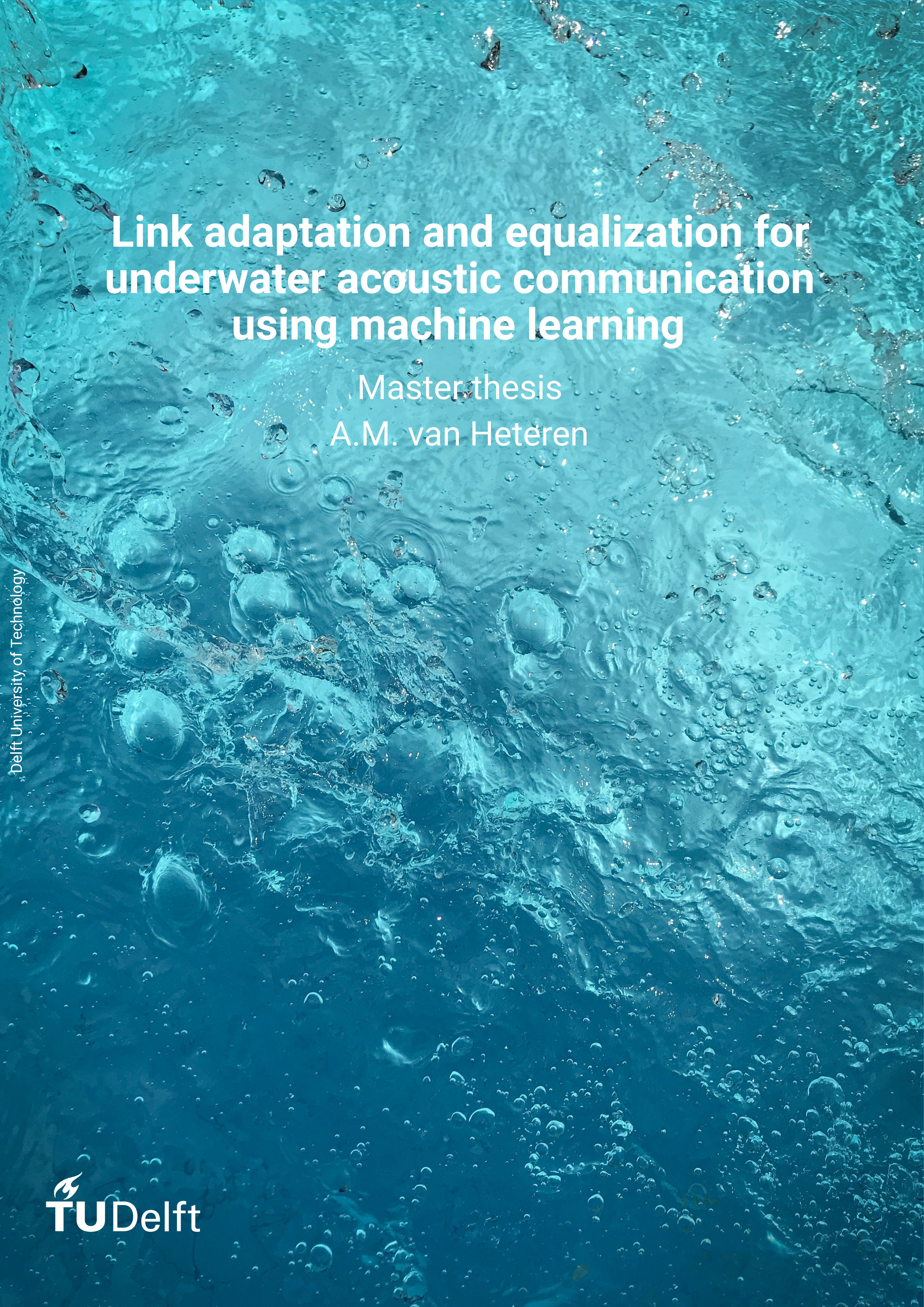


Link adaptation and equalization for underwater acoustic communication using machine learning

Master thesis

A.M. van Heteren

Delft University of Technology

TU Delft

Link adaptation and equalization for underwater acoustic communication using machine learning

by

A.M. van Heteren

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday June 17, 2022 at 14:00 PM.

Student number:	4712145	
Project duration:	November 8, 2022 – June 17, 2022	
Thesis committee:	Prof. dr. ir. G. J. T. Leus,	TU Delft, supervisor
	Dr. ir. H. Jamali-Rad,	TU Delft, Shell
	Dr. ir. K. C. H. Blom,	TNO

Abstract

The underwater acoustic environment is amongst the most challenging mediums for wireless communications. The three distinct challenges of underwater acoustic communication are the low and nonuniform propagation speed, frequency-dependent attenuation and time-varying multipath propagation.
To cope with these challenges, physical layer communication systems allow the selection of communication parameters based on environmental conditions and constraints. This is also known as link adaptation. In this thesis, the frequency-repetition spread-spectrum (FRSS) physical layer is studied. Various channel parameters are used to classify the optimal FRSS format. Furthermore, different machine learning classifiers are implemented to solve the classification problem. It is determined that the output signal-to-noise ratio provides enough information to switch effectively between transmission formats. Among the implemented machine learning classifiers, the decision tree strikes a good balance between performance and computational complexity. It is shown that a small performance gain can be achieved when custom channel parameters are extracted from the estimated impulse response and the equalizer error sequence using a deep neural network.
Optimizing the equalization process is another method to better cope with difficult environmental conditions. Various adaptive filter algorithms are implemented for the decision feedback equalizer used in the FRSS receiver. The optimal algorithm parameters are found by means of algorithm unrolling. It is shown that the standard least mean squares algorithm cannot be outperformed by various other optimization algorithms that use linear or non-linear filters.
The Watermark channel simulator is used to study the performance of the link adaptation and equalization optimization solutions for a wide range of underwater channels.

Preface

After working on my master thesis for seven months, it has finally come to an end. Several people helped me in the process of making this thesis and I would like to thank them. First of all, I would like to thank Koen Blom. Koen, I enjoyed your enthusiastic guidance during the project. You gave me the opportunity to do my research in underwater communications and you always took the time to answer my questions. I would also like to thank Geert Leus. Your comments and feedback during the progress sessions helped me in improving my thesis. An acknowledgement to Henry Dol, Mario Coutiño, and the other colleagues at TNO is also in place. Thank you for your valuable ideas and comments regarding my thesis. Lastly, I would like to thank my wife, Jeanne, for her mental support during this project. You are always there for me and I cannot thank you enough for that.

A.M. van Heteren
Zwijndrecht, June 2022

Contents

Acronyms

AI	artificial intelligence
AMC	adaptive modulation and coding
ANN	artificial neural network
AWGN	additive white Gaussian noise
BER	bit error rate
CART	classification and regression tree
CE	cross-entropy
CIR	channel impulse response
CRC	cyclic redundancy check
DFE	decision feedback equalizer
DFT	discrete Fourier transform
DL	deep learning
DNN	deep neural network
DSSS	direct sequence spread spectrum
DT	decision tree
FRSS	frequency-repetition spread-spectrum
ICI	inter-carrier interference
IDFT	inverse discrete Fourier transform
ISI	inter-symbol interference
KLMS	kernel LMS
LA	link adaptation
LMS	least mean squares
LOS	line-of-sight
MCS	modulation and coding scheme
MCSS	multi-carrier spread spectrum
MFSK	multiple frequency-shift keying
ML	machine learning
MSE	mean squared error
MSFRSS	multi-stream FRSS
NLMS	normalized LMS
NN	nearest neighbours
OFDM	orthogonal frequency division multiplexing
PAM	pulse amplitude modulation
PDR	packet delivery ratio
PLL	phase-locked loop
PSD	power spectral density
PSK	phase-shift keying
QAM	quadrature amplitude modulation
QPSK	quadrature phase-shift keying

RL	reinforcement learning
RLS	recursive least squares
RMS	root mean square
RNN	recurrent neural network
SINR	signal-to-interference-plus-noise ratio
SNR	signal-to-noise ratio
SPL	sound pressure level
SSP	sound speed profile
SVM	support vector machine
UAV	underwater autonomous vehicle
VS-LMS	variable step size LMS

1 Introduction

Nowadays, wireless communication systems are embedded in everyday life. Applications like broadcast radio, Bluetooth technology, satellite communication or Wi-Fi all rely on wireless communication techniques. These applications use radio waves to transmit signals through the air. Communication methods are implemented for environments ranging from closed indoor environments up to wide outdoor environments. Many different signal processing tools and communication methods have been developed over the years [1–3]. However, current wireless communication methods are mainly focused on terrestrial applications.
Another communication medium is water. Since a large portion of the surface of the earth is covered in water, this offers abundant space for all kinds of interesting applications for wireless communications. Nonetheless, the underwater acoustic channel is seen as being amongst the most challenging communication media currently used [4]. In the past years, applications for acoustic underwater communication have been gaining interest. Both military and civil projects have been initiated. Examples include underwater monitoring of the environment for maintenance or security reasons. Communication networks between unmanned vehicles, underwater nodes and ships for challenging or dangerous environments are extensively explored as well [5–7].

1.1. Underwater channel

The choice of using acoustics for underwater communication is motivated by the fact that electromagnetic waves attenuate faster than acoustic waves in water. The three distinct challenges of the underwater channel are low sound propagation speed, frequency-dependent attenuation and time-varying multipath propagation [8].
The propagation speed of sound in water is low compared to radio waves. Furthermore, it is not constant, but dependent on the depth, temperature and salinity of the water. The transmission loss of acoustic waves in water depends on distance, as well as frequency. Using longer distances and higher frequencies both result in a higher transmission loss.
Due to the low propagation speed of sound in water, the time between the arrival of the transmitted signal and its latest reflection can be in the order of tens or hundreds of milliseconds, which is huge compared to radio communications. As a consequence, distortion between transmitted signals at the receiver can be experienced. Due to the multipath channel, signals that follow different paths in the underwater environment add constructively or destructively. Therefore, frequency fades can occur. This results in some frequencies being attenuated, while others are not. Also, due to moving transducers and water dynamics, each path can have a different Doppler shift, resulting in a Doppler spread on the received signal.
In addition, due to time variations in the channel, the channel constantly changes, leading to time-varying frequency-selectivity. In practice, the delay and Doppler spreads are the limiting factors for underwater communication, as opposed to noise limitations. To cope with these limitations, transducers communicate by using both smart signal design and equalization techniques. This is done to reduce the channel effect on communication performance.

1.2. Machine learning for wireless communication

Although more and more advanced signal processing techniques for communication have been designed over the years, improvements can still be made. Recently, artificial intelligence (AI) has become more and more popular. In AI, intelligent agents are assigned to solve hard problems, instead of humans. In this thesis, AI will be used to solve communication problems. More specifically, machine learning (ML) will be used. Machine learning is a subfield of artificial intelligence and deep learning (DL) is a subfield of machine learning. Nowadays, ML is used for a large variety of scenarios. For example, language processing, image classification and computer vision.

Recently, interest in applying ML and DL to wireless communication problems has increased. Applying these methods can provide an increase in throughput and performance. An interesting example of a DL concept applied to a communication system is an end-to-end channel autoencoder [9]. A channel autoencoder is a concatenation of two neural networks, one for the transmitter side and one for the receiver side. Various reasons can be mentioned on why DL could provide increased performance in communications. Firstly, algorithms that are used for signal processing in communications are often optimal or close to optimal. However, this optimality is based on a mathematically tractable model. Assumptions like stationarity, linearity and Gaussian distributions, although tractable, are not always valid. In practice, many imperfections exist, as well as non-linearities [10]. They can be introduced by the used amplifiers or quantizers, for example. Therefore, real models soon become intractable. This suggests an ML approach, as ML and neural networks do not require tractability.
Another reason for the use of DL in communications is coming from the conventional block structure of a communication system. The different blocks in the chain are used to be able to deal with the communication channel. These are individually optimized and have a tractable function [11]. All combined, however, they do not necessarily have to be optimal. For example, the separation between source and channel coding is not optimal, as explained in [12]. Moreover, separating encoding from modulation is suboptimal as well [13]. This also holds for separate equalization and decoding. A communication system based on neural networks does not require a strict block structure.
Lastly, algorithms based on neural networks can be run in parallel on concurrent architectures with data types that have low precision. Therefore, they can be executed faster and with lower energy costs [9, 14].

On the other hand, designing practical autoencoders comes with its challenges. For example, selecting the optimal structure of the neural networks can be difficult. Implementing expert domain knowledge into the design of the networks usually increases the performance of the system. Many different structures have already been proposed [15–20]. Another challenge is that of training the autoencoder. Since training requires the calculation of the gradient over the network, this becomes a problem when the underwater channel is unknown. Some works overcome this problem by estimating the channel or using a reinforcement learning (RL) scheme [21–25]. Furthermore, other difficulties exist when it comes to the practical implementation of the DL communication system. For instance, the size of the symbol alphabet and the transmission block size should be limited to restrict the network size.

Currently, there are no published works combining channel autoencoders with underwater acoustic communication. Combining these concepts is a broad and challenging problem. Therefore, in this work, a step towards a full ML communication system is taken by applying ML techniques for link adaptation and equalization.

1.3. Link adaptation

Link adaptation (LA), or adaptive modulation and coding (AMC) is defined as the method of selecting communication system parameters based on environmental conditions and constraints. Examples of different communication parameters that can be adapted include symbol constellation, transmit power, number of frequency bands and data rate [26–30]. Examples of useful environmental or channel parameters that could be used are: ambient noise level, received signal power, delay spread, Doppler shift and channel coherence time [26].

Selecting the best channel parameters and making the optimal mapping between channel parameters and communication schemes is not trivial. One of the first works regarding this problem for underwater acoustic communication is described in [31]. The authors use the signal-to-noise ratio (SNR), multipath spread and Doppler spread to change between multiple frequency-shift keying (MFSK) and phase-shift keying (PSK).
Another single-carrier AMC method is given in [32]. The parameters used to decide between modulation and coding scheme (MCSs) are the information rate with i.i.d. Gaussian inputs and the post-equalization SNR. To

obtain the needed threshold to switch between MCSs, the first step of the authors is to send different messages using all possible MCSs. The channel parameters are then calculated and thresholds are fixed heuristically.
AMC has been implemented for underwater multicarrier systems as well, for example in [33, 34]. In these works, multiple transmit modulations, so multiple rates, can be selected. Furthermore, in [33], the power allocation of subcarriers is changed adaptively.
Other publications have also explored the use of ML techniques to classify the optimal MCS based on underwater channel conditions. For example, in [35], the authors implement a decision tree to find the optimal thresholds for a single-carrier AMC system. The used channel parameters are SNR, delay spread and Doppler spread. Training data is gathered by transmitting with the different schemes in a simulated channel, obtained using known sound speed profiles.
Another ML method is described in [36]. In this work, the used parameters are the SNR, the delay spread, the delay of the strongest path, the total power of the first three paths, the total power of all paths and the normalized amplitude of the first path. Again, data is first gathered by transmitting using the different MCSs in simulated channels using known sound speed profiles. After obtaining the training data, the k-nearest neighbours (k-NN) algorithm is used to select the best MCS for transmission.
Other approaches for implementing ML-AMC in underwater acoustic communication are taken in [37–39]. These works try to solve the problem using RL.

A related problem to link adaptation is that of performance prediction of underwater communications using channel parameters. The authors in [40] implement various ML methods to predict the bit error rate (BER) based on features like the SNR and delay spread. A similar work attempts to extract useful channel features from the estimated channel impulse response (CIR) using a neural network [41]. These features are then used to perform performance prediction.

1.4. Equalization

Since the underwater channel distorts the transmitted signal, this has to be compensated for by the receiver. The process of compensating for the channel distortion is called equalization. Equalization is an essential step in the communication chain, as it allows for finding the originally transmitted message.

Over the years, many works have studied methods to perform channel equalization. The structure of the channel equalizer depends on the physical layer communication protocol that is used. One of the early equalizers consists of a decision feedback structure for single carrier communications [42]. This has been extended to multi-carrier systems as well [43, 44]. The decision feedback equalizer structure is also used in this thesis.

The equalizer compensates for the channel in an adaptive manner, meaning the filter coefficients are updated adaptively. To achieve this, optimization algorithms are implemented. Typically, optimization algorithms make use of (hyper)parameters to tune metrics such as the convergence speed and the algorithm error. In the case of channel equalization, the exact values of the algorithm parameters are of great importance, since they influence the performance of the equalizer substantially.
The least mean squares (LMS) and recursive least squares (RLS) are common adaptive optimization algorithms for decision feedback equalizers with a linear filter[42–44]. Many variations of the LMS and RLS algorithms have been introduced over the years in attempts to increase the performance of equalizers in terms of convergence speed and error [45]. Furthermore, publications show that the use of non-linear filters in the decision feedback equalizer can improve the performance of the equalizer in some scenarios [46–48].

1.5. Problem formulation

In this thesis, the frequency-repetition spread-spectrum (FRSS) physical layer will be used and studied. Specifics about FRSS are given in Chapter 2. This thesis aims to answer the following questions about link adaptation for underwater acoustic communications using the FRSS physical layer:

- Which channel or communication parameters provide valuable information for link adaptation?
- Which classification method can switch effectively between different transmission formats?
- What is the effect of feedback delay on the performance of link adaptation methods?

Furthermore, the thesis has the goal of answering the following questions about decision feedback equalization optimization for FRSS:

- Can an alternative equalizer algorithm increase the performance of the FRSS equalizer?
- Can the use of a non-linear filter increase the performance of the equalizer?

1.6. Thesis outline

In order to find answers to the stated research questions, an understanding of the underwater environment and its distinct challenges is needed. This background information is provided in Chapter 2. The chapter first introduces the underwater channel, after which a system model is given. The goal of Chapter 3 is to explain various machine learning algorithms that are implemented to obtain answers concerning the problems of link adaptation and equalization optimization. Chapter 4 elaborates on the methodology used for link adaptation and presents the acquired results. Similarly, Chapter 5 discusses the methodology and results related to the equalization optimization problem. Finally, Chapter 6 concludes the thesis and provides directions for future work.

2 Background

2.1. Introduction

After the introduction in the previous chapter, a more detailed overview of the underwater channel characteristics will be given in this chapter. Knowledge about these characteristics is needed to come up with a model to describe underwater acoustic communication. This model, together with the used physical layer communication scheme, is introduced in this chapter as well.

2.2. Underwater acoustics

2.2.1. Passive sonar equation

To physically communicate underwater, acoustic pressure waves are used. The standard unit for pressure is Pa, which is $\mathrm{N/m^2}$. These pressure waves are measured by underwater transducers, which transform the pressure differences into a voltage. However, instead of describing underwater signals in terms of pressure, acoustic or sound intensity is usually used. Sound intensity is defined as the energy per unit area that is carried by a sound wave. The sound pressure level (SPL) is equal to the sound intensity, for plane and spherical waves [49].
The SPL is defined with respect to a reference pressure level p_{ref^2}. In dB, this can be expressed as:

$$\mathrm{SPL}_{\mathrm{dB\ re\ }\mu\mathrm{Pa}^2} = 10\log_{10}\frac{p^2}{p_{\mathrm{ref}^2}} \tag{2.1}$$

where the subscript dB re $\mu\mathrm{Pa}^2$ indicates that a reference pressure level p_{ref} of 1μPa root mean square (RMS) is used. The transmitted signal is sent through the channel to the receiver. The acoustic intensity at the receiver will be lower due to the spreading of the acoustic power. Also, the noise will affect the received signal. This is described in the passive sonar equation [50]. The narrowband SNR at the receiver is given by:

$$\mathrm{SNR}_{\mathrm{dB}}(l,f) = \mathrm{SL}_{\mathrm{dB\ re\ }\mu\mathrm{Pa}^2} - \mathrm{TL}_{\mathrm{dB}}(l,f) - \Big(\mathrm{NL}_{\mathrm{dB\ re\ }\mu\mathrm{Pa}^2}(f) - \mathrm{DI}_{\mathrm{dB}}(f)\Big) \tag{2.2}$$

As can be seen, the SNR at the receiver, $\mathrm{SNR}_{\mathrm{dB}}(l,f)$ is distance and frequency dependent. The distance l is in km and the centre frequency f in kHz. $\mathrm{SL}_{\mathrm{dB\ re\ }\mu\mathrm{Pa}}$ is the sound intensity (SPL) of the source at a standard reference distance of 1 meter, $\mathrm{TL}_{\mathrm{dB}}(l,f)$ the transmission loss at the same reference distance, $\mathrm{NL}_{\mathrm{dB\ re\ }\mu\mathrm{Pa}^2}(f)$ the ambient noise SPL and $\mathrm{DI}_{\mathrm{dB}}(f)$ the directivity of the receiver. Directivity is the amount of rejection of omnidirectional noise by the receiver [50]. The receiver is assumed to be omnidirectional and therefore the directivity level is set to zero.

2.2.2. Transmission loss

The acoustic wave send by the transmitter loses acoustic energy over distance since the power is spread out. The spreading loss is not the same for all underwater environments. In case the water surface and bottom act as perfect reflectors, a spreading factor of $k = 1$ is used [51, 52]. This is called cylindrical spreading. For deep waters, the spreading is spherical and therefore the spreading factor is set to 2. The relation between the spreading and transmission loss is given by:

$$\mathrm{TL_{dB}}(l,f) = k \cdot 10\log_{10}\left(\frac{l}{l_0}\right) + l \cdot a(f) \tag{2.3}$$

where l_0 is a reference distance of 1m and $a(f)$ is the absorption coefficient in dB/km. In practice, often a spreading factor of $k = 1.5$ is used, which is called practical spreading [52].
The absorption coefficient for seawater can be expressed by Thorp's equation [53]:

$$a(f) = \left(\frac{0.11f^2}{1+f^2}\right) + \left(\frac{44f^2}{4100+f^2}\right) + \left(2.75 \cdot 10^{-4} f^2\right) + 0.003 \tag{2.4}$$

The absorption coefficient is shown in Figure 2.1. It can be seen that for higher frequencies, the absorption increases.

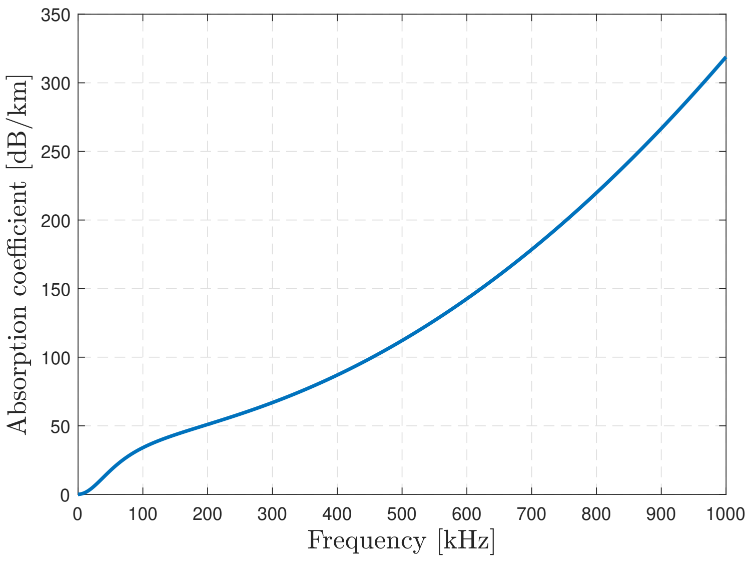


Figure 2.1: Absorption coefficient $a(f)$.

2.2.3. Ambient noise

The ambient noise level in the ocean can be decomposed into four main sources; turbulence, shipping, waves and thermal noise [52]. The turbulence noise is dominant in the low-frequency range, $f < 10$ Hz. Ambient shipping noise influences the region between 10 Hz to 100 Hz. Noise due to waves that are driven by the wind contributes to the noise in the frequency range between 100 Hz to 100 kHz. Lastly, thermal noise is present in the high-frequency region, $f > 100$ kHz. The cause of this noise is random pressure fluctuations at the receiver due to thermally agitated water molecules [54]. The noise sources and total noise spectral density level expressed in $\mu\mathrm{Pa}^2\mathrm{Hz}^{-1}$ can be represented by the following empirical equations [52]:

$$\begin{aligned}
N_\mathrm{t}(f) &= 10^{\left(17-30\log_{10}\left(\frac{f}{f_\mathrm{r}}\right)\right)/10} \\
N_\mathrm{s}\left(f,s_\mathrm{n}\right) &= 10^{\left(40+20(s_\mathrm{n}-0.5)+26\log_{10}\frac{f}{f_\mathrm{r}}-60\log_{10}\left(\frac{f}{f_\mathrm{r}}+0.03\right)\right)/10} \\
\mathrm{N_w}\left(f,w_\mathrm{n}\right) &= 10^{\left(50+7.5\sqrt{\frac{w_\mathrm{n}}{w_\mathrm{r}}}+20\log_{10}\frac{f}{f_\mathrm{r}}-40\log_{10}\left(\frac{f}{f_\mathrm{r}}+0.4\right)\right)/10} \\
N_\mathrm{th}(f) &= 10^{\left(-15+20\log_{10}\frac{f}{f_\mathrm{r}}\right)/10} \\
N\left(f,s_\mathrm{n},w_\mathrm{n}\right) &= N_\mathrm{t}(f)+N_\mathrm{s}\left(f,s_\mathrm{n}\right)+N_\mathrm{w}\left(f,w_\mathrm{n}\right)+N_\mathrm{th}(f)
\end{aligned} \tag{2.5}$$

where f_r is a reference frequency of 1 Hz, s_n the shipping factor (between 0-1), w_n is the wind speed in ms^{-1} and w_r is the reference wind speed of 1 ms^{-1}. The noise power spectral density (PSD) is shown in Figure 2.2. To connect the PSD in Equation 2.5 to the passive sonar equation, for a bandwidth of 1 Hz around frequency f, $\mathrm{NL}_{\mathrm{dB\ re}\ \mu\mathrm{Pa}^2}(f) = N\left(f,s_\mathrm{n},w_\mathrm{n}\right)$.

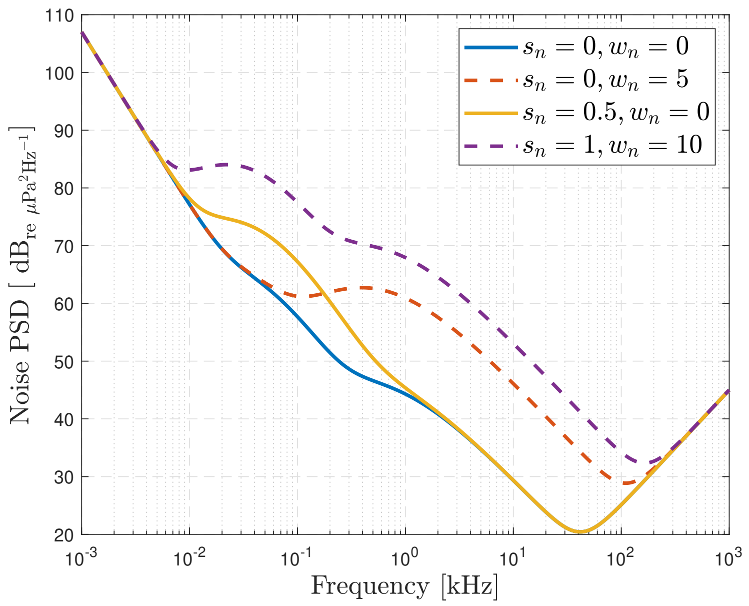


Figure 2.2: PSD of the ambient noise for different wind speeds and shipping factors.

2.2.4. SNR at receiver

Finally, the narrowband SNR at the receiver, as described in Equation 2.2, can be plotted as a function of distance and frequency. This is shown in Figure 2.3. Using longer distances results in a decrease in usable bandwidth. Furthermore, when using higher frequencies, the narrowband SNR is smaller.

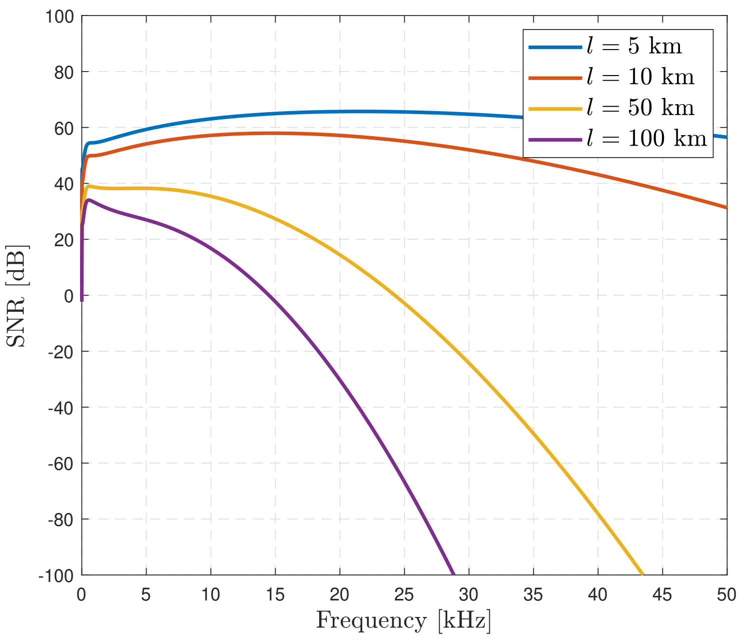


Figure 2.3: SNR as described by the passive sonar equation. In this scenario, $k = 1.5$, $s_n = 0.75$ and $w_n = 10$ ms^{-1}. The sound intensity of the source is set to 180 dB$_{\text{re }\mu\text{Pa}^2\text{ Hz}^{-1}}$.

2.2.5. Sound speed

As explained before, the body of the sea is not uniform in sound speed. The speed is dependent on temperature, depth and salinity. The ocean can be divided into several layers with different sound speed properties. The

three layers are: the surface layer, the main thermocline and the deep isothermal layer [4]. In the first layer, the sound speed is relatively constant. In the main thermocline, the speed rapidly decreases with depth and in the isothermal layer, the speed increases with depth.
These variations in sound speed have an effect on the path that the sound rays follow through the water. For example, when transmitting in between the second and third layer, the rays may refract up and down to form an acoustic waveguide.
Similar effects can cause areas in the ocean to receive almost no sound rays, because rays are refracted from that area. This is known as an acoustic shadow zone [4].

2.2.6. Multipath

When sound is transmitted underwater, sound rays will arrive at the receiver via different paths. This is caused by the variability in sound speed and the reflection of sound onto the surface and bottom. Therefore, a ray travelling over a refracted path may even be received earlier than a direct line-of-sight (LOS) ray. Due to these reflections and refractions, each path carries its own delay, power and phase shift. The time difference between the first and last arrival at the receiver is called the delay spread. In practice, only arrivals above a certain power level are considered. In underwater environments, the delay spread can be in the order of tens to hundreds of milliseconds [4].
When sending symbols faster than the delay spread duration, distortion between symbols will occur. This is because the speed of sound is relatively low and the delay spread is large. The distortion between symbols is called inter-symbol interference (ISI).
The transmitted symbols arrive at different times, introducing constructive and destructive interference. In frequency domain, this will result in fades. Therefore, some frequencies will be more attenuated than other frequencies. This is called frequency-selective fading. Furthermore, these fades change with time since the underwater environment is time-varying. Therefore, the channel can be characterized as time-varying frequency-selective.

2.2.7. Doppler effect

Not only delay spread affects the received signal, but the Doppler effect also influences the characteristics of the received signal. Due to motion of the transmitter, receiver and the water surface, the frequency of the signal changes. This is called a Doppler shift. The Doppler frequency shift can be calculated using the following equation:

$$\Delta f_d = f_c \frac{\Delta v}{c} \cos\theta \tag{2.6}$$

where Δv is the relative change in velocity, c is the sound speed and θ is the angle. Each ray in the multipath propagation will have its own Doppler shift, resulting in a Doppler spread.
When sending symbols on multiple close carrier frequencies, the Doppler shifts may lead to overlapping signals in frequency domain. The distortion caused by this effect is inter-carrier interference (ICI).

2.3. Underwater acoustic system model

After giving an overview of the underwater acoustic channel characteristics in the previous sections, a model of the communication system will be introduced. A schematic overview of this system can be found in Figure 2.4.

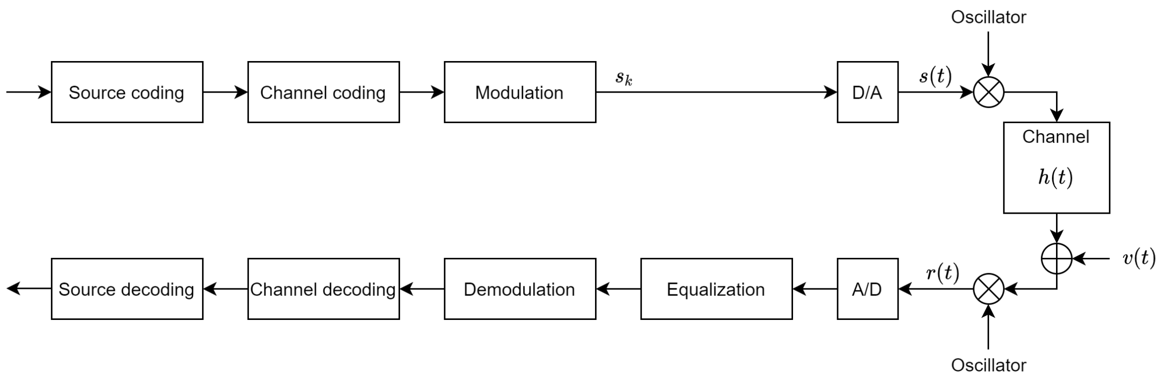


Figure 2.4: Schematic overview of the communication system.

2.3.1. Transmitter

The input to the model in Figure 2.4 is the raw data that should be transmitted. The source coding block takes this input and makes a mapping to a sequence of alphabet symbols (bits).
Then, in the channel encoder, error-correcting techniques are applied for increased performance in terms of bit errors. For example, additional bits can be added to the sequence. These are called parity bits. A well-known example of an error-correcting code is a convolutional code. A disadvantage is that sending additional bits decreases the overall data rate. A coding rate can be defined as the number of output bits per information bit. Interleaving is a technique that is often applied in combination with channel coding. This decreases the probability of burst errors occurring. A burst error, as opposed to a single bit error, happens when two or more nearby bits in a sequence are flipped. This makes it hard to decode the message, even with error-correcting techniques. By using interleaving, the order of the transmitted bits is changed. Therefore, the errors are spread and the probability of a burst of errors occurring is lower, making it easier to decode the message.

After encoding, the bits are modulated. In this block, the binary sequence is mapped into a sequence with a different alphabet, or constellation. The new sequence is called s_k and can be real or complex. This step is included since it can improve bandwidth efficiency. The higher the modulation level, the higher the bandwidth efficiency, due to the increased size of the modulation alphabet. On the other hand, larger constellations are harder to demodulate. Since the constellation points are closer together, symbols are more susceptible to noise when decoded.
When expressing s_k as a complex number $x_k + jy_k = R_k e^{j\theta_k}$, it is clear that a modulation alphabet can use various values for the amplitude and phase. For example, in M-ary phase-shift keying (M-PSK), M different constellation points with varying phases are used as a modulation alphabet. In M-ary pulse amplitude modulation (M-PAM), different amplitudes are adopted for modulation. A combination between PSK and PAM is M-ary quadrature amplitude modulation (M-QAM), where each constellation point encodes the signal using both phase and amplitude.
The modulated sequence is transformed into an analog, time-continuous signal. In practice, this sequence is convolved with a pulse shape function. However, in this model, the pulse shape function is included in the channel function. Therefore, the baseband analog signal $s(t)$ can be described as:

$$s(t) = \sum_{k=0}^{K-1} s_k \delta(t - kT) \tag{2.7}$$

where K is the number of transmitted symbols, T is the symbol time and $\delta(t)$ is the unit impulse. In practice, this method will not work, since the pulse function has infinite bandwidth. Therefore, a pulse shape function that is both local in time and frequency domain is used. In some derivations, this function is written as part of the analog transmitted signal [55]. However, in this work, it is assumed to be part of the channel. The definition of the pulse function will follow in the channel definition in the next section.

Lastly, the baseband analog signal $s(t)$ is modulated onto a carrier frequency f_c before sending it over the channel. However, for the rest of this thesis the Equations will be expressed in baseband. The up-conversions and down-conversions are assumed implicitly.

2.3.2. Channel

The model for the channel $h(t)$ includes the characteristics of the underwater acoustic channel as described in the previous chapter.
Firstly, the Doppler effect is taken into account using the wideband model described in [56]. The received signal in case the channel is only suffering from a single Doppler shift is:

$$r(t) = \sqrt{|\alpha|}s(\alpha(t-\tau)) \tag{2.8}$$

where α is the Doppler scaling. It can be approximated as: $\alpha \approx 1 + \frac{v}{c}$, where v is the relative velocity and c is the sound velocity. If $\alpha > 1$, that means the signal is time-compressed. On the other hand, when $0 < \alpha < 1$, the signal is stretched with respect to time. The term $\sqrt{|\alpha|}$ is a sort of normalization, because the power due to a Doppler shift does not change. The time delay of the signal is $\tau = \frac{d}{c}$, where d is the distance of the path between transmitter and receiver.

The description in Equation 2.8 suggests that the received signal is a Doppler-shifted version of the transmitted signal. However, due to the multipath environment, each path had its own Doppler shift, resulting in a Doppler spread. The channel can be described as:

$$h(t) = \sum_{l=1}^{L} \beta_l \sqrt{|\alpha_l|} p(\alpha_l(t-\tau_l)) \tag{2.9}$$

where L is the number of paths and β_l are the channel coefficients. The fading of the channel coefficients is sometimes presumed to follow a Rayleigh distribution [4, 57]. For shallow water and medium-range, it has been found that this is an appropriate model [58, 59]. However, for other environments, this might not be the case. A Rician fading model or a deterministic model might be a better fit [60].
Lastly, $p(t)$ is the pulse shape function. Equation 2.7, suggests that the transmitted signal contains Dirac-pulses. However, that would mean infinite bandwidth is used. In practice, a pulse shape function that is local in both time and frequency domain is used. An example of a function that approximately has these properties is the root raised cosine pulse. This function is defined as [61]:

$$p(t) = \frac{\sin\left(\frac{\pi t}{T}\right)}{\frac{\pi t}{T}} \cdot \frac{\cos\left(\frac{\gamma \pi t}{T}\right)}{1 - \left(\frac{2\gamma t}{T}\right)^2} \tag{2.10}$$

where γ is the roll-off factor. This parameter determines how smoothly the amplitude decays in frequency domain. Furthermore, it determines the amplitude of the tail in the time domain.

Lastly, also noise $v(t)$ is added by the channel. This noise is sometimes modelled as additive white Gaussian noise (AWGN). However, in practice it is frequency and situational dependent, as discussed in Section 2.2.3.

2.3.3. Receiver

With the channel and noise described, the received signal is:

$$\begin{aligned} r(t) &= h(t) * s(t) + v(t) \\ &= \int_0^{PT} h(\tau)s(t-\tau)d\tau + v(t) \\ &= \int_0^{PT} h(\tau) \sum_{k=0}^{K-1} s_k \delta(t-\tau-kT)d\tau + v(t) \\ &= \sum_{k=0}^{K-1} s_k \int_0^{PT} h(\tau)\delta(t-\tau-kT)d\tau + v(t) \\ &= \sum_{k=0}^{K-1} s_k h(t-kT) + v(t) \end{aligned} \tag{2.11}$$

where $P \cdot T$ is the length of the channel.
In case the received signal is sampled at symbol times and synchronized, the following discrete convolution is obtained:

$$r_n = \sum_{k=0}^{K-1} s_k h_{n-k} + v_n = \sum_{k=0}^{P-1} h_k s_{n-k} + v_n = h_n * s_n + v_n \tag{2.12}$$

where $r_n = r(nT)$ and $h_n = h(nT)$. Stacking the samples in a vector gives the following model description:

$$\mathbf{r} = \mathbf{Hs} + \mathbf{v} \tag{2.13}$$

where

$$\mathbf{r} = \begin{bmatrix} r_0 \\ \vdots \\ r_N \end{bmatrix}, \mathbf{H} = \begin{bmatrix} h_{P-1}^{-P+1} & h_{P-2}^{-P+2} & \cdots & h_0^0 & & & \\ & h_{P-1}^{-P+2} & h_{P-2}^{-P+3} & \cdots & h_0^1 & & \\ & & \ddots & & \ddots & & \\ & & & h_{P-1}^0 & h_{P-2}^1 & \cdots & h_0^N \end{bmatrix}, \mathbf{s} = \begin{bmatrix} s_{-P+1} \\ \vdots \\ s_0 \\ \vdots \\ s_m \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_0 \\ \vdots \\ v_N \end{bmatrix} \tag{2.14}$$

where the subscript and superscript of each h describes the sampling time and channel number, respectively. For example, h_3^1 indicates the channel function $h(t)$ for symbol s_1, sampled at time $t = 3T$. This is needed due to the time-varying nature of the channel.

Alternatively, sampling can also be done at a rate higher than the symbol time $T_{\text{samp}} < T$. In that case, the received samples for one symbol can be written as:

$$\mathbf{r}' = \begin{bmatrix} r(nT - N_1 T_{\text{samp}}) \\ \vdots \\ r(nT) \\ \vdots \\ r(nT + N_2 T_{\text{samp}}) \end{bmatrix} \tag{2.15}$$

where N_1 and N_2 are parameters that determine how many samples are taken before and after the symbol time. Note that $\mathbf{r}'$ now contains ISI from past and future symbols.

2.4. Communication systems

2.4.1. Single carrier technique

Practical underwater acoustic communication systems have been developed over the years. One of the first publications uses a single carrier technique, which means the message is sent using only a single frequency band [42]. The publication showed a receiver structure that performs channel equalization and phase synchronization jointly. In this receiver structure, the received signal is first brought back to baseband, low-pass filtered and frame synchronized. A fractionally spaced decision feedback equalizer (DFE) is used with sampling frequency $T_{\text{samp}} < T$ to obtain sample vector $\mathbf{r}'$, as described in Equation 2.15. The DFE is shown in Figure 2.5. Then, these samples are filtered using a feedforward filter $\mathbf{c}$ to produce an estimate of the transmitted symbol. The carrier phase offset, however, is accounted for separately, since it changes more rapidly than the channel tap gains. The resulting symbol estimate (including ISI) is:

$$p_n = \mathbf{c}^T \mathbf{r}' \tag{2.16}$$

The last step in this receiver is to eliminate the ISI. This is done by subtracting it using a feedback filter that operates on M previously estimated symbols:

$$\hat{s}_n = p_n - \mathbf{b}^T \tilde{\mathbf{s}} \qquad \tilde{\mathbf{s}} = \begin{bmatrix} \hat{s}_{n-1} & \cdots & \hat{s}_{n-M} \end{bmatrix} \tag{2.17}$$

The structure of using a feedforward and feedback filter is called a DFE. A phase-locked loop (PLL) is used for phase synchronization. The equalizer and synchronizer parameters are updated jointly. To accomplish that, an algorithm like the least mean squares (LMS) or recursive least squares (RLS) algorithm can be adopted.

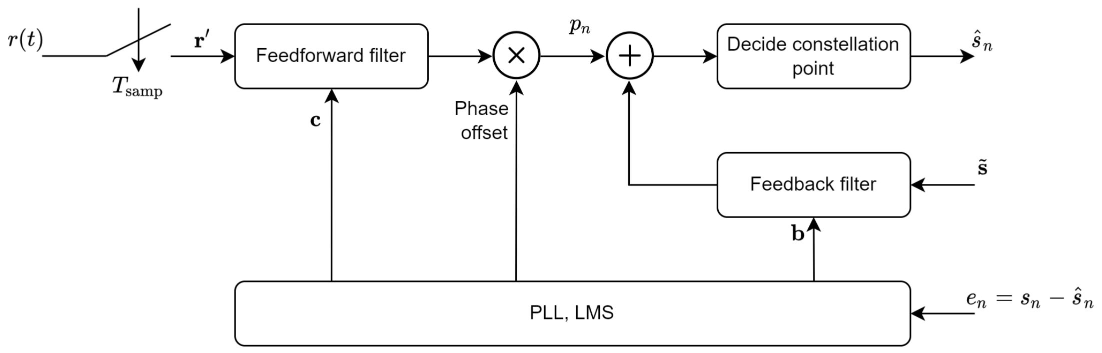


Figure 2.5: Overview of the single carrier decision feedback equalizer.

2.4.2. OFDM

Instead of using a wideband single carrier system, a multiple carrier system with narrower bands can also be used. In that case, the symbol rate per band is lower, resulting in reduced ISI. This is exploited in orthogonal frequency division multiplexing (OFDM).
The orthogonal carrier frequencies are obtained using the inverse discrete Fourier transform (IDFT). So instead of directly using the modulated symbols $\mathbf{s}$, it is first multiplied with $\mathbf{F}^H$ to get $\mathbf{x} = \mathbf{F}^H\mathbf{s}$. Where $\mathbf{F}^H$ is the IDFT matrix (and $\mathbf{F}$ is the DFT matrix). After cyclic prefixing, transmission and taking the DFT, the received signal is:

$$\mathbf{r} = \mathbf{F}\mathbf{y} = \mathbf{F}\mathbf{H}_c\mathbf{x} = \mathbf{F}\mathbf{H}_c\mathbf{F}^H\mathbf{s} = \mathbf{A}\mathbf{s} \tag{2.18}$$

where $\mathbf{H}_c$ is the cyclic prefixed version of $\mathbf{H}$ in Equation 2.14 and ^{H} is the Hermitian transpose. Cyclic prefixing ensures that the linear channel convolution turns into a circular convolution. This enables simpler processing in the frequency domain, as the circular convolution in time domain is a multiplication in frequency domain. It also provides a guard interval to prevent ISI.
When an estimate of $\mathbf{A}$ is available, the transmitted symbols can be obtained using an equalizer. Multiple choices can be made for equalization. For example, full block equalizers or partial block equalizers [57].
It should be noted that this only holds if the channel is time-invariant over one OFDM symbol, i.e. when all the rows in $\mathbf{H}$ are identical up to a shift. When this is not the case, banded equalizers could be used to remove the ICI [62].

2.4.3. FRSS

Another technique that has been developed previously is multi-carrier spread spectrum (MCSS) [43, 44]. This is a multi-band technique. MCSS shares similarities with direct sequence spread spectrum (DSSS). In DSSS, the symbols are multiplied with a spreading code consisting of a chip sequence. The frequency of the spreading code is larger than the frequency of the symbols, such that the symbols are spread in time. On the contrary, in MCSS, the signal is spread in frequency. Therefore, the number of chips defines the number of subbands used. MCSS was later simplified and adapted. It was also given a different name: frequency-repetition spread-spectrum (FRSS) [63]. The multi-band structure, however, was not changed. FRSS is jointly developed by TNO and FFI (Norwegian Defence Research Establishment) [64]. In Figure 2.6, the DSSS and FRSS modulation methods are shown schematically (adapted from [65]).

In this thesis, FRSS will be used as the benchmark wireless communication method. The same structure as in Figure 2.6 is used. Actually, when referring to FRSS, not only the multi-band structure is meant, but the entire physical layer system. The data part of the package is modulated onto multiple frequency bands. Similar as done by Van Walree et al. [65], four different configurations, or rates, are used: $R \in \{1,2,3,4\}$. The number of frequency subbands is related to the rate as: $B = 2^R - 1$. The frequency range used in this work is 4-8 kHz. The number of initial training bits N_t is related to the FRSS rate as: $N_t = 2^{10-R} - 1$. In this thesis, packet lengths are considered with a length of 156 information bits. The first 24 bits contain the header of the packet.

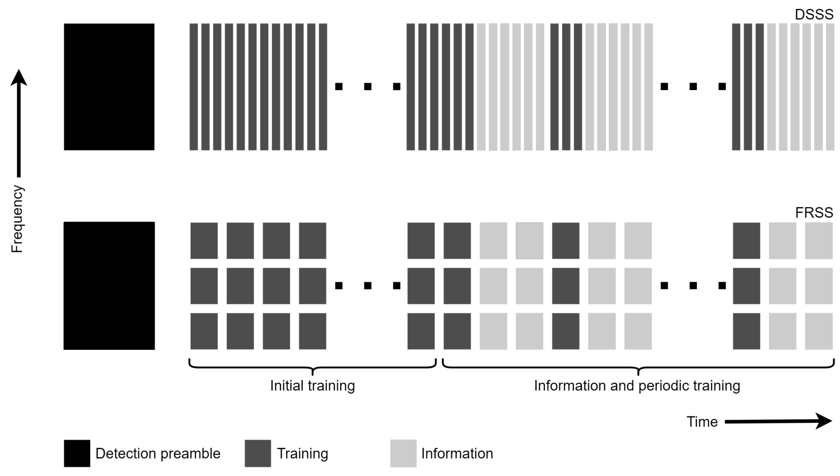


Figure 2.6: Schematic overview of DSSS and FRSS modulation for 3 chips, adapted from [65].

Transmitter

Similar as given by Van Walree et al. [65], before sending the information symbols, first a single-carrier detection preamble is transmitted. The preamble contains an m-sequence, which can be matched filtered to a bank of m-sequence replicas with different Doppler shifts. This allows for the detection of a transmission and for synchronization in time. Since the preamble is modulated onto a single frequency band, it can be written in the same way as Equation 2.7 as:

$$s_p(t) = \sum_{k=0}^{K_p-1} m_k \delta(t - kT) \tag{2.19}$$

where m_k is the m-sequence and K_p the number of preamble symbols. A root raised cosine pulse is used as the transmit pulse shape, which is set as part of the channel model, similar as in Equation 2.9.

After the preamble, the payload part of the message is transmitted. However, before they are transmitted, first they are scrambled to ensure a random-like bitstream. Then channel coding is achieved by using a rate-1/2 convolutional encoder. In order to decrease the probability of burst errors, the bits are also interleaved. To create the symbol sequence, quadrature phase-shift keying (QPSK) is used. So the bits are grouped in pairs and mapped as $\{11, 01, 00, 10\} \to \{(1+i)/\sqrt{2}, (-1+i)/\sqrt{2}, (-1-i)/\sqrt{2}, (1-i)/\sqrt{2}\}$.

The transmitted data part of the package in baseband is:

$$s_d(t) = \left(\sum_{k=0}^{K-1} s_k \delta(t/B - kBT)\right)\left(\sum_{b=1}^{B} c_b e^{-2\pi j f_b t}\right) \tag{2.20}$$

where c_b are the frequency chips, f_b the subband frequencies, B the number of subbands and K the number of data symbols. The subband center frequencies are spaced at equal distances around the overall center frequency.

Receiver

After transmitting the signal, the received signal $\tilde{r}(t)$ is basebanded with respect to the overall center frequency. The preamble part of the incoming signal is then correlated with a bank of replicas of the m-sequence with different Doppler shifts. This way, the Doppler shift can be estimated. The remaining part of the package is then basebanded per frequency band and the Doppler shift is accounted for. Also, the symbol sequences for each frequency band are bandpass filtered and normalized.

After these operations, the baseband signals for each frequency band b are written as $y_b(t)$. For equalization, the DFE-PLL structure, as given in Section 2.4.1. is implemented. However, note that the feedback filter part of the equalizer as described in Section 2.4.1 is not used in FRSS, although it is supported. The signals $y_b(t)$ are downsampled to a samples per symbol and phase-shifted using the PLL. Then they are stacked in a single vector and fed to the fractionally spaced $(d \cdot T/a)$ equalizer with L filter taps per subband [43]. So per symbol k:

$$\mathbf{y}_k = \left[\mathbf{y}_{1,k}^T, \ldots, \mathbf{y}_{B,k}^T\right]^T \tag{2.21}$$

where [43]:

$$\mathbf{y}_{b,k} = \begin{bmatrix} y_b\left((k-1)BT - \frac{(L-1)dBT}{2a}\right) \\ y_b\left((k-1)BT - \frac{(L-3)dBT}{2a}\right) \\ \vdots \\ y_b\left((k-1)BT + \frac{(L-3)dBT}{2a}\right) \\ y_b\left((k-1)BT + \frac{(L-1)dBT}{2a}\right) \end{bmatrix} \exp\left[-j\theta_b(k-1)\right] \tag{2.22}$$

The equalization then gives the following symbol estimate:

$$\hat{s}_k = \mathbf{y}_k^T \mathbf{c}_k \tag{2.23}$$

where $\mathbf{c}_k$ are the filter coefficients. To update the filter coefficients, the LMS algorithm is used. Details about this algorithm can be found in Chapter 5. The initial training of the equalizer is achieved using the known initial training symbols that precede the information symbols, as can be seen in Figure 2.6. Furthermore, periodic training is done using the known periodic training symbols. In the case of the initial and periodic training symbols, $s_{k,\text{ref}}$ is the known symbol. In the case of the information symbols, $s_{k,\text{ref}}$ is the closest constellation point.

The symbol estimates calculated by the equalizer and the periodic training symbols are then utilized to calculate the log-likelihood ratio for each symbol [43]. The log-likelihood ratios are then deinterleaved and decoded by a soft-decision Viterbi decoder. Lastly, the estimated bits are descrambled to get the original bit structure back. The full overview of the FRSS receiver is shown in Figure 2.7.

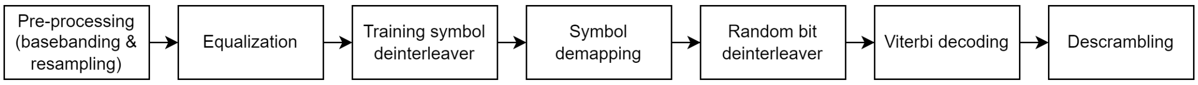


Figure 2.7: Schematic overview of the FRSS receiver.

2.5. Underwater channel simulation

Channel simulators

The methods introduced in the next chapters will need to be trained and tested in underwater channels. It is important that both training and testing is done in realistic underwater environmental conditions. If this is not done, the methods are not likely to perform well in real channel conditions. Also, the entries in the training and testing dataset should be created using a large variety of time-varying channels.

In order to simulate underwater acoustic communications, time-varying impulse responses are required. As explained in Section 2.3.2, there is no general model for the impulse response. An option for simulating underwater communications while incorporating multipath propagation in various underwater conditions is Bellhop [66]. The Bellhop simulator uses a sound speed profile (SSP) and bathymetry to generate a CIR. A disadvantage of Bellhop is that many input parameters have to be set to make a realistic environment. For example, bottom sediment type, channel geometry parameters, small-scale and large-scale channel displacement parameters, Doppler parameters, noise, etc.

Another option for underwater acoustic communication simulation is the underWater AcousTic channEl Replay benchMARK (Watermark) [67]. This simulator uses recorded channels to do channel replay. In other words, Watermark convolves the acoustic waveform with a recorded time-varying channel. Furthermore, it allows for adding AWGN to the received signal. The channel replay in Watermark is achieved using the Mime channel simulator [68].
An advantage of using Watermark as opposed to Bellhop is that it is able to simulate realistic underwater communication without the need for many user input parameters. However, recorded channels need to be available. Fortunately, Watermark offers a library of recorded channels. Furthermore, TNO has a large variety of

recorded channels stored as well. Watermark version 1.0 is used in this thesis. The choice of using Watermark instead of other channel simulation tools is one of the reasons that make this thesis novel compared to other publications discussing machine learning aided link adaptation and equalization.

Recorded channels
The recorded channels that are used to create a training and testing dataset come from an experiment conducted in the Oslofjord, from April 30 2019, 23:00, to May 2 2019, 01:00 [69]. A network consisting of 8 bottom nodes was deployed underwater. The network configuration is shown in 2.8.

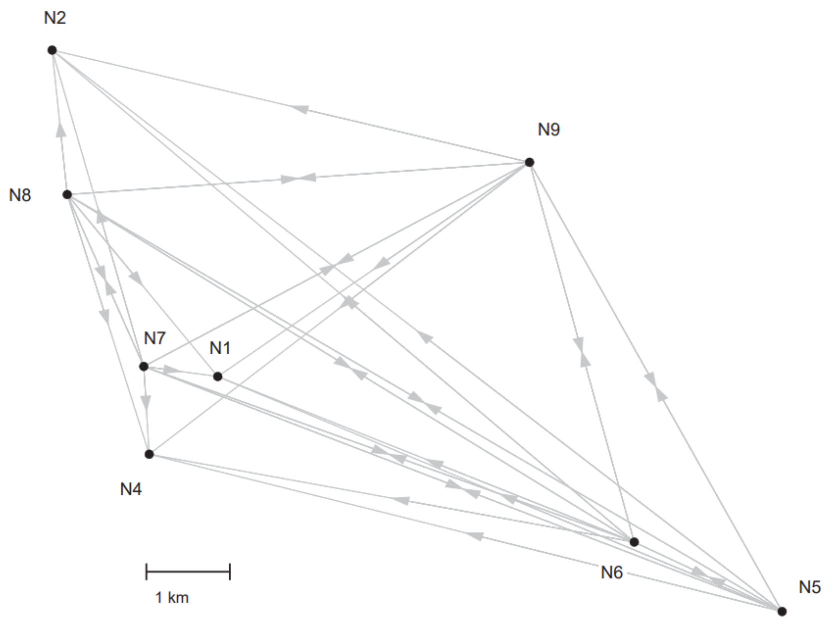


Figure 2.8: Schematic representation of the connections between nodes in the underwater network [69].

To estimate the channel delay profile, 250 ms chirp probe sequences were transmitted. They were subsequently transmitted up to a duration of 8 s. Per node pair a new chirp sequence was transmitted every 3 minutes until 520 cycles were reached. Only high-quality receptions (SNR > 15 dB) were used to generate channel delay profiles. In total, this yielded a total of 2842 unique profiles each covering 8 s.

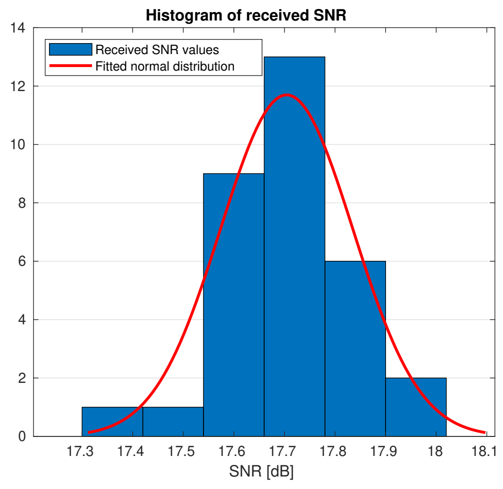


Figure 2.9: Example of received SNR values with fitted normal distribution.

Added noise

When using the recorded channels in Watermark, noise has to be added to the received waveforms to create a realistic simulation. Watermark allows the use of AWGN. However, the user has to define the noise level for the simulation. In practice, the noise level is not constant, but changes with time. Therefore, this should be considered to make the simulations realistic.

In order to come up with noise levels and how the noise levels change over time, the received chirp sequences of the Oslofjord experiment are used. The noise level over time for each chirp sequence is analyzed.

Actually, in Watermark, the SNR is the input parameter, as opposed to the noise level. However, since stationary nodes with fixed transmission power levels were used in the experiment, analyzing the SNR is similar to analyzing the noise level.

To incorporate the behaviour of the received SNR during simulation, it was assumed to be a random variable. The distribution of the received SNR was found by analyzing the SNR for each of the received chirps. It was found that a normal distribution for the SNR in dB within each channel segment provided a good fit. The range of realistic SNR mean values was found to be $[-5, 25]$ dB.

To illustrate the methodology, an example is shown in Figure 2.9. In this example, 32 chirps were transmitted consecutively. The SNR in dB at the receiver is calculated for each chirp. A normal distribution is fitted to the data.

Figure 2.10 shows standard deviation values observed when analyzing all the distribution shapes for the received SNR batches.

During simulation, the SNR for each packet in a channel fragment is therefore drawn uniformly from a normal distribution with the mean being a value in the range $[-5, 25]$ dB and the standard deviation in the range $(0, 1]$ dB.

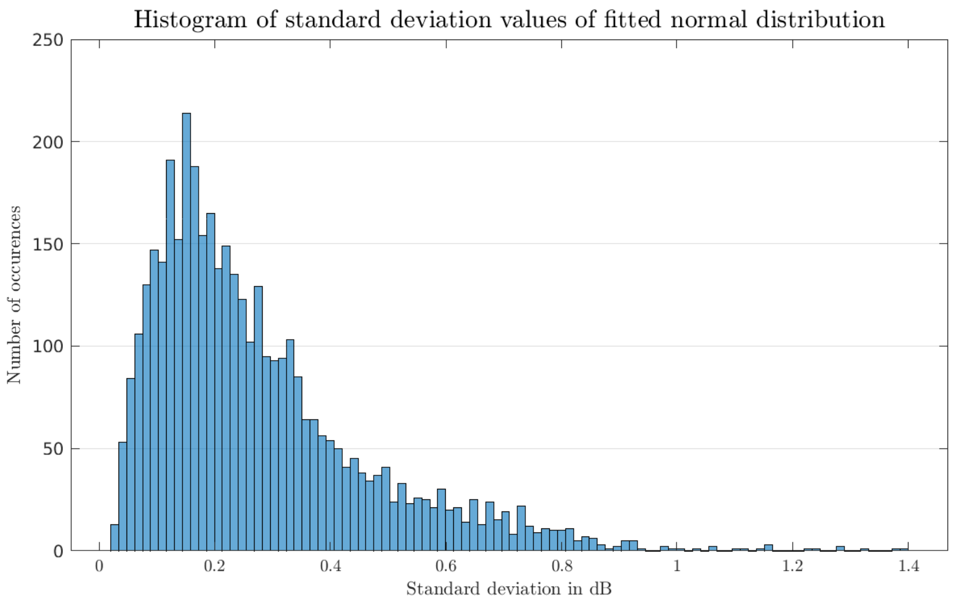


Figure 2.10: Histogram of standard deviation values found for a fitted Gaussian distribution.

3
Machine learning

3.1. Introduction

In this chapter, some machine learning methods will be discussed. These methods are implemented in the next chapters. First, machine learning (ML) algorithms for classification are explained. Subsequently, algorithm unrolling is introduced.

3.2. Classification methods

Multiple classification methods are identified to solve the problem of finding the best transmission strategies. This section gives an overview of the methods that are implemented.

Decision tree

An intuitive machine learning method that can be used for classification is a decision or classification tree. Decision trees can be used for classification and regression tasks. To goal of the method is to create a model that can predict the correct class by learning decision rules to partition the input data space. A decision tree creates rectangular, piecewise constant, decision boundaries.

An advantage of decision trees is that they are simple to visualize and understand. In other words, decision trees are a white-box approach. When providing an input sample, the conditions to come to an output class can be explained using the tree.

The tree is made up of nodes and branches. The starting node is called the root node. The final nodes are the leaf nodes. Each node splits into two branches, which each connects to another node. The depth of the tree is defined as the number of node levels.

Learning a decision tree can be achieved using multiple algorithms. In this thesis, the classification and regression tree (CART) algorithm will be used [70]. The CART algorithm works as follows. The training data D consists of features $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_L]$ and classes $\mathbf{y}$. The total number of training entries is P. The data that is split at node n is D_n, consisting of P_n entries. Splits are made starting from the root node. A split is defined as $s = (l, \varepsilon)$. Where l is the feature used for splitting and ε is the threshold value of that feature. After splitting, the left and right subsets of data are:

$$\begin{aligned} D_n^{\text{left}}(s) &= \left\{ S(\mathbf{X}, \mathbf{y}) \mid S(\mathbf{x}_l) \leq \varepsilon_n \right\} \\ D_n^{\text{right}}(s) &= \left\{ S(\mathbf{X}, \mathbf{y}) \mid S(\mathbf{x}_l) > \varepsilon_n \right\} \end{aligned} \tag{3.1}$$

where S is an operator that selects the dataset rows that are used at node n. Splits are chosen based on a loss or impurity function H. For example, the entropy loss function $H(D_n) = -\sum_k p_{nk} \log(p_{nk})$ can be used. Where p_{nk} is the fraction of class k observations in node n. The best split is:

$$s^* = \operatorname{argmin}_s G(D_n, s) \tag{3.2}$$

where:

$$G(D_n, s) = \frac{P_n^{\text{left}}}{P_n} H\left(D_n^{left}(s)\right) + \frac{P_n^{\text{right}}}{P_n} H\left(D_n^{\text{right}}(s)\right) \tag{3.3}$$

Splitting is executed until the desired tree depth is reached.
The decision tree classifier is implemented in Python using scikit-learn sklearn.tree.DecisionTreeClassifier [71].

k-nearest neighbors
The k-nearest neighbours (k-NN) algorithm can also be used for classification tasks. Again, the training set D consists of features $\mathbf{X}$ and classes $\mathbf{y}$. So each training point can be written as $(\mathbf{x}_i, y_i)$. The k-NN method is relatively simple. The training feature values and the corresponding classes are simply stored. If a new point needs to be classified, the algorithm finds the k nearest points in the feature space. A majority vote is used to classify the new point. To measure the distance between data points, the Euclidean distance is applied.
The k-nearest neighbours classifier is implemented in Python using scikit-learn sklearn.neighbors. KNeighborsClassifier [71].

Support vector machine
Another classification method is a support vector machine (SVM). Just like a decision tree, an SVM attempts to find a boundary between classes to predict the correct class based on the input features. The support vectors are the data points that lie closest to the boundary. The decision boundaries are found by solving a quadratic programming problem. Just as before, the training data D consists of features $\mathbf{X}$ and classes $\mathbf{y}$, such that each training entry is of the form $(\mathbf{x}_i, y_i)$. The total number of training entries is P. For now, it is assumed there are two different classes, either 1 or -1. The goal is to find a hyperplane that separates the two groups. The best hyperplane is the one that maximizes the distance between the hyperplane and the nearest point $\mathbf{x}_i$ of each class. A hyperplane is the set of points $\mathbf{x}$ that satisfies: $\mathbf{w}^T\mathbf{x} - b = 0$.
When the classes can be separated linearly, two hyperplanes can be defined which divide the classes. The goal is to have these hyperplanes as far apart as possible. The hyperplanes are:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i - b &= 1 \\ \mathbf{w}^T\mathbf{x}_i - b &= -1 \end{aligned} \tag{3.4}$$

where anything on or above the boundary in the first equation belongs to the class $y = 1$ and vice versa for the bottom equation. Since the distance between the hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, the goal is to minimize $\frac{1}{2}\|\mathbf{w}\|$.
A constraint is that there are no data points between the two hyperplanes, i.e.:

$$y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1, \quad \text{for all } 1 \leq i \leq P \tag{3.5}$$

The full optimization problem is then:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 \geq 0, \quad i = 1, \ldots, P \end{aligned} \tag{3.6}$$

To solve this problem, the method of Lagrange multipliers is used. The Lagrange multiplier are $a_i \geq 0$. The Lagrange function is:

$$L_p(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{P} a_i\left\{y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1\right\} \tag{3.7}$$

However, in some cases the classes are not linearly separable. In that case, a loss function can be used; for example, the hinge loss. The hinge loss is proportional to the distance to the hyperplane in the case the data is on the wrong side, i.e. $\zeta_i = \max\left(0, 1 - y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right)\right)$. The optimization problem then becomes:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \tfrac{1}{P}\textstyle\sum_{i=1}^{P}\zeta_i + \lambda\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i \end{aligned} \tag{3.8}$$

And the Lagrange function:

$$L_p(\mathbf{w}, b, \mathbf{a}) = \lambda\|\mathbf{w}\|^2 + \sum_{i=1}^{P}\zeta_i - \sum_{i=1}^{P} a_i\left\{y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \zeta_i\right\} - \sum_{i=1}^{P}\mu_i\zeta_i \tag{3.9}$$

with Lagrange multipliers $a_i \geq 0$ and $\mu_i \geq 0$.
Finally, the dual function is:

$$\begin{aligned} \underset{\mathbf{a}}{\text{maximize}} \quad & \textstyle\sum_{i=1}^{P} a_i - \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P} y_i a_i \left(\mathbf{x}_i^T \mathbf{x}_j\right) y_j a_j \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{P} a_i y_i = 0 \text{ and } 0 \leq a_i \leq \frac{1}{2P\lambda}, \text{ for all } i \end{aligned} \tag{3.10}$$

which can efficiently be solved by a quadratic programming algorithm. When a_i is found, vector $\mathbf{w}$ can be calculated as $\mathbf{w} = \sum_{i=1}^{P} a_i y_i \mathbf{x}_i$.

Instead of looking for linear boundaries, one might also be interested in non-linear boundaries between classes. The same method as before can be applied for the non-linear case. Non-linear classification is executed using a transformed dataset $\phi(\mathbf{x}_i)$. The dot product in Equation 3.10 then changes to a kernel function: $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi(\mathbf{x}_i) \cdot \phi\left(\mathbf{x}_j\right)$. Different kernel functions can be used. Examples are a polynomial $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = (\mathbf{x}_i \cdot \mathbf{x}_j + r)^d$ and a Gaussian radial basis $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$ kernel function.
In the problem of link adaptation, there are more than two different classes. Therefore, a multi-class SVM (one-versus-one) is implemented. This means 2-class SVMs are trained for each pair of classes. Then, classification is done using a majority vote over all SVMs.
The decision tree classifier is implemented in Python using scikit-learn sklearn.svm.SVC [71].

Neural network
The last classification method that will be used, is a neural network. Neural networks, or multilayer perceptrons, are building blocks of deep learning (DL), which is a subfield of ML. Nowadays, these networks are applied to many different problems. Examples include solutions for domains like language processing, image classification and computer vision. DL offers solutions for problems where a mathematical model is hard to come up with. A classic example is the task of classifying handwritten numbers. It is certainly difficult to come up with a mathematical model to do this. However, nowadays, it is rather straightforwardly solved with DL.

In its most common form, the artificial neural network (ANN) or deep neural network (DNN) consists of multiple layers of neurons. Layers between the input and output layers are called hidden layers. Each neuron holds a value called activation. Neurons of subsequent layers are connected using connections with specific weights. Every single neuron can have a bias as well. Furthermore, a non-linear function is applied. The connection between one neuron activation and the previous layer is schematically represented in Figure 3.1 and is described as:

$$a_0^{(1)} = \sigma\left(w_{0,0}^{(1)} a_0^{(0)} + w_{0,1}^{(1)} a_1^{(0)} + \cdots + w_{0,n}^{(1)} a_n^{(0)} + b_0^{(1)}\right) = \sigma(z_0^{(1)}) \tag{3.11}$$

where $a_n^{(i)}$ is the activation of neuron n in layer i, $w_{p,n}^{(i)}$ is the weight between neuron p of layer i and neuron n of layer $i-1$ and $b_n^{(i)}$ is the bias corresponding to neuron n in layer i. Lastly, σ is the non-linear or activation function. Different activation functions can be used. Well-known examples are the ReLu function; $\sigma(x) = \max(0, x)$ and Sigmoid function; $\sigma(x) = 1/(1+e^{-x})$. The latter function is useful when the activation should be defined between 0 and 1.

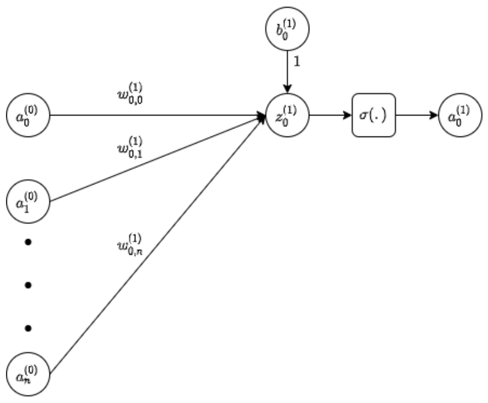


Figure 3.1: Schematic representation of the connection between neural network layers.

The network is trained by changing the network parameters, i.e. the weights and biases. To do that, first, a loss function or cost function is defined as a measure of the network performance. Commonly used examples of loss functions are the mean squared error (MSE) and the cross-entropy (CE) loss functions:

$$L_{\text{MSE}} = \sum_{d=1}^{D} \sum_{k=0}^{K-1} (a_{k,d}^{(i)} - m_{k,d})^2 \qquad L_{\text{CE}} = -\sum_{d=1}^{D} \sum_{k=0}^{K-1} m_{k,d} \log a_{k,d}^{(i)} \tag{3.12}$$

where K is the size of the label vector and output layer, i is the output layer number and m is the sample label. The loss for each entry d out of a dataset of size D is calculated and summed. The CE loss function is mostly used in classification tasks, whereas the MSE loss is used in regression tasks. This is due to fact that the CE function more heavily penalizes predictions that are further from the correct output.

To optimize the network parameters, the gradient of the loss function ∇L is calculated with respect to the weights and biases. Due to the number of parameters and the non-linearity of the network, the gradient cannot directly be set to 0 to calculate the optimal parameter values. Therefore, an optimization algorithm is adopted. Usually, a form of gradient descent is used. Gradient descent minimizes a function by taking steps in the direction of the negative gradient:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \alpha \nabla L(\boldsymbol{\theta}_n) \tag{3.13}$$

with network parameter vector $\boldsymbol{\theta}$, step number n and step size α. Instead of using the whole training set to calculate the negative gradient, usually, a group of samples (minibatch) is taken to approximate the negative gradient. Each gradient descent step then runs on a different minibatch. The algorithm to calculate the negative gradient is called backpropagation. In practice, gradient-descent-based algorithms like Adam [72] are used to train the network.
The neural network classifier is implemented in Python using PyTorch [73].

3.3. Algorithm unrolling

Another machine learning method that will be used in this thesis is algorithm unrolling or unfolding. This technique allows for the development of efficient, high-performance and interpretable neural network architectures [74]. In this thesis it is used for (hyper)parameter optimization. The basic idea of algorithm unrolling is presented in Figure 3.2.

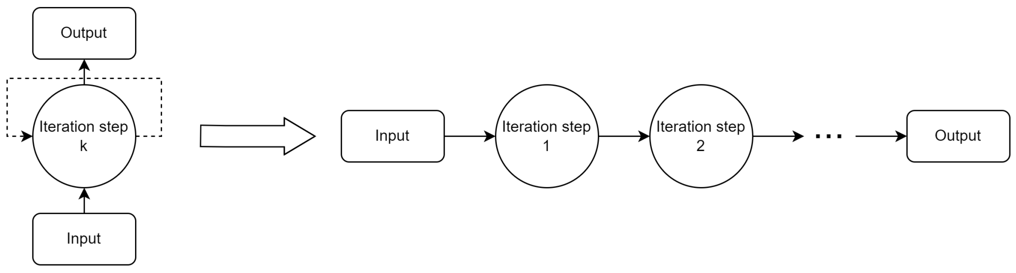


Figure 3.2: Concept of algorithm unrolling.

An iterative algorithm is unrolled over multiple iterations. The iterative algorithm uses certain hyperparameters, such as step size, that needs to be optimized for.
In case the correct output is known and the iterations consist of tractable functions, a loss can be calculated. With this loss computed, a gradient descent based algorithm can be used to optimize for the parameters of the iterative algorithm. Provided enough data, this can provide an accurate result more computationally efficient than a grid search or parameter sweep, for example [75].
Algorithm unrolling is implemented in Python using PyTorch [73].

4 Machine learning aided link adaptation

4.1. Introduction

This chapter discusses the problem of link adaptation for underwater acoustic communication in more detail. Choices for link adaptation are based on information about the underwater channel. Therefore, this chapter starts with a description of channel features used for link adaptation. Then, the implemented methodology and structure are explained. Lastly, acquired results are shown.

4.2. Channel features

This section will state the sources of information that can be used for the problem of link adaptation. The introduced channel features contain information about how difficult (e.g., benign or volatile) the underwater acoustic channel is when it comes to wireless communication. Amongst others, three physical layer channel descriptors/features are explained. These are implemented in the FRSS physical layer and therefore called FRSS channel descriptors.

4.2.1. Input SNR

An intuitive channel descriptor that could be used for link adaptation, is the level of the received signal power with respect to the noise. This is called the input SNR of the received signal. It is defined as:

$$\mathrm{SNR}_{\mathrm{in}} = 10\log_{10}\left(\frac{S}{N}\right) = 10\log_{10}\left(\frac{S+N}{N} - 1\right) \; [\mathrm{dB}] \tag{4.1}$$

where S is the signal level and N is the noise level. Usually, however, the noise and signal are measured together $(S+N)$.

To obtain an estimate of the input SNR, the method is as follows. For the signal plus noise, the received signal after the preamble is taken. To get a noise-only measurement, a window of the same size is taken, but positioned just before the preamble. Both are measured within the frequency band of interest. The calculation of the input SNR has been incorporated into the FRSS receiver. The input SNR is one of the three so-called channel descriptors.

4.2.2. Estimated channel impulse response

More information about the communication channel can be obtained using the preamble. The preamble is correlated with Doppler-shifted replicas of itself to estimate the Doppler shift. However, while doing this, not only the Doppler shift is estimated, but also the channel impulse response (CIR). Similar as in Equation 2.19, the transmitted signal (in baseband, including pulse shape $p(t)$) is:

$$s_p(t) = \sum_{k=0}^{K_p-1} m_k p(t-kT) \tag{4.2}$$

The Doppler-shifted replicas available at the receiver can be written as [43]:

$$r_v(t) = D_v\left(\sum_{k=0}^{K_p-1} m_k p(t-kT)\right) \times \exp\left(-2\pi j f_c \frac{v}{c} t\right) \tag{4.3}$$

where v is the relative velocity between the transmitter and receiver and c is the nominal sound speed. The Doppler-shifts are accounted for by resampling with a factor $1+\frac{v}{c}$, indicated by operator D_v.

The received basebanded preamble $r_p(t)$ is bandpass filtered to filter out the out-of-band noise::

$$v(t) = B_{\text{tot}} \int \operatorname{sinc}\left(B_{\text{tot}}(t-\tau)\right) r_p(\tau) \mathrm{d}\tau \tag{4.4}$$

where B_{tot} is the total bandwidth.
Then, the CIR is obtained by means of matched filtering and normalizing:

$$M_v(\tau) = \frac{\frac{1}{K_p T} \int v(t+\tau) r_v^*(t) \mathrm{d}t}{\sqrt{\frac{1}{K_p T} \int_{\tau}^{\tau+K_p T} |v(t)|^2 \, \mathrm{d}t}} \tag{4.5}$$

The correlator with the highest peak amongst all the Doppler channels $|M|$ is taken as the estimated CIR.

The estimated CIR gives information about the state of the channel. It allows for the calculation of the root mean square (RMS) delay spread T_{RMS}. This is calculated using the power delay profile $P_\tau(\tau) = |h(\tau)|^2$. Where $h(\tau)$ is the estimated CIR. To define the RMS delay spread, the distribution of the power in the impulse response $p_{P_\tau}(\tau)$ is used. It is defined as [2]:

$$p_{P_\tau}(\tau) = \frac{P_\tau(\tau)}{\int_0^{\tau_{\max}} P_\tau(\tau) \mathrm{d}\tau} \tag{4.6}$$

The RMS delay spread is the standard deviation of the power delay spread. It is a measure of the power dispersion in time. This power dispersion is the cause of inter-symbol interference (ISI). The RMS delay spread is therefore an informative descriptor of the difficulty of the channel. It can be written as: $T_{\text{RMS}} = \sqrt{E\left[\tau^2\right] - E^2[\tau]}$.

The definition of the RMS delay spread is common in wireless communications. However, it is quite sensitive to long tails in the power delay profile. Another measure for the delay spread of the channel is the Γ%-energy delay spread. This is defined as [76]:

$$E(\tau) = \frac{\int_0^{\tau} P_\tau(\tau) \mathrm{d}\tau}{\int_0^{\tau_{\max}} P_\tau(\tau) \mathrm{d}\tau} \tag{4.7}$$

$$\max_{\tau_1} \min_{\tau_2} \{\tau_1, \tau_2 \mid E(\tau_2) - E(\tau_1) \geq \Gamma\} \tag{4.8}$$

In other words, this is an interval $T_{\text{en}} = \tau_2 - \tau_1$ that is as short as possible and encloses Γ% of the signal power. The calculation of the 45%-energy delay spread has been incorporated into the FRSS receiver. It is another FRSS channel descriptor. In the remainder of this work, the delay spread refers to the 45%-energy delay spread.

4.2.3. Equalizer performance

As explained in Section 2.4.3, the FRSS receiver uses a DFE-PLL structure as its equalizer. The equalization process provides insights into the difficulty of the channel and how well the transmitted signal can be reproduced.

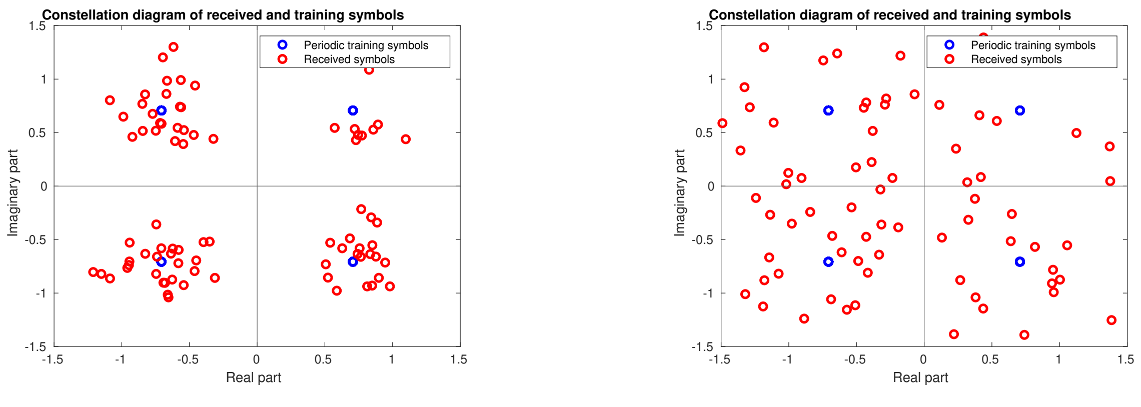
Output SNR

A measure of the performance of coherent communication methods that can be derived from the equalizer output symbols is the output SNR. This measure is inversely related to the size of the point clouds created by the periodic training symbols in the constellation diagram. The definition of the output SNR is similar as in [77]. However, in this thesis, it is corrected for by the number of subbands that is used. Therefore, it can also be called the rate-independent output SNR. It is defined as:

$$\text{SNR}_{\text{out}} = 10\log_{10}\left[\frac{\mathrm{E}\left(|s_n|^2\right)}{B \cdot \mathrm{E}\left(\left|\hat{s}_n/\gamma - s_n\right|^2\right)}\right] \tag{4.9}$$

where B is the number of subbands and $\gamma = \mathrm{E}(\hat{s}_n / s_n)$ is a factor that accounts for the bias in the equalizer. The bias is a result of the equalizer used in FRSS. The equalizer minimizes the mean squared error (MSE) between

the transmitted training symbols and received symbols. So it attempts to maximize the output SNR. This leads to a bias [78].
To illustrate the output SNR measure, Figure 4.1 shows two point clouds for a QPSK modulation for different SNR values. Figure 4.1a contains smaller point clouds, yielding a larger output SNR value. On the other hand, Figure 4.1b contains larger point clouds, so its output SNR will be smaller. The calculation of the output SNR as in Equation 4.9 has been incorporated into the FRSS receiver. This is the third FRSS channel descriptor.
It should be noted that the noise measure at the output of the equalizer does not only consist of true noise. It also includes residual ISI that was not resolved by the feedforward filter (convolutional error). Therefore, the output SNR is a measure of the effective noise, which includes both true noise and ISI. Some authors use the term signal-to-interference-plus-noise ratio (SINR) instead.

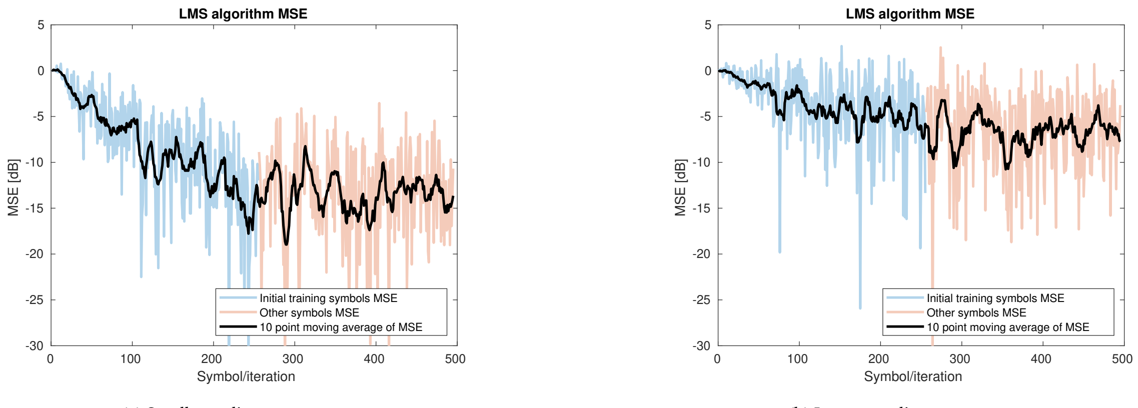


(a) Point clouds with small spread (b) Point clouds with large spread

Figure 4.1: Constellation diagrams with point clouds of estimated and training symbols.

Equalizer error

The output SNR is a measure that uses the error between the estimated training symbols and the true training symbols. However, the entire equalizer error sequence, i.e. the difference between s_k and $\hat{s}_k$ in Equation 2.23, can also be used. If the error is large, that means the equalizer has a difficult time coping with the channel. Also, if the error increases with time, it means the equalizer cannot keep up with the fast-changing channel. To illustrate what the equalizer error over time looks like, convergence plots are given in Figure 4.2. Note that the MSE can still vary significantly after convergence.



(a) Small equalizer output error. (b) Large equalizer output error.

Figure 4.2: Equalizer output error.

4.2.4. Cyclic redundancy check

Knowing if bit errors are made at the receiver, gives indirect information about the difficulty of the channel. A cyclic redundancy check (CRC) is an error-detecting technique that can be used to know if bit errors have been made or not. In FRSS, a CRC is implemented on the 24 header bits.

4.3. Method

This section describes the structure of the problem of link adaptation in more detail. An overview is given concerning the timing structure of the received channel information. Furthermore, the methodology used to find a training dataset is presented.

4.3.1. Schematical overview

Figure 4.3 shows the structure of the link adaptation problem seen from a transmitter. At time t, package p needs to be transmitted with a certain FRSS rate. The selected rate is based on information of M previous received packages. This information is assumed to be gathered from feedback messages coming from the receiver, available at the receiver at times $t-k_1$ to $t-k_M$. The time after which the feedback is received is T_{fb}. Furthermore, the small time delay T_d is used for the processing and detection delay. The length of a package depends on the chosen FRSS rate and the number of bits. The number of bits is fixed to 156 bits. Since four different rates are used, the package length can take one of four values: $T_{P_i} \in \{T_{R_1}, T_{R_2}, T_{R_3}, T_{R_4}\}$.

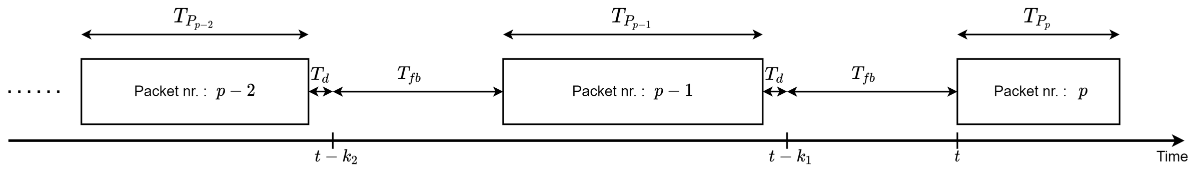


Figure 4.3: Schematic overview of link adaptation structure in time, seen from the transmitter side.

For the remainder of this work, T_d is considered negligible compared to T_{fb}. Thus, the time factor k_m is defined as:

$$k_m = \begin{cases} T_{fb} & \text{if } m = 1 \\ m \cdot T_{fb} + \sum_{i=2}^{m-1} T_{P_{p-i+1}} & \text{if } m > 1 \end{cases} \tag{4.10}$$

The schematic overview of the link adaptation algorithms used in this chapter can be shown as in Figure 4.4. Here $\mathbf{d}(t-k_1)$ contains the information of the previous transmission and so on.

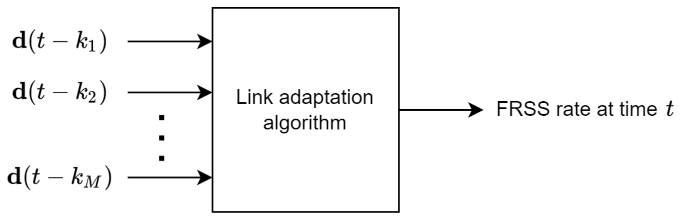


Figure 4.4: Schematic overview of link adaptation algorithm inputs and output.

The motivation behind the assumption of information feedback is based on the network protocol used in combination with FRSS. The network protocol called Gossiping in Underwater Acoustic Mobile Ad-hoc Networks (GUWMANET) [79] has been been used in combination with FRSS before [7].
When implemented in a node network, the protocol uses multiple hops to communicate between nodes. In a typical network, a receiving node would send a feedback acknowledgement packet back to the transmitting to indicate that it received the message correctly. Then, it would transmit its message to the next node.
However, in GUWMANET implicit feedback acknowledgement messages are implemented. This can be done due to the sharing of the medium in the underwater wireless network. The transmitting node overhears if the next hop forwards its message. Therefore, it gets an implicit acknowledgement. In GUWMANET, the implicit link feedback can also include channel descriptors. For example, assume node A transmits to node C via node B. Node B receives the packet. Then, node B transmits the packet to node C and adds a channel descriptor value about the link quality between node A and B. Node A overhears this transmission and therefore it receives implicit feedback [80].

4.3.2. Dataset

With the underwater communication simulation method introduced in Section 2.5, a dataset can be created that can be used for training and testing classifiers. These classifiers will attempt to select the optimal FRSS rate. The dataset contains the channel descriptors or features from M previous transmissions, as was introduced in Figure 4.4. It also contains the optimal next rate, which is the output in the figure.
The entire dataset will be split up into 85% training data and 15% testing data.

The pseudocode of the algorithm implemented to obtain the dataset is given in Algorithm 1. The dataset is created for a fixed delay time and lookback, as can be seen in line 1. The lookback is the number of previous transmission features the classifier takes into account. Next, the main loop is started, which runs until a satisfactory amount of dataset entries is reached. In line 4 to 6, a channel segment is initiated. Then, in line 7, a loop is started which keeps transmitting packages in the channel segment until there is no time left in the channel. In the meantime, the channel features and optimal rate are stored in the dataset. As stated in line 9 and 11, the channel features are obtained from packet transmissions with random, receivable, FRSS rates. The word receivable indicates that the packet can be detected at the receiver and thus channel features can be obtained. An exception is when the channel is so volatile that none of the FRSS rates are succesful. Then, the features in line 11 are set to predefined values.
When there is not enough time left in the recorded channel to continue this process, the next channel is taken and the process continues.

Algorithm 1 Generate LA dataset

```
 1: procedure GENERATE_LA_DATASET($T_{fb}$, $M$)
    ▷ Creates dataset with nominal feedback delay $T_{fb}$ and lookback $M$
 2:
 3:     while $i$ < desired_dataset_size do                    ▷ Repeat until enough data is gathered
 4:         [$c, t$, stop, $T_{tot}$] = INITIALIZE()   ▷ Initialize $t$ and stop = 0 and $T_{tot}$ = total time in channel segment $c$
 5:         $SNR_{mean}$ = rand(-5,25)                                          ▷ Mean SNR in dB
 6:         $SNR_{std}$ = rand(0,1)                                   ▷ Standard deviation SNR in dB
 7:         while stop $\neq$ 1 do
 8:             for $j$ = 1 to $M$ do
 9:                 rate = RANDOM_RATE($c, t, SNR_{mean}, SNR_{std}$)      ▷ Select a random, receivable, FRSS rate
10:                 $T_R$ = RATE_DURATION(rate)
11:                 features[j,:] = SIMULATE_FRSS($c, t$, rate, $SNR_{mean}, SNR_{std}$)      ▷ Get channel features
12:                 $t = t + T_R + T_{fb}$                                  ▷ Update time in channel
13:             end for
14:             rate_opt = FIND_OPTIMAL_FRSS_RATE($c, t, SNR_{mean}, SNR_{std}$)      ▷ Find the optimal FRSS rate
15:             dataset[$i$] = [features, rate_opt]                                ▷ Save datapoint
16:             $i = i + 1$
17:             if $t$ + RATE_DURATION(rate = 4) > $T_{tot}$ then
18:                 stop = 1         ▷ If there is no time left in the channel segment to send a new packet, stop
19:             end if
20:         end while
21:     end while
22:
23:     return dataset
24: end procedure
```

4.3.3. Performance

To compare different strategies of classifiers to switch between FRSS rates, the average effective bitrate over the testing dataset is used as the performance metric. The bitrate BR in bits per second (bps) for each element in the testing set is defined as:

$$\mathrm{BR} = \begin{cases} \mathrm{nBits}/T_{P_i} & \text{if PDR} = 1 \\ 0 & \text{if PDR} < 1 \end{cases} \tag{4.11}$$

where PDR is the packet delivery ratio, which is 1 if there are no biterros after transmitting with the selected FRSS rate and < 1 otherwise.
Five categories for link adaptation are used. Four for the four different FRSS formats and one for the case no successful transmission is possible using any FRSS rate. For testing, the bitrate of that category is set to be equal to the bitrate of an FRSS-4 packet. This is done as in practice a rate 4 packet will still be sent when no FRSS format is expected to work. Furthermore, feedback information is still needed to decide the best next transmission format and no information can be received if no message is transmitted.

4.4. Results

In this section, results concerning the link adaptation part of this thesis will be given. First, different machine learning strategies are trained using multiple inputs and for various conditions. Then, the best methods are compared using other validation channels.

4.4.1. Benchmark strategies

In order to compare different classification methods, first some benchmark strategies and values are shown in Figure 4.5. The first value is the maximum achievable bitrate, which is the bitrate achieved when the lowest FRSS rate with a PDR of 1 is always selected. The second method is choosing the FRSS rate at random. The third and fourth strategies are selecting only rate 2 and 3, respectively. Lastly, a switching method based on the output SNR value of the previous transmission is also shown. The output SNR thresholds for this method are the same as given in [81]. This is a benchmark method, since it has already been implemented by TNO. Note that the thresholds for this method have been optimized for a PDR of 0.8 [80] and not for a PDR of 1.0 as used in this thesis.
As expected, the method of selecting FRSS rates at random has a poor performance. Furthermore, methods three and four have mediocre performance. This is exactly why the link adaptation classification methods are implemented. The last strategy has the best performance, indicating it selects the optimal FRSS rate more often.

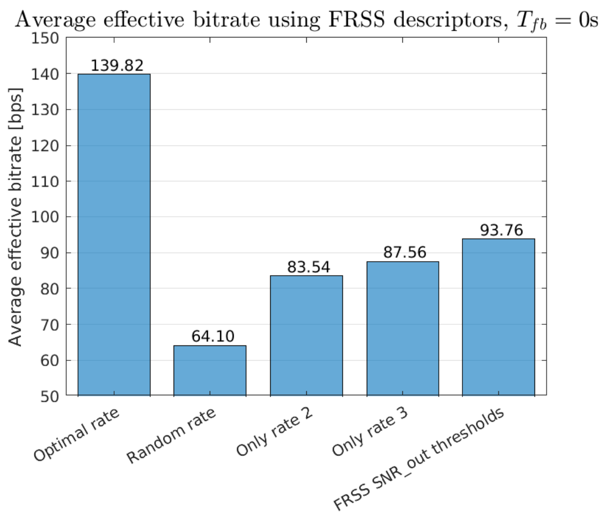


Figure 4.5: Average effective bitrate for benchmark methods.

4.4.2. Classification with FRSS descriptors

The first classification method that is implemented is a decision tree. The results are shown in Figure 4.6. The goal of this test is to verify if the FRSS descriptors provide enough information to effectively switch between rates. Furthermore, the results should provide insight about the usefulness of each descriptor. For this test, the feedback delay $T_{fb} = 0$ s and $M = 1$. The tree uses one or more of the FRSS descriptors and has a maximum depth of six levels. The results were generated using ten times repeated random sub-sampling validation. This also holds for all other results in this chapter.

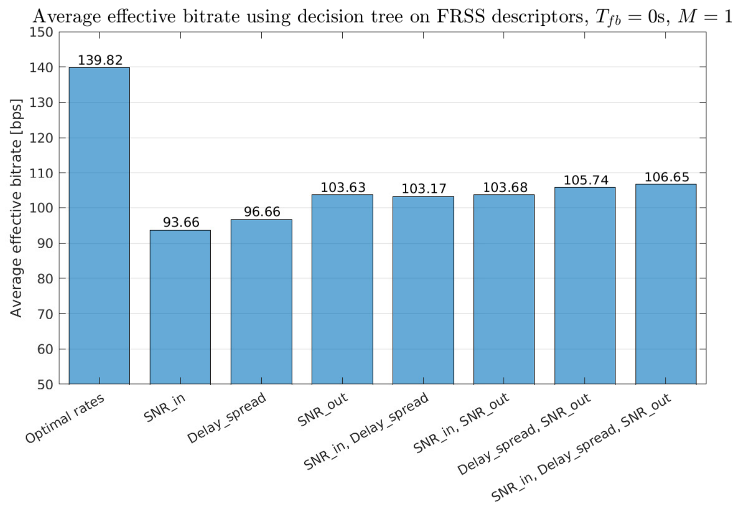


Figure 4.6: Average effective bitrate using decision tree classifier on FRSS descriptor values.

As can be seen from Figure 4.6, out of the three FRSS descriptors, the output SNR is the most informative when it comes to rate switching. Using all three descriptors provides a small increase in performance. Overall, it can be concluded that the FRSS descriptors are useful for link adaptation. The decision tree approach results in higher bitrates than the benchmark methods.
When comparing the average effective bitrate of the output SNR based DT classifier in Figure 4.6 with the benchmark FRSS output SNR method in Figure 4.5, it can be seen that the former achieves a better performance than the latter. This is caused by the fact that the FRSS output SNR method thresholds are based on a PDR of 0.8, as explained in Section 4.4.1.

To further analyze the FRSS descriptors, a correlation plot is shown in Figure 4.7. The input and output SNR values have a positive correlation of 0.49. The reason for this is that for channels with more noise (i.e. lower input SNR), error-less communication is harder. In that case, the equalization will be harder as well, resulting in a lower output SNR. It should be noted that the received signal level should be higher than a certain level to even have a change at successful communication, apart from the channel transfer function.
The correlation between the output SNR and delay spread is negative. This means a larger delay spread value can decrease the output SNR. This is due to an increase in ISI and thus a harder task for the equalizer.
All in all, it makes sense that the output SNR is the best basis for rate selection. It gives information about the difficulty of the channel and includes information about the input SNR and delay spread.

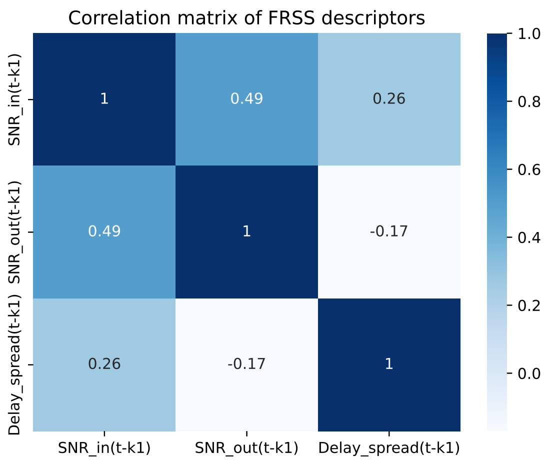


Figure 4.7: Correlation matrix of FRSS descriptors.

To get more insight into the difficulty of classifying FRSS rates, Figure 4.8 shows the pairwise relation between the FRSS descriptors for a part of the training dataset. In general, it is clear that the boundaries between areas with the same optimal rate overlap. Therefore, it will be impossible to perfectly select the optimal rate using a linear boundary based on FRSS descriptors. However, a sub-optimal solution can be found by selecting the correct decision boundaries.
It should be noted that the large spikes in the diagonal plots are originate from predefined values in case the previous transmission was not detected. That is, if FRSS rate 4 was selected in line 9 of Algorithm 1, but the package was not detected. That means no descriptor values can be calculated. This can also happen in practice. Therefore, it is needed to include this case in the training dataset.

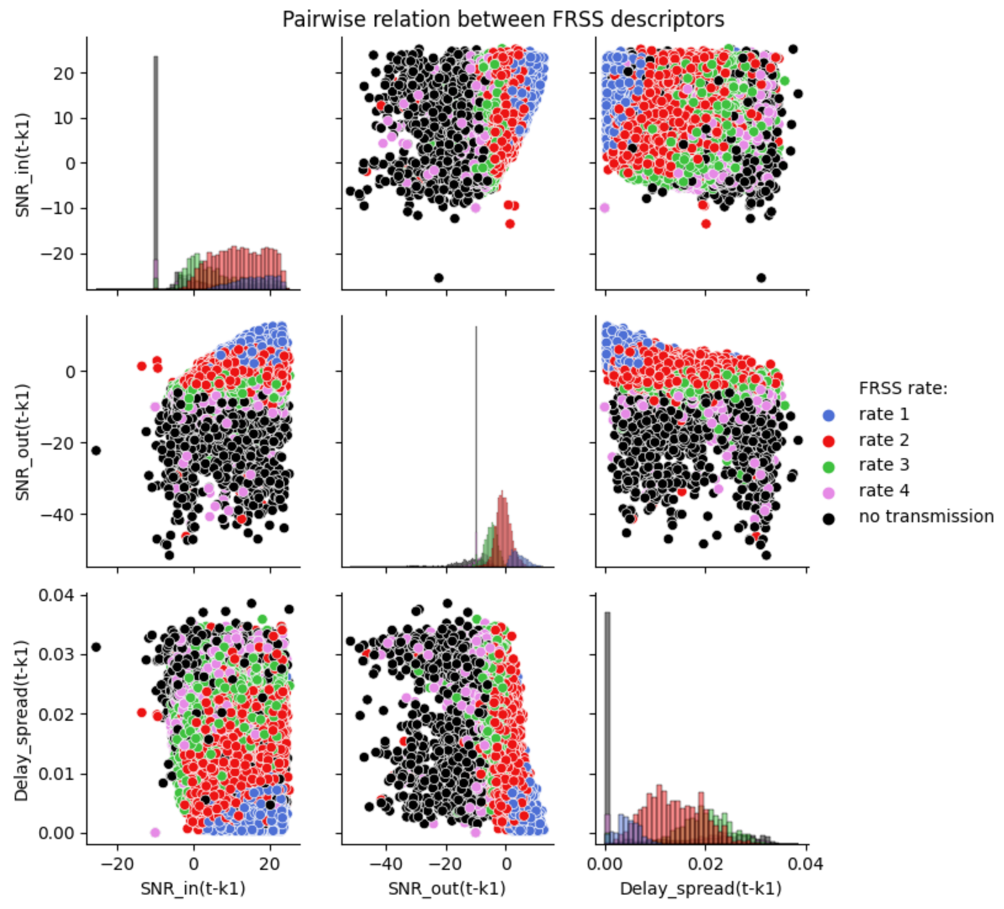


Figure 4.8: Pairwise relation between FRSS descriptors.

In order to select the decision boundary in the best way possible, various machine learning classifiers are tested. The results are shown in Figure 4.9. The methods use all three FRSS descriptors. Three different SVMs have been implemented. They have linear, (fifth order) polynomial and Gaussian radial basis function kernels. Furthermore, a 50-Nearest Neighbours algorithm is executed as well. The decision tree and random forest classifiers have a maximum depth of 6 layers. Finally, the neural network has layer sizes: [3,32,16,8,5] with a hyperbolic tangent as activation function and a softmax function after the last layer.
From Figure 4.9, it can be concluded that classifiers that draw linear boundaries between the descriptor values perform well. Besides the linear classifiers, the non-linear neural network can reach the same performance level.

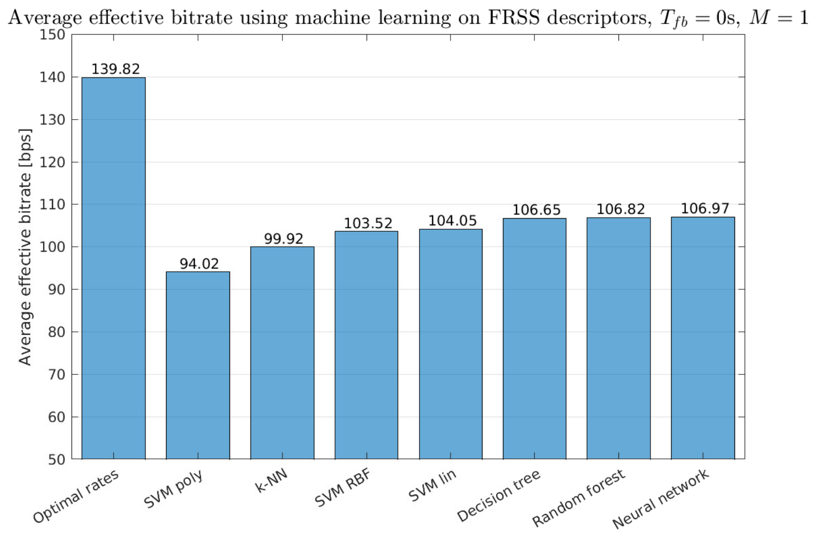


Figure 4.9: Average effective bitrate using machine learning classifiers on FRSS descriptor values.

Two parameters that can influence the performance of the classifier will be analyzed next. These are the feedback delay T_{fb} and lookback M. The goal of this test is to show how their values affect the performance. The results are shown in Figure 4.10. A decision tree is used for classification, but the results are similar for the other classification methods. Again, all three FRSS descriptors are used.
It can be concluded that for longer feedback delays, the classifiers perform worse. This is because the feedback information is more outdated. Due to limitations on the length of the recorded channel impulse responses, delays larger than 3 seconds are not tested. Increasing M to a value of 3 did not provide a gain in terms of bitrate. A reason for this can be that the extra feedback is either outdated or the same as already received from the last transmission.

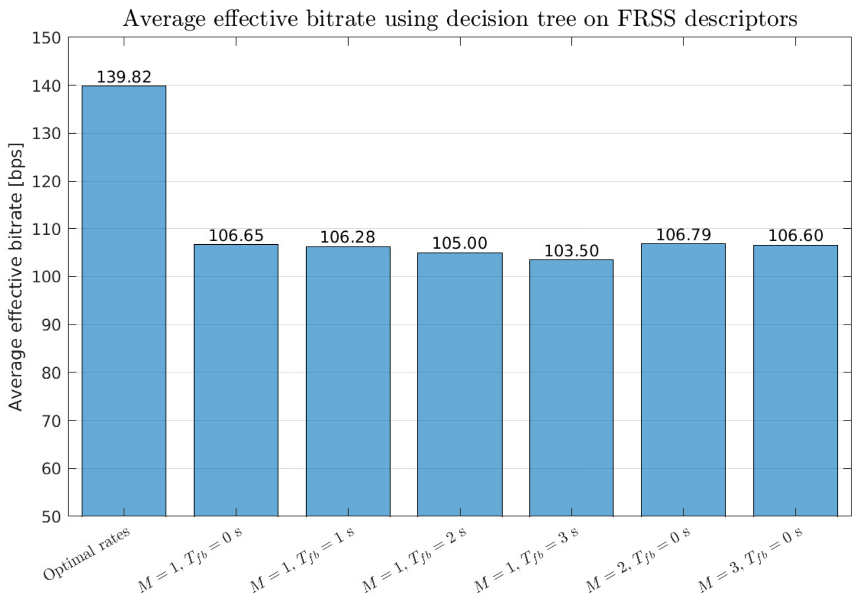


Figure 4.10: Average effective bitrate using decision tree classifier on FRSS descriptor values for various feedback delays and lookbacks.

4.4.3. Classification with alternative features

From the previous results, it can be seen that the performance of classifiers based on the FRSS descriptors has a limit of around 105 bps. From Figure 4.8, it could be seen that this is due to the overlapping boundaries between rate groups. Therefore, other low-level features are created in an attempt to separate groups more clearly.

In order to do this, a neural network is implemented. The inputs of the network are the CIR sequence and/or the equalizer error sequence, as this information is already calculated in the FRSS physical layer chain. The link adaptation neural network can be decomposed into two parts. The first block extracts low-level features from the CIR and/or equalizer error sequences. The second block chooses the optimal next rate based on those features, similar to what was done previously. The number of neurons per layer was chosen such that it decreased with powers of two. For the activation functions, the ReLu function was implemented. The results are shown in Figure 4.11. As can be seen, the neural network that uses both the CIR and equalizer error is able to reach the highest performance yet.

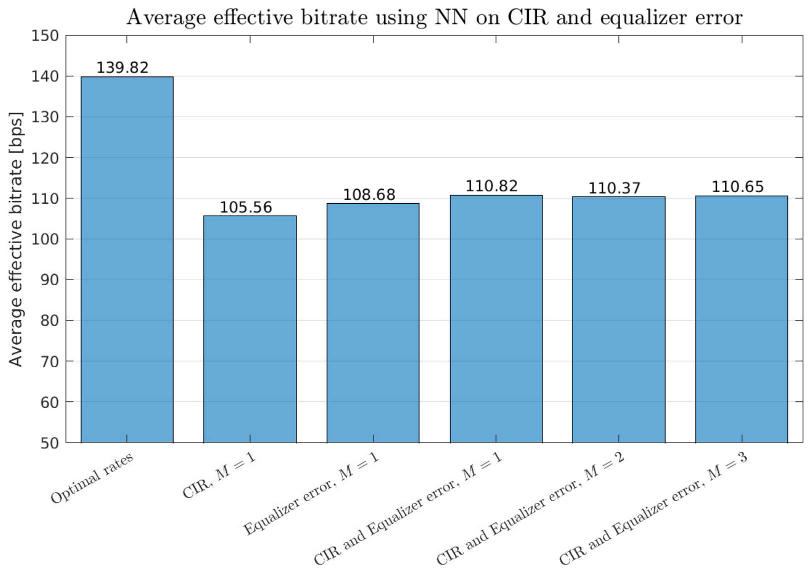
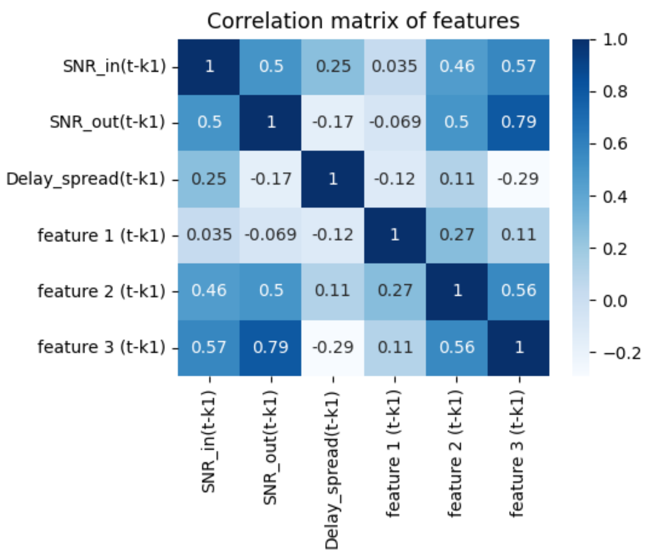


Figure 4.11: Average effective bitrate using neural network classifier on CIR and equalizer error.



	SNR_in(t-k1)	SNR_out(t-k1)	Delay_spread(t-k1)	feature 1 (t-k1)	feature 2 (t-k1)	feature 3 (t-k1)
SNR_in(t-k1)	1	0.5	0.25	0.035	0.46	0.57
SNR_out(t-k1)	0.5	1	-0.17	-0.069	0.5	0.79
Delay_spread(t-k1)	0.25	-0.17	1	-0.12	0.11	-0.29
feature 1 (t-k1)	0.035	-0.069	-0.12	1	0.27	0.11
feature 2 (t-k1)	0.46	0.5	0.11	0.27	1	0.56
feature 3 (t-k1)	0.57	0.79	-0.29	0.11	0.56	1

Figure 4.12: Correlation matrix of FRSS descriptors and low-level features.

In order to gain insight into the features extracted by the neural network, a correlation plot is shown in Figure 4.12. As can be seen, three features were created. The feature 1 dimension was not used by the neural network. So the network was able to create two low-level features that contain enough information to make a choice about the best FRSS rate. These features largely correlate with the FRSS descriptors, as can be seen from the figure.

Figure 4.13 shows the confusion matrix of the neural network classifier using the CIR and equalizer error as input. The matrix shows the normalized number of times the predicted label was the same as the true label. For clarity, only the four FRSS rates are shown as classes here. From the figure, it is clear that the classification is the hardest for FRSS rate 3. In 26% of the cases, rate 2 was chosen over rate 3. Other sources of a decreased performance are the cases where rate 2 was chosen over rate 1 and where rate 3 was chosen over rate 4.

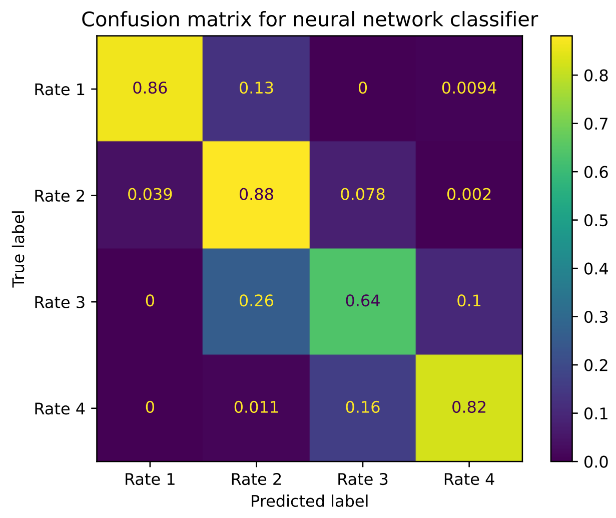


Figure 4.13: Confusion matrix of the neural network classifier. The values are normalized for each true label.

4.4.4. Classification for example channels

The algorithms tested in the previous section only deviate with a maximum of around 10 bps compared to each other. To see more clearly what the difference in performance looks like, the algorithms are tested on additional test channels. This is done for a range of SNR values. In these tests, the decision method with all three FRSS descriptors and the neural network method with the CIR and equalizer error are compared.
As an additional benchmark strategy, a method that switches based on the CRC is implemented. It is given in Algorithm 2 [82]. The CRC feedback of the initial two messages is gathered using rate 4 packages. FRSS rate 1 is not included in this method, as it was found that incorporating rate 1 in the method decreases the overall performance.

Algorithm 2 Link adaptation algorithm using CRC

```
1:  procedure LA_CRC($k$, $\mathbf{CRC}$, FRSSrate)
2:
3:      if CRC($k-2$) == 1 and CRC($k-1$) == 1 then
4:          FRSSrate = max(FRSSrate − 1, 2)                ▷ Try a lower FRSS rate
5:      else if CRC($k-2$) == 0 and CRC($k-1$) == 1 then
6:          FRSSrate = FRSSrate                            ▷ Keep the current FRSS rate
7:      else
8:          FRSSrate = min(FRSSrate + 1, 4)                ▷ Try a higher FRSS rate
9:      end if
10:
11:     return FRSSrate
12: end procedure
```

N9N8

The first example channel is called N9N8. This channel is recorded between bottom nodes 9 and 8 in the same setup as given in Section 2.5, but on another day. The channel has a low delay spread and Doppler spread. Further specifics of the channel are shown in Figure 4.14.

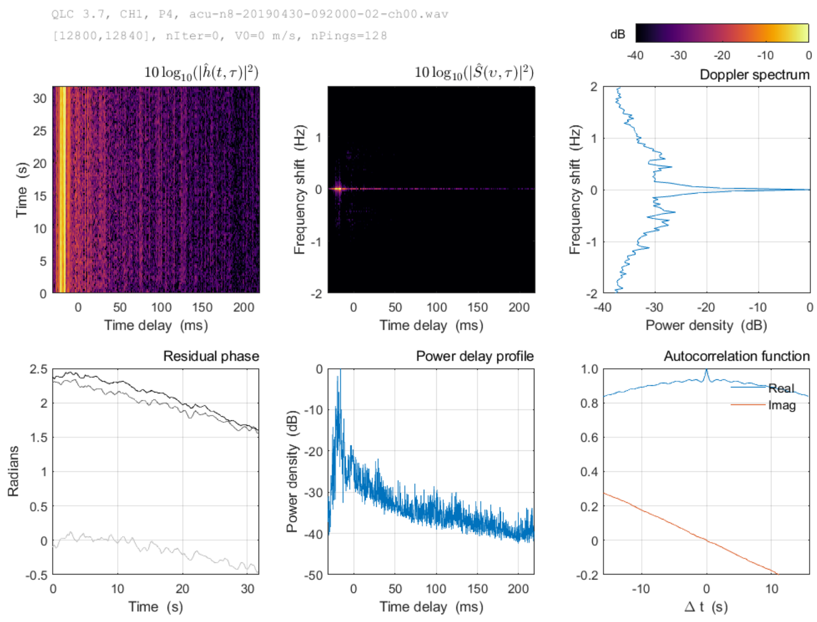
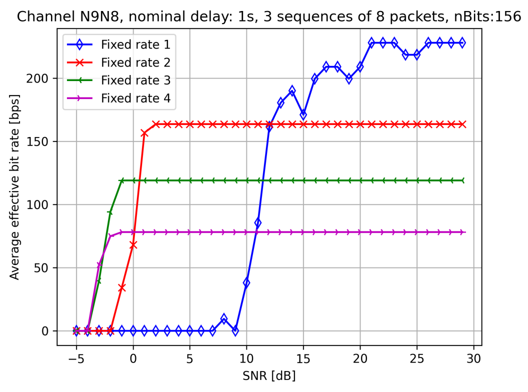
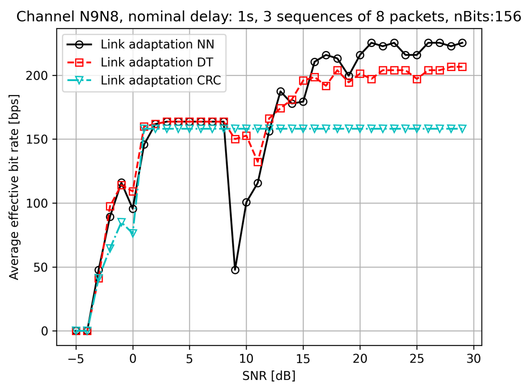


Figure 4.14: Channel N9N8.

Figure 4.15 shows the performance of different communication strategies in this channel. The decision tree (DT) and deep neural network (DNN) methods generally outperform the simple CRC method. However, the DNN method attempts to use FRSS rate 1 at a too low SNR value. This is the cause of the sudden decrease in performance. On the other hand, the DNN outperforms all other methods at high SNR. This is because the network efficiently switches between FRSS rate 1 and 2.



(a) Fixed FRSS rates.



(b) Link adaptation methods.

Figure 4.15: Performance of ML link adaptation techniques compared to fixed FRSS rates in channel N9N8.

N6N4

The second example channel is called N6N4. This channel is recorded between bottom nodes 6 and 4 in the same setup as given in Section 2.5, but on another day. The channel has a modest delay spread and low Doppler spread. Further specifics of the channel are shown in Figure 4.16.

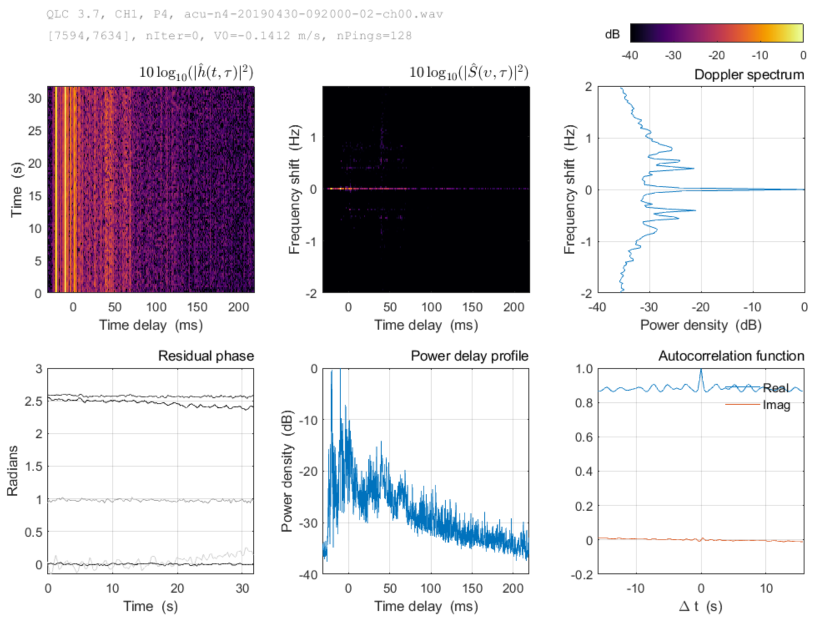
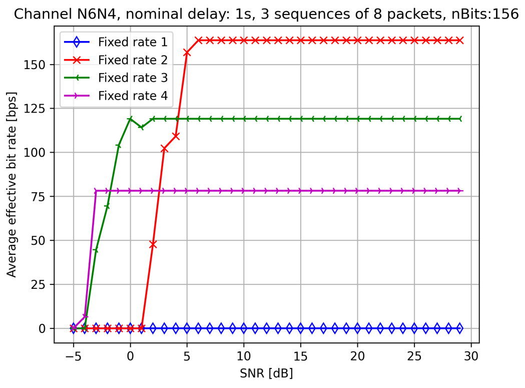
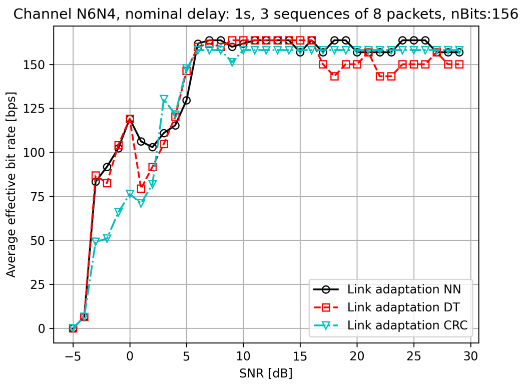


Figure 4.16: Channel N6N4.

Figure 4.17 shows the performance of different communication strategies in this channel. The DT and DNN methods generally outperform the simple CRC method for low SNR values. Since FRSS rate 1 does not work in this channel, the highest possible FRSS rate is 2. This is mostly selected correctly by the different methods.



(a) Fixed FRSS rates.



(b) Link adaptation methods.

Figure 4.17: Performance of ML link adaptation techniques compared to fixed FRSS rates in channel N6N4.

N5N8

The third example channel is called N5N8. This channel is recorded between bottom nodes 5 and 8 in the same setup as given in Section 2.5, but on another day. The channel has a significant delay spread. Further specifics of the channel are shown in Figure 4.18.

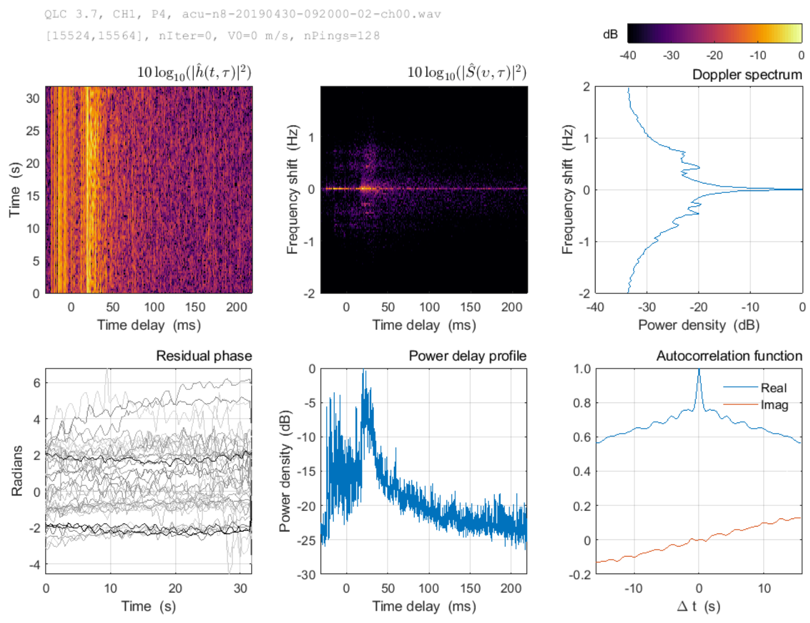
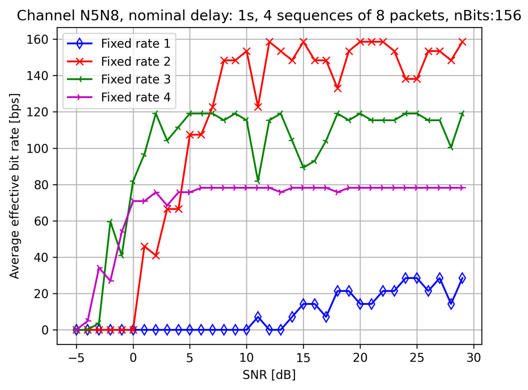
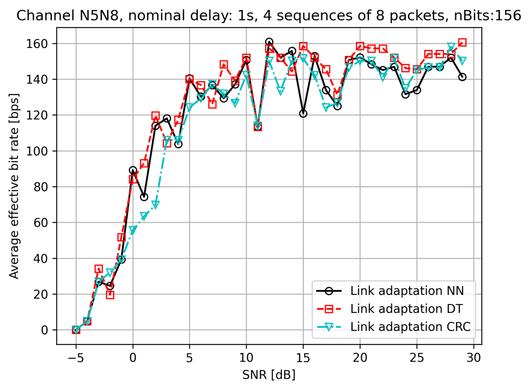


Figure 4.18: Channel N5N8.

Figure 4.19 shows the performance of different communication strategies in this channel. For this channel, the performance of all link adaptation methods is similar.



(a) Fixed FRSS rates.



(b) Link adaptation methods.

Figure 4.19: Performance of ML link adaptation techniques compared to fixed FRSS rates in channel N5N8.

MNN2

The last example channel is called MNN2. This channel is recorded between a moving node and a stationary bottom node. Details can be found in [69]. The channel has significant delay and Doppler spreads. Further specifics of the channel are shown in Figure 4.20.

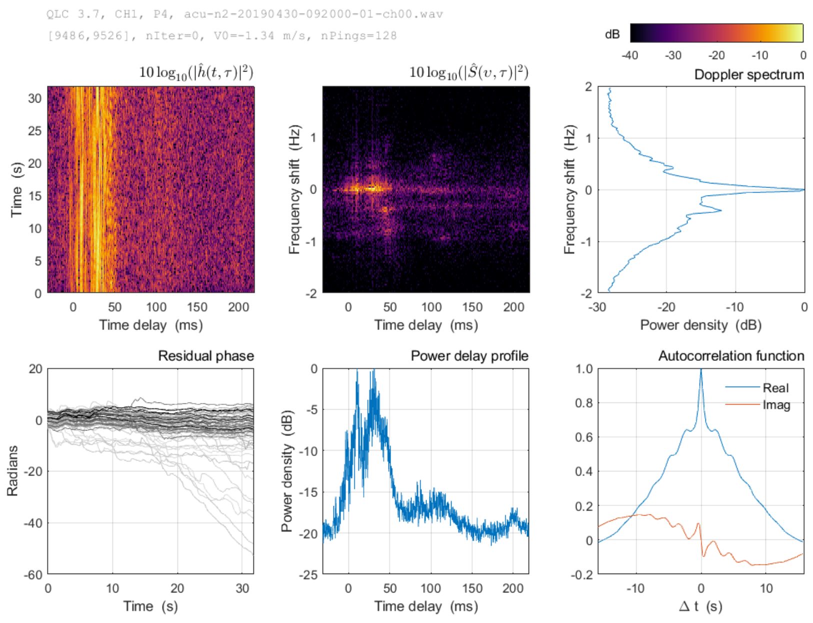
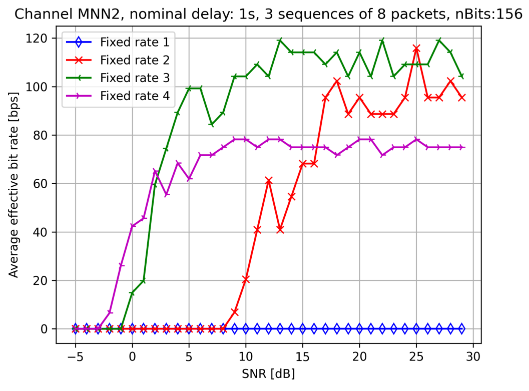
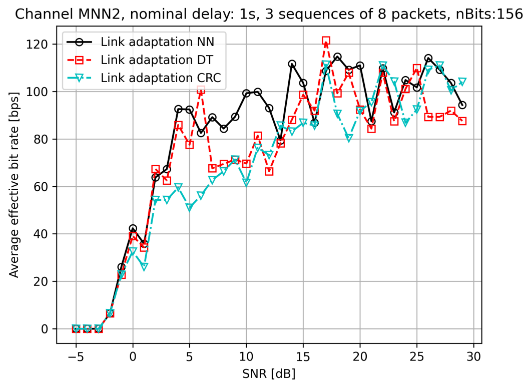


Figure 4.20: Channel MNN2.

Figure 4.21 shows the performance of different communication strategies in this channel. It can be seen that the neural network method has the best performance of all methods. Especially, between 3 and 12 dB SNR, the DNN classifier performs better than the other methods. This due to the fact that the network switches more efficiently between FRSS rate 3 and 4.



(a) Fixed FRSS rates.



(b) Link adaptation methods.

Figure 4.21: Performance of ML link adaptation techniques compared to fixed FRSS rates in channel MNN2.

4.5. Computational complexity

As can be seen from the previous results, the performances of the decision tree and neural network methods do not differ a lot. There is, however, a significant difference in terms of computational complexity. It is important to take the classification computational complexity into consideration when talking about link adaptation algorithms. This has to do with the fact that the algorithms have to be implemented in hardware on each underwater platform. For example, on bottom nodes and underwater autonomous vehicles (UAVs). The computational complexity of each classification method is summarized in Table 4.1.

Table 4.1: Computational complexity of classification methods.

Method	**Classification computational complexity**
If/else statement	$O(1)$
Decision tree	$O(\text{depth})$
Neural network	$O(N_{\text{in}}N_{\text{out}}) + O(N_{\text{out}}) + \cdots$

The benchmark link adaptation method of using the thresholds on the output SNR and the method of using the CRC have low computational complexity. Essentially, they consist of a single if/else statement. Therefore, the complexity will be $O(1)$.
A decision tree consists of multiple levels. At each level, a decision has to be made based on a threshold for a single input value. This is essentially an if/else statement. Therefore, the classification computational complexity is dependent on the tree depth as $O(\text{depth})$.
The computational complexity of a neural network depends on the network structure. A fully connected network with activation functions after each layer will be considered here. A forward pass over each layer is a multiplication of a weight matrix with the input. The weight matrix has a size N_{in} by N_{out}. So simply put, for each layer the computational complexity is $O(N_{\text{in}}N_{\text{out}}) + O(N_{\text{out}})$, where the second term is due to the activation layer. Therefore, a network with many layers and a large input size has a far greater computational complexity than a simple decision tree classifier.

5
Machine learning aided equalization optimization

5.1. Introduction

This chapter describes the equalization optimization problem in more detail. A description of different equalizer filter update algorithms is given. Furthermore, the optimization process of the algorithm parameters is explained. Lastly, acquired results for each algorithm for various input SNR values are shown.

5.2. Method

5.2.1. Equalizer filter update

As was explained in Section 2.4.3, the equalizer in the FRSS physical layer has a decision feedback structure. For a symbol k, the input of the equalizer is the fractionally spaced received signal $\mathbf{y}_k$. From this input, the goal is to estimate the true or transmitted symbol s_k. To achieve this goal, a filter can be used. For example, a linear filter $\mathbf{c}_k$, such that the symbol is estimated as $\hat{s}_k = \mathbf{y}_k^H \mathbf{c}_k$, where ^{H} indicates the Hermitian transpose.

Out of the four FRSS rates, the equalization of rate 3 messages was studied in this thesis. For a rate 3 packet, the input $\mathbf{y}_k$ has a length of 91 (13 fractions for each of the 7 frequency subbands). The methodology followed in this chapter can also be applied to the other FRSS rates.

To find the optimal filter coefficients, many algorithms can be used. Due to the changing channel, it is beneficial for the filter to be updated adaptively. Therefore, adaptive algorithms will be studied in this chapter. The equalizer attempts to estimate the symbol as accurately as possible. In order words, the value of a cost function that measures the error between the true and estimated symbol has to be minimized. Often times, the cost function is taken to be the mean squared error; $C(\mathbf{c}_k) = E[|s_k - \hat{s}_k|^2] = E[|e_k|^2]$.
Newton's method provides an update equation for the filter coefficients using a second-order Taylor expansion. The update is written as:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \mu_k H(\mathbf{c}_k)^{-1} \nabla C(\mathbf{c}_k) \tag{5.1}$$

where μ_k is the step size parameter and $H()$ is the Hessian matrix.
Gradient descent uses only a first-order Taylor expansion and has the following update form:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \mu_k \nabla C(\mathbf{c}_k) \tag{5.2}$$

The structure of both of these equations can be found back in the algorithms explained next.

5.2.2. Linear filter algorithms

In this section, various linear filters are introduced.

LMS

The least mean squares (LMS) uses a gradient-descent-based update equation. In the LMS algorithm, the gradient of the cost function is estimated by the instantaneous error: $e_k = s_k - \hat{s}_k$. Also, it uses a fixed step size. Therefore, the LMS algorithm is described by:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \mu \mathbf{y}_k^* e_k \tag{5.3}$$

Instead of using the instantaneous gradient estimate, it is also possible to estimate the gradient using information from more than one symbol. For example, the update equation can be written as:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \sum_{p=0}^{P-1} \mu_l \mathbf{y}_{k-p}^* e_{k-p} \tag{5.4}$$

where P symbols are used. In this case, $\boldsymbol{\mu}$ is a vector. In other words, more step size parameters need to be found.

Normalized LMS

A modification to the LMS algorithm is made in the normalized LMS (NLMS) algorithm. As the name implies, the step size is normalized in NLMS. The corresponding equation is:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \frac{\mu}{\mathbf{y}_k^H \mathbf{y}_k} \mathbf{y}_k^* e_k \tag{5.5}$$

The normalization of the step size ensures that the algorithm is not susceptible to input power changes.
In Section 2.4.3, it was explained that the received signal is already normalized per subband. Since the sequences of each subband are put together in one vector $\mathbf{y}_k$, this does not mean that $\mathbf{y}_k$ is normalized. Therefore, for NLMS, the normalization is not done for each frequency band, but for vector $\mathbf{y}_k$.

VS-LMS by Harris et al.

The LMS and NLMS algorithms use a single step size parameter in the update equation. A variable step size LMS (VS-LMS) algorithm was proposed by Harris et al. [83]. The algorithm uses individual step size parameters as opposed to a single parameter. The algorithm is described as:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + 2 \cdot \mathrm{diag}\left[\mu_{k,0}, \mu_{k,1}, \ldots \mu_{k,L-1}\right] \mathbf{y}_k^* e_k \tag{5.6}$$

where diag[ ] is a diagonal matrix with its diagonal elements in the vector in the brackets. The algorithm can provide faster convergence and a smaller error than the LMS algorithm. However, it comes at the cost of additional parameters.

VS-LMS by Mikhael et al.

The previous algorithm still requires the selection of appropriate step size parameters. A variable step size LMS algorithm proposed by Mikhael et al. [84] does not require any parameters. Furthermore, it uses individual step sizes, as can be seen in the update equation:

$$\begin{aligned} \mathbf{c}_{k+1} =& \mathbf{c}_k + \mathrm{diag}\left[\mu_{k,0}, \mu_{k,1}, \ldots \mu_{k,L-1}\right] \mathbf{y}_k^* e_k \\ \mu_{k,i} =& \frac{0.5|y_{k,i}|}{\sum_{l=0}^{L-1} |y_{k,l}|^3}, \quad i = 0, \ldots, L-1 \end{aligned} \tag{5.7}$$

Similar as with NLMS, the normalization for each frequency band is omitted. The algorithm already has a normalization step in the calculation of the step sizes.

Momentum

Momentum is an extension of the previously introduced algorithms. Using momentum can prevent oscillatory behaviour of noise gradients in the optimization space. This can improve the performance of the algorithm or improve the convergence speed of the algorithm. For LMS, the equations are:

$$\begin{aligned} \mathbf{v}_k &= \beta \mathbf{v}_{k-1} + (1-\beta) \mathbf{y}_k^* e_k \\ \mathbf{c}_{k+1} &= \mathbf{c}_{k-1} + \mu \mathbf{v}_k \end{aligned} \tag{5.8}$$

where $\beta \in [0,1]$ is the momentum parameter. The same structure can be used for the other algorithms.

AdaGrad

Another well-known adaptive algorithm is AdaGrad, coming from Adaptive Gradient. This algorithm is one of the most popular algorithms for machine learning applications. It has a similar structure as the previous equations. When taking $\mathbf{y}_k^* e_k$ as the gradient estimate, it has the following structure:

$$\begin{aligned} \mathbf{c}_{k+1} &= \mathbf{c}_k + \mu\, \mathrm{diag}(\epsilon \mathbf{I} + \mathbf{G}_k)^{-1/2} \mathbf{y}_k^* e_k \\ \mathbf{G}_k &= \gamma \mathbf{G}_{k-1} + (1-\gamma) \mathbf{y}_k^* e_k (\mathbf{y}_k^* e_k)^H \end{aligned} \tag{5.9}$$

where diag() is a diagonal matrix formed using the diagonal elements of the matrix in the curled brackets. Furthermore, $\mathbf{I}$ is the identity matrix, γ is a momentum parameter and ϵ is a small constant. So AdaGrad adapts the step size parameter individually using a sequence of gradient estimates.

RLS

Besides the LMS algorithm, another well-known algorithm for adaptive filtering is the recursive least squares (RLS) algorithm. The algorithm has a greater computational complexity than the LMS algorithm. The RLS algorithm is defined by the following equations:

$$\begin{aligned} &\mathbf{g}_k = \mathbf{P}_k \mathbf{y}_k \{\lambda + \mathbf{y}_k^H \mathbf{P}_k \mathbf{y}_k\}^{-1} \\ &\mathbf{P}_{k+1} = \lambda^{-1} \mathbf{P}_k - \mathbf{g}_k \mathbf{y}_k^H \lambda^{-1} \mathbf{P}_k \\ &\mathbf{c}_{k+1} = \mathbf{c}_k + \mathbf{g}_k^* e_k \end{aligned} \tag{5.10}$$

where λ is a constant called the forgetting factor.

5.2.3. Non-linear filter algorithms

The previously discussed algorithms provided methods to obtain a linear adaptive filter. Although linear filters have been implemented successfully in many applications, other works show that the use of non-linear filters in the decision feedback equalizer can improve the performance of the equalizer [46–48] compared to linear filters. Two non-linear filters are considered, a kernel adaptive filter and a neural network filter. The non-linear filters estimate the symbol using a function $f()$ as; $\hat{s}_k = f(\mathbf{y}_k)$.

Kernel adaptive filter

In the kernel adaptive filter algorithm, the input $\mathbf{y}_k$ is transformed from the input space to a, possibly non-linear, high dimensional feature space as $\phi(\mathbf{y}_k)$. Where $\phi()$ has the property that the inner products can be computed using a positive definite kernel function that satisfies Mercer's conditions [85]; $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i^H)\phi(\mathbf{x}_j)$. Similar to LMS, the symbol can then be estimated using a filter as: $\hat{s}_k = \phi(\mathbf{y}_k)\mathbf{c}_k$. However, a catch is that $\mathbf{c}_k$ is a high dimensional feature space and therefore it would be impractical to update $\mathbf{c}_k$ as:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \mu \phi(\mathbf{y}_k) e_k \tag{5.11}$$

To circumvent this problem, a similar kernel trick can be used as for the SVM in Section 3.2. The first step is to note that the filter updates can be written as a sum:

$$\mathbf{c}_k = \mu \sum_{i=0}^{k-1} \phi(\mathbf{y}_i) e_i \tag{5.12}$$

provided $\mathbf{c}_0$ is a vector of all zeros. Then, the symbol estimate is [86]:

$$\hat{s}_k = \mu \sum_{i=0}^{k-1} \phi(\mathbf{y}_i)^H \phi(\mathbf{y}_k) e_i = \mu \sum_{i=0}^{k-1} k(\mathbf{y}_i, \mathbf{y}_k) e_i \tag{5.13}$$

this is called the kernel LMS (KLMS) algorithm, which uses the so-called kernel trick. In this equation, it is not required anymore to know $\phi()$ to update $\mathbf{c}_k$. For the kernel $k()$, the Gaussian kernel is used:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp - \left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right) \tag{5.14}$$

Neural network
The neural network adaptive filter estimates the symbol by means of a non-linear function:

$$\hat{s}_k = f(\mathbf{y}_k; \boldsymbol{\theta}_k) \tag{5.15}$$

where $f()$ is the network function consisting of parameters $\boldsymbol{\theta}_k$.
In order to train the filter network, training data is used. Just as in the previous algorithms, the adaptive filter is trained at each step k. This is done using stochastic gradient descent. For training, a set of P previous input-output pairs are used. In other words, the filter coefficients or network parameters are updated as:

$$\boldsymbol{\theta}_{k+1} = \text{SGD}(\mu, \boldsymbol{\theta}_k, [\mathbf{y}_k, s_k], \ldots, [\mathbf{y}_{k-P}, s_{k-P}]) \tag{5.16}$$

where μ is the learning rate of the stochastic gradient descent algorithm.

5.2.4. Dataset

Most of the algorithms introduced in the previous sections require the value of one or more parameters to be specified. In order to find the appropriate parameters, a training dataset is needed. For that, the Watermark simulator with the same channel recordings as given in Section 2.5 is used in combination with the FRSS physical layer. The simulation is run for an FRSS rate 3 message with 284 bits payload and SNR values in the range $[-10, 10]$. For each simulation, the inputs to the equalizer $\mathbf{y}_k$ and the true symbols s_k were stored. From the resulting dataset, 90% was used for training and 10% for validation.
The algorithm parameters are found using algorithm unrolling. First, the algorithms are implemented in Pytorch [73]. Then, an initial algorithm parameter value is set, such as the step size in LMS. Next, the equalizer is run for K symbols and the MSE between the estimated and true symbols is calculated. Subsequently, the gradient with respect to the algorithm parameters is taken, such that an optimization step can be made. This method is shown schematically in Figure 5.1 for three iterations.
Another possible method for parameter optimization is a parameter sweep or grid search. This is feasible in some cases. However, when many parameters need to be optimized for, such as in VS-LMS by Harris, this becomes a highly inefficient method. All possible combinations of parameters should be tested and this takes an incredible amount of time.

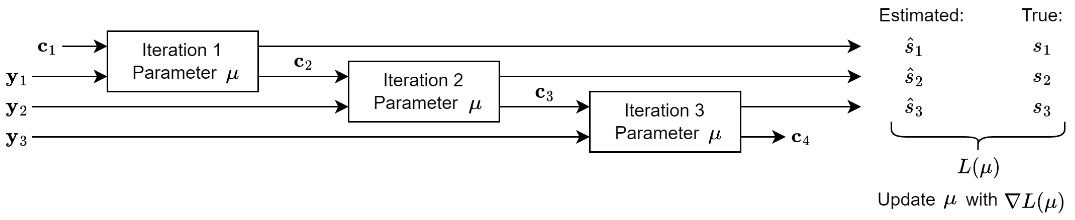


Figure 5.1: Algorithm unrolling structure for three iterations.

5.2.5. Performance

The performance metric that is used is the average PDR for various SNR values. The PDR describes if the decoded message contains bit errors or not. This is influenced by the equalizer, i.e. whether or not the equalizer was able to find the correct symbols. Therefore, it is an appropriate measure of the performance of the equalizer.

5.3. Results

First, results concerning the LMS algorithm are shown. Figure 5.2 shows the average PDR for SNR values in the range $[-10, 10]$. The FRSS LMS algorithm with $\mu = 0.0033$ is compared with an LMS algorithm with parameter $\mu = 0.0039$, obtained using algorithm unrolling. In fact, all of the parameters used in this section are obtained using algorithm unrolling. It should be noted that the step size parameter in FRSS has previously been optimized. Therefore, it is not surprising that no gain in terms of the PDR is visible. However, the fact that the algorithm unrolling method finds a similar value, proves that the method works. The average PDR of the detection preamble is also plotted. This serves as an upper limit on the PDR for FRSS using different update algorithms, as no gain in PDR can be achieved if the signal is not detected.

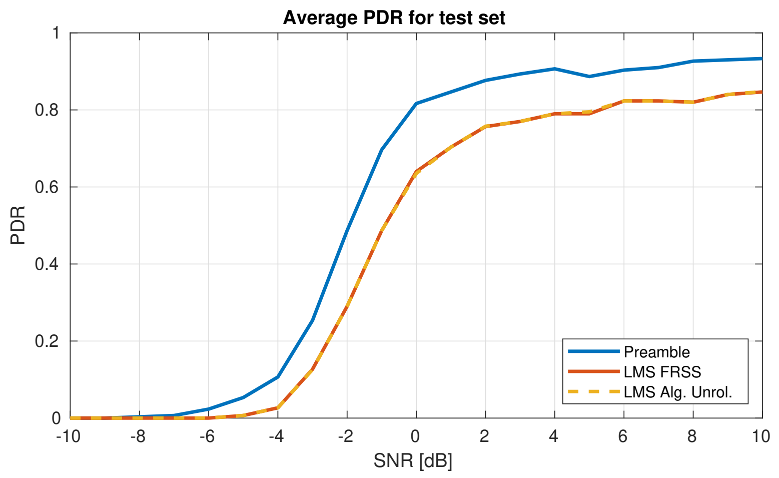


Figure 5.2: Average PDR of detection preamble, LMS filter with FRSS step size parameter and LMS filter with $\mu = 0.0033$ found using algorithm unrolling.

A typical convergence plot of the equalizer is shown in Figure 5.3. The vertical bar indicates the distinction between the initial training symbols and the rest of the symbols. It can be seen that the MSE changes significantly due to the channel variations.

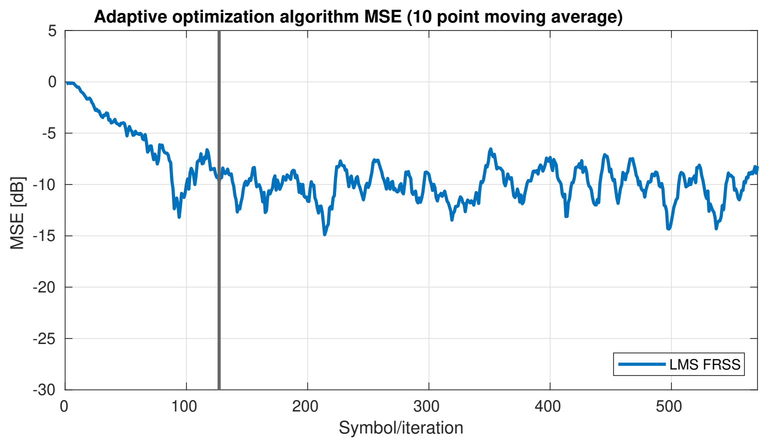


Figure 5.3: MSE of equalizer using LMS with FRSS step size parameter.

As could be seen from Figure 5.3, the LMS algorithm is relatively slow to converge. It uses the same step size for all the symbols. Therefore, having two separate step size parameters, a larger value for the initial training symbols and another value for the other symbols, might benefit the overall performance of the equalizer. The result is shown in Figure 5.4. The step size parameters are $\boldsymbol{\mu} = [0.0048\,0.0025]$.
Two other LMS-based alternatives are tested as well. The first is the LMS with an alternative gradient estimate given in Equation 5.4, with $P = 7$. The second is the LMS algorithm with momentum parameter 0.05, as described by Equation 5.8.
A typical convergence plot for these methods is shown in Figure 5.5. The difference in convergence between the methods is minimal.
The PDR of the algorithms is close to the PDR of the standard LMS method. This is not entirely unsurprising. The step size parameters for the first method are both close to the standard LMS parameter. This also holds for the other two methods. Apparently, the standard LMS form is the better than the alternative forms.

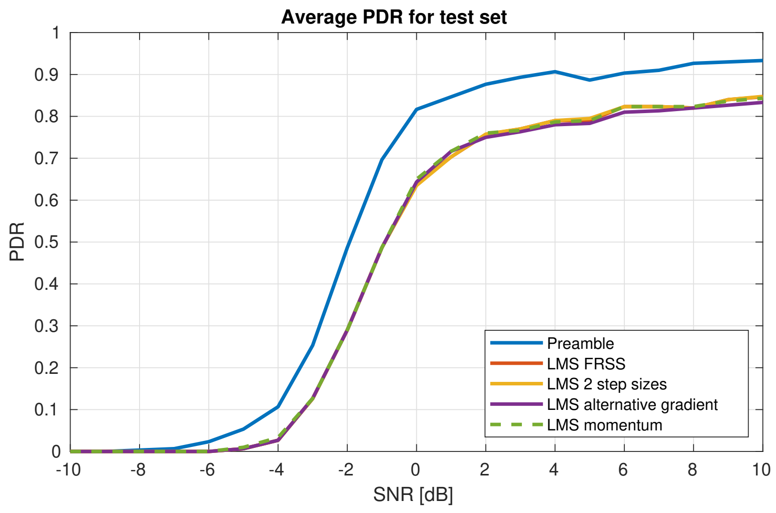


Figure 5.4: Average PDR of detection preamble, LMS filter with FRSS step size parameter, LMS filter with two step sizes, LMS filter with alternative gradient and LMS with momentum.

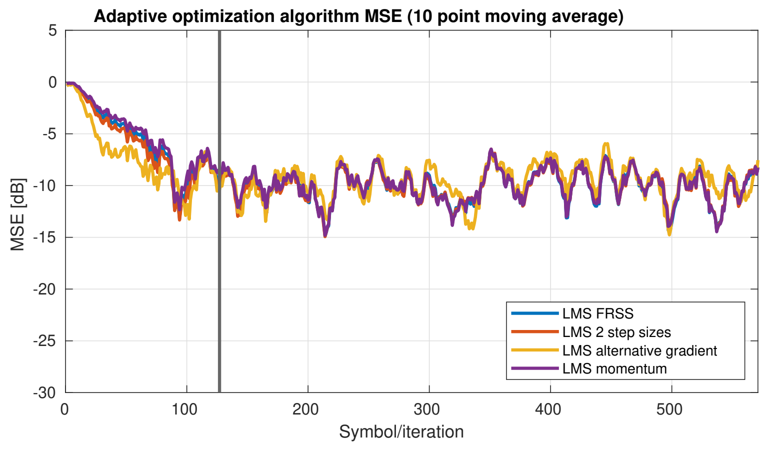


Figure 5.5: MSE of equalizer using LMS filter with FRSS step size parameter, LMS filter with two step sizes, LMS filter with alternative gradient and LMS with momentum.

Figure 5.6 shows the PDR of the NLMS algorithm with parameter 0.31. Furthermore, both the variable step size LMS algorithms by Harris and Mikhael are tested as well. The same conclusions can be drawn as for the previous test. The standard LMS performs similar to or better than the other algorithms.
In Figure 5.7 the convergence plot is shown. Again, the difference in convergence between the methods is minimal.
The fact that the standard LMS algorithm performs similarly to the NLMS algorithm could be due to the subband normalization of the input data for LMS. As explained in Section 5.2.2, the input data for LMS is normalized per subband, which is not entirely the same as normalizing for the complete input vector. However, the subband normalization makes the LMS algorithm insusceptible to input power changes. This has the same effect as the normalization in NLMS and therefore the performances are similar. The same argument holds for the VS-LMS by Mikhael.
For the VS-LMS by Harris, it was noticed that the learned parameters were all close to the standard LMS step size again.

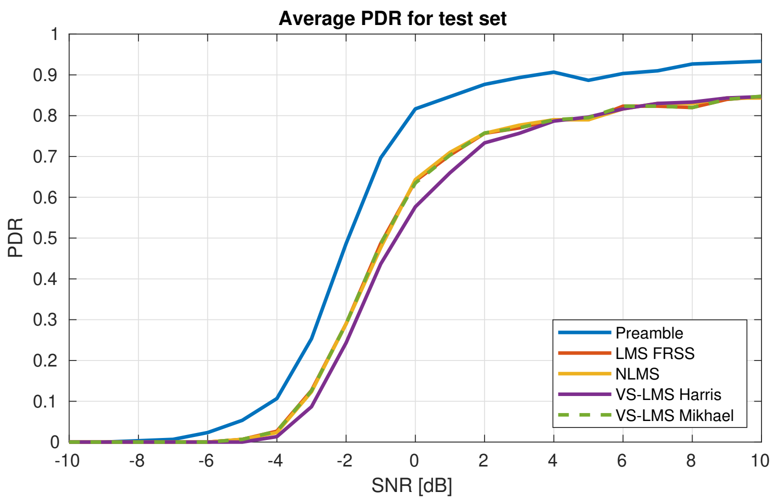


Figure 5.6: Average PDR of detection preamble, LMS filter with FRSS step size parameter, NLMS, VS-LMS by Harris and VS-LMS by Mikhael.

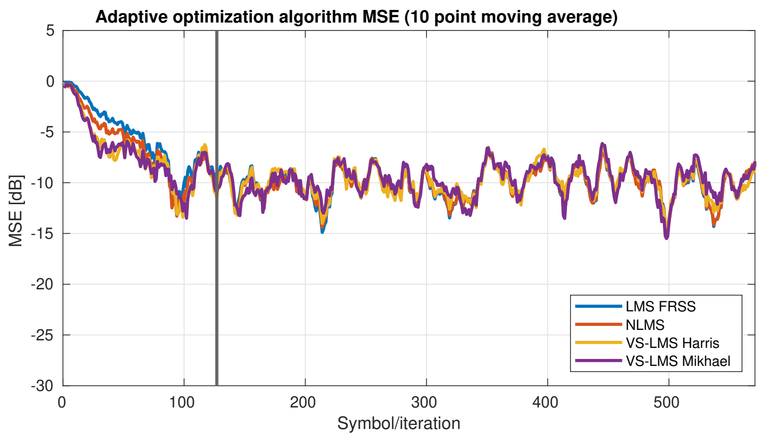


Figure 5.7: MSE of equalizer using LMS filter with FRSS step size parameter, NLMS, VS-LMS by Harris and VS-LMS by Mikhael.

The performance of the last two linear filters is shown in Figure 5.8 and 5.9. The AdaGrad algorithm was implemented with $\mu = 6.14 \times 10^{-4}$ and $\epsilon = 10^{-7}$. Furthermore, a momentum parameter of $\gamma = 0.75$ was also used. The RLS algorithm was implemented with a forgetting factor of $\lambda = 0.9965$.
Again, gain in terms of PDR was noticed. The average PDR of the RLS algorithm even lacks behind compared to the PDR of the LMS algorithm. It appears that the standard LMS algorithm finds a solution that is optimal to some extent, as it cannot be beaten by the other methods.
However, it should be noted that the AdaGrad and RLS algorithms converge faster to a roughly fixed error level than the LMS algorithm. This can be exploited by sending fewer initial training symbols to increase the data rate.

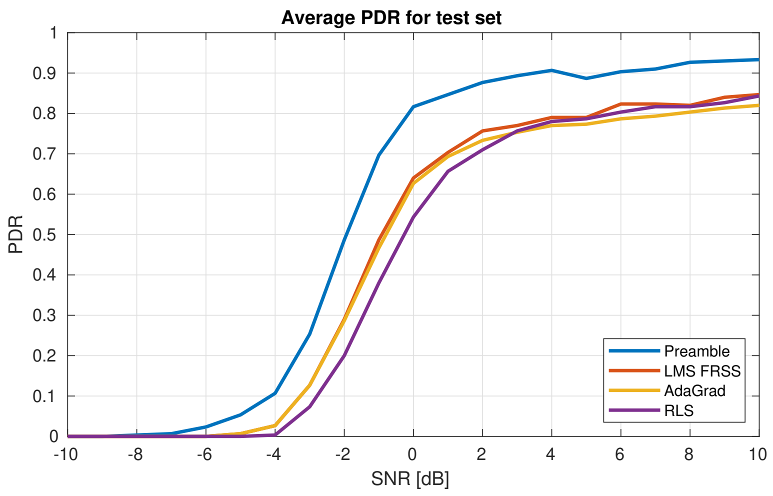


Figure 5.8: Average PDR of detection preamble, LMS filter with FRSS step size parameter, AdaGrad and RLS.

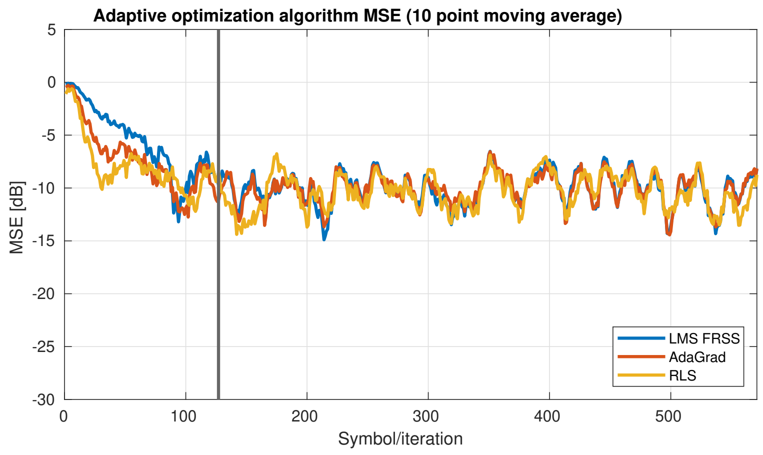


Figure 5.9: MSE of equalizer using LMS filter with FRSS step size parameter, AdaGrad and RLS.

To verify if the performance of the linear filter algorithms can be improved, the non-linear filter algorithms are tested next. The results are presented in Figures 5.10 and 5.11. The KLMS algorithm was implemented with $\mu = 0.005$ and $\sigma = 4.8$. The neural network had layer sizes [182, 32, 4]. The input vector $\mathbf{y}_k$ was split up in real and imaginary parts. The DNN was trained using stochastic gradient descent with momentum. The learning rate $\mu = 0.26$ and the momentum parameter $\beta = 0.6$. Furthermore, the $P = 25$ previous input-output pairs were used as a training set per symbol/iteration. Also, per iteration, 5 epochs were executed, i.e. Equation 5.16 was executed 5 times per iteration. However, it should be noted that the algorithm unrolling learning behaviour was quite unstable. In other words, the optimal neural network learning parameters could not be determined precisely.
The performance in terms of PDR is quite poor, as can be seen from Figure 5.10. The non-linearity in the KLMS algorithm did not benefit the overall performance. The behaviour of the DNN equalizer is interesting. Figure 5.11 shows that the DNN method is able to achieve much lower error levels than the previously discussed algorithms. However, the MSE has a large variation. For some symbols, the error is remarkably small. On the other hand, for other symbols, the error is so large that the symbol is classified incorrectly. This behaviour decreases the average PDR. The DNN lacks the stability of the LMS algorithm.

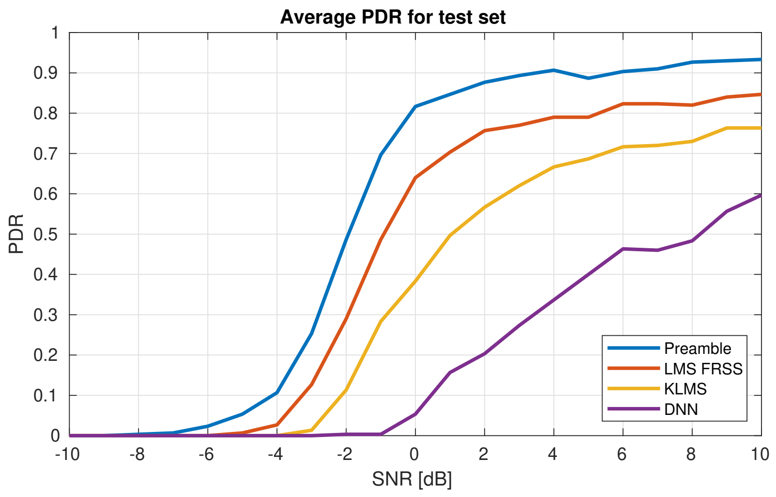


Figure 5.10: Average PDR of detection preamble, LMS filter with FRSS step size parameter, KLMS and DNN.

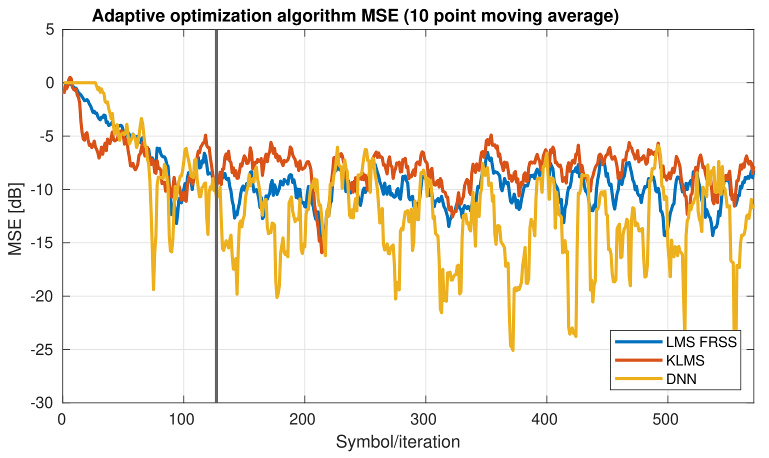


Figure 5.11: MSE of equalizer using LMS filter with FRSS step size parameter, KLMS and DNN.

The previously presented results were all based on FRSS rate 3 packets. To show that these results mostly generalize for the other FRSS rates, Figures 5.12, 5.13 and 5.14 show the average PDR for the other FRSS rates. The VS-LMS algorithm by Mikhael and the AdaGrad algorithm were implemented, since those methods performed well in the previous tests. Overall the results are similar to those of FRSS rate 3 packets.
A difference is that for FRSS rate 1, the AdaGrad method outperformed the LMS algorithm. Furthermore, for FRSS rate 2, the VS-LMS algorithm performed slightly worse than the LMS algorithm.

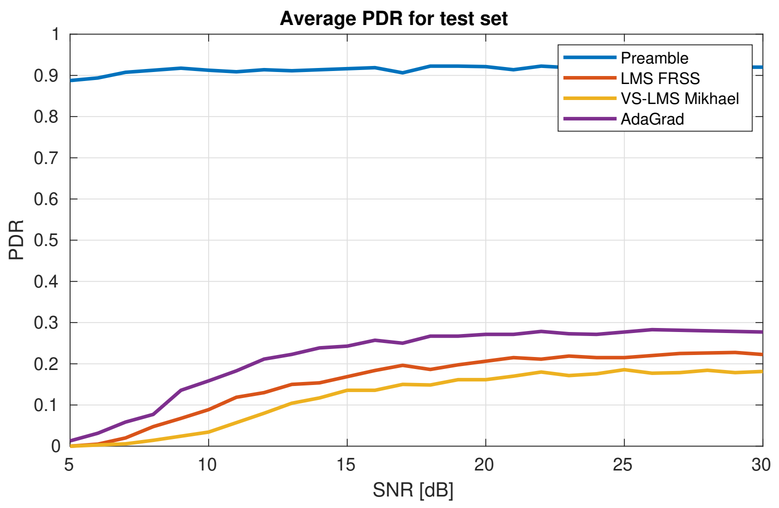


Figure 5.12: Average PDR of detection preamble, LMS filter with FRSS step size parameter, VS-LMS by Mikhael and AdaGrad for FRSS rate 1 packets.

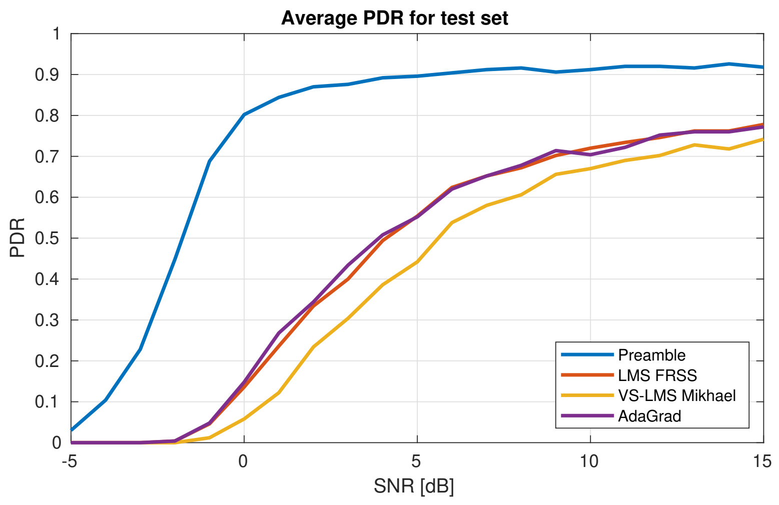


Figure 5.13: Average PDR of detection preamble, LMS filter with FRSS step size parameter, VS-LMS by Mikhael and AdaGrad for FRSS rate 2 packets.

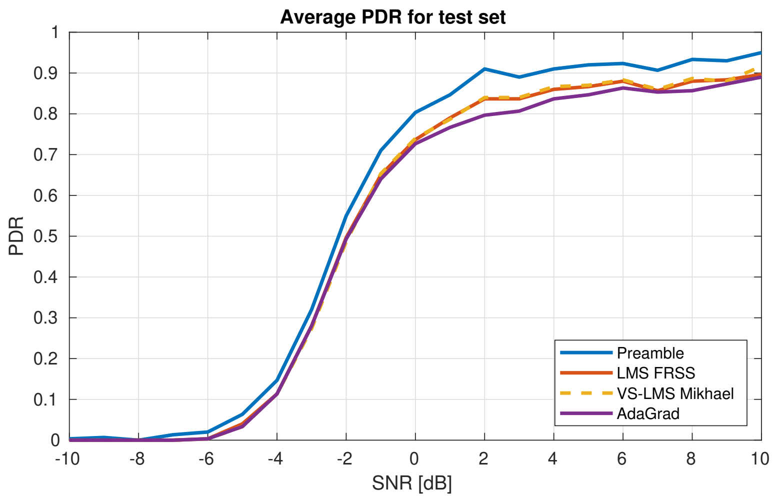


Figure 5.14: Average PDR of detection preamble, LMS filter with FRSS step size parameter, VS-LMS by Mikhael and AdaGrad for FRSS rate 4 packets.

6 Conclusion and future work

This thesis has studied link adaptation and equalization for underwater acoustic communication using machine learning to answer several research questions. Each research question will be answered next.

6.1. Link adaptation

Which channel or communication parameters provide valuable information for link adaptation?

The FRSS receiver calculates three channel descriptors: input SNR, delay spread and output SNR. Machine learning classifiers were trained to make decisions based on these descriptors. It was found that the output SNR provided the highest quality information for FRSS rate selection. Furthermore, slightly higher performance was obtained by training a neural network on the channel impulse response (CIR) and equalizer error sequences.

Which classification method can switch effectively between different transmission formats?

Benchmark methods, as well as various machine learning methods, were used to select FRSS rates. It was found that the machine learning classifiers performed better on average than the benchmark methods. The neural network provided the highest bit rate. However it also has the highest computational complexity. A good trade-off between performance and computational complexity was achieved by a decision tree classifier.

What is the effect of feedback delay on the performance of link adaptation methods?

Due to the feedback delay, the channel descriptors are more outdated when arriving at the transmitter. It was found that larger feedback delays negatively influence the performance of the classifiers. However, for modest delay values, the performance only decreased slightly. Furthermore, it was found that using descriptors from more than one previous transmission did not increase the average bit rate.

6.2. Equalization

Can an alternative equalizer algorithm increase the performance of the FRSS equalizer?

Besides the LMS equalizer implemented in the FRSS phsyical layer, multiple alternative linear equalizer algorithms have been implemented. These include variations to the LMS algorithm, such as NLMS and VS-LMS algorithms. Furthermore, more sophisticated methods like RLS and AdaGrad have been tested as well. The optimal algorithm parameters have been found using algorithm unrolling. The methods have not shown improvements in terms of PDR.

Can the use of a non-linear filter increase the performance of the equalizer?

Two non-linear filter algorithms have been implemented and tested; the KLMS algorithm and a DNN. It has been shown that the KLMS algorithm with a Gaussian kernel did not increase the performance of the equalizer. The DNN showed potential, since in some cases it was able to achieve lower error levels than the

LMS algorithm. However, overall the DNN made more bit erros than the LMS algorithm, resulting in a lower performance.

6.3. Recommendations

Recommendations for future work to improve and build upon the methods presented in this thesis will be given next.

In order to increase the average effective bit rate, link adaptation has been implemented. However, only four different FRSS formats have been used. The main difference between the four FRSS rates is the number of subbands used. Further research can take other communication parameters into account as well. For example, adaptive constellation schemes, transmit power per subband and number of payload bits. A first step could be to use an extension to FRSS, called multi-stream FRSS (MSFRSS) [8]. MSFRSS allows for the use of multiple streams of frequency bands, each with its own modulation scheme. However, it should be noted that it becomes increasingly inefficient to create a training dataset with optimal transmission formats when there are many available formats. At least, if the same methodology to obtain a dataset is used as in this thesis. The problem could be solved more efficiently by using RL. In RL, an agent attempts to find the optimal solution based on channel information by means of trial and error. A few works have been published discussing this topic [37–39].

Regarding the equalization optimization, the tested alternative equalization algorithms did not improve the performance in terms of PDR. However, from the typical convergence plots, it was noticed that a selection of algorithms seemed to converge faster than the LMS algorithm. This can possibly be exploited by transmitting fewer initial training symbols, such that the data rate is increased. Future research is needed to analyze this and to quantify the difference in the number of required initial training symbols for the various algorithms.

The DNN equalizer algorithm showed that it was able to outperform the LMS equalizer in terms of MSE for some symbols. Future work could focus on analyzing the DNN equalizer more. The number of possible configurations and learning algorithms for a neural network equalizer are incredibly large.
For example, a recurrent neural network (RNN) could be trained to take previous estimated symbols into consideration in a more efficient way. This was already briefly explored in a published thesis [87]. It was found that an RNN can indeed outperform a standard DNN.
Another possibility to increase performance would be to use a pretrained network. In this thesis, the network parameters were initiated at random. However, it is also possible to set them to known values. Future research could explore if general network parameters can be found to serve as a pretrained network, such that the network converges faster or reaches a lower MSE.

Both the methods used for link adaptation and equalization optimization were trained and tested on datasets generated using Watermark. Although a large set of underwater channels was used, they were extracted from the same experiment. Future research should include more time-varying impulse responses from a larger amount of experiments and geographical locations. Doing this makes the performance of the algorithms less susceptible to varying environmental scenarios.

Bibliography

[1] William C Jakes and Donald C Cox. *Microwave mobile communications*. Wiley-IEEE press, 1994.

[2] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[3] Theodore S Rappaport et al. *Wireless communications: principles and practice*, volume 2. prentice hall PTR New Jersey, 1996.

[4] Milica Stojanovic and James Preisig. Underwater acoustic communication channels: Propagation models and statistical characterization. *IEEE communications magazine*, 47(1):84–89, 2009.

[5] Mohsin Murad, Adil A Sheikh, Muhammad Asif Manzoor, Emad Felemban, and Saad Qaisar. A survey on current underwater acoustic sensor network applications. *International Journal of Computer Theory and Engineering*, 7(1):51, 2015.

[6] Artur Zolich, David Palma, Kimmo Kansanen, Kay Fjørtoft, João Sousa, Karl H Johansson, Yuming Jiang, Hefeng Dong, and Tor A Johansen. Survey on communication and networks for autonomous marine systems. *Journal of Intelligent & Robotic Systems*, 95(3):789–813, 2019.

[7] Henry Dol. *EDA-SALSA: Towards smart adaptive underwater acoustic networking*. IEEE, 2019.

[8] KCH Blom, HS Dol, and MK Prior. *Development of a wideband underwater acoustic modulation providing quality-of-service support*. IEEE, 2017.

[9] Timothy O'shea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

[10] Tim Schenk. *RF imperfections in high-rate wireless systems: impact and digital compensation*. Springer Science & Business Media, 2008.

[11] Tugba Erpek, Timothy J O'Shea, Yalin E Sagduyu, Yi Shi, and T Charles Clancy. Deep learning for wireless communications. In *Development and Analysis of Deep Learning Architectures*, pages 223–266. Springer, 2020.

[12] Andrea Goldsmith. Joint source/channel coding for wireless channels. In *1995 IEEE 45th Vehicular Technology Conference. Countdown to the Wireless Twenty-First Century*, volume 2, pages 614–618. IEEE, 1995.

[13] Ephraim Zehavi. 8-PSK trellis codes for a Rayleigh channel. *IEEE Transactions on Communications*, 40(5): 873–884, 1992.

[14] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on CPUs. 2011.

[15] Timothy J O'Shea, Latha Pemula, Dhruv Batra, and T Charles Clancy. Radio transformer networks: Attention models for learning to synchronize in wireless systems. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 662–666. IEEE, 2016.

[16] Sebastian Dörner, Sebastian Cammerer, Jakob Hoydis, and Stephan Ten Brink. Deep learning based communication over the air. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):132–143, 2017.

[17] Alexander Felix, Sebastian Cammerer, Sebastian Dörner, Jakob Hoydis, and Stephan Ten Brink. OFDM-autoencoder for end-to-end learning of communications systems. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2018.

[18] Thien Van Luong, Youngwook Ko, Michail Matthaiou, Ngo Anh Vien, Minh-Tuan Le, and Vu-Duc Ngo. Deep learning-aided multicarrier systems. *IEEE Transactions on Wireless Communications*, 20(3):2109–2119, 2020.

[19] Timothy J O'Shea, Tugba Erpek, and T Charles Clancy. Deep learning based MIMO communications. *arXiv preprint arXiv:1707.07980*, 2017.

[20] Amr S Hares, Mohamed A Abdallah, Mohamed A Abohassan, and Doaa A Altantawy. Recurrent Neural Networks for Pilot-aided Wireless Communications. In *2021 38th National Radio Science Conference (NRSC)*, volume 1, pages 167–176. IEEE, 2021.

[21] Hao Ye, Geoffrey Ye Li, Biing-Hwang Fred Juang, and Kathiravetpillai Sivanesan. Channel agnostic end-to-end learning based communication systems with conditional GAN. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–5. IEEE, 2018.

[22] Timothy J O'Shea, Tamoghna Roy, and Nathan West. Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 681–686. IEEE, 2019.

[23] Vishnu Raj and Sheetal Kalyani. Backpropagating through the air: Deep learning at physical layer without channel models. *IEEE Communications Letters*, 22(11):2278–2281, 2018.

[24] Fayçal Ait Aoudia and Jakob Hoydis. Model-free training of end-to-end communication systems. *IEEE Journal on Selected Areas in Communications*, 37(11):2503–2516, 2019.

[25] Fayçal Ait Aoudia and Jakob Hoydis. End-to-end learning of communications systems without a channel model. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 298–303. IEEE, 2018.

[26] Henry Dol, Koen Blom, Paul van Walree, Roald Otnes, Håvard Austad, Till Wiegand, and Dimitri Sotnik. Adaptivity at the Physical Layer. In *Cognitive Underwater Acoustic Networking Techniques*, pages 13–40. Springer, 2020.

[27] Parastoo Qarabaqi and Milica Stojanovic. Adaptive power control for underwater acoustic communications. In *OCEANS 2011 IEEE-Spain*, pages 1–7. IEEE, 2011.

[28] Josep Miquel Jornet, Milica Stojanovic, and Michele Zorzi. On joint frequency and power allocation in a cross-layer protocol for underwater acoustic networks. *IEEE Journal of Oceanic engineering*, 35(4): 936–947, 2010.

[29] Yishan Su, Yibo Zhu, Haining Mo, Jun-Hong Cui, and Zhigang Jin. A joint power control and rate adaptation MAC protocol for underwater sensor networks. *Ad Hoc Networks*, 26:36–49, 2015.

[30] Joachim Hagenauer. Rate-compatible punctured convolutional codes (RCPC codes) and their applications. *IEEE transactions on communications*, 36(4):389–400, 1988.

[31] A Benson, J Proakis, and M Stojanovic. Towards robust adaptive acoustic communications. In *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No. 00CH37158)*, volume 2, pages 1243–1249. IEEE, 2000.

[32] Sanjay Mani, Tolga M Duman, and Paul Hursky. Adaptive coding-modulation for shallow-water UWA communications. *Journal of the Acoustical Society of America*, 123(5):3749, 2008.

[33] Andreja Radosevic, Rameez Ahmed, Tolga M Duman, John G Proakis, and Milica Stojanovic. Adaptive OFDM modulation for underwater acoustic communications: Design considerations and experimental results. *IEEE Journal of Oceanic Engineering*, 39(2):357–370, 2013.

[34] Lei Wan, Hao Zhou, Xiaoka Xu, Yi Huang, Shengli Zhou, Zhijie Shi, and Jun-Hong Cui. Adaptive modulation and coding for underwater acoustic OFDM. *IEEE Journal of Oceanic Engineering*, 40(2):327–336, 2014.

[35] Konstantinos Pelekanakis, Luca Cazzanti, Giovanni Zappa, and João Alves. Decision tree-based adaptive modulation for underwater acoustic communications. In *2016 IEEE third underwater communications and networking conference (UComms)*, pages 1–5. IEEE, 2016.

[36] Lihuan Huang, Qunfei Zhang, Weijie Tan, Yue Wang, Lifan Zhang, Chengbing He, and Zhi Tian. Adaptive modulation and coding in underwater acoustic communications: a machine learning perspective. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):1–25, 2020.

[37] Chaofeng Wang, Zhaohui Wang, Wensheng Sun, and Daniel R Fuhrmann. Reinforcement learning-based adaptive transmission in time-varying underwater acoustic channels. *IEEE access*, 6:2541–2558, 2017.

[38] Jiamin Lin, Wei Su, Liang Xiao, and Xialin Jiang. Adaptive modulation switching strategy based on Q-learning for underwater acoustic communication channel. In *Proceedings of the Thirteenth ACM International Conference on Underwater Networks & Systems*, pages 1–5, 2018.

[39] Wei Su, Jiamin Lin, Keyu Chen, Liang Xiao, and Cheng En. Reinforcement learning-based adaptive modulation and coding for efficient underwater communications. *IEEE access*, 7:67539–67550, 2019.

[40] Evan Lucas and Zhaohui Wang. Supervised learning for performance prediction in underwater acoustic communications. In *Global Oceans 2020: Singapore–US Gulf Coast*, pages 1–6. IEEE, 2020.

[41] Evan Lucas and Zhaohui Wang. Performance prediction of underwater acoustic communications based on channel impulse responses. *Applied Sciences*, 12(3):1086, 2022.

[42] Milica Stojanovic, Josko A Catipovic, and John G Proakis. Phase-coherent digital communications for underwater acoustic channels. *IEEE journal of oceanic engineering*, 19(1):100–111, 1994.

[43] Paul A Van Walree and Geert Leus. Robust underwater telemetry with adaptive turbo multiband equalization. *IEEE Journal of Oceanic Engineering*, 34(4):645–655, 2009.

[44] Paul Van Walree, Erland Sangfelt, and Geert Leus. *Multicarrier spread spectrum for covert acoustic communications*. IEEE, 2008.

[45] Dariusz Bismor, Krzysztof Czyz, and Zbigniew Ogonowski. Review and comparison of variable step-size lms algorithms. *International Journal of Acoustics and Vibration*, 21(1):24–39, 2016.

[46] Sheng Chen, Bernard Mulgrew, and Steve McLaughlin. Adaptive bayesian equalizer with decision feedback. *IEEE Transactions on Signal Processing*, 41(9):2918–2927, 1993.

[47] S Siu, GJ Gibson, and CFN Cowan. Decision feedback equalization using neural network structures. In *1989 First IEE International Conference on Artificial Neural Networks,(Conf. Publ. No. 313)*, pages 125–128. IET, 1989.

[48] Zhe Chen and AC de C Lima. A new neural equalizer for decision-feedback equalization. In *Proceedings of the 2004 14th IEEE Signal Processing Society Workshop Machine Learning for Signal Processing, 2004.*, pages 675–684. IEEE, 2004.

[49] TG Leighton. The Acoustic Bubble.| Academic. *Press, London*, pages 234–243, 1994.

[50] William C Knight, Roger G Pridham, and Steven M Kay. Digital signal processing for sonar. *Proceedings of the IEEE*, 69(11):1451–1506, 1981.

[51] Robert J Urick. Principles of underwater sound 3rd edition. *Peninsula Publising Los Atlos, California*, 22: 23–24, 1983.

[52] Milica Stojanovic. On the relationship between capacity and distance in an underwater acoustic communication channel. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(4):34–43, 2007.

[53] Y Lysanov and LM Brekhovskikh. *Fundamentals of ocean acoustics*, volume 8. Springer, 1982.

[54] Michael A Ainslie. *Principles of sonar performance modelling*, volume 707. Springer, 2010.

[55] John G.. Proakis and Masoud Salehi. *Digital communications*. McGraw-Hill., 2008.

[56] Ye Jiang and Antonia Papandreou-Suppappola. Discrete time-scale characterization of wideband time-varying systems. *IEEE Transactions on Signal Processing*, 54(4):1364–1375, 2006.

[57] Urbashi Mitra and Geert Leus. Equalizers for multi-scale/multi-lag wireless channels. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5. IEEE, 2010.

[58] M Zheng. Experimental study on statistical characteristics of pulse transmission in the shallow water. *PROCEEDINGS-INSTITUTE OF ACOUSTICS*, 15:142–142, 1993.

[59] R Galvin and RFW Coates. Analysis of the performance of an underwater acoustic communications system and comparison with a stochastic model. In *Proceedings of OCEANS'94*, volume 3, pages III–478. IEEE, 1994.

[60] A Essebbar, G Loubet, and F Vial. Underwater acoustic channel simulations for communication. In *Proceedings of OCEANS'94*, volume 3, pages III–495. IEEE, 1994.

[61] Edward A Lee and David G Messerschmitt. *Digital communication*. Springer Science & Business Media, 2012.

[62] Luca Rugini, Paolo Banelli, and Geert Leus. Low-complexity banded equalizers for ofdm systems in doppler spread channels. *EURASIP Journal on Advances in Signal Processing*, 2006:1–13, 2006.

[63] Roald Otnes, Paul A van Walree, Helge Buen, and Heechun Song. Underwater acoustic network simulation with lookup tables from physical-layer replay. *IEEE Journal of Oceanic Engineering*, 40(4):822–840, 2015.

[64] Henry Dol, Koen Blom, and Ernest van der Spek. Experiences with JANUS and efforts towards a common heavy-duty underwater communication stack. In *2016 IEEE Third Underwater Communications and Networking Conference (UComms)*, pages 1–5. IEEE, 2016.

[65] Paul van Walree, Helge Buen, and Roald Otnes. A performance comparison between DSSS, M-FSK, and frequency-division multiplexing in underwater acoustic channels. In *2014 Underwater Communications and Networking (UComms)*, pages 1–5, 2014. doi: 10.1109/UComms.2014.7017133.

[66] Michael B Porter. The bellhop manual and user's guide: Preliminary draft. *Heat, Light, and Sound Research, Inc., La Jolla, CA, USA, Tech. Rep*, 260, 2011.

[67] Paul A van Walree, François-Xavier Socheleau, Roald Otnes, and Trond Jenserud. The watermark benchmark for underwater acoustic modulation schemes. *IEEE Journal of Oceanic Engineering*, 42(4):1007–1018, 2017.

[68] Paul A Van Walree, Trond Jenserud, and Morten Smedsrud. A discrete-time channel simulator driven by measured scattering functions. *IEEE journal on selected areas in communications*, 26(9):1628–1637, 2008.

[69] Paul van Walree and Mathieu Colin. In-situ performance prediction of a coherent acoustic modem. In *2021 Fifth Underwater Communications and Networking Conference (UComms)*, pages 1–5. IEEE.

[70] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. Belmont, CA: Wadsworth. *International Group*, 432:151–166, 1984.

[71] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[72] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[73] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[74] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

[75] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

[76] Paul van Walree. Channel sounding for acoustic communications: techniques and shallow-water examples. 2011.

[77] Paul van Walree. On the definition of receiver output SNR and the probability of bit error. In *2013 MTS/IEEE OCEANS-Bergen*, pages 1–9. IEEE, 2013.

[78] John M Cioffi, Glen P Dudevoir, M Vedat Eyuboglu, and G David Forney. MMSE decision-feedback equalizers and coding. I. Equalization results. *IEEE transactions on Communications*, 43(10):2582–2594, 1995.

[79] Michael Goetz and Ivor Nissen. GUWMANET—Multicast routing in underwater acoustic networks. In *2012 military communications and information systems conference (MCC)*, pages 1–8. IEEE, 2012.

[80] KCH Blom, HS Dol, F Berning, and PA van Walree. *Development of a Physical Layer for Adaptive Underwater Acoustic Communications*. Under review, 2022.

[81] Koen Blom, Henry Dol, Till Wiegand, and Frank Berning. Definition and description of smart adaptive physical-layer methods. Internal document, 2022.

[82] Fredrik Lindqvist. Link adaptation based on CRC feedback. Internal document, 2022.

[83] R Harris, D Chabries, and F Bishop. A variable step (VS) adaptive filter algorithm. *IEEE transactions on acoustics, speech, and signal processing*, 34(2):309–316, 1986.

[84] W Mikhael, F Wu, L Kazovsky, G Kang, and L Fransen. Adaptive filters with individual adaptation of parameters. *IEEE Transactions on circuits and systems*, 33(7):677–686, 1986.

[85] Vladimir N Vapnik. The nature of statistical learning theory, 1995.

[86] Puskal P Pokharel, Weifeng Liu, and Jose C Principe. Kernel LMS. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 3, pages III–1421. IEEE, 2007.

[87] Martin Allander. Channel equalization using machine learning for underwater acoustic communications, 2020.