Adaptive and Efficient Nonlinear Channel Equalization for Underwater Acoustic Communication

Dariush Kari [a], Nuri Denizcan Vanli [b], Suleyman S. Kozat [a,*]

[a] *Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey, 06800*

[b] *Laboratory of Information and Decision Systems, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139*

Abstract

We investigate underwater acoustic (UWA) channel equalization and introduce hierarchical and adaptive nonlinear channel equalization algorithms that are highly efficient and provide significantly improved bit error rate (BER) performance. Due to the high complexity of nonlinear equalizers and poor performance of linear ones, to equalize highly difficult underwater acoustic channels, we employ piecewise linear equalizers. However, in order to achieve the performance of the best piecewise linear model, we use a tree structure to hierarchically partition the space of the received signal. Furthermore, the equalization algorithm should be completely adaptive, since due to the highly non-stationary nature of the underwater medium, the optimal MSE equalizer as well as the best piecewise linear equalizer changes in time. To this end, we introduce an adaptive piecewise linear equalization algorithm that not only adapts the linear equalizer at each region but also learns the complete hierarchical structure with a computational complexity only polynomial in the number of nodes of the tree. Furthermore, our algorithm is constructed to directly minimize the final squared error without introducing any ad-hoc parameters. We demonstrate the performance of our algorithms through highly realistic experiments performed on accurately simulated underwater acoustic channels.

Key words: Underwater acoustic communication, nonlinear channel equalization,

1 Introduction

Underwater acoustic (UWA) domain has become an important research field due to proliferation of new and exciting applications [1, 2]. However, due to poor physical link quality, high latency, constant movement of waves and chemical properties of water, the underwater acoustic channel is considered as one of the most adverse communication mediums in use today [3–5]. These adverse properties of the underwater acoustic channel should be equalized by in order to provide reliable communication [2, 3, 6–13]. Furthermore, due to rapidly changing and unpredictable nature of underwater environment, constant movement of waves and transmitter-receivers, such processing should be adaptive [2, 7, 9, 11]. However, there exist significant practical and theoretical difficulties to adaptive signal processing in underwater applications, since the signal generated in these applications show high degrees of non-stationarity, limit cycles and, in many cases, are even chaotic. Hence, the classical adaptive approaches that rely on assumed statistical models are inadequate since there is usually no or little knowledge about the statistical properties of the underlying signals or systems involved [3, 14, 15].

In this paper, in order to rectify the undesirable effects of underwater acoustic channels, we introduce a radical approach to adaptive channel equalization and seek to provide robust adaptive algorithms in an individual sequence manner [16]. Since the signals generated in this domain have high degrees of non-stationarity and uncertainty, we introduce a completely novel approach to adaptive channel equalization and aim to design adaptive methods that are

* Corresponding author.
Email addresses: kari@ee.bilkent.edu.tr (Dariush Kari), denizcan@mit.edu (Nuri Denizcan Vanli), kozat@ee.bilkent.edu.tr (Suleyman S. Kozat).

mathematically guaranteed to work uniformly for all possible signals without any explicit or implicit statistical assumptions on the underlying signals or systems [16].

Although linear equalization is the simplest equalization method, it delivers an extremely inferior performance compared to that of the optimal methods, such as Maximum A Posteriori (MAP) or Maximum Likelihood (ML) methods [10,17,18]. Nonetheless, the high complexities of the optimal methods, and also their need of the channel information [8, 10, 12, 19, 20], make them practically infeasible for UWA channel equalization, because of the extremely large delay spread of UWA channels [8, 18, 21–23]. Hence, we seek to provide powerful nonlinear equalizers with low complexities as well as linear ones. To this end, we employ piecewise linear methods, since the simplest and most effective as well as close to the nonlinear equalizers are piecewise linear ones [16, 24]. By using piecewise linear methods, we can retain the breadth of nonlinear equalizers, while mitigating the over-fitting problems associated with these models [24, 25]. As a result, piecewise linear filters are used in a vast variety of applications in signal processing and machine learning literature [25].

In piecewise linear equalization methods, the space of the received signal is partitioned into disjoint regions, each of which is then fitted a linear equalizer [18, 24]. We use the term "linear" to refer generally to the class of "affine" rather than strictly linear filters. In its most basic form, a fixed partition is used for piecewise linear equalization, i.e., both the number of regions and the region boundaries are fixed over time [24,25]. To estimate the transmitted symbol with a piecewise linear model, at each specific time, exactly one of the linear equalizers is used [18]. The linear equalizers in every region should be adaptive such that they can match the time varying channel response. However, due to the non-stationary statistics of the channel response, a fixed partition over time cannot result in a satisfactory performance. Hence, the partitioning should be adaptive as well as the linear equalizers in each region.

To this aim, we use a novel piecewise linear algorithm in which not only the linear equalizers in each region, but also the region boundaries are adaptive [16]. Therefore, the regions are effectively adapted to the channel response and follow the time variations of the best equalizer in highly time varying UWA channels. In this sense, our algorithm can achieve the performance of the best piecewise linear equalizer with the same number of regions, i.e., the linear equalizers as well as the region boundaries converge to their optimal linear solutions.

Nevertheless, due to the non-stationary channel statistics, there is no knowledge about the number of regions of the best piecewise linear equalizer, i.e., even with adaptive boundaries, the piecewise linear equalizer with a certain number of regions, does not perform well. Hence, we use a tree structure to construct a class of models, each of which has a different number of regions [24,26]. Each of these models can be then employed to construct a piecewise linear equalizer with adaptive filters in each region and also adaptive region boundaries [16]. In [26], they choose the best model (subtree) represented by a tree over a fixed partition. However, the final estimates of all of these models should be effectively combined to achieve the performance of the best piecewise linear equalizer within this class [16]. For this, we assign a weight to each model and linearly combine the results generated by each of them. However, due to the high computational complexity resulted from running a large number of different models, we introduce a technique to combine the node estimates to produce the exactly same result. We emphasize that we directly combine the node estimates with specific weights rather than running all of these doubly exponential [24] number of models. Furthermore, the algorithm adaptively learns the node combination weights and the region boundaries as well as the linear equalizers in each region, to achieve the performance of the best piecewise linear equalizer. Specifically, we apply a computationally efficient solution to the UWA channel equalization problem using turning boundaries trees [16].

As a result, in highly time varying UWA channels, we significantly outperform other piecewise linear equalizers constructed over a fixed partition.

In this paper, we introduce an algorithm that is shown *i)* to provide significantly improved BER performance over the conventional linear and piecewise linear equalization methods in realistic UWA experiments *ii)* to have guaranteed performance bounds without any statistical assumptions. Our algorithm not only adapts the corresponding linear equalizers in each region, but also learns the corresponding region boundaries, as well as the "best" linear mixture of a doubly exponential number of piecewise linear equalizers. Hence, the algorithm minimizes the final soft squared error, with a computational complexity only polynomial in the number of nodes of the tree. In our algorithm, we avoid any artificial weighting of models with highly data dependent parameters and, instead, "directly" minimize the squared error. Hence, the introduced approach significantly outperforms the other tree based approaches such as [24], as demonstrated in our simulations.

The paper is organized as follows: In section 2 we describe our framework mathematically and introduce the notations. Then, in section 3 we first present an algorithm to hierarchically partition the space of the received signal. We then present an upper bound on the performance of the promised algorithm and construct the algorithm. In section 4 we show the performance of our method using highly realistic simulations, and then conclude the paper with section 5.

2 Problem Description

We denote the received signal by $\{r(t)\}_{t\geq 1}$, $r(t) \in \mathbb{R}$, and our aim is to determine the transmitted bits $\{b(t)\}_{t\geq 1}$, $b(t) \in \{-1, 1\}$. All vectors are column vectors and denoted by boldface lower case letters. For a vector $\boldsymbol{x}$, $\boldsymbol{x}^T$ is the

ordinary transpose.

In UWA communication, if the input signal is bandlimited, the baseband signal at the output is modeled as follows [27]

$$y(t) = \sum_{p=0}^{K} g_p(t)x(t - pT_s) + \nu(t), \tag{1}$$

where $y(t)$ is the channel output, T_s is the sampling interval, K is the minimum number beyond which the tap gains $g_p(t)$ are negligible, $\nu(t)$ indicates the ambient noise, and $g_p(t)$ is defined by

$$g_p(t) \triangleq \int_{-\infty}^{\infty} c(\tau, t)\ \mathrm{sinc}(\frac{\tau - pT_s}{T_s})\ d\tau. \tag{2}$$

where $c(\tau, t)$ indicates the channel response at time t related to an impulse launched at time $t - \tau$, and τ is the delay time. The input signal $x(t)$ is the pulse shaped signal generated from the sequence of bits $\{b(t)\}_{t\geq 1}$ transmitted every T_s seconds. Note that the effects of time delay and phase deviations are usually addressed at the front-end of the receiver. Hence, we do not deal with this representation of the received signal. Instead, we assume that channel is modeled by a discrete time impulse response (i.e., a tap delay model). With a small abuse of notation, in the rest of the paper, we denote the discrete sampling times by t, such that the received signal can be represented as

$$r(t) = \sum_{k=-N_2}^{N_1} b(k)\tilde{g}(t - k) + \nu(t), \tag{3}$$

where $r(t) \triangleq y(tT_s)$ is the output of the discrete channel model, $\tilde{g}(k)$ is the kth tap of the discrete channel impulse response, and $\nu(t)$ represents the ambient noise. We have assumed that the discrete channel can be effectively represented by N_1 causal and N_2 anti-causal taps. The input symbols $b(t)$ are transmitted every T_s seconds and our aim is to estimate the transmitted bits $\{b(t)\}_{t\geq 1}$ according to the channel outputs $\{r(t)\}_{t\geq 1}$. In this setup, a linear channel

equalizer can be constructed as

$$\hat{b}(t) = \boldsymbol{w}^T(t)\boldsymbol{r}(t), \tag{4}$$

where $\boldsymbol{r}(t) \triangleq [r(t), \ldots, r(t-h+1)]^T$ is the received signal vector at time t, $\boldsymbol{w}(t) \triangleq [w_0(t), \ldots, w_{h-1}(t)]^T$ is the linear equalizer at time t, and h is the equalizer length. The tap weights $\boldsymbol{w}(t)$ can be updated using any adaptive filtering algorithm such as the least mean squares (LMS) or the recursive least squares (RLS) algorithms [28] in order to minimize the squared error loss function, where the soft error at time t is defined as

$$e(t) = b(t) - \hat{b}(t).$$

However, we can get significantly better performance by using adaptive nonlinear equalizers, because such linear equalization methods usually yield unsatisfactory performance [18]. Thus, we employ piecewise linear equalizers, which serve as the most natural and computationally efficient extension to linear equalizers [25], since the equalizer lengths are significantly large in UWA channels [29]. The block diagram of a sample adaptive piecewise linear equalizer is shown in the Fig. 1. In such equalizers, the space of the received signal (here, $\mathbb{R}^h$) is partitioned into disjoint regions, to each of which a different linear equalizer is assigned.

As an example, in Fig. 2, we use the received signal $\boldsymbol{r}(t) \triangleq [r(t), r(t-1)]^T \in \mathbb{R}^2$ to estimate the transmitted bit $b(t)$. We partition the space $\mathbb{R}^2$ into two regions R_1 and R_2, and use different linear equalizers $\boldsymbol{w}_1 \in \mathbb{R}^2$ and $\boldsymbol{w}_2 \in \mathbb{R}^2$ in these regions respectively. Hence the estimate $\hat{b}(t)$ is calculated as

$$\hat{b}(t) = \begin{cases} \boldsymbol{w}_1^T(t)\boldsymbol{r}(t) + c_1(t) & \text{if } \boldsymbol{r}(t) \in R_1 \\ \boldsymbol{w}_2^T(t)\boldsymbol{r}(t) + c_2(t) & \text{if } \boldsymbol{r}(t) \in R_2, \end{cases}$$

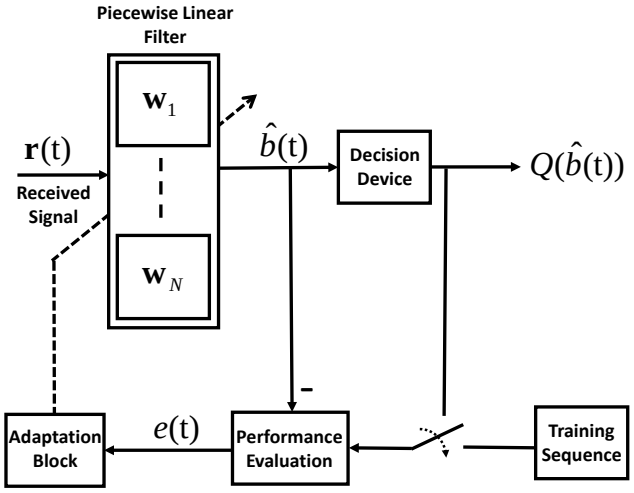

Fig. 1. The block diagram of an adaptive piecewise linear equalizer. This equalizer consists of N different linear filters, one of which is used for each time step, based on the region (a subset of $\mathbb{R}^h$, where h is the length of each filter) in which the received signal vector lies.

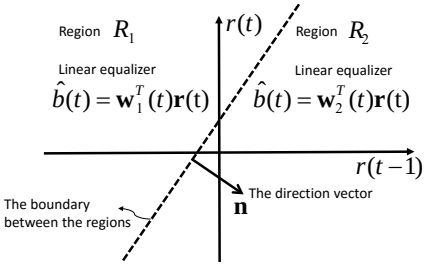

Fig. 2. A simple two region partition of the space $\mathbb{R}^2$. We use different equalizers $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ in regions R_1 and R_2 respectively. The direction vector $\boldsymbol{n}$ is an orthogonal vector to the regions boundary (the hyper-plane used to separate the regions).

where $c_1(t) \in \mathbb{R}$ and $c_2(t) \in \mathbb{R}$ are the offset terms, which can be embedded into $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, i.e., $\boldsymbol{w}_j \triangleq [\boldsymbol{w}_j^T \quad c_j]^T, j = 1, 2$, and $\boldsymbol{r} \triangleq [\boldsymbol{r}^T \quad 1]^T$. Hence the above expression can be rewritten as

$$\hat{b}(t) = \begin{cases} \boldsymbol{w}_1^T(t)\boldsymbol{r}(t) & \text{if } \boldsymbol{r}(t) \in R_1 \\ \boldsymbol{w}_2^T(t)\boldsymbol{r}(t) & \text{if } \boldsymbol{r}(t) \in R_2. \end{cases}$$

Because of the high complexity of the best linear minimum mean squared error (MMSE) equalizer in each region [18], as well as the rapidly changing characteristics of the UWA channel, we use low complexity adaptive techniques to achieve the best linear equalizer in each region [28]. Hence we update the

equalizer's coefficients using least mean squares (LMS) algorithm as

$$\begin{aligned}
\boldsymbol{w}_1(t+1) &= \boldsymbol{w}_1(t) + \mu_1\ e(t)\ \boldsymbol{r}(t) \quad \text{if } \boldsymbol{r}(t) \in R_1 \\
\boldsymbol{w}_2(t+1) &= \boldsymbol{w}_2(t) + \mu_2\ e(t)\ \boldsymbol{r}(t) \quad \text{if } \boldsymbol{r}(t) \in R_2,
\end{aligned}$$

Note that the complexity of the MMSE method is quadratic in the equalizer length [18], while the LMS method has a complexity only linear in the equalizer length.

To obtain a general expression, consider that we use a partition P with N subsets (regions) to divide the space of the received signal into disjoint regions, i.e.,

$$\begin{aligned}
&P = \{R_1, \ldots, R_N\} \\
&\mathbb{R}^h = \cup_{j=1}^{N} R_j \\
&\hat{b}(t) = \hat{b}_j(t) = \boldsymbol{w}_j^T(t)\boldsymbol{r}(t) \quad \text{if } \boldsymbol{r}(t) \in R_j,
\end{aligned} \tag{5}$$

which can be rewritten using indicator functions as

$$\begin{aligned}
\hat{b}(t) &= \sum_{j=1}^{N} \hat{b}_j(t)\ \mathrm{id}_j(\boldsymbol{r}(t)) \\
&= \sum_{j=1}^{N} \boldsymbol{w}_j^T(t)\boldsymbol{r}(t)\ \mathrm{id}_j(\boldsymbol{r}(t)),
\end{aligned} \tag{6}$$

where the indicator function $\mathrm{id}_j(\boldsymbol{r}(t))$ determines whether the received signal vector $\boldsymbol{r}(t)$ lies in the region R_j or not, i.e.,

$$\mathrm{id}_j(\boldsymbol{r}(t)) = \begin{cases} 1 & \text{if } \boldsymbol{r}(t) \in R_j \\ 0 & \text{otherwise.} \end{cases}$$

Remark 1: Note that this algorithm can be directly applied to DFE equalizers. In this scenario, we partition the space of the extended received signal

vector. To this end, we append the past decided symbols to the received signal vector as

$$\tilde{\boldsymbol{r}}(t) \triangleq [r(t), \ldots, r(t-h+1), \bar{b}(t-1), \ldots, \bar{b}(t-h_f)]^T,$$

where h_f is the length of the feedback part of the equalizer, i.e., we partition $\mathbb{R}^{(h+h_f)}$. Also, $\bar{b}(t) = Q(\hat{b}(t))$ denotes the quantized estimate of the transmitted bit $b(t)$. Furthermore, corresponding to this extension in the received signal vector, we merge the feed-forward and feedback equalizers in each region to obtain an extended filter of length $h + h_f$ as

$$\tilde{\boldsymbol{w}}_j(t) \triangleq [\boldsymbol{w}_j^T(t) \quad \boldsymbol{f}_j^T(t)]^T,$$

where $\boldsymbol{f}_j(t)$ represents the feedback filter corresponding to the jth region at time t. Hence, the jth region estimate is calculated as

$$\hat{b}_j(t) = \tilde{\boldsymbol{w}}_j^T(t)\ \tilde{\boldsymbol{r}}(t).$$

In the next section, we extend these expressions to the case of an adaptive partition, both in the region boundaries and number of regions, and introduce our final algorithm.

3 Adaptive Partitioning of The Received Signal Space

3.1 An Adaptive Piecewise Linear Equalizer with a Specific Partition

Due to the non-stationary nature of underwater channel, a fixed partitioning over time cannot match well to the channel response, i.e., the partitioning should be adaptive. Hence we use a partition with adaptive boundaries. To this end, we use hyper-planes with adaptive direction vectors (a vector orthogonal

to the hyper-plane) as boundaries. We use $\boldsymbol{n}$ to refer to the direction vector of a hyperplane.

As an example consider a partition with two regions as depicted in Fig. 2, hence, there is one boundary, the direction vector to which is shown by $\boldsymbol{n}$. The indicator functions for these regions are calculated as

$$\begin{aligned} \mathrm{id}_1(\boldsymbol{r}(t)) &= \sigma(\boldsymbol{r}(t)) \\ \mathrm{id}_2(\boldsymbol{r}(t)) &= 1 - \sigma(\boldsymbol{r}(t)), \end{aligned}$$

where

$$\sigma(\boldsymbol{r}(t)) = \begin{cases} 1 & \text{if } \boldsymbol{r}(t) \in R_1 \\ 0 & \text{if } \boldsymbol{r}(t) \in R_2, \end{cases}$$

represents the hard separation of the regions. However, in order to learn the region boundaries, we use a soft separator function, which is defined as

$$\sigma(\boldsymbol{r}) \triangleq \frac{1}{1 + e^{\boldsymbol{r}^T \boldsymbol{n} + b}} , \tag{7}$$

which yields

$$\sigma(\boldsymbol{r}) = \begin{cases} 1 & \text{if } \boldsymbol{r}^T \boldsymbol{n} + b \ll 0 \\ 0 & \text{if } \boldsymbol{r}^T \boldsymbol{n} + b \gg 0. \end{cases}$$

Although this separator function is not a hard separator function, it is a differentiable function, hence, it can be used to simply update the direction vector $\boldsymbol{n}$ using LMS algorithm, resulting in an adaptive boundary. For simplicity, with a small abuse of notation, we redefine $\boldsymbol{n}$ and $\boldsymbol{r}$, as $\boldsymbol{n} \triangleq [\boldsymbol{n}^T \quad b]^T$ and

$\boldsymbol{r} \triangleq [\boldsymbol{r}^T \quad 1]^T$, hence (7) can be rewritten as

$$\sigma(\boldsymbol{r}) \triangleq \frac{1}{1+e^{\boldsymbol{r}^T\boldsymbol{n}}} \ . \tag{8}$$

We use the LMS algorithm to update the direction vector $\boldsymbol{n}$. Hence,

$$\begin{aligned}
\boldsymbol{n}(t+1) &= \boldsymbol{n}(t) - \frac{1}{2}\mu \ \nabla_{\boldsymbol{n}(t)} e^2 \\
&= \boldsymbol{n}(t) + \mu \ e(t) \ \frac{\partial \ \hat{b}(t)}{\partial \ \boldsymbol{n}(t)} \\
&= \boldsymbol{n}(t) + \mu \ e(t) \ \left(\frac{\partial \ \mathrm{id}_1(\boldsymbol{r}(t))}{\partial \ \boldsymbol{n}(t)} \hat{b}_1(t) + \frac{\partial \ \mathrm{id}_2(\boldsymbol{r}(t))}{\partial \ \boldsymbol{n}(t)} \hat{b}_2(t) \right) \\
&= \boldsymbol{n}(t) + \mu \ e(t) \ \sigma(\boldsymbol{r})(\sigma(\boldsymbol{r})-1) \left(\hat{b}_1(t) - \hat{b}_2(t) \right) \ \boldsymbol{r}(t),
\end{aligned}$$

since

$$\begin{aligned}
\frac{\partial \sigma(\boldsymbol{r})}{\partial \boldsymbol{n}} &= \frac{-\boldsymbol{r} \ e^{\boldsymbol{r}^T\boldsymbol{n}+b}}{(1+e^{\boldsymbol{r}^T\boldsymbol{n}+b})^2} \\
&= -\boldsymbol{r}\sigma(\boldsymbol{r})(1-\sigma(\boldsymbol{r})).
\end{aligned} \tag{9}$$

Since the region boundaries as well as the linear filters in each region are adaptive, if every filter converges, this equalizer can perform better than other piecewise linear equalizers with the same number of regions.

Remark: The piecewise linear equalizers are not limited to the BPSK modulation and one can easily extend these results to higher order modulation schemes like QAM or PAM. However, for the complex valued data (e.g., in QAM modulations) the separating function should change as

$$\sigma(\boldsymbol{r}) \triangleq \frac{1}{1+e^{\boldsymbol{r}_{\mathrm{re}}^T\boldsymbol{n}_{\mathrm{re}}+\boldsymbol{r}_{\mathrm{im}}^T\boldsymbol{n}_{\mathrm{im}}}} \tag{10}$$

where the subscripts "re" and "im" denote the real and imaginary part of each

vector respectively, e.g.,

$$\begin{aligned}\boldsymbol{r}_{\text{re}} &= [\text{Re}\{r(t)\}, \ldots, \text{Re}\{r(t-h+1)\}]^T \\ \boldsymbol{r}_{\text{im}} &= [\text{Im}\{r(t)\}, \ldots, \text{Im}\{r(t-h+1)\}]^T. \end{aligned} \tag{11}$$

3.2 The Completely Adaptive Equalizer Based on a Turning Boundaries Tree

The block diagram of a sample adaptive piecewise linear equalizer with adaptive regions is shown in Fig. 3. Given a fixed number of regions, we can achieve the best piecewise linear equalizer with the algorithm described in Section 3.1. However, there is no a priori knowledge about the number of regions of the best piecewise linear equalizer, and the best linear equalizer will change in time, due to the highly non-stationary nature of underwater medium. In order to provide an acceptable performance with a relatively small computational complexity, we introduce a tree-based piecewise linear equalization algorithm, where we hierarchically partition the space of the received signal, i.e., $\mathbb{R}^h$. Every node of the tree represents a region and is fitted a linear equalizer, as shown in Fig. 4. As shown in Fig. 3, each node j provides its own estimate $\hat{b}_j(t)$, which are then combined to generate the final estimate $\hat{b}(t)$ as

$$\begin{aligned}\hat{b}(t) &= \sum_{j=1}^{2^{d+1}-1} u_j(t)\boldsymbol{w}_j^T(t)\boldsymbol{r}(t), \\ &= \boldsymbol{u}^T(t)\hat{\boldsymbol{b}}(t), \end{aligned} \tag{12}$$

where $\boldsymbol{u}(t) = [u_1(t), \ldots, u_{2^{d+1}-1}(t)]^T$ is the combination weight vector, which is updated each time, and $\hat{\boldsymbol{b}}(t) = [\hat{b}_1(t), ..., \hat{b}_{2^{d+1}-1}(t)]^T$ is the vector of the node estimates.

As depicted in Fig. 5, this tree introduces a number of partitions with different number of regions, each of which can be separately used as a piecewise linear equalizer [16]. Note that in our method, both the region boundaries and the

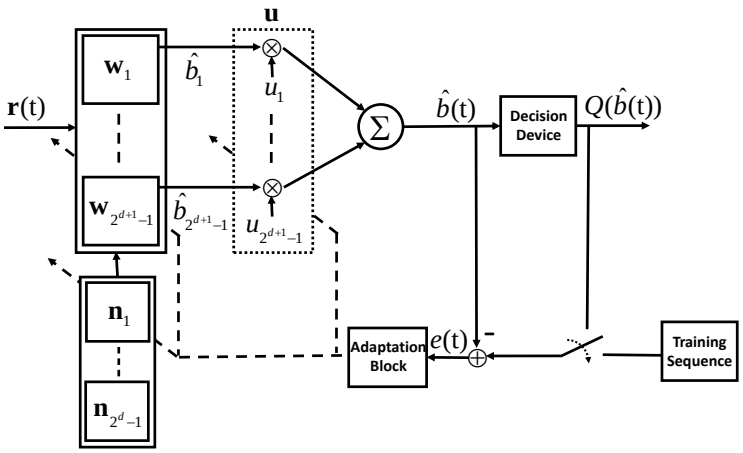

Fig. 3. The block diagram of a turning boundaries tree (TBT) equalizer. The received signal space is partitioned using a depth d tree, and corresponding to each node i there is a linear filter $\boldsymbol{w}_i$. Furthermore, the direction vectors of the separating hyper-planes, $\boldsymbol{n}$'s, are adaptive resulting in an adaptive tree. The weight vector $\boldsymbol{u}$, which contains the combination weights for each node's contribution, is also adaptive.

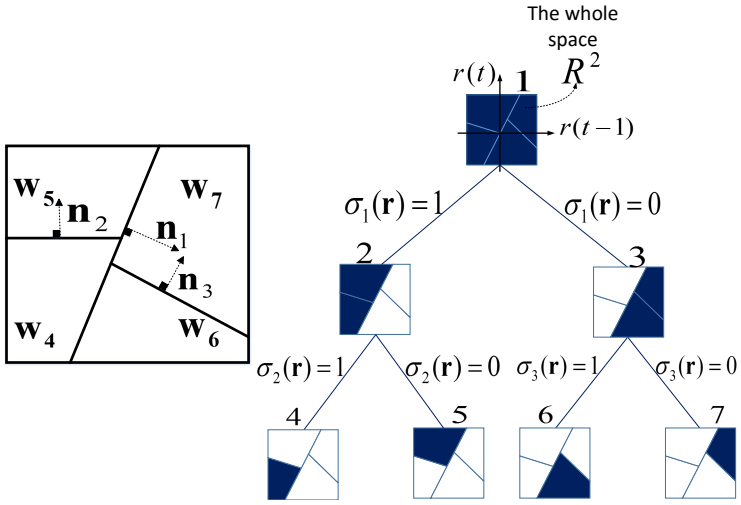

Fig. 4. Partitioning the space $\mathbb{R}^2$ using a depth-2 tree structure. Hyper-planes (lines) are used to divide the regions. The direction vectors are the orthogonal vectors to the hyper-planes.

channel equalizers in each region are adaptive.

To achieve the performance of the best piecewise linear equalizer, we hierarchically partition the space of the received signal. To this aim we use a tree structure in which, each node represents a region that is the union of the regions assigned to its left and right children [30], as shown in Fig. 4. We denote the root node by 1, and the left and right children of the node j by $2j$ and

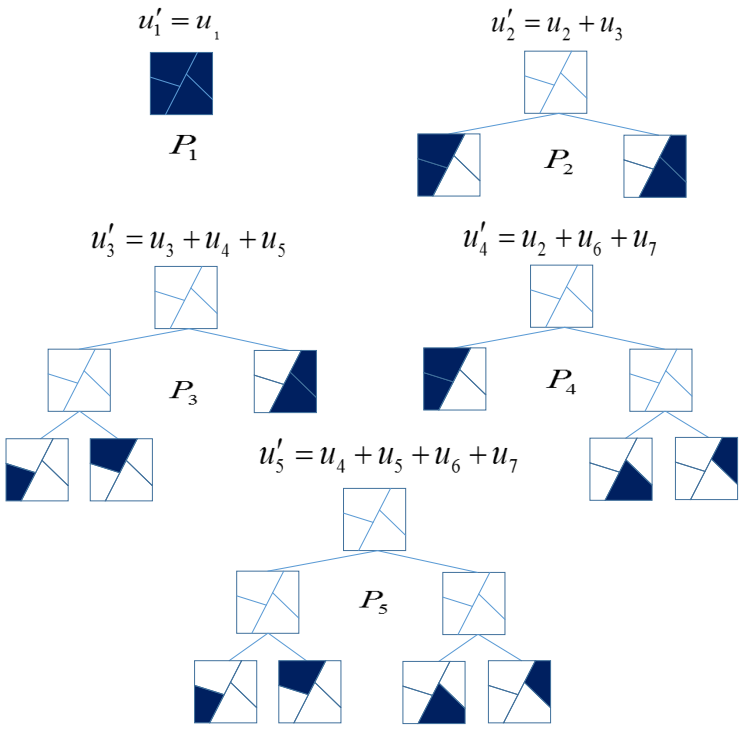

Fig. 5. All different partitions of the received signal space that can be obtained using a depth-2 tree. Any of these partition can be used to construct a piecewise linear equalizer, which can be adaptively trained to minimize the squared error. These partitions are based on the separation functions shown in Fig. 4.

$2j+1$, respectively. Obviously the root node indicates the whole space of the received signal, i.e., $\mathbb{R}^h$. The estimate generated by node j is calculated as

$$\hat{b}_j(t) = \boldsymbol{w}_j^T(t)\ \boldsymbol{r}(t).$$

We denote by α_d the number of partitioning trees with depth$\leq d$. Hence,

$$\alpha_{d+1} = \alpha_d^2 + 1,$$

which shows that there are a doubly exponential number of models embedded in a depth-d tree (See Fig. 5), each of which can be used to construct a piecewise linear equalizer [24]. Each of these models consists of a number of nodes. However, the number of regions (leaf nodes) in each model can be different with that of other models, as shown in Fig. 5, e.g., P_2 has two regions, while P_5 has 4 regions. Therefore, we implicitly run all of the piecewise

linear equalizers constructed based on these partitions, and linearly combine their results to estimate the transmitted bit. We then adaptively learn the combination weights to achieve the best estimate at each time.

To clarify the framework, suppose the corresponding space of the received signal vector is two dimensional, i.e., $\boldsymbol{r}(t) \in \mathbb{R}^2$, and we partition this space using a depth-2 tree as in Fig. 4. A depth-2 tree is represented by three separating functions $\sigma_1(\boldsymbol{r}(t))$, $\sigma_2(\boldsymbol{r}(t))$ and $\sigma_3(\boldsymbol{r}(t))$, which are defined using three hyper-planes with direction vectors $\boldsymbol{n}_1(t)$, $\boldsymbol{n}_2(t)$ and $\boldsymbol{n}_3(t)$, respectively (See Fig. 4). The left and right children of the node j are $2j$ and $2j+1$ respectively, therefore, the indicator functions are defined as

$$\begin{aligned} \mathrm{id}_1(\boldsymbol{r}) &= 1 \\ \mathrm{id}_{2j}(\boldsymbol{r}) &= \sigma_j(\boldsymbol{r}) \ \times \mathrm{id}_j(\boldsymbol{r}) \\ \mathrm{id}_{2j+1}(\boldsymbol{r}) &= (1-\sigma_j(\boldsymbol{r})) \ \times \mathrm{id}_j(\boldsymbol{r}), \end{aligned}$$

where $j \leq 2^d - 1$ and

$$\sigma_j(\boldsymbol{r}) \triangleq \frac{1}{1+e^{\boldsymbol{r}^T \boldsymbol{n}_j}}.$$

Due to the tree structure, three separating hyper-planes generate four regions, each corresponding to a leaf node on the tree given in Fig. 4. The partitioning is defined in a hierarchical manner, i.e., $\boldsymbol{r}(t)$ is first processed by $\sigma_1(\boldsymbol{r}(t))$ and then by $\sigma_i(t)$, $i = 2, 3$. A complete tree defines a doubly exponential number, $O(2^{2^d})$, of models each of which can be used to partition the space of the received signal vector. As an example, a depth-2 tree defines 5 different partitions as shown in Fig. 5, where each of these subtrees is constructed using the leaves and the nodes of the original tree.

Consider the fifth model in Fig. 5, i.e., P_5, where this partition consists of 4 disjoint regions each corresponding to a leaf node of the original complete tree in Fig. 4, labeled as 4, 5, 6, and 7. At each region, say the 4th region, we generate the estimate $\hat{b}_4(t) = \boldsymbol{w}_4^T(t)\boldsymbol{r}(t)$, where $\boldsymbol{w}_4(t) \in \mathbb{R}^h$ is the tap weights of the

linear equalizer assigned to region 4. Considering the hierarchical structure of the tree and having calculated the region estimates, $\hat{b}_j(t)$, the final estimate of P_5 is given by

$$\hat{b}^{(5)}(t) = \sum_{j=4}^{7} \mathrm{id}_j(\boldsymbol{r}(t))\hat{b}_j(t), \tag{13}$$

for an arbitrary selection of the separator functions $\sigma_1, \sigma_2, \sigma_3$ and for any $\boldsymbol{r}(t)$. We emphasize that any P_i, $i = 1, \ldots, 5$ can be used in a similar fashion to construct a piecewise linear channel equalizer. Based on these model estimates, the final estimate of the transmitted bit $b(t)$ is obtained by

$$\begin{aligned} \hat{b}(t) &= \sum_{i=1}^{\alpha_d} \hat{b}^{(i)}(t)\ u_i'(t) \\ &= \hat{\boldsymbol{b}}'(t)^T\ \boldsymbol{u}'(t), \end{aligned} \tag{14}$$

where $\hat{\boldsymbol{b}}'(t) \triangleq [\hat{b}^{(1)}(t), \ldots, \hat{b}^{(\alpha_d)}(t)]^T$ and $\hat{b}^{(k)}(t)$ represents the estimate of $b(t)$ generated by the kth piecewise linear channel equalizer, $k = 1, \ldots, \alpha_d$. We use the LMS algorithm to update the weighting vector $\boldsymbol{u}'(t)$. Note that in our method, which is given in Algorithm 1, we linearly combine the estimates generated by all α_d models, using the weighting vector $\boldsymbol{u}'(t) \triangleq [u_1'(t), \ldots, u_{\alpha_d}'(t)]^T$, to estimate the transmitted bit $b(t)$, such that we can achieve the best performance on the tree.

Under the moderate assumptions on the cost function that $e^2(\boldsymbol{u}'(t))$ is a λ-strong convex function [31] and also its gradient is upper bounded by a constant number, the following theorem provides an upper bound on the error performance of our algorithm (given in Algorithm 1).

Theorem 1: *Let* $\{b(t)\}_{t\geq 1}$ *and* $\{r(t)\}_{t\geq 1}$ *represents arbitrary and real-valued sequences of transmitted bits and channel outputs. The algorithm for* $\hat{b}(t)$ *given in Algorithm 1 when applied to any sequence with an arbitrary length* $L \geq 1$

yields

$$E\{\sum_{t=1}^{L}\Big(b(t)-\hat{b}(t)\Big)^2\}-\min_{\boldsymbol{z}\in\mathbb{R}^{\alpha_d}}E\{\sum_{t=1}^{L}\Big(b(t)-\boldsymbol{z}^T\hat{\boldsymbol{b}}(t)\Big)^2\}\leq$$
$$E\{\sum_{t=1}^{L}\Big(b(t)-\hat{b}(t)\Big)^2\}-E\{\min_{\boldsymbol{z}\in\mathbb{R}^{\alpha_d}}\sum_{t=1}^{L}\Big(b(t)-\boldsymbol{z}^T\hat{\boldsymbol{b}}(t)\Big)^2\}\leq O\Big(\log L\Big),\quad(15)$$

where $\boldsymbol{z}$ is an arbitrary constant combination weight vector, used to combine the results of all models.

Outline of the proof: *Since we use a stochastic gradient method to update the weighting vector in Algorithm 1, from Chapter 3 of [32] it can be straightforwardly shown that*

$$\sum_{t=1}^{L}\Big(b(t)-\hat{b}(t)\Big)^2-\min_{\boldsymbol{z}\in\mathbb{R}^{\alpha_d}}\sum_{t=1}^{L}\Big(b(t)-\boldsymbol{z}^T\hat{\boldsymbol{b}}(t)\Big)^2\leq O\Big(\log L\Big),$$

in a strong deterministic sense, which is a well known result in computational learning theory [32]. Taking the expectation of both sides of this deterministic bound yields the result in (15).

This theorem implies that the algorithm given in Algorithm 1 asymptotically achieves the performance of the optimal linear combination of the $O(2^{2^d})$ different "adaptive" piecewise linear equalizers, represented using a depth-d tree, in the MSE sense, with a computational complexity $O(h4^d)$ (i.e., only polynomial in the number of nodes).

Regarding this theorem, for $\alpha_d \approx (1.5)^{2^d}$ different models that are embedded within a depth-d tree, the introduced algorithm (given in Algorithm 1) asymptotically achieves the same cumulative squared error as the optimal combination of the best adaptive equalizers. Moreover, note that as the data length increases and each region becomes dense enough, the linear equalizer in each region, converges to the corresponding linear MMSE equalizer in that region [24]. In addition, since in our algorithm the tree structure is also adaptive, it can follow the data statistics effectively even when the channel is highly time

varying. Therefore, our algorithm outperforms the conventional methods and asymptotically achieves the performance of the best piecewise linear equalizer.

We update the combination weights using LMS algorithm to achieve the performance of the best piecewise linear equalizer. Hence,

$$\begin{aligned}\boldsymbol{u}'(t+1) &= \boldsymbol{u}'(t) - \frac{1}{2}\mu\nabla_{\boldsymbol{u}'(t)}e^2(t)\\ &= \boldsymbol{u}'(t) + \eta\ e(t)\ \hat{\boldsymbol{b}}(t).\end{aligned}$$

Note that, as depicted in Fig. 5, each model weight equals the sum of the weights assigned to its leaf nodes, hence we have

$$u'_k(t) = \sum_{i\in P_k} u_i(t),$$

which in turn results in the following node weights update algorithm

$$u_j(t+1) = u_j(t) + \mu\ e(t)\ \hat{b}_j(t)\ \mathrm{id}_j(\boldsymbol{r}(t)),$$

where $u_j(t)$ denotes the weight assigned to the jth node at time t.

So far we have shown how to construct a piecewise linear equalizer using separating functions and how to combine the estimates of all models to achieve the performance of the best piecewise linear equalizer. However, there are a doubly exponential number of these models, hence it is computationally prohibited to run all of these models and combine their results. In order to reduce this complexity while reaching exactly the same result, we directly combine the node estimates, i.e., instead of running all possible models, we combine the node estimates with special weights, which yields the same result. We now illustrate how to directly combine the node weights in our algorithm.

The overall estimate using all models contributions is

$$\begin{aligned}\hat{b}(t) &= \sum_{i=1}^{\alpha_d} \hat{b}^{(i)}(t)\ u_i'(t) \\ &= \sum_{i=1}^{\alpha_d} \hat{b}^{(i)}(t) \left(\sum_{j \in P_i} u_j(t) \right) \\ &= \sum_{i=1}^{\alpha_d} \left(\sum_{k \in P_i} \mathrm{id}_k(\boldsymbol{r}(t))\ \hat{b}_k(t) \right) \left(\sum_{j \in P_i} u_j(t) \right) \\ &= \sum_{i=1}^{\alpha_d} \left(\sum_{j,k \in P_i} \mathrm{id}_k(\boldsymbol{r}(t))\ \hat{b}_k(t) u_j(t) \right), \end{aligned} \tag{16}$$

where j and k indicate two arbitrary nodes. For each node k, we define $z_k(t) \triangleq \mathrm{id}_k(\boldsymbol{r}(t))\ \hat{b}_k(t)$. Hence we have

$$\hat{b}(t) = \sum_{i=1}^{\alpha_d} \left(\sum_{j,k \in P_i} z_k(t) u_j(t) \right).$$

Consider that $\Gamma = \{\Gamma_1, \ldots, \Gamma_{\theta_d(d_k)}\}$ is the family of models (subtrees) in all of which the node k is a leaf node,where $\theta_d(d_k)$ denotes the number of such models. Therefore the final estimate of our algorithm can be rewritten as:

$$\hat{b}(t) = \sum_{k=1}^{2^{d+1}-1} z_k(t) \left[\sum_{j \in \Gamma_1} u_j(t) + \cdots + \sum_{j \in \Gamma_{\theta_d(d_k)}} u_j(t) \right].$$

We denote by $\rho(j_0, k)$ the number of models in all of which the nodes j_0 and k appear as the leaf nodes simultaneously. The weight of each node j_0 (i.e., u_{j_0}) appears in the above expression exactly $\rho(j_0, k)$ times, which yields the following expression for the final estimate

$$\hat{b}(t) = \sum_{k=1}^{2^{d+1}-1} z_k(t)\ \beta_k(t),$$

where

$$\beta_k(t) \triangleq \sum_{j_0=1}^{2^{d+1}-1} u_{j_0}(t) \rho(j_0, k).$$

We now illustrate how to calculate $\rho(j,k)$ in a depth-d tree. We use $\theta_d(d_j)$ to denote the number of models extracted from a depth-d tree, in all of which j is a leaf node. It can be shown that

$$\theta_d(d_j) = \prod_{l=1}^{d_j} \alpha_{d-l},$$

where $d_j = \lfloor \log_2(j) \rfloor$ denotes the depth of the jth node [16]. To calculate $\rho(j,k)$ we first note that $\rho(j,k) = \rho(k,j)$ and $\rho(j,j) = \theta_d(j)$. Therefore we obtain

$$\rho(j,k) = \begin{cases} \theta_d(j) & \text{if } j = k \\ \frac{\theta_{d-l-1}(d_k - l - 1)}{\alpha_{d-l-1}} \theta_d(d_j) & \text{if } j \neq k, \end{cases}$$

where, l represents the depth of the nearest common ancestor of the nodes j and k in the tree, i.e., an ancestor of both nodes j and k, none of the children of that is a common ancestor of j and k. This parameter can be calculated using the following algorithm. Assume that, without loss of generality, $j \leq k$. Obviously if j is an ancestor of k, it is also the nearest common ancestor, i.e., $l = d_j$. However, if j is not an ancestor of k, we define $j' \triangleq 2^{d_k - d_j} j$, which is a grandchild of the node j. Hence, the nearest common ancestor of j' and k is that of j and k. The following procedure computes the parameter l.

$l = 0$;
$\delta = d_k$;
while *($l \leq d_k$)* **do**
  $\delta = \delta - l$;
  if $(j', k \leq 2^{\delta-1} + 2^{\delta}$ *or* $j', k \geq 2^{\delta-1} + 2^{\delta})$ **then**
    $l = l + 1$;
  else
    stop;
  end
end

In order to update the region boundaries, we update their direction vectors as follows

$$\boldsymbol{n}_j(t+1) = \boldsymbol{n}_j(t) - \frac{1}{2}\mu \nabla_{\boldsymbol{n}_j(t)} e^2(t), \tag{17}$$

where $\nabla_{\boldsymbol{n}_j(t)} e^2(t)$ is the derivative of $e^2(t)$ with respect to $\boldsymbol{n}_j(t)$. Since $e(t) = b(t) - \hat{b}(t)$ the updating expression can be calculated as follows

$$\begin{aligned}
\boldsymbol{n}_j(t+1) &= \boldsymbol{n}_j(t) - \frac{1}{2}\mu \nabla_{\boldsymbol{n}_j(t)} e^2(t) \\
&= \boldsymbol{n}_j(t) + \mu\ e(t)\ \frac{\partial\ \hat{b}(t)}{\partial\ \boldsymbol{n}_j(t)} \\
&= \boldsymbol{n}_j(t) + \mu\ e(t)\ \sum_{k=1}^{2^{d+1}-1} \frac{\partial\ \hat{b}(t)}{\partial\ z_k(t)} \frac{\partial\ z_k(t)}{\partial\ \boldsymbol{n}_j(t)} \\
&= \boldsymbol{n}_j(t) + \mu\ e(t)\ \sum_{k=1}^{2^{d+1}-1} \beta_k(t)\ \hat{b}_k(t) \frac{\partial\ \mathrm{id}_k(\boldsymbol{r}(t))}{\partial\ \boldsymbol{n}_j(t)} \\
&= \boldsymbol{n}_j(t) + \mu\ e(t)\ \sum_{k=1}^{2^{d+1}-1} \beta_k(t)\ \hat{b}_k(t) \frac{\partial\ \mathrm{id}_k(\boldsymbol{r}(t))}{\partial\ \sigma_j(\boldsymbol{r}(t))} \frac{\partial\ \sigma_j(\boldsymbol{r}(t))}{\partial\ \boldsymbol{n}_j(t)} \\
&= \boldsymbol{n}_j(t) + \mu\ e(t)\ \frac{\partial\ \sigma_j(\boldsymbol{r}(t))}{\partial\ \boldsymbol{n}_j(t)} \sum_{k=1}^{2^{d+1}-1} \beta_k(t)\ \hat{b}_k(t) \frac{\partial\ \mathrm{id}_k(\boldsymbol{r}(t))}{\partial\ \sigma_j(\boldsymbol{r}(t))}.
\end{aligned}$$

However note that not all of the $\mathrm{id}_k(\boldsymbol{r}(t))$ functions involve $\sigma_j(\boldsymbol{r}(t))$, i.e., only the nodes of the subtree with the root node j are included. Hence,

$$\begin{aligned}
\sum_{k=1}^{2^{d+1}-1} \beta_k(t)\ \hat{b}_k(t) \frac{\partial\ \mathrm{id}_k(\boldsymbol{r}(t))}{\partial\ \sigma_j(\boldsymbol{r}(t))} &= \sum_{m=1}^{d-d_j} \sum_{s=0}^{2^{m+1}-1} \beta_{2^m j+s}(t)\ \hat{b}_{2^m j+s}(t) \frac{\partial\ \mathrm{id}_{2^m j+s}(\boldsymbol{r}(t))}{\partial\ \sigma_j(\boldsymbol{r}(t))} \\
&= \sum_{m=0}^{d-d_j-1} \Bigg(\sum_{s=0}^{2^m-1} \beta_{2^{m+1} j+s}(t)\ \hat{b}_{2^{m+1} j+s}(t) \frac{\mathrm{id}_{2^{m+1} j+s}(\boldsymbol{r}(t))}{\sigma_j(\boldsymbol{r}(t))} \\
&\qquad - \sum_{s=2^m}^{2^{m+1}-1} \beta_{2^{m+1} j+s}(t)\ \hat{b}_{2^{m+1} j+s}(t) \frac{\mathrm{id}_{2^{m+1} j+s}(\boldsymbol{r}(t))}{\sigma_j(\boldsymbol{r}(t))} \Bigg).
\end{aligned}$$

We have presented the algorithm 1 for a "turning boundaries tree" equalizer, which is completely adaptive to the channel response. Especially in our algorithm both the number of regions and the region boundaries as well as the linear equalizers in each region are adaptive. We emphasize that the learning rates and initial values of all filters can be different.

3.3 Complexity

Consider that we use a depth-d tree to partition the space of the received signal, $\mathbb{R}^h$. First note that each node estimate needs h computation. Since we update all the linear filters corresponding to each region at each specific time, it generates a computational complexity of $O(h(2^{d+1}-1))$. Also, updating the separator functions results in a computational complexity of $O(h(2^d-1))$. Moreover, note that we compute the cross-correlation of every node estimate and every node weight, which results in the complexity of $O(hN_d^2) = O(h4^d)$. Hence our algorithm has the complexity $O(h4^d)$ which is only polynomial in the number of the tree nodes.

From the construction of this algorithm, it can be straightforwardly shown that the algorithm is completely adaptive such that it converges to the optimal linear filters in every region and optimal partition. Therefore, the proposed equalizer achieves the performance of the best linear combination of all possible piecewise linear equalizers embedded in a depth-d tree, with a complexity only polynomial in the number of tree nodes.

Compute $\rho(j,k)$ for all pairs $\{j,k\}$ of nodes;
for $t = 1$ *to* L **do**
  $\boldsymbol{r} = [r(t), \ldots, r(t-h+1)]^T$;
  for $k = 1$ *to* $2^d - 1$ **do**
    $\sigma_k = \frac{1}{1+e^{\boldsymbol{r}^T \boldsymbol{n}_k}}$;
  end
  $id_1 = 1$;
  for $k = 1$ *to* $2^d - 1$ **do**
    $id_{2k} = id_k \sigma_k$;
    $id_{2k+1} = id_k(1 - \sigma_k)$;
  end
  $\hat{b} = 0$;
  for $k = 1$ *to* $2^{d+1} - 1$ **do**
    $\hat{b}_k = \boldsymbol{w}_k^T \boldsymbol{r}$;
    $z_k = \hat{b}_k id_k$;
    $\beta_k = 0$;
    for $j = 1$ *to* $2^{d+1} - 1$ **do**
      $\beta_k = \beta_k + u_j \ \rho(j,k)$;
    end
    $\hat{b}(t) = \hat{b}(t) + z_k \beta_k$;
  end
  if *train mode* **then**
    $\bar{b} = b(t)$;
  else
    $\bar{b} = Q(\hat{b}(t))$;
  end
  $e = \bar{b} - \hat{b}(t)$;
  for $k = 1$ *to* $2^{d+1} - 1$ **do**
    $\boldsymbol{w}_k = \boldsymbol{w}_k + \mu_k \ e \ id_k \boldsymbol{r}$;
    $u_k = u_k + \eta \ e \ z_k$;
  end
  for $j = 1$ *to* $2^d - 1$ **do**
    $d_j = \lfloor \log_2(j) \rfloor$;
    for $m = 0$ *to* $d - d_j - 1$ **do**
      for $p = 0$ *to* $2^m - 1$ **do**
        $i = 2^{m+1} j + p$;
        $S_1 = S_1 + \beta_i \ \hat{b}_i \ \frac{id_i}{\sigma_j}$;
      end
      for $p = 2^m$ *to* $2^{m+1} - 1$ **do**
        $i = 2^{m+1} j + p$;
        $S_2 = S_2 + \beta_i \ \hat{b}_i \ \frac{id_i}{\sigma_j}$;
      end
      $S = S + S_1 - S_2$;
    end
    $\boldsymbol{n}_j = \boldsymbol{n}_j + \zeta_j \ e \ \sigma \ (\sigma - 1) \ S \ \boldsymbol{r}$;
  end
end

Remark: Although we have introduced our equalization method in a single carrier framework, one can see that is directly extended to the OFDM framework as well. For this purpose, one can use a tree-based piecewise linear equalizer for each subcarrier in the OFDM modulation, which will improve the performance in highly nonstationary underwater acoustic channels. Furthermore, our method can be straightforwardly used in MIMO communications, i.e., one can embed all of the received symbols from all of the outputs in one vector, and then apply the proposed piecewise linear equalizer to them.

4 Simulations

In this section, we illustrate the performance of our algorithm under a highly realistic UWA channel equalization scenario. The UWA channel response is generated using the algorithm introduced in [29], which presents highly accurate modeling of the real life UWA communication experiments as illustrated in [29]. Particularly, the surface height is set to 100m, transmitter antenna is placed at the height of 20m, the receiver antenna is placed at the height of 50m, and the channel distance is 1000m. We compare the performances of the following equalization algorithms: the Turning Boundaries Tree (TBT) equalizer of Theorem 1, the Fixed Boundaries Tree (FBT) equalizer of [16], Finest Partition with Fixed Boundaries (FF), Finest Partition with Turning Boundaries (FT) (all having depths $d = 2$), and the linear LMS equalizer. We have compared the performance of our algorithm with the context tree weighting (CTW), as shown in Fig. 7 and 9. The Finest Partition refers to the partition consisted of all leaf nodes of the tree (the P_5 model in Fig. 5). Also, we use FBT to refer to an equalizer with fixed boundaries, which adaptively update the node weights as well as TBT algorithm. We use the LMS algorithm in the linear equalizer of each node for all algorithms, and the step sizes are set to

$\mu = 0.01$ for all equalizers algorithms. In all algorithms we have used length 362 equalizers.

We sent 1000 repeated Turyn sequences [33] (of 28 bits) over the simulated UWA channel. Fig. 8 shows the normalized time accumulated squared errors of the equalizers, when SNR = 30dB. We emphasize that the TBT equalizer significantly outperforms its competitors, where the FBT equalizer cannot provide a satisfactory result since it commits to the initial partitioning structure. Moreover, the linear equalizer yields unacceptable results due to the structural commitment to the linearity. Note that the TBT equalizer adapts its region boundaries and can successfully perform channel equalization even for a highly difficult UWA channel. The Fig. 6 shows the bit error rate performance in different SNRs for different equalizers. In the Fig. 10, it is shown that the boundaries are turning during TBT algorithm, which results in an adaptive partition. The results are averaged over 10 repetitions, and show the extremely superior performance of our algorithm over other methods.

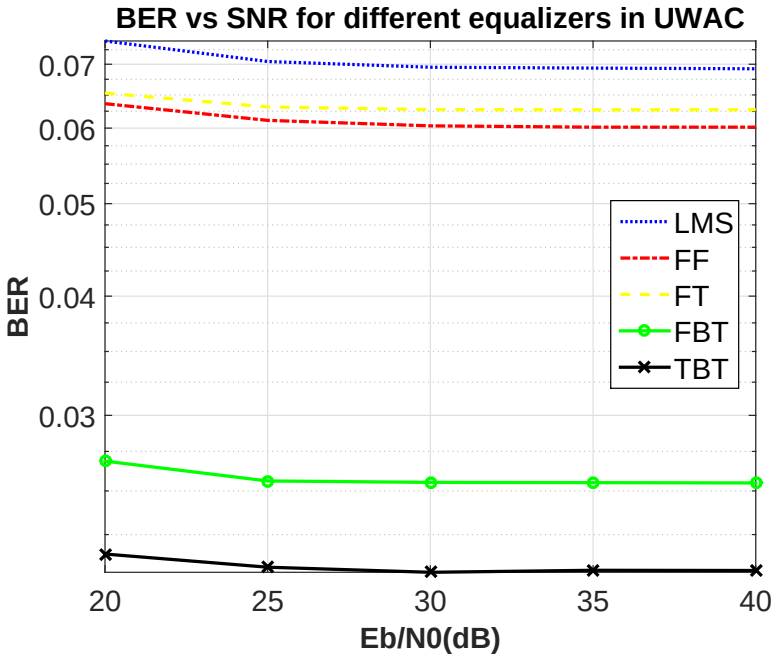

Fig. 6. BER performances for the UWA channel response generated by [29]. The BERs for the TBT, FBT, FF and FT equalizers (all using depth-2 tree structure), and for the linear equalizer are presented.

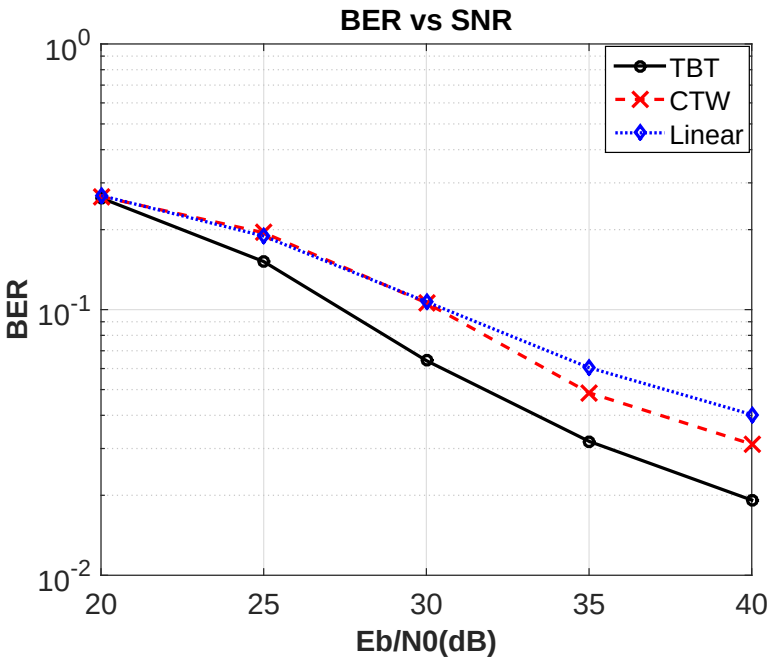

Fig. 7. BER performances for the UWA channel response generated by [29]. The BERs for the TBT and CTW equalizers (both using depth-2 tree structure), and for the linear equalizer are presented.

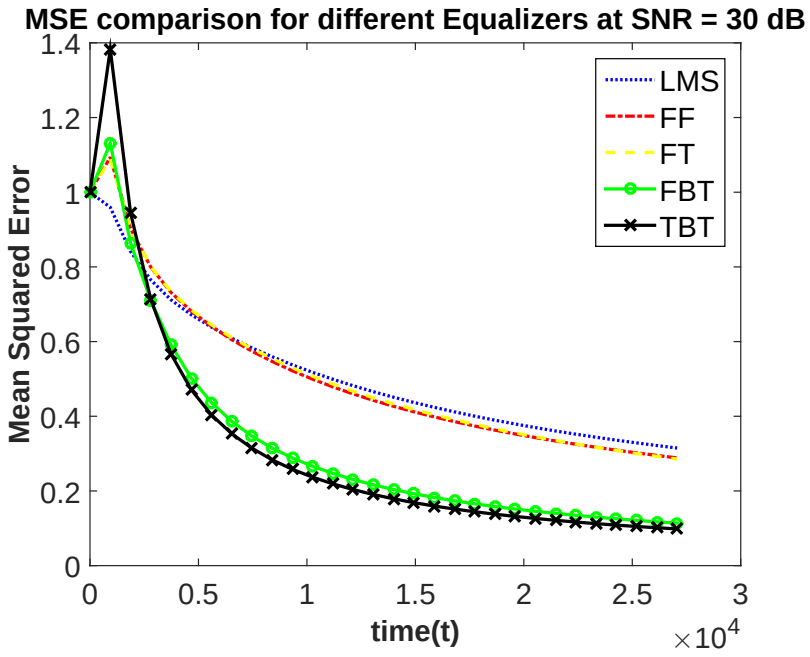

Fig. 8. Squared error performances for the UWA channel response generated by [29] for SNR = 30dB. The time accumulated normalized squared errors for the TBT, FBT, FF, and FT equalizers (all using depth-2 tree structure), and for the linear equalizer are presented.

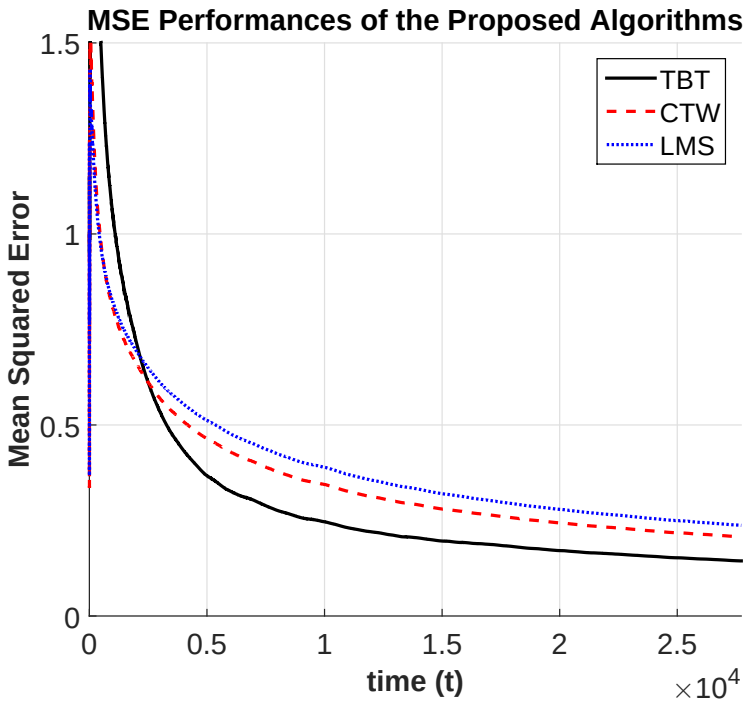

Fig. 9. Squared error performances for the UWA channel response generated by [29] for SNR = 30dB. The time accumulated normalized squared errors for the TBT and CTW equalizers (both using depth-2 tree structure), and for the linear equalizer are presented.

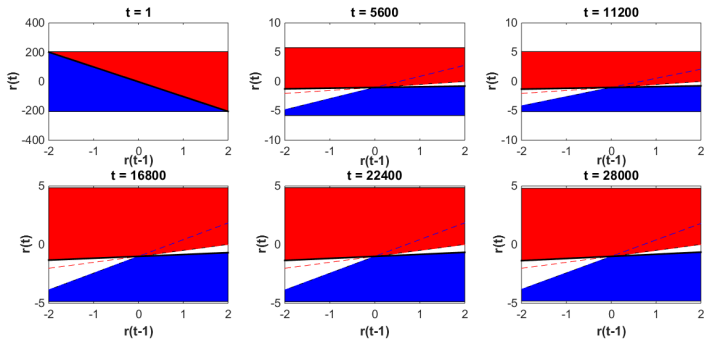

Fig. 10. An adaptive partition using a depth 2 tree. The region boundaries are changing during the TBT algorithm converging to the optimal partition. In this experiment, SNR = 30dB

In the second experiment we sent 10000 repeated Turyn sequence over the simulated channel, and used TBT algorithm with different depths to equalize the channel. The results, as shown in Fig. 11 and 12, demonstrate that increasing the depth of the tree improves the performance. However, as the depth of the

tree increases, the effect of the depth diminishes. This is because increasing the depth introduces finer partitions, i.e., the partitions with more regions. As the number of the regions in a partition increases, the data congestion in each region decreases, hence, the linear filters in these regions cannot fully converge to their MMSE solutions. As a result, the estimates of these regions (nodes) will be contributed to the final estimate with a much lower combination weight than other nodes, which are also present in a lower depth tree. Therefore, although increasing the depth of the tree improves the result, we cannot get a significant improvement in the performance by only increasing the depth.

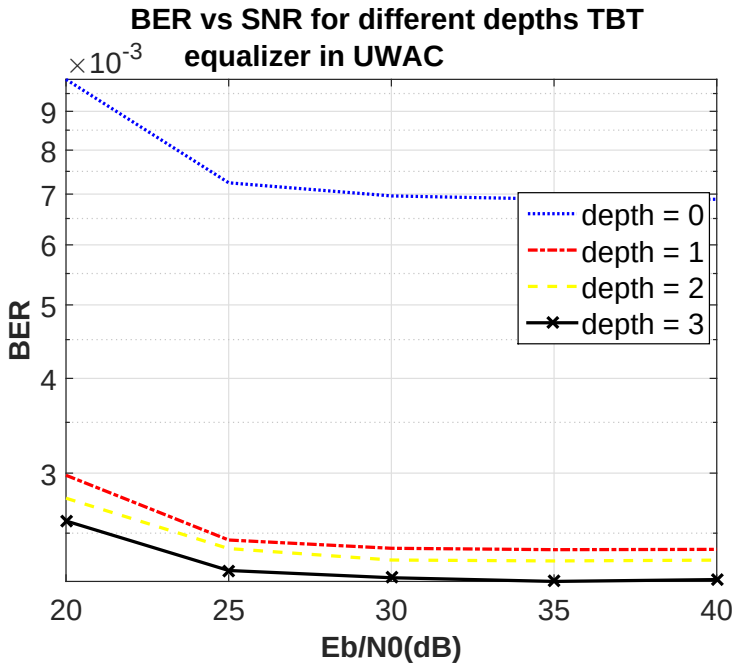

Fig. 11. BER performances for different depths TBT equalizers. This figure shows that increasing the depth of the tree improves the BER performance.

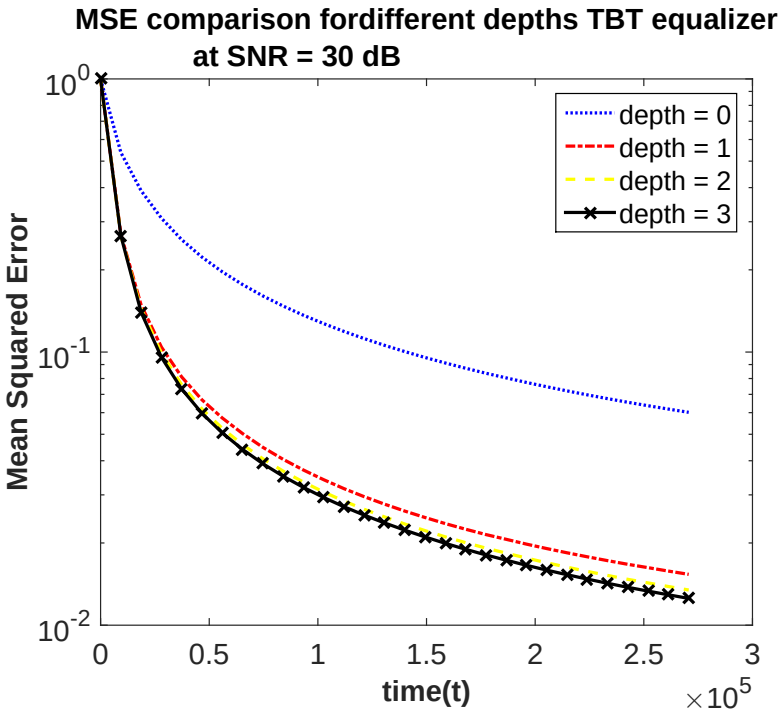

Fig. 12. Squared error performances for different depths TBT equalizers for SNR = 30dB. This figure shows that increasing the depth of the tree improves the MSE performance. However, as the depth increases this effect diminishes.

Also, the node combination weights in the second experiment are shown in Fig. 13. This figure shows that node 1, the root node, has the largest weight, which means that it has the most contribution to the final estimate. Note that for an arbitrarily chosen parent node, a larger portion of the data is used to train the linear filter assigned to that node compared to its children nodes, which in turn, yields a better convergence for the parent node's filter. Hence, the contribution of the parent node to the final estimation is more than that of the children nodes. As a result the weight of each node is greater than both the weights of its left and right children, e.g., node 2 has a greater weight than node 4 and 5.

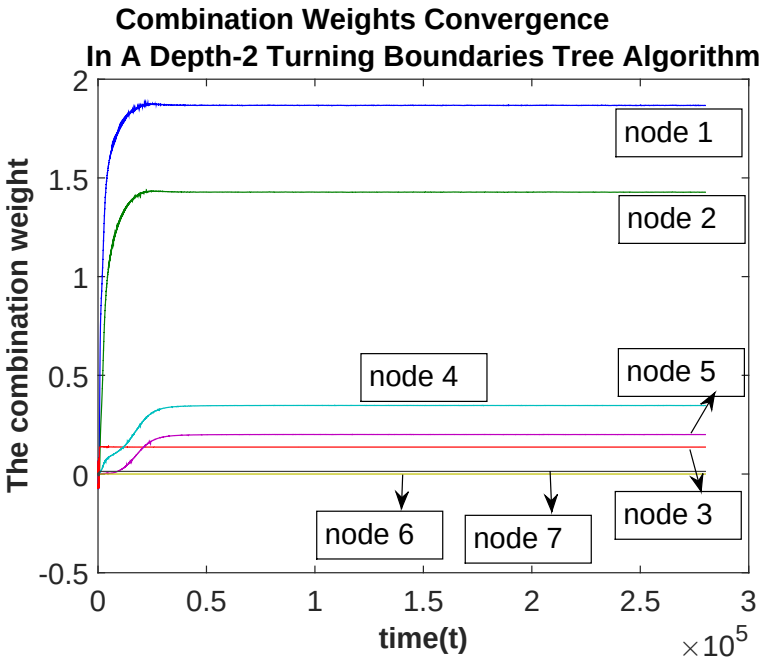

Fig. 13. The node combination weights in TBT algorithm. In this experiment, SNR = 30dB. Each node has a greater weight than its children, after convergence.

5 Conclusion

We study nonlinear UWA channel equalization using hierarchical structures, where we partition the received signal space using a nested tree structure and use different linear equalizers in each region. In this framework, we introduce a tree based piecewise linear equalizer that both adapts its linear equalizers in each region as well as its tree structure to best match to the underlying channel response. Our algorithm asymptotically achieves the performance of the best linear combination of a doubly exponential number of adaptive piecewise linear equalizers represented on a tree with a computational complexity only polynomial in the number of tree nodes. Since our algorithm directly minimizes the squared error and avoid using any artificial weighting coefficients, it strongly outperforms the conventional linear and piecewise linear equalizers as shown in our experiments.

6 Acknowledgements

This work is in part supported by Turkish Academy of Sciences, Outstanding Researcher Programme and TUBITAK, Contract No:112E161.

References

[1] B. Xerri, J.-F. Cavassilas, B. Borloz, Passive tracking in underwater acoustic, Signal Process. 82 (8) (2002) 1067–1085. doi:10.1016/S0165-1684(02)00240-2.
URL http://dx.doi.org/10.1016/S0165-1684(02)00240-2

[2] A. C. Singer, J. K. Nelson, S. S. Kozat, Signal processing for underwater acoustic communications, IEEE Communications Magazine 47 (1) (2009) 90–96.

[3] D. B. Kilfoyle, A. B. Baggeroer, The state of the art in underwater acoustic telemetry, Oceanic Engineering, IEEE Journal of 25 (1) (2000) 4–27. doi:10.1109/48.820733.

[4] J. W. Choi, T. J. Riedl, K. Kim, A. C. Singer, J. C. Preisig, Adaptive linear turbo equalization over doubly selective channels, IEEE Journal of Oceanic Engineering 36 (4) (2011) 473–489.

[5] M. Stojanovic, Recent advances in high-speed underwater acoustic communications, IEEE Journal of Oceanic Engineering 21 (2) (1996) 125–136.

[6] J. C. Patra, P. K. Meher, G. Chakraborty, Nonlinear channel equalization for wireless communication systems using legendre neural networks, Signal Process. 89 (11) (2009) 2251–2262. doi:10.1016/j.sigpro.2009.05.004.

[7] H. Zhang, Y. Shi, A. Saadat Mehr, H. Huang, Robust fir equalization for time-varying communication channels with intermittent observations via an lmi approach, Signal Process. 91 (7) (2011) 1651–1658.

[8] C. P. Shah, C. C. Tsimenidis, B. S. Sharif, J. A. Neasham, Low complexity iterative receiver design for shallow water acoustic channels., EURASIP Journal on Advances in Signal Processing 2010.

[9] N. Iqbal, A. Zerguine, N. Al-Dhahir, Decision feedback equalization using particle swarm optimization, Signal Processing 108 (Complete) (2015) 1–12. doi:10.1016/j.sigpro.2014.07.030.

[10] Y. Zhang, J. Zhao, Y. Guo, J. Li, Blind adaptive mmse equalization of underwater acoustic channels based on the linear prediction method, Journal of Marine Science and Application 10 (1) (2011) 113–120. doi:10.1007/s11804-011-1050-9.

[11] S. Fki, M. Messai, A. Aissa El Bey, T. Chonavel, Blind equalization based on pdf fitting and convergence analysis, Signal processing 101 (2014) 266 – 277.

[12] J. Huang, J. Huang, C. Berger, S. Zhou, P. Willett, Iterative sparse channel estimation and decoding for underwater mimo-ofdm, EURASIP Journal on Advances in Signal Processing 2010.

[13] A. Radosevic, T. Duman, J. Proakis, M. Stojanovic, Channel prediction for adaptive modulation in underwater acoustic communications, OCEANS, 2011 IEEE - Spain (2011) 1–5doi:10.1109/Oceans-Spain.2011.6003438.

[14] B. Li, S. Zhou, M. Stojanovic, L. Freitag, P. Willett, Multicarrier communication over underwater acoustic channels with nonuniform doppler shifts, IEEE Journal of Oceanic Engineering 33 (2) (2008) 198–209.

[15] M. Stojanovic, J. Preisig, Underwater acoustic communication channels: Propagation models and statistical characterization, IEEE Communications Magazine 47 (1) (2009) 84–89.

[16] N. D. Vanli, S. S. Kozat, A comprehensive approach to universal piecewise nonlinear regression based on trees, IEEE Transactions on Signal Processing 62 (20) (2014) 5471–5486.

[17] J. Tao, Y. Zheng, T. Yang, W.-B. Yang, Channel equalization for single carrier

mimo underwater acoustic communications, EURASIP Journal on Advances in Signal Processing 2010. doi:10.1155/2010/281769.

[18] K. Kim, N. Kalantarova, S. S. Kozat, A. C. Singer, Linear MMSE-optimal turbo equalization using context trees, IEEE Transactions on Signal Processings 61 (12) (2013) 3041–3055.

[19] H. C. Myburgh, J. C. Olivier, Low complexity mlse equalization in highly dispersive rayleigh fading channels., EURASIP Journal on Advances in Signal Processing 2010.

[20] Y. Xiao, F. liang Yin, Blind equalization based on rls algorithm using adaptive forgetting factor for underwater acoustic channel, China Ocean Engineering 28 (3) (2014) 403–408. doi:10.1007/s13344-014-0032-5.

[21] C. P. Shah, C. C. Tsimenidis, B. S. Sharif, J. A. Neasham, Iterative equalization for underwater acoustic channels using bit interleaved coded modulation and decision feedback equalization, OCEANS 2009 - EUROPE (2009) 1 – 5doi:10.1109/OCEANSE.2009.5278176.

[22] S.-J. Hwang, P. Schniter, Efficient multicarrier communication for highly spread underwater acoustic channels, Selected Areas in Communications, IEEE Journal on 26 (9) (2008) 1674 – 1683. doi:10.1109/JSAC.2008.081207.

[23] C. P. Shah, C. C. Tsimenidis, B. S. Sharif, J. A. Neasham, Low-complexity iterative receiver structure for time-varying frequency-selective shallow underwater acoustic channels using bicm-id: Design and experimental results, Oceanic Engineering, IEEE Journal of 36 (3) (2011) 406–421. doi:10.1109/JOE.2011.2144670.

[24] S. S. Kozat, A. C. Singer, G. C. Zeitler, Universal piecewise linear prediction via context trees, IEEE Transactions on Signal Processing 55 (7) (2007) 3730–3745.

[25] O. J. J. Michel, A. O. Hero, A.-E. Badel, Tree-structured nonlinear signal modeling and prediction, IEEE Transactions on Signal Processing 47 (11) (1999) 3027–3041. doi:10.1109/78.796437.

[26] S. Gelfand, C. Ravishankar, E. Delp, Tree-structured piecewise linear adaptive equalization, Communications, IEEE Transactions on 41 (1) (1993) 70–82. doi:10.1109/26.212367.

[27] P. van Walree, T. Jenserud, S. Smedsrud, A discrete-time channel simulator driven by measured scattering functions, Selected Areas in Communications, IEEE Journal on 26 (9) (2008) 1628 – 1637. doi:10.1109/JSAC.2008.081203.

[28] A. H. Sayed, Fundamentals of Adaptive Filtering, John Wiley & Sons, NJ, 2003.

[29] P. Qarabaqi, M. Stojanovic, Statistical characterization and computationally efficient modeling of a class of underwater acoustic communication channels, IEEE Journal of Oceanic Engineering 38 (4) (2013) 701–717.

[30] F. M. J. Willems, Y. M. Shtarkov, T. J. Tjalkens, The context-tree weighting method: basic properties, IEEE Transactions on Information Theory 41 (3) (1995) 653–664. doi:10.1109/18.382012.

[31] E. Hazan, A. Agarwal, S. Kale, Logarithmic regret algorithms for online convex optimization, Machine Learning 69 (2-3) (2007) 169–192.

[32] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning, and Games, Cambridge University Press, New York, NY, USA, 2006.

[33] R. J. Turyn, Sequences with small correlation, In Error Correcting Codes: Proceedings of a Symposium (1968) 195–228.