Adaptive filtering methods : on methods use a priori information in order to reduce complexity while maintaining convergence properties

Citation for published version (APA):
Sommen, P. C. W. (1992). *Adaptive filtering methods : on methods use a priori information in order to reduce complexity while maintaining convergence properties*. [Phd Thesis 1 (Research TU/e / Graduation TU/e), Electrical Engineering]. Technische Universiteit Eindhoven. https://doi.org/10.6100/IR375310

DOI:
10.6100/IR375310

Document status and date:
Published: 01/01/1992

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 05. Oct. 2023

Adaptive Filtering Methods

On methods to use a priori information in order to reduce complexity while maintaining convergence properties

Proefschrift

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van
de Rector Magnificus, prof. dr. J.H. van Lint, voor
een commissie aangewezen door het College van
Dekanen in het openbaar te verdedigen op
dinsdag 16 juni 1992 om 16.00 uur

door

PETRUS CHRISTIANUS WILHELMUS SOMMEN

Geboren te Ulicoten (N.Br.)

Dit proefschrift is goedgekeurd
door de promotoren
prof. dr. ir. W.M.G. van Bokhoven
en
prof. dr. -ing. H.J. Butterweck

©Copyright 1992 P.C.W. Sommen.
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission from the copyright owner.

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Sommen, Petrus Christianus Wilhelmus

Adaptive filtering methods : on methods to use a priori
information in order to reduce complexity while
maintaining convergence properties / Petrus Christianus
Wilhelmus Sommen. - [S.l. : s.n.]. - Ill., fig
Proefschrift Eindhoven. - Met lit. opg., reg. - Met
samenvatting in het Nederlands.
ISBN 90-9005143-0
NUGI 832
Trefw.: adaptieve filters / digitale signaalverwerking.

Aan mijn Moeder
en ter nagedachtenis aan ons Pa

Summary

In comparison to fixed filters, adaptive filters use extra complexity to update the weights according to some specific algorithm. With such an algorithm it becomes possible to perform satisfactory in an environment where complete knowledge of the relevant signal characteristics is not available. The performance is, among other things, measured by the speed and accuracy of convergence. In many practical cases at least some a priori information is available about the environment and/or the relevant signal characteristics. This thesis explores some *methods to use the available a priori information to reduce complexity, while maintaining convergence properties.*

The acoustic echo canceller, a typical application of an adaptive filter, was one of the research subjects of the Radio and Data Transmission group at Philips Research Laboratories in the period 1984–1989. In this application a speech signal is reflected via an acoustic echo path of 100–200 msec. as an undesired echo into a microphone. The task of the adaptive filter is to produce an estimate of the unknown acoustic echo signal. The main problems of this application are besides the length of the acoustic echo path (modelled with a transversal structure of 1000–2000 coefficients), the non-stationarities in the speech signal and the time variant character of the echo path. Although not a subject of research in this thesis, the acoustic echo canceller has served as a motivation for most parts of this report. Almost all material has been published in [49]–[60] and [14].

In Chapter 1 a general introduction of adaptive filters is given and the used symbols and definitions are explained.

Since block processing techniques play a central role in this thesis, Chapter 2 gives a derivation and analysis of the well known and robust Block Normalized Least Mean Square (BNLMS) algorithm. The BNLMS algorithm makes one update of all N adaptive coefficients every L samples, with L the processing delay. In literature it is often stated that this, low complexity, algorithm has bad convergence properties when a coloured input signal is applied to the adaptive filter. However, from the analytical and experimental results it follows that both the input signal statistics and the initialization of the adaptive weights influence the convergence properties. This implies that convergence properties can become both better and worse for a coloured input signal. A priori knowledge about the "matching" of the signal charac-

teristics and the "unknown" system can be used to initialize the algorithm as good as possible.

In Chapter 3 it is shown that for large filters (the acoustic echo canceller needs an adaptive transversal filter with 1000–2000 coefficients) the BNLMS algorithm can be implemented very efficiently in frequency domain. For the transformation between time- and frequency domain Fast Fourier Transforms (FFTs), with length $B = N+L-1$, are used. This efficient implementation is one of the first reasons to implement the acoustic echo canceller in frequency domain. Furthermore it is known that two well known procedures to carry out this efficient implementation for fixed filters are given by the overlap–save and the overlap–add method. In literature [8] it is asserted that, for complexity reasons, in adaptive filter configurations the overlap–save method is to be preferred to the overlap–add method. This statement is contradicted in this chapter, and it is shown that both methods can be implemented in adaptive filters with equivalent complexity.

Statistical properties of a speech signal are time dependent. When using such a non–stationar input signal, and applying the BNLMS algorithm for the updating of the adaptive coefficients, convergence properties can change very much during adaptation. For this and many other practical situations it is desirable to change the update algorithm in such a way that convergence properties of the adaptive filter become independent of the input signal statistics. Since it is known from literature that decorrelation can be accomplished relatively easy with frequency domain techniques, this is the second motivation to implement the acoustic echo canceller in frequency domain. It is shown in Chapter 4 that in frequency domain decorrelation is performed by normalizing the spectrum of each separate frequency component. With this method an approximation is made of the required time domain decorrelation. First it is shown under what conditions this approximation is acceptable. Applying this spectral normalization to the efficiently implemented BNLMS algorithm in frequency domain, leads to the Block Frequency Domain Adaptive Filter (BFDAF). Roughly there are two variants of the BFDAF known in literature. The first one, with five FFTs, was introduced as the constrained BFDAF since it requires a window that forces a constraint in adjusting the frequency domain weights based on overlap-save sectioning. The second method is the unconstrained BFDAF, since it removes the window. This unconstrained structure only needs three FFTs. An analysis is given of a generalized BFDAF structure, suitable for both structures. From this it follows that, in general, the constrained method (5FFTs) has bettter convergence properties than the unconstrained method (3FFTs). Furthermore it is known that many practical systems, such as the acoustic echo canceller, have a global decaying function as impulse response. When this a priori information is available, an efficient implementation of the BFDAF is introduced, using 3 FFTs, with convergence properties equivalent to the constrained BFDAF (5FFTs).

One of the main problems of block processing techniques is the large processing delay of L samples (usually L is in the order of the filter length N). Furthermore, when performing decorrelation in frequency domain by spectral normalization, the resolution of the spectrum equals the number B of frequency components. However, the statistical properties of the input signal, and thus the required number of divisions, have no resemblance at all with the segment length B. By partitioning the original BFDAF into K smaller parts, with $1 \leq K \leq N$, and implementing this in an efficient way leads to the Partitioned Block Frequency Domain Adaptive Filter (PBFDAF) as dicussed in Chapter 5. This structure has, in comparison with the BFDAF, a reduced processing delay. Furthermore, when some a priori information is available of the input signal spectrum, this information can be used to reduce complexity, since decorrelation is performed with less than B divisions in the PBFDAF approach.

In Chapter 6 the adaptive filtering problem is described in a geometrical way. Generalizing this approach leads to a Block Orthogonal Projection (BOP) method. With this method it is possible to decorrelate the input signal of an adaptive filter with the inverse of an $L \times L$ autocorrelation matrix with $L \geq 1$. This in contrast to the Recursive Least Squares (RLS) method, that uses an $N \times N$ (inverse) autocorrelation matrix (with N the length of the adaptive filter). When some a priori information of the input signal is available, it is possible to match the dimension L of the required autocorrelation matrix more properly to decorrelate the input signal. Both BOP and PBFDAF methods reduce the required number of degrees of freedom for the decorrelation of the input signal of the adaptive filter. For this reason the relationship between these two methods is also discussed in Chapter 6. Furthermore it is known from literature that a speech signal can be modelled by an auto regressive (ar) process. An Efficient Orthogonal Projection (EOP) algorithm is introduced that can decorrelate ar–signals.

Contents

Summary **V**

Contents **VIII**

1 General Introduction to Adaptive Filters **1**

1.1 Applications of adaptive filters 2
1.2 Generic form and general assumptions 8
1.3 Definitions and notations 10
1.4 Symbols 18
1.5 Various measures describing adaptive filter qualities 23

2 The Block Normalized Least Mean Square Algorithm **27**

2.1 NLMS algorithm 28
2.2 BNLMS algorithm 30
2.3 Dynamic Behaviour 33
2.4 Physical Interpretation 40
2.5 Experiments 41
  2.5.1 Influence of adaptation constant α_L 42
  2.5.2 Influence of block length L 44
  2.5.3 Influence of coloured input signal 46
  2.5.4 Initialization 47
2.6 Discussion 49

3 Efficient Implementation of BNLMS algorithm **51**

3.1 Overlap–save method for fixed filters 52
3.2 Overlap–save implementation of BNLMS 55
3.3 Overlap–add for fixed and adaptive filters 57

3.4 Efficient overlap–add method for adaptive filters 58
3.5 Discussion 62

4 **Frequency Domain Adaptive Filters** **64**
4.1 Frequency Domain Adaptive Filter 65
4.2 Decorrelation conditions for the FDAF algorithm 70
4.3 Block Frequency Domain Adaptive Filter 74
4.4 Analysis of the BFDAF algorithm 79
    4.4.1 Unconstrained window function 84
    4.4.2 Constrained window function 85
    4.4.3 Efficient window function 87
4.5 Experiments 89
    4.5.1 FDAF ($L = 1$) 90
    4.5.2 BFDAF 92
4.6 Discussion 93

5 **Partitioned Frequency Domain Adaptive Filters** **95**
5.1 Partitioned BFDAF 96
5.2 Complexity PBFDAF in relation to BFDAF 104
5.3 Mixed–PFDAF 107
    5.3.1 Relation with incomplete Decimation–In–Time FFT . . 113
    5.3.2 Relation with incomplete Decimation In Frequency FFT 113
5.4 Decorrelation conditions for the mixed–PFDAF 114
5.5 Experiments 118
5.6 Discussion 121

6 **Time domain Adaptive filters** **123**
6.1 Decorrelation in time domain with $N \times N$ autocorrelation matrix 124
    6.1.1 LMS/Newton 124
    6.1.2 Recursive Least Square Algorithm 125
6.2 Geometric interpretation of the NLMS algorithm 129
6.3 L–step Orthogonal Projection Algorithm 131
6.4 BOP algorithm with Gram–Schmidt procedure 134
6.5 An efficient OP algorithm for AR(p) processes 136
6.6 Experiments 140
6.7 Discussion 142

7 Conclusions **144**

A Derivation of dynamic behavior of BNLMS algorithm **148**

Bibliography **154**

Samenvatting **162**

Bedankwoord **166**

Curriculum Vitae **167**

Chapter 1

General Introduction to Adaptive Filters

When designing a Wiener filter a priori knowledge about the actual statistical properties of the data to be processed is required. Only when the properties match the a priori information on which the design of the filter is based, the filter is optimum. It may impossible to design the Wiener filter because this information is not known completely and an appropriate design may no longer be optimum. A possible solution to this problem is to first estimate the statistical parameters of the relevant signals and then compute the filter parameters. For real–time operation this procedure may require costly hardware. A more efficient method is to use an adaptive filter. Such a device is self–designing in that the adaptive filter relies for its operation on a recursive algorithm, which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal statistics is not available, or when the statistics slowly vary in time. The algorithm starts with a set of initial conditions, representing all information available about the environment. In a stationary environment, after successive iterations of the algorithm it converges, in average, to the optimum Wiener solution in some statistical sence. In a nonstationary environment, the algorithm offers tracking capability, whereby it can track the variations in the statistics of the relevant signals yielding some local solution, provided that the variations are sufficiently slow. From textbooks [21,27,38,68,10,1,26,7] it follows that adaptive digital filters are extremely useful devices in many applications of digital signal processing.

As a direct consequence of the application of a recursive algorithm, by which the parameters of an adaptive filter are updated from one iteration to the next, the parameters become time dependent. Since this time dependence is caused by the relevant signals an adaptive filter is a nonlinear device. Note that one iteration of the adaptive filter has not necessarily to be performed between two succesive samples.

In another context, an adaptive filter is often referred to as linear in the sense that the output of the filter is obtained as a linear combination of the available set of observations applied to the filter input. This report is restricted to linear, finite impulse response adaptive filters. Furthermore all used signals are discrete in time.

In Section 1.1 of this chapter some adaptive filtering applications are given in order to establish the connection between the material presented in later sections and the application of interest. From these examples a generic form of an adaptive filter is derived in Section 1.2, while in addition general assumptions are given used further on in this report. Section 1.3 gives the main definitions and notations used in later chapters, while Section 1.4 describes the used symbols. Finally in Section 1.5 different factors, describing adaptive filters, are given.

1.1 Applications of adaptive filters

This section describes different applications of adaptive filters.

1. **Signal Estimation** (Fig. 1.1):
   The input signal x is leaked through an unknown system with impulse response h as signal e together with signal s, that is uncorrelated with x, into the measurable signal $\tilde{e}$. The main task of the adaptive filter is to make an estimate $\hat{e}$ of the unknown signal e.

   Some examples that belong to this class are:

   (a) *Echo cancellation* [25]:
   There are roughly two different types of echo cancellers:

   - Voice, acoustic [63,28] (Fig. 1.2):
     In Fig. 1.2 only the near–end acoustic echo canceller is given. An equivalent system will be present at the far–end. Possible

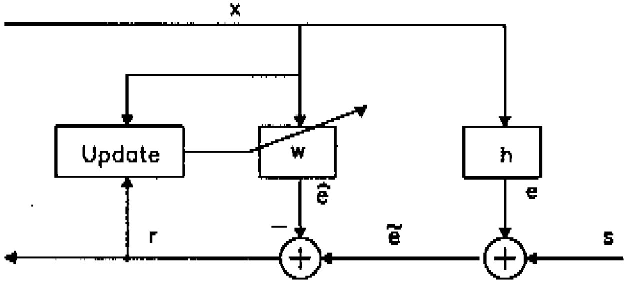

Figure 1.1: *Signal Estimation*

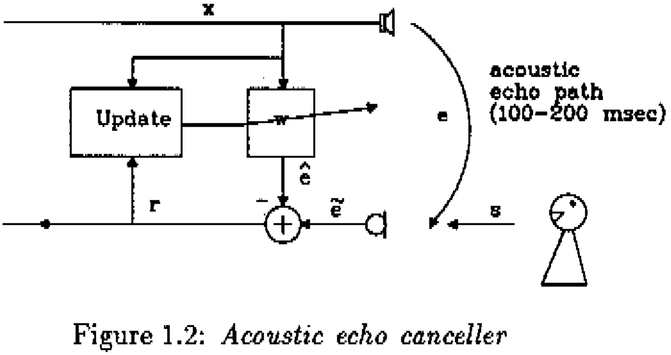

Figure 1.2: *Acoustic echo canceller*

applications of the acoustic echo canceller are: audio–video teleconferencing and loudspeaking telefony systems. In these applications the far–end speech signal x enters the near–end acoustic room through a loudspeaker. Here signal x is reflected, via an acoustic echo path of 100–200 msec, as an undesired echo e into a microphone, together with the near-end speech signal s. Without the acoustic echo canceller the echo signal e will enter the far–end system, where it will be reflected again. The result will be an unacceptable roundsinging or ringing effect.

The task of the near–end acoustic echo canceller is to produce an estimate $\hat{e}$ of the unknown acoustic echo signal e, resulting from the far–end speech signal x. The main problems here are besides the length of the acoustic echo path, the non–stationarities in the input signal x and the time variant character of the echo path. Adaptation is typically carried out in the absence of the near–end speech signal. When double talk is detected (both near- and far–end signals present), updating of the echo canceller coefficients is inhibited.

- Data [67,18] (Fig. 1.3):

  In Fig. 1.3 only the near–end echo canceller is given. An equivalent system is present at the far–end. In a modem,

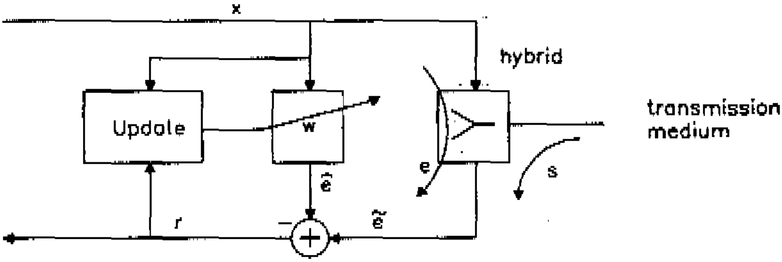

Figure 1.3: *Data echo canceller*

containing (among other things) a transmitter and a receiver for data signals, the near–end data signal x has to be transmitted while the far–end data signal s has to be received. An hybrid directs signal x to, and signal s from a transmission medium. Since the hybrid is not perfect, it will produce an undesired 'echo' signal e of the near–end signal x in the receiver. In practical situations e can be up to 20 dB stronger than the desired signal s.

The near–end echo canceller has to produce an estimate $\hat{e}$ of the echo e, resulting from the 'near–end' signal x. Filter adaptation is typically required to be continued in the presence of a large echo signal e which is correlated with the near–end transmitted data signal x.

(b) *Noise cancellation* [43]:

An unknown system with impulse response h coloures the measurable noise source x. The adaptive filter produces an estimate $\hat{e}$ of the signal e (Fig. 1.1). One example is that of cancelling noise from the pilot's speech signal in the cockpit of an aircraft. In this case $\tilde{e}$ may be the pickup from a microphone in the pilot's helmet and x is the ambient noise picked up by another microphone placed in the cockpit.

(c) *Adaptive Arrays* [5] (Fig. 1.4):

In this application a number of input signals, e.g. from an array of receiving antennas or microphones, are processed through an array of adaptive filters whose outputs are summed together. The radiation diagram of the array can be adjusted by adjusting the amplitude and phase of each array element with an adaptive filter. In practice the filters can consist of only one coefficient. The adaptation can be done to create a null in the direction of an interfering transmitter (adaptive null steering) or to generate a maximum output for a desired signal from an unknown direction (adaptive beam forming [66]).

2. **Signal Correction** (Fig. 1.5):

The desired signal e is distorted by an unknown channel with impulse response h. Together with a noise signal n, the output of this channel is available as input signal x of the adaptive filter. The adaptive filter has to correct this signal x in such a way that the channel distortion is removed and the desired signal e can be estimated.

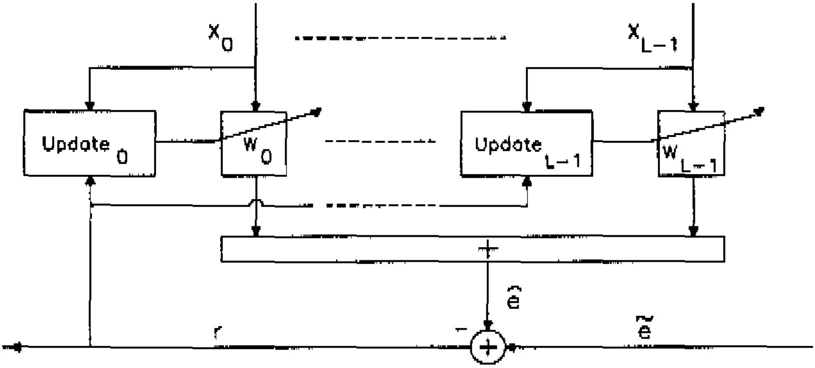

Figure 1.4: *Adaptive Array*

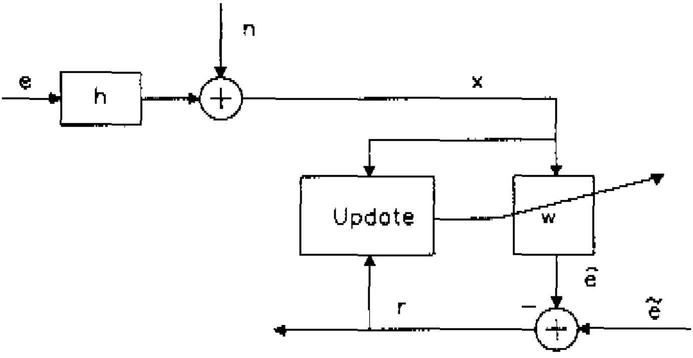

Figure 1.5: *Signal Correction*

The most important example of this class is the linear equalizer [41]. After the initial training period (if there is one), the coefficients w of the adaptive equalizer may be continually adjusted in a decision–directed manner. In this mode, the residual signal r is derived from the final (not necessarily correct) receiver estimate $\tilde{e}$ of the transmitted sequence e. In normal operation, the receiver decisions are correct with high probability, so that the error estimates are correct often enough to allow the adaptive equalizer to maintain precise equalization. Moreover, a decision–directed adaptive equalizer can track slow variations in the channel characteristics or linear perturbations in the receiver front end, such as slow jitter in the sampler phase.

3. **Signal Prediction** (Fig. 1.6):
   The original signal $\tilde{e}$ consists of a predictable and an unpredictable part. The adaptive filter has to produce an estimate $\hat{e}$ of the predictable part. The residual signal $r = \tilde{e} - \hat{e}$ will equal the unpredictable part of $\tilde{e}$. An example is the Adaptive Line Enhancer (ALE) [44,19]. The ALE can be used to detect a low level sine wave s embedded in a background additive noise n with a broadband spectrum ($\tilde{e} = s + n$). The main function of the delay T_L is to remove correlation that may exist between the noise component n in the original input signal $\tilde{e}$ and the noise component in the delayed predictor input x. An ALE must be viewed as an adaptive filter that is designed to suppress broadband components (e.g. white noise) contained in the input signal, while at the same time passing narrowband components (e.g. sine wave) with little attenuation. In other words, it can be used to enhance the

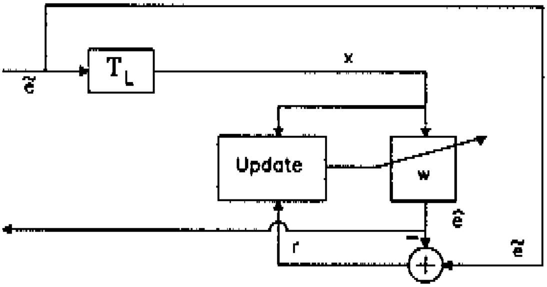

Figure 1.6: *Signal Prediction*

presence of sine waves (whose spectrum consists of harmonic lines) in an adaptive manner.

Another well known example is linear predictive coding (LPC) of speech where the end result is the set of estimated LPC coefficients [44,19]. Due to the nonstationary nature of the speech signal, LPC coefficients are typically obtained separately for each new frame (10 to 25 ms) of the speech signal.

1.2 Generic form and general assumptions

There is no unique solution to the adaptive filtering problem. For the purpose of further development a generic form is given (Fig. 1.7), which is based on Wiener filter theory. It is assumed that the unknown system function w_{opt}, can be modelled with a Finite Impulse Response (FIR) filter. For the adaptive filter also an FIR structure is used as the structural basis for implementing the adaptive filter, from which the order is assumed to be greater than (but at least equal to) the order of the unknown optimum Wiener filter. For the case of stationary inputs, the mean-squared error J (i.e. the mean-squared value of the difference between the signal $\tilde{e}$ and the FIR filter output $\hat{e}$) is precisely a quadratic function of the adaptive weights w in the FIR filter. The dependence of the mean-squared error J on the unknown weights may be visualized in the form of a multidimensional paraboloid with a uniquely defined

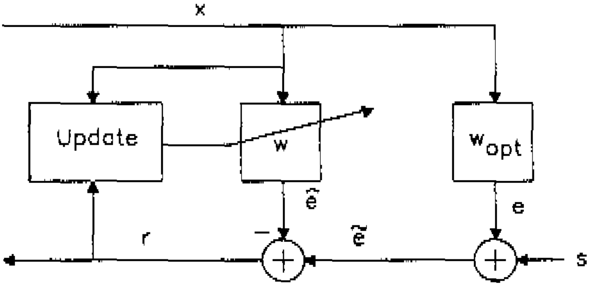

Figure 1.7: *Generic form of adaptive filter*

bottom or minimum point J_{min}. The weights corresponding to this minimum point define the optimum Wiener solution vector $\underline{\mathbf{w}}_{\text{opt}}$. It can be shown that the gradient vector $\underline{\nabla} = \delta J / \delta \underline{\mathbf{w}}$ always points away from the minimum J_{min} as depicted for a single adaptive weight in Fig. 1.8. The coefficient vector $\underline{\mathbf{w}}$ of the adaptive filter is updated in such a way that after convergence the adaptive weights will equal, in average, the unknown optimal FIR Wiener solution vector $\underline{\mathbf{w}}_{\text{opt}}$. This can be reached by using a steepest-descent update, from which the update scheme looks like:

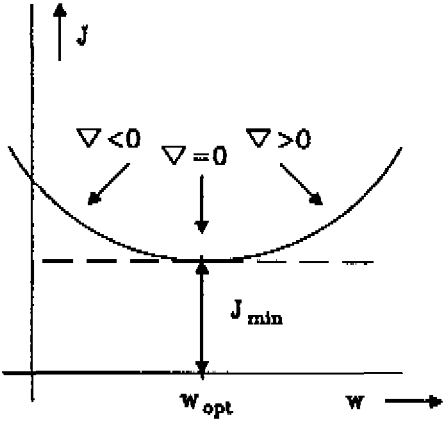

Figure 1.8: *J as function of a single adaptive weight w*

$$\underline{\mathbf{w}} = \underline{\mathbf{w}} - \alpha \cdot \underline{\nabla} \tag{1.1}$$

where α is the adaptation constant ($\alpha > 0$). In general this will result in an update algorithm for the adaptive weights which will have the following general form:

$$\left(\begin{array}{c}\text{new}\\ \text{coefficient}\\ \text{vector}\end{array}\right) = \left(\begin{array}{c}\text{old}\\ \text{coefficient}\\ \text{vector}\end{array}\right) + \left(\begin{array}{c}\text{adaptation}\\ \text{constant}\\ (\alpha)\end{array}\right) \cdot \left(\begin{array}{c}\text{input}\\ \text{signal}\\ \text{vector}\end{array}\right) \cdot \left(\begin{array}{c}\text{residual}\\ \text{signal}\\ \end{array}\right)$$

Generally it is assumed in this report that the reference signal $\tilde{e}$ consists of two components $\tilde{e} = e + s$. The first signal e is the result of a linear convolution of the input signal x with the unknown optimal

FIR Wiener filter w_{opt}, while the second signal s is uncorrelated with x. Furthermore it is assumed that both x and s have zero mean. In practical applications, e.g. for the acoustic echo canceller as described in Section 1.1, a double talk detector is needed: the result is that, during adaptation, signal s may be approximated as a white noise signal. This approximation will be used for all analytical calculations. In order to make analysis of some adaptive algorithms more tractable, the assumption is made that the adaptation constant α is sufficiently small.

As mentioned before an adaptive filter offers tracking capabilities in nonstationary environments. It can track variations in the unknown optimum Wiener filter. Furthermore the tracking quality is dependent of the nonstationarity of the input signal. Since tracking is not the main topic in this report it is assumed that the signals are stationary and that the environment, the unknown Wiener solution, is not time variant.

1.3 Definitions and notations

For the purpose of further development in this section some frequently used definitions and notations are listed.

All signals in time domain are assumed to be real and discrete in time (denoted by square brackets). Thus $x[k]$ denotes signal x at discrete time k, corresponding to $k \cdot T$ in continuous time.

Furthermore, with N denoting the order of the adaptive filter, signals and weights are represented in vectors as follows:

$$\begin{aligned}
\underline{\mathbf{x}}[k] &= (x[k-N+1], \cdots, x[k-1], x[k])^t \\
\underline{\mathbf{w}}[k] &= (w_{N-1}[k], \cdots, w_1[k], w_0[k])^t \\
\underline{\mathbf{w}}_{\mathrm{opt}} &= (w_{\mathrm{opt},N-1}, \cdots, w_{\mathrm{opt},1}, w_{\mathrm{opt},0})^t \\
\underline{\mathbf{d}}[k] &= \underline{\mathbf{w}}_{\mathrm{opt}} - \underline{\mathbf{w}}[k]
\end{aligned}$$

where t denotes transpose. In general a bold-face and underlined character is used for a vector, while a bold-face not underlined letter denotes a matrix. In some cases it is necessary to denote the dimension of a vector or matrix explicitly. For example when it is not obvious that the dimension of vector $\underline{\mathbf{x}}[k]$ is B then it is denoted with a capital subscript as:

$$\underline{\mathbf{x}}_B[k] = (x[k-B+1], \cdots, x[k-1], x[k])^t \tag{1.2}$$

The $N \times N$ identity matrix is given by $\mathbf{I}_N$, while a circular shift of the data over L positions in an N dimensional vector is carried out by the matrix:

$$\mathbf{I}_N^L = \begin{pmatrix} \mathbf{0} & \mathbf{I}_{N-L} \\ \mathbf{I}_L & \mathbf{0} \end{pmatrix} \tag{1.3}$$

in which the zero matrices $\mathbf{0}$ have appropriate dimensions. Note that $\mathbf{I}_N^0 = \mathbf{I}_N$. The data of an N dimensional vector can be mirrored with the $N \times N$ mirrored matrix $\mathbf{J}_N$ that is defined as:

$$\mathbf{J}_N = \begin{pmatrix} & & & & 1 \\ & 0 & & 1 & \\ & & \cdot & & \\ & & \cdot & & \\ & 1 & & 0 & \\ 1 & & & & \end{pmatrix}. \tag{1.4}$$

A reverse circular shift over L positions, in opposite direction to $\mathbf{I}_N^L$, is carried out by the matrix

$$\mathbf{J}_N^L = \begin{pmatrix} \mathbf{J}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{N-L} \end{pmatrix}. \tag{1.5}$$

Note that $\mathbf{J}_N^0 = \mathbf{J}_N$.

When an N dimensional vector $\underline{\mathbf{x}}_N[k]$ is changed to a B dimensional vector $\underline{\mathbf{y}}_B[k]$ (with $B = N+L-1$) by adding $L-1$ zeros, this is denoted by the following vector matrix product:

$$\underline{\mathbf{y}}_B[k] = \begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \cdot \underline{\mathbf{x}}_N[k]. \tag{1.6}$$

On the other hand when a B dimensional vector $\underline{\mathbf{x}}_B[k]$ is changed to an N dimensional vector $\underline{\mathbf{y}}_N[k]$ by throwing away the last $L-1$ elements, this is carried out by

$$\underline{\mathbf{y}}_N[k] = \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix} \cdot \underline{\mathbf{x}}_B[k]. \tag{1.7}$$

In the last part of this report the stochastic signal $x[k]$, that is represented in the N dimensional vector $\underline{\mathbf{x}}[k]$, is considered in a geometrical way. For this the innerproduct between two real vectors $\underline{\mathbf{x}}[k]$ and $\underline{\mathbf{w}}_{\mathrm{opt}}$ is defined as:

$$< \underline{\mathbf{x}}[k], \underline{\mathbf{w}}_{\mathrm{opt}} >= \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{w}}_{\mathrm{opt}} = \sum_{i=0}^{N-1} w_{\mathrm{opt},i} x[k-i] \tag{1.8}$$

and the L^2 norm of the real vector $\underline{\mathbf{x}}[k]$ is defined as:

$$||\underline{\mathbf{x}}[k]|| = \sqrt{< \underline{\mathbf{x}}[k], \underline{\mathbf{x}}[k] >}. \tag{1.9}$$

Furthermore, with $E\{\cdot\}$ denoting the mathematical expectation, the variances of the stationary stochastic signals $x[k]$ and $s[k]$, both having zero mean, are defined as $E\{x^2[k]\} = \sigma_x^2$ and $E\{s^2[k]\} = \sigma_s^2$. For stationary real signals $x[k]$ the autocorrelation function $\rho[\tau]$ is defined as [37]:

$$\rho[\tau] = E\{x[k]x[k-\tau]\}. \tag{1.10}$$

This autocorrelation function can be represented with an $N \times N$ real symmetric Toeplitz autocorrelation matrix $\mathbf{R}$ that is defined as:

$$\mathbf{R} = E\{\underline{\mathbf{x}}[k] \cdot \underline{\mathbf{x}}^t[k]\} = \mathbf{R}^t$$

whose $(k,l)^{th}$ element is given by:

$$(\mathbf{R})_{k,l} = \rho[|k-l|] \tag{1.11}$$

or in matrix form:

$$\mathbf{R} = \begin{pmatrix} \rho[0] & \rho[1] & \cdot & \cdot & \rho[N-1] \\ \rho[1] & \rho[0] & \cdot & \cdot & \rho[N-2] \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \rho[N-1] & \rho[N-2] & \cdot & \cdot & \rho[0] \end{pmatrix}. \tag{1.12}$$

With $\mathbf{Q}^h$ denoting the hermetian (=complex conjugate $(*)$ and transpose (t)) of matrix $\mathbf{Q}$, the following unitary or similarity transformation is often useful:

$$\mathbf{Q}^h \mathbf{R} \mathbf{Q} = \mathbf{\Lambda} \tag{1.13}$$

where $\Lambda = diag\{\lambda_0, \cdots, \lambda_{N-1}\}$ is a diagonal matrix containing the eigenvalues of $\mathbf{R}$ and $\mathbf{Q} = (\underline{\mathbf{q}}_0, \cdots, \underline{\mathbf{q}}_{N-1})$ is an orthonormal matrix containing the eigenvectors. The following properties of $\mathbf{Q}$ and $\mathbf{R}$ are used (see also [26]):

$$\begin{aligned} \mathbf{Q}^h \cdot \mathbf{Q} = \mathbf{I} \;\; \Leftrightarrow \;\; & \underline{\mathbf{q}}_l^h \cdot \underline{\mathbf{q}}_m = \begin{cases} 1 & \text{if } l = m \\ 0 & \text{if } l \neq m \end{cases} \\ \mathbf{Q}^h \mathbf{Q} = \mathbf{I} \;\; \Leftrightarrow \;\; & \mathbf{Q}\mathbf{Q}^h = \mathbf{I} \\ \sum_{l=0}^{N-1} \lambda_l \;\; = \;\; & trace\{\Lambda\} = trace\{\mathbf{Q}^h \mathbf{R} \mathbf{Q}\} = trace\{\mathbf{R}\mathbf{Q}\mathbf{Q}^h\} \\ = \;\; & trace\{\mathbf{R}\} = N\sigma_x^2 \\ \lambda_l \;\; \geq \;\; & 0 \quad \text{for} \quad l = 0, 1, \cdots, N-1. \end{aligned} \tag{1.14}$$

while furthermore the "relative eigenvalue" $\overline{\lambda}_l$ is defined as:

$$\overline{\lambda}_l = \frac{\lambda_l}{\frac{1}{N}\sum_{q=0}^{N-1} \lambda_q} = \frac{\lambda_l}{\sigma_x^2}. \tag{1.15}$$

The Eigenvalue Ratio (ER) is defined as the ratio of the maximum and minimum eigenvalue:

$$ER = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{1.16}$$

Both the autocorrelation function $\rho[\tau]$ and the autocorrelation matrix $\mathbf{R}$ represent a time–domain description of the second order statistics of a stationary discrete–time stochastic signal. The power spectral density function (psdf) $P(e^{j\theta})$, that is a real and positive function, is related to this autocorrelation function via the Fourier Transformation for Discrete signals (FTD) as follows:

$$P(e^{j\theta}) = \sum_{\tau=-\infty}^{\infty} \rho[\tau] e^{-j\tau\theta} = \rho[0] + 2\sum_{\tau=1}^{\infty} \rho[\tau]\cos(\theta\tau). \tag{1.17}$$

When comparing theoretical and experimental results different types of input signals are used. These signals are generated by signal models. All these models derive signal $x[k]$ from a white noise signal $n[k]$ through appropriate filtering (colouring). The signal $n[k]$ has zero mean ,$E\{n[k]\} = 0$, and variance $E\{n^2[k]\} = \sigma_n^2$. In the sequel for each separate model the difference equation, describing the model, and the autocorrelation function $\rho[\tau]$ are given. Furthermore an expression is given for the psdf $P(e^{j\theta})$. From literature [26] it is known that the eigenvalues are bounded by the maximum and minimum of the psdf [22], and this bound for the ER is also given. In order to be able to make correct comparisons between convergence properties of an adaptive filter with different kind of input signals, different signal models are chosen in such a way that their total power is the same for all these models. This implies that the model parameters are chosen in such a way that

$$\frac{1}{2\pi}\int_0^{2\pi} P(e^{j\theta})d\theta = \rho[0] = \sigma_n^2. \qquad (1.18)$$

White noise signal model:

Each sample of the white noise signal $n[k]$ has no relation (correlation) with all other samples $n[k-i]$ for $i \neq 0$.

$$\begin{aligned} \text{Model} \quad &: \quad x[k] = n[k] \qquad (1.19) \\ \text{Autocorrelation} \quad &: \quad \rho[\tau] = \begin{cases} \sigma_n^2 & \text{for } \tau = 0 \\ 0 & \text{elsewhere.} \end{cases} \\ \text{Spectrum} \quad &: \quad P(e^{j\theta}) = \sigma_n^2 \\ \text{Eigenvalue Ratio} \quad &: \quad ER = 1 \end{aligned}$$

Moving Average signal model of order 1 (ma):

A typical example of a signal that belongs to this class is an "AMI" (Alternative Mark Inversion) code [13]. This code possesses several characteristics that are desirable in baseband pulse transmission. It has no DC component and contains only a small amount of low-frequency components. Timing information can easily be recovered from the received line signal.

The ma-signal model of order 1 is defined as:

$$\begin{aligned} \text{Model} \quad &: \quad x[k] = \frac{1}{\sqrt{1+a^2}} \cdot (n[k] + a n[k-1]) \qquad (1.20) \\ \text{Autocorrelation} \quad &: \quad \rho[\tau] = \begin{cases} \sigma_n^2 & \text{for } \tau = 0 \\ \frac{a}{1+a^2}\sigma_n^2 & \text{for } \tau = \pm 1 \\ 0 & \text{elsewhere.} \end{cases} \end{aligned}$$

$$\begin{aligned} \text{Spectrum} \quad &: \quad P(e^{j\theta}) = \left(1 + \frac{2a}{1+a^2}\cos(\theta)\right)\sigma_n^2. \\ \text{Eigenvalue Ratio} \quad &: \quad ER \leq \left(\frac{1+a}{1-a}\right)^2 \end{aligned}$$

Auto Regressive signal model of order 1 (ar):

The vocal tract of a speech signal can typically be described by an ar signal of higher order [48].

The ar–signal model of order 1, with $|a| < 1$, is defined as:

$$\begin{aligned} \text{Model} \quad &: \quad x[k] = (\sqrt{1-a^2}) \cdot n[k] + ax[k-1] \quad (1.21) \\ \text{Autocorrelation} \quad &: \quad \rho[\tau] = a^{|\tau|}\sigma_n^2 \\ \text{Spectrum} \quad &: \quad P(e^{j\theta}) = \frac{(1-a^2)\sigma_n^2}{1+a^2-2a\cos(\theta)} \\ \text{Eigenvalue Ratio} \quad &: \quad ER \leq \left(\frac{1+a}{1-a}\right)^2 \end{aligned}$$

An N dimensional discrete–time vector $\underline{\mathbf{x}}[k]$ is transformed to the frequency domain vector $\underline{\mathbf{X}}[k]$ via the Discrete Fourier Transform (DFT), by using the $N \times N$ Fourier matrix $\mathbf{F}$. Thus:

$$\underline{\mathbf{X}}[k] = (X_0[k], \cdots, X_{N-1}[k])^t = \mathbf{F} \cdot \underline{\mathbf{x}}[k] \tag{1.22}$$

with the $(p,q)^{\text{th}}$ element of $\mathbf{F}$ given by:

$$(\mathbf{F})_{p,q} = e^{-jpq\theta_N} \quad \text{with} \quad \theta_N = \frac{2\pi}{N}. \tag{1.23}$$

Note that the indices $p, q$are defined in the range $0, 1, \cdots, N-1$. This Fourier matrix has symmetrical and unitary propertiea, namely:

$$\mathbf{F}^t = \mathbf{F} \qquad \mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^h. \tag{1.24}$$

In this report two types of circulant matrices, that can be constructed by a circular shift of the first row, are used. These are the "I–circulant" matrix, that is defined as

$$\mathbf{C} = \begin{pmatrix} c[0] & c[1] & \cdot & \cdot & c[N-1] \\ c[N-1] & c[0] & \cdot & \cdot & c[N-2] \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ c[1] & c[2] & \cdot & \cdot & c[0] \end{pmatrix} \tag{1.25}$$

and the ”J–circulant” matrix

$$\mathbf{C} = \begin{pmatrix} c[0] & \cdot\ \cdot & c[N-2] & c[N-1] \\ c[1] & \cdot\ \cdot & c[N-1] & c[0] \\ \cdot & \cdot\ \cdot & \cdot & \cdot \\ \cdot & \cdot\ \cdot & \cdot & \cdot \\ c[N-1] & \cdot\ \cdot & c[N-3] & c[N-2] \end{pmatrix}. \qquad (1.26)$$

Since the DFT has a circular property, the Fourier matrix **F** diagonalizes the I–circulant matrix **C** as follows [12]:

$$\mathbf{FCF}^{-1} = \frac{1}{N}\mathbf{FCF}^{h} = \mathbf{P} = diag\{P_0, \cdots, P_{N-1}\}. \qquad (1.27)$$

When **C** is an J–circulant matrix, this is diagonalized by the Fourier matrix **F** as follows:

$$\frac{1}{N}\mathbf{FCF} = \mathbf{P} = diag\{P_0, \cdots, P_{N-1}\}. \qquad (1.28)$$

When the autocorrelation function $\rho[\tau]$ is restricted $\rho[\tau] = 0$ for $|\tau| > \tau_{\mathrm{max}}$, with $\tau_{\mathrm{max}} \ll N$, it is possible to construct the I–circulant autocorrelation matrix **C** out of the Toeplitz matrix **R**. A possibility is to define the first row of the circulant matrix **C** as follows:

$$c[\tau] = \begin{cases} \rho[\tau] & \text{for } \tau = 0, \cdots, \tau_{\mathrm{max}} \\ \rho[N-\tau] & \text{for } \tau = N - \tau_{\mathrm{max}}, \cdots, N-1 \\ 0 & \text{elsewhere .} \end{cases} \qquad (1.29)$$

Some important similarities, extensively treated in [22,23], between the autocorrelation **R** and the I–circulant autocorrelation matrix **C**, constructed as in (1.29), are used in this report. The most important ones are listed below:

- With the I–circulant autocorrelation matrix **C**, as constructed in (1.29), the eigenvalues P_l of **C** are approximated as:

$$P_l \approx P(e^{j\theta})|_{\theta = l \cdot \theta_N} \quad \text{with} \quad \theta_N = \frac{2\pi}{N}. \qquad (1.30)$$

- The eigenvalues of the autocorrelation matrix $\mathbf{R}$ are bounded by the minimum and maximum of the psdf:

$$\lambda_{\max} \leq \underset{\theta}{\max} \left\{P(e^{j\theta})\right\} \quad \text{and} \quad \lambda_{\min} \geq \underset{\theta}{\min} \left\{P(e^{j\theta})\right\}. \tag{1.31}$$

- The eigenvalues λ_l of the Toeplitz autocorrelation matrix $\mathbf{R}$ and the sampled psdf $P(e^{j\theta})|\theta = l \cdot \theta_N$ are asymptotically equally distributed. This result can easily be shown for the given models, since each of these models have monotonuously increasing/decreasing psdf. The result is plotted in Fig. 1.9. The signals are generated with the above mentioned models with $\sigma_n^2 = 1$ and $|a| = 0.8182$ ($ER \approx 100$). The same figure shows in point-type the ordered eigenvalues of the 32×32 autocorrelation matrix $\mathbf{R}$, at a distance of $\theta_N = \frac{2\pi}{N}$. From this figure it follows that for these simple models the ordered eigenvalues are almost equal to the sampled values of the psdf:

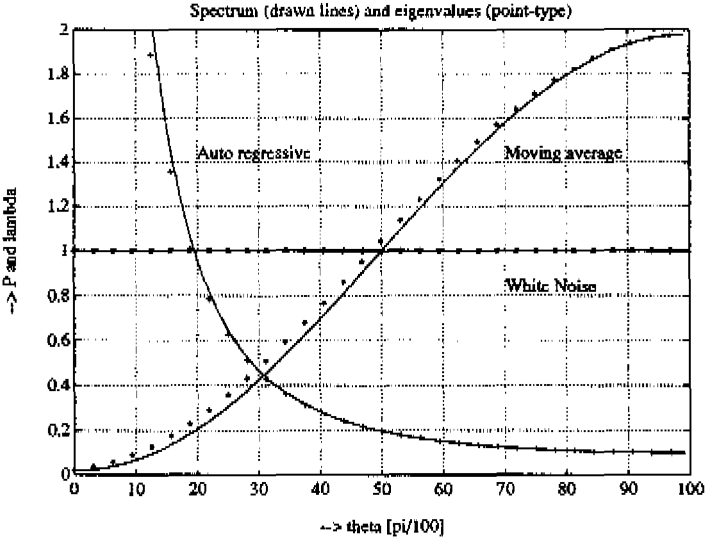

Figure 1.9: *Spectrum and eigenvalues for different signal models*

$$\lambda_l \approx P(e^{j\theta})|_{\theta = l \cdot \theta_N} \quad \text{for} \quad l = 0, 1, \cdots, N-1. \tag{1.32}$$

This similar behaviour of Toeplitz and circulant matrices should seem reasonable since the normal equations $\mathbf{R}\cdot\underline{\mathbf{q}} = \lambda\cdot\underline{\mathbf{q}}$ and $\mathbf{C}\cdot\underline{\mathbf{q}} = P\cdot\underline{\mathbf{q}}$ for $N \gg 2\tau_{\mathrm{max}} + 1$ are essentially the same difference equations of order τ_{max}, with different boundary conditions. In fact the "nice" boundary conditions make P easy to find exactly, by using a DFT, while exact solution for λ are usually more complicated.

In general different frequency components $X_l[k]$ and $X_m[k]$ are correlated with each other (see for an an exact description of this interbin correlation the course books [32] (part of B. Picinbono)). Here however it is assumed that different frequency components $X_l[k]$ are approximately uncorrelated, thus:

$$\frac{1}{N}E\{X_l[k]X_m^*[k]\} \approx \begin{cases} P_l & \text{for } l = m \\ 0 & \text{elsewhere.} \end{cases} \tag{1.33}$$

This approximation makes use of the circulant approximation as described above. This can be visualized in the following expression:

$$\begin{aligned} \frac{1}{N}E\{\underline{\mathbf{X}}[k]\underline{\mathbf{X}}^h[k]\} &= \frac{1}{N}\mathbf{F}E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}\mathbf{F}^h = \frac{1}{N}\mathbf{F}\mathbf{R}\mathbf{F}^h \\ &\overset{\mathbf{R}\to\mathbf{C}}{\approx} \frac{1}{N}\mathbf{F}\mathbf{C}\mathbf{F}^h = \mathbf{P} \\ &= diag\{P_0,\cdots,P_{N-1}\}. \end{aligned} \tag{1.34}$$

1.4 Symbols

All experiments with the adaptive filter structures and algorithms in this report have been carried out on a PC in which a DSP board (TMS320C30) was mounted. The used symbols, listed below, are such that it is straightforward to implement them on this PC/DSP combination.

A signal is denoted by a single line arrow (→) while a vector, containing more signals, is indicated by a double line arrow (⇒). When a vector $\underline{\mathbf{v}} = (v_0,\cdots,v_{N-1})^t$ contains signal samples then always element zero will equal the sample with the lowest time index. Thus $v_0 = x[k-N+1],\cdots,v_{N-1} = x[k]$. In Fig. 1.10 the first two symbols are introduced

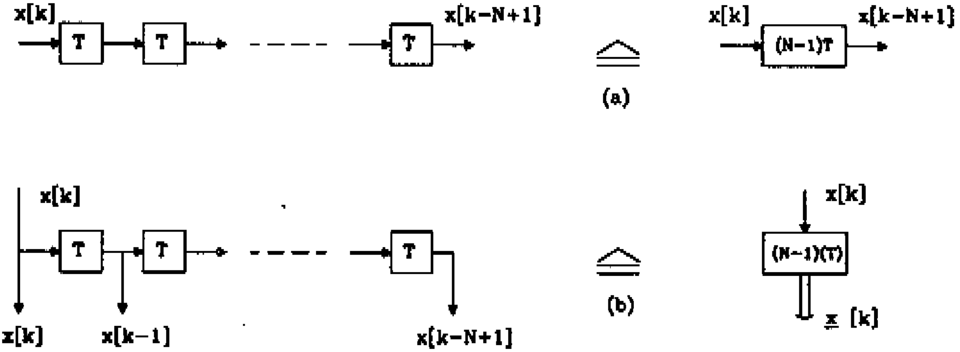

Figure 1.10: *Symbols for cascading delay elements (a) and delay line (b)*

on the right–hand side of the figure. The first symbol, having one input and one output, denotes the cascading of $(N-1)$ delay elements T, containing N samples of signal $x[k]$. The second symbol is used for a delay line with one input signal sample $x[k]$ and a length N vector $\underline{\mathbf{x}}[k]$ of output samples.

Different symbols to reorder elements of a vector $\underline{\mathbf{y}}$ are depicted in Fig. 1.11. The first symbol denotes a mirroring of all elements in a vector. Mathematically this is denoted as

$$\underline{\mathbf{y}} = \mathbf{J}_N \cdot \underline{\mathbf{x}}. \tag{1.35}$$

The second symbol is used to shift the elements of vector $\underline{\mathbf{x}}$ in a circular way over L positions or mathematically:

$$\underline{\mathbf{y}} = \mathbf{I}_N^L \underline{\mathbf{x}}. \tag{1.36}$$

In some cases it is necessary to change the dimension of a vector by adding zeros. This is carried out, for $B = N + L - 1$, in the first symbol of Fig. 1.12, or mathematically:

$$\underline{\mathbf{y}}_B = \begin{pmatrix} \mathbf{I}_N \\ 0 \end{pmatrix} \underline{\mathbf{x}}_N. \tag{1.37}$$

The second symbol in this figure is used to switch from an B dimen-

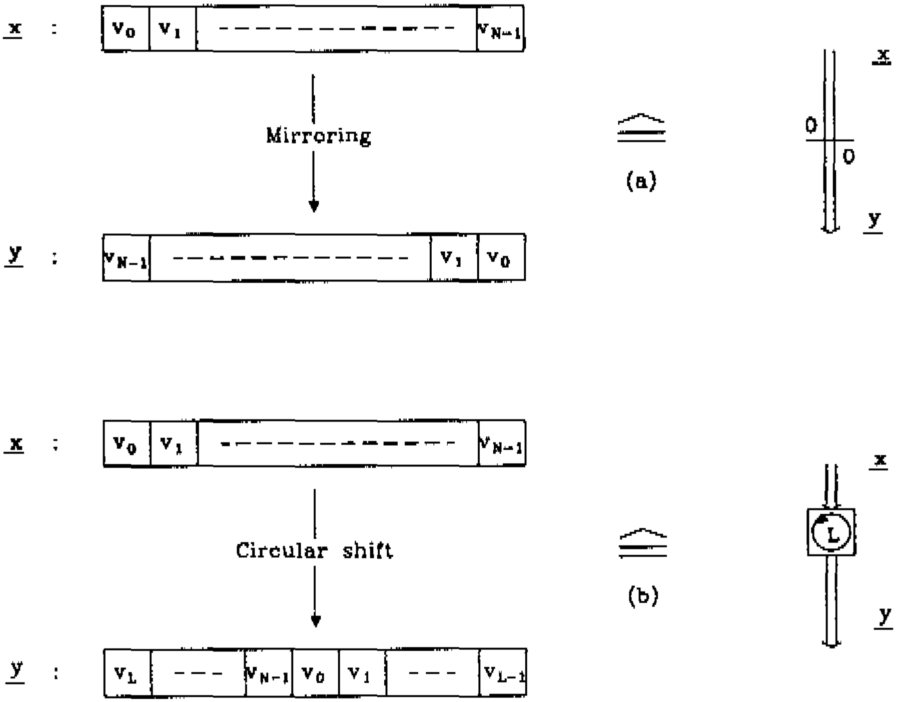

Figure 1.11: *Reordering symbols*

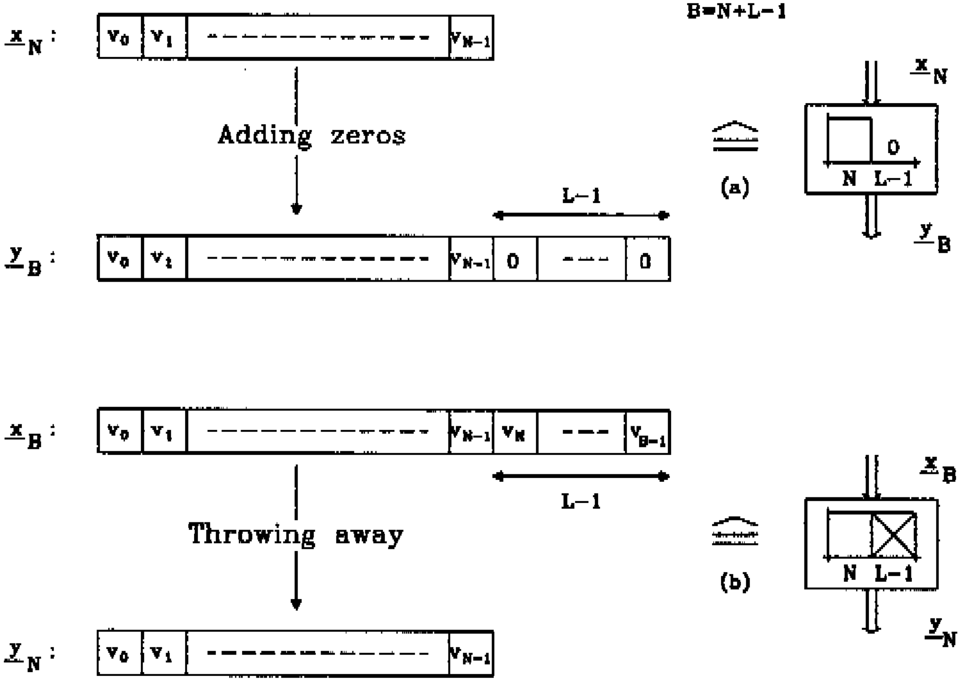

Figure 1.12: *Symbols to change the dimension of a vector*

sional vector $\underline{\mathbf{x}}_B$ to an N dimensional vector $\underline{\mathbf{y}}_N$, by discarding $L-1$ elements, or mathematically:

$$\underline{\mathbf{y}}_N = \begin{pmatrix} \mathbf{I}_N & 0 \end{pmatrix} \underline{\mathbf{x}}_B. \tag{1.38}$$

Fig. 1.13 shows some addition and multiplication symbols. These are defined as:

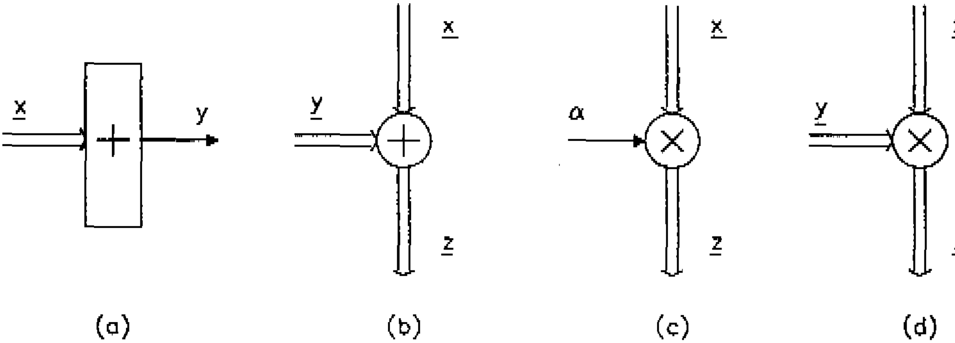

Figure 1.13: *Addition and multiplication symbols*

$$\begin{array}{lll} \text{(a)} & y = & \sum_{i=0}^{N-1} (\underline{\mathbf{x}})_i \\ \text{(b)} & (\underline{\mathbf{z}})_i = & (\underline{\mathbf{x}})_i + (\underline{\mathbf{y}})_i \quad \text{for} \quad i = 0, \cdots, N-1 \\ \text{(c)} & (\underline{\mathbf{y}})_i = & \alpha \cdot (\underline{\mathbf{x}})_i \quad \text{for} \quad i = 0, \cdots, N-1 \\ \text{(d)} & (\underline{\mathbf{z}})_i = & (\underline{\mathbf{x}})_i \cdot (\underline{\mathbf{y}})_i \quad \text{for} \quad i = 0, \cdots, N-1 \end{array}$$

When using block processing techniques the input signal $x[k]$ has to be segmented into overlapping blocks. This way of processing is implemented in this report by using up- and down-samplers [11]. When a signal $x[k]$ is down-sampled by a factor N, it is denoted as $x[kN]$. For this example a delay element, in the down-sampled domain, corresponds to $T_N = N \cdot T$ seconds. Fig. 1.14 shows an example of a down-sampler followed by one delay element T_N and an up-sampler. The output signal $y[k]$ of this example is given by:

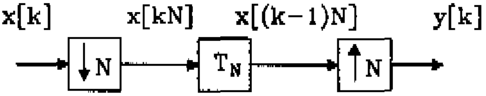

Figure 1.14: *Down–sampler, up–sampler and delay element symbol*

$$y[k] = \begin{cases} x[k-N] & \text{for} \quad k = 0, N, 2N, \cdots \\ 0 & \text{elsewhere.} \end{cases} \tag{1.39}$$

Finally Fig. 1.15 shows an example of arranging the input signal samples in such a way that block processing techniques can be applied. The input signal $x[k]$ is split into segments with B samples $x[k-B+1], \cdots, x[k-1], x[k]$. With $B = N + L - 1$ these segments have an overlap of $N - 1$ samples with the previous segment. This splitting and overlapping is performed in the delay line followed by a down sampler. These down samplers all shift the data at the same moment. Now the block processing can take place, denoted in the dashed area in the figure. Using circular techniques, the block processing is often in such a way that for the calculation of L output signal samples $y[k], \cdots, y[k_L + 1]$, all $B = N + L - 1$ input signal samples are needed. The result of such block processing techniques is usually a vector from which only a part is needed. A window throws away the first $N - 1$ elements of the vector $\underline{y}[kL]$. Returning to the original sampling rate is carried out by an upsampler, that shifts all the data at the same moment, and a transposed delay line. Block processing techniques will always introduce a processing delay. In this example the first output sample is $y[k - L + 1]$, and thus the processing delay is $L - 1$ samples.

1.5 Various measures describing adaptive filter qualities

A wide variety of recursive algorithms has been developed in the literature for the operation of adaptive filters. In most practical cases the choice of one algorithm in favour of another is determined by various factors, depending on the exchange between complexity requirements and convergence properties of the adaptive filter. In practice the main

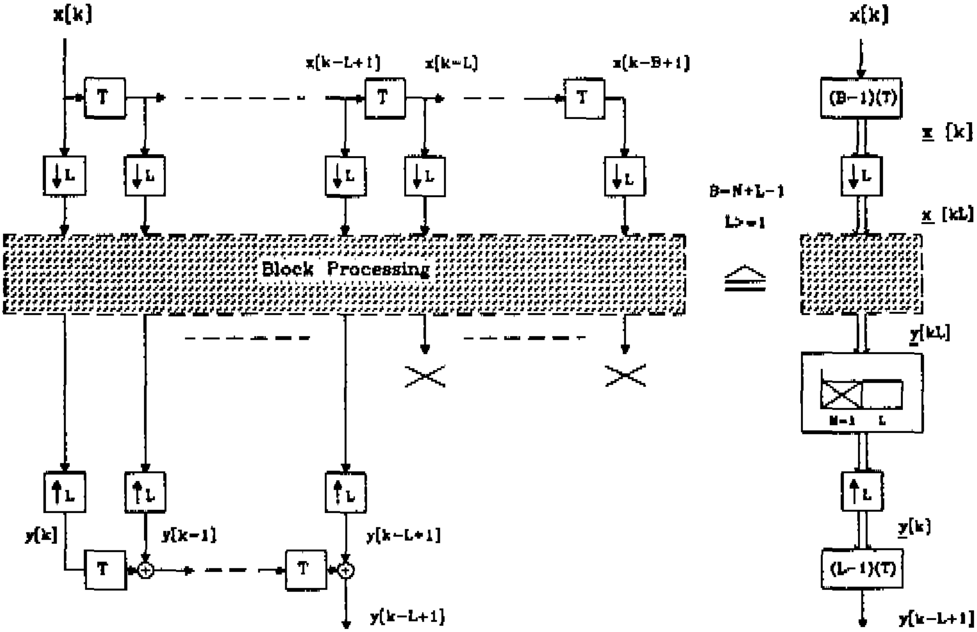

Figure 1.15: *Example of block processing with overlapping input signal segments*

goal in designing an adaptive filter is to reach a certain accuracy "as soon as possible" with the least amount of complexity. As a rough measure for complexity the number of real multiplications, needed to calculate one output sample, is used. Furthermore in this report the main emphasis concerning convergence properties of adaptive filters will be on the following points:

- *Misadjustment* (J):
  A quantity of interest, describing the convergence properties of an adaptive filter, is the mean–squared error of the residual signal $J = E\{r^2\}$ with minimum $J_{\text{min}} = E\{s^2\}$. For an algorithm of interest, this parameter J provides a quantitative measure of the amount by which the final value of the mean–squared error, averaged over an ensemble of adaptive filters, deviates from the minimum mean–squared error that produces the unknown Wiener filter.
- *Relative Misadjustment* ($\tilde{J}$):
  The "relative" misadjustment is defined as:

$$\tilde{J} = \frac{E\{(e - \hat{e})^2\}}{E\{s^2\}} = \frac{J - J_{\text{min}}}{J_{\text{min}}} = \frac{J_{\text{ex}}}{J_{\text{min}}}. \tag{1.40}$$

- *Final Misadjustment* ($\overline{J}$):
  The fractional amount by which the steady state misadjustment exceeds the minimum attainable misadjustment J_{min} defined as: $\overline{J} = \tilde{J}_{\infty}$.
- *Rate of convergence (ν_{20}):*
  In signal estimation problems, such as echo cancellation, this rate of convergence is defined as a quantity which is related to the number of iterations required for an algorithm, in response to stationary signals, to decrease the quantity 10log($\tilde{J}$) by 20 dB. A fast rate of convergence allows the algorithm to adapt rapidly to a stationary environment of unknown statistics. Furthermore, it enables the algorithm to track statistical variations when operating in a nonstationary environment.

Although not a subject of research in this report some other points of interest when describing and comparing adaptive filters are:

- *Convergence region* ($\alpha_{\max}$):
  This is the region of allowed values for the adaptation constant α, for which the algorithm converges. Most of the times this region is given by

  $$0 < \alpha < \alpha_{\max}$$

  where $\alpha_{\max}$ is the largest value of the adaptation constant α that yields a stable algorithm.

- *Numerical properties*:
  When an algorithm is implemented numerically, inaccuracies are produced due to round–off noise and representation errors in the computer.

- *Structure*:
  This refers to the structure of information flow in an algorithm, determining the manner in which it is implemented in hard–ware form. For example, an algorithm whose structure exhibits high modularity, parallelism or concurrency is well–suited for implementation using Very Large Scale Integration (VLSI).

- *Robustness*:
  This refers to the ability of the algorithm to operate satisfactorily with ill conditioned input data.

- *Chip area:*
  The ultimate area needed to implement the algorithm on a chip.

Chapter 2

The Block Normalized Least Mean Square Algorithm

In literature often the suggestion is made that the Normalized Least Mean Square (NLMS) algorithm always requires a large number of iterations to converge when the Eigenvalue Ratio (ER) is large [7]. The main goal of this chapter is to show that also the squared magnitude of the system function (smf) of the initial difference vector (i.e. the difference between the optimal Wiener solution and the initial adaptive weights) plays an important role too. In order to do so first the NLMS algorithm is derived in Section 2.1. This algorithm makes one updating of all adaptive weights after every sampling period ($= T$ time units). The Block Normalized Least Mean Square (BNLMS) algorithm, that is described in Section 2.2, makes this updating only once every L samples ($= L \cdot T$ time units). Obviously with $L \geq 1$ the BNLMS algorithm is as a generalization of the NLMS algorithm. The analysis of the BNLMS algorithm is presented in Section 2.3. The results of this section are used to derive the most important convergence properties of the BNLMS algorithm. In Section 2.4 these results are interpreted in a physical way and are verified by experiments in Section 2.5. In Section 2.6 the main results are discussed. Literature with more information about the BNLMS algorithm is given in [46,15,17].

2.1 NLMS algorithm

In this section first the Normalized Least Mean Square (NLMS) algorithm is derived, as given in [68]. The popularity of the (N)LMS algorithm is largely due to the simplicity of its computational structure, low storage requirements, and the relative ease with which it may be mathematically analyzed. Fig. 2.1 shows an adaptive filter that uses the NLMS algorithm for the updating od the coefficients. As stated in Chapter 1 it is assumed that $\tilde{e}[k]$ is a sum of the convolution of signal $x[k]$ with an unknown optimum Wiener filter (w_{opt}) and a signal $s[k]$

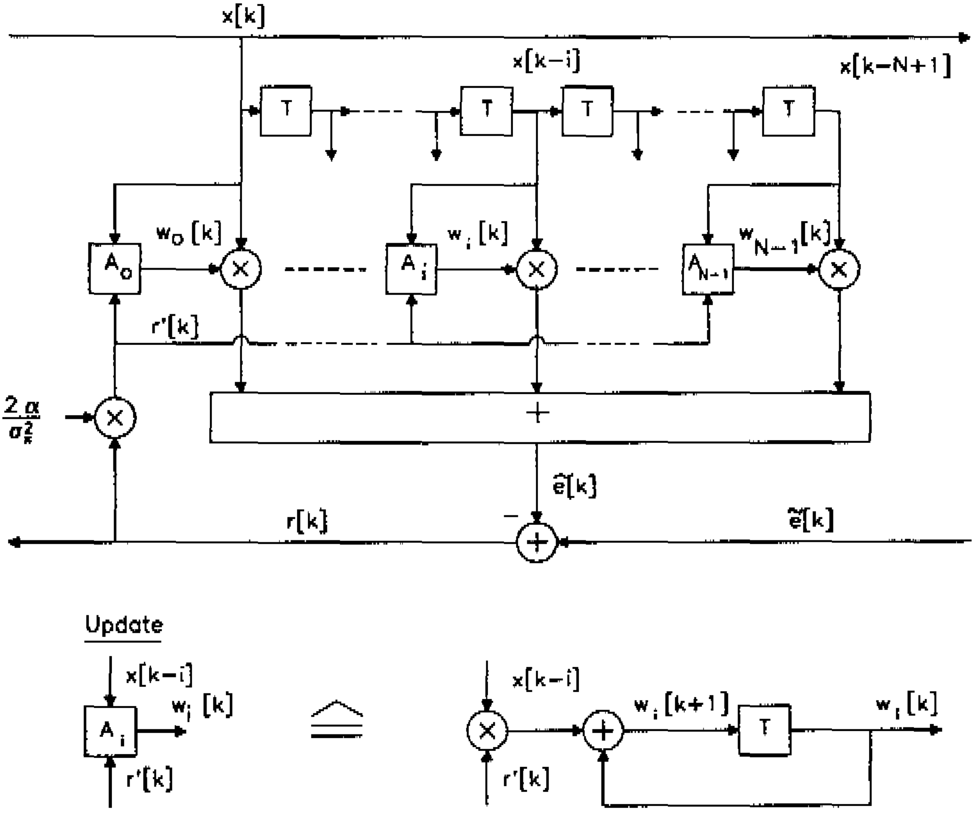

Figure 2.1: *Adaptive filter using the NLMS algorithm*

that is uncorrelated with signal $x[k]$. Thus:

$$\tilde{e}[k] = \sum_{i=0}^{N-1} x[k-i]w_{\text{opt},i} + s[k] = \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{w}}_{\text{opt}} + s[k]. \tag{2.1}$$

As shown in Fig. 2.1 a transversal structure is choosen for the adaptive filter and for this reason the residual signal $r[k]$ can be written as follows:

$$r[k] = \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{d}}[k] + s[k] \tag{2.2}$$

with the difference vector $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}_{\text{opt}} - \underline{\mathbf{w}}[k]$.

The LMS algorithm (without normalization) basically adapts, in average, to the unknown Wiener solution. This is done by minimizing the mean–squared error of the residual signal, that is given by:

$$J[k] = E\{r^2[k]\} = E\{\left(\underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{d}}[k] + s[k]\right)^2\} \tag{2.3}$$

This expression is a quadratic function of the adaptive filter coefficients having an absolute minimum J_{min}. To calculate this minimum the gradient of $J[k]$ with respect to the vector $\underline{\mathbf{w}}[k]$ is considered, that is defined as:

$$\underline{\nabla}[k] = \left(\frac{\delta J[k]}{\delta w_{N-1}[k]}, \cdots, \frac{\delta J[k]}{\delta w_0[k]}\right)^t. \tag{2.4}$$

Evaluation of this expression leads to:

$$\underline{\nabla}[k] = -2E\{\underline{\mathbf{x}}[k]r[k]\}. \tag{2.5}$$

The LMS algorithm is based on the steepest–descent method, that is given in the updating scheme

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] - \alpha\underline{\nabla}[k] \tag{2.6}$$

where α is an adaptation constant ($\alpha > 0$). As already stated in the introduction (Fig. 1.8), the gradient $\underline{\nabla}[k]$ always points away from the minimum J_{min}. The steepest–descent method therefore simply goes opposite to the gradient direction to find the minimum. In the LMS algorithm the gradient (2.5) is now approximated by:

$$\underline{\hat{\nabla}}_{LMS}[k] = -2\underline{\mathbf{x}}[k]r[k]. \tag{2.7}$$

Since this is a very rough (or noisy) estimate, the LMS algorithm is also referred to as the noisy gradient or gradient approximation algorithm. Using this estimate in the steepest-descent update equation (2.6) it leads to the LMS algorithm:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\underline{\mathbf{x}}[k]r[k]. \tag{2.8}$$

The residual signal $r[k]$ contains delayed signal samples $x[k-i]$, and from this it follows that the quantity $E\{2\alpha\underline{\mathbf{x}}[k]r[k]\}$ is depending on $E\{x^2[k]\} = \sigma_x^2$. Thus the convergence properties of the LMS algorithm are dependent on the variance σ_x^2. This effect can be cancelled by normalizing the adaptation constant α by an estimate $\hat{\sigma}_x^2[k]$ of the variance σ_x^2 of the input signal $x[k]$. This results in the NLMS algorithm:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \frac{2\alpha}{\hat{\sigma}_x^2[k]}\underline{\mathbf{x}}[k]r[k]. \tag{2.9}$$

A possible estimate for the variance is:

$$\hat{\sigma}_x^2[k] = \beta\hat{\sigma}_x^2[k-1] + (1-\beta)x^2[k] \quad \text{with} \quad 0 < \beta < 1. \tag{2.10}$$

Unless stated otherwise it is assumed that a perfect estimator for this variance is available, thus $\hat{\sigma}_x^2 = \sigma_x^2$. As a rough measure for complexity, the number of multiplications and divisions needed to calculate one new output sample are used. For the NLMS these numbers are in the order of the following figures:

$$\begin{aligned} MUL_{NLMS} &= 2N \\ DIV_{NLMS} &= 1. \end{aligned} \tag{2.11}$$

2.2 BNLMS algorithm

The NLMS algorithm adapts all weights every T time units. The Block Normalized Least Mean Square (BNLMS) algorithm performs this updating only once every $L \cdot T$ time units ($L \geq 1$). In this section it is shown that the BNLMS algorithm makes a more accurate estimate in comparison to the NLMS algorithm, while complexity is in the same order. On the other hand, since updating of the adaptive coefficients is performed less frequently, the BNLMS has a slower rate of convergence in comparison to the NLMS algorithm. However, as will be shown in Chapter 3, the main reason to apply the BNLMS algorithm is that for large L this algorithm can be implemented in a very efficient way by using block processing techniques.

In the LMS algorithm the $N-1-i^{th}$ element of the estimate of the gradient vector is given by:

$$(\hat{\underline{\nabla}}_{LMS}[k])_{N-1-i} = -2x[k-i]r[k] \quad \text{for} \quad i = N-1, \cdots, 1, 0. \tag{2.12}$$

This product $x[k-i]r[k]$ is an estimate of the crosscorrelation between the signals x and r. By averaging this crosscorrelation over a block with length $L(\geq 1)$, and calculating this estimate only once every L samples, this results in the following equation:

$$\begin{aligned}(\hat{\underline{\nabla}}_{BLMS}[kL])_{N-1-i} &= -\frac{2}{L}\sum_{q=0}^{L-1} x[kL-i-q]r[kL-q] \\ &= \underline{\mathbf{x}}_L^t[kL-i] \cdot \underline{\mathbf{r}}_L[kL]\end{aligned} \tag{2.13}$$

with

$$\begin{aligned}\underline{\mathbf{x}}_L[kL-i] &= (x[kL-i-L+1], \cdots, x[kL-i-1], x[kL-i])^t \\ \underline{\mathbf{r}}_L[kL] &= (r[kL-L+1], \cdots, r[kL-1], r[kL])^t.\end{aligned} \tag{2.14}$$

These equations shows that for the calculation of this estimate both the input signals and the residual signals have to be down sampled by a factor L. Furthermore, using the same approach as in the previous section, it is obvious that the normalized updating equation for the i^{th} coefficient ($i = 0, 1, \cdots, N-1$) is given by:

$$w_i[(k+1)L] = w_i[kL] + \frac{2\alpha_L}{L\sigma_x^2}\underline{\mathbf{x}}_L^t[kL-i]\underline{\mathbf{r}}_L[kL] \tag{2.15}$$

with α_L is the adaptation constant for the BNLMS algorithm. This mechanism is depicted in Fig. 2.2. For the updating of every adaptive coefficient, L consecutive samples $x[k-i], \cdots, x[k-i-L+1]$ of the input signal are needed. These samples are available in a delay line containing

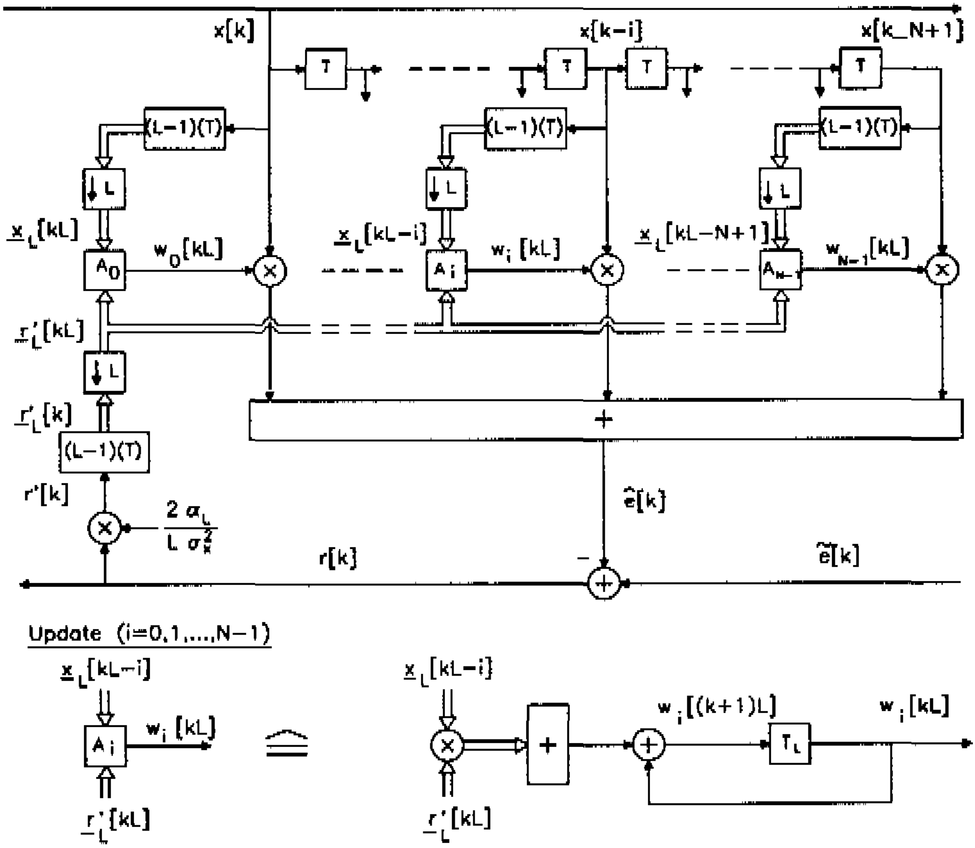

Figure 2.2: *Adaptive filter using BNLMS update algorithm*

$(L-1)$ delay elements as the vector $\underline{\mathbf{x}}_L[k-i]$. Each element of this vector is down sampled, at the same moment, by a factor L resulting in a vector $\underline{\mathbf{x}}_L[kL-i]$. An equivalent procedure is performed for the residual signal $r[k]$: First signal $r[k]$ is multiplied by $2\alpha_L/L\sigma_x^2$ (resulting in $r'[k]$) then stored in a delay line (resulting in $\underline{\mathbf{r}}'_L[k]$) and down sampled by a factor L (giving $\underline{\mathbf{r}}'_L[kL]$). Finally the update according equation (2.15) is performed every $T_L = L \cdot T$ time units. Using the $N \times L$ input signal matrix $\chi[kL]$, that is defined as:

$$\chi[kL] = \begin{pmatrix} \underline{\mathbf{x}}_L^t[kL-N+1] \\ \cdots \\ \cdots \\ \cdots \\ \underline{\mathbf{x}}_L^t[kL-1] \\ \underline{\mathbf{x}}_L^t[kL] \end{pmatrix} = (\underline{\mathbf{x}}[kL-L+1], \cdots, \underline{\mathbf{x}}[kL-1], \underline{\mathbf{x}}[kL]) \tag{2.16}$$

the BNLMS algorithm can be rewritten in vector–matrix notation as follows:

$$\underline{\mathbf{w}}[(k+1)L] = \underline{\mathbf{w}}[kL] + \frac{2\alpha_L}{L\sigma_x^2}\chi[kL]\underline{\mathbf{r}}_L[kL]. \tag{2.17}$$

The number of multiplications and divisions needed to produce L output samples is in the order of $L \cdot 2N$ and L respectively. To produce one new output sample this results in:

$$\begin{aligned} MUL_{BNLMS} &= 2N \\ DIV_{BNLMS} &= 1. \end{aligned} \tag{2.18}$$

2.3 Dynamic Behaviour

In order to gain more insight into the performance of the BNLMS algorithm, this section gives an analysis of this algorithm when a small adaptation constant α_L is used. A more general analysis is given in appendix A.

By using the difference vector $\underline{\mathbf{d}}[kL] = \underline{\mathbf{w}}_{\text{opt}} - \underline{\mathbf{w}}[kL]$ and rewriting the residual vector as:

$$\underline{\mathbf{r}}_L[kL] = \chi^t[kL]\underline{\mathbf{d}}[kL] + \underline{\mathbf{s}}_L[kL] \tag{2.19}$$

with $\underline{\mathbf{s}}_L[kL] = (s[kL-L+1], \cdots, s[kL-1], s[kL])^t$ the BNLMS update equation (2.17) can be written as follows:

$$\underline{\mathbf{d}}[(k+1)L] = \left(\mathbf{I} - \frac{2\alpha_L}{L\sigma_x^2}\chi[kL]\chi^t[kL]\right)\underline{\mathbf{d}}[kL] - \frac{2\alpha_L}{L\sigma_x^2}\chi[kL]\underline{\mathbf{s}}_L[kL]. \quad (2.20)$$

The analysis of various algorithms is performed by treating the adaptive weight vector $\underline{\mathbf{w}}[kL]$, and the difference vector $\underline{\mathbf{d}}[kL]$ as random vectors. The analysis is complicated by the fact that, during adaptation, the residual signal $\underline{\mathbf{r}}_L[kL]$, and therefore the vectors $\underline{\mathbf{w}}[(k+1)L]$ and $\underline{\mathbf{d}}[(k+1)L]$, are nonstationary, even if the signals x and s are stationary. Accordingly, quantities of interest, such as the mean–squared error $J[kL] = \frac{1}{L}E\{\underline{\mathbf{r}}_L^t[kL]\underline{\mathbf{r}}_L[kL]\}$ are functions of the number of iterations k.

First the average behaviour of the difference vector $\underline{\mathbf{d}}[kL]$ is studied as a function of k. By using the assumptions that signal x is independent of s, and using $E\{s[k]\} = 0$, the above update equation can be rewritten as:

$$E\{\underline{\mathbf{d}}[(k+1)L]\} = E\{(\mathbf{I} - \frac{2\alpha_L}{L\sigma_x^2}\chi[kL]\chi^t[kL]) \cdot \underline{\mathbf{d}}[kL]\}. \quad (2.21)$$

In this equation one can observe two different processes, with different time constant, as a function of time: the input signal (in matrix $\chi[kL]$) and the adaptive weights (in vector $\underline{\mathbf{d}}[kL]$). Since the adaptation constant α_L is assumed to be small, the variation in $\underline{\mathbf{d}}[kl]$ is much slower in comparison to the variation in matrix $\chi[kL]$. The input signal matrix $\chi[kL]$ and the difference vector $\underline{\mathbf{d}}[kL]$ may be separated under $E\{\cdot\}$, and for this reason the above difference equation is approximated as follows:

$$E\{\underline{\mathbf{d}}[(k+1)L]\} \approx (\mathbf{I} - \frac{2\alpha_L}{L\sigma_x^2}E\{\chi[kL]\chi^t[kL]\}) \cdot E\{\underline{\mathbf{d}}[kL]\}. \quad (2.22)$$

Using the definition of the input signal matrix $\chi[kL]$ and the autocorrelation matrix $\mathbf{R}$ it follows that

$$E\{\chi[kL]\chi^t[kL]\} = L \cdot \mathbf{R} \quad (2.23)$$

from which it follows that the above equation can be rewritten as:

$$E\{\underline{\mathbf{d}}[(k+1)L]\} \approx (\mathbf{I} - \frac{2\alpha_L}{\sigma_x^2}\mathbf{R})E\{\underline{\mathbf{d}}[kL]\}. \tag{2.24}$$

When the input signal is a non–white signal, the autocorrelation matrix **R** also contains elements unequal to zero outside the main diagonal. For such cases it follows that all elements of the vector $E\{\underline{\mathbf{d}}[(k+1)L]\}$ are interleaved. To overcome this problem both sides of this equation can be transformed as follows:

$$E\{\underline{\mathbf{D}}[(k+1)L]\} = (\mathbf{I} - \frac{2\alpha_L}{\sigma_x^2}\mathbf{\Lambda})E\{\underline{\mathbf{D}}[kL]\} \tag{2.25}$$

with:

$$\begin{aligned}\underline{\mathbf{D}}[kL] &= \mathbf{Q}^h\underline{\mathbf{d}}[kL] = \mathbf{Q}^h(\underline{\mathbf{w}}_{\text{opt}} - \underline{\mathbf{w}}[kL]) \\ &= \underline{\mathbf{W}}_{\text{opt}} - \underline{\mathbf{W}}[kL].\end{aligned} \tag{2.26}$$

Since $\mathbf{\Lambda}$ is diagonal the above set contains N uncoupled difference equations that can be solved separately. Using the initial vector (that is no random variable) $E\{\underline{\mathbf{D}}[0]\} = \underline{\mathbf{W}}_{\text{opt}} - \underline{\mathbf{W}}[0] = \underline{\mathbf{D}}[0]$ this results for $l = 0, 1, \cdots, N-1$ in:

$$E\{D_l[kL]\} = (1 - 2\alpha_L\overline{\lambda}_l)^k D_l[0] \tag{2.27}$$

or equivalently:

$$E\{W_l[kL]\} = \left(1 - (1 - 2\alpha_L\overline{\lambda}_l)^k\right) W_{\text{opt},l} + (1 - 2\alpha_L\overline{\lambda}_l)^k W_l[0] \tag{2.28}$$

with the relative eigenvalue $\overline{\lambda}_l = \lambda_l/\sigma_x^2$. From this equation it follows that the average behavior of the vector $E\{\underline{\mathbf{D}}[kL]\}$ converges to zero, or equivalently $E\{\underline{\mathbf{W}}[kL]\}$ converges to $\underline{\mathbf{W}}_{\text{opt}}$, for $k \to \infty$, provided that the following condition is satisfied:

$$0 < \alpha_L < \frac{1}{\overline{\lambda}_{\max}} \tag{2.29}$$

where $\overline{\lambda}_{\max}$ is the largest relative eigenvalue of the autocorrelation matrix **R**. An example of both the average and instantaneous convergence

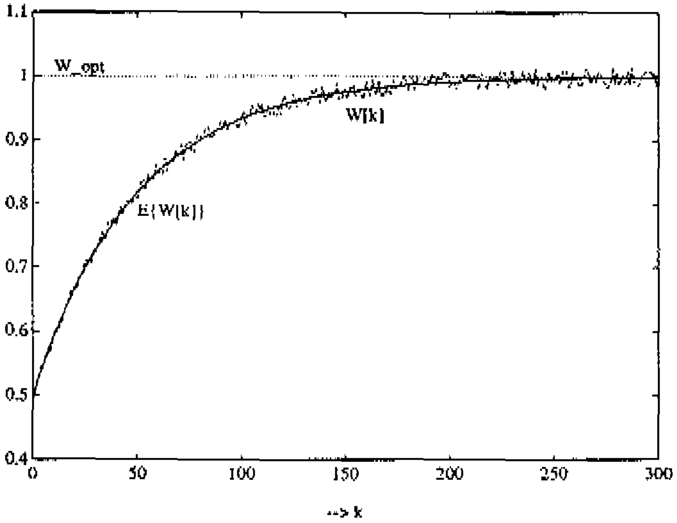

Figure 2.3: *Average and instantaneous behaviour of a single adaptive weight W*

behaviour of a single adaptive weight $W[kL]$, with $L = 1$, is plotted in Fig. 2.3. The conclusion of this analysis is that when the number of iterations k approaches infinity, provided that the adaptation constant α_L is set within bounds defined by equation (2.29), the average of the weight vector $\underline{\mathbf{W}}[kL]$ computed by the BNLMS algorithm converges to the unknown optimum Wiener solution $\underline{\mathbf{W}}_{\mathrm{opt}}$, or equivalently $\underline{\mathbf{w}}[kL]$ converges to $\underline{\mathbf{w}}_{\mathrm{opt}}$.

However this average convergence is not sufficient as far as the algorithm convergence is concerned. There is no guarantee that the average will converge within finite variance. Hence, analysis of the second order statistics is required in order to get more insight into the convergence properties of the algorithm.

As mentioned before a quantity of interest, describing the convergence properties of an adaptive filter with N weights and using a block update mechanism, is the mean squared error of a block with L residual signal samples:

$$J[kL] = \frac{1}{L}E\{\underline{\mathbf{r}}_L^t[kL]\underline{\mathbf{r}}_L[kL]\} \tag{2.30}$$

with minimum:

$$J_{\min} = J[kL]|_{\underline{\mathbf{w}}[kL]=\underline{\mathbf{w}}_{\text{opt}}} = \frac{1}{L}E\{\underline{\mathbf{s}}_L^t[kL]\underline{\mathbf{s}}_L[kL]\} = \sigma_s^2. \tag{2.31}$$

With signals x and s independent this quantity can be written as:

$$\begin{aligned} J[kL] &= \frac{1}{L}E\{(\chi^t[kL]\underline{\mathbf{d}}[kL] + \underline{\mathbf{s}}_L[kL])^t \cdot (\chi^t[kL]\underline{\mathbf{d}}[kL] + \underline{\mathbf{s}}_L[kL])\} \\ &= \frac{1}{L}E\{\underline{\mathbf{d}}^t[kL]\chi[kL]\chi^t[kL]\underline{\mathbf{d}}[kL]\} + J_{\min} \end{aligned} \tag{2.32}$$

The input signal matrix $\chi[kL]$ and the difference vector $\underline{\mathbf{d}}[kL]$ may again be separated under $E\{\cdot\}$ for small adaptation constant α_L. Furthermore by using $E\{\chi[kL]\chi^t[kL]\} = L \cdot \mathbf{R}$ the above expression can be approximated as:

$$J[kL] \approx E\{\underline{\mathbf{d}}^t[kL]\mathbf{R}\underline{\mathbf{d}}[kL]\} + J_{\min} = J_{\text{ex}}[kL] + J_{\min} \tag{2.33}$$

with $J_{\text{ex}}[kL]$ the excess mean–squared error. By using the unitary transformation $\mathbf{Q}$ this excess mean–squared error can be rewritten as:

$$\begin{aligned} J_{\text{ex}}[kL] &= E\{\underline{\mathbf{d}}^t[kL]\mathbf{Q}\mathbf{Q}^h\mathbf{R}\mathbf{Q}\mathbf{Q}^h\underline{\mathbf{d}}[kL]\} = E\{\underline{\mathbf{D}}^h[kL]\mathbf{\Lambda}\underline{\mathbf{D}}[kL]\} \\ &= \sum_{l=0}^{N-1} \lambda_l E\{|D_l[kL]|^2\} = trace\{\mathbf{\Lambda} \cdot \Delta[kL]\} \end{aligned} \tag{2.34}$$

with:

$$\Delta[kL] = E\{\underline{\mathbf{D}}[kL]\underline{\mathbf{D}}^h[kL]\}. \tag{2.35}$$

One of the quantities of interest:

$$\tilde{J}[kL] = \frac{J[kL] - J_{\min}}{J_{\min}} = \frac{J_{\text{ex}}[kL]}{J_{\min}} = \frac{\sum_{l=0}^{N-1} \lambda_l E\{|D_l[kL]|^2\}}{\sigma_s^2}. \tag{2.36}$$

This quantity is composed of a sum of product components $\lambda_l E\{|D_l[kL]|^2\}$ For this reason an expression is derived for each component, by first taking a closer look to $\Delta[kL]$. Transforming equation (2.20) with the unitary matrix $\mathbf{Q}$ results in:

$$\begin{aligned}\underline{\mathbf{D}}[(k+1)L] &= \left(\mathbf{I}-\frac{2\alpha_L}{L\sigma_x^2}\mathbf{Q}^h\chi[kL]\chi^h[kL]\mathbf{Q}\right)\underline{\mathbf{D}}[kL] \\ &\quad -\frac{2\alpha_L}{L\sigma_x^2}\mathbf{Q}^h\chi[kL]\underline{\mathbf{s}}_L[kL] \qquad (2.37)\end{aligned}$$

Using a small adaptation constant α_L it follows:

$$\begin{aligned}\Delta[(k+1)L] &= E\{\underline{\mathbf{D}}[(k+1)L]\underline{\mathbf{D}}^h[(k+1)L]\} \\ &= E\{\left(\mathbf{I}-\frac{2\alpha_L}{L\sigma_x^2}\mathbf{Q}^h\chi[kL]\chi^h[kL]\mathbf{Q}\right)\underline{\mathbf{D}}[kL] \\ &\quad \cdot\underline{\mathbf{D}}^h[kL]\left(\mathbf{I}-\frac{2\alpha_L}{L\sigma_x^2}\mathbf{Q}^h\chi[kL]\chi^h[kL]\mathbf{Q}\right)\} \\ &\quad +\frac{4\alpha_L^2}{L^2\sigma_x^4}E\{\mathbf{Q}^h\chi[kL]\underline{\mathbf{s}}_L[kL]\underline{\mathbf{s}}_L^h[kL]\chi^h[kL]\mathbf{Q}\}. \qquad (2.38)\end{aligned}$$

By using the assumption that the input signal (χ) and the difference vector ($\underline{\mathbf{D}}$) may be separated under $E\{\cdot\}$, and using the white noise assumption for signal s this equation can be approximated as follows:

$$\begin{aligned}\Delta[(k+1)L] &\approx \Delta[kL]-\frac{2\alpha_L}{\sigma_x^2}\mathbf{\Lambda}\Delta[kL]-\frac{2\alpha_L}{\sigma_x^2}\Delta[kL]\mathbf{\Lambda} \\ &\quad +\frac{4\alpha_L^2}{L\sigma_x^4}J_{\min}\mathbf{\Lambda}. \qquad (2.39)\end{aligned}$$

For one product component of the sum (2.36) only the diagonal elements of the above matrix $\Delta[kL]$ are needed. With the relative eigenvalue $\overline{\lambda}_l$, one product component can now be written as:

$$\lambda_l E\{|D_l[(k+1)L]|^2\} = \left(1-4\alpha_L\overline{\lambda}_l\right)\lambda_l E\{|D_l[kL]|^2\}+4\frac{\alpha_L^2}{L}\overline{\lambda}_l^2\sigma_s^2. \qquad (2.40)$$

Equations (2.36) and (2.40) fully describe the dynamic behavior of the BNLMS algorithm for small adaptation constant α_L. From equation (2.40) it is possible to calculate the contribution of each separate l^{th} product component to the total quantity $\tilde{J}[kL]$. Two valuable expressions that can be derived from equation (2.40) are:

- The final value, that is reached in steady state, given by:

$$\lim_{k\to\infty} \lambda_l E\{|D_l[kL]|^2\} = \lambda_l |D_l[\infty]|^2 = \frac{\alpha_L}{L} \cdot \overline{\lambda}_l \cdot \sigma_s^2. \tag{2.41}$$

- The number of samples needed for a decrease of the quantity $\lambda_l E\{|D_l[kL]|^2\}$ by a factor e is given by the integer value that is closest to the number:

$$\tau_l = \frac{-1}{\ln|1 - 4\alpha_L \overline{\lambda}_l|} \cdot L \approx \frac{1}{4\frac{\alpha_L}{L}\overline{\lambda}_l}. \tag{2.42}$$

  A small value for τ_l implies fast adaptation.

From these quantities the final misadjustment and the rate of convergence of the adaptive filter can be derived as follows:

Final Misadjustment:

The final misadjustment $\overline{J}$ is defined as:

$$\overline{J} = \lim_{k\to\infty} \left(\tilde{J}[kL]\right) = \frac{J_{\mathrm{ex}}[\infty]}{J_{\min}}. \tag{2.43}$$

Using equation (2.41) and $\sum_{l=0}^{N-1} \overline{\lambda}_l = N$ this leads to:

$$\overline{J} = \frac{\sum_{l=0}^{N-1} \lambda_l |D_l[\infty]|^2}{\sigma_s^2} = \frac{\alpha_L}{L} \cdot N. \tag{2.44}$$

Note that this quantity is independent of the input signal statistics.

Rate of Convergence:

In general it is difficult to speak about the rate of convergence of the whole process. After all it follows from equation (2.36) that the quantity $\tilde{J}[kL]$ is composed of a sum of N product components $\lambda_l E\{|D_l[kL]|^2\}$ each having its own rate of convergence. Hence a "local" time constant τ of the adaptive filter is defined as that time constant τ_l for which the product component results in the largest contribution to the quantity $\tilde{J}[kL]$.

As mentioned in the introduction the quantity ν_{20} is used in this report. This quantity gives the number of samples needed to reduce $10\log(\tilde{J}[0])$ by 20 dB, and thus ν_{20} is also a composition of different τ_l. Converting τ, that gives the number of samples to reduce this logarithmic quantity by 10log(e) dB, gives:

$$\nu_{20} = \frac{20\tau}{10\log(e)}. \tag{2.45}$$

Convergence region:

Until now it was assumed that the adaptation constant α_L was small. But what is small? In order to answer this question some knowledge must be available about the region of adaptation constants for which the adaptive filter still converges. For this an analysis is made of an adaptive filter without making the restriction of using a small adaptation constant. This analysis is given in Appendix A. In order to get an impression about this convergence area the results are summarized here. The convergence region of the BNLMS algorithm is given by:

$$\begin{aligned} \frac{1}{L}\sum_{l=0}^{N-1}\frac{\alpha_L\overline{\lambda}_l}{1-\alpha_L\overline{\lambda}_l} &< 1 \\ 0 < \alpha_L < \frac{1}{\overline{\lambda}_{\max}} &= \alpha_{\max} \end{aligned} \tag{2.46}$$

2.4 Physical Interpretation

In order to interpret the results of the previous section in a physical way, the Toeplitz–circulant approximation is used as discussed in Chapter 1, With the autocorrelation function $\rho[\tau]$ restricted to

$$\rho[\tau] = 0 \quad \text{for} \quad |\tau| > \tau_{\max} \quad \text{with} \quad \tau_{\max} \ll N \tag{2.47}$$

the excess mean–squared error of equation (2.33) can be rewritten as follows:

$$J_{\text{ex}}[kL] = E\{\underline{\mathbf{d}}^t[kL]\mathbf{R}\underline{\mathbf{d}}[kL]\} \approx E\{\underline{\mathbf{d}}^t[kL]\mathbf{C}\underline{\mathbf{d}}[kL]\}. \tag{2.48}$$

In this equation is $\mathbf{C}$ the circulant extension of the Toeplitz autocorrelation matrix $\mathbf{R}$ as discussed in Chapter 1 (Section 1.3). The approximation (2.48) imposes the following restriction on the first and the last $\tau_{\max} - 1$ components of the difference vector $\underline{\mathbf{d}}[kL]$:

$$(\underline{\mathbf{d}}[kL])_i = 0 \text{ for } \begin{cases} i \in \{0, 1, \cdots, \tau_{\max} - 2\} \\ i \in \{N - 1 - (\tau_{\max} - 2), \cdots, N - 2, N - 1\} \end{cases} \tag{2.49}$$

Note that this restriction becomes more and more true when the adaptation process continues, since $E\{\underline{\mathbf{d}}[kL]\} \to 0$ for $k \to \infty$. The circulant autocorrelation matrix can be diagonalized with the Fourier matrix $\mathbf{F}$, and thus (2.48) can be rewritten as:

$$\begin{aligned} J_{\text{ex}}[kL] &= E\{\underline{\mathbf{d}}^t[kL]\mathbf{F}^{-1}\mathbf{F}\mathbf{C}\mathbf{F}^{-1}\mathbf{F}\underline{\mathbf{d}}[kL]\} = \frac{1}{N}E\{\underline{\mathbf{D}}^h[kL]\mathbf{P}\underline{\mathbf{D}}[kL]\} \\ &= \frac{1}{N}\sum_{l=0}^{N-1} P_l E\{|D_l[kL]|^2\}. \end{aligned} \tag{2.50}$$

In this equation the transformed difference vector is defined as $\underline{\mathbf{D}}[kL] = \mathbf{F}\underline{\mathbf{d}}[kL]$ and the diagonal power matrix is given by $\mathbf{P} = \mathbf{F}\mathbf{C}\mathbf{F}^{-1}$.

In conclusion, within the above mentioned restrictions on the autocorrelation function (2.47) and the differnce vector (2.49), the results of the previous section can be physically interpreted by replacing the eigenvalue λ_l with the power P_l and the transformed difference vector $\underline{\mathbf{D}}[kL]$ is the Fourier transformation ($\mathbf{F}$) of the time domain difference vector $\underline{\mathbf{d}}[kL]$. These interpretations will be used in the following section, when experiments are carried out.

2.5 Experiments

The analytical results of Section 2.3 are verified in this section by exploring the influence of different parameters on the convergence properties of the BNLMS algorithm. For the experiments the system as given in Fig. 1.7 is used, with an adaptive filter of length $N = 128$. Two quantities of importance in the equations (2.36) and (2.40) are the eigenvalues λ_l and the transformed initial difference components

$|D_l[0]|^2 = |W_{\mathrm{opt},l} - W_l[0]|^2$. In the previous section it was argued that, within the restrictions (2.47) and (2.49), in these equations the eigenvalues λ_l may be replaced by the powers P_l and that the transformed difference vector is defined as $\underline{\mathbf{D}}[kL] = \mathbf{F}\underline{\mathbf{d}}[kL]$, with $\mathbf{F}$ the Fourier matrix.

In these experiments three different input signals (random, ma(1) and ar(1)) are used, from which the psdf are plotted in Fig. 1.9. The "unknown" Wiener system is a low–pass filter, whose impulse response w_{opt} and the absolute value of the frequency response are plotted in Fig. 2.4. In first instance, unless stated otherwise, an adaptation constant $\alpha_L = 1/12800$ is used, while the adaptive weights are initialized in such a way that all $|D_l[0]|^2$ are equal. Signal $s[k]$ is a white noise signal, independent of $x[k]$, with $E\{s^2[k]\} = \sigma_s^2 = J_{\min}$, that is chosen in such a way that 10log($\tilde{J}[0]$)=20 dB.

In the next subsection the parameters α_L, L, the input signal $x[k]$ and finally the initialization of the adaptive filter coefficients are varied separately.

2.5.1 Influence of adaptation constant α_L

When using a white noise input signal $x[k]$, the equations for final misadjustment and rate of convergence reduce to:

$$\overline{J} \approx \left(\frac{\alpha_L}{L}\right) \cdot N \quad \text{and} \quad \nu_{20} \approx \frac{1.15}{\left(\frac{\alpha_L}{L}\right)}. \tag{2.51}$$

From these equations it follows that by increasing the adaptation constant α_L the adaptive filter converges faster (ν_{20} smaller), but it becomes less accurate ($\overline{J}$ larger), while decreasing α_L leads to the opposite result. In order to show this, Fig. 2.5 gives the quantity 10 log($\tilde{J}[kL]$) as function of the discrete time index k. The first curve is plotted for $L = 1$ and $\alpha_L = \alpha = 1/12800$ resulting in 10log($\overline{J}$) $\doteq$ −20dB and $\nu_{20} \approx 14720$ samples. The second curve gives the result when the adaptation constant is doubled (1/12800 → 1/6400). The final misadjustment is indeed 3dB (a factor 2) worse, while the rate of convergence is twice as fast ($\approx$ 7360 samples). The smooth lines are the analytical

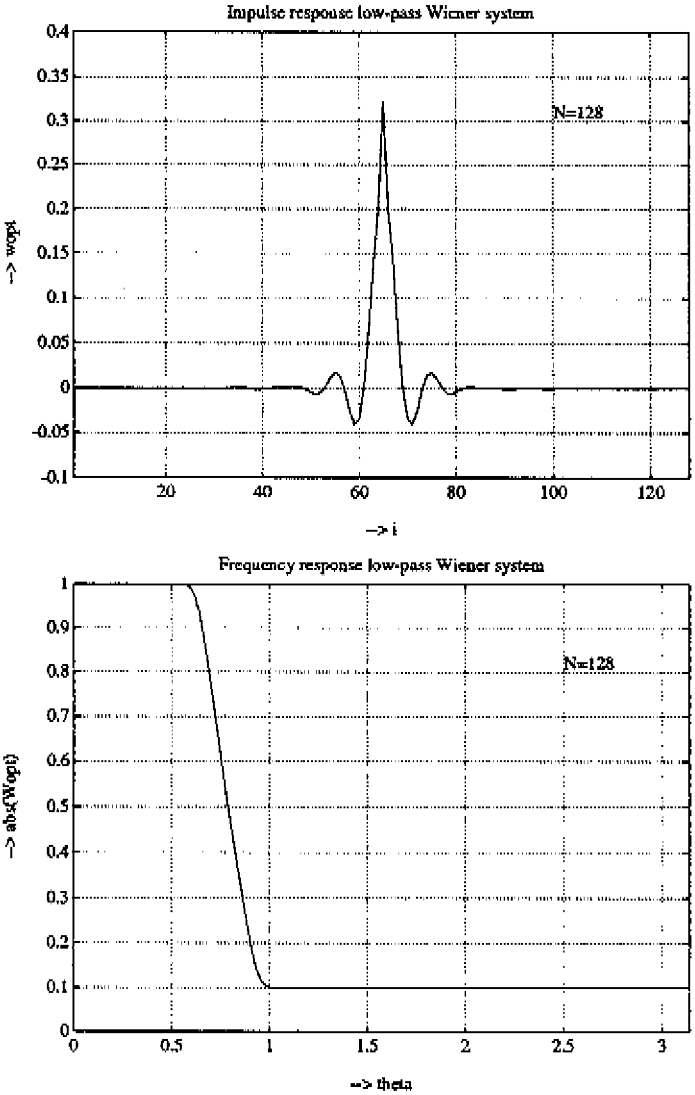

Figure 2.4: *Impulse and frequency response of "unknown" Wiener system*

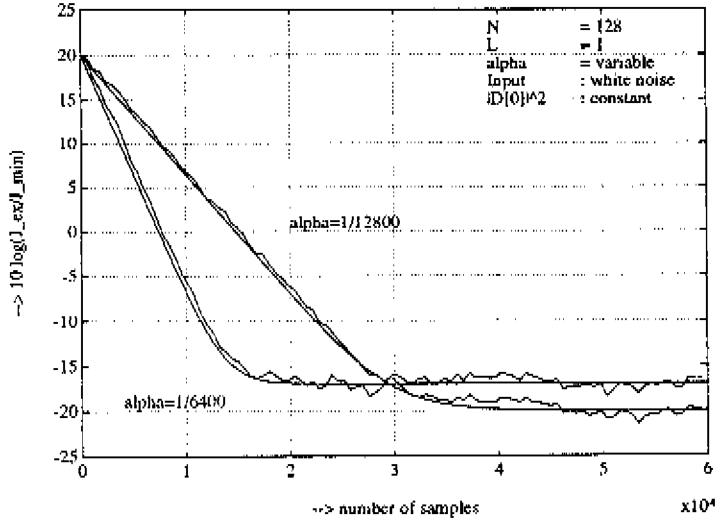

Figure 2.5: *10 log($J_{ex}[kL]/J_{min}$) as function of the number of samples*

results from equations (2.36) and (2.40), or equivalently the ensemble averages. The "noisy" lines represent experimental results that are single runs from the PC/DSP programs.

Finally it follows from the above equations (2.51) that by increasing the number of adaptive coefficients N the final misadjustment $\overline{J}$ will increase, since more and more adaptive weights are fluctuating around their final steady state value. On the other hand the initial rate of convergence ν_{20} is not depending on N.

2.5.2 Influence of block length L

From equations (2.51) it follows that for a white noise input signal the adaptation constant α_L can be eliminated as follows:

$$\overline{J} = \frac{1.15 \cdot N}{\nu_{20}}. \tag{2.52}$$

In Fig. 2.6 this theoretical function ($10\log(\overline{J})$) is plotted (solid line) as a function of the rate of convergence (ν_{20}), with α_L as a parameter.

The lower right part of the curve corresponds to very small adaptation

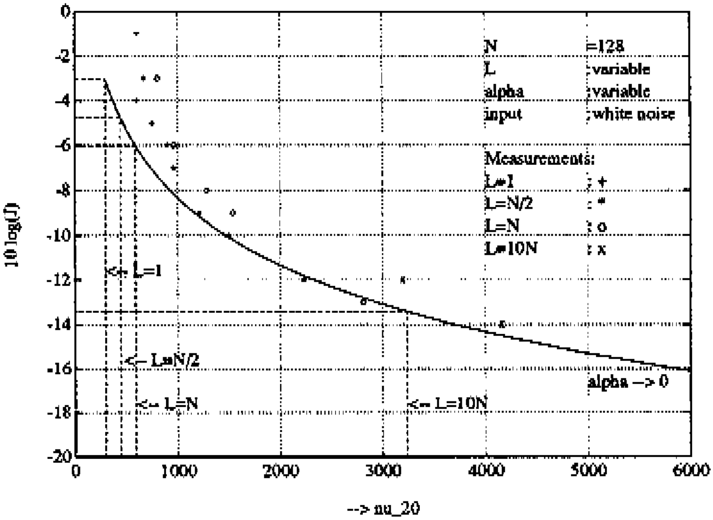

Figure 2.6: *10 log($\overline{J}$) as function of ν_{20} for different L*

constant ($\alpha_L \rightarrow 0$). From (2.52) it follows that convergence properties are independent of the block length L. On the other hand from equation (2.46) it follows that the convergence region is dependent on L, and for white noise given by:

$$0 < \alpha_L < \alpha_{\max} = \frac{L}{N+L}. \tag{2.53}$$

In Fig. 2.6 the adaptation constant α_L is varied for different L in the range $0 < \alpha_L < \alpha_{\max}/2$. The different "maximum" points are indicated for $L = 10N$, $L = N$, $L = N/2$ and $L = 1$. From this it follows that for large adaptation constant the NLMS algorithm outperforms the BNLMS algorithm. This algorithm is "too accurate", and thus too slow in comparison to the NLMS algorithm. On the other hand a strong point of the BNLMS algorithm is that, for large L, it can be implemented very efficiently in frequency domain, as will be shown in Chapter 3.

For different values of L measurements are carried out with the PC/DSP programs, and the results are marked in Fig. 2.6 with different symbols. From these results it follows that the derived theoretical results match reasonably well with the experimental results, even for large adaptation constant α_L. An explanation for this may be as follows:
The main assumption for the analysis was a small adaptation constant, in order to be able to separate the input signal $x[k]$ and the adaptive weights $w_i[k]$ under $E\{\cdot\}$. For large adaptation constant the residual signal is decreasing very fast, resulting in a steady state final value of the adaptive weights, and again the above mentioned separation under $E\{\cdot\}$ may be applied.

2.5.3 Influence of coloured input signal

For this experiment three different input signals (white noise, ma(1) and ar(1)) are used, as described in Chapter 1. The white noise signal has a flat psdf, the ma(1) signal has a psdf containing more power at the higher frequencies, while the ar(1) signal is chosen to have more power in the lowest frequencies. The Eigenvalue Ratio (ER) of these last two signals equals 100. These spectra are plotted in Fig. 1.9. The experimental results are plotted in Fig. 2.7. These results show that a large ER of the input signal can indeed slow down the adaptation process in comparison to the white noise case [26]. But what is slow? Convergence of the adaptive filter with an ar(1) or an ma(1) signal as input is initially faster than using a white noise signal. As mentioned in the introduction chapter, all signal models generate signals with normalized spectra. Thus both ar(1) and ma(1) signals have spectral values larger and smaller than the white noise signal spectrum. The larger parts result in a faster initial rate of convergence, while the smaller part results in a slower rate of convergece at the end. From this and Fig. 2.7 it follows that even when applying different input signals, with equal ER, convergence properties of the BNLMS algorithm can differ.

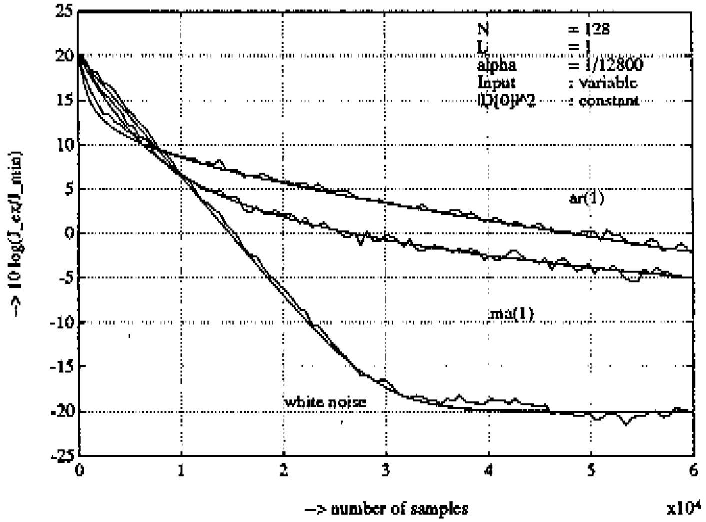

Figure 2.7: *Convergence of BNLMS algorithm with different input signals*

2.5.4 Initialization

In many practical situations the adaptive weights are initialized with zeros. This implies that the initial difference vector is given by: $|D_l[0]|^2 = |W_{l,\mathrm{opt}}|^2$. In this experiment this "unknown" Wiener filter is a low–pass function, as plotted in Fig. 2.4. The input signals that are used are the same as of the previous subsection. The results of this experiment are plotted in Fig. 2.8. From these curves it follows that an input signal with a large ER can both slow down (ma(1) signal) or speed up (ar(1) signal) the adaptation process, in comparison to the white noise case. This depends on the "similarity" between the psdf of the input signal and the squared magnitude of the system function (smf) from the initial difference vector. In this experiment the spectrum of the ar(1) signal and the smf of the initial difference vector have much resemblance, since both have a "low–pass" character. This results in a fast rate of convergence. The ma(1) signal however has a "high–pass" character, which results in a very slow adaptation process.

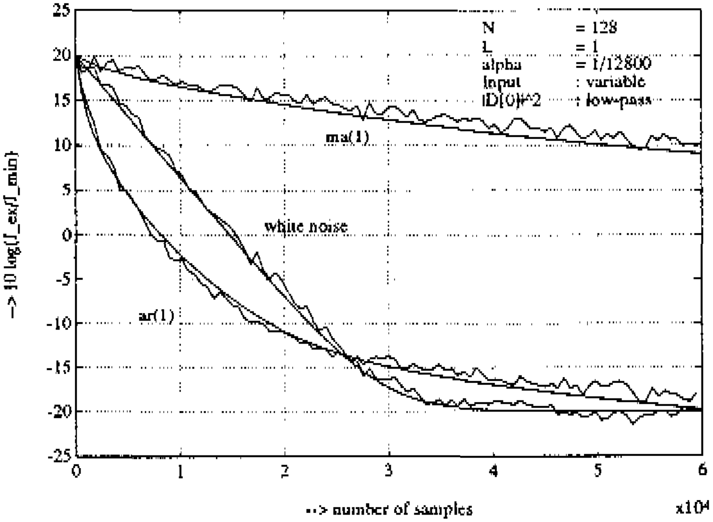

Figure 2.8: *Convergence of BNLMS with zero initialization and different input signals*

2.6 Discussion

In this chapter it is shown that, for small adaptation constant α_L, the BNLMS and NLMS algorithms have equal convergence properties. For large adaptation constant NLMS outperforms BNLMS.

When denoting the number of multiplications and divisions needed to calculate one new output sample as a rough measure of complexity , the BNLMS and the NLMS algorithm have equal complexity. The BNLMS algorithm however can be implemented in frequency domain in an efficient way for large L, by using FFTs, which will be shown in the next chapter.

The dynamic behaviour of the adaptive filter using the BNLMS algorithm is fully described by equations (2.36) and (2.40). From the discussion in Section 2.4 it follows that, within the restrictions (2.47) and (2.49), in these equations the eigenvalues λ_l may be approximated by the power P_l and that $\underline{\mathbf{D}}[kL]$ is the Fourier transformation of the difference vector $\underline{\mathbf{d}}[kL]$. With this the quantity $E\{|D_l[kL]|^2\}$ represents the smf of the difference vector.

From the equations and experimental results it follows that not only the ER of the input signal is important to descibe the convergence properties of the adaptive filter. It is shown that both the psdf of the input signal and its resemblance with the smf of the initial difference vector play an important role too.

In later chapters of this report some techniques are given to decorrelate the input signal by normalizing each component l. This is done by dividing out the eigenvalue λ_l, represented by the power P_l. The result of this decorrelation is that convergence properties will equal the "white noise case". Or equivalently it will lead to an "average" convergence result of the adaptive filter, that is not dependent on the input signal statistics. Without this normalization convergence may be worse or better, depending both on the psdf of the input signal and the resemblance between this psdf and the smf of the initial difference vector. The conclusion is that when having enough a priori information about the input signal statistics and the unknown optimum Wiener filter, it may be better not to decorrelate the input signal, when both spectra have much resemblance. This situation occurs in the acoustic echo canceller, where both the psdf of the input (speech) signal and the smf of the acoustic echo path have a "low-pass" character. On the other hand, when there is no resemblance, or when the a priori knowledge about the input signal statistics and/or the unknown optimum Wiener filter is not available ,decorrelation is a good "average" solution.

Chapter 3

Efficient Implementation of BNLMS algorithm

From the previous Chapter it follows that the two main operations in the BNLMS algorithm are: 1) a linear convolution, to perform the filtering of the input signal with the adaptive weights, and 2) a linear correlation, to calculate an estimate of the gradient that is needed for the update of the adaptive weights. For large filter lengths N these operations can be carried out very efficiently in frequency domain by using Fast Fourier Transforms (FFTs) for the transformation between time– and frequency–domain [37]. Overlap–save and overlap–add are two wellknown techniques to convolve an infinite length input sequence (e.g. $x[k]$) with a finite length impulse response (e.g. N adaptive weights $w_i[kL]$). With these methods the infinite length input sequence is split into segments which are processed separately by applying block processing techniques. The desired signal is a composition of these separate signal. The way of splitting the input sequence and composing the desired result differs for both methods.

In literature [9] it is asserted that, for complexity reasons, in adaptive filter configurations the overlap–save method is to be preferred to the overlap–add method. The main goal of this chapter is to contradict this statement [52]. It is shown that the only limitation of the overlap–add method, used in adaptive filter configurations, is that the choice of the parameters is more restricted in comparison to the overlap–save method. This however is a direct consequence of the way of processing in the overlap–add method.

Both methods will be explained first for fixed filter coefficients and after that for adaptive coefficients. All methods will be implemented efficiently in frequency domain by using FFTs and block processing techniques. Section 3.1 describes the overlap–save method for fixed filters. The results of this section are used in Section 3.2 to derive an efficient implementation of the BNLMS algorithm for large filter length N. Section 3.3 describes the overlap–add method for fixed– and adaptive filters. For adaptive filters this method leads to a more complex result in comparison to the overlap–save method. The reason for this is that during the calculations in the overlap–add method a previously calculated result has to be added to the present one, while the adaptive weights have changed in the meanwhile. In Section 3.4 a method is given to implement the overlap–add method for adaptive filters in an efficient way. Applying this to the BNLMS algorithm leads to an efficient implementation of this algorithm with a complexity comparable to that of the overlap–save method. The chapter is concluded with a discussion in Section 3.5.

3.1 Overlap–save method for fixed filters

This method is based on the partial convolution of a length B segment of the input signal $x[k]$ and a length N weight vector $\underline{\mathbf{w}}$. With $B = N - 1 + L$ this method generates L new output samples $\hat{e}[k]$ each step. This method can be implemented with DFTs, or FFTs when B is a power of two: it is depicted for fixed filters in Fig. 3.1. The input signal $x[k]$ is split into segments of length B that have an overlap with the previous segment of $N - 1$ samples. This segmentation with an overlap of input signal samples is carried out by the delay line and down samplers shown in the figure. The result is a length B vector $\underline{\mathbf{x}}[kL]$. Furthermore the length N weight vector $\underline{\mathbf{w}} = (w_{N-1}, \cdots, w_0)^t$ is first mirrored and than added with zeros to a vector of length B. The (cyclic) convolution is carried out in frequency domain by a multiplication of the transformed weight vector by the transformed input signal vector. The result is

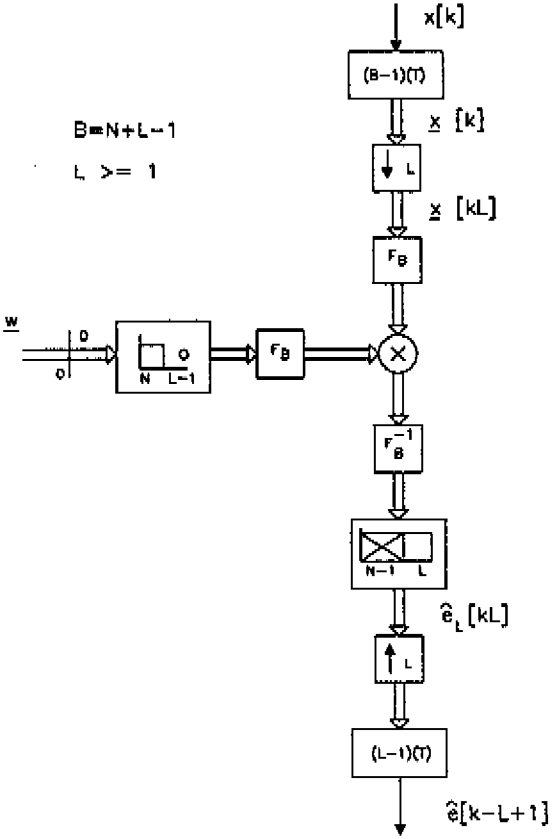

Figure 3.1: *Overlap–save method for fixed filters implemented with FFTs*

transformed back to time domain by an inverse FFT. Only L out of B samples from this cyclic convolution represent a linear convolution result. Thus $N-1$ samples have to be discarded, resulting in a length L vector $\underline{\hat{\mathbf{e}}}_L[kL]$. The original sample rate is obtained by upsampling this vector with a factor L and desegmenting it into samples $\hat{e}[k-L+1]$, that is the output signal of a transposed delay line. Applying this block processing technique, results in a processing delay of L samples ($=L\cdot T$ time units).

From Fig. 3.1 it follows that this method costs 3 Fourier transforms. Note that one FFT is superfluous if the weights $\underline{\mathbf{w}}$ are constant. When B is a power of two these can be implemented with FFTs. The complexity of each FFT is roughly equal to $\frac{B}{2}^2\log(B)$ multiplications, with each complex multiplication equal to 4 real multiplications. In general B will not be equal to a power of two. For simplicity reasons however it is assumed in this thesis that all DFTs can be implemented as FFTs. In due course the exact length of B can always be changed in such a way that it matches the nearest power of two.

Furthermore it follows from Fig. 3.1 that two complex-valued length B vectors have to be multiplied. Since both the input signal $x[k]$ and the weight vector $\underline{\mathbf{w}}$ are real, the resulting vectors in frequency domain have symmetry properties and only half of the frequency components have to be calculated. The number of real multiplications needed to calculate one new output sample with the implementation of Fig. 3.1 is roughly given by:

$$\frac{\frac{1}{2}(3\cdot 4\cdot \frac{B}{2}^2\log(B)+4B)}{L}=\frac{1}{2}\left(\frac{3\cdot 2B^2\log(B)+4B}{L}\right). \tag{3.1}$$

On the other hand when implementing Fig. 3.1 with a transversal filter in time domain, each new output sample costs N real multiplications. Comparing these two complexity numbers shows that for large B ($=N+L-1$) the overlap-save method, implemented with FFTs as depicted in Fig. 3.1, is much more efficient.

Note:

In practical situations not only the number of (real) multiplications is of importance when realising an FFT for large B in a Digital Signal

Processor (DSP) or on a chip . Also the storage, needed for the internal butterfly results, must be counted, and incorporated in the eventual cost of the filter. However, as mentioned before in this thesis only the number of multiplications/divisions are counted.

3.2 Overlap–save implementation of BNLM

In this section the overlap–save method of the Section 3.1 is used to implement the BNLMS algorithm of Fig. 2.2. The result is depicted in Fig. 3.2. As mentioned before two main operations in the BNLMS algorithm are the linear convolution, to perform the filtering of the input signal with the adaptive weights, and a linear correlation, to calculate the gradient estimation that is needed for the update of the adaptive weights. These operations are carried out in the blocks "CONVOLUTION" and "CORRELATION" respectively in Fig. 3.2. The convolution is a straightforward replica of Fig. 3.1, except for the return to the original sampling rate. Here the signal $\tilde{e}[k]$ is segmented and down sampled in a length L vector $\underline{\tilde{e}}_L[kL]$. The length L residual signal vector is given by $\underline{r}_L[kL] = \underline{\tilde{e}}_L[kL] - \underline{\hat{e}}_L[kL]$. The return to the original sampling rate is carried out by upsampling this residual signal vector and applying this result to a transposed delay line, resulting in the delayed residual signal $r[k-L+1]$. Before calculating the correlation between the residual signal vector $\underline{r}_L[kL]$ and the input signal vector $\underline{x}[kL]$, each element of $\underline{r}_L[kL]$ is first multiplied by the adaptation constant $(2\alpha_L)/(L\sigma_x^2)$. To correlate two signals is an equivalent operation as to a convolve two signals, except for an extra mirroring of the input signal. This mirroring is carried out in frequency domain by using the conjugate $(*)$ operator. The result of the cyclic convolution contains only N correct linear convolution values. This result has to be mirrored because the ordering is chosen such that element zero of the vector $\underline{w}[kL]$ is $w_{N-1}[kL]$.

Again as a rough measure of complexity the number of real multiplications and divisions, needed to produce one output sample, is used. For the efficient overlap–save implementation of the BNLMS algorithm

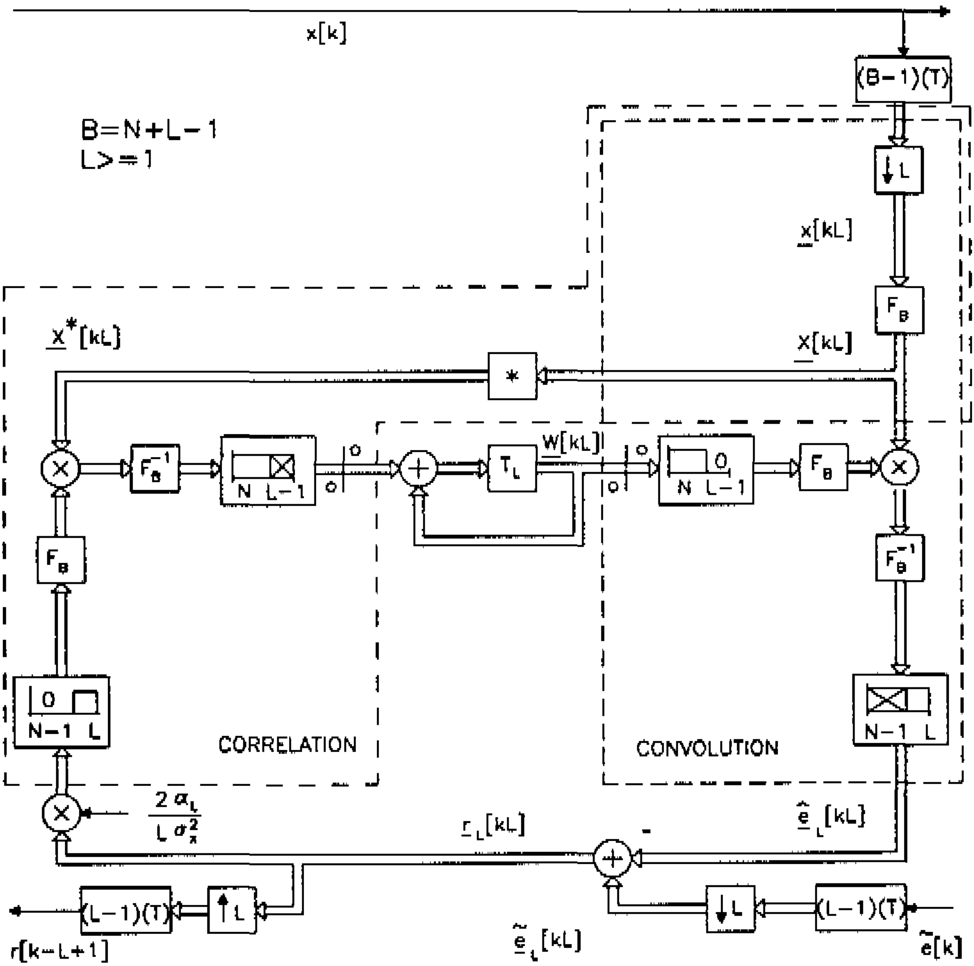

Figure 3.2: *Overlap–save implementation of BNLMS, using 5 FFTs*

with 5 FFTs, as depicted in Fig. 3.2, these numbers are as follows:

$$\begin{aligned} MUL_{EF-BNLMS} &= \frac{1}{2}\left(\frac{5 \cdot 2B^2 \log(B) + 2 \cdot 4B}{L}\right) \\ DIV_{EF-BNLMS} &= 1. \end{aligned} \qquad (3.2)$$

Note that this quantities give an order of magnitude and are not ment to be exact. For example the multiplication needed for the scaling with the number $2\alpha/L\sigma_x^2$ and the calculations needed for the estimate $\hat{\sigma}_x^2$ are not included. Comparing these complexity numbers with the complexity number of the BNLMS algorithm, equation (2.18), it follows that for large B ($=N+L-1$) the implementation as depicted in Fig. 3.2 is more efficient.

Notes:

- In Fig. 3.2 it is possible to combine the two mirror operations, one before and one after the updating of the adaptive weight vector, and leave them out. However in order to keep the separate implementations of the convolution and correlation operations visible, this has not been done here in the figure.
- In contrast to the implementation in Fig. 2.2, here both the update and the filter use block processing techniques, resulting in a processing delay of $(L-1)$ samples. The implementation of Fig. 2.2 has no processing delay, since only the update of the adaptive weights uses block processing techniques.
- The FFTs can be implemented efficiently when B is a power of two. For this reason N is generally chosen as a power of two, and $B = 2N$, resulting in $L = N+1$ new samples for each iteration.

3.3 Overlap–add for fixed and adaptive filters

The overlap–add method is based on the calculation of a complete convolution of a length L segment of the input signal $x[k]$ and a length N weight vector $\underline{w}$. Each step generates L new output samples $\hat{e}[k]$.

The method for fixed filters, implemented with FFTs and using block processing techniques, is depicted in Fig. 3.3 (left hand figure) with $B = N + L - 1$. The input signal $x[k]$ is split into segments of length L. After that the signal vector is down-sampled by a factor L and filled up with $N - 1$ zeros. The resulting vector is applied to an FFT of length B. The N length weight vector $\underline{\mathbf{w}}$ is mirrored and added with $L - 1$ zeros. This length B vector is transformed to frequency domain, resulting in the vector $\underline{\mathbf{W}}$. The (cyclic) convolution is carried out in frequency domain by multiplication of these two transformed vectors. The result of this multiplication is transformed back to time domain with an inverse FFT. In this way a complete linear convolution between the length L segment of the input signal and the length N weight vector is calculated by a cyclic convolution. The desired linear convolution of the infinite length input signal and the length N weight vector is composed as follows: In each iteration k the last $N-1$ samples of the previous iteration $(k-1)$ have to be added to the present result. This is done by first (circular) shifting over L samples then discarding the last L samples and after that adding L zeros, and delaying over $T_L = L \cdot T$ time units. Only L values are correct linear convolution samples. For simplicity reasons it will be assumed that all operations are such that only one addition is needed, as depicted in the right hand side of Fig. 3.3 . This leads to the condition: $L \geq N - 1$.

The problem with the above mentioned method for adaptive filters is that the last addition is not allowed any more: After all, from iteration $(k-1)$ to k the adaptive weights have changed. The right hand side of Fig. 3.3 shows the overlap–add method when applied to filters with adaptive weights. The input signal vector, in frequency domain, is delayed. Both this delayed vector and the present frequency domain input signal vector are multiplied by the adaptive weight vector $\underline{\mathbf{W}}[kL]$. When applying this procedure to the implementation of the BNLMS algorithm the result is an implementation with 7 FFTs [8].

3.4 Efficient overlap–add method for adap tive filters

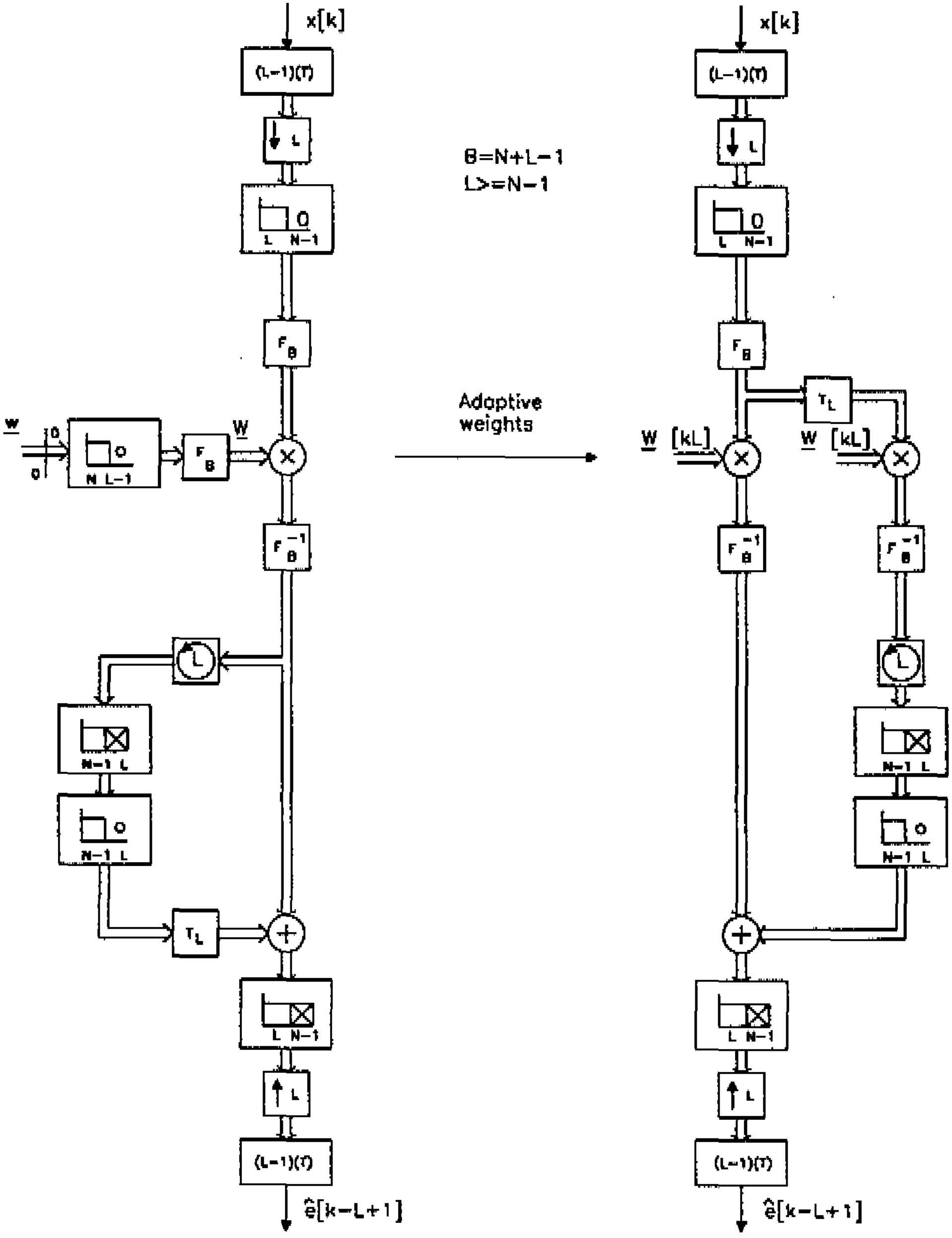

Figure 3.3: *Overlap–add method for fixed– and adaptive–filters implemented with FFTs*

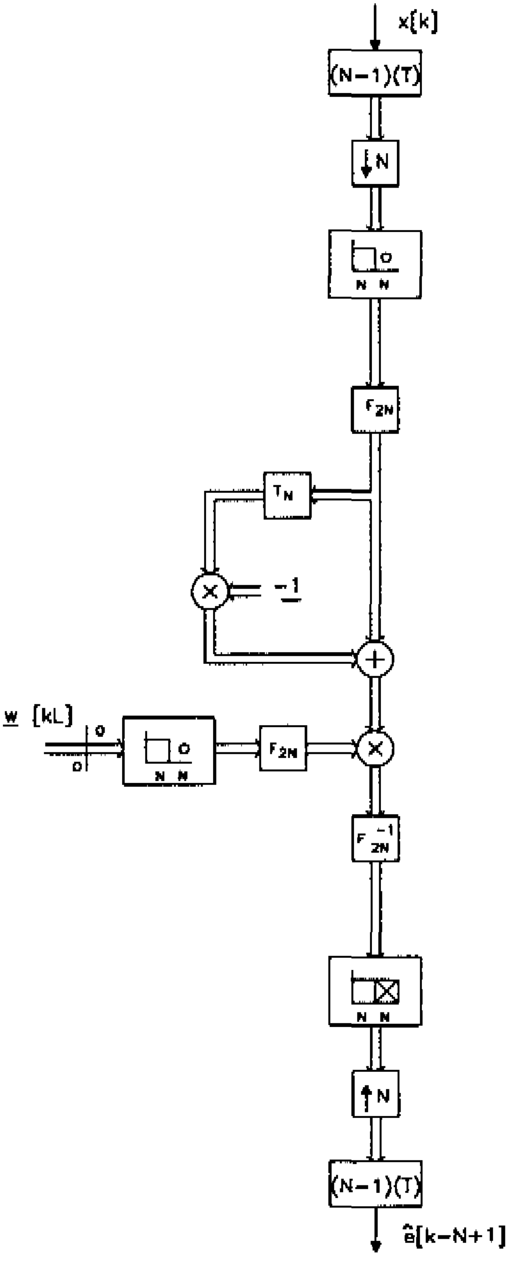

Figure 3.4: *Efficient overlap–add method for adaptive filters*

In this section it is shown that using the overlap–add method for the BNLMS algorithm, implies a restriction in the choice of the parameters N and L. After that an efficient implementation is derived [52].

As mentioned before the BNLMS algorithm needs a convolution and a correlation. When applying the overlap–add method, as explained in the previous secion, the result of the convolution is a length L signal vector $\underline{\hat{e}}_L[kL]$. The only condition for the overlap–add method was: $L \geq N-1$ in order to have only one addition of a segment with one previous segment. This length L vector $\underline{\hat{e}}_L[kL]$ results in a length L residual vector $\underline{r}_L[kL]$, that is used to calculate the needed correlation with the input signal. The result of this correlation, that is an estimate of the gradient vector, must generate a length N vector. This leads to the choice $L = N$.

On the other hand it is possible to combine, under certain restrictions, two FFTs of Fig. 3.3. The two windows, that throw away the last L samples and augment this with L zeros, can be combined with the window after the addition point if $L = N - 1$. Furthermore the cyclic shift can easily be implemented in frequency domain, by multiplying each frequency component l by $e^{-j(2\pi/B)Ll}$. By doing so the two FFTs can be combined to one FFT after the addition point, while the two multiplications with the adaptive weights can be performed after the addition point too.

A compromise between the two above mentioned conditions can be found by segmenting the input signal in length N vectors and add these with N zeros, thus $B = 2N$. The FFTs can still be combined in this way while moreover the cyclic shift in frequency domain is now given by $(e^{-j\pi})^l = (-1)^l$ for $l = 0, \cdots, N-1$. The only drawback of this choice is that every iteration one superfluous output sample is generated, but for large N and L this does not causes a real problem. This method is an efficient overlap–add implementation for adaptive filters as shown in Fig. 3.4. Note that in this figure the vector $\underline{-1}$ is $2N$ dimensional vector, from which the components are alternating 1 and -1, defined as:

$$\underline{-1} = (1, -1, 1, -1, \cdots)^t. \tag{3.3}$$

This result can be used in a straightforward way to the BNLMS algorithm [52], and leads to the implementation as depicted in Fig. 3.5, where 5 FFTs are used in stead of 7 FFTs.

3.5 Discussion

Efficient implementations of the BNLMS algorithm are given using 5 FFTs for both the overlap–save and the overlap–add method. The used block processing technique results in an extra processing delay. At first instance it seems that, for adaptive filter configurations, the overlap–add method is more restricted in the choice of the parameters L and B in comparison to the overlap–save method. This however is a direct consequence of the overlap–add technique. It is shown that, when using the overlap–add method for adaptive filters, a good compromise is found by using length N segments of the input signal $x[k]$ and length $B = 2N$ FFTs. In many practical situations the length of the FFT is chosen as $B = 2N$. For this case it is shown that, in contrast to a statement in literature [9], both overlap–save and overlap–add can be implemented with five FFTs. On the other hand, when L is chosen more freely (such that $B \neq 2N$), it is shown that the adaptive filter structure can not be implemented with the overlap–add method.

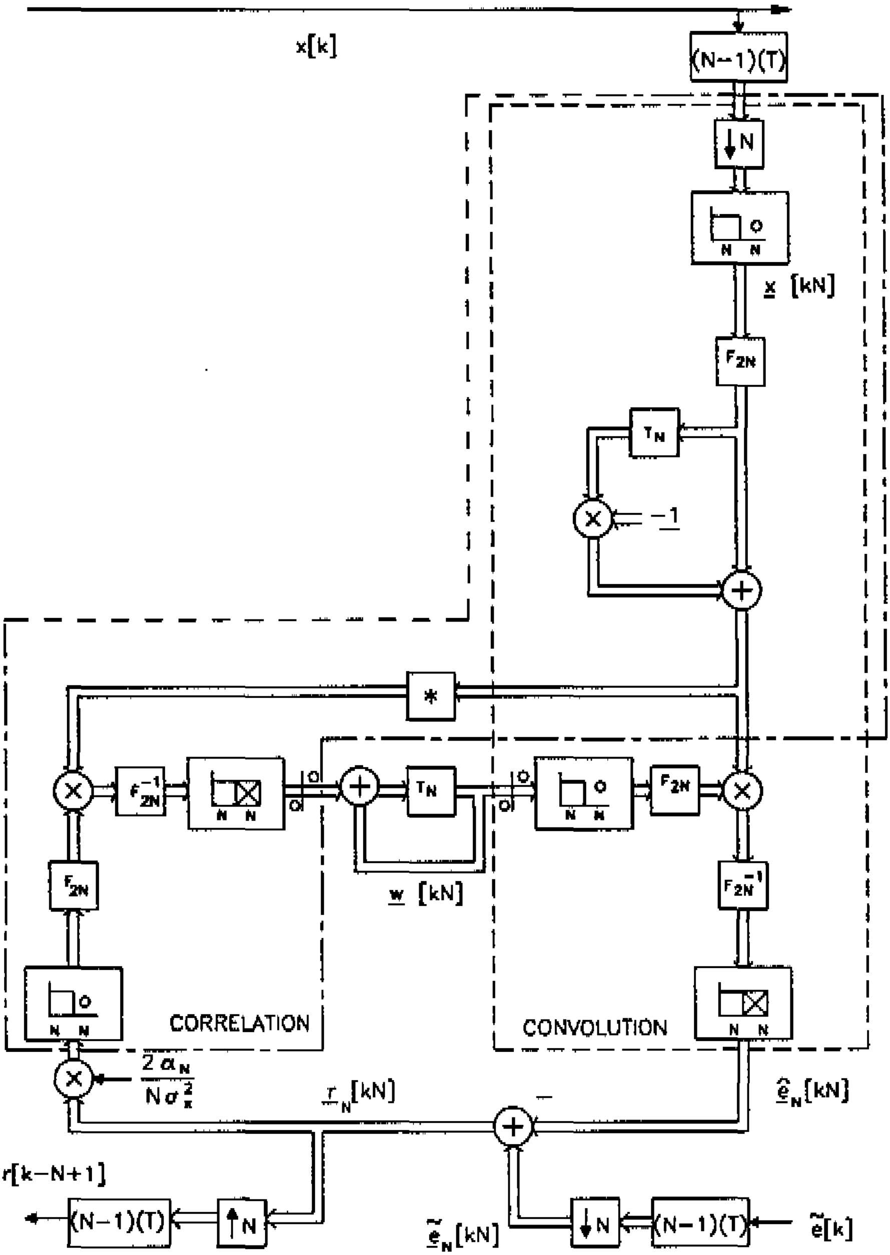

Figure 3.5: *Efficient overlap-add implementation of BNLMS with 5 FFTs*

Chapter 4

Frequency Domain Adaptive Filters

In Chapter 2 it is shown that convergence properties of gradient–based adaptive methods in general, and LMS in particular, are dependent on the input signal statistics. Since many physical processes of interest, such as speech and special codes, are highly correlated, this has served as motivation for deriving other methods of adaptive filtering which are not so sensitive to the input signal statistics.

In this chapter adaptive filters are discussed of which the weights are adjusted independently. This is achieved by using an orthogonal transform that is performed with a fixed preprocessing consisting of the Discrete Fourier Transform (DFT) or the fast implementation of this: the Fast Fourier Transformation FFT. Since the autocorrelation function and the psdf form a Fourier transform pair, decorrelation can be performed in frequency domain. However, as a result of the cyclic nature of the DFT, perfect decorrelation will never be reached.

In Section 4.1 it is shown how decorrelation can be performed in frequency domain. This is done by choosing the adaptation constant for each frequency component l equal to the overall adaptation constant divided by an estimate of the input power of this frequency component. Using this approach leads to the Frequency Domain Adaptive Filter (FDAF). In Section 4.2 it is shown under which circumstances and restrictions the normalization in frequency domain shows a close resemblance with the decorrelation of the input signal. Another motivation for switching to frequency domain is the efficient implementation of a convolution using FFTs when block processing is applied, as discussed in the previous chapter. This leads to the Block Frequency Domain Adaptive Filter (BFDAF) that is represented in section 4.3. This BFDAF approach tackles two problems simultanously:

(a) Under the restrictions, as given in Section 2, convergence properties are made (almost) independent of input signal statistics by spectrum normalization.

(b) Complexity is reduced, as proposed in the previous chapter, by implementing the convolution and correlation in frequency domain, with FFTs as transformation between time– and frequency domain.

Roughly there are two variants of the BFDAF known in literature. The first one, containing five FFTs, is explained in Section 4.3. This structure was introduced in [9] as the constrained BFDAF, since it requires a constraint in adjusting the frequency domain weights based on overlap–save sectioning. In [34] an unconstrained structure is introduced by removing the window. This structure only needs three FFTs. The main goal of the following sections is to get an insight into differences of convergence properties of these structures. For this reason Section 4.4 gives an analysis of the BFDAF by using a generalized window function [51,50]. As a result of this analysis it is shown that, under certain circumstances, an efficient window function can be used as introduced in [49,59]. Results of this analysis are supported by experiments, given in Section 4.5. This chapter is closed with a short discussion in Section 4.6.

4.1 Frequency Domain Adaptive Filter

Applying the concept of the Frequency Domain Adaptive Filter (FDAF) [36] results, for the generic adaptive filter of Fig. 1.7, in a changing of each of the components in amplitude and phase by the unknown optimum Wiener filter w_{opt}. These changes of each component can be estimated by an adaptive filter, that is implemented as a filter bank parallel to w_{opt}. By using the Fourier matrix $\mathbf{F}$ this FDAF concept can be derived by rewriting the output signal $\hat{e}[k]$ of the transversal filter of Fig. 1.7 as

$$\begin{aligned}\hat{e}[k] &= \sum_{i=0}^{N-1} x[k-i]w_i[k] = \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{w}}[k] \\ &= \underline{\mathbf{x}}^t[k]\mathbf{F} \cdot \mathbf{F}^{-1}\underline{\mathbf{w}}[k] = \frac{1}{N}\left(\mathbf{F}\underline{\mathbf{x}}[k]\right)^t \cdot \left(\mathbf{F}^*\underline{\mathbf{w}}[k]\right) \\ &= \frac{1}{N}\underline{\mathbf{X}}^t[k] \cdot \underline{\mathbf{W}}^*[k] = \frac{1}{N}\sum_{l=0}^{N-1} X_l[k]W_l^*[k]. \qquad (4.1)\end{aligned}$$

Thus the estimate $\hat{e}[k]$ of the signal $e[k]$ can be rewritten as the above summation, with

$$\begin{aligned}\underline{\mathbf{X}}[k] &= (X_0[k], \cdots, X_{N-1}[k])^t = \mathbf{F} \cdot \underline{\mathbf{x}}[k] \\ \underline{\mathbf{W}}^*[k] &= (W_0[k], \cdots, W_{N-1}[k])^t = \mathbf{F}^* \cdot \underline{\mathbf{w}}[k]. \qquad (4.2)\end{aligned}$$

Multiplying both sides of the LMS updating algorithm with the matrix $\mathbf{F}^*$ gives:

$$\mathbf{F}^*\underline{\mathbf{w}}[k+1] = \mathbf{F}^*\underline{\mathbf{w}}[k] + 2\alpha\mathbf{F}^*\underline{\mathbf{x}}[k]r[k] \qquad (4.3)$$

resulting in the following LMS algorithm that is implemented with one DFT:

$$\underline{\mathbf{W}}^*[k+1] = \underline{\mathbf{W}}^*[k] + 2\alpha\underline{\mathbf{X}}^*[k]r[k]. \qquad (4.4)$$

This principle is used in the FDAF that is depicted in Fig. 4.1. Note that the calculation of the output signal $\hat{e}[k]$ of the adaptive filter needs a factor $1/N$. This is accomplished in the figure by multiplying the residual signal $r[k]$ by the scaled adaptation constant $2\alpha/N$.

With each new input sample the data slides one step down a delay line of length N, acting as a rectangular window, and a new FFT is computed. Each of the FFT outputs $X_l[k]$, with $l = 0, 1, \cdots, N-1$, is associated with a specific frequency band. The FFT used in this manner can be considered as a means of implementing a bank of band–pass filters uniformly spaced in frequency between zero and half the

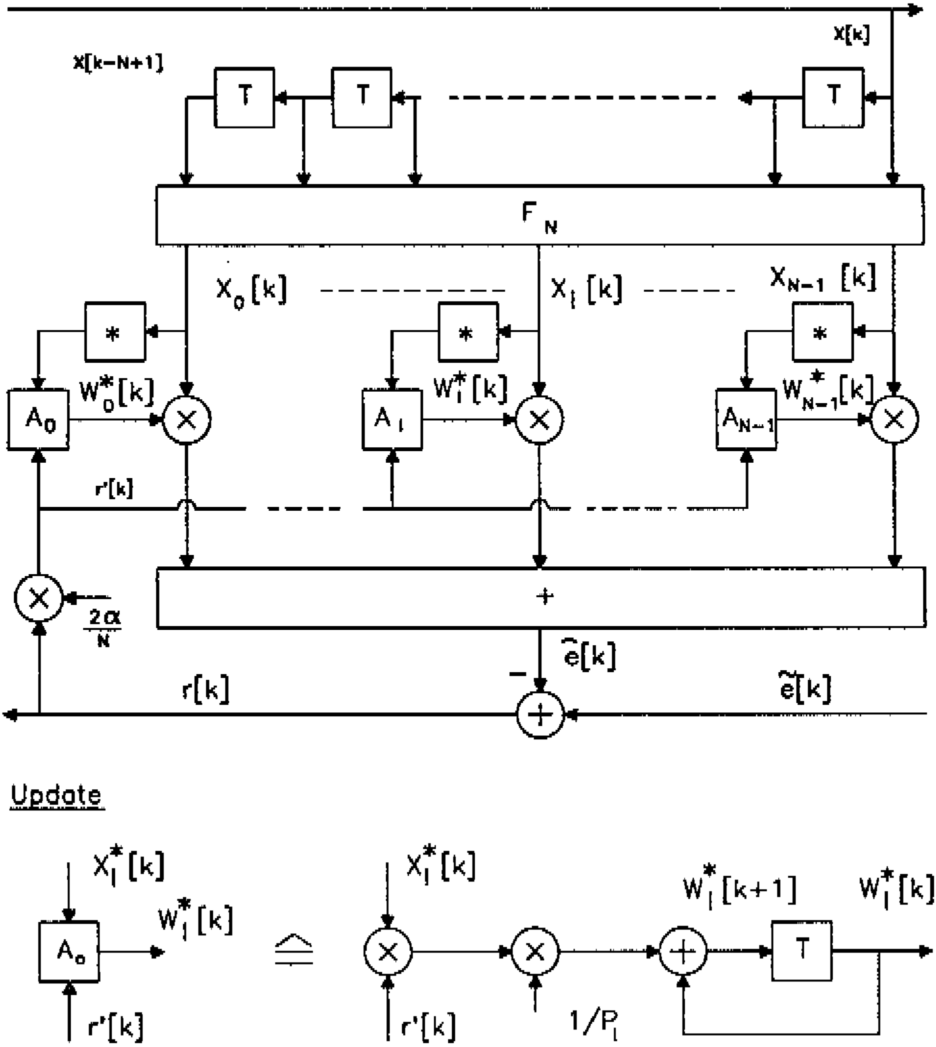

Figure 4.1: *Adaptive filter using FDAF updating algorithm*

sampling frequency. Note that because of the rectangular weighting of the input signal, each bandpass filter has a $\sin(x)/x$ character. The FFT outputs in Fig. 4.1 are complex discrete functions of the sampling index k. They are approximately uncorrelated with each other, being in different frequency bands. The frequency components are not perfectly uncorrelated because the FFT band–pass filters overlap somewhat, causing leakage of signal components from one band to another. For an exact description of this interbin decorrelation we refer to [32] (part of B. Picinbono). These complex output signals of the FFT are weighted in Fig. 4.1 with complex adaptive weights $W_i^*[k]$ to produce $\hat{e}[k]$. In fact these weights are such that they "perform" an inverse Fourier transform since $\underline{\mathbf{W}}^*[k] = \mathbf{F}^*\underline{\mathbf{w}}[k] = N\mathbf{F}^{-1}\underline{\mathbf{w}}[k]$. The N weights are updated in accordance to the transformed LMS algorithm as described above. When dealing with real signals and impulse responses the adaptive weight vector $\underline{\mathbf{W}}^*[k]$ and the input signal vector $\underline{\mathbf{X}}[k]$ have symmetry properties, which can be used to lower the computation load of the algorithm.

Under certain circumstances, described in the next section, the convergence of the above described transformed LMS update algorithm can be made independent of the input signal statistics by normalizing each of the FFT outputs to equal power levels. This result follows from the analysis of the average value of the transformed LMS algorithm (4.4). For this the residual signal $r[k]$ is first rewritten as:

$$r[k] = \frac{1}{N}\underline{\mathbf{X}}^t[k] \cdot \underline{\mathbf{D}}^*[k] + s[k] \tag{4.5}$$

with

$$\underline{\mathbf{D}}^*[k] = \mathbf{F}^* \cdot \underline{\mathbf{d}}[k] = \mathbf{F}^*\left(\underline{\mathbf{w}}_{\mathrm{opt}} - \underline{\mathbf{w}}[k]\right) = \underline{\mathbf{W}}^*_{\mathrm{opt}} - \underline{\mathbf{W}}^*[k]. \tag{4.6}$$

Assuming a small adaptation constant α it follows that, as in Chapter 2, the input signal and the adaptive weights may be separated under $E\{\cdot\}$. Together with $E\{s[k]\} = 0$ it follows that averaging equation (4.4) reduces to:

$$E\{\underline{\mathbf{D}}^*[k+1]\} \approx \left(\mathbf{I} - 2\alpha\frac{1}{N}E\{\underline{\mathbf{X}}^*[k]\underline{\mathbf{X}}^t[k]\}\right) \cdot E\{\underline{\mathbf{D}}^*[k]\}. \tag{4.7}$$

Thus convergence of each separate component $E\{D_l^*[k]\}$ is dependent on the input signal statistics, that are given by the matrix $\frac{1}{N}E\{\underline{\mathbf{X}}^*[k]\underline{\mathbf{X}}^t[k]\}$. Under the assumption that different frequency components are uncorrelated this matrix reduces to the diagonal matrix $\mathbf{P}$ as follows:

$$\begin{aligned} \frac{1}{N}E\{\underline{\mathbf{X}}^*[k]\underline{\mathbf{X}}^t[k]\} &\approx diag\{\frac{1}{N}E\{|X_0[k]|^2\},\cdots,\frac{1}{N}E\{|X_{N-1}[k]|^2\}\} \\ &= diag\{P_0,\cdots,P_{N-1}\} = \mathbf{P}. \end{aligned} \tag{4.8}$$

Note that here the same approximation is applied as mentioned in the Chapter 1, where the symmetric circulant autocorrelation matrix $\mathbf{C}$ is constructed from the Toeplitz autocorrelation matrix $\mathbf{R}$. Since the circulant matrix $\mathbf{C}$ can be diagonalized by the Fourier matrix $\mathbf{F}$ as follows:

$$\begin{aligned} \frac{1}{N}E\{\underline{\mathbf{X}}^*[k]\underline{\mathbf{X}}^t[k]\} &= \frac{1}{N}\mathbf{F}^*E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}\mathbf{F} = \mathbf{F}^{-1}\mathbf{R}\mathbf{F} \\ &\approx \mathbf{F}^{-1}\mathbf{C}\mathbf{F} = \mathbf{P}. \end{aligned} \tag{4.9}$$

Thus normalizing each of the FFT outputs with P_l for $l = 0, 1, \cdots, N-1$ to equal power levels makes the convergence properties independent of the input signal statistics. This results in the Frequency Domain Adaptive Filter (FDAF) that is given by the following updating equation:

$$\underline{\mathbf{W}}^*[k+1] = \underline{\mathbf{W}}^*[k] + 2\alpha\mathbf{P}^{-1}\underline{\mathbf{X}}^*[k]r[k] \tag{4.10}$$

and the update scheme is depicted in Fig. 4.1.

Finally this section is concluded with some general comments:

- Counting as a rough measure for complexity the number of real multiplications and divisions, results for the FDAF algorithm in the following numbers:

$$\begin{aligned} MUL_{FDAF} &\approx \frac{1}{2}\left(2N^2\log(N) + 4N\right) \\ DIV_{FDAF} &\approx \frac{1}{2}\cdot 4N. \end{aligned} \tag{4.11}$$

- From the update equation (4.10) it follows that in average the final value is a transformed version of the optimal Wiener solution, namely

$$\lim_{k\rightarrow\infty}\left(E\{\underline{\mathbf{W}}^*[k]\}\right) = \frac{1}{N}\mathbf{F}^*\underline{\mathbf{w}}_{\mathrm{opt}}. \tag{4.12}$$

- The power levels for $l = 0, \cdots, N-1$ can be estimated by exponential time averaging as follows:

$$\hat{P}_l[k+1] = \beta \hat{P}_l[k] + (1-\beta)\frac{|X_l[k]|^2}{N} \quad \text{with} \quad 0 < \beta < 1. \quad (4.13)$$

  In [51,50] a detailed analysis is given of the influence of this estimation scheme on the convergence properties of the frequency domain adaptive filter.

- Since the update scheme needs the inverse of P_l, it may be usefull to search for estimators for this inverse function. This is equivalent to the approach as used in the Recursive Least Square (RLS) method that will be shortly discussed in Chapter 6.

4.2 Decorrelation conditions for the FDA algorithm

As mentioned in the introduction of this chapter, perfect decorrelation can never be reached by the power normalization of each separate frequency component because of the cyclic nature of the DFT. This section describes two conditions under which the power normalization acts as a reasonable approximation for the desired decorrelation of the input signal of the adaptive filter.

The first condition is a direct consequence of the Toeplitz–circulant matrix approximation as discussed in Chapter 1. A restriction for this approximation was that the autocorrelation function $\rho[\tau]$ has negligible values for $\tau_{\max}$ greater than half the length of the Fourier transformation. For the FDAF this results in:

$$|\tau_{\max}| < N/2. \quad (4.14)$$

A direct consequence of this condition is that the closer the poles of a signal model lie to the unit circle, the larger the DFT length has to be chosen in order to enable decorrelation such an input signal.

The second condition is mainly due to the fact that the inverse value of the power spectrum $(1/P_l)$ is needed in the update equation. This causes problems for input signals that can be modelled with zeros very close to the unit circle. In order to be able to give a quantative measure of the second condition, the impact of the DFT length N on the power normalization is analysed. In the FDAF algorithm the psdf

$$P(e^{j\theta}) = \sum_{\tau=-\infty}^{\infty} \rho[\tau]e^{-j\tau\theta} = \rho[0] + 2\sum_{\tau=1}^{\infty} \rho[\tau]\cos(\tau\theta) \tag{4.15}$$

is estimated, and normalized, at frequency $\theta_N \cdot l = (2\pi/N) \cdot l$ with the function:

$$P_l = \frac{1}{N}E\{|X_l[k]|^2\} = \rho[0] + 2\sum_{\tau=1}^{N-1} \frac{N-\tau}{N}\rho[\tau]\cos(\theta_N l\tau). \tag{4.16}$$

Transforming this equation to time domain results in an expression for the cyclic estimate $\hat{\rho}$ of the autocorrelation function ρ. Namely for $\tau = 0, 1, \cdots, N-1$:

$$\hat{\rho}[\tau] = \frac{N-\tau}{N}\rho[\tau] + \frac{\tau}{N}\rho[N-\tau] \tag{4.17}$$

from which two error sources in the estimation of $\hat{\rho}$ are evident:

(a) Basing the estimate on N samples introduces a bias that results in a windowing effect $\Rightarrow \frac{N-\tau}{N}$.

(b) Sampling the power spectral density function $P(e^{j\theta})$ results in a periodic repetition of the function ρ with period $N \Rightarrow \frac{\tau}{N}\rho[N-\tau]$.

Ideally the power normalization of the FDAF algorithm implies that all power levels of the input signal are reduced to 1, and hence the convergence properties for all adaptive weights become equal. Now the Power Decrease Ratio (PDR) [57] is defined as:

$$PDR_l = \frac{P(e^{j\theta})|_{\theta=\theta_N l}}{P_l}. \tag{4.18}$$

This function evaluates for frequency component l how far the original power $P(e^{j\theta})|_{\theta=\theta_N l}$ is decreased or increased to 1 by the power normalization as applied in the FDAF algorithm with $1/P_l$.

For the ma(1) and ar(1) models, as described in chapter 1, this PDR function is evaluated as:

$$\begin{aligned} PDR_{MA,l} &= \frac{1+a^2+2a\cos(\theta_N l)}{1+a^2+2a\frac{N-1}{N}\cos(\theta_N l)} \\ PDR_{AR,l} &= \frac{\frac{1-a^2}{1+a^2-2a\cos(\theta_N)}}{1+2\sum_{\tau=1}^{N-1}\frac{N-\tau}{N}a^{|\tau|}\cos(\theta_N l\tau)}. \end{aligned} \tag{4.19}$$

In Fig. 4.2 these functions are plotted with $a = 0.8182$ ($\Rightarrow ER = 100$) for different N. From these figures it follows: the larger the DFT length, the better the power spectral normalization. It also follows from this figure that for small values of N the PDR is reasonable flat for a signal model with a pole (ar(1)). For this model $PDR \approx 0.9$ for $N = 32$. To increase this value further from 0.9 to 1 a DFT of infinite length is needed. Furthermore it follows that the PDR, with $N = 32$, for signal models with a spectral zero (ma(1)) is equalized to 1 for $N = 32$ over a large spectral range, except the spectral range near the spectral zero. In order to increase this spectral zero further from 0.4 to 1, a DFT of infinite length is needed. In many practical cases however this DFT length can be restricted for the folowing two reasons: The approximation of the decorrelation by the power normalization needs not to be perfect, while moreover this approximation needs not to be that good for the whole frequency range. It follows from Fig. 4.2 that the ma(1) signal can be decorrelated reasonably well when the unknown optimal Wiener filter has a low-pass character. The ar(1) signal can be decorrelated satisfactory for large enough N as long as $|\tau_{\max}| < N/2$, "independent" of the frequency response of the unknown optimal Wiener solution.

In conclusion it follows that in general frequency domain adaptive filters can decorrelate an input signal acceptably well by normalizing each separate frequency component by its power spectrum when the following decorrelation conditions are satisfied:
For ar-signal models (spectral poles):
The autocorrelation function $\rho[\tau]$ of the input signal must be restricted by:

$$|\tau_{\max}| < N/2 \tag{4.20}$$

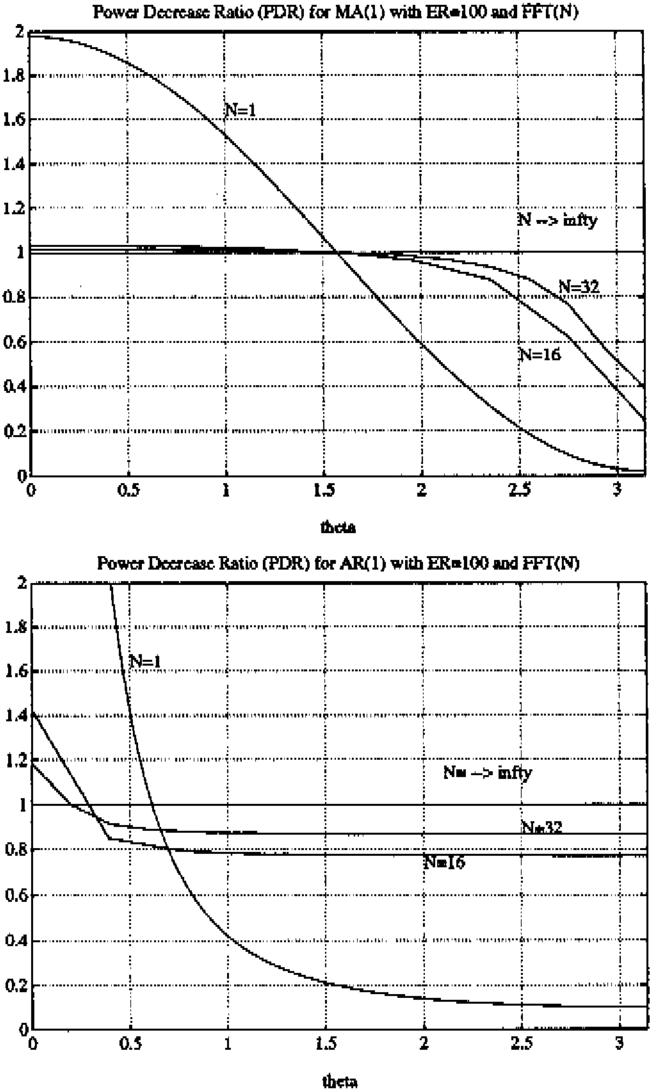

Figure 4.2: *PDR for ma(1) (top) and ar(1) for different DFT length N*

For ma–signal models (spectral zeros):
The length of the DFT must be such that the power decrease ratio, that is defined as

$$PDR_l = \frac{P(e^{j\theta})|_{\theta=\theta_{Nl}}}{P_l} \tag{4.21}$$

is reasonably equalized to 1 in the frequency range that is of importance for the unknown optimum Wiener system.
These results are verified by experiments described in Section 4.5.

4.3 Block Frequency Domain Adaptive Filter

Implementing the BNLMS update equation in frequency domain, as described in Chapter 3, and performing the power normalization as discussed in the previous sections leads to the Block Frequency Domain Adaptive Filter (BFDAF). This structure is derived in the present section. The first step is to describe the overlap–save implementation of the BLMS update algorithm, as depicted in Fig. 3.2 (without power normalization), in mathematical forms. The second step is to determine the update not in time, but in frequency domain. The last step is to make the convergence properties independent of input signal statistics by power normalization, as described in the previous sections.

The update equation of the BLMS algorithm is given by:

$$\underline{\mathbf{w}}[(k+1)L] = \underline{\mathbf{w}}[kL] + \frac{2\alpha}{L}\chi[kL]\underline{\mathbf{r}}_L[kL] \tag{4.22}$$

with

$$\begin{aligned}
\underline{\mathbf{w}}[kL] &= (w_{N-1}[kL], \cdots, w_1[kL], w_0[kL])^t \qquad (4.23)\\
\underline{\mathbf{r}}_L[kL] &= (r[kL-L+1], \cdots, r[kL-1], r[kL])^t\\
\chi[kL] &= (\underline{\mathbf{x}}[kL-L+1], \cdots, \underline{\mathbf{x}}[kL-1], \underline{\mathbf{x}}[kL])\\
\underline{\mathbf{x}}[kL-i] &= (x[kL-i-N+1], \cdots, x[kL-i-1], x[kL-i])^t
\end{aligned}$$

In Fig. 3.2 this algorithm is calculated by using a cyclic correlation that is performed in frequency domain. With $B = N + L - 1$ first the

$B \times B$ matrix $\chi^c[kL]$ is defined as the I–circulant expansion of the $N \times L$ mirrored matrix $\mathbf{J}_N \cdot \chi[kL]$. This is done by putting $\mathbf{J}_N \cdot \chi[kL]$ in the upper right corner of $\chi^c[kL]$ and filling in the missing elements in such a way that $\chi^c[kL]$ becomes I–circulant. By doing this, the following relationship is obvious:

$$\begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix} \cdot \chi^c[kL] \cdot \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_L \end{pmatrix} = \mathbf{J}_N \cdot \chi[kL]. \tag{4.24}$$

Noticing that $\mathbf{J}_N \cdot \mathbf{J}_N = \mathbf{I}_N$ the mirror matrix $\mathbf{J}_N$ can be placed at the left–hand side of this formula. With this the above update equation can be written as:

$$\underline{\mathbf{w}}[(k+1)L] = \underline{\mathbf{w}}[kL] + \frac{2\alpha}{L} \mathbf{J}_N \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix} \cdot \chi^c[kL] \cdot \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_L \end{pmatrix} \underline{\mathbf{r}}_L[kL]. \tag{4.25}$$

The I–circulant matrix can be diagonalized as follows:

$$\mathbf{F}\chi^c[kL]\mathbf{F}^{-1} = \mathbf{X}^*[kL] = diag\{\underline{\mathbf{X}}^*[kL]\} \tag{4.26}$$

with

$$\underline{\mathbf{X}}^*[kL] = (X_0^*[kL], X_1^*[kL], \cdots, X_{B-1}^*[kL])^t = \mathbf{F}^* \cdot \underline{\mathbf{x}}_B[kL]. \tag{4.27}$$

Thus by multipying both sides of the I–circulant matrix $\chi^c[kL]$ in equation (4.24) with the $B \times B$ identity matrix $\mathbf{I}_B = \mathbf{F}^{-1} \cdot \mathbf{F}$, the update equation (4.22) can be written as follows:

$$\underline{\mathbf{w}}[(k+1)L] = \underline{\mathbf{w}}[kL] + \frac{2\alpha}{L} \mathbf{J}_N \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix} \mathbf{F}^{-1}\mathbf{X}^*[kL]\underline{\mathbf{R}}'[kL] \tag{4.28}$$

with

$$\underline{\mathbf{R}}'[kL] = \mathbf{F} \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_L \end{pmatrix} \underline{\mathbf{r}}_L[kL]. \tag{4.29}$$

Note that with the assumption

$$\frac{1}{B} E\{X_l^*[kL] X_m[kL]\} = \begin{cases} P_{X_l} & \text{for } l = m \\ 0 & \text{elsewhere} \end{cases} \tag{4.30}$$

it follows that the following relationship holds:

$$\frac{1}{B} E\{\underline{\mathbf{X}}^*[kL]\underline{\mathbf{X}}^t[kL]\} = \frac{1}{B} E\{\mathbf{X}^*[kL]\mathbf{X}[kL]\} = \mathbf{P}. \tag{4.31}$$

Hence it is allowed to use a diagonal matrix notation $\mathbf{X}[kL] = diag\{\underline{\mathbf{X}}[kL$ in formulas, and a vector notation $\underline{\mathbf{X}}[kL]$ in figures.

For the second step of this section update equation (4.28) has to be described as if it was implemented after the second mirroring and windowing of Fig. 3.2. This can be done by multiplying both sides of the above update equation with $\begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \mathbf{J}_N$. Noticing furthermore that $\mathbf{J}_N \cdot \mathbf{J}_N = \mathbf{I}_N$ and by defining the window

$$\mathbf{g} = \begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix} \tag{4.32}$$

this results in

$$\begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \mathbf{J}_N \underline{\mathbf{w}}[(k+1)L] = \begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \mathbf{J}_N \underline{\mathbf{w}}[kL] + \frac{2\alpha}{L} \mathbf{g} \mathbf{F}^{-1} \mathbf{X}^*[kL] \underline{\mathbf{R}}'[kL]. \tag{4.33}$$

Now this update equation will be implemented in frequency domain. This can be done by multiplying both sides of the last update equation with the Fourier matrix $\mathbf{F}$, resulting in the following equation:

$$\underline{\mathbf{W}}[(k+1)L] = \underline{\mathbf{W}}[kL] + \frac{2\alpha}{L} \mathbf{G} \mathbf{X}^*[kL] \underline{\mathbf{R}}'[kL] \tag{4.34}$$

with

$$\underline{\mathbf{W}}[kL] = \mathbf{F} \begin{pmatrix} \mathbf{I}_N \\ 0 \end{pmatrix} \mathbf{J}_N \underline{\mathbf{w}}[kL] \quad \text{and} \quad \mathbf{G} = \mathbf{F} \mathbf{g} \mathbf{F}^{-1}. \tag{4.35}$$

This equation describes the BLMS update algorithm, implemented efficiently in frequency domain. The update part is depicted in Fig. 4.3. In order to make the convergence properties independent of the input signal statistics, the last step of this section is to use the same power normalization as used in the previous sections. The result of this is the Block Frequency Domain Adaptive Filter (BFDAF), from which the update equation is given by

$$\underline{\mathbf{W}}[(k+1)L] = \underline{\mathbf{W}}[kL] + \frac{2\alpha}{L} \mathbf{G} \mathbf{P}^{-1} \mathbf{X}^*[kL] \underline{\mathbf{R}}'[kL] \tag{4.36}$$

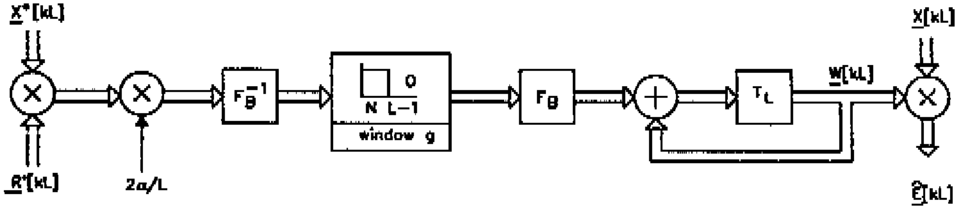

Figure 4.3: *Update part of efficient BLMS algorithm implemented in frequency domain*

An implementation of this algorithm is depicted in Fig. 4.4. Note that in this figure the power normalization is performed with the vector $\underline{\mathbf{P}}^{-1}$ that is defined as:

$$\underline{\mathbf{P}}^{-1} = (1/P_0, \cdots, 1/P_{N-1})^t. \tag{4.37}$$

As mentioned in the introduction of this chapter, there are roughly two variants of the BFDAF known in literature. The first one is explained in this section with 5 FFTs. This structure was introduced in [9] as the constrained BFDAF, since it requires a constraint (window **g**) in adjusting the frequency domain weights based on overlap–save sectioning. In [34] an unconstrained structure is introduced by removing the window **g**. This structure is less complex since it requires only 3 FFTs. As a measure for complexity the number of real multiplications and divisions is used needed to calculate one output sample:

$$\begin{aligned} MUL_{BFDAF} &= \frac{1}{2}\left(\frac{(3+2win)\cdot 2B^2\log(B) + 2\cdot 4B}{L}\right) \\ DIV_{BFDAF} &= \frac{1}{2}\cdot\frac{4B}{L} \end{aligned} \tag{4.38}$$

with $B = N + L - 1$ and the processing delay $L \geq 1$. Above that the parameter *win* is used to denote the difference in complexity between the unconstrained (*win* = 0) and constrained (*win* = 1) approach.
Notes:

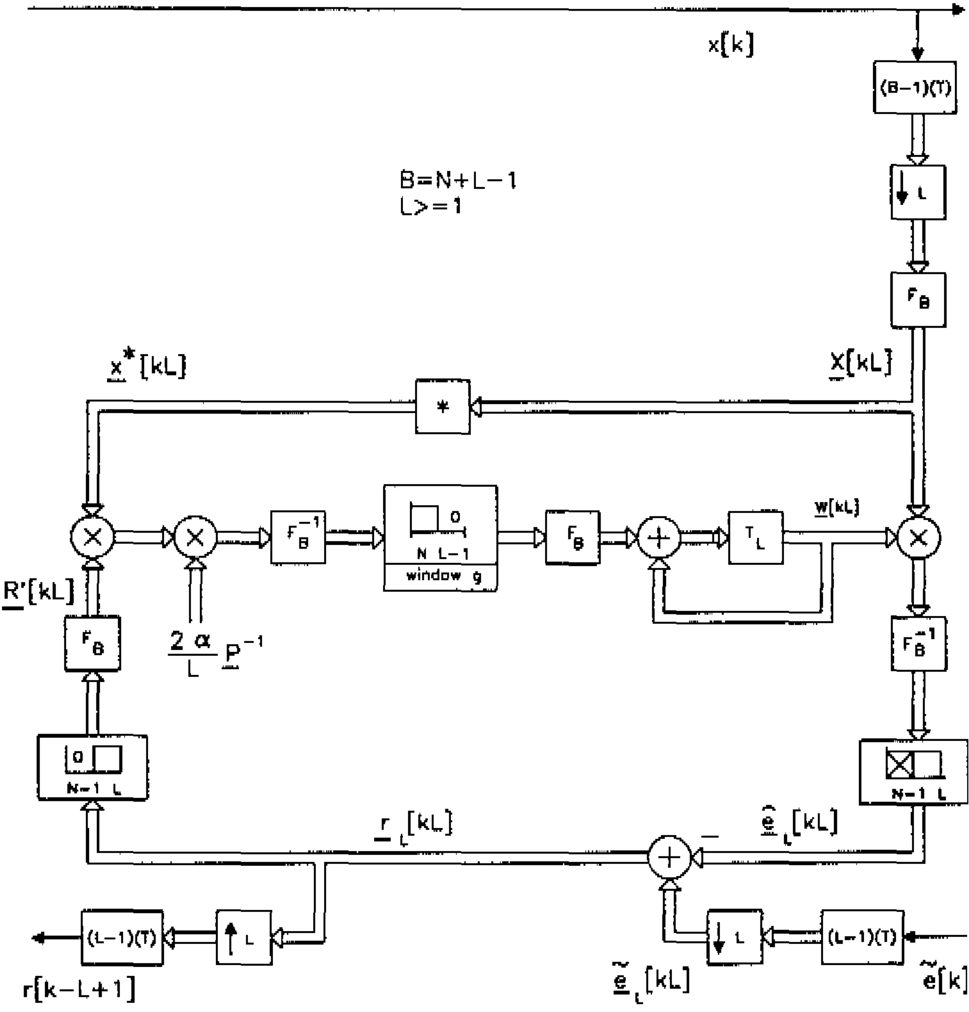

Figure 4.4: *Adaptive filter using BFDAF update algorithm*

- It is known from literature that an FFT can be implemented most efficiently when the length B is a power of two [37]. In many practical cases this is done by choosing N as a power of two and $L = N + 1$. This results in $B = 2N$. Another possibility is to choose N in stead of $N - 1$ samples of the previous segment, and choosing $L = N$. The number of output samples is now $L + 1 = N + 1$, from which the first sample is already calculated in the previous iteration.
- The FDAF of section 1 is, of course, a special case of the BFDAF with $L = 1$. This can be shown as follows:
  For $L = 1$ the BFDAF update equation (4.36) reduces to:

  $$\mathbf{F}\mathbf{J}_N\underline{\mathbf{w}}[k+1] = \mathbf{F}\mathbf{J}_N\underline{\mathbf{w}}[k] + 2\alpha\mathbf{P}^{-1}diag\{\underline{\mathbf{X}}_N^*[k]\}\underline{\mathbf{e}}r[k] \qquad (4.39)$$

  with $\underline{\mathbf{e}} = (1, \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}(N-1)}, \cdots, \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}(N-1)(N-1)})^t$. This vector is the frequency domain equivalent of a time domain cyclic shift $\mathbf{J}_N^{-1}$. By multiplying both sides of the above update equation, in time domain, with the operator $\mathbf{J}_N^1$ this cyclic shift can be made undone. Furthermore with $\mathbf{F}\mathbf{J}_N\underline{\mathbf{w}}[k] = \underline{\mathbf{W}}[k]$, and thus with $\mathbf{F}\mathbf{J}_N^1\underline{\mathbf{w}}[k] = \underline{\mathbf{W}}^*[k]$, this update equation reduces to the FDAF update equation (4.10).

4.4 Analysis of the BFDAF algorithm

In order to get an insight into differences of convergence properties of the constrained and unconstrained structure this section gives an analysis of the BFDAF as proposed in the previous section. In this analysis it is assumed that the power matrix $\mathbf{P}$ is constant. In references [51,50] a more detailed analysis is given when the exponential power spectral estimation scheme of equation (4.13) is used. Furthermore the analysis of this section uses a generalized window function $\mathbf{g} = diag\{g_0, \cdots, g_{B-1}\}$. For the constrained structure this window is defined as $\mathrm{g} = \mathrm{g_{con}}$ with

$$\mathrm{g_{con}} = \begin{pmatrix} \mathbf{I}_N \\ 0 \end{pmatrix} \begin{pmatrix} \mathbf{I}_N & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{I}_N & 0 \\ 0 & 0 \end{pmatrix} \qquad (4.40)$$

while for the unconstrained structure this window is a through connection and is defined as $\mathbf{g} = \mathbf{g}_{\mathrm{unc}} = \mathbf{I}_B$. Note that the definition of the transformed adaptive weight vector as

$$\underline{\mathbf{W}}[kL] = \mathbf{F}\begin{pmatrix} \mathbf{I}_N \\ 0 \end{pmatrix} \mathbf{J}_N \underline{\mathbf{w}}[kL] \tag{4.41}$$

only holds for the constrained window function $\mathbf{g}_{\mathrm{con}}$. A more general transformed adaptive weight vector is defined as:

$$\underline{\mathbf{W}}[kL] = \mathbf{F}\mathbf{J}_B \underline{\mathbf{w}}'[kL] = \mathbf{F}(w'_0[kL], \cdots, w'_{B-1}[kL])^t \tag{4.42}$$

will be used for the analysis. Furthermore it is assumed that both decorrelation conditions, as described in Section 4.2, are satisfied

For analytical purposes the update scheme of Fig. 4.4 is modified. This is done by placing the inverse FFT and window, needed to produce $\underline{\hat{\mathbf{e}}}_L[kL]$, after the addition point. This part is depicted in Fig. 4.5. By

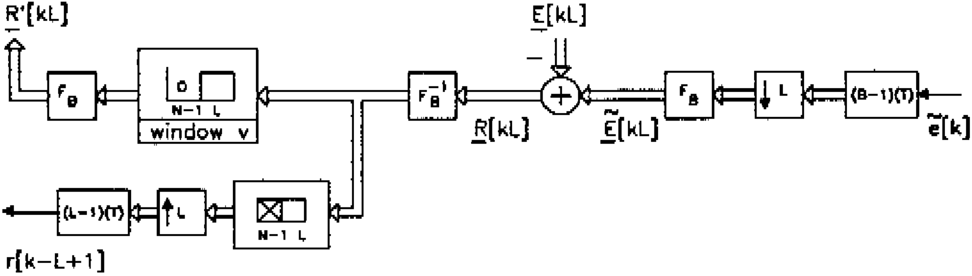

Figure 4.5: *Modification of BFDAF update scheme for analytical purposes*

doing so the residual transformed vector $\underline{\mathbf{R}}[kL]$ can be written as:

$$\begin{aligned} \underline{\mathbf{R}}[kL] &= \mathbf{X}[kL]\underline{\mathbf{D}}[kL] + \underline{\mathbf{S}}[kL] \\ \underline{\mathbf{D}}[kL] &= \underline{\mathbf{W}}_{\mathrm{opt}} - \underline{\mathbf{W}}[kL] \\ \underline{\mathbf{W}}_{\mathrm{opt}} &= \mathbf{F}\begin{pmatrix} \mathbf{I}_N \\ 0 \end{pmatrix} \mathbf{J}_N \underline{\mathbf{w}}_{\mathrm{opt}} \\ \underline{\mathbf{S}}[kL] &= \mathbf{F}\mathbf{s}_B[kL] \end{aligned} \tag{4.43}$$

With these notations it is obvious to write

$$\underline{\mathbf{R}}'[kL] = \mathbf{F} \begin{pmatrix} 0 \\ \mathbf{I}_L \end{pmatrix} \cdot \begin{pmatrix} 0 & \mathbf{I}_L \end{pmatrix} \mathbf{F}^{-1}\underline{\mathbf{R}}[kL]. \tag{4.44}$$

With the window function

$$\mathrm{v} = \begin{pmatrix} 0 \\ \mathbf{I}_L \end{pmatrix} \cdot \begin{pmatrix} 0 & \mathbf{I}_L \end{pmatrix} \quad \text{and} \quad \mathbf{V} = \mathbf{F}\mathrm{v}\mathbf{F}^{-1} \tag{4.45}$$

this transformed and windowed residual vector can be rewritte as:

$$\underline{\mathbf{R}}'[kL] = \mathbf{V}\underline{\mathbf{R}}[kL]. \tag{4.46}$$

Now update equation (4.36) can be rewritten as:

$$\underline{\mathbf{W}}[(k+1)L] = \underline{\mathbf{W}}[kL] + \frac{2\alpha}{L}\mathbf{G}\mathbf{P}^{-1}\mathbf{X}^*[kL]\mathbf{V}\underline{\mathbf{R}}[kL] \tag{4.47}$$

or equivalently, the update equation for the difference vector $\underline{\mathbf{D}}[kL]$ is given by:

$$\begin{aligned} \underline{\mathbf{D}}[(k+1)L] &= \left(\mathbf{I} - \frac{2\alpha}{L}\mathbf{G}\mathbf{P}^{-1}\mathbf{X}^*[kL]\mathbf{V}\mathbf{X}[kL]\right) \cdot \underline{\mathbf{D}}[kL] \\ &\quad - \frac{2\alpha}{L}\mathbf{G}\mathbf{P}^{-1}\mathbf{X}^*[kL]\mathbf{V}\underline{\mathbf{S}}[kL]. \end{aligned} \tag{4.48}$$

As in chapter 2, the analysis is performed in two steps. First the average behaviour of the difference vector $\underline{\mathbf{D}}[kL]$ is studied, and after that the mean squared error of a block with L residual samples is analysed. Both steps use the assumptions that signal x is independent of signal s, and that the adaptation constant is small. With these assumptions the average behaviour of the difference vector $\underline{\mathbf{D}}[kL]$ is given by:

$$E\{\underline{\mathbf{D}}[(k+1)L]\} \approx \left(\mathbf{I} - \frac{2\alpha}{L}\mathbf{G}\mathbf{P}^{-1}E\{\mathbf{X}^*[kL]\mathbf{V}\mathbf{X}[kL]\}\right) \cdot E\{\underline{\mathbf{D}}[kL]\}. \tag{4.49}$$

Since the different frequency components are assumed to be mutually uncorrelated, the calculation of the $(s,t)^{\text{th}}$ element of the matrix $E\{\mathbf{X}^*[kL]\mathbf{V}\mathbf{X}[kL]\}$ is given by:

$$\begin{aligned} (E\{\mathbf{X}^*[kL]\mathbf{V}\mathbf{X}[kL]\})_{s,t} &= E\{X_s^*[kL]X_t[kL]\} \cdot (\mathbf{V})_{s,t} \\ &\approx \begin{cases} B \cdot P_s(\mathbf{V})_{s,s} & \text{if } s = t \\ 0 & \text{elsewhere.} \end{cases} \end{aligned} \tag{4.50}$$

The definition of the window function $\mathbf{v}$ is such that

$$(\mathbf{V})_{s,s} = \frac{1}{B}\sum_{i=0}^{B-1} v_i = \frac{L}{B} \tag{4.51}$$

and with this

$$E\{\mathbf{X}^*[kL]\mathbf{V}\mathbf{X}[kL]\} \approx L\mathbf{P}. \tag{4.52}$$

Thus the above difference equation reduces to:

$$E\{\underline{\mathbf{D}}[(k+1)L]\} \approx (\mathbf{I} - 2\alpha\mathbf{G})\, E\{\underline{\mathbf{D}}[kL]\} \tag{4.53}$$

or equivalently

$$E\{\underline{\mathbf{W}}[(k+1)L]\} \approx (\mathbf{I} - 2\alpha\mathbf{G})\, E\{\underline{\mathbf{W}}[kL]\} + 2\alpha\mathbf{G}\underline{\mathbf{W}}_{\text{opt}}. \tag{4.54}$$

By using the definition of $\underline{\mathbf{W}}_{\text{opt}}$, as given in equation (4.43), and with $\underline{\mathbf{w}}'[kL] = (w'_{B-1}[kL], \cdots, w'_1[kL], w'_0[kL])^t$ and $\underline{\mathbf{W}}[kL] = \mathbf{F}\mathbf{J}_B\underline{\mathbf{w}}'[kL]$ the above equation can be rewritten as:

$$\mathbf{J}_B E\{\underline{\mathbf{w}}'(k+1)L] \approx (\mathbf{I} - 2\alpha\mathbf{g})\, \mathbf{J}_B E\{\underline{\mathbf{w}}'[kL]\} + 2\alpha\mathbf{g}\begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix}\mathbf{J}_N\underline{\mathbf{w}}_{\text{opt}}. \tag{4.55}$$

From this it follows that, independent of the window function $\mathbf{g}$, the adaptive weight vector converges in average to the optimum Wiener solution as follows:

$$\lim_{k\to\infty} \mathbf{J}_B\underline{\mathbf{w}}'[kL] = \begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix}\mathbf{J}_N\underline{\mathbf{w}}_{\text{opt}} \tag{4.56}$$

under the condition that the adaptation constant α is chosen in the average convergence area:

$$0 < \alpha < \frac{1}{g_{\max}} \quad \text{with} \quad g_{\max} = \max\{g_0, \cdots, g_{M-1}\}. \tag{4.57}$$

As mentioned in Chapter 2 the second quantity of interest is $\tilde{J}[kL] = J_{\text{ex}}[kL]/\sigma_s^2$ with

$$J_{\text{ex}}[kL] = \frac{1}{L}E\{(\underline{\mathbf{e}}_L[kL] - \hat{\underline{\mathbf{e}}}_L[kL])^t(\underline{\mathbf{e}}_L[kL] - \hat{\underline{\mathbf{e}}}_L[kL])\} \tag{4.58}$$

from which the difference signal is defined as

$$\underline{\mathbf{e}}_L[kL] - \hat{\underline{\mathbf{e}}}_L[kL] = \begin{pmatrix} \mathbf{0} & \mathbf{I}_L \end{pmatrix} \mathbf{F}^{-1}\mathbf{X}[kL]\underline{\mathbf{D}}[kL]. \tag{4.59}$$

With this, $J_{\mathrm{ex}}[kL]$ can be rewritten as

$$\begin{aligned} J_{\mathrm{ex}}[kL] &= \frac{1}{L}E\{\underline{\mathbf{D}}^h[kL]\mathbf{X}^h[kL](\mathbf{F}^{-1})^h \begin{pmatrix} \mathbf{0} & \mathbf{I}_L \end{pmatrix}^h \cdot \\ &\quad \cdot \begin{pmatrix} \mathbf{0} & \mathbf{I}_L \end{pmatrix} \mathbf{F}^{-1}\mathbf{X}[kL]\underline{\mathbf{D}}[kL]\} \\ &\approx \frac{1}{BL}E\{\underline{\mathbf{D}}^h[kL]E\{\mathbf{X}^*[kL]\mathbf{V}\mathbf{X}[kL]\}\underline{\mathbf{D}}[kL]\} \\ &\approx \frac{1}{B}E\{\underline{\mathbf{D}}^h[kL]\mathbf{P}\underline{\mathbf{D}}[kL]\} = \frac{1}{B}trace\{\mathbf{P}\Delta[kL]\} \end{aligned} \tag{4.60}$$

with $\Delta[kL] = E\{\underline{\mathbf{D}}[kL]\underline{\mathbf{D}}^h[kL]\}$. Furthermore by defining the transformed difference vector as $\underline{\mathbf{D}}[kL] = \mathbf{F} \cdot \underline{\mathbf{d}}[kL]$ with the time–domain difference vector given by:

$$\underline{\mathbf{d}}[kL] = \left(\begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \mathbf{J}_N \underline{\mathbf{w}}_{\mathrm{opt}} - \mathbf{J}_B \underline{\mathbf{w}}'[kL] \right). \tag{4.61}$$

The quantity $J_{\mathrm{ex}}[kL]$ can also be expressed in time–domain as follows:

$$J_{\mathrm{ex}}[kL] = \frac{1}{B}E\{\underline{\mathbf{d}}^t[kL]\left(\mathbf{F}^h\mathbf{P}\mathbf{F}\right)\underline{\mathbf{d}}[kL]\} = trace\{\mathbf{C}\delta[kL]\} \tag{4.62}$$

with the I–circulant autocorrelation matrix $\mathbf{C}$ and the matrix $\delta[kL]$ defined as:

$$\mathbf{C} = \mathbf{F}^{-1}\mathbf{P}\mathbf{F} \quad \text{and} \quad \delta[kL] = E\{\underline{\mathbf{d}}[kL]\underline{\mathbf{d}}^t[kL]\}. \tag{4.63}$$

First an expression is given for $\Delta[kL]$ by using equation (4.48). For small adaptation constant this leads to:

$$\begin{aligned} \Delta[(k+1)L] &\approx \Delta[kL] - \frac{2\alpha}{L}\mathbf{G}\mathbf{P}^{-1}E\{\mathbf{X}^*[kL]\mathbf{V}\mathbf{X}[kL]\}\Delta[kL] \\ &\quad -\frac{2\alpha}{L}\Delta[kL]E\{\mathbf{X}^*[kL]\mathbf{V}^h\mathbf{X}[kL]\}\mathbf{P}^{-1}\mathbf{G}^h \\ &\quad +\frac{4\alpha^2}{L^2}\mathbf{G}\mathbf{P}^{-1}E_1\mathbf{P}^{-1}\mathbf{G}^h \end{aligned} \tag{4.64}$$

with

$$E_1 = E\{\mathbf{X}^*[kL]\mathbf{V}E\{\underline{\mathbf{S}}[kL]\underline{\mathbf{S}}^h[kL]\}\mathbf{V}^h\mathbf{X}[kL]\}. \tag{4.65}$$

Since s is assumed to be a white noise signal it follows that

$$E\{\underline{\mathbf{S}}[kL]\underline{\mathbf{S}}^h[kL]\} = B\sigma_s^2\mathbf{I}. \tag{4.66}$$

Together with the following quantities

$$\begin{aligned} E\{\mathbf{X}^*[kL]\mathbf{V}\mathbf{X}[kL]\} &= E\{\mathbf{X}^*[kL]\mathbf{V}^h\mathbf{X}[kL]\} = L\mathbf{P} \\ E\{\mathbf{X}^*[kL]\mathbf{V}\mathbf{V}^h\mathbf{X}[kL]\} &= L\mathbf{P} \end{aligned} \tag{4.67}$$

the above difference equation reduces to:

$$\begin{aligned} \Delta[(k+1)L] &= \Delta[kL] - 2\alpha\mathbf{G}\Delta[kL] - 2\alpha\Delta[kL]\mathbf{G}^h \\ &\quad +4\frac{B}{L}\sigma_s^2\alpha^2\mathbf{G}\mathbf{P}^{-1}\mathbf{G}^h. \end{aligned} \tag{4.68}$$

With $\Delta[kL] = \mathbf{F}\delta[kL]\mathbf{F}^h$ and $\mathbf{F}^{-1}\mathbf{P}^{-1}\mathbf{F} = \mathbf{C}^{-1}$ this equation is transformed back to time domain as:

$$\delta[(k+1)L] = \delta[kL] - 2\alpha\mathbf{g}\delta[kL] - 2\alpha\delta[kL]\mathbf{g} + \frac{4\alpha^2}{L}\sigma_s^2\mathbf{g}\mathbf{C}^{-1}\mathbf{g}. \tag{4.69}$$

This equation is analysed for different window functions in the next subsections.

4.4.1 Unconstrained window function

The window function is now defined as $\mathbf{g} = \mathbf{g}_{\mathrm{UNC}} = \mathbf{I}_B$ and with this the difference equation (4.69) reduces to:

$$\delta[(k+1)L] = (1-4\alpha)\delta[kL] + \frac{4\alpha^2}{L}\sigma_s^2\mathbf{C}^{-1}. \tag{4.70}$$

With this the quantity of interest $\bar{J}[kL]$ can be expressed with the following difference equation:

$$\bar{J}[kL] = \frac{trace\{\mathbf{C}\delta[kL]\}}{\sigma_s^2} = (1-4\alpha)\bar{J}[kL] + 4\alpha^2\frac{B}{L}. \tag{4.71}$$

From this equation it follows that convergence properties of the unconstrained BFDAF (3 FFTs) are independent of the input signal statistics. The rate of convergence ν_{20} and the final misadjustment $\overline{J}$ are given by

$$\nu_{20} \approx \frac{1.15}{\alpha} L \quad \text{and} \quad \overline{J} \approx \alpha \frac{B}{L}. \tag{4.72}$$

Comparison these results with the convergence properties of the BNLMS algorithm (2.45), that uses a white noise input signal, shows that both algorithms have the same rate of convergence ν_{20}. On the other hand for the unconstrained BFDAF algorithm B $(= N + L - 1)$ adaptive weights are fluctuating around their final value, while these are only N coefficients in the BNLMS case. For this reason the final misadjustment $\overline{J}$ of the unconstrained BFDAF is a factor B/N worse. In many practical situations, the processing delay L is choosen in the order of the number of adaptive weights (e.g. $N + 1$). For this situation the factor B/N equals 2 (=3dB).

In conclusion it follows that the unconstrained BFDAF is capable to decorrelate a coloured input signal. The windowing, that is needed for a correct overlap–save convolution/correlation, is performed by the adaptive filter itself at the cost of a factor B/N in accuracy for the final misadjustment. Note finally that, of course, the two decorrelation conditions, as mentioned in Section 4.2, must be satisfied. For the unconstrained case this implies that $\tau_{\max} < B/2 = (N + L - 1)/2$.

4.4.2 Constrained window function

The window function is now defined as:

$$\mathbf{g} = \mathbf{g}_{\text{con}} = \begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{4.73}$$

When, as in many practical situations, the adaptive weights are initiated with zeros it is obvious that the $B \times B$ matrix $\delta[kL]$ contains only elements different from zero in the upper left $N \times N$ corner. Thus with the $N \times N$ matrix $\delta'[kL]$, the $B \times B$ matrix $\delta[kL]$ can be written as:

$$\delta[kL] = \begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \delta'[kL] \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix}. \tag{4.74}$$

Substituting this in equation (4.69) and using the expression

$$\tilde{J}[kL] = \frac{trace\{\mathbf{C}\delta[kL]\}}{\sigma_s^2} \tag{4.75}$$

this results in the following difference equation:

$$\tilde{J}[kL] = (1-4\alpha)\tilde{J}[(k-1)L] + \frac{4\alpha^2}{L} trace\{\mathbf{C}\mathbf{g}\mathbf{C}^{-1}\mathbf{g}\}. \tag{4.76}$$

In this equation the $trace\{\cdot\}$ can be rewritten as:

$$trace\{\mathbf{C}\mathbf{g}\mathbf{C}^{-1}\mathbf{g}\} = trace\{(\mathbf{C}^{-1})' \cdot \mathbf{C}'\} \tag{4.77}$$

with the $N \times N$ matrices $\mathbf{C}'$ and $(\mathbf{C}^{-1})'$ defined as the $N \times N$ upper left part of the $B \times B$ matrices $\mathbf{C}$ and $\mathbf{C}^{-1}$ respectively, defined as:

$$\begin{aligned} \mathbf{C}' &= \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix} \mathbf{C} \begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \\ (\mathbf{C}^{-1})' &= \begin{pmatrix} \mathbf{I}_N & \mathbf{0} \end{pmatrix} \mathbf{C}^{-1} \begin{pmatrix} \mathbf{I}_N \\ \mathbf{0} \end{pmatrix} \end{aligned} \tag{4.78}$$

From this equation it follows that the constrained BFDAF (5 FFTs) has the following convergence properties:

$$\nu_{20} \approx \frac{1.15}{\alpha} L \quad \text{and} \quad \overline{J} \approx \alpha \frac{N}{L} \cdot \left(\frac{trace\{(\mathbf{C}^{-1})' \cdot \mathbf{C}'\}}{N} \right). \tag{4.79}$$

Thus the contstrained BFDAF has almost equal convergence properties as the BNLMS algorithm, with a white noise input signal. The only difference is that the final misadjustment $\overline{J}$ contains a deviation factor f that is defined as:

$$f = \frac{trace\{(\mathbf{C}^{-1})' \cdot \mathbf{C}'\}}{N}. \tag{4.80}$$

In order to get some insight into the quantitative value of this factor the following table shows some results. In this table the factor f is calculated for $N = 16$, and for different L. Two different input signals (ma(1) and ar(1)) with different Eigenvalue Ratios (ER) are given.

N	L	ER	ma(1) f	ar(1) f
16	8	1	1.00	1.00
		10	1.04	1.04
		100	1.12	1.20
		1000	1.15	1.58
16	16	1	1.00	1.00
		10	1.04	1.04
		100	1.13	1.21
		1000	1.18	1.62
16	32	1	1.00	1.00
		10	1.04	1.04
		100	1.16	1.22
		1000	1.23	1.69

From this table it follows that in general the deviation factor f is close to one. For large ER this is not correct any more. By the assumption $\tau_{\text{max}} < B/2$ the ER is restricted, and thus the deviation factor may be approximated to 1, as long as ER is not too large (or $\tau_{\text{max}} < B/2$). In [51] (p794 and 796) it was argued that this deviation factor is in practice restricted to be smaller than 2.

In conclusion it follows that the constrained BFDAF (5 FFTs) is, within the decorrelation conditions of Section 4.2, able to decorrelate a coloured input signal. The final misadjustment is, for many practical situations, a factor B/N better in comparison to the unconstrained BFDAF (3FFTs).

4.4.3 Efficient window function

As mentioned in the previous subsections all B weights converge to a final value with a certain variance for the unconstrained BFDAF. After convergence, B weights fluctuate and add to the final misadjustment, whereas only N weights are needed. In the constrained BFDAF this is brought about by forcing the last $L-1$ weights to zero, so as to lower the final misadjustment by a factor $B/N = (N+L-1)/N$, while maintaining the rate of convergence. This is done by multiplication in the time domain by the function $\mathbf{g}_{\mathrm{con}}$, necessitating, however, the use of two extra FFTs in comparison to the unconstrained BFDAF. Lowering the final misadjustment by the same amount in the unconstrained approach would reduce the rate of convergence by the same factor B/N. On the other hand, a direct convolution in the frequency domain, based on the transform of $\mathbf{g}_{\mathrm{con}}$, is very complicated.

In order to dispense with these two extra FFTs, but obtain the same rate of convergence and final misadjustment as with the constrained BFDAF, in [49,59] an BFDAF is proposed having a window function $\mathbf{g}_{\mathrm{cos}}$ in the loop that can easily be implemented in the frequency domain. This window function is defined for $i = 0, \cdots, B-1$ as

$$(\mathbf{g}_{\mathrm{cos}})_i = \frac{1}{2} + \frac{1}{2}\cos(\frac{2\pi}{B}i) \qquad (4.81)$$

that is transformed in the frequency domain to the matrix $\mathbf{G}_{\mathrm{cos}}$. This is an I–circulant tridiagonal matrix with

$$(\mathbf{G}_{\mathrm{cos}})_{k,l} = \begin{cases} B/2 & \text{for } k = l \\ B/4 & \text{for } k = l \pm 1 \\ B/4 & \text{for } k = 0 \text{ and } l = B-1 \\ B/4 & \text{for } k = B-1 \text{ and } l = 0 \\ 0 & \text{elsewhere.} \end{cases} \qquad (4.82)$$

In many practical situations some a priori knowledge of the "unknown" Wiener system is present. Here it is assumed that the impulse response of this function is globally decaying function. Thus the coefficients are greatest at the beginning, the rate of convergence of an adaptive filter being largely determined by the speed at which the greatest coefficients converge. The window function $\mathbf{g}_{\mathrm{cos}}$, a raised cosine function, will give these first coefficients the same speed as $\mathbf{g}_{\mathrm{con}}$. When choosing in first instance $L = N+1$ and thus $B = 2N$ the sum of values of $(\mathbf{g}_{\mathrm{cos}})_i$ and $(\mathbf{g}_{\mathrm{con}})_i$ are both equal to N, leading to the same final misadjustment. In contrast to the window function $\mathbf{g}_{\mathrm{con}}$, the multiplication in the time domain with the diagonal matrix $\mathbf{g}_{\mathrm{cos}}$ can easily be implemented in frequency domain by a simple cyclic convolution with the three components of the tridiagonal I–circulant matrix $\mathbf{G}_{\mathrm{cos}}$ as sketched in Fig. 4.6. The three components of $\mathbf{G}_{\mathrm{cos}}$ are moreover powers of two, which

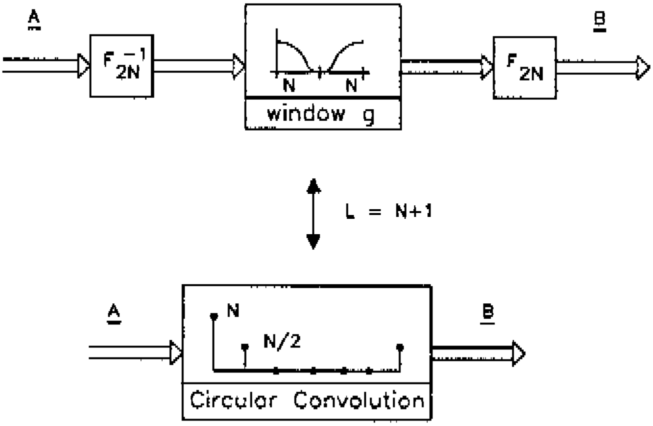

Figure 4.6: *Efficient implementation of raised cosine window function*

makes the few extra multiplications needed very simple. In conclusion it follows that for a global decaying Wiener system the given efficient BFDAF, with a raised cosine window $g_{\cos}$, has convergence properties of the constrained BFDAF and it can be implemented with the complexity that is in the order of the unconstrained BFDAF (3FFTs).

Notes:

- For the more general case $B = N + L - 1$ and $L \geq N + 1$ this technique can also be applied, but the gain can only be a factor two in stead of B/N.
- In some practical cases it is better to use a shifted version of the raised cosine window function as described in [59]. Examples are a causal linear phase low or high pass filter, or an impulse response with a delay at the beginning.

4.5 Experiments

The main analytical results of the previous sections are verified in this section with experiments. For these experiments the system, as given in Fig. 1.7, is used.

4.5.1 FDAF ($L = 1$)

In this experiment it is shown that the decorrelation properties of a frequency domain adaptive filter are limited by the two main decorrelation conditions as described in Section 4.2. In order to do so an adaptive filter with $N = 32$ coefficients uses an FDAF algorithm ($L = 1$) with DFT length $B = N + L - 1$. The adaptive weights are initialized in such a way that $|D_l[0]|^2$ is constant, and the adaptation constant is chosen as $\alpha = 4.6/1000$. The results of this experiment are given in Fig. 4.7 for an ma(1) input signal and in Fig. 4.8 for an ar(1) signal. Both signals have an $ER = 100$ ($\rightarrow a = 0.8182$). The function that is plotted in these figures is:

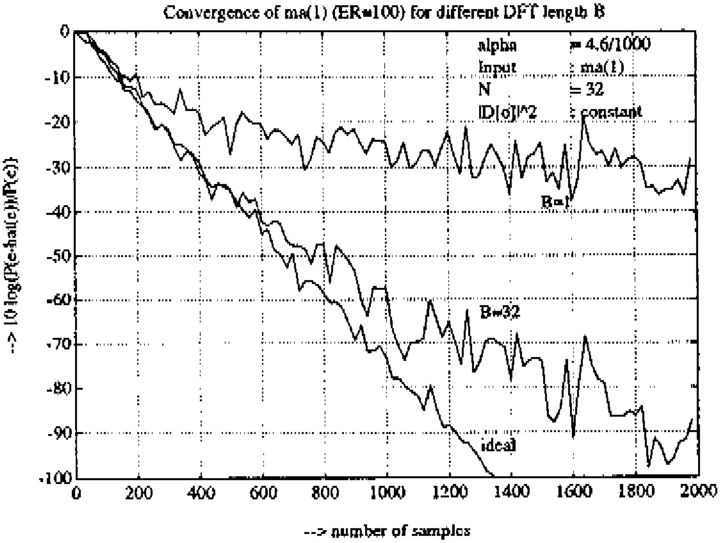

Figure 4.7: *FDAF with ma(1) input signal for different DFT length B*

$$10 \cdot {}^{10}\log\left(\frac{E\{(e[k] - \hat{e}[k])^2\}}{E\{e^2[k]\}}\right). \tag{4.83}$$

From this function it is possible to measure the deviation from the "ideal" rate of convergence $\nu_{20} = 1.15/\alpha = 250$ samples, that is reached

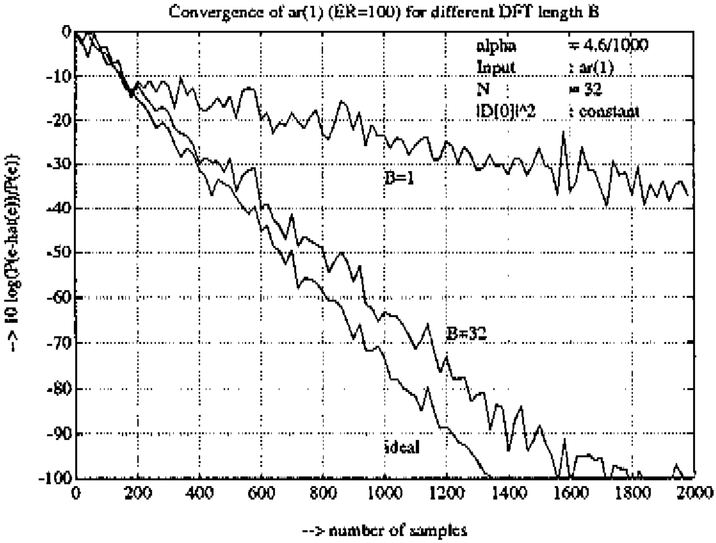

Figure 4.8: *FDAF with ar(1) input signal for different DFT length B*

when the input signal is perfectly decorrelated. From the PDR functions, plotted in Fig. 4.2, it follows that this ideal situation only occurs when the DFT length B approaches infinity. In the experiment this "ideal" situation is simulated by using an FDAF with a white noise input signal. Furthermore the situation with an DFT length $B = 1$ is simulated with a the time domain adaptive filter using an NLMS update algorithm: No decorrelation takes place.

From the PDR function of the ma(1) signal, top figure of Fig. 4.2, it follows that an FDAF with $B = 32$ is capable to equalize the first part of the power spectrum to 1. The spectral zero (at $\theta = \pi$) is, for this value of B, only increased to 0.4. In order to increase this level towards 1, an DFT of length $B \rightarrow \infty$ is needed. From the experimental result of an FDAF with $B = 32$, as plotted in Fig. 4.7, it follows that in first instance the rate of convergence equals the ideal one. Decreasing below -70 dB shows that indeed to decrease another 20 dB costs $250/0.4 \approx 600$ samples.

On the other hand it follows from the PDR function of the ar(1) signal, bottom figure of Fig. 4.2, that an FDAF with $B = 32$ is capable to equalize the complete power spectrum to 0.9 . In order to increase this level to 1, a DFT with a very large length is needed. From the experiment with $B = 32$, as plotted in Fig. 4.8, it follows that for this situation the rate of convergence ν_{20} is indeed in the order of $250/0.9 \approx 280$ samples.

The conclusions of these experiments is that a frequency domain adaptive filter can decorrelate an input signal acceptably, when the two main decorrelation conditions of Section 4.2 are satisfied.

4.5.2 BFDAF

In this experiment it is shown with $N = 32$ and $L = 33$ ($\rightarrow B = 64$), that the difference in final misadjustment between the constrained (5FFTs) and unconstrained BFDAF (3FFTs) is roughly a factor $B/N = 2$ (=3 dB). Above that it is shown that the efficient cosine BFDAF (3FFTs) has almost the same convergence properties as the constrained BFDAF. Fig. 4.9 gives the results of this experiment. In this figure

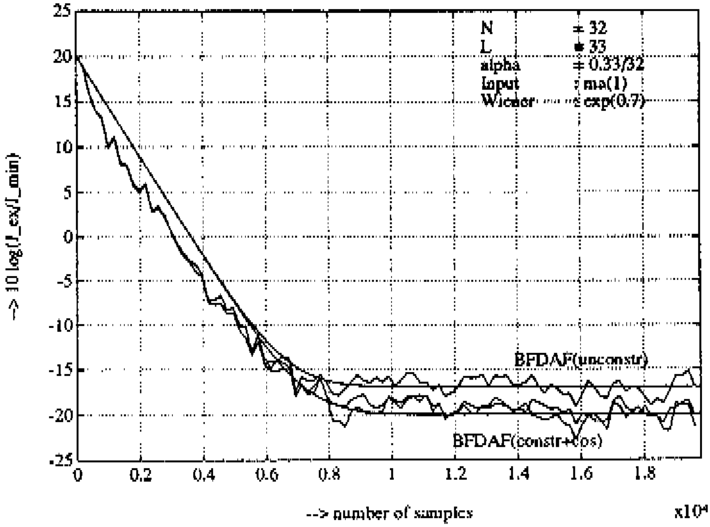

Figure 4.9: *Convergence of different BFDAFs*

the function $10 \cdot {}^{10}\log(\tilde{J}[kL])$ is plotted as a function of the number of input samples. The input signal is an ma(1) signal with $ER = 100$ ($\rightarrow a = 0.8182$). The unknown Wiener system is an exponential decaying function defined as:

$$w_{\mathrm{opt},i} = (0.7)^i \quad \text{for} \quad i = 0, 1, \cdots, 31. \tag{4.84}$$

Furthermore with $\alpha = 0.33/32$ the "ideal" results are as follows:

$$\begin{aligned} \nu_{20} &\approx 3680 \\ 10 \cdot {}^{10}\log(\overline{J}_{\mathrm{constr}}) &\approx -20 \\ 10 \cdot {}^{10}\log(\overline{J}_{\mathrm{uncons}}) &\approx -17 \\ 10 \cdot {}^{10}\log(\overline{J}_{\cos}) &\approx -20 \end{aligned}$$

4.6 Discussion

In this chapter it is shown that, within the two main decorrelation conditions of Section 4.2, convergence properties of the BFDAF can be made independent of the input signal statistics by simple power normalization. Moreover complexity is reduced by implementing the convolution and correlation in frequency domain with FFTs as transformation between time- and frequency domain.

The final misadjustment of the constrained BFDAF (5FFTs) is a factor B/N better in comparison to the unconstrained BFDAF (3FFTs). In many practical cases, when from a priori knowledge it is known that the "unknown" Wiener system is a globally decaying function, then the cosine BFDAF (3FFTs) is an efficient alternative, with convergence properties equal to the constrained BFDAF (5FFTs) and complexity equivalent to the unconstrained BFDAF (3FFTs).

Finally it is noted that from literature [37] it is known that the power of a signal can be calculated by correlating the signal with itself using equivalent overlap save procedures in frequency domain as mentioned in Chapter 3. Following this method in the given BFDAF introduces an extra FFT in order to calculate the power in a correct way. Thus although the estimate $P_l = \frac{1}{B}E\{X_l[kL] \cdot X^*[kL]\}$ is not correct it is shown in this chapter that for the BFDAF algorithm decorrelation can be performed in an acceptable way within the given decorrelation conditions.

Chapter 5

Partitioned Frequency Domain Adaptive Filters

The previous chapter gives an overview of some Frequency Domain Adaptive Filtering approaches. Decorrelation of the input signal is carried out in frequency domain by normalizing the power spectral density function. This is done by dividing each separate frequency component by its power spectral density function. The resolution of this function equals the number $B = N + L - 1$ of frequency components, with N the adaptive length filter and L the block length or processing delay. On the other hand the statistical properties of the input signal, and thus the needed number of divisions, has no relation at all with the segment length B. Assume for example that the autocorrelation function of the input signal has only a few nonzero values within the segment length. The spectral density function of such a signal is smooth, and the first question is: Is I possible to reduce complexity by performing the complete decorrelation with less than B divisions?

Another practical problem is that the length B of the FFTs used in the BFDAF approach must be a power of two. This implies that for large N (as for an acoustic echo canceller $N \approx 1024$ samples), also the processing delay L must be chosen very large (e.g. $L = 1025$ samples). This however may be an unacceptable number in practice. A second question is: Is it possible to obtain more freedom in the choice of the processing delay L while still being able to implement the needed Fourier transform of the adaptive filter with FFTs?

A possible solution to these problems is to partition the adaptive filter, and by that the update algorithm, in separate parts. Implementing this in an efficient way leads to partitioned frequency domain adaptive filter approaches.

In Section 5.1 the adaptive filter, using the BFDAF algorithm, is partitioned in separate consecutive parts. Implementing this in an efficient way leads to the Partitioned Block Frequency Domain Adaptive Filter (PBFDAF) [61,6,4,69,30,57]. In Section 5.2 complexity of the BFDAF and the PBFDAF is compared for a practical example. In Section 5.3 the consecutive partitioning concept is generalized for the "sliding" FDAF. It is not necessary to partition the adaptive filter in consecutive parts, also parts may be interleaved. This results in the mixed Partitioned Frequency Domain Adaptive Filter (mixed-PFDAF) [56,58], in which a DFT of length $M = N/K$ is used. With this mixed concept it is possible to search a way of partitioning for which convergence of the partitioned filter, with smaller DFT length M, has equal convergence properties as the "non–partitioned" FDAF structure with DFT length N. Section 5.4 describes decorrelation conditions for the mixed–PFDAF structure. The aim of this section is to search those conditions for the input signal statistics for which convergence properties of the partitioned structure are equal to those of the original frequency domain adaptive filter. In Section 5.5 experimental results are given, while Section 5.6 gives some conclusions.

5.1 Partitioned BFDAF

In this section an adaptive filter is partitioned in K consecutive separate smaller adaptive filters each having length $M = N/K$, and each using an BFDAF update algorithm. The parameter K is restricted to be an integer in the range $\{1, 2, \cdots, N\}$, and with this parameter it is possible to vary the new structure between a frequency domain and time domain structure. Namely for $K = 1$ the new structure equals the BFDAF as given in Chapter 4, while for $K = N$ this structure reduces to the NLMS algorithm.

The first step of the partition concept is to split the impulse response $\underline{\mathbf{w}}[kL]$ of the original adaptive filter in K consecutive parts of equal length M as:

$$\underline{\mathbf{w}}[kL] = ((\underline{\mathbf{w}}_{K-1}[kL])^t, \cdots, (\underline{\mathbf{w}}_1[kL])^t, (\underline{\mathbf{w}}_0[kL])^t)^t \tag{5.1}$$

with for $q = 0, 1, \cdots, K-1$

$$\underline{\mathbf{w}}_q[kL] = (w_{(q+1)M-1}[kL], \cdots, w_{qM+1}[kL], w_{qM}[kL])^t. \tag{5.2}$$

The $N \times L$ input signal matrix $\chi[kL]$ can also be partitioned into K equal separate parts as:

$$\chi[kL] = \begin{pmatrix} \chi_{K-1}[kL] \\ \cdot \\ \cdot \\ \cdot \\ \chi_1[kL] \\ \chi_0[kL] \end{pmatrix} \tag{5.3}$$

with for $q = 0, 1, \cdots, K-1$ the $M \times L$ matrices $\chi_q[kL]$ defined as:

$$\chi_q[kL] = (\underline{\mathbf{x}}[kL - qM - L + 1], \cdots, \underline{\mathbf{x}}[kL - qM - 1], \underline{\mathbf{x}}[kL - qM]) \tag{5.4}$$

in which the length M vector $\underline{\mathbf{x}}[kL - qM - u]$ for $u = 0, 1 \cdots, L-1$ contains the elements:

$$(x[kL - qM - u - M + 1], \cdots, x[kL - qM - u - 1], x[kL - qM - u])^t. \tag{5.5}$$

With this the original length L output signal vector $\hat{e}_L[kL]$ of the adaptive filter can be rewritten as:

$$\hat{e}_L[kL] = \chi^t[kl]\underline{\mathbf{w}}[kL] = \sum_{q=0}^{K-1} \chi_q^t[kL]\underline{\mathbf{w}}_q[kL]. \tag{5.6}$$

From the given partitioning of the input signal matrix $\chi[kL]$ it follows also that each part $\chi_q[kL]$ can be written as a delayed version of the first part $\chi_0[kL]$ as

$$\chi_q[kL] = \chi_0[kL - qM] \quad \text{with} \quad q = 0, 1, \cdots, K-1. \tag{5.7}$$

All K separate adaptive weights vectors are updated using a block update algorithm with segment length $B = M + L - 1$, adaptive filter length $M = N/K$ and processing delay $L \geq 1$. This is depicted in Fig. 5.1. The next step is to use a BFDAF structure for each separate part. For this reason each partitioned J–circulant input signal matrix $\chi_q[kL]$ is, as in the previous chapter, related to the I–circulant matrix $\chi_q^c[kL]$ as follows:

$$\begin{pmatrix} \mathbf{I}_M & \mathbf{0} \end{pmatrix} \cdot \chi_q^c[kL] \cdot \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_L \end{pmatrix} = \mathbf{J}_M \cdot \chi_q[kL]. \tag{5.8}$$

This I–circulant matrix is transformed to frequency domain with the $B \times B$ Fourier matrix $\mathbf{F}$ as follows:

$$\mathbf{F}\chi_q^c[kL]\mathbf{F}^{-1} = \mathbf{X}_q^*[kL] = diag\{\underline{\mathbf{X}}_q^*[kL]\} \tag{5.9}$$

with

$$\underline{\mathbf{X}}_q[kL] = (X_{q,0}[kL], \cdots, X_{q,B-1}[kL])^t. \tag{5.10}$$

The adaptive weight vector $\underline{\mathbf{w}}_q[kL]$ is windowed and transformed to frequency domain as:

$$\underline{\mathbf{W}}_q[kL] = \mathbf{F} \begin{pmatrix} \mathbf{I}_M \\ \mathbf{0} \end{pmatrix} \mathbf{J}_M \underline{\mathbf{w}}_q[kL]. \tag{5.11}$$

As in the previous chapter this definition only holds for the constrained case. A more general definition is:

$$\underline{\mathbf{W}}_q[kL] = \mathbf{F}\mathbf{J}_B\underline{\mathbf{w}}'_q \tag{5.12}$$

in which $\underline{\mathbf{w}}'_q$ is a vector of dimension $B = M + L - 1$ of the adaptive weights. Furthermore each BFDAF needs a $B \times B$ window matrix $\mathbf{G} = \mathbf{F}\mathrm{g}\mathbf{F}^{-1}$ with

$$\mathrm{g} = \begin{pmatrix} \mathbf{I}_M \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I}_M & \mathbf{0} \end{pmatrix} \tag{5.13}$$

and the $B \times 1$ transformed version of the length L residual signal vector $\underline{\mathbf{r}}_L[kL]$:

$$\underline{\mathbf{R}}'[kL] = \mathbf{F} \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_L \end{pmatrix} \underline{\mathbf{r}}_L[kL]. \tag{5.14}$$

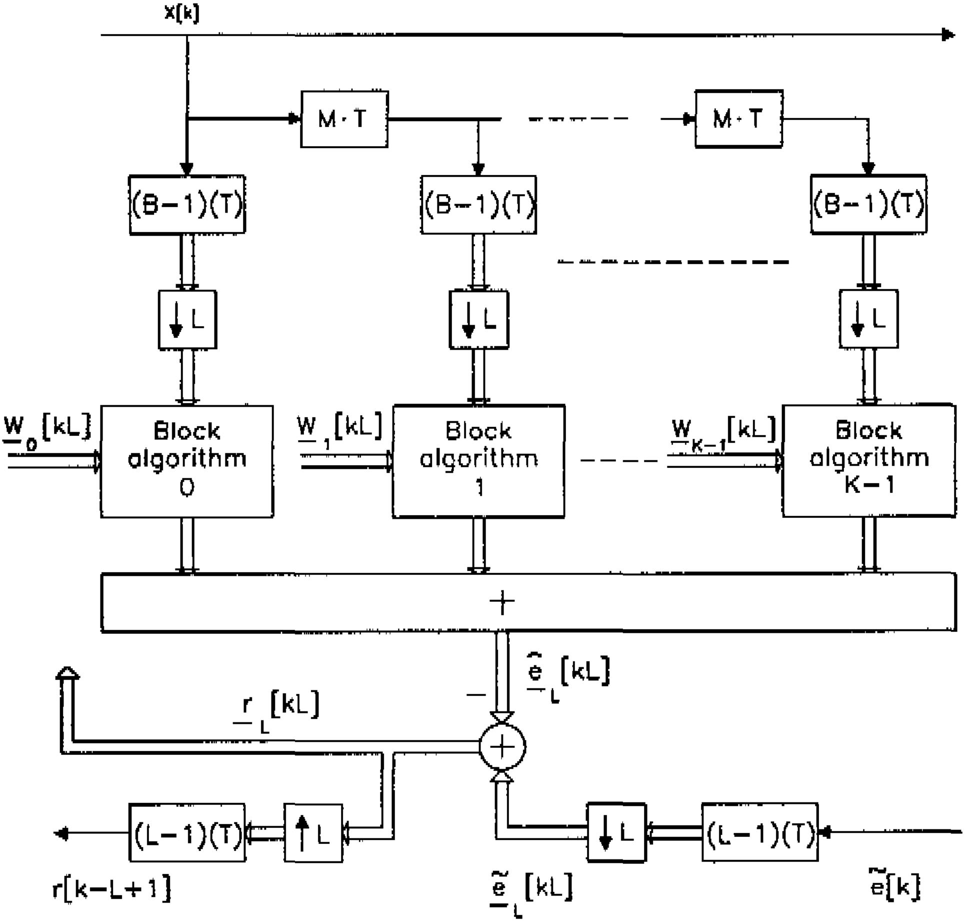

Figure 5.1: *Partitioning concept for block update algorithm with* $B = M + L - 1$, $M = N/K$ *and* $L \geq 1$

Now the BFDAF update equation for each separate adaptive weight vector $\underline{\mathbf{W}}_q[kL]$ $(q = 0, 1, \cdots, K-1)$ is given by:

$$\underline{\mathbf{W}}_q[(k+1)L] = \underline{\mathbf{W}}_q[kL] + \frac{2\alpha}{L}\mathbf{G}\mathbf{P}_q^{-1}\mathbf{X}_q^*[kL]\underline{\mathbf{R}}'[kL] \qquad (5.15)$$

in which the decorrelation takes place by the power normalization, that is performed by the inverse of the $B \times B$ diagonal power matrix

$$\mathbf{P}_q = \frac{1}{B}E\{\mathbf{X}_q^*[kL]\mathbf{X}_q[kL]\}. \qquad (5.16)$$

By combining different DFTs of the separate BFDAFs the structure can be implemented more efficiently. As mentioned above all partitioned input signal matrices $\chi_q[kL]$ are delayed versions of the first one $\chi_0[kL]$. This also holds for the circulant forms and their frequency domain parts:

$$\mathbf{X}_q^*[kL] = \mathbf{F}\chi_q^c[kL]\mathbf{F}^{-1} = \mathbf{F}\chi_0^c[kL - qM]\mathbf{F}^{-1} = \mathbf{X}_0^*[kL - qM]. \qquad (5.17)$$

With the restriction

$$M = \mu L \quad \text{with} \quad \mu \in \{1, 2, \cdots, M\} \qquad (5.18)$$

it follows that it is possible to write the time index $kL - qM$ as $(k - q\mu)L$ and the delayed versions $\mathbf{X}_0[(k - q\mu)L]$ can simply be obtained by delaying $\mathbf{X}_0[kL]$ in frequency domain over $q \cdot \mu$ delay elements of length $T_L = L \cdot T$. Thus under the given restrictions (5.18) all separate DFTs, that transform delayed versions of the input signal, can be combined to one DFT while the delays are performed in frequency domain. Furthermore every separate BFDAF has an inverse DFT and discards the first $M - 1$ samples. These functions can be combined to one DFT with the "throwing away" part after the addition point. Finally each separate BFDAF needs the same vector $\underline{\mathbf{R}}'[kL]$, and thus only one DFT is needed to calculate this vector. The computational complexity can be further reduced when the input signal is stationary. For such signals it can be shown that the diagonal power matrix $\mathbf{P}_q$ from equation (5.16) is independent of the partitioning index q and thus

$$\mathbf{P}_q = \mathbf{P} = diag\{P_0, P_1, \cdots, P_{B-1}\}. \qquad (5.19)$$

The power normalization can be performed directly on the $B \times 1$ transformed residual vector $\underline{\mathrm{R}}'[kL]$, and only B divisions are needed. The above mentioned combinations of DFTs and the reduced number of divisions, lead to the Partitioned Block Frequency Domain Adaptive Filter (PBFDAF), that is shown in Fig. 5.2. The filtering part of the PBFDAF structure (right hand side of figure 5.2) is depicted in an alternative way in Fig. 5.3. From this figure it follows that each frequency component $X_{0,l}[kL]$, with $l = 0, 1, \cdots, B-1$, is used as a (complex) input signal for a transversal filter with delay elements $\mu \cdot T_L$. The K adaptive weights $W_{0,l}[kL], \cdots, W_{K-1,l}[kL]$ of each transversal filter are updated by using an NLMS algorithm with complex weights. The normalization of component l is performed with $1/P_l$. From this figure it also follows that each new segment of B input signal samples has an overlap of $M-1$ samples with the previous segment. By this it is obvious that, even when a white noise input signal is applied, the frequency component $X_{0,l}[kL]$ is "correlated in time" with delayed versions of it that are used in the adaptive complex NLMS filter of component l. This "correlation in time" for each frequency component l can degrade the decorrelation properties of the PBFDAF structure in comparison to the BFDAF structure. On the other hand, the windows (used for the constrained approach), do cancel a part of this "correlation in time". From the experimental results at the end of this chapter it follows that, even for colourd input signals, the convergence properties are not degraded seriously for small K.

As a final comment of this section it is noted that the PBFDAF method has much resemblance with the filterbank approach as described in [31]. In general, subband filters are designed such that, in contrast to the DFT, they have a good frequency selectivity. The impulse response of such a subband filter also introduces a "correlation in time" in the subbands. Furthermore, when implementing these subband filters as causal filters, they also introduce a processing delay.

Thus, besides a good frequency selectivity the subband filters of the filterbank approach must be designed, from the adaptive point of view, in such a way that they introduce minimal "correlation in time", and that the processing delay is still acceptable.

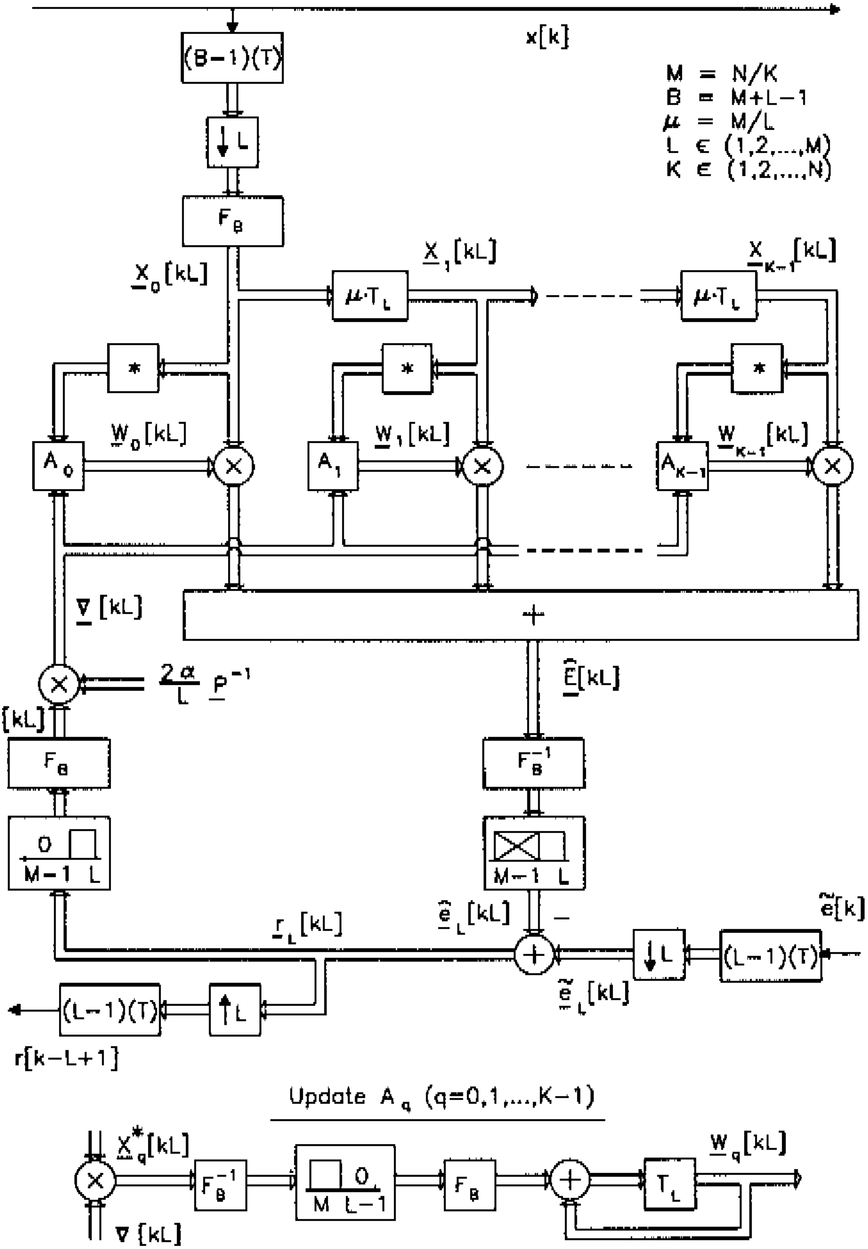

Figure 5.2: *Efficient implementation of PBFDAF structure*

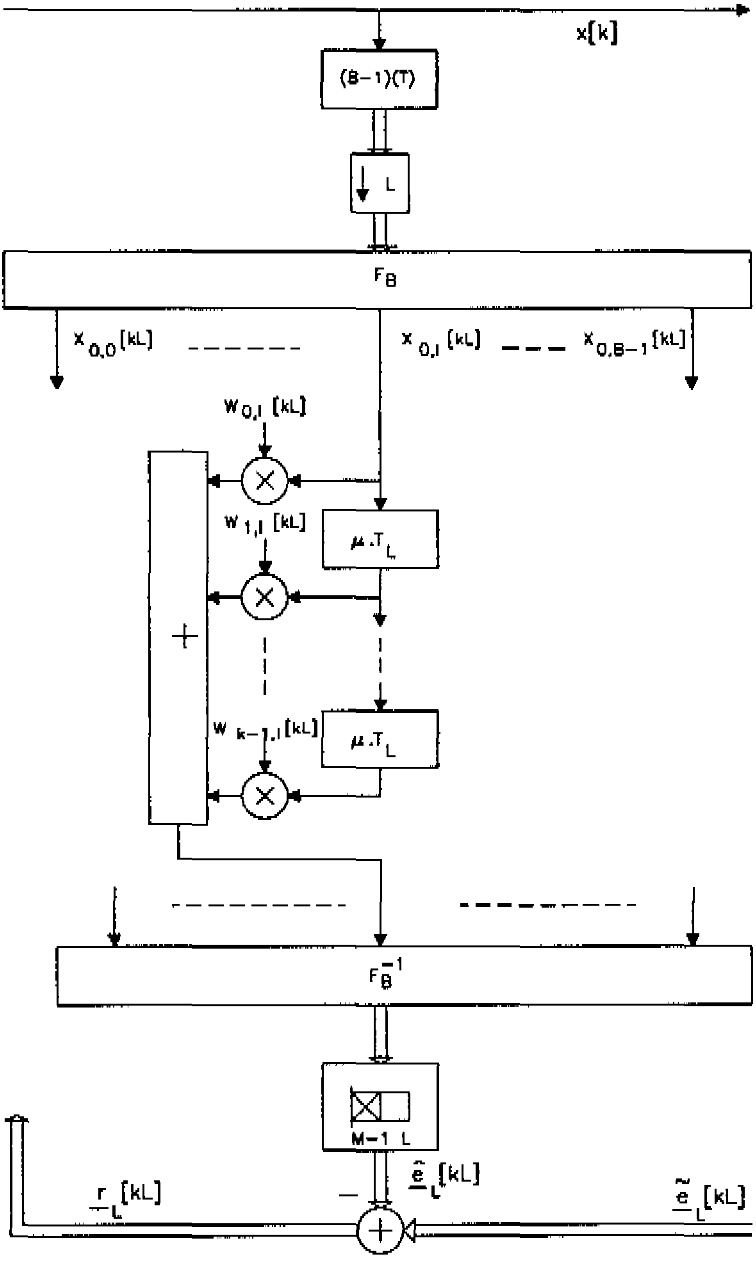

Figure 5.3: *Alternative figure for filtering part of PBFDAF*

5.2 Complexity PBFDAF in relation to BFDAF

The complexity for the efficiently implemented PBFDAF structure, as depicted in Fig. 5.2, is given by:

$$\begin{aligned} MUL_{PBFDAF} &= \frac{1}{2}\left(\frac{(3+2K\cdot win)\cdot 2B^2\log(B)+2\cdot 4KB}{L}\right) \\ DIV_{PBFDAF} &= \frac{1}{2}\cdot\frac{4B}{L} \end{aligned} \tag{5.20}$$

with

$$\begin{aligned} B &= M+L-1 \\ M &= N/K \quad \text{with} \quad K\in\{1,2,\cdots N\} \\ L &\in \{1,2,\cdots,(N/K)\} \\ win &= \begin{cases} 1 & \text{constrained case} \\ 0 & \text{unconstrained case .} \end{cases} \end{aligned} \tag{5.21}$$

With $K=1$ and $L\geq 1$ these numbers equal, of course, the complexity of the BFDAF. For the BFDAF structure the complexity needed to calculate one new output sample becomes smaller for increasing L. On the other hand, since L equals the processing delay, this value may in practice not increase above some maximum allowable value L_{max}. An example is the acoustic echo canceller [31], from which the acoustic echo to be cancelled is in the order of 100 msec. On the other hand, in order not to be audible, the processing delay may not exceed say 25 msec. With a sampling frequency of 10 kHz, in the area of speech applications, this yields an adaptive transversal filter with at least $N_{\text{min}}=1000$ coefficients, while the maximal allowable processing delay is in the order of $L_{\text{max}}\approx 250$ samples. For this example, the needed number of multiplications for the PBFDAF structure is compared to the number for the BFDAF structure for different K, and the result is plotted in Fig. 5.4. For all structures the processing delay is chosen as $L=L_{\text{max}}=250$ while the total number N of adaptive weights ($N=M\cdot K$) is chosen in such a way that the Fourier transform can be implemented as an FFT, thus the segment length $B=M+L-1$ must be a power

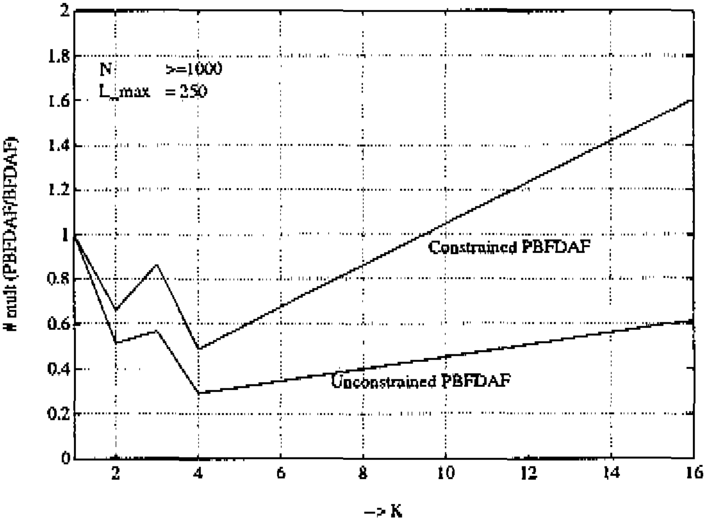

Figure 5.4: *Relative number of real multiplications of PBFDAF/BFDAF as function of K*

of two. This implies that N can be larger than N_{min}. For example when $K = 1$ (BFDAF) and with $L = 250$, the segment length must be $B = 2048$, resulting in $N = 1789$. Furthermore, in order to make an acceptable comparison, it is assumed that all structures are able to decorrelate the applied coloured input signal, and thus the convergence properties are described by the equations from the previous chapter. The initial speed of convergence (ν_{20}) is for all structures the same, and is given by the following number of samples:

$$\nu_{20} \approx \frac{1.15}{\alpha} 250. \tag{5.22}$$

The final misadjustment is given by the following formula

$$\overline{J} = \alpha \frac{M \cdot K}{L} \tag{5.23}$$

and this quantity differs from one structure to the other. All used numbers are given in the following table:

K	M	$N = M \cdot K$	B	$\overline{J}$
1	1789	1789	2048	7.2 α
2	775	1550	1024	6.2 α
3	775	2325	1024	9.3 α
4	263	1052	512	4.2 α
5	263	1315	512	5.3 α
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
141	263	37083	512	148.3 α
142	7	994	256	4.0 α

For this example it follows from this table and Fig. 5.4 that the PBFDAF, with $K = 4$, is more accurate (factor $7.2/4.2 = 2.3$ dB) and needs less multiplications (factor 0.5) in comparison to the BFDAF structure. Finally it is noted that the PBFDAF requires less divisions than the BFDAF approach, since the normalization is carried out over less frequency components.

5.3 Mixed–PFDAF

As mentioned in the previous section the consecutive partitioning scheme results in an adaptive filter structure whose convergence properties can degrade in comparison to the original frequency domain adaptive filter. In this section it is shown that the consecutive partitioning concept as given in Section 1 can be generalized in an efficient way for the sliding FDAF approach ($L = 1$ and $B = N + L - 1 = N$) in such a way that this consecutive partitioning is a subclass of this generalization. The same generalization can be applied to block processing techniques ($L > 1$), but the only subclass that can be implemented efficiently is the given consecutive partitioning. In Section 5.4 input signal conditions are derived for this generalized scheme in order to have equal convergence properties in comparison to the FDAF approach.

The first step needed for this generalization is to rewrite the output signal $\hat{e}[k]$ of the adaptive filter

$$\hat{e}[k] = \sum_{i=0}^{N-1} x[k-i]w_i[k] = \underline{\mathbf{x}}^t[k] \cdot \underline{\mathbf{w}}[k] \tag{5.24}$$

by partitioning both the input signal vector $\underline{\mathbf{x}}[k]$ and the adaptive weight vector $\underline{\mathbf{w}}[k]$. To this end the two vectors are "mixed" in K separate vectors of length M as follows:

$$\begin{aligned} \underline{\mathbf{x}}^{\mathrm{m}}[k] &= ((\underline{\mathbf{x}}^{\mathrm{m}}_{K-1}[k])^t, \cdots, (\underline{\mathbf{x}}^{\mathrm{m}}_0[k])^t)^t \\ \underline{\mathbf{w}}^{\mathrm{m}}[k] &= ((\underline{\mathbf{w}}^{\mathrm{m}}_{K-1}[k])^t, \cdots, (\underline{\mathbf{w}}^{\mathrm{m}}_0[k])^t)^t. \end{aligned} \tag{5.25}$$

While the exact way of "mixing" is defined further down, the above convolution sum $\hat{e}[k]$ can be rewritten as:

$$\hat{e}[k] = (\underline{\mathbf{x}}^{\mathrm{m}}[k])^t \cdot \underline{\mathbf{w}}^{\mathrm{m}}[k] = \sum_{q=0}^{K-1} (\underline{\mathbf{x}}^{\mathrm{m}}_q[k])^t \cdot \underline{\mathbf{w}}^{\mathrm{m}}_q[k]. \tag{5.26}$$

With this the length N convolution sum $\hat{e}[k]$ is rewritten as K separate smaller length $M = N/K$ convolutions sums. Furthermore, as depicted in Fig. 5.5, each of these smaller convolution sums can be performed

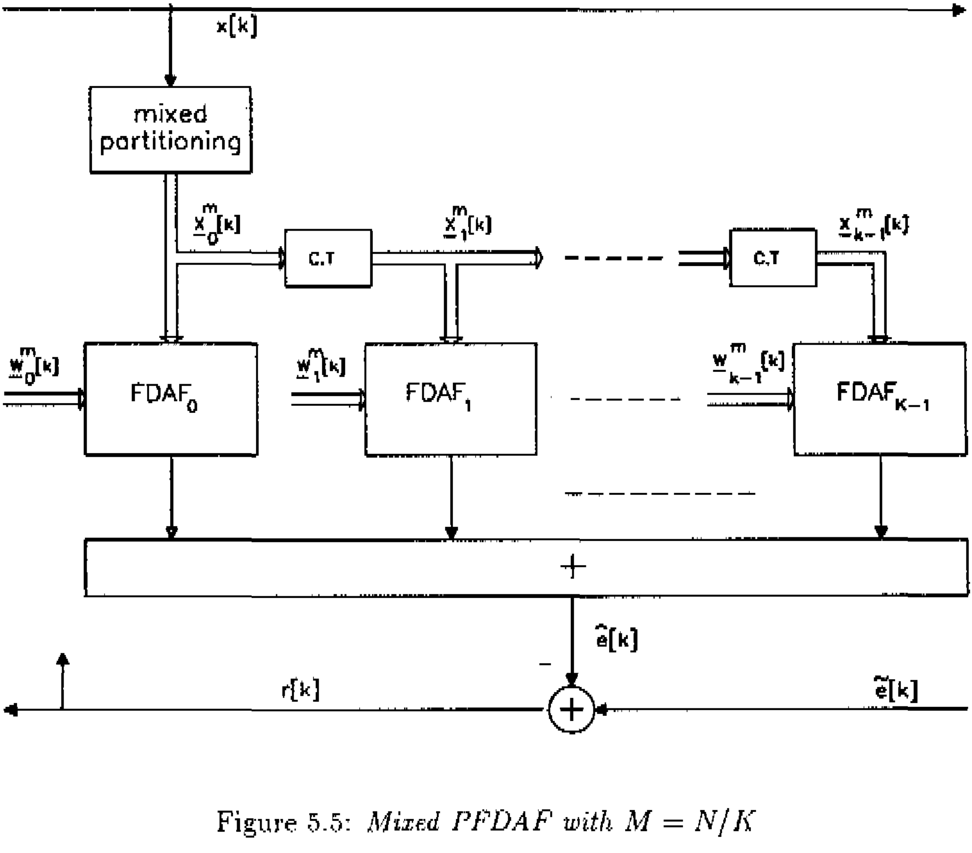

Figure 5.5: *Mixed PFDAF with* $M = N/K$

with an FDAF, containing one DFT of length M. To this end the convolution sum is rewritten as follows:

$$\begin{aligned}\hat{e}[k] &= \sum_{q=0}^{K-1} (\underline{\mathbf{x}}_q^{\mathrm{m}}[k])^t \left(\mathbf{F}_M \mathbf{F}_M^{-1}\right) \underline{\mathbf{w}}_q^{\mathrm{m}}[k] = \frac{1}{M} \sum_{q=0}^{K-1} (\mathbf{F}_M \cdot \underline{\mathbf{x}}_q^{\mathrm{m}}[k])^t \cdot (\mathbf{F}_M^{*} \cdot \underline{\mathbf{w}}_q^{\mathrm{m}}[k \\ &= \frac{1}{M} \sum_{q=0}^{K-1} (\underline{\mathbf{X}}_q^{\mathrm{m}}[k])^t \cdot (\underline{\mathbf{W}}_q^{\mathrm{m}}[k])^{*}. \qquad (5.2\end{aligned}$$

The next step is to implement the mixed structure efficiently by combining DFTs. By defining the transformed vector

$$\underline{\mathbf{X}}_0^{\mathrm{m}}[k] = \mathbf{F}_M \cdot \underline{\mathbf{x}}_0^{\mathrm{m}}[k] = (X_0^{\mathrm{m}}[k], \cdots, X_{M-1}^{\mathrm{m}}[k])^t \qquad (5.28)$$

it follows that only one DFT of length M has to be applied to the input signal when the other Fourier transforms are simlpy delayed versions of this vector:

$$\underline{\mathbf{X}}_q^{\mathrm{m}}[k] = \underline{\mathbf{X}}_0^{\mathrm{m}}[k - qC] \quad \text{with} \quad q = 0, \cdots, K-1 \qquad (5.29)$$

with C some constant. This can be archieved by defining the vectors $\underline{\mathbf{x}}_q^{\mathrm{m}}[k]$ for $q = 1, 2, \cdots, K-1$ as delayed versions, over C samples, of the first vector $\underline{\mathbf{x}}_0^{\mathrm{m}}[k]$ as shown in Fig. 5.5 with

$$\underline{\mathbf{x}}_q^{\mathrm{m}}[k] = \underline{\mathbf{x}}_0^{\mathrm{m}}[k - qC] \quad \text{for} \quad q = 1, \cdots, K-1. \qquad (5.30)$$

By defining $M = C \cdot I$ it follows that the first vector $\underline{\mathbf{x}}_0^{\mathrm{m}}[k]$ of length M, contains the input samples

$$x[k - (u + sN/I)] \quad \text{with} \quad u = 0, \cdots, C-1 \quad \text{and} \quad s = 0, \cdots I-1. \qquad (5.31)$$

In literature [56] two partitioning schemes for the FDAF are known: In the "consecutive" partitioning scheme K times M consecutive samples are selected, while on the other hand the "comb" partitioning scheme selects K times M samples, but every following sample is interleaved by leaving out $(K-1)$ samples. The "mixed" partitioning, conceptually is depicted in Fig. 5.6(a), is in between these two schemes: The M samples are split in $M = C \cdot I$ samples, where C gives the number of consecutive samples. With I in the range $1, 2, \cdots, M$ it follows that

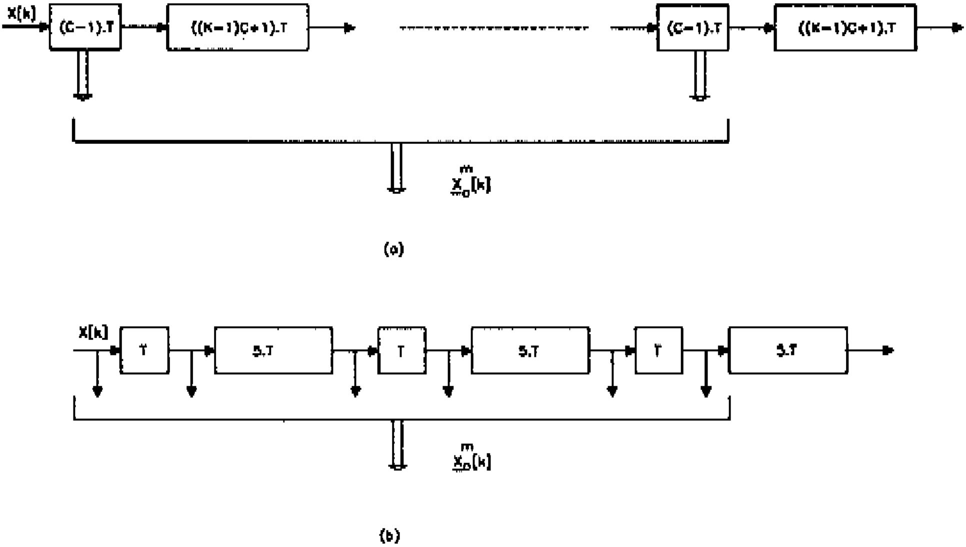

Figure 5.6: *(a) Mixed partitioning concept with* $M = N/K$ *and* $I = M/C$
(b) Example with $N = 18, K = 3$ *and* $C = 2$

the mixed partitioning equals the consecutive–partitioning for $I = 1$ and equals the comb–partitioning for $I = M$. An example of choosing M out of N samples, according to the above mixed concept, is given in Fig. 5.6(b). Since the length of a DFT must be an integer it is obvious that all above used values must be chosen such that N, M, K, I and C are integer values. When the DFT is implemented as an FFT, M must be a power of two as well. On the other hand when $M = N$, and thus $K = 1$ the values of I and C are not relevant any more and may be chosen arbitrarily as $C = M$ and $I = 1$. Furthermore the next K adaptive weight vectors of length M are needed for $q = 0, \cdots, K-1$:

$$\underline{\mathbf{W}}_q^{\mathrm{m}}[k] = (W_{q,0}^{\mathrm{m}}[k], \cdots, W_{q,M-1}^{\mathrm{m}}[k])^t. \tag{5.32}$$

Equivalent to the FDAF approach the update scheme for these K adaptive weight vectors is given by K separate (length M) FDAF update algorithms for $q = 0, \cdots, K-1$:

$$(\underline{\mathbf{W}}_q^{\mathrm{m}}[k+1])^* = (\underline{\mathbf{W}}_q^{\mathrm{m}}[k])^* + 2\alpha(\mathbf{P}^{\mathrm{m}})^{-1}(\underline{\mathbf{X}}_0^{\mathrm{m}}[k-qC])^* r[k] \tag{5.33}$$

with, for stationary signals, the diagonal power matrix $\mathbf{P}^{\mathrm{m}}$ defined as follows:

$$\begin{aligned}
\mathbf{P}^{\mathrm{m}} &= \frac{1}{M} E\{\underline{\mathbf{X}}_0^{\mathrm{m}}[k-qC](\underline{\mathbf{X}}_0^{\mathrm{m}}[k-qC])^h\} \\
&\approx \frac{1}{M} diag(E\{|X_0^{\mathrm{m}}[k]|^2\}, \cdots, E\{|X_{M-1}^{\mathrm{m}}[k]|^2\}) \\
&= diag(P_0^{\mathrm{m}}, \cdots, P_{M-1}^{\mathrm{m}}).
\end{aligned} \tag{5.34}$$

This update scheme is depicted in Fig. 5.7 with the vector $(\underline{P}^m)^{-1}$ containing the diagonal elements of the diagonal matrix $(\mathbf{P}^m)^{-1}$.

The complexity of the mixed–PFDAF structure is roughly given by the following equations:

$$\begin{aligned}
MUL_{mixed-PFDAF} &\approx \frac{1}{2}\left(2M^2 \log(M) + K \cdot 4M\right) \\
DIV_{mixed-PFDAF} &\approx \frac{1}{2} \cdot 4M
\end{aligned} \tag{5.35}$$

with $M = N/K$ and $K = 1, 2, \cdots, N$.

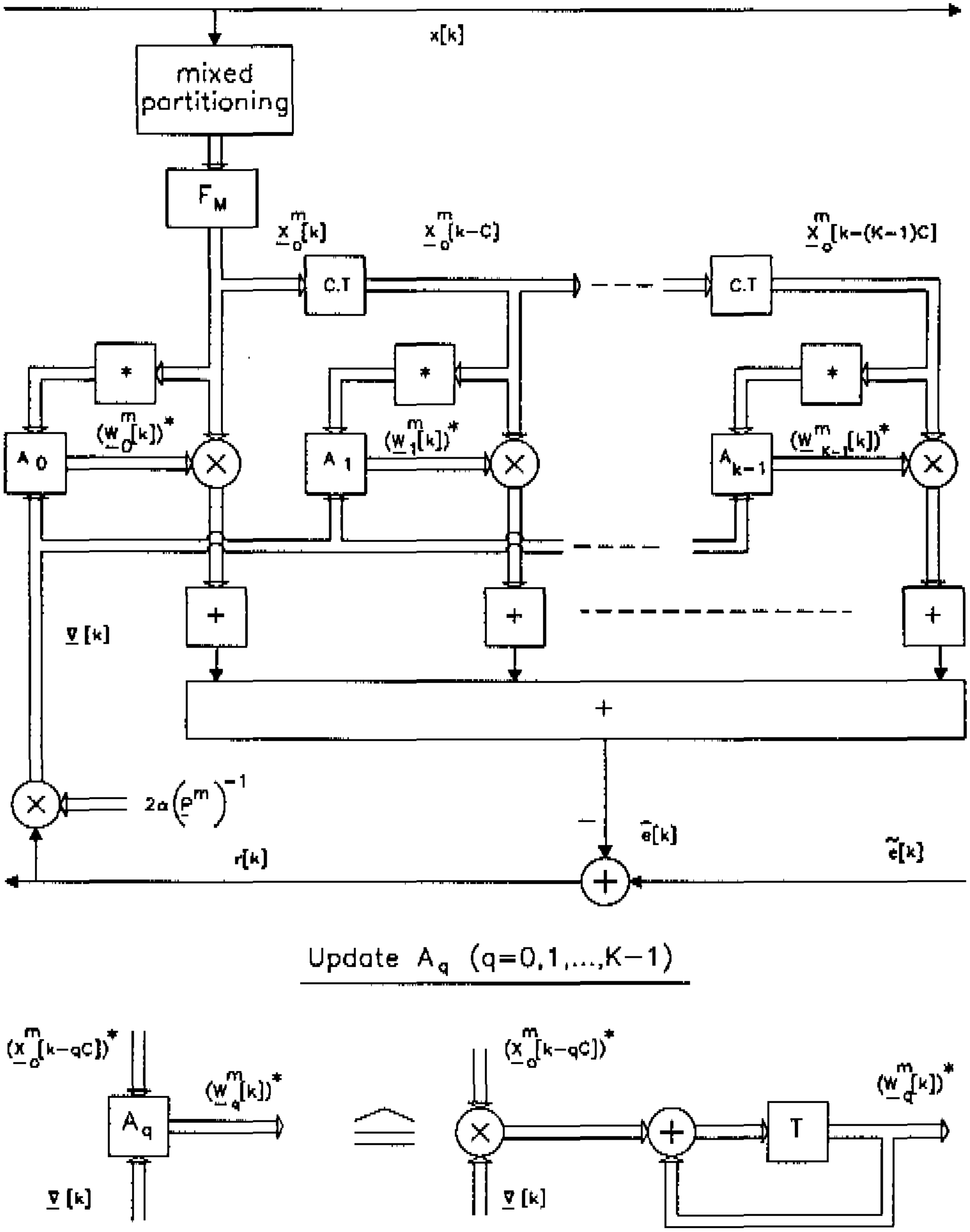

Figure 5.7: *Efficient Mixed-PFDAF with one DFT* ($M = N/K$, $M = I \cdot C$)

5.3.1 Relation with incomplete Decimation–In–Time FFT

From Fig.5.6 it follows that for $I = M$, and thus $C = 1$, the M input samples are chosen in such a way that every following sample is taken by leaving out $(K - 1)$ samples. This corresponds to the comb–partitioning scheme as given in [56]. This comb–PFDAF can be related to an incomplete Decimation In Time (DIT) FFT [37] by looking to the first steps of this DIT–FFT procedure. These steps, to implement an N point DIT-FFT, are:

- Split the input signal in an "even" and an "odd" part.
- Apply FFTs of length $N/2$ to both parts separately.
- Combine the $N/2$ frequency components of both FFTs with an appropriate butterfly stage.

When applying signal vector $\underline{\mathbf{x}}[k]$ to this DIT-FFT the result of the "odd" $N/2$ point FFT equals the result of the "even" $N/2$ point FFT, delayed over T seconds. Thus these two $N/2$ point FFTs can be combined to one $N/2$ point FFT, after which each frequency component is delayed over one delay element of T seconds. Now leaving out the last butterfly stage, as described above, results in an incomplete DIT–FFT that is used in the comb-PFDAF with $K = 2$. Generalizing this concept leads to the comb–PFDAF as given in [56], or equivalently the mixed–PFDAF of Fig. 5.7 with $I = M$.

5.3.2 Relation with incomplete Decimation In Frequency FFT

When chosing $I = 1$, and thus $C = M$, it follows from Fig. 5.6 that M consecutive samples are chosen, which corresponds to the consecutive–PFDAF as given in [56]. This consecutive–PFDAF can be related to an incomplete Decimation In Frequency (DIF) FFT [37] by verifying it with the first steps of the DIF–FFT procedure:

- Split the input signal in a "left" and a "right" part, each containing $N/2$ samples of the input signal.

- Combine the $N/2$ samples of both parts with an appropriate butterfly stage.
- Apply two separate FFTs of length $N/2$.

When leaving out the butterfly stage, and applying the signal vector $\underline{\mathbf{x}}[k]$ to the two $N/2$ point FFTs the result of the "right" $N/2$ point FFT equals the "left" $N/2$ point FFT, delayed over $(N/2)$ delay elements of T seconds. Combining these two FFTs to one FFT of length $N/2$ gives an incomplete DIF–FFT which is used in the consecutive–PFDAF, with $K = 2$. Generalizing this concept leads to the consecutive–PFDAF as given [56], or equivalently the mixed–PFDAF of Fig. 5.7 with $I = 1$.

5.4 Decorrelation conditions for the mixe PFDAF

The filtering part of the mixed–PFDAF structure, depicted at the right hand side of Fig. 5.7, is redrawn in an alternative way in Fig. 5.8. In this figure each frequency component $X_l^{\mathrm{m}}[k]$ is a complex input signal of a transversal filter with delay elements $C \cdot T$ and K adaptive weights. The K adaptive weights, for each separate frequency component $l = 0, 1, \cdots, K-1$, are updated by using a "complex NLMS" algorithm:

$$W_{q,l}^{\mathrm{m}}[k+1] = W_{q,l}^{\mathrm{m}}[k] + \frac{2\alpha}{P_l^{\mathrm{m}}}(X_l^{\mathrm{m}}[k-qC])^* r[k] \tag{5.36}$$

with $N = K \cdot M$ and $q = 0, \cdots, K-1$. The input signal of each separate complex NLMS algorithm can have "correlation in time", that may degrade convergence properties of the mixed–PFDAF structure.

In this section those statistical conditions of the input signal are searched for which the convergence properties of the mixed–PFDAF with DFT length $M = N/K$, are equal to those of the FDAF with DFT length N [58]. First all relevant signals are combined into the following form:

$$\begin{pmatrix} \mathbf{F}_M & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \mathbf{F}_M & 0 \\ 0 & 0 & 0 & \mathbf{F}_M \end{pmatrix} \cdot \begin{pmatrix} \underline{\mathbf{x}}_{K-1}^{\mathrm{m}}[k] \\ \cdots \\ \underline{\mathbf{x}}_1^{\mathrm{m}}[k] \\ \underline{\mathbf{x}}_0^{\mathrm{m}}[k] \end{pmatrix} = \begin{pmatrix} \underline{\mathbf{X}}_{K-1}^{\mathrm{m}}[k] \\ \cdots \\ \underline{\mathbf{X}}_1^{\mathrm{m}}[k] \\ \underline{\mathbf{X}}_0^{\mathrm{m}}[k] \end{pmatrix} \tag{5.37}$$

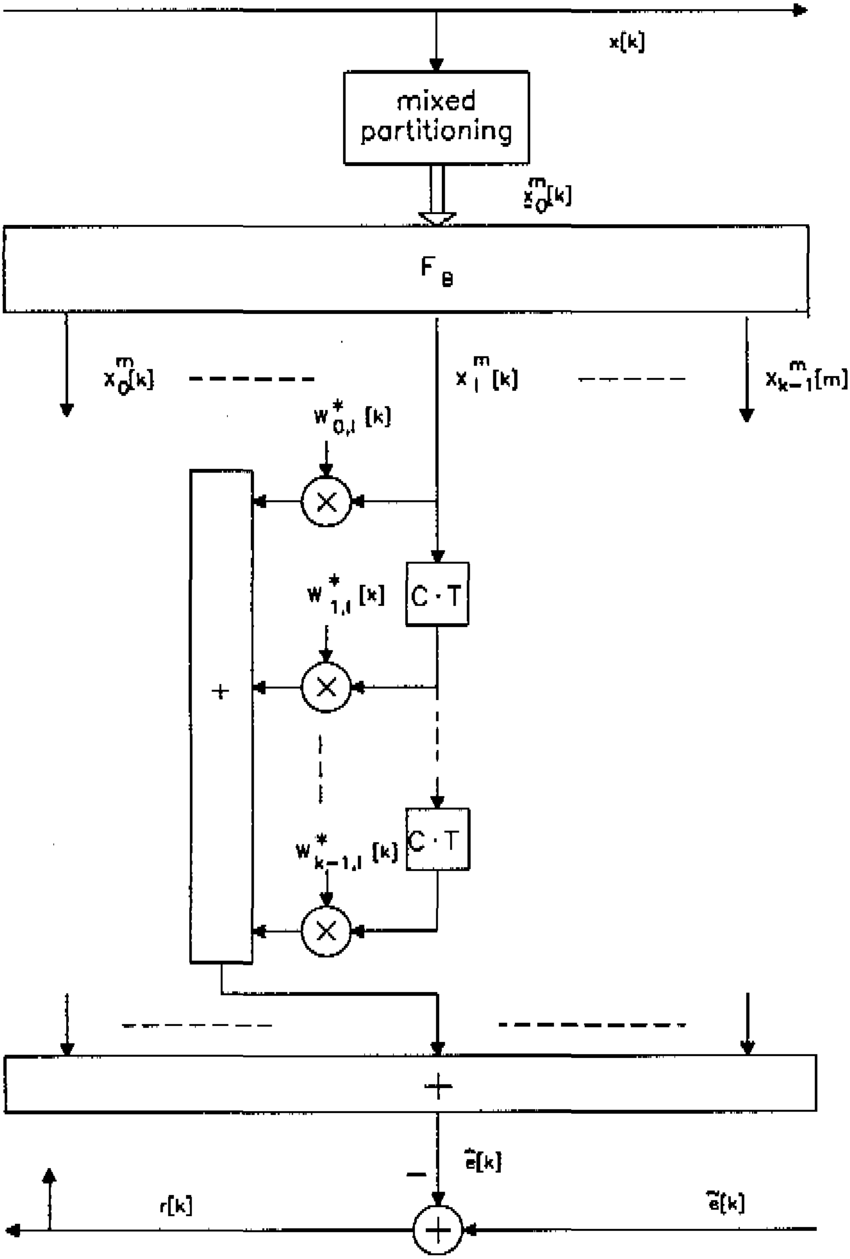

Figure 5.8: *Alternative figure for filtering part of mixed–PFDAF*

or in compact notation:

$$\mathbf{F}^{m} \cdot \underline{\mathbf{x}}^{\mathrm{m}}[k] = \underline{\mathbf{X}}^{\mathrm{m}}[k]. \tag{5.38}$$

All relevant correlations can be calculated by the expression

$$\begin{aligned} E\{\underline{\mathbf{X}}^{\mathrm{m}}[k](\underline{\mathbf{X}}^{\mathrm{m}}[k])^{h}\} &= \mathbf{F}^{m} \cdot E\{\underline{\mathbf{x}}^{\mathrm{m}}[k](\underline{\mathbf{x}}^{\mathrm{m}}[k])^{t}\} \cdot (\mathbf{F}^{m})^{h} \\ &= \mathbf{F}^{m} \cdot \mathbf{R}^{\mathrm{m}} \cdot (\mathbf{F}^{m})^{h}. \end{aligned} \tag{5.39}$$

As mentioned in Chapter 4 an input signal can be decorrelated in frequency domain by simple power normalization if the decorrelation conditions are satisfied. One of these restrictions was, in order to approximate the Toeplitz autocorrelation matrix by a circular one, that the length of the autocorrelation function is limitted by τ_{max} with $|\tau_{\mathrm{max}}| < M/2$. Using this property it follows from the above equations that if the "mixed" autocorrelation matrix

$$\mathbf{R}^{\mathrm{m}} = E\{\underline{\mathbf{x}}^{\mathrm{m}}[k](\underline{\mathbf{x}}^{\mathrm{m}}[k])^{t}\} \tag{5.40}$$

is block–diagonal, with only autocorrelation function values up to $M/2$ elements in the main diagonal blocks, then the convergence properties of the mixed–PFDAF are "equivalent" to the FDAF. Note that that this equivalence is not exact since the Toeplitz–circulant approximation is carried out for different matrix dimensions. In order to fulfill this restriction it follows from equation (5.39) that the autocorrelation function $\rho[\tau] = E\{x[k]x[k-\tau]\}$ of the input signal may have values unequal to zero, or undefined, for $\tau = \lambda N/I$ with $\lambda = 0, \pm 1, \cdots, (I-1)/2$. For all other values of τ this autocorrelation function must vanish. On the other hand one of the decorrelation conditions for the FDAF was that $\tau_{\mathrm{max}} < N/2$. Thus the mixed–PFDAF, with DFT length $M = N/K$, has equivalent convergence properties as the FDAF, with DFT length N, when the autocorrelation function of the input signal has the properties

$$\rho[\tau] = \begin{cases} \text{arbitrary} & \text{for } \tau = 0, \pm\frac{N}{I}, \pm 2 \cdot \frac{N}{I} \cdots, \pm\frac{I-1}{2} \cdot \frac{N}{I} \\ 0 & \text{elsewhere} \end{cases} \tag{5.41}$$

with $I = 1, 2, \cdots, M$. The interpretation of equation (5.41) is as follows:

Assume the autocorrelation function $\rho'[\tau]$ with $I/2$ degrees of freedom is defined as follows:

$$\rho'[\tau] = \begin{cases} \text{arbitrary} & |\tau| < \frac{I-1}{2} \\ 0 & \text{elsewhere} \end{cases} \tag{5.42}$$

then the autocorrelation function $\rho[\tau]$ of the input signal $x[k]$ equals this function $\rho'[\tau]$ interleaved with $(N/I) - 1$ zeros. Thus the power spectral density function

$$P(e^{j\theta}) = \sum_{\tau=-\infty}^{\infty} \rho[\tau] e^{-j\theta\tau} \tag{5.43}$$

contains N/I mirror images.
The interpretation for the comb–PFDAF and consecutive–PFDAF is as follows:

Comb-PFDAF ($I = M$):
The comb–PFDAF has equal convergence properties as the FDAF if the input signal has an autocorrelation with the properties

$$\rho[\tau] = \begin{cases} \text{arbitrary} & \text{for } \tau = 0, \pm K, \cdots, \pm(M-1) \cdot K \\ 0 & \text{elsewhere} \end{cases} \tag{5.44}$$

The spectrum of the input signal contains K mirror images. The complete decorrelation is performed by using the information of one mirror image. This is done by applying a DFT of length M on the comb partitioned input signal.

□

Consecutive-PFDAF ($I = 1$):
Equation (5.41) shows that the consecutive–PFDAF has the same convergence properties as the FDAF if the input signal has one undefined autocorrelation value $\rho[0]$. Thus, as already stated in Section 5.2, consecutive partitioning of the input signal will always degrade convergence properties of the adaptive filter in comparison to FDAF, unless a white

noise input signal is applied to the adaptive filter.
On the other hand when $\rho[\tau]$ has up to M degrees of freedom as:

$$\rho[\tau] = \begin{cases} \text{arbitrary} & |\tau| < M/2 \\ 0 & \text{elsewhere} \end{cases} \tag{5.45}$$

then the power spectral density function of such a signal is a "smooth" function. For such a signal a possible solution is not actual to apply any partitioning scheme at all, but simply using an FDAF, with transformation length N, in which every K consecutive components are normalized with the same power function. Note, however, that this approach does not reduce complexity: Still N divisions and a DFT of length N are needed.

□

Similary, if the autocorrelation function of the input signal does not fulfil the restrictions given in equation (5.41) more and more block-diagonal terms of equation (5.39) will influence the convergence properties of the adaptive filter. Normalizing the mixed–partitioned frequency components by P_l^{in} for $l = 0, 1, \cdots, M-1$ is not enough any more to decorrelate the input signal completely. This also follows directly from the DIT– or DIF–FFT point of view: Too many stages are left out!

5.5 Experiments

Results of the previous section are verified here with some experiments. For the first two experiments the system as given in Fig. 1.7 is used, of which the "unknown" Wiener system has an exponential decaying impulse response of length $N = 64$. The adaptive weights are initialized with zeros, and the white noise signal $s[k]$ is such that the quantity $10\log(\tilde{J}[kL])$ starts with 20 dB. The partition factor K is varied from $K = 1$ (PBFDAF = BFDAF) to $K = N = 64$ (PBFDAF = BNLMS). The processing delay was choosen to $L = M + 1$, with the partitions length $M = N/K$. With these parameters the length of the DFT equals $B = 2M$. Fig. 5.9 shows the results for an ar(1) signal, with parameter

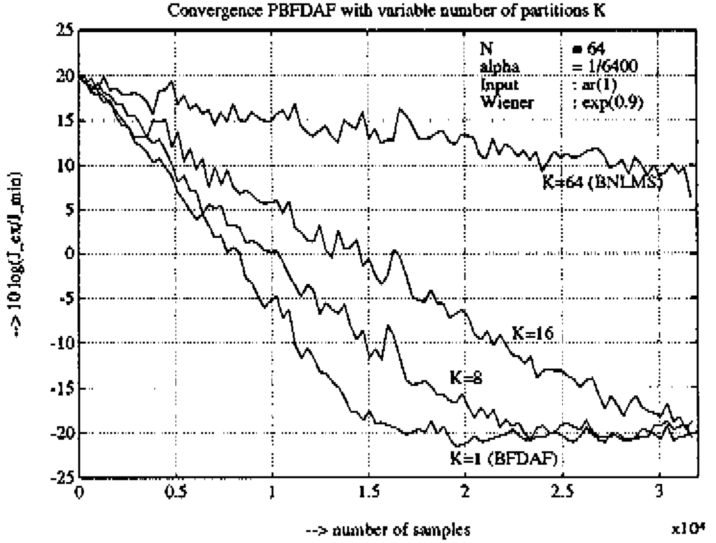

Figure 5.9: *PBFDAF with $M = N/K$ and DFT length $B = 2M$, ar(1) input*

$a = -0.8182$, and Fig. 5.10 gives the results for an ma(1) signal, with parameter $a = -0.8182$. From these experiments it follows that, with $N = 64$, the PBFDAF is capable to decorrelate an ar(1) input signal reasonably well in the range $K = 1, 2 \cdots, 8$. For an ma(1) signal this range is $K = 1, 2, 3, 4$.

With the last experiment of this chapter the results of the mixed–PFDAF will be verified. For this an "unknown" Wiener system is chosen with $N = 32$ coeffiecients. The adaptive weights are initialized with zeros, and the signal $s[k] = 0$. Thus the quantity of interest is:

$$10\log\left(\frac{E\{(e[k] - \hat{e}[k])^2\}}{E\{e^2[k]\}}\right). \tag{5.46}$$

The input signal is generated by an ar(4) model, that is defined as:

$$x[k] = (\sqrt{1 - a^2}) \cdot n[k] + a \cdot x[k-4] \tag{5.47}$$

with $n[k]$ a white noise signal, having average zero and $E\{n^2[k]\} = 1/3$, and $a = 0.8182$. The adaptation constant is $\alpha = 4.6/1000$. The results

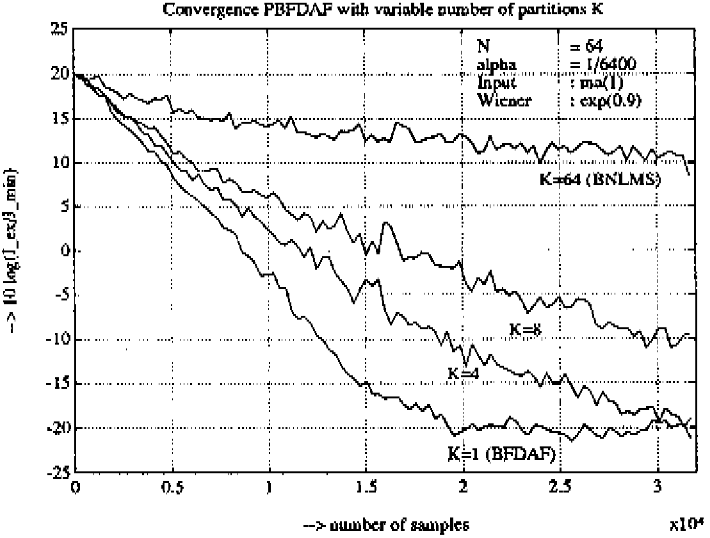

Figure 5.10: *PBFDAF with $M = N/K$ and DFT length $B = 2M$, ma(1) input*

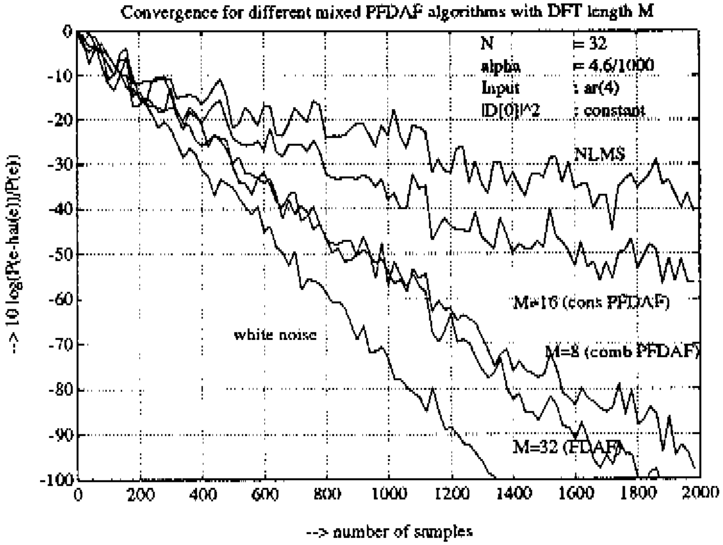

Figure 5.11: *Experimental results of mixed-PFDAF*

of different experiments are plotted in Fig. 5.11. Two reference curves are plotted: the ideal decorrelation case (white noise) and the situation where no decorrelation takes place (NLMS). After that the result of the FDAF algorithm ($K = 1$) is plotted.

First the FDAF is simulated using the mixed–PFDAF structure with DFT length M equal to the adaptive filter length $M = N = 32$. For this situation no partitioning takes place. As discussed in Chapter 4, the resulting curve slightly deviates from the ideal curve (white noise).

Furthermore it is shown in the figure that the given ar(4) signal can be decorrelated by using a mixed–PFDAF with $C = 1$ (comb–partitioning) and $K = 4$. The DFT used has length $M = N/4 = 8$.

Finally it is shown in the figure that the consecutive way of partitioning degrades convergence already for $K > 1$. This is simulated with the consecutive partitioning for $M = N/2 = 16$.

5.6 Discussion

In this chapter some techniques are presented to decouple the spectral resolution and the filter length N. These techniques are carried out by partitioning the impulse response, and by that the update algorithm, in separate parts. Implementing this in an efficient way for the block processing approach leads to the (consecutive) PBFDAF structure, in which the impulse response is separated in K consecutive parts of length $M = N/K$. In general convergence properties will degrade when using more and more partitions. It is shown by experimental results that for small K this degradation is minimal. On the other hand it is shown that, for a practical example, complexity of the PBFDAF structure is less than the complexity needed to implement the BFDAF structure.

For the "sliding" FDAF a new mixed–PFDAF was introduced from which the transformation length of the DFT is a factor K smaller than that of the DFT used in the FDAF structure. Furthermore it was shown that with one extra parameter $I = M/C$ this structure describes both the comb–PFDAF ($I = M$) and consecutive–PFDAF ($I = 1$) as given in [56]. The comb– and consecutive–structures are respectively equivalent to an incomplete Decimation In Time or Decimation In Frequency method as applied to FFT schemes. Moreover it is shown that the mixed–PFDAF with $K = N/M$ and $I = M/C$ has the same convergence properties as FDAF if statistical properties of input signal, given by the autocorrelation function $\rho[\tau]$, has I degrees of freedom according to equation (5.41). The power spectral density function of such a signal contains N/I mirror images. This implies that the comb–PFDAF $(I = M)$ can have up to M degrees of freedom in the autocorrelation function (5.44) with equal convergence properties as the FDAF. The consecutive–PFDAF on the other hand can only have one degree of freedom: For an input signal with M degrees of freedom with autocorrelation function as given in equation (5.45): consecutive partitioning will always degrade convergence properties.

Finally some comments for future research are given:

- The given mixed structure is such that each separate frequency component has a transversal filter structure, from which the adaptive weights are updated according to a "complex" NLMS algorithm. Research can be done to investigate the possibilities to choose both structure and algorithm for each separate frequency component in more agreement with the requirements of that component.
- In this chapter techniques are described to <u>decrease</u> the spectral resolution of the adaptive filter from N to $M = N/K$ components.

  In Chapter 4 it is shown that frequency domain adaptive filters require two decorrelation conditions. If the length of the Fourier transform however is such that these conditions are not satisfied, it may be possible to search for methods to <u>increase</u> the DFT length (at the cost of complexity).

Chapter 6

Time domain Adaptive filters

In this chapter some time domain techniques are given that can decorrelate an input signal of an adaptive filter with N weights, by using an $L \times L$ autocorrelation matrix ($L \geq 1$).

In Section 6.1 first a short overview is given of some well known techniques that use an $N \times N$ autocorrelation matrix. As an introduction to techniques used later in this chapter, Section 6.2 describes a geometrical interpretation of an NLMS algorithm. This method uses a projection of the difference vector on a one–dimensional space, representing all available information in the adaptive filter. This method is called the Orthogonal Projection (OP) method. Using more information from the past, this geometrical interpretation is generalized in Section 6.3, where a projection is made of the difference vector on a L dimensional hyperplane, representing the available information. In order to reduce complexity, this algorithm is performed on block bases, making one update every L samples. This results in the Block Orthogonal Projection (BOP) algorithm. It is shown that this method decorrelates the input signal with an $L \times L$ autocorrelation matrix. With the dimension $L \geq 1$ this method offers more freedom in tuning this dimension to the requirements of the input signal. Another possibility is discussed in Section 6.4. In this section the L–dimensional hyperplane is orthogonalized by using the Gram–Schmidt procedure before applying to the

BOP method. This results in the Gram–Schmidt Block Orthogonal Projection (GS-BOP) algorithm. In Section 6.5 it is shown that for signals generated with an auto regressive model of order p (ar(p)) this Gram–Schmidt approach can be implemented very efficiently resulting in the Efficient Orthogonal Projection (EOP) algorithm. This EOP algorithm can decorrelate ar(p) input signals with roughly the same amount of complexity as the NLMS algorithm. All theoretical results are verified by experiments in Section 6.6 and this chapter is concluded in Section 6.7 with a discussion.

6.1 Decorrelation in time domain with $N \times N$ autocorrelation matrix

Although not a subject of research in this thesis, this section gives a short overview of some well known techniques that decorrelate an input signal with an $N \times N$ autocorrelation matrix.

6.1.1 LMS/Newton

In this section the assumption is made that the autocorrelation matrix $\mathbf{R} = E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]\}$, or an estimate of it, as well as the inverse $\mathbf{R}^{-1}$, are completely known. It is shown how $\mathbf{R}^{-1}$ can be used in order to make convergence properties of the adaptive algorithm independent of statistical properties from signal $x[k]$.

Premultiplying only the update part in the LMS algorithm (2.8) by the inverse autocorrelation matrix $\mathbf{R}^{-1}$, results in the LMS/Newton algorithm [68]:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\mathbf{R}^{-1}\underline{\mathbf{x}}[k]r[k]. \tag{6.1}$$

This algorithm equals the NLMS algorithm when the input signal is a white noise signal with variance σ_x^2. In this case the autocorrelation matrix is simply diagonal given by $\mathbf{R} = \sigma_x^2\mathbf{I}$. Note that in literature [68] the LMS/Newton algorithm is also known in such a form that adapts to the optimum in one step solution. This equals the above algorithm by choosing $\alpha = 1/2$. As in previous chapters two steps are studied: First the convergence behaviour of the average weight vector is investigated and second an expression is derived for the quantity $\tilde{J}[k] = J_{\text{ex}}/\sigma_s^2$.

Calculating the average of both sides from (6.1) and using $r[k] = \underline{\mathbf{x}}^t[k](\underline{\mathbf{w}}_{\text{opt}} - \underline{\mathbf{w}}[k]) + s[k]$ and assuming $E\{s[k]\} = 0$ gives:

$$\begin{aligned} E\{\underline{\mathbf{w}}[k+1]\} &\approx E\{\underline{\mathbf{w}}[k]\} + 2\alpha\mathbf{R}^{-1}E\{\underline{\mathbf{x}}[k]\underline{\mathbf{x}}^T[k]\}E\{\underline{\mathbf{d}}[k]\} \\ &= 2\alpha\underline{\mathbf{w}}_{\text{opt}} + (1-2\alpha)E\{\underline{\mathbf{w}}[k]\} \\ &= 2\alpha\sum_{i=0}^{k}(1-2\alpha)^i\underline{\mathbf{w}}_{\text{opt}} + (1-2\alpha)^{k+1}\underline{\mathbf{w}}[0]. \end{aligned} \tag{6.2}$$

Thus for α within the convergence area $(0 < \alpha < 1)$ the LMS/Newton algorithm converges, in average, to the optimum Wiener solution:

$$\lim_{k\to\infty} E\{\underline{\mathbf{w}}[k]\} = \underline{\mathbf{w}}_{\text{opt}}. \tag{6.3}$$

For the onvergence behaviour of the LMS/Newton algorithm of the second order statistics, first the updating equation can be rewritten for the difference vector $\underline{\mathbf{d}}[k] = \underline{\mathbf{w}}_{\text{opt}} - \underline{\mathbf{w}}[k]$ as follows:

$$\underline{\mathbf{d}}[k] = (\mathbf{I} - 2\alpha\mathbf{R}^{-1}\underline{\mathbf{x}}[k-1]\underline{\mathbf{x}}^t[k-1])\underline{\mathbf{d}}[k-1] - 2\alpha\mathbf{R}^{-1}\underline{\mathbf{x}}[k-1]s[k-1]. \tag{6.4}$$

This difference vector is substituted in the equation

$$\tilde{J}[k] = \frac{J_{\text{ex}}[k]}{J_{\text{min}}} = \frac{E\{\underline{\mathbf{d}}^t[k]\mathbf{R}\underline{\mathbf{d}}[k]\}}{\sigma_s^2} \tag{6.5}$$

and results in the following recursive relation for the dynamic behavior:

$$\tilde{J}[k] == (1-4\alpha)\tilde{J}[k-1] + 4\alpha^2 N. \tag{6.6}$$

These results are clearly independent of statistical properties of the input signal $x[k]$.

6.1.2 Recursive Least Square Algorithm

The problem with the LMS/Newton algorithm is that exact knowledge is required of the matrix $\mathbf{R}^{-1}$. This matrix is in general not known a priori and moreover it may slowly change in time.

The family of methods known as (Recursive) Least Square ((R)LS) algorithms belong also to the class of techniques that are theoretically less sensitive to the statistical properties of the input signal. The main difference between gradient–based methods and RLS techniques is that the RLS algorithms minimize an exact error criterion constructed from the actual data in contrast with the statistical error criterion for the LMS. Clearly, this exact optimization for every point in time implies quite a sophisticated processing algorithm. The traditional Least–Squares approach to compute an N^{th} order adaptive filter would require in the order of N^3 arithmetic operations per time update, largely due to the computation of the $N \times N$ matrix inverse.

Instead of minimizing the mean square error, as in the LMS case, the Least Squares (LS) approach minimizes an exponential weighted sum of squared errors (the LS–cost function) usually defined as [3]:

$$\varepsilon[k] \doteq \sum_{i=0}^{k} \lambda^{k-i}(\tilde{e}[i] - \underline{\mathbf{w}}^t[k] \cdot \underline{\mathbf{x}}[i])^2. \tag{6.7}$$

Thus the present adaptive weight vector $\underline{\mathbf{w}}[k]$ is convolved with previous input signal vectors $\underline{\mathbf{x}}[i]$ and compared with previous reference signals $\tilde{e}[i]$. The result is exponentially weighted with a window from which the effective length is approximately equal to $\frac{1}{1-\lambda}$ samples. A value of $\lambda = 1$ signifies that all data is equally weighted, and this case is often referred to in literature as the prewindowed case. Another choice would be to weight with a window of finite and equal length for all k. This choice is sometimes called a sliding window weighting. It says that only the most recent samples are used to do the estimation and these samples are weighted equally. Both this sliding window and the exponential weighting methods can be used to handle slow time variations in the unknown Wiener system. Depending on the type of time variation, one scheme may be better than the other. For simplicity here only the exponential weighting is considered, because it can be realized in a recursive way.

Now a short work through the mathematics. Minimization of the LS–cost function leads to the following optimum adaptive weight vector at time instant k:

$$\underline{\mathbf{w}}[k] = \hat{\mathbf{R}}^{-1}[k]\underline{\mathbf{c}}[k] \tag{6.8}$$

with:

$$\begin{aligned}\hat{\mathbf{R}}[k] &= \sum_{i=0}^{k} \lambda^{k-i} \underline{\mathbf{x}}[i]\underline{\mathbf{x}}^t[i] \\ \underline{\mathbf{c}}[k] &= \sum_{i=0}^{k} \lambda^{k-i} \tilde{e}[i]\underline{\mathbf{x}}[i]. \end{aligned} \tag{6.9}$$

Direct solution of these equations should require in the order of N^3 operations. In order to reduce this computational effort, recursive solutions are used leading to the Recursive Least Square (RLS) algorithm. The first step is to compute $\hat{\mathbf{R}}[k]$ and $\underline{\mathbf{c}}[k]$ recursively according to the next equations:

$$\begin{aligned}\hat{\mathbf{R}}[k] &= \lambda\hat{\mathbf{R}}[k-1] + \underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k] \\ \underline{\mathbf{c}}[k] &= \lambda\underline{\mathbf{c}}[k-1] + \underline{\mathbf{x}}[k]\tilde{e}[k]. \end{aligned} \tag{6.10}$$

Since equation (6.8) needs the inverse autocorrelation matrix, the second step is to compute $\hat{\mathbf{R}}^{-1}[k]$ recursively by using a well known matrix inversion lemma [26] (p385). This results in the following equation:

$$\hat{\mathbf{R}}^{-1}[k] = \frac{1}{\lambda}\left(\hat{\mathbf{R}}^{-1}[k-1] - \underline{\mathbf{g}}[k]\underline{\mathbf{x}}^t[k]\hat{\mathbf{R}}^{-1}[k-1]\right) \tag{6.11}$$

with the gain vector $\underline{\mathbf{g}}[k]$ defined as:

$$\underline{\mathbf{g}}[k] = \frac{1}{\lambda + \underline{\mathbf{x}}^t[k]\hat{\mathbf{R}}^{-1}[k-1]\underline{\mathbf{x}}[k]} \cdot \hat{\mathbf{R}}^{-1}[k-1]\underline{\mathbf{x}}[k]. \tag{6.12}$$

By multiplying both sides of equation (6.11) with the vector $\underline{\mathbf{x}}[k]$ it follows that:

$$\hat{\mathbf{R}}^{-1}[k]\underline{\mathbf{x}}[k] = \underline{\mathbf{g}}[k]. \tag{6.13}$$

Thus the gain vector is a transformed vector obtained by rotating the input signal vector.

The RLS updating equation for the adaptive weights can now be derived as follows:
Filling in equation (6.10) in (6.8) and using (6.13) gives:

$$\underline{\mathbf{w}}[k] = \lambda\hat{\mathbf{R}}^{-1}[k-1]\underline{\mathbf{c}}[k-1] + \underline{\mathbf{g}}[k]\tilde{e}[k]. \tag{6.14}$$

Using expression (6.11) for the quantity $\lambda\hat{\mathbf{R}}^{-1}[k-1]$ results in:

$$\begin{aligned}\underline{\mathbf{w}}[k] &= \underline{\mathbf{w}}[k-1] - \underline{\mathbf{g}}[k]\underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k-1] + \underline{\mathbf{g}}[k]\tilde{e}[k] \\ &= \underline{\mathbf{w}}[k-1] + \underline{\mathbf{g}}[k]r[k]. \end{aligned} \tag{6.15}$$

Thus after initialization every iteration consists of the following steps:

(a) Calculate the residual signal $r[k] = \tilde{e}[k] - \underline{\mathbf{x}}^t[k]\underline{\mathbf{w}}[k-1]$.

(b) Calculate the gain vector according to equation (6.12)

(c) Calculate the new weight vector according to (6.15)

(d) Calculate the new inverse autocorrelation matrix $\hat{\mathbf{R}}^{-1}[k]$ according equation (6.11).

Comparing the LMS update algorithm (2.8) with the above RLS equations (6.15) shows that in the LMS case the gain vector $\underline{\mathbf{g}}[k] = 2\alpha\underline{\mathbf{x}}[k]$. This implies that the input signal vector is not transformed or rotated. The complexity of the above RLS update equations is in the order of N^2.

In the past decade, new algorithms have been derived that further reduce the number of required multiplications plus divisions per iteration such that they become linear in N. Basically, they capitalize on a property of the autocorrelation matrix $\mathbf{R}$ that is not exploited in the previous methods, namely that the vector $\underline{\mathbf{x}}[k]$ acts like a shift register such that $\underline{\mathbf{x}}[k+1]$ is only a "push up" version of $\underline{\mathbf{x}}[k]$ with a new sample at the bottom. This new algorithm can be broadly classified into two categories by their approaches to solution. One is the Fast Kalman type or the fixed-order recursive least–squares algorithms, and the other is the ladder type algorithms. Loosely speaking, the fast Kalman algorithms belong to the framework given in the above RLS update equations, and use more efficient methods to update the gain vector $\underline{\mathbf{g}}[k]$ at each iteration. The ladder algorithm resorts to another formulation of $\underline{\mathbf{w}}[k]$ and finds simpler ways of updating it. A tutorial of fast RLS is given in [2].

In conclusion it follows that the LMS/Newton can perfectly decorrelate a correlated input signal, when the exact autocorrelation matrix, and its inverse, are known a priori. The complexity of the algorithm is very large. The recursive schemes are less complex and have good decorrelation properties too. Still this complexity is too high for many practical applications. Fast RLS schemes with low complexity and good decorrelation properties have been developed in the last years.

6.2 Geometric interpretation of the NLMS algorithm

In this section a geometric interpretation is given of the signal estimation problem of Fig. 1.7. For simplicity reasons it is assumed, in first instance, that the signal $s[k]$ is zero.
The update procedure is as follows (see also Fig. 6.1):

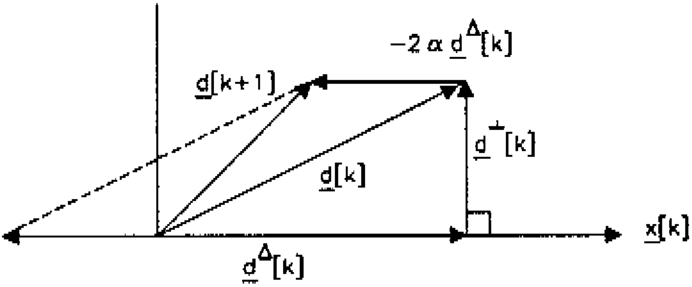
$-2\alpha\underline{d}^{\Delta}[k]$
$\underline{d}[k+1]$
$\underline{d}^{\perp}[k]$
$\underline{d}[k]$
$\underline{x}[k]$
$\underline{d}^{\Delta}[k]$

Figure 6.1: *Geometric interpretation of the NLMS algorithm*

Make a projection of the difference vector $\underline{\mathbf{d}}[k]$ on the available data which is present in the vector $\underline{\mathbf{x}}[k]$. By doing so $\underline{\mathbf{d}}[k]$ is decomposed as:

$$\underline{\mathbf{d}}[k] = \underline{\mathbf{d}}^{\perp}[k] + \underline{\mathbf{d}}^{\Delta}[k] \qquad (6.16)$$

with $\underline{\mathbf{d}}^{\perp}[k]$ orthogonal and $\underline{\mathbf{d}}^{\Delta}[k]$ parallel to $\underline{\mathbf{x}}[k]$. This implies that:

$$\begin{aligned} < \underline{\mathbf{x}}[k], \underline{\mathbf{d}}^{\perp}[k] > &= 0 \\ \underline{\mathbf{d}}^{\Delta}[k] &= c \cdot \underline{\mathbf{x}}[k] \end{aligned} \qquad (6.17)$$

with c some scalar. With $s[k] = 0$ it follows from Fig. 1.7 that the residual signal $r[k]$ can be written as:

$$\begin{aligned} r[k] &= \underline{\mathbf{x}}^t[k]\underline{\mathbf{d}}[k] = < \underline{\mathbf{x}}[k], \underline{\mathbf{d}}[k] > = \\ &= < \underline{\mathbf{x}}[k], \underline{\mathbf{d}}^{\Delta}[k] > = c||\underline{\mathbf{x}}[k]||^2. \end{aligned} \qquad (6.18)$$

From this equation the parameter c can be calculated since both the signal $r[k]$ and the quantity $||\underline{\mathbf{x}}[k]||^2$ are known. With this the parallel component $\underline{\mathbf{d}}^{\Delta}[k]$ can be written as:

$$\underline{\mathbf{d}}^{\Delta}[k] = c \cdot \underline{\mathbf{x}}[k] = \frac{r[k]}{||\underline{\mathbf{x}}[k]||^2} \cdot \underline{\mathbf{x}}[k]. \tag{6.19}$$

The main purpose of the adaptive algorithm is to both reduce the length and rotate vector $\underline{\mathbf{d}}[k]$ in such a way that it becomes "more orthogonal" to $\underline{\mathbf{x}}[k]$. This can be achieved by subtracting a small part of the vector $\underline{\mathbf{d}}^{\Delta}[k]$ from the vector $\underline{\mathbf{d}}[k]$, as shown in Fig.6.1. Thus:

$$\underline{\mathbf{d}}[k+1] = \underline{\mathbf{d}}[k] - 2\alpha\underline{\mathbf{d}}^{\Delta}[k] = \underline{\mathbf{d}}[k] - 2\alpha\frac{\underline{\mathbf{x}}[k]r[k]}{||\underline{\mathbf{x}}[k]||^2} \tag{6.20}$$

which leads, with the definition of the difference vector, to the following update equation:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\frac{\underline{\mathbf{x}}[k]r[k]}{||\underline{\mathbf{x}}[k]||^2}. \tag{6.21}$$

Comparing this algorithm with the NLMS algorithms shows that both are very similar. The above algorithm is the deterministic interpretation of the (stochastic) NLMS algorithm. Heuristically this geometric approach can be explained as follows:
First assume orthogonal vectors $\underline{\mathbf{x}}[k]$ and $\underline{\mathbf{x}}[k+i]$ $\forall i$ (the geometrical interpretation of a stochastic white noise process). For these orthogonal vectors the update algorithm rotates the vector $\underline{\mathbf{d}}[k]$ in such a way that the inner product $<\underline{\mathbf{d}}[k+1], \underline{\mathbf{x}}[k]>$, which is a measure for the final misadjustment, becomes smaller in comparison to the previous inner product. This reduction can be accomplished in two ways: by reducing the length of $\underline{\mathbf{d}}[k]$ and/or by rotating $\underline{\mathbf{d}}[k]$ in such a way that it becomes more orthogonal to the vector $\underline{\mathbf{x}}[k]$. The above algorithm achieves both. For example when choosing a large adaptation constant $\alpha = 1/2$ the algorithm rotates $\underline{\mathbf{d}}[k]$ in one step orthogonal to $\underline{\mathbf{x}}[k]$. This implies that the inner product becomes zero, while the length of $\underline{\mathbf{d}}[k+1]$ needs not to be zero. When however the "noisy" signal $s[k]$ is present, the inner product can have large fluctuations since the vector $\underline{\mathbf{d}}[k]$ can fluctuate around the orthogonal position. This can result in a large

final misadjustment. On the other hand choosing a small adaptation constant α leads to a very slow algorithm, whith the final difference vector both small and orthogonal to the input signal vector, resulting in a small final misadjustment.

When the input signal vectors are not orthogonal, representing a coloured process, the new update direction is not orthogonal to the previous update directions. An update made in one iteration can both be helped or counteracted in the next iteration: Convergence properties of the above algorithm are sensitive to the geometric distribution of the input signal vectors. Or in stochastic terms: Convergence properties of the NLMS algorithm are dependent of the input signal statistics, as was shown in in Chapter 2.

6.3 L–step Orthogonal Projection Algorithm

In [39,16] an L–step Orthogonal Projection (OP) method is introduced, that extends the geometric approach of the previous section to L dimensions. With this OP method a projection is made on an L dimensional hyperplane rather than on a line. In general there are two variants to implement this idea:

(a) The "sliding" procedure that uses every iteration only one new input signal sample. This is the L-step Orthogonal Projection method (OP) as described in [39].

(b) The "block" approach that uses L new samples and performs only one update of the adaptive weight vector every L samples. This is the Block Orthogonal Projection (BOP) algorithm [55]. Since the latter algorithm can be implemented more efficiently it is described here.

The procedure of the BOP method is as follows: make a projection of $\underline{\mathbf{d}}[kL]$ on an L dimensional plane, spanned by the L vectors $\underline{\mathbf{x}}[kL], \cdots, \underline{\mathbf{x}}[kL-L+1]$. Thus:

$$\underline{\mathbf{d}}[kL] = \underline{\mathbf{d}}^{\Delta}[kL] + \underline{\mathbf{d}}^{\perp}[kL] \tag{6.22}$$

with

$$
\begin{aligned}
< \underline{\mathbf{d}}^{\perp}[kL], \underline{\mathbf{x}}[kL-i] > &= 0 \quad \text{for all } i = 0, \cdots L-1 && (6.23)\\
\underline{\mathbf{d}}^{\Delta}[kL] &= \sum_{i=0}^{L-1} c_i[kL]\underline{\mathbf{x}}[kL-i] = \chi[kL]\underline{\mathbf{c}}[kL] && (6.24)
\end{aligned}
$$

with the $N \times L$ matrix $\chi[kL]$ and the L dimensional vector $\underline{\mathbf{c}}[kL]$ defined as:

$$
\begin{aligned}
\chi[kL] &= (\underline{\mathbf{x}}[kL-L+1], \cdots, \underline{\mathbf{x}}[kL-1], \underline{\mathbf{x}}[kL])\\
\underline{\mathbf{c}}[kL] &= (c_{L-1}[kL], \cdots, c_1[kL], c_0[kL])^t. && (6.25)
\end{aligned}
$$

With the assumption (in first instance) that signal $s[k] = 0$, the L dimensional residual signal vector $\underline{\mathbf{r}}_L[kL]$ can be written as:

$$\underline{\mathbf{r}}_L[kL] = \chi^t[kL]\underline{\mathbf{d}}[kL]. \tag{6.26}$$

Now an equivalent procedure as in the previous section can be used to calculate the coefficient vector $\underline{\mathbf{c}}[kL]$, by using the fact that $\underline{\mathbf{d}}^{\perp}[kL]$ is perpendicular to all vectors $\underline{\mathbf{x}}[kL-i]$ for $i = 0, 1, \cdots, L-1$. In vector matrix notation this leads to:

$$
\begin{aligned}
\underline{\mathbf{r}}_L[kL] &= \chi^t[kL]\underline{\mathbf{d}}[kL] = \chi^t[kL]\left(\underline{\mathbf{d}}^{\Delta}[kL] + \underline{\mathbf{d}}^{\perp}[kL]\right)\\
&= \chi^t[kL]\underline{\mathbf{d}}^{\Delta}[kL] = \chi^t[kL]\chi[kL]\underline{\mathbf{c}}[kL]. && (6.27)
\end{aligned}
$$

The solution of these equations is given by:

$$\underline{\mathbf{c}}[kL] = \hat{\mathbf{R}}^{-1}[kL]\underline{\mathbf{r}}[kL] \tag{6.28}$$

with the $L \times L$ "autocorrelation matrix" is defined as:

$$\hat{\mathbf{R}}[kL] = \chi^t[kL]\chi[kL] \tag{6.29}$$

from which the $(p,q)^{th}$ element equals:

$$
\begin{aligned}
(\hat{\mathbf{R}}[kL])_{p,q} &= < \underline{\mathbf{x}}[kL-L+1+p], \underline{\mathbf{x}}[kL-L+1+q] >\\
&= \underline{\mathbf{x}}^t[kL-L+1+p]\underline{\mathbf{x}}[kL-L+1+q]. && (6.30)
\end{aligned}
$$

The length of vector $\underline{\mathbf{d}}[kL]$ can now be reduced with some amount according to the following update equation:

$$\begin{aligned}\underline{\mathbf{d}}[(k+1)L] &= \underline{\mathbf{d}}[kL] - 2\alpha\underline{\mathbf{d}}^{\Delta}[kL] \qquad (6.31)\\ &= \underline{\mathbf{d}}[kL] - 2\alpha\chi[kL]\hat{\mathbf{R}}^{-1}[kL]\underline{\mathbf{r}}_L[kL]. \qquad (6.32)\end{aligned}$$

This equation leads, with the definition of the difference vector, to the BOP update equation:

$$\underline{\mathbf{w}}[(k+1)L] = \underline{\mathbf{w}}[kL] + 2\alpha\chi[kL]\hat{\mathbf{R}}^{-1}[kL]\underline{\mathbf{r}}_L[kL]. \qquad (6.33)$$

Comparing this algorithm with the BNLMS update equation shows that the BOP algorithm is a generalization of the BNLMS algorithm. The input signal is decorrelated by an $L \times L$ "autocorrelation matrix" $\hat{\mathbf{R}}[kL]$. On the other hand this matrix needs not to have dimension $N \times N$ as in the RLS approach. In the experimental results at the end of this chapter it is shown that indeed for specific input signals decorrelation can be performed with the inverse of an $L \times L$ autocorrelation matrix, with $L < N$. Finally some general notes:

- In [45] (p69) the usefull suggestion is made to calculate the inverse of the autocorrelation matrix $\hat{\mathbf{R}}^{-1}[kL]$ by using equivalent "recursive" techniques as used for the RLS algorithm.
- Since both BOP and PBFDAF methods decorrelate the input signal with less than N degrees of freedom, a strong relationship between these two methods is expected. This relationship is investigated in [14]. The first step in this paper is to partition the adaptive weight vector in length L vectors. The BOP updating algorithm (6.33) is partitioned in an equivalent way. Using the same techniques as discussed in chapter 4, these partitioned BOP algorithm is implemented in frequency domain and is compared with the PBFDAF method. By doing so it follows that, in comparison to the PBFDAF algorithm, the BOP method performs a more accurate decorrelation, using a Toeplitz autocorrelation matrix, but it costs more complexity to implement it. On the other hand it is also shown that the decorrelation properties of the, relative low complexity, PBFDAF method are reasonably well.

6.4 BOP algorithm with Gram–Schmidt procedure

The purpose of this section is to rewrite the BOP updating equation (6.33) in such a way, that it can be used in the following section.

The BOP concept of the previous section is to make a projection of the vector $\underline{\mathrm{d}}[kL]$ on a space that was spanned by the basis:

$$\{\underline{\mathrm{x}}[kL-L+1],\cdots,\underline{\mathrm{x}}[kL]\}. \tag{6.34}$$

The first step in this section is to construct from this basis a new orthogonal basis by using the Gram–Schmidt procedure. After that the same BOP procedure is used, with this new basis, as given in the previous section.

For $i=0$ to $L-1$ the Gram–Schmidt procedure leads to the following set of orthogonal vectors:

$$\underline{\tilde{\mathrm{x}}}_i[kL] = \underline{\mathrm{x}}[kL-i] - \sum_{q=0}^{i-1} \frac{<\underline{\mathrm{x}}[kL-i],\underline{\tilde{\mathrm{x}}}_q[kL]>}{||\underline{\tilde{\mathrm{x}}}_q[kL]||^2}\underline{\tilde{\mathrm{x}}}_q[kL] \tag{6.35}$$

with the sum $\sum_{q=0}^{-1}$ defined as zero. Note furthermore that these constructed orthogonal vectors are not shifted versions of each other, in contrast to the input signal vectors, thus:

$$(\underline{\tilde{\mathrm{x}}}_i[kL])_t \neq (\underline{\tilde{\mathrm{x}}}_0[kL])_{t+i}. \tag{6.36}$$

The Gram–Schmidt procedure (6.35) is such that the new orthogonal basis and the initial basis span the same space. From this it is obvious that it is always possible to write each new basis vector $\underline{\tilde{\mathrm{x}}}_i kL]$ as a linear combination of initial basis vectors $\underline{\mathrm{x}}[kL-q]$. Thus for $i=0,\cdots,L-1$:

$$\underline{\tilde{\mathrm{x}}}_i[kL] = \sum_{q=0}^{i} \gamma_q^i[kL]\underline{\mathrm{x}}[kL-q]. \tag{6.37}$$

with $\gamma_i^i[kL]=1$ for all $i=0,1,\cdots,L-1$. In matrix notation these equations can be written as follows:

$$\tilde{\chi}[kL] = \chi[kL]\cdot\mathbf{\Gamma}[kL] \tag{6.38}$$

with the $L \times L$ lower triangular matrix

$$\Gamma[kL] = \begin{pmatrix} \gamma_{L-1}^{L-1}[kL] & 0 & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \gamma_1^{L-1}[kL] & \cdot & \cdot & \cdot & \gamma_1^1[kL] & 0 \\ \gamma_0^{L-1}[kL] & \cdot & \cdot & \cdot & \gamma_0^1[kL] & \gamma_0^0[kL] \end{pmatrix} \tag{6.39}$$

and the $N \times L$ matrix containing the orthogonal set of basis vectors as follows:

$$\tilde{\chi}[kL] = (\tilde{\underline{\mathbf{x}}}_{L-1}[kL], \cdots, \tilde{\underline{\mathbf{x}}}_0[kL])\,. \tag{6.40}$$

The BOP concept is to make a projection of the vector $\underline{\mathbf{d}}[kL]$ on the orthogonal basis. This leads to:

$$\begin{aligned} < \underline{\mathbf{d}}^{\perp}[kL], \tilde{\underline{\mathbf{x}}}_i[kL] > &= 0 \quad \text{for } i = 0, \cdots, L-1 \\ \underline{\mathbf{d}}^{\Delta}[kL] &= \chi[kL]\underline{\mathbf{c}}[kL] = \tilde{\chi}[kL]\tilde{\underline{\mathbf{c}}}[kL] \end{aligned} \tag{6.41}$$

with $\tilde{\underline{\mathbf{c}}}[kL] = \mathbf{\Gamma}^{-1}[kL]\underline{\mathbf{c}}[kL]$. Furthermore by defining the rotated (transformed) residual signal vector as:

$$\tilde{\underline{\mathbf{r}}}_L[kL] = \mathbf{\Gamma}^t[kL]\underline{\mathbf{r}}_L[kL] = (\tilde{r}_{L-1}[kL], \cdots, \tilde{r}_0[kL])^t \tag{6.42}$$

the following Gram-Schmidt BOP update algorithm (GS–BOP) can be derived in a similar way as described in the previous section:

$$\underline{\mathbf{w}}[(k+1)L] = \underline{\mathbf{w}}[kL] + 2\alpha\tilde{\chi}[kL]\tilde{\mathbf{R}}^{-1}[kL]\tilde{\underline{\mathbf{r}}}_L[kL]. \tag{6.43}$$

In this equation the $L \times L$ rotated (transformed) "autocorrelation" matrix $\tilde{\mathbf{R}}[kL]$ is diagonal because of the orthogonality property of the vectors $\tilde{\underline{\mathbf{x}}}_i[kL]$. Thus:

$$\tilde{\mathbf{R}}[kL] = diag\{||\tilde{\underline{\mathbf{x}}}_{L-1}[kL]||^2, \cdots, ||\tilde{\underline{\mathbf{x}}}_0[kL]||^2\}. \tag{6.44}$$

Now it is straightforward to rewrite the GS–BOP algorithm (6.43) as:

$$\underline{\mathbf{w}}[(k+1)L] = \underline{\mathbf{w}}[kL] + 2\alpha \sum_{i=0}^{L-1} \frac{\tilde{\underline{\mathbf{x}}}_i[kL](\tilde{\underline{\mathbf{r}}}_L[kL])_{L-1-i}}{||\tilde{\underline{\mathbf{x}}}_i[kL]||^2}. \tag{6.45}$$

From this update equation it follows that every iteration, thus every L samples, one "NLMS" correction is made for the difference vector in each of the L orthogonal directions. Thus a correction made in one direction is not influenced by a correction made in another direction.

As a final comment of this section it is noted that in [40] (p99) some efficient alternatives are given to compute the matrix $\mathbf{\Gamma}[kL]$. One of them uses the Householder transform [64].

6.5 An efficient OP algorithm for AR(p) processes

Applying the GS–BOP concept of the previous section to the "sliding" approach, results in an algorithm equivalent to update equation (6.45). At time $k \cdot T$ a set of L orthogonal vectors is calculated spanning an L dimensional plane, and the adaptive weight vector is updated in L different orthonormal directions. At time $(k+1) \cdot T$ a complete new set of L orthogonal vectors is calculated spanning a new L dimensional plane. Updates in orthogonal directions do not influence each other. Since, however the L dimensional planes at time $k \cdot T$ and at time $(k+1) \cdot T$ need not to be orthogonal, it is obvious that an update of the adaptive weight vector in one iteration can be influenced by previous or following updates.

From literature [35] it is known that a speech signal can be modelled by using an ar(p) model, with p in the order of $8, \cdots, 12$. This section gives an algorithm is given (derived from (6.45)) that perfectly decorrelates ar(p) input signals. After that an efficient realisation of this scheme is given.

A ar(p) signal is described by the following difference equation:

$$x[k] = \sum_{i=1}^{p} a_i x[k-i] + n[k] \tag{6.46}$$

with $n[k]$ a white noise process. Describing this equation in N dimensional vectors $\underline{\mathrm{x}}[k]$ and $\underline{\mathrm{n}}[k]$ gives:

$$\underline{\mathrm{x}}[k] = \sum_{i=1}^{p} a_i \underline{\mathrm{x}}[k-i] + \underline{\mathrm{n}}[k]. \tag{6.47}$$

Since $n[k]$ is a white noise process it follows that the vector $\underline{\mathrm{n}}[k]$ is independent of $\underline{\mathrm{n}}[k-i]$ for all $i > 0$. Now the following signal is defined

$$\underline{\tilde{\mathrm{x}}}_p[k] = \underline{\mathrm{x}}[k] - \sum_{i=1}^{p} \hat{a}_i \underline{\mathrm{x}}[k-i] \tag{6.48}$$

with $\underline{\tilde{\mathrm{x}}}_p[k] = (\tilde{x}_p[k-N+1], \cdots, \tilde{x}_p[k])^t$ and $\hat{a}_i$ some estimate of the coefficient a_i. Note that a recursive scheme (with low complexity) to calculate the coefficients $\hat{a}_i$ is the following LMS update algorithm [29]:

$$\hat{a}_i[k+1] = \hat{a}_i[k] + 2\beta \tilde{x}_p[k] x[k-i]. \tag{6.49}$$

By comparing the "new" vector $\underline{\tilde{\mathrm{x}}}_p[k]$ of equation (6.48) with the definition of an ar(p) process, it is obvious that $\underline{\tilde{\mathrm{x}}}_p[k]$ is an estimate of the white noise process $\underline{\mathrm{n}}[k]$. This implies that when $\hat{a}_i = a_i$

$$\underline{\tilde{\mathrm{x}}}_p[k] \perp \underline{\tilde{\mathrm{x}}}_p[k-i] \quad i > 0. \tag{6.50}$$

In contrast to the result of the previous section: For ar(p) signals it is better to a make every iteration only a correction in direction $\underline{\tilde{\mathrm{x}}}_p[k]$, since this vector is already orthogonal to all previous update directions. Thus for ar(p) processes only the first part of (6.45) is needed. This results in the following update algorithm, that decorrelates an ar(p) input signal completely:

$$\underline{\mathrm{w}}[k+1] = \underline{\mathrm{w}}[k] + 2\alpha \frac{\underline{\tilde{\mathrm{x}}}_p[k] \tilde{r}_p[k]}{||\underline{\tilde{\mathrm{x}}}_p[k]||^2} \tag{6.51}$$

with:

$$\begin{aligned} \tilde{r}_p[k] &= r[k] - \sum_{i=1}^{p} \hat{a}_i[k] r_i[k] \\ r_i[k] &= \underline{\mathrm{x}}^t[k-i]\underline{\mathrm{d}}[k] + s[k-i] \quad \text{for} \quad i = 0, \cdots, p. \end{aligned}$$

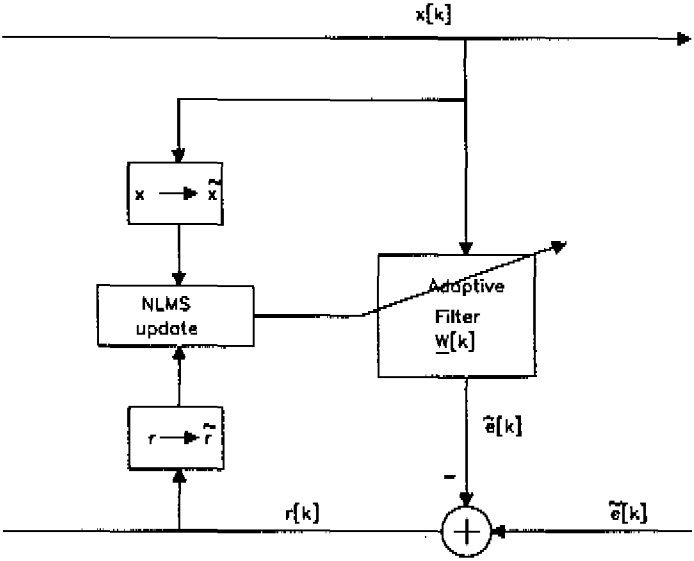

Figure 6.2: *OP algorithm for AR signals*

This update scheme is refered to as the Orthogonal Projection algorithm for Auto Regressive signals (OP–AR), and a schematic interpretation is depicted in Fig. 6.2. In comparison to the NLMS algoritm this OP–AR algorithm first "whitens" both the input signal ($x \rightarrow \tilde{x}$) and the "x-component in the residual signal" ($r \rightarrow \tilde{r}$). With these "white" signals an NLMS update is made. Thus convergence properties are independent of the input signal statistics. In fact the OP–AR algorithm is a parametric algorithm to solve the decorrelation problem, whereas the frequency domain and RLS approaches are non–parametric approaches.

The problem however with the OP–AR algorithm is complexity. Every iteration $p-1$ extra convolutions of length N have to be calculated to generate the residual signals $r_i[k]$. Since, however, the algorithm decorrelates perfectly, the following approximation can be used to lower down this complexity:

$$E\{\underline{d}[k]\} \approx (1-\frac{2\alpha}{N})^i E\{\underline{d}[k-i]\}. \tag{6.52}$$

Using this approximation it follows that an estimate of $\hat{r}_i[k]$ of the signal $r_i[k]$ is given by:

$$\hat{r}_i[k] \approx (1 - \frac{2\alpha}{N})^i r[k-i]. \tag{6.53}$$

Using this approximation complexity can drastically be reduced. Together with a simple low pass filter for the power average, leads to the following Efficient OP–AR (EOP–AR) algorithm [53,60]:

$$\underline{\mathrm{w}}[k+1] = \underline{\mathrm{w}}[k] + 2\alpha \frac{\tilde{\underline{\mathrm{x}}}_p[k]\hat{\tilde{r}}_p[k]}{N\hat{\sigma}_x^2[k]} \tag{6.54}$$

with:

$$\begin{aligned}
\tilde{\underline{\mathrm{x}}}_p[k] &= \underline{\mathrm{x}}[k] - \sum_{i=1}^{p} \hat{a}_i[k]\underline{\mathrm{x}}[k-i] &(6.55)\\
\hat{\tilde{r}}_p[k] &= r[k] - \sum_{i=1}^{p} \hat{a}_i[k]\hat{r}_i[k] \\
r[k] &= \tilde{e}[k] - \hat{e}[k] \\
\hat{r}_i[k] &= (1 - \frac{2\alpha}{N})^i r[k-i] \quad \text{for } i = 1, \cdots, p \\
\hat{a}_i[k+1] &= \hat{a}_i[k] + 2\beta\tilde{x}[k]x[k-i] \quad \text{for } i = 1, \cdots, p \\
\hat{\sigma}_x^2[k+1] &= \gamma\hat{\sigma}_x^2[k] + (1-\gamma)x^2[k].
\end{aligned}$$

This EOP–AR algorithm can <u>completely</u> decorrelate an ar(p) input signal, while the complexity (the number of multiplications) has the same order as the NLMS algorithm. An implementation for the boxes $x \rightarrow \tilde{x}$ and $r \rightarrow \tilde{r}$ of Fig. 6.2 for this EOP–AR algorithm is given in Fig. 6.3 for $p = 1$.

Finally it is noted that it can be proved that the first update algorithm given in [39] (page 21) is, as the authors say, a specific form of the OP algorithm for $\alpha = \frac{1}{2}$. In [70] however this specific algorithm is used for arbitrary values of α (in the convergence area). This leads to the next algorithm:

$$\underline{\mathrm{w}}[k+1] = \underline{\mathrm{w}}[k] + 2\alpha \frac{\tilde{\underline{\mathrm{x}}}_p[k]r[k]}{< \underline{\mathrm{x}}[k], \tilde{\underline{\mathrm{x}}}_p[k] >} \tag{6.56}$$

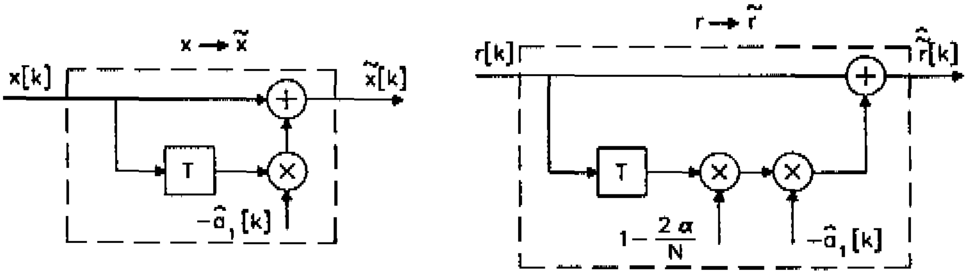

Figure 6.3: *Implementation of $x \to \tilde{x}$ and $r \to \tilde{r}$ for EOP-AR algorithm with $p = 1$*

This however is no "real" OP algorithm as proposed in [39] (page 23 under formula (3)), but it looks like an "OP-AR" algorithm. The "whitening" however is only performed in the x component and not in the residual signal r. The result is that this algorithm will not completely decorrelate an ar input signal.

6.6 Experiments

Results of the previous section are verified in this section with some experiments. For these experiments the system as given in Fig. 1.7 is used, from which the "unknown" Wiener system has an impulse response of length $N = 32$. The adaptive weights are initialized with zeros. The signal $s[k]$ is zero, thus the quantity of interest is:

$$10\log\left(\frac{E\{(e[k]-\hat{e}[k])^2\}}{E\{e^2[k]\}}\right). \tag{6.57}$$

The input signal is an ma(1) or ar(1) signal with $ER = 100$. This is done by chosing for both models the parameter $a = +0.8182$ and $n[k]$ is taken as a white noise signal, having zero average and $E\{n^2[k]\} = 1/3$.

The adaptation constant is $\alpha = 4.6/1000$. In the first experiment it is shown that it is possible to perform the decorrelation with the inverse of an $L \times L$ autocorrelation matrix with $L \leq N$. The results of this experiment are plotted in Fig. 6.4 and Fig. 6.5. From both figures it follows that when lowering the dimension L of the autocorralation matrix convergence properties degrade. Comparing however the result for the ma(1) signal (Fig. 6.4) and the ar(1) signal (Fig. 6.5) shows that the ma(1) signal is much more sensitive for lowering this dimension L. Convergence properties with an ma(1) signal as input already degrades for $L \leq 16$, while the ar(1) degrades for $L \leq 8$, even for $L = 2$ convergence is reasonable for the ar(1) process. These results can be explained as follows: since the ma(1) model contains a zero, it is obvious that the inverse autocorrelation matrix is a full matrix. Making the dimension smaller results in throwing away relevant information. The ar(1) model however contains a pole, resulting in an tri–diagonal inverse autocorrelation matrix. Lowering down the dimension is allowed.

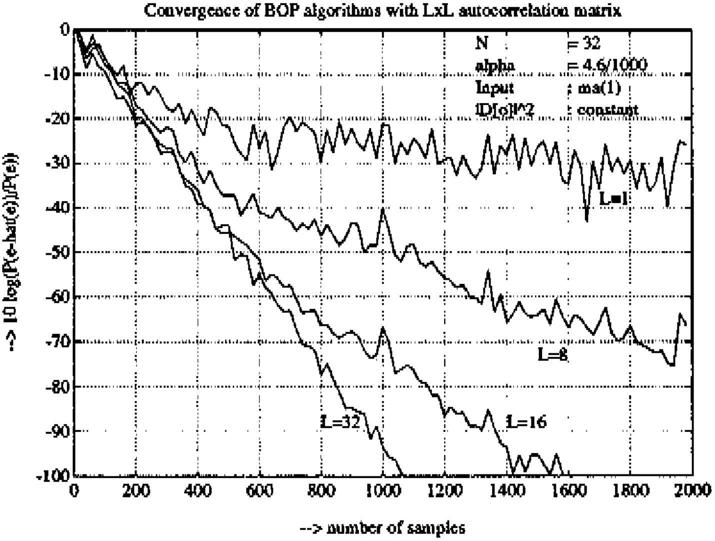

Figure 6.4: *Convergence BOP algorithm with $L \times L$ matrix for ma(1) signal*

The second experiment shows that the EOP algorithm indeed decorre-

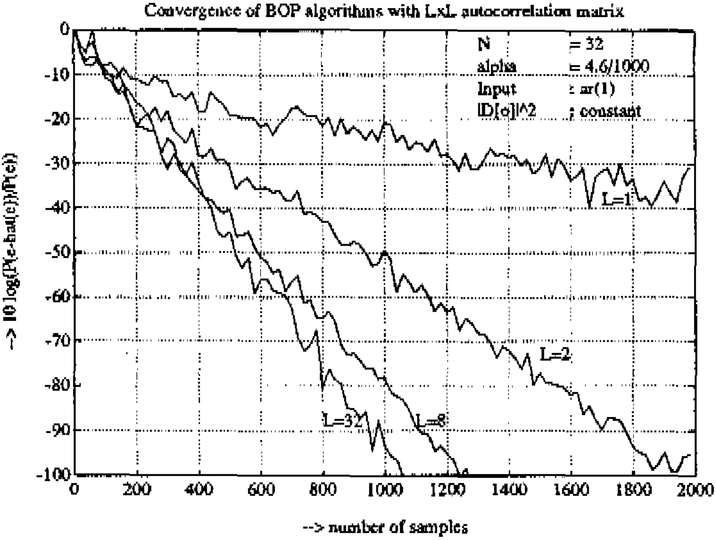

Figure 6.5: *Convergence BOP algorithm with $L \times L$ matrix for ar(1) signal*

lates a signal that is generated by an ar model. The result is plotted in Fig. 6.6. As a reference the white noise curve represents the ideal decorrelation case. The input signal is an ar(1) signal, and also the result of convergence of the NLMS algorithm is plotted. The third curve shows the result from the EOP algorithm as depicted in Fig. 6.3. This curve is indeed close to the ideal case, which shows the decorrelation capacity of the EOP algorithm for ar(1) signals.

6.7 Discussion

In this chapter it is shown that it is possible to decorrelate an adaptive filter with N weights , with an $L \times L$ autocorrelation matrix with $L \geq 1$. For signals with spectral poles L can be relative small, resulting in a small amount of extra complexity to decorrelate such a signal. Signal models with spectral zeros need a higher dimension L.

When from a priori knowledge it is known that the input signal is generated with an ar(p) model it is shown that the EOP algorithm decor-

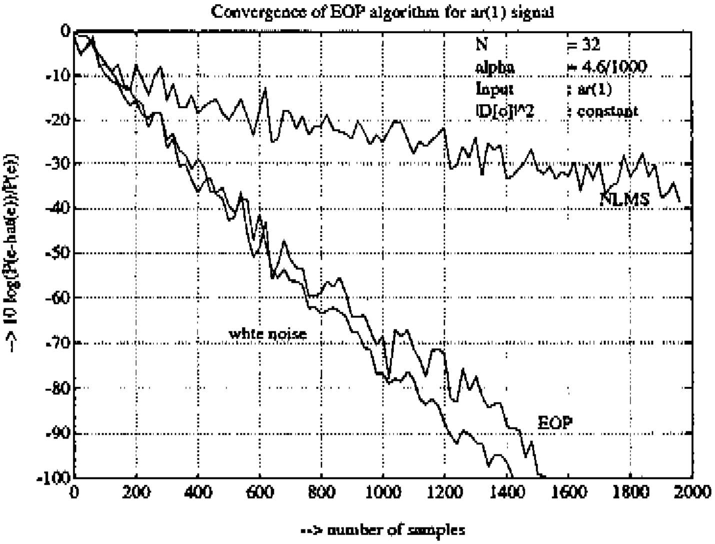

Figure 6.6: *Convergence of EOP algorithm for ar(1) signal*

relates such a signal, with complexity in the order of the complexity needed for the NLMS algorithm.

Chapter 7

Conclusions

This thesis explores some *methods to use available a priori information in order to reduce complexity while maintaining convergence properties* of adaptive filters. The most important results are listed below:

A reasonable approximation of the convergence properties of the Block Normalized Least Mean Square (BNLMS) algorithm can be deduced from the product of the spectrum of the input signal with the squared magnitude of the system function (smf) from the difference vector. A priori knowledge about the "matching" of the signal characteristics and the "unknown" system can be used to initialize the algorithm as well as possible.

For large filters the BNLMS algorithm can be implemented very efficiently in frequency domain by using Fast Fourier Transforms (FFTs) for the transformation between time– and frequency domain. The length of these transforms is given by the number $B = N + L - 1$, with N the number of adaptive weights and L the processing delay (resulting from the block processing approach). Both overlap–save and overlap–add method can be implemented with equivalent complexity.

Convergence properties of an adaptive filter can be made independent of the input signal statistics by decorrelation of the input signal. This decorrelation can be accomplished relatively easy with frequency domain techniques by normalizing the spectrum of each separate frequency component. With this method an approximation is made of the required time domain decorrelation. Applying this spectral normalization to the efficiently implemented BNLMS algorithm in frequency domain, leads to the Block Frequency Domain Adaptive Filter (BFDAF). Roughly there are two variants of the BFDAF. The first method is the constrained BFDAF, since it requires a window that forces a constraint in adjusting the frequency domain weights based on overlap–save sectioning. The implementation of this structure requires five FFTs. The second method is the unconstrained BFDAF, since it removes the window. This method uses three FFTs. For input signals that have an autocorrelation function $\rho[\tau]$, that is neglectable for $|\tau| < N/2$ and $L > N/2$, the constrained BFDAF has better convergence properties than the unconstrained BFDAF: Because of the constaint window less weight are fluctuating around their final value resulting in a smaller final misadjustment. Furthermore when the impulse response of the unknown system is a global decaying function it is possible to use an efficient implementation of the BFDAF, using 3 FFTs, with convergence properties equivalent to the constrained BFDAF (5FFTs).

By partitioning the original BFDAF into smaller parts and implementing this in an efficient way leads to the Partitioned Block Frequency Domain Adaptive Filter (PBFDAF). In comparison with the BFDAF, the PBFDAF structure can be realized with smaller FFTs, resulting in a reduced processing delay. Furthermore when some a priori information is available of the input signal spectrum this information can be used to reduce complexity, since decorrelation is performed by less divisions in the PBFDAF approach.

With the Block Orthogonal Projection (BOP) method it is possible to decorrelate the input signal of an adaptive filter with the inverse of an autocorrelation matrix from which the dimension can be chosen in accordance to the input signal statistics. The BOP and the PBFDAF are strongly related, since both methods reduce the required number of degrees of freedom in order to decorrelate the input signal of the adaptive filter. Furthermore it is possible to decorrelate auto regressive signals with an Efficient Orthogonal Projection (EOP) algorithm.

Finally some recommendations for future reserch are listed below:

- Still a *complete* analysis of the (BN)LMS algorithm is required that is both relatively simple and that uses correct assumptions.
- From several discussions given in this thesis it follows that convergence properties of adaptive filters are depending on statistical properties of the input signal of the adaptive filter. On the other hand it is known that multirate techniques introduces spectral deformations (i.e. aliasing, mirroring). From this it follows that, when using adaptive techniques in a multirate environment [11], a description of the convergence properties is not trivial. New fundamental concepts are needed in order to descibe this *multirate adaptive digital signal processing* area. Some papers in this area are: [31,33,20].
  Note that the PBFDAF structure of Chapter 5 of this thesis is an example of a multirate adaptive filter: Each separate frequency component is filtered with a separate filter, having a sample rate that differs from the sample rate of the input signal $x[k]$.
- The Efficient Orthogonal Projection (EOP) algorithm (Section 6.5) uses a signal model, with only poles, to decorrelate the input signal of the adaptive filter in an efficient way. It is usefull to extend this technique for general signal models containing poles and/or zeros for the following two reasons:
  - The input signal is decorrelated by using only relevant information needed for the decorrelation.
  - This method is suitable for the tracking problem: The signal model can be used to track the nostationarities of the input signal, while the adaptive filter itself can be used to track the nonstationarities of the unknow system.
- When applying techniques known from fixed filters to adaptive filters, this must always be done with some care since the adaptive weights change every iteration. An example is the discussing of the overlap–add method for adaptive filters, as given in Section 3.3 and 3.4. Another challenge is the area of *parallel adaptive digital signal processing*: splitting the original problem into parallel parts has to be done with much care when adaptive weights are involved.

This area needs new fundamental research. Some literature in this area is given in [26](Chapter 10) and its references.

Appendix A

Derivation of dynamic behavior of BNLMS algorithm

In this appendix more accurate formulas are given that describe the dynamic behavior of the BNLMS algorithm. This is done by giving a derivation that is mainly based on results given in [15]. Although this derivation has some weak points, mentioned explicitly here, no alternative derivation is given because theoretical and experimental results fit reasonably well.

From the BNLMS update equation (2.17) and with $\overline{\alpha}_L = \frac{\alpha_L}{L\sigma_x^2}$ it follows:

$$\underline{\mathbf{d}}[(k+1)L] = \left(\mathbf{I} - 2\overline{\alpha}_L \chi[kL]\chi^t[kL]\right)\underline{\mathbf{d}}[kL] - 2\overline{\alpha}_L \chi[kL]\underline{\mathbf{s}}_L[kL]. \quad (\mathrm{A.1})$$

Transforming this equation with the unitary matrix $\mathbf{Q}$ gives:

$$\begin{aligned}\underline{\mathbf{D}}[(k+1)L] &= \left(\mathbf{I} - 2\overline{\alpha}_L \mathbf{Q}^h\chi[kL]\chi^t[kL]\mathbf{Q}\right)\underline{\mathbf{D}}[kL] - 2\overline{\alpha}_L \mathbf{Q}^h\chi[kL]\underline{\mathbf{s}}_L[k \\ &= \left(\mathbf{I} - 2\overline{\alpha}_L \mathbf{X}[kL]\mathbf{X}^h[kL]\right)\underline{\mathbf{D}}[kL] - 2\overline{\alpha}_L \mathbf{X}[kL]\underline{\mathbf{s}}_L[kL] \quad (\mathrm{A.}\end{aligned}$$

with

$$\mathbf{X}[kL] = \mathbf{Q}^h\chi[kL]. \quad (\mathrm{A.3})$$

This matrix has the property

$$\begin{aligned} E\{\mathbf{X}[kL]\mathbf{X}^h[kL]\} &= E\{\mathbf{Q}^h\chi[kL]\chi^t[kL]\mathbf{Q}\} \\ &= L\mathbf{Q}^h\mathbf{R}\mathbf{Q} = L \cdot \mathbf{\Lambda}. \end{aligned} \tag{A.4}$$

Furthermore for small adaptation constant the input signal x and the adaptive weights w may be separated under $E\{\cdot\}$. This results in:

$$\begin{aligned} & \Delta[(k+1)L] = E\{\underline{\mathbf{D}}[(k+1)L]\underline{\mathbf{D}}^h[(k+1)L]\} \\ \approx\ & E\left\{(\mathbf{I} - 2\overline{\alpha}_L\mathbf{X}[kL]\mathbf{X}^h[kL]) \cdot \Delta[kL] \cdot (\mathbf{I} - 2\overline{\alpha}_L\mathbf{X}[kL]\mathbf{X}^h[kL])\right\} \\ & +4\overline{\alpha}_L^2 E\left\{\mathbf{X}[kL] E\left\{\underline{\mathbf{s}}_L[kL]\underline{\mathbf{s}}_L^t[kL]\right\}\mathbf{X}^h[kL]\right\} \\ =\ & \Delta[kL] - 2\overline{\alpha}_L E\{\mathbf{X}[kL]\mathbf{X}^h[kL]\}\Delta[kL] \\ & -2\overline{\alpha}_L\Delta[kL]E\{\mathbf{X}[kL]\mathbf{X}^h[kL]\} \\ & +4\overline{\alpha}_L^2 E\{\mathbf{X}[kL]\mathbf{X}^h[kL] \cdot \Delta[kL] \cdot \mathbf{X}[kL]\mathbf{X}^h[kL]\} \\ & +4\overline{\alpha}_L^2 J_{\min} E\{\mathbf{X}[kL]\mathbf{X}^h[kL]\} \\ =\ & \Delta[kL] - 2L\overline{\alpha}_L\mathbf{\Lambda}\Delta[kL] - 2L\overline{\alpha}_L\Delta[kL]\mathbf{\Lambda} \\ & +4\overline{\alpha}_L^2 E\{\mathbf{X}[kL]\mathbf{X}^h[kL]\Delta[kL]\mathbf{X}[kL]\mathbf{X}^h[kL]\} \\ & +4L\overline{\alpha}_L^2 J_{\min}\mathbf{\Lambda}. \end{aligned} \tag{A.5}$$

The first weak point of this analysis is that the above equation only holds for small adaptation constant, while later on the same equations are used as if they are valid for large adaptation constant too! A possible validation can be that for large adaptation constant the residual signal is decreasing very fast, resulting in a steady state final value of the adaptive weights. Again the above mentioned separation under $E\{\cdot\}$ may be applied.

Skipping the time index for one moment, the $(i,j)^{th}$ element of the last matrix between brackets is given by:

$$E\{\mathbf{X}\mathbf{X}^h\Delta\mathbf{X}\mathbf{X}^h\}_{ij} = \sum_{p=0}^{L-1}\sum_{t=0}^{N-1}\sum_{r=0}^{N-1}\sum_{s=0}^{L-1} E\{\mathbf{X}_{ip}\mathbf{X}_{tp}^*\mathbf{X}_{rs}\mathbf{X}_{js}^*\}\Delta_{tr}. \tag{A.6}$$

Now the assumption is made that the elements of matrix $\mathbf{X}$ have a complex circular Gaussian distribution function. With this assumption this fourth order moment can be split into three different parts [42]:

$$\begin{aligned} E\{\mathbf{X}_{ip}\mathbf{X}^*_{tp}\mathbf{X}_{rs}\mathbf{X}^*_{js}\} &= E\{\mathbf{X}_{ip}\mathbf{X}^*_{tp}\}E\{\mathbf{X}_{rs}\mathbf{X}^*_{js}\} + E\{\mathbf{X}_{ip}\mathbf{X}_{rs}\}E\{\mathbf{X}^*_{tp}\mathbf{X}^*_{js} \\ &\quad + E\{\mathbf{X}_{ip}\mathbf{X}^*_{js}\}E\{\mathbf{X}_{tp}\mathbf{X}^*_{rs}\}. \qquad \text{(A.7} \end{aligned}$$

Using the definition of matrix $\mathbf{X}[kL] = \mathbf{Q}^h\chi[kL]$ this gives for the $(p,t)^{th}$ element:

$$\mathbf{X}_{pt} = \underline{\mathbf{q}}_p^h\underline{\mathbf{x}}[kL-L+1-t] = \underline{\mathbf{x}}^t[kL-L+1-t]\underline{\mathbf{q}}_p^* \tag{A.8}$$

and in general:

$$E\{\mathbf{X}_{pt}\mathbf{X}^*_{mn}\} = \underline{\mathbf{q}}_p^h E\{\underline{\mathbf{x}}[kL-L+1-t]\underline{\mathbf{x}}^t[kL-L+1-n]\}\underline{\mathbf{q}}_m. \tag{A.9}$$

The second weak point of the analysis is that it is assumed that the main contributions of this matrix are on the diagonal elements, thus for $t = n$, and that the contributions for $t \neq n$ may be neglected in the sequel. Using this, it follows furthermore that

$$\begin{aligned} E\{\mathbf{X}_{pl}\mathbf{X}^*_{ml}\} &= \underline{\mathbf{q}}_p^h E\{\underline{\mathbf{x}}[kL-L+1-l]\underline{\mathbf{x}}^t[kL-L+1-l]\}\underline{\mathbf{q}}_m \\ &= \underline{\mathbf{q}}_p^h\mathbf{R}\underline{\mathbf{q}}_m = \begin{cases} \lambda_p & \text{for } p = m \\ 0 & \text{for } p \neq m \end{cases} \end{aligned} \tag{A.10}$$

On the other hand it follows from the circular Gaussian assumption that

$$E\{\mathbf{X}_{pt}\mathbf{X}_{mn}\} = 0. \tag{A.11}$$

Using these results this leads to:

$$E\{\mathbf{X}\mathbf{X}^h\Delta\mathbf{X}\mathbf{X}^h\} \approx L^2\mathbf{\Lambda}\Delta\mathbf{\Lambda} + L\cdot trace\{\mathbf{\Lambda}\Delta\}\mathbf{\Lambda}. \tag{A.12}$$

With this, the above expression for $\Delta[(k+1)L]$ can be written as:

$$\begin{aligned} \Delta[(k+1)L] &= \Delta[kL] - 2L\overline{\alpha}_L\left(\mathbf{\Lambda}\Delta[kL] + \Delta[kL]\mathbf{\Lambda}\right) \\ &\quad + 4L\overline{\alpha}_L^2\left(L\mathbf{\Lambda}\Delta[kL]\mathbf{\Lambda} + trace\{\mathbf{\Lambda}\Delta[kL]\}\mathbf{\Lambda}\right) \\ &\quad + 4L\overline{\alpha}_L^2 J_{\min}\mathbf{\Lambda}. \end{aligned} \tag{A.13}$$

Since only the diagonal elements of Δ are of interest, this expression can be rewritten as:

$$\underline{\mathbf{\Delta}}[(k+1)L] = \mathbf{A}\underline{\mathbf{\Delta}}[kL] + 4L\overline{\alpha}_L^2 J_{\min}\underline{\lambda} \tag{A.14}$$

with

$$\mathbf{A} = \mathbf{I} - 4L\overline{\alpha}_L\mathbf{\Lambda} + 4L^2\overline{\alpha}_L^2\mathbf{\Lambda}^2 + 4L\overline{\alpha}_L^2\underline{\lambda}\cdot\underline{\lambda}^t \tag{A.15}$$

and

$$\begin{aligned} \underline{\lambda} &= (\lambda_0, \cdots, \lambda_{N-1})^t \\ \underline{\Delta}[kL] &= (\Delta_{0,0}[kL], \cdots, \Delta_{N-1,N-1}[kL])^h. \end{aligned} \tag{A.16}$$

This equation fully describes the dynamic behavior of the adaptive filter using the BNLMS update scheme. Following the same strategy as in [15] from this equation the following important convergence characteristics can be derived.

Convergence Area:

It can be shown ([15]) that the BNLMS algorithm converges if:

$$1 + 4L\overline{\alpha}_L^2 \sum_{l=0}^{N-1} \frac{\lambda_l^2}{1 - 4L\overline{\alpha}_L\lambda_l + 4L^2\overline{\alpha}_L^2\lambda_l^2 - 1} > 0 \tag{A.17}$$

and

$$1 - 4L\overline{\alpha}_L\lambda_l + 4L^2\overline{\alpha}_L^2\lambda_l^2 < 1. \tag{A.18}$$

Using the definition for the relative eigenvalue

$$\overline{\lambda}_l = \frac{\lambda_l}{\frac{1}{N}\sum_{q=0}^{N-1}\lambda_q} = \frac{\lambda_l}{\sigma_x^2} \tag{A.19}$$

and introducing

$$\Sigma_{\alpha,L}(\overline{\lambda}) = \frac{1}{L}\sum_{l=0}^{N-1}\frac{\alpha_L\overline{\lambda}_l}{1 - \alpha_L\overline{\lambda}_l} \tag{A.20}$$

it can be concluded that the convergence region of the BNLMS algorithm is given by:

$$0 < \alpha_L < \frac{1}{\overline{\lambda}_l} \quad \text{and} \quad \Sigma_{\alpha,L}(\overline{\lambda}) < 1. \tag{A.21}$$

Final Misadjustment:

Using the expression describing the dynamic behavior of the BNLMS algorithm for $k \rightarrow \infty$ the final misadjustmant can be written as:

$$\overline{J} = \lim_{k\rightarrow\infty}\left(\frac{J[kL] - J_{\text{min}}}{J_{\text{min}}}\right) = 4L\overline{\alpha}_L^2\underline{\lambda}^t(\mathbf{I} - \mathbf{A})^{-1}\underline{\lambda} \tag{A.22}$$

with

$$\mathbf{I} - \mathbf{A} = 4L\overline{\alpha}_L\left(\Gamma + \gamma\underline{\lambda}\underline{\lambda}^t\right) \tag{A.23}$$

and

$$\begin{aligned} \Gamma &= \mathbf{\Lambda} - L\overline{\alpha}_L\mathbf{\Lambda}^2 \\ \gamma &= -\overline{\alpha}_L. \end{aligned} \tag{A.24}$$

Using the Barlett formula (as in [15]):

$$\begin{aligned} \underline{\lambda}^t\left(\Gamma + \gamma\underline{\lambda}\underline{\lambda}^t\right)^{-1}\underline{\lambda} &= \underline{\lambda}^t\left(\Gamma^{-1} - \frac{\gamma}{1 + \gamma\underline{\lambda}^t\Gamma^{-1}\underline{\lambda}}\cdot\Gamma^{-1}\underline{\lambda}\cdot\underline{\lambda}^t\Gamma^{-1}\right)\underline{\lambda} \\ &= \frac{\underline{\lambda}^t\Gamma^{-1}\underline{\lambda}}{1 + \gamma\underline{\lambda}^t\Gamma^{-1}\underline{\lambda}}. \end{aligned} \tag{A.25}$$

Combing this with the expression for the final misadjustment gives:

$$\overline{J} = \frac{\Sigma_{\alpha,L}(\overline{\lambda})}{1 - \Sigma_{\alpha,L}(\overline{\lambda})}. \tag{A.26}$$

Note that indeed for small adaptation constant α_L this quantity can be approximated with the equation given in chapter 2, namely:

$$\overline{J} \approx \frac{\alpha_L}{L}N. \tag{A.27}$$

which is indeed the result used in chapter 2.

Rate of Convergence:

A closed expression for the rate of convergence as given in [15] can be derived as follows:

$$\begin{aligned} J_* &= L\cdot\sum_{k=0}^{\infty}\left(J[kL] - J[\infty]\right) = L\cdot\sum_{k=0}^{\infty}\left(J_{\text{ex}}[kL] - J_{\text{ex}}[\infty]\right) \\ &= L\cdot\underline{\lambda}^t\sum_{k=0}^{\infty}\left(\underline{\mathbf{\Delta}}[kL] - \underline{\mathbf{\Delta}}[\infty]\right) \end{aligned} \tag{A.28}$$

Furthermore with:

$$\underline{\Delta}[kL] - \underline{\Delta}[\infty] = \mathbf{A} \cdot (\underline{\Delta}[(k-1)L] - \underline{\Delta}[\infty])\,. \tag{A.29}$$

this implies for the rate of convergence:

$$\begin{aligned} J_s &= L \cdot \underline{\lambda}^t \cdot \sum_{k=0}^{\infty} \mathbf{A}\,(\underline{\Delta}[(k-1)L] - \underline{\Delta}[\infty]) \\ &= L \cdot \underline{\lambda}^t \cdot \sum_{k=0}^{\infty} \left\{\mathbf{A}^k\right\} \cdot (\underline{\Delta}[0] - \underline{\Delta}[\infty]) \\ &= L \cdot \underline{\lambda}^t \cdot (\mathbf{I} - \mathbf{A})^{-1} \underline{\Delta}_{\mathrm{ini}} \end{aligned} \tag{A.30}$$

with

$$\underline{\Delta}_{\mathrm{ini}} = \underline{\Delta}[0] - \underline{\Delta}[\infty]. \tag{A.31}$$

Using again a modification of the Barlett formula gives:

$$\begin{aligned} \underline{\lambda}^t \left(\Gamma + \gamma \underline{\lambda}\underline{\lambda}^t\right)^{-1} &= \underline{\lambda}^t \left(\Gamma^{-1} - \frac{\gamma}{1 + \gamma \underline{\lambda}^t \Gamma^{-1} \underline{\lambda}} \Gamma^{-1} \underline{\lambda}\underline{\lambda}^t \Gamma^{-1}\right) \\ &= \frac{\underline{\lambda}^t \Gamma^{-1}}{1 + \gamma \underline{\lambda}^t \Gamma^{-1} \underline{\lambda}} \end{aligned} \tag{A.32}$$

which results in the next expression for the rate of convergence of the BNLMS algorithm:

$$J_s = \frac{1}{4\overline{\alpha}_L} \cdot \frac{\sum_{l=0}^{N-1} \frac{\Delta_{\mathrm{ini},l}}{1 - \overline{\alpha_L \lambda_l}}}{1 - \Sigma_{\alpha,L}(\overline{\lambda})}. \tag{A.33}$$

For small adaptation constant α this equation reduces to

$$J_s \approx \frac{1}{4\overline{\alpha}_L} \sum_{l=0}^{N-1} \Delta_{\mathrm{ini},l} = \frac{\sigma_x^2}{4\frac{\alpha_L}{L}} \sum_{l=0}^{N-1} \Delta_{\mathrm{ini},l}. \tag{A.34}$$

A problem with this measure for the rate of convergence is that a fast, or slow, initial part is neglected by definition.

The rate of convergence in Chapter 2 gives the number of samples needed to reduce the quantity $10\log J_{\mathrm{ex}}[0]/J_{\min}$ by 20 dB, and for small adaptation constant this number was given by:

$$\nu_{20} \approx \frac{1.15}{\frac{\alpha_L}{L}\overline{\lambda}_l}. \tag{A.35}$$

Bibliography

[1] S. Thomas Alexander
"Adaptive Signal Processing", Springer Verlag, 1986, ISBN 0 387 96380 4

[2] S.T. Alexander
"Fast Adaptive Filters: A Geometrical Approach"; IEEE ASSP Magazine, oct. 1986, pp 18–28.

[3] S.T. Alexander
"Adaptive Signal Processing, Theory and Applications", Springer–Verlag, 1986

[4] M. G. Amin, F.D. Allen Jr.
"A Generalised Noise Canceller using Orthogonal Transformations", EUSIPCE conference 1988, Grenoble, pp419–422

[5] S.P. Applebaum
"Adaptive Arrays", IEEE Trans. on Ant. and Prop. (special issue on Adaptive Antennas), vol. AP–24, no.5, 1976, pp 585–598

[6] M.R. Asharif, F. Amano, S. Unagami, K. Murano
"Acoustic echo canceller based on Frequency Bin Adaptive Filter", GLOBECOM conference, nov. 15–18, 1987, Tokyo, Japan, pp49.2.1–5

[7] Maurice Bellanger
"Adaptive Digital Signal Processing", Marcel Dekker Inc., 1987, ISBN 0 8247 7784 0

[8] G. Clark, S.K. Mitra, S.R. Parker
"Block Implementation of Adaptive Digital Filters", IEEE Trans. on ASSP, vol. ASSP–29, no.3, june 1981, pp744–752

[9] G. Clark, S.K. Mitra, S.R. Parker
"A Unified approach to Time– and Frequency–Domain Realization of FIR Adaptive Digital Filters", IEEE Trans. on ASSP, vol ASSP–31, no. 5, oct. 1983, pp1073–1083

[10] C.F.N. Cowan, P.M. Grant
"Adaptive Filters", Prentice–Hall, 1985, ISBN 0 013 004037 1 01

[11] R.E. Crochiere, L.R. Rabiner
"Multirate Digital Signal Processing", Prentice–Hall, N.J., 1983, ISBN 0-13-605162-6

[12] P.J. Davis
"Circulant Matrices", New York: Wiley, 1979

[13] N.Q. Duc, B.M. Smith
"Line Coding for Digital Data Transmission", A.T.R. vol. 11, no.2, 1977

[14] G. Egelmeers, P.C.W. Sommen
"Relation between reduced dimension time and frequency domain adaptive algorithm", Proc. EUSIPCO 1992, Brussels

[15] A.Feuer
"Performance Analysis of the Block Least Mean Square Algorithm", IEEE Trans. on CAS, vol. CAS–32, no.9, sept. 1985, pp960–963

[16] Kaoru Furosawa, Takuji Furusawa
"A geometric interpretation of adaptive algorithms"; Globecom conference, Tokyo, Japan, nov. 15–18, 1987, pp 49.7.1–49.7.5

[17] W.A. Gardner
"Learning Characteristics of Stochastic–Gadient–Descent Algorithms: A General study, Analysis, and Critique", Signal Processing 6 (1984) pp113–133, North–Holland

[18] P.J. van Gerwen, N.A.M. Verhoeckx and T.A.C.M. Claasen
"Design Considerations for a 144 kbit/sec Digital Transmission unit for the Local Telephony network", IEEE Journal on selected areas in Communications, vol. SAC–2, no.2, march 1984, pp 314–323

[19] J.D. Gibson
"Adaptive Prediction for Speech Encoding", IEEE ASSP Magazine, vol.1, no.3, 1984, pp12–26

[20] A. Gilloire, M. Vetterli
"Adaptive Filtering in Sub–bands", Proc. ICASSP88, New York, pp 1572–1575

[21] Graham C. Goodwin
"Adaptive Filtering Prediction and Control", Prentice–Hall 1984, ISBN 0 013 004069 - X

[22] R.M. Gray
"Toeplitz and Circulant matrices: A Review", Stanford Electroctronics Laboratories, Tech. Rep. 6502-1, 1971

[23] R.M. Gray
"On the Asymptotic Eigenvalue Distribution of Toeplitz Matrices", IEEE Trans. on IT, vol. IT–18, n0.6, nov. 1972, pp725–730

[24] L.J. Griffiths
"An adaptive lattice structure for noise cancelling applications", Proceedings ICASSP–78, p. 87

[25] C.W.K. Gritton
"Echo Cancellation Algorithms", IEEE ASSP Magazine; april 1984

[26] Simon Haykin
"Adaptive Filter Theory", Prentice Hall, 1986, ISBN 0 13 004052 5 025

[27] Michael L. Honig, David G. Messerschmitt
"Adaptive Filters, Structures, Algorithms and Applications", Kluwer Academic Publishers, 1984, ISBN 0 89838 163 0

[28] O.A. Horna
"Echo control in teleconferencing", Proc. GLOBECOM 1983 (San Diego); pp 16.2.1 – 7

[29] S. Karni, G. Zeng
"An adaptive IIR algorithm with unimodal performance surfaces", IEEE Trans. on ASSP, vol. ASSP 36, no.2, Feb. 1988, pp 286–287

[30] W. Kellermann
"Kompensation akustischer Echos in Frequenzteilbändern", Frequenz, 39, 1985, 7/8, pp209-215

[31] W. Kellermann
"Zur Nachbildung physikalischer Systeme durch parallelisierte digitale Ersatzsysteme im Hiblick auf die Kompensation akustischer Echos", PhD thesis Darmstadt, VDI Verlag Düsseldorf, Reihe 10: Informatik/Kommunikationstechnik Nr. 102

[32] J.L. Lacoume, T.S. Durani and R.Stora
"Signal Processing", course books, Les Houches, 1985

[33] J.C. Lee, C.K. Un
"Block Realization of Multirate Adaptive Filters", IEEE Trans. on ASSP, vol. ASSP–34, no.1, feb. 1986, pp105–117

[34] D. Mansour, A.H. Gray Jr.
"Unconstrained Frequency Domain Adaptive Filter", IEEE Trans. on ASSP, vol ASSP–30, no.5, pp 726–734, oct. 1982

[35] J.D. Markel, A.H. Gray Jr.
"Linear Prediction of speech", Springer Verlag, 1976

[36] S.S. Narayan, A.M. Peterson and M.J. Narasimha
"Transform Domain LMS Algorithm", IEEE Trans on ASSP, vol. ASSP–31, No. 3, june 1983

[37] A.V. Oppenheim, R.W. Schafer
"Digital Signal Processing",Prentice–Hall, Englewood Cliffs, N.J., 1975

[38] Sophocles J. Orfanidis
"Optimum Signal Processing, an introduction", Macmillan Publishing Company, 1984, ISBN 0 02 949860 0

[39] Kazuhiko Ozeki, Tetsuo Umeda
"An adaptive Filtering Algorithm Using an Orthogonal Projection to an Affine Subspace and its properties"; Electronics and Communications in Japan, vol. 67-A, no.5, 1984, pp 19–27
Translated from Denshi Tsushin Gakkai Ronbunshi, vol. 67A, no. 2, Feb. 1984, pp 126–132

[40] Beresford N. Parlett
"The Symmetric Eigenvalue Problem", Prentice–Hall, Inc., Englewood Cliffs, N.J. 07632

[41] S.U.H. Qureshi
"Adaptive Equalization", Proc. IEEE, vol. 73, no.9, sep. 1985, pp 1349–1387

[42] I.S. Reed
"On a Moment Theorem for Complex Gaussian Processes", IRE Transactions on Information Theory, April1962, pp 194–195

[43] M.R. Sambur
"Adaptive Noise Cancelling for Speech Signals", IEEE Trans. on ASSP, vol. ASSP–26, no.5, 1978, pp 419–423

[44] M.R. Schroeder
"Linear Prediction, entropy and signal analysis", IEEE ASSP Magazine, vol.1, no.3, 1984, pp3–11

[45] U. Schültheiss
"Über die Adaption eines Kompensators für akustische Echos", Phd Report 1988, Darmstadt, Fortschritt–berichte, VDI Verlag Düsseldorf, Reihe 10, Informatik Kommunikationstechnik, Nr.90

[46] K.D. Senne
"Adaptive Linear Discrete–Time Estimation", Ph.D. report Stanford University,Technical Report no. 6778–5, SU–68–090, june 1968

[47] John J. Shynk and Richard P. Gooch
"Frequency–Domain Adaptive Pole–Zero Filtering", Proc. IEEE, vol. 73, no. 10, oct. 1985, pp1526–1528

[48] R.J. Sluyter
"Digitization of Speech", Philips Technical Review, vol. 41, 1983/84, no. 7/8

[49] P.C.W. Sommen
"Frequency–Domain Adaptive Filter with an efficient window function", Proceedings of the ICC86 conference, Toronto Canada, june 1986, pp60.6.1–5

[50] P.C.W. Sommen
"On the convergence behaviour of a frequency-domain adaptive filter with an efficient window function", Proc. EUSIPCO 1986, The Hague 1986, pp211-214

[51] P.C.W. Sommen, P.J. van Gerwen, H.J. Kotmans and A.J.E.M. Janssen
"Convergence Analysis of a Frequency-Domain Adaptive Filter with Exponential Power Averaging and Generalized Window Function", IEEE Trans. on CAS, special issue on Adaptive Systems and Applications, vol. Cas–34, no.7, july 1987, pp788–798

[52] P.C.W. Sommen, J.A.K.S. Jayasinghe
"On Frequency Domain Adaptive Filters using the Overlap–Add Method", Proceedings of the ISCAS88 Conference, Espoo, Finland, pp27–30

[53] P.C.W. Sommen, C.J. van Valburg
"Efficient realisation of adaptive filter using an Orthogonal Projection method", Proc. ICASSP 89 conference, Glasgow, Scotland

[54] P.C.W. Sommen, C.J. van Valburg
"Adaptive Filtering Methods with Decorrelation Properties", Philips Nat. Lab. report no.6349, march 1989

[55] P.C.W. Sommen
"On the Orthogonal Projection method for Adaptive Filters", Mathematical Theory of Networks and Systems, edited by M.A. Kaashoek et. al., vol. III, pp283–290, presented at the MTNS conference 1989, Amsterdam

[56] P.C.W. Sommen
"Partitioned Frequency Domain Adaptive Filters", Proc. Asilomar conference on Signals and Systems, 1989, Pacific Grove, California, USA, pp 677–681

[57] P.C.W. Sommen
"On the convergence properties of a Partitioned Block Frequency Domain Adaptive Filter (PBFDAF)", Proc. EUSIPCO 1990, Barcelona, Spain, pp201–204

[58] P.C.W. Sommen, E. de Wilde
" Equal Convergence Conditions for Normal– and Partitioned Frequency Domain Adaptive Filters", Proc. ICASSP92, San Francisco, USA

[59] P.C.W. Sommen, T.A.C.M. Claasen, P.J. van Gerwen, H.J. Kotmans
"Frequency Domain Block Adaptive Filter", Patent, PH 11.794, 20–6–1986

[60] P.C.W. Sommen, C.J. van Valburg
"Adaptief tijddiscreet transversaal filter" (in dutch), Patent, PHN 12955, may 1989

[61] J.S. Soo, K.K. Pang
"A new structure for Block FIR Adaptive Digital Filters", IREEE Conference, sep. 14– 18, Sydney, 1987, pp 364–367

[62] Jian Sien Soo, Khee K. Phang
"A multistep Size (MSS) Frequency Domain Adaptive Filter", IEEE Trans. ASSP, vol. 39, no. 1, jan. 1991, pp 115–121

[63] C.R. South, C.E. Horna, and A.V. Lewis
"Adaptive Filters to improve a loudspeaking telephone", Electron. Lett., vol.15, no.21, pp 673–674, oct. 1979.

[64] Allan O. Steinhardt
"Householder Transforms in Signal Processing", IEEE Magazine, July 1988, pp 4–12

[65] C.P.J. Tzeng
"An Analysis of a Sub–Band Echo Canceller", GLOBECOM conference, 1987, pp49.1.1–4

[66] B.D. van Veen and K.M. Buckley
"Beamforming: A Versatile Approach to Spatial Filtering", IEEE ASSP Magazine, vol. 5, no.2, 1988, pp4–24

[67] N.A.M. Verhoeckx, H.C. van den Elzen, F.A.M. Snijders and P.J. van Gerwen
"Digital Echo Cancellation for Baseband Data Transmission", IEEE Trans. on ASSP, vol. ASSP–27, no.6, dec. 1979, pp 768–781

[68] Bernard Widrow, Samuel D. Stearns
"Adaptive Signal Processing", Prentice Hall, 1985, ISBN 0 013 004029 01

[69] M. Xu, Y. Grenier
"Acoustic Channel Identification", EUSIPCO conference, 1988, Grenoble, pp 1401-1404

[70] Hiroshi Yasukawa, Shoji Shimada, Isao Furukawa
"Acoustic echo canceller with high speech quality"; ICASSP88 conference, Dallas Texas USA, april 6–9, 1988, pp 49.8.1–49.8.4

Samenvatting

In vergelijking met vaste filters hebben adaptieve filters extra (rekenkundige) complexiteit nodig om de coëfficiënten aan te passen volgens een specifiek regelalgoritme. Met zo'n algoritme is het mogelijk om goede resultaten te verkrijgen in een omgeving waar een aantal signaaleigenschappen niet volledig bekend zijn. De convergentieeigenschappen van zo'n adaptief filter worden onder andere gekenmerkt door de snelheid en nauwkeurigheid van het adaptatieproces. In veel praktische situaties is enige a priori informatie beschikbaar van de omgeving en/of de van belang zijnde signaaleigenschappen. Dit proefschrift behandelt enkele *methoden om de beschikbare a priori informatie te gebruiken om hiermee de (rekenkundige) complexiteit te verminderen met behoud van de convergentieeigenschappen.*

De akoestische echo compensator, een typische toepassing van een adaptief filter, was een van de onderwerpen waaraan, in de periode 1984–1989, onderzoek werd gedaan bij de Radio en Data Transmissie groep op het Natuurkundig Laboratorium van Philips. In deze toepassing wordt een spraaksignaal via een akoestisch echopad, van 100–200 msec., als een ongewenst signaal in een microfoon gereflecteerd. Het adaptief filter moet nu een schatting maken van dit ongewenste echosignaal. De belangrijkste problemen van deze toepassing zijn, buiten de lengte van het akoestisch echopad (1000–2000 coëfficiënten), de niet stationariteiten van het spraaksignaal en het tijdvariante karakter van het echopad. Hoewel geen specifiek onderzoeksgebied van dit proefschrift, heeft de akoestische echo compensator toch gediend als motivatie voor de meeste gedeelten uit dit proefschrift. Het meeste materiaal is geplubliceerd in [49]–[60] en [14].

In Hoofdstuk 1 wordt een algemene inleiding gegeven van adaptieve filters en de gebruikte symbolen en notaties worden uitgelegd.

Omdat blok processing technieken een centrale rol vervullen in dit proefschrift, wordt in Hoofdstuk 2 een afleiding en een analyse gegeven van het "Block Normalized Least Mean Squares" (BNLMS) algoritme. Dit BNLMS algoritme maakt een keer per L monsters een aanpassing van alle N adaptieve coëfficiënten. Hierin is L de procesvertraging. In de literatuur wordt vaak beweerd dat dit algoritme, dat met weinig rekenkundige complexiteit gerealiseerd kan worden, slechte convergentieeigenschappen heeft als een gekleurd signaal wordt toegevoegd aan de ingang van het adaptief filter. Uit de analyse

en experimentele resultaten van dit proefschrift volgt dat zowel de statistische eigenschappen van het ingangssignaal als de initialisatie van de adaptieve coëfficiënten de convergentieeigenschappen beïnvloeden. Dit heeft tot gevolg dat bij een gekleurd ingangssignaal de convergentieeigenschappen zowel beter als slechter kunnen worden. Enige a priori kennis van de ingangssignaal eigenschappen kan gebruikt worden om het algoritme zo goed mogelijk te initialiseren.

In Hoofdtsuk 3 wordt aangetoond dat het BNLMS algoritme voor grote filters (de akoestische echo compensator heeft een adaptief transversaal filter nodig van 1000-2000 coëfficiënten) heel efficiënt kan worden uitgevoerd in het frequentie domein. Voor de transformatie tussen het tijd– en frequentie domein wordt hierbij gebruik gemaakt van de "Fast Fourier Transformatie" (FFT) met een lengte van $B = N + L - 1$. Dit is een van de eerste redenen om de akoestische echo compensator in het frequentie domein uit te voeren. In de literatuur zijn, voor vaste filters, twee methoden bekend om deze efficiënte implementatie uit te voeren. Dit zijn de zogenaamde "overlap–save" en de "overlap–add" methode. Een bewering in de literatuur is dat bij adaptieve filters de "overlap–save" procedure met minder FFT's gerealiseerd kan worden dan de "overlap–add" methode. In dit hoofdstuk wordt aangetoond dat dit onjuist is en dat beide methoden bij adaptieve filters met ongeveer dezelfde hoeveelheid rekenkundige complexiteit gerealiseerd kunnen worden.

Statistische eigenschappen van een spraak signaal zijn tijdsafhankelijk. Als in zo'n situatie het BNLMS algoritme voor de aanpassing van de adaptieve coëfficiënten wordt toegepast, dan kunnen de convergentieeigenschappen sterk fluctueren. Voor zo'n geval, en voor vele andere praktische toepassingen, is het wenselijk om het update algoritme zodanig aan te passen dat de convergentieeigenschappen van het adaptief filter onafhankelijk worden van de statistische eigenschappen van het ingangssignaal. Uit de literatuur is bekend dat het relatief eenvoudig is om in het frequentie domein deze decorrelatie uit te voeren. Dit is de tweede motivatie om de akoestische echo compensator in het frequentie domein te realiseren. In Hoofdstuk 4 wordt aangetoond dat decorrelatie kan worden uitgevoerd in het frequentie domein door iedere afzonderlijke frequentie component spectraal te normeren. Met deze methode wordt een benadering gemaakt van de gewenste tijd domein decorrelatie. Eerst wordt in dit hoofdtsuk aangetoond onder welke voorwaarden deze benadering acceptabel is. Toepassing van deze spectrale normalisatie bij het efficiënt uitgevoerde BNLMS algoritme in het frequentie domein, leidt tot het "Block Frequency Domain Adaptive Filter" (BFDAF). Globaal zijn er twee varianten van het BFDAF algoritme bekend. De eerste, die vijf FFTs nodig heeft voor de realisatie, is geïntroduceerd als de "constrained" BFDAF. Bij deze methode wordt de aanpassing van de adaptieve coëfficiënten door een venster zodanig geconditioneerd, dat voldaan wordt aan de voorwaarde, die nodig is voor de "overlap–save" procedure. De tweede methode is de "unconstrained" BFDAF, omdat hierbij het venster niet nodig is. Deze methode kan worden gerealiseerd met slechts drie FFTs. In het hoofdstuk wordt een analyse gegeven van een gegeneraliseerde structuur van de BFDAF, die voor beide methoden kan worden gebruikt. Uit deze analyse volgt dat in het algemeen de "constrained" methode (5 FFTs) betere convergentieeigenschappen heeft dan de "unconstrained" methode (3FFTs). Verder is het bekend dat vele praktische systemen, zoals de akoestische echo compensator, een globaal afnemende impuls responsie hebben. Als deze a priori informatie beschikbaar is, dan kan gebruik gemaakt worden van een efficiënte implementatie van de BFDAF die geraliseerd kan worden met drie FFTs, terwijl de convergentieeigenschappen vergelijkbaar zijn met de BFDAF van vijf FFTs.

Een van de grootste nadelen van signaalbewerking op basis van blokken is de procesvertaging van L monsters (meestal is L in de orde grootte van de filter lengte N). Verder is bij de uitvoering van de decorrelatie in het frequentie domein door spectrale normalisatie, de resolutie van het spectrum gelijk aan het aantal frequentie componenten B. Echter, de statistische eigenschappen van het ingangssignaal, en dus het benodigd aantal frequentie domein delingen, heeft geen enkele relatie met de lengte B. Door nu het originele BFDAF in K kleinere stukken te partitioneren, met $1 \leq K \leq N$, en deze verkregen structuur op een efficiënte manier te implementeren, wordt het "Partitioned Block Frequency Domain Adaptive Filter" (PBFDAF) verkregen, die wordt besproken in Hoofdstuk 5. Deze structuur heeft, in vergelijking met het BFDAF, een gereduceerde procesvertraging. Verder kan eventueel aanwezige a priori informatie van het spectrum van het ingangssignaal gebruikt worden om de rekenkundige complexiteit te reduceren. Dit omdat de decorrelatie bij de PBFDAF methode wordt uitgevoerd met minder dan B delingen.

In Hoofdstuk 6 wordt het adaptief filter probleem beschreven op een geometrische manier. Een generalisatie van dit concept leidt tot de "Block Orthogonal Projection" (BOP) methode. Met deze methode is het mogelijk om het ingangssignaal van een adaptief filter te decorreleren met een $L \times L$ inverse autocorrelatie matrix, met $L \geq 1$. Dit in tegenstelling tot de "Recursive Least Squares" (RLS) methode, die hiervoor een $N \times N$ (inverse) autocorrelatie matrix nodig heeft. Als nu enige a priori informatie bekend is van het ingangssignaal is het mogelijk om met de BOP benadering de dimensie L beter aan te passen op de benodigde dimensie om het ingangssignaal te decorreleren. Omdat verder zowel de BOP als de PBFDAF methode met een gereduceerd aantal vrijheidsgraden het ingangssignaal van het adaptief filter kunnen decorreleren, wordt hun onderlinge relatie ook in Hoofdstuk 6 besproken. Verder is het bekend uit de literatuur dat een spraaksignaal gemodelleerd kan worden met een auto regressief (ar) proces. Een "Efficient Orthogonal Projection" (EOP) algoritme wordt geïntroduceerd om ar–signalen op eenvoudige wijze te decorreleren.

Bedankwoord

Hierbij wil ik iedereen bedanken die op een directe of indirecte manier heeft bijgedragen aan de totstandkoming van dit proefschrift.

Op de eerste plaats alle ex–collega's van het Natuurkundig Laboratorium. Met name Theo Claasen, die mij heeft weten te interesseren voor het signaal bewerkingsgebied, en Piet van Gerwen, die me steeds weer wist uit te dagen met de vraag: "Wat stelt die gekke formule voor?"

Prof. Wim van Bokhoven wil ik bedanken voor het feit dat hij bereid is geweest om in deze, met name voor hem, zeer drukke periode op te treden als eerste promotor. Vooral wil ik hem bedanken voor de ruimte en vrijheid die ik de afgelopen 3 jaren in zijn groep heb gekregen voor het uitoefenen van mijn eigen stuk onderwijs/onderzoek. Mijn tweede promotor, Prof. H.J. Butterweck, wil ik bedanken voor de vele verhelderende discussies die wij hebben gehad tijdens het uitgebreid doorwerken van het eerst concept van dit proefschrift. Verder wil ik alle andere leden van de comissie bedanken voor de vaak zeer nuttige kritiek.

Dan wil ik alle collega's van de vakgroep EEB bedanken voor de prettige werksfeer die ik de afgelopen drie jaar heb ervaren. Speciaal woord van dank aan kamergenoot Gerard Egelmeers, voor het geleverde zinvolle commentaar op het eerste concept van dit proefschrift, en Linda Balvers voor het maken van de tekeningen.

Ik ben mijn ouders ook heel dankbaar voor het feit dat ik altijd de mogelijkheid heb gekregen om zoveel mogelijk "mijn eigen weg" te gaan. Ik denk dat het opdragen van dit boekje aan mijn moeder en mijn onlangs overleden vader, wel het minste is wat ik terug kan doen.

Het vastleggen van een boeiend, maar vooral leuk gedeelte van mijn werkzaamheden in een proefschrift, heb ik als zeer positief ervaren. Dat ik hierbij de afgelopen periode, afgezien van misschien de laatste paar weken, niet als te gespannen heb ervaren, heb ik vooral te danken aan de mate waarin ik op het thuisfront zoveel mogelijk ben ontzien.
Ria, Sjoerd, Jelle en Evi bedankt

Curriculum Vitea

Petrus Christianus Wilhelmus Sommen was born in Ulicoten, the Netherlands, on Februari 17, 1954. He received the Ingenieur degree in electrical engineering from Delft University of Technology, Delft, the Netherlands, in 1981. In the period 1981–1989, he joined the Philips Research Laboratories, Eindhoven, the Netherlands. First (1981–1984) he was engaged in research on CAD for circuit design, and after that (1984–1989) in research on (adaptive) digital signal processing. Since 1989 he is teacher/researcher (UD) in the area of (adaptive) digital signal processing at Eindhoven University of Technology, Eindhoven, the Netherlands.

STELLINGEN

Behorende bij het proefschrift

Adaptive Filtering Methods

On methods to use a priori information in order to reduce complexity while maintaining convergence properties

door P.C.W. Sommen

1. Ten onrechte wordt in de literatuur de suggestie gewekt dat de convergentieeigenschappen van het "Least Mean Square" algoritme slecht zijn als het ingangssignaal gekleurd is.
   (Bron: Bellanger, M.G. (1987). *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker, Inc., New York, ISBN 0-8247-7784-0 (p.130))

2. Bij de analyse van het "Least Mean Square" algoritme worden aannames gedaan waaraan zelden voldaan wordt. De juistheid van de resultaten impliceert niet dat de aannames gerechtvaardigd zijn.
   (Bron: Widrow B., Stearns S.D. (1985), *Adaptive Signal Processing*, Prentice–Hall, Englewood Cliffs, New Yersey 07632, ISBN 0-13-004029-0 (p. 102))

3. De bewering dat, bij de uitvoering van een adaptief filter in het frequentiedomein, voor de toepassing van de "overlap–add" procedure meer Fourier transformaties nodig zijn dan voor de toepassing van de "overlap–save" procedure is onjuist.
   (Bron: G. Clark, S.R. Parker, S.K. Mitra, *A Unified approach to Time- and Frequency–Domain Realization of FIR Adaptive Digital Filters*, IEEE Trans. on ASSP, vol.31, no.5, oct. 1983, pp. 1073–1083)

4. Bij een adaptief filter kan het "tracking" probleem in twee delen worden opgesplitst. Hierop kan, door gebruik te maken van signaalmodellen voor het ingangssignaal, op een adequate manier worden ingespeeld.
   (Bron: Haykin, S. (1986). *Adaptive Filter Theory*, Prentice–Hall, Englewood Cliffs, New Jersey 07632, ISBN 0-13-004052-5 (p. 251))

5. Adaptieve filters die gebruik maken van frequentiedomein technieken zijn, voor veel praktische toepassingen, een goed alternatief voor de "Recursive Least Squares" methode.
(Bron: Dit proefschrift)

6. Bij toepassingen waar een akoestische echo (100–200 msec.) nog steeds een probleem vormt, verdient het aanbeveling om te zoeken naar andere oplossingen dan het parallel schakelen van een adaptieve echo compensator.

7. Onder de weggebruikers belasten de voetganger en de fietser het milieu het minst. In de praktijk wordt dit nog steeds niet voldoende gewaardeerd in de vorm van regelgevingen en voorzieningen ten behoeve van deze weggebruikers.

8. Voor het productierijp maken van een massaproduct is een van de vereisten dat het ontwikkelteam eensgezind aan de slag gaat. Bij het doen van innovatief onderzoek is daarentegen enige mate van eigenwijsheid van de teamleden een vereiste.

9. Een TUE–medewerker die een dienstreis naar het buitenland maakt is van te voren nooit zeker of hij voldoende tegen ongevallen verzekerd is. De "aanbeveling" om op eigen kosten een passende verzekering af te sluiten getuigt van een onzorgvuldig personeelsbeleid.
(Bron: Regeling voor vergoeding van reiskosten voor dienstreizen, p7)

10. Met behulp van vrij eenvoudige apparatuur (PC, DSP) is het in veel gevallen wel degelijk mogelijk om goed en professioneel wetenschappelijk onderzoek te verrichten.
(Bron: Cursor, 1-12-89, p6–7)

11. Het geven van stellingen bij een proefschrift zou niet verplicht moeten zijn.

Piet Sommen
Valkenswaard, 16 juni 1992