2-19

Comprehensive Behavioural Model for an Electronic Amplifier used in an Optical Communications System

Patricio Fuentes Ugartemendia

Comprehensive Behavioural Model for an Electronic Amplifier used in an Optical Communications System

PROYECTO
presentado para optar
al Título de Máster en Ingeniería de
Telecomunicaciones por
Patricio Fuentes Ugartemendia
bajo la supervisión de
Pedro Crespo Bofill
John C. Cartledge
Wai-Yip Geoffrey Chan

Donostia-San Sebastián, Febrero 2019

tecnun
Universidad
de Navarra

Proyecto Fin de Máster

INGENIERIA DE TELECOMUNICACIONES

Comprehensive Behavioural Model for an Electronic Amplifier used in an Optical Communications Systems

Patricio Fuentes Ugartemendia

San Sebastián, febrero de 2019

Pº Manuel Lardizabal, 13. 20018 Donostia-San Sebastián, Gipuzkoa
Tel. 943 219 877 · Fax 943 311 442 · www.tecnun.es

Tecnun-Universidad de Navarra

&

Queen's University

Abstract

Faculty of Engineering & Applied Science

Communications & Signal Processing Department

Master of Applied Science in Telecommunications Engineering

by Patricio Fuentes Ugartemendia

Correctly modelling the behaviour of power amplifiers to include the nonlinear aspects of their response has been a matter of interest in the fields of electronic and wireless communications for a considerable amount of time. Throughout the last decades, numerous techniques that succeed in accurately representing these nonlinearities and that provide ways to reduce their impact on system performance have been derived. An essential application of these techniques is the construction of amplifier models that can be used to obtain a deeper understanding of the phenomena that make these devices more or less nonlinear. These models have been used in numerous scenarios to comprehend and devise ways to mitigate the impact of nonlinearity in the response of many amplifiers. However, ultra-broadband RF electronic drive amplifiers used in optical communications environments represent a small niche in which extensive derivation of these amplifier models has not yet been accomplished. In this thesis, a device-specific behavioural model for one such electronic drive amplifier will be constructed based on existing modelling techniques and extensive lab measurements. In addition, changes in the performance of a communications system when these behavioural models are integrated within it will also be studied. Ultimately, this thesis will strive to determine the specific phenomena that are linked to the nonlinearity present in the response of an ultra-broadband RF electronic drive amplifier and how variation of these factors affects the behaviour of this device.

Acknowledgements

This thesis is a compulsory requirement to obtain the Master of Applied Science Degree in Telecommunications Engineering from the Tecnun-School of Engineering. The research work for this thesis has been carried out in the Lightwave Systems Research laboratory of the Faculty of Engineering and Applied Science at Queen's University in Kingston, Ontario, Canada, between September 2018 and February 2019.

I would like to express my gratitude to my thesis supervisor at my home university, Professor Pedro Crespo, for his endless encouragement and support prior and during the course of this work. I am also very grateful to my co-supervisors at Queen's University, Professors John C. Cartledge and Wai Yip Geoffrey Chan, for their willingness to introduce me into their respective research groups as a visiting research student, the time they devoted to helping me work through many technical and practical issues, and their guidance throughout my entire stay in Kingston.

My deepest thanks go to a fellow Master's Degree student, Peng Wang, for helping me learn the ropes of performing lab measurements and giving me a helping hand whenever it was needed. I would also like to thank Queen's University Professor Carlos Saavedra, who always made time to discuss both technical and personal matters with me even though he had no obligation to do so. In addition, many thanks go to Phd candidates Ahmed Ahmed and Aazar Kashi, for helping me get settled at Queen's and providing me with very useful programming insight.

I must also express my gratitude to the people that handled the logistics of completing this thesis abroad, Javier Ganuza and Estibaliz Oliva in Spain, and Debra Fraser and Darlene Homer in Canada. I am also grateful to Joaquin de No for his continued support and detailed advice regarding the contents of the thesis.

Finally, I would like to express how grateful I am towards my family and friends for their endless support during my studies.

Contents

List of Figures

List of Tables

Abbreviations

NARMAX	**N**onlinear **A**uto**R**egressive **M**oving **A**verage model with e**X**ogenous inputs
LMS	**L**east **M**ean **S**quares
RLS	**R**ecursive **L**east **S**quares
VLMS	**V**olterra **L**east **M**ean **S**quares
VRLS	**V**olterra **R**ecursive **L**east **S**quares
NRZ	**N**on **R**eturn to **Z**ero
mVpp	**m**ili**V**olts **p**eak-to-**p**eak
RHS	**R**ight **H**and **S**ide
MSE	**M**ean **S**quare **E**rror
SG	**S**ignal **G**enerator
BPG	**B**it **P**attern **G**enerator
DCA	**D**igital **C**ommunications **A**nalyzer
RSM	**R**emote **S**ampling **M**odule
PTM	**P**recision **T**imebase **M**odule
ISI	**I**nter**s**ymbol **I**nterference
COLA	**C**onstant **O**ver**L**ap and **A**dd
AWGN	**A**dditive **W**hite **G**aussian **N**oise
LDPC	**L**ow **D**ensity **P**arity **C**heck code
DVB-S	**D**igital **V**ideo **B**roadcasting by **S**atellite
PCM	**P**arity **C**heck **M**atrix
QAM	**Q**uadrature **A**mplitude **M**odulation
PAM	**P**ulse **A**mplitude **M**odulation
BPSK	**B**inary **P**hase **S**hift **K**eying
BER	**B**it **E**rror **R**ate

FEC	**F**orward **E**rror **C**orrection
SNR	**S**ignal to **N**oise **R**atio
LN	**L**i**N**ear
NL	**N**on-**L**inear
HD	**H**ard **D**ecision
SD	**S**oft **D**ecision
PRBS	**P**seudo**R**andom **B**inary **S**equence
DMC	**D**iscrete **M**emoryless **C**hannel
MI	**M**utual **I**nformation
ML	**M**aximum **L**ikelihood
BW Decoder	**B**it-**W**ise Decoder
GMI	**G**eneralized **M**utual **I**nformation
LLR	**L**og-**L**ikelihood **R**atio
DP	**D**ual **P**olarization

Symbols

M	memory of a Volterra model
P	order of a Volterra model
$h_p(m_1, \ldots, m_p)$	Volterra model kernels
$\tilde{h}_1$	Linear normalized Volterra kernel weight
$\tilde{h}_2$	Quadratic normalized Volterra kernel weight
$\tilde{h}_3$	Cubic normalized Volterra kernel weight
μ	LMS algorithm step size
μ_1	Linear VLMS algorithm step size
μ_2	Quadratic VLMS algorithm step size
μ_3	Cubic VLMS algorithm step size
λ	RLS algorithm forgetting factor
G_x, S_x	Amplifier input signal normalization parameters
G_d, S_d	Amplifier output signal normalization parameters
α	total number of Volterra kernels of a given model
$\mathbf{H}$	LDPC PCM
ρ	System spectral efficiency
$\frac{E_b}{N_0}$	SNR per bit
$P_{b,2}$	Uncoded bit error probability of 2-PAM system
BER_{post}	BER after FEC decoding
BER_{pre}	BER before FEC decoding
d_{nl}	System performance loss assessment parameter
d_{th}	System performance loss assessment parameter
d_{38}, d_r	Parameters to study system performance losses caused by input signal data rate
$c_{\tilde{h}_1}$	Parameter to study the evolution of

	the linear normalized Volterra kernel
$c_{\tilde{h}_3}$	Parameter to study the evolution of the cubic normalized Volterra kernel
d_{400}, d_v	Parameters to study system performance losses caused by input signal peak-to-peak voltage
$I(X;Y)$	Mutual Information between X and Y
C	Capacity of a given communication channel

Chapter 1

Introduction

1.1 Background

The immense demand for mobile communication technologies capable of meeting the constantly increasing data rate, capacity, flexibility, and reliability requirements of today's day and age is pushing system designers to efficiently optimize resource use more than ever before. System transmit power and bandwidth stand out among those resources whose use must be specifically controlled to fulfill the necessities of contemporary wireless communications devices.

Amplifiers are an essential component of almost any communications system, wireless technology being no exception, and they will directly affect the transmit power and bandwidth of a system. Because of the defining role amplifiers play in wireless technology, derivation of models that accurately represent the behaviour of these components has been a matter of significant importance for the past decades. As do all system components, real amplifiers deviate from their ideal behaviour depending on their operating conditions. In their specific case, this deviation manifests itself as the presence of nonlinearity in the response of the device. Due to the fact that this nonlinear aspect of an amplifiers behaviour has the potential to substantially impact the quality of the designed system, a prime goal of this modelling research has been to design methodologies that correctly emulate the nonlinearity displayed by these components. Through the years, numerous generic nonlinearity modelling methodologies have been successfully designed. Although degrees of accuracy vary slightly between these techniques, they provide a highly precise manner to represent the behaviour of devices that are notorious for presenting nonlinear responses under certain conditions.

Simultaneously within this same time frame, the main research interest in the closely related field of optical communications has been to accurately describe the effects of transmitting data through a fiber channel. This medium induces nonlinear effects on the signal, hence the study of nonlinearities has also been a matter of significant interest in this field, albeit with a more intense focus on channel-induced nonlinear effects instead of amplifier-induced phenomena. As a consequence of prioritizing the analysis of channel-induced nonlinear effects, the behaviour of ultra-broadband RF electronic drive amplifiers that are used in coherent optical communications systems has not been extensively studied. Application of the extensive methods of amplifier nonlinearity modelling that have been derived in the field of wireless technologies to characterize these electronic drive amplifiers represents a relatively novel approach to a decade-old research topic which may potentially provide some interesting insights.

1.2 Objective & Scope

The goal of this thesis is to characterize the behaviour of an ultra-broadband RF electronic drive amplifier to study the presence of nonlinearity in its response. This characterization will aim to derive a model that can be used to determine which factors are critical in morphing the response of this device to become more nonlinear. Overall, the aim of the research is to design a comprehensive model for a specific drive amplifier to obtain insight on what phenomena make the behaviour of this device nonlinear, as well as to determine the subsequent effects of this nonlinear content of the device response on the performance of a generic communications system.

1.3 Document Organization

The structure of the thesis is illustrated in Figure 1.1. The text can be divided into three parts, where Chapter 2 is made up of the theoretical concepts on which the work is based, Chapters 3-5 are compromised by the research and discussions that constitute the bulk of the thesis, and Chapter 6 explains potentially interesting future research avenues.

Chapter 2 describes the chosen nonlinearity modelling technique and the methodology selected to build the amplifier behavioural model. Chapter 3 covers the design and implementation of the software tools that are necessary to construct the model discussed in the previous chapter. The setup and execution of the lab experiments, along with a preliminary analysis of the results obtained from them, is also included in it. The focus of this chapter is to explain the layout of how the data is extracted and processed, the results are treated much more thoroughly later in the text.

In Chapter 4, the design of a synthetic communications system, as well as the integration of the model obtained in Chapter 3 within it, is explained. Theoretical concepts that are useful when assessing the impact on system performance as a result of including the nonlinear model within the implemented system are also discussed in this chapter, as is a set of important analytic tools.

In Chapter 5, the results obtained from the experiments explained in Chapter 3 are studied based on the analytic tools discussed in Chapters 4 and 5.

Chapter 6 provides an overview of possible improvements as well as suggestions for future work. Finally, Chapter 7 presents the conclusions and Chapter 8 details the budget of the thesis.

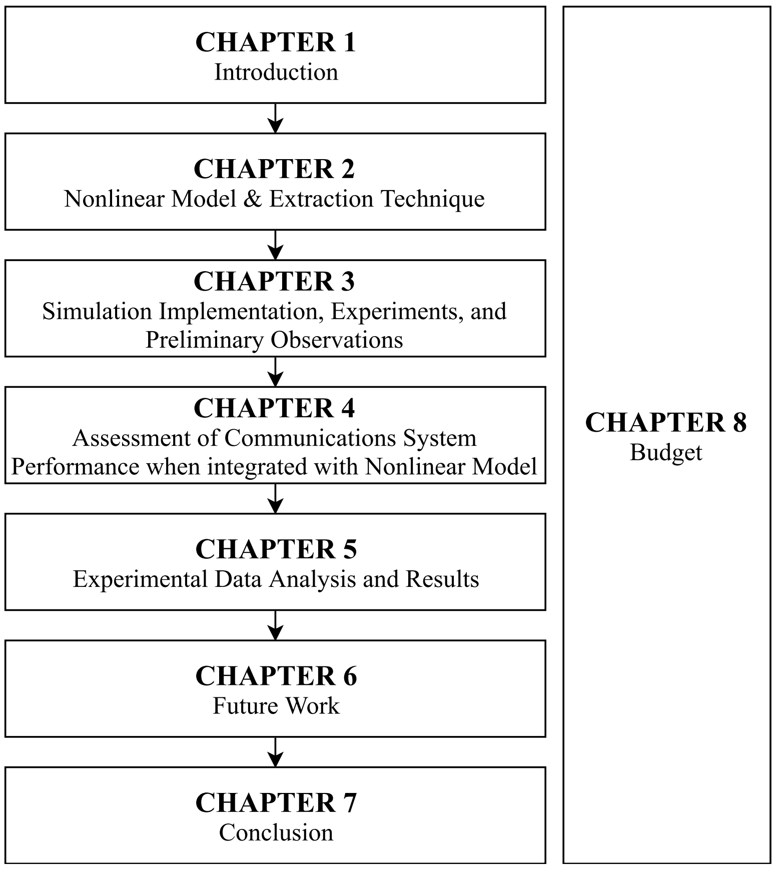

CHAPTER 1
Introduction
CHAPTER 2
Nonlinear Model & Extraction Technique
CHAPTER 3
Simulation Implementation, Experiments, and Preliminary Observations
CHAPTER 4
Assessment of Communications System Performance when integrated with Nonlinear Model
CHAPTER 5
Experimental Data Analysis and Results
CHAPTER 6
Future Work
CHAPTER 7
Conclusion
CHAPTER 8
Budget

FIGURE 1.1: Organization of the thesis.

Chapter 2

Nonlinearity & Amplifiers

Nonlinear systems, devices, and phenomena can be found across many fields of scientific interest. A nonlinear system is defined as any system that does not satisfy the superposition principle. Conversely, linear systems are those that satisfy this principle.

The superposition principle states that for a system to be linear, the net system response caused by two or more stimuli must equal the sum of the individual system responses to each of those stimuli. This can be easily understood by providing an example: given a system in which input X_1 produces output Y_1 and input X_2 results in output Y_2, if said system is linear, an input compromised of the sum of both these inputs, $X_1 + X_2$, will result in an output equal to the sum of the outputs obtained previously for each individual input, $Y_1 + Y_2$.

Linear systems can be perfectly described using methods based on the superposition principle. However, because nonlinear systems do not satisfy this principle, linear methods will not be suitable to represent them and different tools capable of accurately mimicking their nonlinear effects must be employed. Fortunately, a wide variety of techniques exists to perform nonlinear system identification[1] and obtain appropriate behavioural models. Due to the complexity and large amount of existing approaches to nonlinear system modelling, this chapter will begin by providing a simple overview of the most significant among these techniques. Once the reader has been briefly introduced to the discipline of nonlinear system identification, the focus shifts towards thoroughly explaining the specific technique selected to construct the amplifier model.

[1]System identification is defined as the process of identifying or measuring the mathematical model of a system from measurements of the system inputs and outputs.

2.1 Nonlinear System Identification

Electronic drive amplifiers are known to behave nonlinearly under specific operating conditions. As such, they can be classified as nonlinear systems and modelled using the wide variety of techniques available for this purpose. All of these methods provide different alternatives to perform nonlinear system identification, which is the method of obtaining the mathematical model of the system in question by means of system input and output measurements. It can be broken down into the following steps:

1. *Data Extraction:* The gathering of data sets is essential to construct nonlinear system models. It involves the recording of different inputs and outputs of the nonlinear system that is being identified. This information is later used to construct the model. The extraction of sufficiently exhaustive and appropriate data is paramount when building models for nonlinear systems.
2. *Model Construction:* The data obtained during the extraction process is applied to adaptive algorithms which are able to build nonlinear system models based on nonlinear modelling approaches. Selecting the modelling technique that is most suitable to each particular scenario will be critical in determining how good the model is.
3. *Model Validation:* The inputs recorded in the data extraction process are applied to the derived nonlinear models and the subsequent "modelled" outputs are compared to the previously gathered real outputs to determine the accuracy and validity of the constructed model.

2.1.1 Nonlinear Modelling Methods

As has previously been mentioned, many techniques can be used to build nonlinear models. The following list is a brief summary of the most relevant techniques used during the model construction procedure.

- *Volterra Series:* Volterra Series were the first method used to construct models for nonlinear systems. They provide a relatively simple way to mathematically describe the behaviour of a nonlinear system and are widely used to this day, remaining relevant in the field of nonlinear system identification. However, drawbacks related to the processing requirements of the algorithms that construct the Volterra series based models have recently been identified.

- *Block structured models (Hammerstein/Wiener Models):* Block structured models encompass a set of nonlinear modelling techniques that divide system behaviour into two separate elements that represent its linear and nonlinear aspects. These block structured models have been shown to have lower processing requirements than the Volterra series based algorithms, making the identification procedure more manageable.

- *Neural networks:* Artificial neural networks imitate the structure of neurons in the human brain by segmenting complex computation into large numbers of simple processing elements. A typical neural network consists of a number of simple processing units interconnected to form a complex system. These simple units are arranged into different layers so that the system input data is entered at the first layer and passed along through all the other layers before reaching the output layer. Generally, these networks require "training"; they operate on the difference between the real output data and the output of the network, changing the arrangement of layers and processing nodes in an attempt to diminish this difference until it is negligible, or in other words, until the real output and network output are essentially identical. The principal drawback of neural networks is that the models they produce are almost always impossible to describe mathematically and analyze.

- *NARMAX Models:* NARMAX stands for Nonlinear Autoregressive Moving Average model with eXogenous inputs. It has quickly become the most popular technique in the study of nonlinear systems, as it encompasses a large variety of other techniques, including the three previously described ones.

As has just been shown, there are numerous manners to build a nonlinear model for a specific system or device. Given that the focus of this thesis is to construct a model for an electronic drive amplifier, which is not an overly complex example of a nonlinear system, a technique that produces a relatively simple and accurate model which can be mathematically described and numerically assessed is desirable.

In view of these requirements, Volterra series are the most appropriate choice to construct the model for the amplifier studied in this thesis. Because they were the first nonlinear modelling technique, a large body of work and research exists related to their use in characterizing nonlinearities, which makes them an even better selection for our purposes. Having briefly introduced the principles of nonlinear system identification, the rest of this chapter will cover how Volterra series can be applied to derive a nonlinear model for an electronic drive amplifier.

2.2 Nonlinear Amplifier Model based on Truncated Volterra Series

Volterra Series are a wide-spread tool when it comes to modeling practical nonlinear systems. The generic discrete Volterra series of finite order and finite memory is given by (2.1), where $x(n)$ and $y(n)$ are the system input and output signals, M is the memory of the model, P represents the order of the nonlinear effects, and $h_p(m_1, \ldots, m_p)$ are the respective Volterra kernels of each memory element.

$$y(n) = h_0 + \sum_{p=1}^{P} \sum_{m_1=0}^{M-1} \cdots \sum_{m_P=0}^{M-1} h_p(m_1, \ldots, m_p) \prod_{i=1}^{P} x(n - m_i). \tag{2.1}$$

This expression can be simplified by assuming that $h_0 = 0$ and that the Volterra kernels are symmetric[2]. The assumption of symmetry reduces the number of coefficients that have to be calculated in half, as is reflected in the following expression,

$$y(n) = \sum_{p=1}^{P} \sum_{m_1=0}^{M-1} \sum_{m_2=m_1}^{M-1} \cdots \sum_{m_P=m_{P-1}}^{M-1} h_p(m_1, \ldots, m_p) \prod_{i=1}^{P} x(n - m_i).$$

Based on the existing literature, expanding the series to include nonlinear orders beyond $P = 3$ results in small increases in modelling accuracy and much larger strains on computing resources. Hence for our purpose, the drive amplifier will be modeled with a truncated Volterra series where $P = 2$ or $P = 3$. The series expansion for a $P = 3$ model is given in (2.2), where $h_1(m_1)$ are the linear or first order Volterra kernels, $h_2(m_1, m_2)$ are the cuadratic or second order Volterra kernels, and $h_3(m_1, m_2, m_3)$ are the cubic or third order Volterra kernels. Models with different values for M and P will be used throughout the text. Their expanded expression can be obtained by substituting the appropriate M value in (2.2) and disregarding the third order terms if $P = 2$. As an example, a third order $M = 5$ model will entail the calculation of a total of 55 Volterra Kernels, of which 5 are linear, 15 are quadratic and 35 are cubic.

$$\begin{gathered} y(n) = \sum_{m_1=0}^{M-1} h_1(m_1) x(n - m_1) + \ldots \\ \ldots + \sum_{m_1=0}^{M-1} \sum_{m_2=m_1}^{M-1} h_2(m_1, m_2) x(n - m_1) x(n - m_2) + \ldots \\ \ldots + \sum_{m_1=0}^{M-1} \sum_{m_2=m_1}^{M-1} \sum_{m_3=m_2}^{M-1} h_3(m_1, m_2, m_3) \\ x(n - m_1) x(n - m_2) x(n - m_3). \end{gathered} \tag{2.2}$$

[2]This symmetry implies that $h(m_1, m_2, ..., m_n) = h(m_2, m_1, ..., m_n)$.

2.3 Adaptive Algorithms

To compute the Volterra kernels necessary to construct the model, an algorithm capable of calculating them from the input and output signals, $x(n)$ and $y(n)$, is needed. A wide variety of such methods exists for the linear case, out of which the least mean squares (LMS) algorithm and the recursive least squares (RLS) algorithm are some of the more popular options.

2.3.1 Volterra LMS algorithm for the calculation of kernels

The application of a gradient-type LMS adaptive algorithm for the purpose of learning a nonlinear model based on truncated Volterra series will require that a set of specific modifications be applied to the linear version of the algorithm. The derivation of a VLMS (Volterra LMS) algorithm by applying these modifications to the linear LMS algorithm is valid due to the fact that the output of the Volterra series is a linear function with regard to the kernels (the nonlinearity is introduced by means of the cross products of the input signal).

The LMS algorithm for the linear case finds the set of filter coefficients that minimize the instantaneous square of the error signal, which is defined as the difference between the output signal and the estimated signal. This is shown in (2.3), where $\mathbf{x}(n)$ is the input signal vector, $e(n)$ is the error of the algorithm, $d(n)$ is the desired signal, $\hat{\mathbf{h}}(n)$ is the estimated set of filter coefficients, and μ is the step-size of the algorithm.

$$\mathbf{x}(n) = [x(n), x(n-1), \dots, x(n-(M-1))]^\top \tag{2.3a}$$

$$e(n) = d(n) - \hat{\mathbf{h}}^\top(n)\mathbf{x}(n) \tag{2.3b}$$

$$\hat{\mathbf{h}}(n+1) = \hat{\mathbf{h}}(n) + \mu e(n)\mathbf{x}(n) \tag{2.3c}$$

The LMS algorithm extended to the nonlinear case differs in terms of the input vector $\mathbf{x}(n)$, as it must now contain the linear terms (individual samples of the input signal) along with the cross-product terms (cross-products of the input signal samples) that will mimic the nonlinear behaviour of the system. For the third order truncated Volterra series, it is useful to visualize this input vector as the concatenation of three input vectors: $\mathbf{x}_1(n)$ which contains the input signal samples present in the first sum of (2.2), $\mathbf{x}_2(n)$ which contains the products between two input signal samples present in the second sum of (2.2), and $\mathbf{x}_3(n)$ which contains the products between three input signal samples present in the third sum of (2.2). This is shown in (2.4a).

$$\begin{aligned}
\mathbf{x}(n) &= [\mathbf{x}_1(n)^\top \ \mathbf{x}_2(n)^\top \ \mathbf{x}_3(n)^\top]^\top && \text{(2.4a)}\\
\mathbf{x}_1(n) &= [x(n) \ x(n-1) \ \dots \ x(n-(M-1))]^\top\\
\mathbf{x}_2(n) &= [x^2(n) \ x(n)x(n-1) \ \dots \ x(n)x(n-(M-1))\\
&\quad x^2(n-1) \ \dots \ x^2(n-(M-1))]^\top\\
\mathbf{x}_3(n) &= [x^3(n) \ x^2(n)x(n-1) \ \dots\\
&\quad x^2(n)x(n-(M-1)) \ x(n)x^2(n-1) \ \dots\\
&\quad x^3(n-1) \ \dots \ x^3(n-(M-1))]^\top
\end{aligned}$$

In addition, the set of estimated coefficients for the nonlinear case, usually referred to as Volterra kernels, will be different. They are obtained in the form $\hat{\mathbf{h}}(n) = [\hat{\mathbf{h}}_1(n)^\top \hat{\mathbf{h}}_2(n)^\top \hat{\mathbf{h}}_3(n)^\top]^\top$, where each of the sub-indexed vectors contains the linear, quadratic and cubic kernels, respectively. The update rules of the nonlinear LMS algorithm are given in (2.5), where it is easy to see how the linear, quadratic and cubic coefficients are updated independently.

$$\begin{aligned}
\hat{\mathbf{h}}_1(m_1; n+1) &= \hat{\mathbf{h}}_1(m_1; n) + \mu_1 e(n)\mathbf{x}_1(n) && \text{(2.5a)}\\
\hat{\mathbf{h}}_2(m_1, m_2; n+1) &= \hat{\mathbf{h}}_2(m_1, m_2; n) + \dots && \text{(2.5b)}\\
&\quad \dots + \mu_2 e(n)\mathbf{x}_2(n)\\
\hat{\mathbf{h}}_3(m_1, m_2, m_3; n+1) &= \hat{\mathbf{h}}_3(m_1, m_2, m_3; n) + \dots && \text{(2.5c)}\\
&\quad \dots + \mu_3 e(n)\mathbf{x}_3(n)
\end{aligned}$$

How good the coefficient estimates shown in (2.5) are, will depend on the chosen LMS step-sizes, which play a significant role in the algorithms performance. It is well known that the linear LMS algorithm will converge only when $0 < \mu < \frac{2}{\lambda_{\max}}$, where $\lambda_{\max}$ denotes the maximum eigenvalue of the auto-correlation matrix of the input vector $\mathbf{x}(n)$ defined in (2.3). This bound can be extended to the Volterra LMS algorithm by substituting the input vector shown in (2.3) by $\mathbf{x}(n)$ given in (2.4a). However, the use of a single step size makes the convergence rate of the Volterra LMS algorithm depend on the eigenvalue spread of the auto-correlation matrix of the input vector $\mathbf{x}(n)$, which, being much larger than for the linear case often results in slow convergence [3]. Therefore, it is desirable to update the equations shown in (2.5) using different step-sizes; μ_1 for the linear kernels, μ_2 for the quadratic kernels, and μ_3 for the cubic kernels.

It can be difficult to derive bounds for these different step-sizes, since as is shown in [4], the Volterra LMS algorithm does not lend itself well to mathematically tractable analysis that produces concrete upper bounds for them. Step size upper bounds are much simpler to obtain for a variant of the VLMS algorithm known as the partially decoupled VLMS [4]. Regardless, it has been shown in [5] and [6] that despite the fact that no step-size bounds are available for the general case of nonlinear filters, it is reasonable to expect that second-order Volterra filters, due to their property of having an output that is linear in the filter parameters, will share the same convergence condition as the linear case, albeit after slight modifications. The extension of the linear step size LMS bound to the nonlinear second order Volterra case is given by:

$$0 < \mu_i < \frac{2}{\lambda_{\text{max}}}, \tag{2.6}$$

where λ_{max} denotes the maximum eigenvalue of the auto-correlation matrix of the input vectors $\mathbf{x}_i(n)$ shown in (2.4a), where $i = 1, 2$. Although (2.6) has not been proven for the cubic Volterra filter, we will assume its extension to the third order terms as a guideline to select values for μ_3. It has also been shown in [7] that setting a larger step size for linear terms and smaller step sizes for nonlinear terms is beneficial for obtaining more accurate kernel estimates. Therefore, the step sizes for the linear, cuadratic and cubic Volterra kernels will be chosen to fulfill the bounds shown in (2.6) following the recommendation given in [7].

2.3.2 Volterra RLS algorithm for the calculation of kernels

Another option for the estimation of the Volterra kernels is to employ the RLS algorithm. Because it is not the primary algorithm choice in this thesis, the text merely provides a summary of its functioning. Extensive analysis of the RLS algorithm is conducted in [8]. Maintaining conventional RLS algorithm notation, the computative aspects of the linear RLS algorithm are shown in (2.7), where λ is known as the forgetting factor, $\mathbf{R}^{-1}(n)$ is an $M \times M$ matrix, $\mathbf{k}(n)$ is an M-dimensional vector known as the Kalman gain, and the rest of the parameters are identical to the LMS method. The last two expressions given in (2.7) are the update rules of the algorithm.

$$\mathbf{x}(n) = [x(n), x(n-1), \dots, x(n-(M-1))]^\top \tag{2.7a}$$

$$e(n) = d(n) - \mathbf{x}^\top(n)\hat{\mathbf{h}}(n) \tag{2.7b}$$

$$\mathbf{k}(n) = \frac{\mathbf{R}^{-1}(n)\mathbf{x}(n)}{\lambda + \mathbf{x}^\top(n)\mathbf{R}^{-1}(n)\mathbf{x}(n)} \tag{2.7c}$$

$$\mathbf{R}^{-1}(n+1) = \frac{1}{\lambda}(\mathbf{R}^{-1}(n) - \mathbf{k}(n)\mathbf{x}^\top(n)\mathbf{R}^{-1}(n)) \tag{2.7d}$$

$$\hat{\mathbf{h}}(n+1) = \hat{\mathbf{h}}(n) + e(n)\mathbf{k}(n) \tag{2.7e}$$

The linear RLS algorithm must also be adapted in order to be validly applied in the estimation of nonlinear kernels. This procedure is straightforward given the LMS derivation conducted in the previous section. For a third order Volterra truncated series, the input vector $\mathbf{x}(n)$ is given by (2.4a), which leads to the following modifications to the linear RLS update rules:

$$\begin{aligned}
\mathbf{k}_i(n) &= \frac{\mathbf{R}_i^{-1}(n)\mathbf{x}_i(n)}{\lambda + \mathbf{x}_i^\top(n)\mathbf{R}_i^{-1}(n)\mathbf{x}_i(n)}, \\
\mathbf{R}_i^{-1}(n+1) &= \frac{1}{\lambda}(\mathbf{R}_i^{-1}(n) - n\mathbf{k}_i(n)\mathbf{x}_i^\top(n)\mathbf{R}_i^{-1}(n), \\
\hat{\mathbf{h}}(n+1) &= \hat{\mathbf{h}}(n) + e(n)\begin{bmatrix}\mathbf{k}_1 \\ \mathbf{k}_2 \\ \mathbf{k}_3\end{bmatrix}.
\end{aligned}$$

where $i = 1, 2, 3$.

For the RLS algorithm to run, it requires the setting of initial values for λ and $\mathbf{R}_i^{-1}(0)$. In general, $\mathbf{R}_i^{-1}(0) = \delta_i I$, where I is the identity matrix of size M_ixM_i and δ_i is a positive scalar chosen to ensure that $\mathbf{R}_i^{-1}(n)$ is not singular for small values of n. This guarantees convergence of the algorithm in its initial stages. The setting of these initialization parameters is discussed in more detail in chapter 3.

Chapter 3

Practical extraction of amplifier behavioural models

In this chapter, we will discuss the methods, data, and tools that are employed to extract the previously mentioned nonlinear Volterra models. In addition, the techniques that are derived to study the extracted models will be explained. The chapter is closed with a discussion regarding a set of preliminary results.

The correct derivation of the nonlinear behavioural models will require two distinct factors: a software framework and real amplifier measurements. The implemented software must process the amplifier data so that it can be fed into the adaptive algorithm, also software designed, that will generate the Volterra model. This software framework will also include the necessary tools to study the learned amplifier models. The amplifier measurements must provide the necessary input and output data that will allow the adaptive algorithm to learn appropriate Volterra models of the device. As will be shown throughout the rest of the text, a plethora of amplifier measurements will be required to achieve this goal.

3.1 Software simulation framework

To learn the necessary Volterra models of the electronic drive amplifier, the software framework must implement one of the kernel estimating algorithms explained in Chapter 2. The framework will be programmed using Matlab. Figure 3.1 provides a graphical description of how the Matlab implementation of the estimating algorithm functions.

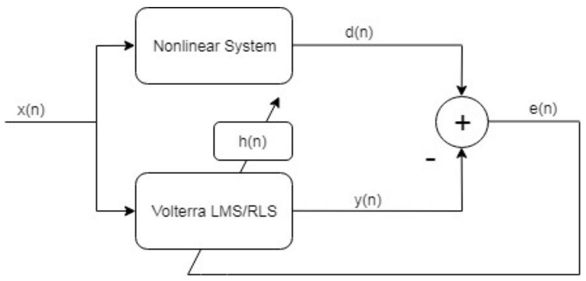


FIGURE 3.1: Matlab framework overview.

As can be seen in the above figure, the estimating algorithm will require the input signal of the drive amplifier, $x(n)$, and its corresponding output signal (also referred to as the desired signal), $d(n)$. How these amplifier input and output signals are recorded is discussed in section 3.2. The rest of this section explains the processing of this raw amplifier data before it can be provided to the kernel estimating algorithm.

3.1.1 Signal Time Alignment

It is necessary for the adaptive algorithm to correctly estimate the Volterra kernels that the drive amplifier input and output signals, $x(n)$ and $d(n)$, be aligned in time. This means that for any given sample index n_1, the output signal sample $d(n_1)$ was produced when the drive amplifier received signal sample $x(n_1)$ at its input. Therefore, the first step that is conducted in Matlab is to check the time alignment of the input and output signals, since the way in which they where measured may have led to them being delayed with regard to one another.

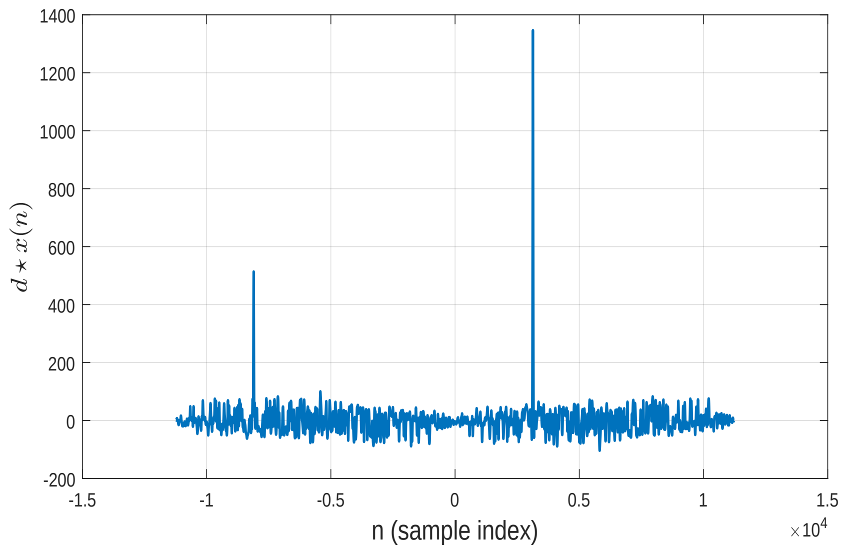


FIGURE 3.2: Cross correlation between $x(n)$ and $d(n)$. The input is a 32 Gb/s NRZ signal with a 511 bit pattern length and a 450 mV peak-to-peak voltage. The equivalent time sampling rate was 22 samples/symbol.

The time alignment is ensured by calculating the cross-correlation between $x(n)$ and $d(n)$, finding the location of it's maximum, and then shifting the delayed signal forward by the amount of samples that the cross-correlation maximum was located at. Figure 3.2 shows the cross-correlation of an experimentally measured input and output signal pair. As is shown in the figure, one of the signals is delayed with regard to the other (the cross-correlation maximum does not occur at 0) and will require sample shifting to guarantee that $x(n)$ and $d(n)$ are aligned in time.

3.1.2 Signal Normalization

Once the input and output signals have been aligned in time, the next step is to normalize their sample values. Two normalization methods are presented in this subsection. The first one can be applied to restrict the signal sample values so that $x(n)$, $d(n) \in [-1, 1]$, $\forall n$. This is shown in (3.1), where $\tilde{x}(n)$ and $\tilde{d}(n)$ denote the normalized amplifier input and output signals, respectively.

$$\begin{aligned}\tilde{x}(n) &= \frac{x(n) - S_x}{G_x} \;\forall n,\\ \tilde{d}(n) &= \frac{d(n) - S_d}{G_d} \;\forall n,\end{aligned} \tag{3.1}$$

$$\begin{aligned}S_x &= \frac{\max(x(n)) + \min(x(n))}{2},\\ G_x &= \frac{\max(x(n)) - \min(x(n))}{2},\\ S_d &= \frac{\max(d(n)) + \min(d(n))}{2},\\ G_d &= \frac{\max(d(n)) - \min(d(n))}{2}.\end{aligned}$$

The second method of normalization is quite similar to the first, but instead of taking the minimum and maximum values of each signal, the average values of the eye diagram rails are taken. This normalization procedure is shown in (3.2), where $\hat{x}_1$, $\hat{x}_{-1}$ denote the average values of the eye diagram rails of $x(n)$ and $\hat{d}_1$, $\hat{d}_{-1}$ represent those average values for $d(n)$.

$$\begin{aligned}\tilde{x}(n) &= \frac{x(n) - A_x}{B_x} \;\forall n,\\ \tilde{d}(n) &= \frac{x(n) - A_d}{B_d} \;\forall n,\end{aligned} \tag{3.2}$$

$$\begin{aligned}A_x &= \frac{\hat{x}_1 + \hat{x}_{-1}}{2},\\ B_x &= \frac{\hat{x}_1 - \hat{x}_{-1}}{2},\\ A_d &= \frac{\hat{d}_1 + \hat{d}_{-1}}{2},\\ B_d &= \frac{\hat{d}_1 - \hat{d}_{-1}}{2}.\end{aligned}$$

Figure 3.3 depicts a close-up image of an eye diagram corresponding to one of the drive amplifier output NRZ signals measured in the lab. It shows how the normalization values for both methods are picked. These methods effectively decrease the dynamic range of $d(n)$ (amplified signal) and increase it for $x(n)$ (input signal). The goal of this normalization is to ensure that the Volterra models that are learned are not conditioned by the signal level. This will allow us to draw comparisons between Volterra model coefficients when the voltage swing of the signals for which these coefficients were learned is different. The normalization will cast the amplifier in the framework of a unit gain component. However, this is not an issue since our interest lies with modelling the amplifier nonlinearities and not with analyzing the multiplicative factor that is its gain. For the simulations and results shown in later parts of this thesis, the first normalization method has been employed.

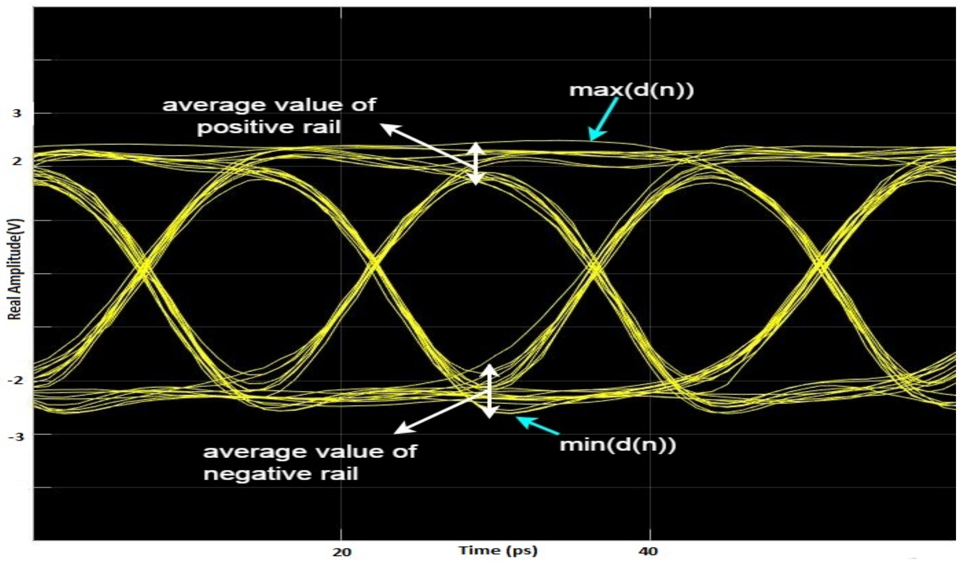


FIGURE 3.3: Eye diagram of a measured drive amplifier output NRZ signal. The blue arrows show how the first normalization procedure picks its values while the white arrows portray the way the second procedure does it.

3.1.3 Algorithm Initialization

At this point, the drive amplifier input and output signals have been successfully time aligned and normalized. The final step before running the kernel estimating algorithm is to define its corresponding initialization parameters to guarantee algorithm convergence. For both the LMS and RLS Volterra algorithms, the kernel vector, $\hat{\mathbf{h}}(n)$, is initialized with its entries set to 0.

For the Volterra LMS algorithm, the only other condition that must be fulfilled is given by (2.6). Ideally, the steps of the VLMS algorithm would be defined by multiplying the RHS of (2.6) by a factor $\sigma_i < 1$. This way, $\mu_i = \sigma_i \frac{2}{\lambda_{\max}} < \frac{2}{\lambda_{\max}}$, which would ensure the convergence of the algorithm. In practice, to avoid the computational cost of constantly calculating the auto-correlation matrices of the input vector, values small enough to perpetually satisfy (2.6) are defined. The optimum step size values out of those that fulfill the previous condition are determined by conducting simulations and comparing which set of μ_i values produces the smallest mean square error. The MSE is calculated as shown in (3.3), where N is the length of the sliding rectangular window used to time average $e^2(n)$. The results of these simulations, as well as the specific μ_i values that were chosen for the algorithm, are shown later in this section.

$$MSE(\text{dB}) = 10\log_{10}\big(\frac{1}{N}\sum_{n=1}^{N} e^2(n)\big) \tag{3.3}$$

For the VRLS algorithm, the forgetting factor, λ, as well as $\mathbf{R}_i^{-1}(0)$ must be defined. Typically $0 \ll \lambda < 1$, with lower values of λ leading to better tracking capabilities of the algorithm (used in real-time applications) and higher values leading to better stability and adjustment. In terms of $\mathbf{R}_i^{-1}(0)$, we know that $\mathbf{R}_i^{-1}(0) = \delta_i I$, where the δ_i values must be chosen to guarantee that $\mathbf{R}_i^{-1}(n)$ does not become singular for small values of n. The value for δ_i will be conditioned by the choice of λ, with [8] and [9] recommending the choice of a large values for δ_i. A detailed analysis of how to optimally select this parameter is provided in [10].

3.2 Lab Experiments

This section discusses the nature, structure, and characteristics of the lab experiments conducted to obtain amplifier input and output signals. In section 3.3, these signal measurements will be processed using the computational tools explained previously in this chapter. Broadly speaking, these lab experiments come down to feeding an electronic drive amplifier with different input signals, and recording both the input and amplifier output signals.

Naturally, it is difficult to accurately determine which input signals will yield the data necessary to completely characterize the amplifier on the first try. Therefore, two lab experiments had to be performed throughout the duration of this thesis. The main objective of the first lab visit was to obtain data that would enable the development and verification of specific computational tools. These tools could later be used to thoroughly analyze posterior lab measurements. In addition, the data collected in this first experiment was also meant to provide knowledge on which measurements to perform in later lab visits so that the amplifier nonlinear behaviour could be completely characterized.

Although not random, this first iteration of measurements was not exhaustive enough to infer or deduce rigorous conclusions from the data regarding amplifier nonlinearity.

After working with the first set of measurements and once the necessary computational and analytic tools were in place, a second lab experiment was conducted. The data recorded during this second iteration of measurements was significantly more extensive than in the previous iteration. Intuitively, we know that the Volterra models that are learned by the adaptive algorithm will change according to the input signals fed to the amplifier. In consequence, the aim of this second lab experiment was to record sufficient and appropriate data to completely understand the effects on the learned Volterra model of varying input signal characteristics, with the end goal of designing the most complete behavioural amplifier model possible.

3.2.1 Experimental Setup

The lab setup employed to record the necessary signal measurements was identical for both experiments.

In Table 3.1 all the equipment that was used during the experiments is listed, while in figures 3.4 and 3.5, the equipment configuration used to record the signal measurements is shown.

The functioning of the measurement setup is briefly described as follows:

1. The Anritsu SG generates the clock signal and provides it to the BPG.
2. The BPG will synthesize the desired data signal as well as relaying the clock signal to the modules of the DCA.
3. The data signal generated by the BPG is fed into the amplifier if we are recording output signals, or directly to the DCA if we are recording input signals.
4. The DCA stores the desired data.

Although relatively simple in principle, the equipment had to be configured following careful and particular steps if it was to record the correct data. The following subsection explains the details of appropriately deploying and connecting each piece of lab hardware that was used during the experiment.

Table 3.1: Equipment used in the lab experiments.

Equipment	Brand	Model
Signal Generator (SG)	Anritsu	MG3697C 67 GHz
Bit Pattern Generator (BPG)	SHF	12103A
Digital Communications Analyzer (DCA)	Agilent Tech.	86100D
Remote Sampling Module	Agilent Tech.	86118A 60 GHz
Precision Timebase Module	Agilent Tech.	86107A 10/20 GHz
Electronic Amplifier	SHF	807 30397 30 GHz
6dB Attenuator	-	-
10 dB Attenuator	-	-
Remote Sampling Head	-	-

3.2.2 Equipment Configuration

The SG generates the clock signal and provides it to the BPG. The BPG has different selectable signal channels. Therefore, the clock signal frequency must be adjusted according to the selected channel on the BPG. In our experiments, we chose BPG channel A×C, which is generated by multiplexing channels A and C. This means that the signal frequency specified as the input to the SG has to be half of the desired signal frequency as the final signal will be created by multiplexing two lower frequency channels of the BPG. For example, if we want to produce a 50 Gbps data signal, we will need a 50 GHz clock. However, we must input a frequency of 25 GHz to the SG. This provides a 25 GHZ clock signal to the BPG, which will synthesize two independent 25 Gbps data signals on channels A and C that will ultimately be multiplexed into a 50 Gbps signal present at the output of channel A×C.

The BPG receives the clock signal from the SG and generates the input data signal for our experiment. This data signal is a pseudo-random binary sequence (PRBS) with a specific peak-to-peak voltage, data rate, and bit pattern length modulated using a non-return to zero (NRZ) line code. In order for the DCA to be able to correctly capture this data signal, we must reduce the high frequency clock signal by using the selectable clock feature of the BPG and dividing the clock signal coming from the SG by an appropriate factor. Recalling the previous example, the BPG sees a 25 GHz signal from the SG. We select the Clk/4 setting of the BPG, which results in a 6,25 GHz clock signal.

At the output of the BPG, the clock signal is split into two different paths so that it can be provided to the precision timebase module of the DCA, as well as its front panel trigger. Using the precision timebase module enables the pattern lock function of the DCA, which is capable of automatically acquiring transmitted bit patterns in the event that they are periodic and recording the entire waveform in an excel file. This feature is quite useful, but must be carefully set up as follows:

1. Set the front panel trigger as the trigger source of the DCA.
2. In the Pattern Lock panel of the DCA, select the Clock/Divided (3 GHz - 13 GHz) trigger bandwidth option. This will let the DCA know that the we are using the selectable clock feature of the BPG.
3. Finish setting up Pattern Lock by manually introducing the pattern length of the signal being transmitted by the BPG.
4. Allow the DCA to automatically acquire the data rate and trigger divide ratio of the data. The trigger divide ratio seen by the DCA will be half of that selected at the BPG because of the fact that it sees the multiplexed channel AxC as a single channel. For our previous example, because the DCA sees a single 50 Gbps channel and a 50 GHz clock signal, it would acquire a 1:8 trigger divide ratio, since $\frac{50}{8} = \frac{25}{4} = 6,25GHz$.
5. Given that the Pattern Lock feature allows the DCA to perform automatic data acquisition, the DCA will autoconfigure the Acquisition Setup.
6. The last step is to acquire waveforms that are as smooth as possible. This is ensured by selecting the Average Smoothing feature of the DCA, which instead of acquiring a single run of the desired waveform, will obtain a specific number of runs and then average them. For our experiments, this feature was set to average over 16 different waveform acquisitions.

We must now distinguish the parts of the setup that vary depending on whether input or output signals are being measured. When amplifier output signals are being recorded, the data signal generated by the BPG, which is essentially a bit pattern of a specific length and data rate, is fed into a 6 dB attenuator. The attenuator is connected to the electronic amplifier, whose output is fed to a 20 dB attenuator (created by connecting two 10 dB attenuators in series). The output of this 20 dB attenuator is connected to the remote sampling head, which is attached to the remote sampling module of the DCA. The 6 dB and 20 dB attenuators are added to decrease the voltage swing of the amplifier input and output signals so that, after the signals are amplified, the output peak-to-peak swing does not become too large for the DCA. This is shown in figure 3.4. The DCA interface must be told of the presence of these attenuators, as it can account for them when performing data acquisition.

When recording input signals (amplifier is not connected), the data produced by the BPG is fed to the remote sampling head after travelling through the 6 dB attenuator. The 6 dB attenuator is maintained so that the same amplifier input signals can be recorded. As previously, the sampling head is connected to the DCA front panel trigger. This is shown in figure 3.5.

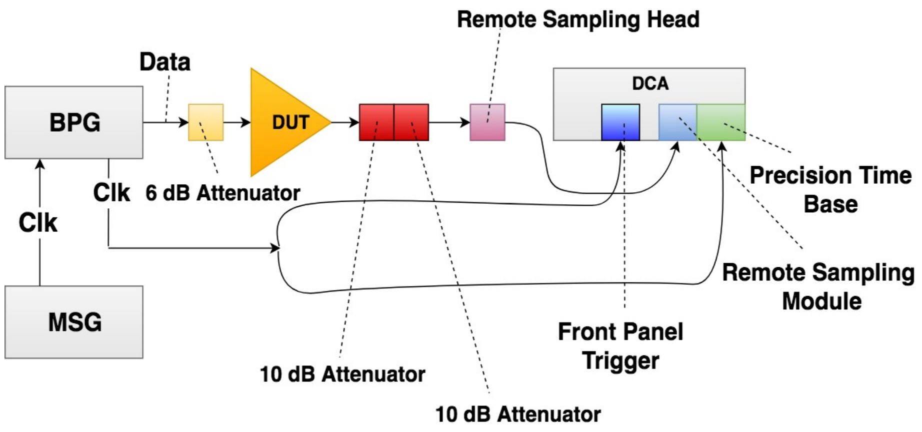


FIGURE 3.4: Configuration used in the lab to record the amplifier output signal measurements.

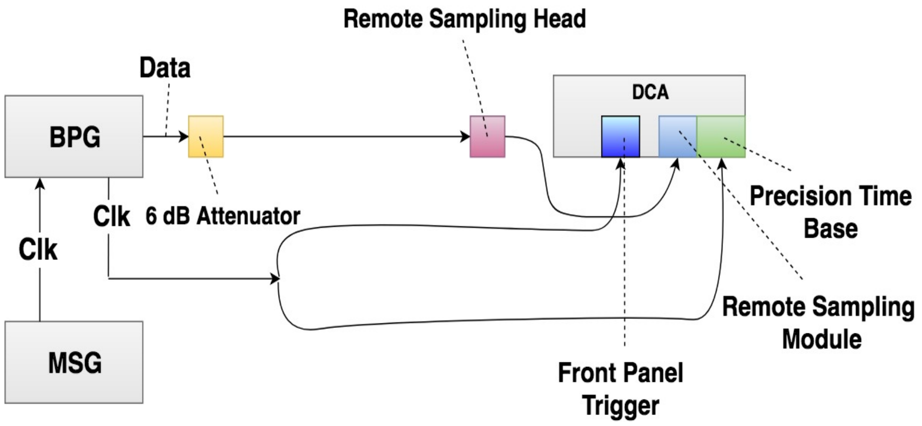


FIGURE 3.5: Configuration used in the lab to record the amplifier input signals.

The pairs of input and output waveforms are stored by the DCA after averaging the signal samples over 16 acquisitions of that same signal. This DCA acquisition mode averages the collected data on a point-by-point basis, improving the signal-to-noise ratio, removing uncorrelated noise, and making the repetitive waveform smoother. The stored signals are then introduced in the Matlab framework to learn the corresponding Volterra models and develop further computational analytic tools.

3.2.3 Lab Experiment #1

As has been mentioned previously in this section, the goal of the first lab experiment was to obtain input and output signal measurements that would provide initial insight and sufficient information to develop the necessary tools to analyze posterior lab measurements. Table 3.2 shows the input and output signals that where stored in this first measurement iteration. In both experiments, all of the input signals where NRZ[1] signals.

TABLE 3.2: Characteristics of the signals used as the input of the electronic drive amplifier in the first lab visit. The equivalent time sampling rate is discussed in later in this chapter.

Modulation	Rate (Gb/s)	Pattern Length (bits)	Voltage (mVpp)	Equivalent Time Sampling Rate (Samples/symbol)	Waveform Length (samples)
NRZ	32	2^9-1	450	22	11241
NRZ	32	2^9-1	675	22	11241
NRZ	32	2^9-1	900	22	11241
NRZ	32	$2^{15}-1$	450	22	720873
NRZ	32	$2^{15}-1$	675	22	720873
NRZ	32	$2^{15}-1$	900	22	720873
NRZ	50	2^9-1	525	29	14818
NRZ	50	2^9-1	650	29	14818
NRZ	50	$2^{15}-1$	525	29	950242
NRZ	50	$2^{15}-1$	650	29	950242

3.2.4 Lab Experiment #2

Because of the time spent working on the initial measurements, it was much easier to determine which signals to use for the second batch of experiments. The aim of this second lab visit was to record data that would allow us to determine the effects of varying input signal amplitude, pattern length, and data rate on the learned Volterra models. Because of this, this second measurement iteration is much more exhaustive than the first, as it aims to completely characterize the impact caused by these input signal parameters. Given that there are three input signal parameters of interest, hence three degrees of freedom, the measurements are defined by repeating experiments in which one degree of freedom is varied and the others are kept the same. This is shown in Table 3.3, where the characteristics of the input signals used in the second visit to the lab can be seen.

Later in this section, the measurements obtained in the first lab experiment are used to obtain results and make preliminary observations. The second iteration of measurements is discussed in Chapter 5.

[1]An NRZ line code is a simple binary coding scheme in which ones are typically represented by a positive voltage V and zeros are represented by a negative voltage $-V$.

TABLE 3.3: Characteristics of the signals used as the input of the electronic drive amplifier in the second lab visit.

Modulation	Rate (Gb/s)	Pattern Length (bits)	Voltage (mVpp)	Equivalent Time Sampling Rate (Samples/symbol)	Waveform Length (samples)
NRZ	50	2^9-1	650	15	7664
NRZ	50	$2^{11}-1$	650	15	30704
NRZ	50	$2^{15}-1$	650	15	491504
NRZ	38	$2^{15}-1$	650	19	622572
NRZ	44	$2^{15}-1$	650	16	524271
NRZ	56	$2^{15}-1$	650	15	425970
NRZ	56	$2^{15}-1$	600	13	425970
NRZ	56	$2^{15}-1$	500	13	425970
NRZ	56	$2^{15}-1$	400	13	425970
NRZ	50	$2^{15}-1$	600	15	491504
NRZ	50	$2^{15}-1$	500	15	491504
NRZ	50	$2^{15}-1$	400	15	491504

3.3 Processing the Initial Measurements

We will now discuss the processing of the drive amplifier data (input and output signals) measured in the first lab experiment with the adaptive algorithms explained in Chapter 2. As has been shown at the beginning of this section, prior to applying the signals recorded during the experiment to the Volterra adaptive algorithm, they are time aligned and normalized to the bipolar range ([-1 1]) by means of the normalization method given in 3.1. The algorithm chosen to learn the Volterra models is the VLMS algorithm. The Matlab code which implements this algorithm is shown in Appendix A. The number of iterations for which the VLMS algorithm is run is discussed in the following paragraph. The memory and order of the learned model are set to the default values of $M = 5$ and $P = 3$. The appropriate manner of selecting the values for M and P is discussed in a later chapter of the thesis.

3.3.1 Application of the VLMS Algorithm

The VLMS Algorithm explained in Chapter 2 can be used to extract Volterra models from the measurements conducted in the first lab experiment. Performing an objective and unbiased study of these learned models will require that an appropriate way to establish the values of VLMS iterations and VLMS step sizes be devised.

3.3.1.1 VLMS Iterations

As is shown in Table 3.2, the number of samples of the input/output signal pairs varies depending on the characteristics of the input signal. This implies that the VLMS algorithm will require more or less iterations (total number of signal samples) to learn the model depending on the type of amplifier input signal. Said variation in algorithm iterations as a function of signal length will not permit an objective study of its convergence, making the step-sizes of the VLMS algorithm dependant on the sample set length of the input/output signal pairs.

To circumvent this issue, the number of algorithm iterations is decoupled from the length of the discrete waveforms by periodically replicating them. The shorter sequence signals are successively concatenated to construct a periodic 'repetition' waveform consisting of as many symbols, ergo as many samples, as the longer sequence signals. This entails concatenating the signals with shorter pattern length ($2^9 - 1$) enough times to guarantee that the length of these extended sequences is approximately that of the signals with longer pattern length ($2^{15} - 1$), which results in the periodic 'repetition' waveform being constructed by concatenating the $2^9 - 1$ pattern length sequences 64 times ($64 \approx \frac{2^{15}-1}{2^9-1}$). This way, the issue of adjusting algorithm step sizes in relation to signal sample set length is effectively averted, allowing the study of algorithm convergence behaviour for a common set of step sizes for all recorded signals.

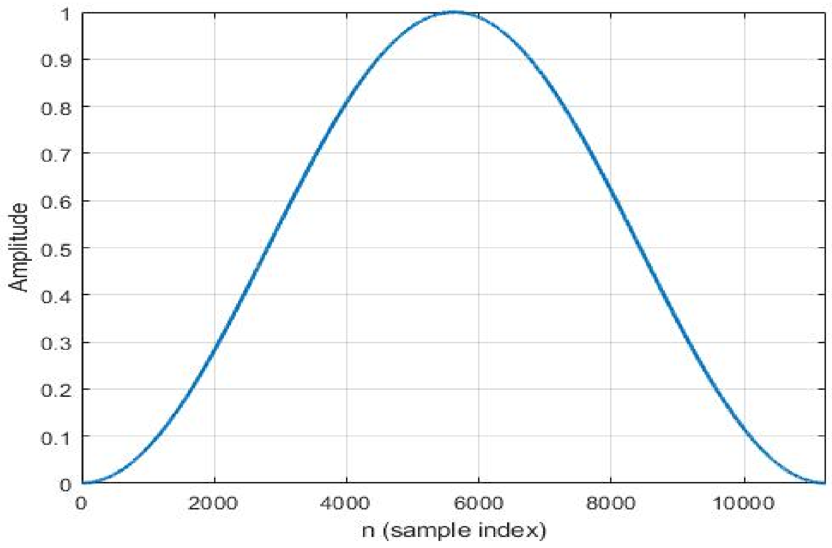


FIGURE 3.6: Impulse response of a Hanning window of length $L = 11242$. It is shown later in this chapter how such a window is an appropriate choice for tapering when using the overlap and add method.

To soften the effects of sequence concatenation, a tapering window can be applied to the successively concatenated waveform so that the extended periodic 'repetition' signal does not exhibit strong discontinuities at the instances where the sequence endpoint and beginning meet. The falling taper at the end of the last concatenated sequence slowly decreases to zero while the rising taper at the beginning of the sequence to be added will gradually increase to one, which serves to ameliorate the effects of abrupt sequence endpoints. The tapering window selected for this purpose has to satisfy a condition in the time domain akin to the Nyquist zero ISI criterion. An example of a such a window is the Hanning window defined according to the expression shown in 3.4. This topic is will be further discussed in 3.4.1.

$$w(n) = 0.5\big(1 - \cos\big(2\pi\frac{n}{N}\big)\big),\ 0 \le n \le N. \tag{3.4}$$

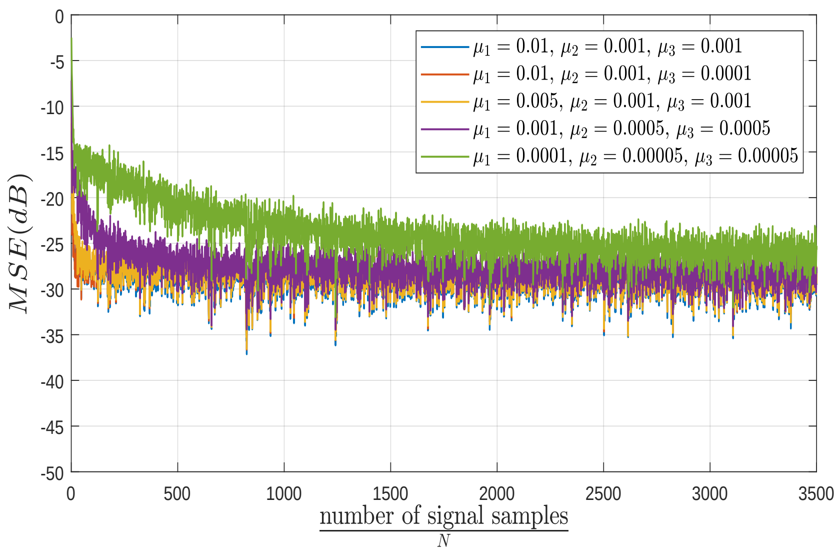
where n represents the signal the window is being applied to and $L = N + 1$ is the window length. Figure 3.6 shows the shape of an $L = N + 1$ ($N = 11241$) Hanning window that could be applied to the short sequence signal prior to concatenation.

The concept of applying a tapering window was initially proposed due to a lack of knowledge regarding the exact contents of the sequences being concatenated. Windowing alone or in conjunction with overlap and add methods constitutes an effective way of softening the addition of the sequence endpoints and beginnings. However, as will be shown in 3.4.1, the sequences that are being concatenated end and begin in approximately the same point in time, hence, direct concatenation will be optimal and no softening of endpoints will be required.

3.3.1.2 VLMS Step Size

The optimum step sizes for the VLMS algorithm were determined by conducting Matlab simulations. Table 3.4 shows the different values of μ_i that where tested along with their final MSE (MSE computed during the last 10 windowing events of (3.3)). Figure 3.7 portrays the algorithm convergence curves obtained for said μ_i combinations. The top image of Figure 3.7 serves to discard two of the μ_i combinations, while the bottom one shows how the curve obtained for $\mu_1 = 0.01, \mu_2 = 0.001, \mu_3 = 0.001$ yields the lowest MSE.

The choice of setting $\mu_2, \mu_3 < \mu_1$ is based on observations made in [11], which show that better estimates are made by Volterra equalizers when the nonlinear step-sizes are kept smaller than the linear one. A principled way to select the step sizes for the nonlinear coefficients is provided in [12].



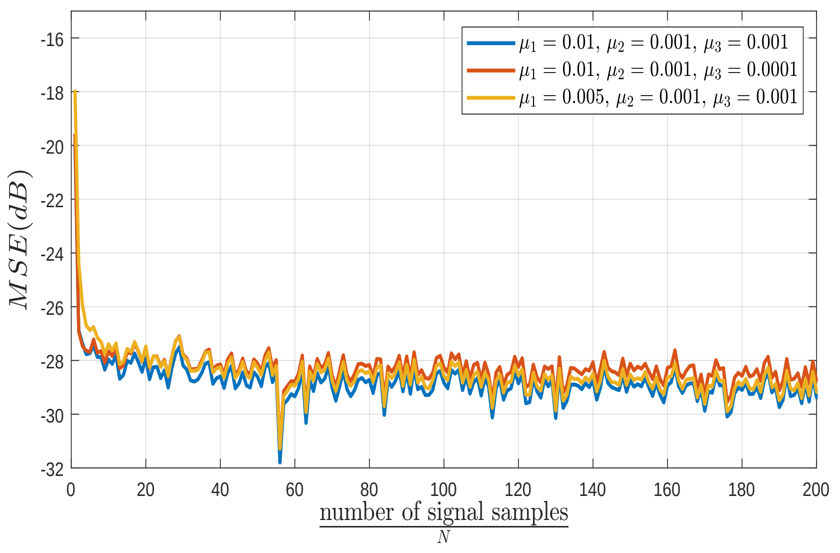


FIGURE 3.7: Evolution of the MSE as a function of the VLMS iterations. The top image has been obtained by using a sliding rectangular window of length $N = 400$. The bottom image has been obtained by using a rectangular window of length $N = 4000$.

TABLE 3.4: Logarithmic MSE results for different values of μ_i.

μ_1	μ_2	μ_3	MSE(dB)
0.01	0.001	0.001	-29.17
0.01	0.001	0.0001	-28.61
0.005	0.001	0.001	-29.01
0.001	0.0005	0.0005	-28.26
0.0001	0.00005	0.00005	-25.91

3.3.2 Interpretation of the Volterra models

Using the Volterra kernels estimated by the VLMS algorithm, Volterra models for each specific pair of measured input and output signals can be constructed by substituting those estimates in (2.2) for $M = 5$. However, it will be complicated to compare models and make observations based solely on the individual values of their Volterra kernels. Ideally, we should find some kind of measure capable of providing information about the amplifier response without having to study the entire kernel content of the associated Volterra model.

3.3.2.1 Volterra Normalized Kernel Weights

An appropriate set of metrics to characterize the Volterra models is designed by combining the kernels that make up the model in a relatively straightforward way. Given a Volterra model of order P, there will be P metrics that completely characterize the Volterra model: one metric will weigh the value of the linear Volterra kernels, another will weigh the value of the quadratic Volterra kernels, a third one will weigh the value of the cubic Volterra kernels, and so on until the final P-th metric, which will weigh the value of the Volterra kernels of order P. These metrics represent the normalized square sums of each subset of kernels of a specific nonlinear order and will be referred to from this point onwards as the normalized kernel weights.

Calculation of these normalized kernel weights is achieved by processing the Volterra kernels as shown in (3.5), where $\tilde{h}_1$, $\tilde{h}_2$, and $\tilde{h}_3$ represent the normalized linear, quadratic, and cubic kernel weights, respectively. Computing the expressions shown in (3.5) for each of the learned Volterra models provides a quicker and more efficient way to study and compare their behaviour. For instance, by looking at Figures 3.8 and 3.9 which depict the normalized kernels of two Volterra models learned from the measurements recorded in the first experiment, we can see how these metrics vary depending on the characteristics of the input signals.

$$\tilde{h}_1 = \frac{\sum_{m=0}^{4} h_1^2(m)}{\nu}, \tag{3.5a}$$

$$\tilde{h}_2 = \frac{\sum_{m_1=0}^{4} \sum_{m_2=m_1}^{4} h_2^2(m_1, m_2)}{\nu}, \tag{3.5b}$$

$$\tilde{h}_3 = \frac{\sum_{m_1=0}^{4} \sum_{m_2=m_1}^{4} \sum_{m_3=m_2}^{4} h_3^2(m_1, m_2, m_3)}{\nu}, \tag{3.5c}$$

$$\nu = \sum_{m=0}^{4} h_1^2(m) + \sum_{m_1=0}^{4} \sum_{m_2=m_1}^{4} h_2^2(m_1, m_2) + \ldots$$

$$\ldots + \sum_{m_1=0}^{4} \sum_{m_2=m_1}^{4} \sum_{m_3=m_2}^{4} h_3^2(m_1, m_2, m_3).$$

Table 3.5 provides the numerical values of the normalized kernel weights of four different models learned from the measurements obtained during the first lab experiment. Once more, variation in the normalized kernel weights is observed as input signal parameters change. As will be explained in the following subsection, these results serve to provide valuable insight which will be used to determine which specific measurements to record in future lab work.

TABLE 3.5: Normalized Volterra kernel weights for various input signals.

Signal	$\tilde{h}_1$	$\tilde{h}_2$	$\tilde{h}_3$	$\tilde{h}_2 + \tilde{h}_3$
32 Gb/s $2^9 - 1$ bits (pattern) 675 mVpp input	0.952	0.021	0.027	0.048
32 Gb/s $2^{15} - 1$ bits (pattern) 675 mVpp input	0.950	0.004	0.046	0.050
50 Gb/s $2^9 - 1$ bits (pattern) 650 mVpp input	0.905	0.004	0.091	0.095
50 Gb/s $2^{15} - 1$ bits (pattern) 650 mVpp input	0.899	0.006	0.095	0.101

3.3.2.2 Preliminary Observations

Having come up with a way of effectively characterizing each learned Volterra model, we can now use the normalized kernel weights to compare and study these models.

Prior to the experiment and simulation, one might have intuitively assumed that an increase in amplifier nonlinearity would be observed when the input signal voltage was

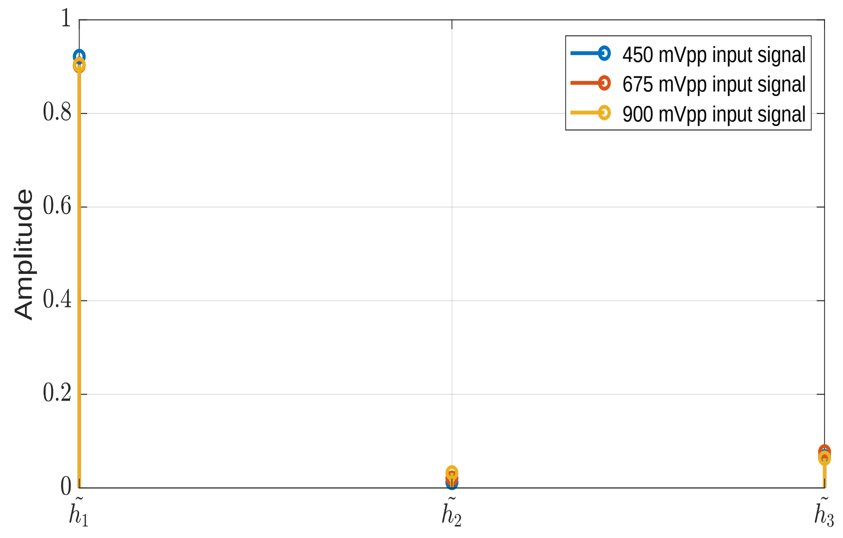


FIGURE 3.8: Normalized Volterra kernel weights learned for signals with a pattern length of 511 bits and a data rate of 32 Gb/s.

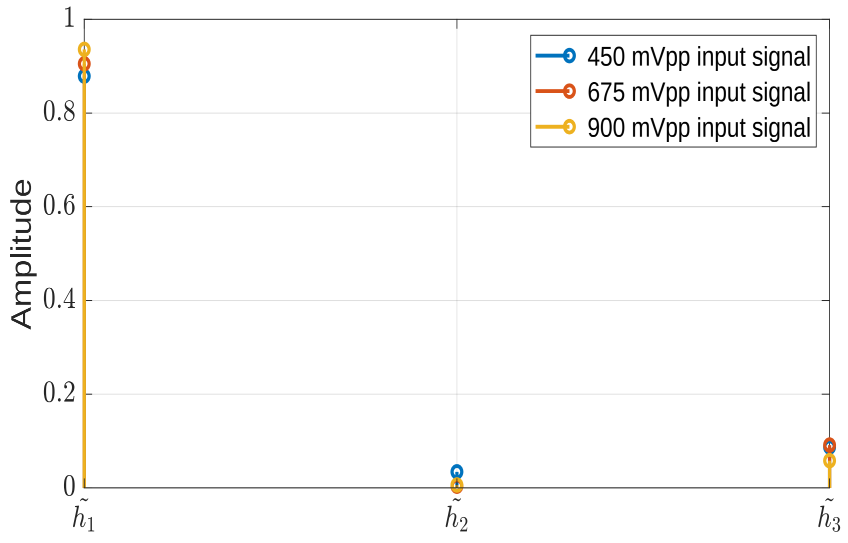


FIGURE 3.9: Normalized Volterra kernel weights learned for signals with a pattern length of 32767 bits and a data rate of 32 Gb/s.

increased. In addition, a change in amplifier behaviour may also have been expected if input signal data rate was varied. We can now verify both these hypotheses by studying the normalized kernel weights of the Volterra models shown in Table 3.5 and the multiple figures in this section. Figure 3.9 shows how the learned Volterra model changes when varying the input signal voltage and keeping the data rate and bit pattern length constant. Table 3.5, shows how the weight of $\tilde{h}_2 + \tilde{h}_3$ increases when the data rate goes up and the input voltage[2] and bit pattern length are kept constant. Curiously,

[2]The input voltage changes slightly from 675 mVpp at 32 Gb/s to 650 mVpp at 50 Gb/S, but this variation is so small that it is assumed to play a negligible part in the change in nonlinear amplifier content.

the results in Table 3.5 reveal that the increase in the weight of the nonlinear kernels is slightly larger for a pattern length of $2^{15}-1$ bits than for a pattern length of $2^{9}-1$ bits.

These preliminary trends serve to show that amplifier behaviour and it's nonlinear content, ergo the learned Volterra models, are tied to three input signal parameters: the peak-to-peak voltage, the data rate, and the bit pattern length. However, the data sets derived from the first lab experiment are not sufficient to study and determine the role each input signal parameter plays in modifying the amplifier response and so more lab measurements will be required. In Chapter 5, the relationship between these parameters and amplifier response is studied much more thoroughly based on the denser set of measurements obtained during the second lab experiment.

3.3.3 Model Fidelity, Validity & Accuracy

Despite all of the work and insight that has been obtained up to this point of the thesis, only a single way of determining the accuracy of the Volterra models has been shown. This method was discussed in subsection 3.1.3, and it revolves around the computation of the MSE, which represents a measure of the error that the adaptive algorithm commits when modeling the output signal from the input. The lower the value of the MSE of the model, the more accurate it will be.

Another possibility when analyzing how accurate a portrayal of the amplifier output the Volterra models provide, is to study the eye diagrams of recorded lab output signals and compare them to the eye diagrams of the corresponding Volterra modelled output signals. Such comparisons can be seen in Figures 3.10 and 3.11, where the input, real output, and modelled output signal eye diagrams for two different measurement sets are shown. These measurement sets are the input and output signal pairs recorded for a 32 Gb/s NRZ input signal with 675 mVpp input voltage and for a 50 Gb/s NRZ signal with 650 mVpp input Voltage. Both signals had a pattern length of $2^{15}-1$ bits. All of the characteristics of these measurements are shown in Table 3.2. Although difficult to numerically quantify the fidelity of the models using this technique, the eye diagrams provide a visual way of checking it.

A final and much more precise way of determining model accuracy and fidelity is discussed in Chapter 5. This method also represents an appropriate technique to select the optimum values for Volterra model order and memory.

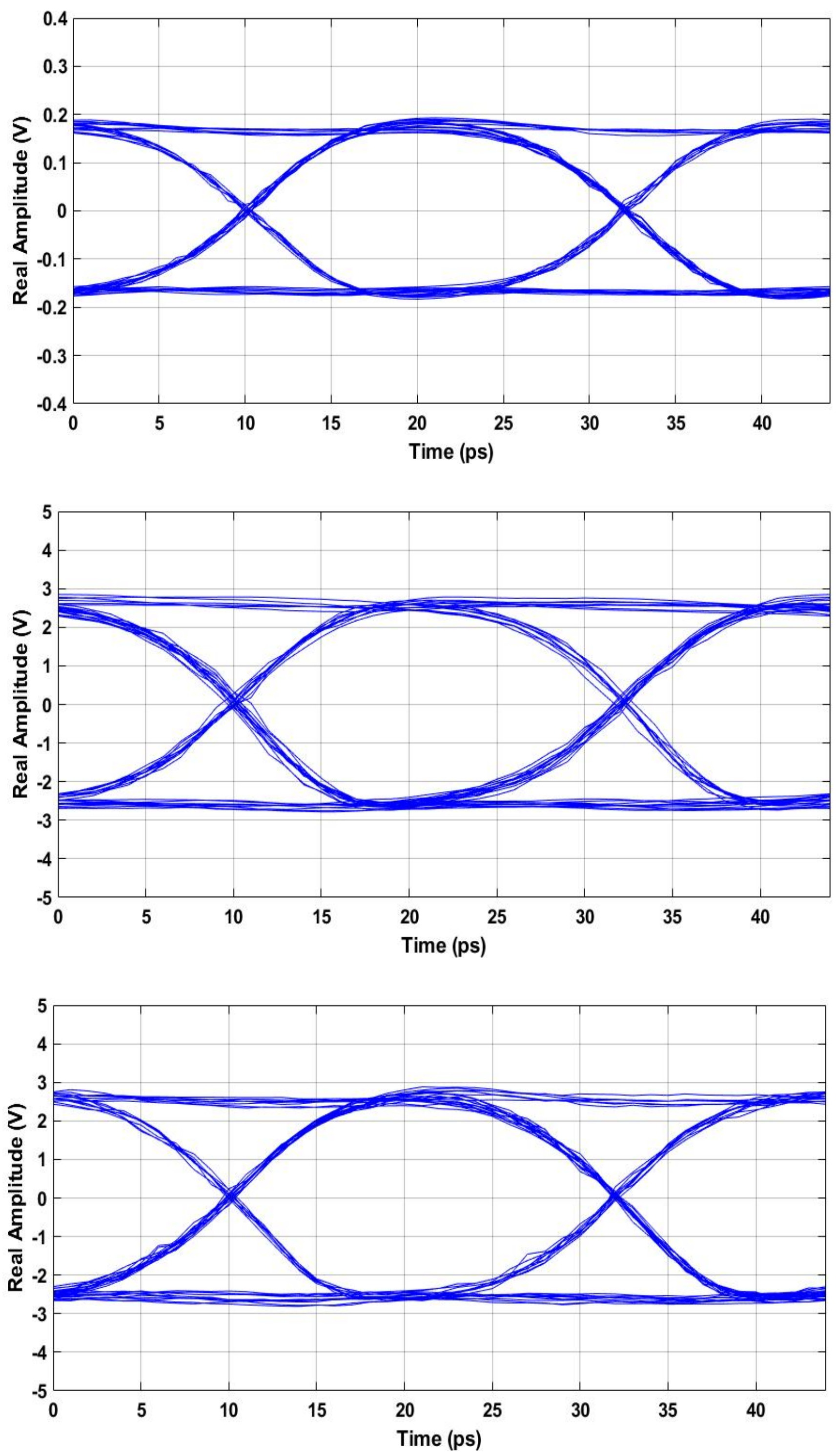


FIGURE 3.10: Eye diagrams corresponding to the input and output signals, as well as the modelled output. The input signal was a 32 Gb/s NRZ signal with a 650 mVpp Voltage and a pattern length of $2^{15} - 1$ bits. The top image shows the eye diagram corresponding to the input signal, the middle image shows the eye diagram of the amplifier output signal, and the bottom image shows the eye diagram corresponding to the modelled output signal.

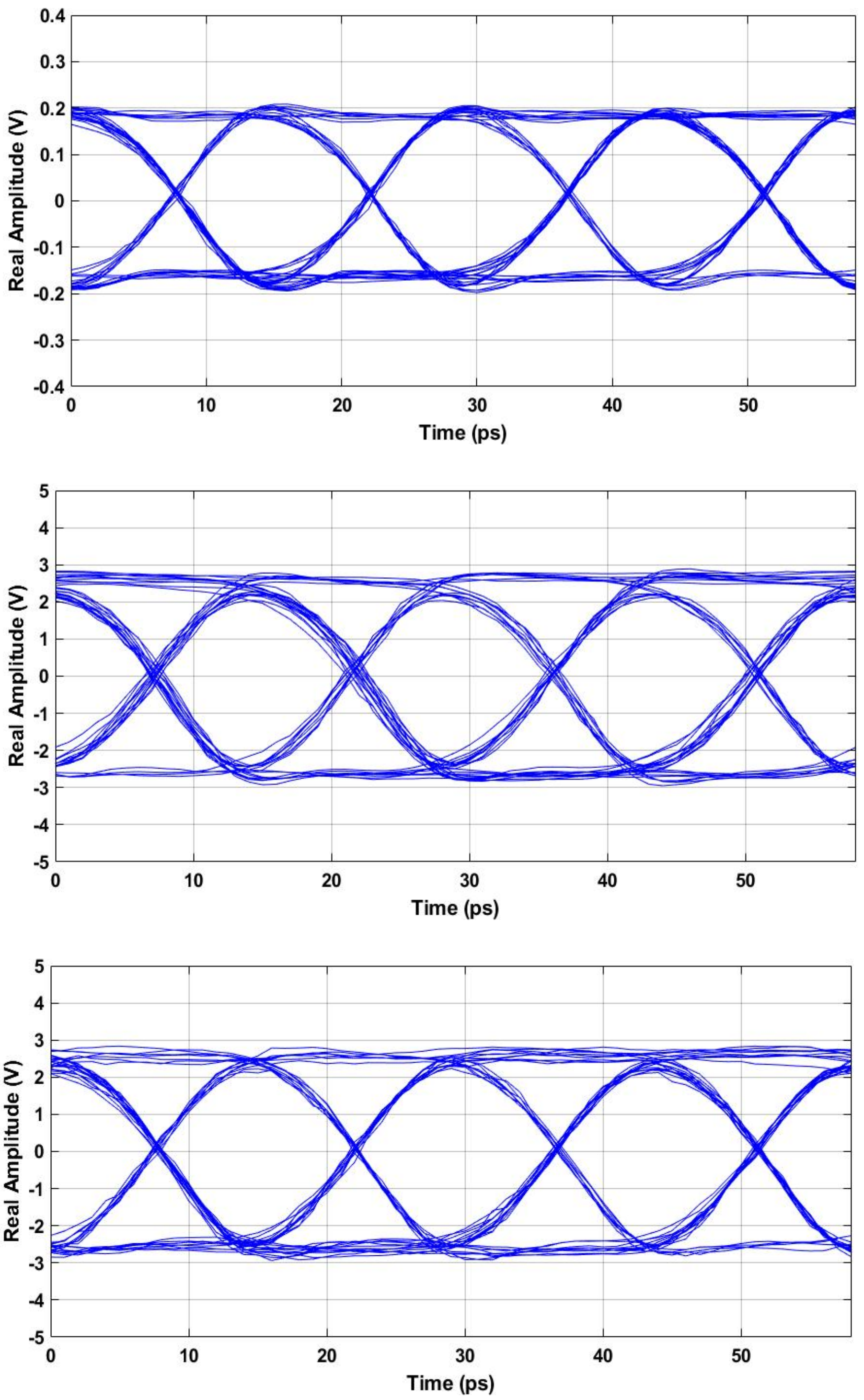


Figure 3.11: Eye diagrams corresponding to the input and output signals, as well as the modelled output. The input signal was a 50 Gb/s NRZ signal with a 650 mVpp Voltage and a pattern length of $2^{15} - 1$ bits. The top image shows the eye diagram corresponding to the input signal, the middle image shows the eye diagram of the amplifier output signal, and the bottom image shows the eye diagram corresponding to the modelled output signal.

3.4 Preliminary Observations

This section summarizes the main issues, ideas, and observations encountered while processing the data obtained from the first lab experiment and the subsequent construction of the Volterra models. The aim is to provide insight on why certain techniques mentioned throughout this chapter were not actually applied and to explain the results obtained from this first set of measurements. These preliminary findings will later be used to determine which measurements to record in the second lab experiment.

3.4.1 Sampling rate, content of recorded waveforms & considerations for window selection

Earlier in this chapter, the concept of using a tapering window to extend shorter length sequences was touched upon. In the end, because direct sequence concatenation could be applied, this tool became unnecessary. The following discussion will explain how this understanding was reached.

The results that have been shown up to this point have all been obtained assuming that the waveforms recorded from the lab measurements represent a single run of the corresponding symbol sequence. For example, this implies that all the recorded input signals of rate 32 Gb/s and pattern length $2^9 - 1$ bits contain $2^9 - 1$ symbols[3]. This can be verified by dividing the waveform length (in samples) of the captured waveform by the equivalent time sampling rate (in samples/symbol), which for a 32 Gb/s signal yields $\frac{11241}{22} \approx 2^9 - 1$. However, if the true sampling rate of the oscilloscope is 22 samples/symbol, it means that it would have been operating at $32 \times 22 = 704$ GSa/s during the experiment, which is a sampling rate well beyond the capabilities of any modern day equipment. Naturally, the oscilloscope was not operating at such a high sampling rate. The reason why we were able to capture that many samples/symbol is because the scope was operated in an equivalent time sampling mode. This way the oscilloscope acquires portions of the waveform during multiple trigger events and is able, over time, to assemble these portions into a complete waveform. The true sampling rate of the oscilloscope remains unchanged and is much lower than 704 GSa/s, which is why in Tables 3.2 and 3.3 the parameter that gives us the samples/symbol has been defined as the "equivalent time" sampling rate.

To further verify that the waveforms recorded with the oscilloscope do indeed represent a single run of the corresponding sequences, their autocorrelation can be computed. This operation is used to determine if the signal is periodic, in which case the recorded

[3]Because of the simplicity of an NRZ modulation scheme, each bit is represented by a single symbol.

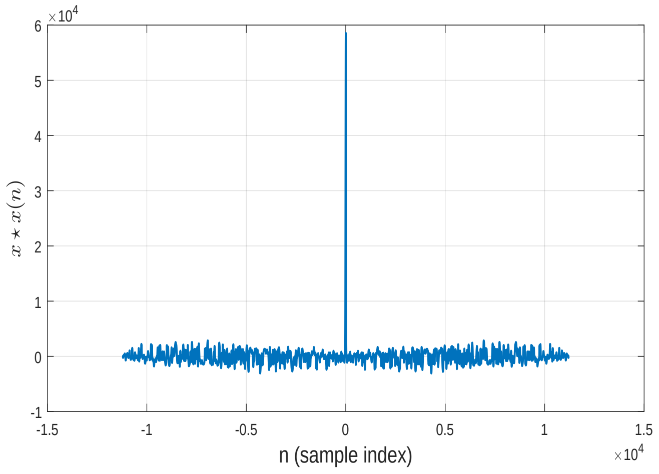


FIGURE 3.12: Autocorrelation of waveform recorded from the oscilloscope. It is a 32 Gb/s, $2^9 - 1$ pattern length, 675 mVpp amplifier output signal.

waveforms will contain multiple runs of the same sequence. The autocorrelation of a specific signal is shown in Figure 3.12, where it is easy to see by its single peak that the waveform is not periodic.

Knowing that the recorded waveforms are not made up of shorter periodic sequences, plotting the beginning and the end of each of the recorded waveforms will serve to finally determine if they contain a single run of the corresponding bit sequence. In fact, these plots are how we reached the conclusion that windowing shorter sequences to soften the effects of their endpoints when concatenating them was actually unnecessary for our purposes. Figure 3.13 shows the end and the beginning of a 32 Gb/s, $2^9 - 1$ pattern length amplifier output signal. It shows how the waveform ends in approximately the same spot where it begins, which coupled with the fact that the waveform is not periodic, justifies the idea that the captured waveforms represent a single run of each sequence. In addition, Figure 3.13 depicts how smooth the transition between two copies of the same signal is when applying direct concatenation.

Despite the fact that a tapering window ended up being unnecessary, it remains valuable to document some of the conclusions that were reached in relation to the topic. In fact, as is shown in the following paragraphs, had we applied the intended windowing and overlap and add combination, the outcome would not have been what we intended.

It was mentioned earlier in this chapter that the window selected for the purposes of tapering must satisfy a condition in the time domain that is similar to the Nyquist zero ISI criterion in the frequency domain. This condition (3.6), which is known as the constant-overlapp-add (COLA) constraint, is satisfied when successively R-time shifted versions of a specific window always add up to one, where R represents the delay of each window impulse response in samples.

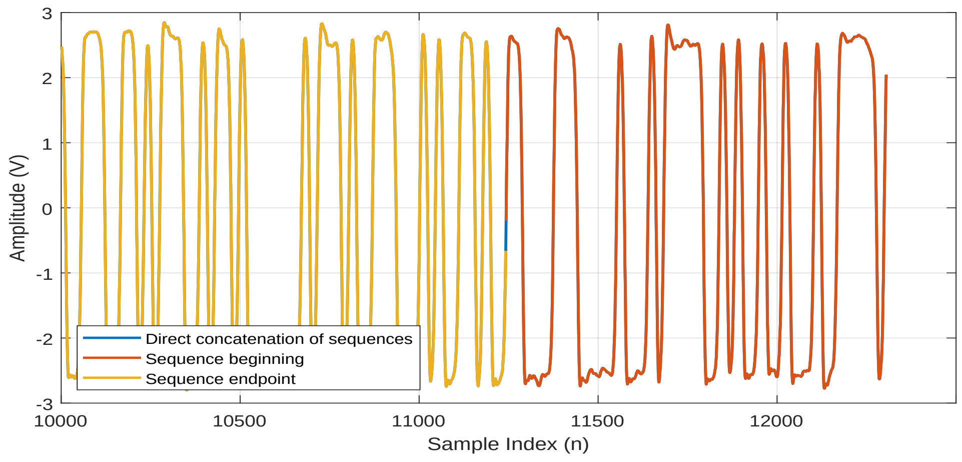


FIGURE 3.13: Direct concatenation of two copies of a waveform recorded from the oscilloscope. It's characteristics are: 32 Gb/s, $2^9 - 1$ pattern length, 675 mVpp amplifier output signal.

$$\sum_{m \in \mathbb{Z}} w(n - mR) = 1, \;\forall n \;\in\; \mathbb{Z}. \tag{3.6}$$

The Nyquist criterion for zero ISI, which states that for a channel impulse response to be free of intersymbol interference, frequency shifted versions of its Fourier transform, $H(f - \frac{k}{T_s})$, must add up to a constant value, is given in (3.7). This serves to showcase its similarity with the COLA constraint.

$$\frac{1}{T_s} \sum_{k=-\infty}^{k=\infty} H(f - \frac{k}{T_s}) = 1, \;\forall f. \tag{3.7}$$

Figure 3.14 shows how the Hanning window of length $L = N + 1$ and delay $R = L/2$ satisfies this condition and how the Tukey (tapered cosine) window with $r = 0.1$ does not, where N is the length, in samples, of the signal to be windowed, and r is the ratio of cosine-tapered section length to the entire window length. Figure 3.15 portrays the effect of applying a Hanning window to taper the endpoint and beginning of a sequence, as well as an example of using said window to perform 50% overlapp-add (50% comes from setting $R = \frac{N+1}{2}$ in (3.6), where $L = N + 1$ is the window length). As can be seen in the bottom image of Figure 3.15, the waveform that results when performing 50% overlapp-add is corrupted and does not resemble the original signal shown in Figure 3.13. Overlap-Add is generally employed to decompose a lengthy input signal into smaller, easier to work with subsets called frames. These frames are processed individually and can then be recombined using the overlap-add procedure to obtain the original signal.

Essentially, overlap-add constitutes a way of avoiding the cumbersome processing operations that large input signals usually entail. The reconstructed signal will be the same as the original because the falling taper and rising taper of the frames that overlap and are then added, correspond to the same portion of the original signal. In our simulation scenario, the falling and rising tapers are not applied to the same signal segments (they are applied to the sequence endpoint and beginning), which results in a corrupted waveform that no longer resembles the original recorded signal. Therefore, even if applied, this "makeshift" overlap-add procedure[4] would not have achieved the desired results of sequence endpoint softening for which it was originally intended.

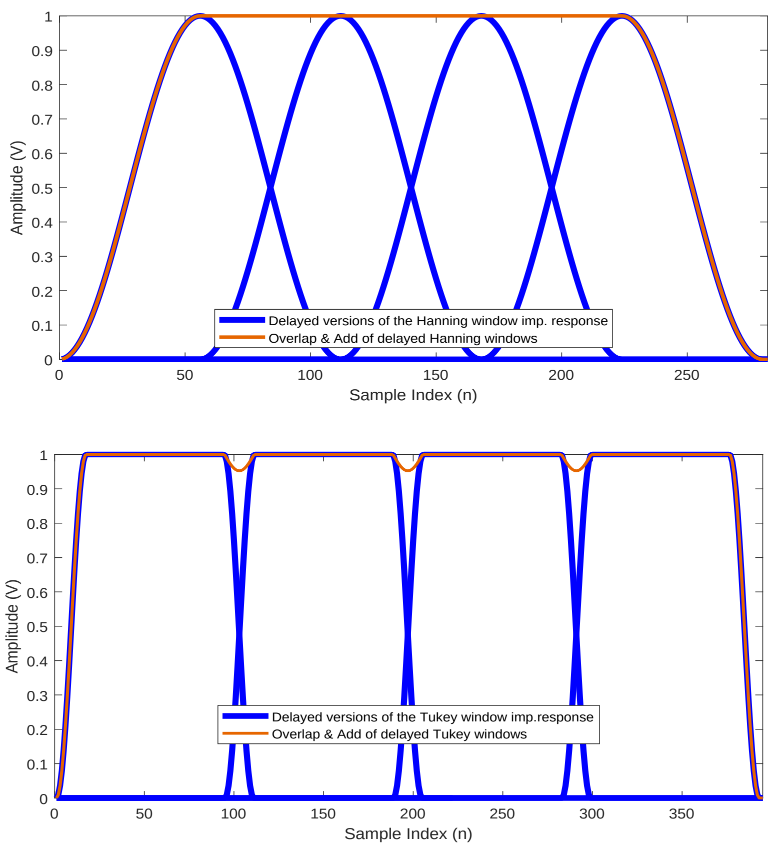


FIGURE 3.14: Verification of the COLA constraint for the Hanning window and the Tukey window. The orange curve is obtained by adding the time shifted versions of the window impulse responses. It shows how the Hanning window satisfies (3.6) and how the Tukey window with $r = 0.1$ does not.

[4]The overlap-add procedure shown in Figure 3.15 is not identical to the widespread overlap-add technique used in signal processing which is why it is referred to as a "makeshift" overlap-add procedure.

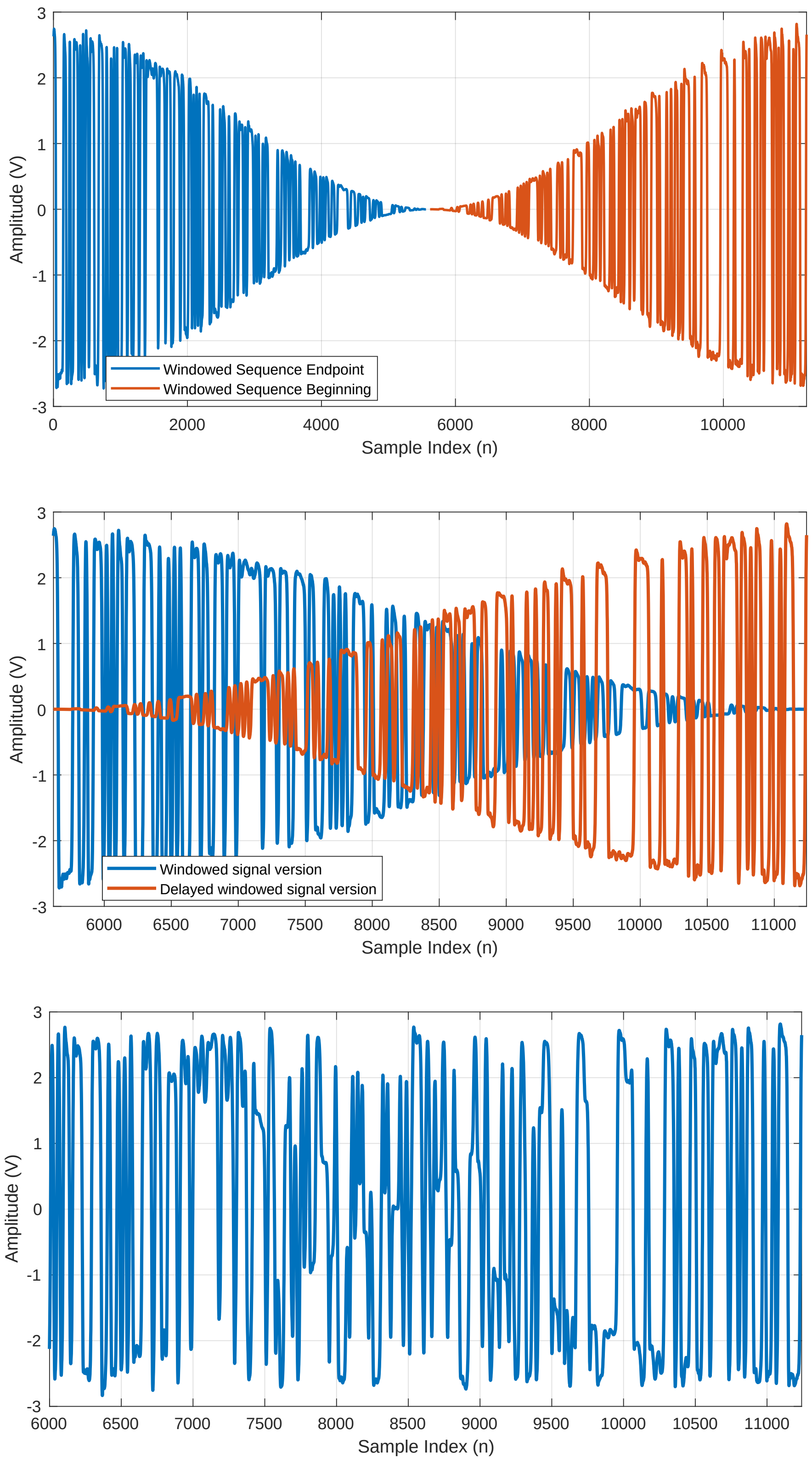


FIGURE 3.15: 50 % Overlapp-add procedure: The top image shows the aftermath of applying a Hanning window to taper the end and beginning of a sequence. The middle image shows the windowed versions of the sequence shown in the top image with a 50 % overlap ($R = \frac{N+1}{2}$ in (3.6)). The bottom image shows the result of adding the overlapped and windowed versions of the sequence shown in the middle image.

3.4.2 Relationship between pattern length & amplifier nonlinearity

It was shown in a prior section of this chapter how the drive amplifier responds slightly differently, in terms of the nonlinear content of the learned models, to signals with different pattern lengths. This was reflected in Table 3.5. A possible explanation for the variation between the normalized nonlinear Volterra kernel weights of these models may be the difference in harmonic content between signals with different pattern lengths.

Let us look at the frequency spectrum of the signals recorded in the first lab experiment. Figures 3.16 and 3.17 portray the signal spectrum of two input signals with pattern lengths of $2^{15} - 1$ and $2^9 - 1$ bits, respectively. As can be seen in these images, the frequency spectrum associated to the longer length sequence appears to be denser and has larger harmonic content than the spectrum of the shorter length signal. This makes it plausible that differences in learned Volterra models for different bit patterns of an input signal stem from variation in the harmonic content of these waveforms. In Chapter 5, when data related to more pattern lengths is available, a much better explanation for the variability of the Volterra models as a function of the input signal spectrum is provided.

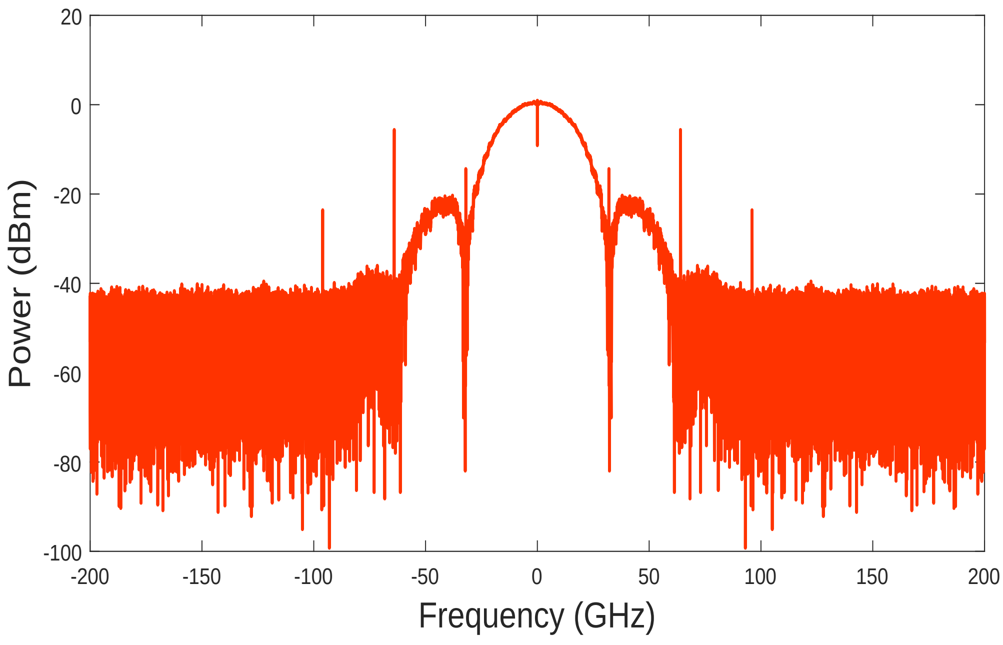


FIGURE 3.16: Power Spectral Density of a 32 Gb/s NRZ input signal with a pattern length of $2^{15} - 1$ bits.

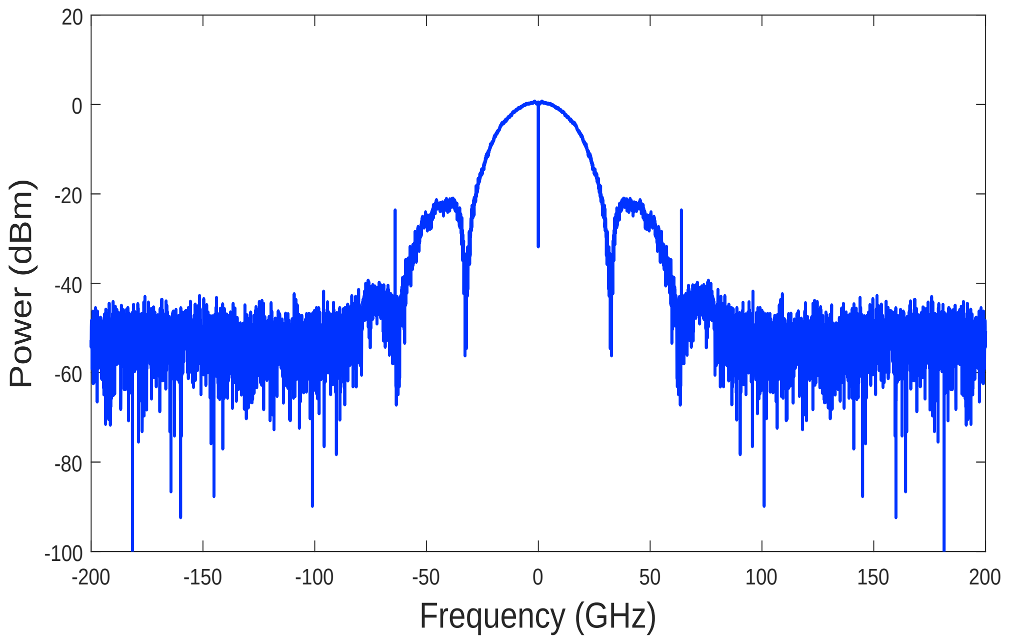


FIGURE 3.17: Power Spectral Density of a 32 Gb/s NRZ input signal with a pattern length of $2^9 - 1$ bits.

3.4.3 Learning models via the VRLS algorithm

Recalling the final section of Chapter 2, we know that the VRLS algorithm can be employed to learn the Volterra models, and that it might represent a superior option to the VLMS algorithm. Thus, in order to determine the optimal algorithm, a practical comparison between the two must be performed.

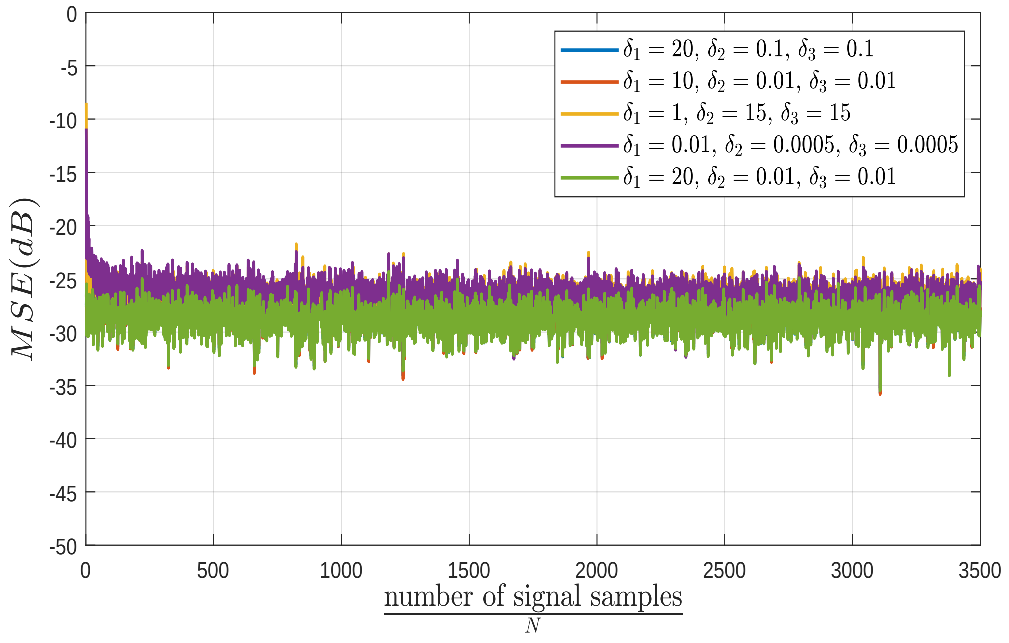
As has been done earlier for the VLMS algorithm, Figure 3.18 portrays the convergence curves of the VRLS algorithm for different values of δ_i. Selecting the combination of δ_i values that produce the lowest MSE curve ($\delta_1 = 20$, $\delta_2 = 0.01$, and $\delta_3 = 0.01$), and choosing $\lambda = 1$ the VRLS algorithm is used to compute the normalized kernel weights for the signals shown in Table 3.5. Table 3.6 compares the normalized kernel weights obtained using the VRLS algorithm with the ones obtained using the VLMS algorithm.

As was foreseen in Chapter 2, both the VLMS and VRLS algorithms yield very similar results and both embody viable adaptive algorithms to learn the Volterra models. In terms of their differences, the VRLS algorithm converges to a steady state MSE value quicker than the VLMS algorithm, but it requires the computation of larger matrices and the definition of more initialization parameters than its counterpart.

Given that there are no significant advantages to using the VRLS algorithm, we will continue to use the VLMS algorithm to learn the Volterra models.

Table 3.6: Normalized Volterra kernel weights for various input signals computed using the VLMS and VRLS algorithms. The superscripts indicate the adaptive algorithm that has been used, where ‘r’ indicates VRLS and ‘l’ indicates VLMS.

Signal	$\tilde{h}_1^{\mathrm{l}}$	$\tilde{h}_2^{\mathrm{l}} + \tilde{h}_3^{\mathrm{l}}$	$\tilde{h}_1^{\mathrm{r}}$	$\tilde{h}_2^{\mathrm{r}} + \tilde{h}_3^{\mathrm{r}}$
32 Gb/s $2^9 - 1$ bits (pattern) 675 mVpp input	0.902	0.098	0.908	0.092
32 Gb/s $2^{15} - 1$ bits (pattern) 675 mVpp input	0.905	0.095	0.891	0.109
50 Gb/s $2^9 - 1$ bits (pattern) 650 mVpp input	0.871	0.129	0.887	0.113
50 Gb/s $2^{15} - 1$ bits (pattern) 650 mVpp input	0.826	0.174	0.831	0.169



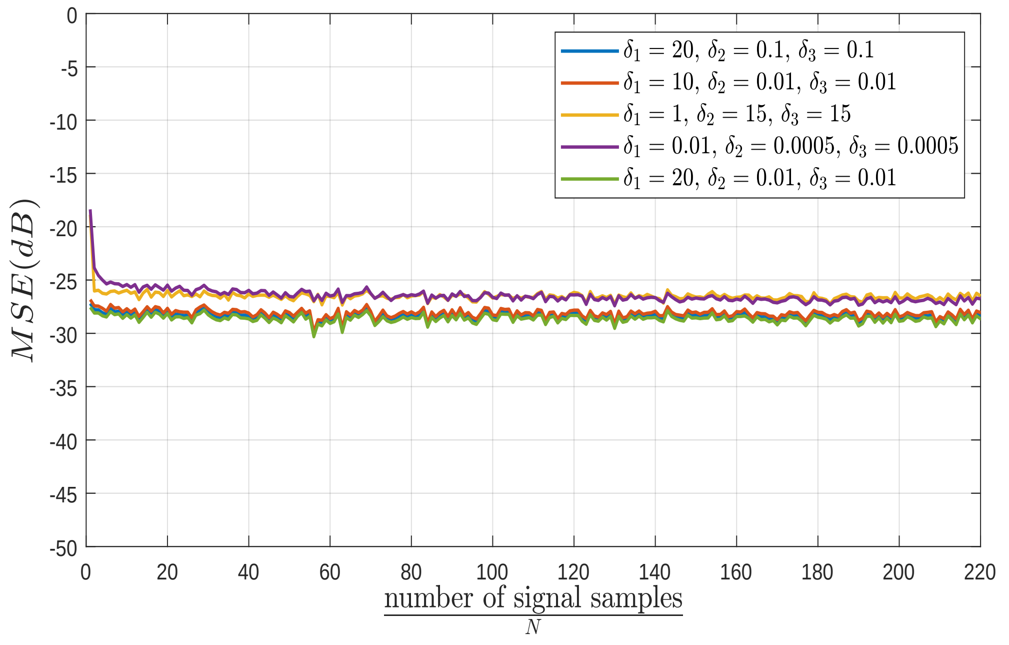


FIGURE 3.18: Evolution of the MSE as a function of the VRLS iterations. The top image has been obtained by using a sliding rectangular window of length $N = 400$. The bottom image has been obtained by using a rectangular window of length $N = 4000$.

Chapter 4

Amplifier model impact on a generic communications system

The primary goal of this thesis is to understand and characterize the nonlinear behaviour of an amplifier based on mathematical models and simulation solutions. Studying the impact amplifier nonlinearities have on a generic communications system will help in achieving this objective. This chapter will explain the integration of the the truncated Volterra amplifier models derived in the previous chapter with a communications system operating in the presence of additive white Gaussian noise (AWGN). The simulation of the entire synthesized system when including models learned in the first lab experiment will also be discussed.

4.1 Inclusion of Volterra models into an AWGN Communications System

4.1.1 Communications System

The system and AWGN channel are constructed within a Matlab environment using a simulation composed of a series of blocks. Each block performs a specific function in the transmission, channel, or reception chain. This is shown in Figure 4.1, which provides a graphical representation of the communications system and the channel. A thorough description of each block is included in the following enumeration.

1. *Block Encoder*: The data is encoded using an LDPC code from the DVBS2 standard. The dimensions of the LDPC parity check matrix (PCM) will vary according

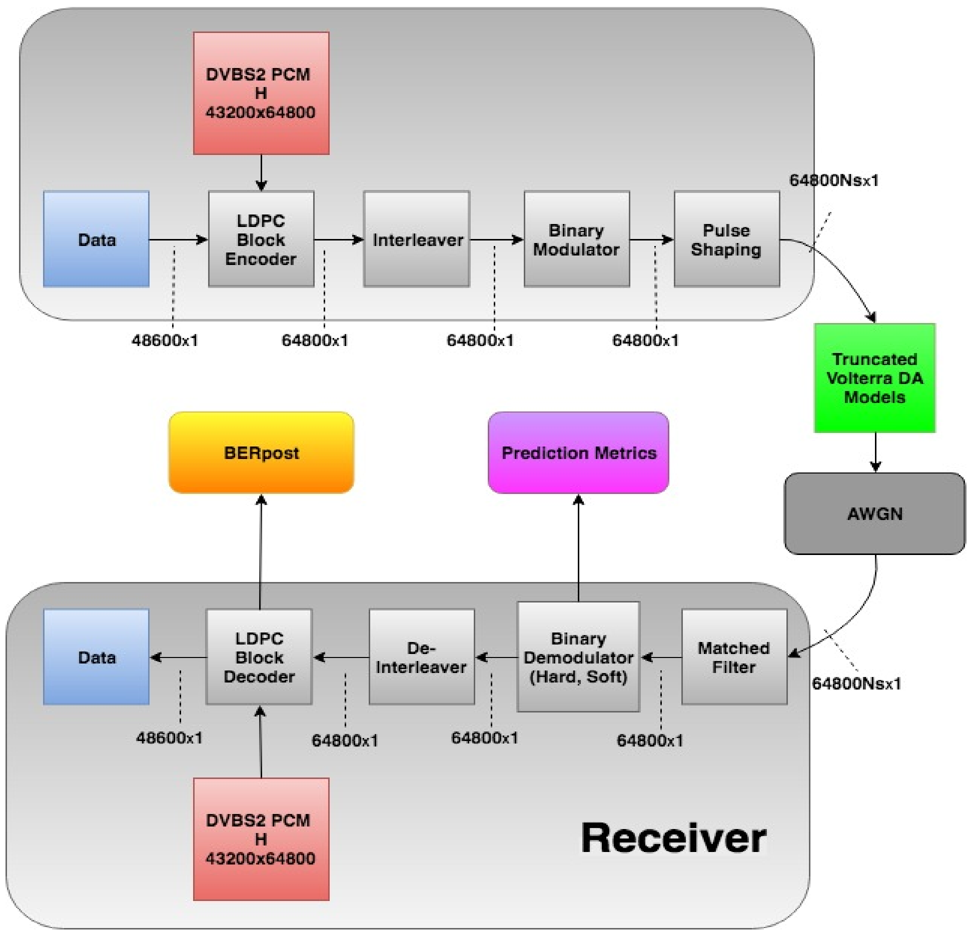


FIGURE 4.1: Representation of the transmission of a single data block in the simulated communications system and channel. Ns represents the number of samples/symbol.

to the chosen code rate. The simulations shown in this letter have been conducted for a code rate of $R_{\text{LDPC}} = \frac{3}{4}$, which results in a PCM, **H**, of size 16200×64800. The message length of the code is obtained by subtracting the columns and rows of **H**, which for $R_{\text{LDPC}} = \frac{3}{4}$ is 48600 bits. In order to obtain reliable simulation results, it is necessary to transmit sequences that are much longer than the code message length. Therefore, the data sequence is fragmented into blocks of 48600 bits which are encoded separately. A random interleaver is introduced at the output of the LDPC block encoder to reduce correlated error sensitivity.

2. *M-QAM Modulator*: Once the data is encoded, the code bits are mapped to the appropriate constellation symbols. Since the amplifier models considered in this letter have been learned from NRZ signals, binary modulation (BPSK/2PAM) is employed.

3. *Pulse Shaping*: In the lab, the signals used to make the amplifier measurements are NRZ signals. In Matlab, we create these NRZ signals by filtering the binary symbols with a filter that has a rectangular impulse response. Although NRZ

signals are oftentimes said to lack pulse shaping, we can consider them as signals that have been pulse shaped using the most basic of pulse shapes: rectangles. The objective of pulse shaping is to limit the effective bandwidth of the signal and to control the channel induced intersymbol interference by changing the shape of the transmitted signals. It should not affect the BER performance of the system (unless ISI appears due to an incorrect combination of pulse shaping and matched filtering). It may be interesting in future research to analyze if using different pulse shaping, like root raised cosine filtering, changes the impact of the Volterra models.

4. *Volterra Model*: To include the effects of the nonlinear device, after pulse shaping, the waveform must be degraded in a manner befitting the drive amplifier. For that purpose, the Volterra kernels learned previously are used to create a filter that will affect the signal in a similar way to the real amplifier. Initially, the idea of normalizing the coefficients of the Volterra model prior to synthesizing the filter was conceived so that the sum of the kernel squares of the model would add up to one.

   This procedure, shown in (4.1), is no longer used, as it was increasing the weight of the linear kernels and diminishing that of the nonlinear kernels, slightly modifying the final simulation results. In (4.1), $\hat{h}_i$ represents the Volterra kernel at position i, $i = 1 \ldots \alpha$, were α represents the total number of kernels of the model (α varies depending on model order and memory).

$$\hat{h}_i = \frac{h_i}{\sqrt{\sum_{j=1}^{\alpha} h_j^2}} \tag{4.1}$$

5. *AWGN Channel*: After applying a specific Volterra model, white Gaussian noise is added to the transmitted signal. Because the symbols of the selected modulation format are real (they only have an in-phase component), the noise added in the AWGN channel will only be included in the in-phase signal component.

6. *Metric computation & Decoding*: Once the waveform has traversed through the Volterra model and the AWGN channel, the final steps in the simulation involve performing matched filtering to correctly recover the corrupted symbols, demodulating these symbols, and then decoding them into codebits. During different stages of this reception process, prediction metrics can be calculated. The system BER is computed at the end of the reception block. This is discussed at length in the rest of this report.

4.1.2 Applied Volterra Models

Simulating the communications system with the Volterra models learned from the first set of lab measurements, especially the computation of the BER after FEC decoding, takes up a considerable amount of time and computing resources. Longer memory and higher order Volterra models increase the simulation requirements. However, although it has not yet been proven, intuition suggests that simpler models will not provide as accurate a portrayal of the amplifier behaviour. This means that there will be a trade-off between model complexity and simulation time. It is thus a reasonable endeavour to relearn different Volterra models and look at their accuracy and impact on the simulations. Despite the fact that a third order $M = 5$ model has already been learned and analyzed in Chapter 3, lower order and shorter memory models cannot be constructed by simply selecting a subset of the kernels of this model. Due to the nature of the Volterra series and the VLMS algorithm, the coefficient values will vary whenever P (order) or M (memory) is changed, which means that every time a model with a different P or M needs to be tested, it will have to be relearned by means of the adaptive algorithm for the corresponding values of P and M.

The memory and order of the Volterra models stand out among a growing number of parameters that play important roles in the BER simulations that are shown later in this thesis. It is paramount in our endeavor of building an accurate behavioural amplifier model to determine the relationship of these simulations with the aforementioned parameters. Therefore, achieving a complete understanding of the BER results warrants the relearning of Volterra models with the goal of asserting how the variation of P and M affects the final results.

Lower memory and lower order Volterra models lack terms in the Volterra series to reflect the complete nonlinear behaviour of the amplifier. This results in a more linear than realistic portrayal of its behaviour, and so the BER simulation results will appear "better" (less degraded), in terms of performance, than they really are. Models with higher values of M and P include more terms of the Volterra series, resulting in larger nonlinear kernel weights and a more accurate approximation of the amplifiers real-world behaviour. Because these models contain more nonlinear terms, the expectation is that the resulting AWGN simulations will be more degraded than for lower M and P models.

Another phenomenon worthy of inspection is simulation result behaviour as a function of VLMS step sizes. In terms of the effects caused by changing these step sizes, we know that variations between models learned with different step-sizes appear in the values of their Volterra kernels. As it only makes sense to select those step-sizes that guarantee algorithm convergence, and the difference between kernel values is negligible when said step-sizes are chosen, simulating models of same M and P but learned with different VLMS step-sizes should result in almost identical outcomes.

The simulation results shown in the following chapter, which will be used to verify the predictions that have just been made, have been obtained using a set of models with different order and memory. The VLMS algorithm step-sizes used to learn these models are $\mu_1 = 0.01$, $\mu_2 = 0.001$, and $\mu_3 = 0.001$ (in the case that $P = 3$). The normalization variables calculated as shown in 3.1 are also included in Table 4.1. In addition, Table 4.2 provides the defining characteristics of the learned models, where $\tilde{h}_i$ for $i = 1, 2, 3$ is computed as shown in (3.5), and α represents the total number of kernels of each model. The normalized kernel weights, $\tilde{h}_i$, are used to measure the linear and nonlinear content of the models. They provide a way of determining how linear/nonlinear the learned model is. The series expansions of these models can be obtained from (2.2) by setting the appropriate value of M, and disregarding or including the third order terms depending on the order of the model in question.

TABLE 4.1: Values of the normalization parameters G_x, S_x, G_d, S_d, where $x(n)$ and $d(n)$, are the input and output signals explained in Chapter 3.

Model	S_x	G_x	S_d	G_d
32 Gb/s $2^{15} - 1$ bits 675 mVpp	0.0047	0.1907	0.0245	2.8979
50 Gb/s $2^{15} - 1$bits 650 mVpp	-0.0091	0.2063	-0.0247	2.9819

4.2 Simulation of the complete nonlinear AWGN system

Having explained how the Volterra models that have been learned are integrated with the AWGN communications system, we are now in a position to discuss how relevant information can be obtained from this setup. This section explains the different types of simulations that the complete system can be put through to quantify the effects caused by the addition of the nonlinear Volterra models to the AWGN communications system. How the inclusion of these models reflects on the performance of the LDPC decoder shown in Figure 4.1 is analyzed. Moreover, a prediction metric is proposed as a viable way of studying the impact these nonlinearities have. Finally, a method to select Volterra model memory and order, and to determine the models' overall accuracy is proposed.

TABLE 4.2: Characteristics of the models learned from the first set of lab measurements.

Model	P	M	α	$\tilde{h}_1$	$\tilde{h}_2$	$\tilde{h}_3$
32 Gb/s $2^{15}-1$ bits 675 mVpp	2	12	65	0.9910	0.0090	-
32 Gb/s $2^{15}-1$ bits 675 mVpp	3	3	19	0.9706	0.0081	0.0213
32 Gb/s $2^{15}-1$ bits 675 mVpp	3	4	34	0.9595	0.0089	0.0316
32 Gb/s $2^{15}-1$ bits 650 mVpp	3	5	55	0.9501	0.0040	0.0459
50 Gb/s $2^{15}-1$ bits 650 mVpp	2	12	65	0.9947	0.0053	-
50 Gb/s $2^{15}-1$ bits 650 mVpp	3	3	19	0.9237	0.0027	0.0736
50 Gb/s $2^{15}-1$ bits 650 mVpp	3	4	34	0.9068	0.0012	0.0.092
50 Gb/s $2^{15}-1$ bits 650 mVpp	3	5	55	0.8998	0.0052	0.0950

4.2.1 Theoretical baselines & the post-FEC BER

An analysis of the effects caused by adding the Volterra models to the communications system will require the study of a variety of different scenarios. A good method to begin with is to draw comparisons and seek similarities between established theoretical baselines, linear AWGN waveforms[1], and transmissions affected by both the nonlinear Volterra models and AWGN. In terms of theoretical baselines for systems that use FEC, the Shannon limit is extensively used. It provides the SNR value for a specific spectral efficiency, denoted by ρ[2] and measured in bits per 2 real dimensions ($\frac{\text{bits}}{2D}$), below which reliable communication is not possible. In the Matlab simulations, depending on the value selected for R_{LDPC}, both the system spectral efficiency, ρ_s, and the specific theoretical limit for that spectral efficiency will vary. When the spectral efficiency approaches zero (low rate LDPC codes), a final limit, known as the ultimate Shannon limit, can be obtained. The Shannon limit for a given ρ is shown in (4.2), where $\frac{E_b}{N_0}$ is known as the

[1]In this context, the term linear implies that the signal has not been subjected to amplifier nonlinearities during the simulation. This is accomplished by applying a Volterra model in which only the first order kernels are present, the higher order kernels having been zeroed out. In other words, the signal goes through a Volterra model in which the nonlinear components have been 'shut off'.

[2]Throughout this letter, the spectral efficiency of an arbitrary communications system will be denoted by ρ. The subscripted version of this notation, ρ_s, is used to refer to the spectral efficiency of the system considered in the Matlab simulations.

SNR per bit and is the de facto SNR measure in channel coding systems where $\rho \leq 2$. $\frac{E_b}{N_0}$ is a normalized version of the conventional SNR[3].

$$\frac{E_b}{N_0} > \frac{2^\rho - 1}{\rho} \tag{4.2}$$

The ultimate Shannon limit is obtained from (4.2) as shown in (4.3), where we have used the fact that $\lim_{\rho \to 0} 2^\rho - 1 \approx \rho \ln 2$. From this point onwards in this letter, in accordance with channel coding notation used for systems with $\rho \leq 2$, the SNR per bit, $\frac{E_b}{N_0}$, will be employed.

$$\frac{E_b}{N_0}^{\text{ult}} > \lim_{\rho \to 0} \frac{2^\rho - 1}{\rho} \approx \ln 2 \approx 0.69 \ (-1.59\text{dB}). \tag{4.3}$$

The Shannon limit provides a way to measure the quality of FEC codes by defining the theoretically lowest attainable SNR that guarantees error-free communication for a specific value of ρ. Another comparison possibility is to look at the performance of an uncoded system (one that does not use FEC) with a specific ρ and compare it to our system with FEC. This will allow us to quantify how much is gained by introducing FEC in the communications system. Because the simulations considered in this letter use binary modulation, which without FEC coding and pulse shaped with Nyquist pulses has $\rho = 2$, we will take the bit error probability of binary pulse modulation (2-PAM) for baseband channels or 2×2-QAM for passband channels as our uncoded baseline.

The theoretical expression for the BER of an uncoded system with $\rho = 2$ working in the presence of AWGN, which we will denote by $P_{b,2}$, is given in [13] and shown in (4.4), where $Q(x)$[4] is the Q-function and erfc() is the complementary error function.

$$P_{b,2} = Q\left(\sqrt{2\frac{E_b}{N_0}}\right) = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right). \tag{4.4}$$

By evaluating the performance of the uncoded modulation scheme for $\rho = 2$ and comparing it to the Shannon limit, it is possible to determine how much is gained by using FEC codes. The coding gain, defined as the difference in SNR values for a given error probability between the uncoded BER and the FEC BER, is a measure of this improvement. The spectral efficiency of the communications system that is simulated in this letter is given by $\rho_s = 2R_{\text{LDPC}}[\frac{\text{bits}}{2D}]$, where R_{LDPC} is the rate of the selected LDPC code. Since $R_{\text{LDPC}} < 1$, the gain obtained by using these LDPC codes must be compared to the uncoded performance for $\rho = 2$ relative to their respective Shannon limits. This

[3]The relationship between $\frac{E_b}{N_0}$ and the SNR is given by $\text{SNR} = \rho\frac{E_b}{N_0}$.

[4]The Q-function represents the probability that a Gaussian random variable will take a value larger than x standard deviations. It is defined as $Q(x) = \frac{1}{2\pi}\int_x^\infty \exp(-\frac{u^2}{2})du$. It can be expressed in terms of the complementary error function as $Q(x) = \frac{1}{2}\text{erfc}(\frac{x}{\sqrt{2}})$.

way, numerical comparisons between the simulation results, uncoded performance and the Shannon limits will be possible, potentially enabling the study of how the Volterra models affect the coding gain of the FEC scheme. For contextual purposes, although a coding gain comparison is not conducted in the text, the results shown in this section will include the Shannon limit for the spectral efficiency of the system ρ_s, the ultimate Shannon limit, and the uncoded BER baseline for $\rho = 2\ \frac{\text{bits}}{2D}$ given in (4.4).

To be able to compare the aforementioned theoretical values to the simulated transmissions, a measure of their system BER is required. Such a measure can be obtained by computing the bit error rate after FEC decoding, known as the post-FEC decoding BER, and denoted by BER_{post}. A theoretical expression for BER_{post} is provided in [14]. In the simulations, BER_{post} is obtained by computing the number of decoded bits in error for each transmitted waveform. This is performed by comparing the decoded data block with the corresponding original transmitted data block. Since these simulations do not include outer FEC coding and since the comparison of data blocks implies a comparison between information bits, hard decision detection is performed on the soft outputs produced by the SD-FEC LDPC decoder. This is shown in Figure 4.2.

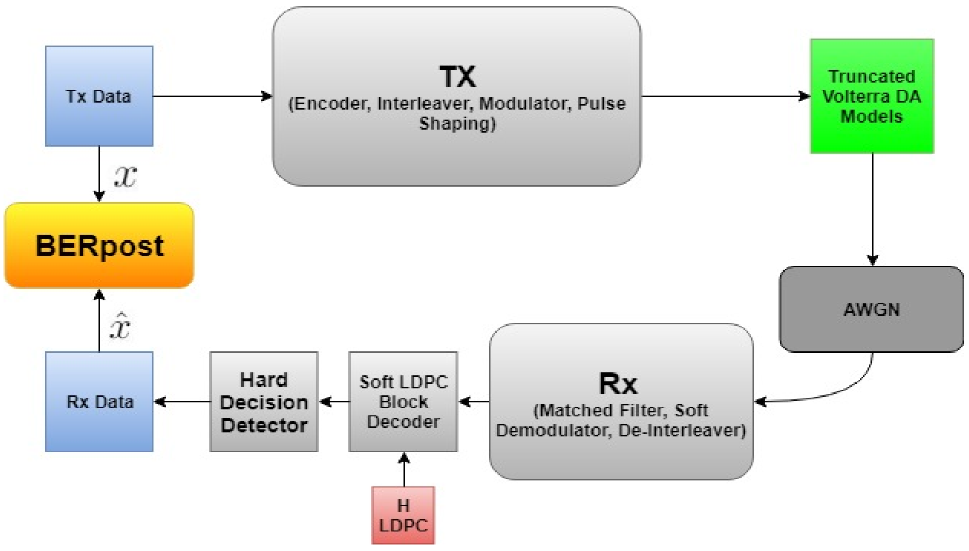

Tx Data
TX
(Encoder, Interleaver, Modulator, Pulse Shaping)
Truncated Volterra DA Models
x
BERpost
AWGN
$\hat{x}$
Rx
(Matched Filter, Soft Demodulator, De-Interleaver)
Rx Data
Hard Decision Detector
Soft LDPC Block Decoder
H LDPC

FIGURE 4.2: Computation of BER_{post}.

An important factor to note when computing BER_{post} is the maximum number of iterations the LDPC decoder is allowed to perform before producing data bit estimates. The amount of bit errors the decoder makes will be conditioned by the number of decoding iterations. To study the relationship between this variable and the simulation results, the decoder is run for two separate maximum numbers of decoding iterations; 50 and 100. Looking at this factor might also be useful to determine if an increase in decoding iterations mitigates the impact caused by the Volterra models[5].

[5]It is well known that up to a certain number of iterations, increasing the time the decoder is allowed to run results in improved bit estimates. In turn, this will lead to lower BER_{post} results.

4.2.1.1 Post-FEC BER: Expected results

Before looking at the outcome of the simulations, predictions on what these results will look like can be made according to the theory derived previously in this section.

The Matlab framework tests models learned for two different data rates: 32 and 50 Gb/s. Based on the definition of the spectral efficiency given earlier, because the modulation scheme remains the same for both data rates, their spectral efficiency will be identical, being given by $\rho_s = 2R_{\text{LDPC}}$. However, it seems reasonable that an increase in signal data rate from 32 to 50 Gb/s should be accounted for somewhere in the theoretical calculations. We have defined our SNR measure as the SNR per bit, $\frac{E_b}{N_0}$. The energy per symbol, which is identical to the energy per bit in our case, can be expressed as $E_b = PT_s$, where P is the output power of the system and T_s is the symbol period of our rectangular NRZ pulses. As shown in [15], we can re-express the SNR as a quotient of powers by multiplying both the numerator and denominator of $\frac{E_b}{N_0}$ by $\frac{1}{T_s}$. This is given by

$$SNR = \frac{E_b}{N_0} = \frac{P}{N_0 W},$$

where $N_0 W$ is the noise power and $W = \frac{1}{T_s}$ is the bandwidth of the noise. It is now easy to see how an increase in signal data rate will not be reflected in the value of the SNR per bit. Increasing the data rate implies a shorter symbol period T_s which means an increase in W. As a result, the ratio $\frac{P}{N_0 W} = \frac{E_b}{N_0}$ will remain unchanged. Therefore, the expected outcome of the simulations is that linear AWGN transmissions (model nonlinearities are turned off) at 32 Gb/s and 50 Gb/s produce the same post-FEC decoding curves. In other words, that their bit error performance will be identical.

With regard to the nonlinear models, the expectation is that at 32 Gb/s, the applied Volterra models cause a degradation in performance such that higher $\frac{E_b}{N_0}$ values than in the linear scenario are required for the same BER_{post} values. This effect should not be significant, as all of the 32 Gb/s models shown in Table 4.2 have near negligible nonlinear kernel weights. At 50 Gb/s, a heavier performance loss than at 32 Gb/s is expected for the full nonlinear model. The deterioration of the BER curves should still remain relatively modest, as the nonlinear weights of the 50 Gb/s models shown in Table 4.2 are a lot smaller than the linear kernel weights. The fact that the performance loss should not be substantial is further supported by the eye diagrams shown in Figures 3.10 and 3.11, which show how there is an increase in nonlinear behaviour when going from 32 to 50 Gb/s transmissions but that the resulting eye pattern is not overly corrupted.

4.2.1.2 Post-FEC BER: Simulation results

The results analyzed in this subsection have been obtained by running a simulation that computes BER_{post} as a function of the SNR per bit for a variety of transmission scenarios: linear AWGN signals (Volterra model without nonlinearities) at 32 and 50 Gb/s, and nonlinear AWGN signals (full Volterra model) at 32 and 50 Gb/s. The rate of the LDPC code was set at $R_{\text{LDPC}} = \frac{3}{4}$. The Volterra model was a $P = 3$ $M = 5$ model. Its characteristics are shown in Table 4.2. Figure 4.3 portrays the post-FEC BER of these transmission configurations. Each simulated transmission results in two BER_{post} curves, where one denotes the result obtained when the LDPC decoder is run for 50 iterations and the other one denotes the result for 100 iterations. Table 4.3 is provided to numerically ground the differences between the curves of Figure 4.3. The parameter d_{nl} denotes the distance between the linear AWGN curves and the nonlinear curves at a specific data rate, in dB. It is obtained by subtracting $\frac{E_b}{N_0}(\text{NL-AWGN}) - \frac{E_b}{N_0}(\text{LN-AWGN})$ when $\text{BER}^{50}_{\text{post}}(\text{NL-AWGN}) = \text{BER}^{50}_{\text{post}}(\text{LN-AWGN}) = 10^{-2}$ for the 32 or 50 Gb/s curves.

TABLE 4.3: Data obtained from Figure 4.3. The signal type field identifies the curves shown in the figure. LN stands for linear and NL stands for nonlinear. The nonlinearities are simulated using the $P = 3$ and $M = 3$ models shown in Table 4.2.

Signal Type	Signal Rate (Gb/s)	$\frac{E_b}{N_0}$ (dB)	$\text{BER}^{50}_{\text{post}}$ (dB)	$\text{BER}^{100}_{\text{post}}$ (dB)	d_{nl} (dB)
LN-AWGN	32	0.8159	0.0100	0.0085	-
LN-AWGN	50	0.8201	0.0100	0.0092	-
NL-AWGN	32	1.3200	0.0100	0.0098	0.5041
NL-AWGN	50	1.8110	0.0100	0.0083	0.9909

Prior to discussing the results portrayed by Figure 4.3, it is important to restate how the 32 Gb/s and 50 Gb/s linear AWGN curves are obtained. These BER_{post} curves are computed from signals that have gone through the corresponding $P = 3$ $M = 5$ linear version of the nonlinear model from Table 4.2. The linear version of the nonlinear model is obtained by simply "shutting off" (zeroing out) its nonlinear components. In other words, the NRZ signal is solely subjected to the first order kernels (the higher order terms have been zeroed out) of the respective Volterra models prior to the addition of AWGN.

If we now analyze the curves shown in Figure 4.3, we will be able to verify if they match the expected outcomes mentioned in the previous subsection. Based on the definition of the spectral efficiency in bits per two real dimensions, one of the theoretical predictions was that the linear AWGN transmissions at 32 and 50 Gb/s should result in identical bit error performance. This can be seen in Figure 4.3, which shows how the linear

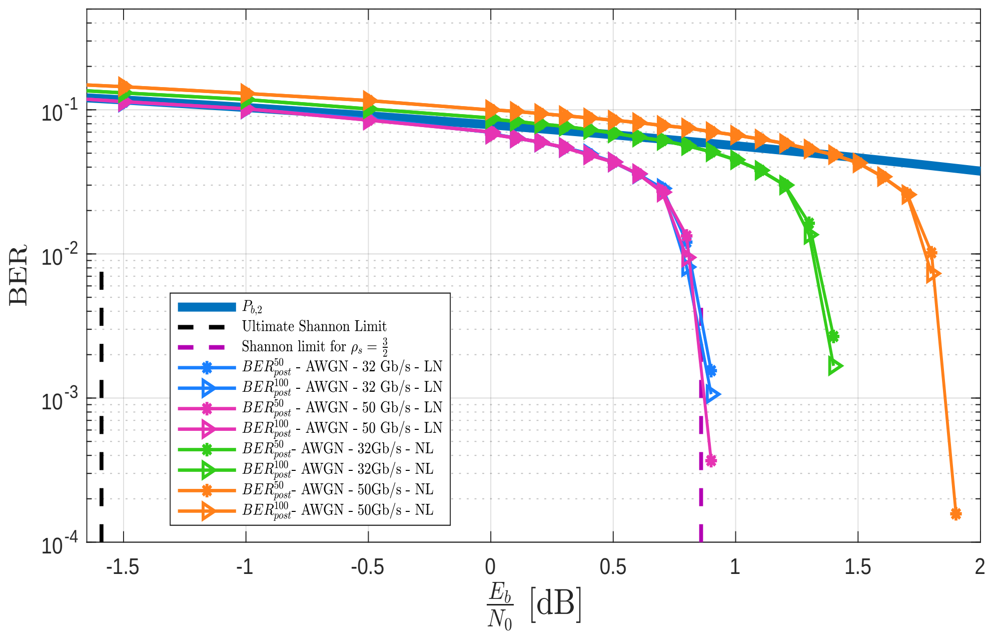


FIGURE 4.3: BER_{post} curves for various transmitted signals and different LDPC decoder iterations. The superscripts of the BER_{post} entries represent the number of LDPC decoding iterations that were used. The Volterra model used to mimic the amplifier nonlinearities is a third order $M = 5$ model. The acronyms LN and NL represent the words "linear" and "nonlinear", respectively.

BER_{post} curves for 32 and 50 Gb/s and for both decoder iteration scenarios are essentially identical (the slight variation is attributed to simulation conditions). Moreover, Figure 4.3 also portrays how close the FEC system with $\rho_s = \frac{3}{2}$ is to the Shannon limit for that spectral efficiency (0.859dB).

Another theoretical assumption was that inclusion of the full nonlinear models would result in a worsened bit error performance, and that this degradation would be higher at 50 Gb/s than at 32 Gb/s. This bit error performance deterioration can be assessed by comparing the BER_{post} curves for the nonlinear Volterra models at a specific data rate with their corresponding linear AWGN curves. We quantify the assessment based on the parameter d_{nl} given in Table 4.3, which measures the separation at a specific BER value between the linear AWGN curve at a given data rate and its nonlinear counterpart at that same rate. At a BER_{post} value of 10^{-2}, for 32 Gb/s, $d_{\text{nl}} = 0.5041$ dB. At that same BER_{post} value, $d_{\text{nl}} = 0.9909$ at 50 Gb/s. This means that, in the linear regime of the amplifier (data rate of 32 Gb/s), the nonlinear components of the model cause an SNR loss of about 0.5 dB. At 50 Gb/s, the SNR loss due to inclusion of the nonlinearities doubles, embodied by $d_{\text{nl}} \approx 1$ dB. This verifies our previous theoretical assumption. Although the SNR loss values of 0.5 dB at 32 Gb/s and 1 dB at 50 Gb/s seem to be relatively small values, data for different transmissions will be necessary to determine just how significant these losses are.

We can also determine the effectiveness of increasing the LDPC decoder iterations as a mitigation measure for the nonlinearities of the Volterra models by observing the $\mathrm{BER}_{\mathrm{post}}^{100}$ curves of Figure 4.3. It seems quite clear from the simulation results that increasing the decoder iterations does not lead to any improvements of relevance in terms of the bit error performance. Despite the fact that there is a very slight enhancement ($\mathrm{BER}_{\mathrm{post}}^{100}$ decreases more rapidly than $\mathrm{BER}_{\mathrm{post}}^{50}$), it does not appear to be a plausible way to bridge the gap between the linear AWGN curves and the respective nonlinear 32 Gb/s and 50 Gb/s curves.

Finally, upon further inspection of Figure 4.3, one may notice that the $\mathrm{BER}_{\mathrm{post}}$ curves are cut short after 10^{-4}. This phenomenon is caused because of the fact that the number of transmitted bits in the simulation is not sufficiently large. The simulation conducted to obtain Figure 4.3 considered the transmission of 100 data frames of 48600 bits, resulting in a total of 4860000 data bits and 6480000 code bits. Increasing the accuracy with which $\mathrm{BER}_{\mathrm{post}}$ values can be calculated, in other words, computing values below 10^{-4} for the curves shown in Figure 4.3, would require simulating the transmission of an even larger quantity of data frames. In reality, $\mathrm{BER}_{\mathrm{post}}$ values should be as low as 10^{-12} or 10^{-15}. Since these values cannot be reliably estimated using Monte-Carlo simulations, a different strategy will be required. The conventional design method adopted as the substitute of $\mathrm{BER}_{\mathrm{post}}$ computation involves simulating the system without FEC coding, optimizing it for a much higher BER value, and assuming that said BER value can be reduced to the desired $\mathrm{BER}_{\mathrm{post}}$ values. This is discussed in the following subsection. Nevertheless, the results shown in Figure 4.3 are useful to analyze the relationships between the simulated transmissions, theoretical results, and other factors, such as the impact of the maximum number of decoding iterations.

4.2.2 The pre-FEC BER

The computation of reliable and low enough values of $\mathrm{BER}_{\mathrm{post}}$ by means of Monte-Carlo simulations is impractical because of the heavy time and computational costs they entail. The complexity of calculating the post-FEC BER by means of these simulations can be circumvented by computing a set of prediction metrics that will also enable us to study the impact of the nonlinear Volterra models on the system.

The most widely employed prediction metric is the pre-FEC decoding BER, denoted by $\mathrm{BER}_{\mathrm{pre}}$. It is used as a standard performance measure for uncoded systems. As is mentioned in the previous subsection, the design strategy revolves around the assumption that below a certain sufficiently small value of $\mathrm{BER}_{\mathrm{pre}}$, subsequent values of this uncoded BER are guaranteed to be reducible to the desired $\mathrm{BER}_{\mathrm{post}}$ values via previously verified FEC implementations. This assumption is known as the *FEC limit paradigm*, and it has been extensively applied in the field of optical communications. The validity of this paradigm is discussed in Chapter 6.

In [14], a mathematical expression for BER_{pre} is given. In our Matlab simulations, we compute the pre-FEC BER by comparing received and transmitted codebits instead. The method is analogous to the computation of BER_{post}, albeit the comparison is performed between code bits and not information bits. Figure 4.4 provides a visual representation of how BER_{pre} is calculated in the simulation. It requires the inclusion of a hard decision detector at the output of the soft binary demodulator. This HD detector will minimize BER_{pre} by making hard decisions on the *a posteriori* logarithmic likelihood ratios computed by the soft demodulator. These ratios are given by (4.5), where $k = 1, \ldots, n$, n is the total number of transmitted code bits, and $P_{C_k|Y_k}(1|y_k)$ and $P_{C_k|Y_k}(0|y_k)$ are the probabilities that $C_k = 1$ or $C_k = 0$ given that y_k was received, respectively.

$$L_k^{\text{apo}} = \log \frac{P_{C_k|Y_k}(1|y_k)}{P_{C_k|Y_k}(0|y_k)} \tag{4.5}$$

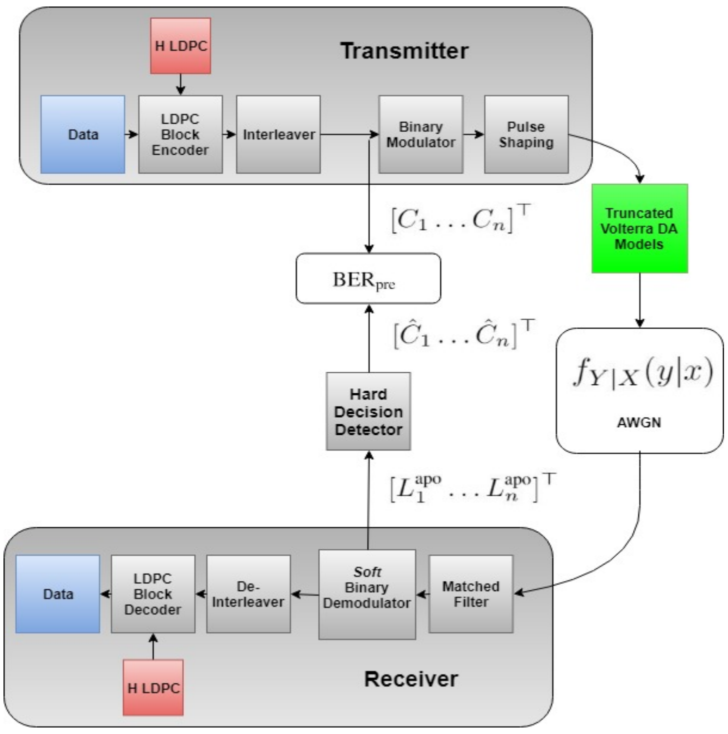


FIGURE 4.4: Computation of BER_{pre}.

We can now simulate the same transmission scenarios considered to obtain the BER_{post} curves shown in Figure 4.3, but computing BER_{pre} instead. The computation of BER_{pre} will be identical to that of BER_{post}, with the difference being that we will calculate code bits in error instead of decoded bits in error. As previously, predictions on what the results should look like can be made based on theory. We predict, as we did for BER_{post}, that the linear AWGN 32 Gb/s and 50 Gb/s transmissions should produce identical BER_{pre} curves. In addition, since FEC coding is no longer being used, and given that $\rho_s = 2$, these linear AWGN curves should be very close to the theoretical uncoded limit $P_{b,2}$.

As in the case with FEC coding, the nonlinear Volterra models for a specific data rate should cause a deviation (increase in required $\frac{E_b}{N_0}$ for same BER_{pre}) from the corresponding linear AWGN BER_{pre} curve at that data rate. The deviation caused by the nonlinear model at 50 Gb/s should be more severe than for 32 Gb/s.

Figure 4.5 shows the result of computing BER_{pre} for signals that have traversed through linear and nonlinear models for 32 and 50 Gb/s (extracted from measurements conducted in the first lab experiment) and the AWGN channel. As was foreseen, the BER_{pre} curves exhibit the same behaviour shown by the BER_{post} results of Figure 4.3. The 32 Gb/s and 50 Gb/s linear AWGN BER_{pre} curves are almost identical, being slightly above the theoretical BER limit. The nonlinear model at 32 Gb/s causes a slight detereoration in the BER_{pre} curve, reflected by the separation between the linear 32 Gb/s and the nonlinear result at that data rate. At 50 Gb/s, applying the full nonlinear Volterra model causes an even larger loss in performance.

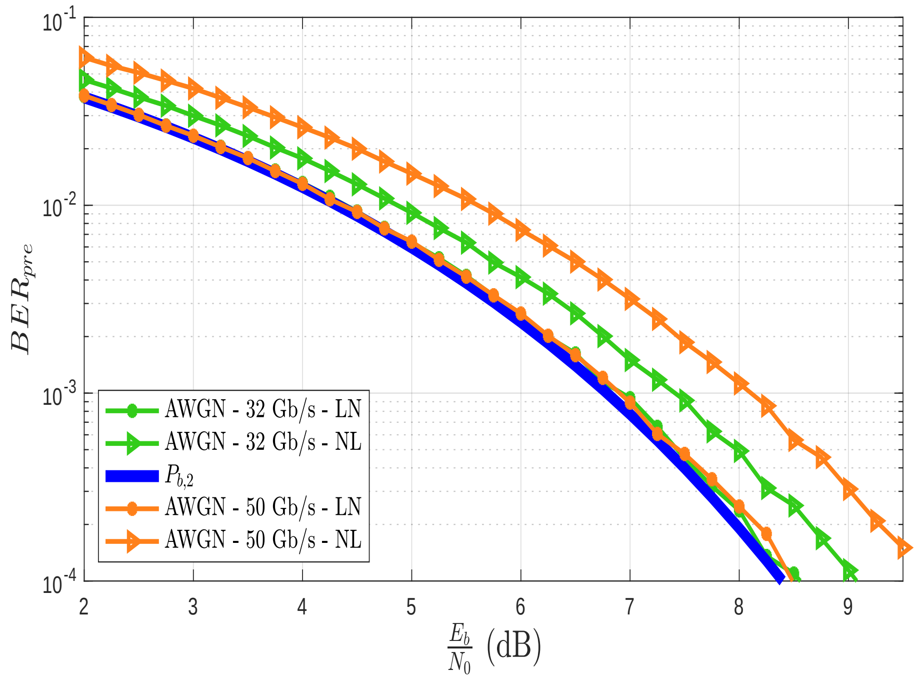


FIGURE 4.5: BER_{pre} curves for 32 Gb/s and 50 Gb/s signals that have gone through the linear and nonlinear versions of a $P = 3$ $M = 5$ Volterra model. The LDPC code rate is $R_{LDPC} = \frac{3}{4}$. Results for signals with the same data rate are plotted in the same colour, while results for the same type of model (LN/NL) are pictured with the same data marker.

4.2.3 Volterra model accuracy

Earlier in this thesis, two different ways of verifying the accuracy of the learned Volterra models have been explained. The first method utilizes the mean square error of the adaptive algorithm as a measure of accuracy, while the second calls for the comparison of measured output signal and modelled output signal eye diagrams. A variant of this second method that has not yet been suggested could be to compare the output signal waveform with the modelled output waveform. Although these techniques provide a useful way of "eyeballing" how accurate the Volterra models are, a more precise way to determine model fidelity is desirable.

Given that we have constructed a simulation that computes BER_{pre}, we can look for potential ways of using this metric to determine the accuracy of the learned Volterra models. Considering that BER_{pre} simulations do not require FEC coding, a simple way of comparing lab output signals with Matlab modelled outputs could be to simply transmit the recorded lab output signals and the Matlab Volterra model output signals through the synthetic AWGN channel and computing the corresponding BER_{pre} curves. In the case of lab recorded signals, the computation of BER_{pre} requires the comparison of the codebits of the input signal with the codebits of the amplifier output signal after it has gone through the synthetic AWGN channel. Gauging the similarities and differences between the lab signal and modelled signal curves, how accurate a portrayal of the real amplifier nonlinearities the models provide might be assessed. The computation of the BER_{pre} curves required for this comparison method is shown in Figure 4.6.

Based on the simulation infrastructure shown in the previous figure, a set of BER_{pre} curves related to lab measurements from the first experiment is computed. This is shown in Figure 4.7. The set of BER_{pre} curves is explained in the following enumeration:

1. The BER_{pre} curve computed for a recorded lab amplifier input signal at 50 Gb/s, 450 mVpp voltage, and a pattern length of $2^{15}-1$ bits after transmitting it through an AWGN channel. Because we are only comparing code bits influenced by additive white Gaussian noise, this curve should be extremely similar to the theoretical curve for a BPSK constellation in an AWGN channel, $P_{b,2}$.

2. The BER_{pre} curve computed for a recorded lab amplifier input and output signal pair at 50 Gb/s, 450 mVpp voltage, and a pattern length of $2^{15}-1$ bits after transmitting it through an AWGN channel. As is shown in Figure 4.6, to compute this BER_{pre} curve, the comparison occurs between the input signal code bits recorded in the lab and the amplifier output signal code bits recorded in the lab after they have travelled through the AWGN channel in the simulation. This information

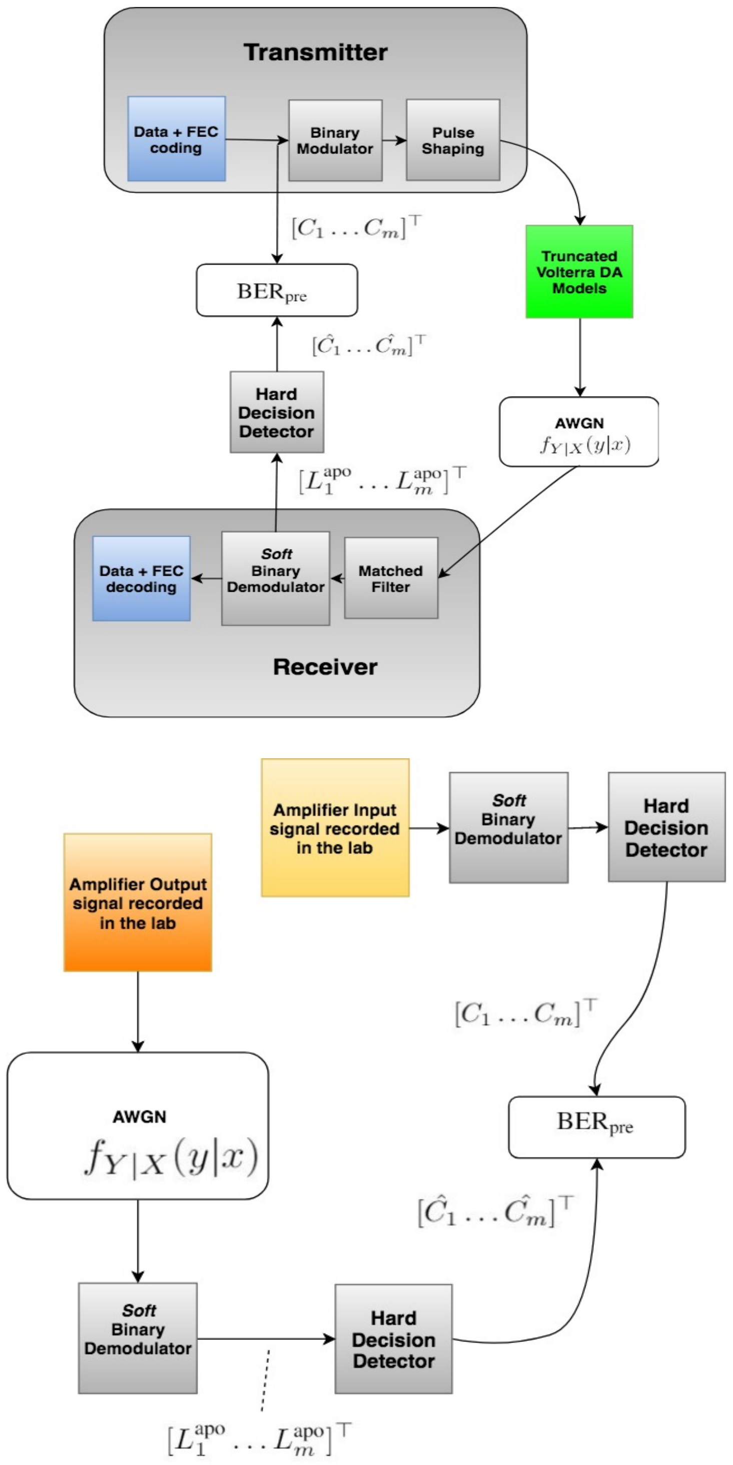


FIGURE 4.6: Methodology devised to determine Volterra model accuracy. BER$_{\text{pre}}$ values for the AWGN simulation results that include learned Volterra models (top image) are compared to BER$_{\text{pre}}$ values for lab recorded signals solely influenced by the synthetic AWGN channel (bottom image).

should portray the real nonlinear effects of the amplifier for this specific input signal.

3. The BER$_{\text{pre}}$ curves computed for the lab amplifier input signal at 50 Gb/s, 450 mVpp voltage, and a pattern length of $2^{15} - 1$ bits corrupted by two corresponding Volterra models of different order prior to transmission through the AWGN channel. These curves should portray the nonlinear effects of the amplifier that each of the models can successfully mimic for this specific input signal.

4. The theoretical BER$_{\text{pre}}$ curve for a BPSK constellation in an AWGN channel, $P_{b,2}$.

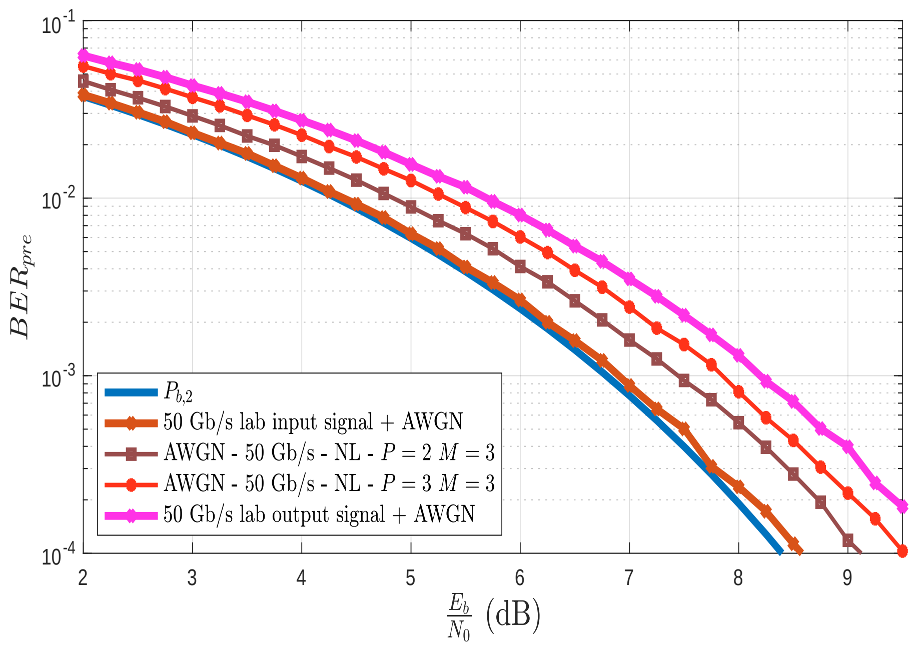


FIGURE 4.7: BER$_{\text{pre}}$ curves obtained from lab recorded signals as well as from Volterra modelled signals. This comparison represents a viable way of assessing Volterra model accuracy.

Upon inspection of Figure 4.7, a set of important observations can be made. To begin with, the result of transmitting the amplifier input signal through the AWGN signal results in an identical curve to $P_{b,2}$. This coincides with the expected result, as the amplifier input signal has not been subjected to any nonlinearities and thus should not exhibit a more degraded performance than the theoretical uncoded baseline. Differences arise when we compare the modelled signal BER$_{\text{pre}}$ curves with the lab recorded signal pair BER$_{\text{pre}}$ curve. By looking at the modelled signal BER$_{\text{pre}}$ curves, one can observe how both of the applied Volterra models fail at capturing the full extent of the performance degradation caused by the amplifier nonlinearity. This is represented by the separation on the x-axis between the aforementioned curves and the BER$_{\text{pre}}$ curve for the lab data. Interestingly, it seems as though the curve related to the higher order model ($P = 3$) provides a more accurate representation of the performance degradation caused by the amplifier than the curve related to the lower order model ($P = 2$). This means that the BER_{pre} curves might also constitute an effective way of selecting Volterra model order and memory.

In summary, the BER_{pre} metric provides an accurate way of assessing the accuracy of the learned Volterra models. Moreover, by previously accounting for model complexity, BER_{pre} may also be used to determine the optimum values for M and P of a Volterra model.

4.2.3.1 Determining the optimum order and memory for a Volterra model

Expanding upon the use of BER_{pre} to determine model accuracy, we can now study the tradeoff between between model complexity and accuracy to find the optimum values for M and P of each specific Volterra model.

Figure 4.8 shows the result of computing BER_{pre} for the transmission of a 32 Gb/s, 450 mVpp voltage, and $2^{15} - 1$ bit pattern lab recorded input signal through different order and memory nonlinear models and through the AWGN channel. It also includes the BER_{pre} result obtained for the corresponding input and output lab recorded signal pair (as shown in the bottom image of Figure 4.6) to gauge the accuracy of the Volterra models. The order of these models is either $P = 2$ or $P = 3$ and their memory is set at $M = 3$, $M = 4$, and $M = 5$. The linear AWGN transmissions for each model have been omitted to improve clarity of the graph[6].

Analyzing the curves shown in Figure 4.8 the following can be observed. At 32 Gb/s, neither increasing model memory nor model order seems to change model accuracy, reflected by the fact that the BER_{pre} curves for all three models are essentially identical to each other and the curve associated to the recorded input and output signal pair.

Figure 4.9 shows the result of computing the same BER_{pre} curves as in 4.8 but for signals at a data rate of 50 Gb/s. It is easy to see how at this higher data rate, the narrative differs with respect to 32 Gb/s. At 50 Gb/s, increasing the value of M results in BER_{pre} curves that are much closer to the result obtained for the lab recorded signal pairs.

The phenomena observed at 32 and 50 Gb/s can be explained by studying the characteristics of the applied models. When model memory is increased, more nonlinear terms are added to the Volterra series. At 32 Gb/s and an input voltage of 450 mVpp,

[6]When the nonlinear components of these models are “shut off” to obtain the linear versions of said models, variation between the values of their first order kernels is minimal. As a result, the linear AWGN curves for all the models are almost identical.

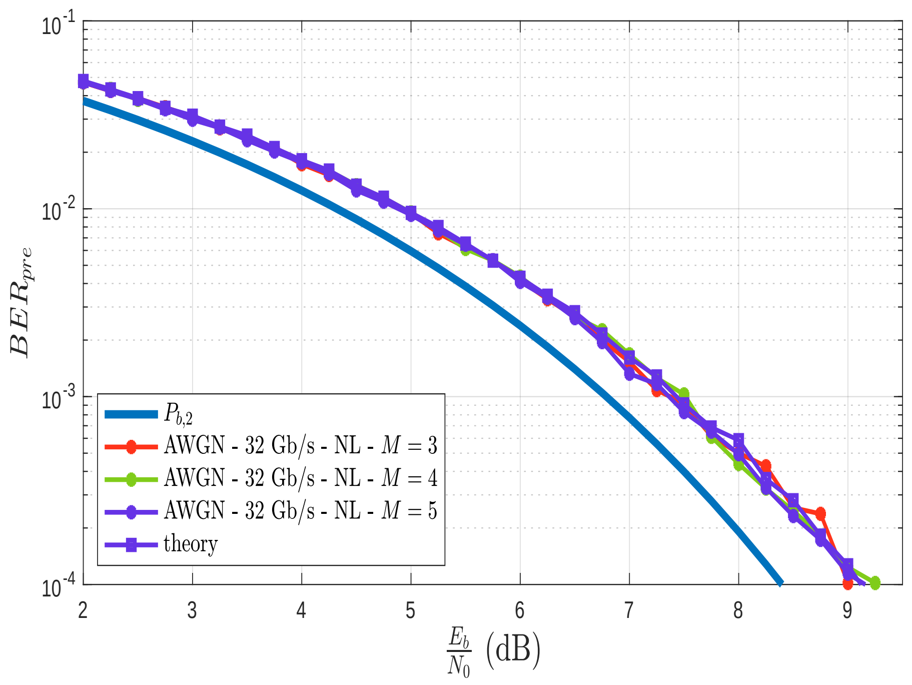


FIGURE 4.8: Comparison between the nonlinear third order $M = 3$, $M = 4$, and $M = 5$ models learned from 32 Gb/s input signals. Colours identify the results obtained for a specific model while data markers distinguish between signal data rates.

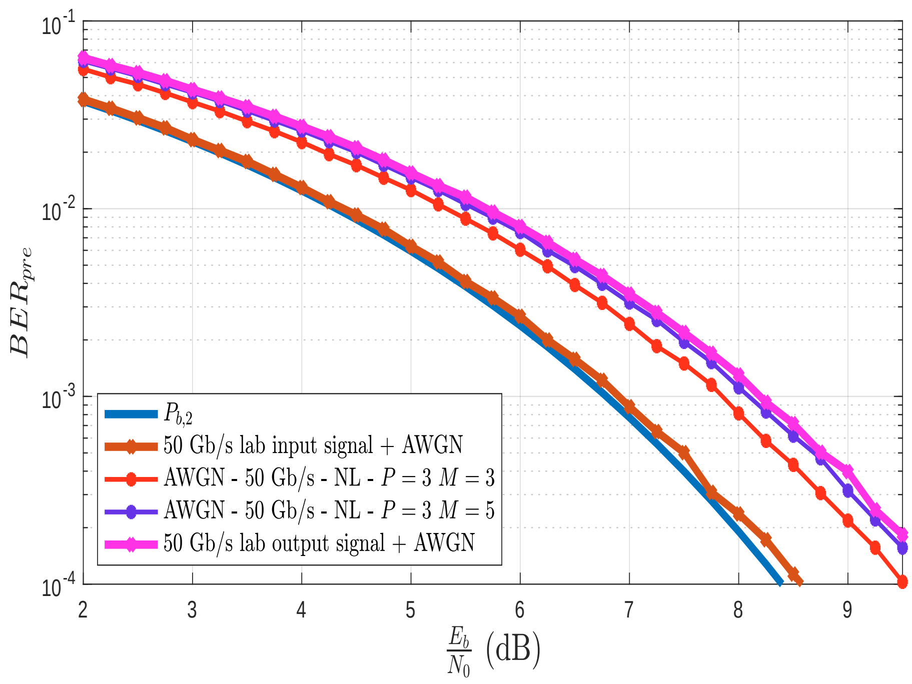


FIGURE 4.9: Comparison between the nonlinear third order $M = 3$ and $M = 5$ models learned from 50 Gb/s input signals. The $M = 5$ model achieves the most accurate portrayal of the amplifier behaviour.

the value of these nonlinear kernels is negligible. Therefore, although α goes from 19 at $M = 3$ to 55 at $M = 5$, this causes no noticeable differences on the BER_{pre} curves associated to each model. At 50 Gb/s and a voltage of 650 mVpp, the amplifier seems to exhibit augmented nonlinear behaviour, which is reflected by nonlinear kernels that are no longer negligible. At this data rate, adding more nonlinear terms to the Volterra model, because of the added non-negligible kernels, results in a closer BER_{pre} curve to the lab recorded signal pair result. This means that the nonlinear terms learned for higher M models have significantly larger values than they did at 32 Gb/s, as shown by the separation between the 50 Gb/s $M = 3$ and $M = 5$ curves of Figure 4.9. These observations prove the earlier prediction which stated that increasing model memory should result in more accurate models.

An explanation as to why these phenomena are taking place might be found by relating them to two factors: the input signal voltage and the amplifiers behaviour. The study of the relationship between input signal amplitude and amplifier nonlinearity will be discussed in Chapter 5, which works with the second lab measurements.

Part of the original hypothesis, which stated that model accuracy would increase with model order, has not yet been verified. This can be checked by simulating Volterra models with $P = 2$ and comparing the results to those obtained for the third order models. Figure 4.10 shows the bit error curves shown in Figure 4.9 along with the BER_{pre} curve when using a $P = 2$ $M = 3$ model. Their characteristics can be seen in Table 4.2.

Observing the results shown in Figure 4.10, it is easy to see how the second order model is incapable of emulating the nonlinear behaviour of the amplifier at 50 Gb/s. This can be explained by looking at the data shown in Table 4.2, specifically the magnitude of the second order kernels, $\tilde{h}_2$. For the $P = 2$ model, the value of $\tilde{h}_2$ is almost negligible. When $P = 3$, the same can be said for $\tilde{h}_2$ but not for $\tilde{h}_3$, whose value is two orders of magnitude larger than $\tilde{h}_2$. Therefore, the BER curve difference at 50 Gb/s when $P = 2$ and $P = 3$ must surely be caused by the cubic kernels. In view of these results, it is obvious that the hypothesis that increasing model order increases model accuracy is correct.

Based on the above observations and the data shown in Table 4.2, the best way to select the values for M and P for a given Volterra model comes down to minimizing Volterra model complexity while maximizing Volterra model accuracy. This can be summarized by the following phrase: *Select the smallest possible values for M and P that yield BER_{pre} curves that are acceptably close to the lab recorded data.* Determining what constitutes an acceptably close BER_{pre} curve will be conditioned by different requirements and will be further explained in the following chapter.

To close this subsection, after the variety of results shown within it, we can conclude that the magnitude of $\tilde{h}_2$ will remain constant and insignificant in comparison to $\tilde{h}_1$ and $\tilde{h}_3$ regardless of how M and P are varied. This phenomenon might explain the reason why in other papers, such as [16], [17], and [18], the quadratic kernels are not included in Volterra series expressions.

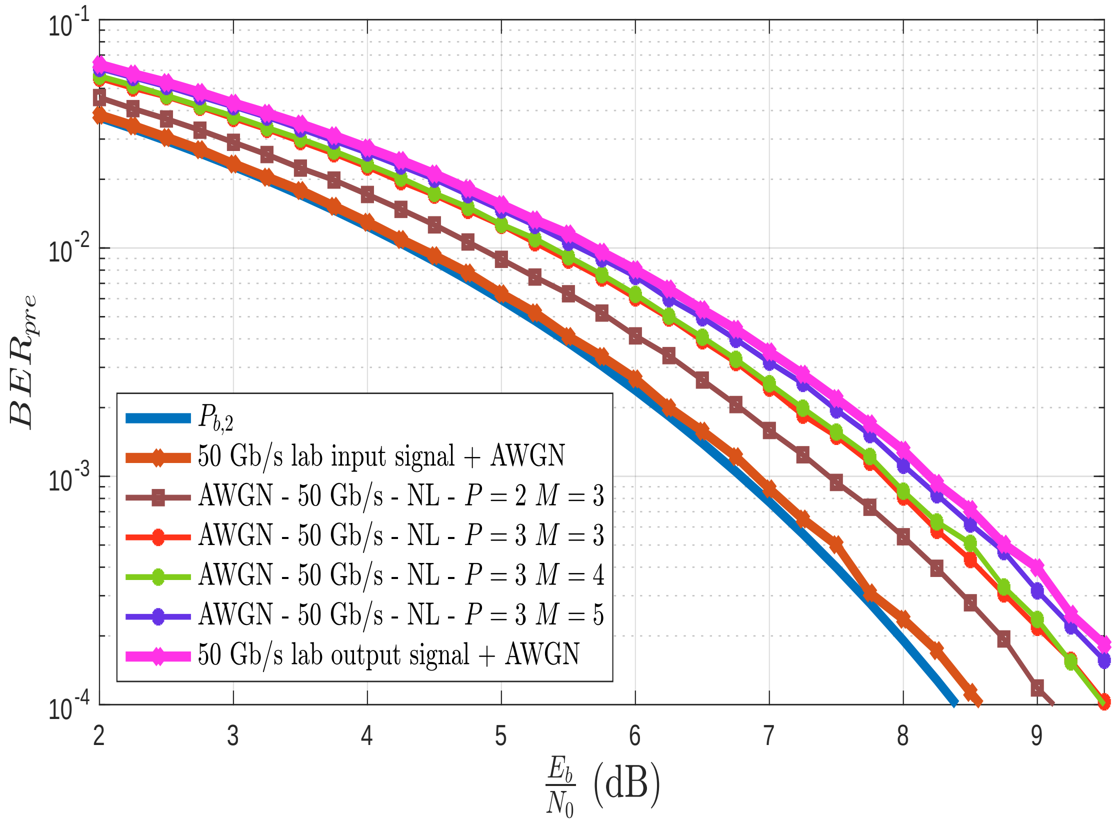


FIGURE 4.10: Comparison between the linear and nonlinear second order $M = 12$ model at 32 Gb/s and 50 Gb/s with the third order $M = 5$ model at those same data rates. The second order model is unable to reflect the increased nonlinearity that appears when the amplifier is fed with a 50 Gb/s signal.

4.2.4 Effects of varying the VLMS step size

It was mentioned earlier that BER results obtained when simulating models of same M and P but learned with different VLMS step-sizes, if these step sizes guarantee algorithm convergence, should result in almost identical outcomes. With the purpose of maintaining a certain degree of rigour, the previous statement should be verified. Figure 4.11 shows the results obtained when running the same simulation used to obtain the curves shown in Figure 4.5, but for an $M = 3$ $P = 5$ Volterra model relearned for different VLMS step size combinations. As was expected, variation in the results is negligible. This serves as a sanity check, proving the adaptive Volterra algorithm works as intended, and that the variation in the step sizes does not lead to drastically different results.

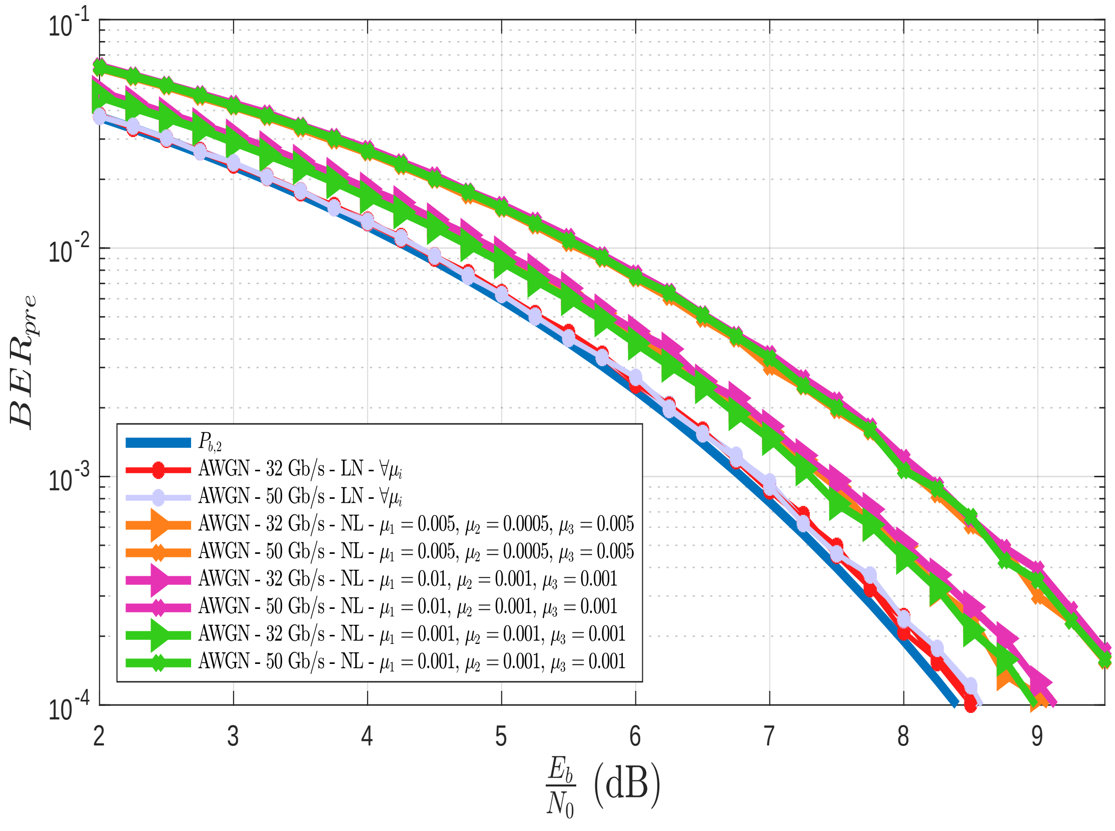


FIGURE 4.11: BER_{pre} curves for 32 Gb/s and 50 Gb/s transmissions for an $M = 3$ $P = 5$ model learned for different values of the VLMS step-sizes, μ_1, μ_2, and μ_3. The linear AWGN curves at 32 Gb/s and 50 Gb/s are almost identical for all the tested models which is why a single colour and curve format represents the linear results at a given data rate. As in Figure 4.9, the colours of the curves distinguish results for each model while markers differentiate between signal data rates.

Chapter 5

Factors that critically affect the content of nonlinearity in the amplifier response

In this chapter, we will use the computational and analytic tools derived in the prior contents of this thesis to process lab measurements recorded in the second experiment. The aim is to achieve a complete understanding of the relationship between specific input signal characteristics and the nonlinear content of the electronic amplifier response.

5.1 Model adjustment to input signal parameters

Throughout previous chapters of this thesis, characteristics of the amplifier input signals that cause change in its behaviour and hence variation of the learned Volterra models have been analyzed. The first set of lab measurements served to identify three specific aspects of the input signals that can be associated with change in the learned Volterra models. The first one is the data rate of the amplifier input signal, whose effects on model variability have previously been discussed to some extent. The second one is the pattern length of the amplifier input signal. In Chapter 3, a hypothesis for why said parameter may account for learned Volterra model variations was ventured. The third and final parameter related to changes in amplifier response is the peak-to-peak voltage of the input signal. It's impact on the learned Volterra models has not yet been discussed.

In order to conduct an exhaustive analysis on how these parameters cause variation in the learned Volterra models, precise lab recorded data sets are required. Although useful as a starting point, the measurements recorded in the first lab visit are not suitable for such a task. This initial experiment was meant to obtain just enough data to develop processing tools, so the input/output signal pairs that were recorded were not sufficient to thoroughly study the effects of these input signal parameters on amplifier behaviour. This is where the measurements conducted in the second lab visit come into play, as they where exclusively designed to analyze the impact of the aforementioned parameters. The input and output signal pairs recorded in the second lab experiment are shown in Table 5.1 (repetition of Table 3.3).

TABLE 5.1: Characteristics of the signals used as the input of the electronic drive amplifier in the second lab visit.

Modulation	Rate (Gb/s)	Pattern Length (bits)	Voltage (mVpp)	Equivalent Time Sampling Rate (Samples/symbol)	Waveform Length (samples)
NRZ	50	$2^9 - 1$	650	15	7664
NRZ	50	$2^{11} - 1$	650	15	30704
NRZ	50	$2^{15} - 1$	650	15	491504
NRZ	38	$2^{15} - 1$	650	19	622572
NRZ	44	$2^{15} - 1$	650	16	524271
NRZ	56	$2^{15} - 1$	650	15	425970
NRZ	56	$2^{15} - 1$	600	13	425970
NRZ	56	$2^{15} - 1$	500	13	425970
NRZ	56	$2^{15} - 1$	400	13	425970
NRZ	50	$2^{15} - 1$	600	15	491504
NRZ	50	$2^{15} - 1$	500	15	491504
NRZ	50	$2^{15} - 1$	400	15	491504

The aim of these measurements was to obtain data that would allow us to assess the impact caused by varying each of the three input signal parameters individually. With this goal in mind, numerous recordings of amplifier input/output signal pairs in which only one input signal parameter is varied and the other two are kept constant were extracted. By jointly analyzing these results, the relationship between the parameters and the learned models can be established and the derivation of a complete behavioural model might be achieved.

In summary, this chapter will study the impact of varying each of the three input signal parameters independently. The goal is to completely understand how the amplifier and the subsequent Volterra models will behave according to variations of these input signal parameters.

5.1.1 Analysis Guidelines

The data studied in this chapter will be analyzed based on the procedure summarized by the following set of guidelines:

1. In order to run the adaptive algorithm an equal number of iterations for every input/output signal measurement pair, shorter[1] input/output signal pairs are extended by sequentially concatenating them. This technique is more thoroughly explained previously in subsection 3.3.1.1.

2. All Volterra models extracted from the amplifier input/output signals recorded in the second lab experiment are learned for the combination of VLMS step-sizes: $\mu_1 = 0.01$, $\mu_2 = 0.001$, and $\mu_3 = 0.001$

3. All Volterra models extracted from the amplifier input/output signal pairs recorded in the second lab experiment are learned for $P = 3$ and $M = 5$. As was shown in chapter 8, these model memory and order values yield the most accurate nonlinear Volterra models while simultaneously maintaining model and algorithm complexity manageable. Higher values of M severely increase the time required by the VLMS to learn the model while providing negligible improvements in accuracy. Higher values of P have not been tested due to the augmented complexity of the adaptive algorithm[2].

4. Analysis of the signal measurements recorded in the second lab experiment and their associated results will be based on the BER_{pre} metric and the Matlab simulation that computes it. This metric and simulation were introduced in the previous chapter. The computation of BER_{pre} is shown in Figure 4.4. When necessary, eye diagrams will be included to provide additional information.

5. Throughout this chapter, different phenomena are said to cause an "SNR loss", "SNR degradation", or loss in performance. These phrases constitute a simple way of referring to an increase in the SNR required by a system in the presence of a specific phenomenon to achieve the same BER_{pre} values of that same system when not in the presence of the phenomenon in question. This "SNR loss" is embodied in the graphical results as the distance on the x-axis, at a given BER_{pre} value, between the BER_{pre} curves that are being compared.

6. When an input signal parameter is varied, the response of the electronic amplifier changes, which is the ultimate reason for variation of the learned Volterra models.

[1]In this context, shorter makes reference to signals that have a lower waveform length in samples.

[2]Increasing the nonlinear orders the adaptive algorithm is capable of learning represents an interesting endeavor for future work.

Although not mentioned explicitly at times throughout this chapter, when saying that the input signal parameters cause change in the Volterra models, it is indirectly implied that the reason for these variations in the models is the amplifier response.

5.2 Data Rate

In earlier chapters of this document, changes in the data rate of the amplifier input signal have been shown to cause variation in the learned Volterra models. Based on the set of measurements shown in Table 5.2, we will now ascertain the complete extent of the impact caused by varying the data rate of the input signal. The data shown in said table is compromised of input and output amplifier signal pairs recorded for four different data rates while keeping the pattern length and peak-to-peak voltage at constant values.

TABLE 5.2: Input signals used to study the relationship between the data rate and learned Volterra models.

Modulation	Rate (Gb/s)	Pattern Length (bits)	Voltage (mVpp)
NRZ	38	$2^{15}-1$	650
NRZ	44	$2^{15}-1$	650
NRZ	50	$2^{15}-1$	650
NRZ	56	$2^{15}-1$	650

Making use of the VLMS algorithm, the Volterra models corresponding to the data shown in the above table are obtained. Their characteristics are shown in Table 5.3. Based on these models, we employ the simulations introduced in Chapter 8 to compute the corresponding BER_{pre} curves. Figure 5.1 shows the BER_{pre} curves obtained when applying the models shown in Table 5.3 to the appropriate Matlab simulation. With these results in hand, information on how the data rate changes the learned Volterra models may be garnered.

Inspection of Figure 5.1 yields some interesting insights. The most notable fact is the performance degradation (in terms of the SNR) that results from an increase in the signal data rate. This phenomenon was foreshadowed by the results obtained after processing the first lab measurements (see Figure 4.5), and these new results can now be employed to reach a deeper understanding of its behaviour. If we simultaneously look at Table 5.3 as well as the above Figure, as the data rate of the signals increases and the resulting

TABLE 5.3: Characteristics of the Volterra models learned from the data shown in Table 5.2.

Model	$\tilde{h}_1$	$\tilde{h}_2$	$\tilde{h}_3$
38 Gb/s $2^{15}-1$ bits 650 mVpp	0.9296	0.0065	0.0639
44 Gb/s $2^{15}-1$ bits 650 mVpp	0.9094	0.0061	0.0845
50 Gb/s $2^{15}-1$ bits 650 mVpp	0.8985	0.0068	0.0947
56 Gb/s $2^{15}-1$ bits 650 mVpp	0.8633	0.0059	0.1308

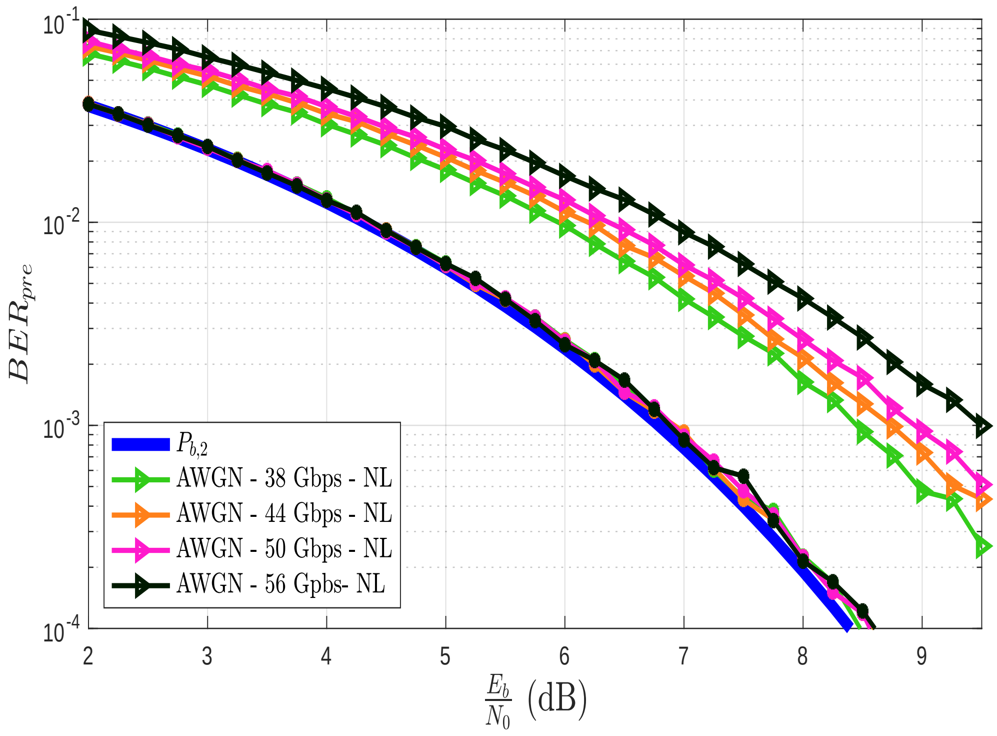


FIGURE 5.1: Computation of BER_{pre} curves when applying the models shown in Table 5.3. In the legend, NL stands for nonlinear, identifying the curves obtained when the full nonlinear Volterra models are applied in the simulation. Although not included in the legend to improve clarity, the curves obtained when applying linear Volterra models are maintained as a sanity check. They can be seen clustered around $P_{b,2}$.

BER_{pre} curves deviate further from $P_{b,2}$, a slight decrease in $\tilde{h}_1$ and an increase in $\tilde{h}_3$ of the learned Volterra models is observed.

Another factor to note is that all of the nonlinear BER_{pre} curves shown in Figure 5.1 display a significant overall loss in performance, more severe than any of the results that have been encountered up to this point in the thesis. All of the $\mathrm{BER}_{\mathrm{pre}}$ curves related to the different data rate signals shown in 5.1 are relatively close to each other and the SNR degradation related to changing signal data rates does not seem too severe in comparison to the overall loss exhibited by all of the curves. The separation between $P_{b,2}$ (the theoretical uncoded curve) and the 38 Gb/s curve is larger than the distance between the 38 Gb/s and the 56 Gb/s curves. Because of this, the major loss in performance must be due to one of the other input signal parameters studied in this chapter: the input signal voltage or pattern length.

We will now continue by trying to relate the changes in normalized Volterra kernels that occur as a consequence of modified input signal data rate with the respective $\mathrm{BER}_{\mathrm{pre}}$ curves, leaving the discovery of the reason for the overall loss in performance for posterior discussions in this chapter.

5.2.1 Relationship between input signal data rate related performance losses and normalized Volterra model kernel weights

When the data rate of the amplifier input signal is higher, the system requires higher signal SNRs to attain the same BER performance it achieves when using lower data rate input signals. This has been shown by the BER results depicted in Figure 5.1. If this loss in performance can somehow be related to changes in the normalized Volterra kernels of the learned models, the effects of input signal data rate on the amplifier response may potentially be characterized. Ideally, such a relationship would permit us to infer the Volterra model and $\mathrm{BER}_{\mathrm{pre}}$ curve associated to an input signal to the amplifier of any given data rate without requiring any lab measurements.

We begin with the mathematical characterization of the performance losses mentioned earlier in this section. If we select a specific value of $\mathrm{BER}_{\mathrm{pre}}$, a numerical comparison between SNR losses can be conducted based on the curves of Figure 5.1. Figure 5.2 shows the same curves of the aforementioned image for the more specific SNR range of 8 to 10 dB.

For the purposes of the comparison, the following set of metrics is defined:

- d_{th}: represents the distance, on the x-axis, between $P_{b,2}$ and the $\mathrm{BER}_{\mathrm{pre}}$ curve in question.
- d_{38}: represents the distance, on the x-axis, between the $\mathrm{BER}_{\mathrm{pre}}$ curve associated to the 38 Gb/s measurements and the $\mathrm{BER}_{\mathrm{pre}}$ curve in question.

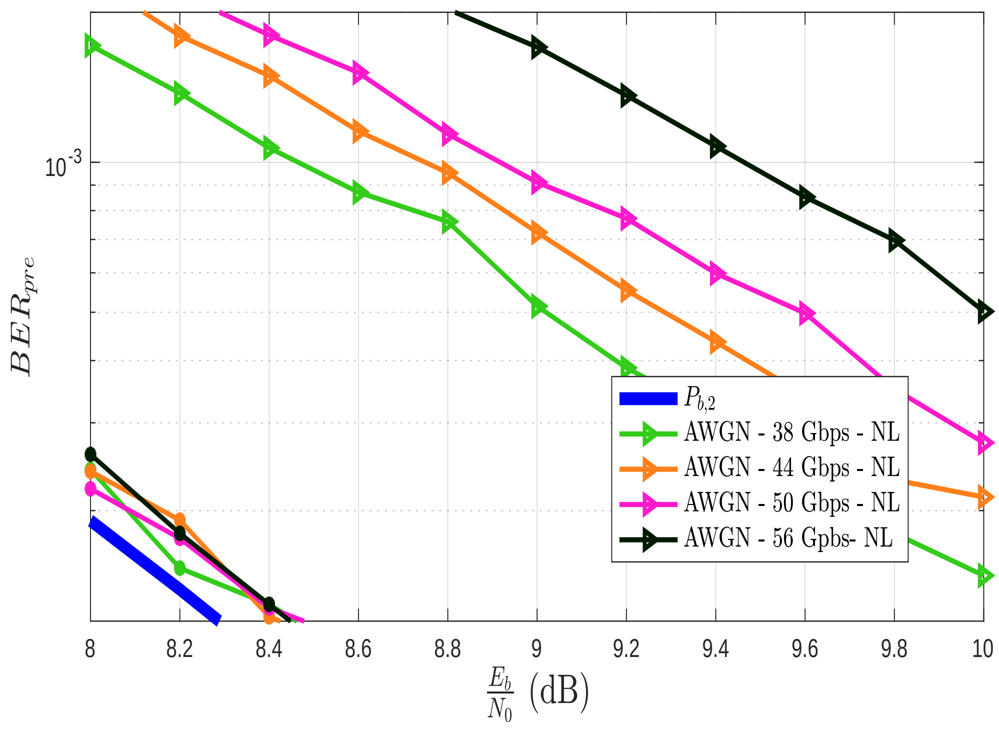


FIGURE 5.2: Zoomed in capture of the BER_{pre} curves of Figure 5.2.

- d_{r}: represents the distance, on the x-axis, between the closest lower data rate BER_{pre} curve and the BER_{pre} curve in question.

The values taken by these metrics based on the simulation results of Figure 5.2 are shown in Table 5.4. The BER_{pre} value for which the metrics have been computed is 10^{-3}. The linear BER_{pre} curves are excluded from this comparison as they are assumed to be identical to $P_{b,2}$.

TABLE 5.4: Characterization of the SNR loss caused by increases in input signal data rate for a $\text{BER}_{\text{pre}} = 10^{-3}$.

Signal Data Rate (Gb/s)	d_{th}	d_{38}	d_{r}
38 Gb/s	1.65	0	0
44 Gb/s	1.95	0.3	0.3
50 Gb/s	2.1	0.45	0.15
56 Gb/s	2.65	1	0.55

The results shown in Table 5.4 are intriguing in a number of ways. To begin with, the smallest value of d_{th} is significantly larger than the biggest value of d_{38}. The d_{th} metric provides a way of assessing the overall performance loss while d_{38} represents a method of determining the loss specific to an increase in data rate. Therefore, as was stated prior to this subsection, the increase in data rate is not the cause for the large loss in performance exhibited by all the nonlinear curves of Figures 5.1 and 5.2 and portrayed by all the d_{th} values.

To better understand the data in Table 5.4 we will now plot some of its contents. Figure 5.3 portrays the evolution of d_{38} as a function of the signal data rate. The increase in data rate related SNR loss is quite apparent in said image. The d_{r} metric, which serves to measure the loss in performance relative to the closest lower data rate curve, will serve to determine the step by step nature of the curve shown in Figure 5.3. The values of d_{r} in Table 5.4 are different for the same data rate increment of 6 Gb/s from measurement to measurement, which means that the relationship between d_{r} and the signal data rate is not linear. This explains the jagged appearance of the curve in Figure 5.3.

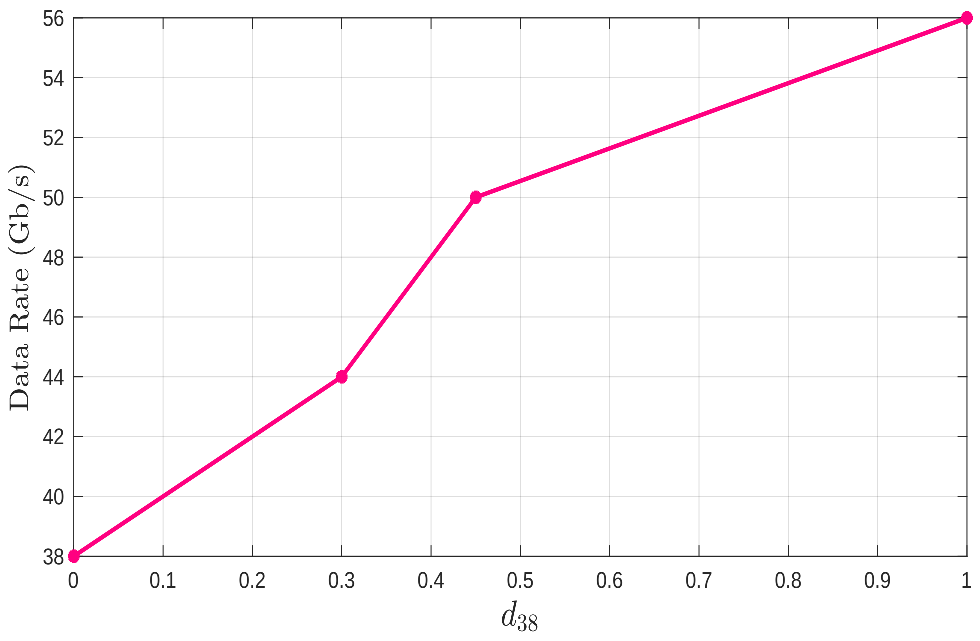


FIGURE 5.3: Evolution of d_{38} as a function of the input signal data rate.

The next step is to determine if the variations in d_{38} or d_{r} can be related to the changes observed in the normalized kernels of each model. In Table 5.3, the values of $\tilde{h}_2$ are shown to be various orders of magnitude smaller than the values of the other two normalized kernels. Because of the negligible values of $\tilde{h}_2$ and the fact that they remain essentially the same from model to model[3], we will ignore them for the purposes of the following analysis. Our main concern will be to determine how the linear and cubic nonlinear normalized kernels, $\tilde{h}_1$ and $\tilde{h}_3$, respectively, are related to the SNR loss observed when varying input signal data rates.

Figure 5.4 is a graphical representation of $\tilde{h}_1$ and $\tilde{h}_3$ as functions of d_{38}. The trend mentioned earlier in this section: decrease in $\tilde{h}_1$ and increase in $\tilde{h}_3$ as the data rate

[3]This was already foreseen in section 4.2.3 of the chapter 8.

goes up is now graphically visible, since d_{38} is an increasing function of the data rate. Moreover, the appearance of the plots of this figure seem to suggest that the behaviour of $\tilde{h}_1$ and $\tilde{h}_3$ as functions of d_{38} is, in a sense, linear.

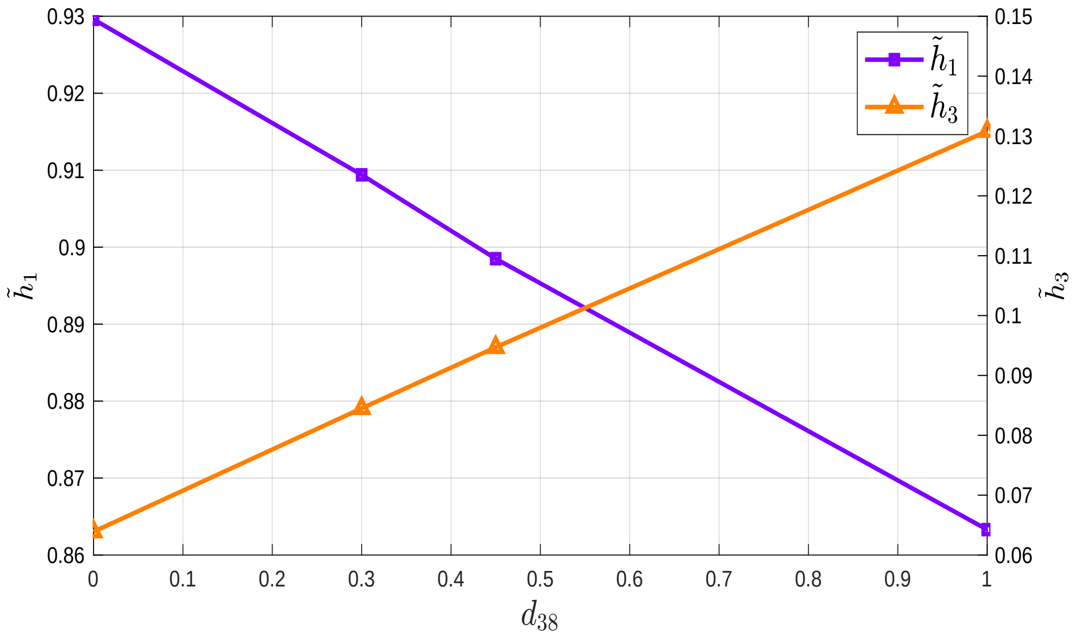


Figure 5.4: Evolution of $\tilde{h}_1$ and $\tilde{h}_3$ as functions of d_{38}.

To better understand the evolution of $\tilde{h}_1$ and $\tilde{h}_3$ we define parameters $c_{\tilde{h}_1}$ and $c_{\tilde{h}_3}$. They serve to measure the change in $\tilde{h}_1$ and $\tilde{h}_3$ from one model to the next as the loss d_{38} increases. A negative value of $c_{\tilde{h}_1}$ or $c_{\tilde{h}_3}$ represents a decrease of the normalized kernel in question while a positive value represents an increase. These metrics are included in Table 5.5.

Table 5.5: Evolution of $\tilde{h}_1$ and $\tilde{h}_3$ as functions of the signal data rate and d_{38}.

Signal Data Rate (Gb/s)	$\tilde{h}_1$	$\tilde{h}_3$	d_{38}	d_r	$c_{\tilde{h}_1}$	$c_{\tilde{h}_3}$
38 Gb/s	0.9296	0.0639	-	-	-	-
44 Gb/s	0.9094	0.0845	0.3	0.3	-0.0202	0.0206
50 Gb/s	0.8985	0.0947	0.45	0.15	-0.0109	0.0102
56 Gb/s	0.8633	0.1308	1	0.55	-0.0352	0.0361

Based on the results shown in Table 5.5 and Figure 5.4, the following observations can be made:

- The linear and cubic normalized kernels of the models learned for the different tested data rates decrease and grow in a linear fashion as the data rate is increased. If we look at the values of $c_{\tilde{h}_1}$ and $c_{\tilde{h}_3}$, we can see that when we go from 38 to 44 Gb/s, the value of $\tilde{h}_1$ decreases by essentially the same value by which $\tilde{h}_3$ increases.

This relationship holds when studying the rest of the data rates, so we can make the claim that $|c_{\tilde{h}_1}| \approx |c_{\tilde{h}_3}|$. This is an obvious observation, somewhat, as $\tilde{h}_2$ is negligible and thus being ignored.

- The values of $c_{\tilde{h}_1}$ and $c_{\tilde{h}_3}$ are linearly related to d_{38}. We define $c_{\tilde{h}}$ at a given data rate R Gb/s as

$$c_{\tilde{h}} = \sum_{i=38}^{R} \frac{|c_{\tilde{h}_1,i}| + |c_{\tilde{h}_3,i}|}{2}, \tag{5.1}$$

where the sum in 5.1 is computed over the discrete set of 6 Gb/s incrementing data rate measurements we possess. If we then compute $\frac{c_{\tilde{h}}}{d_{38}}$, we obtain the results shown below:

TABLE 5.6: Values of $\frac{c_{\tilde{h}}}{d_r}$.

Signal Data Rate (Gb/s)	$c_{\tilde{h}}$	d_{38}	$\frac{c_{\tilde{h}}}{d_{38}}$
44	0.0204	0.3	0.0684
50	0.0309	0.45	0.0688
56	0.0665	1	0.0665

As can be seen above, the values for $\frac{c_{\tilde{h}}}{d_{38}}$ remain very similar as the signal data rate increases. This behaviour makes it plausible to obtain d_{38} from $\tilde{h}_1$ and $\tilde{h}_3$ based on the quasi-linear curves of Figure 5.4.

- Although $\tilde{h}_1$ and $\tilde{h}_3$ might be linearly related to d_{38}, the amount by which d_{38} varies for each 6 Gb/s increment of the measurements does not respond to a linear law. This can be seen in Figure 5.3. It will later be shown to be related to the value of the input signal voltage of the measurements: 650 mVpp.

Should our purpose be to devise an inference-based amplifier model, in light of these observations, the conundrum revolves around the accurate prediction of d_{38} for a specific data rate. One possibility would be to assume a linear relationship between d_{38} and the input signal data rate, but based on Figure 5.3, this would be a very crude approximation, at best. All in all, although some enticing results have been obtained, the fact that the change in d_{38} is different for each 6 Gb/s increment of our measurements renders coming up with accurate predictions a somewhat pointless endeavour, as we have no way of determining how to make these predictions.

The main takeaway from the analysis we have conducted in this section is that an increase in data rate results in an increase in performance loss, and that the normalized kernels appear to provide a way of determining the amount of nonlinearity exhibited by the amplifier in each scenario. The larger the weight of the linear normalized kernel (or smaller the weight of the cubic normalized kernel) the more linear the system is and the lower the SNR loss will be. The smaller the weight of the linear normalized kernel (or larger the weight of the cubic normalized kernel) the more nonlinear the system is and the larger the SNR loss will be. From there, it is easy to obtain an idea on what the impact on the performance of the system will be under specific operating conditions.

If this behaviour is maintained for the other two input signal parameters, these normalized kernels will become increasingly useful. For example, they might provide a way to determine, depending on the amount of performance loss we want to tolerate, which amplifier operating conditions (input signal parameter configurations) are acceptable and which ones are not.

5.3 Bit Pattern Length

Another of the input signal parameters whose variation plays a role in the change of the learned Volterra models is the length of the bit pattern that compromises the input signal to the amplifier.

In chapter 3, we introduced the notion that there might be a relationship between the variation in the normalized kernel weights of learned Volterra models and the harmonic content of the input signal. This idea is grounded on the difference between the frequency spectrums of signals with different PRBS pattern length.

The set of measurements obtained in the first lab experiment was compromised of only two PRBS pattern lengths. Although these measurements provided some useful insight, acquiring data related to more PRBS pattern lengths serves to solidify the claim that longer sequences lead to the learning of models that better represent the amplifiers behaviour. This is accomplished with the second batch of lab measurements, which provides the necessary data to determine the validity of the proposition made in chapter 3.

The specific set of measurements from the second lab experiment that will be used to assess the effects of varying input signal bit pattern lengths on the learned Volterra models can be seen in Table 5.7. The data shown in said table is compromised of input and output amplifier signal pairs recorded for 3 different PRBS pattern lengths. The input signal data rate and peak-to-peak voltage were maintained at constant values.

As was done previously when studying the effects caused by the variation of the input signal data rate, we compute the BER_{pre} curves that result from applying the nonlinear models shown in Table 5.8 to the simulations derived in chapter 8. The models contained

Table 5.7: Characteristics of the electronic drive amplifier input signals used to study the effects of signal data rate on learned Volterra models.

Modulation	Rate (Gb/s)	Pattern Length (bits)	Voltage (mVpp)
NRZ	50	$2^9 - 1$	650
NRZ	50	$2^{11} - 1$	650
NRZ	50	$2^{15} - 1$	650

within said table are learned from the data shown in Table 5.7. Figure 5.5 shows the BER$_{\text{pre}}$ curves obtained when applying the models shown in Table 5.8 to the appropriate Matlab simulation (Figure 4.4).

Table 5.8: Characteristics of the Volterra models learned from the data shown in Table 5.7.

Model	$\tilde{h}_1$	$\tilde{h}_2$	$\tilde{h}_3$
50 Gb/s $2^9 - 1$ bits 650 mVpp	0.9122	0.0090	0.0788
50 Gb/s $2^{11} - 1$ bits 650 mVpp	0.9036	0.0054	0.0910
50 Gb/s $2^{15} - 1$ bits 650 mVpp	0.8985	0.0068	0.0947

The results shown in Figure 5.5 show an overall SNR degradation that is very similar to what was observed when studying the BER$_{\text{pre}}$ curves in section 5.2. Once again, the curves obtained for the full nonlinear Volterra models are relatively close together and substantially far from the curves that result from applying the linear versions of said models.

Considering that this phenomenon is observed both when varying the data rate (Figure 5.1) and the bit pattern length (Figure 5.5) of the input signal, and given that the only input signal parameter that has remained constant during these simulations is the input signal peak-to-peak voltage, it seems like the latter parameter is the root cause for the overall performance loss present in all of the nonlinear BER$_{\text{pre}}$ curves we have analyzed up to this point. This will be discussed in the next section of the chapter.

If we continue our inspection of Figure 5.5 by looking at the BER$_{\text{pre}}$ curves associated to the full nonlinear models of each pattern length, very slight increases in SNR degradation can be observed as the input signal pattern becomes longer. These increases are so small

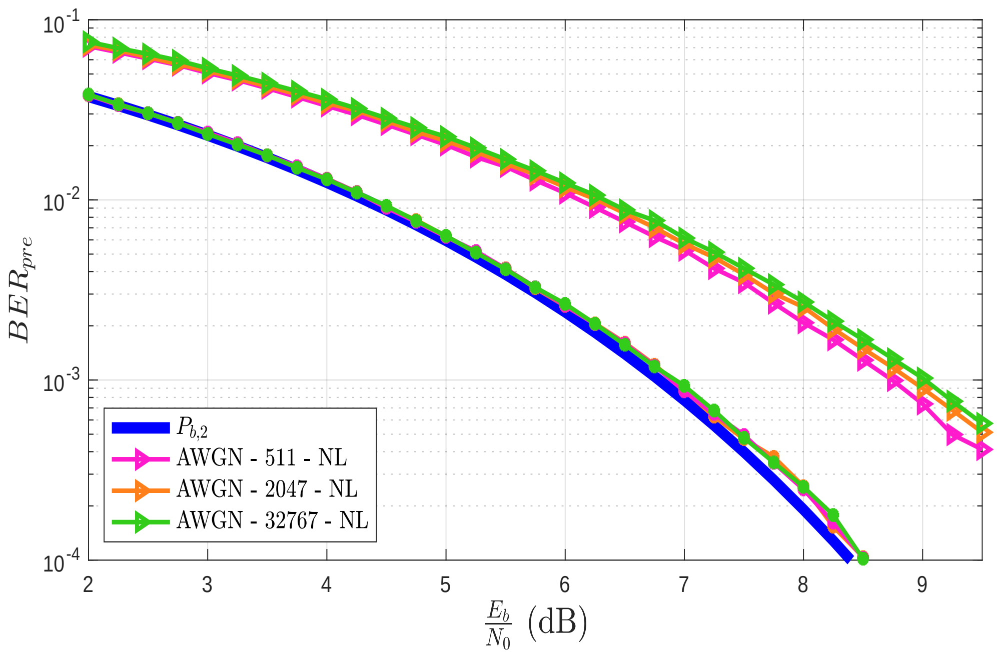


FIGURE 5.5: Computation of BER_{pre} curves when applying the models shown in Table 5.8. The curves obtained when applying linear Volterra models are maintained as a sanity check.

that there is very little difference between the curves. The one sided frequency spectrums of the three tested pattern lengths (measured during the second lab experiment by means of a spectrum analyzer) can be seen in Figure 5.6. Although similar to each other, it is possible to see how the data captured by the spectrum analyzer becomes more neat as the tested sequences become longer. The following paragraph provides an explanation for both these phenomena.

Pseudo-random binary sequences are approximations of infinitely long true random sequences typically used in simulation environments as substitutes of actual data signals. Naturally, the longer the pseudo-random binary sequence, the better the approximation. In most simulations, data signals are constructed by serially concatenating the same pseudo-random binary sequence. Apart from determining how good the approximation to a true data signal is, the length of a pseudo-random binary sequence will also determine the number of data bits that are transmitted before that same sequence is repeated. If we take a signal constructed with a PRBS of length $2^9 - 1$, the data sequence will be composed of the serial time repetition of that same PRBS of length $2^9 - 1$ bits. In the frequency domain, this serial repetition of a sequence that is an approximation of a truly random signal results in a signal spectrum that exhibits void or empty areas referred to as “spectral holes”. The number of these spectral holes is larger for data signals constructed based on shorter PRBS’ because they represent worse approximations of a real data signal.

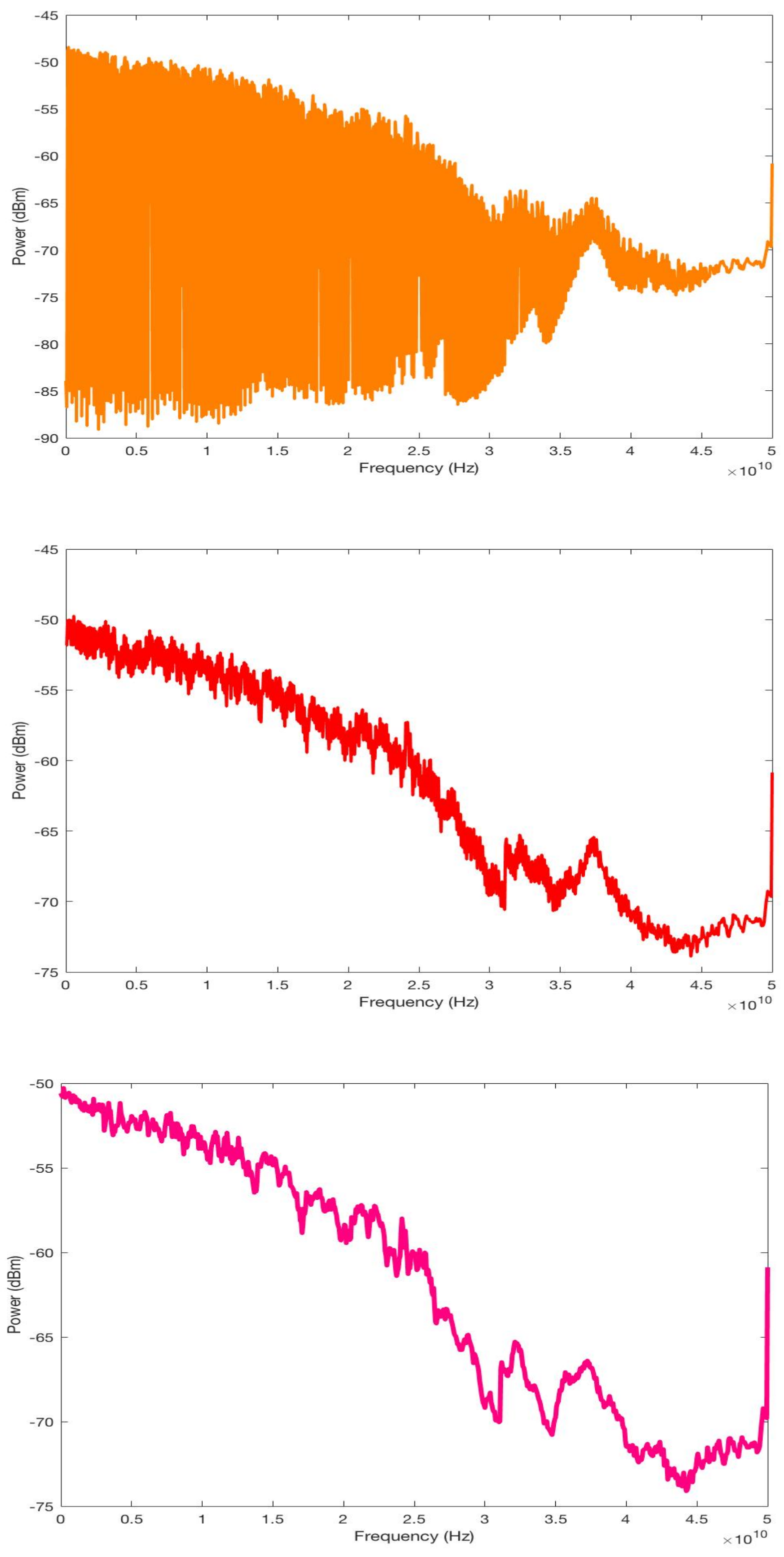


FIGURE 5.6: Measured one sided frequency spectrum for all the three different pseudo-random bit sequences that were tested. From top to bottom: Measured frequency spectrum for a PRBS of $2^9 - 1$ bits. Measured frequency spectrum for a PRBS of $2^{11} - 1$ bits. Measured frequency spectrum for a PRBS of $2^{15} - 1$ bits.

When applied as the input to the amplifier, these holes will fail to elicit a response from the device at those frequencies, which means that for those specific frequencies, the adaptive algorithm will be unable to map the behaviour of the amplifier to the Volterra model. As PRBS length is increased, the approximation improves and the number of spectral holes decreases, which means that more complete learning of amplifier behaviour will take place when using longer sequence lengths. This explains why in Figure 5.5 more SNR loss can be seen in the curves related to longer pattern lengths, as they are able to learn a more complete amplifier response. However, as was mentioned earlier, this accuracy improvement is not very substantial. This decrease in the quantity of spectral holes as pattern length increases may also be the reason for the difference in the quality and neatness of the measured signal spectrums shown in Figure 5.6.

Because the behaviour observed in the BER_{pre} results shown in Figure 5.5 and the above explanation are consistent with the prediction that was made in Chapter 3, we can state with a high degree of certainty that signals that use longer pseudo-random bit sequences allow the adaptive algorithm to produce more accurate Volterra models. As was explained in the previous paragraph, the more complete frequency spectrum of signals constructed using longer bit sequences will be able to elicit a behavioural response from the amplifier closer to the one we would obtain if we had an infinitely long random sequence. In short, the longer the selected bit sequence, the more accurate the learned Volterra model.

However, there is an important caveat: working with lengthier PRBS sequences leads to a significant increase in the experimental and memory requirements of the necessary equipment. For instance, an attempt was made during the second lab experiment to record data based on pseudo-random bit sequences of $2^{23} - 1$ bits, but the combination of the time required by the DCA to obtain a good enough waveform capture of such length and the size of the resulting file, made it an impractical endeavor.

Although we have no way of accurately determining the magnitude of the representation error that occurs because of using shorter pattern lengths (we are physically limited by our capability of generating longer sequences after all), it seems reasonable to try and estimate it. The relative proximity of the nonlinear BER results shown in Figure 5.5 seems to suggest it is not too substantial, as there is little difference between the nonlinear curves obtained for the different PRBS pattern lengths. As we will see in the following section, this error will be negligible in comparison to the effects caused by varying the data rate or input signal voltage.

Given the fact that the length of the longest PRBS sequence manageable by the equipment at hand was $2^{15} - 1$ bits, we assume that sequences of said length will be capable of exciting an almost complete behavioural response from the amplifier and choose to ignore the underlying PRBS length related representation error.

5.4 Peak-to-peak Voltage

The final input signal parameter that changes the learned Volterra models when it is modified is the peak-to-peak voltage of the input signal. Previously in this chapter, this parameter was proposed as the main reason for the recurring loss in SNR performance observed in the BER results studied in sections 5.2 and 5.3. Assuming this is true, changes in this parameter cause significant variations in BER performance and learned Volterra models. By making use of the data obtained in the second lab experiment, we will be able to verify this assumption and determine how significant these variations are in comparison to the effects caused by varying the other two parameters analyzed in this chapter.

The measurements used to analyze the effects of input signal voltage on the learned Volterra models can be seen in Table 5.9. The data shown in said table is compromised of input and output amplifier signal pairs recorded for four different peak-to-peak voltage values. The selected input signal bit pattern was $2^{15}-1$, the longest possible PRBS the available resources could work with[4]. Measurements for two different data rates were recorded. The Volterra models learned for the data shown in said table can be seen in Table 5.10.

TABLE 5.9: Characteristics of the electronic drive amplifier input signals used to study the effects of signal peak-to-peak voltage on learned Volterra models.

Modulation	Rate (Gb/s)	Pattern Length (bits)	Voltage (mVpp)
NRZ	50	$2^{15}-1$	400
NRZ	50	$2^{15}-1$	500
NRZ	50	$2^{15}-1$	600
NRZ	50	$2^{15}-1$	650
NRZ	56	$2^{15}-1$	400
NRZ	56	$2^{15}-1$	500
NRZ	56	$2^{15}-1$	600
NRZ	56	$2^{15}-1$	650

Figure 5.7 shows the BER_{pre} curves obtained when applying the 50 Gb/s models shown in Table 5.10 to the appropriate Matlab simulation.

[4]As was shown in section 5.3, the use of longer bit sequences results in the learning of more accurate Volterra models but also increases the storage and processing requirements.

TABLE 5.10: Characteristics of the Volterra models learned from the data shown in Table 5.9.

Model	$\tilde{h}_1$	$\tilde{h}_2$	$\tilde{h}_3$
50 Gb/s $2^{15}-1$ bits 400 mVpp	0.9919	0.0071	0.0010
50 Gb/s $2^{15}-1$ bits 500 mVpp	0.9612	0.0078	0.0310
50 Gb/s $2^{15}-1$ bits 600 mVpp	0.9286	0.0063	0.0631
50 Gb/s $2^{15}-1$ bits 650 mVpp	0.8985	0.0068	0.0947
56 Gb/s $2^{15}-1$ bits 400 mVpp	0.9885	0.0090	0.0025
56 Gb/s $2^{15}-1$ bits 500 mVpp	0.9582	0.0082	0.0336
56 Gb/s $2^{15}-1$ bits 600 mVpp	0.9236	0.0042	0.0722
56 Gb/s $2^{15}-1$ bits 650 mVpp	0.8633	0.0059	0.1308

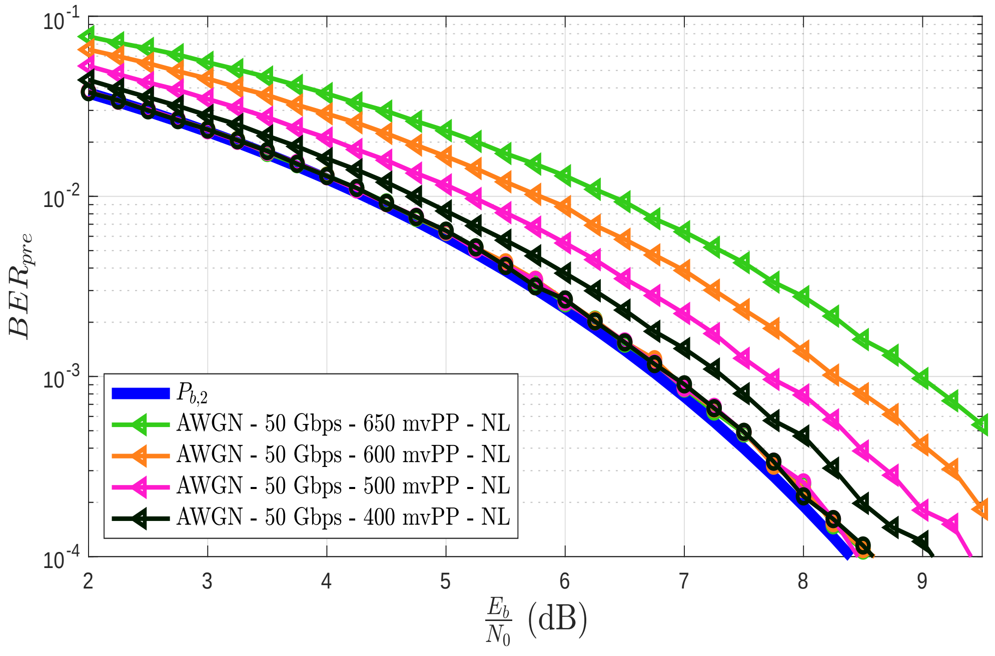


FIGURE 5.7: Computation of BER_{pre} curves when applying the models shown in Table 5.10.

The first thing that stands out from the results shown in Figure 5.7 is the difference, in terms of the curves associated to each input signal peak-to-peak voltage, of the SNR degradation they portray. The larger the peak-to-peak voltage of the input signal, the more the corresponding $\mathrm{BER}_{\mathrm{pre}}$ curve deviates from the curve associated to the linear version of the corresponding Volterra model. This can also be seen in the eye diagrams of the amplifier output signals obtained for the 50 Gb/s data of Table 5.9. They are shown in Figure 5.8. In said diagrams, the loss in SNR can be seen as the progressive closing of the eye as the input signal voltage is increased.

Both Figures 5.7 and 5.8 show that the largest SNR loss occurs at the highest input signal voltage of 650 mVpp. Recalling the data and results that were studied in the previous sections of this chapter, they were all obtained from measurements recorded for an input signal voltage value of 650 mVpp. A large overall SNR loss was observed in all the results associated to these data sets, and because it was the only parameter that was kept constant in all of them, the input signal voltage was thought to be the potential cause for it. The correlation between input signal voltage and SNR degradation observed in Figures 5.7 and 5.8 serves to solidify this idea, providing proof for how the main driving factor for the performance loss present in all the previously analyzed results is indeed the voltage of the input signal to the amplifier.

Another matter of interest is the variation between the $\mathrm{BER}_{\mathrm{pre}}$ curves associated to different input signal voltages. It is more significant than the variations that where observed between curves associated to different data rates (Figure 5.1) and substantially larger than the variations between curves associated to different PRBS pattern lengths (Figure 5.5). In view of these results, we hypothesize that the input signal voltage will be the factor (more so than the other two parameters) that most influences the presence, or lack thereof, of nonlinearity in the amplifier behavioural response and subsequent changes in the learned Volterra models.

This hypothesis can be explained by looking at the transfer function of a generic electronic amplifier. The transfer function of an electronic device is a tool used to express the device's output as a function of its input. For amplifiers, this is generally represented by a graph of the voltage at the output as a function of the voltage applied to the input. An example of an electronic amplifier transfer function is shown in 5.9. As can be seen in said image, two different behavioural regions can be observed: the linear region and the saturation or nonlinear region. The ideal characteristic curve of the amplifier is provided for context. If we look at the actual response of the amplifier, even within the linear region, as the input signal voltage is increased, the amplifier will behave in an increasingly nonlinear manner. Beyond the saturation point, increasing input signal

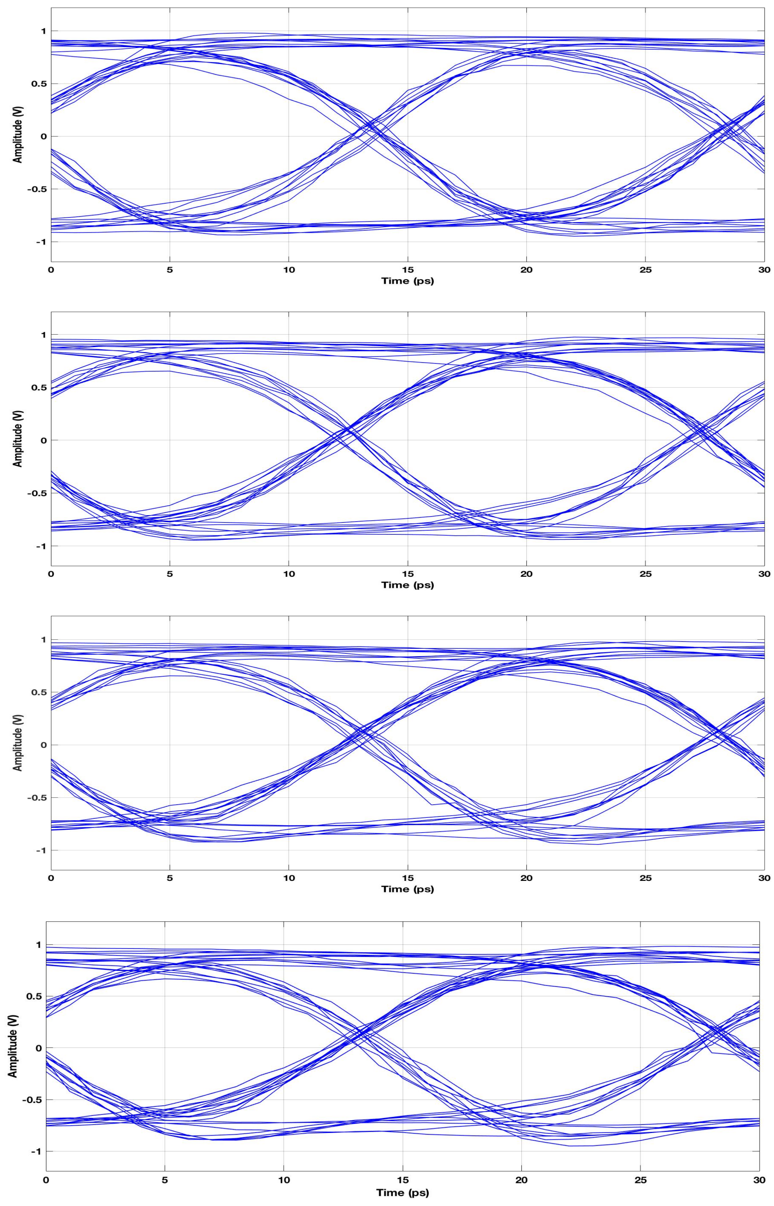


Figure 5.8: Eye diagrams corresponding to the output signals of the 50 Gb/s data shown in Table 5.9. The peak-to-peak voltage of the input signals associated to each eye diagram is, from top to bottom: 400 mVpp, 500 mVpp, 600 mVpp and 650 mVpp. The voltage of the signals has been normalized to the bipolar range to permit visual comparisons of the closing of the eye as voltage increases.

voltage will only lead to higher nonlinear content in the output signal and negligible gain increase.

This theory explains the input signal voltage related change in Volterra models and increase in performance loss as follows: As has just been shown, increases in input signal voltage lead to augmented nonlinear content in the output, which the learned Volterra model will theoretically reflect in the values of its normalized kernels (increases in nonlinear kernels). Application of Volterra models learned for increasing input signal voltages to the BER_{pre} simulation will result in progressively higher SNR losses, as the

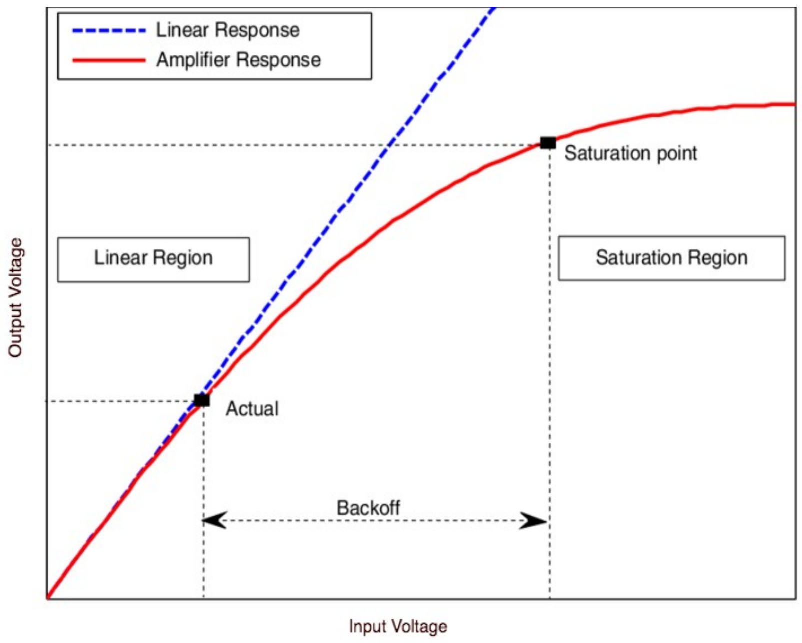


FIGURE 5.9: Generic transfer function of an electronic drive amplifier. Original image is obtained from [1].

nonlinear kernels of the Volterra models change in response to the incrementing input signal voltages.

The next step is to determine the exact relationship between the normalized kernels of the Volterra models and the SNR losses observed in Figure 5.7, as it will allow us to understand the change in system performance related to change in input signal voltage. Moreover, this would ideally allow the construction of an inference-based model for the input signal voltage similar to the one we tried to derive in section 5.2 for the input signal data rate.

Prior to doing so, let us study the 56 Gb/s data shown in Table 5.9. According to the results and conclusions that were reached in section 5.2 of this chapter, these higher data rate measurements should reflect an increased loss in performance for all four input signal voltages in comparison to the 50 Gb/s results shown in Figure 5.7. Plotting the 400 mVpp and 650 mVpp measurements for both of these data rates will allow us to verify these expected outcomes. These plots can be seen in Figure 5.10.

As was expected, both the 56 Gb/s 400 mVpp and 650 mVpp BER_{pre} curves resulting from the full nonlinear Volterra models of Table 5.10 show a higher performance degradation than the same curves obtained for a data rate of 50 Gb/s. Interestingly, this data rate related degradation is higher at 650 mVpp than at 400 mVpp. This might mean that given two input signal voltages X, Y such that $X > Y$, increasing the data rate for input signal voltage X will result in larger nonlinear content of the learned

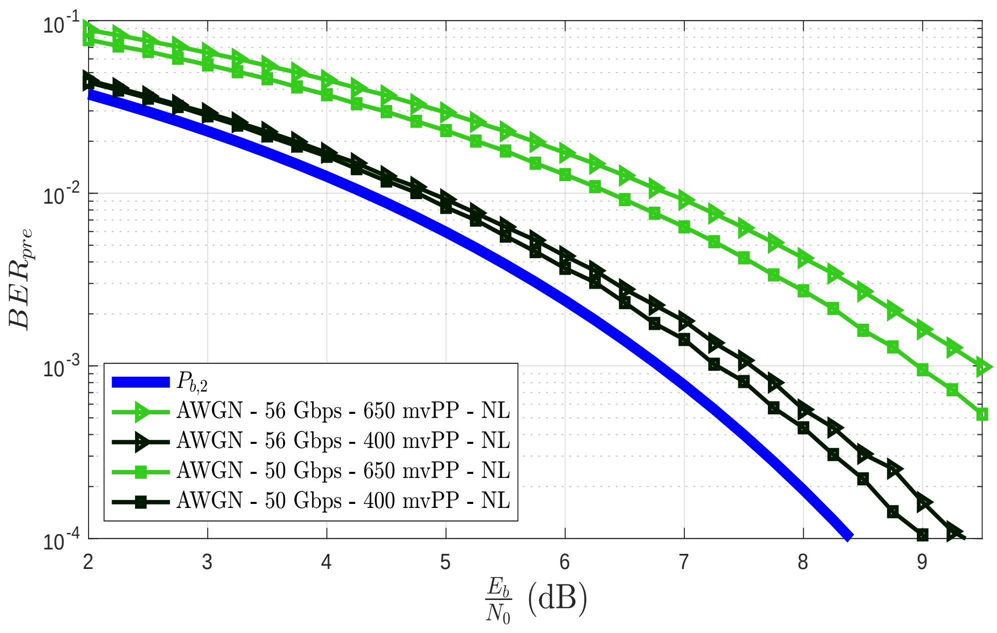


FIGURE 5.10: BER_{pre} curves obtained when applying the 400 mVpp and 650 mVpp models for 50 and 56 Gb/s data rates shown in Table 5.10.

Volterra models than that same data rate increase at input signal voltage Y. This will be discussed in the following subsection.

5.4.1 Relationship between input signal voltage performance losses and normalized Volterra model kernels

The results discussed previously in this section have proven that the learned Volterra models change in response to the amplifier becoming more nonlinear as the input signal voltage increases. This results in higher signal SNRs being needed by systems using signals with an input voltage V_1 to achieve the same BER values of systems with an input signal voltage V_2, if $V_1 > V_2$. Let us proceed by determining the nature of this relationship between input signal voltage and system performance degradation.

5.4.1.1 Analysis of the performance losses due to variation in the peak-to-peak voltage of the input signals

As we did previously, we begin by numerically characterizing the performance losses shown in Figure 5.7. These results are portrayed in the more specific SNR range of 6 to 10 dB in Figure 5.11.

This analysis is conducted based on the set of metrics listed below, which are computed for the generic BER_{pre} value of 10^{-3}.

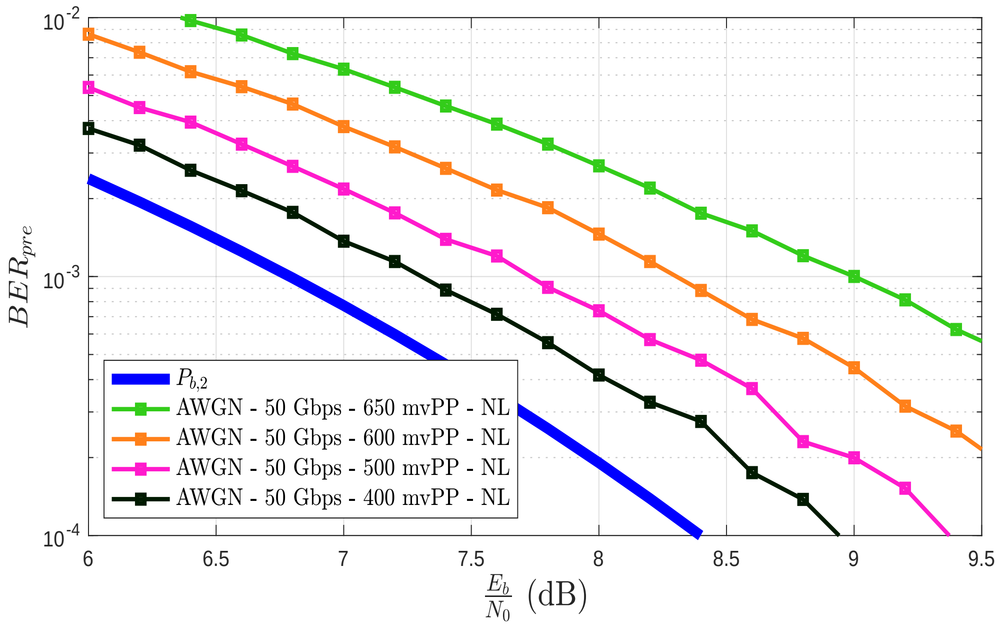


FIGURE 5.11: Zoomed in capture of the results shown in Figure 5.7.

- d_{th}: represents the distance, on the x-axis, between $P_{b,2}$ and the BER$_{\mathrm{pre}}$ curve in question.
- d_{400}: represents the distance, on the x-axis, between the BER$_{\mathrm{pre}}$ curve for the 400 mVpp measurements and the BER$_{\mathrm{pre}}$ curve in question.
- d_{v}: represents the distance, on the x-axis, between the closest lower input voltage BER$_{\mathrm{pre}}$ curve and the BER$_{\mathrm{pre}}$ curve in question.

These metrics will be denoted with a superscript representing the specific data rate of the measurements from which they were computed. The notation without the superscript, i.e d_{400}, encompasses all the superscripted metrics. The superscript is a simple way of indicating the measurements from which the values of the metrics were obtained, but the metrics themselves are rate independent and specifically designed to compare results that originate from different data rate measurements.

The values of these metrics for measurements at 50 Gb/s are shown in Table 5.11. The large increase in the values of d_{th}^{50} as the signal input voltage grows (0.5 dB approximately per measurement) are apparent in said table. Even though the signal data rate is 50 Gb/s (the amplifier had a bandwidth of 30 GHz), at an input voltage of 400 mVpp d_{th}^{50} has a value of 0.55 dB, which would more than likely be an acceptable performance loss in most scenarios. However, as the input voltage goes up, the loss in SNR grows significantly, getting up to 2.1 dB at an input voltage of 650 mVpp. The variation in d_{th}^{50} as a function of the input signal voltage also serves to, once again, prove how this parameter is responsible for the overall losses observed in all the previous results of this chapter.

TABLE 5.11: Characterization of the SNR loss caused by increases in input signal voltage for a $\mathrm{BER_{pre}} = 10^{-3}$.

Signal Input Voltage (mVpp)	Signal Data Rate (Gb/s)	d^{50}_{th}	d^{50}_{400}	d^{50}_{v}
400	50 Gb/s	0.55	0	0
500	50 Gb/s	0.95	0.45	0.45
600	50 Gb/s	1.55	1	0.6
650	50 Gb/s	2.1	1.55	0.55

Let us now look at how the SNR losses increase as functions of the input signal voltage. Figure 5.12 shows the evolution of d^{50}_{400} as the input signal voltage is increased. d^{50}_{v} is implicit in this image, represented by the differences between each pair of consecutive points in the curve.

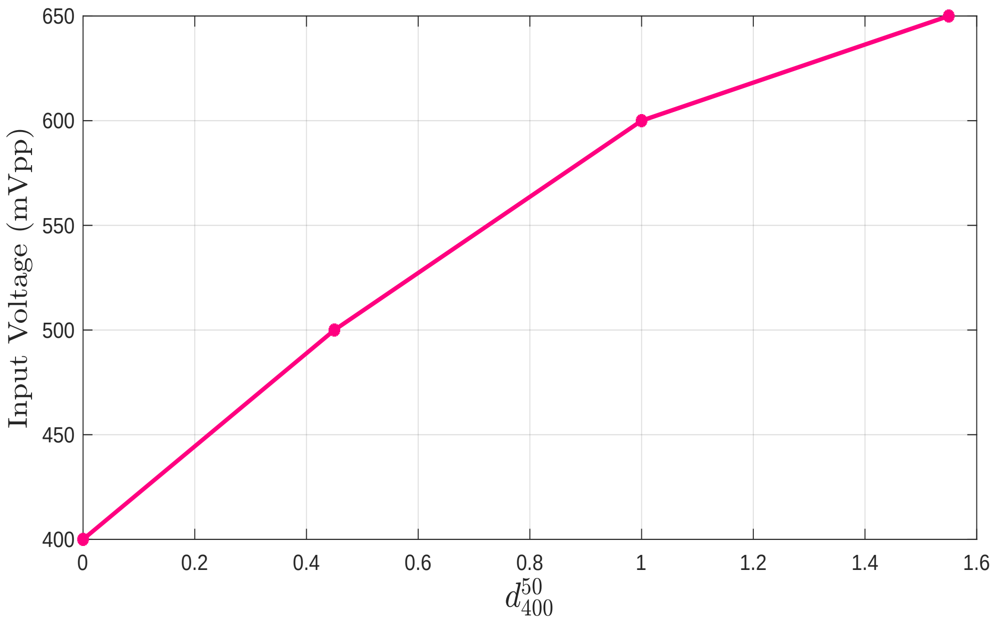


FIGURE 5.12: Evolution of d_{400} as a function of the input signal voltage.

The next step is to compute the same curve for the 56 Gb/s data in order to determine if the behaviour of the metric d_{400} as a function of the input voltage is maintained at a different data rate. Following the same procedure employed previously, the metrics d_{th}, d_{400}, and d_v are computed at a $\mathrm{BER_{pre}}$ value of 10^{-3} for the results related to the 56 Gb/s measurements. They will be distinguished from the previous metrics by the superscript indicating the higher data rate of the measurements. The values of these metrics are shown in Table 5.12. Figure 5.13 shows the evolution of d^{56}_{400} as a function of the input signal voltage alongside the same curve obtained for the 50 Gb/s data, d^{50}_{400}.

TABLE 5.12: Characterization of the SNR loss caused by increases in input signal voltage for a $\text{BER}_{\text{pre}} = 10^{-3}$.

Signal Input Voltage (mVpp)	Signal Data Rate (Gb/s)	d^{56}_{th}	d^{56}_{400}	d^{56}_{v}
400	56 Gb/s	0.75	0	0
500	56 Gb/s	1.15	0.4	0.4
600	56 Gb/s	1.75	1	0.6
650	56 Gb/s	2.65	1.9	0.9

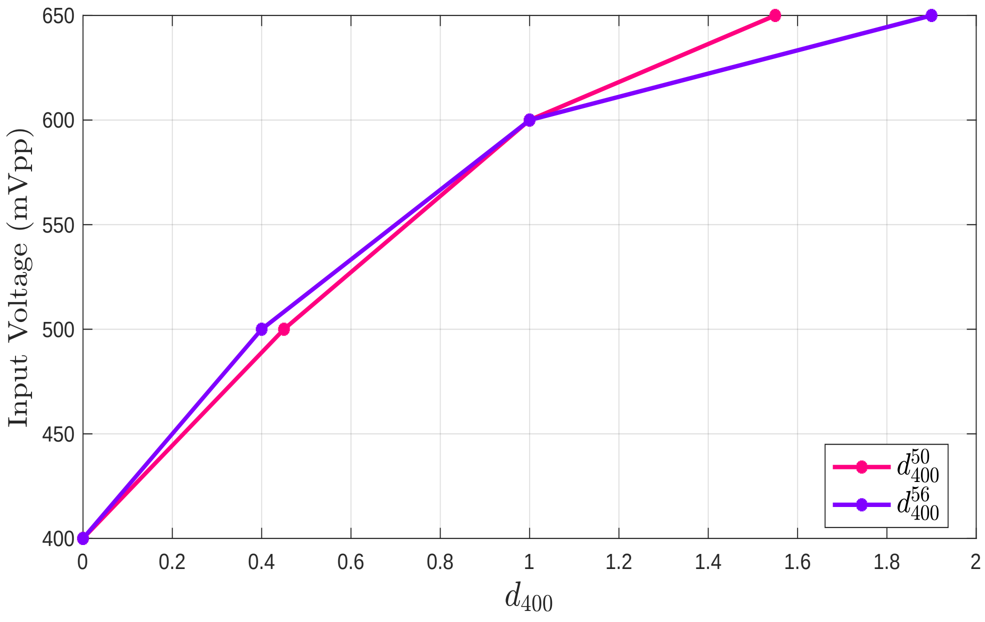


FIGURE 5.13: Evolution of d_{400} as a function of the input signal voltage.

This last figure shows how for both the 50 and 56 Gb/s data sets, the evolution of d_{400} as a function of the increasing input voltage is almost identical up until the value of 650 mVpp. At the largest measured input voltage, the value of d^{56}_{400} is 0.35 dB larger than the value obtained for d^{50}_{400}.

Earlier in this section we mentioned that given input voltages X, Y, such that $X > Y$, an increase in signal data rate within a system working at voltage X might lead to a larger SNR loss than working at voltage Y. Seeking insight regarding this matter and the discrepancy at 650 mVpp between d^{56}_{400} and d^{50}_{400} mentioned in the previous paragraph, we proceed by plotting the values of d_{th} obtained from the data rate measurements given in Tables 5.11 and 5.12. This can be seen in Figure 5.14. Taking a closer look at the curves depicted in the aforementioned figure, we observe that until a voltage of 600 mVpp (for the three smallest voltage measurements), d^{56}_{th} is merely a version of d^{50}_{th} that has been shifted towards the right of the x-axis. However, as was just observed for d_{400}, at 650 mVpp d^{56}_{th} and d^{50}_{th} differ.

If we now take a numerical approach and consider the actual differences between d^{50}_{th}

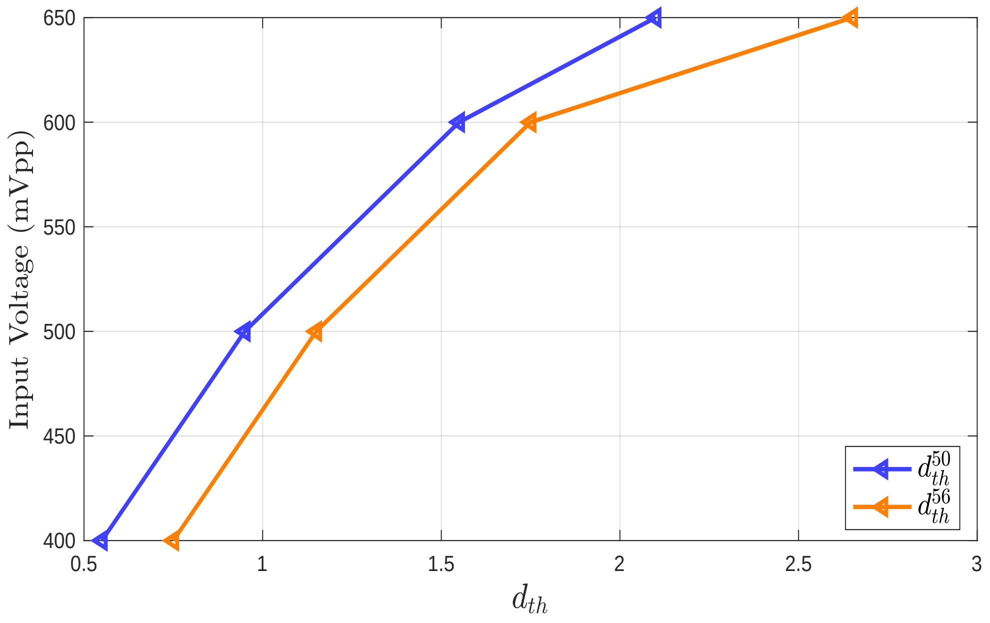


FIGURE 5.14: Computed d_{th} values for two different data rates as a function of the input signal voltage.

and d_{th}^{56}, it can be observed that for input voltages of 400, 500 and 600 mVpp, $d_{th}^{56} - d_{th}^{50}$ yields the same value of 0.2 dB. At 650 mVpp of input voltage, $d_{th}^{56} - d_{th}^{50} = 0.55$. An explanation for this phenomenon might be that for signals with an input voltage of 650 mVpp the amplifier is close to being or already outside the linear operational range, causing it to respond "worse"[5] when said signals are of a higher data rate. It is currently difficult to come up with a method to characterize this variation in amplifier response to different data rates at 650 mVpp, more lab measurements will most likely be necessary. Nevertheless, because for input voltages of 400, 500, and 600 mVpp, d_{400}^{50} and d_{400}^{56} are very similar and $d_{th}^{56} - d_{th}^{50} = 0.2$, it seems reasonable to further analyze the 400-600 mVpp range.

5.4.1.2 General model for effects caused by input signal voltages

This section will be closed by determining the nature of the relationship between the normalized Volterra kernels and the input signal peak-to-peak voltage. To do so, we must recover the kernels of the corresponding models along with specific analytic tools used in earlier sections of this chapter. All of this is provided in Tables 5.13 and 5.14. The parameters $c_{\tilde{h}_1}$ and $c_{\tilde{h}_3}$ were defined when studying the effects of the signal data rate. They served to measure the change in $\tilde{h}_1$ and $\tilde{h}_3$ from one model to the next as the rate dependant loss measuring metric d_{38} increased. They will now be used to measure those same changes but when the voltage dependant loss d_{400} increases. As previously,

[5]In this contest, the term "worse" implies that more SNR loss is observed.

negative values of $c_{\tilde{h}_1}$ or $c_{\tilde{h}_3}$ represent a decrease of the normalized kernel in question while positive values represents an increase.

Table 5.13: Metrics that define the evolution of $\tilde{h}_1$ and $\tilde{h}_3$ as functions of the input signal voltage and a data rate of 50 Gb/s.

Input Voltage (mVpp)	Signal Data Rate (Gb/s)	$\tilde{h}_1$	$\tilde{h}_3$	d_{th}^{50}	d_{400}^{50}	$c_{\tilde{h}_1}$	$c_{\tilde{h}_3}$
400	50	0.9919	0.0010	0.55	0	0	0
500	50	0.9612	0.0310	0.95	0.45	-0.0307	0.03
600	50	0.9286	0.0631	1.55	1	-0.0326	0.0321
650	50	0.8985	0.0947	2.1	1.55	-0.0301	0.0316

Table 5.14: Metrics that define the evolution of $\tilde{h}_1$ and $\tilde{h}_3$ as functions of the input signal voltage and a data rate of 56 Gb/s.

Input Voltage (mVpp)	Signal Data Rate (Gb/s)	$\tilde{h}_1$	$\tilde{h}_3$	d_{th}^{56}	d_{400}^{56}	$c_{\tilde{h}_1}$	$c_{\tilde{h}_3}$
400	56	0.9885	0.0025	0.75	0	-	
500	56	0.9582	0.0336	1.15	0.4	-0.0303	0.0311
600	56	0.9236	0.0722	1.75	1	-0.0346	0.0386
650	56	0.8633	0.1308	2.65	1.9	-0.0603	0.0586

In our endeavor to better understand the data given in the above tables, we choose to plot some of it, as is shown by Figures 5.15 and 5.16. Figure 5.15 shows, on the same graph, how $\tilde{h}_1$ and $\tilde{h}_3$ evolve as functions of d_{400}. To distinguish between the curves, the notation $\tilde{h}_i(d_{400}^x)$ is used. This notation denotes the values of the kernel $\tilde{h}_i$ associated to the losses d_{400}^x, where $i = 1, 3$ and $x = 50, 56$ represent the order of the kernel and the data rate of the signal from which the model was learned. For example, $\tilde{h}_1(d_{400}^{50})$ refers to the values of $\tilde{h}_1$ that correspond to the losses d_{400}^{50}, both parameters being given in Table 5.13.

Figure 5.15 includes the results obtained for measurements at 650 mVpp. Although at said voltage and for varying data rates, as was shown previously, the SNR losses behave very differently than for the data below 600 mVpp, this figure reflects the similarity in the behaviour of the normalized kernels from both data rate models as the rate independent loss d_{400} increases.

Because we were unable to really identify what governs the data rate related loss variations that occur at 650 mVpp, we exclude those results from the current analysis. The same plot of Figure 5.15 is shown in Figure 5.16 without the information derived from 650 mVpp input signal measurements. This figure reveals that $\tilde{h}_1(d_{400}^{56})$ is essentially a version of $\tilde{h}_1(d_{400}^{50})$ downshifted along the y-axis . The same can be said for $\tilde{h}_3(d_{400}^{56})$, except it is a version of $\tilde{h}_3(d_{400}^{50})$ upshifted along the y-axis. This is a relevant result because it shows that varying the data rate of a signal within an input voltage range

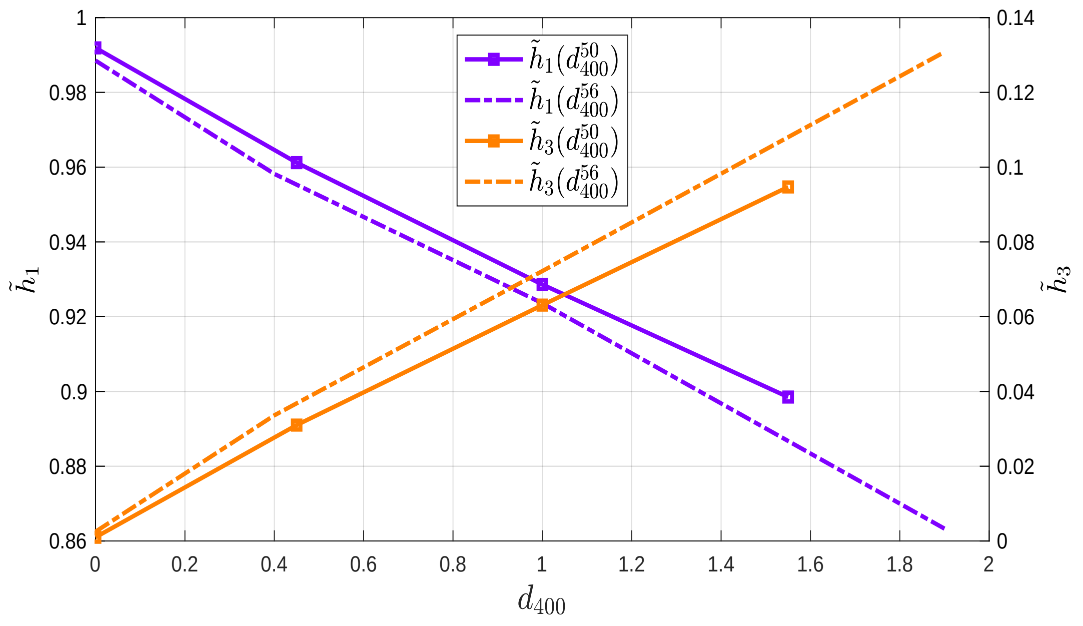


FIGURE 5.15: Behaviour of the normalized kernels $\tilde{h}_1$ and $\tilde{h}_3$ as functions of the loss d_{400}.

of 400-600 mVpp displaces the normalized kernel curves along the y-axis but it does not change it's slope. In other words, changing the signal data rate when operating in an input voltage range of 400-600 mVpp causes changes that can be linearly predicted. This means that the characteristics of a Volterra model for an input signal voltage X, such that $400 \geq X \leq 600$, and a signal data rate R, can easily be determined from an expanded version of Figure 5.16 in two simple steps:

1. Select the value of d_{400} related to the input signal voltage in question from an expanded version of Figure 5.13.

2. Find the appropriate $\tilde{h}_1$ and $\tilde{h}_3$ related to the specific value of d_{400} for that voltage and the data rate in question from a complete version of Figure 5.16.

Because of the linear behaviour observed when changing the data rate in the input signal voltage range of 400-600 mVpp, the complete versions of Figures 5.13 and 5.16 can be obtained easily by simply shifting some of the curves upwards or downwards on the respective axis depending on the value of the data rate. For example, the expanded version of Figure 5.16 is obtained by shifting the $\tilde{h}_1(d_{400}^{50})$ and $\tilde{h}_3(d_{400}^{50})$ curves upwards or downwards on the y-axis depending on the value of the data rate. These complete figures are shown in the following section.

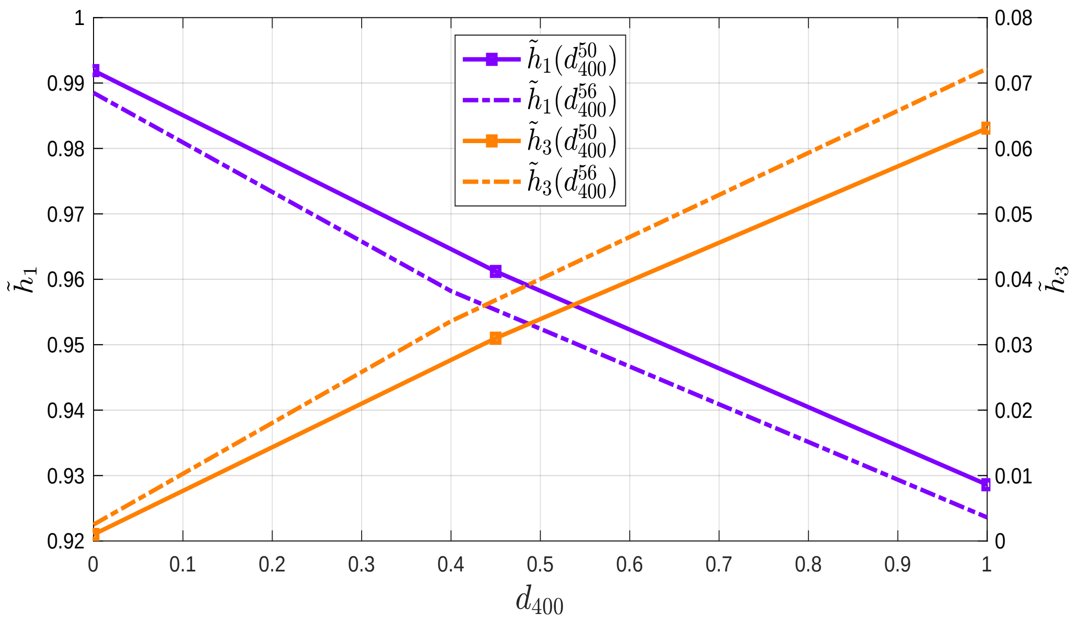


FIGURE 5.16: Behaviour of the normalized kernels $\tilde{h}_1$ and $\tilde{h}_3$ as functions of the loss d_{400}. The data obtained from 650 mVpp input signal measurements has been excluded.

5.5 Final considerations for model variability in relation to amplifier input signal characteristics

This final section briefly summarizes the main topics that have been discussed in this chapter, which primarily focused on the changes in amplifier behaviour and learned Volterra models that occur when each input signal characteristic is varied independently. In addition, it also explains an inference technique that can be used to obtain amplifier behavioural Volterra models without performing additional lab measurements.

5.5.1 Impact of varying individual input signal characteristics:

1. Data rate: The data rate of the input signal to the amplifier directly affects the response of the amplifier and the subsequent Volterra model that is learned. The measurements and analysis conducted in this chapter have shown that changes in the data rate of the signal will cause different effects depending on its peak-to-peak voltage: If the input voltage is within the range of 400-600 mVpp, increases or decreases in data rate will affect the amplifier response in a predictable manner. Outside of this range (the specific measurement was conducted at 650 mVpp), the amplifier responds differently to each data rate in a way that is unpredictable given the measurementes at our disposal.

2. Bit Pattern length: The length of the bit pattern that compromises the data sequence of the input signal will determine the accuracy with which a model of the amplifiers behaviour can be learned. Longer length bit patterns will result in more accurate models, while shorter length sequences have a larger representation error. However, there is a trade-off between sequence length and analysis complexity, as longer sequences become increasingly difficult to handle. All in all, the representation error has been shown to be small, and a pattern length of $2^{15} - 1$ bits seems appropriate to achieve a good compromise between the representation penalty and analysis complexity.

3. Peak-to-peak Voltage: The peak-to-peak voltage of the input signal is the most influential input signal parameter with regard to the content of nonlinearity in the amplifier response. Higher input signal peak-to-peak voltages result in an increasingly nonlinear amplifier response, which results in considerable system performance degradation. For example, changing the input voltage from 400 mVpp to 600 mVpp increases the required SNR to maintain BER performance by around 1.5 dB, which is quite significant in comparison to the 0.4 dB loss caused by a 6 Gb/s increment of the data rate. If the input signal voltage is kept within the range of 400 mVpp to 600 mVpp, the change in amplifier behaviour related to modifying the data rate or the voltage itself, and the losses caused by these changes, occur in a predictable manner. At 650 mVpp of input signal voltage, the amplifier responds in a way that we have not been able to ascertain.

5.5.2 Regression tool to obtain amplifier behavioural Volterra models for the input signal peak-to-peak voltage range of 400-600 mVpp

The goal of this thesis is to develop ways to model and study the nonlinear content of an amplifier's behavioural response to different input stimuli. The modelling of the amplifier has been accomplished using truncated Volterra series, but the models themselves provide little information about the behaviour of the device. Throughout the text, two main analysis tools have been derived to obtain information from these models. First off, we have the normalized Volterra kernels given in (3.5). The second tool is compromised by the metrics that measure the separation between the BER_{pre} curves that have been studied in this chapter.

The normalized Volterra kernels provide a quick and easy way of evaluating how nonlinear the amplifier is under specific operating conditions by simply looking at the values of the linear and cubic normalized kernels. The separation metrics, such as d_{th} or d_{400}, determine the losses in performance in comparison to different benchmarks. By relating both of these tools and using measurement-based regression, we will be able to infer models for amplifier behaviour within the 400-600 mVpp range for various data rates. Moreover, we will be able to compute the system performance loss caused by said device behaviour. Most importantly, this technique will allow us to obtain this information without having to conduct any more lab measurements.

Before explaining this regressive methodology, it must be stated that this technique is untested. It is in no way suggested that it may serve in a different scenario or for different amplifier models, and it might not even work for different amplifiers of the same kind. The sole purpose of including this method in the thesis is to document the first steps that have been taken down the path of creating an inference-based nonlinear amplifier model that could potentially characterize devices without the need for lab measurements.

The procedure works as follows:

1. Based on the specific input signal voltage V and data rate R, extract the corresponding value of d_{th} from Figure 5.17. The associated system peformance BER_{pre} curve is readily obtained by shifting $P_{b,2}$ along the right of the x-axis by the specific value of d_{th}.

2. Based on that same value of d_{th}, obtain $d_{400} = d_{\text{th}}^{R}(V) - d_{\text{th}}^{R}(400)$ using Figure 5.17, where $d_{\text{th}}^{R}(V)$ and $d_{\text{th}}^{R}(400)$ represent the values of d_{th}^{R} at voltages V and 400 mVpp for the data rate R, respectively. Based on the value extracted for d_{400}, obtain $\tilde{h}_1$ and $\tilde{h}_3$ from Figure 5.18 for the data rate R.

To close out this chapter we will show the validity of this prediction tool by estimating the BER_{pre} curve and normalized kernels for a set of specific operating conditions measured during the first lab experiment and then comparing these estimations to the simulated BER_{pre} results obtained for those same measurements. As was mentioned previously, it has only been tested based on the data that was recorded from the single amplifier we measured. These estimations will be made for the measurements shown in Table 5.15. The estimated normalized kernels, denoted by $\ddot{h}_i$ where $i = 1, 3$, are compared to the learned normalized kernels in Table 5.16. The estimated and simulated BER_{pre} results are shown in Figure 5.19.

As can be seen by the almost negligible difference between the estimated and simulated results shown in Table 5.16 and Figure 5.19, this method constitutes a valid way of obtaining the normalized kernels and losses in performance caused by the amplifier without the need for lab data.

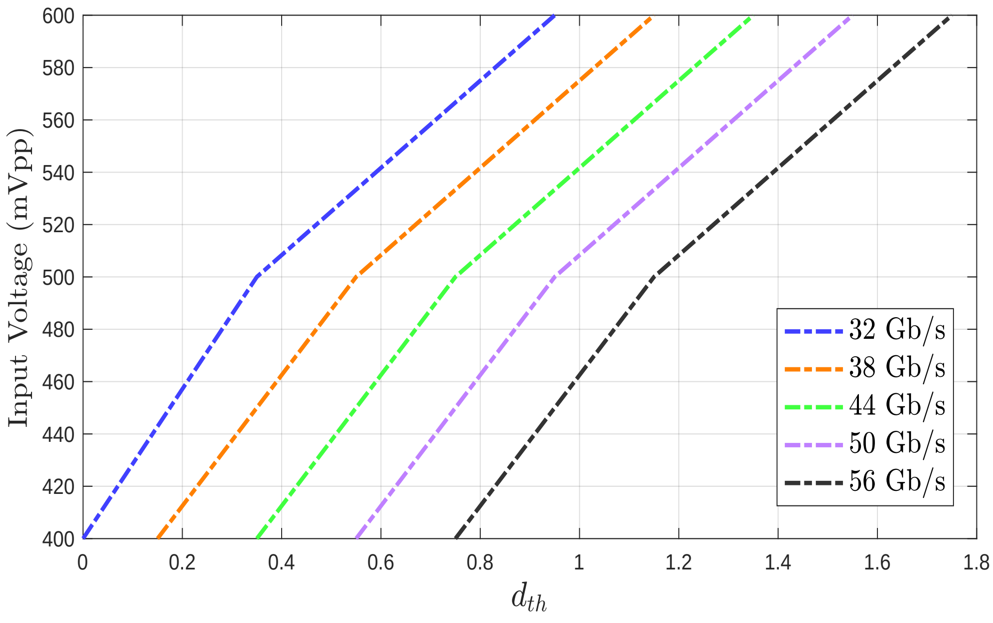


FIGURE 5.17: SNR loss represented by d_{th} for various input signal data rates and the input signal voltage range 400-600 mVpp.

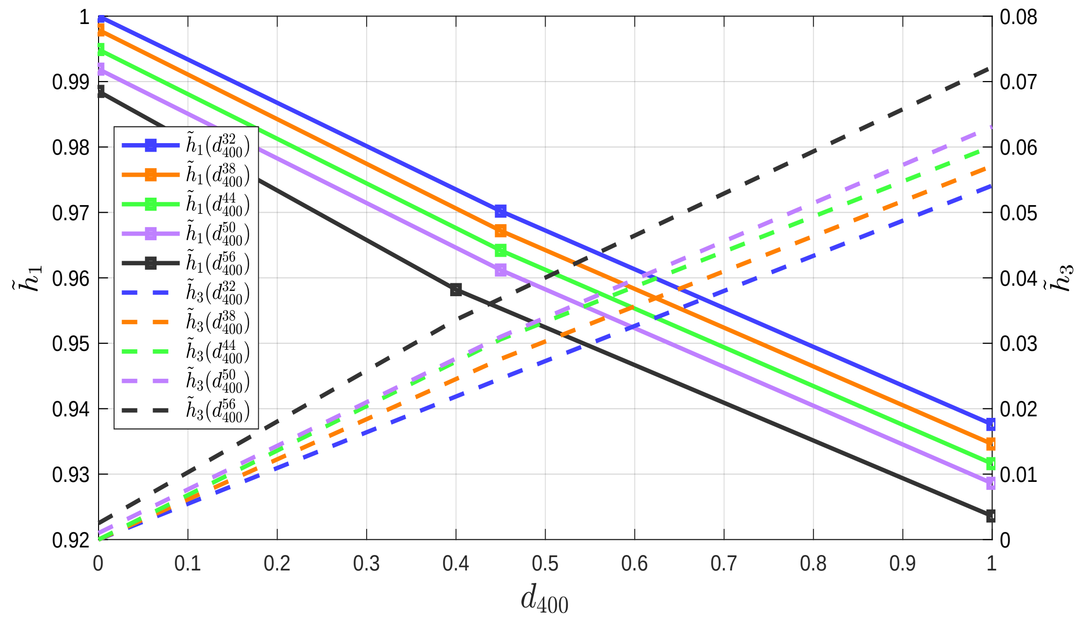


FIGURE 5.18: Behaviour of the normalized kernels $\tilde{h}_1$ and $\tilde{h}_3$ as functions of the loss d_{400} for various input signal data rates. The superscript denotes the data rate associated to each curve.

TABLE 5.15: Measurements from the first experiment whose effects will be estimated.

Modulation	Rate (Gb/s) (bits)	Pattern Length	Voltage (mVpp)
NRZ	32	$2^{15}-1$	450
NRZ	50	$2^{15}-1$	525

TABLE 5.16: Estimated and simulated normalized kernel values.

Rate (Gb/s)	Voltage (mVpp)	$\tilde{h}_1$	$\tilde{h}_3$	$\ddot{h}_1$	$\ddot{h}_3$
32	450	0.9878	0.0011	0.9890	0.0010
50	525	0.9512	0.0425	0.9520	0.0400

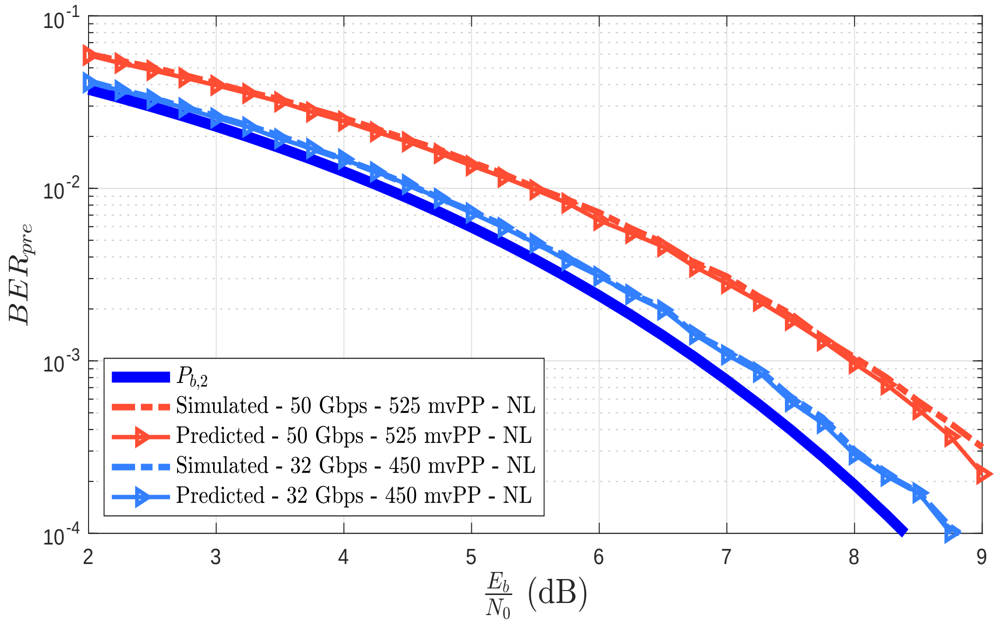


FIGURE 5.19: Comparison between estimated and simulated BER_{pre} curves.

Chapter 6

Future Work: Alternate Prediction Metrics

In Chapter 4, the concept of optimizing a communications system for a sufficiently small value of the BER before FEC decoding was introduced. This idea operates based on assuming that below said sufficiently small uncoded BER, subsequent values of the BER before FEC decoding are guaranteed to be reducible to the desired BER after FEC decoding values based on previously verified FEC implementations. This is known as the *FEC limit paradigm*.

It has recently been shown in [14] and [19] that the *FEC limit paradigm* can lead to the underestimation of the spectral efficiency, that it is not applicable to SD-FEC decoders, and that better prediction metrics exist. The search for different prediction metrics not based on this paradigm has been and continues to be an open research problem in optical communications. A variety of methods to predict the post-FEC BER instead of using $\mathrm{BER}_{\mathrm{pre}}$ have been derived by studying the achievable rates of systems that use FEC coding. These prediction metrics might potentially allow to study the impact caused by the nonlinear models on the FEC codecs of the communications system. This chapter serves as an introduction to some of the post-FEC BER prediction methods that have been derived in recent times based on the achievable rates of FEC optical communications systems. Application of these measures to the research that has been explained previously in this text presents an interesting approach for future work on the topic of this thesis.

6.1 Mutual Information: Basic Concepts

To design a successful FEC coding scheme for transmission through a communication channel, we will require a codebook, an encoder, and a decoder. The encoder will take an information sequence D and associate it to a codeword X by means of a one-to-one mapping procedure. The decoder receives a random sequence Y that results from X being corrupted by the channel, and will have to guess the sequence D using a deterministic rule $g : \hat{D} = g(Y)$, that maps noisy channel observations onto information sequences. The codebook $\mathbf{C}$ is the set of all possible codewords.

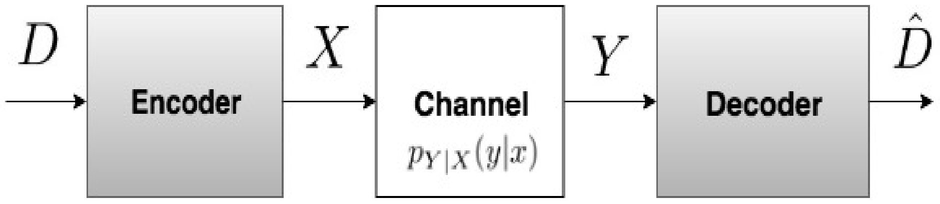


FIGURE 6.1: Discrete Memoryless Channel.

Let us now consider the discrete input discrete output memoryless communication channel shown in Figure 6.1, where $p(y|x)$ is the probability mass function that represents the transition probability of the channel (the probability of observing the output symbol y when symbol x is sent), X is the channel input, and Y is the channel output. The channel is said to be memoryless because the probability distribution of the output depends only on the input at that time and is conditionally independent of previous channel inputs or outputs. We define the metric known as the mutual information, denoted by $I(X;Y)$, as a measure of the amount of information that Y contains about X. In other words, it quantifies the knowledge the channel output provides with regard to the input. The generic expression for the mutual information is given in (6.1). It is extensively used in many applications. For example, the capacity of the DMC shown in Figure 6.1, C_{DMC}, is expressed in terms of the mutual information as shown by (6.2). The complete derivation of C_{DMC} is provided in [20].

$$I(X;Y) = \mathbb{E}_{X,Y}\left[\log_2 \frac{p_{Y|X}(y|x)}{p_Y(y)}\right] \tag{6.1}$$

$$C_{\text{DMC}} = \max_{p_x(x)} I(X;Y) \tag{6.2}$$

Shannon's Capacity Theorem defines the capacity of a communication channel as the maximum rate at which a system can send information over the channel and recover the information at the output with a vanishing probability of error. Therefore, we can say a rate R_s is achievable, if a system transmitting information at said rate across a given channel, can function with arbitrarily low probability of error. This can be stated in a simple expression as $R_s \leq C$, where C represents the capacity of the channel in question and R_s represents the rate of the system. The rate of a system, in information bits per symbol, is given by the combined rate of its FEC encoder and its modulator as $R_s = mR_{\text{FEC}}$, where $m = \log_2 M$ and M represents the cardinality of the chosen discrete constellation, and R_{FEC} is the rate of the chosen FEC code in information bits per code symbol. For a communications system using FEC to achieve arbitrarily low probability of error, the condition $R_s \leq C$ must be met in the limit of large FEC block length. This can be written mathematically as

$$\lim_{n\to\infty} P_e = 0.$$

6.2 Mutual Information: Simulations

It has been shown in [21] that the MI is a superior metric, in comparison to the pre-FEC BER, when it comes to predicting the post-FEC BER. As such, it is advantageous to derive an expression for (6.1) particularized for our simulations so that its computation can be included in the Matlab framework. Having provided the necessary definitions in the previous subsection, we are in a position to derive such an expression for the $I(X;Y)$.

We begin by summarizing the derivations shown in [14]. Let $\mathbf{B} = [B_1, \ldots, B_m]$ be a random vector representing the transmitted bits $[c_{1,l}, \ldots, c_{m,l}]$ at any time instant l, which are mapped to the corresponding symbol $X_l \in \mathbb{X}$ with $l = 1, 2, \ldots, n$. The symbols X_l belong to a discrete M-ary constellation $\mathbb{X}$ such that $M = 2^m$. Assuming a memoryless channel modeled by the transition probability $f_{Y|X}(y|x)$, an FEC code with codebook $\mathbf{C}$, and equally likely codewords, we define the maximum-likelihood (ML) receiver as the receiver that will choose the transmitted codeword based on an observed sequence $[y_1, \ldots, y_n]$ according to the rule

$$\mathbf{c}^{\text{ml}} = \underset{\mathbf{c}\in\mathbf{C}}{\operatorname{argmax}} \sum_{l=1}^{n} \log f_{Y|\mathbf{B}}(y_l|c_{1,l}, \ldots, c_{m,l}). \tag{6.3}$$

As per Shannon's channel coding theorem, reliable transmission with the ML decoder given in (6.3) is possible at arbitrarily low error probability if $R_s = R_{\text{FEC}}m \leq C$, where C is the capacity of the channel in consideration. The mutual information consitutes an achievable rate R_s for a system using a discrete constellation $\mathbb{X}$ that transmits through a channel with transition probability $f_{Y|X}(y|x)$ and uses an ML decoder. The expression for $I(X;Y)$ given in (6.1) can be expressed, based on this system, as

$$I(X;Y) = \sum_{x\in\mathbb{X}} P_X(x) \int_{\mathbb{C}} f_{Y|X}(y|x) \log_2 \frac{f_{Y|X}(y|x)}{f_Y(y)} dy, \tag{6.4}$$

where $\mathbb{C}$ denotes the set of complex numbers. The MI given in (6.4) can be approximated by making use of Monte Carlo integration. Following the work conducted in [22], Monte-Carlo integration can be used to approximate an integral via a finite sum [23], namely

$$\int_{\mathbb{C}} f_R(r)g(r)dr \approx \frac{1}{D}\sum_{n=1}^{D} g(r_n), \tag{6.5}$$

where r is an arbitrary random variable defined over $\mathbb{C}$. In (6.5), $g(r) : \mathbb{C} \to \mathbb{R}$, where $\mathbb{R}$ and $\mathbb{C}$ denote the set of real and complex numbers, respectively, is an arbitrary real-valued function and r_n with $n = 1, \ldots, N$ are samples from the distribution $f_R(r)$. In light of (6.5), a Monte-Carlo approximation of (6.6) can be readily obtained. Considering that $\log_2(r) : \mathbb{R}^+ \to \mathbb{R}$, having applied $\mathbb{R}^+ \in \mathbb{C}$, where r is now defined as $r = \frac{f_{Y|X}(y|x)}{f_Y(y)}$, we obtain a Monte-Carlo estimate of (6.6) as,

$$I(X;Y) \approx \sum_{x\in\mathbb{X}} P_X(x) \left[\frac{1}{n}\sum_{l=1}^{n} \log_2 r_l\right], \tag{6.6}$$

where r_l with $l = 1, 2, \ldots, n$ are samples of the random variable r distributed according to $f_R(r)$. If we now substitute $r_l = \frac{f_{Y|X}(y_l|x)}{f_Y(y_l)}$ into (6.6), we obtain

$$I(X;Y) \approx \frac{1}{n}\sum_{x\in\mathbb{X}} P_X(x) \sum_{l=1}^{n} \log_2 \frac{f_{Y|X}(y_l|x)}{f_Y(y_l)}, \tag{6.7}$$

where y_l with $l = 1, 2, \ldots, n$ are samples of the random variable Y conditioned on the transmitted bits X. We can rewrite the above expression as

$$I(X;Y) \approx \frac{1}{n}\sum_{i=1}^{M} P_X(x_i) \sum_{l=1}^{n} \log_2 \frac{f_{Y|X}(y_l^{(i)}|x_i)}{f_Y(y_l^{(i)})}, \tag{6.8}$$

with $y_l^{(i)}$ with $l = 1, 2, \ldots, n$ are samples of the random variable Y conditioned on $X = x_i$, where $i = 1, \ldots, M$. Knowing that in our present framework $M = 2$, (6.7) can be specifically tailored to the simulations considered in this text[1] as

[1]This particularization is performed under the assumption that our system uses an ML decoder. As will be shown in the next subsection, because binary modulation is used, said assumption is correct.

$$
\begin{aligned}
I(X;Y) &\approx \frac{1}{n}\sum_{i=1}^{2} P_X(x_i)\sum_{l=1}^{n}\log_2 \frac{f_{Y|X}(y_l^{(i)}|x_i)}{f_Y(y_l^{(i)})} \\
&= \frac{1}{n}\Bigg[P_X(x_1)\sum_{l=1}^{n}\log_2 \frac{f_{Y|X}(y_l^{(1)}|x_1)}{f_Y(y_l^{(1)})} + \dots \\
&\quad \dots + P_X(x_2)\sum_{l=1}^{n}\log_2 \frac{f_{Y|X}(y_l^{(2)}|x_2)}{f_Y(y_l^{(2)})}\Bigg],
\end{aligned}
\tag{6.9}
$$

where x_1 and x_2 represent the symbols of a 2-PAM constellation (binary modulation). Figure 6.2 shows how (6.9) would be computed in the Matlab simulation. The probability density functions $f_{Y|X}(y_l|x_i)$ and $f_Y(y_l)$ are obtained by estimating their respective mean and variance from the simulated data, and then, based on the Gaussian nature of the channel, combining them with the probability density function of the normal distribution[2] to create the corresponding probability density functions.

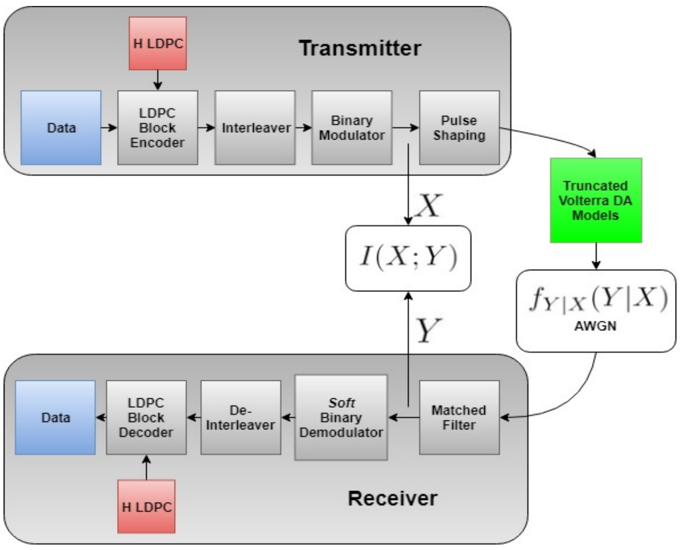


FIGURE 6.2: Computation of $I(X;Y)$.

6.2.1 The BW receiver & the GMI

Figure 4.1 shows the functioning of the communications system that is implemented in Matlab. Initially, the receiver performs soft demodulation by computing log-likelihood ratios, following it with binary SD decoding. This detection architecture is known as

[2]The Gaussian distribution, also referred to as the normal distribution, has a probability density function that is given by $f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, where μ and σ are the mean and variance of X, respectively, and X is a normally distributed random variable. When working with multiple variables; considering both quadratures instead of just the in-phase component for example, the univariate normal pdf must be generalized to include multiple variables. Such a distribution is known as the multivariate normal distribution and is given by $f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d|\omega|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top\omega^{-1}(\mathbf{x}-\mu)\right)$, where $\mathbf{X}$ is a $1\times d$ random vector, μ is the $1\times d$ mean vector of random vector $\mathbf{X}$, and ω is the $d\times d$ covariance matrix of $\mathbf{X}$.

the BW (bit-wise) decoder. In typical transceiver schemes that employ a BW decoder (commonly used in coherent optical communication systems) an additional outer FEC coding stage is included. This stage adds an outer FEC encoder prior to the LDPC block encoder and an HD-FEC outer decoder after the binary SD decoder. This is shown in Figure 6.3.

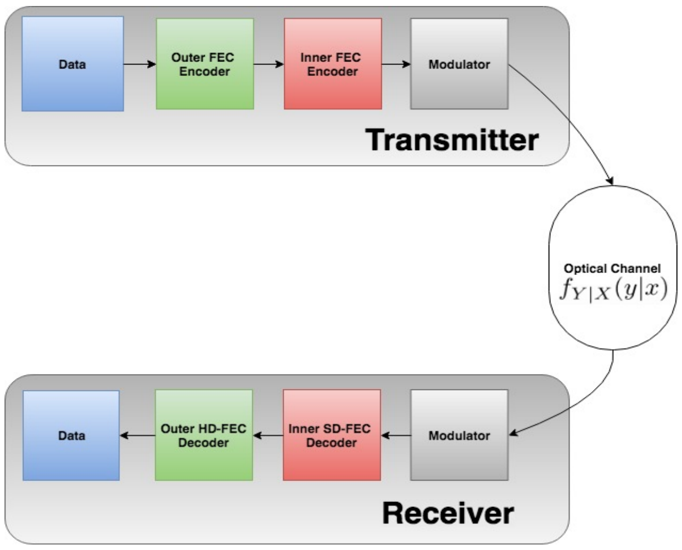


FIGURE 6.3: Architecture of a BW transceiver. It is a simplified representation: multiple stages, such as the interleaving or pulse shaping blocks, have been omitted to increase clarity.

Let us assume for the purpose of the following derivation that our system is a generic BW transceiver that uses outer and inner FEC coding. Once again, the following expressions are fully derived in [14].

The BW decoder can be defined by the BW decoding rule

$$\mathbf{c}^{\text{bw}} = \underset{\mathbf{c}\in\mathbf{C}}{\text{argmax}} \sum_{l=1}^{n} \log \prod_{k=1}^{m} f_{Y|B_k}(y_l|c_{k,l}). \tag{6.10}$$

As was considered in the previous subsection for the derivation of a generic expression for the mutual information, in (6.10), $\mathbf{B} = [B_1, \dots, B_m]$ is a random vector representation of the transmitted bits $[c_{1,l}, \dots, c_{m,l}]$ at any time instant l, which are mapped to the corresponding symbol $X_l \in \mathbb{X}$ with $l = 1, 2, \dots, n$. The symbols X_l belong to an M-ary constellation $\mathbb{X}$ such that $M = 2^m$. In the specific cases considered in this report, because strictly binary modulation (2-PAM or BPSK) is used, the equality $[c_{1,l}, \dots, c_{m,l}] = c_{1,l} = c_l$ holds. This means that index k can be dropped from (6.10), resulting in expression (6.11), where B is now a random variable representing transmitted bit c_l.

$$\mathbf{c}^{\text{bw}}_{\text{BPSK}} = \underset{\mathbf{c}\in\mathbf{C}}{\text{argmax}} \sum_{l=1}^{n} \log f_{Y|B}(y_l|c_l), \tag{6.11}$$

In order to maintain generality, and following the derivations shown in [14], let us assume that we are using constellation $\mathbb{X}$ with $M > 2$ and that our BW decoder is defined by (6.10). If we compare the expression shown in (6.10) to the the optimal maximum-likelihood detection rule given in (6.3), certain differences can be appreciated. We stated previously that the mutual information given in (6.4) represents an achievable rate for ML receivers. In general, because $\mathbf{c}^{\text{bw}} \neq \mathbf{c}^{\text{ml}}$, the $I(X;Y)$ will not be an achievable rate for BW decoders. As a result, it will be necessary to derive an achievable rate for BW decoders.

An achievable rate for BW decoders can be found by casting them in the framework of a mismatched decoder. This can be done as follows. The channel under consideration is a symbol-wise channel defined by the channel law $f_{Y|X}(y|x)$. If we consider the symbol-wise metric

$$q(\mathbf{b}, y) = \prod_{k=1}^{m} f_{Y|B_k}(y|b_k), \tag{6.12}$$

the BW rule given in (6.10) can be expressed as

$$\mathbf{c}^{\text{bw}} = \operatorname*{argmax}_{\mathbf{c}\in\mathbf{C}} \sum_{l=1}^{n} \log q(\mathbf{b}_l, y_l)), \tag{6.13}$$

where $\mathbf{b}_l = [c_{1,l}, \ldots, c_{m,l}]$. In this context, the ML decoder given in (6.3) can be viewed as a mismatched decoder with a metric $q(\mathbf{b}_l, y_l) = f_{Y|X}(y_l|x_l)$ that is matched to the symbol-wise channel. Under this same guise, the BW decoder uses a metric matched to the bits $f_{Y|B_k}(y|b_k)$ instead of the channel symbols, which is where its name comes from.

It has been shown in [24] and [25] that a metric known as the *generalized mutual information*, GMI in short, constitutes an achievable rate for a BW decoder. It represents a bound on the number of bits/symbol that can be reliable transmitted through a channel. Moreover, [14] shows how the GMI is a superior prediction metric when it comes to predicting BER_{post} for BW transceivers than both the MI and BER_{pre}. A general expression for the GMI is shown below as is defined in [24],

$$\text{GMI} = \max_{s\geq 0} \mathbb{E}_{\mathbf{B},Y}\left[\log_2 \frac{q(\mathbf{B}, Y)^s}{\sum_{\boldsymbol{b}\in\mathbb{B}^m} P_{\mathbf{B}}(\boldsymbol{b}) q(\boldsymbol{b}, Y)^s}\right], \tag{6.14}$$

The GMI expression in (6.14) is general in the sense that it holds for any metric $q(\mathbf{b}, y)$ and for any symbol distribution $P_{\mathbf{B}}(\mathbf{b})$. This expression can be further simplified by making two assumptions. First, that the bits $[B_1, \ldots, B_m]$, represented by the random vector $\mathbf{B}$, are independent. Second, that the receiver uses the BW metric defined in (6.12). Both of these assumptions are valid for our simulation scenario. Considering independent bits $[B_1, \ldots, B_m]$ is valid for any encoder that induces a uniform symbol distribution (which is the case we consider in this paper), and is valid for any constellation and labeling. The BW metric defined in (6.12) is used when log likelihood ratios are computed for each bit independently, which also happens to be the case (more details are provided later in this section). Therefore, for the BW metric in (6.12) and assuming independent bits $B_1, \ldots, B_m$, where $\mathbb{B}$ is the set of possible bit sequences, (6.14) can be expressed as [24]

$$\begin{aligned}\text{GMI} &= \max_{s\geq 0} \sum_{k=1}^{m} \mathbb{E}_{B_k,Y}\left[\log_2 \frac{f_{Y|B_k}(Y|B_k)^s}{\sum_{\boldsymbol{b}\in\mathbb{B}} P_{B_k}(b) f_{Y|B_k}(Y|b)^s}\right] \\ &= \sum_{k=1}^{m} \mathbb{E}_{B_k,Y}\left[\log_2 \frac{f_{Y|B_k}(y|b_k)}{\sum_{b\in\mathbb{B}} P_{B_k}(b) f_{Y|B_k}(y|b)}\right] \\ &= \sum_{k=1}^{m} I(B_k;Y),\end{aligned} \tag{6.15}$$

where the optimization over s in this case gives $s = 1$. A further expanded expression of (6.15) is provided in [22]. Up to this point in the letter, the definition of the GMI has been made in terms of the channel observations Y. Recalling the architecture of the BW receiver shown in Figure 6.3, we know that they usually include an inner SD-FEC decoder. Typically, these decoders will operate based on information units known as ‘soft bits’, also referred to as L-values or log-likelihood ratios (LLRs). These soft bits are real numbers whose magnitude represents the certainty the decoder has regarding their actual ‘hard’ value. An L-value close to zero means that the decoder is uncertain regarding the ‘hard’ value of the bit in question. In contrast, a large and positive LLR implies that we are certain the transmitted code bit was a one. If the LLR value is large and negative, we are sure the transmitted code bit was a zero. Since LLRs embody another way of representing information, the GMI can be redefined in terms of these L-values. The soft binary demodulator computes log-likelihood ratios as

$$L_k = L_k^{\text{apo}} - L_k^{\text{apri}} = \log \frac{\sum_{x\in\mathbb{X}_k^1} P_{X|C_k}(x|1) f_{Y|X}(y|x)}{\sum_{x\in\mathbb{X}_k^0} P_{X|C_k}(x|0) f_{Y|X}(y|x)}, \tag{6.16}$$

where $\mathbb{X}_k^b$ is the set of constellation symbols labeled by bit $b \in \mathbb{B} = \{0, 1\}$ at bit position $k \in \{1, \ldots, m\}$, L_k^{apo} are the *a posteriori* L-values computed as shown in (4.5), $L_k^{\text{apri}} = \log \frac{P_{C_k}(1)}{P_{C_k}(0)}$ are the *a priori* L-values, Y is a random variable representing the channel observations, and $f_{Y|X}(y|x)$ is the probability density function of the channel. If we consider that all symbols are equally likely (as is the case), (6.16) can be written as

$$L_k = \log \frac{\sum_{x \in \mathbb{X}_k^1} f_{Y|X}(y|x)}{\sum_{x \in \mathbb{X}_k^0} f_{Y|X}(y|x)}. \tag{6.17}$$

It is shown in [25] that when the L-values are calculated following (6.16), $I(B_k;Y) = I(B_K;L_K)$, and thus the GMI given in (6.15) becomes

$$\text{GMI} = \sum_{k=1}^{m} I(B_k;L_k). \tag{6.18}$$

It is important to note that the equality given in (6.18) only holds when the LLRs are obtained based on the exact computations shown in (6.16) and (6.17). When approximations of these LLRs are used to alleviate the computational complexity of (6.16), the equality $\sum_{k=1}^{m} I(B_k;Y) = \sum_{k=1}^{m} I(B_k;L_k)$ will no longer hold, and there will be a loss in achievable rate. The most commonly used approximation of the L-values is the max-log approximation, which is discussed at length in [26].

Regardless of the computation of the L-values (exact or approximated) the expression of the GMI given in (6.15) can be approximated via Monte Carlo integration [24]. This is shown in (6.19) where we have used the property $I(A|B) = H(A) - H(A|B)$, $\lambda_{k,b}^{l}$ with $l = 1, 2, \ldots, n$ are i.i.d random variables distributed according to the PDF of the L-values $f_{L_k|B_k}(\lambda|b)$, and $H_b(p)$ is the binary entropy function.

$$\begin{aligned} \text{GMI} \approx& \sum_{k=1}^{m} H_b(P_{B_k}(0)) \\ &- \frac{1}{n} \min_{s \geq 0} \sum_{k=1}^{m} \sum_{b \in \mathbb{B}} P_{B_k}(0) \sum_{l=1}^{n} \log_2 \left(1 + e^{s(-1)^b \lambda_{k,b}^{l}}\right). \end{aligned} \tag{6.19}$$

Prior to particularizing the estimate in (6.19) to our simulation scenario, it is interesting to consider the differences between the achievable rates defined in (6.7) and (6.19). Earlier in this section, we stated that the $I(X;Y)$ constitutes an achievable rate for ML decoders. We also mentioned that it does not represent an achievable rate for BW decoders. Following this, the GMI was introduced as a metric that represents an achievable rate for BW decoders. Although the GMI has not been proven to be the largest achievable rate for a BW receiver[3], it is known for predicting the performance of

[3]A different rate, known as the LM rate, has been studied in [27]. In addition, in the case where unequally likely constellation points are allowed, a new achievable rate has been derived in [28].

systems (like the one considered in this letter) that use capacity-approaching SD-FEC codes quite well.

In general, $I(X;Y) \geq \text{GMI}$ [24], where the rate penalty $I(X;Y) - \text{GMI}$ is understood as the penalty caused by the use of the suboptimal bit-wise decoder. The decoding is suboptimal because the bit-wise decoding rule (6.10) ignores the dependency between the bits in a symbol (it assumes they are independent). In view of this fact, the GMI in (6.15) and (6.18) can be interpreted as a sum of unconditional bit-wise mutual informations.

A set of experimental results that showcase the rate penalty of using (suboptimal) BW decoding, or in other words, the difference between the achievable rates defined in (6.7) and (6.19), is provided in [2]. Figure 6.4 portrays the MI and GMI of a DP (dual polarization) system as a function of the SNR for different modulation orders. It is a copy of the second figure included in [2], having been obtained by replicating and simulating the experimental conditions of [2] using a Matlab simulation. In addition, Figure 6.4 serves as a visual example of the penalty present in the GMI relative to the MI incurred due to suboptimal decoding. In [2], the GMI is also shown to be an appropriate metric for selection of the optimum modulation order of a system as a function of the SNR. Moreover, said work introduces a normalized version of the GMI, which can be used to obtain the minimum required FEC code rate to achieve the maximum number of bits per symbol that can be reliably transmitted through a channel at a specific SNR.

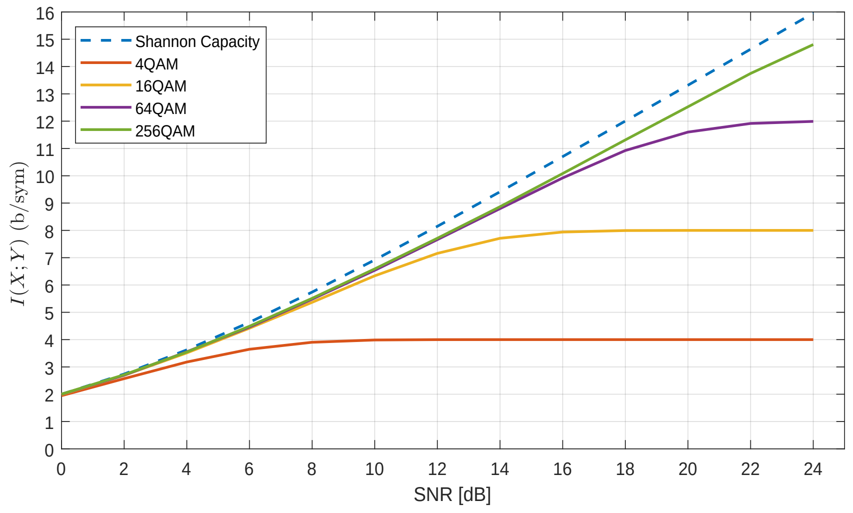
Having obtained a generic estimate for the GMI of a BW communications scheme, we can proceed by obtaining a specific version of (6.19) for the system considered in this letter. Given that our Matlab simulation operates in the real (we only consider the in-phase signal component) memoryless AWGN channel, the L-values given in (6.17) can now be computed as

$$L_k = \log\left[\frac{\sum_{x\in\mathbb{X}_k^1}\exp(\frac{-|y-x|^2}{\sigma_z^2})}{\sum_{x\in\mathbb{X}_k^0}\exp(\frac{-|y-x|^2}{\sigma_z^2})}\right], \tag{6.20}$$

where σ_z is the variance of the Gaussian noise added in the channel. For given sequences of mn transmitted bits $c_{k,l}$ and mn L-values $\lambda_{k,l}$ computed via (6.17) for $k = 1, \ldots, m$ and $l = 1, \ldots, n$, the GMI in (6.19) can be estimated as

$$\text{GMI} \approx m - \frac{1}{n}\sum_{k=1}^{m}\sum_{l=1}^{n}\log_2\left(1 + e^{(-1)^{c_{k,l}}\lambda_{k,l}}\right), \tag{6.21}$$

where the optimization over s is obtained for $s = 1$ once again. The minimization of (6.19) will only occur for $s = 1$ if the L-values are calculated as shown in (6.17).



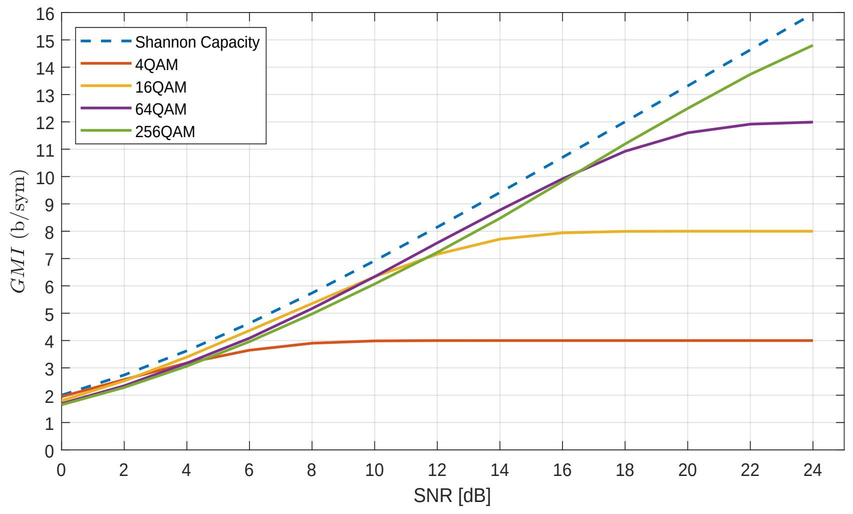


FIGURE 6.4: MI and GMI as functions of the SNR for the DP-mQAM optical transceiver experimental scenario used in [2]. The MI and the GMI values are twice as large as they would be for a single channel system like the one considered in this paper.

Computation of the L-values based on inexact approximations will cause (6.19) to be optimized for $s \neq 1$. This follows from [24]. Finally, we can further simplify these expressions by applying the specific conditions of the framework considered in this paper.

Recalling that in our simulations $m = 1$ holds, the ML decoder rule given in (6.3) can be written as

$$\mathbf{c}^{\text{ml}}_{\text{BPSK}} = \underset{\mathbf{c} \in \mathbf{C}}{\text{argmax}} \sum_{l=1}^{n} \log f_{Y|B}(y_l | c_l). \tag{6.22}$$

If we compare (6.22) to the expression for the binary BW decoding rule given in (6.11), it is easy to see that $\mathbf{c}^{\text{ml}}_{\text{BPSK}} = \mathbf{c}^{\text{bw}}_{\text{BPSK}}$. For the cases considered in this letter, this means that the BW decoder and the ML decoder are one and the same, which results in their respective achievable rates being equal. In other words, the mutual information and the GMI will be identical. It is easy to explain this phenomenon by recalling the assumption made by the suboptimal BW decoder. The rule that defines such a decoder, given in (6.10), ignores the dependency of the bits in a symbol (it assumes they are independent). Throughout this paper, we have represented said symbols by the random variable X, which is drawn from an M-ary constellation $\mathbb{X}$. Throughout the derivations of this subsection, we assumed the cardinality of said constellation fulfilled $M > 2$. If $M > 2$, the symbols X of constellation $\mathbb{X}$ will be represented by more than one information bit, and so ignoring their dependence will result in a loss in achievable rate. However, when $M = 2$, which happens to be the scenario considered currently, constellation $\mathbb{X}$ only has two symbols represented by a single bit. Therefore, when a constellation where $M = 2$ is chosen, the BW decoding rule will be no different to the ML decoding rule due to the fact that there is only one bit per symbol (no dependencies are ignored). Knowing that in our simulations $I(X;Y) = \text{GMI}$, both (6.9) and the expression for the GMI derived below and given in (6.24), will define an achievable rate for our simulated system.

In a memoryless AWGN channel, when $m = 1$, the L-values shown in (6.20) can be computed as

$$L_k = \log\left[\frac{\exp(\frac{-|y_k-1|^2}{\sigma_z^2})}{\exp(\frac{-|y_k+1|^2}{\sigma_z^2})}\right] = \frac{2}{\sigma_z^2}y_k. \tag{6.23}$$

where $k = 1, \ldots, n$. For given sequences of n transmitted code bits c_l and n L-values λ_l where $l = 1, \ldots, n$ computed according to (6.16), the Monte-Carlo estimate shown in (6.21) can be reduced to

$$\text{GMI} \approx 1 - \frac{1}{n}\sum_{l=1}^{n}\log_2\left(1 + e^{(-1)^{c_l}\lambda_l}\right). \tag{6.24}$$

How (6.24) can be computed in the Matlab framework is shown in Figure 6.5.

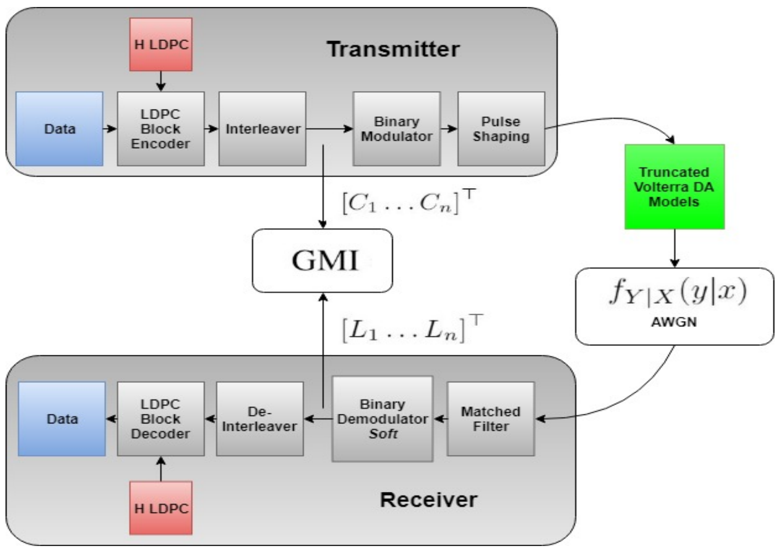


FIGURE 6.5: Computation of the GMI as given by (6.24).

Chapter 7

Conclusion

The objective of this thesis was to characterize an ultra-broadband RF electronic drive amplifier in order to study and understand what factors give rise to nonlinearity in its behaviour. To achieve this goal, we began with the introduction and development of the algorithms and modelling techniques used to perform the nonlinear system identification of the amplifier. Following this, the experimental activities and software simulations used to extract practical Volterra models were discussed. Next, how to assess the impact caused by introducing these models in a generic communications system was explained. After that, the parameters that determine the extent to which the response of the amplifier is nonlinear were thoroughly analyzed. Finally, considerations regarding future work and improvements were provided.

Nonlinear devices like electronic drive amplifier's can be modelled using numerous techniques. These techniques can also be implemented based on different adaptive algorithms. Once characterized, the nature of an electronic drive amplifiers response, including the reasons for it's display of nonlinearity, can be much easier understood.

In this thesis, truncated third order Volterra-series based models for an SHF-807-303987 30 GHz amplifier under different operating conditions have been obtained based on the Volterra least mean squares algorithm. Metrics capable of determining the content of nonlinearity in the response of said amplifier, which have been named as the normalized volterra kernel weights, have also been derived. In addition, the impact caused by integrating these Volterra models in a generic communications system operating in the presence of additive white Gaussian noise has been assessed. Moreover, the set of factors and the extent to which they impact the appearance of nonlinearity in the response of the amplifier, as well as the effect this nonlinear response has on the performance of the communications system, has been determined.

These factors are all characteristics of the signals used as inputs to the amplifier. Among them, the peak-to-peak voltage of the amplifier input signal has been shown to be heavily related to the amount of nonlinearity in the amplifier response, with an increase in said parameter causing significant losses in system performance. A relationship between the nonlinear content of the amplifier response and the input signal data rate has also been found. In contrast, modifying the bit pattern length of the input signal has been shown to lead to negligible changes in the content of nonlinearity in the amplifier response. Following the assessment of these input signal characteristics, preliminary work related to devising a method that could potentially be used to obtain amplifier behavioural models without the need for lab measurements was introduced.

The results included in this text show that the objective of this thesis has been met considerably well. However, it should be noted that all of the extracted models and conclusions derived from them are based on measurements conducted on a single device for a specific amplifier model, and thus their applicability to different devices of that same model, as well as other amplifier types, should be verified in later work. In terms of future research, proceeding with the development of a regressive model that enables the derivation of amplifier behaviour in the absence of lab data seems to be the most enticing topic. Aside from this, it would be interesting to analyze the impact caused by nonlinear amplifier behaviour on the FEC blocks of a communications system.

Chapter 8

Budget

The contents of this chapter detail the budget required to complete this thesis. It contains tables describing the costs of the following budget item categories:

- *Tangible Assets:* A detailed list of the components employed in the research experiments is provided.
- *Expendable Equipment:* This table contains records of the material resources and and office supplies that have been consumed throughout the development of the research.
- *Fixed Assets:* In this table, the costs related to the use of lab equipment is detailed. These expenses are computed based on the average depreciation cost and the time each specific asset has been used.
- *Software:* The different software platforms used throughout the thesis are listed. The costs are calculated based on the average amortization cost of the software licences.
- *Labour Cost:* This table includes the expenses related to the man hours required to complete all of the necessary research tasks.

Table 8.1: Tangible Asset Budget.

Quantity	Item	Description	Cost/Unit (€)	Total Cost (€)
1	Amplifier	SHF-807-303987	1482,88	1482,88
1	6dB Attenuator	Anritsu 41/43	316,29	316,29
2	10dB Attenuator	Anritsu 41/43	316,29	632,58
1	Remote Electrical Sampling Head	N1045A 60 GHz 2/4 Port	107,53	107,53
3	Female Connector Protective Cap	N1027A-1CF	45,69	137,03
3	Male Connector Protective Cap	N1027A-1CM	43,93	131,79
Total (€)				2808,10

Table 8.2: Expendable Equipment Budget.

Quantity	Item	Description	Cost/Unit (€)	Total Cost (€)
3	Pen	Bic Velocity Ball	0,72	2,16
250	Paper	A4 Sheets	0,02	4,75
4	Folder	A4 Paper Sheet Storage	0,52	2,08
15	Paper Clips	Jumbo Corrugated Clips	0,01	0,15
10	Staples	Intended for use with 20 Sheet Capacity Stapler	0,01	0,08
1	Calculator	T.I 84	120,22	138,33
Total (€)				147,55

Table 8.3: Fixed asset Budget.

Item	Acquisition Cost (€)	Depreciation Time (years)	Monthly Depreciation Expense(€)	Time Used (months)	Depreciation Cost (€)
SG Anritsu MG3697C	43.456,00	5	724,26	2	1.448,54
BPG SHF 12103A	20.647,53	5	344,13	2	688,25
DCA Agilent 86100D	24.474,47	7	291,36	2	582,72
RSM Agilent 86118A	8.597,61	7	102,35	2	204,70
PTM Agilent 86107A	44.323,72	7	527,66	2	1.055,33
Macbook 13-inch Retina Display	1986,00	4	41,38	6	248,25
Total (€)					4.227,79

Table 8.4: Software Budget.

Software	Acquisition Cost (€)	Depreciation Time (years)	Monthly Depreciation Expense(€)	Time Used (months)	Depreciation Cost (€)
Matlab	1958,97	1	163,25	6	979,49
Advanced Waveform Analysis Software	4911,71	7	58,47	2	116,95
Flex Eye Channel Control & Acquisition	10.893,22	1	129,68	7	778,09
Total (€)					1.874,53

Table 8.5: Labour Cost Budget.

Task	Duration (Hours)	Cost/Hour (€)	Total Cost (€)
Analytic & Simulation Framework Development	650	116,64	75.816,00
Lab Experiment Setup	10	266,61	2.666,10
Lab Measurements	50	116,64	5.832,00
Weekly Progress Reports	100	116,4	11.640,00
Weekly Assessment Meetings	22	582,74	12.820,28
Total (€)			108.774,38

8.1 Bugdet Summary

The following table provides a summary of the total budget required to complete this MASc thesis, which comes to a grand total of 139.808,08 euros.

Table 8.6: Budget Summary.

Budget Item	Partial Cost (€)	Cumulative Cost
Tangible Assets	2.808,10	2.808,10
Expendable Equipment	147,55	147,55
Fixed Assets	4.227,79	4.227,79
Software	1.874,53	1.874,53
Labour Cost	108.774,38	108.774,38
Indirect Costs (5%)		5.891,62
Cost (Before Tax)		123.723,97
Cost (After Tax - 13% HST)		139.808,08

Appendix A

Matlab implementation of the Cubic Volterra Least Means Squares Algorithm

```
function [volterra_kernels, error_squared, modelled_output] = ...
    cubic_vlms(uncorrupted_signal, received_signal, memory, mu1, mu2, mu3, max_iterations)

    x_input = [zeros(1,memory) transpose(real(uncorrupted_signal))];
    modelled_output = zeros(1,length(received_signal));
    order_2_base_term = 3; % Smallest number of quadratic Volterra kernels possible (occurs when memory = 2);
    order_3_base_term = 1; % Smallest number of cubic Volterra kernels possible (occurs when memory = 2);
    adjustable = order_2_base_term + sum(3:memory);
    if memory ==2
        adjustable2 = order_3_base_term + sum(3:3:(memory-1)*3); %4
    elseif memory == 3
        adjustable2 = order_3_base_term + sum(3:3:(memory-1)*3); %10
    elseif memory == 4
        adjustable2 = order_3_base_term + sum(3:3:(memory-1)*3) + 1; %20
    elseif memory == 5
        adjustable2 = memory + sum(3:3:(memory-1)*3); %35
    elseif memory == 6
        adjustable2 = 11 + sum(3:3:(memory-1)*3); %56
    else
        adjustable2 = 21 + sum(3:3:(memory-1)*3); %94 (adjustable2_mem6 + ajustable_mem7)
    end
    WvlmsI = zeros(1,memory+adjustable+adjustable2);

    for n=1:max_iterations % number of samples of longest signal

        X1 = fliplr(x_input(n+1:n+memory));

        for i=1:memory
            if i==1
                X2 = X1(i)*X1;
            else
                X2 = [X2 X1(i)*X1(i:end)];
            end
        end

        if memory == 2
            for ii=1:memory
                if ii==1
                    X3 = X1(ii)*X2;
                else
                    X3 = [X3 X1(ii)*X2(end)];
                end
            end

        elseif memory == 3
            for ii=1:memory
                if ii==1
                    X3 = X1(ii)*X2;
                elseif ii==2
                    X3 = [X3 X1(ii)*X2((ii-1)+memory:end)];
                else
                    X3 = [X3 X1(ii)*X2(end)];
                end
            end

        elseif memory ==4

            for ii=1:memory
                if ii==1
                    X3 = X1(ii)*X2;
                elseif ii==2
                    X3 = [X3 X1(ii)*X2((ii-1)+memory:end)];
                elseif ii==3
                    X3 = [X3 X1(ii)*X2((ii-1)+memory+2:end)];
                else
                    X3 = [X3 X1(ii)*X2(end)];
                end

            end

        elseif memory == 5
```

```

            for ii=1:memory
                if ii==1
                    X3 = X1(ii)*X2;
                elseif ii==2
                    X3 = [X3 X1(ii)*X2((ii-1)+memory:end)];
                elseif ii==3
                    X3 = [X3 X1(ii)*X2((ii-1)+memory+2:end)];
                elseif ii==4
                    X3 = [X3 X1(ii)*X2((ii-1)+memory+6:end)];
                else
                    X3 = [X3 X1(ii)*X2(end)];
                end

            end
        end

        Y_rx_real = real(received_signal(n));

        y_modelled = WvlmsI*[X1 X2 X3]';
        modelled_output(n) = y_modelled;
        error_y = Y_rx_real - y_modelled;                    % Instantaneous error of VLMS

        WvlmsI = WvlmsI +  error_y * [mu1*X1 mu2*X2 mu3*X3];  % Weight update rule of VLMS

        error_squared(n) = error_y^2;
    end
    MSE = 10*log10(mean(error_squared));
    volterra_kernels = [WvlmsI(1:memory) WvlmsI(memory+1:memory+3+sum(3:memory)) WvlmsI(memory+3+sum(3:memory)+1:end)];
end
```

Bibliography

[1] G.A. Hussein I. M. Eldokany S. El-Dolil O. Oraby R. A. Bakr, E. S. Hassan and F. E. Abd El-Samie. Continuous phase modulation with chaotic interleaving for optical ofdm systems. *Optical and Quantum Electronics*, Jan. 2015. doi: 10.1007/s11082-015-0130-5.

[2] D. Lavery R. Maher, A. Alvarado and P. Bayvel. Modulation order and code rate optimisation for digital coherent transceivers using generalised mutual information. *European Conference on Optical Communication (ECOC)*, Oct. 2015. doi: 10.1109/ECOC.2015.7341621.

[3] V. J. Mathews. Adaptive polynomial filters. *IEEE Signal Processing Magazine*, 8 (3):10 – 25, Jul. 1991. doi: 10.1109/79.127998.

[4] D. W. Griffith and G. R. Arce. Partially decoupled volterra filters: Formulation and lms adaptation. *IEEE Transactions on Signal Processing*, 45(6):1485 – 1494, Jun. 1997. doi: 10.1109/78.599973.

[5] S. Kalluri and G. R. Arce. A general class of nonlinear normalized adpative filtering algorithms. *IEEE Transactions on Signal Processing*, 47(8):2262 – 2272, Aug. 1999. doi: 10.1109/78.774769.

[6] P. M. Clarkson and M. V. Dokic. Stability and convergence behaviour of second-order lms volterra filter. *Electronics Letters*, 27(5):441 – 443, Feb. 1991. doi: 10.1049/el:19910279.

[7] A. Gutierrez and W. E. Ryan. Performance of volterra and mlsd receivers for nonlinear bandlimited satellite systems. *International Communications Conference*, 1995-1996.

[8] A. Gutierrez and W. E. Ryan. Convergence of the rls and lms adaptive filters. *IEEE Transactions on Circuits and Systems*, 34(7):799 – 803, Jul. 1987. doi: 10.1109/TCS.1987.1086206.

[9] L. Ljung L. Guo and P. Priouret. Performance analysis of the forgetting factor rls algorithm. *Proceedings of 31st IEEE Conference on Decision and Control*, Dec. 1992. doi: 10.1109/CDC.1992.371639.

[10] G. V. Moustakides. Study of the transient phase of the forgetting factor rls. *IEEE Transactions on Signal Processing*, 45(10):2468 – 2476, Oct. 1997. doi: 10.1109/78.640712.

[11] A. Gutierrez and W. E. Ryan. Performance of volterra and mlsd receivers for nonlinear bandlimited satellite systems. *IEE Transactions on Communications*, 48(7):1171 – 1177, Jul. 2000. doi: 10.1109/26.855524.

[12] J. Tsimbinos. Identification and compensation of nonlinear distortion. *Thesis, School of Electronic Engineering, University of South Australia*, Feb. 1995. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.46.833&rep=rep1&type=pdf.

[13] M. Divya. Bit error rate performance of bpsk modulation and ofdm-bpsk with rayleigh multipath channel. *International Journal of Engineering and Advanced Technology (IJEAT)*, 2(4):2249 8958, Apr. 2013.

[14] D. Lavery R. Maher A. Alvarado, E. Agrell and P. Bayvel. Replacing the soft-decision fec limit paradigm in the design of optical communication systems. *Journal of Lightwave Technology*, 33(20):4338 – 4352, Jun. 2015. doi: 10.1109/JLT.2015.2450537.

[15] P. J. Winzer G. J. Foschini R. Essiambre, G. Kramer and B. Goebel. Capacity limits of optical fiber networks. *Journal of Lightwave Technology*, 28(4):662 – 701, Feb. 2010. doi: 10.1109/JLT.2009.2039464.

[16] S. Calabr E. De Man G. Khanna, B. Spinnler and N. Hanik. A robust adaptive pre-distortion method for optical communication transmitters. *IEEE Photonics Technology Letters*, 28(7):752 – 755, Dec. 2015. doi: 10.1109/LPT.2015.2509158.

[17] S. Calabr E. De Man U. Feiste T. Dresnki G. Khanna, B. Spinnler and N. Hanik. A memory polynomial based adaptive digital pre-distorter for optical communication transmitters. *International Conference on Transparent Optical Networks (ICTON)*, Jul. 2017. doi: 10.1109/ICTON.2017.8024786.

[18] S. Calabr E. De Man U. Feiste T. Dresnki G. Khanna, B. Spinnler and N. Hanik. A memory polynomial based digital pre-distorter for high power transmitter components. *Optical Fibre Communications Conference and Exhibition (OFC)*, Mar. 2017.

[19] D. Lavery A. Alvarado, E. Agrell and P. Bayvel. Ldpc codes for optical channels: Is the fec limit" a good predictor of post-fec ber?,. *Optical Fibre Communications Conference and Exhibition (OFC)*, Mar. 2015. doi: 10.1364/OFC.2015.Th3E.5.

[20] T. M. Cover and J. A. Thomas. *Elements of Information Theory: 2nd Edition*. Wiley Series in Telecommunications and Signal Processing, Wiley-Interscience, 2006.

[21] L. Schmalen S. ten Brink A. Leven, F. Vacondio and W. Idler. Estimation of soft fec performance in optical transmission experiments. *IEEE Photonics Technology Letters*, 23(20):1547 – 1549, Jul. 2011. doi: 10.1109/LPT.2011.2162725.

[22] B. Chen A. Alvarado, T. Fehenberger and F. M. J. Willems. Achievable information rates for fiber optics: Applications and computations. *Journal of Lightwave Technology*, 36(2):424 – 439, Dec. 2017. doi: 10.1109/JLT.2017.2786351.

[23] W. Tranter. *Principles of communication systems simulation with wireless applications*. Prentice Hall Professional Technical Reference, Upper Saddle River, NJ, USA, 2004.

[24] L. Szczecinski and A. Alvarado. *Bit-Interleaved Coded Modulation: Fundamentals, Analysis, and Design*. Wiley, New York, NY, USA, 2015.

[25] G. Caire A. Martinez, A. Guilln i Fbregas and F.M.J. Willems. Bit-interleaved coded modulation revisited: A mismatched decoding perspective. *IEEE Transactions on Information Theory*, 55(6):2756 – 2765, May 2009. doi: 10.1109/TIT.2009.2018177.

[26] A. J. Viterbi. An intuitive justification and a simplified implementation of the map decoder for convolutional codes. *IEEE Journal on Selected Areas in Communications*, 16(2):260 – 264, Feb. 1998. doi: 10.1109/49.661114.

[27] L. Peng. Fundamentals of bit-interleaved coded modulation and reliable source transmission. *PhD dissertation, University of Cambridge, Cambridge, UK*, Dec. 2012. URL `http://itc.upf.edu/biblio/1066`.

[28] G. Bcherer. Probabilistic signal shaping for bit-metric decoding. *IEEE International Symposium on Information Theory*, Jul. 2014. URL `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6874869`.