Novel Adaptive Signal Processing Techniques for Underwater Acoustic Communications

Teyan Chen

Doctor of Philosophy (Ph.D.)

University of York

Department of Electronics

September 2011

Abstract

The underwater acoustic channel is characterized by time-varying multipath propagation with large delay spreads of up to hundreds of milliseconds, which introduces severe intersymbol interference (ISI) in digital communication system. Many of the existing channel estimation and equalization techniques used in radio frequency wireless communication systems might be practically inapplicable to underwater acoustic communication due to their high computational complexity.

The recursive least squares (RLS)-dichotomous coordinate descent (DCD) algorithm has been recently proposed and shown to perform closely to the classical RLS algorithm while having a significantly lower complexity. It is therefore a highly promising channel estimation algorithm for underwater acoustic communications. However, predicting the convergence performance of the RLS-DCD algorithm is an open issue. Known approaches are found not applicable, as in the RLS-DCD algorithm, the normal equations are not exactly solved at every time instant and the sign function is involved at every update of the filter weights. In this thesis, we introduce an approach for convergence analysis of the RLS-DCD algorithm based on computations with only deterministic correlation quantities.

Equalization is a well known method for combatting the ISI in communication channels. Coefficients of an adaptive equalizer can be computed without explicit channel estimation using the channel output and known pilot signal. Channel-estimate (CE) based equalizers which re-compute equalizer coefficients for every update of the channel estimate, can outperform equalizers with the direct adaptation. However, the computational complexity of CE based equalizers for channels with large delay spread, such as the underwater acoustic channel, is an open issue. In this thesis, we propose a low-complexity CE based adaptive linear equalizer, which exploits DCD iterations for computation of equalizer coefficients. The proposed technique has as low complexity as $\mathcal{O}(N_u(K+M))$ operations per sample, where K and M are the equalizer and channel estimator length, respectively, and N_u is the number of iterations such that $N_u \ll K$ and $N_u \ll M$. Moreover, when using the RLS-DCD algorithm for channel estimation, the computation of equalizer coefficients is multiplication-free and division-free, which makes the equalizer attractive for hardware design. Simulation results show that the proposed adaptive equalizer performs close to the

minimum mean-square-error (MMSE) equalizer with perfect knowledge of the channel.

Decision feedback equalizers (DFEs) can outperform LEs, provided that the effect of decision errors on performance is negligible. However, the complexity of existing CE based DFEs normally grows squarely with the feedforward filter (FFF) length K. In multipath channels with large delay spread and long precursor part, such as in underwater acoustic channels, the FFF length K needs to be large enough to equalize the precursor part, and it is usual that $K > M$. Reducing the complexity of CE based DFEs in such scenarios is still an open issue. In this thesis, we derive two low complexity approaches for computing CE based DFE coefficients. The proposed DFEs operate together with partial-update channel estimators, such as the RLS-DCD channel estimator, and exploit complex-valued DCD iterations to efficiently compute the DFE coefficients. In the first approach, the proposed DFE has a complexity of $\mathcal{O}(N_u l \log_2 2l)$ real multiplications per sample, where l is the equalizer delay and N_u is the number of iterations such that $N_u \ll l$. In the second proposed approach, DFE has a complexity as low as $\mathcal{O}(N_u K) + \mathcal{O}(N_u B) + \mathcal{O}(N_u M)$ operations per sample, where B is the feedback filter (FBF) length and $N_u \ll M$. Moreover, when the channel estimator also exploits the DCD iterations, e.g. such as in the RLS-DCD adaptive filter, the second approach is multiplication-free and division-free, which makes the equalizer attractive for hardware implementation. Simulation results show that the proposed DFEs perform close to the RLS CE based DFE, where the CE is obtained using the classical RLS adaptive filter and the equalizer coefficients are computed according to the MMSE criterion.

Localization is an important problem for many underwater communication systems, such as underwater sensor networks. Due to the characteristics of the underwater acoustic channel, localization of underwater acoustic sources is challenging and needs to be accurate and computationally efficient. The matched-phase coherent broadband matched-field (MF) processor has been previously proposed and shown to outperform other advanced broadband MF processors for underwater acoustic source localization. It has been previously proposed to search the matched phases using the simulated annealing, which is well known for its ability for solving global optimization problems while having high computational complexity. This prevents simultaneous processing of many frequencies, and thus, limits the processor performance. In this thesis, we introduce a novel iterative technique based on coordinate descent optimization, the phase descent search (PDS), for searching the matched phases. We show that the PDS algorithm obtains matched phases similar to that obtained by the simulated annealing, and has significantly lower complexity. Therefore, it enables to search phases for a large number of frequencies and significantly improves the processor performance. The proposed processor is applied to experimental data for locating a moving acoustic source and shown to provide accurate localization of the source well matched to GPS measurements.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Dr. Yuriy Zakharov for his constant support of my Ph.D research, for his patience, enthusiasm, and immense knowledge in signal processing. His guidance helped me in all the time of my research and writing of this thesis. It is an honor for me to have such a great supervisor for my Ph.D study.

I would like to acknowledge Chunshan Liu for his support and helpful discussions.

I would also like to thank all my colleagues in the Communications Research Group for their kind support during my Ph.D study.

This thesis is dedicated to my parents, who brought me up with their endless love, gave me the freedom to learn, encouraged me to pursue this degree and constantly support me in everything I do; to my wife for her unconditional love, understanding, encouragement and her belief in me; to my son, who gave me a lot of happy moments during my study; and to my parents-in-law for their understanding and support.

Especially, I would like to dedicate this thesis to the memory of my grandmother, whom I love and miss and will never forget.

Declaration

Some of the research presented in this thesis has resulted in some publications. These publications are listed at the end of Chapter 1.

All work presented in this thesis as original is so, to the best knowledge of the author. References and acknowledgements to other researchers have been given as appropriate.

Glossary

AR	**A**uto-**R**egressive
ASSA	**A**daptive **S**implex **S**imulated **A**nnealing
AWGN	**A**dditive **W**hite **G**aussian **N**oise
BER	**B**it-**E**rror-**R**ate
BPSK	**B**inary **P**hase-**S**hift **K**eying
CD	**C**oordinate **D**escent
CE	**C**hannel-**E**stimate
CG	**C**onjugate **G**radient
CW	**C**onstant-**W**ave
CTD	**C**onductivity, **T**emperature, and **D**epth
DA	**D**irect **A**daptation
dB	Decibel
DCD	**D**ichotomous **C**oordinate **D**escent
DFE	**D**ecision **F**eedback **E**qualizer
ERLS	**E**xponentially weighted **R**ecursive **L**east **S**quares
FFF	**F**eedforward **F**ilter
FBF	**F**eedback **F**ilter
FFT	**F**ast **F**ourier **T**ransform
GPS	**G**lobal **P**ositioning **S**ystem
Hz	**H**ertz
ISI	**I**nter-**S**ymbol **I**nterference
LE	**L**inear **E**qualizer
LMS	**L**east **M**ean **S**quare
MF	**M**atched-**F**ield
MFP	**M**atched-**F**ield **P**rocessing
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
MLSE	**M**aximum **L**ikelihood **S**equence **e**stimation
MMSE	**M**inimum **M**ean **S**quare **E**rror
MSD	**M**ean **S**quare **D**eviation
MSE	**M**ean **S**quare **E**rror
PDS	**P**hase **D**escent **S**earch
QPSK	**Q**uadrature **P**hase-**S**hift **K**eying

RLS	**R**ecursive **L**east **S**quares
RLS-DCD	**R**ecursive **L**east **S**quares-**D**ichotomous **C**oordinate **D**escent
SNR	**S**ignal to **N**oise **R**atio
VLA	**V**ertical **L**ine **A**rray

Chapter 1

Introduction

Contents

1.1 Overview

The underwater acoustic channel is considered to be one of the most challenging communication media in use today [1]. It is characterized by many factors which prevent implementation of high-speed and reliable communications, such as time-varying multipath propagation with large delay spreads of up to hundreds milliseconds [1–3]. In digital communication systems, the effect of multipath propagation with large delay spreads is severe intersymbol interference (ISI) that can extend from over several tens to several hundreds of symbol periods. This makes many of the techniques widely used in radio frequency wireless communication systems practically inapplicable to underwater acous-

tic communications, from a computational complexity point of view. For example, the classical recursive least squares (RLS) channel estimator [4, 5] has a computational complexity that grows squarely with the delay spread of the channel (channel length), and the MMSE channel-estimate (CE) based decision-feedback equalizers (DFEs) [6, 7] have a computational complexity that grows at least squarely with the feedforward filter (FFF) length which normally needs to be greater than the channel length for channels with a long precursor part. In this thesis, we investigate low-complexity channel estimation and CE based equalization techniques to overcome the ISI problem in underwater acoustic communications. Localization is important for underwater acoustic communications in many aspects, such as time synchronization, underwater sensor networks [8–10] and networking protocols [11]. In underwater sensor networks, distributed sensors are used to collect specific data. However, the collected data can be meaningless if the location of the sensor is unknown. Although many techniques have been developed for terrestrial localization, most of these techniques cannot be applied directly to underwater acoustic source localization mainly due to the variable speed of sound in underwater, and the unavoidable movement of sensors. Moreover, due to the large propagation delay in the underwater acoustic channel and the limited computational power of sensors, localization needs to be accurate and computationally efficient. In this thesis, we also investigate low-complexity techniques for underwater acoustic source localization.

Although the underwater acoustic channel exhibits large delay spreads, it normally has a limited number of multipath components. This enables the use of sparse channel estimation techniques [12–15] which have low computational complexity when comparing with the classical channel estimation techniques, such as the RLS channel estimator. The dichotomous coordinate descent (DCD) algorithm has been proposed to efficiently solve the linear least-squares problem without involving any multiplications nor divisions. It is even more efficient when the expected solution vector is sparse [16]. In [17], low-complexity RLS algorithms using DCD iterations have been proposed and shown by empirical analysis to perform closely to the classical RLS algorithm. The RLS-DCD channel estimator is therefore a highly promising candidate for underwater acoustic communications. However, predicting the convergence performance of the RLS-DCD algorithms is still an open issue. Traditional methods for convergence analysis of the RLS algorithms [4] are difficult to apply, since, in the RLS-DCD algorithm, the normal equations are not exactly solved at every time instant and the sign function is involved at every update of the weights. A

general framework for analysis of adaptive filtering algorithms is introduced in [18, 19]. However, in the RLS-DCD algorithm, due to multiple iterations at each time instant, it is difficult to represent the RLS-DCD algorithm using such a framework. A statistical analysis of the affine projection algorithm proposed in [20] is based on some statistical properties of a residual vector. However, such a residual vector does not exist in the RLS algorithms. It is desirable to find a new approach for convergence analysis of the RLS-DCD algorithm, and other adaptive algorithms based on multiple iterations at a single time instant.

Equalization is a well known method for combatting the ISI in communication channels [21]. Coefficients of an adaptive equalizer can be computed without explicit channel estimation using the channel output and known pilot signal [21]. However, CE based equalizers which re-compute equalizer coefficients for every update of the channel estimate, can outperform equalizers with the direct adaptation [22]. In CE based adaptive linear equalizers (LEs), computation of equalizer coefficients normally requires generation and inversion of a $K \times K$ channel autocorrelation matrix, where K is the equalizer length. In general, it results in a complexity of $\mathcal{O}(K^3)$ operations per sample. Exploiting structural properties of the matrix, the complexity can be reduced down to $\mathcal{O}(K^2)$ operations [7]. For channel estimation, the RLS adaptive filtering algorithms [4] which are known to possess fast convergence, have a complexity of $\mathcal{O}(M^2)$ operations per update, where M is the channel estimator length. It is usual that $K > M$, thus the complexity of computing the equalizer coefficients determines the total complexity. Moreover, the RLS-DCD algorithms [17] only require a complexity of $\mathcal{O}(N_u M)$ operations per sample, where $N_u << M$. Thus, adaptive channel estimation can be significantly simpler than CE based computation of equalizer coefficients. To reduce the whole complexity, the computation of equalizer coefficients should be simplified.

It is known that DFEs can outperform LEs, provided that the effect of decision errors on performance is negligible [21]. However, the computational complexity of CE based DFEs can be higher than that of CE based LEs. Extensive effort has been made to reduce the complexity of computing the DFE taps (see [6, 7, 23–30] and references therein). However, the complexity normally grows squarely with the FFF length K. In multipath channels with large delay spread and long precursor part, such as in underwater acoustic channels [31], the FFF length K needs to be large enough to equalize the precursor part,

and it is usual that $K > M$. Reducing the complexity of CE based DFEs in such scenarios is still an open issue.

For underwater source localization, broadband (or multi-frequency) MFP has been actively investigated in the past two decades [32–39]. It is found that coherent combining of ambiguity surfaces obtained at different frequencies provides better performance compared to incoherent combining [38]. In scenarios where an acoustic source transmits sound at multiple frequencies, phases of the source frequencies contribute in the measured acoustic data. The phase shifts between different frequencies are often unknown and need to be compensated. A matched-phase coherent processor proposed in [38] compensates for these phase shifts and has been shown to outperform other advanced MF processors, especially when the ambient noise level and environment mismatch are significant [38]. In [38], it is proposed to search the phase shifts by using the simulated annealing algorithm, which is well known for its ability of solving global optimization problems while having high computational complexity. Although different approaches have been proposed to reduce the complexity [40, 41], it is still computationally consuming and increases dramatically as the number of free parameters increases. This prevents simultaneous processing of many frequencies, and thus, limits the processor performance. Furthermore, for most of the simulated annealing methods, it is found to be exhausting to determine some algorithm parameters such as the initial temperature and the cooling schedule, which need to be carefully set. Reducing the complexity of searching the matched phases for the matched-phase coherent processor is highly desirable.

1.2 Objectives

This research aims to reduce the computational complexity of signal processing techniques for underwater acoustic communications by using iterative techniques, such as the DCD algorithm. We start with a convergence analysis of the RLS-DCD algorithm, which will be used for channel estimation in the equalizers that we derive in this thesis. We then focus on developing low-complexity CE based adaptive LE and DFE which exploit DCD iterations for computation of equalizer coefficients. We also investigate the application of the matched-phase coherent MF processor for underwater source localization. Specifi-

cally, we are interested in reducing the complexity of searching the matched phases for the matched-phase coherent MF processor, by applying a DCD based phase search algorithm, the phase descent search (PDS) algorithm.

1.3 Notations

In this thesis, we use capital and small bold fonts to denote matrices and vectors, respectively; e.g. $\mathbf{G}$ is a matrix and $\mathbf{r}$ a vector. Elements of the matrix and vector are denoted as $G_{n,p}$ and r_n, respectively. A pth column and nth row of $\mathbf{G}$ are denoted as $\mathbf{G}^{(p)}$ and $\mathbf{G}_{(n)}$, respectively. We also denote: $\mathbf{r}^T$ and $\mathbf{G}^T$ are transpose of the vector $\mathbf{r}$ and matrix $\mathbf{G}$, respectively; $\mathbf{G}^H$ is conjugate transpose of matrix $\mathbf{G}$; $\mathbf{r}^*$ is the complex conjugate of the vector $\mathbf{r}$; $\mathbf{I}_K$ is a $K \times K$ identity matrix; $\mathbf{0}_{K\times M}$ is a $K \times M$ matrix of all zeros; $E\{\cdot\}$ is the expectation; tr$\{\cdot\}$ denotes the trace operator; $\Re\{\cdot\}$ and $\Im\{\cdot\}$ are the real and imaginary part of a complex number, respectively. The variable n is used as a time index and i is iteration index. The symbol j is an imaginary unit $j = \sqrt{-1}$.

1.4 Fundamental techniques

In this section, we first briefly discuss the existing techniques for solving the normal equations. Fundamental techniques used throughout this thesis are then introduced. These are: DCD algorithm; time-varying channel models; and time-varying underwater acoustic channel model.

1.4.1 Solving normal systems of equations

Many of the signal processing techniques for communications require solving the linear least-square (LS) problem in real time, such as channel estimation [4], equalization [21] and adaptive array processing [5]. It is known that solving the LS problem is equivalent to solving a system of linear equations, called the normal equations $\mathbf{Rh} = \boldsymbol{\beta}$, where $\mathbf{R}$ is an

$M \times M$ symmetric positive definite matrix and both $\mathbf{h}$ and $\boldsymbol{\beta}$ are $M \times 1$ column vectors. The matrix $\mathbf{R}$ and the vector $\boldsymbol{\beta}$ are known, whereas the vector $\mathbf{h}$ needs to be estimated. Techniques for solving the normal equations is mainly divided into two categories: direct methods and iterative methods.

Direct methods for solving the normal equations, such as Gaussian elimination, LU decomposition, Cholesky decomposition and QR decomposition, find an exact solution through a finite number of pre-specified operations [42]. The direct methods can only provide solutions after the pre-specified operations. Moreover, they normally involve divisions and multiplications, and have a complexity of $\mathcal{O}(M^3)$ operations [42]. Therefore, the direct methods are too complex for real-time implementation, especially when solving the very large or very sparse systems of linear equations.

Iterative methods solve the normal equations iteratively, and at each iteration, they find better approximations to the optimal solution [42]. Comparing with the direct methods, the iterative methods have lower complexity, and is easier for real-time implementation [42]. The iterative methods are also known to be more efficient than the direct methods when solving both very large and very sparse systems of linear equations [42]. Moreover, the iterative methods have the ability to use a good initial guess of the solution, which may reduce the computational complexity.

The iterative methods can be further divided into two types: stationary methods and non-stationary methods. Stationary methods, such as Jacobi and Gauss-Seidel methods [42], normally have a complexity as low as $\mathcal{O}(M)$ operations per iteration. However, their convergence speed is usually much slower than that of the non-stationary methods [43]. Non-stationary methods, such as conjugate gradient (CG) and coordinate descent (CD) algorithms, possess fast convergence, but have a high complexity of $\mathcal{O}(M^2)$ operations per iteration. These algorithms also involve divisions and multiplications, which make them expensive for real-time implementation. The DCD algorithm as a non-stationary iterative technique, performs a similar convergence speed to the CG and CD algorithms, while it does not require any multiplication or division, and has a complexity as low as $\mathcal{O}(M)$ additions per successful iterations [16].

1.4.2 DCD algorithm

The DCD algorithm with a leading element [17] which solves the system of normal equations, $\mathbf{Rh} = \boldsymbol{\beta}$, is presented in Table 1.1. In Table 1.1, the DCD algorithm finds a 'leading' (pth) element in the solution vector $\mathbf{h}$ to be updated according to an element of a residual vector $\mathbf{r}$, which has the largest absolute value. The step-size α is chosen from one of M_b predefined values, which correspond to binary representation of elements of the solution vector $\mathbf{h}$ with M_b bits within an amplitude range $[-H, H]$, where H is preferably a power-of-two number. The step-size $\alpha = 2^{-m}H$ is therefore also a power-of-two number. With such settings, operations required in the DCD algorithm are only additions as all multiplications and divisions are replaced by bit-shifts. Due to the quantized step-size, there are 'unsuccessful' iterations (decided at step 4) without updates of the solution and 'successful' iterations where the solution and the residual vector are updated (steps 5 and 6). With N_u successful iterations, the complexity of the DCD algorithm is upper limited by $(2M + 1)N_u + M_b$ additions, which corresponds to a worst-case scenario when the condition at step 3 is never satisfied.

The DCD algorithm has been widely used for real-time implementation of some adaptive filtering algorithms, such as the affine projection algorithm and the RLS algorithm [17, 44–47]. It is multiplication-free and division-free, and has significantly lower complexity than the Cholesky decomposition and other known techniques. It is therefore attractive for hardware implementation and has been implemented on FPGA and DSP platforms [48–53]. In this thesis, the DCD algorithm will be considered and applied to different signal processing techniques for underwater communications.

1.4.3 Time-varying Rayleigh fading channel models

Two time-varying Rayleigh fading channel models are used in the subsequent chapters: first order autoregressive (AR) model and modified Jakes' model.

For modeling the time-varying channel impulse response $\mathbf{h}(n)$ with length M, the first order AR model $\mathbf{h}(n) = \sqrt{\upsilon}\ \mathbf{h}(n-1) + \sqrt{1-\upsilon}\ \boldsymbol{\omega}(n)$ is used [54], where $\sqrt{\upsilon}$ is the autoregressive factor and $\boldsymbol{\omega}(n)$ are zero-mean independent random Gaussian vectors,

Table 1.1: DCD algorithm for solving a system of normal equations $\mathbf{Rh} = \boldsymbol{\beta}$

Step	Equation	$+$
	Initialization: $\mathbf{h} = 0$, $\mathbf{r} = \boldsymbol{\beta}$, $\alpha = H/2$, $m = 1$	
	for $k = 1, \ldots, N_u$	
1	$p = \arg\max_{n=1,\ldots,M}\{\lvert r_n \rvert\}$, go to step 4	$M-1$
2	$m = m + 1$, $\alpha = \alpha/2$	
3	if $m > M_b$, the algorithm stops	
4	if $\lvert r_p \rvert \leq (\alpha/2)R_{p,p}$, then go to step 2	1
5	$h_p = h_p + \text{sign}(r_p)\alpha$	1
6	$\mathbf{r} = \mathbf{r} - \text{sign}(r_p)\alpha\mathbf{R}^{(p)}$	M
	Total: $\leq (2M+1)N_u + M_b$ adds	

whose elements have variance $1/M$. Jakes' model [55] as a simplified version of Clarkes model [56] has been widely used for modeling time-varying Rayleigh fading channels. In this thesis, we adopt the modified Jakes' model proposed in [57] for modeling the time-varying channel impulse response.

1.4.4 Time-varying underwater acoustic channel model

In this thesis, we also employ the simulator recently proposed in [58, 59] for modeling underwater acoustic signals propagating through a time-varying multipath underwater acoustic channel caused by transmitter and/or receiver motions. In this simulator, the movement trajectory is sampled at a low rate and the 'waymark' impulse responses are computed at these sampling instances by solving the wave propagation equation. Some well developed programs, such as the normal mode method KRAKEN and the ray tracing method BELLHOP [60–62] are used for solving the wave equation. The waymark impulse responses are then interpolated in time using local B-splines [63] to obtain impulse responses for each sampling instant of the source signal.

1.5 Contributions

Major contributions in this thesis can be summarized as follows:

- An approach for convergence analysis of the RLS-DCD adaptive filtering algorithm based on computations with only deterministic quantities has been derived. Deterministic expressions for time dependent correlation quantities have been obtained without involving any stochastic processes and used to form the normal equations. Deterministic equations for evaluating the predicted MSE and MSD learning curves of the RLS-DCD algorithm have been derived. Simulation results show good agreement between the predictions and the practical learning curves.

- A low-complexity CE based adaptive linear equalizer has been derived, which exploits DCD iterations for computation of equalizer coefficients. It has been shown that, when using the RLS-DCD algorithm for channel estimation, the computation of equalizer coefficients is multiplication-free and division-free, which makes the equalizer attractive for hardware design. The performance of the proposed adaptive equalizer over the channels with large delay spreads are shown to be close to that of the MMSE equalizer with perfect knowledge of the channel.

- Two partial-update CE based adaptive DFEs have been proposed, both of which can operate together with partial-update channel estimators and exploit complex-valued DCD iterations to efficiently compute the DFE taps. The first proposed DFE has been implemented not only in the conventional structure, for which a simple recursive method has been derived for computing the FBF taps, but also in the modified structure which does not require computing the FBF. The second proposed DFE is derived and implemented in the conventional structure with even lower computational complexity. It has been shown that, when using the channel estimator which also exploits the DCD iterations, all multiplications involved in computation of the equalizer taps can be replaced by bit-shift operations. The proposed approach for computing the linear equalizer coefficients has been extended to the complex-valued case. The proposed DFEs have been applied to different time-varying channels with small and large delay spreads and shown to perform very close to the RLS CE based DFE, where the CE is obtained using the classical RLS adaptive filter and the equalizer taps are computed according to the MMSE criterion.

- The complex-valued DCD iterations and the complex-valued RLS-DCD adaptive filtering algorithm for channel estimation have been introduced.

- One of the most advanced underwater acoustic channel simulators recently proposed has been used to model acoustic signals propagating through a time-varying multipath underwater acoustic channel caused by transmitter motion. The later proposed DFE has been applied to the time-varying underwater acoustic channel and shown to perform very close to the RLS CE based DFE.

- The phase decent search (PDS) algorithm has been introduced to the matched-phase coherent broadband matched-field (MF) processor for searching the matched phases. The proposed PDS algorithm has been compared with simulated annealing algorithm and shown to have significantly lower complexity, which enables simultaneous processing of many frequencies and improves processor performance. The proposed processor has been applied to experimental data for source localization. The proposed processor has been shown to have better performance when processing more frequencies. The estimated range trajectory is obtained by using the proposed processor and shown to be well matched to GPS measurements.

1.6 Thesis Outline

The rest of this thesis is organized into the following chapters.

- Chapter 2: Convergence Analysis of RLS-DCD Algorithm

  In this chapter, the RLS-DCD algorithm is briefly introduced. We then derive an approach for convergence analysis of the RLS-DCD algorithm based on computations with only deterministic quantities obtained from the second order statistics. Finally, we compare the practical MSE and MSD learning curves of the RLS-DCD algorithm with the predictions obtained by using the proposed approach.

- Chapter 3: Low-complexity channel-estimate based adaptive linear equalization

  In this chapter, the data models and the normal equations for computing the MMSE LE coefficients are firstly given. We then introduce the assumptions that we use in our derivation, and derive a low-complexity approach for computing the LE coefficients. Simulation results which compare the performance and computational complexity of the proposed LE against known techniques are also presented.

- Chapter 4: Partial-update channel-estimate based adaptive decision feedback equalizer: Approach 1

  In this chapter, we start with introducing the structure of the conventional DFE and giving expressions for computing the DFE taps. Assumptions made for deriving the partial-update DFE are then given. We also introduce the complex-valued DCD iterations and RLS-DCD algorithm for adaptive channel estimation. We derive a low-complexity approach for computing the FFF taps and recursive computation of the FBF taps. The modified DFE structure is also introduced. We finally present numerical results that demonstrate the performance and computational complexity of the proposed DFE against known techniques.

- Chapter 5: Partial-update channel-estimate based adaptive decision feedback equalizer: Approach 2

  In this chapter, we derive another partial-update DFE which has even lower computational complexity. We apply the proposed DFE to the underwater acoustic channel and present numerical results that demonstrate the performance and computational complexity of the proposed DFE against known techniques.

- Chapter 6: Matched-phase coherent broadband matched-field processor using phase descent search

  In this chapter, the matched-phase coherent MF processor and the cross-frequency incoherent processor are reviewed. We then introduce the PDS algorithm to the match-phase coherent MF processor for searching the matched phases, and the frequency estimator based on the dichotomous search of the periodogram peak for estimating the compression factor in the experiment. We apply the proposed processor to experimental data, and compare the localization performance and computational complexity of the PDS algorithm against simulated annealing algorithm.

1.7 Publication List

Journal Papers [59, 64–66]

1. T. Chen, Y. V. Zakharov, and C. Liu, “Source localization using matched-phase

matched-field processing with phase descent search", under revision by IEEE Journal on Oceanic Engineering, 2010.

2. T. Chen, Y. V. Zakharov, and C. Liu, "Low-complexity channel-estimate based adaptive linear equalizer", IEEE Signal Processing Letters, vol. 18, no. 7, pp. 427-430, 2011.

3. C. Liu, Y. V. Zakharov, and T. Chen, "Doubly-selective underwater acoustic channel model for moving transmitter/receiver", under revision by IEEE Transactions on Vehicular Technology, 2011.

4. T. Chen, Y. V. Zakharov, and C. Liu, "Partial-update channel-estimate based adaptive decision feedback equalizer", under revision by IEEE Transactions on Signal Processing, 2011.

Conference Papers [58, 67–71]

1. T. Chen and Y. Zakharov, "Convergence analysis of RLS-DCD algorithm", in IEEE/SP 15th Workshop on Statistical Signal Processing, SSP09. IEEE, 2009, pp. 157C160.

2. T. Chen, C. Liu, and Y. V. Zakharov, "Matched-phase coherent broadband matched-field processor using phase descent search", in Tenth European Conference on Underwater Acoustics, ECUA, 2010, pp. 590-595, Istanbul, Turkey.

3. C. Liu, T. Chen, and Y. V. Zakharov, "Source localization using sparsity based iterative adaptive beamforming", in Tenth European Conference on Underwater Acoustics, ECUA, 2010, pp. 604-610, Istanbul, Turkey.

4. C. Liu, T. Chen, and Y. V. Zakharov, "Matched field inversion for sound speed profile in a deep water environment by using simplex simulated annealing", in Tenth European Conference on Underwater Acoustics, ECUA, 2010, pp. 597-602, Istanbul, Turkey.

5. C. Liu, T. Chen, and Y.V. Zakharov, "Broadband underwater source localization by solving basis pursuit de-noising using coordinate descent search", in Wireless Communication Systems (ISWCS), 2010 7th International Symposium on. IEEE, pp. 1-5.

6. C. Liu, Y. V. Zakharov, and T. Chen, “Modeling of time-varying underwater acoustic channels”, in Proceedings of the 4th Int. Conf. “Underwater acoustic measurements: Technologies and Results”, Kos, Greece, 20-24 June 2011, pp. 1423-1430.

Chapter 2

Convergence Analysis of RLS-DCD Algorithm

Contents

The recursive least squares (RLS)-dichotomous coordinate descent (DCD) algorithm introduced in [17] for adaptive filtering is characterized by low complexity, while possessing fast convergence. However, predicting the convergence performance of the RLS-DCD algorithm is still an open issue. Known approaches are found not applicable, as in the RLS-DCD algorithm, the normal equations are not exactly solved at every time instant and the sign function is involved at every update of the filter weights. In this chapter, we propose an approach for convergence analysis of the RLS-DCD algorithm based on computations with only deterministic correlation quantities. This new approach can be also used for other adaptive filtering algorithms based on iterative solving the normal equations.

This chapter is organized as follows. In the next section, an introduction is given. In

Section 2.2, the data models and the RLS-DCD algorithm are briefly introduced. Section 2.3 presents the derivation of the proposed approach for convergence analysis of the RLS-DCD algorithm. Simulation results which compare practical leaning curves and the predictions are given in Section 2.4. Section 2.5 finally draws conclusions.

2.1 Introduction

The classical RLS algorithm is well known for its fast convergence. However, it has a high computational complexity [4, 5]. The RLS-DCD algorithm is characterized by low complexity, while possessing fast convergence. Empirical analysis of the RLS-DCD algorithm has been presented in [17] to show that its performance can be made very close to that of the RLS algorithm. However, predicting the convergence performance of the RLS-DCD algorithms is still an open issue. In this chapter, we propose an approach for convergence analysis of this adaptive algorithm based on computations with deterministic quantities derived from the second order statistics.

Traditional methods for convergence analysis of the exponentially weighted RLS (ERLS) algorithm presented in [4] are difficult to apply, since, in the RLS-DCD algorithm, the normal equations are not exactly solved at every time instant and the sign function is involved at every update of the weights. A general framework for analysis of adaptive filtering algorithms introduced in [18] is specified by a generic filter-weight update equation, and correspondence between special cases of this equation and various adaptive filtering algorithms. The subsequent transient analysis of adaptive filters based on this framework is proposed in [19]. However, in the RLS-DCD algorithm, due to multiple iterations at each time instant, it is difficult to derive such an update equation. Furthermore in [19], estimates of some quantities are obtained from a *single realization* of signals involved in the adaptive filtering, *i.e.* stochastic signals are involved. We want to use only statistical characteristics of signals in our analysis. A statistical analysis of the affine projection (AP) algorithm for a unity step size and autoregressive inputs is proposed in [20] based on some statistical properties of a residual vector. However, such a residual vector does not exist in the RLS algorithm. It is desirable to find a new approach for convergence analysis of the RLS-DCD algorithm, and other adaptive algorithms based

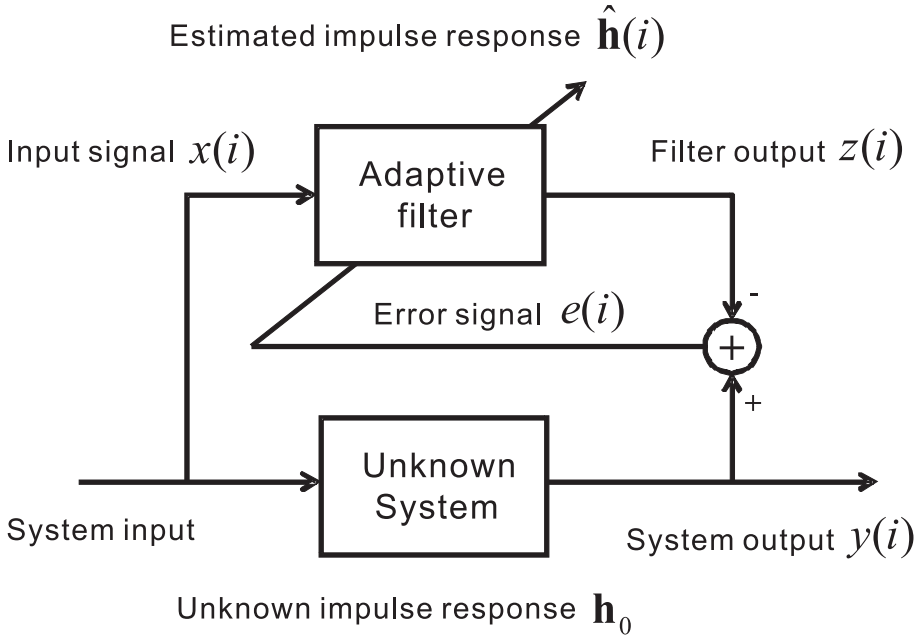


Figure 2.1: Adaptive filtering for identification scenario

on multiple iterations at a single time instant.

2.2 Data models

We consider the application of adaptive filtering for a general identification scenario as shown in Figure 2.1, in which an adaptive filter is used to estimate the impulse response of an unknown system. We consider that the unknown system output (the desired response) $y(i)$ and the M-length input data vector $\mathbf{x}(i)$ are related by the *multiple linear regression model* [4]:

$$y(i) = \mathbf{x}^T(i)\mathbf{h}_0 + \nu(i), \tag{2.1}$$

where $\mathbf{x}(i) = [x(i)\, x(i-1) \ldots\, x(i-M+1)]^T$, $\mathbf{h}_0$ is the unknown impulse response that we want to estimate, and $\nu(i)$ is the measurement noise; the vector $\mathbf{h}_0$ is constant, the measurement noise $\nu(i)$ is white with zero mean and variance σ_ν^2, and the $M \times 1$ data vector $\mathbf{x}(i)$ has a positive-definite covariance matrix $\mathbf{R}_{\mathbf{xx}}$, given by

$$\mathbf{R}_{\mathbf{xx}} = E\left\{\mathbf{x}(i)\mathbf{x}^T(i)\right\}.$$

The adaptive filter adjusts its estimate of the impulse response $\hat{\mathbf{h}}(i)$ in such a way to minimize the error signal $e(i) = y(i) - z(i)$, where $z(i) = \mathbf{x}^T(i)\hat{\mathbf{h}}(i)$. Most often,

an adaptive algorithm adjusts its estimate to minimize the mean square error (MSE) $E\left\{|e(i)|^2\right\} = E\left\{\left|y(i) - \mathbf{x}^T(i)\hat{\mathbf{h}}(i)\right|^2\right\}$. According to the orthogonality principle [5], the error signal $e(i)$ should be orthogonal to the input data vector $\mathbf{x}(i)$, which gives $E\left\{\mathbf{x}(i)\left[y(i) - \mathbf{x}^T(i)\hat{\mathbf{h}}(i)\right]\right\} = \mathbf{0}$. Therefore, solving the MSE minimization problem is equivalent to solving the normal equations $\mathbf{R}_{\mathbf{xx}}\mathbf{h}(i) = \boldsymbol{\beta}_{\mathbf{x}y}$, where $\boldsymbol{\beta}_{\mathbf{x}y} = E\left\{\mathbf{x}(i)y(i)\right\}$.

In the RLS problem, at every time instant i, an adaptive algorithm should find a solution to the normal equations

$$\mathbf{R}(i)\mathbf{h}(i) = \boldsymbol{\beta}(i), \tag{2.2}$$

where $\mathbf{R}(i)$ and $\boldsymbol{\beta}(i)$ are an instantaneous autocorrelation matrix of the filter input signal and instantaneous crosscorrelation vector between the input signal and the desired signal, respectively. $\mathbf{R}(i)$ is assumed to be a symmetric positive-definite matrix of size $M \times M$, $\boldsymbol{\beta}(i)$ and $\mathbf{h}(i)$ are M- length vectors. The matrix $\mathbf{R}(i)$ and vector $\boldsymbol{\beta}(i)$ are known, whereas the vector $\mathbf{h}(i)$ should be estimated. It has been shown in [17] that, when using iterative techniques, such as DCD iterations, an approach which is based on transforming the original sequence of normal equations into a sequence of auxiliary normal equations, is preferable.

Let $\mathbf{r}(i-1) = \boldsymbol{\beta}(i-1) - \mathbf{R}(i-1)\hat{\mathbf{h}}(i-1)$ be a residual vector for the approximate solution $\hat{\mathbf{h}}(i-1)$ at time instant $(i-1)$. We denote $\Delta\mathbf{R}(i) = \mathbf{R}(i) - \mathbf{R}(i-1)$, $\Delta\boldsymbol{\beta}(i) = \boldsymbol{\beta}(i) - \boldsymbol{\beta}(i-1)$ and $\Delta\mathbf{h}(i) = \mathbf{h}(i) - \hat{\mathbf{h}}(i-1)$. The normal equations (2.2) can then be rewritten as

$$\mathbf{R}(i)\left[\hat{\mathbf{h}}(i-1) + \Delta\mathbf{h}(i)\right] = \boldsymbol{\beta}(i).$$

This can be represented as a system of equations with respect to $\Delta\mathbf{h}(i)$

$$\begin{aligned}
\mathbf{R}(i)\Delta\mathbf{h}(i) &= \boldsymbol{\beta}(i) - \mathbf{R}(i)\hat{\mathbf{h}}(i-1) \\
&= \boldsymbol{\beta}(i-1) - \mathbf{R}(i-1)\hat{\mathbf{h}}(i-1) + \Delta\boldsymbol{\beta}(i) - \Delta\mathbf{R}(i)\hat{\mathbf{h}}(i-1) \\
&= \mathbf{r}(i-1) + \Delta\boldsymbol{\beta}(i) - \Delta\mathbf{R}(i)\hat{\mathbf{h}}(i-1).
\end{aligned}$$

The original sequence of normal equations is now transformed into a sequence of auxiliary normal equations, given by [17]

$$\mathbf{R}(i)\Delta\mathbf{h}(i) = \boldsymbol{\beta}_0(i),$$

where $\boldsymbol{\beta}_0(i) = \mathbf{r}(i-1) + \Delta\boldsymbol{\beta}(i) - \Delta\mathbf{R}(i)\hat{\mathbf{h}}(i-1)$. A recursive approach for solving a sequence of systems of equations is presented in Table 2.1.

Table 2.1: Recursively solving a sequence of equations

Step	Equation
	Initialization: $\mathbf{r}(-1)=\mathbf{0}$,$\boldsymbol{\beta}(-1)=\mathbf{0}$,$\hat{\mathbf{h}}(-1)=\mathbf{0}$
	for $i=0,1,\ldots$
1	Find $\Delta\boldsymbol{R}(i)$ and $\Delta\boldsymbol{\beta}(i)$
2	$\boldsymbol{\beta}_0(i)=\mathbf{r}(i-1)+\Delta\boldsymbol{\beta}(i)-\Delta\mathbf{R}(i)\hat{\mathbf{h}}(i-1)$
3	Solve $\mathbf{R}(i)\Delta\mathbf{h}=\boldsymbol{\beta}_0(i)\Rightarrow\Delta\hat{\mathbf{h}}(i),\ \mathbf{r}(i)$
4	$\hat{\mathbf{h}}(i)=\hat{\mathbf{h}}(i-1)+\Delta\hat{\mathbf{h}}(i)$

Table 2.2: Exponentially Weighted RLS Algorithm

Step	Equation
	Initialization: $\hat{\mathbf{h}}(-1)=0$, $\mathbf{r}(-1)=0$, $\mathbf{R}(-1)=\boldsymbol{\Pi}$
	for $i=0,1,\ldots$
1	$\mathbf{R}(i)=\lambda\mathbf{R}(i-1)+\mathbf{x}(i)\mathbf{x}^T(i)$
2	$z(i)=\mathbf{x}^T(i)\hat{\mathbf{h}}(i-1)$
3	$e(i)=y(i)-z(i)$
4	$\boldsymbol{\beta}_0(i)=\lambda\mathbf{r}(i-1)+e(i)\mathbf{x}(i)$
5	Solve $\mathbf{R}(i)\Delta\mathbf{h}(i)=\boldsymbol{\beta}_0(i)\Rightarrow\Delta\hat{\mathbf{h}}(i),\ \mathbf{r}(i)$
6	$\hat{\mathbf{h}}(i)=\hat{\mathbf{h}}(i-1)+\Delta\hat{\mathbf{h}}(i)$

In the exponentially weighted RLS (ERLS) algorithm, the vector $\mathbf{h}(i)$ is found by solving the normal equations (2.2) with the instantaneous estimates to the correlation quantities, given by [4]:

$$\begin{aligned}\mathbf{R}(i) &= \sum_{j=0}^{i}\lambda^{i-j}\mathbf{x}(j)\mathbf{x}^T(j)+\lambda^i\boldsymbol{\Pi},\\ \boldsymbol{\beta}(i) &= \sum_{j=0}^{i}\lambda^{i-j}\mathbf{x}(j)y(j),\end{aligned}$$

where $\boldsymbol{\Pi}$ is a regularization matrix and $0<\lambda\leq 1$ is a forgetting factor. The regularization matrix is usually chosen as a diagonal matrix and is used for stabilizing the algorithm [4]. After applying the method in Table 2.1 to this problem, we can summarize the ERLS algorithm as shown in Table 2.2 [17]. The DCD algorithm has firstly been introduced in [16]. In this work, we are interested in the new DCD algorithm [17], which is shown in Table 2.3. In Table 2.3, elements of the matrix and vector are denoted as $R_{p,n}$ and r_n, respectively, and a pth column of $\mathbf{R}$ is denoted as $\mathbf{R}^{(p)}$. This DCD algorithm is then used to solve the normal equation at step 5 in Table 2.2.

Table 2.3: DCD algorithm with leading element

Step	Equation
	Initialization: $\Delta\hat{\mathbf{h}} = 0$, $\mathbf{r} = \boldsymbol{\beta}_0$, $\alpha = H/2$, $m = 1$
	for $k = 1, \ldots, N_u$
1	$p = \arg\max_{n=1,\ldots,M}\{\lvert r_n\rvert\}$, go to step 4
2	$m = m + 1$, $\alpha = \alpha/2$
3	if $m > M_b$, the algorithm stops
4	if $\lvert r_p\rvert \leq (\alpha/2)R_{p,p}$, then go to step 2
5	$\Delta\hat{h}_p = \Delta\hat{h}_p + \text{sign}(r_p)\alpha$
6	$\mathbf{r} = \mathbf{r} - \text{sign}(r_p)\alpha\mathbf{R}^{(p)}$

2.3 Convergence analysis

In this section, we derive an approach for convergence analysis of the RLS-DCD algorithm based on computations with only deterministic correlation quantities. We explore deterministic expressions for time dependent correlation quantities without involving any stochastic processes, and then solve modified normal equations with only deterministic quantities by using the DCD algorithm. Finally, we derive the deterministic equations for the Mean Squared Error (MSD) and Mean Squared Deviation (MSD). The common *independence assumptions* [4] are employed in our analysis.

2.3.1 Deterministic correlation quantities

The auto-correlation matrix $\mathbf{R}(i)$ at instant $i \geq M$, can be approximated by the deterministic expression [72]

$$\mathbf{R}_d(i) = a(i)\mathbf{R}_{\mathbf{xx}} + \lambda^i\boldsymbol{\Pi}, \tag{2.3}$$

where $a(i) = (1 - \lambda^i)/(1 - \lambda)$. For the cross-correlation vector $\boldsymbol{\beta}(i)$, according to the desired response $y(i)$ in (2.1), we obtain

$$\boldsymbol{\beta}(i) = \sum_{j=0}^{i} \lambda^{i-j}\mathbf{x}(j)\mathbf{x}^T(j)\mathbf{h}_0 + \sum_{j=0}^{i} \lambda^{i-j}\mathbf{x}(j)\nu(j).$$

Let a deterministic cross-correlation vector $\boldsymbol{\beta}_d(i)$ be expressed as

$$\boldsymbol{\beta}_d(i) = \boldsymbol{\beta}_{nf}(i) + \boldsymbol{\beta}_n(i),$$

where $\boldsymbol{\beta}_{nf}(i)$ denotes the instantaneous cross-correlation vector between the noise free desired response $y(i) = \mathbf{x}^T(j)\mathbf{h}_0$ [$\nu(i) = 0$ in (2.1)] and the data vector $\mathbf{x}(i)$, and $\boldsymbol{\beta}_n(i)$ denotes the cross-correlation vector between the measurement noise $\nu(i)$ and the data vector $\mathbf{x}(i)$. The vector $\boldsymbol{\beta}_{nf}(i)$ can be derived by using the similar approximation as in $\mathbf{R}_d(i)$. At time instant $i \geq M$, $\boldsymbol{\beta}_{nf}(i)$ can be approximated by

$$\boldsymbol{\beta}_{nf}(i) = a(i)\mathbf{R}_{\mathbf{xx}}\mathbf{h}_0. \tag{2.4}$$

The vector $\boldsymbol{\beta}_n(i)$ involves the measurement noise process $\nu(i)$, which is an obstacle for us to evaluate $\boldsymbol{\beta}_n(i)$. Instead of evaluating the vector $\boldsymbol{\beta}_n(i)$, we are more interested in deriving the deterministic expression for the auto-correlation matrix of $\boldsymbol{\beta}_n(i)$, given as

$$\mathbf{R}_{\boldsymbol{\beta}_n}(i) = E\left\{\boldsymbol{\beta}_n(i)\boldsymbol{\beta}_n^T(i)\right\}.$$

Since the measurement noise $\nu(i)$ is assumed to be white and with variance σ_ν^2, we have

$$E\left\{\nu(j)\nu(l)\right\} = \begin{cases} \sigma^2, & j = l \\ 0, & j \neq l \end{cases}$$

Therefore, we obtain

$$\mathbf{R}_{\boldsymbol{\beta}_n}(i) = \sigma_\nu^2 \sum_{j=0}^{i} \lambda^{2(i-j)} E\left\{\mathbf{x}(j)\mathbf{x}^T(j)\right\},$$

which can then be expressed at every time instant $i \geq M$ by

$$\mathbf{R}_{\boldsymbol{\beta}_n}(i) = \sigma_\nu^2 b(i)\mathbf{R}_{\mathbf{xx}}, \tag{2.5}$$

where $b(i) = (1 - \lambda^{2i})/(1 - \lambda^2)$.

We now replace the instantaneous correlation quantities $\mathbf{R}$ and $\boldsymbol{\beta}$ by their deterministic expressions $\mathbf{R}_d$ and $\boldsymbol{\beta}_d$, respectively, into the RLS normal equations (2.2), which gives at every time instant i:

$$\mathbf{R}_d(i)\mathbf{h}_p(i) = \boldsymbol{\beta}_{nf}(i) + \boldsymbol{\beta}_n(i), \tag{2.6}$$

where $\mathbf{h}_p(i)$ denotes the prediction of the solution vector. Since on the right hand side of (2.6), the deterministic quantity $\boldsymbol{\beta}_n(i)$ is unknown and difficult to derive, we let

$$\mathbf{h}_p(i) = \mathbf{h}_{nf}(i) + \mathbf{h}_n(i), \tag{2.7}$$

where we assume

$$\mathbf{R}_d(i)\mathbf{h}_{nf}(i) = \boldsymbol{\beta}_{nf}(i), \tag{2.8}$$

$$\mathbf{R}_d(i)\mathbf{h}_n(i) = \boldsymbol{\beta}_n(i). \tag{2.9}$$

Equations (2.8) are now dealing with only deterministic quantities that we have derived: $\mathbf{R}_d(i)$ given by (2.3) and $\boldsymbol{\beta}_{nf}(i)$ given by (2.4). As we considered in Section 2.1, the RLS-DCD algorithm deals with the auxiliary normal equations. In order to solve the equations (2.8) by using the DCD algorithm, we need to transform these deterministic equations into a sequence of deterministic auxiliary equations, defined by

$$\mathbf{R}_d(i)\Delta\mathbf{h}_{nf}(i) = \boldsymbol{\beta}_{0\ d}(i).$$

According to the recursive approach for solving the auxiliary equations in Table 2.1, and by substituting all the variables by their deterministic expressions, we obtain a sequence of equations with only deterministic quantities as shown in Table 2.4. Here, $\beta_{0\ d}(i)$ is the deterministic expression for vector $\beta_0(i)$, and according to the equation at step 2 in Table 2.1, it is given by

$$\boldsymbol{\beta}_{0\ d}(i) = \mathbf{r}_d(i-1) + \Delta\boldsymbol{\beta}_{nf}(i) - \Delta\mathbf{R}_d(i)\mathbf{h}_{nf}(i-1), \tag{2.10}$$

where $\mathbf{r}_d(i) = \boldsymbol{\beta}_{nf}(i) - \mathbf{R}_d(i)\mathbf{h}_{nf}(i)$ is the deterministic residual vector for the solution vector at time instant i, and

$$\begin{aligned}\Delta\mathbf{R}_d(i) &= \mathbf{R}_d(i) - \mathbf{R}_d(i-1),\\ \Delta\boldsymbol{\beta}_{nf}(i) &= \boldsymbol{\beta}_{nf}(i) - \boldsymbol{\beta}_{nf}(i-1).\end{aligned}$$

Equation (2.10) can then be rewritten as

$$\begin{aligned}\boldsymbol{\beta}_{0\ d}(i) &= \boldsymbol{\beta}_{nf}(i-1) - \mathbf{R}_d(i-1)\mathbf{h}_{nf}(i-1) + \boldsymbol{\beta}_{nf}(i) - \boldsymbol{\beta}_{nf}(i-1)\\ &\quad - [\mathbf{R}_d(i) - \mathbf{R}_d(i-1)]\,\mathbf{h}_{nf}(i-1)\\ &= \boldsymbol{\beta}_{nf}(i) - \mathbf{R}_d(i)\mathbf{h}_{nf}(i-1),\end{aligned}$$

as given at step 2 in Table 2.4. At every time instant i, the DCD algorithm is used to solve the equation at step 3 in Table 2.4, and the solution vector $\mathbf{h}_{nf}(i)$ is deterministic.

For equations (2.9), the deterministic quantity $\boldsymbol{\beta}_n(i)$ is unknown to us, and thus, the deterministic solution vector $\mathbf{h}_n(i)$ can not be obtained. However, for the analysis of mean square performance of adaptive filtering algorithms, such as mean square error (we will consider in the next section), we are more interested in second order statistical quantities. Therefore, instead of evaluating $\mathbf{h}_n(i)$, we evaluate the autocorrelation matrix of $\mathbf{h}_n(i)$, denoted by

$$\mathbf{R}_{\mathbf{h}_n}(i) = E\left\{\mathbf{h}_n(i)\mathbf{h}_n^T(i)\right\}.$$

Table 2.4: A sequence of equations with only deterministic quantities

Step	Equation
	Initialization: $\mathbf{h}_{nf}(-1) = \mathbf{0}$
	for $i = 0, 1, \ldots$
1	Find $\boldsymbol{R}_d(i)$ and $\boldsymbol{\beta}_{nf}(i)$
2	$\boldsymbol{\beta}_{0\ d}(i) = \boldsymbol{\beta}_{nf}(i) - \mathbf{R}_d(i)\mathbf{h}_{nf}(i-1)$
3	Solve $\mathbf{R}_d(i)\Delta\mathbf{h}_{nf} = \boldsymbol{\beta}_{0\ d}(i) \Rightarrow \Delta\mathbf{h}_{nf}(i)$
4	$\mathbf{h}_{nf}(i) = \mathbf{h}_{nf}(i-1) + \Delta\mathbf{h}_{nf}(i)$

By assuming that $\mathbf{h}_n(i) = \mathbf{R}_d^{-1}(i)\boldsymbol{\beta}_n(i)$, we then have

$$\mathbf{R}_{\mathbf{h}_n}(i) = \mathbf{R}_d^{-1}(i)\mathbf{R}_{\boldsymbol{\beta}_n}(i)\mathbf{R}_d^{-1}(i),$$

where $\mathbf{R}_{\boldsymbol{\beta}_n}(i)$ is the deterministic auto-correlation matrix of $\boldsymbol{\beta}_n(i)$, given by (2.5).

2.3.2 Deterministic equations for evaluating MSE and MSD

The Mean Squared Error (MSE) at each time instant i is defined by [5]:

$$\text{MSE} = E\left\{|e(i)|^2\right\}, \text{where } \ e(i) = y(i) - \mathbf{x}^T(i)\hat{\mathbf{h}}(i).$$

Under the assumption that the measurement noise $\nu(i)$ is *i.i.d.* and statistically independent of the input vector $\mathbf{x}(i)$, the Mean Squared Error (MSE) can then be evaluated by [5]

$$\text{MSE} = E\left\{|e_a(i)|^2\right\} + \sigma_\nu^2, \tag{2.11}$$

where $e_a(i)$ is the *a priori* error at instant i and defined by

$$e_a(i) = \mathbf{x}^T(i)\left[\mathbf{h}_0 - \hat{\mathbf{h}}(i)\right].$$

Considering the predicted solution $\mathbf{h}_p(i)$ defined by (2.7) instead of $\hat{\mathbf{h}}(i)$, we have $e_a(i) = \mathbf{x}^T(i)\left[\boldsymbol{\epsilon}_d(i) - \mathbf{h}_n(i)\right]$, where $\boldsymbol{\epsilon}_d(i) = \mathbf{h}_0 - \mathbf{h}_{nf}(i)$. The vector $\boldsymbol{\epsilon}_d(i)$ is deterministic, since both $\mathbf{h}_0$ and $\mathbf{h}_{nf}(i)$ are deterministic.

According to the *independence assumption*, the error vector $\boldsymbol{\epsilon}(i) = \mathbf{h}_0 - \hat{\mathbf{h}}(i)$ is independent of $\mathbf{x}(i)$, and therefore, we can obtain the mean square *a priori* error as

$$\begin{aligned} E\left\{|e_a(i)|^2\right\} = {} & \boldsymbol{\epsilon}_d^T(i)\mathbf{R}_{\mathbf{xx}}\boldsymbol{\epsilon}_d(i) - E\left\{2\boldsymbol{\epsilon}_d^T(i)\mathbf{R}_{\mathbf{xx}}\mathbf{h}_n(i)\right\} \\ & + E\left\{\mathbf{h}_n^T(i)\mathbf{R}_{\mathbf{xx}}\mathbf{h}_n(i)\right\}. \end{aligned} \tag{2.12}$$

The first term on the right hand side of (2.12) is deterministic, since $\boldsymbol{\epsilon}_d(i)$ is deterministic and $\mathbf{R}_{\mathbf{xx}}$ is a constant matrix. According to the assumption, $\mathbf{h}_n(i) = \mathbf{R}_d^{-1}(i)\boldsymbol{\beta}_n(i)$, the second term on the right hand side of (2.12) can be expressed as

$$E\left\{2\boldsymbol{\epsilon}_d^T(i)\mathbf{R}_{\mathbf{xx}}\mathbf{h}_n(i)\right\} = 2E\left\{\boldsymbol{\epsilon}_d^T(i)\mathbf{R}_{\mathbf{xx}}\mathbf{R}_d^{-1}(i)\boldsymbol{\beta}_n(i)\right\}.$$

According to the *independence assumption*, we find that $\boldsymbol{\epsilon}_d(i)$, $\boldsymbol{\beta}_n(i)$ and $\mathbf{R}_{\mathbf{xx}}\mathbf{R}_d^{-1}(i)$ are independent of each other, and as $E\left\{\boldsymbol{\beta}_n(i)\right\} = \mathbf{0}$, we obtain $E\left\{2\boldsymbol{\epsilon}_d^T(i)\mathbf{R}_{\mathbf{xx}}\mathbf{h}_n(i)\right\} = 0$. The third term on the right hand side of (2.12) can be expressed as

$$E\left\{\mathbf{h}_n^T(i)\mathbf{R}_{\mathbf{xx}}\mathbf{h}_n(i)\right\} = tr\left\{\mathbf{R}_{\mathbf{xx}}\mathbf{R}_{\mathbf{h}_n}(i)\right\},$$

which is deterministic. The mean squared *a priori* error given by (2.12), can now be evaluated by a deterministic expression, which gives

$$E\left\{|e_a(i)|^2\right\} = \boldsymbol{\epsilon}_d^T(i)\mathbf{R}_{\mathbf{xx}}\boldsymbol{\epsilon}_d(i) + tr\left\{\mathbf{R}_{\mathbf{xx}}\mathbf{R}_{\mathbf{h}_n}(i)\right\}. \tag{2.13}$$

Therefore, the prediction of the MSE learning curves can be computed by substituting (2.13) into (2.11), which gives

$$\text{MSE}(i) = \boldsymbol{\epsilon}_d^T(i)\mathbf{R}_{\mathbf{xx}}\boldsymbol{\epsilon}_d(i) + tr\left\{\mathbf{R}_{\mathbf{xx}}\mathbf{R}_{\mathbf{h}_n}(i)\right\} + \sigma_\nu^2. \tag{2.14}$$

The Mean Squared Deviation (MSD) at each time instant i is defined by [5]: $\text{MSD}(i) = E\left\{\left[\mathbf{h}_0 - \hat{\mathbf{h}}(i)\right]^T\left[\mathbf{h}_0 - \hat{\mathbf{h}}(i)\right]\right\}$. Again, considering the predicted solution $\mathbf{h}_p(i)$ defined by (2.7) instead of $\hat{\mathbf{h}}(i)$, we obtain

$$\text{MSD}(i) = \boldsymbol{\epsilon}_d^T(i)\boldsymbol{\epsilon}_d(i) - E\left\{2\boldsymbol{\epsilon}_d^T(i)\mathbf{h}_n(i)\right\} + E\left\{\mathbf{h}_n^T(i)\mathbf{h}_n(i)\right\}. \tag{2.15}$$

After some algebra and employing the *independence assumption* similar to that used for derivation of the MSE as presented above, we finally obtain a deterministic equation for predicting the learning curve of MSD at every time instant i, given by

$$\text{MSD}(i) = \boldsymbol{\epsilon}_d^T(i)\boldsymbol{\epsilon}_d(i) + tr\left\{\mathbf{R}_{\mathbf{h}_n}(i)\right\}. \tag{2.16}$$

2.4 Simulation results

Below, we present simulation results which compare the prediction of the MSE and MSD learning curves obtained by using the proposed approach versus the practical learning

curves for the ERLS-DCD algorithm. The desired response $y(i)$ is generated according to (2.1). The input vector to the filter $\mathbf{x}(i) = [x(i)\ x(i-1)\ldots\ x(i-M+1)]^T$ contains autoregressive correlated random numbers generated according to $x(i) = vx(i-1) + w(i)$, where v is the autoregressive factor ($0 \leq v < 1$) and $w(i)$ are uncorrelated zero-mean random Gaussian numbers of unit variance. In our simulations, the regularization matrix $\mathbf{\Pi}$ is chosen as a diagonal matrix $\mathbf{\Pi} = \eta\mathbf{I}_M$. For generating the predicted learning curves, the auto-correlation matrix $\mathbf{R_{xx}}$ is derived according to the autoregressive model by $(\mathbf{R_{xx}})_{m,n} = r_{\mathbf{xx}}(|m-n|)$ [4], where $r_{\mathbf{xx}}(n) = v^n\sigma_x^2$ and σ_x^2 is the variance of the input samples, which can be evaluated by $\sigma_x^2 = 1/(1-v^2)$ [73]. The autoregressive factor v and vector $\mathbf{h}_0$ are given. For generating the practical learning curves, the MSE curve is computed by averaging as [74]:

$$\hat{E}(i) = (1/N_{exp})\sum_{l=1}^{N_{exp}} e^{(l)}(i)^2, \tag{2.17}$$

where $e^{(l)}(i) = \left[y(i) - \mathbf{x}^T(i)\hat{\mathbf{h}}(i-1)\right]^{(l)}$ and N_{exp} is the number of independent experiments. The MSD curve is computed by averaging as [74]:

$$\hat{D}(i) = (1/N_{exp})\sum_{l=1}^{N_{exp}} \left|\boldsymbol{\epsilon}^{(l)}(i)\right|^2, \tag{2.18}$$

where $\boldsymbol{\epsilon}^{(l)}(i) = \left[\mathbf{h}_0 - \hat{\mathbf{h}}(i)\right]^{(l)}$.

In the following simulations, the practical curves are obtained by ensemble average over 200 independent experiments ($N_{exp} = 200$). Our predictions to the learning curves start at the time instant $i = M$, since the approximation of the deterministic auto-correlation matrix, $\mathbf{R}_d$, expressed by (2.3) is only valid for $i \geq M$. Figures 2.2 and 2.3 show the predicted learning curves against the simulated curves for the ERLS-DCD algorithm with different autoregressive factors v and with different number of updates N_u, respectively. It is seen that the prediction shows good agreement with the practical learning curves, although our predictions are somewhat optimistic.

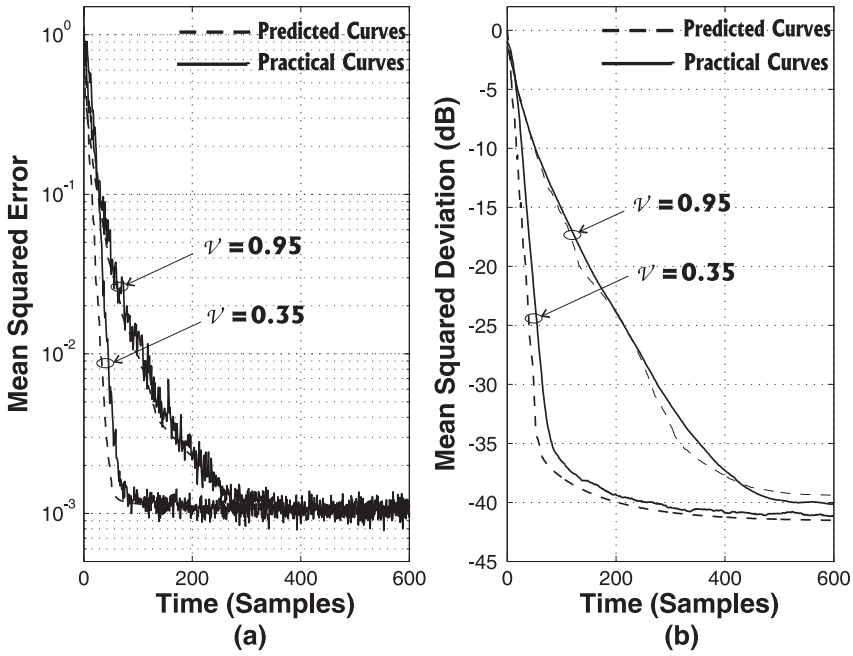


Figure 2.2: Predicted vs Practical learning curves for the ERLS-DCD algorithm with different autoregressive factors v: (a) MSE; (b) MSD. Simulation parameters: $M = 16$, $\sigma_\nu^2 = 10^{-3}$, $\lambda = 1 - 1/(8M)$, $\eta = 10^{-6}$, $N_u = 1$, $H = 1$, $M_b = 16$, $N_{exp} = 200$.

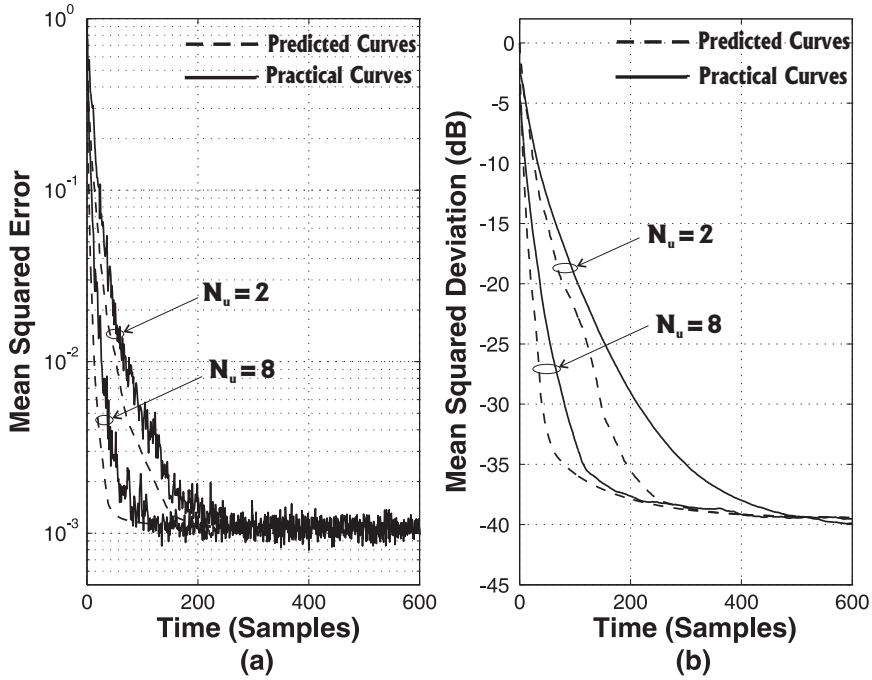


Figure 2.3: Predicted vs Practical learning curves for the ERLS-DCD algorithm with different successful updates N_u: (a) MSE; (b) MSD. Simulation parameters: $M = 16$, $\sigma_\nu^2 = 10^{-3}$, $\lambda = 1 - 1/(8M)$, $\eta = 10^{-6}$, $v = 0.9$, $H = 1$, $M_b = 16$, $N_{exp} = 200$.

2.5 Conclusions

In this chapter, we have presented a new approach for convergence analysis of the RLS-DCD adaptive filtering algorithm. This approach is based on computation with only deterministic quantities, which are derived from statistical characteristics of signals. The approach can be also used for other adaptive filtering algorithms based on iteratively solving the normal equations with one or more iterations at a time instant. Deterministic equations for predicting the MSE and MSD learning curves of the RLS-DCD algorithm have been obtained, and simulation results have shown good agreement when $\lambda \lesssim 1$, although the predictions are somewhat optimistic. From our analysis, we have observed fast convergence of the RLS-DCD algorithm. In the next chapter, we will introduce a low-complexity channel-estimate based adaptive linear equalizer, in which the computation of equalizer coefficients can be multiplication-free and division-free, when using the RLS-DCD algorithm for channel estimation and the DCD iterations in the equalizer.

Chapter 3

Low-Complexity Channel-Estimate Based Adaptive Linear Equalization

Contents

In Chapter 2, we have presented the convergence analysis of the RLS-DCD algorithm and shown its fast convergence. It is also known that the RLS-DCD algorithm has a complexity as low as $\mathcal{O}(N_u M)$ operations per sample, where M is the filter length and N_u is the number of iterations such that $N_u \ll M$ [17].

In this chapter, we propose a low-complexity channel-estimate based adaptive linear equalizer, which exploits DCD iterations for computation of equalizer coefficients. The proposed technique has as low complexity as $\mathcal{O}(N_u(K + M))$ operations per sample, where K and M are the equalizer and channel estimator length, respectively, and N_u is the number of iterations such that $N_u \ll K$ and $N_u \ll M$. Moreover, when using the RLS-DCD algorithm for channel estimation, the computation of equalizer coefficients is multiplication-free and division-free, which makes the equalizer attractive for hardware

design. Simulation shows that the proposed adaptive equalizer performs close to the minimum mean-square-error (MMSE) equalizer with perfect knowledge of the channel.

This chapter is organized as follows. In the next section, an introduction is given. Section 3.2 gives the data models and normal equations for computing the MMSE LE coefficients. In Section 3.3, we firstly introduce the assumptions used in our derivation, and then propose a low-complexity approach for computing the LE coefficients. Simulation results which compare the performance and computational complexity of the proposed LE against known techniques are given in Section 3.4, followed by conclusions in Section 3.5.

3.1 Introduction

Equalization is a well known method for combatting the inter-symbol interference in communication channels [21]. Coefficients of an adaptive linear equalizer (LE) can be computed without explicit channel estimation using the channel output and known pilot signal [21]. However, channel-estimate (CE) based equalizers can outperform LEs with the direct adaptation [22]. The CE based adaptive equalizers re-compute equalizer coefficients for every update of the channel estimate, preferably for every sample of a received signal. This requires generation and inversion of a $K \times K$ channel autocorrelation matrix, where K is the equalizer length. In general, it results in a complexity of $\mathcal{O}(K^3)$ operations per sample. Exploiting structural properties of the matrix, the complexity can be reduced down to $\mathcal{O}(K^2)$ operations [7]. RLS adaptive channel estimators have a complexity of $\mathcal{O}(M^2)$, where M is the channel estimator length [4]. It is usual that $K > M$, thus the complexity of computing the equalizer coefficients determines the total complexity. Moreover, the RLS-DCD adaptive algorithm has been proposed to have a complexity as low as $\mathcal{O}(N_u M)$ operations per sample, where $N_u << M$, while to perform very close to the RLS algorithm [17]. Thus, adaptive channel estimation can be significantly simpler than CE based computation of equalizer coefficients. To reduce the whole complexity, computation of equalizer coefficients should be simplified.

In this chapter, we propose a novel CE based adaptive LE. The proposed equalizer

is applicable for using together with channel estimators based on adaptive algorithms with partial update (see [75] and references therein), including adaptive algorithms with coordinate descent iterations [16,17,76,77]. Moreover, we show that when using the DCD iterations, computation of equalizer coefficients can be multiplication-free and division-free. When using the DCD algorithm in both the channel estimator and equalizer, the overall complexity of the equalization is as low as $\mathcal{O}(N_u(K+M))$ operations per sample.

3.2 Linear MMSE equalizer

We consider that the received signal $y(n)$ is given by

$$y(n) = \mathbf{x}^T(n)\mathbf{h}(n) + \nu(n), \tag{3.1}$$

where $\mathbf{x}(n) = [x(n)\, x(n-1) \dots\, x(n-M+1)]^T$, $x(n)$ is the transmitted signal, $\mathbf{h}(n) = [h_1(n)\, h_2(n) \dots\, h_M(n)]^T$ is the channel impulse response, and $\nu(n)$ is the white noise with zero mean and variance σ_ν^2; $\mathbf{x}(n)$, $\mathbf{h}(n)$ and $\nu(n)$ are real-valued. At time instant n, a K-length LE with the tap weight coefficient vector $\mathbf{f}(n)$ estimates the transmitted signal as $\hat{x}(n) = \mathbf{y}^T(n)\mathbf{f}(n)$, where $\mathbf{y}(n) = [y(n)\, y(n-1) \dots y(n-K+1)]^T$. Figure 3.1 shows the block diagram of the direct adaptation LE. The equalizer vector $\mathbf{f}(n)$ is adjusted to minimize the mean square error (MSE) $E\{[x(n) - \hat{x}(n)]^2\}$. For CE based equalization, minimizing the MSE requires solving the normal equations [21]

$$\mathbf{G}(n)\mathbf{f}(n) = \boldsymbol{\xi}(n), \tag{3.2}$$

where $\mathbf{G}(n) = \mathbf{H}^T(n)\mathbf{H}(n) + \sigma_\nu^2\mathbf{I}_K$, $\boldsymbol{\xi}(n) = \mathbf{H}^T(n)\mathbf{e}_l$, $\mathbf{e}_l$ is a $(K+M-1)\times 1$ vector of all zeros except the lth element, which equals one and corresponds to the equalizer delay, and $\mathbf{H}(n)$ is a $(K+M-1)\times K$ time-varying channel convolution matrix. In practice, as the time-varying channel is unknown, estimates $\hat{\mathbf{h}}(n-j), j = 0, \dots, K-1$, of the

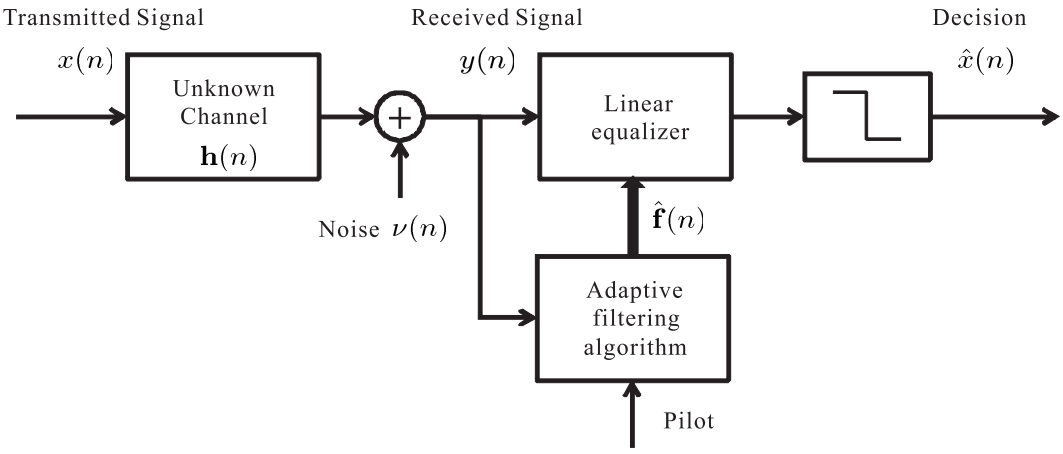


Figure 3.1: Direct adaptation LE.

channel impulse response are used to form $\mathbf{H}(n)$ as given by

$$\mathbf{H}(n) = \begin{bmatrix} \hat{h}_1(n) & 0 & \cdots & 0 & 0 \\ \hat{h}_2(n) & \hat{h}_1(n-1) & \ddots & \vdots & \vdots \\ \vdots & \hat{h}_2(n-1) & \ddots & 0 & \vdots \\ \hat{h}_M(n) & \vdots & \ddots & \hat{h}_1(n-K+2) & 0 \\ 0 & \hat{h}_M(n-1) & \ddots & \hat{h}_2(n-K+2) & \hat{h}_1(n-K+1) \\ \vdots & 0 & \ddots & \vdots & \hat{h}_2(n-K+1) \\ \vdots & \vdots & \ddots & \hat{h}_M(n-K+2) & \vdots \\ 0 & 0 & \cdots & 0 & \hat{h}_M(n-K+1) \end{bmatrix}. \quad (3.3)$$

Figure 3.2 shows the block diagram of the Channel estimate based adaptive LE.

3.3 Low-complexity CE based adaptive LE

3.3.1 Assumptions

We use the following assumptions:

1) For every time sample n, the channel estimate can be updated N_u times. We will be using the index $i = (n-1)N_u + k$, where $k = 1, \ldots, N_u$, to indicate such an update. Correspondingly, the sequence of the normal equations to be solved in the MMSE LE is

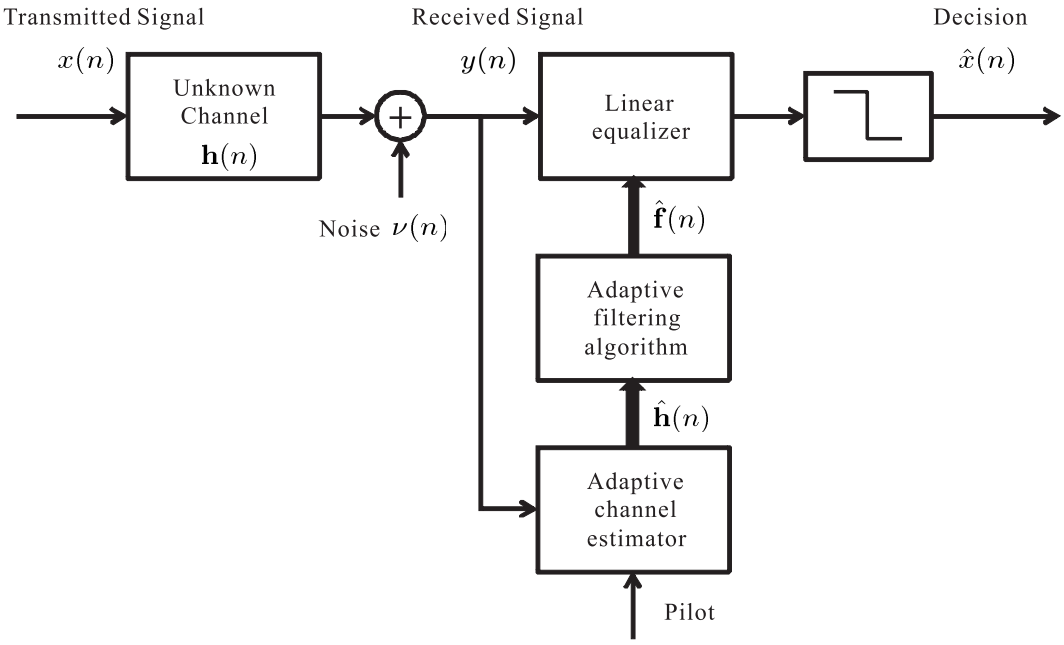


Figure 3.2: Channel estimate based adaptive LE.

now given by

$$\mathbf{G}(i)\mathbf{f}(i) = \boldsymbol{\xi}(i). \tag{3.4}$$

2) For every i, the channel estimator updates only one, $p(i)$th, element in $\hat{\mathbf{h}}(i)$ as

$$\hat{h}_{p(i)}(i) = \hat{h}_{p(i)}(i-1) + \Delta\hat{h}(i).$$

3) For every i, only one, $q(i)$th, equalizer coefficient in $\hat{\mathbf{f}}(i)$ is updated as

$$\hat{f}_{q(i)}(i) = \hat{f}_{q(i)}(i-1) + \Delta\hat{f}(i).$$

Here, $\hat{\mathbf{f}}(i)$ denotes an approximation to the MMSE solution $\mathbf{f}(i)$ at iteration i.

4) The convolution matrix (3.3) can be approximated for each i as

$$\hat{\mathbf{H}}(i) = \begin{bmatrix} \hat{h}_1(i) & 0 & \cdots & 0 & 0 \\ \hat{h}_2(i) & \hat{h}_1(i) & \ddots & \vdots & \vdots \\ \vdots & \hat{h}_2(i) & \ddots & 0 & \vdots \\ \hat{h}_M(i) & \vdots & \ddots & \hat{h}_1(i) & 0 \\ 0 & \hat{h}_M(i) & \ddots & \hat{h}_2(i) & \hat{h}_1(i) \\ \vdots & 0 & \ddots & \vdots & \hat{h}_2(i) \\ \vdots & \vdots & \ddots & \hat{h}_M(i) & \vdots \\ 0 & 0 & \cdots & 0 & \hat{h}_M(i) \end{bmatrix}. \tag{3.5}$$

The number of iterations for computing the equalizer coefficients after an update of the channel estimate can be made greater than one. This is a straightforward extension of the algorithm described below. However, our simulation has shown little improvement in the equalizer performance compared to the case of one iteration (as given by assumption 3).

3.3.2 Derivation

Equations (3.4) can be transformed into a sequence of auxiliary normal equations $\mathbf{G}(i)\Delta\mathbf{f}(i) = \boldsymbol{\xi}_0(i)$ [17]. A recursive approach for solving the equations is described in Table 3.1 [17], where: $\mathbf{r}(i)$ is the residual vector $\mathbf{r}(i) = \boldsymbol{\xi}(i) - \mathbf{G}(i)\hat{\mathbf{f}}(i)$; $\Delta\mathbf{G}(i) = \mathbf{G}(i) - \mathbf{G}(i-1)$; and $\Delta\boldsymbol{\xi}(i) = \boldsymbol{\xi}(i) - \boldsymbol{\xi}(i-1)$.

Although, by considering the Toeplitz structure of matrices, most of the computations in Table 3.1 can be simplified. Step 1 requires finding $\Delta\mathbf{G}(i)$ which involves computation of the matrix $\mathbf{G}(i) = \hat{\mathbf{H}}^T(i)\hat{\mathbf{H}}(i)$ with a complexity of $\mathcal{O}(M^2)$. Step 2 requires $\mathcal{O}(MK)$ operations to compute $\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$. These are still the most computationally demanding operations and below we show how these operations can be simplified when using our assumptions.

Computation of $\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$:

Table 3.1: Recursively solving a sequence of equations

Step	Equation
	Initialization: $\mathbf{r}(0) = \mathbf{0}$, $\boldsymbol{\xi}(0) = \mathbf{0}$, $\hat{\mathbf{f}}(0) = \mathbf{0}$
	for $i = 1, 2, \ldots$
1	Find $\Delta\mathbf{G}(i)$ and $\Delta\boldsymbol{\xi}(i)$
2	$\boldsymbol{\xi}_0(i) = \mathbf{r}(i-1) + \Delta\boldsymbol{\xi}(i) - \Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$
3	Solve $\mathbf{G}(i)\Delta\mathbf{f} = \boldsymbol{\xi}_0(i) \Rightarrow \Delta\hat{\mathbf{f}}(i),\ \mathbf{r}(i)$
4	$\hat{\mathbf{f}}(i) = \hat{\mathbf{f}}(i-1) + \Delta\hat{\mathbf{f}}(i)$

Let $\hat{\mathbf{H}}(i) = \hat{\mathbf{H}}(i-1) + \boldsymbol{\Delta}(i)$, then we have

$$\Delta\mathbf{G}(i) = \boldsymbol{\Delta}^T(i)\hat{\mathbf{H}}(i-1) + \hat{\mathbf{H}}^T(i-1)\boldsymbol{\Delta}(i) + \boldsymbol{\Delta}^T(i)\boldsymbol{\Delta}(i), \tag{3.6}$$

and

$$\begin{aligned}\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1) = {} & \boldsymbol{\Delta}^T(i)\hat{\mathbf{H}}(i-1)\hat{\mathbf{f}}(i-1) \\ & + \hat{\mathbf{H}}^T(i-1)\boldsymbol{\Delta}(i)\hat{\mathbf{f}}(i-1) + \boldsymbol{\Delta}^T(i)\boldsymbol{\Delta}(i)\hat{\mathbf{f}}(i-1).\end{aligned} \tag{3.7}$$

Denoting $\mathbf{b}(i-1) = \hat{\mathbf{H}}(i-1)\hat{\mathbf{f}}(i-1)$, we obtain

$$\mathbf{b}(i-1) = [\hat{\mathbf{H}}(i-2) + \boldsymbol{\Delta}(i-1)][\hat{\mathbf{f}}(i-2) + \Delta\hat{\mathbf{f}}(i-1)],$$

which gives a recursion for $\mathbf{b}(i-1)$:

$$\mathbf{b}(i-1) = \mathbf{b}(i-2) + \hat{\mathbf{H}}(i-2)\Delta\hat{\mathbf{f}}(i-1) + \boldsymbol{\Delta}(i-1)\hat{\mathbf{f}}(i-1). \tag{3.8}$$

Note that $\boldsymbol{\Delta}(i-1)$ is a Toeplitz matrix whose first column is $\Delta\hat{h}(i-1)\mathbf{e}_{p(i-1)}$. We also have $\Delta\hat{\mathbf{f}}(i-1) = \Delta\hat{f}(i-1)\mathbf{e}_{q(i-1)}$. Then (3.8) can be rewritten as

$$\begin{aligned}\mathbf{b}(i-1) = {} & \mathbf{b}(i-2) + \Delta\hat{f}(i-1)\hat{\mathbf{h}}^{[q(i-1)]}(i-2) \\ & + \Delta\hat{h}(i-1)\hat{\mathbf{f}}^{[p(i-1)]}(i-1),\end{aligned}$$

where $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$ is a $(K+M-1)\times 1$ vector obtained by shifting elements of $\hat{\mathbf{h}}(i-2)$ by $q(i-1)$ positions down, and the other elements of $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$ are zeros. Definition for $\hat{\mathbf{f}}^{[p(i-1)]}(i-1)$ is similar to that of $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$. Thus, the first term on the right hand side of (3.7) is given by

$$\begin{aligned}\boldsymbol{\Delta}^T(i)\hat{\mathbf{H}}(i-1)\hat{\mathbf{f}}(i-1) &= \boldsymbol{\Delta}^T(i)\mathbf{b}(i-1) \\ &= \Delta\hat{h}(i)\mathbf{b}_{p(i):p(i)+K-1}(i-1),\end{aligned} \tag{3.9}$$

where $\mathbf{b}_{p(i):p(i)+K-1}(i-1)$ is a $K \times 1$ vector whose elements are obtained by extracting the $p(i)$th to $p(i)+K-1$th elements from the vector $\mathbf{b}(i-1)$.
Let $\hat{\mathbf{H}}^T(i-1)\mathbf{\Delta}(i) = \check{\mathbf{\Delta}}^T(i)\check{\mathbf{H}}^T(i-1)$, where $\check{\mathbf{\Delta}}(i)$ is a Toeplitz matrix whose first column is $\Delta\hat{h}(i)\mathbf{e}_{M-p(i)+1}$, and the matrix $\check{\mathbf{H}}(i)$ is given by

$$\check{\mathbf{H}}(i) = \begin{bmatrix} \hat{h}_M(i) & 0 & \cdots & 0 & 0 \\ \hat{h}_{M-1}(i) & \hat{h}_M(i) & \ddots & \vdots & \vdots \\ \vdots & \hat{h}_{M-1}(i) & \ddots & 0 & \vdots \\ \hat{h}_1(i) & \vdots & \ddots & \hat{h}_M(i) & 0 \\ 0 & \hat{h}_1(i) & \ddots & \hat{h}_{M-1}(i) & \hat{h}_M(i) \\ \vdots & 0 & \ddots & \vdots & \hat{h}_{M-1}(i) \\ \vdots & \vdots & \ddots & \hat{h}_1(i) & \vdots \\ 0 & 0 & \cdots & 0 & \hat{h}_1(i) \end{bmatrix}.$$

The second term on the right hand side of (3.7) can then be expressed as

$$\hat{\mathbf{H}}^T(i-1)\mathbf{\Delta}(i)\hat{\mathbf{f}}(i-1) = \check{\mathbf{\Delta}}^T(i)\check{\mathbf{H}}^T(i-1)\hat{\mathbf{f}}(i-1).$$

Denoting $c(i-1) = \check{\mathbf{H}}^T(i-1)\hat{\mathbf{f}}(i-1)$, we obtain

$$\mathbf{c}(i-1) = [\check{\mathbf{H}}(i-2) + \check{\mathbf{\Delta}}(i-1)][\hat{\mathbf{f}}(i-2) + \Delta\hat{\mathbf{f}}(i-1)],$$

which gives a recursion for $\mathbf{c}(i-1)$:

$$\begin{aligned} \mathbf{c}(i-1) &= \mathbf{c}(i-2) + \check{\mathbf{H}}(i-2)\Delta\hat{\mathbf{f}}(i-1) + \check{\mathbf{\Delta}}(i-1)\hat{\mathbf{f}}(i-1) \\ &= \mathbf{c}(i-2) + \Delta\hat{f}(i-1)\hat{\mathbf{u}}^{[q(i-1)]}(i-2) \\ &\quad + \Delta\hat{h}(i-1)\hat{\mathbf{f}}^{[M-p(i-1)+1]}(i-1), \end{aligned}$$

where elements of the vector $\hat{\mathbf{u}}(i-2)$ are given by

$$\hat{u}_m(i-2) = \hat{h}_{M-m+1}(i-2), m = 1, \ldots, M.$$

The second term on the right hand side of (3.7) is now given by

$$\begin{aligned} \hat{\mathbf{H}}^T(i-1)\mathbf{\Delta}(i)\hat{\mathbf{f}}(i-1) &= \check{\mathbf{\Delta}}^T(i)\mathbf{c}(i-1) \\ &= \Delta\hat{h}(i)\mathbf{c}_{M-p(i)+1:M-p(i)+K}(i-1). \end{aligned} \tag{3.10}$$

Since $\mathbf{\Delta}^T(i)\mathbf{\Delta}(i) = \Delta\hat{h}^2(i)\mathbf{I}_K$, the third term on the right hand side of (3.7) is given by

$$\mathbf{\Delta}^T(i)\mathbf{\Delta}(i)\hat{\mathbf{f}}(i-1) = \Delta\hat{h}^2(i)\hat{\mathbf{f}}(i-1). \tag{3.11}$$

From (3.9), (3.10) and (3.11), denoting $\mathbf{z}(i) = \Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$, we finally obtain a simplified expression for (3.7):

$$\mathbf{z}(i) = \Delta\hat{h}(i)\left[\mathbf{b}_{p(i):p(i)+K-1}(i-1) + \mathbf{c}_{M-p(i)+1:M-p(i)+K}(i-1) + \Delta\hat{h}(i)\hat{\mathbf{f}}(i-1)\right].$$

Computation of $\Delta\mathbf{G}(i)$:

The matrix $\Delta\mathbf{G}(i)$ can be obtained by using (3.6), which can also be written as

$$\Delta\mathbf{G}(i) = \mathbf{\Delta}^T(i)\hat{\mathbf{H}}(i) + \hat{\mathbf{H}}^T(i-1)\mathbf{\Delta}(i). \tag{3.12}$$

Since the matrix $\mathbf{G}(i)$ is a symmetric Toeplitz matrix, $\Delta\mathbf{G}(i)$ is also a symmetric Toeplitz matrix. Therefore, for each update of the channel estimate, only the first column of $\Delta\mathbf{G}(i)$ given by (3.12) needs to be updated. In this column, only the first M elements are nonzero, which are given by

$$\begin{aligned}\Delta G_{1,1}(i) &= \Delta\hat{h}(i)\left[\hat{h}_{p(i)}(i) + \hat{h}_{p(i)}(i-1)\right], \\ \Delta G_{1,m}(i) &= \Delta\hat{h}(i)\left[\hat{h}_{p(i)-m+1}(i-1) + \hat{h}_{p(i)+m-1}(i-1)\right],\end{aligned} \tag{3.13}$$

where $m = 2, \ldots, M$.

The proposed technique for computing the equalizer coefficients is now summarized in Table 3.2. Here, we assume that the noise variance σ_ν^2 is known. Table 3.2 also shows the complexity of the computation steps in terms of multiplications and additions. The complexity of computing the LE coefficients will depend on the iterative technique used for solving the equation $\mathbf{G}\Delta\mathbf{f} = \boldsymbol{\xi}_0$ at step 6, where P_{mu} and P_{ad} denote the number of multiplications and additions, respectively.

3.3.3 DCD iterations

We propose to use the DCD iteration described in Table 3.3, which is simple for implementation and shows fast convergence to optimal performance [17]. When using the DCD iteration, it is assumed that the equalizer coefficients are represented as M_b-bit fixed-point numbers within an interval $[-A, A]$, where A is preferably a power-of-two number. The

Table 3.2: Low-complexity computation of CE based equalizer coefficients

Step	Equation	×	+
	Initialization: $i=0$, $\mathbf{G}(0)=\sigma_\nu^2\mathbf{I}_K$, $\hat{\mathbf{f}}(0)=\mathbf{0}$, $\mathbf{r}(0)=\mathbf{0}$, $\mathbf{b}(0)=\mathbf{0}$, $\mathbf{c}(0)=\mathbf{0}$		
	for $n=1,2,\ldots$		
	for $k=1,\ldots,N_u$		
1	$i=i+1$		
2	Obtain $\hat{\mathbf{h}}(i)$, $\Delta\hat{h}(i)$ and position $p(i)$ from a channel estimator		
3	$\mathbf{z}(i)=\Delta\hat{h}(i)[\mathbf{b}_{p(i):p(i)+K-1}(i-1)+\mathbf{c}_{M-p(i)+1:M-p(i)+K}(i-1)$ $+\Delta\hat{h}(i)\hat{\mathbf{f}}(i-1)]$	$2K$ $2K$	$2K$ $2K$
4	$\boldsymbol{\xi}_0=\mathbf{r}(i-1)+\Delta\hat{h}(i)\mathbf{e}_{l+p(i)}-\mathbf{z}(i)$	–	$K+1$
5	Compute $\Delta\mathbf{G}^{(1)}(i)$ using (3.13) and update $\mathbf{G}^{(1)}(i)=\mathbf{G}^{(1)}(i-1)+\Delta\mathbf{G}^{(1)}(i)$	M	$2M$
6	Use one iteration to solve $\mathbf{G}\Delta\mathbf{f}=\boldsymbol{\xi}_0$ and obtain $\Delta\hat{f}(i)$, $q(i)$, and $\mathbf{r}(i)$	P_{mu}	P_{ad}
7	$\hat{f}_{q(i)}(i)=\hat{f}_{q(i)}(i-1)+\Delta\hat{f}(i)$	–	1
8	$\mathbf{b}(i)=\mathbf{b}(i-1)+\Delta\hat{f}(i)\hat{\mathbf{h}}^{[q(i)]}(i-1)+\Delta\hat{h}(i)\hat{\mathbf{f}}^{[p(i)]}(i)$	$K+M$	$K+M$
9	$\mathbf{c}(i)=\mathbf{c}(i-1)+\Delta\hat{f}(i)\hat{\mathbf{u}}^{[q(i)]}(i-1)+\Delta\hat{h}(i)\hat{\mathbf{f}}^{[M-p(i)+1]}(i)$	$K+M$	$K+M$
	Total for each sample n: $N_u(4K+3M+P_{mu})$ mult. and $N_u(6K+4M+1+P_{ad})$ adds		

Table 3.3: DCD algorithm with one update

Step	Equation	+
	Initialization: $\mathbf{r}=\boldsymbol{\xi}_0$, $\alpha=A/2$, $a=0$	
1	$q=\arg\max_{j=1,\ldots,K}\{\lvert r_j\rvert\}$, go to step 4	$K-1$
2	$a=a+1$, $\alpha=\alpha/2$	–
3	if $a>M_b$, the algorithm stops	–
4	if $\lvert r_q\rvert\le(\alpha/2)G_{q,q}$, then go to step 2	1
5	$\Delta\hat{f}=\text{sign}(r_q)\alpha$	1
6	$\mathbf{r}=\mathbf{r}-\text{sign}(r_q)\alpha\mathbf{G}^{(q)}$	K
	$\Delta\hat{f}(i)=\Delta\hat{f}$, $q(i)=q$, $\mathbf{r}(i)=\mathbf{r}$	
	Total: $P_{mu}=0$ and $P_{ad}\le 2K+M_b+1$	

step-size parameter α is $\alpha=2^{-a}A$, i.e. also a power-of-two number. With such settings, operations required in the DCD algorithm are only additions as all multiplications and divisions are replaced by bit-shifts; see more details on the parameter choice in [17]. If, in addition, the adaptive channel estimator is implemented using the RLS-DCD adaptive filter of complexity $\mathcal{O}(N_uM)$ [17], the increments $\Delta\hat{h}(i)$ will be power-of-two numbers. Therefore, all multiplications in Table 3.2 can be replaced by bit-shift operations. With the DCD iteration, step 6 in Table 3.2 is multiplication-free and requires no more than $2K+M_b+1$ additions.

3.4 Simulation results

In this section, we compare the performance of seven LEs:

1) MMSE LE. For every time sample n, the convolution matrix $\mathbf{H}(n)$ is formed using the perfectly known channel response $\mathbf{h}(n-j), j = 0, \ldots, K-1$, instead of its estimate as in (3.3). The equalizer vector $\hat{\mathbf{f}}_0$ is found by solving (3.2);

2) RLS CE based adaptive LE. The time-varying channel is estimated using the classical RLS algorithm with a forgetting factor λ, and for every sample n, the convolution matrix $\mathbf{H}(n)$ is formed using (3.3). The equalizer vector $\hat{\mathbf{f}}$ is obtained by solving (3.2);

3) LMS CE based adaptive LE. This is similar to the RLS CE based adaptive LE except that the time-varying channel is estimated using the classical LMS algorithm [4];

4) RLS CE based adaptive LE (K samples). This is the RLS CE based adaptive LE which estimates the time-varying channel for every sample n, while the equalizer coefficients are computed once for K samples;

5) RLS directly adaptive (DA) LE. The equalizer coefficients are directly computed for every sample n using the RLS algorithm [21];

6) LMS DA LE. The equalizer coefficients are directly computed for every sample n using the LMS algorithm [21];

7) Proposed LE. The time-varying channel is estimated using the RLS-DCD algorithm from [17] with a forgetting factor λ and for every i, the leading index $p(i)$ is chosen according to the position of the maximum in the residual vector (see [17]). The choice of N_u for the DCD algorithm is investigated in [17, 52]. The equalizer vector $\hat{\mathbf{f}}$ is obtained using the algorithm in Table 3.2.

To simulate the time-varying channel impulse response $\mathbf{h}(n)$, we adopt the first order autoregressive model given by $\mathbf{h}(n) = \sqrt{\upsilon}\,\mathbf{h}(n-1) + \sqrt{1-\upsilon}\,\boldsymbol{\omega}(n)$ [54], where $\sqrt{\upsilon}$ is the autoregressive factor and $\boldsymbol{\omega}(n)$ are zero-mean independent random Gaussian vectors,

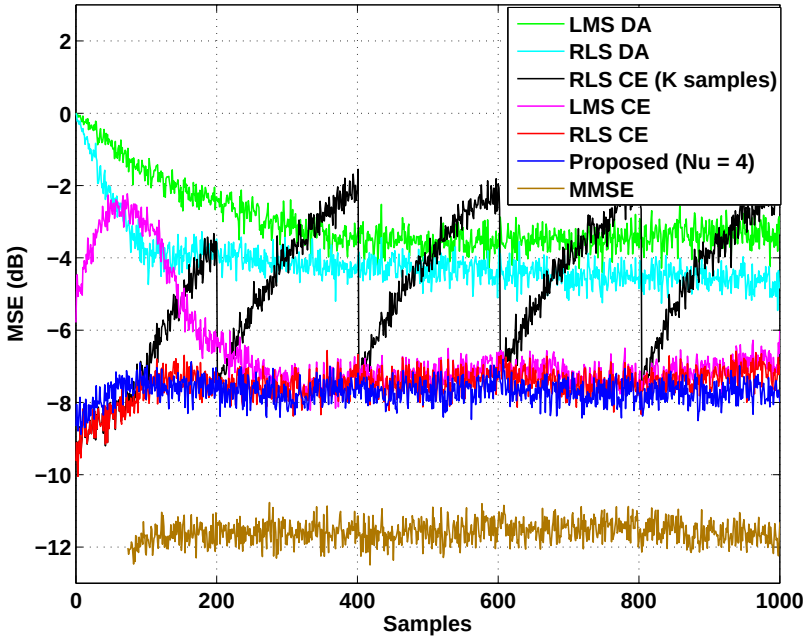


Figure 3.3: MSE performance of LEs: SNR $= 20$ dB, $\upsilon = 1 - 10^{-3}$, $M = 51$, $K = 201$, $M_b = 16$, $A = 1$, forgetting factor is 0.9804 for the RLS CE and the proposed LE and 0.995 for the RLS DA LE, step size is 0.005 for the LMS DA LE and 0.02 for the LMS CE.

whose elements have variance $1/M$. The channel length is $M = 51$ and the equalizer length is $K = 201$. We use $l = (K + M)/2 = 126$ as the equalizer delay [78]. Different signal to noise ratios (SNRs) are considered, and for each SNR, simulation results are obtained by averaging over 500 independent simulation trials. For each trial, 1000 BPSK pilot symbols of unit power are transmitted.

Fig.3.3 compares the MSE performance of the seven LEs for SNR $= 20$ dB and the time-varying channel with $\upsilon = 1 - 10^{-3}$. For computing the MSE for each n, a 1000-length data sequence independent of the pilot is filtered with the equalizer vector $\hat{\mathbf{f}}(n)$ derived using the pilot. It is seen that the proposed LE performs very close to the RLS CE based adaptive LE and outperforms the other LEs.

Fig.3.4 and Fig.3.5 compare the MSE performance of LEs at different SNRs, for time-varying channels with $\upsilon = 1-10^{-5}$ and $\upsilon = 1-10^{-4}$, respectively. For a simulation trial, the steady-state MSE is evaluated as MSE $= \frac{1}{926}\sum_{n=75}^{1000}[x(n) - \mathbf{y}^T(n)\hat{\mathbf{f}}(n)]^2$. It is seen

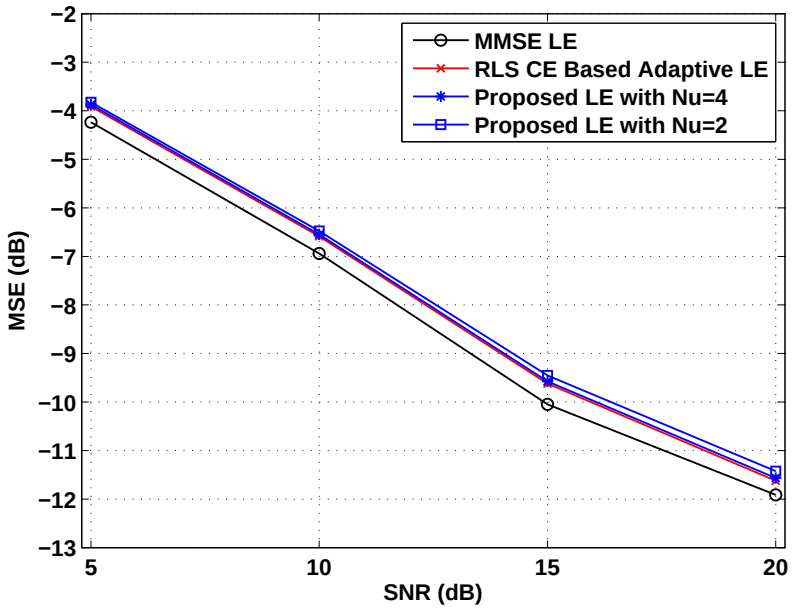


Figure 3.4: MSE performance of the three LEs at different SNRs: $\upsilon = 1-10^{-5}$, $M = 51$, $K = 201$, $M_b = 16$, $A = 1$; forgetting factor is 0.9975 for SNR $= 5$ and 10 dB, 0.9951 for SNR $= 15$ and 20 dB, and 0.9902 for SNR $= 25$ dB.

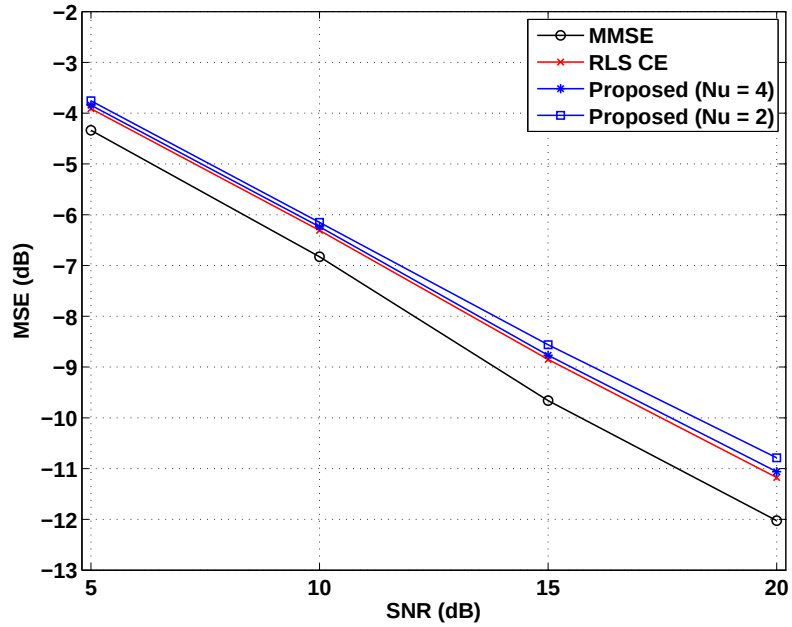


Figure 3.5: Steady-state MSE performance of the three LEs at different SNRs: $\upsilon = 1-10^{-4}$, $M = 51$, $K = 201$, $M_b = 16$, $A = 1$; forgetting factor is 0.9951 for SNR $= 5$ and 10 dB, 0.9902 for SNR $= 15$ and 20 dB, and 0.9804 for SNR $= 25$ dB.

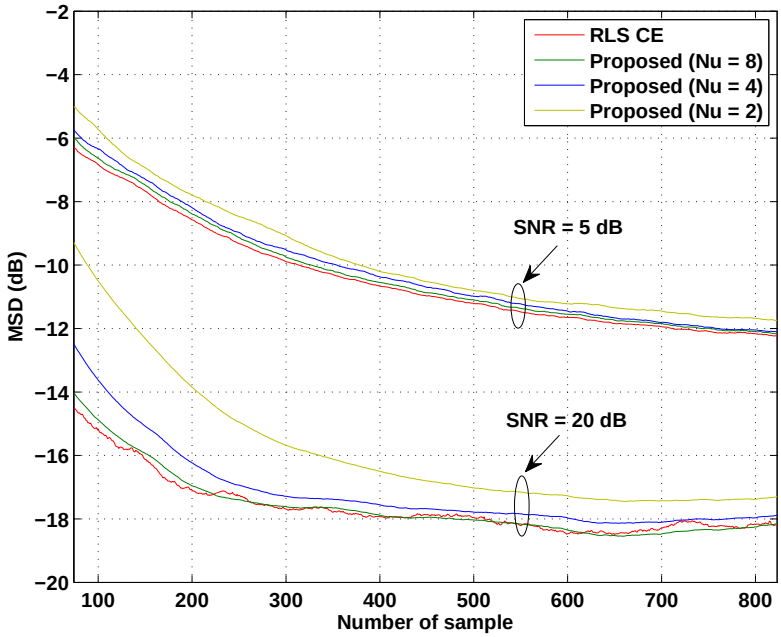


Figure 3.6: MSD performance of the proposed LE and the RLS CE based adaptive LE: $\upsilon = 1 - 10^{-5}$, $M = 51$, $K = 201$, $M_b = 16$, $A = 1$; forgetting factor is 0.9975 for SNR $= 5$, and 0.9951 for SNR $= 20$ dB.

that with $N_u = 4$ and even $N_u = 2$, the proposed LE provides performance very close to that of the RLS CE based adaptive LE, and close to that of the MMSE LE. We also applied the proposed LE with $N_u = 2$ but with 2 iterations for computing the equalizer coefficients after an update of the channel estimate, to the same simulation trials as we ran to obtained Fig.3.5. We observed for SNR $= 15$ and 20 dB, little improvement of 0.07 dB in the MSE performance of the equalizer compared to the performance of the proposed LE with $N_u = 2$ as shown in Fig.3.5.

Fig.3.6 compares the mean square deviation (MSD) performance of the proposed LE and the RLS CE based adaptive LE. For each LE, at every time sample n, the MSD is evaluated as $\text{MSD}(n) = [\hat{\mathbf{f}}_0(n) - \hat{\mathbf{f}}(n)]^T[\hat{\mathbf{f}}_0(n) - \hat{\mathbf{f}}(n)]/[\hat{\mathbf{f}}_0^T(n)\hat{\mathbf{f}}_0(n)]$, where $\hat{\mathbf{f}}_0(n)$ is the equalizer vector obtained from the MMSE LE. From Fig.3.6, it is seen that as N_u increases, the proposed LE performs close to the RLS CE based adaptive LE.

Table 3.4 compares the number of multiplications required in different LEs at each sample for different equalizer lengths. For the MMSE LE, only the multiplications in-

Table 3.4: Number of multiplications per sample ($M = 51$)

		$K = 51$	$K = 101$	$K = 201$	$K = 401$
MMSE		3.9×10^5	2.5×10^6	1.8×10^7	1.3×10^8
RLS CE		4.1×10^5	2.6×10^6	1.8×10^7	1.3×10^8
RLS CE (K samples)		8113	2.6×10^4	9.1×10^4	3.4×10^5
LMS CE		3.9×10^5	2.5×10^6	1.8×10^7	1.3×10^8
RLS DA		1.6×10^4	6.2×10^4	2.4×10^5	9.7×10^5
LMS DA		153	303	603	1203
Proposed	$N_u = 2$	1020	1470	2370	4170
	$N_u = 4$	1734	2584	4284	7684

volved in the computation of the equalizer coefficients are considered, where the equalizer coefficients are obtain by solving (3.2) directly. For other CE based LEs, the multiplications involved in both the channel estimation and the computation of the equalizer coefficients are taken in to account, where the equalizer coefficients are also obtain by solving (3.2) directly. From Table 3.4, we can find that, for CE based LEs, the complexity of computing the equalizer coefficients determines the total complexity. It is seen that the proposed LE has much lower computational complexity than the other CE based LEs. Its complexity is also significantly lower than that of the RLS DA LE.

3.5 Conclusions

In this chapter, we have proposed a channel-estimate based adaptive LE with a complexity as low as $\mathcal{O}(N_u(K+M))$ operations per sample, where $N_u \ll K$ and $N_u \ll M$. The proposed technique exploits coordinate descent iterations for computing the equalizer coefficients. Moreover, when using the dichotomous coordinate descent iterations, computation of the equalizer coefficients is multiplication-free and division-free, which makes it attractive for hardware design. Simulation results show that, with only a few updates per sample, the proposed LE performs very close to the RLS CE based adaptive LE and close to the MMSE LE with perfect knowledge of the channel. It is shown in [21] that the MMSE LE may have poor performance on channels with severe inter-symbol interference, while the MMSE decision-feedback equalizer (DFE) outperforms the MMSE LE and yields good performance, provided that the decision errors are negligible. In the next two chapters, we will introduce two partial-update CE based adaptive DFEs, both of which are based on DCD iterations for computing the equalizer coefficients.

Chapter 4

Partial-Update Channel-Estimate Based Adaptive Decision Feedback Equalizer: Approach 1

Contents

In the previous chapter, we have proposed a low-complexity channel-estimate (CE) based adaptive linear equalizer, which performs very close to the RLS CE based adaptive linear equalizer (LE) and close to the minimum mean-square error (MMSE) LE with perfect knowledge of the channel. However, for channels with severe inter-symbol interference, such as the underwater acoustic channel, the MMSE decision-feedback equalizer (DFE) can outperform the MMSE LE, provided that the effect of decision errors on per-

formance is negligible [21].

In this chapter, we propose a novel CE based adaptive DFE. In addition to the low complexity computation of equalizer taps, the proposed technique can operate with low-complexity channel estimators. Specifically, it is assumed that every CE update involves only one channel tap, e.g. the CE is generated by a partial update adaptive filter. The proposed DFE exploits DCD iterations and has complexity as low as $\mathcal{O}(N_u l \log_2 2l)$ multiplications per sample, where l is the equalizer delay and N_u is the number of updates per sample such that $N_u \ll l$. For every update of the equalizer, only one feedforward filter tap is updated, while feedback filter taps can be computed recursively, or they do not need to be computed when a modified DFE structure is used. Thus, we have a partial-update equalizer.

The proposed DFE is especially efficient when the channel estimator also exploits the DCD iterations, e.g. such as in the RLS-DCD adaptive filter. Then the channel estimator has a complexity significantly lower than that of the equalizer. Moreover, most of the multiplications involved in the computation of the equalizer coefficients can now be replaced by bit-shift operations, which makes the equalizer attractive for hardware design. Simulation results show that the proposed DFE performs very close to the CE based DFE, where the CE is obtained using the classical RLS adaptive filter and the equalizer taps are computed according to the MMSE criterion.

This chapter is organized as follows. In the next section, an introduction is given. In Section 4.2, we introduce the structure of the conventional DFE and give expressions for computing the DFE taps. Section 4.3 introduces assumptions made for deriving the partial-update DFE. The complex-valued DCD iterations and RLS-DCD algorithm for adaptive channel estimation are presented in Section 4.4. In Section 4.5, a low-complexity approach for computing the FFF taps is proposed. In Section 4.6, recursive computation of the FBF taps is derived, and the modified DFE structure is introduced. Section 4.7 presents numerical results that demonstrate the performance and computational complexity of the proposed DFE against known techniques. Finally, Section 4.8 draws conclusions.

4.1 Introduction

Decision-feedback equalizers (DFEs) are widely used for combatting the inter-symbol interference in communication channels [21,79]. Taps of an adaptive DFE can be computed without explicit channel estimation, by direct adaptation (DA) using the channel output and known pilot signal [21]. However, channel-estimate (CE) based DFEs can outperform DFEs with the direct adaptation [22]. As the demand for broadband communications increases, the computational complexity of CE based DFEs becomes an important issue. Extensive effort has been made to reduce the complexity (see [6, 7, 23–30] and references therein). However, the complexity normally grows squarely with the length K of the feedforward filter (FFF). In multipath channels with large delay spread and long precursor part, such as in underwater acoustic channels [31], the FFF length K needs to be large enough to equalize the precursor part, and it is usual that $K > M$, where M is the channel estimator length. Reducing the complexity of CE based DFEs in such scenarios is still an open issue.

In [7], an efficient approach for computing the minimum mean-square error (MMSE) DFE taps [6] with a complexity of $\mathcal{O}(K(K+M))$ is proposed. It relies on fast Cholesky factorization, which is still difficult for practical implementation. In [26], a DFE with a complexity of $\mathcal{O}((2K+1)^2)$ is proposed.

In [28, 80], an alternative approach for fast computation of MMSE DFE taps is proposed; it has a complexity of $\mathcal{O}(lK + K\log_2 2K))$ for $K > M$, where l is the equalizer delay. In [28], the FFF taps are obtained by solving a set of linear equations using the fast RLS algorithm [81], and the feedback filter (FBF) taps are computed by convolving the FFF taps with the channel impulse response. In [28], the fast RLS algorithm is simplified and it is stated to have some advantages in stabilization for finite precision implementation. However, there is still no guarantee of stability in practice, and such a fast RLS algorithm can still exhibit instability [5].

In Chapter 3, we proposed a CE based adaptive linear equalizer with a complexity as low as $\mathcal{O}(N_u K) + \mathcal{O}(N_u M)$, where N_u is the number of updates such that $N_u \ll K$ and $N_u \ll M$. The equalizer exploits dichotomous coordinate descent (DCD) iterations for the tap computation. It is attractive to apply this approach for fast computation of DFE

taps. However, the approach in Chapter 3 is not directly applicable to the computation of the DFE taps. Besides, the approach in Chapter 3 has been derived for real-valued signals and channels. Moreover, complex-valued DCD iterations have not been introduced yet. The linear equalizer from Chapter 3 is especially efficient when using the RLS-DCD adaptive filter [17] as the channel estimator. In this case, the tap computation is multiplication-free and division-free, which is attractive for hardware implementation [52]. However, an RLS-DCD adaptive filter for complex-valued signals and channels has not been introduced yet.

In this chapter, we propose a novel CE based adaptive DFE which can operate together with partial-update channel estimators (see [75–77] and reference therein). The proposed DFE exploits complex-valued DCD iterations (that we introduce here) to efficiently compute the DFE taps. The equalizer tap computation has a complexity as low as $\mathcal{O}(N_u l \log_2 2l))$ multiplications per sample. The proposed DFE is especially efficient if the channel estimation is performed by the complex-valued RLS-DCD adaptive filter that we also introduce here. The DFE can be implemented in the conventional or modified structure [24–26]. For the conventional structure, we also propose a simple recursive method for computing the FBF taps, whereas the modified structure does not require computing the FBF.

Notations: We use capital and small bold fonts to denote matrices and vectors, respectively; e.g. $\mathbf{G}$ is a matrix and $\mathbf{r}$ a vector. Elements of the matrix and vector are denoted as $G_{n,p}$ and r_n, respectively. A pth column and nth row of $\mathbf{G}$ are denoted as $\mathbf{G}^{(p)}$ and $\mathbf{G}_{(n)}$, respectively. We also denote: $\mathbf{G}^T$ and $\mathbf{G}^H$ are transpose and conjugate transpose of the matrix $\mathbf{G}$, respectively; $\mathbf{r}^*$ is the complex conjugate of vector $\mathbf{r}$; $\mathbf{I}_K$ is a $K \times K$ identity matrix; $\mathbf{0}_{K\times M}$ is a $K \times M$ matrix of all zeros; $E\{\cdot\}$ is the expectation; $\Re\{\cdot\}$ and $\Im\{\cdot\}$ are the real and imaginary part of a complex number, respectively. The variable n is used as a time index and i is the iteration index.

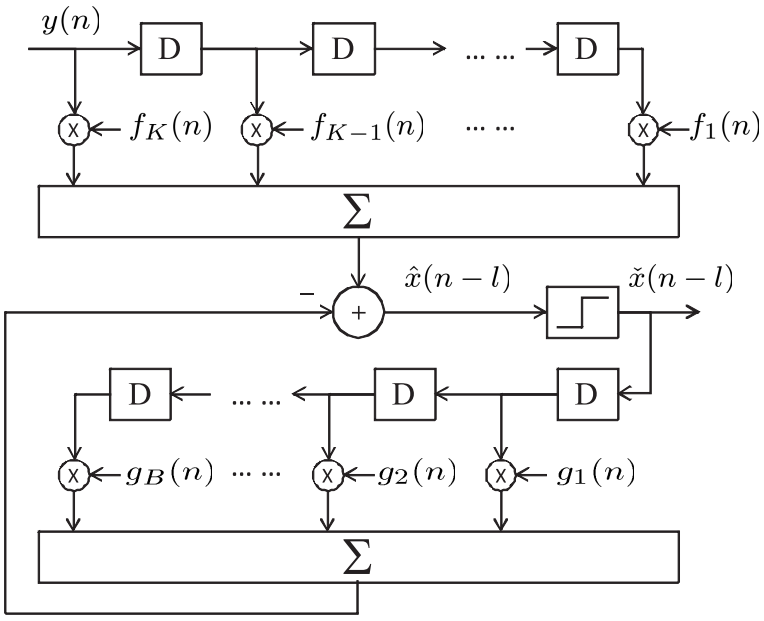


Figure 4.1: Conventional structure of a symbol-spaced DFE.

4.2 MMSE decision-feedback equalizer

We consider that the received signal $y(n)$ at a time-instant n is given by

$$y(n) = \mathbf{x}^T(n)\mathbf{h}(n) + \nu(n) \tag{4.1}$$

where $\mathbf{h}(n) = [h_1(n)\, h_2(n) \ldots\, h_M(n)]^T$ is the channel impulse response, $\mathbf{x}(n) = [x(n)\, x(n-1) \ldots\, x(n-M+1)]^T$ is a sequence of transmitted symbols, and $\nu(n)$ is the white noise with zero mean and variance σ_ν^2; $\mathbf{x}(n)$, $\mathbf{h}(n)$, and $\nu(n)$ are complex-valued.

Fig.4.1 shows the structure of a DFE [21] consisting of a K-length FFF with the tap vector $\mathbf{f}(n)$ and B-length FBF with the tap vector $\mathbf{g}(n)$. At time instant n, the DFE estimates the transmitted symbol $x(n-l)$ as

$$\hat{x}(n-l) = \mathbf{y}^T(n)\mathbf{f}(n) - \check{\mathbf{x}}^T(n)\mathbf{g}(n) \tag{4.2}$$

where $\mathbf{y}(n) = [y(n)\, y(n-1)\, \ldots y(n-K+1)]^T$, $\check{\mathbf{x}}(n) = [\check{x}(n-l-1)\, \ldots\, \check{x}(n-l-B)]^T$ and l is the equalizer delay. The received data vector $\mathbf{y}(n)$ can be expressed as

$$\mathbf{y}(n) = \mathbf{H}^T(n)\bar{\mathbf{x}}(n) + \boldsymbol{\nu}(n) \tag{4.3}$$

where $\bar{\mathbf{x}}(n) = [x(n)\, x(n-1) \ldots\, x(n-M-K+2)]^T$, $\boldsymbol{\nu}(n) = [\nu(n)\, \nu(n-1) \ldots \nu(n-K+1)]^T$ and $\mathbf{H}(n)$ is a $(K+M-1) \times K$ time-varying

channel convolution matrix

$$\mathbf{H}(n) = \begin{bmatrix} h_1(n) & 0 & \cdots & 0 & 0 \\ h_2(n) & h_1(n-1) & \ddots & \vdots & \vdots \\ \vdots & h_2(n-1) & \ddots & 0 & \vdots \\ h_M(n) & \vdots & \ddots & h_1(n-K+2) & 0 \\ 0 & h_M(n-1) & \ddots & h_2(n-K+2) & h_1(n-K+1) \\ \vdots & 0 & \ddots & \vdots & h_2(n-K+1) \\ \vdots & \vdots & \ddots & h_M(n-K+2) & \vdots \\ 0 & 0 & \cdots & 0 & h_M(n-K+1) \end{bmatrix}. \quad (4.4)$$

By introducing two $(B+K)\times 1$ vectors

$$\mathbf{s}(n) = \begin{bmatrix} \mathbf{y}(n) \\ \hline -\check{\mathbf{x}}(n) \end{bmatrix} \text{ and } \mathbf{w}(n) = \begin{bmatrix} \mathbf{f}(n) \\ \hline \mathbf{g}(n) \end{bmatrix},$$

the symbol estimate (4.2) can be rewritten as

$$\hat{x}(n-l) = \mathbf{s}^T(n)\mathbf{w}(n). \quad (4.5)$$

The vectors $\mathbf{f}(n)$ and $\mathbf{g}(n)$ are adjusted to minimize the mean square error (MSE)

$$E\{|x(n-l) - \hat{x}(n-l)|^2\} = E\{|x(n-l) - \mathbf{s}^T(n)\mathbf{w}(n)|^2\}.$$

Solving this minimization problem is equivalent to solving a set of linear equations [21]

$$\mathbf{\Gamma}(n)\mathbf{w}(n) = \boldsymbol{\zeta}_s(n), \quad (4.6)$$

where $\mathbf{\Gamma}(n) = E\{\mathbf{s}(n)\mathbf{s}^H(n)\}$ and $\boldsymbol{\zeta}_s(n) = E\{\mathbf{s}(n)x^*(n-l)\}$.

In the DA DFE, $\mathbf{\Gamma}(n)$ and $\boldsymbol{\zeta}_s(n)$ are estimated without explicit channel estimation using the received data and known pilot or estimated data symbols. The equalizer taps are then obtained by solving (4.6) using an adaptive algorithm [21, 82], such as the RLS algorithm [4, 5]. Fig.4.2 shows the block diagram of the DA DFE.

For a CE based MMSE equalizer, assuming that the transmitted symbols $x(n)$ are independent and identically distributed with unit power, the matrix $\mathbf{\Gamma}(n)$ and vector $\boldsymbol{\zeta}_s(n)$ can be represented as [28]

$$\mathbf{\Gamma}(n) = \begin{bmatrix} \mathbf{G}(n) & -\tilde{\mathbf{H}}^H(n) \\ \hline -\tilde{\mathbf{H}}(n) & \mathbf{I}_B \end{bmatrix} \quad (4.7)$$

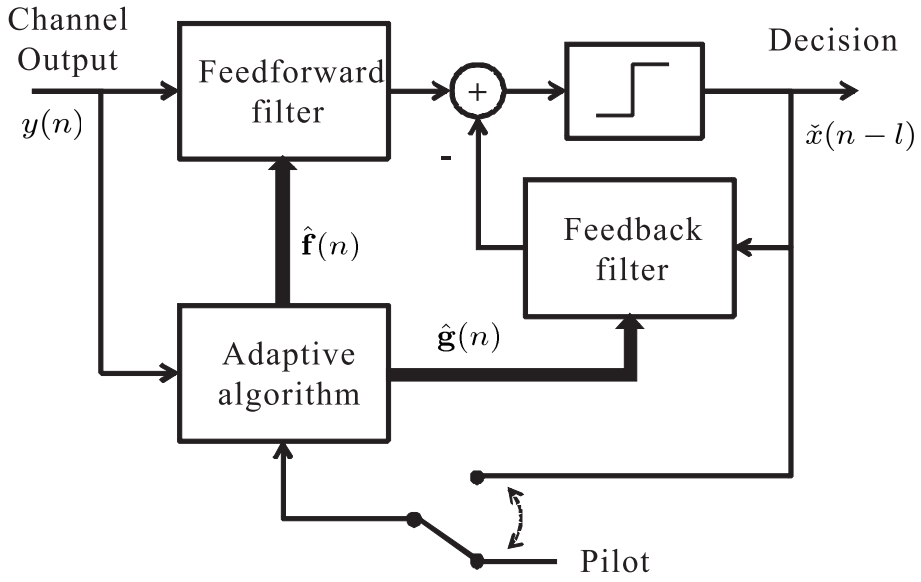


Figure 4.2: Direct adaptation DFE.

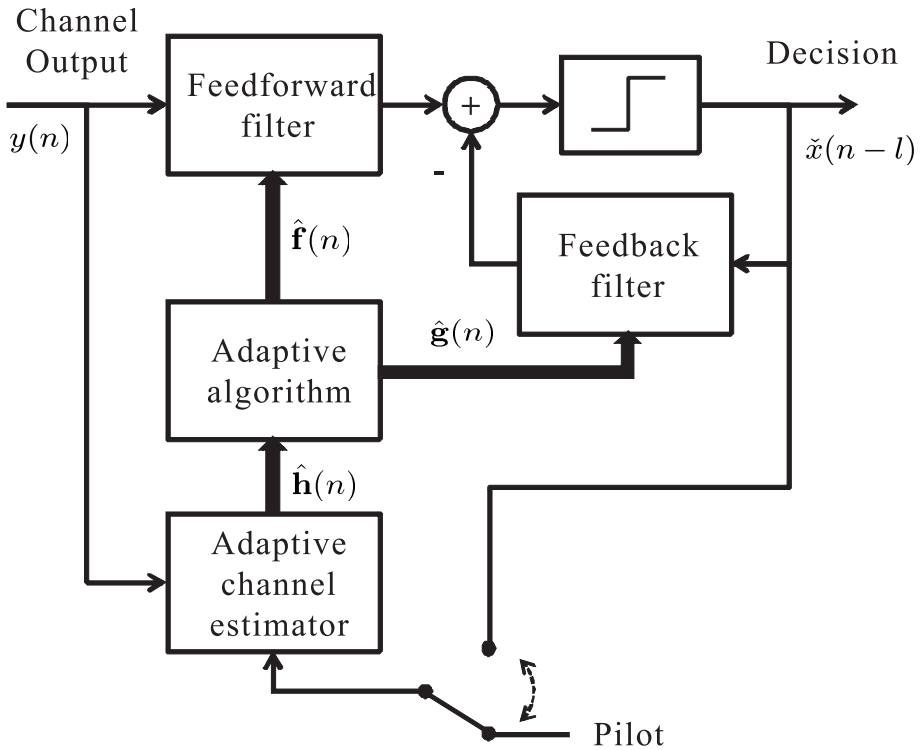


Figure 4.3: Channel estimate based adaptive DFE.

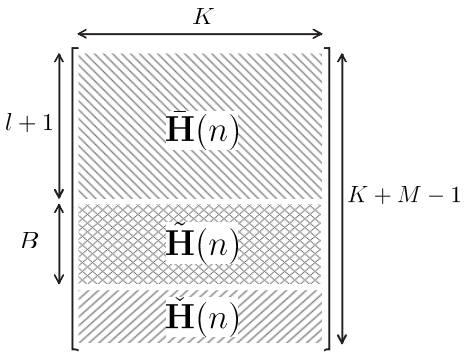


Figure 4.4: Submatrices of the channel convolution matrix $\mathbf{H}(n)$

and

$$\boldsymbol{\zeta}_s(n) = \left[\begin{array}{c} \boldsymbol{\zeta}(n) \\ \hline \mathbf{0}_{B\times 1} \end{array}\right], \tag{4.8}$$

where

$$\mathbf{G}(n) = \mathbf{H}^H(n)\mathbf{H}(n) + \sigma_\nu^2 \mathbf{I}_K, \tag{4.9}$$

and

$$\boldsymbol{\zeta}(n) = \mathbf{H}^H(n)\mathbf{e}_l. \tag{4.10}$$

$\tilde{\mathbf{H}}(n)$ is a $B \times K$ submatrix of the channel convolution matrix $\mathbf{H}(n)$ as illustrated in Fig.4.4; $\mathbf{e}_l$ is a $(K+B-1)\times 1$ vector of all zeros except the lth element, which equals one and corresponds to the equalizer delay.

In the CE based adaptive DFE, shown in Fig.4.3, the channel impulse response is estimated by using the received data and pilot or estimated data symbols. The quantities $\boldsymbol{\Gamma}(n)$ and $\boldsymbol{\zeta}_s(n)$ are then computed using (4.7) and (4.8), respectively. Finally, the equalizer taps are obtained by solving (4.6) using an adaptive algorithm.

Using (4.6), (4.7) and (4.8), we obtain

$$\mathbf{G}(n)\mathbf{f}(n) - \tilde{\mathbf{H}}^H(n)\mathbf{g}(n) = \boldsymbol{\zeta}(n), \tag{4.11}$$

$$\mathbf{g}(n) = \tilde{\mathbf{H}}(n)\mathbf{f}(n). \tag{4.12}$$

Substituting (4.12) into (4.11), we have

$$\left[\bar{\mathbf{H}}^H(n)\bar{\mathbf{H}}(n) + \check{\mathbf{H}}^H(n)\check{\mathbf{H}}(n) + \sigma_\nu^2\mathbf{I}_K\right]\mathbf{f}(n) = \boldsymbol{\zeta}(n). \tag{4.13}$$

where $\bar{\mathbf{H}}(n)$ and $\check{\mathbf{H}}(n)$ are submatrices of the channel convolution matrix $\mathbf{H}(n)$ as illustrated in Fig.4.4. The size of matrix $\bar{\mathbf{H}}(n)$ is $(l+1) \times K$.
Since we consider $B \geq M$, equation (4.13) can be rewritten as [28]

$$\bar{\mathbf{G}}(n)\mathbf{f}(n) = \boldsymbol{\zeta}(n), \tag{4.14}$$

where $\bar{\mathbf{G}}(n) = \bar{\mathbf{H}}^H(n)\bar{\mathbf{H}}(n)$. The MMSE DFE taps can also be obtained by solving the normal equations (4.14) for the FFF vector $\mathbf{f}(n)$ using an adaptive algorithm, and then computing the FBF vector $\mathbf{g}(n)$ using (4.12). For the CE based DFE, the FFF and FBF taps can also be obtained by solving separate MMSE optimization problems [83]. In [84], the separate MMSE DFE optimization has been demonstrated to be equivalent to the simultaneous optimization, which is considered in this chapter.

4.3 Assumptions

For computation of the DFE taps, we use the following assumptions:

1) In practice, as the time-varying channel is unknown, estimates $\hat{\mathbf{h}}(n)$ of the channel impulse response $\mathbf{h}(n)$ are used for computing the DFE taps.

2) For every time sample n, the channel impulse response estimate $\hat{\mathbf{h}}(n)$ can be updated N_u times. We will be using index i to indicate such an update. Correspondingly, the sequence of normal equations to be solved for the FFF taps is now given by

$$\bar{\mathbf{G}}(i)\mathbf{f}(i) = \boldsymbol{\zeta}(i). \tag{4.15}$$

3) For every iteration i, the convolution matrix (4.4) can be approximated as

$$\hat{\mathbf{H}}(i) = \begin{bmatrix} \hat{h}_1(i) & 0 & \cdots & 0 & 0 \\ \hat{h}_2(i) & \hat{h}_1(i) & \ddots & \vdots & \vdots \\ \vdots & \hat{h}_2(i) & \ddots & 0 & \vdots \\ \hat{h}_M(i) & \vdots & \ddots & \hat{h}_1(i) & 0 \\ 0 & \hat{h}_M(i) & \ddots & \hat{h}_2(i) & \hat{h}_1(i) \\ \vdots & 0 & \ddots & \vdots & \hat{h}_2(i) \\ \vdots & \vdots & \ddots & \hat{h}_M(i) & \vdots \\ 0 & 0 & \cdots & 0 & \hat{h}_M(i) \end{bmatrix}. \tag{4.16}$$

4) For every iteration i, the channel estimator updates only one, $p(i)$th, element in $\hat{\mathbf{h}}(i)$ as

$$\hat{h}_{p(i)}(i) = \hat{h}_{p(i)}(i-1) + \Delta\hat{h}(i).$$

Note that, when using the DCD iteration in channel estimation, for every i, $\Delta\hat{h}(i)$ is a power-of-two number.

5) For every i, only one, $q(i)$th, FFF coefficient in $\hat{\mathbf{f}}(i)$ is updated as

$$\hat{f}_{q(i)}(i) = \hat{f}_{q(i)}(i-1) + \Delta\hat{f}(i),$$

As we will propose to use the DCD iteration in the computation of the FFF taps, for every i, $\Delta\hat{f}(i)$ is also a power-of-two number.

The number of iterations for computing the FFF taps after an update of the channel estimate can be made greater than one. This is a straightforward extension of the algorithm described below. However, our simulation (not presented here) has shown little improvement in the equalizer performance compared to the case of one iteration (as given by assumption 5).

6) We assume that the noise variance σ_ν^2, and thus the signal to noise ratio (SNR) are known.

Table 4.1: Complex-valued RLS algorithm

Step	Equation
	Initialization: $\hat{\mathbf{h}}(0) = \mathbf{0}_{M\times 1}$, $\mathbf{r}(0) = \mathbf{0}_{M\times 1}$, $\mathbf{R}(0) = \varepsilon\mathbf{I}_M$
	for $n = 1, 2, \ldots$
1	$\mathbf{R}(n) = \lambda\mathbf{R}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n)$
2	$\hat{y}(n) = \mathbf{x}^T(n)\hat{\mathbf{h}}(n-1)$
3	$e(n) = y(n) - \hat{y}(n)$
4	$\boldsymbol{\beta}_0(n) = \lambda\mathbf{r}(n-1) + e^*(n)\mathbf{x}(n)$
5	Solve $\mathbf{R}(n)\Delta\mathbf{h}(n) = \boldsymbol{\beta}_0(n) \Rightarrow \Delta\hat{\mathbf{h}}(n)$, $\mathbf{r}(n)$
6	$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \Delta\hat{\mathbf{h}}(n)$

4.4 Partial-update adaptive channel estimation

To satisfy assumption 2 above, we need to use a partial-update channel estimator. For this purpose, we propose to use the RLS-DCD adaptive algorithm due to its low complexity, stability, and fast convergence [17, 52]. Moreover, the channel estimate updates in the RLS-DCD algorithm allow the DFE taps to be computed without explicit multiplications. However, the RLS-DCD algorithm in [17] deals with real-valued signals. Therefore, here we present a complex-valued version of the RLS-DCD algorithm which is a straightforward extension of the RLS-DCD algorithm in [17].

In the RLS algorithm, the vector $\mathbf{h}(n)$ is found by solving the normal equations $\mathbf{R}(n)\mathbf{h}(n) = \boldsymbol{\beta}(n)$, where [4]

$$\begin{aligned}\mathbf{R}(n) &= \sum_{m=0}^{n}\lambda^{n-m}\mathbf{x}(m)\mathbf{x}^H(m) + \lambda^n\varepsilon\mathbf{I}_M,\\ \boldsymbol{\beta}(n) &= \sum_{m=0}^{n}\lambda^{n-m}\mathbf{x}(m)y^*(m),\end{aligned}$$

$\varepsilon > 0$ is a regularization factor and $0 < \lambda \leq 1$ is a forgetting factor. This can be done recursively as presented in Table 4.1. The implementation of this variant of the RLS algorithm is especially efficient when, at step 5, DCD iterations are used for solving the system $\mathbf{R}(n)\Delta\mathbf{h}(n) = \boldsymbol{\beta}_0(n)$. In this case, step 5 can be implemented without explicit multiplications and divisions.

Table 4.2 describes the complex-valued DCD algorithm with a leading element [17], which is used to solve the normal equations at step 5 in Table 4.1. For convenience, we

Table 4.2: Complex-valued DCD algorithm.

Step	Equation	+
	Initialization: $\Delta\hat{\mathbf{h}} = \mathbf{0}_{M\times 1}$, $\mathbf{r} = \boldsymbol{\beta}_0$, $\alpha = A$, $a = 1$, $\mathbf{d} = [\alpha, -\alpha, j\alpha, -j\alpha]$	
	for $k = 1, \ldots, N_u$	
1	$\boldsymbol{\vartheta} = [\Re\{\mathbf{r}^T\}\ \Im\{\mathbf{r}^T\}]$, $p = \arg\max_{m=1,\ldots,2M}\{\lvert\vartheta_m\rvert\}$, if $p > M$, then $p \leftarrow p - M$, go to step 4	$2M-1$
2	$a \leftarrow a + 1$, $\alpha \leftarrow \alpha/2$, $\mathbf{d} = [\alpha, -\alpha, j\alpha, -j\alpha]$	–
3	if $a > M_b$, the algorithm stops	–
4	$\mu = \arg\min\{[-\Re\{r_p\},\ \Re\{r_p\}, -\Im\{r_p\},\ \Im\{r_p\},\ -\alpha R_{p,p}/2]\}$ if $\mu > 4$, then go to step 2	2
5	$\Delta\hat{h}_p = \Delta\hat{h}_p + d_\mu$	1
6	$\mathbf{r} = \mathbf{r} - d_\mu \mathbf{R}^{(p)}$	$2M$
	Total: 0 real mult. and $\leq N_u(4M+2)$ real adds.	

omit here the time index n. It is assumed that the channel taps are represented as M_b-bit fixed-point numbers within an amplitude interval $[-A, A]$, where A is preferably a power-of-two number. The step-size parameter α is given by $\alpha = 2^{-a}A$, where a is a positive integer number, i.e. α is also a power-of-two number (see more details on the parameter choice in [17]). With such settings, operations required in the DCD algorithm are only additions as all multiplications and divisions are replaced by bit-shifts.

4.5 Low-complexity computation of FFF taps

For computation of the FFF taps, equations (4.15) can be transformed into a sequence of auxiliary normal equations $\bar{\mathbf{G}}(i)\Delta\mathbf{f}(i) = \boldsymbol{\zeta}_0(i)$ [17]. A recursive approach for solving these equations can be derived, which is described in Table 4.3. In Table 4.3, $\hat{\mathbf{f}}(i)$ denotes an approximate solution obtained at iteration i; $\boldsymbol{\epsilon}(i)$ is the residual vector $\boldsymbol{\epsilon}(i) = \boldsymbol{\zeta}(i) - \bar{\mathbf{G}}(i)\hat{\mathbf{f}}(i)$; $\Delta\bar{\mathbf{G}}(i) = \bar{\mathbf{G}}(i) - \bar{\mathbf{G}}(i-1)$; and $\Delta\boldsymbol{\zeta}(i) = \boldsymbol{\zeta}(i) - \boldsymbol{\zeta}(i-1)$. In Table 4.3, step 1 requires finding $\Delta\bar{\mathbf{G}}(i)$ which involves computation of the matrix $\bar{\mathbf{G}}(i) = \bar{\mathbf{H}}^H(i)\bar{\mathbf{H}}(i)$ with a complexity of $\mathcal{O}((l+1)K^2)$. Step 2 requires $\mathcal{O}(K^2)$ operations to compute $\Delta\bar{\mathbf{G}}(i)\hat{\mathbf{f}}(i-1)$. These are the most computationally demanding operations in the computation of the FFF taps. For computation of the FBF taps, equation (4.12) can be computed directly with a complexity of $\mathcal{O}(BK)$, which is also computa-

Table 4.3: Recursively solving a sequence of equations for computation of FFF taps

Step	Equation
	Initialization: $\boldsymbol{\epsilon}(0) = \mathbf{0}_{K\times 1}$, $\boldsymbol{\zeta}(0) = \mathbf{0}_{K\times 1}$, $\hat{\mathbf{f}}(0) = \mathbf{0}_{K\times 1}$
	for $i = 1, 2, \ldots$
1	Find $\Delta\bar{\mathbf{G}}(i)$ and $\Delta\boldsymbol{\zeta}(i)$
2	$\boldsymbol{\zeta}_0(i) = \boldsymbol{\epsilon}(i-1) + \Delta\boldsymbol{\zeta}(i) - \Delta\bar{\mathbf{G}}(i)\hat{\mathbf{f}}(i-1)$
3	Solve $\bar{\mathbf{G}}(i)\Delta\mathbf{f} = \boldsymbol{\zeta}_0(i) \Rightarrow \Delta\hat{\mathbf{f}}(i),\ \boldsymbol{\epsilon}(i)$
4	$\hat{\mathbf{f}}(i) = \hat{\mathbf{f}}(i-1) + \Delta\hat{\mathbf{f}}(i)$

tionally consuming. In the followings of this chapter, we show how these operations can be simplified when using our assumptions.

4.5.1 Computation of $\Delta\bar{\mathbf{G}}(i)\hat{\mathbf{f}}(i-1)$

Let $\bar{\mathbf{H}}(i) = \bar{\mathbf{H}}(i-1) + \bar{\boldsymbol{\Delta}}(i)$, then we have $\bar{\mathbf{G}}(i) = \bar{\mathbf{G}}(i-1) + \bar{\boldsymbol{\Delta}}^H(i)\bar{\mathbf{H}}(i-1) + \bar{\mathbf{H}}^H(i-1)\bar{\boldsymbol{\Delta}}(i) + \bar{\boldsymbol{\Delta}}^H(i)\bar{\boldsymbol{\Delta}}(i)$ and thus,

$$\begin{aligned}\Delta\bar{\mathbf{G}}(i)\hat{\mathbf{f}}(i-1) &= \bar{\boldsymbol{\Delta}}^H(i)\bar{\mathbf{H}}(i-1)\hat{\mathbf{f}}(i-1) \\ &\quad + \bar{\mathbf{H}}^H(i-1)\bar{\boldsymbol{\Delta}}(i)\hat{\mathbf{f}}(i-1) + \bar{\boldsymbol{\Delta}}^H(i)\bar{\boldsymbol{\Delta}}(i)\hat{\mathbf{f}}(i-1). \end{aligned} \tag{4.17}$$

For convenience, we rewrite (4.17) as

$$\mathbf{z}(i) = \mathbf{c}(i) + \mathbf{b}(i) + \mathbf{D}(i)\hat{\mathbf{f}}(i-1), \tag{4.18}$$

where

$$\begin{aligned}\mathbf{z}(i) &= \Delta\bar{\mathbf{G}}(i)\hat{\mathbf{f}}(i-1), \\ \mathbf{c}(i) &= \bar{\boldsymbol{\Delta}}^H(i)\bar{\mathbf{H}}(i-1)\hat{\mathbf{f}}(i-1), \\ \mathbf{b}(i) &= \bar{\mathbf{H}}^H(i-1)\bar{\boldsymbol{\Delta}}(i)\hat{\mathbf{f}}(i-1), \\ \mathbf{D}(i) &= \bar{\boldsymbol{\Delta}}^H(i)\bar{\boldsymbol{\Delta}}(i).\end{aligned}$$

Now, we derive expressions for computing these variables: $\mathbf{c}(i)$, $\mathbf{b}(i)$ and $\mathbf{D}(i)$. Note that $\bar{\boldsymbol{\Delta}}(i)$ is a $(l+1)\times K$ submatrix of a $(K+M-1)\times K$ Toeplitz matrix whose first column is $\Delta\hat{h}(i)\mathbf{e}_{p(i)}$ and elements of the first row are zeros if $p(i) > 1$. For $p(i) = 1$, $\bar{\boldsymbol{\Delta}}(i)$ is a rectangular diagonal matrix whose diagonal elements are $\Delta\hat{h}(i)$. The structure of the matrix $\bar{\boldsymbol{\Delta}}(i)$ is illustrated in Fig.4.5.

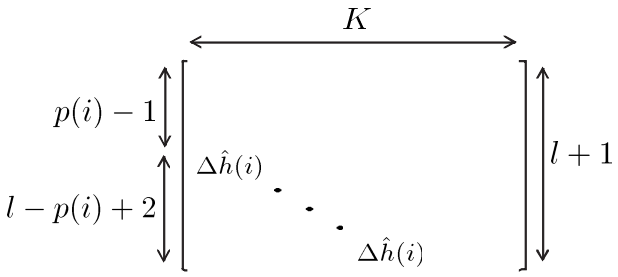


Figure 4.5: Structure of the matrix $\bar{\mathbf{\Delta}}(i)$

The matrix $\bar{\mathbf{\Delta}}^H(i)\bar{\mathbf{H}}(i-1)$ can then be expressed as

$$\bar{\mathbf{\Delta}}^H(i)\bar{\mathbf{H}}(i-1) = \Delta\hat{h}^*(i)\begin{bmatrix}\mathbf{F}(i-1)\\ \hline \mathbf{0}_{(K-l+p(i)-2)\times K}\end{bmatrix}, \tag{4.19}$$

where $\mathbf{F}(i-1)$ is a $(l-p(i)+2)\times K$ submatrix of $\bar{\mathbf{H}}(i-1)$ as illustrated in Fig.4.6(a). The structure of matrix $\mathbf{F}(i-1)$ is shown in Fig.4.6(b). For convenience, we now define a $(l-p(i)+2)\times(l+1)$ matrix $\bar{\mathbf{F}}(i-1)$, which is a submatrix of $\mathbf{F}(i-1)$ as illustrated in Fig.4.6(c).

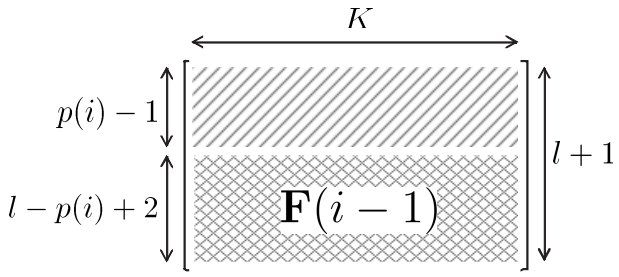
According to the structure of $\mathbf{F}(i-1)$ and by using (4.19), $\mathbf{c}(i)$ can be expressed as

$$\mathbf{c}(i) = \Delta\hat{h}^*(i)\begin{bmatrix}\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)\\ \hline \mathbf{0}_{(K-l+p(i)-2)\times 1}\end{bmatrix}, \tag{4.20}$$

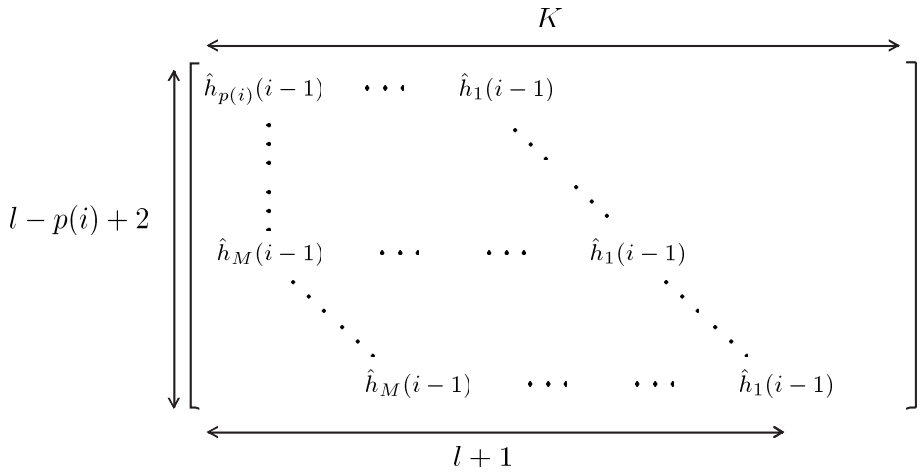
where $\hat{\mathbf{f}}_{1:l+1}(i-1)$ denotes the first $(l+1)$ elements of the vector $\hat{\mathbf{f}}(i-1)$. $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$ can be computed directly with $(l-p(i)+2)(l+1)/2$ complex multiplications. However, when a large FFF length ($K \gg M$, and thus $l > M$) is used, this operation will be computationally consuming. Alternatively, this can be implemented using the fast fourier transform (FFT) [85] as explained in Appendix A, with $2(l+1)+6(l+1)\log_2 2(l+1)$ complex multiplications. In Appendix B, we also derive a recursion for computing $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$, which only requires $2(K+M)$ real multiplications and $2(K+M)$ real additions for each i. Moreover, as we propose to use the DCD iteration in both channel estimation and the computation of the FFF taps, all the multiplications required in this recursive approach can be replaced by bit-shift operations.

From (4.19), we obtain

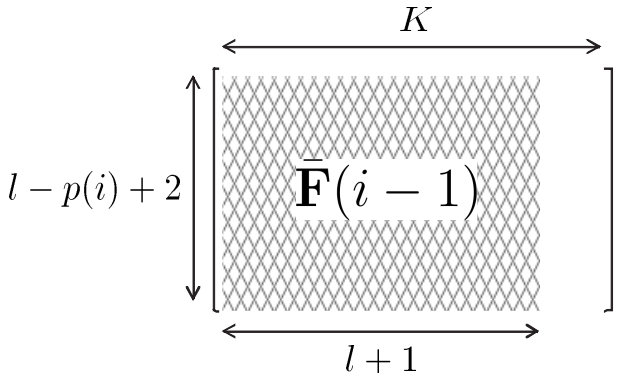
$$\bar{\mathbf{H}}^H(i-1)\bar{\mathbf{\Delta}}(i) = \Delta\hat{h}(i)\left[\mathbf{F}^H(i-1) \mid \mathbf{0}_{K\times(K-l+p(i)-2)}\right].$$



(a) Structure of the matrix $\bar{\mathbf{H}}(i-1)$



(b) Structure of the matrix $\mathbf{F}(i-1)$



(c) $\bar{\mathbf{F}}(i-1)$ as a submatrix of $\mathbf{F}(i-1)$

Figure 4.6: Matrices structures

According to the structure of $\mathbf{F}(i-1)$ (see Fig.4.6), $\mathbf{b}(i)$ can be expressed as

$$\mathbf{b}(i) = \Delta\hat{h}(i) \begin{bmatrix} \bar{\mathbf{F}}^H(i-1)\hat{\mathbf{f}}_{1:l-p(i)+2}(i-1) \\ \hline \mathbf{0}_{(K-l-1)\times 1} \end{bmatrix}, \tag{4.21}$$

where the vector $\bar{\mathbf{F}}^H(i-1)\hat{\mathbf{f}}_{1:l-p(i)+2}(i-1)$ can be computed directly with $(l-p(i)+2)(l+1)/2$ complex multiplications. Alternatively, it can also be computed using an approach based on the FFT similar to that as given in Appendix A, with $2(l+1)+6(l+1)\log_2 2(l+1)$ complex multiplications. However, a recursive computation is not available for this vector because the number of elements of $\hat{\mathbf{f}}_{1:l-p(i)+2}(i-1)$ involved in the computation varies according to the position $p(i)$.

Since $\mathbf{D}(i) = \bar{\mathbf{\Delta}}^H(i)\bar{\mathbf{\Delta}}(i)$, where $\bar{\mathbf{\Delta}}(i)$ has the structure as given in Fig.4.5, elements of $\mathbf{D}(i)$ are given by

$$D_{m,n} = \begin{cases} |\Delta\hat{h}(i)|^2, & m = n = 1, \ldots, (l-p(i)+2), \\ 0, & \text{otherwise}. \end{cases}$$

Thus, we have

$$\mathbf{D}(i)\hat{\mathbf{f}}(i-1) = |\Delta\hat{h}(i)|^2\hat{\mathbf{f}}_{1:l-p(i)+2}(i-1). \tag{4.22}$$

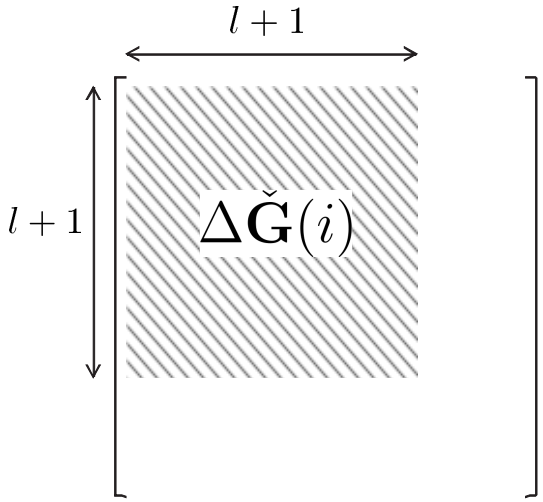
From (4.20), (4.21) and (4.22), $\mathbf{z}(i)$ in (4.18) can finally be computed by

$$\begin{aligned} \mathbf{z}(i) = \Delta\hat{h}^*(i) & \begin{bmatrix} \bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1) \\ \hline \mathbf{0}_{(K-l+p(i)-2)\times 1} \end{bmatrix} \\ & + \Delta\hat{h}(i) \begin{bmatrix} \bar{\mathbf{F}}^H(i-1)\hat{\mathbf{f}}_{1:l-p(i)+2}(i-1) \\ \hline \mathbf{0}_{(K-l-1)\times 1} \end{bmatrix} \\ & + |\Delta\hat{h}(i)|^2\hat{\mathbf{f}}_{1:l-p(i)+2}(i-1). \end{aligned} \tag{4.23}$$

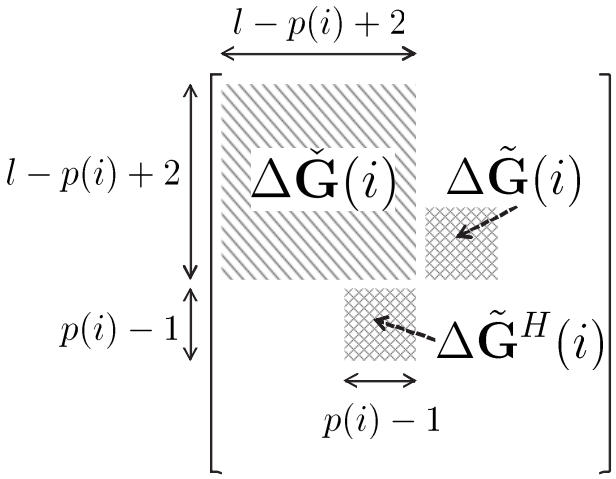
4.5.2 Computation of $\Delta\bar{\mathbf{G}}(i)$

The matrix $\Delta\bar{\mathbf{G}}(i)$ will have three different structures as illustrated in Fig.4.7, depending on the position of update $p(i)$ from the channel estimator.

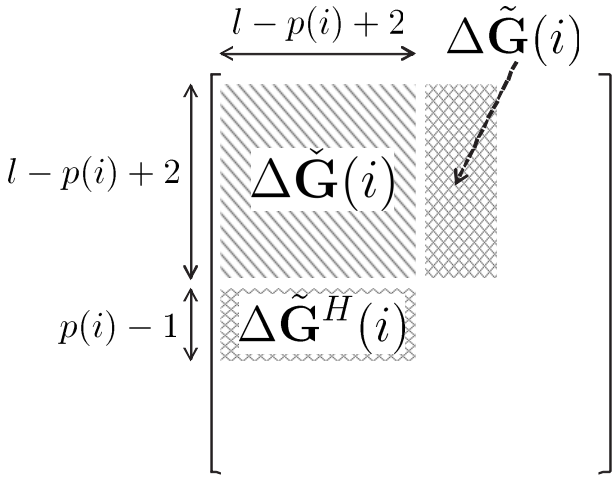
1) For $p(i) = 1$, $\Delta\bar{\mathbf{G}}(i)$ contains an upper-left corner $(l+1)\times(l+1)$ matrix $\Delta\check{\mathbf{G}}(i)$



(a) Case 1: $p(i) = 1$



(b) Case 2: $1 < p(i) \leq (l+3)/2$



(c) Case 3: $p(i) > (l+3)/2$

Figure 4.7: Structure of matrix $\Delta\bar{\mathbf{G}}(i)$

which is a Hermitian Toeplitz matrix, and the other elements of $\Delta\bar{\mathbf{G}}(i)$ are zeros. In such a case, only the first column of matrix $\Delta\check{\mathbf{G}}(i)$ needs to be computed.

2) For $1 < p(i) \leq (l+3)/2$, $\Delta\bar{\mathbf{G}}(i)$ contains three Toeplitz submatrices: one is an upper-left corner $(l-p(i)+2)\times(l-p(i)+2)$ matrix $\Delta\check{\mathbf{G}}(i)$ which is a Hermitian Toeplitz matrix, and the other two are Toeplitz square matrices $\Delta\tilde{\mathbf{G}}(i)$ and $\Delta\tilde{\mathbf{G}}^H(i)$, which are conjugate transpose of each other. Furthermore, $\Delta\tilde{\mathbf{G}}(i)$ is a $(p(i)-1)\times(p(i)-1)$ lower triangle matrix. The other elements of $\Delta\bar{\mathbf{G}}(i)$ are zeros. In this case, only the first column of matrix $\Delta\check{\mathbf{G}}(i)$ and the first column of matrix $\Delta\tilde{\mathbf{G}}(i)$ need to be computed.

3) For $p(i) > (l+3)/2$, $\Delta\bar{\mathbf{G}}(i)$ also contains three Toeplitz submatrices: one is an upper-left corner $(l-p(i)+2)\times(l-p(i)+2)$ matrix $\Delta\check{\mathbf{G}}(i)$ which is a Hermitian Toeplitz matrix, and the other two are Toeplitz matrices $\Delta\tilde{\mathbf{G}}(i)$ and $\Delta\tilde{\mathbf{G}}^H(i)$, which are conjugate transpose of each other. The size of matrix $\Delta\tilde{\mathbf{G}}(i)$ is $(l-p(i)+2)\times(p(i)-1)$. The other $\bar{\mathbf{G}}(i)$ are zeros. In this case, the first column of matrix $\Delta\check{\mathbf{G}}(i)$, and the first column and row of matrix $\Delta\tilde{\mathbf{G}}(i)$ need to be computed.

According to these matrix structures, $\Delta\bar{\mathbf{G}}(i)$ can be obtained as described in Table 4.4. For each i, the complexity of computing $\Delta\bar{\mathbf{G}}(i)$ is a function of the position $p(i)$, where $1 \leq p(i) \leq M$. Therefore, we can only evaluate the maximum complexity. Computation of $\Delta\bar{\mathbf{G}}(i)$ requires no more than $M+2l/3+3$ real multiplications and $l+3$ real additions. Moreover, since $\Delta\hat{h}(i)$ is a power-of-two number, all the multiplications required in the computation of $\Delta\bar{\mathbf{G}}(i)$ can be done by bit-shift operations.

Table 4.5 summarizes the proposed technique for computing the FFF taps. Here, we assume that the noise variance σ_ν^2, and thus the signal to noise ratio (SNR) are known. Table 4.5 also shows the complexity of the computation steps in terms of multiplications and additions. The complexity of computing the FFF taps will depend on the iterative technique used for solving the equation $\bar{\mathbf{G}}\Delta\mathbf{f} = \boldsymbol{\zeta}_0$ at step 8 (P_{mu} multiplications and P_{ad} additions). Since we propose to use adaptive algorithms with partial update for solving the equation at step 8, for every i, only the $q(i)$th column of $\bar{\mathbf{G}}(i)$ needs to be updated, $\bar{\mathbf{G}}^{(q(i))}(i) = \bar{\mathbf{G}}^{(q(i))}(i-1) + \Delta\bar{\mathbf{G}}^{(q(i))}(i)$. Therefore, at step 7, we can only take into account the operations for updating the $q(i)$th column of $\bar{\mathbf{G}}(i)$, which will only require $3l/2+2$ real additions. Note that the total number of real additions provided at the bottom

Table 4.4: Computation of $\Delta\bar{\mathbf{G}}(i)$

Step	Equation
	Initialization: $p = p(i)$, $\Delta\hat{h} = \Delta\hat{h}(i)$, $\Delta\bar{\mathbf{G}}(i) = \mathbf{0}_{K\times K}$ $\boldsymbol{\rho} = \mathbf{0}_{(l-p+2)\times 1}$, $v = \min\{l+1, M\}$
1	$\rho_1 = \Delta\hat{h}[\hat{h}^*_p(i-1) + \Delta\hat{h}^*]$ $\rho_m = \Delta\hat{h}\hat{h}^*_{p+m-1}(i-1), m = 2, \ldots, v-p+1,$
2	$\rho_m = \rho_m + \Delta\hat{h}\hat{h}^*_{p-m+1}(i-1),$ $\begin{cases} m = 1, \ldots, l-p+2 & \text{for } p > (l+2)/2 \\ m = 1, \ldots, p & \text{otherwise} \end{cases}$
3	$\Delta\check{\mathbf{G}}^{(1)}(i) = \boldsymbol{\rho}$
4	if $1 < p \le (l+3)/2$ $\Delta\tilde{\mathbf{G}}^{(1)}(i) = \Delta\hat{h}^*\hat{\mathbf{h}}_{1:p-1}(i-1)$ if $p(i) > (l+3)/2$ $\Delta\tilde{\mathbf{G}}^{(1)}(i) = \Delta\hat{h}^*\hat{\mathbf{h}}_{2p-l-2:p-1}(i-1)$ $\Delta\tilde{\mathbf{G}}_{(1)}(i) = \Delta\hat{h}^*\hat{\mathbf{u}}_{2:p-1}(i-1)$ where $\hat{u}_m(i-1) = \hat{h}_{M-m+1}(i-1), m = 1, \ldots, M$

of the table is the upper bound of the complexity. This is due to the fact that the actual complexity of the computation in step 7 will depend on the position $p(i)$. However, since the complexity of computing $\mathbf{g}(i)$ in step 3 determines the total complexity, this upper bound will be close to the actual complexity of the proposed algorithm.

4.5.3 DCD iterations

For solving the auxiliary normal equations at step 8 in Table 4.5, we propose to use the DCD iteration with one update as described in Table 4.6. As we have mentioned in Section 4.4, in the DCD iteration, it is assumed that the FFF taps are represented as M_b-bit fixed-point numbers within an interval $[-A, A]$, where A is preferably a power-of-two number. The step-size parameter α is $\alpha = 2^{-a}A$, i.e. also a power-of-two number. Therefore, operations required in the DCD algorithm are only additions as all multiplications and divisions are replaced by bit-shifts. When using the DCD iteration in both channel estimation and computation of the FFF taps, the increments $\Delta\hat{h}(i)$ and $\Delta\hat{f}(i)$ are power-of-two numbers. This means that multiplications which involve either $\Delta\hat{h}(i)$ or $\Delta\hat{f}(i)$ can be done by bit-shift operations. Therefore, our proposed approach as presented in Table 4.5 does not require any multiplication except those involved in the computation of

Table 4.5: Low-complexity computation of FFF taps

Step	Equation	$\times$	+
	Initialization: $i=0$, $\bar{\mathbf{G}}(0)=\sigma_v^2\mathbf{I}_K$, $\hat{\mathbf{f}}(0)=\mathbf{0}_{K\times 1}$, $\boldsymbol{\epsilon}(0)=\mathbf{0}_{K\times 1}$,		
	for $n=1,2,\ldots$		
	for $k=1,\ldots,N_u$		
1	$i=i+1$		
2	Use one iteration in the channel estimator and obtain $\hat{\mathbf{h}}(i)$, $\Delta\hat{h}(i)$ and position $p(i)$		
3	Compute $\mathbf{c}(i)$ using (4.20) and $\mathbf{b}(i)$ using (4.21)	ψ_{mu}	ψ_{ad}
4	$\mathbf{z}(i)=\mathbf{c}(i)+\mathbf{b}(i)+\lvert\Delta\hat{h}(i)\rvert^2\hat{\mathbf{f}}(i-1)$	$-$	$4K$
5	$\boldsymbol{\zeta}_0=\boldsymbol{\epsilon}(i-1)+\Delta\hat{h}^*(i)\mathbf{e}_{l-p(i)+2}-\mathbf{z}(i)$	$-$	$2K+2$
6	$\mathbf{v}=[\Re\{\boldsymbol{\epsilon}^T\}\ \Im\{\boldsymbol{\epsilon}^T\}]$, $q(i)=\arg\max_{m=1,\ldots,2K}\{\lvert v_m\rvert\}$ if $q(i)>K$, then $q(i)\leftarrow q(i)-K$	$-$	$2K$
7	Compute $\Delta\bar{\mathbf{G}}(i)$ using Table 4.4 and update $\bar{\mathbf{G}}(i)=\bar{\mathbf{G}}(i-1)+\Delta\bar{\mathbf{G}}(i)$	$-$	$\leq(3l+4)(l+1)/2$
8	Use one iteration to solve $\bar{\mathbf{G}}\Delta\mathbf{f}=\boldsymbol{\zeta}_0$ and obtain $\Delta\hat{\mathbf{f}}(i)$ and $\boldsymbol{\epsilon}(i)$	P_{mu}	P_{ad}
9	$\hat{f}_{q(i)}(i)=\hat{f}_{q(i)}(i-1)+\Delta\hat{f}(i)$	$-$	1
	Total for each sample n: $N_u[\psi_{mu}+P_{mu}]$ real mult. and $\leq N_u[\psi_{ad}+3l^2/2+8K+7l/2+P_{ad}+4]$ real adds, where $\psi_{mu}=8l+8+24(l+1)\log_2 2(l+1)$ and $\psi_{ad}=2K+2M+12(l+1)\log_2 2(l+1)$		

$\mathbf{b}(i)$ in step 3.

4.6 Low-complexity implementation of FBF

4.6.1 Recursive computation

For the conventional DFE whose structure is shown in Fig.4.1, the FBF taps, in addition to the FFF taps, need to be computed at every iteration i. Computation of the FBF taps can be done directly using (4.12) with a complexity of $\mathcal{O}(BK)$. Alternatively, as the matrix $\tilde{\mathbf{H}}(i)$ is a block of the channel convolution matrix (see Fig.4.4), the FBF taps can be obtained by convolving the FFF taps $\hat{\mathbf{f}}(i)$ and the vector $\hat{\mathbf{u}}(i)$ whose elements are given by (A.1) in Appendix A. This can be computed using the FFT [85] similar to the approach explained in Appendix A, with $2K+6K\log_2 2K$ complex multiplications (for $K>M$). In Appendix B, we derive a recursion for computing the vector $\boldsymbol{\varphi}(i)$ as a result of convolving the vector $\hat{\mathbf{f}}(i)$ and $\hat{\mathbf{u}}(i)$:

$$\begin{aligned}\boldsymbol{\varphi}(i)=\boldsymbol{\varphi}(i-1)&+\Delta\hat{f}(i)\hat{\mathbf{h}}^{[q(i)]}(i-1)\\&+\Delta\hat{h}(i)\hat{\mathbf{f}}^{[p(i)]}(i),\end{aligned}$$

Table 4.6: DCD algorithm with one update

Step	Equation	+
	Initialization: $q = q(i)$, $\Delta\hat{\mathbf{f}} = \mathbf{0}_{K\times 1}$, $\boldsymbol{\epsilon} = \boldsymbol{\zeta}_0$, $\alpha = A/2$, $a = 1$, $\mathbf{d} = [\alpha, -\alpha, \alpha j, -\alpha j]$	
1	go to step 4	–
2	$a \leftarrow a + 1$, $\alpha \leftarrow \alpha/2$, $\mathbf{d} = [\alpha, -\alpha, \alpha j, -\alpha j]$	–
3	if $a > M_b$, the algorithm stops	–
4	$\mu = \arg\min\{[-\Re\{\epsilon_q\}, \Re\{\epsilon_q\}, -\Im\{\epsilon_q\}, \Im\{\epsilon_q\}, \alpha G_{q,q}/2]\}$ if $\mu > 4$, then go to step 2	5
5	$\Delta\hat{f}_q = \Delta\hat{f}_q + d_\mu$	1
6	$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - \alpha\bar{\mathbf{G}}^{(q)}$	$2K$
	$\Delta\hat{\mathbf{f}}(i) = \Delta\hat{\mathbf{f}}$, $\boldsymbol{\epsilon}(i) = \boldsymbol{\epsilon}$	
	Total: $P_{mu} = 0$ and $P_{ad} \leq 2K + 6 + M_b$	

The FBF taps $\hat{\mathbf{g}}(i)$ can then be obtained from $\boldsymbol{\varphi}(i)$ as

$$\hat{\mathbf{g}}(i) = \boldsymbol{\varphi}_{l+2:l+B}(i). \tag{4.24}$$

By using such a recursion, the computation of the FBF taps only requires $2(K + M)$ real multiplications and $2(K + M)$ real additions. Moreover, when using the recursive approach given in Appendix B for the computation of the FFF taps, which also involves the computation of $\boldsymbol{\varphi}(i)$ at every iteration i, the FBF taps can be obtained directly from (4.24) without any extra computation.

4.6.2 Modified DFE

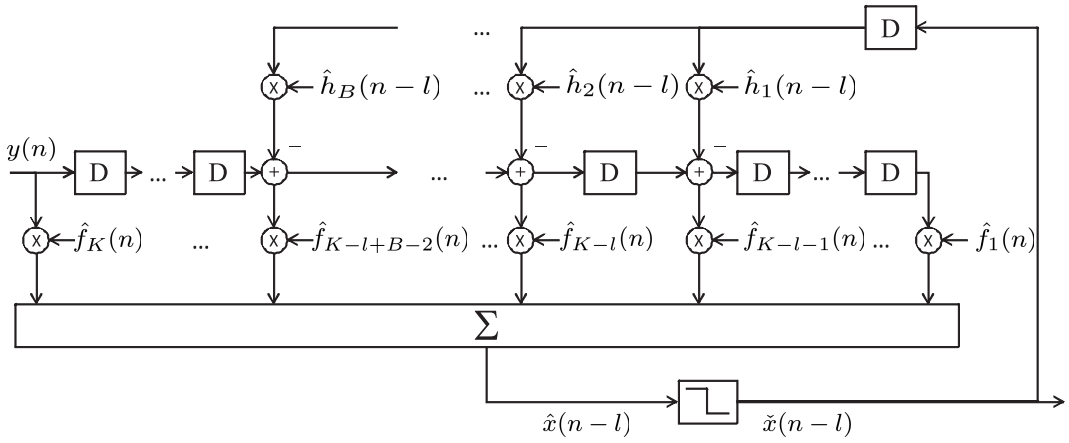


Figure 4.8: Modified structure of symbol-spaced DFE

Table 4.7: DFEs used for simulation

Algorithm	Channel	Decision	Structure	Convolution matrix $\mathbf{H}(n)$	Computation of FFF taps	Computation of FBF taps
MMSE (known decision)	known	known	Fig.4.1	formed using (4.4)	solving (4.14) directly	using (4.12)
RLS CE (known decision)	estimated using RLS	known	Fig.4.1	formed using (4.4)	solving (4.14) directly	using (4.12)
RLS CE	estimated using RLS	estimated	Fig.4.1	formed using (4.4)	solving (4.14) directly	using (4.12)
RLS DA	not required	estimated	Fig.4.1	not required	estimated using RLS	estimated using RLS
Proposed (known decision)	estimated using RLS-DCD	known	Fig.4.8	formed using (4.16)	using Table 4.5	not required
Proposed	estimated using RLS-DCD	estimated	Fig.4.8	formed using (4.16)	using Table 4.5	not required

DFEs can also be implemented using the modified structure as shown in Fig.4.8 [24, 25], which does not require any computation for the FBF taps. In this modified DFE, for every time sample n, the FBF simply multiplies the detected data symbol $\check{x}(n-l)$ with the estimated channel impulse response $\hat{\mathbf{h}}(n-l)$. Outputs from the FBF are fed into the FFF directly in order to precancel postcursors according to the equalizer delay l. Note that the modified DFE structure shown in Fig.4.8 applies for the case when $K \geq M$. For the case when $K < M$, the equalizer structure is slightly different and can be found in [25]. In this modified DFE, the FBF taps are the coefficients of the estimated impulse response, which can be obtained directly from the channel estimator without any extra computation.

4.7 Simulation results

In this section, we compare the performance of six DFEs as summarized in Table 4.7:

1) MMSE DFE (known decision). For every time sample n, the convolution matrix $\mathbf{H}(n)$ is formed using (4.4), in which the channel impulse response $\mathbf{h}(n)$ is perfectly known. The FFF vector $\mathbf{f}(n)$ is found by solving (4.14), and the FBF vector $\mathbf{g}(n)$ is obtained using (4.12). Correct decisions are perfectly known and used in the FBF.

2) RLS CE based adaptive DFE (known decision). The time-varying channel is estimated using the classical RLS algorithm [4] with a forgetting factor λ, and for every

sample n, the convolution matrix $\mathbf{H}(n)$ is given by (4.4). The FFF vector $\mathbf{f}(n)$ is found by solving (4.14), and the FBF vector $\mathbf{g}(n)$ is obtained using (4.12). Correct decisions are perfectly known and used in the FBF.

3) RLS CE based adaptive DFE. This is similar to the DFE in 2) except that the correct decisions are unknown; decisions obtained from the equalizer are used in the FBF.

4) RLS directly adaptive (DA) DFE. In the DA DFE, $\mathbf{\Gamma}(n)$ and $\boldsymbol{\zeta}(n)$ are estimated without explicit channel estimation using the received data and known pilot or estimated data symbols. The equalizer taps are then obtained by solving (4.6) using the RLS algorithm.

5) Proposed DFE (known decision). The time-varying channel is estimated using the RLS-DCD algorithm presented in Section 4.4. For every i, the FFF vector is obtained using the algorithm in Table 4.5. Correct decisions are perfectly known and used in the FBF.

6) Proposed DFE. This is similar to the DFE in 5) except that the correct decisions are unknown; decisions obtained from the equalizer are used in the FBF.

We adopt the first order autoregressive model given by $\mathbf{h}(n) = \sqrt{v}\ \mathbf{h}(n-1) + \sqrt{1-v}\ \boldsymbol{\omega}(n)$ [54], to simulate the time-varying channel impulse response $\mathbf{h}(n)$, where $\sqrt{v}$ is the autoregressive factor and $\boldsymbol{\omega}(n)$ are zero-mean independent random Gaussian vectors, whose elements have variance $1/M$. In our simulation, we consider two different channels: one is a short channel ($M = 21$), for which the initial impulse response is generated as a complex zero-mean independent random Gaussian vector, whose elements have variance $1/M$; the other is a long channel ($M = 101$), for which the initial impulse response is generated as a vector with 11 non-zero elements, whose positions are uniformly distributed between 1 and 101. These non-zero elements are complex zero-mean independent random Gaussian numbers with variance $1/11$. Different signal to noise ratios (SNRs) are considered, and for each SNR, simulation results are obtained by averaging over 200 independent simulation trials. For each trial, a 4000-length data sequence of unit power is transmitted, which contains a sequence of L pilot symbols followed by $4000 - L$ unknown data symbols. In the CE based DFEs, channel estimates are

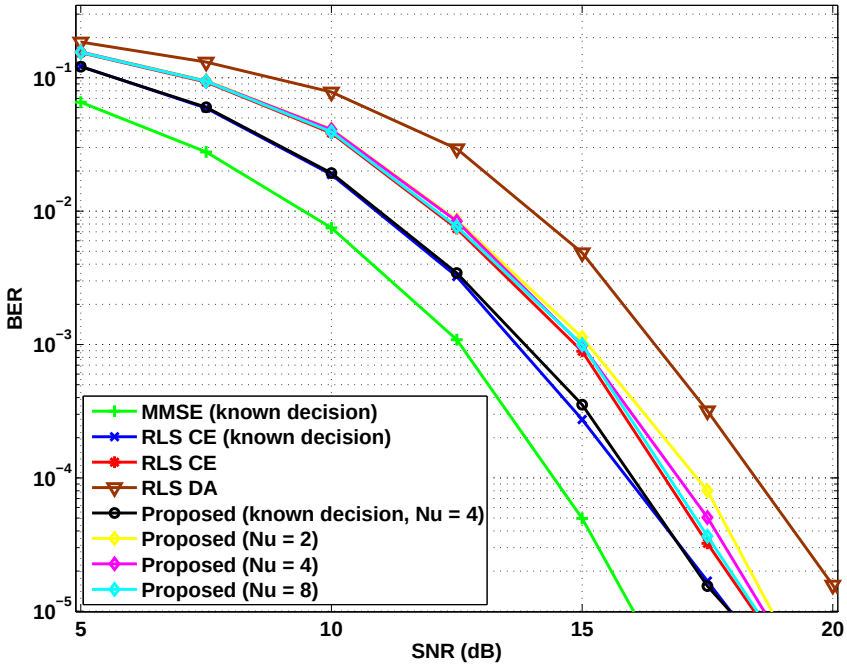


Figure 4.9: BER performance of the DFEs over short time-varying channels: $\upsilon = 1 - 10^{-4}$, $M = 21$, $K = 41$, $B = 21$, $M_b = 16$, $A = 1$, The number of pilot symbols for the RLS DA DFE is $L = 600$; for the other DFEs: $L = 100$.

first obtained from the pilot symbols and then from the equalized data symbols. In all the simulation scenarios, QPSK symbols are transmitted.

Fig.4.9 and Fig.4.10 compare the performance of the DFEs over short and long time-varying channels, respectively. For each SNR, the RLS forgetting factor is chosen in the interval $0.988 \leq \lambda \leq 0.997$, so that the minimum BER is achieved. It is seen that, with $N_u = 4$ and even $N_u = 2$, the proposed DFE performs very close to the RLS CE based adaptive DFE and outperforms the RLS DA DFE.

Fig.4.11 compares the complexity in computation of DFE taps using the proposed approach and the approach in [28]. The comparison is made in terms of the number of real multiplications required per sample, for different FFF length K. It is seen that, as K increases, the computational complexity of the proposed approach grows linearly and becomes much lower than that of the approach in [28].

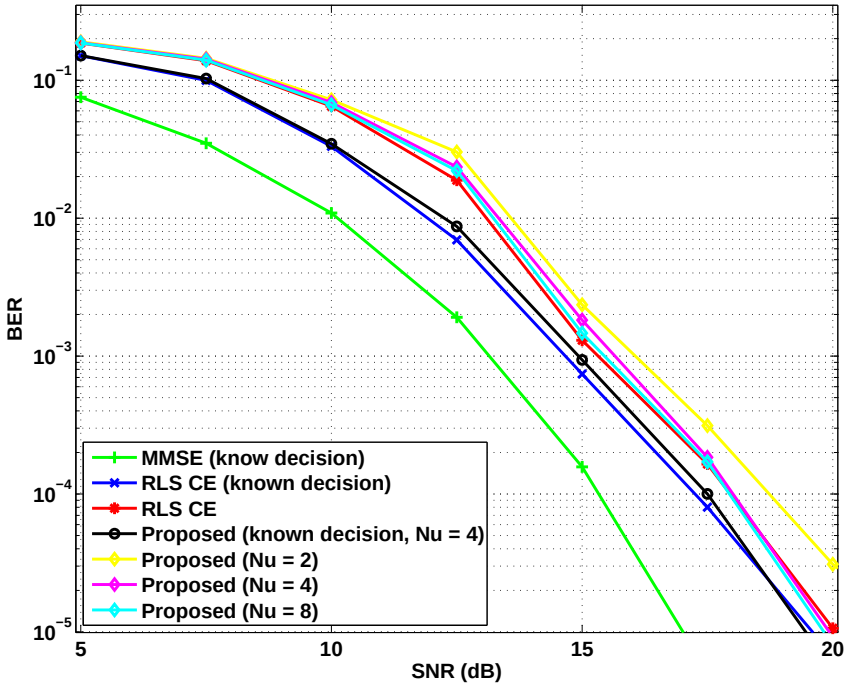


Figure 4.10: BER performance of the DFEs over long time-varying channels: $\upsilon = 1 - 10^{-4}$, $M = 101$, $K = 201$, $B = 101$, $M_b = 16$, $A = 1$, $L = 300$.

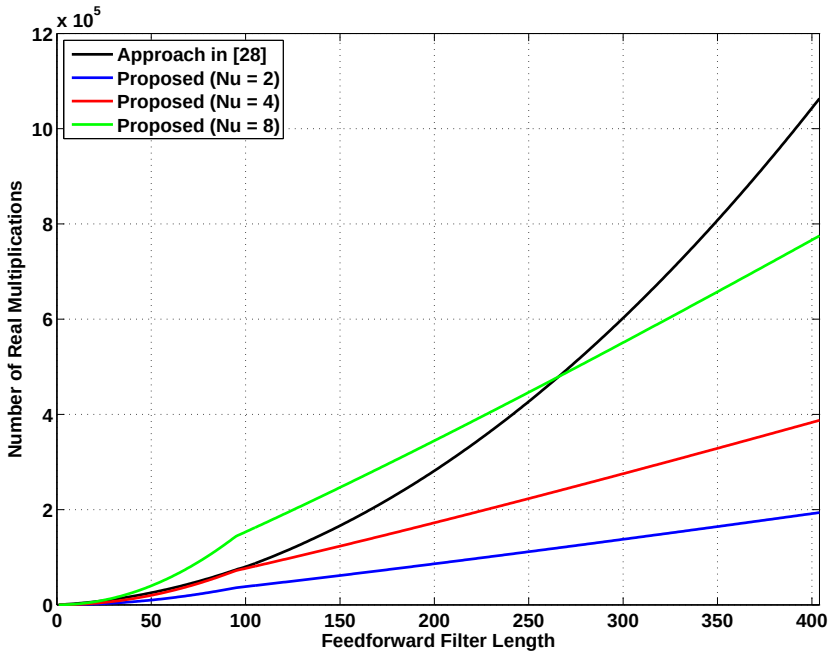


Figure 4.11: Number of real multiplications required for computation of DFE taps per sample for different FFF length K: $M = 101$, $B = 101$, $l = K - 1$.

4.8 Conclusions

In this chapter, we have proposed a CE based adaptive DFE with a complexity as low as $\mathcal{O}(N_u(l+1)\log_2 2(l+1))$ real multiplications per sample, where l is the equalizer delay and N_u is the number of iterations such that $N_u \ll l$. We have also presented a complex-valued RLS adaptive filtering algorithm that is preferable for the channel estimator used together with the proposed DFE, as well as a complex-valued DCD algorithm used for both the channel estimation and computation of the DFE taps. In the proposed DFE, the DCD iteration is used in both channel estimation and computation of the equalizer taps. In such a case, most of the multiplications involved in the computation of equalizer taps can be replaced by bit-shift operations, which makes the equalizer attractive for hardware design. Simulation shows that, even with a small number of N_u, the proposed DFE significantly outperforms the DA DFE and performs very close to the known CE based DFEs. However, the proposed DFE also involves $\mathcal{O}(N_u(l^2 + l\ \log_2 2l))$ real additions per sample, which is still computationally consuming in practice, especially for channels with large delay spreads, such as the underwater acoustic channels. In the next chapter, we will derive an even lower complexity method for recursive computation of CE based DFE taps.

Chapter 5

Partial-Update Channel-Estimate Based Adaptive Decision Feedback Equalizer: Approach 2

Contents

In the previous chapter, we have presented a partial-update channel-estimate (CE) based adaptive decision-feedback equalizer (DFE), which has a complexity as low as $\mathcal{O}(N_u l \log_2 2l)$ real multiplications per sample, where l is the equalizer delay and N_u is the number of updates per sample such that $N_u \ll l$. However, it still requires $\mathcal{O}(N_u(l^2 + l \log_2 2l))$ real additions per sample. In this chapter, we propose an even lower complexity method for recursive computation of CE based DFE taps. The proposed DFE exploits DCD iterations and is especially efficient when the recursive-least-squares DCD (RLS-DCD) algorithm is used for channel estimation. Simulation results show that the proposed DFE performs close to the CE based DFE, where the CE is obtained using the classical RLS adaptive filter and the equalizer taps are computed according to the mini-

mum mean-square error criterion.

This chapter is organized as follows. In the next section, an introduction is given. Section 5.2 introduces assumptions made for deriving the partial-update DFE. In Section 5.3, the partial update DFE is derived. Section 5.4 presents numerical results that demonstrate the performance and computational complexity of the proposed DFE against known techniques. Finally, Section 5.5 draws conclusions.

5.1 Introduction

In this chapter, we propose another novel CE based adaptive DFE which can operate together with partial-update channel estimators (see [75–77] and reference therein). At every sample, only a few CE taps may be updated, and every such an update involves only one tap, e.g. the CE can be generated by a low complexity partial update adaptive filter. Every update of the equalizer involves one tap in the feedforward filter (FFF) and one tap in the feedback filter (FBF). The proposed DFE exploits complex-valued DCD iterations and has a complexity as low as $\mathcal{O}(N_u K) + \mathcal{O}(N_u B) + \mathcal{O}(N_u M)$ operations per sample, where K is the FFF length, B the FBF length, M the channel estimator length, and N_u the number of updates such that $N_u \ll M$. The proposed DFE is especially efficient when the channel estimator also exploits the complex-valued DCD iterations, e.g. such as in the the complex-valued RLS-DCD adaptive filter that we have introduced in Section 4.4. Then all multiplications involved in the computation of the equalizer taps can be replaced by bit-shift operations. This makes the equalizer attractive for hardware design.

5.2 Assumptions

For computation of the DFE taps, we use the following assumptions, which are essentially the same as we introduced in 4.3. The difference is in assumption 5).

1) In practice, as the time-varying channel is unknown, estimates $\hat{\mathbf{h}}(n)$ of the channel impulse response $\mathbf{h}(n)$ are used for computing the DFE taps.

2) For every time sample n, the channel impulse response estimate $\hat{\mathbf{h}}(n)$ can be updated N_u times. We will be using index i to indicate such an update. Correspondingly, the sequence of normal equations to be solved for the equalizer taps (4.6) is now written as

$$\mathbf{\Gamma}(i)\mathbf{w}(i) = \boldsymbol{\zeta}(i), \;\; i = 0, 1, \ldots, nN_u, \ldots. \tag{5.1}$$

For convenience, in this chapter, we use $\boldsymbol{\zeta}(i)$ to represent $\boldsymbol{\zeta}_s(i)$ in (4.6).

3) For every iteration i, the convolution matrix (4.4) can be approximated as

$$\hat{\mathbf{H}}(i) = \begin{bmatrix} \hat{h}_1(i) & 0 & \cdots & 0 & 0 \\ \hat{h}_2(i) & \hat{h}_1(i) & \ddots & \vdots & \vdots \\ \vdots & \hat{h}_2(i) & \ddots & 0 & \vdots \\ \hat{h}_M(i) & \vdots & \ddots & \hat{h}_1(i) & 0 \\ 0 & \hat{h}_M(i) & \ddots & \hat{h}_2(i) & \hat{h}_1(i) \\ \vdots & 0 & \ddots & \vdots & \hat{h}_2(i) \\ \vdots & \vdots & \ddots & \hat{h}_M(i) & \vdots \\ 0 & 0 & \cdots & 0 & \hat{h}_M(i) \end{bmatrix}. \tag{5.2}$$

4) For every iteration i, the channel estimator updates only one, $p(i)$th, element in $\hat{\mathbf{h}}(i)$ as

$$\hat{h}_{p(i)}(i) = \hat{h}_{p(i)}(i-1) + \Delta\hat{h}(i).$$

5) For every iteration i, only one, $q(i)$th, FFF coefficient in $\hat{\mathbf{f}}(i)$ is updated as

$$\hat{f}_{q(i)}(i) = \hat{f}_{q(i)}(i-1) + \Delta\hat{f}(i),$$

and only one, $\tau(i)$th, FBF coefficient in $\hat{\mathbf{g}}(i)$ is updated as

$$\hat{g}_{\tau(i)}(i) = \hat{g}_{\tau(i)}(i-1) + \Delta\hat{g}(i).$$

The number of iterations for computing the equalizer taps after an update of the channel estimate can be made greater than one. This is a straightforward extension of the algorithm described below. However, our simulation (not presented here) has shown little improvement in the equalizer performance compared to the case of one iteration.

6) We assume that the noise variance σ_ν^2, and thus the signal to noise ratio (SNR) are known.

Table 5.1: Recursively solving a sequence of equations for computation of equalizer taps

Step	Equation
	Initialization: $\boldsymbol{\epsilon}(0) = \mathbf{0}_{(K+B)\times 1}$, $\boldsymbol{\zeta}(0) = \mathbf{0}_{(K+B)\times 1}$, $\hat{\mathbf{w}}(0) = \mathbf{0}_{(K+B)\times 1}$
	for $i = 1, 2, \ldots$
1	Find $\Delta\boldsymbol{\Gamma}(i)$ and $\Delta\boldsymbol{\zeta}(i)$
2	$\boldsymbol{\zeta}_0(i) = \boldsymbol{\epsilon}(i-1) + \Delta\boldsymbol{\zeta}(i) - \Delta\boldsymbol{\Gamma}(i)\hat{\mathbf{w}}(i-1)$
3	Solve $\boldsymbol{\Gamma}(i)\Delta\mathbf{w} = \boldsymbol{\zeta}_0(i) \Rightarrow \Delta\hat{\mathbf{w}}(i),\ \boldsymbol{\epsilon}(i)$
4	$\hat{\mathbf{w}}(i) = \hat{\mathbf{w}}(i-1) + \Delta\hat{\mathbf{w}}(i)$

5.3 Partial-update DFE

We use (5.1) for computation of the equalizer taps $\mathbf{w}(i)$. The sequence of equations (5.1) can be transformed into a sequence of auxiliary normal equations $\boldsymbol{\Gamma}(i)\Delta\mathbf{w}(i) = \boldsymbol{\zeta}_0(i)$ as described in Table 5.1 (see [17]). In Table 5.1, $\hat{\mathbf{w}}(i)$ denotes an approximate solution for the weight vector $\mathbf{w}(i)$ obtained at iteration i, $\boldsymbol{\epsilon}(i)$ is the residual vector $\boldsymbol{\epsilon}(i) = \boldsymbol{\zeta}(i) - \boldsymbol{\Gamma}(i)\hat{\mathbf{w}}(i)$, $\Delta\boldsymbol{\Gamma}(i) = \boldsymbol{\Gamma}(i) - \boldsymbol{\Gamma}(i-1)$, and $\Delta\boldsymbol{\zeta}(i) = \boldsymbol{\zeta}(i) - \boldsymbol{\zeta}(i-1)$.

In Table 5.1, step 1 requires finding $\Delta\boldsymbol{\Gamma}(i)$ which, in general, involves computation of the matrix $\mathbf{G}(i)$ using (4.9) with a complexity of $\mathcal{O}((K+M-1)K^2)$. Step 2 requires $\mathcal{O}(K(K+2B))$ operations to compute $\Delta\boldsymbol{\Gamma}(i)\hat{\mathbf{w}}(i-1)$. Direct computation at step 3 requires $\mathcal{O}((K+B)^3)$ operations. Thus, direct computations according to steps in Table 5.1 would result in significant computational load. In the following, we show how these operations can be simplified when using assumptions presented in Section 5.2.

5.3.1 Computation of $\Delta\boldsymbol{\Gamma}(i)\hat{\mathbf{w}}(i-1)$

Let $\Delta\mathbf{G}(i) = \mathbf{G}(i) - \mathbf{G}(i-1)$, $\tilde{\boldsymbol{\Delta}}(i) = \tilde{\mathbf{H}}(i) - \tilde{\mathbf{H}}(i-1)$, then according to (4.7), we have

$$\Delta\boldsymbol{\Gamma}(i) = \left[\begin{array}{c|c} \Delta\mathbf{G}(i) & -\tilde{\boldsymbol{\Delta}}^H(i) \\ \hline -\tilde{\boldsymbol{\Delta}}(i) & \mathbf{0}_{B\times B} \end{array}\right]. \tag{5.3}$$

Since

$$\hat{\mathbf{w}}(i-1) = \left[\begin{array}{c} \hat{\mathbf{f}}(i-1) \\ \hline \hat{\mathbf{g}}(i-1) \end{array}\right], \tag{5.4}$$

from (5.3) and (5.4) we obtain

$$\Delta\mathbf{\Gamma}(i)\hat{\mathbf{w}}(i-1) = \begin{bmatrix} \Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1) - \tilde{\mathbf{\Delta}}^{H}(i)\hat{\mathbf{g}}(i-1) \\ -\tilde{\mathbf{\Delta}}(i)\hat{\mathbf{f}}(i-1) \end{bmatrix}. \tag{5.5}$$

Now, we derive expressions for computing the following vectors: $\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$, $\tilde{\mathbf{\Delta}}(i)\hat{\mathbf{f}}(i-1)$ and $\tilde{\mathbf{\Delta}}^{H}(i)\hat{\mathbf{g}}(i-1)$.

Computation of $\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$: For computing $\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$ we extend the approach in Chapter 3 to the complex-valued case, and obtain a set of recursive equations (see the derivation in Appendix C). We have

$$\begin{aligned} \Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1) &= \Delta\hat{h}^{*}(i)\mathbf{b}_{p(i):p(i)+K-1}(i-1) \\ &+ \Delta\hat{h}(i)\mathbf{c}_{M-p(i)+1:M-p(i)+K}(i-1) + |\Delta\hat{h}(i)|^{2}\hat{\mathbf{f}}(i-1), \end{aligned} \tag{5.6}$$

where $\mathbf{b}_{p(i):p(i)+K-1}(i-1)$ is a $K\times 1$ vector whose elements are obtained by extracting the $p(i)$th to $p(i)+K-1$th elements from a vector $\mathbf{b}(i-1)$; $\mathbf{c}_{M-p(i)+1:M-p(i)+K}(i-1)$ is a $K\times 1$ vector whose elements are obtained by extracting the $M-p(i)+1$th to $M-p(i)+K$th elements from a vector $\mathbf{c}(i-1)$. The vectors $\mathbf{b}(i-1)$ and $\mathbf{c}(i-1)$ are obtained using the following recursions:

$$\begin{aligned} \mathbf{b}(i-1) &= \mathbf{b}(i-2) + \Delta\hat{f}(i-1)\hat{\mathbf{h}}^{[q(i-1)]}(i-2) \\ &+ \Delta\hat{h}(i-1)\hat{\mathbf{f}}^{[p(i-1)]}(i-1) \end{aligned} \tag{5.7}$$

and

$$\begin{aligned} \mathbf{c}(i-1) &= \mathbf{c}(i-2) + \Delta\hat{f}(i-1)\hat{\mathbf{u}}^{[q(i-1)]}(i-2) \\ &+ \Delta\hat{h}(i-1)\hat{\mathbf{f}}^{[M-p(i-1)+1]}(i-1), \end{aligned} \tag{5.8}$$

where $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$ is a $(K+M-1)\times 1$ vector obtained by shifting elements of $\hat{\mathbf{h}}(i-2)$ by $q(i-1)$ positions down, and the other elements of $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$ are zeros. Definitions for $\hat{\mathbf{f}}^{[p(i-1)]}(i-1)$, $\hat{\mathbf{u}}^{[q(i-1)]}(i-2)$ and $\hat{\mathbf{f}}^{[M-p(i-1)+1]}(i-1)$ are similar to that of $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$. Elements of the vector $\hat{\mathbf{u}}(i-2)$ are given by

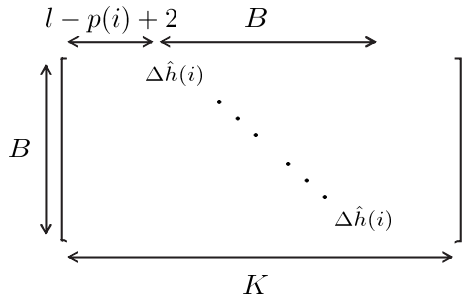
$$\hat{u}_m(i-2) = \hat{h}_{M-m+1}(i-2), \quad m = 1,\ldots,M.$$

Since we propose to use DCD iterations in both the channel estimation and computation of the equalizer taps, $\Delta\hat{h}(i)$ and $\Delta\hat{f}(i)$ are power-of-two numbers. Therefore, all multiplications in (5.6), (5.7) and (5.8), required for computation of $\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$, can be replaced by bit-shift operations.

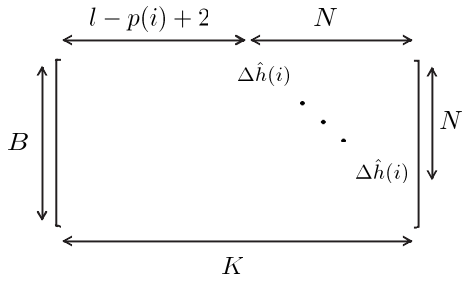
Computation of $\tilde{\mathbf{\Delta}}(i)\hat{\mathbf{f}}(i-1)$: Depending on the position $p(i)$, the matrix $\tilde{\mathbf{\Delta}}(i)$ has different structures as illustrated in Fig.5.1. Thus, the vector $\tilde{\mathbf{\Delta}}(i)\hat{\mathbf{f}}(i-1)$ is given by

$$\tilde{\mathbf{\Delta}}(i)\hat{\mathbf{f}}(i-1) = \Delta\hat{h}(i)\times \begin{cases} \hat{\mathbf{f}}_{l-p(i)+3:l-p(i)+B+2}(i-1), & P < p(i) \le l+2 \\ \left[\begin{array}{c} \hat{\mathbf{f}}_{l-p(i)+3:K}(i-1) \\ \hline \mathbf{0}_{(B-K+l-p(i)+2)\times 1} \end{array}\right], & p(i) \le l+2 \text{ and } p(i) \le P \\ \left[\begin{array}{c} \mathbf{0}_{(p(i)-l-2)\times 1} \\ \hline \hat{\mathbf{f}}_{1:B-p(i)+l+2}(i-1) \end{array}\right], & p(i) > l+2 \text{ and } p(i) \ge P \\ \left[\begin{array}{c} \mathbf{0}_{(p(i)-l-2)\times 1} \\ \hline \hat{\mathbf{f}}(i-1) \\ \hline \mathbf{0}_{(B-K-p(i)+l+2)\times 1} \end{array}\right], & l+2 < p(i) < P \end{cases} \tag{5.9}$$

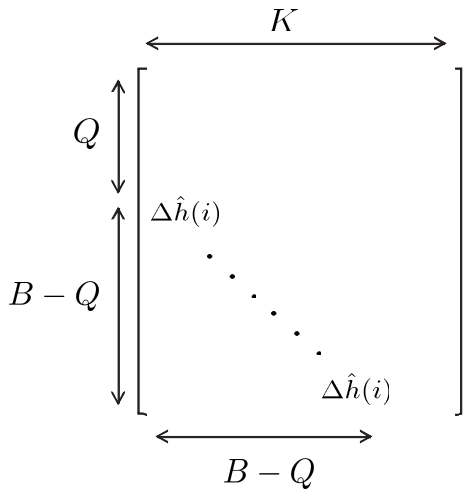
where $P = B - K + l + 2$.



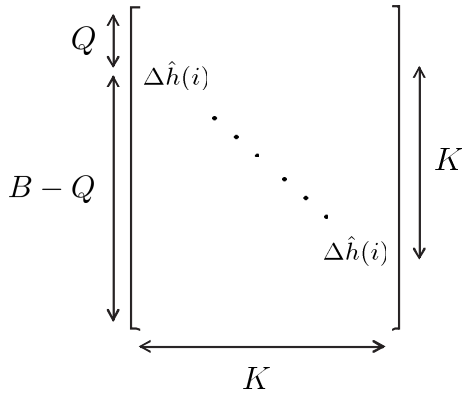
(a) $P < p(i) \leq l+2$



(b) $p(i) \leq l+2$ and $p(i) \leq P$



(c) $p(i) > l+2$ and $p(i) \geq P$



(d) $l+2 < p(i) < P$

Figure 5.1: Structure of matrix $\tilde{\mathbf{\Delta}}(i)$; $P = B-K+l+2$; $N = K-l+p(i)-2$; $Q = p(i)-l-2$.

Computation of $\tilde{\mathbf{\Delta}}^H(i)\hat{\mathbf{g}}(i-1)$: According to Fig.5.1, $\tilde{\mathbf{\Delta}}^H(i)\hat{\mathbf{g}}(i-1)$ can be represented as

$$\tilde{\mathbf{\Delta}}^H(i)\hat{\mathbf{g}}(i-1) = \Delta\hat{h}^*(i)\times \begin{cases} \left[\begin{array}{c} \mathbf{0}_{(l-p(i)+2)\times 1} \\ \hline \hat{\mathbf{g}}(i-1) \\ \hline \mathbf{0}_{(K-B-l+p(i)-2)\times 1} \end{array}\right], & P < p(i) \le l+2 \\ \left[\begin{array}{c} \mathbf{0}_{(l-p(i)+2)\times 1} \\ \hline \hat{\mathbf{g}}_{1:K-l+p(i)-2}(i-1) \end{array}\right], & p(i) \le l+2 \text{ and } p(i) \le P \\ \left[\begin{array}{c} \hat{\mathbf{g}}_{p(i)-l-1:B}(i-1) \\ \hline \mathbf{0}_{(K-B+p(i)-l-2)\times 1} \end{array}\right], & p(i) > l+2 \text{ and } p(i) \ge P \\ \hat{\mathbf{g}}_{p(i)-l-1:p(i)-l+K-2}(i-1), & l+2 < p(i) < P \end{cases} \tag{5.10}$$

Note that with the DCD iterations, $\Delta\hat{h}(i)$ is a power-of-two number. Consequently, all multiplications in (5.9) and (5.10) can be replaced by bit-shift operations. Thus, the computation of $\Delta\mathbf{\Gamma}(i)\hat{\mathbf{w}}(i-1)$ is multiplication-free and division-free.

5.3.2 Computation of $\mathbf{\Gamma}(i)$

According to (4.7), the matrix $\mathbf{\Gamma}(i)$ is defined by $\mathbf{G}(i)$ and $\tilde{\mathbf{H}}(i)$. The matrices $\mathbf{G}(i)$ and $\Delta\mathbf{G}(i)$ are Hermitian Toeplitz matrices and thus defined by the first columns. Consequently, for updating $\mathbf{G}(i)$ only the first column of $\Delta\mathbf{G}(i)$ needs to be computed. In this column, only the first M elements are nonzero and given by

$$\begin{aligned} \Delta G_{1,1}(i) &= 2\Re\left\{\Delta\hat{h}^*(i)\hat{h}_{p(i)}(i-1)\right\} + |\Delta\hat{h}(i)|^2, \\ \Delta G_{1,m}(i) &= \Delta\hat{h}(i)\hat{h}^*_{p(i)-m+1}(i-1) \\ &\quad + \Delta\hat{h}^*(i)\hat{h}_{p(i)+m-1}(i-1), \end{aligned} \tag{5.11}$$

where $m = 2, \ldots, \min\{M, K\}$. Since $\Delta\hat{h}(i)$ is a power-of-two number, all multiplications in (5.11) can be replaced by bit-shift operations.

For $p(i) \leq l+2$, all elements in the first column of $\tilde{\mathbf{H}}(i)$ are zeros. Therefore, for each update of the channel estimate, only element $l - p(i) + 3$ in the first row of $\tilde{\mathbf{H}}(i)$ needs to be updated. For $p(i) > l + 2$, all elements in the first row of $\tilde{\mathbf{H}}(i)$ are zeros, and only element $p(i) - l - 1$ in the first column of $\tilde{\mathbf{H}}(i)$ needs to be updated. Thus, we have

$$\begin{aligned} \tilde{H}_{1,l-p(i)+3}(i) &= \hat{h}_{p(i)}(i), \quad \text{for } p(i) \leq l+2, \\ \tilde{H}_{p(i)-l-1,1}(i) &= \hat{h}_{p(i)}(i), \quad \text{for } p(i) > l+2. \end{aligned}$$

5.3.3 Computation of DFE taps

The proposed technique for computing the equalizer taps is summarized in Table 5.2. The computational complexity will depend on the iterative technique used for solving the equation $\mathbf{\Gamma}\Delta\mathbf{w} = \boldsymbol{\zeta}_0$ at step 10. With the DCD iterations, the equalizer taps are represented as M_b-bit fixed-point numbers within an amplitude interval $[-A, A]$, where A is a power-of-two number. When using the DCD iterations in both the channel estimation and computation of the equalizer taps, the increments $\Delta\hat{h}(i)$, $\Delta\hat{f}(i)$ and $\Delta\hat{g}(i)$ are also power-of-two numbers. Multiplications by these numbers can be replaced by bit-shifts. Therefore, the proposed approach for computing the DFE taps as presented in Table 5.2 is multiplication-free and division-free. Table 5.2 shows the complexity of the computation steps in terms of additions.

Table 5.3 presents a DCD iteration for updating one FFF tap and one FBF tap.

5.4 Simulation results

In this section, we compare the performance of six DFEs as summarized in Table 5.4:

1) MMSE DFE (known decision).

For every time sample n, the convolution matrix $\mathbf{H}(n)$ is formed using (4.4) and the channel impulse response $\mathbf{h}(n)$ is perfectly known. The FFF vector $\mathbf{f}(n)$ is found by solving (4.14), and the FBF vector $\mathbf{b}(n)$ is obtained using (4.12). Correct decisions are perfectly known and used in the FBF.

Table 5.2: Low-complexity computation of equalizer taps

Step	Equation	+
	Initialization: $i=0$, $\mathbf{G}(0)=\sigma_\nu^2\mathbf{I}_K$, $\hat{\mathbf{w}}(0)=\mathbf{0}_{(K+B)\times 1}$, $\boldsymbol{\epsilon}(0)=\mathbf{0}_{(K+B)\times 1}$, $\mathbf{b}(0)=\mathbf{0}_{(K+M-1)\times 1}$, $\mathbf{c}(0)=\mathbf{0}_{(K+M-1)\times 1}$, $\tilde{\mathbf{H}}(0)=\mathbf{0}_{B\times K}$	
	for $n=1,2,\ldots$	
	for $k=1,\ldots,N_u$	
1	$i=i+1$	
2	Use one iteration in the channel estimator and obtain $\hat{\mathbf{h}}(i)$, $\Delta\hat{h}(i)$ and position $p(i)$	
3	$\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)=\Delta\hat{h}^*(i)\mathbf{b}_{p(i):p(i)+K-1}(i-1)$ $+\Delta\hat{h}(i)\mathbf{c}_{M-p(i)+1:M-p(i)+K}(i-1)+\lvert\Delta\hat{h}(i)\rvert^2\hat{\mathbf{f}}(i-1)$	$4K$
4	Compute $\tilde{\boldsymbol{\Delta}}(i)\hat{\mathbf{f}}(i-1)$ and $\tilde{\boldsymbol{\Delta}}^H(i)\hat{\mathbf{g}}(i-1)$ using (5.9) and (5.10), respectively	$-$
5	Use $\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$, $\tilde{\boldsymbol{\Delta}}(i)\hat{\mathbf{f}}(i-1)$ and $\tilde{\boldsymbol{\Delta}}^H(i)\hat{\mathbf{g}}(i-1)$ to represent $\Delta\boldsymbol{\Gamma}(i)\hat{\mathbf{w}}(i-1)$ according to (5.5)	$2K$
6	$\boldsymbol{\zeta}_0=\boldsymbol{\epsilon}(i-1)+\Delta\hat{h}^*(i)\mathbf{e}_{l-p(i)+2}-\Delta\boldsymbol{\Gamma}(i)\hat{\mathbf{w}}(i-1)$	$2(K+B)+1$
7	Compute $\Delta\mathbf{G}^{(1)}(i)$ using (5.11) and update $\mathbf{G}^{(1)}(i)=\mathbf{G}^{(1)}(i-1)+\Delta\mathbf{G}^{(1)}(i)$	$4M-2$
8	$\tilde{\mathbf{H}}_{(1)}(i)=\tilde{\mathbf{H}}_{(1)}(i-1)$ and $\tilde{\mathbf{H}}^{(1)}(i)=\tilde{\mathbf{H}}^{(1)}(i-1)$ For $p(i)\le l+2$, $\tilde{H}_{1,l-p(i)+3}(i)=\hat{h}_{p(i)}(i)$, and for $p(i)>l+2$, $\tilde{H}_{p(i)-l-1,1}(i)=\hat{h}_{p(i)}(i)$	1
9	Use $\mathbf{G}(i)$ and $\tilde{\mathbf{H}}(i)$ to represent $\boldsymbol{\Gamma}$ according to (4.7)	$-$
10	Solve $\boldsymbol{\Gamma}\Delta\mathbf{w}=\boldsymbol{\zeta}_0$ using Table 5.3 and obtain $\Delta\hat{f}(i)$, $\Delta\hat{g}(i)$, $q(i)$, $\tau(i)$ and $\boldsymbol{\epsilon}(i)$	$\le 8K+8B+6$
11	$\hat{f}_{q(i)}(i)=\hat{f}_{q(i)}(i-1)+\Delta\hat{f}(i)$ and $\hat{g}_{\tau(i)}(i)=\hat{g}_{\tau(i)}(i-1)+\Delta\hat{g}(i)$	2
12	$\mathbf{b}(i)=\mathbf{b}(i-1)+\Delta\hat{f}(i)\hat{\mathbf{h}}^{[q(i)]}(i-1)+\Delta\hat{h}(i)\hat{\mathbf{f}}^{[p(i)]}(i)$	$2(K+M)$
13	$\mathbf{c}(i)=\mathbf{c}(i-1)+\Delta\hat{f}(i)\hat{\mathbf{u}}^{[q(i)]}(i-1)+\Delta\hat{h}^*(i)\hat{\mathbf{f}}^{[M-p(i)+1]}(i)$	$2(K+M)$
	Total for each sample n: 0 real mult. and $\le N_u(20K+10B+8M+8)$ real adds.	

2) RLS CE based adaptive DFE (known decision).

The time-varying channel is estimated using the classical RLS algorithm [4] with a forgetting factor λ, and for every sample n, the convolution matrix $\mathbf{H}(n)$ is given by (4.4). The FFF vector $\mathbf{f}(n)$ is found by solving (4.14), and the FBF vector $\mathbf{b}(n)$ is obtained using (4.12). Correct decisions are perfectly known and used in the FBF.

3) RLS CE based adaptive DFE.

This is similar to the DFE in 2) except that the correct transmitted symbols are unknown; the decisions on the symbols obtained from the equalizer are used in the FBF.

4) RLS directly adaptive (DA) DFE.

In the DA DFE, $\boldsymbol{\Gamma}(n)$ and $\boldsymbol{\zeta}(n)$ are estimated without explicit channel estimation using the received data and known pilot or estimated data symbols. The equalizer taps are then obtained by solving (4.6) using the RLS algorithm.

5) Proposed DFE (known decision).

The time-varying channel is estimated using the RLS-DCD algorithm presented in Sec-

Table 5.3: DCD iteration for solving $\mathbf{\Gamma}\Delta\mathbf{w} = \boldsymbol{\zeta}_0$.

Step	Equation	+
	Initialization: $\Delta\hat{\mathbf{w}} = \mathbf{0}_{(K+B)\times 1}$, $\boldsymbol{\epsilon} = \boldsymbol{\zeta}_0$, $\alpha = A/2$, $\Delta\hat{f}(i) = 0$, $\Delta\hat{g}(i) = 0$, $a = 1$, $\mathbf{d} = [\alpha, -\alpha, j\alpha, -j\alpha]$	
	for $\rho = 1, 2$	
1	if $\rho = 1$, $\mathbf{v} = [\Re\{\boldsymbol{\epsilon}_{1:K}^T\}\ \Im\{\boldsymbol{\epsilon}_{1:K}^T\}]$, $q = \arg\max_{m=1,\ldots,2K}\{\lvert v_m\rvert\}$, if $q > K$, then $q \leftarrow q - K$	$2K$
2	if $\rho = 2$, $\mathbf{v} = [\Re\{\boldsymbol{\epsilon}_{K+1:K+B}^T\}\ \Im\{\boldsymbol{\epsilon}_{K+1:K+B}^T\}]$, $q = \arg\max_{m=1,\ldots,2B}\{\lvert v_m\rvert\}$, if $q > B$, then $q \leftarrow K + q - B$, if $q \leq B$, then $q \leftarrow K + q$	$2B$
3	go to step 6	$-$
4	$a \leftarrow a + 1$, $\alpha \leftarrow \alpha/2$, $\mathbf{d} = [\alpha, -\alpha, j\alpha, -j\alpha]$	$-$
5	if $a > M_b$, end the loop	$-$
6	$\mu = \arg\min\{[-\Re\{\epsilon_q\}, \Re\{\epsilon_q\}, -\Im\{\epsilon_q\}, \Im\{\epsilon_q\}, -\alpha\Gamma_{q,q}/2]\}$ if $\mu > 4$, then go to step 4	2
7	$\Delta\hat{w}_q = d_\mu$	1
8	$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - d_\mu \mathbf{\Gamma}^{(q)}$	$2K + 2B$
9	if $\rho = 1$, $\Delta\hat{f}(i) = \Delta\hat{w}_q$ and $q(i) = q$	$-$
10	if $\rho = 2$, $\Delta\hat{g}(i) = \Delta\hat{w}_q$ and $\tau(i) = q - K$	$-$
	$\boldsymbol{\epsilon}(i) = \boldsymbol{\epsilon}$	
	Total: 0 real mult. and $\leq 8K + 8B + 6$ real adds.	

tion 4.4. For every i, the equalizer taps are obtained using the algorithm in Table 5.2. Correct decisions are perfectly known and used in the FBF.

6) Proposed DFE.

This is similar to the DFE in 5) except that the correct decisions are unknown; decisions obtained from the equalizer are used in the FBF.

In the simulation, we consider four time-varying channels:

1) Short time-varying channel (Jakes' model).

The channel has a uniform power delay profile of length $M = 21$ and path variance $1/M$. For modeling the time variation, the Jakes' model as described in [57] with a normalized Doppler frequency (ratio of the Doppler frequency to the symbol rate) of $f_d = 10^{-4}$ is used.

2) Long time-varying channel (Jakes' model).

The channel of length $M = 101$ has a sparse power delay profile with 11 non-zero paths of

Table 5.4: DFEs used in the simulation

Algorithm	Channel	Decision	Convolution matrix $\mathbf{H}(n)$	Computation of FFF taps	Computation of FBF taps
MMSE (known decision)	known	known	formed using (4.4)	solving (4.14) directly	using (4.12)
RLS CE (known decision)	estimated using RLS	known	formed using (4.4)	solving (4.14) directly	using (4.12)
RLS CE	estimated using RLS	estimated	formed using (4.4)	solving (4.14) directly	using (4.12)
RLS DA	not required	estimated	not required	estimated using RLS	estimated using RLS
Proposed (known decision)	estimated using RLS-DCD	known	formed using (5.2)	using Table 5.2	using Table 5.2
Proposed	estimated using RLS-DCD	estimated	formed using (5.2)	using Table 5.2	using Table 5.2

variance $1/11$. Delays of the non-zero paths are randomly generated for every simulation trial. The Jakes' model with a normalized Doppler frequency of $f_d = 10^{-4}$ is used to model the channel time variations.

3) Long time-varying channel (autoregressive model).

This is similar to the channel model 2). However, for modeling the channel time variations, we adopt the first order autoregressive model given by $\mathbf{h}(n) = \sqrt{v}\ \mathbf{h}(n-1) + \sqrt{1-v}\ \boldsymbol{\omega}(n)$ [54], where $\sqrt{v}$ is the autoregressive factor and $\boldsymbol{\omega}(n)$ are zero-mean independent random Gaussian vectors, whose elements have variance $1/M$; $v = 10^{-4}$.

4) Underwater acoustic channel.

The time-varying underwater acoustic channel is modeled as described in [58] for a deep-water environment. In this scenario, the receiver is stationary at a depth of 400 m and the transmitter is moving at a speed of v_s at a depth of 200 m. The initial distance between the transmitter and receiver is 40 km. The delay spread of the channel is about 150 ms and a channel estimator of length $M = 201$ is used. An example of the channel impulse response is shown in Fig.5.2.

In all the simulation scenarios, QPSK symbols are transmitted. In channels 1 to 3, these are baseband symbols. In the underwater acoustic channel, the symbols are transmitted at a carrier frequency of 3072 Hz and the symbol rate is 1024 Hz. The received signal is first transformed to a baseband signal by a carrier frequency shift accounting for the speed of the transmitter. The baseband signal is further resampled to take the time compression

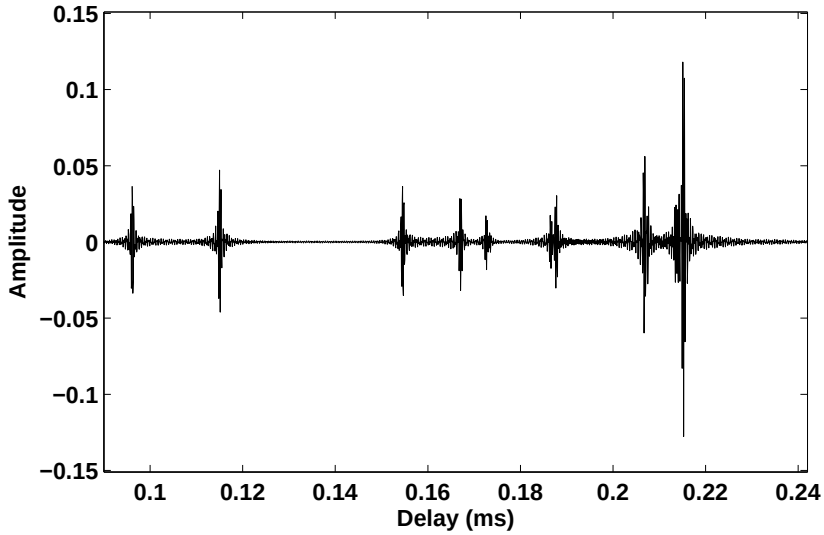


Figure 5.2: An example of the underwater acoustic channel impulse response.

caused by the transmitter motion into account. The baseband resampled signal is used for the equalization.

For every signal to noise ratio (SNR), the bit-error-rate (BER) is computed by averaging over 400 simulation trials. For each trial, a 2000-length data sequence of unit power is transmitted, which contains a sequence of L pilot symbols followed by $2000 - L$ information symbols. In the CE based DFEs, channel estimates are first obtained from the pilot symbols and then from the equalized data symbols.

Fig.5.3 compares the BER performance of the six DFEs in a scenario with the short time-varying channel. For each SNR, the RLS forgetting factor is chosen in the interval $0.988 \leq \lambda \leq 0.997$, so that the minimum BER is achieved. The MMSE (known decision) DFE that possesses the perfect channel knowledge and uses the true transmitted symbols to feed the FBF provides the best performance. The RLS CE DFE (known decision) that also uses the true transmitted symbols in the FBF, but estimates the channel, has an inferior performance by about 1 dB at BER$= 10^{-4}$. The proposed DFE (known decision) with $N_u = 4$ demonstrates almost the same performance as the RLS CE DFE (known decision). When the FBF is fed by decisions made by the equalizer, the performance of the proposed DFE with $N_u = 4$ and $N_u = 8$ is almost the same as that of the RLS CE DFE and about 1.3 dB inferior to the MMSE (known decision) DFE at BER$= 10^{-4}$. With $N_u = 2$ updates, the proposed DFE is inferior to the RLS CE DFE by as little as

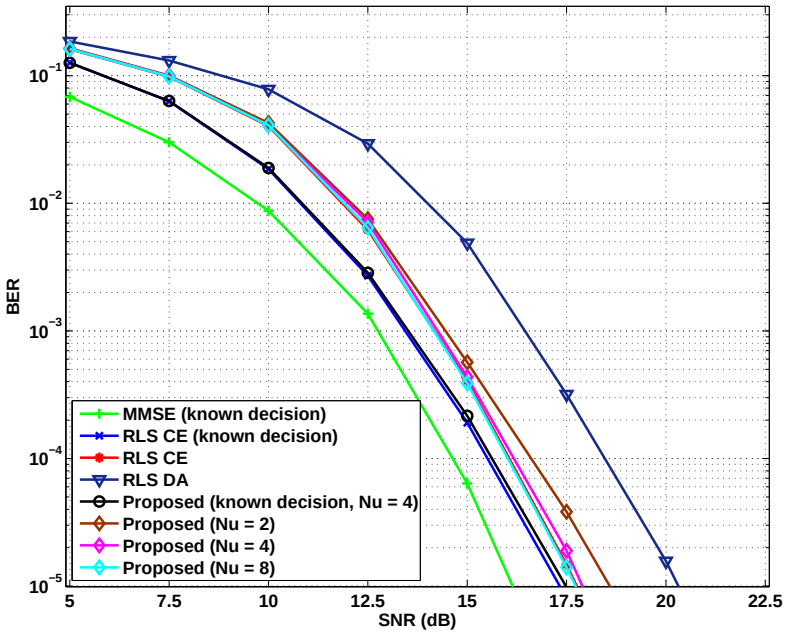


Figure 5.3: BER performance of the six DFEs in the short time-varying channels (Jakes' model): $f_d = 10^{-4}$, $M = 21$, $K = 41$, $B = 21$, $M_b = 16$, $A = 1$. The number of pilot symbols for the RLS DA DFE is $L = 600$; for the other DFEs: $L = 100$.

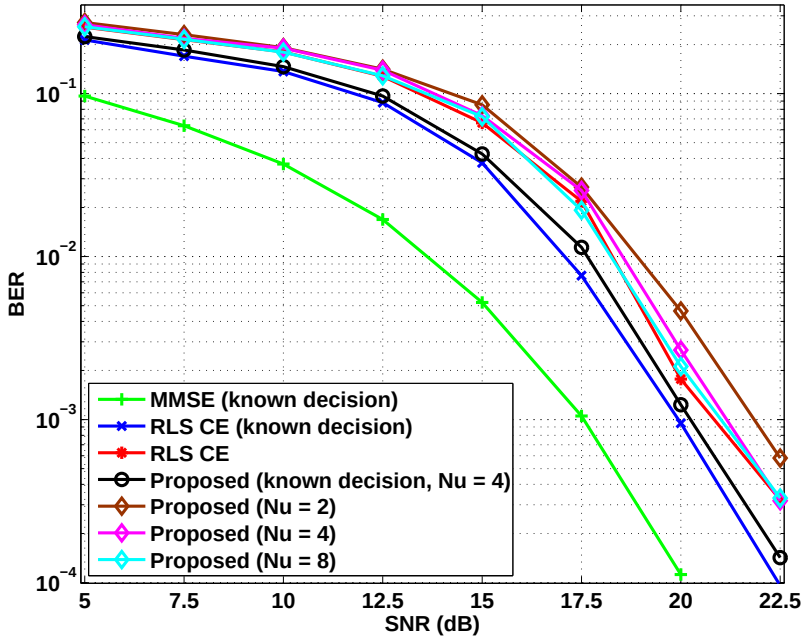


Figure 5.4: BER performance of the DFEs in the long time-varying channels (Jakes' model): $f_d = 10^{-4}$, $M = 101$, $K = 201$, $B = 101$, $M_b = 16$, $A = 1$. The number of pilot symbols is $L = 300$.

0.5 dB. Thus, in this scenario, the proposed DFE with $N_u = 2$ updates provides a BER performance which is very close to that of the RLS CE DFE. The direct adaptation DFE is inferior to the proposed DFE with $N_u = 2$ by about 1.9 dB. Note that the RLS DA DFE also requires a much longer pilot ($L = 600$) than the other techniques ($L = 100$).

Fig.5.4 compares the BER performance of the DFEs in a scenario with the long time-varying channel. The RLS forgetting factor is chosen within the interval $0.988 \leq \lambda \leq 0.997$ to achieve the best BER performance. The performance of the proposed DFE with $N_u = 4$ is very close to that of the RLS CE DFE. However, at BER $= 10^{-4}$, the proposed DFE with $N_u = 2$ is inferior to the RLS CE DFE by about 1.5 dB.

Simulation results for the autoregressive model of the channel variations, shown in Fig.5.5, are similar to that in Fig.5.4 with a common shift of the BER curves towards the lower SNRs. We do not show results for the RLS DA DFE due to its significantly low performance compared to the other DFEs.

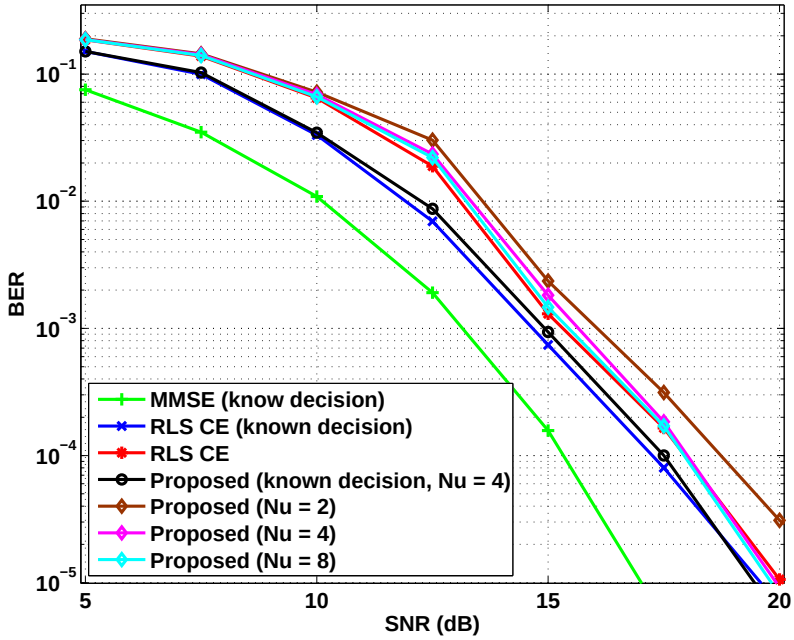
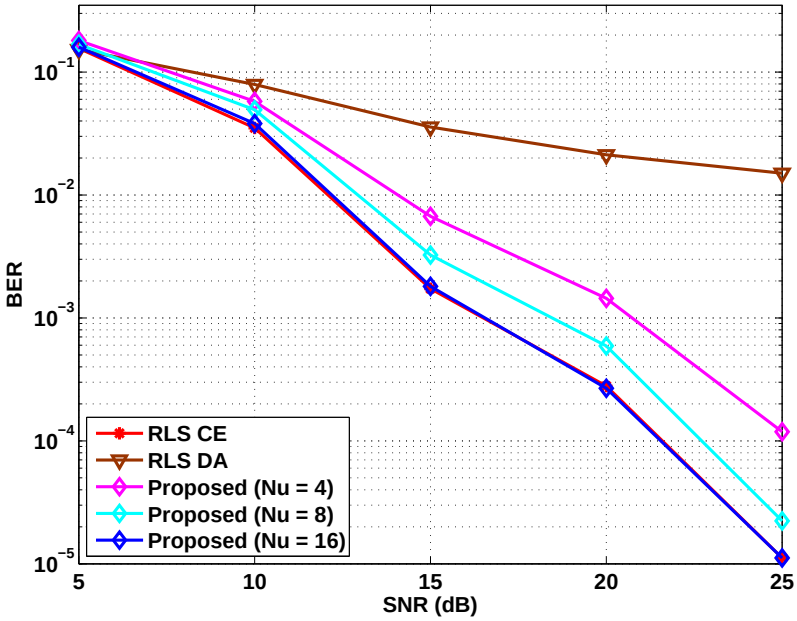


Figure 5.5: BER performance of the DFEs in the long time-varying channels (autoregressive model): $\upsilon = 10^{-4}$, $M = 101$, $K = 201$, $B = 101$, $M_b = 16$, $A = 1$. The number of pilot symbols is $L = 300$.

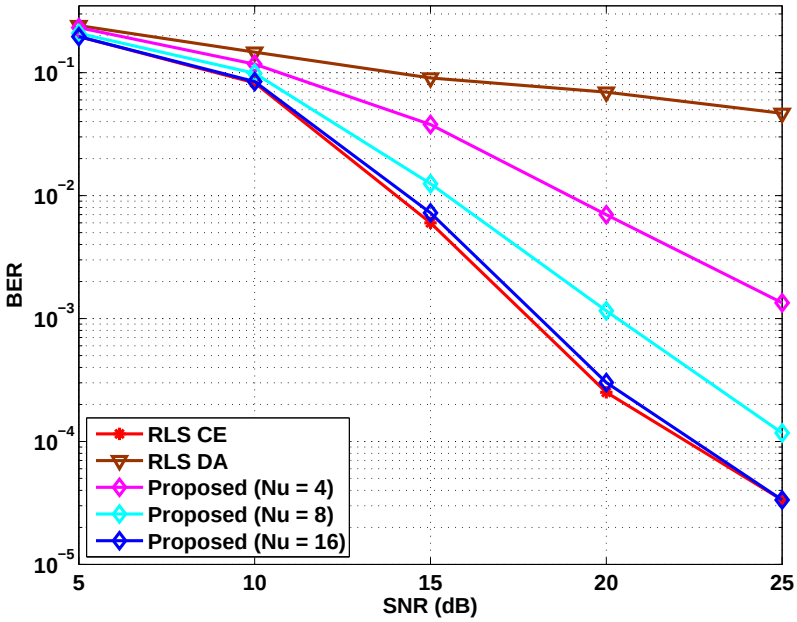
Fig.5.6 compares the BER performance of the RLS CE and proposed DFEs in the scenario with the underwater acoustic channel. Again, for each SNR, the RLS forgetting factor is chosen within the interval $0.988 \leq \lambda \leq 0.997$ to minimize the BER. This is a channel with a high multipath spread (we use $M = 201$ in the channel estimator) and, consequently, the lengths of the FFF and FBF increase to $K = 401$ and $B = 201$, respectively. In this case, the performance of the proposed DFE approaches that of the RLS CE DFE with $N_u = 16$, higher than in the previous scenarios.

It can be seen that the increase in the channel length requires a proportional increase in the number of updates N_u in the proposed DFE to approach closely the performance of the RLS CE DFE. However, in all the cases, we still can see that $N_u \ll M$.

We now analyze the complexity of the proposed DFE in comparison to the complexity of the MMSE DFE where the DFE taps are computed using the fast technique proposed in [28]. We take into account the complexity of computing the equalizer taps and also the equalization (FFF and FBF filtering). Note that the proposed DFE requires no mul-



(a) Speed of the transmitter: $v_s = 0.5$ m/s



(b) Speed of the transmitter: $v_s = 5$ m/s

Figure 5.6: BER performance of DFEs in the underwater acoustic channel: $M = 201$, $K = 401$, $B = 201$, $L = 500$, $M_b = 16$, $A = 1$.

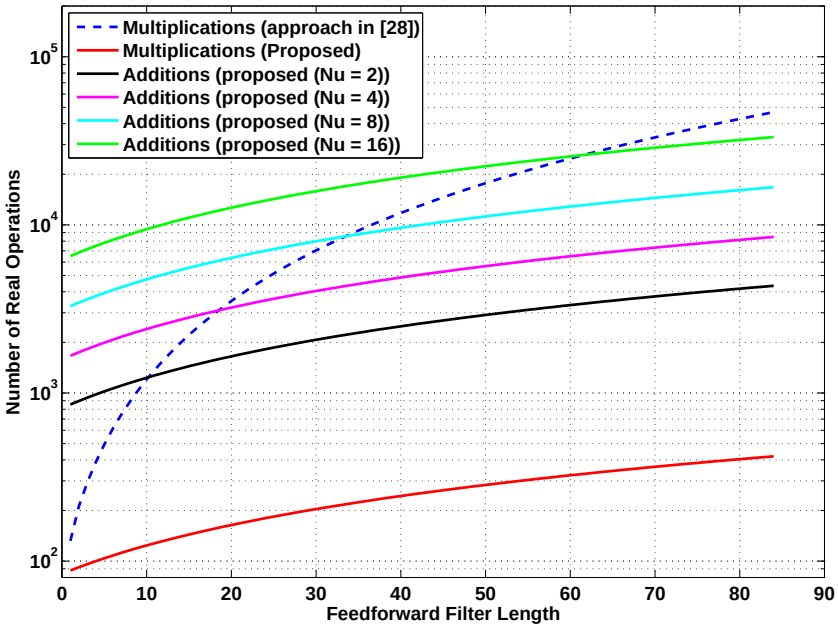


Figure 5.7: Complexity of computing DFE taps and equalization for the short channel against the FFF length K; $M = 21$, $B = 21$, $M_b = 16$.

tiplications for computing the equalizer taps; however, the number of additions can be substantial. Therefore, we analyze both the number of multiplications and number of additions in the DFEs. For the proposed DFE, we use the upper bound on the number of additions given in the lowest row of Table 5.2. For the MMSE DFE [28], the number of additions involved in the computation of the DFE taps is on the same order as the number of multiplications. Thus, only the number of multiplications in the MMSE DFE is considered.

Fig.5.7 shows the number of operations per sample as a function of the FFF length K for the short channel case; the other parameters of the equalizers are as shown in the caption to Fig.5.3. It can be seen that the number of multiplications in the proposed DFE (which are only used for the equalization) is significantly lower than the number of multiplications in the MMSE DFE. For $K = 41$ (as used in the simulation above in Fig.5.3), the difference is approximately 40 times. Assuming that the number of additions required by the MMSE DFE is the same as the number of multiplications, we notice that the proposed DFE requires fewer additions than the MMSE DFE by about 4 times for $N_u = 2$ and about 2 times for $N_u = 4$, when the performance of the two equalizers is

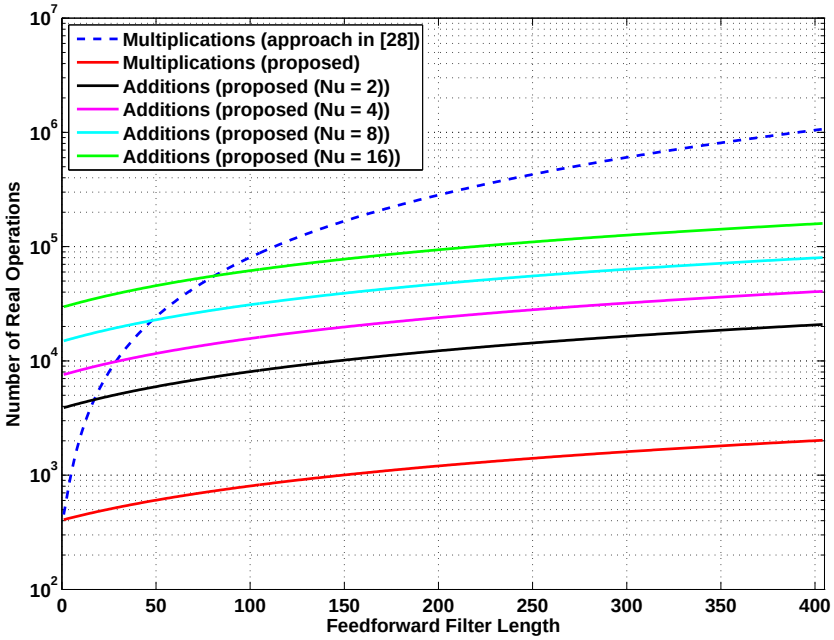


Figure 5.8: Complexity of computing DFE taps and equalization for the long channel against the FFF length K; $M = 101$, $B = 101$, $M_b = 16$.

similar. The significant reduction in the number of multiplications is beneficial when a hardware design platform such as the FPGA platform is used. With increase in the FFF length, the proposed technique shows more significant reduction in complexity compared to the MMSE DFE.

Fig.5.8 shows the complexity of the two equalizers for the long channel case. In this case, for $K = 201$ as used in the simulation above (see Fig.5.4), the proposed technique allows reduction in the number of multiplications by about 300 times. The number of additions for $N_u = 8$, that provides almost the same performance for the two equalizers, is about 3 times lower than that of the MMSE DFE. With the increase in K, the difference in the complexity also increases.

5.5 Conclusions

In this chapter, we have proposed a novel low complexity technique for computation of equalizer taps in the channel estimate (CE) based decision feedback equalizer (DFE). The proposed technique operates with partial-update channel estimators, such as RLS-DCD channel estimator, and based on the dichotomous coordinate descent (DCD) iterations that allow the equalizer tap computation to be multiplication-free and division-free. Thus, this technique is very attractive for design on hardware platforms such as the FPGA platform. The complexity of the proposed technique is upper bounded by a value of $\mathcal{O}(N_u K) + \mathcal{O}(N_u B) + \mathcal{O}(N_u M)$ operations per sample. We have applied the proposed DFE and known DFEs to two time-varying Rayleigh fading channel models and a time-varying underwater acoustic channel model. The simulation results have shown that with $N_u \ll M$, the proposed DFE provides the BER performance similar to that of the RLS CE DFE. Up to this point, we have investigated low-complexity channel estimation and CE based equalization techniques for underwater acoustic communications.

Chapter 6

Matched-phase coherent broadband matched-field processor using phase descent search

Contents

Localization is important for underwater acoustic communications in many aspects. For example, by knowing the location of the transmitter and the acoustic fields, time synchronization can be easily achieved at the receiver. In underwater sensor networks, distributed sensors are used to collect specific data which can be meaningless if the location of the sensor is unknown. Global positioning system (GPS) [86] which uses radio frequency is a well-known technique for terrestrial localization. However, since radio frequency are severely attenuated in underwater [3, 87], GPS can not be used for underwater source localization. Acoustic waves which can propagate over very long distances

in underwater have been considered as the most robust and feasible carrier for underwater source localization [8, 10]. Many beamforming techniques [88–90] which rely on time differences of arrival and direction of arrival estimation, have been developed for terrestrial acoustic source localization. However, most of these techniques are based on plane wave signals which are usually not the case in the ocean waveguide [91]. This is mainly due to the characteristics of underwater acoustic channels, such as variable speed of sound, and unavoidable movement of the source and receiver [1–3]. Matched-field processing (MFP) [32, 91] which explores the spatial complexities of acoustic fields in an ocean waveguide to locate sources has attracted much research interest in the past few decades. It does not rely on plane wave signals and provides superior performance than plane wave methods for underwater source localization [91]. Due to bandwidth limitations of underwater acoustic channels, receivers are required to process broadband communications signals. Therefore, in this chapter, we are interested in broadband MFP techniques [33, 37, 38, 92] for underwater acoustic source localization.

This chapter is organized as follows. In the next section, an introduction is given. In Section 6.2, the data model is described. In Section 6.3, the matched-phase coherent MF processor is introduced and the cross-frequency incoherent processor is reviewed. The PDS algorithm and the frequency estimation technique are introduced in Section 6.4 and Section 6.5, respectively. Application of the proposed processor to experimental data is presented in Section 6.6. Finally, Section 6.7 gives conclusions.

6.1 Introduction

Matched-field processing (MFP) has been widely used in ocean acoustic applications, such as source localization [37, 93] and estimation of ocean parameters [94, 95]. For locating an acoustic source, the MFP computes a set of modeled acoustic fields, ”replicas”, at a hydrophone array. Each replica is produced for a particular source location in the underwater environment of interest. The measured acoustic field, ”data”, collected by the real hydrophone array is then matched with each of the replicas. This produces an ambiguity surface, which shows the correlation between each of the replicas and the data. The

peak in the ambiguity surface should indicate the true source position, where the replica and the data are well correlated, provided that the propagation model used to generate the replicas is accurate.

Broadband (or multi-frequency) MFP has been actively investigated in the past two decades [33–38, 92, 96]. Coherent combining of ambiguity surfaces obtained at different frequencies provides better performance compared to incoherent combining. In scenarios where an acoustic source transmits sound at multiple frequencies, phases of the source frequencies contribute in the measured acoustic data. The phase shifts between different frequencies should be compensated before the MFP; however, they are often unknown. In order to compensate for these phase shifts, a matched-phase coherent processor was proposed [38]. This processor has been shown to outperform other advanced MF processors, especially when the ambient noise level and environment mismatch are significant [38]. A cross-frequency processor, which can be seen as an incoherent version of the matched-phase processor, is then proposed in [39]; it has been shown that this processor provides similar maximum of the ambiguity surface as the matched-phase coherent processor.

In [38], it was proposed to search the phase shifts by using the simulated annealing algorithm, which is well known for its ability for solving global optimization problems while having high computational complexity. Although different approaches have been proposed to reduce the complexity [40, 41], it is still very high and increases dramatically as the number of free parameters increases. This prevents simultaneous processing of many frequencies, and thus, limits the processor performance. Furthermore, for most of the simulated annealing methods, it is found to be exhausting to determine some algorithm parameters such as the initial temperature and the cooling schedule, which need to be carefully set. For all these reasons, we propose to search the matched phases by using a novel iterative technique, the phase descent search (PDS) algorithm [97] which is based on coordinate descent iterations with respect to the unknown phases and constrains the solution to have a unit magnitude. Since coordinate descent optimization is mainly applicable to solving convex problems, it is not clear how it will behave in application to the phase search problem which has been considered as a global optimization problem [38]. In this work, we investigate the application of the PDS algorithm to this problem and show that it can obtain matched phases similar to that obtained by the simulated annealing. The PDS algorithm has significantly lower complexity as compared with simulated annealing

methods, and thus, enables searching matched phases for a large number of processed frequencies. This can significantly improve the processor performance. In addition, the PDS algorithm is simple for practical implementations since all the algorithm parameters can be easily chosen.

For localization of a fast moving acoustic source, frequency correction is required, in order to capture the information on the shifted transmission frequencies before applying MFP. In this work, we employ a frequency estimator with dichotomous search of periodogram peak [98] for estimating the transmitted frequencies in the received data. Due to the fast movement of the source, in order to achieve accurate localization at each time instant, a short data record (a few short snapshots) has to be used for MFP. Thus, the ability of an MF processor to solve the localization problem with a short data record is very important. We apply the proposed MF processor to the data collected in the SWellEx-96 experiment using as short as 1-second snapshots and show accurate localization results.

Notations: In this chapter, we use capital and small bold fonts to denote matrices and vectors, respectively. For example, $\mathbf{R}$ and $\mathbf{d}$ represent a matrix and a vector, respectively. Elements of the matrix and vector are denoted as $R_{m,n}$ and d_i. A pth column of $\mathbf{R}$ is denoted as $\mathbf{R}^{(p)}$. $\mathbf{d}^H$ is the Hermitian transpose of the vector $\mathbf{d}$. diag $\{\mathbf{R}\}$ denotes a vector whose entries are diagonal elements of $\mathbf{R}$. Other notations used throughout this chapter are defined when considered.

6.2 Data Model

We consider a single acoustic source transmitting sound at multiple frequencies, and the source position can be characterized by range and depth. The data model for the signal received by the ith hydrophone of an M-hydrophone array at frequency ω is given by

$$d_i(\omega) = h_i(\omega)s(\omega) + e_i(\omega), \tag{6.1}$$

where $d_i(\omega)$ is the measured acoustic pressure, $h_i(\omega)$ is the channel transfer function, $s(\omega)$ is the source signal, and $e_i(\omega)$ is a zero-mean stochastic process representing additive observation noise. We can define vectors $\mathbf{h}(\omega) = \{h_i(\omega)\}_{i=1}^M$ and $\mathbf{e}(\omega) = \{e_i(\omega)\}_{i=1}^M$ for the channel transfer function and the additive observation noise, respectively. The data

model can then be represented by

$$\mathbf{d}(\omega) = \mathbf{h}(\omega)s(\omega) + \mathbf{e}(\omega), \tag{6.2}$$

where $\mathbf{d}(\omega) = \{d_i(\omega)\}_{i=1}^{M}$ is a "data" vector containing the measured acoustic pressure field at the M-hydrophone array. We also define a complex-valued "replica" vector $\mathbf{p}(\omega, \mathbf{x}) = \{p_i(\omega, \mathbf{x})\}_{i=1}^{M}$ which contains the modeled acoustic pressure field at the M-hydrophone array, where $p_i(\omega, \mathbf{x})$ is a modeled solution to the acoustic wave equation at the ith hydrophone for a source located at $\mathbf{x}$ and transmitting acoustic signal at frequency ω.

6.3 Broadband matched-field processing

In this section, we review the single-frequency Bartlett processor and its extension dealing with multiple frequencies, the multi-frequency coherent processors. Then the matched-phase coherent processor which requires searching the phases of the replica is considered, and an alternative expression of its ambiguity function is derived. Finally, the incoherent version of this matched-phase processor called the cross-frequency incoherent processor is also considered, which does not require any phase search.

6.3.1 Single-frequency Bartlett processor

The single-frequency Bartlett processor is an MF processor which averages the projection of the data vectors $\mathbf{d}(\omega)$ at radial frequency ω on the normalized replica vector $\mathbf{u}(\omega, \mathbf{x}) = \mathbf{p}(\omega, \mathbf{x}) / |\mathbf{p}(\omega, \mathbf{x})|$ at radial frequency ω and spatial coordinate $\mathbf{x}$. It produces the ambiguity function [33]

$$B_B(\omega, \mathbf{x}) = \frac{\left\langle \left| \mathbf{d}^H(\omega)\mathbf{u}(\omega, \mathbf{x}) \right|^2 \right\rangle_T}{tr\left[\mathbf{D}(\omega)\right]} \tag{6.3}$$

where we denote $\langle \ldots \rangle_T$ as the time average, $tr\left[\mathbf{A}\right]$ as the trace of a matrix $\mathbf{A}$, and $\mathbf{D}(\omega) = \left\langle \mathbf{d}(\omega)\mathbf{d}^H(\omega) \right\rangle_T$. By defining a normalized covariance matrix $\mathbf{K}(\omega) = \mathbf{D}(\omega)/tr\left[\mathbf{D}(\omega)\right]$, (6.3) can be written as

$$B_B(\omega, \mathbf{x}) = \mathbf{u}^H(\omega, \mathbf{x})\mathbf{K}(\omega)\mathbf{u}(\omega, \mathbf{x}). \tag{6.4}$$

6.3.2 Matched-phase coherent processor

In [38], the coherent broadband MF processor is defined based on the single-frequency Bartlett processor (6.4) but taking account for the nonzero phase difference between frequencies. It is given by [38]

$$\begin{aligned} B_C(\mathbf{x}) &= \left\langle \left| \frac{1}{L} \sum_{n=1}^{L} \frac{\mathbf{d}^H(\omega_n)\mathbf{u}(\omega_n,\mathbf{x})}{\sqrt{tr\left[\mathbf{D}(\omega_n)\right]}} e^{j\hat{\phi}_n} \right|^2 \right\rangle_T \\ &= \frac{1}{L^2} \sum_{m,n=1}^{L} \mathbf{u}_m^H \mathbf{K}_{m,n} \mathbf{u}_n e^{j(\hat{\phi}_n - \hat{\phi}_m)}, \end{aligned} \tag{6.5}$$

where

$$\mathbf{K}_{m,n} = \mathbf{K}(\omega_m,\omega_n) = \frac{\left\langle \mathbf{d}(\omega_m)\mathbf{d}^H(\omega_n) \right\rangle_T}{\sqrt{tr\left[\mathbf{D}(\omega_m)\right]}\sqrt{tr\left[\mathbf{D}(\omega_n)\right]}}, \tag{6.6}$$

$\mathbf{u}_n = \mathbf{u}(\omega_n,\mathbf{x})$. Here the phase estimates $\hat{\boldsymbol{\phi}} = \left\{\hat{\phi}_n\right\}_{n=1}^{L}$ are given by

$$\hat{\boldsymbol{\phi}} = \arg\max_{[\mathbf{x},\boldsymbol{\phi}]} \left\{ \sum_{m,n=1}^{L} \mathbf{u}_m^H(\mathbf{x}) \mathbf{K}_{m,n} \mathbf{u}_n(\mathbf{x}) e^{j(\phi_n - \phi_m)} \right\}, \tag{6.7}$$

where $\boldsymbol{\phi} = \{\phi_n\}_{n=1}^{L}$. Equation (6.5) can be divided into two terms as

$$\begin{aligned} B_C(\mathbf{x}) &= \frac{1}{L^2} \left[\sum_{m=1}^{L} \mathbf{u}_m^H \mathbf{K}_{m,m} \mathbf{u}_m \right. \\ &\quad \left. + \sum_{m\neq n}^{L} \mathbf{u}_m^H \mathbf{K}_{m,n} \mathbf{u}_n e^{j(\hat{\phi}_n - \hat{\phi}_m)} \right], \end{aligned} \tag{6.8}$$

where the first and second terms are the summation of auto-frequency components and the summation of cross-frequency components, respectively.

In [38], for a matched-phase coherent processor, it is proposed to use only the cross-frequency components. The processor proposed is defined according to the second term of (6.8):

$$B_M(\mathbf{x}) = \frac{1}{L(L-1)} \sum_{m\neq n}^{L} \mathbf{u}_m^H \mathbf{K}_{m,n} \mathbf{u}_n e^{j(\hat{\phi}_n - \hat{\phi}_m)}, \tag{6.9}$$

where the phase terms $\hat{\boldsymbol{\phi}} = \left\{\hat{\phi}_n\right\}_{n=1}^{L}$ are estimated by using (6.7). It has been shown in [38] that by using only the cross-frequency components, the processor $B_M(\mathbf{x})$ has better performance than the coherent broadband processor $B_C(\mathbf{x})$, especially when the ambient noise is significant.

In order to find the phase terms $\hat{\phi}$ in (6.7), the maximization is performed simultaneously with respect to both the phases ϕ's and the location search grid $\mathbf{x}$. It is only possible to directly search over the location grid and relative phases with sufficiently high resolution for a few frequencies ($L \leqslant 3$) [38]. For a larger number of processed frequencies, it was proposed in [38] to search the relative phases using the simulated annealing. It is well known for its ability for solving global optimization problems while having an extremely high computational complexity (The maximum number of processed frequencies considered in [38] was 5). For searching the matched phases, we propose to use a much more efficient phase search method, the phase descent search (PDS) algorithm [97].

In order to apply the phase search algorithm to the matched-phase processor, we find that it is useful to derive an alternative expression for the ambiguity function of the matched-phase processor (6.9), which can be rewritten as

$$\begin{aligned} B_M(\mathbf{x}) &= \frac{1}{L(L-1)} \left[\sum_{m,n=1}^{L} \mathbf{u}_m^H \mathbf{K}_{m,n} \mathbf{u}_n e^{j(\hat{\phi}_n - \hat{\phi}_m)} \right. \\ &\quad \left. - \sum_{m=1}^{L} \mathbf{u}_m^H \mathbf{K}_{m,m} \mathbf{u}_m \right]. \end{aligned} \tag{6.10}$$

We introduce a matrix $\mathbf{R}$ and a column vector $\hat{\mathbf{b}}$, whose elements are defined as

$$R_{m,n} = \mathbf{u}_m^H \mathbf{K}_{m,n} \mathbf{u}_n \tag{6.11}$$

and

$$\hat{b}_n = e^{j\hat{\phi}_n}, n = 1, \ldots, L, \tag{6.12}$$

respectively. Equation (6.10) can then be expressed as

$$B_M(\mathbf{x}) = \frac{1}{L(L-1)} \left\{ \hat{\mathbf{b}}^H \mathbf{R} \hat{\mathbf{b}} - tr\left[\mathbf{R}\right] \right\}. \tag{6.13}$$

The phase search problem in this matched-phase processor can then be interpreted as the problem of finding a vector $\hat{\mathbf{b}}$ by maximizing the quadratic function given by the first term of (6.13):

$$\hat{\mathbf{b}} = \arg \max_{[\mathbf{x}, |b_n|=1, n=1,\ldots,L]} \left\{ \mathbf{b}^H \mathbf{R} \mathbf{b} \right\}. \tag{6.14}$$

6.3.3 Cross-frequency incoherent processor

The cross-frequency incoherent processor proposed in [39] reduces the computational load at the cost of reducing the capability of suppressing sidelobes but still can obtain the same maximum output of B_M as the matched-phase coherent processor. Instead of searching for the matched phases over the location grid for achieving the maximum output of B_M in (6.9), it takes the modules of the quadratic terms across frequency, which results in

$$B_X(\mathbf{x}) = \frac{1}{L(L-1)} \sum_{m \neq n}^{L} \left| \mathbf{u}_m^H \mathbf{K}_{m,n} \mathbf{u}_n \right|. \tag{6.15}$$

6.4 Phase descent search algorithm

The PDS algorithm is based on coordinate descent iterations where coordinates are the unknown phases, and a constraint forcing the solution to have a unit magnitude. Elements of the solution vector $\mathbf{b}$ are given by

$$b_n = e^{j\phi_n},\ n = 1, \ldots, L,\ \phi_n \in [-\pi, \pi] \tag{6.16}$$

The coordinate descent iterations are applied to the phases ϕ_n and the PDS algorithm is derived by applying the dichotomous coordinate descent method [16] to the optimization problem (6.14) with elements b_n from (6.16).

We can describe the PDS algorithm as shown in Table 6.1 [97]. The algorithm starts with initialization of the solution vector $\mathbf{b} = \mathbf{b}_0$, a phase vector $\boldsymbol{\phi} = \boldsymbol{\phi}_0$, a residual vector $\mathbf{r} = -\mathbf{R}\mathbf{b}_0$ where elements of the matrix $\mathbf{R}$ are defined by (6.11), a step-size parameter $\beta = \beta_0$ where $\beta_0 \in [0, 2\pi]$, and an index $n = 0$ which denotes "successful" iterations. For each $m = 1, \ldots, M_b$, the step-size is reduced as $\beta \leftarrow \lambda\beta,\ 0 < \lambda < 1$ and a vector $\boldsymbol{\theta}$ is computed by $\boldsymbol{\theta} = \text{diag}\{\mathbf{R}\}[1 - \cos(\beta)]$. The parameter M_b indicates the number of reductions of the step-size β. For the pth element of the solution vector $\mathbf{b}$, where p is chosen in a circle order $p = 1, \ldots, L$, the element b_p might be updated as $b_{p,1} = e^{j\phi_{p,1}}$ where $\phi_{p,1} = \phi_p + \beta$, or $b_{p,2} = e^{j\phi_{p,2}}$ where $\phi_{p,2} = \phi_p - \beta$. Thus, we have $T_1 = \Re\{\Delta_1^* r_p\}$ where $\Delta_1 = b_{p,1} - b_p$, or $T_2 = \Re\{\Delta_2^* r_p\}$ where $\Delta_2 = b_{p,2} - b_p$, respectively. If one of the inequalities $\theta_p > T_1$ or $\theta_p > T_2$ is satisfied, the iteration is successful, and thus, the

Table 6.1: Phase Descent Search Algorithm

Step	Equation
Init.	$\mathbf{b} = \mathbf{b}_0$, $\boldsymbol{\phi} = \boldsymbol{\phi}_0$, $\mathbf{r} = -\mathbf{R}\mathbf{b}_0, \beta = \beta_0, n = 0$
1	for $m = 1 : M_b$
2	$\beta \leftarrow \lambda\beta$
3	$\boldsymbol{\theta} = \text{diag}\{\mathbf{R}\}[1 - \cos(\beta)]$
4	Flag $= 0$
5	for $p = 1 : L$
6	$\phi_{p,1} = \phi_p + \beta, b_{p,1} = e^{j\phi_{p,1}}$
7	$\Delta_1 = b_{p,1} - b_p, T_1 = \Re\{\Delta_1^* r_p\}$
8	if $\theta_p > T_1$
9	$n \leftarrow n + 1$, Flag $= 1$
10	$\mathbf{r} \leftarrow \mathbf{r} - \Delta_1 \mathbf{R}^{(p)}$
11	$\phi_p = \phi_{p,1}, b_p = b_{p,1}$
12	$\phi_{p,2} = \phi_p - \beta, b_{p,2} = e^{j\phi_{p,2}}$
13	$\Delta_2 = b_{p,2} - b_p, T_2 = \Re\{\Delta_2^* r_p\}$
14	if $\theta_p > T_2$
15	$n \leftarrow n + 1$, Flag $= 1$
16	$\mathbf{r} \leftarrow \mathbf{r} - \Delta_2 \mathbf{R}^{(p)}$
17	$\phi_p = \phi_{p,2}, b_p = b_{p,2}$
18	end the loop over p
19	if $n > N_u$ the algorithm stops
20	if Flag $= 1$ go to step 4
21	end the loop over m

index n is incremented, the phase ϕ_p, the element b_p and the residual vector $\mathbf{r}$ are updated as $\phi_p = \phi_{p,1}$, $b_p = b_{p,1}$ and $\mathbf{r} \leftarrow \mathbf{r} - \Delta_1 \mathbf{R}^{(p)}$ or $\phi_p = \phi_{p,2}$, $b_p = b_{p,2}$ and $\mathbf{r} \leftarrow \mathbf{r} - \Delta_2 \mathbf{R}^{(p)}$, respectively. Otherwise, they are not changed. The index n is compared with a predefined number of "successful" iterations N_u for stopping criterion. The choice of β_0, λ and M_b defines the final phase resolution $\beta_0 \lambda^{M_b}$; e.g., in the case of $\beta_0 = 2\pi$, $\lambda = 1/2$ and $M_b = 5$, the final phase resolution is $\pi/2^{M_b} = \pi/32$.

6.5 Frequency correction

For localization of a moving acoustic source, frequency correction is very important. Due to the movement of the source, the received signal suffers from the Doppler effect. The

frequencies received at the hydrophone array will be shifted and the shifts are usually unknown. We estimate these frequency shifts by using a frequency estimator based on the dichotomous search of the periodogram peak which provides the performance similar to that of the maximum likelihood estimator [98].

Since the source frequencies are transmitted simultaneously, the frequency shifts should be determined by the same compression factor η given by $\eta = \hat{f}/f_o$, where $\hat{f}$ is the received frequency and f_o is the transmitted frequency. Here, it is assumed that the compression factor is constant within a snapshot. We consider three different approaches for choosing the reference compression factor and use for MFP the one which gives the most reliable results. Fig. 6.1 shows the reference compression factors obtained from the data collected during the SWellEx-96 experiment by using these three approaches.

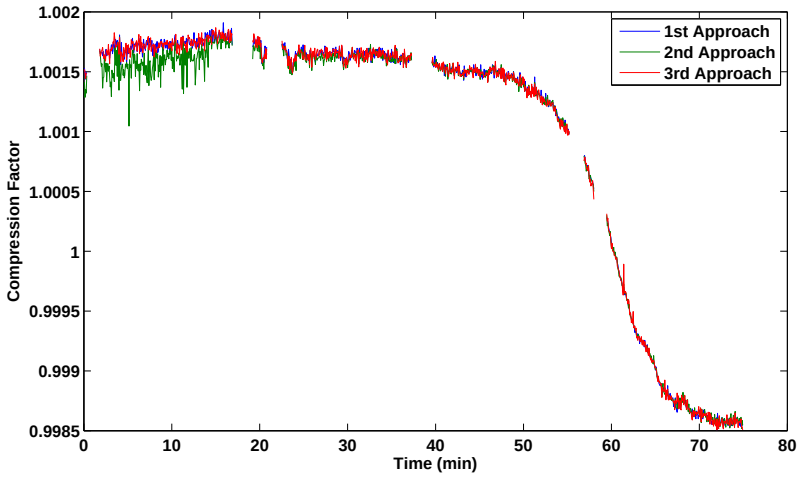


Figure 6.1: Compression factors η_{ref} obtained from the data collected during the SWellEx-96 experiment by using three different approaches.

In the first approach, the frequency shifts are estimated based on the periodogram averaged over the receiver hydrophones. As a result, we obtain a L-length vector of compression factors and a vector of corresponding signal-to-noise ratios (SNRs), and denote the compression factor and the SNR for the nth frequency as η_n and SNR_n, respectively. The compression factor η_{ref} corresponding to the frequency with the highest SNR is chosen for computation of all shifted frequencies: $\eta_{\mathrm{ref}} = \eta_{\hat{n}}$, where $\hat{n} = \arg\max_n \{\mathrm{SNR}_n\}$.

The SNR for each estimated frequency is computed by

$$\text{SNR}_n = \left[\frac{P(\hat{f})}{(1/4)\sum_{k=1}^{4} P(f_k)} - 1 \right], \tag{6.17}$$

where $P(\hat{f})$ is the signal power at the estimated frequency $\hat{f}$, and $P(f_k)$ is the power at the noise reference frequency $f_k = \hat{f} + \epsilon_k/T_s, \boldsymbol{\epsilon} = [-2, -1, 1, 2], k = 1, \ldots, 4$, T_s is the length of snapshot. The reason for using these frequencies as noise references is that they are the nearest frequencies to the estimated transmission frequencies without containing any signal information. The frequency step $1/T_s$ guarantees that frequencies f_k contain purely noise components that are not affected by the transmitted tone.

In the second approach, the frequencies are estimated based on the periodograms obtained from each receiver hydrophone. In such a case, for each snapshot, we obtain a $M \times L$ matrix of compression factors and a matrix of corresponding SNRs, and denote the compression factor and the SNR for the nth frequency at the mth hydrophone as $\eta_{m,n}$ and $\text{SNR}_{m,n}$, respectively. The reference compression factor is computed as $\eta_{\text{ref}} = \sum_{m=1}^{M}\sum_{n=1}^{L} \eta_{m,n}\, \text{SNR}_{m,n}/\text{SNR}_{sum}$, where $\text{SNR}_{sum} = \sum_{m=1}^{M}\sum_{n=1}^{L} \text{SNR}_{m,n}$.

The third approach is almost the same as the second approach, except that the reference compression factor is chosen as $\eta_{\text{ref}} = \eta_{\hat{m},\hat{n}}$, where $[\hat{m}, \hat{n}] = \arg\max_{[m,n]} \{\text{SNR}_{m,n}\}$.

According to Fig 6.1, the first and third approaches provide similar results with smaller fluctuations compared to the second approach. The first approach is computationally less expensive, and thus, is chosen to obtain the reference compression factors for our MFP analysis.

Fig. 6.2 shows SNR of the data collected at every 1-second snapshot for the transmission frequency 338 Hz during the experiment. We can see that, as the source is moving towards the receiver array (see Section 6.6.1), the SNR increases steadily from about 10 dB to 20 dB. As mentioned in [99], the source stopped transmitting the constant-wave (CW) tones at the beginning, midway point, and the end of the track. From Fig. 6.2, we can see the time periods when the source stopped transmission, which are the 2nd, 18th to 20th, 22nd to 23rd, 39th to 40th, 57th and 60th minutes of the data collected during the experiment.

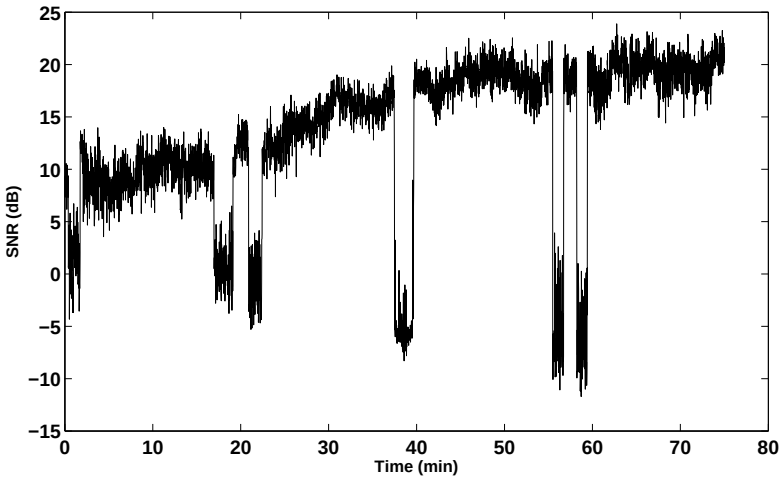


Figure 6.2: Signal-to-noise (SNR) ratio of the data collected at the frequency 338 Hz during the experiment.

6.6 Numerical results

In this section, we present results of application of the coherent matched-phase MF processor using the PDS algorithm to the data obtained in the SWellEx-96 Event S5 experiment. Brief description of the experiment, the source track used for the analysis and the data collection are firstly presented. Then, the coherent matched-phase MF processor using the PDS algorithm is applied to provide range-depth ambiguity surfaces and the estimated range trajectories. The results are compared with that obtained by applying the simulated annealing algorithm.

6.6.1 SWellEx-96 Event S5 experiment

The SWellEx-96 experiment was conducted in May 1996 ten kilometers off the coast of San Diego in California. Details of the experiment can be found in [99]. Fig. 6.3 shows a map of the source track during event S5 and the location of the receiver hydrophone array, a vertical line array (VLA) used for data collection. During the SWellEx-96 event S5 experiment, a shallow source at a supposed depth of 9 m and a deep source at a supposed depth of 54 m were towed along an isobath by a source ship [99]. During this event, the source ship started its track from the south of the array and proceeded northward at a

speed of about 2.5 m/s. Our analysis is based on the data collected on the VLA, which consisted of an array of 21 hydrophones with unequal depth spacing between 94.125 m and 212.25 m. The sampling rate on the VLA is 1500 Hz.

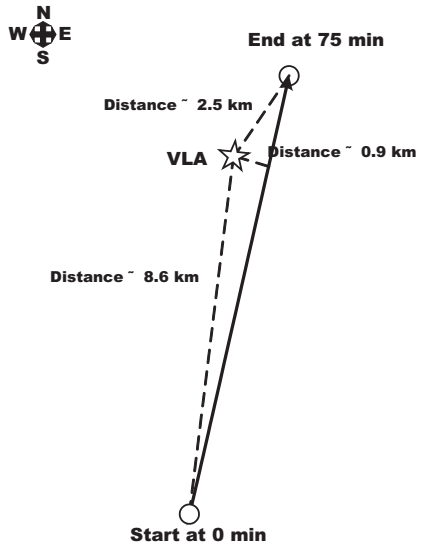


Figure 6.3: Map of the source track and the location of the vertical line array (VLA).

The shallow source transmitted a set of 9 tones which spanned frequencies between 109 Hz and 385 Hz. The frequencies of the set were at 109 Hz, 127 Hz, 145 Hz, 163 Hz, 198 Hz, 232 Hz, 280 Hz, 335 Hz and 385 Hz. The deep source transmitted a tonal pattern consisted of 5 sets of 13 tones each. Each set spanned frequencies between 49 Hz and 400 Hz. The first set of 13 tones which were projected at the maximum level were used in our MFP analysis. The frequencies of the set were at 49 Hz, 64 Hz, 79 Hz, 94 Hz, 112 Hz, 130 Hz, 148 Hz, 166 Hz, 201 Hz, 235 Hz, 283 Hz, 338 Hz and 388 Hz. Fig.6.4 shows the frequency spectrum between 45 Hz and 450 Hz, which is obtained from the experimental data.

A CTD (Conductivity, Temperature, and Depth) survey was conducted during the SWellEx-96 experiment to provide the water column sound speed data. A sound speed profile as recommended by [99] is used in our MFP analysis. This sound speed profile is plotted in Fig. 6.5. The seafloor is modeled by three layers [99]: the first layer is a 23.5 m thick sediment layer with an approximate density of 1.76 g/cm^3 and an compressional attenuation of about 0.2 dB/kmHz. The top and bottom of this sediment layer have compressional sound speeds of 1572.368 m/s and 1593.016 m/s, respectively; the second layer is an 800 m thick mudstone layer with an approximate density of 2.06 g/cm^3 and an

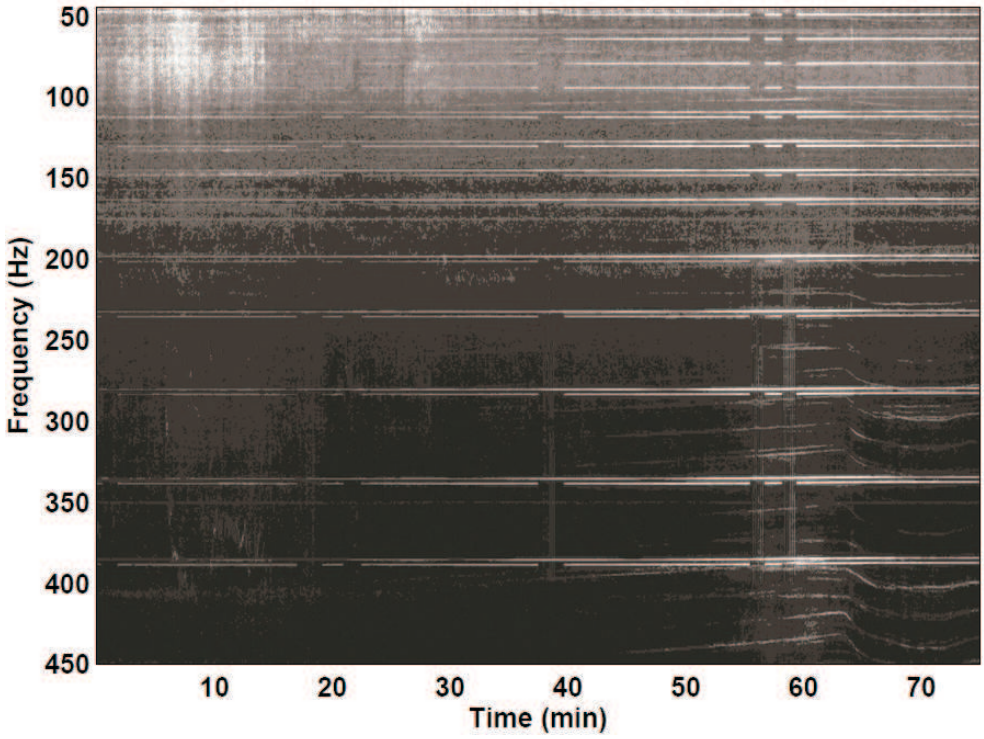


Figure 6.4: Frequency spectrum obtained during the SWellEx-96 experiment.

attenuation of about 0.06 dB/kmHz. The top and bottom sound speeds of the mudstone layer are 1881 m/s and 3245 m/s, respectively; the third layer is modeled as a halfspace with a density of 2.66 g/cm^3, an attenuation of 0.02 dB/kmHz, and a sound speed of 5200 m/s.

6.6.2 MFP analysis

In this analysis, the program KRAKEN [61] implementing the normal mode method was employed to compute the replicas with the resolution of 10 m in range and 1 m in depth. The three-layer seafloor model as described in Section 6.6.1 and the sound speed profile in Fig. 6.5 were used for computation of the acoustic field. The matched-phase coherent processor (6.13) was employed. The PDS algorithm as summarized in Table 6.1 with $\lambda = 1/2, M_b = 5$ was applied for searching the matched phases. The data were divided into snapshots and only one snapshot was used in the MFP.

In order to show the importance of frequency correction for locating the acoustic

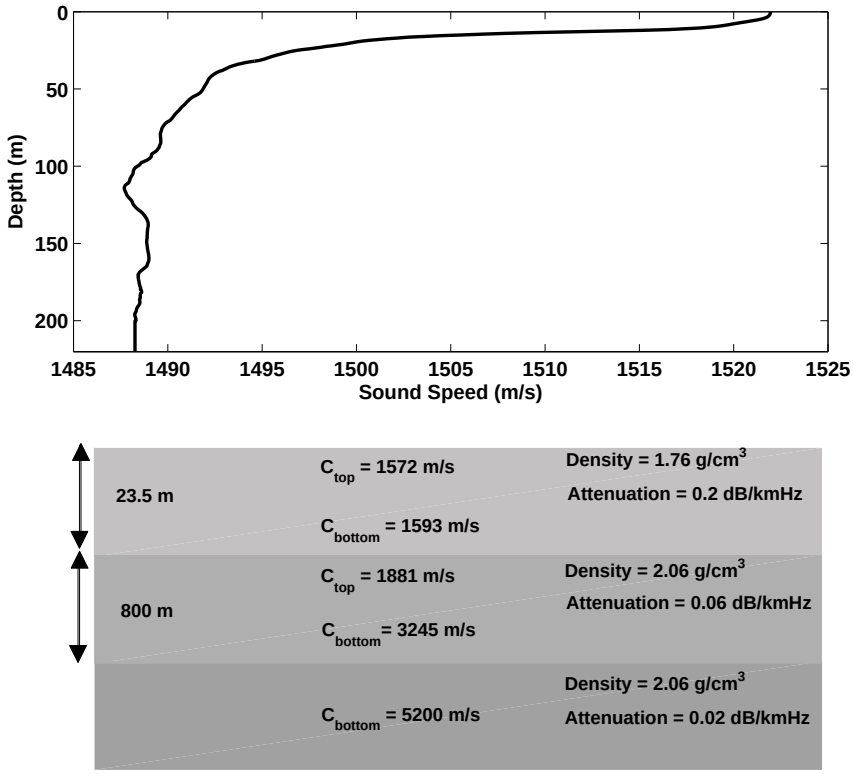


Figure 6.5: Sound speed as a function of depth generated from the local CTD cast during the SWellEx-96 experiment.

source, we applied the proposed MF processor to the experimental data with and without frequency correction. Fig. 6.6 shows the estimated range trajectories for the deep source by using the proposed MF processor with and without the frequency correction. The proposed MF processor was applied to the data collected in 4-second snapshots with 13 frequencies. With the 4-second snapshots, the frequency resolution is 0.25 Hz. With the ship speed of about 2.5 m/s, the maximum Doppler shifts are about 0.08 Hz for the lowest frequency (49 Hz) and about 0.64 Hz for the highest frequency (388 Hz). Without the frequency correction, the high frequencies only contribute noise, and thus, the MFP fails to locate the source at the beginning of the experiment, where SNR is low. Also, from Fig. 6.6, we see that, with frequency correction, the proposed MF processor always provides accurate localization even at the beginning of the experiment. In the remainder of this section, all simulation results were obtained with the frequency correction.

We also implemented the adaptive simplex simulated annealing (ASSA) proposed

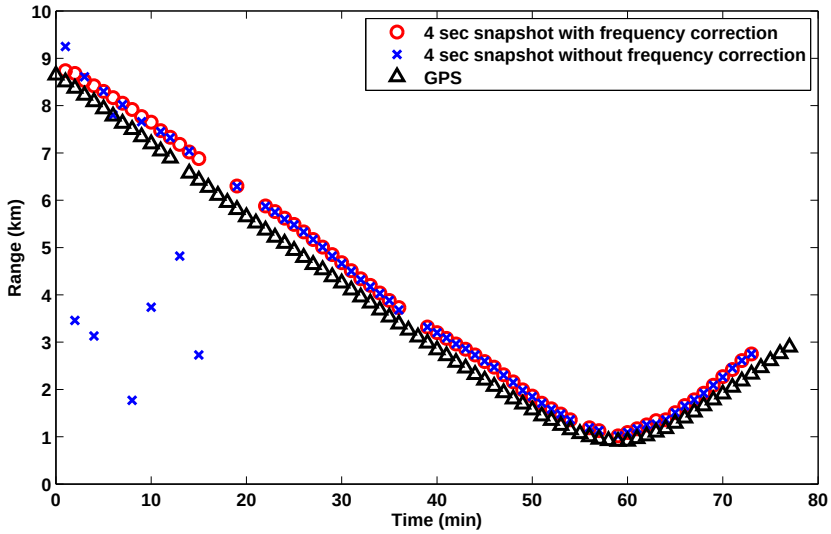


Figure 6.6: Range trajectory estimated by the MFP-PDS processor using 4-second snapshots and 13 frequencies with and without the frequency correction.

in [41] to the matched-phase processor for computing the matched phases, and compared its performance and complexity with that of the MFP-PDS processor. Fig. 6.7 and Fig. 6.8 show the ambiguity surfaces obtained by these two processors for 5 and 13 processed frequencies, respectively. We can see that the matched-phase processor with the PDS algorithm provides similar ambiguity surfaces as the matched-phase processor with ASSA: the peak to sidelobe ratios read from Fig. 6.7 for the processors are 3.16 dB and 3.17 dB, respectively. The peak to sidelobe ratios read from Fig. 6.8 are 6.59 dB and 6.55 dB, respectively. The matched phases obtained by using the two algorithms are listed in Table 6.2. We see that the phases obtained by the PDS algorithm are very close to those obtained by the ASSA.

We compared the computational complexity of the PDS and ASSA algorithms by counting how many times the quadratic form $\mathbf{b}^H\mathbf{R}\mathbf{b}$ was computed. For each point in the location search grid, the quadratic form is computed once in each iteration of the algorithms, and this is the most computationally consuming part of the algorithms. These counts were averaged over the number of positions in the location grid. When processing 5 frequencies, the count for the ASSA algorithm is approximately 12 times of that of the PDS algorithm; specifically, they are 1399 and 116, respectively. When processing 13 frequencies, the ratio is higher; the PDS algorithm computed the quadratic form 356 times, whereas the ASSA algorithm required 16554 computations, i.e., the ASSA com-

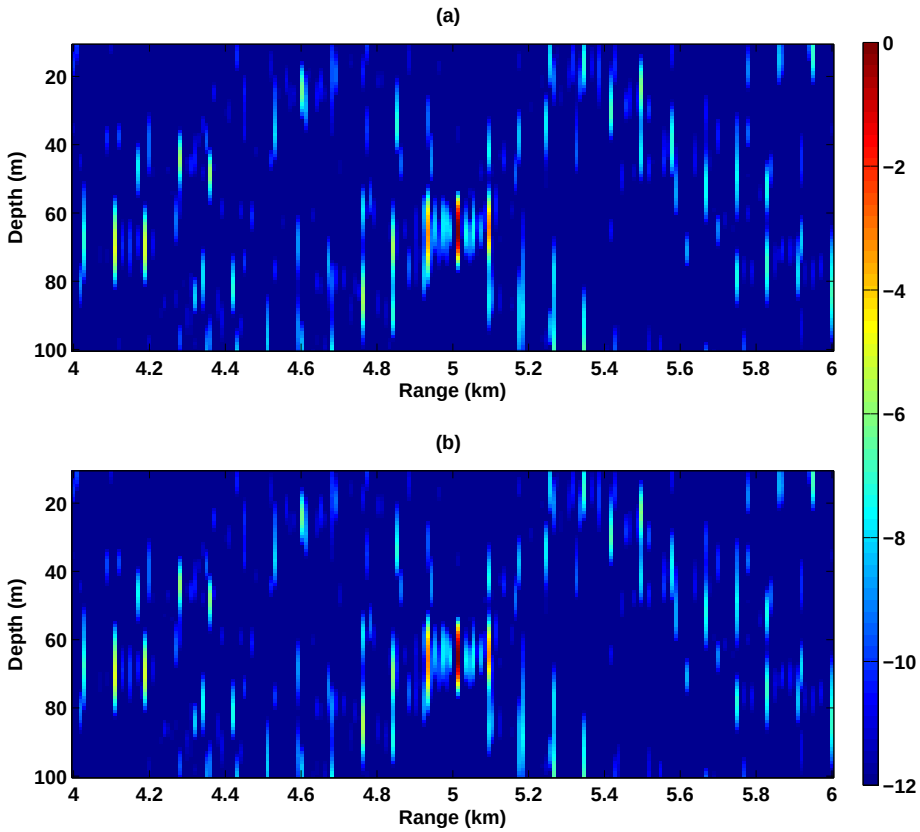


Figure 6.7: Range-depth ambiguity surfaces computed by using (a) matched-phase coherent processor with PDS; (b) matched-phase coherent processor with ASSA. 5 frequencies (Hz): 112 130 148 166 201 were processed in both processors.

plexity was about 46 times the PDS complexity. The difference is further increased as the number of processed frequencies increases.

Fig. 6.9(a)-(c) show the range-depth ambiguity surfaces obtained by using the matched-phase coherent processor with the PDS algorithm for different numbers of processed frequencies. For Fig. 6.9(a), the middle 5 frequencies at 112 Hz, 130 Hz, 148 Hz, 166 Hz and 201 Hz as used in [38] were processed. For Fig. 6.9(b), the 9 frequencies which had the highest SNR were processed. For Fig. 6.9(c), all the frequencies in the first set of tones were used. 1-second snapshot starting at the 9th minute of the experiment data was processed. We can see that, as the number of processed frequencies increases, the performance of the matched-phase coherent processor with PDS algorithm is improved. The peak to sidelobe ratios read from Fig 6.9 (a), (b) and (c) are about 1.6 dB, 5.5 dB and 6.3 dB, respectively.

Table 6.2: Phase shifts obtained by using PDS and ASSA algorithms.

	Phase shifts (in degrees) with respect to the phase at frequency 112 Hz obtained for 5 processed frequencies
PDS	$0, -90, 135, 45, -56$
ASSA	$0, -89, 140, 49, -48$

	Phase shifts (in degrees) with respect to the phase at frequency 49 Hz obtained for 13 processed frequencies
PDS	$0, 112, -124, 33, -56, -146, 78, -11, -112, -112, 90, 101, 157$
ASSA	$0, 108, -129, 29, -63, -156, 74, -16, -116, -118, 83, 97, 147$

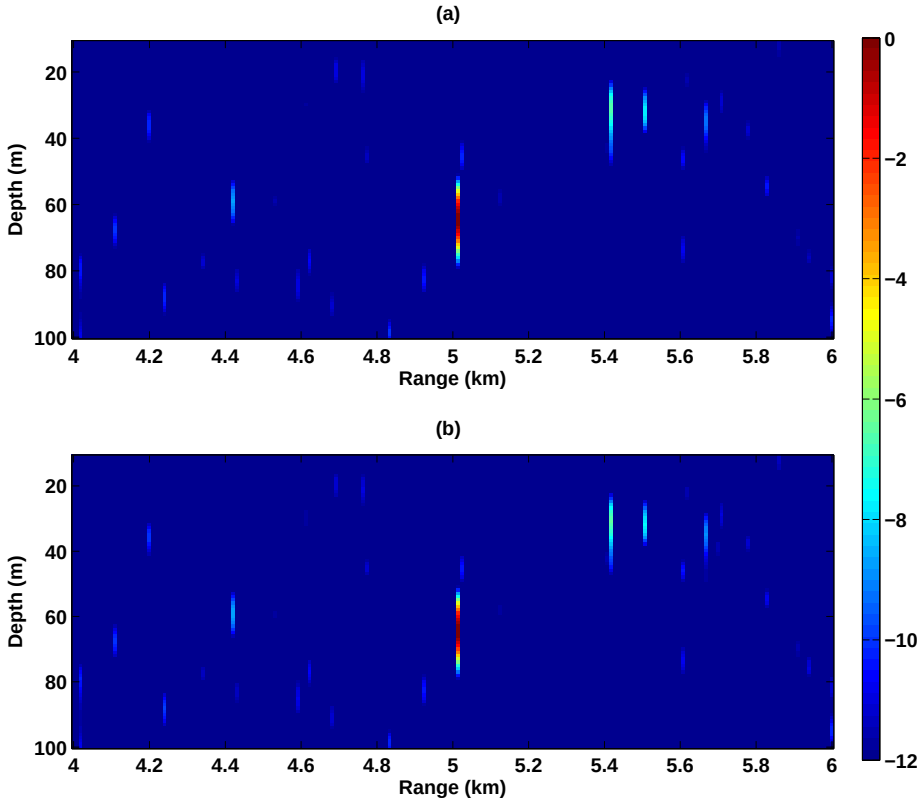


Figure 6.8: Range-depth ambiguity surfaces computed by using (a) matched-phase coherent processor with PDS; (b) matched-phase coherent processor with ASSA. 13 frequencies (Hz): 49 64 79 94 112 130 148 166 201 235 283 338 388 were processed in both processors.

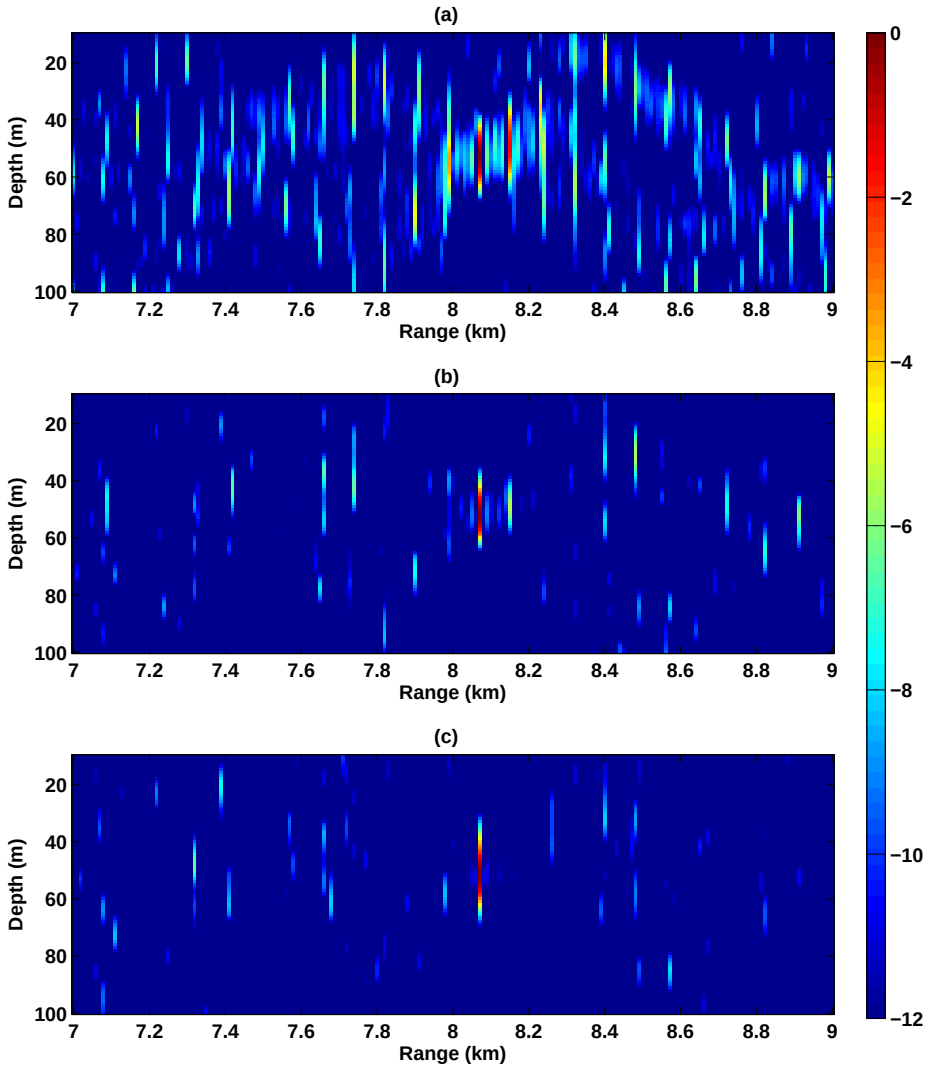


Figure 6.9: Range-depth ambiguity surfaces computed by using matched-phase coherent processors with PDS algorithm, for different numbers of processed frequencies: (a) 5 frequencies (Hz): 112 130 148 166 201; (b) 9 frequencies (Hz): 112 130 148 166 201 235 283 338 388; (c) 13 frequencies (Hz): 49 64 79 94 112 130 148 166 201 235 283 338 388.

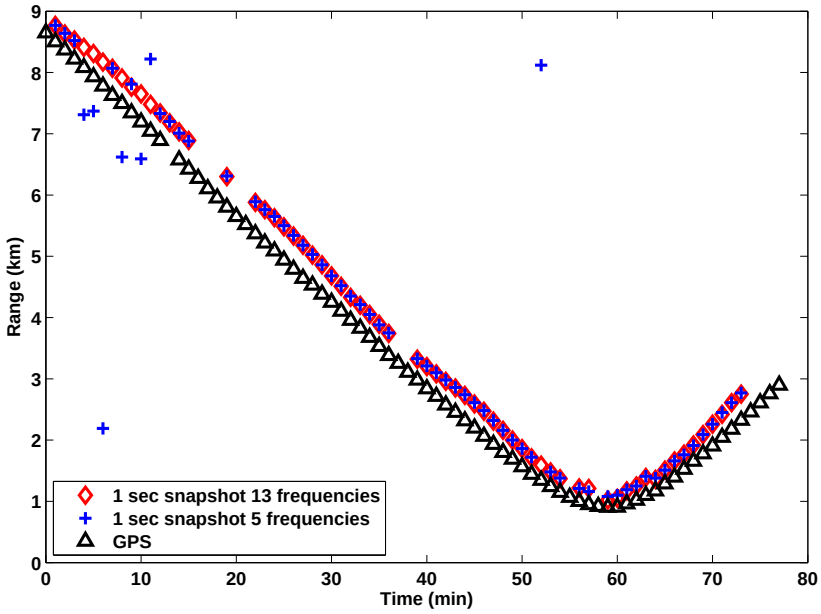


Figure 6.10: Range trajectory obtained by the MFP-PDS processor with 5 and 13 processed frequencies.

Fig.6.10 shows the range trajectory obtained by the MFP-PDS processor with 5 and 13 processed frequencies. We can see how increase in the number of the processed frequencies improves the performance of the processor, thus justifying the need to have a computationally efficient algorithm for the phase search.

Fig. 6.11(a) and 6.11(b) show ambiguity surfaces obtained by the MFP-PDS processor and the cross-frequency incoherent processor [39], respectively. For both the processors, 13 frequencies were used and one 1-second snapshot starting at the 4th minute of the experiment. It is seen that the proposed processor provides the same peak level as the cross-frequency incoherent processor, which has been shown [39] to have the same maximum of the ambiguity surface as the matched-phase coherent processor using the simulated annealing method. It is also seen that, the cross-frequency incoherent processor gives a much wider peak in range and much higher sidelobes. The peak to sidelobe ratios read from Fig 6.11(a) and (6.11(b) are about 6.1 dB and 0.3 dB, respectively.

Fig. 6.12 shows the range trajectory generated from the GPS data recorded during the experiment [99] and the estimated range trajectories for the deep source by applying the matched-phase coherent processor with the PDS algorithm to the snapshots of different

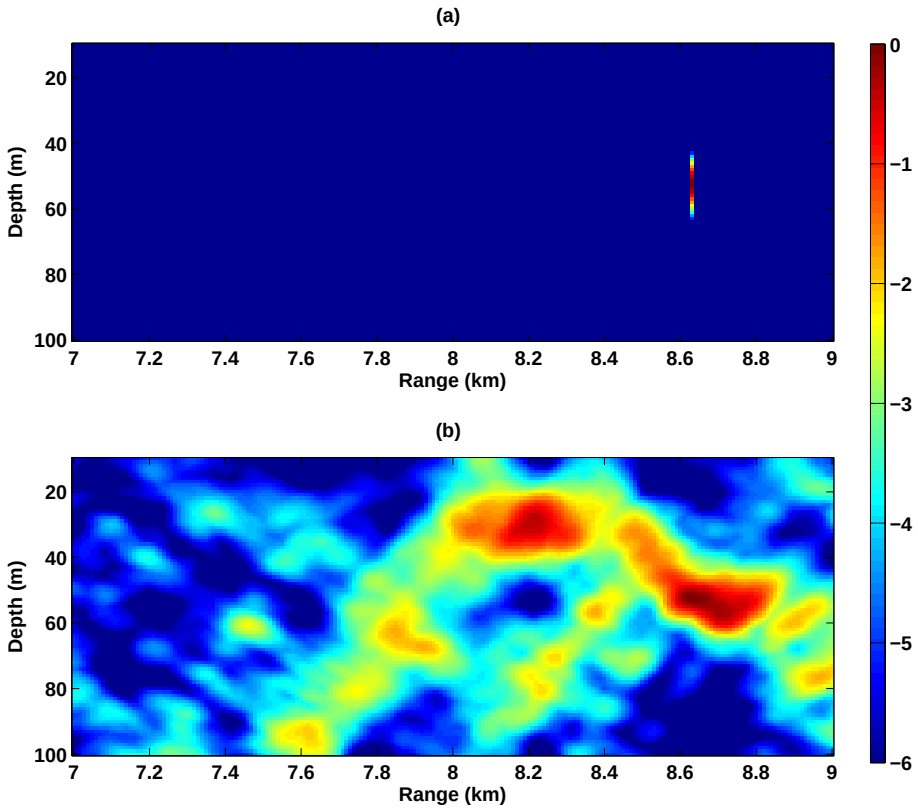


Figure 6.11: Range-depth ambiguity surfaces computed by using (a) matched-phase coherent processor with PDS algorithm $B_M(\mathbf{x})$; (b) cross-frequency incoherent processor $B_X(\mathbf{x})$.

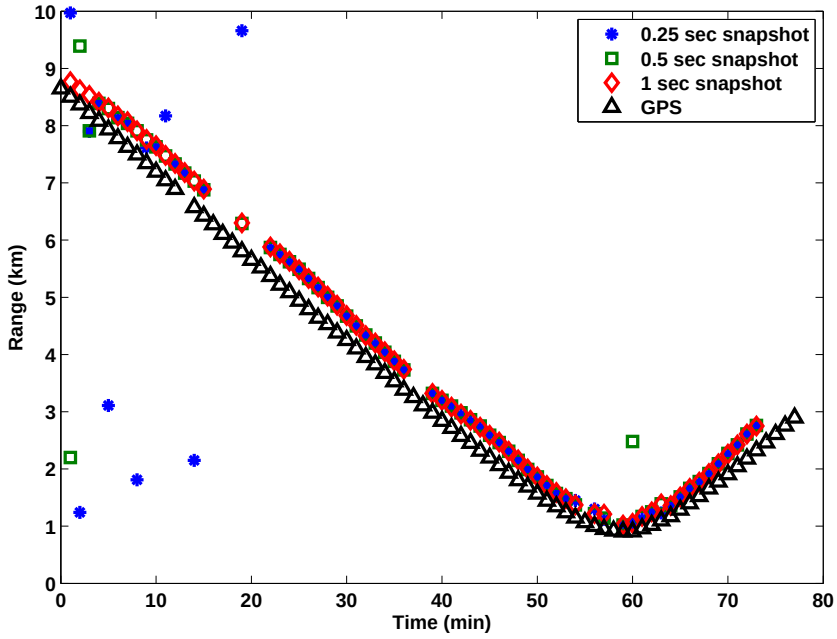


Figure 6.12: Range trajectory obtained from GPS measurements and the estimated range trajectories for the deep source by applying the matched-phase coherent processor with PDS algorithm to the data collected in different snapshot length with 13 frequencies.

length. All the frequencies in the first set of tones were used for estimating the source trajectory. Due to the uncertainty about the correspondence of the starting time of the GPS measurements to the experiment data, the estimated range trajectories were shifted 1 minute forwards to better match the shape of the GPS measurements (the 2nd minute of the experiment data corresponds to the 1st minute of the GPS measurements). Estimates of the range trajectories for those periods when the source stopped transmitting CW tones were removed. From Fig. 6.12, we see that with 0.25-second snapshots, the proposed matched-phase processor failed to locate the source in the first 20 minutes of the experiment and provided accurate localization afterward. It is seen that as with a longer snapshot, the proposed matched-phase processor can precisely locate the source at more positions. We observe that the estimated trajectory obtained by applying the matched-phase processor to 1-second snapshots is well matched to the GPS measurements. This is because the SNR for the data collected in 0.25-second snapshot was much lower, especially when the source was far away from the receiver array. However, shifts of around $50 - 400$ meters between the estimated trajectory and the GPS measurements are also observed. These shifts were probably caused by the mismatch in the bathymetry assumptions which were used for the calculations of the replicas, as reported in [100, 101]. These

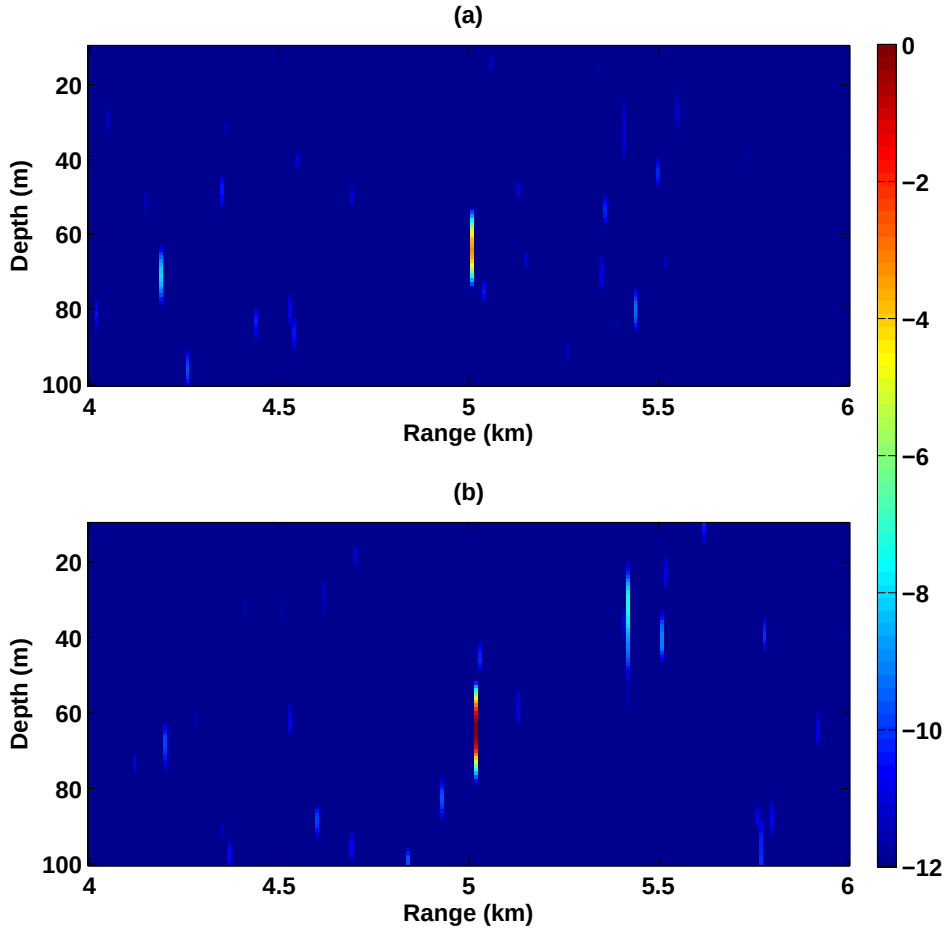


Figure 6.13: Range-depth ambiguity surfaces computed by applying the matched-phase coherent processor with PDS algorithm to the data collected in different snapshot length with 13 frequencies: (a) 0.25-second snapshot; (b) 1-second snapshot.

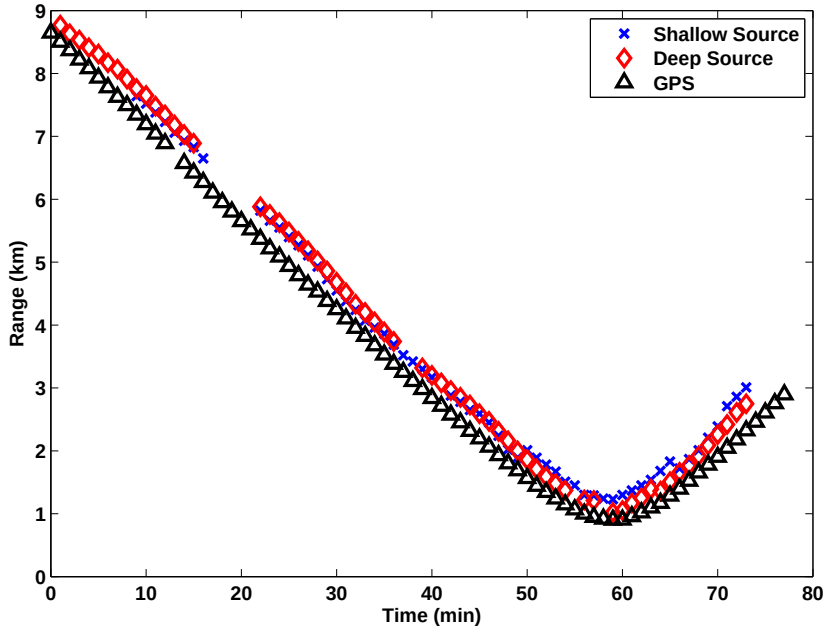


Figure 6.14: Range trajectories for the shallow source and deep source obtained by the MFP-PDS processor.

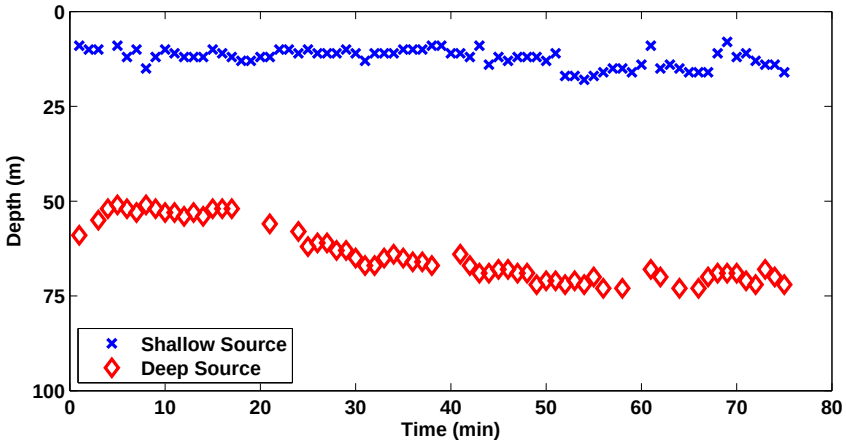


Figure 6.15: Depth trajectories for the shallow source and deep source obtained by the MFP-PDS processor.

shifts are found to be very close to those shown in the comparison of ranges estimated by MFP with GPS measurements in [60, 102].

Fig.6.13 shows the range-depth ambiguity surfaces obtained by applying the matched-phase coherent processor with the PDS algorithm to the data collected in different snapshot length with 13 frequencies. It is seen that with both 0.25-second and 1-second snapshots, the proposed matched-phase processor precisely locate the source. However, with 0.25-second snapshot, the peak value of the ambiguity surface which indicates the source position, is about 3.3 dB lower than that of ambiguity surface obtained with 1-second snapshot. The peak to sidelobe ratios read from Fig 6.13 (a) and (b) are about 7.3 dB and 4.8 dB, respectively.

Finally, Fig. 6.14 and Fig. 6.15 show the range trajectories and the depth trajectories for both the shallow source and deep source, respectively, obtained by the MFP-PDS processor. For locating the shallow source, 1-second snapshots with all the frequencies were used. For locating the deep source, 1-second snapshots with all the frequencies in the first set of tones were used. From Fig. 6.14, it is seen that both the estimated range trajectories of the shallow source and deep source are close to the GPS measurements. From Fig. 6.15, it is seen that the shallow source fluctuated in depth between 8 m and 18 m, and the deep source fluctuated in depth between 50 m and 75 m.

6.7 Conclusions

In this chapter, we have reviewed the matched-phase coherent matched-field processor and introduced the phase descent search (PDS) algorithm to the matched-phase coherent processor for searching the matched phases. The PDS algorithm is based on coordinate descent iterations with respect to the unknown phases and constrains the solution to have a unit magnitude. When compared with simulated annealing algorithm, it has significantly lower complexity, which enables simultaneous processing of many frequencies, and thus, improves processor performance.

The proposed processor has been applied to experimental data for source localization. It has been shown that, by using the proposed PDS algorithm, the matched-phase coherent processor can process more frequencies, and thus, gives better performance in reinforcing the main peak at the source location while reducing the sidelobes. The estimated range trajectory obtained by applying the processor to the data collected in every 1-second snapshot is well matched to GPS measurements.

Chapter 7

Conclusions and Further Work

Contents

This thesis investigated the low-complexity channel estimation, CE based equalization and source localization techniques using DCD algorithm for underwater acoustic communications. We have firstly derived an approach for convergence analysis of the RLS-DCD adaptive filtering algorithm based on computations with only deterministic quantities (Chapter 2). We have then proposed a low-complexity CE based adaptive LE (Chapter 3), in which the computation of equalizer coefficients is multiplication-free and division-free when using the DCD iterations for both channel estimation and equalization. We have presented the complex-valued DCD iterations and the complex-valued RLS-DCD adaptive filtering algorithm for channel estimation (Chapter 4), and derived two partial-update CE based adaptive DFEs (Chapter 4 and 5), both of which operate together with partial-update channel estimators, such as RLS-DCD channel estimator, and exploit complex-valued DCD iterations to efficiently compute the DFE coefficients. Finally, we have investigated the application of MFP for underwater acoustic source localization and introduced the PDS algorithm to the matched-phase coherent broadband MF processor for searching the matched phases (Chapter 6).

7.1 Conclusions

Chapter 1 has stated the motivations, objectives and contributions of the whole work, and briefly introduced fundamental techniques including the DCD algorithm, time-varying channel models and time-varying underwater acoustic channel model, which are used throughout this thesis.

Chapter 2 has presented a new approach for convergence analysis of the RLS-DCD adaptive filtering algorithm. The proposed approach is based on computations with only deterministic quantities obtained from the second order statistics. Deterministic expressions for time dependent correlation quantities have been obtained without involving any stochastic processes and used to form the normal equations. We have derived deterministic equations for predicting the MSE and MSD learning curves of the RLS-DCD algorithm. Simulation results have shown good agreement between the predictions and practical learning curves, although the predictions are somewhat optimistic.

In Chapter 3, we have proposed a channel-estimate (CE) based adaptive linear equalizer (LE) with a complexity as low as $\mathcal{O}(N_u(K+M))$ operations per sample, where K and M are the equalizer and channel estimator length, respectively, and N_u is the number of iterations such that $N_u \ll K$ and $N_u \ll M$. The proposed technique exploits coordinate descent iterations for computing the equalizer coefficients. Moreover, we have shown that when using the DCD iterations in both the channel estimation and equalization, computation of the equalizer coefficients is multiplication-free and division-free, which makes it attractive for hardware implementation. We have compared the performance of the proposed LE with that of the MMSE LE with perfect knowledge of the channel and known LEs over time-varying multipath channel. Simulation results have shown that, with only a few updates N_u per sample, the proposed LE outperforms the RLS directly adaptive (DA) LE, and performs very close to the RLS CE based adaptive LE and close to the MMSE LE.

In Chapter 4, we have introduced complex-valued DCD iterations and the complex-valued RLS-DCD channel estimator, and proposed a partial-update CE based adaptive DFE with a complexity as low as $\mathcal{O}(N_u(l+1)\log_2 2(l+1))$ real multiplications per sample, where l is the equalizer delay and N_u is the number of iterations such that $N_u \ll l$.

The proposed technique operates with partial-update channel estimators, such as the RLS-DCD channel estimator. It is assumed that every CE update involves only one channel coefficient and every update of the equalizer involves only one equalizer coefficient. The proposed DFE has been implemented in both the conventional and modified DFE structures. For the conventional structure, we have also proposed a simple recursive method for computing the FBF coefficients, whereas the modified structure does not require computing the FBF. We have compared the performance of the proposed DFE with that of the MMSE DFE with perfect knowledge of the channel and known DFEs over time-varying multipath channels. Simulation results have show that, even with a small number of updates N_u, the proposed DFE significantly outperforms the DA DFE, and performs very close to the RLS CE based DFE and close to the MMSE DFE. It is found that the proposed DFE also involves $\mathcal{O}(N_u(l^2 + l\ \log_2 2l))$ real additions per sample, which is still computationally consuming for channels with large delay spreads, such as the underwater acoustic channel.

In Chapter 5, we have derived another approach for recursive computation of CE based DFE coefficients with even lower complexity. The proposed DFE operates with partial-update channel estimators, such as RLS-DCD channel estimator, and exploits DCD iterations. It is assumed that every CE update involves only one channel coefficient, and every update of the equalizer involves one coefficient of the FFF and one coefficient of the FBF. The complexity of the proposed DFE is upper bounded by a value of $\mathcal{O}(N_u K) + \mathcal{O}(N_u B) + \mathcal{O}(N_u M)$ operations per sample, where K is the FFF length, B the FBF length, M the channel estimator length, and N_u the number of updates such that $N_u \ll M$. We have shown that when using a channel estimator which also exploits the DCD iterations, such as in the RLS-DCD adaptive filter, all multiplications involved in computation of the equalizer coefficients can be replaced by bit-shift operations, which makes the equalizer attractive for hardware design. We have applied the proposed DFE and known DFEs to two time-varying Rayleigh fading channel models and a time-varying underwater acoustic channel model. Simulation results have shown that with $N_u \ll M$, the proposed DFE provides the BER performance similar to that of the RLS CE DFE, and outperforms the RLS DA DFE.

Chapter 6 has investigated the application of the matched-phase coherent MF processor for underwater acoustic source localization. We have introduced the PDS algorithm

to the matched-phase coherent MF processor for searching the matched phases. The PDS algorithm is based on coordinate descent iterations with respect to the unknown phases and constrains the solution to have a unit magnitude. We have shown that when compared with simulated annealing algorithm, the PDS algorithm has significantly lower complexity, which enables simultaneous processing of many frequencies, and thus, improves processor performance. The proposed processor has been applied to experimental data for source localization. Simulation results have shown that, by using the proposed PDS algorithm, the matched-phase coherent processor can process more frequencies, and thus, gives better performance. We also shown that the estimated range trajectory obtained by applying the processor to experimental data is well matched to GPS measurements.

7.2 Further Work

Based on this research, we have concluded the following suggestions for further work:

In this thesis, we have proposed a novel approach for convergence analysis of the RLS-DCD adaptive filtering algorithms based on computations with only deterministic quantities. This approach can also be used for other adaptive filtering algorithms based on iteratively solving the normal equations with one or more iterations at a time instant [76, 103–105]. Its applications to other adaptive filtering algorithms are worth to be investigated.

The low-complexity LE and DFEs we have derived are symbol-spaced equalizers. It is known that fractionally-spaced equalizers can outperform symbol-spaced equalizers due to the fact that the actual bandwidth of the signal is somewhat larger than expected and the sampling rate of the input signal should be increased in order to satisfy the Nyquist theorem [106]. However, fractionally-spaced equalizers require even more computation. It is therefore interesting to extend our proposed low complexity approaches to fractionally-spaced CE based equalizers.

Multiple-input-multiple-output (MIMO) systems have recently drawn extensive research interest in underwater acoustic communications to increase the transmission data rate over the acoustic channel, which is bandwidth-limited. Although many equal-

ization techniques have been introduced for MIMO underwater acoustic communications [107–110], these techniques have not been considered in the computational complexity point of view. They might therefore be impractical and difficult for real-time implementation. Extension of our proposed low-complexity approaches to MIMO systems can be promising.

The maximum likelihood sequence estimation (MLSE) equalizer [111] is an optimal equalization technique, which finds the minimum over all the possible data sequences of the log-likelihood function. The Viterbi algorithm [111] can be used to exactly solve this minimization problem. However, it involves a computational complexity which grows exponentially with an increase of channel length. It is of great interest to develop a low-complexity MLSE-like equalizer by introducing a DCD based algorithm for solving such a minimization problem.

DFEs are known to suffer from error propagation due to the feedback of error decisions. Error correction coding techniques can help address this issue and ensure low BER performance. Many joint equalization and decoding techniques [112–115] have been proposed to improve the performance of the MMSE DFEs. However, these techniques may have some difficulty with sparse channels [116], and may be too complex for hardware implementation. Hence, it is interesting to investigate and develop a low-complexity joint equalization and decoding approach by applying a DCD based algorithm.

Although the underwater acoustic channel exhibits large delay spreads, it is typically sparse. There has been increasing research interest in the application of the sparse channel estimation techniques for underwater acoustic communications [12–15]. The sparse channel estimators are known to be simple for implementation and only require a very short training sequence. It is interesting to compare these techniques with the RLS-DCD channel estimator, which also takes into account the sparse nature of the channel. In [12], it has been shown that by exploiting the natural sparseness of the underwater acoustic channel, it is possible to ignore the small equalizer taps and obtain sparse equalization, at the cost of slightly worse performance. Extension of our proposed approaches to the sparse equalization techniques may further reduce the computational complexity, which makes it even more attractive for practical implementation.

The matched-phase coherent broadband MF processor using PDS algorithm we proposed in this thesis has been shown to provide accurate localization of a moving source transmitting broadband signal, by using only one short snapshot. However, it cannot be applied to locate multiple sources directly, since only the strongest source will be reinforced as the main peak and all the other weaker sources will be depressed as sidelobes. In underwater sensor networks [8], accurate and efficient localization of multiple sources is highly desirable. It is interesting and worth to investigate how the matched-phase approach can be applied to the multi-source localization problem to achieve high resolution localization of sources with a small number of snapshots.

Appendix A

Computation of $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$: Approach 1

According to the definition of $\bar{\mathbf{F}}(i-1)$ and the structure of $\mathbf{F}(i-1)$ as shown in Fig.4.6(b), $\bar{\mathbf{F}}(i-1)$ is a block of the channel convolution matrix. Therefore, $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$ can be computed by convolving the FFF taps $\hat{\mathbf{f}}_{1:l+1}(i-1)$ and a vector $\hat{\mathbf{u}}(i-1)$ with elements given by

$$\hat{u}_m(i-1) = \hat{h}_{M-m+1}(i-1), \;\; m = 1, \ldots, M. \tag{A.1}$$

As a result, a $2(l+1) \times 1$ vector $\boldsymbol{\phi}(i-1)$ can be obtained, and we have

$$\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1) = \boldsymbol{\phi}_{p(i):l+1}(i-1). \tag{A.2}$$

By applying the fast fourier transforms (FFT) [85], $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$ can be computed as shown in Table A.1, where $\mathcal{F}\{\cdot\}$ and $\mathcal{F}^{-1}\{\cdot\}$ denote FFT and inverse FFT operations, respectively, and $\odot$ denotes point-by-point matrix multiplication.

Table A.1: Low-complexity computation of $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$

Step	Equation	$\times$	+
1	Extend $\hat{\mathbf{h}}(i-1)$ with zeros to length $2(l+1)$ and compute $\mathcal{F}\{\hat{\mathbf{h}}(i-1)\}$	$2(l+1)\log_2 2(l+1)$	$2(l+1)\log_2 2(l+1)$
2 2	Extend $\hat{\mathbf{f}}_{1:l+1}(i-1)$ with zeros to length $2(l+1)$ and compute $\mathcal{F}\{\hat{\mathbf{f}}_{1:l+1}(i-1)\}$	$2(l+1)\log_2 2(l+1)$	$2(l+1)\log_2 2(l+1)$
3	$\mathbf{P}(i-1)=\mathcal{F}\{\hat{\mathbf{h}}^*(i-1)\}\odot\mathcal{F}\{\hat{\mathbf{f}}_{1:l+1}(i-1)\}$	$2(l+1)$	–
4	$\boldsymbol{\phi}(i-1)=\mathcal{F}^{-1}\{\mathbf{P}(i-1)\}$	$2(l+1)\log_2 2(l+1)$	$2(l+1)\log_2 2(l+1)$
5	$\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)=\boldsymbol{\phi}_{p(i):l+1}(i-1)$	–	–
	Total for each iteration i: $2(l+1)(1+3\log_2 2(l+1))$ complex mult. and $6(l+1)\log_2 2(l+1)$ complex adds		

Appendix B

Computation of $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$: Approach 2

According to assumption 5 given in Section 4.3, convolution of the FFF taps $\hat{\mathbf{f}}(i-1)$ and the vector $\hat{\mathbf{u}}(i-1)$ can be computed recursively with a complexity $\mathcal{O}(K+M)$, and thus, $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$ can also be obtained by recursive computation. Derivation of such recursion is given below.

According to the definition of $\hat{\mathbf{u}}(i-1)$ in (A.1) and assumption 4 given in Section 4.3, we have

$$\hat{\mathbf{u}}(i-1) = \hat{\mathbf{u}}(i-2) + \Delta\hat{\mathbf{u}}(i-1),$$

where all the elements of $\Delta\hat{\mathbf{u}}(i-1)$ are zeros, except

$$\Delta\hat{u}_{M-p(i-1)+1}(i-1) = \Delta\hat{h}(i-1). \tag{B.1}$$

We denote the convolution of the FFF taps $\hat{\mathbf{f}}(i-1)$ and the vector $\hat{\mathbf{u}}(i-1)$ as

$$\boldsymbol{\varphi}(i-1) = \hat{\mathbf{u}}(i-1) * \hat{\mathbf{f}}(i-1), \tag{B.2}$$

and using the recursive expressions for $\hat{\mathbf{u}}(i-1)$ and $\hat{\mathbf{f}}(i-1)$, we obtain

$$\begin{aligned}\boldsymbol{\varphi}(i-1) &= [\hat{\mathbf{u}}(i-2) + \Delta\hat{\mathbf{u}}(i-1)] * \left[\hat{\mathbf{f}}(i-2) + \Delta\hat{\mathbf{f}}(i-1)\right] \\ &= \boldsymbol{\varphi}(i-2) + \hat{\mathbf{u}}(i-2) * \Delta\hat{\mathbf{f}}(i-1) \\ &\quad + \Delta\hat{\mathbf{u}}(i-1) * \hat{\mathbf{f}}(i-1).\end{aligned} \tag{B.3}$$

According to (B.1) and assumption 5, (B.3) can be rewritten as

$$\begin{aligned}\boldsymbol{\varphi}(i-1) = \boldsymbol{\varphi}(i-2) &+ \Delta\hat{f}(i-1)\hat{\mathbf{h}}^{[q(i-1)]}(i-2) \\ &+ \Delta\hat{h}(i-1)\hat{\mathbf{f}}^{[p(i-1)]}(i-1),\end{aligned} \tag{B.4}$$

where $\boldsymbol{\varphi}(0) = \mathbf{0}_{2K\times 1}$, $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$ is a $2K \times 1$ vector obtained by shifting elements of $\hat{\mathbf{h}}(i-2)$ by $q(i-1)$ positions down, and other elements of $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$ are zeros. Definition for $\hat{\mathbf{f}}^{[p(i-1)]}(i-2)$ is similar to that of $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$. Finally, $\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1)$ can be obtained from $\boldsymbol{\varphi}(i-1)$ as

$$\bar{\mathbf{F}}(i-1)\hat{\mathbf{f}}_{1:l+1}(i-1) = \boldsymbol{\varphi}_{p(i):l+1}(i-1), \tag{B.5}$$

where $\boldsymbol{\varphi}(i-1)$ is computed using the recursion in (B.4). For each iteration i, this approach requires only $2(K+M)$ real multiplications and $2(K+M)$ real additions. Moreover, as we propose to use the DCD iteration in both channel estimation and the computation of the FFF taps, $\Delta\hat{h}(i)$ and $\Delta\hat{f}(i)$ are power-of-two numbers for every i. Therefore, all the multiplications required in this approach can be replaced by bit-shift operations.

Appendix C

Computation of $\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1)$

The derivation below is a straightforward extension of the real-valued case in Chapter 3 to the complex-valued case.

Let $\mathbf{H}(i) = \mathbf{H}(i-1) + \mathbf{\Delta}(i)$, then we have $\mathbf{G}(i) = \mathbf{G}(i-1) + \mathbf{\Delta}^H(i)\mathbf{H}(i-1) + \mathbf{H}^H(i-1)\mathbf{\Delta}(i) + \mathbf{\Delta}^H(i)\mathbf{\Delta}(i)$ and thus,

$$\begin{aligned}\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1) &= \mathbf{\Delta}^H(i)\mathbf{H}(i-1)\hat{\mathbf{f}}(i-1) \\ &\quad + \mathbf{H}^H(i-1)\mathbf{\Delta}(i)\hat{\mathbf{f}}(i-1) + \mathbf{\Delta}^H(i)\mathbf{\Delta}(i)\hat{\mathbf{f}}(i-1). \end{aligned} \tag{C.1}$$

Denoting $\mathbf{b}(i-1) = \mathbf{H}(i-1)\hat{\mathbf{f}}(i-1)$, we obtain

$$\mathbf{b}(i-1) = [\mathbf{H}(i-2) + \mathbf{\Delta}(i-1)][\hat{\mathbf{f}}(i-2) + \Delta\hat{\mathbf{f}}(i-1)],$$

which gives a recursion for $\mathbf{b}(i-1)$:

$$\begin{aligned}\mathbf{b}(i-1) &= \mathbf{b}(i-2) + \mathbf{H}(i-2)\Delta\hat{\mathbf{f}}(i-1) \\ &\quad + \mathbf{\Delta}(i-1)\hat{\mathbf{f}}(i-1). \end{aligned} \tag{C.2}$$

Note that $\mathbf{\Delta}(i-1)$ is a Toeplitz matrix whose first column is $\Delta\hat{h}(i-1)\mathbf{e}_{p(i-1)}$. We also have $\Delta\hat{\mathbf{f}}(i-1) = \Delta\hat{f}(i-1)\mathbf{e}_{q(i-1)}$. Then (C.2) can be rewritten as

$$\begin{aligned}\mathbf{b}(i-1) &= \mathbf{b}(i-2) + \Delta\hat{f}(i-1)\hat{\mathbf{h}}^{[q(i-1)]}(i-2) \\ &\quad + \Delta\hat{h}(i-1)\hat{\mathbf{f}}^{[p(i-1)]}(i-1),\end{aligned}$$

where $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$ is a $(K+M-1)\times 1$ vector obtained by shifting elements of $\hat{\mathbf{h}}(i-2)$ by $q(i-1)$ positions down, and the other elements of $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$ are zeros. Definition

for $\hat{\mathbf{f}}^{[p(i-1)]}(i-1)$ is similar to that of $\hat{\mathbf{h}}^{[q(i-1)]}(i-2)$. Thus, the first term on the right hand side of (C.1) is given by

$$\begin{aligned}\boldsymbol{\Delta}^H(i)\mathbf{H}(i-1)\hat{\mathbf{f}}(i-1) &= \boldsymbol{\Delta}^H(i)\mathbf{b}(i-1) \\ &= \Delta\hat{h}^*(i)\mathbf{b}_{p(i):p(i)+K-1}(i-1),\end{aligned} \tag{C.3}$$

where $\mathbf{b}_{p(i):p(i)+K-1}(i-1)$ is a $K\times 1$ vector whose elements are obtained by extracting the $p(i)$th to $p(i)+K-1$th elements from the vector $\mathbf{b}(i-1)$. After some algebra, we find that the second term on the right hand side of (C.1) can be expressed as

$$\begin{aligned}\mathbf{H}^H(i-1)\boldsymbol{\Delta}(i)\hat{\mathbf{f}}(i-1) &= \boldsymbol{\Delta}^T(i)\mathbf{c}(i-1) \\ &= \Delta\hat{h}(i)\mathbf{c}_{M-p(i)+1:M-p(i)+K}(i-1),\end{aligned} \tag{C.4}$$

where, for the vector $\mathbf{c}(i-1)$ we obtain a recursion similar to that for $\mathbf{b}(i-1)$:

$$\begin{aligned}\mathbf{c}(i-1) = \mathbf{c}(i-2) &+ \Delta\hat{f}(i-1)\hat{\mathbf{u}}^{[q(i-1)]}(i-2) \\ &+ \Delta\hat{h}^*(i-1)\hat{\mathbf{f}}^{[M-p(i-1)+1]}(i-1),\end{aligned}$$

where elements of the vector $\hat{\mathbf{u}}(i-2)$ are given by

$$\hat{u}_m(i-2) = \hat{h}^*_{M-m+1}(i-2), m = 1,\ldots,M.$$

Since $\boldsymbol{\Delta}^H(i)\boldsymbol{\Delta}(i) = |\Delta\hat{h}(i)|^2\mathbf{I}_K$, the third term on the right hand side of (C.1) is given by

$$\boldsymbol{\Delta}^H(i)\boldsymbol{\Delta}(i)\hat{\mathbf{f}}(i-1) = |\Delta\hat{h}(i)|^2\hat{\mathbf{f}}(i-1). \tag{C.5}$$

From (C.3), (C.4) and (C.5), we finally obtain a simplified expression for (C.1):

$$\begin{aligned}\Delta\mathbf{G}(i)\hat{\mathbf{f}}(i-1) &= \Delta\hat{h}^*(i)\mathbf{b}_{p(i):p(i)+K-1}(i-1) \\ &+ \Delta\hat{h}(i)\mathbf{c}_{M-p(i)+1:M-p(i)+K}(i-1) + |\Delta\hat{h}(i)|^2\hat{\mathbf{f}}(i-1).\end{aligned}$$

Bibliography

[1] M. Stojanovic and J. Preisig, "Underwater acoustic communication channels: Propagation models and statistical characterization", *IEEE Communications Magazine*, vol. 47, no. 1, pp. 84–89, 2009.

[2] D.B. Kilfoyle and A.B. Baggeroer, "The state of the art in underwater acoustic telemetry", *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 4–27, 2000.

[3] A.C. Singer, J.K. Nelson, and S.S. Kozat, "Signal processing for underwater acoustic communications", *IEEE Communications Magazine*, vol. 47, no. 1, pp. 90–96, 2009.

[4] Simon Haykin, *Adaptive filter theory (4th Edition)*, Prentice Hall, 2001.

[5] A.H. Sayed, *Fundamentals of adaptive filtering*, IEEE Press, 2003.

[6] N. Al-Dhahir and J.M. Cioffi, "MMSE decision-feedback equalizers: Finite-length results", *IEEE Transactions on Information Theory*, vol. 41, no. 4, pp. 961–975, 1995.

[7] N. Al-Dhahir and J.M. Cioffi, "Fast computation of channel-estimate based equalizers in packet data transmission", *IEEE Transactions on Signal Processing*, vol. 43, no. 11, pp. 2462–2473, 1995.

[8] V. Chandrasekhar, W.K.G. Seah, Y.S. Choo, and H.V. Ee, "Localization in underwater sensor networks: survey and challenges", in *Proceedings of the 1st ACM international workshop on Underwater networks*. ACM, 2006, pp. 33–40.

[9] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking", in *IEEE Wireless Communications and Networking Conference, WCNC 2006*, 2006, vol. 1, pp. 228–235.

[10] H.P. Tan, Victoria University of Wellington. School of Engineering, and Computer Science, *A survey of techniques and challenges in underwater localization*, School of Engineering and Computer Science, Victoria University of Wellington, 2011.

[11] D. Pompili and I. Akyildiz, "Overview of networking protocols for underwater wireless communications", *IEEE Communications Magazine*, vol. 47, no. 1, pp. 97–102, 2009.

[12] M. Kocic, D. Brady, and M. Stojanovic, "Sparse equalization for real-time digital underwater acoustic communications", in *IEEE Conference on Oceans 1995*, vol. 3, pp. 1417–1422.

[13] S.F. Cotter and B.D. Rao, "Sparse channel estimation via matching pursuit with application to equalization", *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 374–377, 2002.

[14] C. Carbonelli and U. Mitra, "A simple sparse channel estimator for underwater acoustic channels", in *IEEE Conferences on OCEANS 2007*, pp. 1–6.

[15] C.R. Berger, S. Zhou, J.C. Preisig, and P. Willett, "Sparse channel estimation for multicarrier underwater acoustic communication: From subspace methods to compressed sensing", *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1708–1721, 2010.

[16] Y.V. Zakharov and T.C. Tozer, "Multiplication-free iterative algorithm for LS problem", *Electronics Letters*, vol. 40, pp. 567–569, April 2004.

[17] Y.V. Zakharov, G.P. White, and J. Liu, "Low-complexity RLS algorithms using dichotomous coordinate descent iterations", *IEEE Transactions on Signal Processing*, vol. 56, no. 7 Part 2, pp. 3150–3161, 2008.

[18] H.J Husøy, "A unified framework for adaptive filtering", *in Proc. NORSIG, Bergen, Norway*, October 2003.

[19] J.H. Husøy and M.S.E. Abadi, "Transient analysis of adaptive filters using a general framework", *AUTOMATIKA - Journal for Control, Measurement, Electronics, Computing and Communications*, vol. 45, no. 3-4, pp. 121 – 127, December 2004.

[20] S.J.M. de Almeida, J.C.M. Bermudez, N.J. Bershad, and M.H. Costa, "A statistical analysis of the affine projection algorithm for unity step size and autoregressive inputs", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 4, pp. 1394–1405, 2005.

[21] J.G. Proakis, *Digital communications*, McGraw-Hill New York, 1995.

[22] R. A Ziegler, M. W. Al-Dhahir, and J. M. Cioffi, "Nonrecursive adaptive decision-feedback equalization from channel estimates", in *IEEE 42nd Vehicular Technology Conference*, 1992, pp. 600–603.

[23] I. Lee and J.M. Cioffi, "A fast computation algorithm for the decision feedback equalizer", *IEEE Transactions on Communications*, vol. 43, no. 11, pp. 2742–2749, 1995.

[24] S. Chen, ES Chng, B. Mulgrew, and G. Gibson, "Minimum-BER linear-combiner DFE", in *IEEE International Conference on Communications, ICC 1996*, 1996, vol. 2, pp. 1173–1177.

[25] S. Ariyavisitakul and L.J. Greenstein, "Reduced-complexity equalization techniques for broadband wireless channels", *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 1, pp. 5–15, 1997.

[26] I.J. Fevrier, S.B. Gelfand, and M.P. Fitz, "Reduced complexity decision feedback equalization for multipath channels with large delay spreads", *IEEE Transactions on Communications*, vol. 47, no. 6, pp. 927–937, 1999.

[27] R. Lopez-Valcarce, "Realizable linear and decision feedback equalizers: properties and connections", *IEEE Transactions on Signal Processing*, vol. 52, no. 3, pp. 757–773, 2004.

[28] R. Merched and N.R. Yousef, "Fast techniques for computing finite-length MMSE decision feedback equalizers", in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04)*, 2004, vol. 4, pp. iv–1005.

[29] C. Zhang, Z. Wang, C. Pan, S. Chen, and L. Hanzo, "Low-complexity iterative frequency domain decision feedback equalization", *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 1295–1301, 2011.

[30] K. M. Z. Islam, N. Al-Dhahir, R. McKown, R. Dawes, and C. Stillo, "A new fast Householder-based fractionally-spaced FIR MMSE-DFE computation algorithm and its real-time implementation", *IEEE Communications Letters*, vol. 14, no. 11, pp. 1065–1067, 2010.

[31] I.F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges", *Ad Hoc Networks*, vol. 3, no. 3, pp. 257–279, 2005.

[32] H.P. Bucker, "Use of calculated sound fields and matched-field detection to locate sound sources in shallow water", *The Journal of the Acoustical Society of America*, vol. 59, pp. 368–373, 1976.

[33] A.B. Baggeroer, W.A. Kuperman, and H. Schmidt, "Matched field processing: Source localization in correlated noise as an optimum parameter estimation problem", *The Journal of the Acoustical Society of America*, vol. 83, no. 2, pp. 571–587, 1988.

[34] C.S. Clay, "Optimum time domain signal transmission and source location in a waveguide", *The Journal of the Acoustical Society of America*, vol. 81, pp. 660–664, 1987.

[35] L.N. Frazer and P.I. Pecholcs, "Single-hydrophone localization", *The Journal of the Acoustical Society of America*, vol. 88, pp. 995–1002, 1990.

[36] A. Tolstoy, "Computational aspects of matched field processing in underwater acoustics", *Computational Acoustics*, vol. 3, pp. 303–310.

[37] E.K. Westwood, "Broadband matched-field source localization", *The Journal of the Acoustical Society of America*, vol. 91, pp. 2777–2789, 1992.

[38] G.J. Orris, M. Nicholas, and J.S. Perkins, "The matched-phase coherent multi-frequency matched-field processor", *The Journal of the Acoustical Society of America*, vol. 107, pp. 2563–2575, 2000.

[39] C. Soares and S.M. Jesus, "Broadband matched-field processing: Coherent and incoherent approaches", *The Journal of the Acoustical Society of America*, vol. 113, pp. 2587–2598, 2003.

[40] H. Szu and R. Hartley, "Fast simulated annealing", *Physics Letters A*, vol. 122, pp. 157–162, 1987.

[41] S.E. Dosso, M.J. Wilmut, and A.L.S. Lapinski, "An adaptive-hybrid algorithm for geoacoustic inversion", *IEEE Journal of Oceanic Engineering*, vol. 26, no. 3, pp. 324–336, 2001.

[42] G.H. Golub and C.F. Van Loan, *Matrix computations*, Johns Hopkins Univ Pr, 1996.

[43] R. Barrett, *Templates for the solution of linear systems: building blocks for iterative methods*, Society for Industrial Mathematics, 1994.

[44] Y.V. Zakharov and F. Albu, "Coordinate descent iterations in fast affine projection algorithm", *IEEE Signal Processing Letters*, vol. 12, no. 2, pp. 353–356, 2005.

[45] F. Albu, "Efficient multichannel filtered-x affine projection algorithm for active noise control", *Electronics Letters*, vol. 42, no. 7, pp. 421–423, 2006.

[46] Y.V. Zakharov, "Low-complexity implementation of the affine projection algorithm", *IEEE Signal Processing Letters*, vol. 15, pp. 557–560, 2008.

[47] C. Stanciu, C. Anghel, C. Paleologu, J. Benesty, F. Albu, and S. Ciochina, "A proportionate affine projection algorithm using dichotomous coordinate descent iterations", in *IEEE 10th International Symposium on Signals, Circuits and Systems (ISSCS)*, 2011, pp. 1–4.

[48] J. Liu, B. Weaver, and G. White, "FPGA implementation of the DCD algorithm", in *the London Communications Symposium. University College London, London, UK*, 2006.

[49] J. Liu, Z. Quart, and Y. Zakharov, "Parallel FPGA implementation of DCD algorithm", in *IEEE 15th International Conference on Digital Signal Processing*, 2007, pp. 331–334.

[50] Z. Quan, J. Liu, and Y. Zakharov, "FPGA implementation of DCD based CDMA multiuser detector", in *IEEE 15th International Conference on Digital Signal Processing*, pp. 319–322.

[51] J. Liu, B. Weaver, and Y. Zakharov, "FPGA implementation of multiplication-free complex division", *Electronics Letters*, vol. 44, no. 2, pp. 95–96, 2008.

[52] J. Liu, Y.V. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 11, pp. 2425–2438, 2009.

[53] F. Auger, Z. Lou, B. Feuvrie, and F. Li, "Multiplier-free divide, square root, and log algorithms [DSP tips and tricks]", *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 122–126, 2011.

[54] R. Otnes and M. Tuchler, "Low-complexity turbo equalization for time-varying channels", in *IEEE 55th Vehicular Technology Conference, VTC Spring 2002*, 2002, vol. 1, pp. 140–144.

[55] C.J. William and C. Jakes, "Microwave mobile communications", 1994.

[56] R.H. Clarke, "A statistical theory of mobile-radio reception", *Bell Syst. Tech. J*, vol. 47, no. 6, pp. 957–1000, 1968.

[57] Y. R. Zheng and C. Xiao, "Simulation models with correct statistical properties for Rayleigh fading channels", *IEEE Transactions on Communications*, vol. 51, no. 6, pp. 920–928, 2003.

[58] C. Liu, Y. V. Zakharov, and T. Chen, "Modeling of time-varying underwater acoustic channels", in *Proceedings of the 4th Int. Conf. "Underwater acoustic measurements: Technologies and Results", Kos, Greece*, 20–24 June 2011, pp. 1423–1430.

[59] C. Liu, Y. V. Zakharov, and T. Chen, "Doubly-selective underwater acoustic channel model for moving transmitter/receiver", *under revision by IEEE Transactions on Vehicular Technology*, 2011.

[60] Z.H. Michalopoulou, "Matched-impulse-response processing for shallow-water localization and geoacoustic inversion", *The Journal of the Acoustical Society of America*, vol. 108, pp. 2082, 2000.

[61] M.B. Porter, "The KRAKEN normal mode program", Tech. Rep., DTIC Document, 1992.

[62] M. Porter, "Bellhop gaussian beam/finite element beam code", Available in the Acoustics Toolbox, `http://oalib.hlsresearch.com/Rays`.

[63] Y.V. Zakharov, T.C. Tozer, and J.F. Adlard, "Polynomial spline-approximation of Clarke's model", *IEEE Transactions on Signal Processing*, vol. 52, no. 5, pp. 1198–1208, 2004.

[64] T. Chen, Y. V. Zakharov, and C. Liu, "Source localization using matched-phase matched-field processing with phase descent search", *under revision by IEEE Journal on Oceanic Engineering*, 2010.

[65] T. Chen, Y. V. Zakharov, and C. Liu, "Low-complexity channel-estimate based adaptive linear equalizer", *IEEE Signal Processing Letters*, vol. 18, no. 7, pp. 427–430, 2011.

[66] T. Chen, Y. V. Zakharov, and C. Liu, "Partial-update channel-estimate based adaptive decision feedback equalizer", *under revision by IEEE Transactions on Signal Processing*, 2011.

[67] T. Chen and Y. Zakharov, "Convergence analysis of RLS-DCD algorithm", in *IEEE/SP 15th Workshop on Statistical Signal Processing, SSP'09*. IEEE, 2009, pp. 157–160.

[68] T. Chen, C. Liu, and Y. V. Zakharov, "Matched-phase coherent broadband matched-field processor using phase descent search", in *Tenth European Conference on Underwater Acoustics, ECUA*, 2010, pp. 590–595.

[69] C. Liu, T. Chen, and Y. V. Zakharov, "Source localization using sparsity based iterative adaptive beamforming", in *Tenth European Conference on Underwater Acoustics, ECUA*, 2010, pp. 604–610.

[70] C. Liu, T. Chen, and Y. V. Zakharov, "Matched field inversion for sound speed profile in a deep water environment by using simplex simulated annealing", in *Tenth European Conference on Underwater Acoustics, ECUA*, 2010, pp. 597–602.

[71] C. Liu, T. Chen, and Y.V. Zakharov, "Broadband underwater source localization by solving basis pursuit de-noising using coordinate descent search", in *IEEE 7th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, pp. 1–5.

[72] B. Farhang-Boroujeny, *Adaptive filters: Theory and applications*, Wiley, 1998.

[73] G.E.P. Box and G. Jenkins, *Time series analysis, forecasting and control*, Holden-Day, Incorporated, 1990.

[74] V.H. Nascimento and A.H. Sayed, "On the learning mechanism of adaptive filters", *IEEE Transactions on Signal Processing*, vol. 48, no. 6, pp. 1609–1625, 2000.

[75] K. Dogancay and O. Tanrikulu, "Adaptive filtering algorithms with selective partial updates", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 8, pp. 762–769, 2002.

[76] G.F. Xu, T. Bose, and J. Schroeder, "The Euclidean direction search algorithm for adaptive filtering", in *IEEE International Symposium on Circuits and Systems, ISCAS'99*, 1999, vol. 3, pp. 146–149.

[77] G.F. Xu, T. Bose, and J. Schroeder, "Channel equalization using an Euclidean direction search based adaptive algorithm", in *Global Telecommunications Conference*, 1998, vol. 6, pp. 3479–3484.

[78] P. Butler and A. Cantoni, "Noniterative automatic equalization", *IEEE Transactions on Communications*, vol. 23, no. 6, pp. 621–633, 1975.

[79] C.A. Belfiore and J.H. Park Jr, "Decision feedback equalization", *Proceedings of the IEEE*, vol. 67, no. 8, pp. 1143–1156, 1979.

[80] R. Merched and N.R. Yousef, "Fast techniques for computing finite-length MIMO MMSE decision feedback equalizers", *IEEE Transactions on Signal Processing*, vol. 54, no. 2, pp. 701–711, 2006.

[81] J. Cioffi and T. Kailath, "Fast recursive-least-squares transversal filters for adaptive filtering", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-32, pp. 304–337, 1984.

[82] S.U.H. Qureshi, "Adaptive equalization", *Proceedings of the IEEE*, vol. 73, no. 9, pp. 1349–1387, 1985.

[83] E.A. Lee and D.G. Messerschmitt, *Digital communication*, Springer, 1994.

[84] J.E. Smee and N.C. Beaulieu, "On the equivalence of the simultaneous and separate MMSE optimizations of a DFE FFF and FBF", *IEEE Transactions on Communications*, vol. 45, no. 2, pp. 156–158, 1997.

[85] R.N. Bracewell, *The Fourier transform and its applications*, McGraw-Hill, 2000.

[86] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *GPS: Theory and practice*, Springer Vienna, 1994.

[87] W.S. Burdic, *Underwater acoustic system analysis*, Peninsula Pub, 2002.

[88] H. Schau and A. Robinson, "Passive source localization employing intersecting spherical surfaces from time-of-arrival differences", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, no. 8, pp. 1223–1225, 1987.

[89] YT Chan and KC Ho, "A simple and efficient estimator for hyperbolic location", *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905–1915, 1994.

[90] J.C. Chen, K. Yao, and R.E. Hudson, "Acoustic source localization and beamforming: theory and practice", *EURASIP Journal on Applied Signal Processing*, pp. 359–370, 2003.

[91] A.B. Baggeroer, W.A. Kuperman, and P.N. Mikhalevsky, "An overview of matched field methods in ocean acoustics", *IEEE Journal of Oceanic Engineering*, vol. 18, no. 4, pp. 401–424, 1993.

[92] A.B. Baggeroer, "Broadband matched-field processing", *The Journal of the Acoustical Society of America*, vol. 102, pp. 3170, 1997.

[93] N.O. Booth, P.A. Baxley, J.A. Rice, P.W. Schey, W.S. Hodgkiss, G.L. D'Spain, and J.J. Murray, "Source localization with broad-band matched-field processing in shallow water", *IEEE Journal of Oceanic Engineering*, vol. 21, no. 4, pp. 402–412, 1996.

[94] C.E. Lindsay and N.R. Chapman, "Matched field inversion for geoacoustic model parameters using adaptive simulated annealing", *IEEE Journal of Oceanic Engineering*, vol. 18, no. 3, pp. 224–231, 1993.

[95] N.R. Chapman and C.E. Lindsay, "Matched-field inversion for geoacoustic model parameters in shallow water", *IEEE Journal of Oceanic Engineering*, vol. 21, no. 4, pp. 347–354, 1996.

[96] R.K. Brienzo and W.S. Hodgkiss, "Broadband matched-field processing", *The Journal of the Acoustical Society of America*, vol. 94, pp. 2821, 1993.

[97] Z. Quan, Y.V. Zakharov, and J. Zhang, "Multiple phase decoder for MIMO systems", in *42nd Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Oct. 26–29, 2008, pp. 1759–1762.

[98] Y.V. Zakharov and T.C. Tozer, "Frequency estimator with dichotomous search of periodogram peak", *Electronics Letters*, vol. 35, no. 19, pp. 1608–1609, 1999.

[99] J. Murray and D. Ensberg, "The SWellEx-96 Experiment", `http://www.mpl.ucsd.edu/swellex96/`.

[100] G.L. D'Spain, J.J. Murray, W.S. Hodgkiss, N.O. Booth, and P.W. Schey, "Mirages in shallow water matched field processing", *The Journal of the Acoustical Society of America*, vol. 105, pp. 3245–3265, 1999.

[101] D.R. Del Balzo, C. Feuillade, and M.M. Rowe, "Effects of water-depth mismatch on matched-field localization in shallow water", *The Journal of the Acoustical Society of America*, vol. 83, pp. 2180–2185, 1988.

[102] P. Hursky, *Using ambient noise and sources of opportunity to estimate environment parameters and improve matched field source detection and localization*, PhD thesis, University of California, San Diego, Jul 2001.

[103] C.E. Davila, "Line search algorithms for adaptive filtering", *IEEE Transactions on Signal Processing*, vol. 41, no. 7, pp. 2490–2494, 1993.

[104] G.F. Xu and T. Bose, "Analysis of the Euclidean direction set adaptive algorithm", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1998, vol. 3, pp. 1689–1692.

[105] M.Q. Chen, "A direction set based algorithm for adaptive least squares problems in signal processing", *Applied and Computational Control, Signals, and Circuits*, pp. 213–238, 2001.

[106] J.R. Treichler, I. Fijalkow, and Jr. Johnson, C.R., "Fractionally spaced equalizers", *IEEE Signal Processing Magazine*, vol. 13, no. 3, pp. 65–81, 1996.

[107] S. Roy, T.M. Duman, V. McDonald, and J.G. Proakis, "High-rate communication for underwater acoustic channels using multiple transmitters and space–time coding: Receiver structures and experimental results", *IEEE Journal of Oceanic Engineering*, vol. 32, no. 3, pp. 663–688, 2007.

[108] J. Zhang, Y.R. Zheng, and C. Xiao, "Frequency-domain equalization for single carrier MIMO underwater acoustic communications", in *IEEE Conference on OCEANS 2008*, pp. 1–6.

[109] B. Li, J. Huang, S. Zhou, K. Ball, M. Stojanovic, L. Freitag, and P. Willett, "MIMO-OFDM for high-rate underwater acoustic communications", *IEEE Journal of Oceanic Engineering*, vol. 34, no. 4, pp. 634–644, 2009.

[110] J. Tao, Y.R. Zheng, C. Xiao, TC Yang, and W.B. Yang, "Channel equalization for single carrier MIMO underwater acoustic communications", *EURASIP Journal on Advances in Signal Processing*, vol. 2010, pp. 12.

[111] G. Forney Jr, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference", *IEEE Transactions on Information Theory*, vol. 18, no. 3, pp. 363–378, 1972.

[112] EM Sozer, JG Proakis, and F. Blackmon, "Iterative equalization and decoding techniques for shallow water acoustic channels", in *OCEANS, MTS/IEEE Conference and Exhibition*, 2001, vol. 4, pp. 2201–2208.

[113] F. Blackmon, E. Sozer, and J. Proakis, "Iterative equalization, decoding, and soft diversity combining for underwater acoustic channels", in *OCEANS'02 MTS/IEEE*, 2002, vol. 4, pp. 2425–2428.

[114] M. Tuchler, A.C. Singer, and R. Koetter, "Minimum mean squared error equalization using a priori information", *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 673–683, 2002.

[115] J. Egle and J. Lindner, "Iterative joint equalization and decoding based on soft cholesky equalization for general complex valued modulation symbols", *Advanced Signal Processing for Communication Systems*, pp. 267–282, 2002.

[116] M. Chitre, S. Shahabudeen, L. Freitag, and M. Stojanovic, "Recent advances in underwater acoustic communications & networking", in *OCEANS'08 IEEE*, 2008, vol. 2008, pp. 1–10.