MACHINE LEARNING FOR PERFORMANCE IMPROVEMENT OF LONG-HAUL END-TO-END OPTICAL TRANSMISSION SYSTEMS

EGOR SEDOV

Doctor of Philosophy

Aston University
November 2023

Even if you're scared, move forward.
Even if you tremble, move forward.
Even if your legs give out, move forward.
Just keep moving forward!

Tanjiro Kamado, Demon Slayer

To the people who believed in me — my friends, my wife, my family, and my parents...

Acknowledgements

My deepest gratitude goes to my supervisor, Professor Sergei Turitsyn, for his unwavering guidance and for making my doctoral journey possible. I extend my heartfelt thanks to my co-supervisor, Dr. Yaroslav Prilepsky, for his assistance and steadfast commitment to research integrity. My colleagues deserve special recognition for revealing to me the genuine essence of academic life. A special thank you to the TRANSNET program and its Principal Investigator, Polina Bayvel, for supporting my research.

I am indebted to my friends, Misha (who passed away just 9 days after my thesis defense), Valentine (whose first son, Gordei, was born 11 days after my defense), Stepan, and many others; to the members of the "BHYK" creativity group (and related): Ivan, Tusnin, Edos, Tuman, Max, Andrey, Tolyan, Alex, Dremov, Shipko, Mishin, Kuprik, Roman, Sanek, Kudashkin, Gervaz, Valera, Rabus for their constant support and for inspiring me to strive for excellence.

To my parents and family, both present and past, who have nourished my pursuit of knowledge and nurtured the scientist within me, I owe my deepest thanks.

Lastly, the most profound thanks are reserved for my wife, Olga, my steadfast companion through life's most rigorous challenges, and for my beloved dog, Bubaleh, who has brought boundless joy and harmony into my life.

Birmingham, 2023 E. S.

Abstract

The thesis focuses on addressing the challenges faced by optical fiber networks in keeping up with the growing demand for data transfer, especially with the advent of 5G/6G and the Internet of Things (IoT). The rapid expansion in data transfer requirements highlights the limitations of current optical fiber networks and the necessity for improvements in data encoding techniques, spectrum utilization, and signal clarity over long distances. The thesis contributes to this field by developing new methods for applying the Nonlinear Fourier Transform (NFT) to continuous signals, improving signal processing algorithms, and using Machine learning (ML) to understand complex patterns and make data-driven decisions to optimize optical communication systems.

The work is divided into two primary sections. The first section delves into advanced NFT techniques, including their application in optical fiber channel modeling for single and dual-polarization systems, signal processing with a sliding window technique combined with NFT, exploring solitonic components in optical signals, and the use of neural networks for NFT to work with noisy signals. The second section is dedicated to the role of ML in optimizing optical communication systems, discussing the new High-Performance COMmunication library (HpCom) framework for simulating optical channels, the use of Gradient Boosting for nonlinear equalization, studying received symbol distributions using the Gaussian Mixture Model, and summarizing findings with insights for future research. The thesis outlines the creation of innovative techniques to improve optical fiber systems, thus aiding the continued development of the digital world by handling the ever-increasing demands for data transmission.

Keywords: Optical Fiber Communications, Nonlinear Fourier Transform, Nonlinear Equalization, Machine Learning, Signal Processing Algorithms

Contents

List of Figures

List of Tables

List of Publications

1. Bogdanov, S., Shepelsky, D., Vasylchenkova, A., **Sedov, E.**, Freire, P.J., Turitsyn, S.K. and Prilepsky, J.E., 2023. Phase computation for the finite-genus solutions to the focusing nonlinear Schrödinger equation using convolutional neural networks. *Communications in Nonlinear Science and Numerical Simulation*, **125**, p.107311.

2. Srivallapanondh, S., Freire, P.J., Alam, A., Costa, N., Spinnler, B., Napoli, A., **Sedov, E.**, Turitsyn, S.K. and Prilepsky, J.E., 2023. Multi-Task Learning to Enhance Generalizability of Neural Network Equalizers in Coherent Optical Systems. *arXiv preprint arXiv:2307.05374*.

3. **Sedov, E.**, 2023, June. Gradient Boosting for Nonlinear Equalization in Optical Transmission Systems. In *2023 Conference on Lasers and Electro-Optics Europe & European Quantum Electronics Conference (CLEO/Europe-EQEC)* (pp. 1-1). IEEE.

4. Reznichenko, A.V., Chernykh, A.I., **Sedov, E.V**. and Terekhov, I.S., 2022. Optimal input signal distribution for a nonlinear optical fiber channel with small Kerr nonlinearity. *JOSA B*, **39**(3), pp.810-820.

5. **Sedov, E.V.**, Chekhovskoy, I.S. and Prilepsky, J.E.E., 2021. Neural network for calculating direct and inverse nonlinear Fourier transform. *Quantum Electronics*, **51**(12), p.1118.

6. **Sedov, E.V.**, Freire, P.J., Seredin, V.V., Kolbasin, V.A., Kamalian-Kopae, M., Chekhovskoy, I.S., Turitsyn, S.K. and Prilepsky, J.E., 2021. Neural networks for computing and denoising the continuous nonlinear Fourier spectrum in focusing nonlinear Schrödinger equation. *Scientific Reports*, **11**(1), p.22857.

7. Chekhovskoy, I.S., Medvedev, S.B., Vaseva, I.A., **Sedov, E.V.** and Fedoruk, M.P., 2021, June. Fast Eigenvalue Evaluation of the Direct Zakharov-Shabat Problem in Telecommunication Signals Using Adaptive Phase Jump Tracking. In *The European Conference on Lasers and Electro-Optics* (p. ci_p_2). Optica Publishing Group.

8. **Sedov, E.**, Freire, P.J., Chekhovskoy, I., Turitsyn, S. and Prilepsky, J., 2021, September. Neural Networks For Nonlinear Fourier Spectrum Computation. In *2021 European Conference on Optical Communication (ECOC)* (pp. 1-4). IEEE.

9. **Sedov, E.**, Prilepsky, J., Chekhovskoy, I. and Turitsyn, S., 2021, June. Computing continuous nonlinear Fourier spectrum of optical signal with artificial neural networks. In

European Quantum Electronics Conference. Optica Publishing Group.

10. Chekhovskoy, I., Medvedev, S.B., Vaseva, I.A., **Sedov, E.V.** and Fedoruk, M.P., 2021. Introducing phase jump tracking-a fast method for eigenvalue evaluation of the direct Zakharov-Shabat problem. *Communications in Nonlinear Science and Numerical Simulation*, **96**, p.105718.

11. **Sedov, E.V.**, Redyuk, A.A., Fedoruk, M.P. and Turitsyn, S.K., 2020, November. Statistical occurrence of soliton content in the conventional optical WDM signals. In *2020 International Conference Laser Optics (ICLO)* (pp. 1-1). IEEE.

12. **Sedov, E.V.**, Chekhovskoy, I.S., Prilepsky, J.E.E. and Fedoruk, M.P., 2020. Application of neural networks to determine the discrete spectrum of the direct Zakharov–Shabat problem. *Quantum Electronics*, **50**(12), p.1105.

13. Turitsyn, S., **Sedov, E.**, Redyuk, A. and Fedoruk, M., 2019. Nonlinear spectrum of conventional OFDM and WDM return-to-zero signals in nonlinear channel. *Journal of Lightwave Technology*, **38**(2), pp.352-358.

14. **Sedov, E.V.**, Redyuk, A.A., Fedoruk, M.P., Gelash, A.A., Frumin, L.L. and Turitsyn, S.K., 2018. Soliton content in the standard optical OFDM signal. *Optics Letters*, **43**(24), pp.5985-5988.

Introduction

Motivation

Optical fiber channels play a crucial role in today's communication systems, providing the high-speed data transmission worldwide. These channels are not just for data transfer—they are vital for maintaining global connections, contributing to economic development, and supporting essential services in healthcare, finance, education, and government. The rapid expansion of online services, the Internet of Things (IoT), and the increasing need for greater bandwidth highlight the essential role of optical fiber channels in the modern digital era [1, 2].

With the rise of new technologies like 5G/6G, cloud services, and more devices joining the internet, our current optical fiber networks face tough challenges to keep up with the growing demand for data transfer [3]. These challenges include limits on how much data can be sent through and the need to expand the use of the available spectrum. They also involve complex algorithms and the planning of networks. Solving these problems is critical to keep up with the digital world's rapid growth and to enable new advancements in communication technologies.

People from universities, companies, and governments are working hard to overcome the challenges faced by optical fiber networks. There is a strong push in research to find new ways to improve the ability of these networks to handle more data, work more efficiently, and be more reliable. Efforts include looking into better ways to encode data, creating new types of fibers, coming up with smarter network planning methods, and using new tools like machine learning and the Nonlinear Fourier Transform (NFT) to get the most out of optical fiber communications [4].

Optical fiber systems face many challenges, especially the need to increase their capacity to keep up with the growing demand for faster data speeds. One way to tackle this is to use more complex data encoding techniques, like advanced QAM [3]. We're also looking to use more of the light spectrum for data transmission, which means we would use not just the usual C-band but also the L, S, E, and O bands. This requires making new types of amplifiers that work with these broader ranges of light. We use methods like Forward error correction (FEC) and technologies like coherent detection to make signals clearer and reduce data errors over long distances . Expanding the spectrum used in optical fiber communication involves several areas of development. One area is improving fiber amplifiers to work with a broader range of light

frequencies, where technologies like rare-earth-doped fibers and semiconductor amplifiers are the key [3]. Creating components that can operate at these new frequencies is essential to use the entire available spectrum and increase the amount of data transmitted. Alongside hardware, we also need better algorithms to manage these new frequencies efficiently.

Another significant step is to create better fibers to overcome the limits of current ones. This includes making fibers with less loss and better handling of light interactions that can distort the signal. Efforts in this direction include researching fibers that lose less signal over distance and using Space Division Multiplexing (SDM) to send more data through the same cable [5].

Developing algorithms to handle nonlinearity in optical channels is essential. The Nonlinear Fourier Transform (NFT) is a mathematical tool that extends the concept of the classical Fourier transform to include the effects of nonlinearity in signal processing. In optical communication systems, nonlinearity can significantly impact signal propagation in optical fibers, often leading to distortion that can limit the performance and capacity of these systems. [4]. NFT takes into account the nonlinear nature of the optical fiber medium. By analyzing the signal in a nonlinear spectral domain, it can effectively describe the evolution of complex waveforms that are affected by the fiber's nonlinearity, thereby mitigating distortion. Traditional methods of communication approach the Shannon limit because they are based on linear signal processing. NFT offers a way to possibly exceed this limit by using the fiber's nonlinearity to our advantage rather than treating it as a hindrance. Moreover, NFT naturally leads to the concept of solitons [6], which are stable wave packets that can travel over long distances without changing shape. NFT-based systems can use solitons to carry information more reliably over long distances. NFT can be used to improve signal processing algorithms. By transforming the data into the nonlinear domain, NFT allows for the development of new signal processing algorithms that are better suited to dealing with the specific types of distortions and noise that occur in optical fibers. This can lead to improved spectral efficiency by allowing more data to be packed into the same bandwidth.

Machine learning (ML) is becoming an essential tool for optical communication systems due to its ability to understand complex patterns and make data-driven decisions. In the field of optical communications, where signals can be distorted by a variety of factors such as fiber nonlinearity and channel noise, ML stands out as a solution that can adaptively compensate for these impairments without explicit programming. This adaptability is crucial, as it allows communication systems to self-optimize in real-time, ensuring the transmission of data with higher accuracy and speed. Moreover, ML's predictive capabilities enable proactive maintenance, reducing downtime and enhancing the reliability of communication networks.

The utilization of ML in optical communication systems is not just about solving current challenges; it's also about paving the way for future advancements. As data traffic volumes continue to grow, ML provides a scalable approach to managing this increase efficiently. By analyzing vast amounts of transmission data, ML algorithms can predict and manage network loads, optimize routing, and even aid in the design of new fiber optic materials and structures.

This foresight is invaluable for keeping communication systems ahead of the curve, making ML not just a tool for maintenance but also a driving force for innovation in the field of optical communications.

Bringing together these advancements is key to pushing optical fiber communications forward. These technologies ensure that these systems are strong and ready for the growing demands of data transmission. With these tools, we can improve how optical fiber systems work, supporting the ongoing development and growth of our digital world.

Applications in Diverse Fields

Nonlinear Schrödinger Equation and the Manakov system have revolutionized our understanding of nonlinear wave dynamics in various physical settings. These mathematical models capture the essence of wave propagation influenced by nonlinear and dispersive effects, forming the backbone of numerous technological advancements and scientific discoveries [7]. The Nonlinear Fourier Transform stands out as a transformative tool, similar to the classical Fourier Transform but for nonlinear systems, enabling the decomposition of complex wave fields into simpler, analyzable components.

Consider the Bose-Einstein Condensates (BECs), where the NLS equation, known as the Gross-Pitaevskii equation, provides insights into the macroscopic wave function at zero temperature. Efforts to solve the GPE analytically for ultracold atom clouds in the BEC phase have utilized variational techniques, offering insights into the dynamics of these systems [8, 9]. The interplay between kinetic energy, trapping potential, and particle interactions is finely detailed, paving the way for the exploration of quantum effects such as soliton interactions and vortex formations. The NFT's power is magnified by Neural networks, which can predict the evolution of these phenomena from empirical data, even when the system's complexity exceeds traditional analytical capabilities. For instance, Neural networks could forecast the dynamics of solitons in BECs, providing a new lens through which quantum turbulence and particle interactions are viewed.

In plasma physics and water wave studies, the NLSE models the behavior of envelope solitons, essential for grasping Langmuir waves and modulational instability. It also aids in understanding the genesis of rogue waves and their modulation on the ocean's surface. The spatial NLSE's utility in modeling deterministic freak wave generation in water at MARIN illustrates its practical application, guided by the theory of linear perturbation analysis and solitons on a non-vanishing background [10–12]. The NLSE has been used as a framework for studying the emergence of rogue waves in the ocean, attributed to nonlinear energy transfer processes, through both deterministic and stochastic approaches [13–15]. Here, NFT, coupled with Neural network analysis, unravels the nonlinear spectral dynamics, facilitating the prediction of wave behavior under diverse conditions—a crucial step for designing plasma-based devices or mitigating rogue wave impacts.

The Nonlinear Fourier Transform offers a novel approach to analyzing complex wave dynamics in superconductivity, building on the foundational work with the nonlinear Klein-Gordon and sine-Gordon equations. These equations help us understand how waves move and interact in superconductors and have been crucial in studying phenomena like crystal dislocation and particle interactions [16–20]. A key area where NFT shows promise is in examining the Josephson effect, observed in Josephson junctions—quantum devices made of superconductor electrodes separated by a barrier. This effect, where current flows without voltage, highlights the quantum mechanical nature of superconductivity [21, 22]. The sine-Gordon model, used to describe Josephson junctions, aligns well with NFT's capabilities, especially in analyzing superconducting soliton oscillators [23]. Further, the connection between the sine-Gordon system and the NLSE in small-amplitude scenarios opens up possibilities for using NFT to study superconductors under external influences, like alternating current fields. This is where Neural networks come into play. By performing NFT, Neural networks can analyze complex superconducting systems, predict their behavior, and identify patterns that are not immediately apparent [24, 25]. Recent advancements in quantum graph theory illustrate how NFT, coupled with Neural networks, can be particularly effective in superconductivity research. For example, studying localized wave solutions in tricrystal Josephson junctions showcases the potential of Neural networks to understand the interactions within superconducting materials [26–29].

Nonlinear optics, another area enriched by the NLSE, sees the formation of spatial and temporal solitons in nonlinear birefringent media. Optical rogue waves, modeled by a generalized NLSE and supported by experimental data from highly nonlinear microstructured optical fibers, exemplify the fusion of theory and practice in this field [30–33]. The introduction of Neural networks to perform NFT provides a groundbreaking method for analyzing light-matter interactions, facilitating the development of advanced optical devices. In quantum fluids, NFT aids in understanding superfluid turbulence and vortex interactions, crucial for quantum computing and simulation technologies.

The NLS equation's application extends to atmospheric sciences, modeling nonlinear wave propagation including internal waves and solitons in stratified fluids. Leveraging NFT with Neural networks allows for enhanced weather prediction models and a deeper understanding of climate dynamics.

The methods from my thesis, like using Neural networks and a window approach, offer new ways to study and predict complex system behaviors. These tools are great for cases where we might not know the exact math or when systems don't fully match the usual conditions for NFT. Neural networks can find new patterns in data, helping us understand phenomena like rogue waves better. The window approach lets us study complex systems more easily, leading to new research and technologies in various fields, from environmental science to quantum tech and communications. Using NLSE and NFT methods in areas beyond optical communications shows how versatile these approaches are for studying wave dynamics. This research not only adds to academic knowledge but also opens up possibilities for new technologies and

applications in many physical systems.

Nonlinear Fourier Transform in Systems with Higher-Order Effects

Integrable systems, characterized by an infinite number of conservation laws, are idealized models where soliton solutions can propagate without changing shape due to exact balance between nonlinearity and dispersion. The NFT provides a powerful framework for analyzing these systems by transforming the Nonlinear Schrödinger Equation, a model governing soliton dynamics in fiber optics and other fields, into a spectral problem. This transformation enables the study of the system's evolution in terms of its spectral data, simplifying the analysis of soliton interactions and stability.

However, real-world systems are rarely perfectly integrable due to higher-order effects like attenuation, higher-order dispersion, and nonlinearity. These perturbations introduce deviations from the ideal soliton dynamics predicted by the NLSE [34]. For instance, higher-order dispersion can lead to the emission of radiation during soliton propagation, altering the soliton's shape and velocity. Similarly, higher-order nonlinearity can cause soliton self-frequency shift, a phenomenon where the central frequency of a soliton shifts during propagation.

Incorporating these higher-order effects into the NFT analysis requires perturbative techniques. One approach is to treat these effects as small perturbations to the integrable system and study their impact on the spectral data. For example, the effect of perturbations on soliton parameters (such as amplitude, width, and velocity) can be analyzed by examining how the spectral data evolves due to these nonidealities [35–37]. This perturbative analysis helps in understanding the stability of solitons under real-world conditions and in designing optical systems that can mitigate the adverse effects of higher-order corrections.

A concrete example of this analysis is seen in the study [38] of fiber optics, where solitons are used for long-distance communication. The presence of higher-order dispersion and nonlinearity in optical fibers affects the soliton's shape and speed, leading to signal distortion. By applying NFT with perturbative corrections [39], engineers can predict these changes and design optical systems that compensate for these effects, ensuring stable soliton propagation and reliable communication.

In the domain of soliton dynamics within nearly integrable systems, a significant body of research has illuminated the intricate effects of higher-order corrections. The work of Kodama and Hasegawa in 1982 [40], for example, delves into the realm of optical solitons in monomode fibers, focusing on how dissipative damping can be counteracted through periodically applied stimulated Raman scattering. This study stands as a cornerstone in understanding the interplay between soliton stability and external perturbations.

Further extending the discourse [41] embarked on a numerical exploration of a compensation model grounded in the application of a small-amplitude Raman pumping wave. Their findings

offer a deeper comprehension of how external forces can mold soliton dynamics, specifically highlighting the potential for dissipation to be offset in a controlled manner.

The analytical frameworks provided by Kaup and Newell in the late 1970s [24, 42] offer another layer of insight into the perturbed soliton solutions. Their work, through a dynamical systems lens, sheds light on the resilience and behavioral tendencies of solitons when subjected to external perturbations, enriching the dialogue on soliton dynamics with analytical rigor.

For a deeper dive into the latest studies, some numerous contemporary articles and findings shed light on advancements in this field. The article [43] investigates the emission of radiation during fast collisions between two solitons governed by the nonlinear Schrödinger equation with weak cubic loss. Employing perturbation theory and numerical simulations, the study reveals the impact of cubic loss on soliton dynamics, emphasizing the balance between conservative and dissipative perturbations in shaping radiation outcomes. This research contributes to the broader understanding of soliton interactions in nonlinear optical systems. The article [44] explores the effects of fiber optic loss on the dynamics of two-soliton states in nonlinear Schrödinger equations. It particularly focuses on how loss modifies the eigenvalue spectrum of soliton compounds, leading to significant changes in their behavior. This investigation reveals that fiber loss can enhance one soliton at the expense of another and transform static solitons into pairs with distinct velocities, providing new insights into soliton dynamics and their application in fiber-optic communications.

These studies collectively contribute to a nuanced understanding of how higher-order effects, ranging from external perturbations to dissipation, shape the evolution of solitons in nearly integrable systems. The interplay between theory and application revealed through these works underscores the dynamic nature of soliton research and its relevance to fields as diverse as optical communications and fluid dynamics. Through a careful examination of these contributions, one can appreciate the breadth of inquiry and the depth of discovery that characterize the ongoing exploration of soliton dynamics in the face of higher-order corrections.

Another book "Nonlinear Waves in Integrable and Nonintegrable Systems" by Jianke Yang [45] explores the mathematical and computational approaches to understanding nonlinear waves in both integrable and nonintegrable systems. It addresses significant problems in physical, life sciences, and engineering through analytical and computational techniques, promoting an interdisciplinary approach. The book covers various topics including the derivation of nonlinear wave equations, integrable theory, soliton perturbation theories, theories for nonintegrable equations, nonlinear wave phenomena in periodic media, and numerical methods for studying nonlinear wave equations.

The Nonlinear Fourier Transform emerges as a robust tool for analyzing integrable systems even when higher-order effects—such as attenuation and higher-order dispersion—come into play. Through the detailed studies and examples provided, it's evident that NFT can effectively account for these perturbations, enabling a more accurate analysis of soliton dynamics and interactions in real-world systems. By extending traditional NFT techniques with perturbation

theory, researchers can predict and mitigate the impact of these non-idealities, ensuring the stability and reliability of soliton-based technologies in various applications, from optical communications to quantum computing.

Contribution

The contribution of this thesis can be summarized as follows:

- We proposed and developed a new method that applies the NFT to continuous signals. This method involves pre-processing the signal using Chromatic dispersion compensation (CDC) before applying NFT. The outcomes for single polarization signals have shown better performance than Digital back-propagation (DBP) with three steps per span, and for dual polarization, the results are compared with DBP with two steps per span. We have made the developed code publicly available.
- Through the use of NFT, we analyzed WDM and Orthogonal Frequency Division Multiplexing (OFDM) symbols to determine the conditions for the presence of solitons. Our analysis of WDM signals revealed that a dominant portion of the energy (99%) is associated with soliton components.
- We introduced a new approach for conducting both forward and inverse NFT operations using NNs. The chosen NN architecture was further optimized and has demonstrated effectiveness in working with noisy signals, where deterministic NFT algorithms were unsuccessful.
- A new GPU-accelerated framework, High-Performance COMmunication library (HpCom), was introduced for simulating optical fiber channels with various characteristics. This framework serves as a data mining tool for subsequent research, and we have provided code examples along with the resulting data and performance metrics.
- We identified Gradient Boosting (GB) as a promising technique for nonlinear equalization within optical channels.
- We demonstrated the performance of NNs across a range of channel and signal parameters, showcasing their versatility and effectiveness.
- A data-driven approach was employed to uncover new complex structures within the received symbols' constellation, which arise from the nonlinearity of optical fibers. This non-Gaussian structure was evident across all examined "triplets" of received points.
- Furthermore, we provided a demonstration of real-time optical channel simulation as an additional resource.

Thesis outline

This thesis is organised as follows.

The thesis divided into two parts: “Advanced Nonlinear Fourier Transform Techniques” and “Machine Learning for Optimizing Optical Communication Systems”.

In the first part

- Chapter 1 provides a comprehensive introduction to the Nonlinear Fourier Transform (NFT) and its application in modeling optical fiber channels for both single and dual-polarization systems. The chapter further discusses various modulation formats and the performance metrics relevant to this study, concluding with an in-depth look at Wavelength Division Multiplexing (WDM) and Orthogonal Frequency-Division Multiplexing (OFDM), which are central to the research conducted in this thesis.
- Chapter 2 delves into the processing of optical signals using a sliding window technique and introduces an enhanced method combining Chromatic Dispersion Compensation (CDC) with NFT. Results for WDM signals in both single and dual-polarization systems are presented at the end of this chapter.
- Chapter 3 explores the solitonic components of optical signals, beginning with the establishment of criteria for energy and power levels required for soliton existence. This is followed by an analysis of individual OFDM and WDM symbols to assess the probability of soliton occurrence based on symbol parameters. The chapter proceeds with an investigation into the presence of solitons in continuous WDM signals and the distribution of their properties.
- Chapter 4 wraps up the first part of the thesis by introducing the application of Neural Networks (NN) for NFT. It opens with the motivation behind using NN and demonstrates that CNNs can effectively compute NFT. The approach is further validated for its ability to identify solitonic components within a signal. The final section of this chapter provides detailed findings on the optimal structure for NN and its capability to manage NFT in the presence of noise.

In the second part

- Chapter 5 presents the High-Performance COMmunication library (HpCom) framework, a new tool for simulating optical channels. It begins with an overview of the simulation methods, discusses the architecture of the framework, and demonstrates how it can speed up simulations. The chapter concludes with examples demonstrating the framework’s application in simulating optical communication systems.
- Chapter 6 introduces the use of Gradient Boosting (GB) for nonlinear equalisation in optical communication systems. The chapter progresses from the theoretical basis of

gradient boosting to its practical application in complex systems, analyzing its computational complexity. Results showcase the efficacy of GB across various signal powers and transmission distances, and the chapter ends with a example of the broader use of machine learning methods like Neural network (NN) for signal equalization tasks.

- Chapter 7 delves into the study of received symbol distributions in optical communication systems, outlining potential distribution types for initial analysis. It elaborates on employing the Gaussian Mixture Model (GMM) to describe and emulate the distribution of the received constellation.
- Chapter 8 wraps up the thesis with a summary of the findings and provides insights into future research directions within the field of optical communications.

Part I

Advanced Nonlinear Fourier Transform Techniques

1 Nonlinear Fourier Transform Fundamentals

1.1 Channel models

The Nonlinear Schrödinger Equation (NLSE) and the Manakov Equation (ME) are two commonly used mathematical models for describing the behavior of light propagation in a single-mode optical fiber, including the effects of dispersion and nonlinearity [46]. The NLSE, being a generic model describing the interplay between the dispersive and nonlinear effects, is applicable to the description of a vast number of physical phenomena, ranging from the dynamics of magneto-ordered systems [47] to hydrodynamics [48]. It also serves, under certain assumptions, as a principal master model governing the evolution of a single-polarisation slow-varying light envelope propagating along the single-mode fibre [49, 50]. In the context of optical fiber transmission, the Schrödinger equation is used to describe the evolution of the slowly varying optical field envelope, $A(z,t)$, in the presence of dispersion and nonlinearity. The general form of the Schrödinger equation for a single-polarization signal is as follow:

$$\frac{\partial A}{\partial z} = -\frac{\alpha}{2}A - i\frac{\beta_2}{2}\frac{\partial^2 A}{\partial t^2} + i\gamma|A|^2 A, \tag{1.1}$$

where z represents the distance along the fiber, t is time, β_2 is the Group velocity dispersion (GVD) parameter, γ is the nonlinear coefficient, and A is the complex envelope of the optical field.

The Manakov Equation is a more complex model that is used to describe the behaviour of light propagation in a two-polarisation optical fiber. It considers the effects of polarization-mode dispersion and nonlinear interactions between the two polarizations. The Manakov Equation for a two-polarization signal is as follows:

$$\begin{aligned}
\frac{\partial A_x}{\partial z} &= -\frac{\alpha}{2}A_x - i\frac{\beta_2}{2}\frac{\partial^2 A_x}{\partial t^2} + i\gamma\frac{8}{9}\left(|A_x|^2 + |A_y|^2\right)A_x, \\
\frac{\partial A_y}{\partial z} &= -\frac{\alpha}{2}A_y - i\frac{\beta_2}{2}\frac{\partial^2 A_y}{\partial t^2} + i\gamma\frac{8}{9}\left(|A_x|^2 + |A_y|^2\right)A_y,
\end{aligned} \tag{1.2}$$

where A_x and A_y are the complex envelopes of the two polarizations. This model includes attenuation and is applied in simulations of Erbium-Doped Fiber Amplifier (EDFA).

It is important to specifically address the scenario where $\alpha = 0$, which is relevant for NFT systems and their analysis. The use of NFT in the presence of attenuation necessitates extra steps, including the estimation of an effective nonlinear coefficient γ_{eff}. While this discussion is outside the current document's focus, it certainly represents a promising area for future research that could build upon the findings presented here.

In this study, we have simplified the channel model to exclude attenuation, a necessary simplification for applying NFT to optical communications signals. The Nonlinear Schrödinger Equation (NLSE) for a channel model without attenuation is presented in dimensioned units as:

$$i\frac{\partial A}{\partial z} - \frac{\beta_2}{2}\frac{\partial^2 A}{\partial t^2} + \gamma|A|^2 A = 0, \tag{1.3}$$

To effectively utilize NFT, we introduce non-dimensional variables using characteristic scales like the pulse width or symbol interval T_0, dispersion length $L_D = T_0^2/|\beta_2|$, and characteristic power $P_0 = 1/(\gamma L_D)$. These lead to the following dimensionless equation, where $\sigma = \text{sign}(-\beta_2\gamma)$ represents either focusing ($\sigma = 1$) or defocusing ($\sigma = -1$) dispersion:

$$i\frac{\partial q}{\partial \zeta} + \frac{\sigma}{2}\frac{\partial^2 q}{\partial \tau^2} + |q|^2 q = 0, \tag{1.4}$$

In the situation where there is no signal loss ($\alpha = 0$), the Manakov Equation (ME) equations are:

$$\begin{aligned} i\frac{\partial A_x}{\partial z} - \frac{\beta_2}{2}\frac{\partial^2 A_x}{\partial t^2} + \gamma\frac{8}{9}\left(|A_x|^2 + |A_y|^2\right)A_x = 0, \\ i\frac{\partial A_y}{\partial z} - \frac{\beta_2}{2}\frac{\partial^2 A_y}{\partial t^2} + \gamma\frac{8}{9}\left(|A_x|^2 + |A_y|^2\right)A_y = 0, \end{aligned} \tag{1.5}$$

with the same symbol interval T_0, the dispersion length $L_D = T_0^2/|\beta_2|$ reference power level now scaled with $P_0 = 1/(\frac{8}{9}\gamma L_D)$. These choices allow us to define scaled variables $\tau = t/T_0$, $\zeta = z/L_D$, and $q_{x,y} = A_{x,y}/\sqrt{P_0}$:

$$\begin{aligned} i\frac{\partial q_x}{\partial \zeta} + \frac{\sigma}{2}\frac{\partial^2 q_x}{\partial \tau^2} + \left(|q_x|^2 + |q_y|^2\right)q_x = 0, \\ i\frac{\partial q_y}{\partial \zeta} + \frac{\sigma}{2}\frac{\partial^2 q_y}{\partial \tau^2} + \left(|q_x|^2 + |q_y|^2\right)q_y = 0. \end{aligned} \tag{1.6}$$

For clarity, throughout this thesis, we will refer to the scaled fields q (from A) or $q_{x,y}$ (from $A_{x,y}$) simply as 'the signal'.

For the NFT, we work with the NLSE and the ME as set out in Eqs. (1.4) and (1.6). It's important to note that we can adjust the equations using a normalization factor of 2. This means setting

the dispersion length L_D and the characteristic power P_0 as $L_D = 2T_0^2/|\beta_2|$ and $P_0 = 2/(\frac{8}{9}\gamma L_D)$, respectively. After normalization, the fields $q_{x,y} = A_{x,y}/\sqrt{P_0}$ for dual-polarization or simply $q = A/\sqrt{P_0}$ for a single polarization remain the same. The normalized Nonlinear Schrödinger Equation and Manakov Equation equations are then written as:

$$\frac{\partial q}{\partial \zeta} + \sigma \frac{\partial^2 q}{\partial \tau^2} + 2|q|^2 q = 0, \tag{1.7}$$

for a single polarisation, and for dual polarisation:

$$\begin{aligned} i\frac{\partial q_x}{\partial \zeta} + \sigma \frac{\partial^2 q_x}{\partial \tau^2} + 2\left(|q_x|^2 + |q_y|^2\right) q_x = 0, \\ i\frac{\partial q_y}{\partial \zeta} + \sigma \frac{\partial^2 q_y}{\partial \tau^2} + 2\left(|q_x|^2 + |q_y|^2\right) q_y = 0, \end{aligned} \tag{1.8}$$

respectively.

1.2 Nonlinear Fourier Transform

1.2.1 Introduction

Quite often, the evolution of nonlinear systems is well approximated by the nonlinear partial differential equation (PDE). Evidently, there is no universal theory for the solution of nonlinear PDEs, but there exists a distinguished class of nonlinear equations that can be solved with a mathematical rigour: the so-called integrable systems. The history of integrable PDEs started in the 1960s when Gardner et al. [51] discovered a method for finding the infinite families of exact solutions for the Korteweg-de Vries equation. Their method termed the inverse scattering transform, can be deemed as the generalisation of the conventional Fourier Transform (FT) to the nonlinear systems. Thus, the name Nonlinear Fourier Transform (NFT) for it is often used nowadays, especially in the signal processing literature [4, 52]. Shortly after the integration of the Korteweg-de Vries equation, Zakharov and Shabat developed the inverse scattering machinery (i.e. the NFT method) for yet another celebrated PDE: the Nonlinear Schrödinger Equation (NLSE) [53], which will be the focus of our current study.

In a nutshell, for an integrable PDE there exists the canonical transform of dependent variables, converting the original nonlinear system into the so-called action-angle variables; the evolution of the latter is governed by a set of uncoupled trivial (linear) differential equations. Mathematically, this can be treated as the effective linearisation of a nonlinear integrable PDE [54, 55]. For our work, it is important that we know the explicit form of the NFT operations attributed to the NLSE.

Withing modern optical communications, the NFT is used not as a tool for the NLSE solution, but as a signal processing method [4, 52]. This concept originated from the work of Hasegawa and Nyu [56], who proposed to depart from considering the time domain solitonic shapes [50],

but rather use the nonlinear spectrum (the so-called eigenvalues) for the data modulation and transmission. Over the last decade, the NFT-based optical transmission techniques have been resurrected and greatly extended [4, 57]. The most efficient NFT-based optical transmission method is the so-called Periodic Nonlinear Fourier Transform (PNFT) [52], within which we directly modulate the parameters of the nonlinear modes that emerge from the Nonlinear Fourier (NF) signal decomposition. When the optical field propagates down the fibre link, the evolution of the nonlinear modes inside the NF domain stays almost linear, in contrast to the truly nonlinear evolution of signal in the space-time domain. Due to this property, we can theoretically get rid of the infamous nonlinear cross-talk degrading the transmission performance at high signal powers[58].

Generally, when considering the NF decomposition of an arbitrary rapidly decaying waveform, we can have two distinct coexisting parts of the NF spectrum: the continuous part, describing quasi-linear dispersive waves, and the discrete part, corresponding to solitonic modes [4, 52, 54, 55]. The continuous part of NF spectrum is represented by the complex-valued function $r(\xi) \in \mathbb{C}$ of a real argument $\xi \in \mathbb{R}$, where ξ is called the spectral parameter; $r(\xi)$ is called the reflection coefficient, and ξ emerges as the nonlinear analogue of a conventional Fourier frequency. This NF spectrum part converges to the conventional FT of our signal in the low-power limit [59]. The discrete part consists of the complex eigenvalues $\xi_n \in \mathbb{C}^+$, located in the upper complex half-plane, and the associated norming constants r_n (spectral amplitudes) [60]. The graphical summary of the general NF spectrum structure is given in Fig. 1.1. However, we point out that it is exactly the utilisation of the continuous NF spectrum part [61–68] that resulted in the breakthrough in the PNFT technology: this idea, mentioned already in early NFT transmission-related works [52, 59], is in stark contrast with the progenitor soliton-based transmission methods [50]. We mention that the continuous NF spectrum modulation using the special technique coined b-modulation [69–72] has provided the highest PNFT data rates so far [57, 73]. Finally, we note that for the PNFT based on the discrete NF spectrum [74–76], the achieved data rates have been noticeably lower than those for the modulation of continuous NF spectrum, see the comparison in [57, Fig. 1].

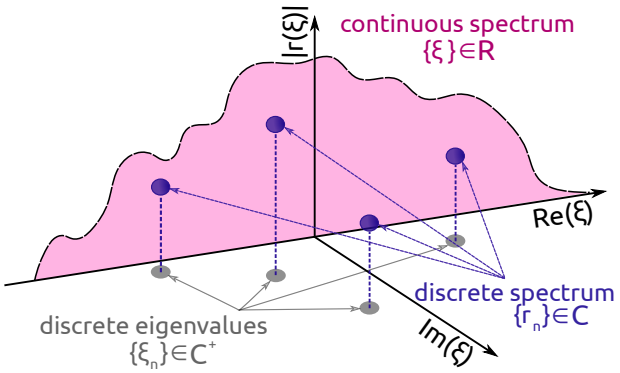


Figure 1.1: The schematic showing the different coexisting parts of a general NF spectrum: the discrete part, represented by the eigenvalues ξ_n and respective norming constants r_n, and the continuous part, shown as the function $|r(\xi)|$ on the real ξ-axis.

The PNFT transmission method relies on the (approximate) integrability of our transmission channel, i.e. we inherently assume that Eq. (1.4) is a very accurate model describing the signal evolution down the fibre. However, aside from second-order dispersion and Kerr nonlinearity present in (1.4), in realistic fibre-optic systems, there are numerous other effects affecting signal's propagation. Optical noise inevitably arising during the amplification process [49] is one of the key challenges in optical communications. The noise results in random NF spectrum disturbances [77, 78], imposing limits on the PNFT transmission quality. Another widespread deviation from idealised model (1.4) is the non-zero nonuniform gain-loss profile occurring in realistic systems for both lumped [63, 67] and distributed [64] amplification schemes. We also mention the effects of polarisation mode dispersion [79, 80], higher-order chromatic dispersion [79], and component-induced impairments, to itemise just several important sources. All these effects bring about the deviations of the true optical channel from integrable NLSE (1.4) such that the NF spectrum of the signal at the end of our transmission system can be significantly distorted, which results in the appearance of errors in the transmitted data [65, 66, 79]. Given that, the machine learning and artificial Neural network (NN) based signal processing methods have recently attracted much attention, as they can effectively render adaptive distortions-resilient signal processing tools, and, thus, using the NNs we can mitigate the impact of detrimental factors mentioned above [81, 82].

Finally, we note that, recently, the interest in using the NFT as a signal-processing tool has risen in fields that are not directly relevant to optical transmission. In particular, the NFT was applied in the so-called integrable turbulence to monitor the appearance of coherent structures, such as breathers, solitons, and rogue waves [83, 84], to the optical microresonators regime analysis [85], to the optical frequency combs characterisation [86], and to the analysis of laser regimes and the emergence of dissipative coherent nonlinear structures [87–89]. The analysis of NFT modes' evolution for such systems often appears to be more informative and convenient than dealing with the conventional Fourier modes. The NFT is also an important tool for the design of fibre Bragg gratings [90, 91]. Thus, we believe that the technique presented in this work can have a much wider range of applications than simply being a processing tool in optical communications. To end up, solving nonlinear differential equations itself by using NNs is a fast-growing area with a range of applications in science and engineering [92–94]. We hope that our work will also advance knowledge in this emerging field.

Periodical Nonlinear Fourier Transform

While this thesis primarily focuses on Nonlinear Fourier Transform techniques with vanishing boundary conditions and the methods developed for this particular case, it is imperative to acknowledge the existence of another significant branch of research in the field of NFT, known as Periodic Nonlinear Fourier Transform (PNFT). The PNFT approach marks a significant advancement in the study and application of NFT techniques, specifically designed for periodic or quasi-periodic boundary conditions. This approach not only extends the utility of NFT methods to a broader range of signal types but also addresses several limitations associated

with conventional NFT methods by offering enhanced control over signal's duration and bandwidth, reducing the processing window at the receiver, and mitigating noise impact [95].

PNFT-based transmission systems have been thoroughly investigated using the algebro-geometric approach, which, despite its computational intensity due to the reliance on the Riemann theta function, has paved the way for innovative solutions in optical communications [96, 97]. To circumvent the computational challenges, the Riemann-Hilbert Problem (RHP) approach has been proposed as a more practical alternative. This method benefits from linear computational complexity proportional to the number of signal samples, facilitating efficient parallelization and computation [98].

The finite-gap theory emerges as a complementary technique within the PNFT framework, offering a rich mathematical foundation for the analysis of continuous (non-decaying) signals. This theory, rooted in the Inverse Scattering Transform (IST) method and the Gelfand-Levitan-Marchenko integral equations, provides a profound understanding of the spectral properties of integrable systems [99]. An alternative perspective on the inverse problem is offered through the formulation of a factorization problem of the Riemann-Hilbert (RH) type, further simplifying the integration of spatial and temporal variables into the analysis [100].

The theory of finite-genus solutions to the NLSE has seen extensive application in the analysis of water (ocean) waves [101, 102]. The finite-gap integration methodology has introduced groundbreaking implications for problems on a finite spatial interval with periodic boundary conditions, enabling the generation of exact solutions known as finite-gap solutions. These solutions are explicitly given in terms of associated theta functions, derived from solving a Jacobi inversion problem on a finite-genus Riemann surface, albeit their computational accessibility remains a challenge [103–105].

In optical communications, the application of finite-gap theory and PNFT has been instrumental in the development of innovative methods based on non-decaying solutions to the NLSE. These methods have demonstrated significant potential in enhancing the efficiency of optical telecommunication systems, leveraging the robustness of finite-gap solutions to address challenges inherent in soliton-based communications [95, 96, 98, 106, 107]. The numerical solution of RH problems plays a crucial role in the modulation and digital signal processing (DSP) methods proposed for PNFT-based transmission, where the inverse problem is solved by constructing a signal in the physical domain from spectral parameters [98, 106].

Thus, while our thesis concentrates on NFT methods for vanishing boundary conditions, the exploration of PNFT and finite-gap theory underscores a broader perspective in the field. These approaches not only enrich our understanding of NFT and its applications but also open new avenues for research and development in optical telecommunication systems, promising more robust, efficient, and adaptable communication technologies for the future.

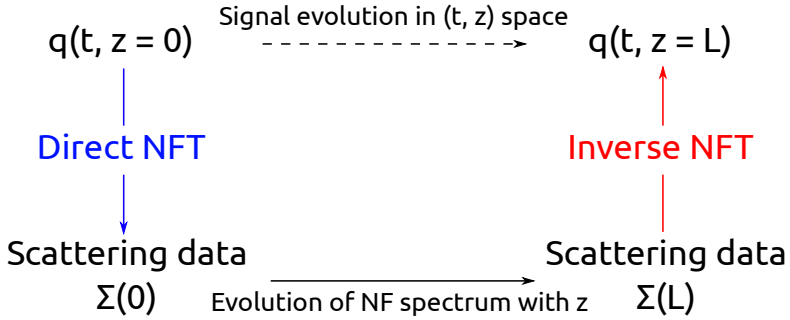


Figure 1.2: Schematic representation of Nonlinear Fourier Transform.

1.2.2 Theory

NFT for Nonlinear Schrödinger Equation

The Nonlinear Schrödinger Equation (1.4) (so does Manakov Equation (1.6)) belongs to the class of the so-called integrable nonlinear partial differential equation (PDE)s that can be integrated by the Inverse Scattering Transform (IST) method, also known as the Nonlinear Fourier Transform (NFT) [53, 108]. The NFT allows one to find the evolution (with z) of the signal $q(t,z)$ described by the NLSE channel by solving two linear problems instead of solving the nonlinear partial differential equation (PDE). A complex signal evolution governed by NLSE is replaced by 3 steps (see Fig. 1.2):

1. direct transform from $q(t,z=0)$ to the so-called scattering data $\Sigma(0)$ (nonlinear spectrum and spectral data of the initial signal)
2. trivial evolution of the scattering data with z, and
3. inverse transform to restore signal $q(t,z)$ at any desired propagation distance z.

The NFT method is similar to standard Fourier approach for the solution of linear evolution equations: present initial field in the spectral domain, consider trivial evolution of each spectral harmonic, reconstruct evolved signal from the spectral components known for any propagation distance.

We will now focus on the NLSE (1.4) presented as:

$$i\frac{\partial q}{\partial z}+\frac{\sigma}{2}\frac{\partial^2 q}{\partial t^2}+|q|^2 q=0, \tag{1.9}$$

where $q(t,z)$ represents the complex envelope of the optical field that decays rapidly as t approaches infinity. The variable z stands for the length of the optical fiber, and t is the time variable. The parameter σ takes a value of -1 for normal dispersion and 1 for anomalous dispersion [46]. This equation, set in a frame moving with the pulse, describes how light pulses $q(t,z)$ travel through fiber optics. The initial conditions (Cauchy problem) for the pulse are

set at a starting point $z = z_0$ and given by:

$$q(t,z)|_{z=z_0} = q_0(t). \tag{1.10}$$

The mathematical approach developed by Zakharov and Shabat [53] integrates the NLSE. This technique, known as the Nonlinear Fourier Transform, converts the optical signal into a Nonlinear Fourier (NF) spectrum. This spectrum is derived from the solutions to the Zakharov-Shabat problem (ZSP).

Equation (1.9) can be written as a condition of compatibility:

$$\frac{\partial L}{\partial z} = ML - LM, \tag{1.11}$$

which are written as

$$L\Psi = \xi\Psi, \quad \frac{\partial \Psi}{\partial z} = M\Psi, \tag{1.12}$$

where $\Psi(t)$ is a complex, two-component function that depends on the real variable t, and

$$L = i\begin{pmatrix} \partial_t & -q \\ -\sigma q^* & -\partial_t \end{pmatrix}, \quad M = i\begin{pmatrix} \sigma\partial_t^2 + \frac{1}{2}|q|^2 & -\sigma q\partial_t - \frac{1}{2}\sigma q_t \\ -q^*\partial_t - \frac{1}{2}q_t & -\sigma\partial_t^2 - \frac{1}{2}|q|^2 \end{pmatrix}. \tag{1.13}$$

The first equation in Eq. (1.12) is the eigenvalue problem for the operator L. When $\sigma = -1$, the operator L is Hermitian, meaning it equals its own complex conjugate transpose ($L = L^\dagger = (L^*)^T$), so the spectral parameter $\xi = \zeta + i\eta$ is a real number $\xi = \zeta \in \mathbb{R}$. If $\sigma = 1$, there is no such constraint, and we find both continuous and discrete parts to the spectrum. The continuous part is on the real axis, while the discrete part is found where the imaginary part of ξ is greater than zero ($\mathrm{Im}(\xi) > 0$).

The first equation in Eq. (1.12) can also be expressed as an evolutionary system:

$$\frac{d\Psi(t)}{dt} = Q(t)\Psi(t), \tag{1.14}$$

where $q = q(t,z)$ is the signal at any point z along the fiber, which we'll often not mention to simplify our equations. The vector $\Psi(t)$ and the matrix $Q(t)$ are defined as

$$\Psi(t) = \begin{pmatrix} \psi_1(t) \\ \psi_2(t) \end{pmatrix}, \quad Q(t) = \begin{pmatrix} -i\xi & q \\ -\sigma q^* & i\xi \end{pmatrix}.$$

Some sources call this the Zakharov-Shabat spectral problem (ZSSP). When we look at the ZSSP for the signal at the start of the fiber, $q(t, z = 0)$, the problem is laid out as:

$$\begin{cases} -\partial_t\psi_1 + q_0(t)\psi_2 = i\xi\psi_1, \\ \partial_t\psi_2 + \sigma q_0^*(t)\psi_1 = i\xi\psi_2. \end{cases} \tag{1.15}$$

In these equations, $q_0(t)$ is our initial signal, and ξ is a complex parameter that we use to solve the system.

The system described by Eq. (1.14) can also be expressed in a gradient form:

$$\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix}_t = J\begin{pmatrix}\frac{\partial H}{\partial \psi_1^*}\\ \frac{\partial H}{\partial \psi_2^*}\end{pmatrix} = J\begin{pmatrix}-i\xi & \sigma q\\ -\sigma q^* & i\xi\end{pmatrix}, \tag{1.16}$$

where H represents the energy of the system and is given by $H = |\psi_1|^2 + \sigma|\psi_2|^2$. When the spectral parameter ξ is a real number, the matrix J becomes skew-Hermitian (which means $J = -J^*$), and this applies for both values of $\sigma = \pm 1$. Therefore, the equation in (1.16) keeps the value of H constant over time.

We can express the energy H and the matrix Q using special matrices called Pauli matrices, denoted as σ_0 and σ_3:

$$H = \begin{cases}(\Psi^*, \sigma_0\Psi), & \text{for } \sigma = 1,\\ (\Psi^*, \sigma_3\Psi), & \text{for } \sigma = -1,\end{cases} \tag{1.17}$$

$$Q = \begin{cases}J\sigma_0, & \text{for } \sigma = 1,\\ J\sigma_3, & \text{for } \sigma = -1,\end{cases}, \tag{1.18}$$

where the curved brackets $(\cdot,\cdot)$ represent the scalar product of complex vectors.

When the signal $q(t)$ decays rapidly as t goes to infinity, we can find specific solutions, known as Jost functions, for the ZSP shown in Eq. (1.12). These solutions are:

$$\Psi = \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} = \begin{pmatrix}e^{-i\xi t}\\ 0\end{pmatrix}[1 + o(1)], \quad t \to -\infty, \tag{1.19}$$

for the left boundary, and

$$\Phi = \begin{pmatrix}\phi_1\\ \phi_2\end{pmatrix} = \begin{pmatrix}0\\ e^{i\xi t}\end{pmatrix}[1 + o(1)], \quad t \to \infty, \tag{1.20}$$

for the right boundary. If we use the left boundary condition in Eq. (1.19), we can say that $H = 1$ for any real spectral parameter. From here, we can find the Jost scattering coefficients $a(\xi)$ and $b(\xi)$:

$$a(\xi) = \lim_{t\to-\infty} \psi_1(t,\xi)e^{i\xi t}, \quad b(\xi) = \lim_{t\to-\infty} \psi_2(t,\xi)e^{-i\xi t}. \tag{1.21}$$

These coefficients always meet the following condition:

$$|a(\xi)|^2 + \sigma|b(\xi)|^2 \equiv 1. \tag{1.22}$$

The spectral data of the ZSP is given by $a(\xi)$ and $b(\xi)$ and can be described as:

1. The zeros of $a(\xi) = 0$ give us the discrete spectrum $\{\xi_k\}, k = 0, K-1$, with the phase coefficients defined by

$$r_k = \left.\frac{b(\xi)}{a'(\xi)}\right|_{\xi=\xi_k}, \quad \text{where } a'(\xi) = \frac{da(\xi)}{d\xi}. \tag{1.23}$$

2. The continuous spectrum is defined by reflection coefficient

$$r(\xi) = b(\xi)/a(\xi), \xi \in \mathbb{R}. \tag{1.24}$$

We use the "left" boundary condition Eq. (1.19) to find these spectral data. Both the left and right conditions Eqs. (1.19) and (1.20) are useful for figuring out the coefficient $b(\xi_k)$ for the discrete part of the spectrum:

$$\Psi(t, \xi_k) = \Phi(t, \xi_k) b(\xi_k). \tag{1.25}$$

The trace formula is given as [108]:

$$C_n = -\frac{1}{\pi}\int_{-\infty}^{\infty} (2i\xi)^n \ln|a(\xi)|^2 d\xi + \sum_{k=0}^{K-1} \frac{1}{n+1}\left[(2i\xi_k^*)^{n+1} - (2i\xi_k)^{n+1}\right], \tag{1.26}$$

This formula links the NLSE integrals C_n with the scattering coefficient $a(\xi)$ and the set of discrete spectral values ξ_k. The integrals C_n are represented as:

$$C_0 = \int_{-\infty}^{\infty} |q|^2 dt,\ C_1 = \int_{-\infty}^{\infty} q q_t^* dt,\ C_2 = \int_{-\infty}^{\infty} (q q_{tt}^* + |q|^4) dt,$$

$$C_3 = \int_{-\infty}^{\infty} (q q_{ttt}^* + 4|q|^2 q q_t^* + |q|^2 q^* q_t) dt.$$

Specifically, Eq. (1.26) for $n = 0$ yields:

$$C_0 = -\frac{1}{\pi}\int_{-\infty}^{\infty} \ln|a(\xi)|^2 d\xi + \sum_{k=0}^{K-1} 2i(\xi_k^* - \xi_k), \tag{1.27}$$

This result is known as the Parseval equality for NLSE and it helps confirm the accuracy of numerical solutions by checking the balance between continuous and discrete spectral energies. The first term on the right in Eq. (1.27) is the energy from the continuous spectrum:

$$E_c = -\frac{1}{\pi}\int_{-\infty}^{\infty} \ln|a(\xi)|^2 d\xi, \tag{1.28}$$

while the energy from the discrete spectrum is:

$$E_d = \sum_{k=0}^{K-1} 2i(\xi_k^* - \xi_k). \tag{1.29}$$

The way scattering data changes with the distance z is described by:

$$r(\xi, z) = r(\xi, z_0)e^{-2i\xi^2(z-z_0)}, \quad r_k(z) = r_k(z_0)e^{-2i\xi_n^2(z-z_0)}. \tag{1.30}$$

The scattering data combines into the core $\Sigma(z)$, which is a sum of the discrete and continuous parts of the nonlinear spectrum:

$$\Sigma_{dis}(z,t) = \sum_{k=0}^{K-1} r_k(z)e^{-i\zeta_n t}, \tag{1.31}$$

for the discrete part, and

$$\Sigma_{con}(z) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} d\xi r(\xi, z)e^{-i\xi t}, \tag{1.32}$$

for the continuous part. Here N represents the total number of discrete eigenvalues in the signal.

The last part of working with the NFT involves reconstructing the signal from the scattering data. To get the signal $q(t,z)$ back at a certain distance z, we use the core $\Sigma(z,t)$ in two integral equations, known as the Gelfand-Levitan-Marchenko (GLM) equations:

$$\Theta_1^*(t,s) + \int_{-s}^{t} \Sigma(s+\tau)\Theta_2(t,\tau)d\tau = 0, \tag{1.33}$$

$$-\Theta_2^*(t,s) + \int_{-s}^{t} \Sigma(s+\tau)\Theta_1(t,\tau)d\tau + \Sigma(t+s) = 0, \tag{1.34}$$

where $-t \le s < t$ and $0 \le t \le T$, with $\Sigma(t) \equiv \Sigma(z,t)$. We solve for two functions, Θ_1 and Θ_2, using these equations. Once we find these functions, we can recover the signal using the straightforward formula:

$$q(t)|_z = -2\Theta_2^*(t,t), \tag{1.35}$$

at the desired spatial point z.[1].

NFT for Manakov Equation

In the same way we can define Nonlinear Fourier Transform for Manakov Equation (1.6):

$$\begin{aligned} i\frac{\partial q_x}{\partial \zeta} + \frac{\sigma}{2}\frac{\partial^2 q_x}{\partial \tau^2} + \left(|q_x|^2 + |q_y|^2\right)q_x = 0, \\ i\frac{\partial q_y}{\partial \zeta} + \frac{\sigma}{2}\frac{\partial^2 q_y}{\partial \tau^2} + \left(|q_x|^2 + |q_y|^2\right)q_y = 0. \end{aligned} \tag{1.36}$$

[1]We suppress the dependence on z for simplicity: $\Sigma(t) \equiv \Sigma(z,t)$, but actually, $\Theta_2^*(t,s)$ also depends on z — denoted as $\Theta_2^*(z,t,s)$.

The Lax operators of the Manakov Equation are given by

$$L_{\mathrm{M}} = i\begin{pmatrix} \partial_t & -q_1 & -q_2 \\ -\sigma q_1^* & -\partial_t & 0 \\ -\sigma q_2^* & 0 & \partial_t \end{pmatrix}, \tag{1.37}$$

$$M_{\mathrm{M}} = i\begin{pmatrix} \sigma\partial_t^2 + \frac{1}{2}(|q_1|^2 - |q_2|^2) & -\sigma q_1\partial_t - \frac{1}{2}\sigma q_{1_t} & -\sigma q_2\partial_t - \frac{1}{2}\sigma q_{2_t} \\ -q_1^*\partial_t - \frac{1}{2}q_{1_t}^* & -\sigma\partial_t^2 - \frac{1}{2}(|q_1|^2 - |q_2|^2) & 0 \\ -q_2^*\partial_t - \frac{1}{2}q_{2_t}^* & 0 & -\sigma\partial_t^2 - \frac{1}{2}(|q_1|^2 - |q_2|^2) \end{pmatrix}. \tag{1.38}$$

Eigenvalues are found by solving the eigenproblem

$$L\Psi = \xi\Psi, \tag{1.39}$$

Zakharov-Shabat system can be written for the Manakov's equations in the following way (Manakov-Zakharov-Shabat (MZS) problem):

$$\begin{cases} -\partial_t\psi_1 + q_1(t)\psi_2 + q_2(t)\psi_3 = i\xi\psi_1, \\ \partial_t\psi_2 + \sigma q_1^*(t)\psi_1 = i\xi\psi_2. \\ \partial_t\psi_3 + \sigma q_2^*(t)\psi_1 = i\xi\psi_3. \end{cases} \tag{1.40}$$

By the analoge for NLSE ZS problem, the basis vectors for the eigenspace (the space of all eigenfunctions of $\mathbf{L}$) for ME are given by

$$\Phi(t) \to \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \exp(j\xi t) \quad \text{and} \quad \bar{\Phi}(t) \to \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \exp(-j\xi t) \quad \text{for } t \to \infty.$$

The same can be done for the boundary $t \to -\infty$ and this leads to the eigenfunctions

$$\Psi(t) \to \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \exp(-j\xi t) \quad \text{for } t \to -\infty,$$

$$\bar{\Psi}(t) \to \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \exp(j\xi t) \quad \text{for } t \to -\infty.$$

We can select either the set of eigenfunctions at the boundary $t \to \infty$, which are Φ and $\bar{\Phi}$, or the set at the boundary $t \to -\infty$, which are Ψ and $\bar{\Psi}$, to serve as the independent basis for the eigenvalue problem. Then, we can represent the other set using this chosen basis. We choose

the eigenfunctions $\Phi, \bar{\Phi}$ as the basis:

$$\begin{aligned} \Psi &= \bar{\Phi} a(\xi) + \Phi b(\xi), \\ \bar{\Psi} &= \Phi \bar{a}(\xi) + \bar{\Phi} \bar{b}(\xi), \end{aligned} \tag{1.41}$$

with dimensions $a \in \mathbb{C}, \bar{a} \in \mathbb{C}^{2\times 2}, b \in \mathbb{C}^{2\times 1}$ and $\bar{b} \in \mathbb{C}^{1\times 2}$. Just one pair of these coefficients, either a and b or $\bar{a}$ and $\bar{b}$, is sufficient to fully describe the signal. These coefficients, $a(\xi)$ and $b(\xi)$, known as the Nonlinear Fourier coefficients, depend on the spectral parameter ξ and the initial position z_0, but they are independent of the time variable t. This allows us to select the basis vectors at any time point we choose for calculating the Nonlinear Fourier coefficients. Let's select the time as it goes to infinity, $t \to \infty$. We already know the values for Φ and $\bar{\Phi}$ at this time from the boundary conditions:

$$\begin{aligned} \lim_{t\to\infty} \Psi &= \lim_{t\to\infty} \left(\bar{\Phi} a(\xi) + \Phi b(\xi) \right) \\ &= \begin{bmatrix} a(\xi) \\ 0 \\ 0 \end{bmatrix} \exp(-j\xi t) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b_1(\xi) \\ b_2(\xi) \end{bmatrix} \exp(j\xi t) = \begin{bmatrix} a(\xi)\exp(-j\xi t) \\ b_1(\xi)\exp(j\xi t) \\ b_2(\xi)\exp(j\xi t) \end{bmatrix} \end{aligned}$$

We can determine Ψ as time approaches infinity ($t \to \infty$) using the boundary condition set as time approaches negative infinity ($t \to -\infty$). Consequently, all eigenvectors mentioned in (1.41) are known at the same point in time, specifically when t approaches infinity. To calculate the Nonlinear Fourier Transform coefficients, we multiply both sides of the equation by either $\exp(j\xi t)$ or $\exp(-j\xi t)$, depending on whether we are finding the a coefficients or $b_{1,2}$ coefficients, respectively. Then, we pick out the appropriate component from the resulting expressions:

$$\lim_{t\to\infty} \Psi \exp(j\xi t) = \begin{bmatrix} a(\xi)\exp(-j\xi t) \\ b_1(\xi)\exp(j\xi t) \\ b_2(\xi)\exp(j\xi t) \end{bmatrix} \exp(j\xi t) = \begin{bmatrix} a(\xi) \\ b_1(\xi)(\exp(j\xi t))^2 \\ b_2(\xi)(\exp(j\xi t))^2 \end{bmatrix}$$

and

$$\lim_{t\to\infty} \Psi \exp(-j\xi t) = \begin{bmatrix} a(\xi)\exp(-j\xi t) \\ b_1(\xi)\exp(j\xi t) \\ b_2(\xi)\exp(j\xi t) \end{bmatrix} \exp(-j\xi t) = \begin{bmatrix} a(\xi)(\exp(-j\xi t))^2 \\ b_1(\xi) \\ b_2(\xi) \end{bmatrix}$$

The NFT coefficients are thus given by

$$a(\xi) = \lim_{t\to\infty} \psi_1 \exp(j\xi t), \quad b_i(\xi) = \lim_{t\to\infty} \psi_{i+1} \exp(-j\xi t)$$

And we can define the same spectral data:

- The zeros of $a(\xi) = 0$ give us the discrete spectrum $\{\xi_k\}, k = 0, K-1$, with the phase

coefficients defined by

$$r_{k,i} = \left.\frac{b_i(\xi)}{a'(\xi)}\right|_{\xi=\xi_k}, \quad \text{where } a'(\xi) = \frac{da(\xi)}{d\xi}. \tag{1.42}$$

- The continuous spectrum is defined by reflection coefficient

$$r_i(\xi) = b_i(\xi)/a(\xi), \xi \in \mathbb{R}. \tag{1.43}$$

Finding the NFT coefficients comes down to solving the MZS system (1.40). This is a scattering problem: in solving (1.40), we analyze how the eigenfunctions are scattered from $t \to -\infty$ to $t \to \infty$. The NFT coefficients are therefore sometimes called the scattering coefficients. They capture the scattering data of $\mathbf{q}(z_0, t)$ and can also be collected in the socalled scattering matrix $\mathbf{S}(z, \xi)$. We can express (1.41) in terms of the scattering matrix as

$$\begin{bmatrix} \Psi & \bar{\Psi} \end{bmatrix} = \begin{bmatrix} \bar{\Phi} & \Phi \end{bmatrix} \underbrace{\begin{bmatrix} a & \bar{b} \\ b & \bar{a} \end{bmatrix}}_{\mathbf{S}(x,\xi)}$$

The inverse problem consists of recovering the potentials $\mathbf{q} = \{q_1(t), q_2(t)\} \in \mathbb{C}^{1\times 2}$, where the time $t \in \mathbb{R}$, $\xi = \zeta + i\eta \in \mathbb{C}^+$, by left spectral data:

$$\left\{\mathbf{r}(\xi), [\xi_k, \mathbf{r}_k]_{k=0}^{K-1}\right\}. \tag{1.44}$$

Here K is the number of the solitons in the signal. The inverse problem is reduced to solving the left system of integral equations

$$\begin{aligned} &\Theta_1^*(t,s) + \int_{-\infty}^{t} \Theta_2\left(t,t'\right)\Sigma\left(t'+s\right)dt' = 0, \quad t \geq s \\ &\mp\Theta_2^*(t,s) + \Sigma(t+s) + \int_{-\infty}^{t} \Theta_1\left(t,t'\right)\Sigma\left(t'+s\right)dt' = 0 \end{aligned} \tag{1.45}$$

where 2-dimensional kernels are defined for all real t by

$$\Sigma(z) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathbf{r}(\xi)e^{-i\xi t}d\xi - i\sum_{k=0}^{K-1} \mathbf{r}_k e^{-i\xi_k t}. \tag{1.46}$$

After solving the system (1.45), the potential $\mathbf{q}(t, z)$ is restored by the formulas

$$\mathbf{q}(t,z) = -2\Theta_2^*(t,t)$$

For most cases in this work we will use NFT for NLSE.

1.2.3 Solitons

The history of solitons begins in fact in 1834, when James Scott Russell observed a solitary shaft of water in a channel, moving without a noticeable change in shape or decrease in speed over several kilometers [109]. Such waves were called solitary, and later, in 1965, the term soliton was introduced to represent their particle-like essence [110]. However, even earlier, in 1831, M. Faraday [111] described an effect in which a fine powder placed on an oscillating surface gathered in small "heaps" , which could be both stationary and moving. The properties of solitons were studied in the 1960s, when the inverse scattering method was introduced, in which solitons arose as separate solutions [51].

In the 20th century, autosolitons (dissipative solitons) were actively studied in physical, chemical, and biological systems. The main work on this topic is related to the reaction-diffusion model [112, 113], which describes the localized structures that arise in the chemical reactions of activators, or in current flows in plasma, gas and semiconductors.

In nonlinear optics, solitons are divided into temporal and spatial. The temporal solitons are localized in time and related to optical pulses that retain their shape, and the spatial ones are self-directed beams, limited in transverse directions, orthogonal to the propagation vector. The formation and existence of such types of solitons is caused by the optical Kerr effect [114–116] — a nonlinear change in the refractive index of the medium, depending on the light intensity and leading to spatial focusing (or defocusing) and temporal self-modulation phase.

Optical solitons are divided into two large classes: conservative and dissipative. Conservative solitons are created due to the balance of nonlinear focusing and linear spreading in transparent media in which there is no energy pumping, and the radiation loss is negligible. Dissipative solitons (autosolitons) are localized as a result of the balance of the inflow and outflow of energy in the system. For them, there may also be an effect of balancing linear spreading and nonlinear focusing.

These two classes of solitons have both common properties and fundamental differences, as a result of their common nature, but of a different method of formation. The set of basic parameters of dissipative solitons is discrete due to the requirement for the energy balance. This leads to their increased stability, and, as a result, dissipative solitons have prospects in various practical applications. Conservative solitons are determined by a continuously varying parameter, for example, intensity or width. Depending on the method of energy input into the optical system, the solitons are divided into coherent and incoherent. Coherent solitons are formed in a beam of continuous coherent radiation, which determines the frequency and phase. The incoherent signal forms incoherent solitons, for which the common radiation phase is arbitrary.

Due to the presence of noise in real systems, the possibility of spontaneous transition between

soliton and soliton-free structures is realized. The soliton itself, although not a stable structure, can exist for a long time, since it is resistant to small perturbations. Transitions can be caused by large fluctuations that are unlikely in real systems. This fact leads to one interesting feature described in [117], when in the process of signal evolution in the system solitons arise.

The study of optical solitons began in the 60s of the 20th century, when the first optical nonlinear systems appeared, and experiments in this field became available. The first example was the article [118] about the possibility of the existence of a spatial conservative soliton in a transparent medium with a self-focusing nonlinearity. Later, a time soliton was discovered in single-mode fibers with the Kerr nonlinearity [6].

1.2.4 Analytical solutions

Satsuma-Yajima signal

As the first analytical signal for which the nonlinear spectrum is known, we take the following:

$$q(t,0) = A\,\mathrm{sech}(t), \tag{1.47}$$

hyperbolic profile with variable amplitude A. We assume that $A > 0$ is real, because otherwise the phase factor only affects the phase of the solution $q(t,z)$, which in this context does not interest us. In 1974, Satsuma and Yajima showed that the signal (1.47) is a particular solution of the Zakharov-Shabat problem (ZSP) (1.15), and its nonlinear spectrum was found.

They found that the function written as follows:

$$\psi_1(t;\xi) = \begin{pmatrix} y_1^{(1)}(s;\xi) \\ -A(\xi - \frac{1}{2}i)^{-1} y_1^{(2)}(s;-\xi) \end{pmatrix}, \tag{1.48}$$

is the Jost function of the problem (1.15), where the functions $y_1^{(1,2)}$ are defined by:

$$y_1^{(1)}(s;\xi) = s^{i\xi/2}(1-s)^{-i\xi/2} F\left(-A, A, i\xi + \frac{1}{2}; s\right) \tag{1.49}$$

$$y_1^{(2)}(s;\xi) = s^{1/2-i\xi/2}(1-s)^{-i\xi/2} F\left(\frac{1}{2} - i\xi + A, \frac{1}{2} - i\xi - A, \frac{3}{2} - i\xi; s\right) \tag{1.50}$$

in which $F(\alpha,\beta,\gamma;s)$ is a hypergeometric function, $s \equiv \frac{1-\tanh(x)}{2}$. Taking into account the properties of the hypergeometric function:

$$F(\alpha,\beta,\gamma;0) = 1$$

$$F(\alpha,\beta,\gamma;s)=\frac{\Gamma(\gamma)\Gamma(\gamma-\alpha-\beta)}{\Gamma(\gamma-\alpha)\Gamma(\gamma-\beta)}F(\alpha,\beta,\alpha+\beta+1-\gamma;1-s)+$$
$$+\frac{\Gamma(\gamma)\Gamma(\alpha+\beta-\gamma)}{\Gamma(\alpha)\Gamma(\beta)}(1-s)^{\gamma-\alpha-\beta}F(\gamma-\alpha,\gamma-\beta,1+\gamma-\alpha-\beta;1-s)$$

it is easy to calculate the asymptotic behavior of the function $\psi_1(t;\xi)$:

$$\psi_1(t;\xi)\to\begin{pmatrix}1\\0\end{pmatrix}e^{-i\xi t},\qquad t\to\infty \tag{1.51}$$

$$\psi_1(t;\xi)\to\begin{pmatrix}\frac{\Gamma^2(i\xi+\frac{1}{2})}{\Gamma(i\xi+\frac{1}{2}+A)\Gamma(i\xi+\frac{1}{2}-A)}e^{-i\xi t}\\-i\sin(\pi A)\operatorname{sech}(\pi\xi)e^{i\xi t}\end{pmatrix},\qquad t\to-\infty \tag{1.52}$$

The function $\frac{\Gamma^2(i\xi+\frac{1}{2})}{\Gamma(i\xi+\frac{1}{2}+A)\Gamma(i\xi+\frac{1}{2}-A)}$ in front of $e^{-i\xi t}$ in the last expression is nothing more than the coefficient $a(\xi)$ whose zeros give us a discrete spectrum for problem (1.15):

$$\xi_n=i\left(A+\frac{1}{2}-n\right), \tag{1.53}$$

where n is a positive integer satisfying the condition $n<A+\frac{1}{2}$. The first eigenvalue appears at $A=\frac{1}{2}$, the second at $A=\frac{3}{2}$ and so on. Thus, for an arbitrary $A>0$, the number of discrete eigenvalues is equal to the largest integer number n satisfying the condition $n-\frac{1}{2}<A$.

Rectangular pulse

Another example of the initial distribution is a rectangular pulse with varying amplitude:

$$q(t,0)=\begin{cases}A, & 0\le t\le 1,\\ 0, & \text{otherwise.}\end{cases} \tag{1.54}$$

Similarly, with the previous type of signal, we assume that $A>0$ and real. The coefficient $a(\xi)$ takes the form:

$$a(\xi)=e^{i\xi}\left(\cos\sqrt{\xi^2+A^2}-\frac{i\xi}{\sqrt{\xi^2+A^2}}\sin\sqrt{\xi^2+A^2}\right). \tag{1.55}$$

This expression is zero if the expression in parentheses is zero, which leads us to the transcendental equation:

$$\cot\sqrt{\xi^2+A^2}=\frac{i\xi}{\sqrt{\xi^2+A^2}} \tag{1.56}$$

This expression can be satisfied only for imaginary ξ. Each subsequent eigenvalue appears when $A=(n-\frac{1}{2})\pi$, where n is a positive integer. This formula can be rewritten as

$$N=\text{int}[1/2+L_1(q)/\pi], \tag{1.57}$$

where int[...] means the integer part of the expression in brackets, and N is the number of solitons. The first level L_1 of the norm calculated by this formula is 1.57, the second is 4.71 and so on.

1.2.5 NF spectrum associated with finite-extent signals

In practical applications, we do not typically deal with the signals defined on the whole infinite t-axis, but rather operate with the truncated wave-forms, meaning that $q(t)$ is non-zero only inside the finite interval of t. In this case, the NF spectrum of the signal is completely characterised by the coefficient $b(\xi)$ from (1.21), which becomes band-limited, appended with the finite discrete set of solitonic parameters $\{\xi_n, r_n\}$ [70, 71]. When, in addition, the discrete NF spectrum is absent, as it is in the case considered, the whole NF spectrum can be defined using just $b(\xi)$ profile[69], while the coefficient $a(\xi)$ can be expressed through $b(\xi)$ in the following way:

$$a(\xi) = \sqrt{1-|b(\xi)|^2}\,\exp\left[\frac{i}{2\pi}\int\limits_{-\infty}^{\infty}\frac{\ln\left(1-|b(s)|^2\right)}{\xi-s}\,ds\right], \tag{1.58}$$

where the integral in the exponent is understood in the principal value sense. So, in practice, instead of $r(\xi)$ (1.24), it is sufficient to compute the b-coefficient, and then find $a(\xi)$ using Eq. (1.58). If needed, we then can use both computed quantities to find the reflection coefficient (1.24). We note that within the b-modulation concept, which has turned out to be the most efficacious PNFT method developed so far, we utilise the $b(\xi)$ functions as information carriers[69–72].

1.2.6 NF spectrum for the weakly-nonlinear case

Let us assume that the amplitude of our signal is small, say $|q(t)| \sim \epsilon$, with $\epsilon \ll 1$. Then, we can derive the following expansions for the NF scattering coefficients [59]:

$$a(\xi) = 1 - \int\limits_{-\infty}^{\infty} dt_1 \int\limits_{-\infty}^{t_1} dt_2\, e^{2i\xi(t_1-t_2)} q(t_1)\bar{q}(t_2), \tag{1.59}$$

up to ϵ^2 (the next expansion term $\sim\epsilon^4$), and

$$b(\xi) = -\int\limits_{-\infty}^{\infty} dt_1\, e^{-2i\xi t_1}\bar{q}(t_1) + \int\limits_{-\infty}^{t} dt_1 \int\limits_{-\infty}^{t_1} dt_2 \int\limits_{0}^{t_2} dt_3\, e^{2i\xi(t_2-t_1-t_3)}\bar{q}(t_1)q(t_2)\bar{q}(t_3), \tag{1.60}$$

up to ϵ^3 (the next expansion term $\sim \epsilon^5$). With the accuracy up to ϵ^4, we have for the reflection coefficient:

$$r(\xi) = -\int_{-\infty}^{\infty} dt_1\, e^{-2i\xi t_1}\bar{q}(t_1) - \int_{-\infty}^{\infty} dt_1 \int_{t_1}^{\infty} dt_2 \int_{-\infty}^{t_2} dt_3\, e^{2i\xi(t_2-t_1-t_3)}\bar{q}(t_1)q(t_2)\bar{q}(t_3). \tag{1.61}$$

So we see that the first linear term in $r(\xi)$ expansion is simply the conjugated FT of our signal up to the frequency scaling factor. Then, $r(\xi)$ from Eq. (1.61) differs from the expression for $b(\xi)$, Eq. (1.60), only by the terms $\sim \epsilon^3$ and higher, but the structure of both expressions is the same.

1.3 Modulation and Quality metrics

1.3.1 Modulation formats

In optical communication systems, a variety of modulation formats are employed to encode information onto light waves for transmission. Among these, On-Off Keying (OOK) is a basic form where the presence or absence of a light pulse represents binary one or zero, respectively. Phase-Shift Keying (PSK) takes a step further by modulating the phase of the optical signal to represent data, with common variants like Binary Phase-Shift Keying (BPSK) and Quadrature Phase-Shift Keying (QPSK) representing one or two bits per symbol. Differential modulation, such as Differential Phase-Shift Keying (DPSK) and Differential Quadrature Phase-Shift Keying (DQPSK), modulates the phase information relative to the previous symbol, which can be advantageous in certain channel conditions. Pulse Amplitude Modulation (PAM) modulates the amplitude of the optical pulses to encode data.

Bitrate and *baud rate* are fundamental concepts in digital communication systems, serving as measures of data transmission and symbol modulation rates respectively. The bitrate, also known as bit rate or data rate, signifies the rate at which bits are transmitted over a communication channel per unit time. It is quantified in bits per second (bps). The bitrate provides insight into the communication system's capacity to convey information over a period. A higher bitrate denotes a higher data transmission rate, which often translates to faster communication. The *baud rate*, on the other hand, represents the rate of symbol changes or modulations on the communication channel per unit time. It is measured in symbols per second or baud (Bd). In digital communications, a symbol could represent one or more bits depending on the modulation scheme. For instance, in BPSK, one symbol corresponds to one bit, but in Quadrature amplitude modulation (QAM), a symbol could represent multiple bits. The relationship between bitrate and baud rate can be expressed using the following formula, where M is the number of bits per symbol:

$$\text{Bitrate} = \text{Baud Rate} \times M \tag{1.62}$$

In this work, our focus narrows down to Quadrature amplitude modulation, a sophisticated modulation format that encodes data by modulating both the amplitude and phase of the optical signal. QAM combines the principles of both amplitude and phase modulation, thereby enabling the transmission of multiple bits per symbol. This feature significantly enhances the spectral efficiency of the communication system. Commonly used QAM formats include 16-QAM, 64-QAM, and 256-QAM, named for the number of distinct symbols or states they have, which are 16, 64, and 256 respectively. The main parameters of the systems, depending on the constellation diagram, are presented in Table 1.1. Although the first three types of

Table 1.1: QAM formats, Bit rate and Baud rate comparison

Modulation type	Bits per symbol	Baud rate	Bit rate
BPSK	1	1 × Bit rate	1 × Baud rate
QPSK	2	1/2 × Bit rate	2 × Baud rate
8PSK	3	1/3 × Bit rate	3 × Baud rate
16QAM	4	1/4 × Bit rate	4 × Baud rate
32QAM	5	1/5 × Bit rate	5 × Baud rate
64QAM	6	1/6 × Bit rate	6 × Baud rate
256QAM	8	1/8 × Bit rate	8 × Baud rate
1024QAM	10	1/10 × Bit rate	10 × Baud rate

modulation in the Table are called Phase-Shift Keying (PSK), they can be represented via QAM modulation.

In QAM, data is represented by a constellation diagram, where each point on the constellation corresponds to a unique symbol. The position of each point is determined by both its amplitude and phase, allowing for the encoding of multiple bits per symbol. Examples of constellation diagrams used in the work are presented in Fig. 1.3. For instance, in 16-QAM, the constellation diagram consists of 16 distinct points arranged in a square grid, allowing for the transmission of 4 bits per symbol. The decoding process in the receiver involves identifying the closest constellation point to the received signal, from which the encoded bits are then retrieved. The choice of QAM as a modulation format strikes a balance between the data rate, spectral efficiency, and system robustness, making it a compelling choice for high-speed optical communication systems.

1.3.2 Performance Metrics in Optical Communication Systems

In optical communication systems, evaluating the performance and reliability of data transmission is crucial. Prominent metrics such as Bit Error Rate (BER), Error Vector Magnitude (EVM), and Mutual information (MI) are employed for this purpose. Here's an overview of these metrics:

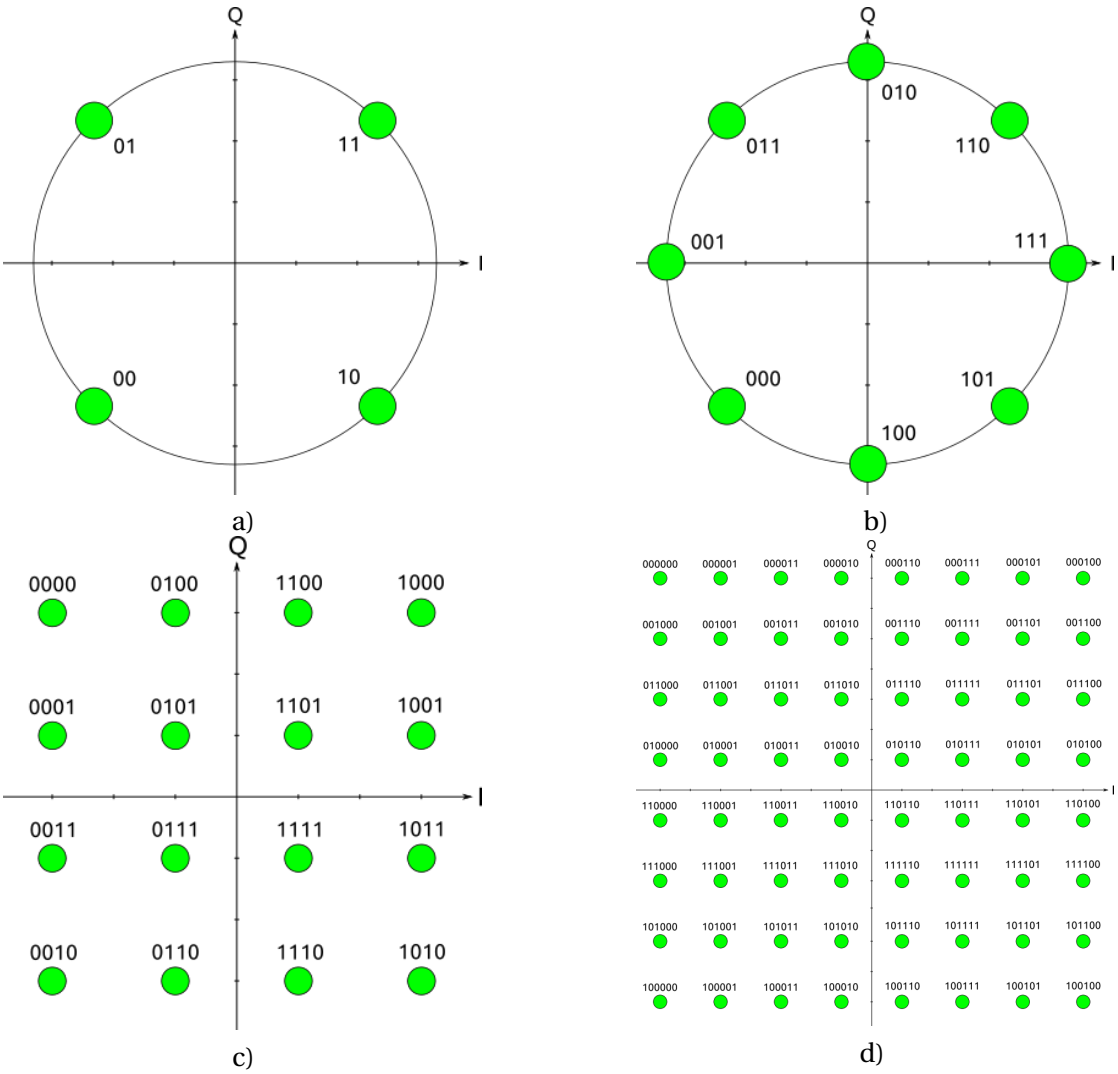


Figure 1.3: Constellation diagrams for (a) QPSK, (b) 8-PSK, (c) 16-QAM and (d) 64-QAM.

Bit Error Rate (BER)

Bit Error Rate (BER) is a pivotal metric that quantifies the number of bit errors (incorrectly received bits) per unit time. It's a direct indicator of the system's error performance.

$$\text{BER} = \frac{\text{Number of bit errors}}{\text{Total number of bits transmitted}} \tag{1.63}$$

The measured (or estimated) BER is usually converted to an equivalent Gaussian noise Q-factor in dB using the expression:

$$Q_{\text{BER}} = 20 \cdot \log_{10}\left(\sqrt{2}\,\text{erfc}^{-1}(2\text{BER})\right), \tag{1.64}$$

where erfc^{-1} is the inverse complementary error function. This sets the reference Q-factor used in the following evaluation of different indirect methods.

Error Vector Magnitude (EVM)

Error Vector Magnitude (EVM) serves as a measure of modulation quality and error performance. It appose the ideal symbol points to the actual symbol points received, providing insight into the distortion induced during transmission.

$$\text{EVM} = \sqrt{\frac{\sum\left(\text{Ideal symbol point} - \text{Received symbol point}\right)^2}{\sum\left(\text{Ideal symbol point}\right)^2}} \tag{1.65}$$

In an optical communication system with QAM format, constellation points represented by complex points. EVM is determined by the formula

$$\text{EVM}_m = \frac{\sigma_{\text{err}}}{|E_{t,m}|},\ \sigma_{\text{err}}^2 = \frac{1}{I}\sum_{i=1}^{I}|E_{\text{err},i}|^2,\ E_{\text{err},i} = E_{r,i} - E_{t,i}. \tag{1.66}$$

where the propagated vector E_r deviates by the vector E_{err} from the ideally transmitted vector E_t, and BER is connected with EVM as

$$\text{BER} \approx \frac{(1-L^{-1})}{\log_2 L}\text{erfc}\left[\sqrt{\frac{3\log_2 L}{(L^2-1)}\frac{1}{(k\text{EVM}_m)^2\log_2 M}}\right]. \tag{1.67}$$

In the formula L is the number of signal levels that are identical in each dimension of the constellation, $\log_2 M$ is the number of bits encoded in each QAM symbol. More information about these parameters can be found in [119].

Mutual Information

MI assesses the amount of information shared between the transmitted and received signals. It's a measure of the system's capacity to convey information reliably.

$$\mathrm{MI} = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{1.68}$$

where $p(x,y)$ is the joint probability distribution of the transmitted signal x and the received signal y, and $p(x)$ and $p(y)$ are the marginal probability distributions of x and y respectively. In our scenario, x and y can be construed as the transmitted and received symbols or bits, respectively, and the probability distribution can be ascertained numerically.

These metrics collectively offer a comprehensive understanding of the system's error performance, modulation quality, and information transmission capacity, serving as vital tools for system analysis and optimization.

1.4 Signal Formats

1.4.1 Wavelength Division Multiplexing

Wavelength Division Multiplexing (WDM) is a pivotal technology in optical communications, enabling the simultaneous transmission of multiple signals along the same fiber-optic cable, each at a unique wavelength. This effectively multiplies the capacity of the channel, making efficient use of the available bandwidth.

The optical signal for WDM transmission is represented as a superposition of modulated signals, each associated with a distinct wavelength or frequency. The expression for $A(t)$ is given by:

$$A(t) = \sqrt{P_0}\sum_{k=1}^{K}\sum_{n=1}^{N_{ch}} c_{kn} f(t-kT)e^{-i2\pi f_n t}, \tag{1.69}$$

where N_{ch} is the number of considered spectral channels, K – number of symbols in signal, P_0 is a average signal power per spectral channel, $f_n = \left(n - \frac{N_{ch}+1}{2}\right)\cdot\Delta$, and Δ is the channel spacing; $c_{k,n}$ is the complex point from modulation constellation (symbol) of the kth symbol on the nth channel, $f(t-kT)$ – pulse shape.

For a system with two polarizations, the optical signal is represented as two separate components $A_x(t)$ and $A_y(t)$:

$$\begin{aligned} A_x(t) &= \sqrt{\frac{P_0}{2}}\sum_{k=1}^{K}\sum_{n=1}^{N_{ch}} c_{kn,x} f(t-kT)e^{-i2\pi f_n t}, \\ A_y(t) &= \sqrt{\frac{P_0}{2}}\sum_{k=1}^{K}\sum_{n=1}^{N_{ch}} c_{kn,y} f(t-kT)e^{-i2\pi f_n t}, \end{aligned} \tag{1.70}$$

where $c_{kn,x}$ and $c_{kn,y}$ are the complex constellation points of the kth symbol on the nth channel for the x and y polarizations, respectively. In this work we always suppose that number of symbols and number of WDM channels for each polarisation are the same.

At the receiver, the received signal $A_{out}(t)$ is demodulated to extract the transmitted symbols. The demodulation process can be represented by the following expression:

$$b_k = \int_{\infty}^{\infty} dt A_{out}(t) f^*(t - kT), \tag{1.71}$$

where b_k is the demodulated symbol, $f^*(t-kT)$ is the complex conjugate of the pulse shape. In the scenario involving two polarizations, $A_{out}(t)$ will be represented as $A_{out,x}(t)$ and $A_{out,y}(t)$ for each polarization, leading to the derivation of $b_{k,x}$ and $b_{k,y}$ respectively.

The matched filter is an essential component in digital communication systems, utilized at the receiver to maximize the Signal-to-noise ratio (SNR) for a given received signal, thereby facilitating optimal symbol detection. The impulse response of a matched filter is crafted to be a time-reversed and conjugated version of the transmitted signal's waveform. In mathematical terms, if $f(t)$ is the transmitted signal waveform, the impulse response $h(t)$ of the matched filter is given by $h(t) = f^*(-t)$, where $*$ denotes complex conjugation.

When the received signal traverses the matched filter, the output at the sampling instant yields the optimal SNR, making it simpler to detect the transmitted symbols amidst noise. The employment of a matched filter is particularly beneficial in environments laden with significant noise and interference, as it markedly enhances the reliability of symbol detection. In our case we have time symmetry and always $f(t) = f(-t)$. So Eq. (1.71) represents matched filter for WDM receiver.

Note that the integral in Eq. (1.71) represents the convolution of the output signal with the function f^*. A similar integral expression, albeit in discrete form, can be observed in Eqs. (1.69) and (1.70). The summation $\sum_{n=1}^{N_{ch}} c_{kn} f(t-kT)$ serves as a discrete analog of the convolution operation between the discrete points c_{kn} and the function $f(t)$. This characteristic is leveraged for practical implementation, aiding in efficient GPU acceleration for numerical simulations, which will be elaborated upon in Chapter 5.

Carrier Pulse

The choice of carrier pulse shape is crucial as it greatly influences the system's performance, particularly regarding bandwidth efficiency and inter-symbol interference.

A common choice for the carrier pulse in WDM systems is the Gaussian pulse, valued for its spectral efficiency and simplicity. The Gaussian pulse has a continuous waveform, with its shape defined by the Gaussian function. Another alternative could be the RC or RRC pulses, frequently chosen due to their ability to control bandwidth and minimize inter-symbol interference.

The choice between RRC and RC filters is driven by the specific requirements of a communication system. However, there are particular advantages associated with the RRC filter which often make it a preferred choice over the RC filter. One of the advantages is the pulse shaping and matched filtering. While the RC filter can be used for pulse shaping at the transmitter or receiver, the RRC filter is designed to be used both at the transmitter and the receiver. When an RRC filter is used at both ends, the combined response is a Raised Cosine filter, which provides optimal ISI mitigation. This split of filtering responsibility between the transmitter and receiver can simplify the design and implementation of the filtering process.

Without loss of generality in this thesis, we consider the following different return-to-zero (RZ) carrier functions. First example is the function which has cosine slopes and flat central part:

$$f(t)=\begin{cases}\frac{1}{2}\left[1-\cos\left(\frac{4\pi t}{T}\right)\right], & 0\le t\le\frac{T}{4}\ \text{or}\ \frac{3T}{4}\le t\le T\\ 1, & \frac{T}{4}<t<\frac{3T}{4}\end{cases} \tag{1.72}$$

where T is symbol interval.

As a carrier function, one can also use other RZ functions, for example one period of cosine function, which shifted to make overall function positive on the time interval.

$$f(t)=\frac{1}{2}\left[1-\cos\left(\frac{2\pi t}{T}\right)\right],\ 0\le t\le T, \tag{1.73}$$

As we mentioned above, one of the most common choice is RC waveform function. In time space, it is expressed as follows:

$$f(t)=\begin{cases}\frac{\pi}{4T}\text{sinc}\left(\frac{1}{2\beta}\right), & t=\pm\frac{T}{2\beta},\\ \frac{1}{T}\text{sinc}\left(\frac{t}{T}\right)\frac{\cos\left(\frac{\pi\beta t}{T}\right)}{1-\left(\frac{2\beta t}{T}\right)^2}, & \text{otherwise.}\end{cases} \tag{1.74}$$

And other is RRC with a roll-off factor β:

$$f(t)=\begin{cases}\frac{1}{T}\left(1+\beta\left(\frac{4}{\pi}-1\right)\right), & t=0,\\ \frac{\beta}{T\sqrt{2}}\left(\left(1+\frac{2}{\pi}\right)\sin\left(\frac{\pi}{4\beta}\right)+\left(1-\frac{2}{\pi}\right)\cos\left(\frac{\pi}{4\beta}\right)\right), & t=\pm\frac{T}{4\beta},\\ \frac{1}{T}\frac{\sin\left(\pi\frac{t}{T}(1-\beta)\right)+4\beta\frac{t}{T}\cos\left(\pi\frac{t}{T}(1+\beta)\right)}{\pi\frac{t}{T}\left(1-\left(4\beta\frac{t}{T}\right)^2\right)}, & \text{otherwise.}\end{cases} \tag{1.75}$$

All waveforms are represented in Fig. 1.4. For NFT analysis, we will focus on the study of symbols with a carrier function in the form (1.72). And for WDM signal analysis we will use RRC (1.72) (as we are interested in real systems).

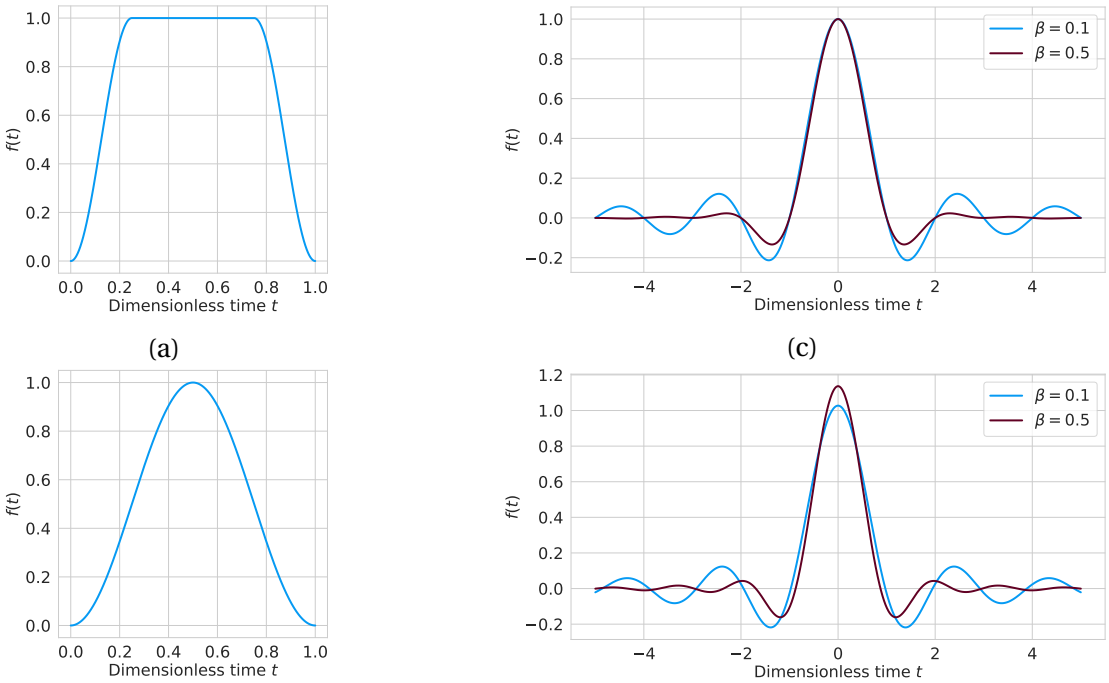


Figure 1.4: Examples of carrier functions for WDM signal. **(a)** function for Eq. (1.72), **(b)** function for Eq. (1.73), **(c)** function for RC (Eq. (1.74)) and **(d)** for RRC (Eq. (1.74)).

1.4.2 Orthogonal Frequency-Division Multiplexing

OFDM signal combines modulation and multiplexing. To form a symbol, the term "subcarrier" is used, which in fact is one of the Fourier harmonics in the frequency space from which the signal is formed. One isolated symbol on a time slot of T is the sum of independent subcarriers modulated according to the chosen modulation type:

$$A_{\text{symb}}(t) = \sum_{k=0}^{N-1} c_k e^{i\frac{2\pi k}{T}t}, \tag{1.76}$$

where N – number of subcarriers, T – symbol duration, c_k – modulated data. Depending on the modulation type, the corresponding constellation diagram is selected (Fig. 1.3).

In practice, the number of subcarriers is chosen as $N = 2^p$ (p is a positive integer) in order to use the Fast Fourier Transform (FFT) algorithms. The input is serial data which is divided into N parallel threads. Each data stream is encoded according to the selected modulation format (QPSK or QAM). Each individual stream corresponds to a specific frequency, and the encoded data determines the complex amplitude for that frequency. Thus obtained N numbers are used further in the FFT, the result of which is the one OFDM symbol.

Orthogonal Frequency Division Multiplexing (OFDM) signal generation in a practical scenario encompasses multiple symbols, a guard interval, and a cyclic prefix to mitigate Inter-Symbol

Interference (ISI) and ensure seamless reception. When transmitting multiple OFDM symbols, the signal $A(t)$ over multiple symbol intervals is represented as a sum over all symbols K and all sub-carriers N as follows:

$$A(t) = \sum_{k=0}^{K-1} \sum_{n=0}^{N-1} c_{kn} \cdot e^{i\frac{2\pi k}{T}(t-kT)} \tag{1.77}$$

where$K$ is the total number of OFDM symbols, T is the symbol duration.

A cyclic prefix is a portion of the OFDM symbol copied and prepended to the symbol to combat ISI. If the cyclic prefix duration is T_{CP}, the signal with the cyclic prefix $A_{\text{CP}}(t)$ is given by:

$$A_{\text{CP}}(t) = \begin{cases} A(t-(T+T_{\text{CP}})k+T_{\text{CP}}), & \text{for } kT \le t < kT+T_{\text{CP}} \\ A(t-k(T+T_{\text{CP}})), & \text{for } kT+T_{\text{CP}} \le t < (k+1)T+kT_{\text{CP}} \end{cases} \tag{1.78}$$

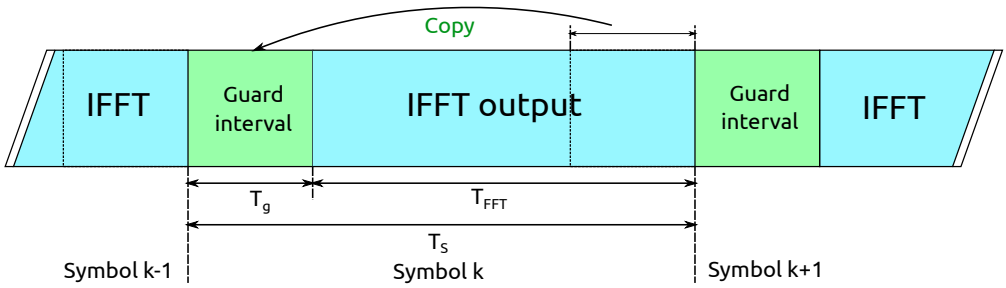


Figure 1.5: Addition of a guard period to an OFDM signal

A guard interval is added between OFDM symbols to prevent inter-symbol interference. The guard interval duration is typically equal to the cyclic prefix duration T_{CP}. The resulting signal $A_{\text{final}}(t)$ which includes multiple symbols, the cyclic prefix, and the guard interval is thus represented by the formula for $A_{\text{CP}}(t)$ over the entire transmission duration (Fig. 1.5). Within the scope of this work, the cycle prefix will not be incorporated into the OFDM signals. Instead, Eq. (1.77) or Eq. (1.76) will be used as references.

To decode a signal, the procedure is performed in the reverse order. The cyclic prefix appended to each OFDM symbol is removed. Once the cyclic prefix is removed, the receiver performs a FFT on each OFDM symbol to convert it from the time domain to the frequency domain. The FFT operation segregates the aggregated OFDM symbol back into its constituent subcarriers. This transformation is the reverse of what was done at the transmitter during the Inverse Fast Fourier Transform (IFFT) operation. Following the FFT, each sub-carrier is individually demodulated to extract the data bits. The demodulation is performed based on the modulation scheme used at the transmitter, for example, QAM or PSK. The data from all sub-carriers is then aggregated to form the complete data stream.

2 Windowed NFT Processing

2.1 Intro

The Nonlinear Fourier Transform (NFT) stands as a potent tool for the analysis and comprehension of nonlinear systems, especially those of a periodic or quasi-periodic nature [53, 108, 120]. In recent years, its application has extended across various domains, including nonlinear and quantum optics [89, 121, 122]. A significant advantage of NFT lies in its capability to compute signal propagation over extended distances through a three-step process [123]: direct NFT, nonlinear spectrum evolution, and inverse NFT. This approach enables the precise modeling of intricate interactions between chromatic dispersion and nonlinear phenomena, such as the Kerr effect. Furthermore, NFT excels at handling signals with arbitrary boundary conditions, rendering it an ideal choice for continuous signal processing.

In light of this, the interest in numerical techniques for solving direct and inverse problems has grown [124–127]. Comprehensive reviews of existing methods are available in [4, 128, 129]. Notably, there exist fast methods for direct and inverse NFT calculations, characterized by lower asymptotic complexity compared to conventional approaches. These are referred to as the Fast Nonlinear Fourier Transform (FNFT) [130–136].

Concerning nonlinear interactions, NFT emerges as a essential tool for enhancing the performance of optical communication systems and augmenting throughput [4, 120, 137–141]. Moreover, NFT yields a more precise depiction of the signal and its nonlinear distortions, fostering the development of more efficient communication systems. This encompasses improved modulation and detection schemes, along with advanced error correction techniques [142–144].

Nonetheless, challenges persist in employing NFT for optical telecommunications. The computational complexity of NFT algorithms can be substantial, especially for systems featuring numerous channels or a broad spectrum of signal parameters. Additionally, implementing NFT-based signal processing may necessitate specialized hardware or software, introducing complexity and elevating system costs. Notwithstanding these challenges, the potential ad-

vantages of NFT in optical telecommunications render it an enticing area of research, with the potential to revolutionize the processing and transmission of optical signals within future communication networks.

The process of partitioning a continuous signal into non-overlapping segments, commonly referred to as the "sliding window" technique, serves as a fundamental method in signal processing. This technique seeks to extract meaningful insights from a signal by independently analyzing each segment or window. The application of NFT to continuous signals presents a notable challenge, as merely segmenting the signal into non-overlapping portions using a sliding window technique falls short of efficient processing.

The idea of the windowing method, as used in the study of hydrodynamic envelope solitons [145–147], involves applying a temporal or spatial window to the data before performing the Inverse Scattering Transform. This technique allows for the analysis of localized structures within a larger dataset by focusing on specific sections where solitonic features are expected or observed. It enhances the ability to detect and analyze solitons in complex systems by isolating their contributions from the surrounding noise or other wave components, thereby providing a clearer understanding of their dynamics and stability.

This part introduces a novel method for enhancing the accuracy and efficiency of continuous signal processing with vanishing boundary conditions using NFT. This method aims to address the challenges associated with traditional NFT algorithms and offers a promising solution for signal processing across diverse fields. Through an in-depth exploration of the underlying mathematical principles and numerical simulations, we demonstrate the effectiveness of this new approach across various scenarios.

We propose an enhanced method that combines Chromatic dispersion compensation (CDC) with the sliding window technique for continuous signal processing within optical communication systems. By pre-compensating chromatic dispersion, we mitigate overlap and distortion among adjacent characters, ultimately improving the performance and applicability of NFT. This approach bears similarities to reducing nonlinearity using neural networks [148–152]. Our method enhances both accuracy and efficiency while resolving issues associated with continuous signal processing, resulting in heightened system performance and increased throughput, particularly in optical telecommunications.

2.2 Sliding window processing

2.2.1 Conventional window

Let's consider a continuous signal, denoted as $A(z = L, t) := A(t)$, which we intend to process using the NFT with vanishing boundary conditions. To clarify, the notation for A in equation (1.3) remains consistent throughout. However, in the case of the Manakov Equation, A represents the vector $[A_x, A_y]$, and all operations are independently applied to each

component.

To achieve accurate processing, it is crucial to account for dispersion that occurs during signal propagation. Dispersion leads to neighboring symbols overlapping and distorting, potentially compromising the accuracy of subsequent signal processing steps.

To address this challenge, we introduce side dispersion intervals, denoted as T_d, on both the left and right sides of a designated processing interval, referred to as T_{proc} (see Fig. 2.1). This approach ensures that the processing interval effectively captures the nonlinear distortions in the signal while allowing the signal to be influenced only by the effects of the side dispersion intervals, rather than other parts of the signal beyond the designated interval.

The length of the dispersion interval, T_d, depends on the propagation distance and can be approximated by:

$$T_d = \beta_2 \times \Omega \times L, \tag{2.1}$$

where β_2 is the group velocity dispersion parameter of the fiber, L is the propagation distance, Ω is the signal bandwidth.

To cut the signal we define the window function $W(t)$ as a combination of Heaviside step functions that selects the time interval for processing:

$$W(t, t_w) = \begin{cases} 1, & t_w - T_d \le t \le t_w + T_{proc} + T_d, \\ 0, & \text{otherwise}, \end{cases} \tag{2.2}$$

where t_w corresponds to the time parameter associated with the current position of the processing window.

Then, we take a signal $A_w(t)$ inside this $T_d + T_{proc} + T_d$ window defined by

$$A_w(t, t_w) = A(t)W(t, t_w) \tag{2.3}$$

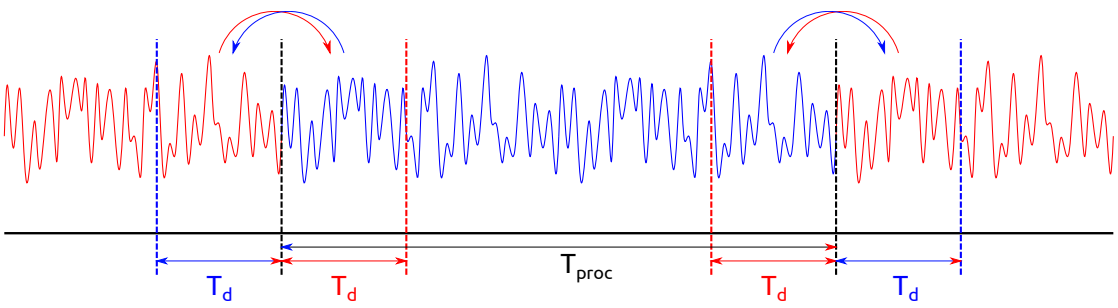


Figure 2.1: Schematic illustration depicting how dispersion intervals from different parts of the signal interact with each other.

and use NFT. It will restore the initial transmitted signal $A_w(z=0,t,t_w)$, but side intervals T_d will be corrupted due to the fact that we windowed the signal and lost information about the previous and following intervals. So we cut again and take only the T_{proc} interval:

$$W_{proc}(t,t_w) = \begin{cases} 1, & t_w \le t \le t_w + T_{proc}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

It will have all the necessary information about nonlinear distortions and other effects from the side intervals T_d. It exclusively considers these effects and disregards any influence from previous or subsequent intervals.

The resulting solution can be expressed by the following equation:

$$A_w(z=0,t,t_w) = W_{proc}(t,t_w) \cdot \mathrm{NFT}[A_w(z=L,t,t_w)]. \tag{2.5}$$

In this equation, A_w represents the value at the output of the processing window. It's calculated as the product of the processing window function, $W_{proc}(t,t_w)$, and the nonlinear Fourier transform of the input signal, which is the product of $A(z=L,t)$ and the window function $W(t,t_w)$ at the fiber's end, located at $z=L$. We can iterate this procedure by incrementing t_w with the value of the processing interval T_{proc}. This allows us to repeat the process for the next position of the processing window.

While this method can be effective in certain scenarios, its numerical accuracy may not always be optimal. To enhance the precision of the calculation methods, it might be necessary to insert zero intervals on both sides of the processing signal. Furthermore, dealing with a signal of high average power can result in a significant number of soliton components, making the computational analysis challenging. Particularly, the computation of nonlinear spectrum phases for solitons can be highly unstable, and as the power of the solitons increases, the numerical calculations may become less reliable.

To address these challenges and enhance the stability of numerical calculations, we propose a more advanced method, which will be described in the subsequent section.

2.2.2 Compensated window

As previously discussed, the dispersion occurring during signal propagation leads to the overlapping and distortion of neighboring symbols. This can significantly compromise the accuracy of subsequent signal processing. However, we can mitigate this issue by leveraging our understanding of the propagation model, such as the Nonlinear Schrödinger Equation for single polarization or the Manakov Equation for dual polarization. We achieve this by effectively "compressing" the signal before processing through the compensation of chromatic dispersion. While this operation restores symbols to their initial positions, it does

introduce some level of corruption due to nonlinear effects during propagation. Following dispersion compensation, we proceed to window the signal and subsequently decompensate the dispersion to obtain a processed signal ready for analysis using the NFT.

To initiate the process, we compensate for the dispersion across the entire signal using the equation:

$$A_{CD}(t) = F^{-1}F[A(t)]e^{i\phi_{CD}(\omega)}, \tag{2.6}$$

Here, $F[\cdot]$ and $F^{-1}[\cdot]$ represent the forward and inverse linear Fourier transforms, respectively. The phase shift ϕ_{CD} can be readily determined from equations (1.3) and (1.5) by setting $\gamma = 0$ and $\alpha = 0$. It is given by:

$$\phi_{CD} = -\frac{\beta_2}{2}\omega^2 L. \tag{2.7}$$

Here, β_2 denotes the second-order dispersion coefficient of the fiber, ω is the angular frequency, and L represents the length of the fiber.

Subsequent to dispersion compensation, we apply the same window operation as described in equation (2.3). We then proceed to decompensate the dispersion using the same phase shift (2.7). This results in:

$$A_{w,CD}(t, t_w) = A_{CD}(t)W(t, t_w), \tag{2.8}$$

$$A_{\tilde{w}}(t) = F^{-1}F[A_{w,CD}]e^{-i\phi_{CD}(\omega)}, \tag{2.9}$$

The final solution for a processing window using the dispersion-compensated window mode is depicted below:

$$A_{\tilde{w}}(z = 0, t, t_w) = W_{proc}(t, t_w) \cdot \mathrm{NFT}[A_{\tilde{w}}(z = L, t, t_w)]. \tag{2.10}$$

Similar to the conventional window mode, we can increment $t_w = t_w + T_{proc}$ to continuously process the full signal.

For the practical implementation of the signal recovery process using the NFT, we need to introduce additional variables. To construct a time window for the recovery of the full signal, we employ the following expression:

$$T_w = T_{proc} + 2 \cdot (R_d \cdot T_d + T_z); \quad T_{proc} = N \cdot T_s. \tag{2.11}$$

Here, T_{proc} represents the processing interval with N symbols, and T_s denotes the duration of one symbol interval. The term R_d acts as a scale factor for the dispersive length, while the term T_z corresponds to the number of additional empty symbol slots added to each side of the processing interval, enhancing the accuracy of the NFT.

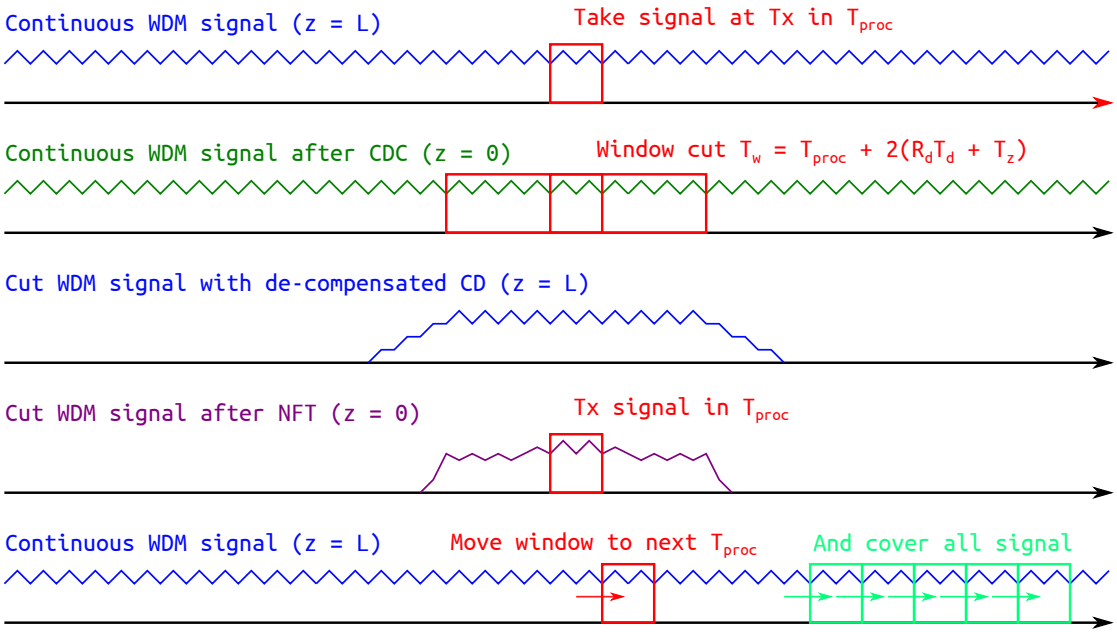


Figure 2.2: Schematic representation of the processing procedure for a continuous WDM signal utilizing NFT. The initial step involves performing CDC for the entire signal. Subsequently, we extract a window of size T_w (as defined by Eq. (2.11)) and compensate for the dispersion. The resulting signal undergoes NFT processing to recover the transmitted signal. This procedure is repeated iteratively for successive processing intervals.

The final procedure for processing a continuous WDM signal with NFT can be summarized as follows (showed at Fig. 2.2):

- Compensate the dispersion of the propagated signal without additional equalization.
- Cut the required window in the compensated signal.
- Decompensate the dispersion for the windowed signal.
- Process the signal in the window using NFT (as it is already localized).
- Collect the full signal.

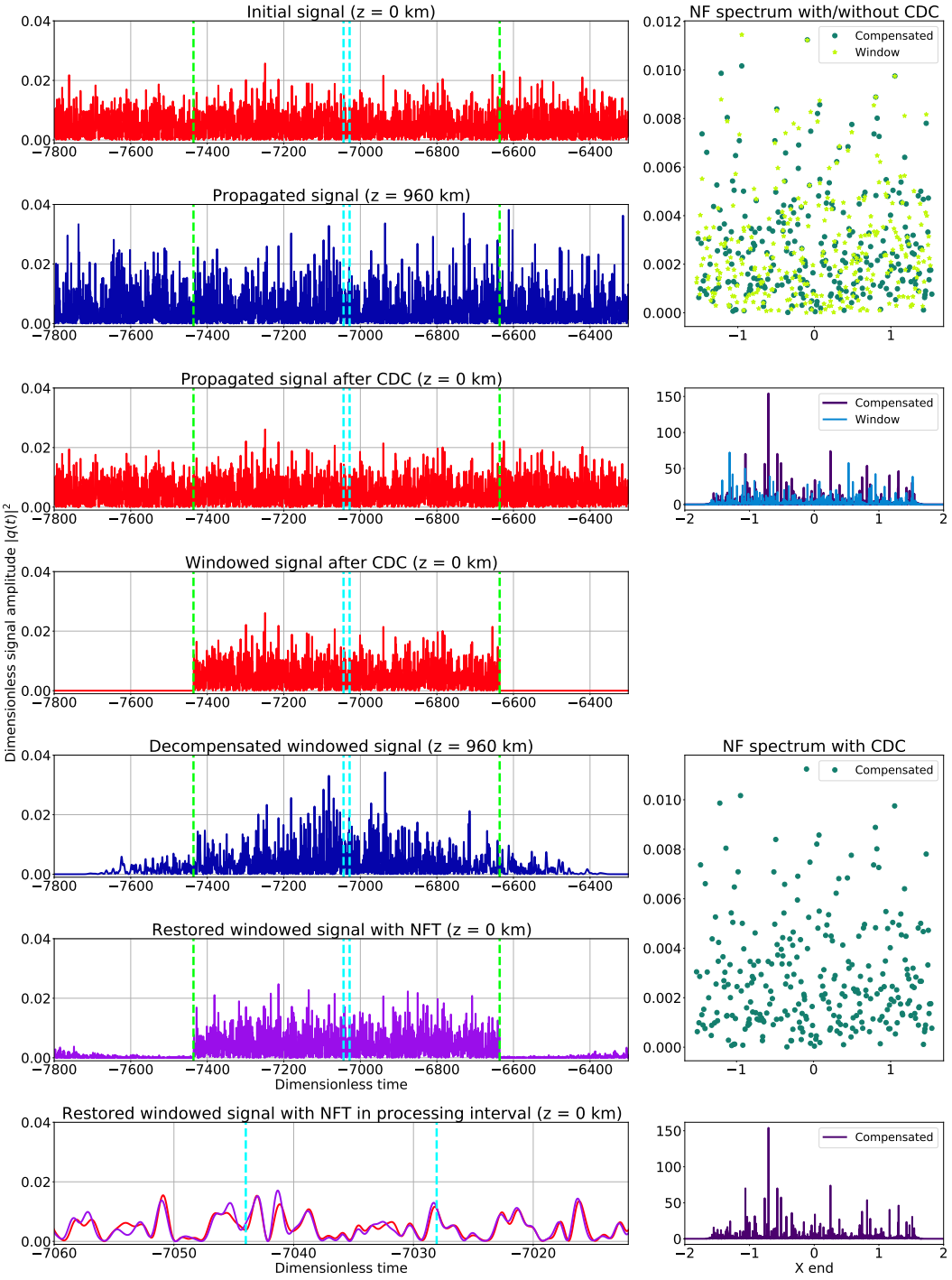


Figure 2.3: Scheme of using compensated window mode. The **left** column shows the signal processing steps. The **right** column shows an example of a signal NF spectrum processed without dispersion precompensation (window) and with the method described above (compensated).

2.3 Results

2.3.1 One polarization (NLSE case)

The first step in researching the window method was to apply it to the NLSE as a simpler model on which we could determine the method's areas of applicability. The application of the window method to the Manakov Equations will impose additional restrictions due to the complexity of the model. Additionally, calculations for the NLSE are faster.

To evaluate the effectiveness of the proposed method, we utilized a single-channel 16-QAM WDM signal (see Eq (1.1)). Simulations involved transmitting the signal over 12×80 [km] spans of standard single-mode fiber (SSFM), complemented with distributed Raman amplification (DRA). Two scenarios were considered: one noiseless scenario and another with DRA noise, mimicking an erbium-doped fiber amplifier noise figure of 4.5 [dB]. The fiber had an attenuation coefficient of $\alpha = 0$ [dB/km], a dispersion coefficient of $D = 16.8$ [ps/{nm$\cdot$km}], and a nonlinear coefficient of $\gamma = 1.2$ [W$\cdot$km]$^{-1}$. Calculations were performed with 2 samples per symbol.

Figure 2.3 illustrates the scheme for utilizing the compensated window mode. The left column displays the signal processing steps. The first plot depicts the initial continuous signal, while the second plot illustrates its propagation over a specified distance z. The third plot showcases the signal after dispersion compensation. Subsequently, the fourth graph demonstrates how the dispersion-compensated signal is windowed within a selected interval. This interval is carefully chosen to ensure that the central (processing) interval remains unaffected by the dispersion spreading from the cut portions of the signal. In the fifth step, dispersion is reintroduced to the signal. Finally, the sixth plot displays the result of NFT reconstruction of the initial signal. Consequently, the central interval of the signal is restored, denoted by the cyan vertical straight lines. After processing one interval, we proceed to the next, sequentially processing the entire signal.

The right column provides an example of an NF spectrum of a signal processed without dispersion precompensation (window) and with the method described above (compensated). In the upper graph, which compares the two spectra, a notable feature in the discrete spectrum becomes apparent: when examining the maximum eigenvalues in the imaginary part, it becomes evident that some of them coincide. Conversely, another subset of the windowed-mode points deviates from the compensated-mode points (we are specifically considering only the maxima in the imaginary part). We attribute this deviation to the influence of the external (clipped) segments of the signal on the processed interval.

In Fig. 2.4 shows a comparison of the algorithms for NFT-CDC (dispersion precompensated) and for NFT without additional processing. It can be seen from the graph that although the NFT tries to repeat the waveform, it lacks reconstruction accuracy. For a pre-compensated signal, the reconstruction accuracy is higher. We also noticed that, thanks to this approach, we automatically satisfy the condition of zeroing the boundary conditions at the ends of the

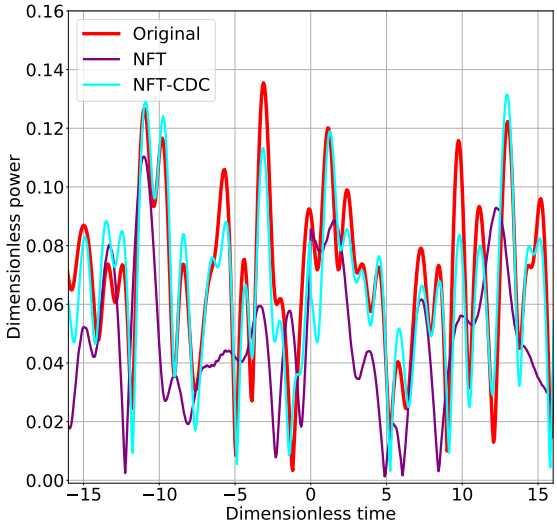


Figure 2.4: Example how NFT algorithm restore initial signal (red). Cyan – window mode with preprocessed chromatic dispersion compensation, purple – NFT restoration for window mode.

time interval to use the forward NFT. Thus, there is no need to fictitiously select a function at the ends of the signals for zeroing. This also reduces the error received.

The presented approach has several parameters for optimization, such as T_{proc}, R_D, and the full window size T_w. For a corresponding propagation distance of 960 km, the dispersion length T_d equals 297 symbols or approximately 440 ps with $T_s = 14.8$ ps.

We performed the inverse NFT using the fast layer-peeling method from the FNFT library (Ablowitz-Ladik scheme) [133]. It operates only with a number of computational nodes equal to the degree of 2. Therefore, for such a long propagation distance, the minimum T_w must be at least equal to 1024 (≈ 1.5 ns) to include $2T_d$ intervals. We also considered a window size of 2048 symbols (≈ 3 ns) in total. The symbol interval of 4096 symbols turned out to be too large for numerical computations due to the $\mathcal{O}(N \log^2 N)$ complexity growth.

Direct NFT was performed in two separate steps. First, the phase jump tracking (PJT) [153, 154] with an adaptive step size was applied for discrete eigenvalue search. This method used an effective exponential scheme of 6th order (ES6) with Padé approximation of the matrix exponential [155]. Despite this, the discrete eigenvalue search was the most time-consuming step of one calculation. Second, continuous spectrum computation was carried out using the fast FNFT_TES4_5B scheme [156, 157], which provided the most stable results in conjunction with the inverse FNFT scheme. We used upsampling of the continuous spectrum by a factor of up to 16 times to enhance the stability of the inverse transform and improve the accuracy of phase jump detection on the real axis.

One of the key practical considerations in our study is determining the appropriate pro-

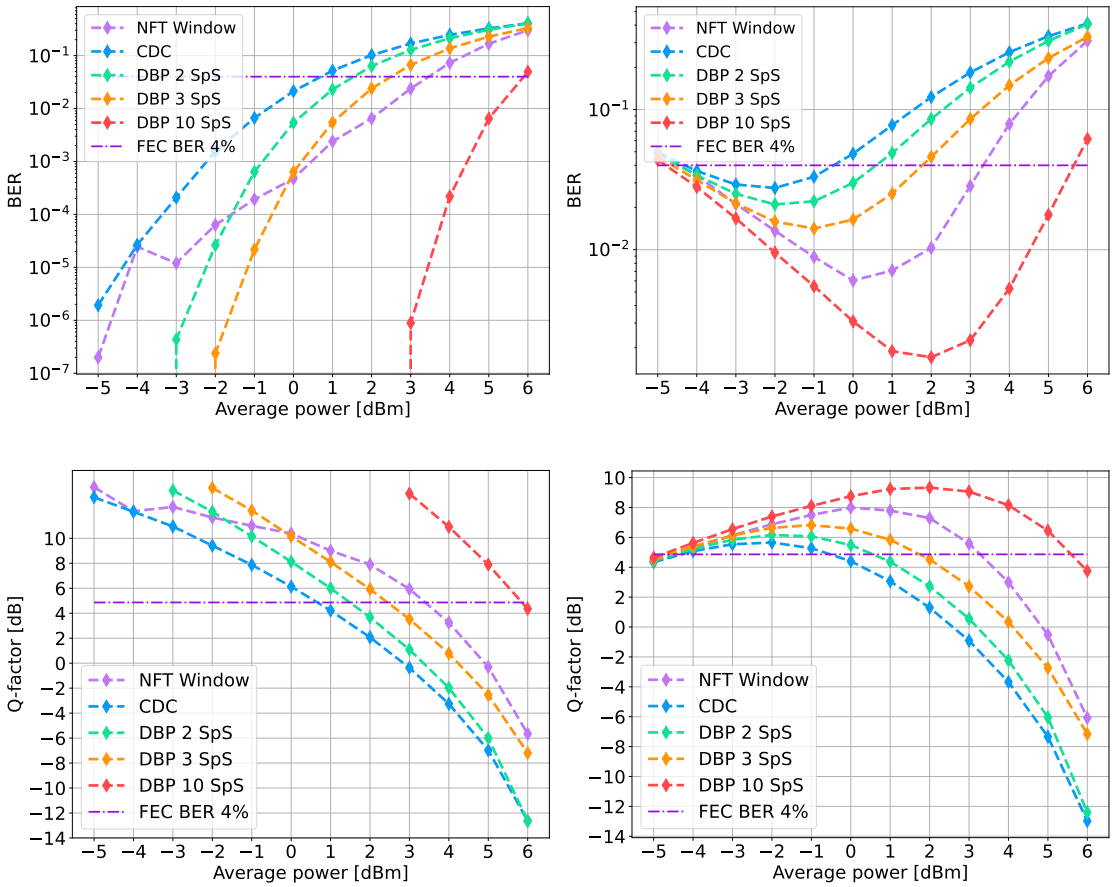


Figure 2.5: The dependence of BER (upper row) and Q-factor (lower row) versus the average signal power P_0. The **left column** corresponds to the noiseless case, the **right column** corresponds to the noise power of 4.5 dB. Total window size is $T_w = 2048$. Processed symbol count $T_{proc} = 512$. The dispersion broadening scale factor $R_d = 1.2$.

cessing interval size, T_{proc}. To minimize additional computational effort compared to the conventional approach, we aimed to maximize the processing interval size. Our investigations revealed that a window of $T_w = 2048$ symbols is a more promising approach, as its wide zero tails (T_z) result in a small step in the frequency domain, which improves the accuracy of optical field restoration.

In our subsequent investigations, we adopted a dispersion broadening scale factor of $R_d = 1.2$, when processed symbol count was set to $T_{proc} = 512$. We also examined the influence of noise on different values of the average power P_0. Figure 2.5 presents a comparison between the noiseless case and the case with noise of power 4.5 dBm. We found that the presence of noise only slightly increases the discrete spectrum size.

Across all considered average power values, the NFT window approach outperformed DBP with 2 steps per span in terms of Bit Error Rate (BER) and Error Vector Magnitude (EVM). In some nonlinear regimes, the NFT approach demonstrated a greater accuracy than DBP with 3 steps per span. However, it should be noted that the accuracy of the NFT approach cannot be compared to that of DBP with 10 steps per span.

2.3.2 Two polarization (Manakov equation case)

The dual-polarization scheme introduces a second polarization for the NLSE equation and the interactions between the two polarizations. For real-world parameters, the effects of the presence of two polarizations are sufficiently small and can be neglected in the first-order approximation. Therefore, we use the approach, introduced in the paper [140], to deal with DP signals. The authors introduced a novel dual-polarization transmission scheme with reduced complexity that separately processes each polarization component. Instead of applying the more complex $\mathrm{NFT_M}$ – the NFT approach based on the Manakov Equation for encoding and decoding information on the nonlinear spectrum – they proposed using independent $\mathrm{NFT_{NLS}}$ processing based on the NLSE channel for each polarization component of the signal. They demonstrated that, in a certain range of system parameters where performance is dominated by the effect of noise on the nonlinear spectrum, such reduced complexity processing can even provide a performance improvement compared to full vector processing, despite the mismatch between the channel model and processing.

Thus, we processed each polarization separately using NFT for NLSE. This approach proved to be effective, and we managed to achieve a high level of performance.

One projection of BER values for 12 spans is presented in Fig. 2.6. Here, the result was obtained by averaging over 100 launches for each value of the average power P_0. When compared to signal recovery, which was modeled by NLSE, accuracy degradation is noticeable. Previously, NFT allowed signal recovery with higher accuracy than that of DBP with 3 steps per span. Now, NFT is applied separately to each polarization. However, the accuracy of a 2-step/span DBP is achieved, and for higher powers, the accuracy approaches that of a 3-step/span DBP.

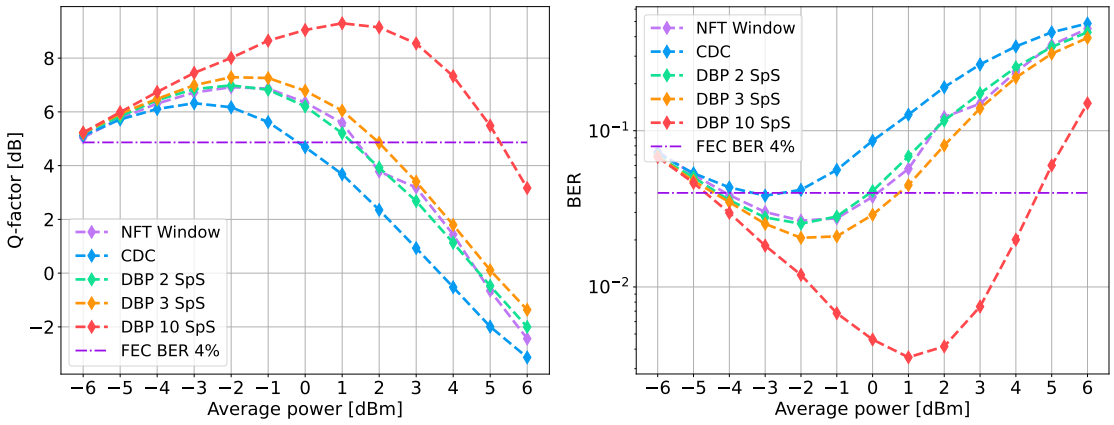


Figure 2.6: The dependence of BER **(left)** and Q-factor **(right)** versus the average signal power P_0. The left column corresponds to the noiseless case, the right column corresponds to the noise power of 4.5 dBm. Total window size is $T_w = 2048$. Processed symbol count $T_{proc} = 512$. The dispersion broadening scale factor $R_d = 1.2$.

2.4 Conclusion

In summary, in this part of the work we introduce a novel method for enhancing the accuracy and efficiency of continuous signal processing using the Nonlinear Fourier Transform, a complex problem in the field of signal processing. The proposed method effectively addresses the limitations of traditional NFT algorithms and offers a promising solution for signal processing applications across diverse fields.

We have conducted an extensive analysis of the underlying mathematical principles and supported our findings with numerical simulations, illustrating the efficacy of this proposed approach in various scenarios. Our methodology is applied to both the Nonlinear Schrödinger Equation, simulating single polarization signal propagation in optical systems, and the Manakov Equation, which characterizes light propagation in a two-polarization optical fiber.

The method leverages a sliding window technique, dividing a continuous signal into non-overlapping segments, with each segment undergoing a nonlinear operation like NFT to transform it into a discrete representation. The size of the window, a crucial parameter influencing the accuracy and resolution of the analysis, is optimized through numerical simulations. There exists a direct correlation between the window size employed in a processing system and the overall system speed. Larger window sizes allow for more data to be processed at once, resulting in increased processing speed. However, it's important to note that there may be diminishing returns as window sizes become excessively large, as the heightened processing power required to handle larger datasets might offset the gains in processing speed. Therefore, achieving an optimal balance between window size and processing power is imperative for system performance optimization.

We evaluated the proposed method using a single-channel Wavelength Division Multiplexing signal, a common feature in optical communication systems. The results demonstrate that the proposed method yields a lower Bit Error Rate compared to the traditional NFT algorithm, underscoring its effectiveness.

This approach paves the way for practical implementation in real communication systems. Its adaptability and flexibility make it suitable for a wide array of signal processing applications spanning various fields, including nonlinear optics and quantum optics.

Future research in this domain could delve into investigating the impact of point drift on signal reconstruction quality using NFT, as well as the development of optimization techniques for various parameters within the proposed method, such as window size, with the aim of further enhancing accuracy and efficiency.

As a final note, we want to stress that an extensive part of this analysis was related to code creation, which can be found in the GitHub repository at nft-processing. The repository includes comprehensive code examples and Jupyter notebooks to reproduce the results.

3 Evaluating OFDM and WDM Soliton Content through Nonlinear Fourier Analysis

3.1 Criteria Analysis for the Existence of Discrete Spectrum

Above, we saw that there is a criteria for signals of a certain form, which determines the possibility of the existence of discrete eigenvalues in the Zakharov-Shabat problem. The imprecise criterion described in [99] is that if L_1 norm, defined by the formula

$$L_1(q) = \|q(t,0)\|_{L_1} = \int_{-\infty}^{+\infty} |q(t,0)|dt \tag{3.1}$$

(where $q(t,z)$ is signal to study, t and z — time and space coordinates), is less than 1.317, then the system (1.15) does not have a discrete spectrum.

Later, in 2003, Klaus and Shaw found an exact criterion in [158]. For the complex initial condition $q(t,0)$, there is no discrete spectrum if

$$\|q(t,0)\|_{L_1} \leq \frac{\pi}{2}. \tag{3.2}$$

If L_1 norm is greater than a given value, then a discrete spectrum may exist (but not necessarily exist).

In this study we use both the L_1 norm (3.1) and the L_2 norm

$$L_2(q) = \|q(t,0)\|_{L_2} = \int_{-\infty}^{+\infty} |q(t,0)|^2 dt, \tag{3.3}$$

for which there is no criterion like (3.2). However, for numerical simulation, we can obtain a criterion that can give us some preliminary information about the L_2 levels of the norm, and therefore about the average signal power, less than which there cannot be any discrete spectrum in the signals. Without loss of generality, we consider the signal $q(t,z)$ for a fixed spatial coordinate z. Subsequent calculations are performed in dimensionless units.

$$L_1 = \|q(t,z)\|_{L_1} = \sum_{n=0}^{N-1} |q_n| \Delta t, \tag{3.4}$$

$$L_2 = \|q(t,z)\|_{L_2} = \sum_{n=0}^{N-1} |q_n|^2 \Delta t, \tag{3.5}$$

where $\Delta t = T/N$, T is the time interval on which the function $q(t,z)$ is defined, N is the number of sampling points, q_n are the values of the function $q(t_n,z)$ at these points. Since we have some finite set of q_n values, we can calculate the average value for this set

$$< |q_n| >= \frac{1}{N} \sum_{n=0}^{N-1} |q_n| = \frac{L_1}{N\Delta t} = \frac{L_1}{T}, \tag{3.6}$$

$$< |q_n|^2 >= \frac{1}{N} \sum_{n=0}^{N-1} |q_n|^2 = \frac{L_2}{N\Delta t} = \frac{L_2}{T}, \tag{3.7}$$

where the brackets $< ... >$ denote the operation of taking the average. The modulus of complex numbers is real and satisfies the condition $|q_n| \geq 0$. Also the value of L_1 is equal to or greater than 0 by definition. Now remember that the average of the square of a positive value is greater than or equal to the square of the average of this value (generalized mean inequality), one can write

$$< |q_n|^2 > \geq < |q_n| >^2 \quad \Rightarrow \quad \frac{L_2}{T} \geq \frac{L_1^2}{T^2}. \tag{3.8}$$

It remains to obtain the constraint on L_1^2 from the criterion 3.2). First of all, we note that $\frac{\pi}{2} > 1$, which implies

$$L_1 > \frac{\pi}{2} \quad \Rightarrow \quad L_1^2 > \frac{\pi^2}{4} \quad \Rightarrow \quad \frac{L_1^2}{T} > \frac{\pi^2}{4T}. \tag{3.9}$$

We can conclude that the criterion for the existence of discrete eigenvalues of the Zakharov-Shabat problem for the L_2 norm:

$$L_2 > \frac{\pi^2}{4T}. \tag{3.10}$$

It is not surprising that in the limit $T \to \infty$ the criterion reduces to a simple $L_2 > 0$. However, in most problems, the signal is localized in time, so the integral over an infinite interval in the formula (3.3) can be replaced by an integral over a finite interval, where the signal is not zero

$$\widetilde{L_2} = \|q(t,z)\|_{\widetilde{L_2}} = \int_{-T/2}^{T/2} |q(t,z)|^2 dt, \tag{3.11}$$

for which the criterion (3.10) works.

3.2 Soliton Content

3.2.1 Methodology

The method of nonlinear Fourier transform allows us to examine the nonlinear component of the signals that were developed in the framework of the linear theory. The mathematical apparatus of the nonlinear Fourier transform allows a better understanding of the structure of the signals, as well as their features associated with nonlinear effects. Standard signals can (at a certain power level) contain coherent structures — solitons. Solitons can be represented as $q(t) = A\text{sech}(At+\delta)e^{2i\omega t+i\theta}$ and are of interest from an applied point of view, since they contain four variable parameters in which information can be encoded: soliton amplitude A, time shift δ, soliton complex frequency ω and phase shift θ.

The fact of the presence of solitons in a signal can be detected using the direct Zakharov-Shabat problem. The spectrum of the operator L in the problem (1.16), where the function $q(t,z)$ is the signal to study, can give a representation of the structure of this signal. If there are discrete values in the spectrum, then there are also solitons. This feature will be used for further analysis of standard optical signals, and the number of solitons is determined by the formula (A.38). By standard signals, we mean WDM and OFDM systems, which are now widely used for optical communication (see Section 1.4). For the futher analysis of signals, we use numerical algorithms, which are described in Appendix A.1 and previously tested on test signals: a rectangular pulse and a Satsuma-Yajima signal.

The signal parameters at which solitons can exist depend on the system. For further study, we consider two parameters inherent in each signal. They are the L_1 and L_2 norms, calculated as follows:

$$L_1(q) = \|q(t,z)\|_{L_1} = \int_{-\infty}^{+\infty} |q(t,z)|dt, \tag{3.12}$$

$$L_2(q) = \|q(t,z)\|_{L_2} = \int_{-\infty}^{+\infty} |q(t,z)|^2 dt, \tag{3.13}$$

where $q(t,z)$ is signal to study, t and z — time and space coordinates.

In general, signals may comprise multiple solitons. When the average power of a signal coincides with the soliton existence threshold, the resulting set of signals with random data may include those without solitons or those with one or more solitons. The left panel of Fig. 3.1 depicts a scenario in which solitons within an OFDM symbol are rare, chosen specifically by setting the symbol average power to -20 dBm. This power level demonstrates that solitons are not always present, highlighting their sporadic emergence influenced by the random distribution of data within the symbol. It is observed that symbols can contain multiple solitons, exhibiting a pattern that aligns with Poisson statistics ($P(x;\lambda) = e^{-\lambda}\lambda^x/x!$), as illustrated on the graph. Conversely, the right panel contrasts with a higher average power of -15 dBm, ensuring that every symbol contains at least one soliton, regardless of its internal data structure.

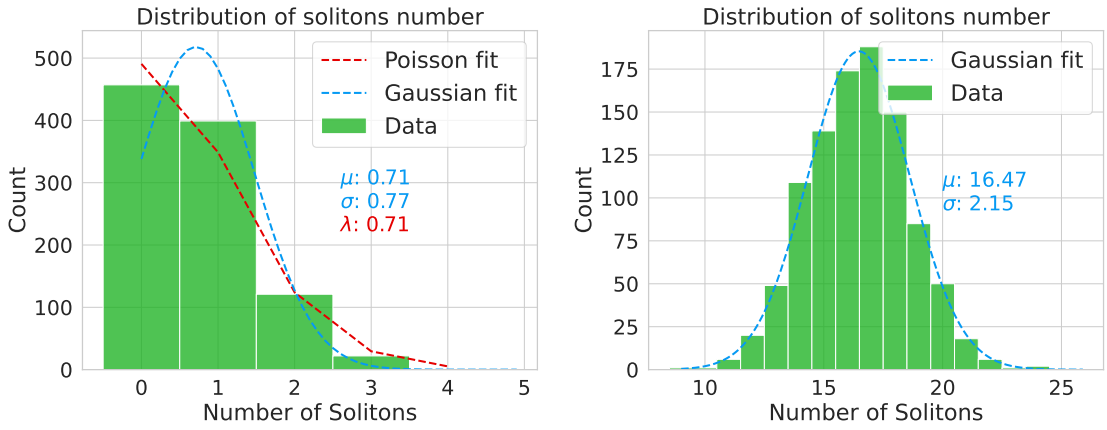


Figure 3.1: The probability of soliton occurrence in an OFDM signal with QPSK modulation, 64 subcarriers and an average power of -20 dBm **(left)** and -15 dBm **(right)**.

This situation reflects a consistent probability of soliton presence, with the average number of solitons being influenced by the power level but showing variability due to the random nature of the OFDM symbol's internal data. This variability adheres to a Gaussian distribution $G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, where μ is the mean and σ is the standard deviation. This emphasize the influence of random data on the statistically determined number of solitons within a symbol.

In the first phase of our study, we focus exclusively on the occurrence of solitons within the signal. Notably, we quantify the presence of solitons rather than their quantity or individual properties. Consequently, we define the probability of a discrete spectrum's presence as the proportion of signals containing at least one soliton relative to the total number of signals with specific parameters.

The signal's average power, denoted as P_{ave}, correlates with the L_2 norm (see Equation 3.3) through the relationship $P_{\text{ave}} = \frac{L_2}{T}$, where T represents the symbol interval, the duration of a single symbol. Power measurements adopt milliwatts (mW) and decibels-milliwatts (dBm), with the conversion given by $P_{\text{dBm}} = 10\log_{10}(P_{\text{mW}})$.

For our numerical simulations, we apply specific optical fiber parameters: the group velocity dispersion $\beta_2 = -21.5$ ps^2/km and the Kerr nonlinearity coefficient $\gamma = 1.27$ W^{-1}/km. It is crucial to note that at this stage, we have not considered signal propagation, meaning our analysis pertains solely to the initial conditions, framing it as a Cauchy problem for the NLSE.

To ascertain the operational ranges for optical signals, we reference the soliton existence criterion as expressed in Eq. (3.10), given in dimensional form by

$$P_{\text{ave}} > \frac{|\beta_2|\pi^2}{4\gamma T_s^2}, \tag{3.14}$$

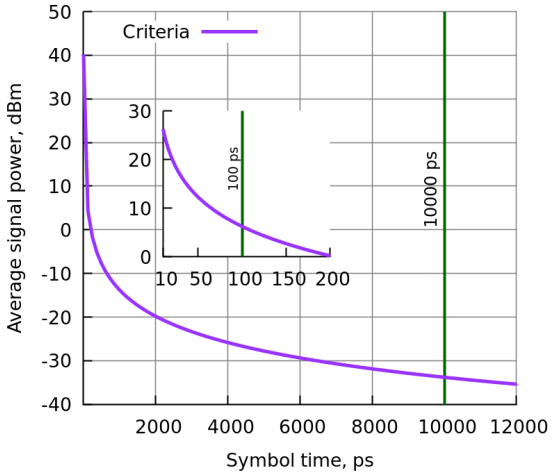


Figure 3.2: The dependence of the criteria (3.14) in dimensional units on the symbol duration T.

where T_s denotes the interval of interest, equivalent to the duration of a single symbol in OFDM or WDM signaling. Figure 3.2 illustrates the relationship between the minimum average power necessary for the potential existence of a discrete spectrum and the symbol duration. The symbol durations examined herein are highlighted by green lines. For instance, at 10 ns, the threshold lies below -30 dBm (specifically, -33.8 dBm), whereas at 100 ps, the requisite threshold significantly increases to 6.2 dBm. These thresholds guide the selection of appropriate power levels for examining each signal type. Notably, the graph displays the aggregate average power; however, in the context of WDM systems, the average power per channel is typically the metric of interest. This distinction is accounted for in subsequent recalculations.

3.2.2 OFDM symbol

The mathematical formulation of an OFDM symbol generation is detailed in Section 1.4.2, specifically in Equation 1.76. Dimensionally, we investigate OFDM symbols with a duration of 10 ns and modulation schemes including QPSK, 16-QAM, 64-QAM, and 1024-QAM. The count of subcarriers ranges from 16 to 1024, with a full Fast Fourier Transform (FFT) size of 1024. Initially, our analysis sought to determine if soliton numbers within a signal were affected by changes in FFT size. Our findings indicate that the soliton count remains invariant, provided the FFT size is fixed at 128.

Further analysis was conducted on 200 simulated symbols with random input data, holding the parameters constant for each point graphed. The Mersenne Twister algorithm, mt19937, was utilized for generating 32-bit pseudo-random numbers with a 19937-bit state size. Figure 3.3 presents the correlation between the probability of soliton formation in a signal and the L_1 norm, for a configuration with 128 subcarriers modulated by QPSK and 16-QAM. The baseline

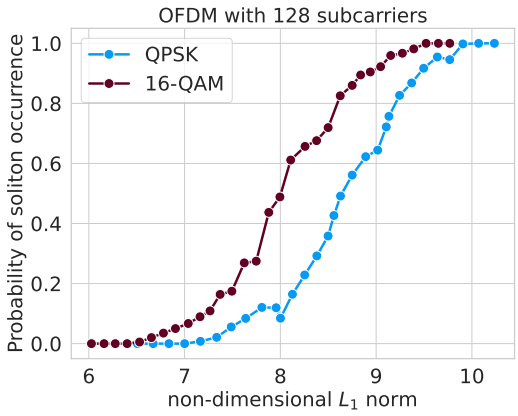
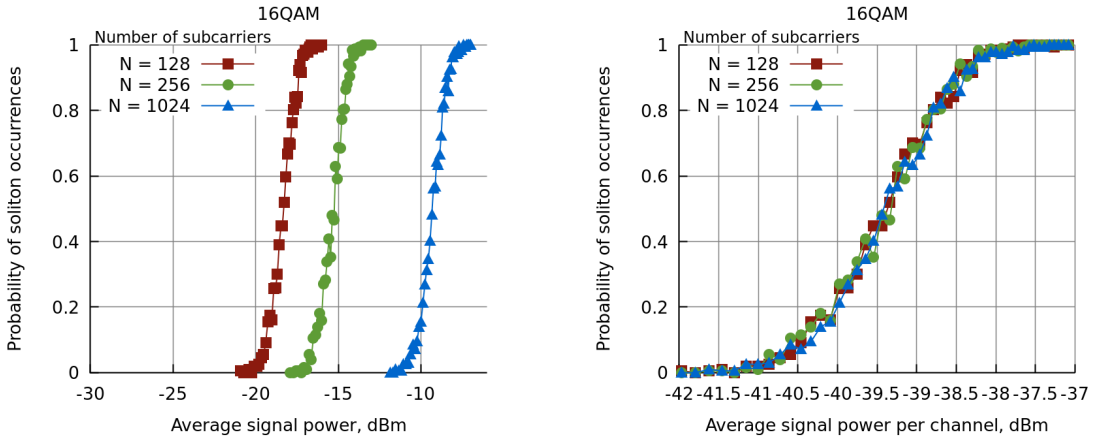


Figure 3.3: Dependence of the probability of soliton occurrence in OFDM signals with 128 subcarriers and QPSK and 16-QAM modulation.

L_1 norm value, calculated as per Equation 1.57 and equal to 1.57, falls outside the left margin of the graph. In the following sections, the discussion explores how the likelihood of soliton occurrence is influenced by the average power of the signal, as this measure has a direct physical interpretation and can be quantified experimentally.



a) b)

Figure 3.4: The dependence of the probability of soliton occurrence for OFDM signals with 16-QAM and a symbol duration of 10 ns from (a) on average signal power, (b) on average signal power per channel.

Figure 3.4(a) displays the relationship between the probability of soliton presence in OFDM signals with 16-QAM modulation and the average signal power. It is observed that as the number of subcarriers increases, so does the required power for soliton formation. This effect can be explained in Figure 3.4 (b), which presents a corresponding graph considering that the average power is distributed across each individual subcarrier. This plot clarifies that the probability of soliton existence is actually governed by the power density within each

subcarrier, independent of the total number of subcarriers. Consequently, an augmentation in the number of subcarriers necessitates a proportional increase in overall signal power to sustain soliton presence. Same trends are detected for different modulation schemes, as depicted in Figure 3.5. The findings suggest that the operational power bandwidth per channel conducive to solitons lies between −42 dBm and −37 dBm. Leveraging this insight allows us to approximate the soliton existence thresholds for varied signal configurations.

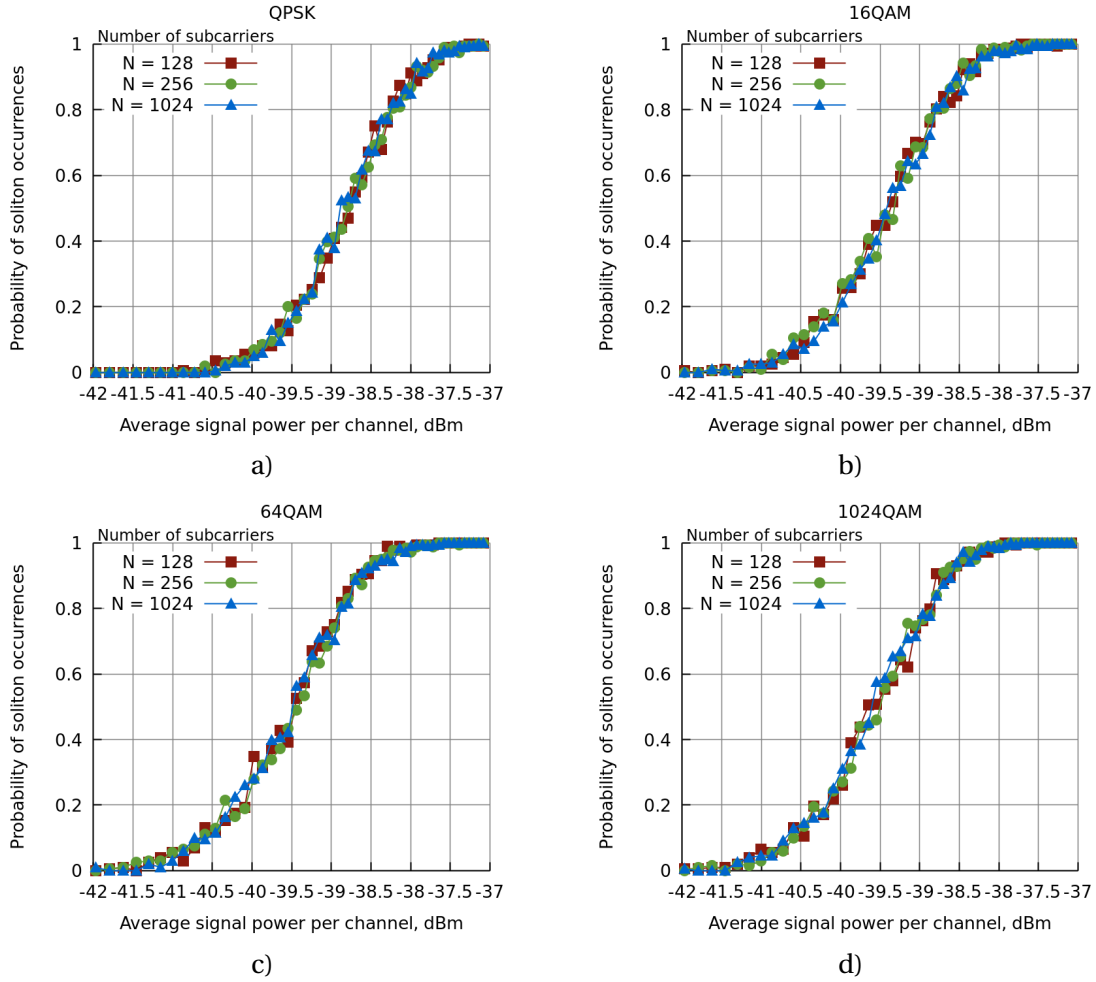


Figure 3.5: The dependence of the probability of soliton occurrence in the OFDM symbol with a duration of 10 ns and (a) QPSK, (b) 16-QAM, (c) 64-QAM, (d) 1024-QAM modulation on the average signal power per channel.

However, the type of modulation also affects the probability distribution. Figure 3.6 demonstrates that with an increase in the order of the constellation diagram, and hence with an increase in the number of bits of information encoded in one symbol, the required average signal power for the existence of solitons decreases. This fact is well seen when comparing QPSK and other types of modulation. Irrespective of the number of subcarriers, there is a tendency that for QPSK modulation, the average signal power is higher than for others. If we

proceed to the dependence on the average power per channel, the situation will not change (Fig. 3.7).

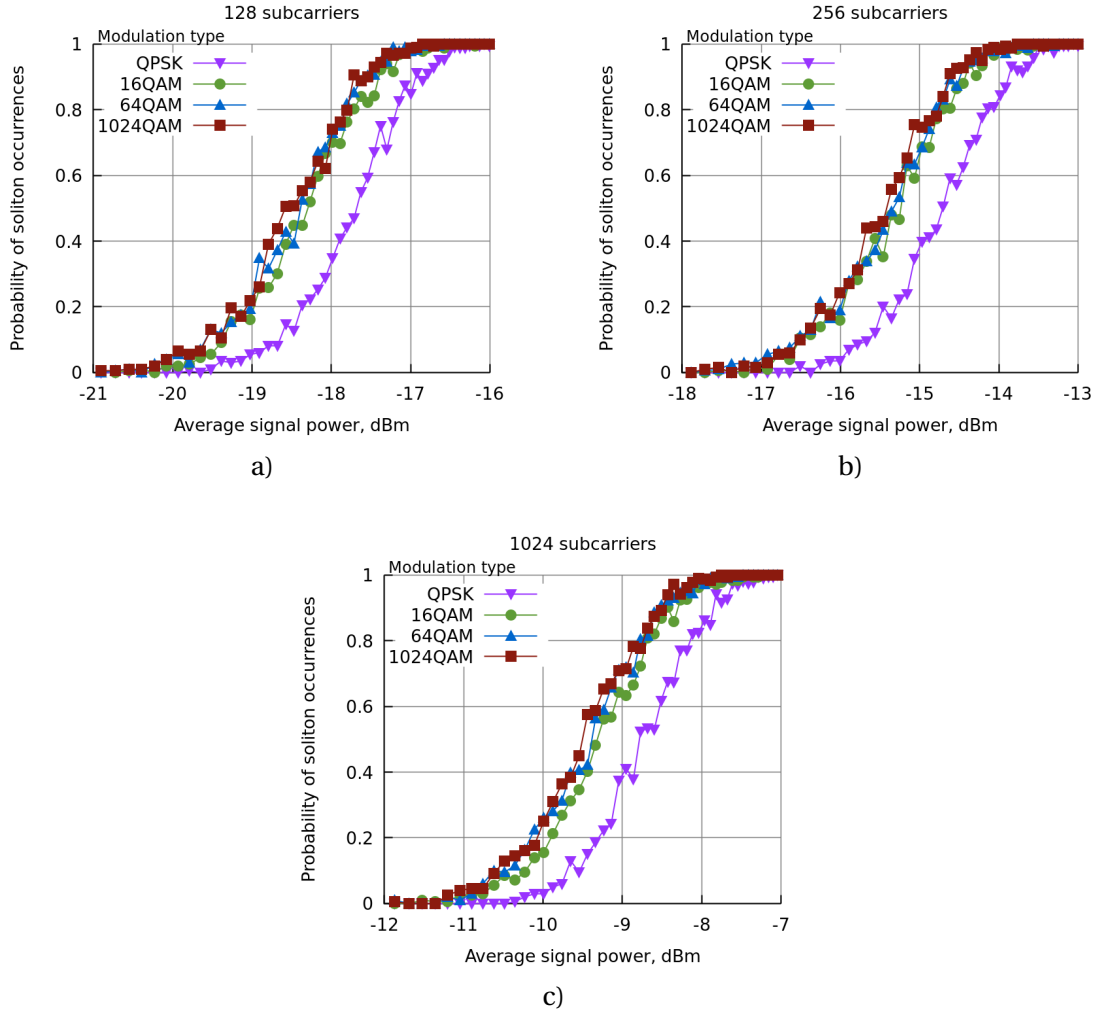


Figure 3.6: The dependence of the probability of soliton occurrence in the OFDM symbol with a duration of 10 ns and (a) 128, (b) 256, (c) 1024 subcarriers on the average signal power.

As expected, for a complex signal the level of L_1 and L_2 norms (and hence P_{ave}), at which solitons exist in the signal, is higher than for a simple rectangular signal. This is consistent with the criteria (3.14), which can also be used for preliminary analysis in other existing systems.

In the general case, solitons and dispersion waves propagate through the fiber in different ways. The most important difference is that the dispersion for solitons is balanced by nonlinear effects. The existence of solitons in a standard OFDM signal has the potential to make some contribution in propagation. Understanding of this fact may affect coding and modulation of signals in optical communication, as it can lead to an improvement in the efficiency of optical communication lines. However, in practice, the appearance of solitons does not have a large effect on the propagation dynamics of the OFDM signal at the considered power levels.

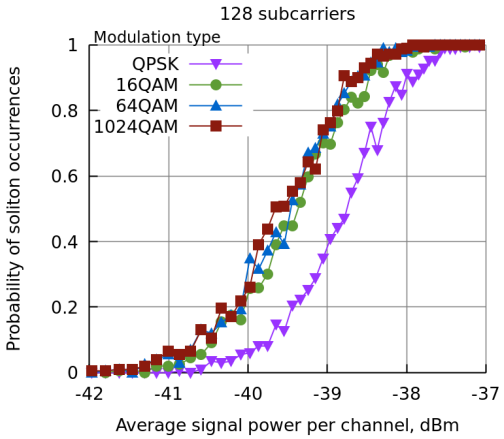


Figure 3.7: The dependence of the probability of soliton occurrence in the OFDM symbol with a duration of 10 ns and 128 subcarriers on the average signal power per channel.

At higher levels, it is already possible to observe the evolution of individual solitons in a common signal, however, these events are rare, and more research is needed to determine the parameters and degree of influence of solitons on the signal.

3.2.3 WDM symbol

WDM technology enables the simultaneous transmission of multiple data channels through a single optical fiber, each at a different frequency. WDM works by sending information on various wavelengths within the same fiber-optic line, with each wavelength known as a "channel," a term we will use throughout.

Data is transmitted using a specific modulation format. The carrier frequency for each channel, f_n, is determined by the formula:

$$f_n = \Delta \cdot n, \quad n = -N/2 \ldots N/2, \tag{3.15}$$

where Δ represents the spacing between channels, which in this case is set to 25 GHz as detailed in Eq. (1.69). The duration of one symbol is 100 ps.

At this point, we're looking at WDM signals that use different modulation formats such as QPSK, and 16-, 64-, and 1024-QAM. We also vary the number of channels from 9 to 51. Just like we did with OFDM signals, we gathered data on 200 WDM symbols for every combination of parameters. Figure 3.8 displays the probability of existence in a signal, depending on the average power per channel for different types of modulation and different number of channels. It's important to note that these measurements are for power in each WDM channel, not the total power across all channels.

When we look at the chart in Fig. 3.8 (a), we see that for QPSK modulation, the necessary power level is quite a bit higher—between 5.8 dBm and 7 dBm—compared to other modulations. For 16-, 64-, and 1024-QAM, this range is broader: from 2 dBm to 6 dBm. As we use more channels, the minimum power needed tends to go down, and this reduction is consistent across all modulation types. Just like with OFDM signals, as the complexity of the modulation increases (from QPSK to 1024-QAM), the power needed to detect solitons gets lower (as shown in Fig. 3.9).

Additionally, for WDM signals, QPSK modulation requires much more power compared to other modulations when it comes to finding solitons. There's also a more pronounced difference between 16-QAM, 64-QAM, and 1024-QAM for WDM signals than for OFDM signals. For instance, if we increase the number of channels to 51, the power required for 16-QAM and 64-QAM starts to diverge significantly, as depicted in Fig. 3.9 (d).

These features demonstrate that, despite the similar signal generation processes, WDM and OFDM systems have a different internal structure, and also depend on the selected parameters in different ways.

The results obtained for WDM and OFDM symbols show that coherent structures are an integral part of these systems, and therefore should be study along with other effects. The results are interesting because the features associated with nonlinearity can be evaluated and used in practice. Now, we moving forward to the analysis of more complex structuers - WDM

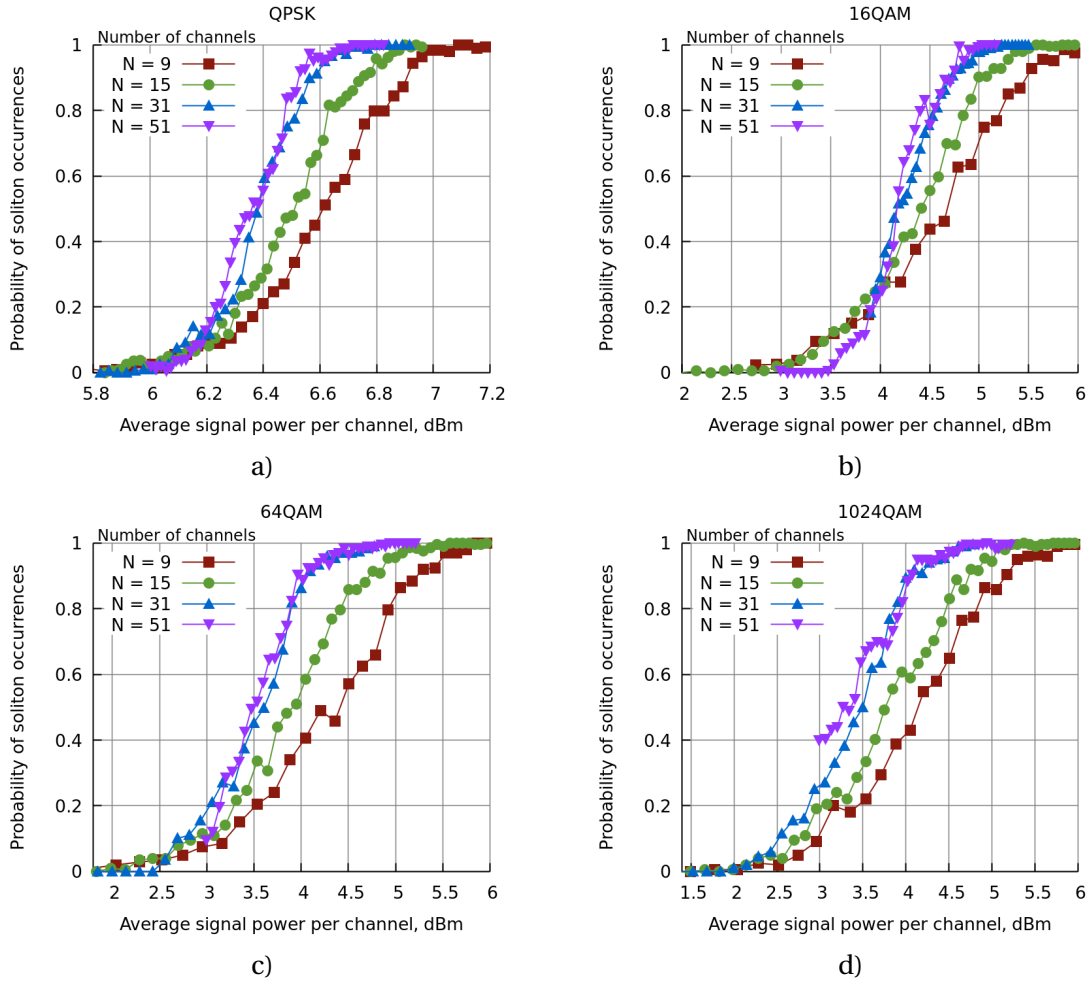


Figure 3.8: The dependence of the probability of solitons occurrence in a WDM signal on the average signal power per channel with a duration of 100 ps and (a) QPSK, (b) 16-QAM, (c) 64-QAM, (d) 1024-QAM modulation.

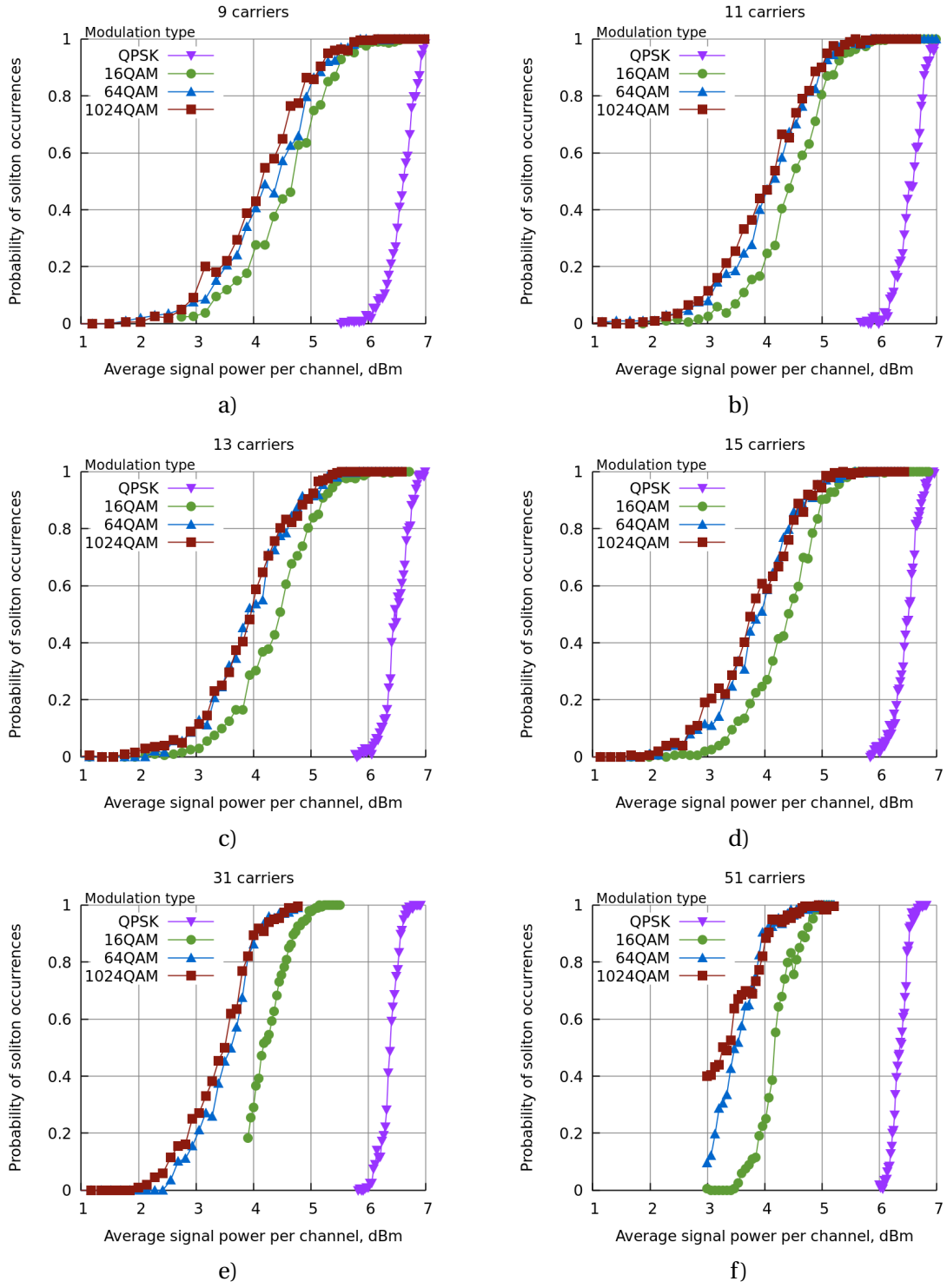


Figure 3.9: The dependence of the probability of soliton occurrence in a WDM signal with (a) 9, (b) 11, (c) 13, (d) 15, (e) 31 and (f) 51 channels.

signals.

3.2.4 WDM signal

In this study, we examined a fiber optic line with a single channel WDM signal using a 16-quadrature amplitude modulation (16-QAM) symbol sequence at a symbol rate of 34 [GBd] in a single polarization. The signal, shaped by a digital Root Raised Cosine (RRC) filter with a 0.1 roll-off factor, covered 12×80 [km] spans of Standard single-mode fiber (SSMF). Signal average power P_{ave} varied between -25 and 6 dBm. Simulations, run in a noise-free environment, utilized an Split-step Fourier method (SSFM) model operating at a wavelength of $\lambda = 1550$ [nm] characterized by an attenuation coefficient $\alpha = 0.0$ [dB/km], a dispersion coefficient $D = 16.8$ ps/[nm$\cdot$km], and a nonlinear coefficient $\gamma = 1.2$ [W$\cdot$km]$^{-1}$. For window processing, the CDC-window mode was employed with parameters: processing interval T_{proc} is 32 symbols and dispersion side intervals $T_d = 300$ symbols.

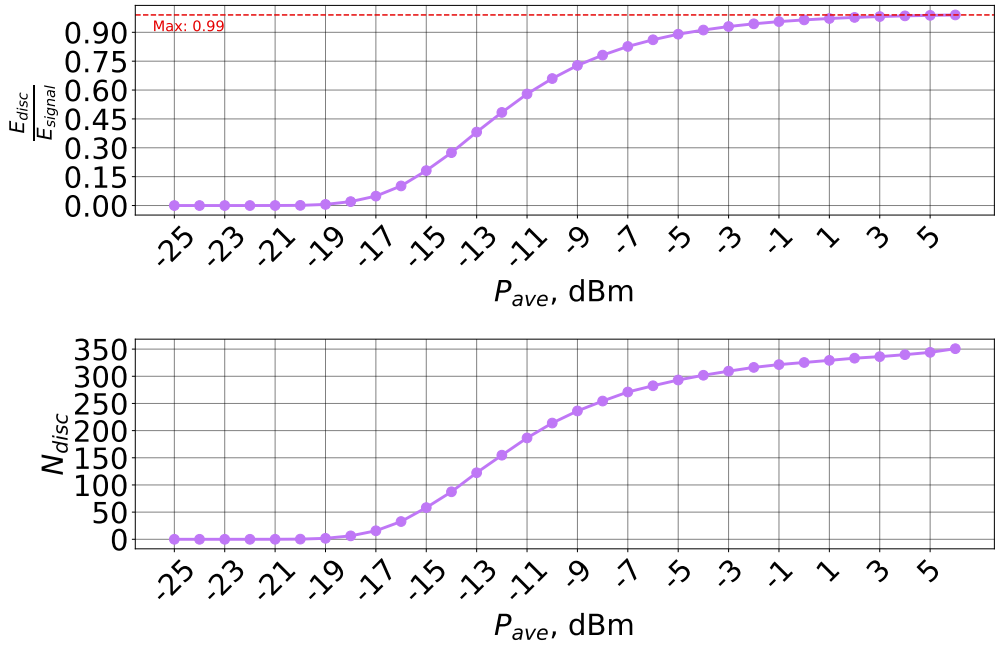


Figure 3.10: **Upper**: Soliton energy proportion in WDM signal. **Lower**: Average soliton count at various power levels.

The upper image in Fig. 3.10 illustrates the proportion of signal energy that comprises the soliton component within a WDM signal. The lower image displays the average quantity of solitons at specific power levels. These representations are based on the statistics of 1024 distinct windowed intervals for the WDM signal. It is important to note that the step shift is equivalent to T_{proc}, resulting in overlapping windows. However, this overlap does not affect the overall trend, as the contributions from identical solitons, which may be counted multiple times, are averaged out. Consequently, we observe consistent behavior for non-overlapping intervals as well.

At lower average signal power levels, the soliton component is absent, with both the number and energy of solitons being zero, as expected. Interestingly, an increase in power leads to a rise in the energy allocated to the soliton component. Commencing at approximately -20 dBm, the count begins to ascend, followed by a corresponding increase in the soliton energy fraction. Near the 1-3 dBm range, almost 99% of the energy attributed to solitons within the signal, and this proportion remains stable even as the average power level continues to increase (up to the maximum observed level of 6 dBm).

The number of solitons does not saturate at the same level (around 1 dBm) and continues to grow with an increase in average power. Notably, even at 5 dBm and 6 dBm, a distinction in the soliton count is evident. This phenomenon suggests that, beyond a certain saturation point of power, where the majority of the energy is soliton-associated, the number of solitons can still proliferate. This implies that the emergence of higher-energy solitons becomes unlikely, while the system favors an increase in lower-energy solitons that collectively assimilate the available energy.

From a physical perspective, systems gravitate towards a state of minimal energy, which is more readily achieved by integrating multiple low-energy solitons rather than a combination of a few high-energy and several very low-energy solitons. Statistically, it is also more likely to obtain uniform results across different signal realizations with typical energy levels than with extreme soliton power values. These observations foster a discussion regarding the implications of the data, setting the stage for following validation of our preliminary interpretations.

Figure 3.11 displays the distribution of discrete points in the NF spectrum, which corresponds to the number of solitons, across different average signal power levels. Starting with the upper left plot at the lowest power of -19 dBm, the power increases from one plot to the next, culminating in the lower left figure at the highest power of 5 dBm. This range of power levels, as seen in Fig. 3.10, spans from scenarios with nearly no solitons to those where the power is predominantly in soliton form.

Additionally, Fig. 3.12 complements the previous figure by presenting the distribution of the maximum imaginary part of the NF spectrum, which is directly related to soliton power (refer to Eq. 5). Our prior analysis on soliton content within single symbols indicates that certain power levels, particularly at the boundaries like -20 dBm (Fig. 3.12) and -19 dBm (first panel of Fig. 3.11), are not well-characterized by a Gaussian distribution. In these instances, a Poisson distribution offers a more accurate representation.

Excluding the borderline power levels up to -16 to -15 dBm, where a Poisson distribution may still be more appropriate, we observe that the standard deviation of the Gaussian distribution tends to decrease with increasing power. For power levels above this range, our preliminary data suggests a Gaussian standard deviation in the vicinity of 7.0 to 7.4. This value narrows to around 6.0 for power levels near -2 to -1 dBm, and even further to 5.0 for power levels between 2 and 5 dBm.

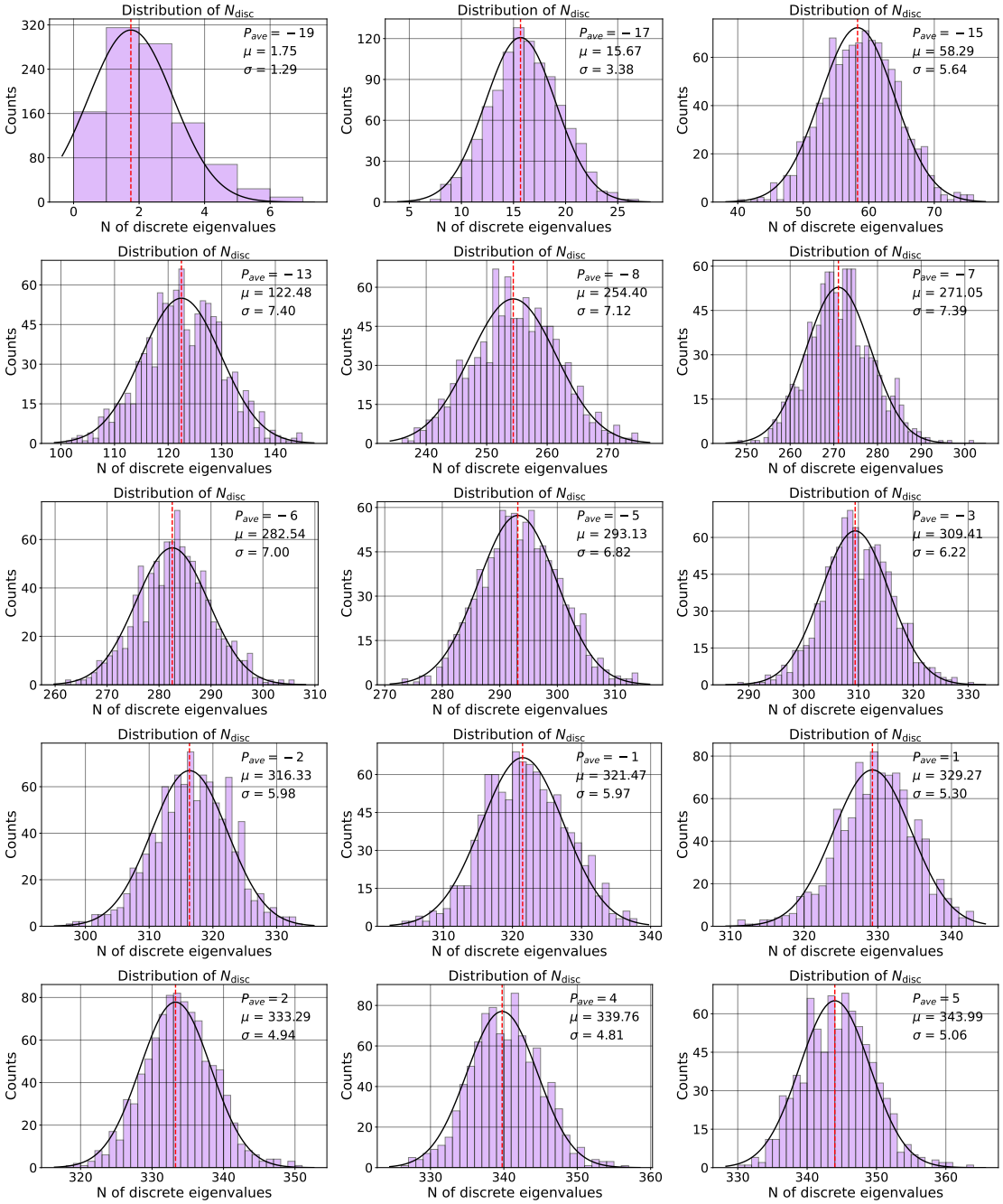


Figure 3.11: Comparison of soliton distibution for WDM signal with differnt average signal power.

This trend hints at a region of stability where the occurrence of solitons is statistically more predictable and exhibits less variation. This stability may indicate the formation of physical structures akin to a “soliton gas”, where solitons exhibit particle-like behavior, and thermodynamic principles could potentially apply.

At 6 dBm, as shown in the lower pane of Fig. 3.12, the standard deviation increases to 5.75. This could suggest either a statistical anomaly or the onset of a more complex system structure, necessitating further investigation. The concept of a soliton gas is an intriguing one that aligns with this observation, but it requires additional research to fully understand the relationship between soliton statistics and thermodynamic properties.

Figure 3.13 illustrates the distribution of discrete eigenvalues in the NF spectrum across the complex plane at varying signal power levels, beginning at −15 dBm in the upper left and concluding at 6 dBm in the lower right panels. The intensity of the blue color represents the density of points within the spectral space. Accompanying the main graph, histograms along the top and right edges display the distributions of the real and imaginary parts, respectively.

At lower power levels, a substantial presence of solitons is evident, with an average count of approximately 60 for −15 dBm and about 214 for −10 dBm. These figures indicate that the real part of the discrete spectrum is evenly distributed, aside from the spectral edges where the signal’s bandwidth limit has an effect. For the imaginary part, the distribution shows a general decline with an increase in the imaginary value, suggesting a higher prevalence of low- and mid-energy solitons over those with high energy. This pattern is consistent even at −5 dBm, as shown in the second row, first panel of Fig. 3.13.

As the average signal power increases, a notable change occurs: low-energy solitons begin to cluster at the boundaries of the spectral intervals. This clustering is observable at 0 dBm and becomes more noticeable at 3 and 6 dBm, where the right histograms reveal a significant peak near the real axis. Beyond this peak, there is a gradient decrease in the imaginary part of the discrete eigenvalues. In Fig. 3.10, we observed that for average signal powers exceeding 1 dBm, the signal reaches a saturation point, with nearly all (99%) of its power comprised of solitons. Yet, as the power continues to increase, so does the number of solitons, suggesting an accumulation of lower-energy solitons and a concentration at the spectral interval edges.

The underlying reasons for this tendency are not immediately clear, but from a physical standpoint, it could be assumed that this pattern represents an energy-efficient arrangement of discrete eigenvalues within the spectral space.

3.2.5 Conclusion

This study demonstrates that solitons are an integral component of optical communication signals. From the analysis of single symbols, which reveals the close relationship between solitons and the signal’s internal structure—such as independent subcarriers in OFDM or different channels in WDM—to the examination of complex WDM signals that may contain

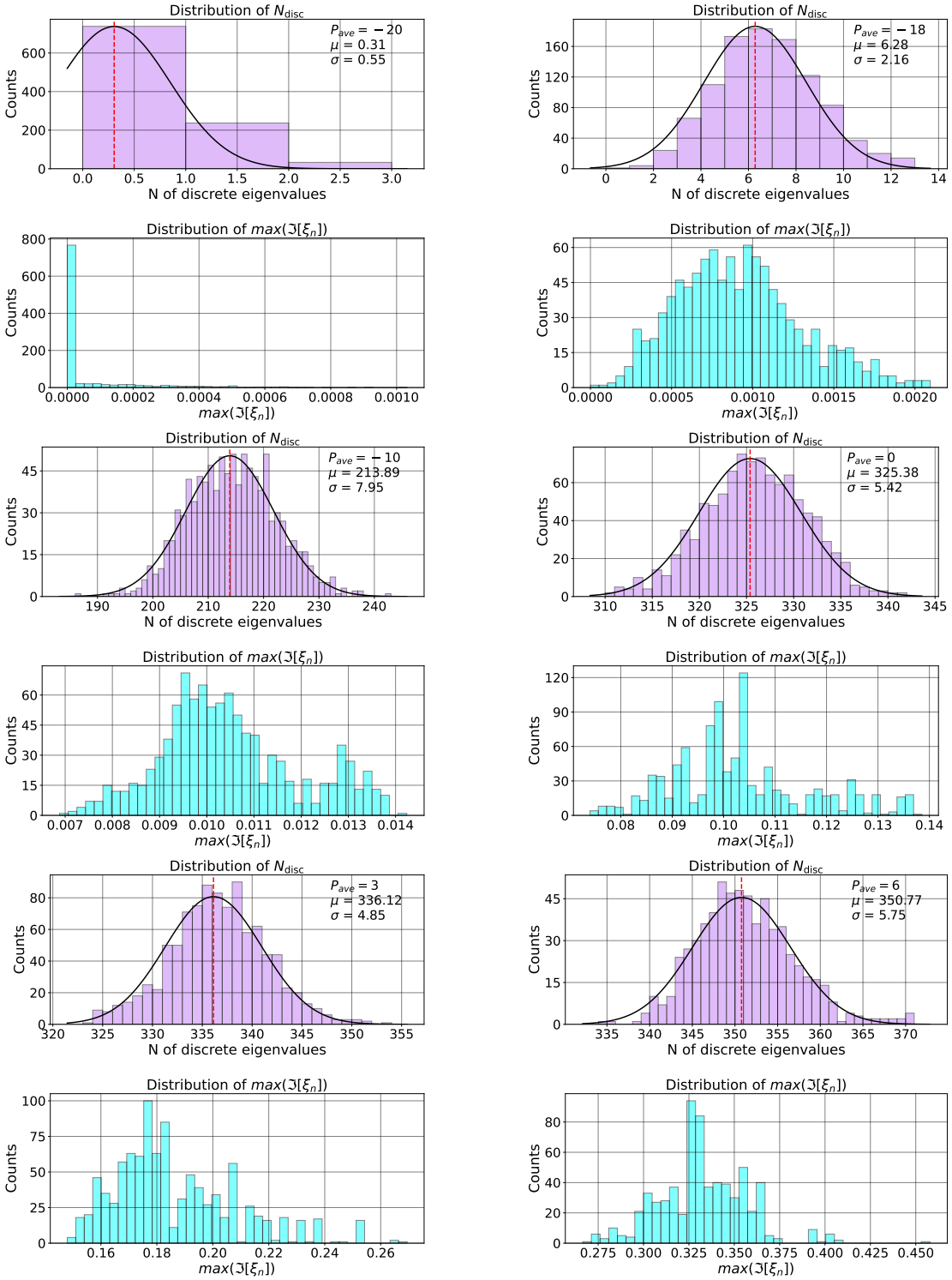


Figure 3.12: Comparison of soliton distibution and distribution of maximum discrete eigenvalue

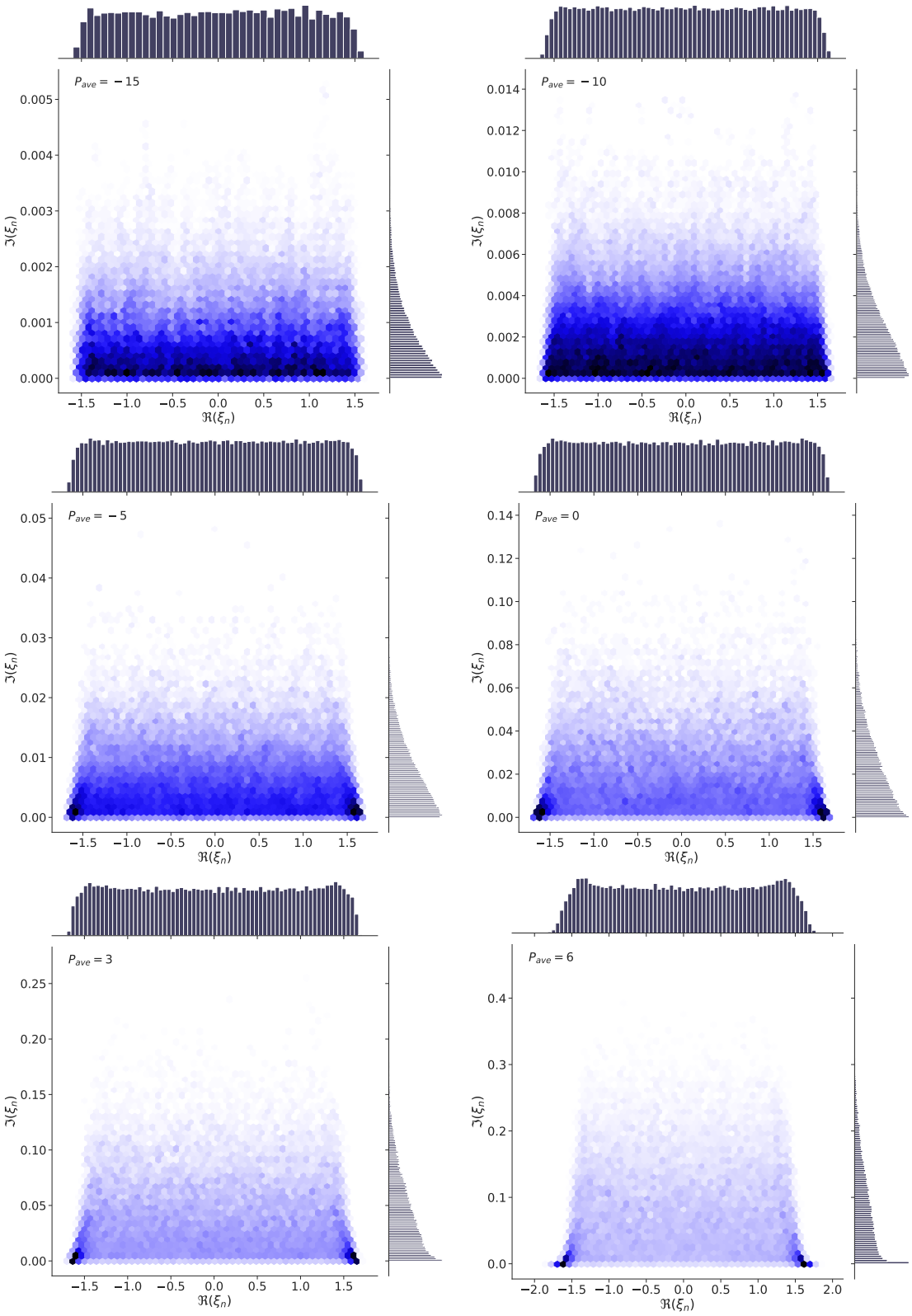


Figure 3.13: Distribution of discrete eigenvalues in the NF spectrum across the complex plane at varying signal power levels, beginning at -15 dBm in the upper left and concluding at 6 dBm in the lower right panels.

dozens or even hundreds of solitons, it is evident that nonlinear structures can have a significant influence on signal formation. This realization provides us with valuable insights that could inform further research and enhancements in modulation techniques or signal formats.

We observed that the statistical behavior of WDM signals offers clues about the internal thermodynamics of the 'soliton gas' present within these signals. Leveraging such statistics could lead to a more profound understanding of the nonlinear interactions occurring within the signal, ultimately contributing to improvements in transmission quality. If we can decipher the effects, we may be able to devise methods to mitigate or even exploit them to our advantage, similar to the use of solitons for data transmission. While the subsequent stages of this research are outside the context of this thesis, it is clear that the nonlinear Fourier properties of signals open up numerous possibilities for future exploration.

4 Neural Network-based Nonlinear Fourier Transform Algorithms

4.1 Introduction

Since linear, nonlinear, and noise effects manifest themselves simultaneously when we transmit data over fibre-optic communication lines, such systems are well fit to be dealt with using the latest advances in machine learning methods. Using these methods, it is possible to solve the problem of multidimensional optimisation (for example, in terms of data transmission quality and data throughput maximisation) without having to iterate through all possible parameter values. Of particular relevance is the problem of identifying some internal features and patterns of the transmitted data, where neural networks can be used to simulate various effects that affect the signal when it propagates through a noisy nonlinear medium. In other words, neural networks can be used to simulate nonlinear transformations without the need for direct calculation of these transformations. The advantage lies in the speed and versatility of the transformation, as well as the flexibility and adaptability of operations based on a neural network: the network does not know what data it processes; it looks for the necessary features in the data that affect the final result, and then extracts them. This process is called feature extraction. Thus, if we want to calculate a certain value of a function, instead of (possibly) complex calculations, we can use a pre-trained network that, with a pre-known number of operations, will give the desired result. The difficulty is that the neural networks need to be trained up-front on the known data. Another advantage of signal processing based on neural networks is that networks can reduce the noise component present in the analysed data [38]. In practice, we almost always encounter a situation where there is some noise in the data, for example, due to the finite accuracy of measurements, and its presence may be critical for accurate data processing methods. A neural network can effectively filter out unnecessary information within itself, leaving only the basic features needed for a specific task. Note that one of the disadvantages of using neural networks is the final accuracy of the result attainable with the use of a trained network. However, in practice, the accuracy of a neural network is sufficient for most tasks and sometimes even exceeds the accuracy of existing numerical methods if the necessary set of training data is available.

The first direction of utilising NNs for NFDM systems consists in applying the additional NN-based processing unit at the receiver to compensate the emerging line impairments and deviations from the ideal model [159–163]. But, despite ensuing transmission quality improvement, this type of NN usage brings about the additional complexity of the receiver. In the other approach, the NFT operation at the receiver is entirely replaced by the NN element. It has been shown that this approach, indeed, results in a considerable improvement of the NFT-based transmission system functioning [76, 164, 165]. But, despite the benefits rendered by such a NN utilisation, the NNs emulating the NFT operation have so far been mostly used in the NFDM systems operating with solitons only, and the NN structure used there was relatively simple. In the only work related to the continuous NF spectrum recovery [166], a standard "imageInputLayer" NN (developed originally for hand-written digits recognition) from MATLAB 2019a deep learning toolbox was adapted to process the signals of a special form. Such an approach, evidently, has limited applicability and flexibility and is not optimal neither in terms of the result's quality nor in the complexity of signal processing. In our current work, we demonstrate how this direction can be significantly extended and optimised by using Neural network (NN) for modeling the continuous part of the Nonlinear Fourier (NF) spectrum. We show how NNs could be used to detect the discrete spectrum and demonstrate the use of optimization tools to choose the best NN design. Our research aims to establish a foundation for creating highly effective NFDM systems that work well with any channel. Additionally, we don't just focus on finding the NF spectrum $r(\xi)$, but also on the b-coefficient, which can be used in the most effective method of NFDM known as b-modulation.

The foundation of this chapter's content is the outcomes reported in [167–169]. This chapter is structured as follows: At the close of the introduction, we detail the data format employed throughout the chapter. Subsequent to this, we start with an initial exploration of utilizing Neural network (NN) to execute both forward and inverse Nonlinear Fourier Transform (NFT) operations on signals without solitons, which have been omitted from the dataset. The following section is dedicated to signals that do contain solitons, focusing on the application of NN to estimate soliton quantities. The core section presents a case study in which a neural network is applied to predict the continuous NF spectrum of soliton-free signals in case of noise interference. The chapter concludes with a discussion on prospective routes for future research.

4.1.1 Data format

In the following examination of optical signals, we utilize the WDM format (see Eq. (1.69)). In this section, we express the WDM signal in dimensionless units, a practical necessity for applying the NFT in our analysis. The mathematical representation of a return-to-zero (RZ) WDM symbol is given by:

$$s(t) = \frac{1}{Q} \sum_{k=1}^{M} C_k e^{i\omega_k t} f(t), \quad 0 \le t < T, \tag{4.1}$$

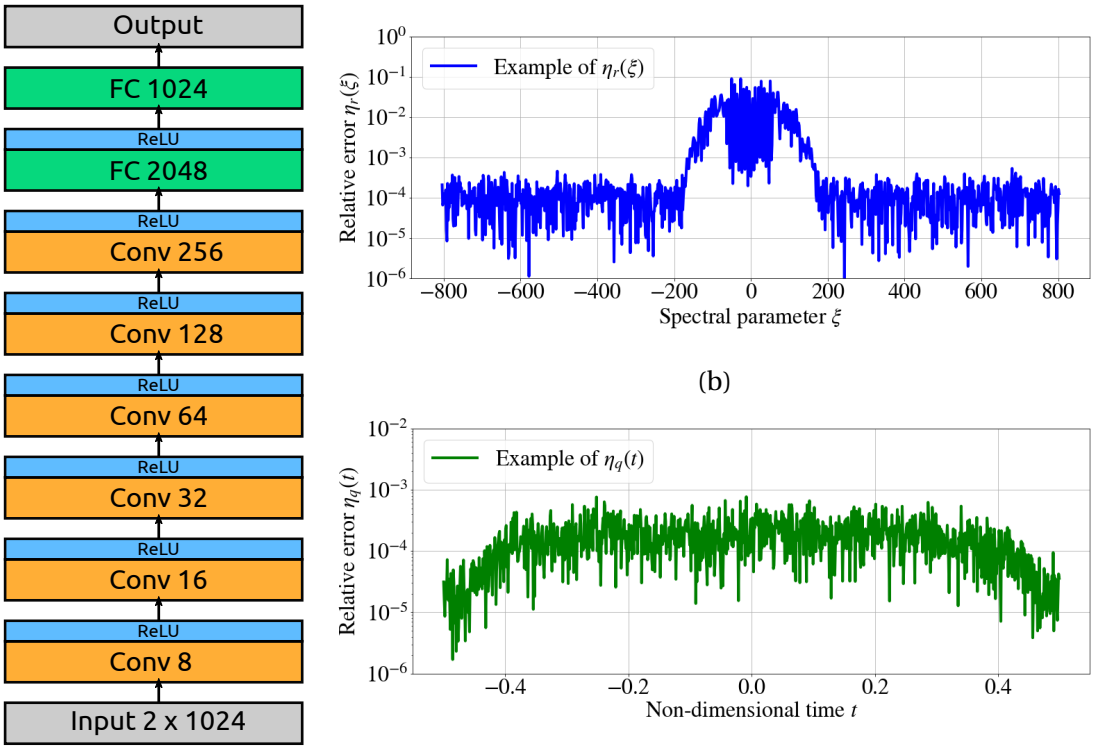


Figure 4.1: **(a)** Proposed architecture of the NN that performs the NFT operations. **(b)** Value of the relative error $\eta_r(\xi)$ between the precomputed and predicted continuous spectrum. **(c)** Value of the relative error $\eta_q(t)$ between the original and predicted signal.

where M is a number of WDM channels, ω_k is a carrier frequency of the k-th channel, C_k corresponds to the digital data in k-th channel, and T defines the symbol interval. Q in (4.1) is the normalisation factor that we use to set the required energy for each signal (the total signal energy is calculated according to Eq. (4.5)). In Eq. (4.1), each coefficient C_k represents a complex number selected from a constellation diagram of a given size, meaning it is randomly chosen from a predetermined set of possible values with uniform probability. For our NF decomposition analysis each time we use a single signal of the form given in Eq. (4.1). $f(t)$ is the waveform of the carrier pulse, defined as (and represented on Fig. 1.4):

$$f(t) = \begin{cases} \frac{1}{2}\left[1-\cos\left(\frac{4\pi t}{T}\right)\right], & 0 \le t \le \frac{T}{4} \text{ or } \frac{3T}{4} \le t \le T, \\ 1, & \frac{T}{4} < t < \frac{3T}{4}. \end{cases} \tag{4.2}$$

4.2 Neural Network for Forward and Inverse Nonlinear Fourier Transform

In this section, we use the NN to predict the continuous NFT spectrum of complex optical signals and transform the spectrum back (the inverse NFT) into a signal. Convolution Neural

network (CNN) are preferred over Feed-forward Neural Network (FNN) due to their efficient processing of spatial hierarchies in data. This efficiency makes CNNs particularly suitable for tasks where context and locality play critical roles. Unlike FNNs, which rely on simple matrix multiplications that cannot capture complex spatial relationships, CNNs utilize layers equipped with convolutional filters. These filters are adept at capturing spatial features at various levels of abstraction, thereby allowing CNNs to identify patterns more effectively. The significance of this capability becomes especially apparent in the context of the Nonlinear Fourier Transform. NFT, not being a linear transformation, means the local context within a time sequence greatly influences the outcome of the transformation. This nuanced influence of local context is something CNNs can handle well, thanks to their spatial processing capability. For the random number generator, both here and in subsequent sections, I employ the Mersenne Twister as the core algorithm. This choice is due to its well-documented high-quality randomness and its exceptionally long period ($2^{19937}-1$), as detailed by Matsumoto and Nishimura (1998) [170].

To assess the quality of the NN prediction, we use the following formula defining the relative error for the continuous NFT spectrum:

$$\eta_r(\xi) = \frac{|r_{\text{predicted}}(\xi) - r_{\text{actual}}(\xi)|}{\langle |r_{\text{actual}}(\xi)| \rangle_\xi}, \tag{4.3}$$

where $\langle\cdot\rangle_\xi$ denotes the mean over the spectral interval, the "predicted" and "actual" indices refer to the NN-predicted and precomputed values of the reflection coefficient $r(\xi)$, respectively. A similar formula for calculating the relative error for signal prediction:

$$\eta_q(t) = \frac{|q_{\text{predicted}}(t) - q_{\text{actual}}(t)|}{\langle |q_{\text{actual}}(t)| \rangle_t}, \tag{4.4}$$

where $q(t)$ is a signal and $\langle\cdot\rangle_t$ here is the mean over the time interval. The above relative error $\eta_r(\xi)$ and $\eta_q(t)$ is determined at the point ξ and t, respectively. We use $\langle\eta_r(\xi)\rangle_\xi$ and $\langle\eta_q(t)\rangle_t$ to estimate the overall mean of the error. We stress that the metric was chosen in such a way as to take into account even the regions where the value of the spectrum or signal is much less than one.

Fig. 4.1a shows the architecture of the NN that performs the NFT operations (direct and inverse transform), with the parameters given inside the figure. The NN consists of sequential convolution layers and fully connected output layers. At the input, the network receives a complex signal consisting of 1024 points. This NN predicts only one component of the continuous NF spectrum, such that two identical NNs have to be used to predict the real and imaginary $r(\xi)$ parts. Similarly, converting the spectrum back to a signal requires two separate NNs for the real and imaginary parts of the signal $q(t)$. Each of the four NNs with the same architectures we trained independently.

A total of 1024 points were selected to strike a balance between the computational complexity of the Nonlinear Fourier Transform and the demonstration of the NN capabilities. This

selection represents a synthetic case designed to start from simple signals, albeit with high resolution, to showcase the potential of NNs to perform NFT operations. The choice of this specific number was motivated by its sufficiency to illustrate NNs' ability to execute both forward and inverse NFT, while preventing the NN model from becoming overly complex. Furthermore, the generation of the dataset required preliminary computations (i.e., performing NFT), and larger signal sizes would have necessitated more extensive calculation time.

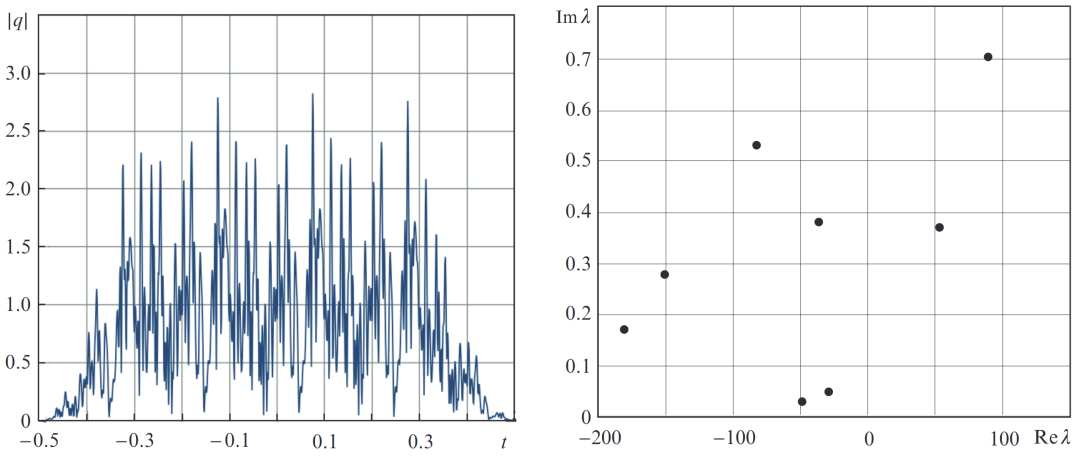
There are 94035 signals in the dataset, 9403 are used for validation and are not involved in the training process. To generate the signals, we used random data sequences encoded in the Quadrature Phase-Shift Keying (QPSK) format. The energy of all signals is the same and is chosen at such a level that nonlinear effects are strong. At the selected energy, some of the signals contained a discrete spectrum, but such signals did not get into the training dataset. The continuous spectrum for each signal had been precomputed using the conventional direct NFT methods. To train the NN, we used the mean squared error (MSE) as the loss function. In training the NN, we employed the Adam (Adaptive Moment Estimation) optimisation algorithm with a learning rate of 1e-4. On average, with the amount of data used, our learning process took 50 000 epochs. To detect and mitigate overfitting during the training process, we monitor the loss function value on a validation dataset, which is distinct from the training dataset to ensure it does not influence the training directly. Additionally, we employ an early stopping callback mechanism. This mechanism tracks the loss values for both the training and validation datasets. Should the validation loss begin to increase – a phenomenon observed over a range of epochs, in our case, from 100 to 500 across different studies – the training process is halted. This strategy allows us to save the model at its peak performance, effectively before the onset of overfitting.

Fig. 4.1b depicts $\eta_r(\xi)$ – the difference between the predicted and actual (precomputed using conventional NFT method) continuous nonlinear spectrum for the example signal, while Fig. 4.1c shows the example of $\eta_q(t)$ for the inverse NFT. We see from the graphs that NN performs both forward and inverse transformations with high accuracy. Some increase in the error at the centre for a continuous spectrum is associated with its localization in the middle. While at the edges of the spectral interval, $r(\xi)$ values tend to 0. There is no such feature for the signal since it is evenly located over the entire time interval. The mean value over the entire validation set of the mean relative prediction error of the continuous spectrum $\langle \eta_r(\xi) \rangle_\xi$ for NN forward NFT is $2.68 \cdot 10^{-3}$. For the inverse transformation, the mean over the entire validation set of the mean relative signal prediction error $\langle \eta_q(t) \rangle_t = 1.62 \cdot 10^{-4}$. The results obtained show that the presented architecture can perform forward and backward NFT with high accuracy.

4.3 Discrete Spectrum Number Estimation

In this part, we used a neural network to predict the number of discrete eigenvalues in a nonlinear spectrum of telecommunication signals. The discrete spectrum reflects the internal structure of the signal, and knowledge of this structure allows one to find out the signal

properties and the features of its propagation over a nonlinear optical fibre.



(a) (b)

Figure 4.2: (a) Example of the dependence of the amplitude of the WDM signal under study on time (one of the possible implementations is given), and (b) example of the location of the discrete spectrum components in the complex half-plane of the spectral parameter ξ for one of the signals under study.

The signal was generated from a random dataset encoded in one of the modulation formats for C_k: QPSK, 16-QAM, 64-QAM, and 1024-QAM, corresponding to 4, 16, 64, and 1024 possible values of C_k, selected from the constellation on a complex plane. The number M of spectral channels for each signal from the set was one of the following: {9, 11, 13, 15, 17, 31, 51}. This set of values was taken to cover the number of channels used in existing WDM systems. An example of the WDM signal amplitude is shown in Fig. 4.2a, while an example of a discrete spectrum for such a signal is shown in Fig. 4.2b. The neural network architecture was based on a simplified version of the VGG-16 network, which is used in image recognition problems (Fig. 4.3a). Such architectures, where convolutional layers with the same number of input channels are sequentially arranged, demonstrate high efficiency by reducing the number of trained parameters while maintaining the overall prediction accuracy. To further improve the accuracy, it is necessary to increase the number of convolutional layers. This increases the learning time of the model; however, it allows us to select more features in the input data, and therefore improve the operation accuracy. The neural network input receives a complex signal consisting of 1024 points. This signal is converted to a vector with 2048 elements, in which the real and imaginary parts of each point of the original complex signal are sequentially arrayed. The signal is then processed by several convolutional layers with activation functions and then passed through the fully connected layers. The network output shows the number of solitons in the signal. The number of trained parameters in the network was 3 834 145.

In total, the training set consists of 174 847 generated signals, which contain from 0 to 20 solitons. The exact number of solitons in each signal was preliminarily calculated using a modification of the method of contour integrals, where the grid step in the spectral space

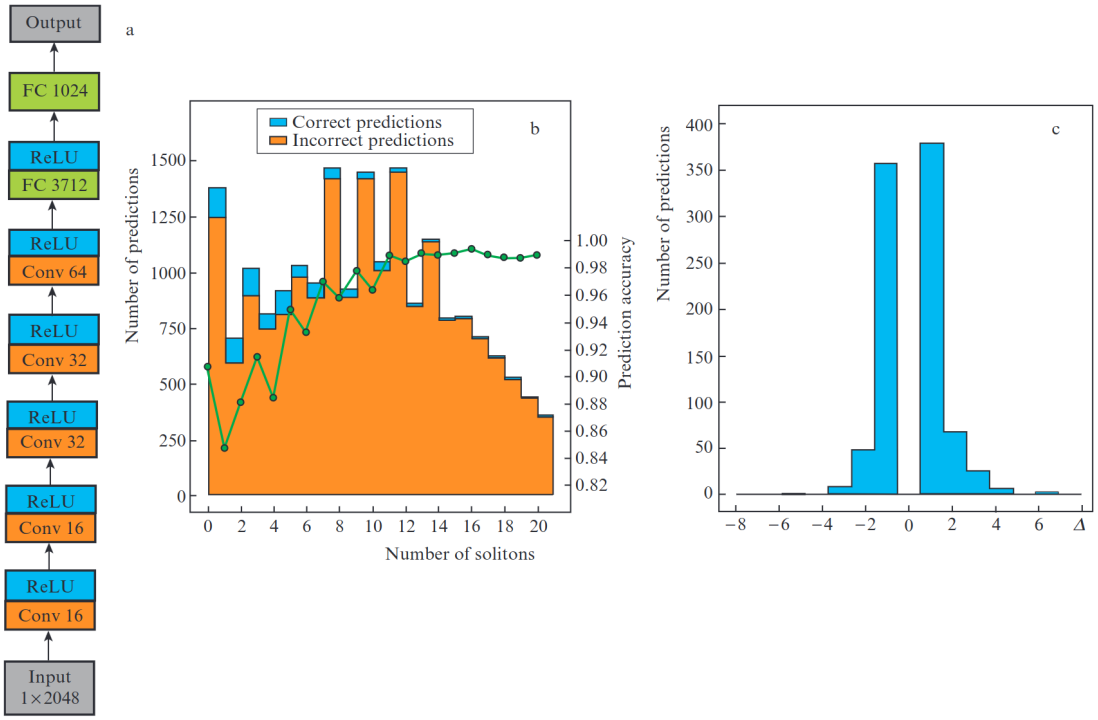


Figure 4.3: **(a)** Neural network architecture for predicting the number of discrete eigenvalues (solitons) in the Nonlinear Fourier spectrum for a WDM signal. **(b)** the distribution of correct and incorrect predictions of the neural network as a function of the number of solitons in signals from the validation sample (the green curve shows the network prediction accuracy for each set of signals with the same number of solitons), and **(c)** the distribution of the deviation Δ of the predicted number of solitons in the signals from the validation sample versus their actual number.

was adaptively calculated depending on the number of solitons in the signal. To speed up the learning process on the training data set, for each signal we calculate only the number of discrete eigenvalues in the nonlinear spectrum, rather than the numerical value of each of the discrete spectral parameters corresponding to the soliton mode. A set of data with complete information on the nonlinear spectrum for each signal will be considered in subsequent works. The accuracy of network predictions was determined using a validation set of 19 427 signals (10 % of the total training set). The network was trained for 300 epochs, and the final prediction accuracy on the validation set was 95.39%. In the training process, we use for optimisation the Adam algorithm (adaptive moment estimation). The learning rate was varied during the training process from 10^{-3} at the beginning to 10^{-5} at the end. Its further reduction did not lead to an increase in the prediction accuracy. Figure 4.3b shows the distribution of correct and incorrect neural network predictions as a function of the number of solitons in the signal for the validation sample. The green curve shows the neural network accuracy as a function of the number of solitons. The network worked best for signals where the number of solitons was greater than 10. For such cases, the accuracy exceeded 98%. Signals with a single soliton component were processed the worst, with an accuracy of 84%. In this case, the maximum 'error' of the network operation (the difference between the real number of solitons in the signal and the soliton number predicted by our neural network) was 8 (Fig. 4.3c). Negative error values correspond to the case when the network predicts a smaller number of solitons in the signal than the true number, and positive values occur when the neural network predicts a larger number of solitons than is present in the signal. Figure 4.3b shows the distribution of the deviation of the predicted number of solitons in the signals generated by the validation sample from their actual number: most of the incorrect results are in the range [–2; 2]. Thus, the neural network's prediction often deviates by a small value. This means that it captures general features indicating the number of solitons but cannot fully identify particular features. Obviously, with an increase in the number of convolutional layers, the neural network will be able to determine the internal features of signals more accurately, which means that its accuracy can be improved. The results show that neural networks have a great potential for implementing various stages of NFT with their help.

4.4 Neural Network Approaches for Continuous Spectrum Analysis in NFT

4.4.1 Introduction

In this section, we describe the main results obtained in the process of finding a suitable NN architecture for computing the continuous NF spectrum of a given signal. The results of this section have been published in [168]. First, we describe which type of signals we used in training and testing. Next, we discuss the Bayesian optimisation application for our finding the best-performing NN architecture and the respective training procedure. Then, we analyse the output accuracy for the proposed NN architecture and compare it with that produced

by a deterministic NFT numerical algorithm. In this study, for the data generation and "conventional" computation we use the Fast Nonlinear Fourier Transform (FNFT) library[171]. At the end of the section, we show that the proposed NN architecture can predict not only the scattering coefficient $r(\xi)$, but also the NF coefficient $b(\xi)$, Eq. (1.21).

4.4.2 Training data generation

In this work, without loss of generality, we analyse the NF decomposition of the signals having the form of Wavelength Division Multiplexing (WDM) format with random modulation and return-to-zero carrier functions, considered in [141, 172].

To train the NN, we precomputed 94035 such signals, with C_k for each carrier randomly drawn from Quadrature Phase-Shift Keying (QPSK) constellations, i.e. the constellations with 4 possible points; the number of optical channels (carriers) in (4.1) is 15. Then we sampled our signal at equidistant points in time, t_m, over the segment of length T, $q(t_m) = q_m$: the number of sample points in each signal representation was $2^{10} = 1024$. The normalised symbol interval T was set to unity so that the time step size used was $\Delta t = 2^{-10}$ (for the explicit normalisations referring to single-mode fibre transmission see, e.g., Ref. [4]). For generated discretised profile, the reflection coefficient $r(\xi)$ was identified for 1024 sample points in ξ variable, calculated using the fast numerical NFT method [171]. The parameter ξ for our computations ranged from $-\pi/(4\Delta t) \approx -804$ to $\pi/(4\Delta t) \approx 804$: this region corresponds to the conventional Fourier spectrum computational bandwidth for the given sampling rate Δt, up to the scaling factor 2 referring to the linear limit correspondence [59]. Each signal in the dataset was eventually normalised so that its energy $E_{\text{signal}} = 39.0$. Some of the signals in the initial dataset for this energy contained solitons, but such signals were singled out and removed from the training and validation datasets. The remaining 94,035 signals did not contain solitons, which means that the discrete nonlinear spectrum for each signal is absent, such that these are used in our analysis. We note that although there are no solitons in the signals, we are still operating in the regime where the signal nonlinearity is not negligible. The more straightforward way of generating the datasets with desired properties would be to use the inverse NFT routines, but these are much more time-consuming, such that we decided to employ the data-generation approach described below: it also allows us to explicitly control the accuracy of the generation process.

Together with the set of deterministic signals, we generated the signal sets with the addition of uncorrelated Gaussian noise, adding the random value to each sample point. In realistic applications, the source of this noise can be the instrumental imperfections of the transceiver or the effects relevant to inline amplifiecation [49]. The Signal-to-noise ratio (SNR) is a traditionally used characteristic for quantifying the level of a noisy corruption:

$$\text{SNR} = \frac{E_{\text{signal}}}{E_{\text{noise}}}, \qquad E_{\text{signal}} = \sum_{m=0}^{N-1} |q_m|^2 \Delta t, \tag{4.5}$$

where E_{signal} and E_{noise} are the signal and noise energies, respectively; q_m is the m-th signal sample, with N being total number of sampling points, Δt is the time sample size. For further training, in addition to the set without noise, which had 84632 signals, we used 8 sets of 423160 signals (5 different noise realisations). Each set corresponds to one of the following SNR values: {0, 5, 10, 13, 17, 20, 25, 30} dB. 9 sets of 9403 signals with the corresponding noise levels were left to validate the network performance. Validation data sets were not used in the training process. We note that the NFT in optical communications is tailored for use in long-haul systems, meaning the high levels of noise (low SNR) is most interesting from the application perspectives. However, we also include the results for high SNR levels to analyse the NN functioning peculiarities in detail.

Turning to the question of soliton appearance from a localised profile, the rigorous criterion for our pulse having no embedded solitons can be formulated for single-lobe profiles as [173]:

$$\int_{-\infty}^{\infty} |q(t)|\, dt < \pi/2, \tag{4.6}$$

and the deterministic profiles used in our work have a much higher normalised energy. For more involved multi-lobe profiles, the soliton-creation threshold is typically higher, but we still had some profiles that contained solitary components, so we had to eliminate them. When we add noise to our signal that initially contains no solitons, a random modulation typically diminishes the probability of solitons appearance [174, 175]. However, we checked out that all randomly perturbed signals used in our study did not contain a solitonic component as well.

To demonstrate the difference between the continuous NFT spectrum and the linear FT spectrum, we calculated (taking into account the necessary transformations and frequency scaling) both spectra for an example signal of the type used in our analysis. As the measure showing the distinction between the conventional Fourier and NF spectra, we use the norm of the difference: $|r(\xi) - r_{FT}(\xi)|$, where $r_{FT}(\xi)$ is given by the first (linear) term in the expansion of $r(\xi)$, Eq. (1.61). Fig. 4.4a shows an example of a nonlinear and conventional Fourier spectrum. The dependence of the difference on the spectral parameter ξ for a typical signal from our testing set is shown in Fig. 4.4b. The critical decrease of the difference at ξ region below -100 and above 100 occurs because the amplitude of the continuous spectrum at that region also tends to zero. The average maximal difference parameter value over the entire spectrum for all signals from the test dataset is ≈ 9. This fact allows us to argue that the nonlinear effects are essential for the selected testing signals, despite their containing no solitons. Thus, the accuracy of the NFT-Net allows us to perceive the truly nonlinear effects.

In our work we used the conventional forward NFT numerical method to generate training and testing data set pairs: the signal and its respective NF spectrum. For the computation of continuous NF spectrum associated with a given profile $q(t)$ (containing no solitons) having the form of Eq. (4.1), we used the exponential scheme ES4 from the FNFT package[171] (non-fast realisation). It has the accuracy proportional to the fourth power of the time sample

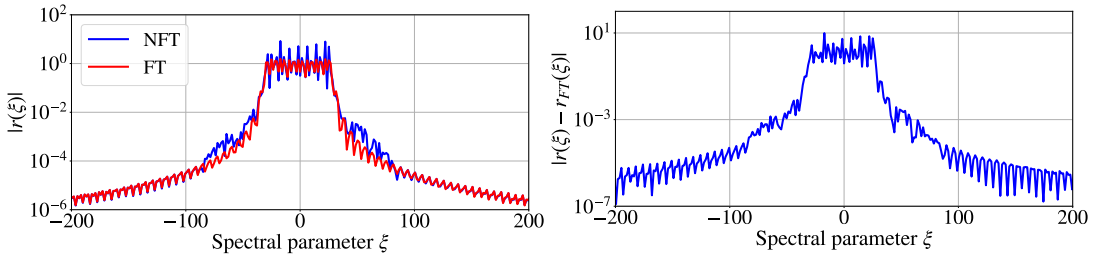


Figure 4.4: **(a)** Example of the amplitudes of Fourier spectrum (FT) and continuous nonlinear Fourier spectrum (NS) for one of the training signals. **(b)** Example of the absolute value of the difference between Fourier spectrum and continuous nonlinear Fourier spectrum for one of the training signal. For both graphs signal energy $E_{\text{signal}} = 39.0$ in non-dimensional units.

size, $\sim (\Delta t)^4$. We note that there exists the fast realisation of the NFT processing with $\sim (\Delta t)^4$ accuracy[176], which can potentially be used for efficient NFT-Net training.

4.4.3 Neural network design and Bayesian optimisation

As mentioned above, the general NF spectrum attributed to a given localised waveform consists of two parts: the discrete spectrum that we do not consider in our current study (our trial pulses do not contain any solitonic component, neither in pure form nor in the noisy case), and the continuous part which is our subject in hand here. The continuous part is retrieved through considering the special Jost solutions (1.16) to the ZS problem (1.15). The goal of our work is to demonstrate the fundamental possibility of replacing the direct calculation of NF spectrum through the numerical solution of the ZS problem (1.15) with the computations employing specially-designed and trained NNs.

The latter task can be addressed using the encoder-decoder approach, where the encoder transforms the input signal into some intermediate vector representation and, later, the decoder converts this representation into the output signal. We notice that the input and output signals can belong to two different data domains. There are several advantages of this approach, e.g. it is quite flexible, so the encoder and decoder structures can differ to match exactly the "nature" of each signal's domain. With this, we train such NNs in the end-to-end style, so the weights of the encoder and decoder will be trained simultaneously and fit each other. A lot of highly efficient encoder-decoder architectures have been designed up to date, e.g. those can demonstrate an efficiency higher than that of a human brain for some specific tasks [177]. For processing quite long sequences (typically more than 1000 data points), the convolutional NNs (CNN) are often more beneficial than the Recurrent neural networks (RNNs). Also, the CNN allows us to parallelise the computations in an efficient way, which is important in our case. Thus, we argue that the encoder-decoder architectures based on CNNs are most suitable for our data and task, though other NN types may also deserve investigation in latter studies.

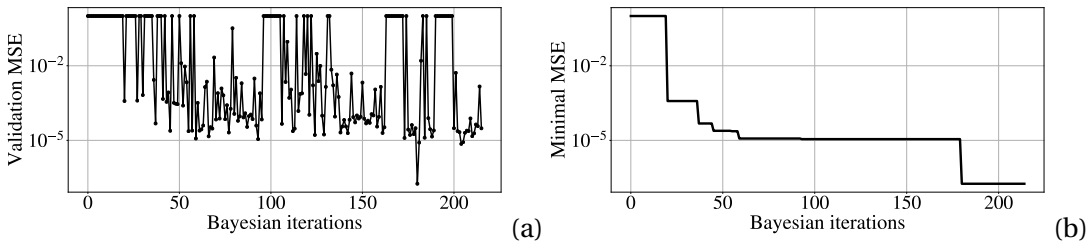


Figure 4.5: **(a)** The dependence of the MSEvalue on the number of Bayesian iteration. **(b)** The same for minimal value of MSE.

As a starting point, we took the WaveNet [178]-based network, which extends the concept of deep CNNs. Models of this type have several advantages, among which we underline the reduction of time required for training the network on long data sequences. However, a significant drawback of this architecture is the requirement to embed a large number of convolutional layers to increase the receptive field. In our work, to increase the effective size of that region, we used convolutions with dilation. This made it possible to exponentially increase the receptive field with the NN depth growth and, therefore, to capture a larger number of data points in the input signal. The choice of Convolution Neural networks (CNNs) for Nonlinear Fourier Transform (NFT) applications over simpler neural network architectures is primarily motivated by CNNs' superior ability to capture spatial hierarchies and complex patterns in data, which are crucial for accurately processing and interpreting signals governed by the nonlinear Schrödinger equation. This inherent capability makes CNNs particularly well-suited for tasks that require an understanding of the complex, nonlinear dynamics present in NFT analyses.

The momentous issue in using NNs to perform any nonlinear transformation is the choice of the optimal network architecture. One of the optimisation methods is to enumerate the possible combinations of NN parameters. But even in the case of a relatively small number of layers, the number of hyperparameters can reach several thousand, which makes the optimisation process very time-consuming, if realisable at all. Thus, the search for an optimisation algorithm for such computationally expensive problems can be extremely difficult. Rather than conducting a conventional dimensioning study for NNs, this research adopts Bayesian Optimization to fine-tune the architecture of the CNN for the NFT-Net. Bayesian Optimization, recognized for its efficiency in finding the optimal distribution of hyperparameters [179], offers a principled and efficient method for exploring hyperparameter spaces. This approach strikes a balance between model complexity and performance, outperforming exhaustive search methods by utilizing prior evaluations to inform subsequent searches. Through the application of Bayesian Optimization, we systematically refined the CNN architecture, achieving optimal performance for tasks involving the NFT without resorting to traditional dimensioning studies.

The Bayesian optimization builds a probabilistic model of the function mapping from hyperparameter values to the objective evaluated on a validation set [179, 180]. By iteratively

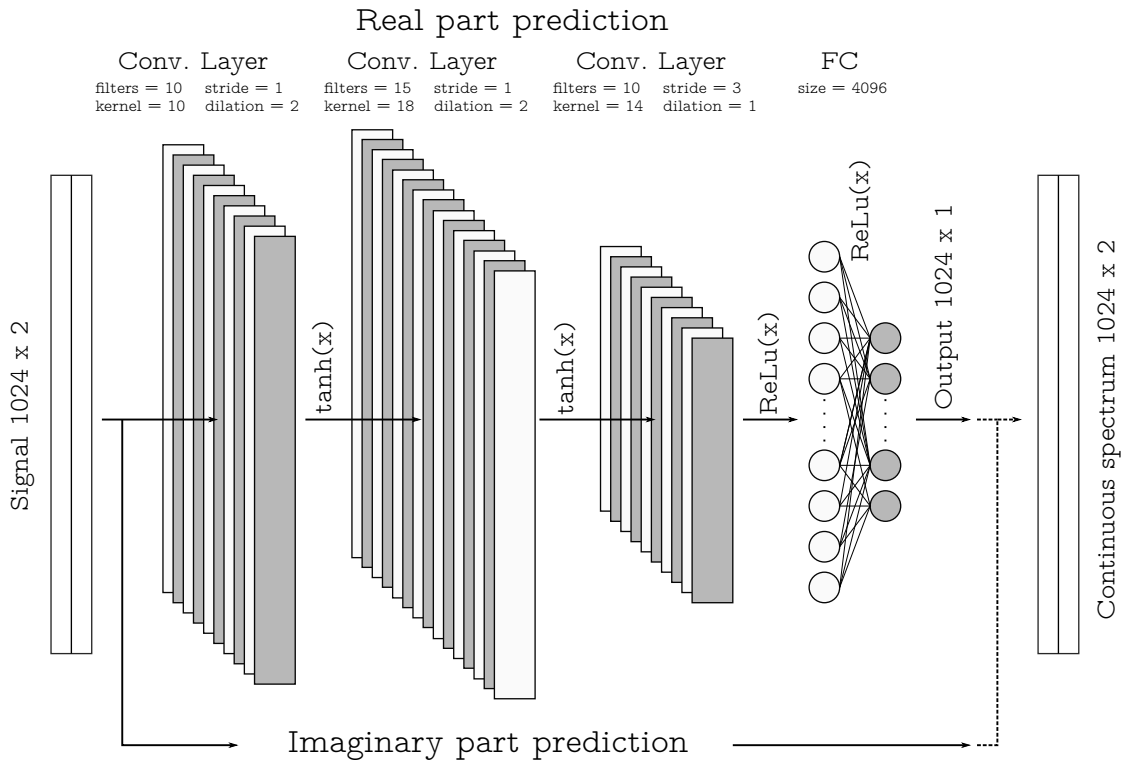


Figure 4.6: The schematic of NFT-Net topology: the extended scheme presents the sequence of operations for the processing of real part; the processing of imaginary part is identical (marked with the long arrow below the scheme). The numbers indicating the layers/arrays sizes refer to our processing 1024 complex-valued signal samples.

evaluating a promising hyperparameter configuration based on the current model, and then updating it, the Bayesian optimization aims to gather observations revealing as much information as possible about this function and, in particular, the location of the optimum. Thus, it tries to balance exploration (hyperparameters for which the outcome is most uncertain) and exploitation (the hyperparameters expected to bring us close to the optimum). An important aspect to note is that the Bayesian optimisation often does not return one specific point in the parameter hyperspace for which the optimised function is minimal. The process converges into some subspace of parameters, where several points can locally minimize the function [179].

We manipulate the following hyperparameters for the convolutional part of the neural network: the number of convolutional layers, the number of filters, the kernel size, stride, dilation, and the activation function for each layer. After the convolutional part, there are 2 fully connected layers, the second of which has a fixed size (1024, which corresponds to the size of the output vector). The size and activation function of the first fully connected layer was also a hyperparameter for optimization. The optimization process included fine-tuning the number of convolutional layers within a range of 2 to 4, allowing for variability in the depth of feature extraction. For each convolutional layer, the number of filters was set to span between

10 to 256, depending on the layer, to diversify the capacity for feature detection. The kernel size, crucial for defining the scope of input considered at each convolutional step, varied from 1×1 to 20×20, offering flexibility in the granularity of analysis. Stride and dilation parameters, both adjustable from 1 to 3, were optimized to alter the convolution operation's coverage and input sampling methodology, respectively. The architecture also incorporated 1 to 3 dense layers, with the neuron count in each layer ranging broadly from 128 to 4096, providing a robust framework for pattern recognition and decision-making. Activation functions – ReLU, sigmoid, and tanh – were selectively employed across the network to introduce nonlinearity, enhancing the model's ability to model complex relationships in the data. This comprehensive and detailed parameterization ensures a thorough exploration of the model's architectural landscape, aiming to identify configurations that yield optimal performance metrics. In each iteration of Bayesian optimization, we retrain the NN and calculate the MSEloss on the validation dataset. The architecture of the NN is considered better if it results in a lower validation loss. Therefore, we select the architecture that achieves the lowest loss value on the validation dataset.

For the optimisation, we used a dataset without additional noise and employed only the real part of the continuous spectrum for the prediction. After that, the "optimal" architecture (but not weights) is fixed, and is no longer changed to predict the imaginary part of the continuous spectrum or for our operating with the datasets with additional noise. The loss function was optimised for each architecture. We used the MSEas the loss function, aiming to minimise the MSE between the network output and the target output computed with the conventional NFT method [171]. In training, we employed the Adam (Adaptive Moment Estimation) optimisation algorithm with the learning rate of 1e-4 [181]. The learning process of each point in the parameter hyperspace was stopped if the value of the loss function did not decrease over 5000 epochs. We chose this large epoch stopping-criterium number to neutralise the factor of randomness in the learning process, which appears due to the random choice of the initial weights. Additionally, we checked the value of the loss function on the validation set to prevent the overfitting, but for the amount of training data used, the overfitting was not observed. Figure 4.5 presents the dependence of the MSE value and dependence of its minimum on the Bayesian iteration number. For architectures with more than 20 million training parameters, we set the value of the loss function to 1.0: this explains the upper cut-off limit in the figure. It is apparent from Fig. 4.5 that the optimisation has identified a subspace where many architectures have approximately the same value of the loss function at the level of 10^{-5}. However, there was a point where the value was at the level of 10^{-7}. Thus, we took this point (a set of hyperparameters) as the optimal one. After finding the optimal architecture, each NN's weights were trained for different SNR but keeping the same optimal architecture parameters. On average, with the amount of data used, our learning process took 50 000 epochs to reach the minimum for each noise level.

The original signal and NF data for the continuous spectrum are complex-valued functions. Therefore, two networks with the same architecture are to be used for the whole transformation; each identical part is responsible for the computation of either the real or imaginary

parts of the resulting arrays, which contain the values of continuous NF spectrum defined in Eq. (1.24). Fig. 4.6 depicts the schematic for the entire optimised NFT-Net architecture. The convolutional part consists of three layers with 10, 15 and 10 filters. Kernel sizes of the first and third convolutional layers are 10, and for the layer between them, it is 18. As noted above, we took the dilation value for each layer as one of the sought hyperparameters. For NFT-Net, the optimisation gave that the first two layers have dilation 2, stride 1 and "tanh" activation function, and for the third layer, the dilation is 1 with stride 3 and "ReLu" activation. After the CNN part we put the flattening layer, not shown in the figure (but affecting the processing complexity), and two fully-connected layers with 4096 and 1024 neurons. The exemplary picture of how the designed NN works on one signal is given in Fig. 4.7. In this figure, we show the results of the NN-based NF spectrum computation for the noiseless case. Already from this figure, we can notice that the result produced by our NN and that obtained from conventional NFT routine[171] are very similar.

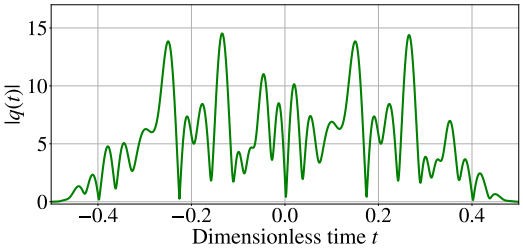
4.4.4 Studying the NFT-Net performance for computing NF spectra of noisy signals

In this section we analyse the NFT-Net performance and the denoising property of the NN. We compare the deviations in the obtained nonlinear spectrum calculated with the NFT-Net and calculated with the conventional NFT applied to the same signal without noise. To quantify the performance rendered by the NFT-Net application with the performance of conventional algorithms applied to noisy signals, we use the following metric:

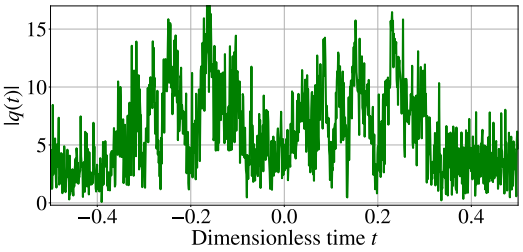
$$\eta = \frac{1}{S}\sum_{i=1}^{S}\langle \eta_i(\xi)\rangle_\xi, \quad \eta_i(\xi) = \frac{|\{r_{\text{predicted}}(\xi)\}_i - \{r_{\text{actual}}(\xi)\}_i|}{\langle |\{r_{\text{actual}}(\xi)\}_i|\rangle_\xi}, \tag{4.7}$$

where S is the total number of signals in the validation set, $\langle\cdot\rangle_\xi$ denotes the mean over the spectral interval, $\{r_{\text{predicted}}(\xi)\}_i$ and $\{r_{\text{actual}}(\xi)\}_i$ correspond to the value of reflection coefficient $r(\xi)$ computed for the signal number i at point ξ (we compare the quantities for the validation data set). The label "predicted" refers to the result produced by the NFT-Net on the noisy signal, and "actual" marks the $r(\xi)$ value obtained using the conventional NFT algorithm[171] for the noiseless signal. The relative error $\eta(\xi)$ is determined at the point ξ, so we use $\langle\eta(\xi)\rangle_\xi$ to estimate the overall mean of the error for one signal, and use Eq. (4.7) to evaluate the error for the entire validation dataset. We stress that the metric was chosen in such a way as to take into account even the regions where the value of the spectrum is much less than one.

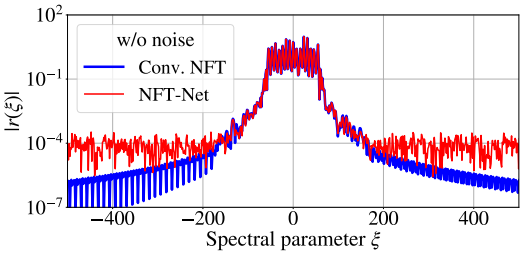
The results of our comparison for $r(\xi)$ computation using different SNR levels for NFT-Net are presented in Table 4.1, and are arranged as follows. The first column of the table identifies the SNR value in dB for the validation signals, i.e. the level of noise for the signals which we analyse. The first row of the table displays the SNR values of noisy signals from the training set, i.e. it shows the noise level of the signals on which the NFT-Net was trained. We notice that the case SNR = 30 dB corresponds to almost negligible noise, while for SNR = 0 dB our noise energy is equal to that of our signal, which signifies a very intensive noisy corruption. Thus, each column in the table corresponds to the results produced by the NN trained on



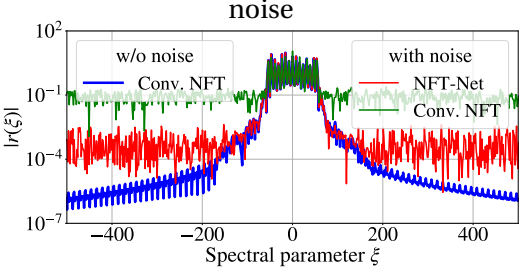
(a) Amplitude of original signal



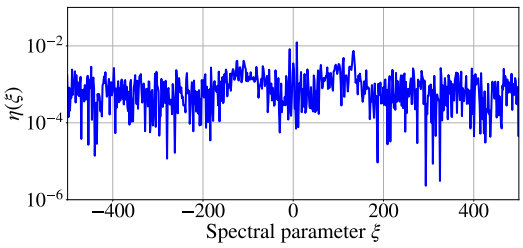
(b) Amplitude of signal with additional noise



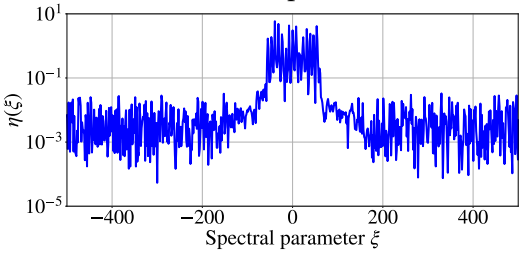
(c) Amplitude of reflection coefficient for continuous spectrum



(d) Amplitude of reflection coefficient for continuous spectrum



(e) Difference between $r(\xi)$ calculated using NFT-Net and Conv. NFT



(f) Difference between $r(\xi)$ calculated using NFT-Net and Conv. NFT

Figure 4.7: Panel **(a)** shows an exemplary amplitude of a original complex WDM signal $q(t)$ versus time. Below (panel **(c)**) is the amplitude for calculated scattering coefficient $r(\xi)$ associated with the signal from the pane above. The blue line corresponds to the data obtained using the conventional NFT method, the red line corresponds to the NFT-Net result. The difference between the scattering coefficients for signal example calculated by these methods is shown in panel **(e)**. Pane **(b)**: the same plot for complex signal $q(t)$ with the addition of Gaussian noise. The SNR value used is 5 dB. Plot **(d)** shows the result of calculating the continuous spectrum for the noisy signal using the FNFT method (green line) and using the NFT-Net trained at the same noise level (SNR = 5 dB, red line) and original spectrum (for noiseless case, blue line). For NFT-Net trained with noise, pane **(f)** below shows the difference between the predicted scattering data for the example of noisy signal and the reflection coefficient calculated by conventional NFT for that signal without noise.

the signals with the chosen level of a noisy corruption. The number in each cell shows the averaged metric value, Eq. (4.7), where for the computation of $\{r_{\text{predicted}}(\xi)\}_i$ we used the NFT-Net trained on the signals with SNR values shown in the first row and applied to the validation signals having the SNR values given in the first column. The "Conv. NFT" column shows the error value for the numerical result of the fast NFT method on the signals with added noise, where the respective SNR is presented, again, in the first column. The value of metric (4.7) corresponding to the conventional NFT method applied to noiseless signals is, obviously, zero: the results provided by the conventional NFT without noise are taken as the true ones. When the NFT-Net produces a less accurate result compared to the conventional NFT applied to the noisy signal, the cell is marked with grey shadowing; when the NFT-Net outperforms the conventional NFT method, i.e. it successfully purifies our signal from noise, the respective cell is not highlighted (white). Whence, the white area size in each table demonstrates how well the NFT-Net retrieves the nonlinear spectrum for noise-corrupted signals.

			Training SNR level, dB								
		Conv. NFT	w/o noise	30	25	20	17	13	10	5	0
Validation SNR level, dB	w/o noise	0	8.39e-4	6.52e-3	9.43e-3	1.26e-2	1.61e-2	2.38e-2	3.59e-2	7.43e-2	1.42e-1
	30	6.91e-2	5.54e-2	9.56e-3	1.11e-2	1.36e-2	1.68e-2	2.42e-2	3.63e-2	7.49e-2	1.44e-1
	25	1.23e-1	9.84e-2	1.40e-2	1.39e-2	1.51e-2	1.78e-2	2.45e-2	3.63e-2	7.47e-2	1.43e-1
	20	2.21e-1	1.74e-1	2.53e-2	2.18e-2	1.97e-2	2.08e-2	2.58e-2	3.65e-2	7.40e-2	1.43e-1
	17	3.10e-1	2.41e-1	3.96e-2	3.23e-2	2.63e-2	2.53e-2	2.78e-2	3.70e-2	7.31e-2	1.42e-1
	13	4.89e-1	3.66e-1	7.74e-2	6.12e-2	4.53e-2	3.97e-2	3.54e-2	3.98e-2	7.06e-2	1.38e-1
	10	6.78e-1	4.88e-1	1.29e-1	1.03e-1	7.36e-2	6.23e-2	5.12e-2	4.85e-2	6.87e-2	1.33e-1
	5	1.16e+0	7.26e-1	2.73e-1	2.31e-1	1.72e-1	1.43e-1	1.15e-1	9.93e-2	7.98e-2	1.17e-1
	0	2.00e+0	9.48e-1	4.79e-1	4.37e-1	3.60e-1	3.12e-1	2.59e-1	2.29e-1	1.74e-1	1.16e-1

Table 4.1: Comparison of the NFT-Net performance against the conventional NFT in the computation of coefficient $r(\xi)$. The table presents the results for the optimised NFT-Net architecture from Fig. 4.6. The values in the cells show the error value (4.7) for each specific pair of training and validation sets SNR. The gray cells correspond to the cases when the accuracy of the NFT-Net nonlinear spectrum restoration is lower than that of fast NFT, i.e. the NN does not denoise the signal well, while the white cells correspond to the cases when the accuracy of the continuous NF spectrum rendered by the NFT-Net is higher, i.e. the NN effectively denoises the result.

Table 4.1 shows the error values for the restoration of $r(\xi)$ coefficient (1.24) of a noiseless and noisy-perturbed signals (4.1), by the NFT-Net architecture given in Fig. 4.6. The first row in the table corresponds to the noiseless case. It is always marked with grey, which means that the NN cannot provide any better results than the benchmark ones rendered by the conventional fast NFT method used to generate the training data.

However, the values of the error for noise-corrupted signals reveal interesting tendencies. It follows from the table that for the low training noise level (up to 10 dB, columns three through nine), the NFT-Net error is typically lowest for the noiseless validation dataset (second row). Thus, the addition of low noise in the training dataset only degrades the NFT-Net restoration capability, even though this decrease is not significant. This NFT-Net feature can be deemed as the NN's being "confused" by the weak noise in its training in the nonlinear transformation identification. For the most interesting case of high noise, the network works best for samples

where the SNR value is the same for the validation and training sets. In such cases, the relative error is about 8-12%, while the error for conventional NFT is at the level of 100-200%. Another fact is that with decreasing noise (rows from bottom to top) in the validation set, the error value remains at approximately the same level after the cell corresponding to the same training and validation noise values. These results confirm that the presented NN architecture is capable of performing the desired nonlinear transformation, the NFT, and, in addition, it can also work as an effective denoising element when the noise level becomes non-negligible.

The examples of original and noise-corrupted signals and the corresponding nonlinear continuous spectra are given in Fig. 4.7, where we used the NFT-Net for the computations. Figures 4.7b and 4.7d show that when the additional noise distorts the signal, the conventional numerical algorithms naturally produce the noise-distorted nonlinear spectra. Figs. 4.7e and 4.7f show the relative error value $\eta(\xi)$ (4.7) for the continuous spectrum prediction with NFT-Net for the signal without noise (left) and the signal with noise (right), and the reflection coefficient computed for the original signal by the conventional NFT (marked as "Conv. NFT" in the panes' legends). In Figs. 4.7c, 4.7e, the NFT-Net is trained on the dataset without adding noise, and in Figs. 4.7d, 4.7f, the NFT-Net is trained on the dataset with additional noise for SNR = 5 dB. Figs. 4.7c and 4.7d show that in the presence of noise, the fast NFT results begin to deviate noticeably from the original (noiseless) values, while the NFT-Net tends to denoise the resulting nonlinear spectrum.

4.4.5 NFT-Net performance for the restoration of NF coefficient $b(\xi)$ attributed to noisy signals

In addition to the coefficient r, from the optical communications perspective it is instructive and important to check how the proposed architecture would work to predict the NF coefficient b, Eq. (1.21). We note that the optical transmission method coined b-modulation [69, 71, 86], where we operate with the modulation of the b-coefficient, has proven to be the most efficacious technique among different NFDM methods proposed [57, 73]. Moreover, for the practical case when our signal has a finite extent, the continuous part of the NF spectrum can be completely described by the b-coefficient only, because the second NF coefficient $a(\xi)$ can be calculated from $b(\xi)$ profile, see Eq. (1.58) in Methods. Our goal here is to demonstrate that the same NFT-Net structures can be used for the both $r(\xi)$ and $b(\xi)$ computation, when the NN is trained on the respective dataset. As the loss function, we now use the MSE build on the b-coefficient samples, and the MSE is also used as our quality metric in the respective tables:

$$\eta_b = \frac{1}{S}\sum_{i=1}^{S}\langle\eta_{b,i}(\xi)\rangle_\xi, \qquad \eta_{b,i}(\xi) = \frac{|\{b_{\text{predicted}}(\xi)\}_i - \{b_{\text{actual}}(\xi)\}_i|}{\langle|\{b_{\text{actual}}(\xi)\}_i|\rangle_\xi}. \tag{4.8}$$

The notations are the same as we used in (4.7): the labels "predicted" and "actual" correspond, respectively, to the result of the NFT-Net applied to noisy signals and the result produced by the conventional NFT routine applied to noiseless signals.

We carried out the analysis of the NFT-Net performance for the restoration of b-coefficient using the same approach as we did in the previous subsection for $r(\xi)$. Our results for noise pulses with the different level of noise are summarised in Table 4.2. We checked that the NFT-Net configurations when applied to the computation and denoising of $b(\xi)$ revealed the same tendencies for the quality of restoration as we observed in the previous subsection devoted to the reflection coefficient $r(\xi)$. A similar situation as was observed for coefficient

			Training SNR level, dB								
		Conv. NFT	w/o noise	30	25	20	17	13	10	5	0
Validation SNR level, dB	w/o noise	0	7.12e-3	5.37e-3	5.87e-3	6.49e-3	8.69e-3	1.07e-2	1.20e-2	1.58e-2	1.77e-2
	30	1.15e-1	5.83e-2	6.73e-3	6.69e-3	7.16e-3	8.95e-3	1.08e-2	1.20e-2	1.58e-2	1.77e-2
	25	2.05e-1	1.02e-1	1.02e-2	8.70e-3	8.48e-3	9.54e-3	1.09e-2	1.21e-2	1.58e-2	1.77e-2
	20	3.64e-1	1.75e-1	2.13e-2	1.51e-2	1.18e-2	1.12e-2	1.14e-2	1.23e-2	1.59e-2	1.78e-2
	17	5.14e-1	2.38e-1	3.66e-2	2.41e-2	1.59e-2	1.33e-2	1.21e-2	1.26e-2	1.60e-2	1.78e-2
	13	8.14e-1	3.44e-1	7.73e-2	4.94e-2	2.64e-2	1.88e-2	1.39e-2	1.36e-2	1.63e-2	1.80e-2
	10	1.15e+0	4.41e-1	1.31e-1	8.58e-2	4.21e-2	2.70e-2	1.67e-2	1.51e-2	1.69e-2	1.84e-2
	5	2.04e+0	6.22e-1	2.70e-1	1.96e-1	9.95e-2	5.91e-2	2.81e-2	2.18e-2	1.95e-2	1.99e-2
	0	3.60e+0	7.98e-1	4.54e-1	3.68e-1	2.24e-1	1.42e-1	6.34e-2	4.43e-2	2.93e-2	2.43e-2

Table 4.2: Comparison of the NFT-Net performance against the fast conventional NFT in the computation of coefficient $b(\xi)$. The table presents the results for the NFT-Net architecture from Fig. 4.6. The values in the cells show the error value (4.8) for each specific pair of training and validation sets SNR. The grey cells correspond to the cases when the accuracy of the NFT-Net nonlinear spectrum restoration is lower than that of fast NFT, i.e. the NN does not denoise the signal well, while the white cells correspond to the cases when the accuracy of the continuous NF spectrum rendered by the NFT-Net is higher, i.e. the NN effectively denoises the signal.

$r(\xi)$, remains in this case. The error is minimal for a noiseless validation set. However, this trend now continues for high noise levels. A similar tendency is observed all over the results: the values above the diagonal vary slightly. The additional observations when dealing the b-coefficient are as follows. An interesting difference from the case relevant to $r(\xi)$, is that the metric value (4.8) in the case of predicting $b(\xi)$ is less, and the grey-shadowed region in Table 4.1 is larger compared to what we see in Table 4.2 for the b-coefficient. From the results, it is clear that the prediction accuracy is higher for the b-coefficient. It means that our NN generally works more accurately for the restoration of coefficient $b(\xi)$ than for $r(\xi)$. This result can be expected, as the noise-perturbed $r(\xi)$ contains the noisy contributions from both $a(\xi)$ and $b(\xi)$, while the b-coefficient involves only its noisy contribution, and thus gets corrupted less. So in the latter case, the NN has to clean off less noise.

Figure 4.8 summarizes the above and shows the calculation errors (4.7) and (4.8) for NFT-Net architecture. The plot actually visualises the values and tendencies from Tables 4.1 and 4.2. For both $r(\xi)$ and $b(\xi)$ coefficients, the NN outperforms the fast NFT results when the NFT-Net gets trained on the data with additional noise.

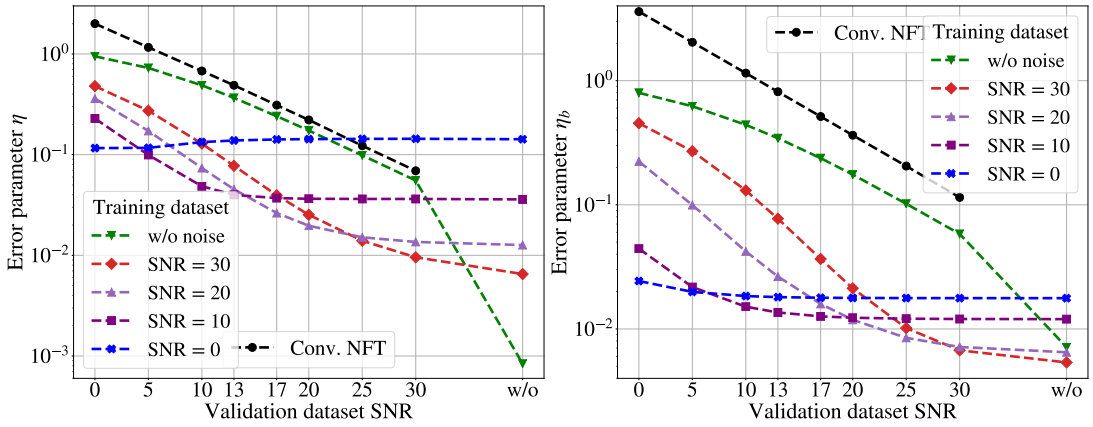


Figure 4.8: **(a)** The dependence of the error parameter η (4.7) for coefficient $r(\xi)$ on the SNR value of the validation dataset for fast conventional NFT and NFT-Net trained at different noise levels. **(b)** The same for the error parameter η_b (4.8) for coefficient $b(\xi)$. The black line represents the error value for fast NFT applied to noisy signals, and the points below this line refer to the cases when the NFT-Net outperforms conventional computations. Other lines show the value of the error in calculating the continuous spectrum using NFT-Net, trained with different noise levels: green – without additional noise, red – with additional noise with SNR = 30 dB, violet – at SNR = 20 dB purple – at SNR = 10 dB, blue – at SNR = 0 dB.

4.4.6 Complexity analysis

One of the important metrics in the development of signal processing tools is the complexity of the processing device, i.e. the number of elementary arithmetic operations that the processing unit employs to reach its goal. Quite often we need to analyse the interplay between the complexity and accuracy of the processing unit. Thus, here we perform the complexity analysis for the NFT-Net.

In our case, we concentrate only on the number of multiplications, since in practical implementation the computational complexity of addition operations is negligible. The number of real multiplications needed for the forward propagation of the model, as introduced in [150] for several types of NN layers, is also used to calculate the computational complexity of the NFT-Net in this research.

The overall complexity C of the NFT-Net can be presented as the sum of two constituents: the complexity of densely-connected block C_{dense} and the complexity of convolutional block C_{conv}. For the calculation of C_{dense} the same formula as in [150] can be used, where we have n_i inputs, n_1 neurons in the hidden layers, and n_o outputs, and the complexity is defined as:

$$C_{\text{dense}} = n_1 \cdot (n_i + n_o), \tag{4.9}$$

In the case of the convolution layer, we can change the equation given in [150] to measure the generalised convolutional layer complexity by taking into account the number of filters f

and kernel size k, as well as the effect of padding p, stride s, and dilation d. The complexity $C_{\text{conv, layer}}$ for one layer when the input shape is $[L_{in}, Q_{in}]$, is specified as follows:

$$C_{\text{conv, layer}} = k \cdot Q_{in} \cdot f \cdot \left(\frac{L_{in} + 2 \cdot p - d \cdot (k-1) - 1}{s} + 1 \right), \tag{4.10}$$

where Q_{in} denotes a number of channels, L_{in} is a length of signal samples sequence. Therefore, the total complexity of the NFT-Net in terms of real multiplications per output sequence (1024 complex valued points) is:

$$C_{\text{conv}} = 2 \cdot (C_{\text{conv, 1}} + C_{\text{conv, 2}} + C_{\text{conv, 3}} + C_{\text{dense}}), \tag{4.11}$$

where the factor 2 in front appears due to the use of two identical NNs to predict the real and imaginary parts of the continuous NF spectrum. Turning to our optimised architecture, to process 1024 complex signal samples, the following number of multiplication operations for the optimised architecture is required:

$$\begin{aligned} C_{\text{conv}} = 2 \cdot (&10 \cdot 2 \cdot 10 \cdot 1006 + 18 \cdot 10 \cdot 15 \cdot 972 + 14 \cdot 15 \cdot 10 \cdot 320 + \\ &+ 3200 \cdot 4096 + 4096 \cdot 1024) = 41598208. \end{aligned} \tag{4.12}$$

4.5 Conclusion and Discussion

In summary, Machine learning and Neural network are cutting-edge technologies that have garnered substantial interest in the realm of nonlinear signal processing and optical communications. The neural network architecture proposed in this study illustrates the initial yet promising capability of NNs for the intricate task of analyzing and (de)modulating the complex optical signals prevalent in communication systems. Such advances herald the potential to enhance current optical communication infrastructures by leveraging intelligent algorithms, such as digital back-propagation based on Nonlinear Fourier Transform and NN, even without comprehensive knowledge of the nonlinear phenomena that influence signal quality. Our research indicates that NNs are not only capable of dissecting the internal structure of optical signals but also of forging new signal patterns through the use of autoencoders. The application of NNs to NFT opens new research frontiers in understanding the inherently nonlinear properties of signal structures and their evolution.

Our goal in this work was to demonstrate that the NN can be successfully used for performing the NFT operation, in particular, for computing the profile of continuous NF spectrum. Note that our interest was not only the computation of the continuous NF spectrum, i.e. the nonlinear transformation, but the possibility to denoise signal using NNs.

Once again, we emphasize that Bayesian optimization does not always give the "best" set of parameters. It provides a subspace of hyperparameters in which neural networks with such parameters are best trained on the available dataset. Due to the fact that neural networks are

universal approximators, any sufficiently complex architecture can be trained for a specific task. We can expect that the optimization process can converge endlessly towards increasing the complexity of the network. However, this is not suitable for our task, where we want to minimize the complexity of the network while improving the accuracy of the work. Therefore, we simultaneously limited the number of trainable parameters in the NN during optimization. In our case, during the optimization process, we found an architecture that gives us the best metric value (4.7) and we chose it as the desired architecture. Further, the optimization process could converge to another subspace of hyperparameters, but we stick to the point with the minimum value of the loss function.

We found that this NN, indeed, can perform the NFT operation and denoise the received NF spectrum: the denoising effect is pronounced at medium to high noise levels. To achieve this effect, several realisations of the noise are needed for the neural network to "understand" the influence of noise on the signal. As expected, denoising is typically best when the training and testing data noise levels coincide, though we observed some deviations from this rule for lower noise levels, where the quality of restoration of the NF spectrum also makes a noticeable contribution in the overall error value. When being trained on different noise levels, the NFT-Net was still able to produce denoising, thus demonstrating the design's flexibility. We have shown that conventional NFT calculation methods give "distorted" results when working added noise. In fact, the "distorted" results are actually correct, but from the nonlinear transformation point of view. But from the application's perspective, we are almost always interested in the denoised signals to reduce the embedded data corruption level.

Finally, we note that the problem of recovering a few solitons from a given pulse utilizing NN has been studied in [76, 164, 165, 182]. However, the NN architectures used in these works are much more simple as far as we need to identify and denoise just several solitonic parameters, while in our work we recovered 1024 complex numbers representing the continuous NF spectrum. Here we considered a larger number of solitary modes, where, however, only the total number of solitons in the pulse was studied. Potentially, it is interesting to combine the NN developed in our work with the additional module that can deal with soliton parameters restoration: such a hybrid tool would be able to perform the complete NFT decomposition of an arbitrary decaying pulse. However, from the position of high-speed optical transmission, the signal processing involving the continuous NF part is most important. We also anticipate that our results can find applications in numerous problems that involve the solution of direct scattering problems where denoising of the profiles is required.

To sum up, we investigated the modelling of the NFT operation associated with the focusing NLSE, using the NN with a special structure, which we coined the NFT-Net. We considered here an almost unexplored case dealing with the computation of the continuous part of the NF spectrum. It was demonstrated that the WaveNet-type NFT-Net structure can satisfactorily perform the task of the NF spectrum computation, and the best-performing architecture was obtained by Bayesian hyperparameters optimisation. Moreover, we showed that the same NFT-Net structure can be used to efficaciously retrieve both the reflection coefficient $r(\xi)$ and the

scattering coefficient $b(\xi)$. The most practically important feature of the developed NN-based method is its capability to perform signal denoising. We demonstrated that the NN-based processing can bring about essential improvements in the quality of NF spectrum restoration attributed to noise-perturbed time-domain profiles, compared to the conventional high-accuracy NFT processing method. The advantage in denoising becomes most pronounced at high noise levels, with the maximum restoration quality typically occurring when the SNR of the training data is the same as that of the validation dataset. In the end, we demonstrated that we can compute and denoise the NF b-coefficient with the same NFT-Net architecture, where the restoration quality follows the same trend as we observed for the recovery of the $r(\xi)$ coefficient. From the physical application's perspective, our methods can be useful in tasks where we are interested in separating the meaningful NF spectrum from the noise: it directly refers to the optical transmission applications and can be used in many similar problems. From the mathematical perspective, our work can be reckoned as a step towards developing artificial intelligence-based tools for the solution of integrable PDEs. We anticipate that the proposed approach can be extended well beyond optical communications, to signal processing, fibre Bragg grating design [90, 91] and other applications.

Part II

Machine Learning for Optimizing Optical Communication Systems

5 HpCom: A Data Mining and Optimization Platform for Optical Communication

5.1 Introduction

In the rapidly evolving field of optical communication systems, researchers face increasingly complex problems and large-scale computations. The need for efficient and high-performance simulation tools is paramount. While there are established simulation methods, there is currently a gap in the availability of high-performance software tailored to the unique requirements of optical communication systems. This work aims to bridge this gap by introducing a GPU-accelerated framework – High-Performance COMmunication library [183], designed to enhance simulation capabilities and accelerate innovation in the field.

The use of GPU-based implementations in optical communication system simulations offers several key advantages over traditional CPU-based methods [184]. One of the primary benefits is the inherent parallelism in GPU architectures. With thousands of cores capable of handling multiple operations simultaneously, GPUs are better suited for the parallel computations often found in optical communication system simulations. This leads to improved performance, particularly for tasks involving large-scale matrix operations, vector manipulations, or Fourier transforms.

Scalability is another significant advantage of GPU implementations, as they can be more easily adapted to handle larger problem sizes or more complex simulations. This flexibility is crucial for researchers working on cutting-edge optical communication systems, as computational demands continue to grow. Moreover, GPUs are known for their power efficiency in parallel computations. For large-scale simulations, using GPUs can result in considerable energy savings compared to CPU-based implementations. This is particularly important when generating vast amounts of data for machine learning techniques or tackling high-dimensional optimization problems [185–187].

The accelerated computation offered by GPUs enables researchers to rapidly test hypotheses [188], iterate on designs, and explore a wider range of scenarios in a shorter amount of time. This results in faster discovery of new insights and innovations, as well as more efficient

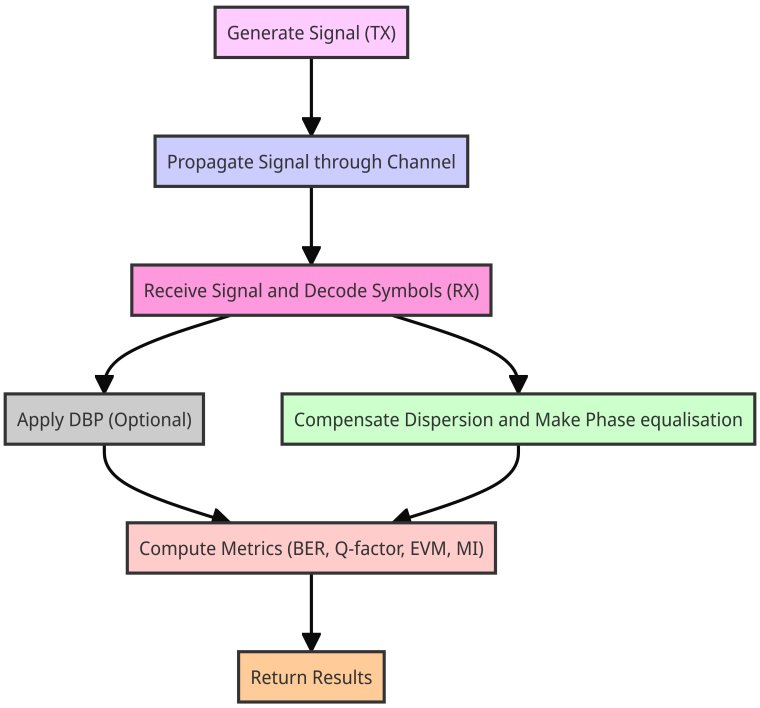


Figure 5.1: Framework architecture for optical communication system simulation, featuring optimized transceiver (Tx) design, GPU-accelerated SSFM-based channel model (with N spans of length L), receiver implementation (Rx), and performance metrics evaluation including BER, EVM, and MI.

use of resources for data analysis, interpretation of results, and further development of novel techniques.

By filling the current gap in high-performance software for optical communication system simulations, this GPU-accelerated framework has the potential to transform research in the field. Its user-friendly design, combined with its powerful capabilities, allows researchers to focus on their novel tasks rather than being constrained by well-established simulations. This part provides an in-depth overview of the framework's architecture, highlighting its benefits and showcasing its potential for driving new discoveries in optical communication systems.

5.2 Methodology

The proposed framework is built on the TensorFlow 2.x library, with Python as the primary programming language. It is designed to be compatible with the MATLAB environment through the pyrun function, enabling seamless integration with existing MATLAB workflows.

The main components of the framework include the transmitter (Tx), the optical channel model (using SSFM and its higher-order variations), the receiver (Rx), and the metrics evalua-

tion block (see Fig. 5.1). Besides the aforementioned modules, there are blocks dedicated to compensation and various signal processing techniques. These encompass chromatic dispersion compensation, nonlinear phase equalization, and digital backpropagation. Notably, within these blocks, users have the liberty to incorporate their own Digital signal processing (DSP) techniques or even employ Machine learning (ML)-based methods. By default, the transmitter generates a WDM signal with properties as described below. Once the signal is formed by the Tx, it is passed to the second block - the channel model, where a Standard single-mode fiber (SSMF) optical channel is simulated with variable numbers of spans and an Erbium-Doped Fiber Amplifier (EDFA) scheme. The Rx consists of a matched filter implementation, Chromatic dispersion compensation (CDC), signal demodulation and hard decision blocks. After processing the received information, it can be used to evaluate desired performance metrics such as BER, EVM, MI, or any custom metric.

The Schrödinger equation and the Manakov equation (see Eqs. (1.1) and (1.2)) are two widely-used mathematical models that describe light propagation in single-mode optical fibers, taking into account dispersion and nonlinearity effects [189]. The Schrödinger equation models the evolution of the slowly varying optical field envelope, while the generalized Manakov equation provides a more sophisticated model for light propagation in a two-polarization optical fiber, including polarization-mode dispersion and nonlinear interactions between polarizations [190, 191]. HpCom incorporates both of these models, utilizing a GPU-accelerated SSFM implementation for the efficient and accurate simulation of optical communication systems. This forms the core of the framework, ensuring that it can handle complex scenarios and deliver reliable results. Further details on the implementation can be found in the accompanying framework documentation[183].

By default, our framework employs a classical WDM channel scheme (see Appendix C.1 and code examples C.3 and C.2) for SSMF with an EDFA [192], and ideal transmitter and receiver. However, users can easily customize their channel line by adjusting various parameters for the simulations. For instance, you can explore other types of optical fibers such as TrueWave Classic[193] (TWC) or Large Effective Area Fiber[194] (LEAF) with corresponding parameters. Additionally, you can modify signal parameters to use not only WDM with QAM but also other types of modulations or signals.

5.3 Framework Architecture

High-Performance COMmunication library (HpCom) is structured into several integral modules, as illustrated at Fig. (5.2). Here's a simplified summary of the modules and their functionalities:

`channel` This module facilitates the establishment of channel parameters such as the length of each span, dispersion, nonlinearity, etc., and provides GPU-accelerated simulation of the optical channel.

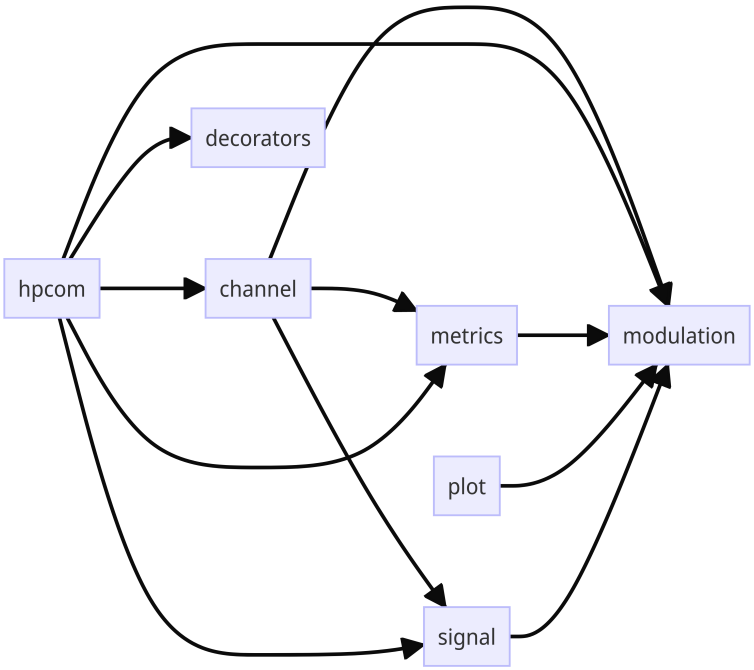


Figure 5.2: Schematic representation of the HpCom library architecture showcasing its modular design for simulating optical communication systems.

`signal` Engaged in the generation and decoding of Wavelength Division Multiplexing or Orthogonal Frequency Division Multiplexing signals.

`modulation` This module is responsible for data modulation, creation of constellations, and modulation/demodulation with various Quadrature amplitude modulation formats (16-, 64-, 256-, 1024-QAM).

`metrics` Metrics module houses optimized functions necessary for computing performance metrics such as BER, MI, EVM, Q-factor or any user-defined metrics. These metrics are essential for evaluating the performance of the communication system.

`plot` This supplementary module aids in technical plotting, providing a visual representation of data and system performance.

`decorators` An additional utility module. It provides python decorators which can enhance the functionality or performance of the code.

5.3.1 Channel Module

The `channel` module in the `hpcom` framework is pivotal for the simulation of signal propagation through a fiber optic channel. It primarily relies on the nonlinear Schrödinger equation

(NLSE) or the Manakov equation to model the propagation dynamics. This module encompasses two main functions: `full_line_model_wdm` and `full_line_model_ofdm`, which are designed for WDM and OFDM signal generation and propagation, respectively.

This module contains several other functions, as enumerated below:

`get_default_channel_parameters`	Constructs a dictionary populated with default channel parameters.
`update_channel_parameters`	Modifies the channel parameters and recalculates internal characteristics to mitigate errors, especially in scenarios where the user overlooks the necessity of manual recalculation.
`update_channel_parameters_from_json`	Initializes default channel parameters, then loads and updates any existing parameters from a specified JSON file.
`create_channel_parameters`	Generates a dictionary with user-provided channel parameters and computes additional values requisite for simulations.

For a comprehensive overview, refer to Appendix C.1. The functions `transceiver_line_model` and `receiver_model` are further discussed in the following subsection.

Functions `full_line_model_wdm` and `full_line_model_ofdm`

The function `full_line_model_wdm` serves as a comprehensive model encompassing signal generation, propagation, reception, demodulation, and metrics computation. Below is a description of the primary parameters:

`channel`	A dictionary containing the channel model parameters (see Appendix C.1 for more details).
`wdm`	A dictionary containing the parameters of the WDM signal (see Appendix C.1 for more details).
`bits` (optional)	Utilized when a specific bit sequence is to be transmitted.
`points` (optional)	Utilized when predefined points are available[1].
`channels_type` (default: 'all')	Specifies whether metrics should be computed for all WDM channels or only the middle channel.

[1] The `points` must be scaled in accordance with the average signal power, refer to the scaling subsection.

`verbose` (default: 0)	Controls the verbosity level of messages, with 0 being minimal verbosity and higher values displaying more information[2].
`dbp` (default: False)	A flag to toggle DBP calculations.
`dbp_parameters` (optional)	Defines the parameters for DBP (see Appendix C.1 for more details).
`optimise` (default: 'not')	When set to a parameter ('ber_x', 'evm_x', 'mi_x'), the function returns only the value for this parameter. Otherwise, function returns all available information (see information below).
`ft_filter_values_tx` (optional)	The Fourier coefficients corresponding to filter values for the transmitter are specified through this parameter. In scenarios where a non-standard filter shape is employed (i.e., a shape divergent from the RRC filter), users have the liberty to define the Fourier modes of the filter explicitly.
`ft_filter_values_rx` (optional)	Fourier coefficients of filter values for the receiver. Same functionality as for `ft_filter_values_tx`.

The function `full_line_model_ofdm` adheres to a similar logic as `full_line_model_wdm`, with a few distinctions. Specifically, it accepts an `ofdm` dictionary to cater to the parameters requisite for OFDM signal processing. Unlike `full_line_model_wdm`, this function does not use the `channels_type`, `ft_filter_values_tx`, and `ft_filter_values_rx` parameters.

Both functions can operate in either single or dual polarization mode. In a single polarization scenario, only the x-polarization is used, while in dual polarization, both x and y polarizations are employed. The functions returns various metrics such as Bit Error Rate (BER), Q-factor, Error Vector Magnitude (EVM), and Mutual information (MI), alongside the constellation points at various stages of the transmission link.

The `full_line_model_wdm` and `full_line_model_ofdm` functions return a dictionary of results when the `optimise` parameter is set to 'not'. The keys in this dictionary, along with their descriptions, are provided below:

`points_orig`	The original points at the transmitter.
`points_noneq`	The constellation points at the receiver, post the matched RRC filtering for WDM and post FFT for OFDM, but prior to any equalization.
`points`	The constellation points after chromatic dispersion compensation.
`points_shifted`	The constellation points after chromatic dispersion compensation and nonlinear phase equalization.

[2]The current maximum verbosity level is 3, subject to change in future versions.

`points_found`	The constellation points found after hard-decision.
`ber`	The BER for transmitted points.
`q`	The Q-factor for transmitted points.
`evm`	The EVM for transmitted points.
`mi`	The MI for transmitted points.

If the `dbp` flag is set to True, indicating that digital backpropagation is to be computed, additional keys are included in the result dictionary:

`points_dbp`	The constellation points post digital backpropagation.
`ber_dbp`	The BER post digital backpropagation.
`q_dbp`	The Q-factor post digital backpropagation.
`evm_dbp`	The EVM post digital backpropagation.
`mi_dbp`	The MI post digital backpropagation.

This organized return structure allows for a thorough analysis of the constellation at the receiver at various stages of post-processing, and facilitates the evaluation of the DBP's impact on the signal transmission quality metrics.

Functions `transceiver_line_model` and `receiver_model`

To accommodate a more flexible tuning of the channel model, two distinct functions are provided: `transceiver_line_model` and `receiver_model`. The former is devoted to the generation of the WDM signal and its propagation through the link, without receiver component, while the latter serves as a receiver for the WDM signal outputted from the fiber model.

The function `transceiver_line_model` facilitates the creation and transmission of a WDM signal through the link, albeit without engaging the receiver functionality. For a comprehensive understanding of the function's inputs, refer to the description provided in the preceding subsection concerning the `full_line_model_wdm` function.

```
result = transceiver_line_model(channel, wdm, bits=None, points=None,
  verbose=0, dbp=False, ft_filter_values_tx=None)
# result contains:
# result = {
#  'signal': [signal_x, signal_y],  # x- and y-polarisations after
  propagation
```

```
#  'points_orig': [points_x_orig, points_y_orig],  # transmitted points for x- and y-
#  'ft_filter_values': [ft_filter_values_x, ft_filter_values_y]  # Fourier coefficients for filter used to create WDM signal for x- and y-
#         }
```

Listing 5.1: Usage of `transceiver_line_model` function

The function `receiver_model` operates as the receiver, taking the WDM signal from the fiber model's output. It employs a matched RRC filter to extract the constellation points, subsequently performing Chromatic dispersion compensation (CDC) and Nonlinear Phase Equalisation (NPE), followed by metrics computation. Similar to transceiver model, the input parameters for this function have the same meaning as for `full_line_model_wdm`, as detailed in the previous subsection.

```
result = receiver_model(wdm, signal, points_orig, ft_filter_values_rx, channels_type='all', verbose=0, optimise='not')
# result same as for full_line_model_wdm function
```

Listing 5.2: Usage of `receiver_model` function

In the current version of the HpCom package, support is extended exclusively to WDM. However, forthcoming iterations will encompass support for OFDM, thereby broadening the scope and applicability of these functions.

5.3.2 Signal Module

The Signal Module encompasses key functionalities pertinent to both Wavelength Division Multiplexing (WDM) and Orthogonal Frequency Division Multiplexing (OFDM) signal generation and decoding. For an introductory exposition on WDM and OFDM, the reader is directed to Section 1.4.

WDM Signal Generation and Demodulation

The principal function for WDM signal generation is given by:

```
signal, info = generate_wdm(wdm, bits=None, points=None, ft_filter_values=None)
```

This function orchestrates the signal generation process. Initially, it checks if the `bits` or `points` parameters are provided; if absent, it generates them based on the stipulated parameters in the `wdm` dictionary, such as average signal power or modulation format. The `points` are scaled in accordance with the average signal power (details in the Appendix C.3).

Subsequently, for each wavelength, the function invokes a subroutine `generate_wdm_base`, which manages the generation process for a single wavelength within the current bandwidth.

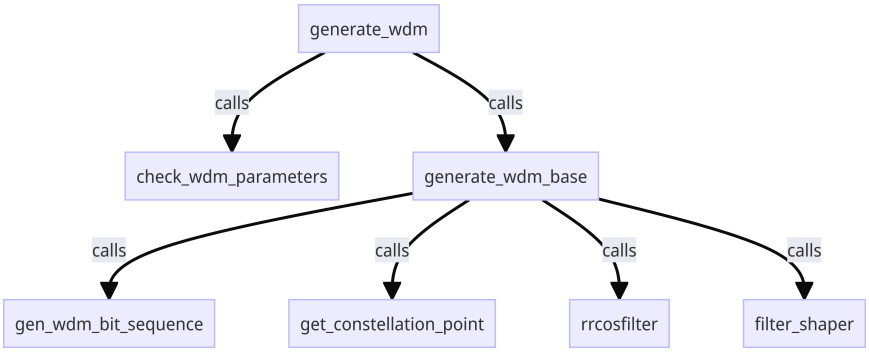


Figure 5.3: Scheme of WDM generation.

The modulated points (e.g., 16-QAM) are taken, and zeros are interspersed between them according to the specified upsampling rate. For instance, with an upsampling rate of 1, every sample corresponds to a constellation point; with a rate of 2, every other sample is a constellation point, with zeros in between, etc.

The convolution operation, as delineated in Eq. 1.69, is central to this process. Leveraging the Fourier Transform, the convolution is transmuted into a multiplication operation in the frequency domain, which is amenable to GPU acceleration. The Fourier spectra of the upsampled points array and the filter are multiplied. Although a RRC filter is typically utilized, the implementation accommodates user-defined filter shapes.

This procedure is reiterated for each WDM channel, engendering the complete WDM signal with a distinct frequency shift for the Fourier spectrum of each channel, as dictated by the `wdm['channel_spacing']` parameter. The convolution operation is executed via the following functions, catering to both time and frequency domain signals:

```
def filter_shaper_spectral(spectrum, ft_filter_val):
 return tf.signal.ifftshift(tf.signal.ifft(tf.signal.ifftshift(spectrum *
   ft_filter_val)))

def filter_shaper(signal, ft_filter_val):
 spectrum = tf.signal.fftshift(tf.signal.fft(signal))
 return filter_shaper_spectral(spectrum, ft_filter_val)
```

With the WDM signal now generated, decoding the constellation necessitates retracing the aforementioned steps in reverse order. As delineated in Section 1.4, the application of a matched filter, as illustrated in Eq. (1.71), is imperative for this process. In the code, this operation is executed by the function `receiver_wdm`, which in turn invokes `matched_filter_wdm`. The latter essentially performs the following actions: it isolates each WDM channel in the frequency domain (the channel separation is dictated by the `wdm['channel_spacing']` parameter), and applies the matched filter to the resultant signal (or spectrum, as reverting to the time domain is not feasible, thus the truncated spectrum is utilized). This matched filtering operation employs the previously referenced `filter_shaper_spectral` function.

The ultimate output of this process are the decoded constellation points. Note that these points are identified prior to the hard decision stage, hence, they can be dispersed across the complex constellation plane.

OFDM Signal Generation and Demodulation

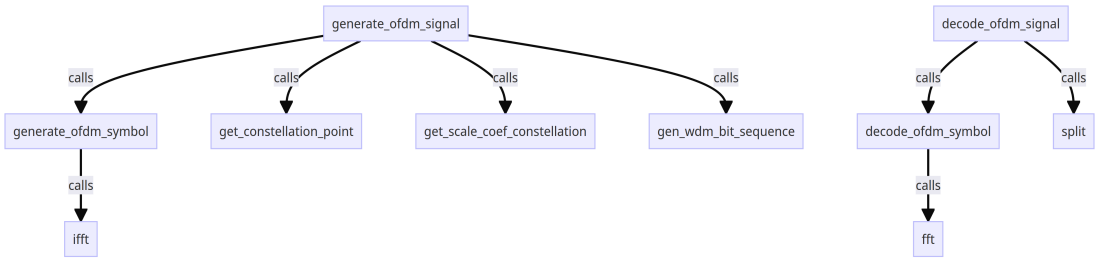


Figure 5.4: Scheme of OFDM signals generating and decoding.

The generation of an OFDM symbol is undertaken by the function `generate_ofdm_symbol`, where, based on the provided or generated bits, constellation points are derived. These points are then subjected to an Inverse Fast Fourier Transform (IFFT) operation to transition from the frequency domain to the time domain, thereby forming the OFDM symbol. If specified, a cyclic prefix is appended to this symbol to mitigate inter-symbol interference.

Subsequently, the `generate_ofdm_signal` function orchestrates the overall generation of the OFDM signal. It initiates by configuring the average power of the signal, especially in scenarios involving two polarizations. A loop iterates through each polarization, within which the bits are either provided or generated anew, followed by the derivation of constellation points. The `generate_ofdm_symbol` function is invoked within this loop to generate the requisite OFDM symbols which are then concatenated to form the complete OFDM signal for the corresponding polarization. This function mirrors the procedural blueprint of OFDM signal generation, extending from bit generation to the aggregation of OFDM symbols.

```
signal, info = generate_ofdm_signal(ofdm_init, bits_init=None,
  points_init=None, seed=0)
```

On the receiving end, the `decode_ofdm_symbol` function is engaged to demodulate a single OFDM symbol. It begins by discarding the cyclic prefix, if present, and then conducts a Fast Fourier Transform (FFT) operation on the OFDM symbol to revert back to the frequency domain, extracting the constellation points.

The broader demodulation of the entire OFDM signal is managed by the `decode_ofdm_signal` function. It dissects the received OFDM signal into individual OFDM symbols, and invokes `decode_ofdm_symbol` for each symbol to retrieve the constellation points. The collected points from all symbols and polarizations are then returned as the output, marking the completion of the OFDM signal demodulation process.

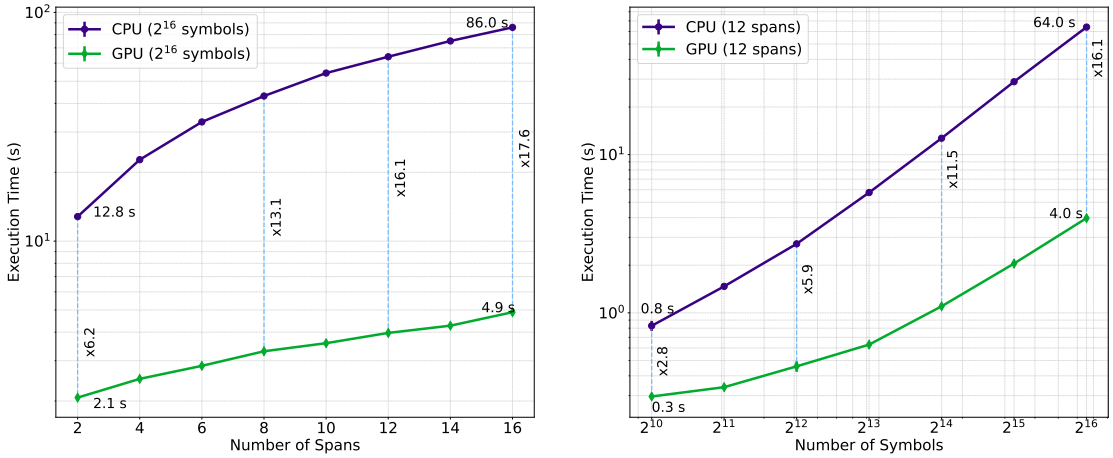


Figure 5.5: The left figure displays the total execution time for all steps of the optical channel simulation with 2^{16} symbols, comparing single CPU core and GPU usage, as a function of the number of spans (SMF, each span is 80 km). The right figure illustrates the relationship between execution time for a single CPU core and GPU usage with a fixed number of spans and varying number of symbols in the simulation.

```
points = decode_ofdm_signal(ofdm_signal, ofdm)
```

5.4 Performance Evaluation

To evaluate the framework's performance, we conducted several tests comparing computation time between CPU and GPU. The signal used was a two-polarization 64-QAM WDM with 1 km per step and 80 km span. The SSMF fiber parameters were: $\gamma = 1.2$ (W$\cdot$ km)$^{-1}$, $D =$ 16.8 ps/(nm$\cdot$km), and $\alpha = 0.21$ dB/km. At the end of each span, optical fiber losses were compensated using an EDFA with a 4.5 dB noise figure. The benchmark calculations were performed using an NVIDIA GeForce RTX 3070 with 8GB GDDR6 and a single core of an AMD Ryzen 9 5900HX with up to 4.6GHz.

The total simulation time consists of four components: the transmitter (Tx) simulation time T_{Tx}, channel simulation time $T_{channel}$, receiver (Rx) simulation time T_{Rx}, and metrics evaluation time $T_{metrics}$. In the current implementation, Tx, Rx, and metrics calculations utilize CPU cores, but employ optimized functions to prevent $O(n^2)$ computational complexity. For instance, with 2^{15} symbols, Tx, Rx, and metrics calculations take approximately 200 ms, significantly less than the propagation time (1230 ms for GPU and 27000 ms for CPU). Optimal results are observed when working within the available GPU memory. When the number of simulation points equals n_symbols $\cdot$ upsampling $= 2^{16} \cdot 2^4$, the overall memory required for calculations is around 6.5 GB. To perform more precise simulations for multi-channel WDM signals, one must control the overall number of points or employ more advanced hardware.

Figure 5.5 displays the dependence of total execution time on the number of spans for a

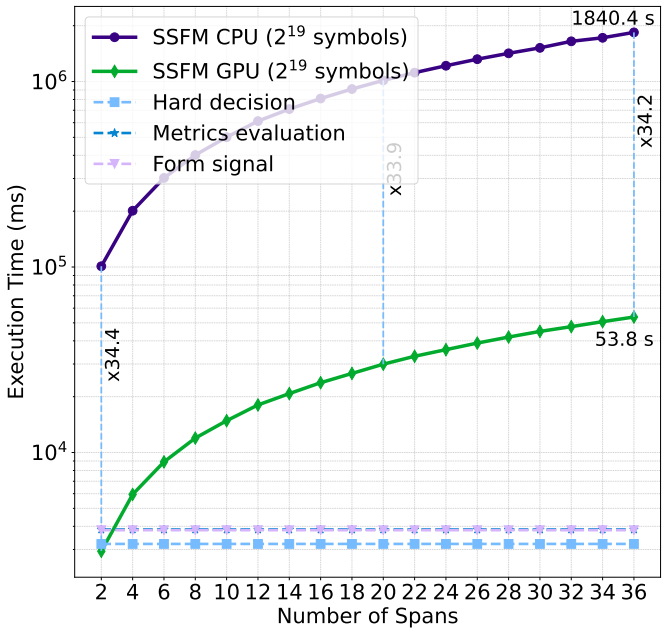


Figure 5.6: Execution time in milliseconds for various simulation steps as a function of the number of spans. Signal formation at the Tx, hard decision at the Rx, and metric evaluation (e.g., BER, EVM, MI) rely on the CPU and are influenced by the number of symbols, depicted as dashed lines. Propagation, which is performed on either the CPU or GPU, is represented by solid lines.

fixed number of symbols (left), and the dependence of total execution time on the number of symbols with fixed propagation length (right). The graphs indicate that as the number of spans or symbols increases, the GPU's speedup factor also grows. For instance, for 2^{16} symbols and 16 spans, the GPU is 17 times faster.

We also evaluated the system performance for an extreme case, where the total memory required significantly exceeds the GPU's capacity. We conducted simulations with 2^{19} symbols (eight times more memory than the current GPU) and separately assessed the time for each component of the simulations (Fig. 5.6). The graph demonstrates that the gain is independent of the propagation length, as expected, and remains approximately 34 times faster on the GPU compared to the CPU. However, for long propagation lengths (or more steps per span), the total time saved is substantial, as the GPU completes the simulation in under a minute (53.8 s), while the CPU takes over 30 minutes.

5.5 Example of Data Mining

Our research examines signal transmission in an optical system that includes a model for SMF with each section being 80 km long. We use WDM, which in this scenarios involves a single channel and single polarization. The system employs a Root Raised Cosine filter with a roll-off

factor of 0.1. We transmit signals at a rate of 34 Gbaud (GBd), with channels spaced 75 GHz apart. The setup has an attenuation coefficient $\alpha = 0.2$ dB/km, an EDFA noise figure of 4.5 dB (though some cases do not include additional EDFA noise), a dispersion coefficient $D = 16.8$ ps/[nm·km], and a nonlinear coefficient $\gamma = 1.2$ [W·km]$^{-1}$.

Figures 5.7 and 5.8 provide examples of data generated with HpCom, highlighting how the BER is calculated for a range of parameters. The fiber length varies from 80 km to 2000 km, equivalent to 1 to 25 spans. The signal average power ranges from -15 dBm to 15 dBm, in increments of 0.5 dBm. The left column of the figures shows results without additional EDFA noise, and the right column includes a 4.5 dB EDFA noise figure. For each parameter set, 2^{16} symbols are simulated. Figure 5.7 presents the scenario for a 16-QAM signal, while Fig. 5.8 is for a WDM signal using 256-QAM. In both figures, the first and second rows depict the BER's dependency on distance and average power. The first row provides a 3D visualization, and the second row shows a 2D color map projection of the same data. The third row offers another view of the second row, with white areas indicating where the BER is below 10^{-6} or above 10^{-1}. In scenarios without additional noise from the EDFA, displayed in the left column, the BER is generally better at lower signal powers because nonlinear effects are minimal. In contrast, the right column, which includes EDFA noise, shows that noise significantly degrades signal quality, especially at low power levels.

The contrast between Fig. 5.7 and Fig. 5.8 is noticeable. In the case of the higher modulation format, 256-QAM shown in Fig. 5.8, the range where we have a low BER is much narrower. Additionally, the third row indicates that, for this higher modulation format, we do not achieve a BER as low as 10^{-6}.

5.6 Conclusion

The GPU-accelerated framework presents a marked improvement in performance over CPU-based implementations, especially for larger problem sizes and complex simulations. The significant speedup achieved with GPU acceleration underscores its wide-ranging benefits in various aspects of optical communication system simulations, such as advanced research, high-dimensional optimization problems, swift validation of theoretical concepts, and adjustment of simulation parameters. Moreover, the accelerated framework facilitates efficient data mining, laying the groundwork for machine learning applications and the development of intelligent systems within the field. This user-friendly, versatile, and powerful framework empowers researchers to concentrate on innovation, ultimately boosting productivity and paving the way for groundbreaking discoveries in the realm of optical communication.

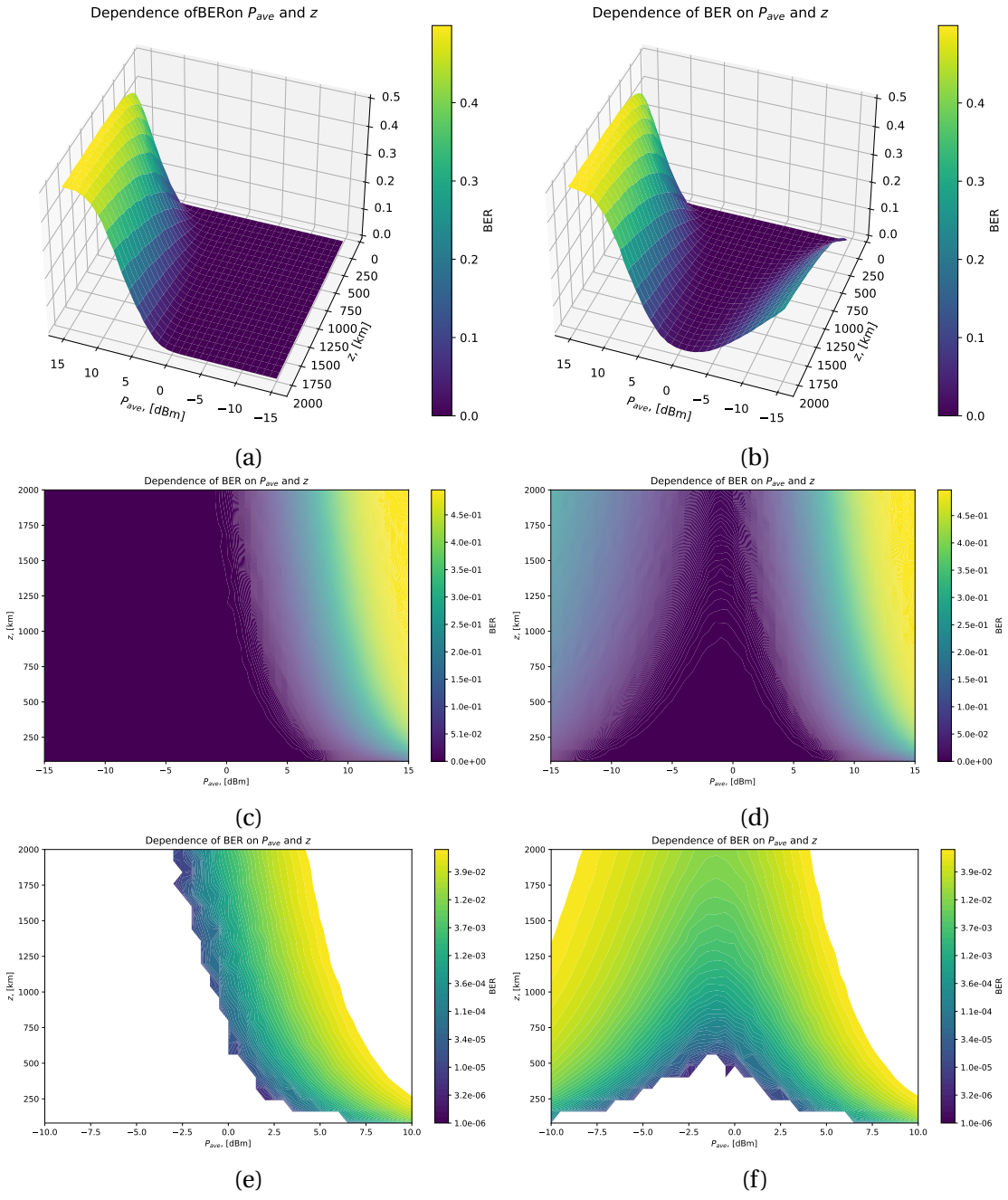


Figure 5.7: 16-QAM. Lenght of the fiber changes from 80 km to 2000 km (from 1 span of fibre up to 25 spans). Signal average power changes from -15 dBm up to 15 dBm (with step of 0.5 dBm). Left column represents the case where there is no additional EDFA noise, right column - 4.5 dB EDFA noise figure. For each set of parameters there is 2^{16} symbols.

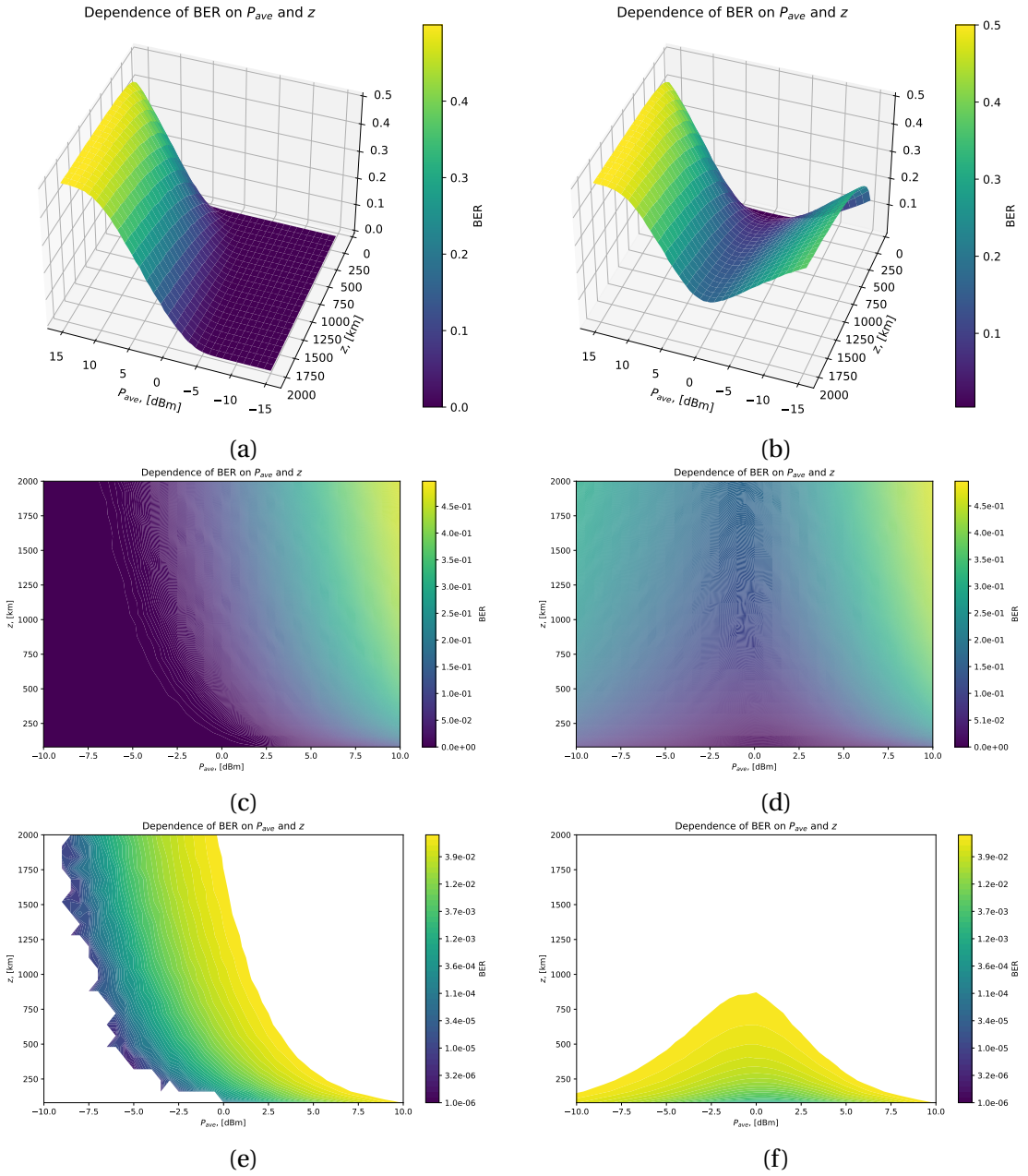


Figure 5.8: 256-QAM. Lenght of the fiber changes from 80 km to 2000 km (from 1 span of fibre up to 25 spans). Signal average power changes from -15 dBm up to 15 dBm (with step of 0.5 dBm). Left column represents the case where there is no additional EDFA noise, right column - 4.5 dB EDFA noise figure. For each set of parameters there is 2^{16} symbols.

6 Data-driven Approach for Nonlinear Equalisation

6.1 Gradient boosting

6.1.1 Introduction

In the realm of machine learning, decision trees have emerged as a potent tool for both classification and regression tasks, owing to their intuitive decision-making process and ease of interpretation. The basic principle behind decision trees is to split the data into subsets based on the values of input features, thus creating a tree-like model of decisions, as illustrated in Fig. 6.1. This methodology is highly effective for data mining applications and has been a subject of extensive research across various disciplines such as statistics, machine learning, and data mining.

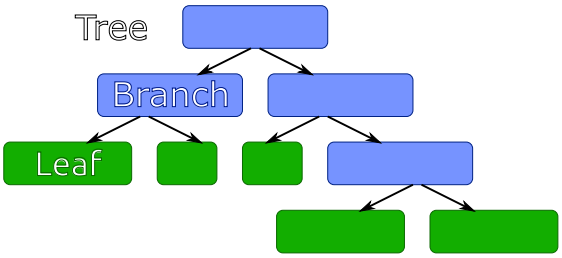


Figure 6.1: One desicion tree - weak learner.

Decision trees are a class of algorithms employed for both classification and regression tasks. They have gained popularity due to their ease of interpretation and resemblance to human decision-making processes [195, 196]. The methodology behind decision trees involves creating a tree-like model (see Fig. 6.1) of decisions based on the values of input features, which is particularly effective for data mining applications [196]. One of the notable advancements in decision tree algorithms came with Ross Quinlan's development of ID3 and its improved version C4.5, which extended the capability to handle both categorical and numerical features through the use of Information Gain Ratio (IGR) (see [197] for information).

Building upon the foundation laid by decision trees, ensemble learning techniques like Random Forest (RF) and Gradient Boosting (GB) have been developed to tackle more complex problems. Random Forests extend the decision tree concept by constructing a 'forest' of decision trees, each trained on a random subset of the data and features. The ensemble's final prediction is an aggregation of the predictions from individual trees, significantly reducing variance compared to a single decision tree, and often achieving higher accuracy.

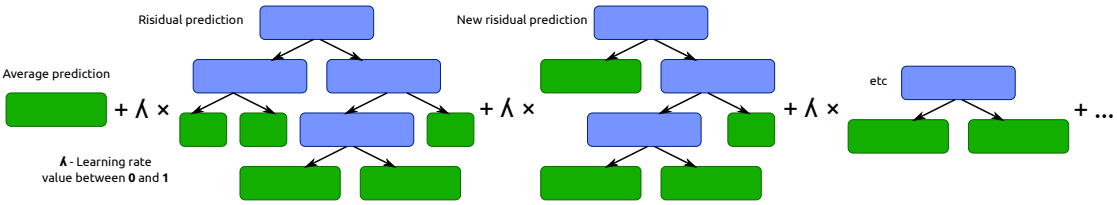


Figure 6.2: Gradient boosting scheme

On the other hand, Gradient Boosting takes a sequential approach to ensemble learning. Unlike Random Forests, which build trees independently, Gradient Boosting iteratively trains decision trees to correct the errors of their predecessors, as depicted in Fig. 6.2. By minimizing a loss function through a gradient descent-like optimization, Gradient Boosting gradually improves the accuracy of its predictions. This iterative correction process enables Gradient Boosting to adapt more effectively to the data, often outperforming other tree-based ensemble methods like Random Forests in terms of predictive accuracy [198]. Variants like XGBoost, LightGBM, and CatBoost have emerged, focusing on enhancing speed and accuracy of this ensemble technique [199].

Gradient Boosting has garnered attention as a promising equalization technique for optical communication systems due to its adaptability and capability to model complex input-output relationships [198–201]. One of its notable advantages is its robustness to overfitting, achieved through techniques like shrinkage, regularization, and early stopping. Furthermore, the inherent parallelizability of Gradient Boosting allows for efficient implementations on modern hardware architectures, including multi-core CPUs, GPUs, and even FPGAs [202], thus offering low-latency and power-efficient solutions crucial for real-time applications in optical communication systems.

In this part, we delve into the application of GB as a novel equalization technique in optical communication systems. We aim to explore its potential to surpass the performance of traditional equalization methods, focusing on increasing predictive performance while simultaneously reducing the computational complexity of the equalization process. This innovative approach positions Gradient Boosting as a promising candidate for equalization tasks in optical communication systems, opening avenues for future advancements in the field.

6.1.2 Regression Trees Foundations

Decision Trees enable the modeling of outcomes based on decision rules inferred from data attributes. While decision trees can be applied for both classification and regression tasks, this work focuses on their use in regression, aiming to predict continuous outcomes.

The operation of regression trees relies on mathematical concepts to split the data at each node in a way that results in subsets with minimized variance. Variance measures the spread of a dataset's values. It is defined as:

$$\text{Var}(Y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{6.1}$$

where Y is the set of target values, y_i is an individual target value, $\bar{y}$ is the mean of the target values, and N is the number of data points.

When selecting the best feature to split on at each node, the decision tree algorithm seeks to maximize the reduction in variance. This is defined as the difference between the variance before the split and the weighted average variance of the two child nodes after the split:

$$\text{Var}_{reduction} = \text{Var(parent)} - \left(\frac{N_{left}}{N}\text{Var(left)} + \frac{N_{right}}{N}\text{Var(right)}\right) \tag{6.2}$$

Maximizing this reduction ensures that the resulting child nodes are as homogenous as possible. Mean Squared Error is another measure used to evaluate the performance of a regression tree, particularly for leaf nodes. It is defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \tag{6.3}$$

where $\hat{y}$ is the predicted value for the data points in the leaf node, typically the mean of the target values of the data points in the node.

1. **Start at the Root:** The entire dataset is considered at the root node.
2. **Feature Selection for Splitting:** The algorithm evaluates each possible split across all features, calculating the variance for the dataset if it were split according to each feature. The reduction in variance for each potential split is determined by comparing the total variance before the split with the weighted sum of the variances of the two resulting subsets. At each node, select the feature that, when split upon, results in child nodes with the lowest total variance.
3. **Splitting the Data:** The dataset is split into subsets based on the selected feature's values, creating child nodes.
4. **Recursive Partitioning:** Repeat the feature selection and splitting process recursively on each child node until stopping criteria are met, such as a maximum tree depth or a

minimum number of samples in a node.

5. **Leaf Nodes and Prediction:** Each leaf node represents a numerical value, typically the mean of the target values in the node. This value is the prediction for new instances that reach this leaf.

6. **Predicting New Instances:** To predict a new instance, traverse the tree from the root, following the branches based on the instance's features, until a leaf node is reached. The value of the leaf node is the predicted outcome.

This approach ensures that the tree is constructed in a way that captures the underlying patterns in the data as efficiently as possible.

6.1.3 Theory of Gradient Boosting

Gradient Boosting (GB) builds a series of weak learners, usually decision trees, and iteratively refines the model by focusing on the areas where the previous models performed poorly. The algorithm starts with an initial model, $F_0(x)$, which could be as simple as predicting the mean (for regression) or the majority class (for classification) for all instances in the dataset.

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma), \tag{6.4}$$

where: N is the number of training instances, L is the loss function, y_i is the true label of the i-th instance, γ is a constant.

In classification tasks, the concept of a "majority class," which denotes the most frequently occurring class in the dataset, is often employed as the starting point for initializing models in algorithms like gradient boosting. This initial model, $F_0(x)$, uniformly predicts the majority class for all instances, providing a baseline from which the algorithm can iteratively improve. However, this approach encounters difficulties in applications like 16-QAM constellation point prediction, where classes (i.e., constellation points) are nearly equally represented, and thus, no single class dominates. In such cases, selecting a class at random for $F_0(x)$ might serve as an unbiased alternative, ensuring a simple yet effective starting point for model refinement. This adaptation underscores the necessity for flexible initial modeling strategies in the face of varied data distributions, particularly in complex classification scenarios.

The algorithm then enters a loop, for $m = 1$ to M (where M is the number of boosting rounds), where in each round it:

1. Computes the pseudo-residuals, which are the gradients of the loss function with respect to the predicted values of the previous model, $F_{m-1}(x)$:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \tag{6.5}$$

2. Fits a weak learner, $h_m(x)$, to the pseudo-residuals, r_{im}.

3. Computes the optimal step size, α_m, that minimizes the loss function when $h_m(x)$ is added to the current model:

$$\alpha_m = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \alpha h_m(x_i)) \tag{6.6}$$

4. Updates the model:

$$F_m(x) = F_{m-1}(x) + \alpha_m h_m(x) \tag{6.7}$$

The final model is the sum of the initial model and the products of the step sizes and the weak learners from each round:

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} \alpha_m h_m(x) \tag{6.8}$$

Each weak learner, $h_m(x)$, is typically a shallow decision tree, and the loss function, L, could be any differentiable loss function such as Mean Squared Error for regression or Logarithmic Loss for classification. The step size, α_m, helps in controlling the contribution of each weak learner, aiding in reducing overfitting and improving the model's generalization capability.

Loss Functions

Loss functions measure the difference between the predicted values and the true values. In Gradient Boosting, differentiable loss functions are used. The two common ones are Mean squared error (MSE) (for regression) and Logistic Loss (for binary classification).

The loss function for MSE is given by:

$$L(y, F(x)) = \frac{1}{N} \sum_{i=1}^{N} (y_i - F(x_i))^2 \tag{6.9}$$

Taking the derivative with respect to $F(x_i)$, we get:

$$\frac{\partial L(y, F(x))}{\partial F(x_i)} = -2(y_i - F(x_i)) \tag{6.10}$$

The loss function for Logistic Loss is given by:

$$L(y, F(x)) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(F(x_i)) + (1-y_i)\log(1-F(x_i))] \tag{6.11}$$

Taking the derivative with respect to $F(x_i)$, we get:

$$\frac{\partial L(y, F(x))}{\partial F(x_i)} = \frac{F(x_i) - y_i}{F(x_i)(1-F(x_i))} \tag{6.12}$$

In these formulas: N is the number of samples, y_i is the true label of the i-th instance, $F(x_i)$ is the predicted value for the i-th instance.

For MSE, the derivative is straightforward and results in a simple expression involving the difference between the actual and predicted values. For Logistic Loss, the derivative involves the quotient rule, and the expression relates the difference between the predicted probability and the actual label, normalized by the product of the predicted probability and its complement.

Regularization

Regularization is a technique used to prevent overfitting by adding a penalty on the complexity of the model. In Gradient Boosting, regularization can be achieved through various means:

1. **Shrinkage (Learning Rate)**: A small learning rate (also known as shrinkage) can be used to reduce the step size at each boosting step, making the boosting process more conservative.

   $$F_m(x) = F_{m-1}(x) + \nu \cdot \alpha_m h_m(x)$$

   where ν is the learning rate, $0 < \nu \leq 1$.

2. **Tree Complexity**: The complexity of the individual trees can be controlled by setting a maximum depth, minimum samples per leaf, or other similar parameters.

3. **L1 (Lasso) and L2 (Ridge) Regularization**: These regularization terms can be added to the loss function to penalize the coefficients of the features.

   $$L_{\text{regularized}}(y, F(x)) = L(y, F(x)) + \lambda_1 \sum |\beta| + \lambda_2 \sum \beta^2$$

   where β are the coefficients of the features, and λ_1 and λ_2 are the regularization parameters.

6.1.4 Methodology

In this study, we introduce a novel approach to nonlinear equalization in optical communication systems by employing a GB algorithm based on decision trees. We used an optical channel model of Standard single-mode fiber (SSMF) with Erbium-Doped Fiber Amplifiers (EDFAs). The signal format is a 16-quadrature amplitude modulation (16-QAM) WDM with dual polarization and a symbol rate of 34.4 GBd. The pulse shaping employed a digital Root Raised Cosine (RRC) filter with a roll-off factor of 0.1. The transmission distance varied between 10, 15, and 20 spans of 80 km each. The EDFA noise figure was set at 4.5 dB. The average signal power range was from 1 to 8 dBm, but for training, we used power from 4 dBm to 8dBm. The signal propagation through the fiber was represented by a generalized Manakov equation using the GPU-accelerated split-step Fourier method[183]. The fiber parameters included a wavelength of $\lambda = 1550$ nm, a dispersion coefficient of $D = 16.8$ ps/(nm$\cdot$km), and a nonlinear coefficient of $\gamma = 1.2\,\mathrm{W}^{-1}\cdot\mathrm{km}^{-1}$.

We generated around 10 million datapoints for each specific average power and propagation distance. The dataset was then divided into training and test sets, with the test set comprising 1% of the full dataset (about 100,000 points).

In the dataset used for training, the input consists of values of the received points for the x-polarization, for which we aim to predict the nonlinear shift. To account for the neighboring points' influence, we include five values of received points to the right and left of the central point (neighbors in the sequence of points). Since we utilize two polarizations, we also incorporate the central point for y-polarization (in the same position in the sequence as for x-polarization) and its five neighboring points on both the left and right. As a result, a total of 22 complex points serve as input for the GB algorithm. The output represents the nonlinear shift (a complex number that must be added to accurately shift the point to its original position in the constellation) for the received point in the x-polarization.

The gradient boosting algorithm was implemented using the XGBoost library[200] with GPU acceleration. The training process involved a large number of estimators, specifically 200,000, which were chosen to test the idea of using decision trees but can be significantly reduced for a particular system. The maximum tree depth did not exceed 20, the learning rate was set to 0.1, and the L_2 and L_1 regularization parameters were both set to 1.0.

To explore why we limited our analysis to only five neighboring symbols, despite the potential for dispersion broadening to impact a significantly larger number of symbols (thus, with nonlinear effects, the effective interaction could extend to dozens or hundreds of symbols), let us provide some context to support this decision.

Machine learning models, with their capacity to serve as universal approximators, can extract valuable information from input data for a specific purpose. This introduction sets the stage for our specific scenario: the presence of neighboring points, which are themselves influenced by their neighbors. This implies that the nonlinear adjustment of neighbors, even

with knowledge only of a neighbor's position at the receiver, can reveal information about subsequent neighbors and the influence on the target point for which we are calculating a correction. This forms a complex network where information about a local group of points contains more insights than initially considered. For instance, the outermost points in our analysis (the fifth neighbors to the left and right) indirectly incorporate information about subsequent points in the transmitted sequence. This information is partial but suggests that each point's shift due to neighboring influences carries implicit data. Although expanding the number of neighbor symbols for analysis could potentially enhance model performance, such an exploration exceeds the scope of this research. However, we emphasize that this is a compelling and valuable area for future investigation and would be an excellent subject for a detailed study.

Now, let's turn our attention to a more quantitative analysis. We will examine how features (the neighboring points) are utilized in a Gradient Boosting model by employing feature importance measurement.

Feature importance in gradient boosting models provides insight into the relative significance of each feature in making predictions. The importance of a feature is often quantified based on how frequently it is used to split the data across all trees in the ensemble. This metric, referred to as the "importance" or "weight," reflects the feature's contribution to the model's decision-making process.

Mathematically, we can define the importance of a feature as follows:

$$I(f_i) = \frac{\sum_{k=1}^{K} \text{Count}(f_i, T_k)}{\sum_{j=1}^{N} \sum_{k=1}^{K} \text{Count}(f_j, T_k)}, \tag{6.13}$$

where N represents the number of features, K the number of trees (estimators) in the model, and $\text{Count}(f_i, T_k)$ is the number of times feature f_i is used as a split in tree T_k.

This definition implies that the importance of a feature is directly proportional to how often it is selected for splitting data points in the model's trees. A higher count indicates a higher significance of the feature in the model's predictions. The essence of this metric lies in its ability to highlight which features the model relies on most heavily to reduce uncertainty or entropy in the data.

It is essential to note that while this measure provides valuable insights into feature relevance, it does not convey the direction of the feature's effect (positive or negative) on the target variable. In the broader context of model interpretation and feature selection, understanding the importance of each feature helps in identifying the variables that contribute most significantly to the predictive power of the model. This knowledge can be crucial for model refinement, simplification, and for gaining deeper insights into the underlying processes being modeled.

Figure 6.3 presents the feature importance in a GB model trained to predict nonlinear shifts at an average signal power of 6 dBm for different transmission distances: (a) 800 km, (b) 1200

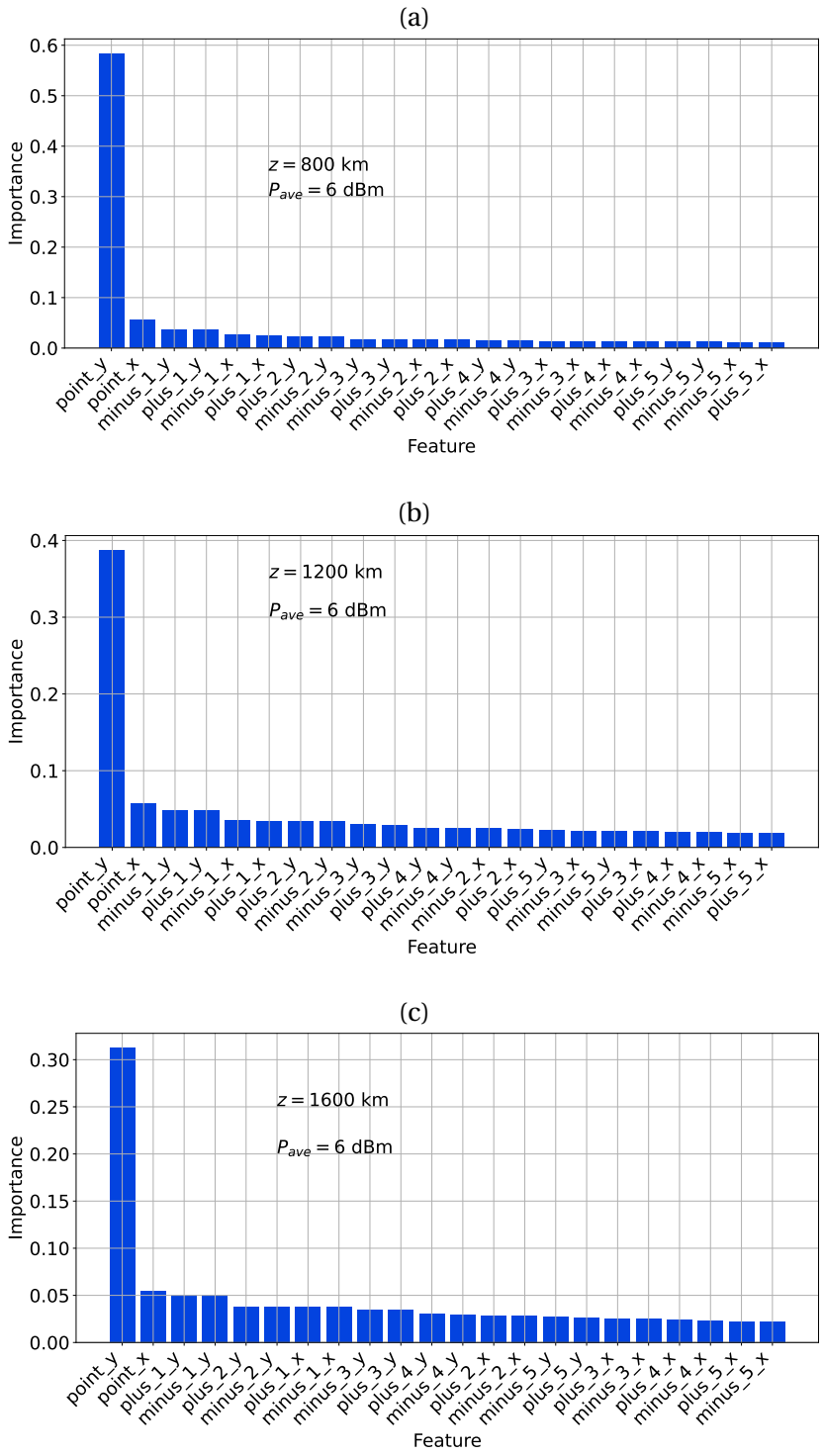


Figure 6.3: Feature importance in the trained GB model for predicting nonlinear shifts at an average signal power of 6 dBm. **(a)** For a transmission distance of 800 km, **(b)** 1200 km, and **(c)** 1600 km. In the feature labels, 'x' and 'y' denote the polarization of the signal point, whereas 'plus' and 'minus' indicate the right and left neighbors in the symbol sequence, respectively.

km, and (c) 1600 km. In the feature labels, 'x' and 'y' represent the polarization of the signal point, while 'plus' and 'minus' signify the right and left neighbors in the symbol sequence, respectively. An intriguing observation is that the most influential feature in the decision trees is not the point for which we predict the correction ('point_x'), but its counterpart in the other polarization ('point_y'). This finding highlights the significant insight that information from the same position but in a different polarization provides about the required correction.

Further analysis reveals that 'point_y' dominates feature usage, significantly more than 'point_x', especially over longer distances (for 1600 km, 'point_y' accounts for over 30% of usage, whereas 'point_x' is just above 5%). The importance of the immediate neighbors ('plus_1_y' and 'minus_1_y') is comparable to 'point_x', but the relevance of further neighbors declines with their sequence number. This decline is more pronounced for shorter distances (800 km) compared to longer ones (1200 and 1600 km), aligning with the expectation that neighbor influence increases with distance. For instance, at 1600 km, the influence of the furthest neighbors analyzed is around 2% each, dropping below 1% for 1200 km, and to approximately 0.6% for 800 km.

From these observations, we conclude that the impact of distant neighbors grows with increasing transmission distance, though not dramatically. Including more symbols could enhance model performance, but the improvement is minimal for short distances and potentially up to 8% for longer distances under optimal conditions. This increase, while notable, does not fundamentally alter the model's overall performance. At a lower average signal power of 3 dBm, where nonlinear effects are reduced, the impact of distant neighbors is even less significant, reinforcing the dominant role of 'point_y' (Fig. 6.4).

This analysis supports our methodological choice of focusing on five neighboring symbols, demonstrating its viability even when the actual channel memory extends significantly further. Our aim was to establish a proof of concept for using the GB model without delving into a comprehensive study of neighbor influence, highlighting this as a promising direction for future research.

6.1.5 Complexity

Constructing

The computational complexity of constructing a decision tree depends on the number of records (N) and the number of features (M). In a balanced tree scenario, the complexity is $O(MN\log N)$, accounting for evaluating all features across all records at each level of the tree down to a depth of $\log N$. In the worst-case, with an unbalanced tree structure, the complexity can escalate to $O(MN^2)$. This is due to the potential for the tree to degenerate into a structure with depth N, requiring extensive computations at each level. Introducing a maximum depth limit (D_{max}) as a stopping criterion alters the complexity. Regardless of being balanced or unbalanced, the tree cannot exceed this depth, making the construction

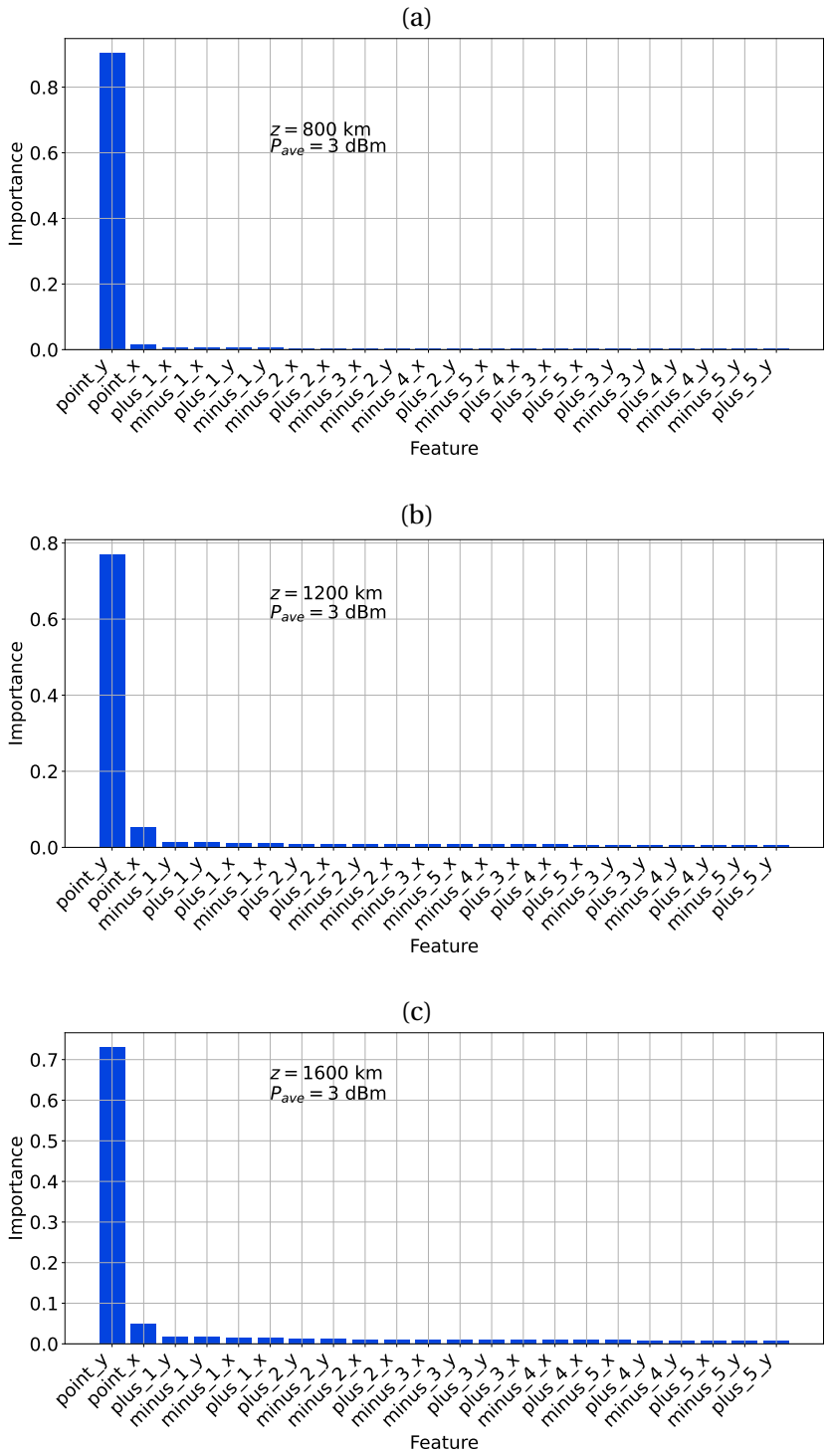


Figure 6.4: Feature importance in the trained GB model for predicting nonlinear shifts at an average signal power of 3 dBm. **(a)** For a transmission distance of 800 km, **(b)** 1200 km, and **(c)** 1600 km. In the feature labels, 'x' and 'y' denote the polarization of the signal point, whereas 'plus' and 'minus' indicate the right and left neighbors in the symbol sequence, respectively.

complexity $O(MND_{max})$. This constraint ensures that the tree's growth is curtailed, preventing the exponential increase in computational cost seen in unbounded scenarios.

The computational complexity of Decision Trees varies significantly between the ideal, balanced tree scenario and the worst-case, unbalanced tree. The introduction of a maximum tree depth as a stopping criterion significantly impacts the computational complexity of Decision Trees, both in construction and prediction phases. It provides a practical mechanism to control the growth of the tree, thereby ensuring that the complexity remains manageable, especially in scenarios that could otherwise lead to highly unbalanced structures.

In the context of Gradient Boosting, the construction of the ensemble model involves sequentially building T decision trees. Consequently, the computational complexity for training the ensemble is influenced by the complexities inherent in constructing individual decision trees, scaled by the number of trees, T. In scenarios where decision trees are balanced, the training complexity is $O(TMN\log N)$. Conversely, in cases where the trees become unbalanced, the computational burden escalates, with complexity potentially reaching $O(TMN^2)$. This increase is attributable to the deeper tree structures that may arise, requiring more extensive computation at each level. To mitigate such variability and enhance predictability in computational demands, imposing a maximum depth, D, for each tree normalizes the complexity to $O(TMND)$.

Making Predictions

For predictions, a balanced tree allows for traversing from the root to a leaf node in $O(\log N)$ computations, given the tree's depth corresponds to $\log N$ in balanced conditions. In an unbalanced tree, the depth could approach N, leading to a worst-case prediction complexity of $O(N)$ for traversing through all nodes from root to a specific leaf. When a maximum depth is enforced, the complexity of making a prediction becomes $O(D_{max})$. This limit ensures that the prediction time remains constant, relative to the depth constraint, regardless of the tree's balance.

Predictive complexity in Gradient Boosting models similarly hinges on the depth of the individual decision trees. Under regular conditions, where trees maintain a balanced structure, the complexity of making predictions for a single instance is $O(T\log N)$, benefiting from the logarithmic depth of balanced trees. However, in the worst-case scenario, where trees may be significantly unbalanced, the complexity could deteriorate to $O(TN)$. To counteract this and secure a uniform prediction time, enforcing a maximum tree depth, D, aligns the predictive complexity to $O(TD)$.

The computational complexity for predictions made by the Gradient Boosting algorithm can be succinctly summarized by the following upper limit:

$$C(\text{GB}) = T \cdot D, \tag{6.14}$$

where T represents the number of trees within the Gradient Boosting model, and D signifies the maximum depth encountered across all these trees. It's important to note that in practical applications, D does not typically surpass a threshold (often around 100). This constraint helps in managing the computational load during the prediction phase, ensuring that the time complexity remains within a feasible range, even as the model scales.

This formulation emphasizes the linear relationship between the algorithm's computational complexity and both the number of trees and their depth. Such an understanding is crucial for optimizing the performance and efficiency of Gradient Boosting models in diverse applications. At first glance, it may appear that the overall complexity for our chosen set of parameters is excessively high. In this study, our primary goal is not to achieve cutting-edge complexity but rather to demonstrate the proof-of-concept. However, it is important to emphasize that each parameter contributing to complexity can be optimized to significantly reduce the overall complexity. Comprehensive information about feature importance allows us to manage input parameters (which can be further reduced) and to control the impact of each decision tree, thereby enabling the removal of unnecessary trees to enhance performance. Moreover, we can manually regulate the maximum depth of each tree, which, after optimization, can ultimately yield the desired performance.

6.1.6 Results

To demonstrate the potential of GB, we assessed the performance of various GB models across a range of propagation distances and diverse input signal average powers. The results were analyzed in terms of Q-factor improvement and the level of HD-FEC (Hard Decision Forward Error Correction), which represents a common error correction scheme used in optical communication systems. The target Bit Error Rate was set at 4%.

For all propagation distances, the GB models showed at least a 1 dB improvement in Q-factor compared to the system with Chromatic Dispersion Compensation (CDC). Specifically, for a 1600 km transmission distance (Fig. 6.5c), the model trained on 8 dBm signals exhibited a 1.10 dB improvement in Q-factor for the same test power. The model trained on 7 dBm signals displayed a 1.19 dB improvement for the same test power, while the model trained on 6 dBm signals showed a 1.22 dB improvement. The 1.22 dB Q-factor improvement for the 6 dBm model allowed us to decrease the BER (increase the Q-factor) below (above) the HD-FEC level.

Interestingly, models trained on higher average power (and thus experiencing higher nonlinear distortions) exhibited good performance when applied to lower test power levels. However, their performance degraded for higher test power levels. This observation suggests that the GB algorithm can successfully mitigate nonlinear effects. This is particularly evident in Fig. 6.5a for a propagation distance of 800 km. The model trained on 8 dBm outperformed other models for power levels from 5 dBm to 8 dBm, showing approximately 1.6 dB improvement (specifically, 1.64 dB for 6 dBm test power and 1.61 dB for 8 dBm test power). For 8 dBm, the improved Q-factor surpassed the HD-FEC level. However, for lower signal powers, the model

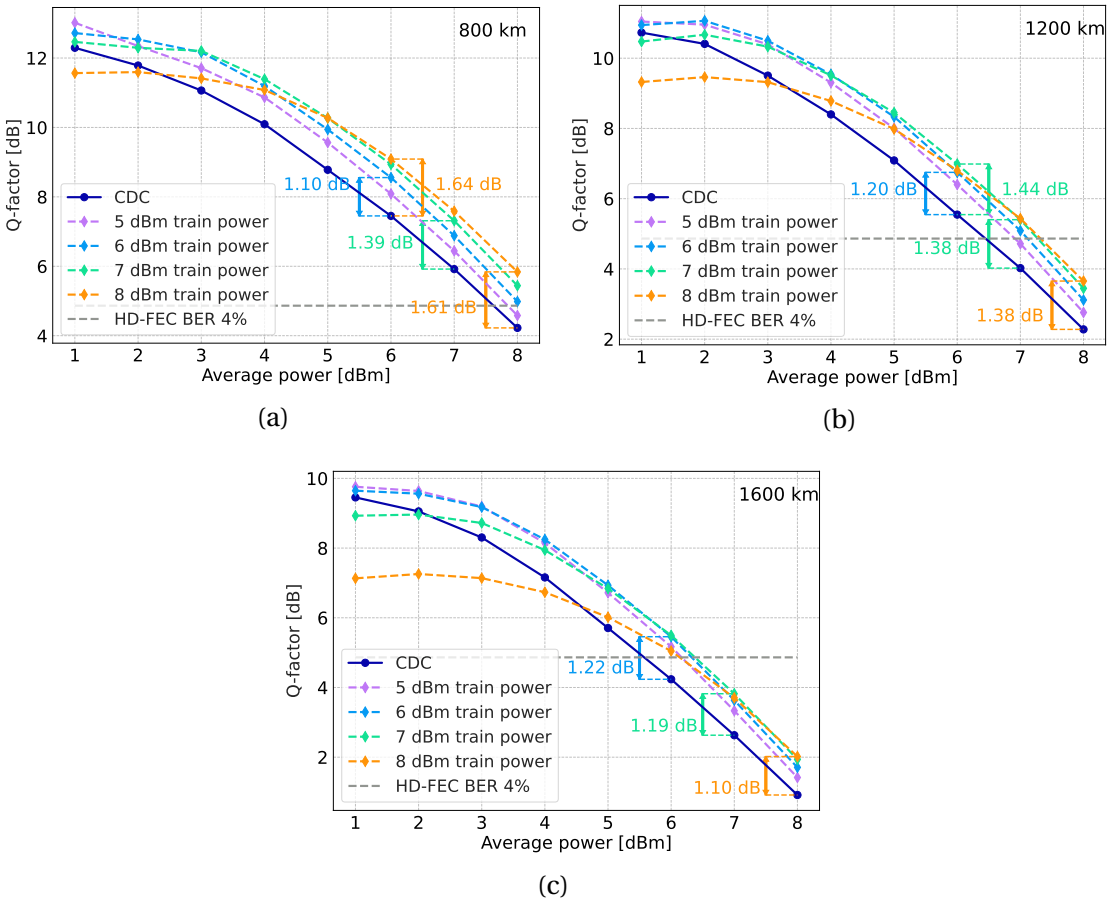


Figure 6.5: Q-Factor performance after nonlinear equalization using GB trained at varying average signal power levels. Solid lines represent the system with CDC, and horizontal dashed lines indicate the HD-FEC threshold for a BER of 4%. Total propagation distance of 800 km **(a)**, 1200 km **(b)** and 1600 km **(c)**.

trained on 8 dBm performed poorly, since these regimes are less influenced by nonlinear effects.

A similar pattern can be observed in Fig. 6.5b for a propagation distance of 1200 km. In this case, the model trained on 7 dBm showed significant improvement for a wide range of test signal powers (from 3 dBm to 8 dBm). This finding indicates that the GB training power can be optimized for specific power ranges in which the system operates. The wide range of applicability allows for the development of a universal equalizer based on GB, which does not require retraining for different power levels (subject to the constraints of the working regime for GB).

6.1.7 Discussion

Perspective of Practical Realization

To address the practical aspects of signal equalization, it is crucial to consider the computational efficiency of the equalization process in real-world implementations. Neural networks (NNs), for instance, comprise multiple layers involving matrix multiplications, typically resulting in computational complexity on the order of $O(N^3)$ for straightforward implementations. While optimizations such as Strassen's algorithm can reduce this complexity, the inherent structure of neural networks, often comprising several layers, significantly increases the operation count, scaling almost cubically with added complexity. This escalation can range from thousands of operations for basic Multi-Layer Perceptron (MLP) architectures to millions for more sophisticated models, which offer improved performance but at a higher computational cost.

The sequential nature of these operations, a characteristic shared with Digital back-propagation (DBP) algorithms employing the Split-step Fourier method (SSFM), necessitates waiting for the completion of previous operations before proceeding. This requirement for shared memory, where the results of prior computations must be immediately accessible for subsequent operations, introduces significant challenges for real-time processing.

The Gradient Boosting approach offers a noteworthy advancement in this context. Once the estimators, or trees, are constructed, they can be utilized independently. This independence allows for the parallel computation of results from each tree without interdependencies, significantly enhancing computational efficiency. Furthermore, the simplicity of GB operations—requiring only D (the depth of the tree) comparisons for input features to derive results from each decision tree—eliminates the need for complex computations. Given that comparison operations are as fast as summations in contemporary systems, the process is highly efficient. Importantly, the depth of these trees, D, is inherently manageable, often not exceeding 20 in practical analyses.

With only 20 basic operations, we can process all trees in a Gradient Boosting model. This efficiency suggests the potential for high-speed performance through parallel processing. An intriguing research direction is exploring how to manage 200,000 estimators, the current count in our system, in parallel within real-world platforms like FPGAs. Furthermore, there's room to optimize and potentially reduce this number of estimators, which hints at the depth of investigation required — a fitting challenge for a dedicated Ph.D. project. In this thesis, my aim was to establish a proof-of-concept for the GB algorithm, demonstrating its viability.

Conclusion

In conclusion, this study highlights the potential of gradient boosting as an innovative equalization technique in optical communication systems. This approach not only offers improved predictive performance but also holds the promise of simplifying the complexity inherent

in traditional methods. Such advancements could significantly enhance the efficiency of contemporary telecommunication networks.

Our initial findings are encouraging, showing that models based on gradient boosting can achieve a notable improvement in system performance, specifically an increase of 1.1 - 1.6 dB in the Q-factor compared to systems that only implement Chromatic dispersion compensation (CDC). This underscores both the practical and theoretical value of the gradient boosting technique in advancing optical communication technologies.

Looking forward, future research should aim at fine-tuning the balance between complexity and performance, ensuring robustness in real-world applications, and examining the prospects of hardware acceleration to bring this technique to practical use. The development of a versatile equalizer using gradient boosting could broaden the scope of its application in optical communications, promising substantial improvements in the field.

6.2 Neural Networks

In this part of our research, we aim to broaden our methods by demonstrating how neural networks (NNs) can be applied across various power levels and fiber distances. These findings represent a potential research direction and offer a glimpse into the practical use of NNs in optical communications. It's important to highlight that these results are basic examples of NN applications.

For our simulations, we used a standard single-mode fiber (SSFM) model with erbium-doped fiber amplifiers (EDFAs). We transmitted signals using 16-quadrature amplitude modulation (16-QAM) WDM with dual polarization at a rate of 34 GBd. The signals were shaped using a digital root-raised cosine (RRC) filter with a roll-off factor of 0.1. We tested transmission distances of 12 and 15 spans, each span being 80 km long (totaling 960 km and 1200 km, respectively). The noise figure for the EDFA was set to 4.5 dB. The range for the average signal power was from -2 to 7 dBm. To model the signal's propagation through the fiber, we used the NLSE (1.1) and applied a GPU-accelerated split-step Fourier method[183]. The fiber's characteristics included a wavelength of $\lambda = 1550$ nm, a dispersion coefficient of $D = 16.8\,\mathrm{ps}/(\mathrm{nm}\cdot\mathrm{km})$, and a nonlinear coefficient of $\gamma = 1.2\,\mathrm{W}^{-1}\cdot\mathrm{km}^{-1}$.

We created a large dataset with over four million points (2^{22}) for each combination of average power and propagation distance. This dataset was then split, setting aside 10% (exactly 419,431 points) to test the neural network's performance.

The neural network is structured as a linear sequence of layers, beginning with a Conv1D layer that performs a convolution operation with 256 filters of kernel size 20 to detect patterns across the input sequence, utilizing the "relu" activation function. This is followed by a Bidirectional LSTM layer with 256 units that processes the data in both forward and reverse directions, capturing dependencies in the sequence data, and is set to return sequences for full temporal

resolution. After the LSTM layer, the network employs a Flatten layer to transform the 3D output to a 2D tensor, suitable for the final layer. The architecture concludes with a Dense layer that has 2 units. This sequential arrangement allows the model to extract spatial features through convolution, learn temporal dynamics with LSTM, and finally make predictions.

```
from keras.models import Sequential
from keras.layers import Conv1D, Bidirectional, LSTM, Flatten, Dense

# Define the model
model = Sequential([
    Conv1D(filters=256, kernel_size=20, activation='relu', input_shape
  =(42, 1)),
    Bidirectional(LSTM(256, return_sequences=True)),
    Flatten(),
    Dense(2, activation='linear')
])
```

Listing 6.1: Default WDM parameters

The neural network's (NN) input layer takes in 42 real numbers. These numbers represent the central data point in a sequence and its 20 neighboring points, with 10 on each side: $(b_{k-10},\ldots,b_{k-1},b_k,b_{k+1},\ldots,b_{k+10})$. The NN's output is two real numbers that combine into one complex number. This complex number is the estimated correction, or equalization, for the received point b_k in the complex plane. Ideally, this correction η_k should be equal to the difference between the transmitted and received points, meaning $\eta_k = c_k - b_k$. Therefore, when we add η_k to b_k, we should get the original transmitted symbol c_k, where c_k is the sent symbol (a point in the signal constellation), and b_k is the point that was actually received.

The NN was trained using the Mean Squared Error (MSE) as the loss function, which measures the average squared difference between the estimated values and what is estimated. The optimization of the network was performed using the Adam method, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments with a learning rate of 10^{-4}. The training process ran for 200 epochs on a Tesla V100 GPU.

In Figure 6.6, we show how the Q-factor, which measures signal quality, changes with different average signal powers for neural networks (NNs) trained at various power levels. The analysis begins with a signal power of 1 dBm (top left) and increases to 6 dBm (bottom right), over a distance of 960 km. Figure 6.7 presents similar data for a distance of 1200 km. In these figures, the blue line shows the Q-factor after applying Chromatic dispersion compensation (CDC) and Nonlinear Phase Equalisation (NPE) followed by a hard decision process. The red line indicates the Q-factor when the NN is tested at the same power level it was trained on. The green lines show how well the NN, trained at a specific power level, performs when tested at different power levels. The dashed line marks the 4% Bit Error Rate (BER) threshold, which is the upper limit for effective Forward error correction.

NNs trained on datasets at -2, -1, 0, and 7 dBm are not included in these graphs because

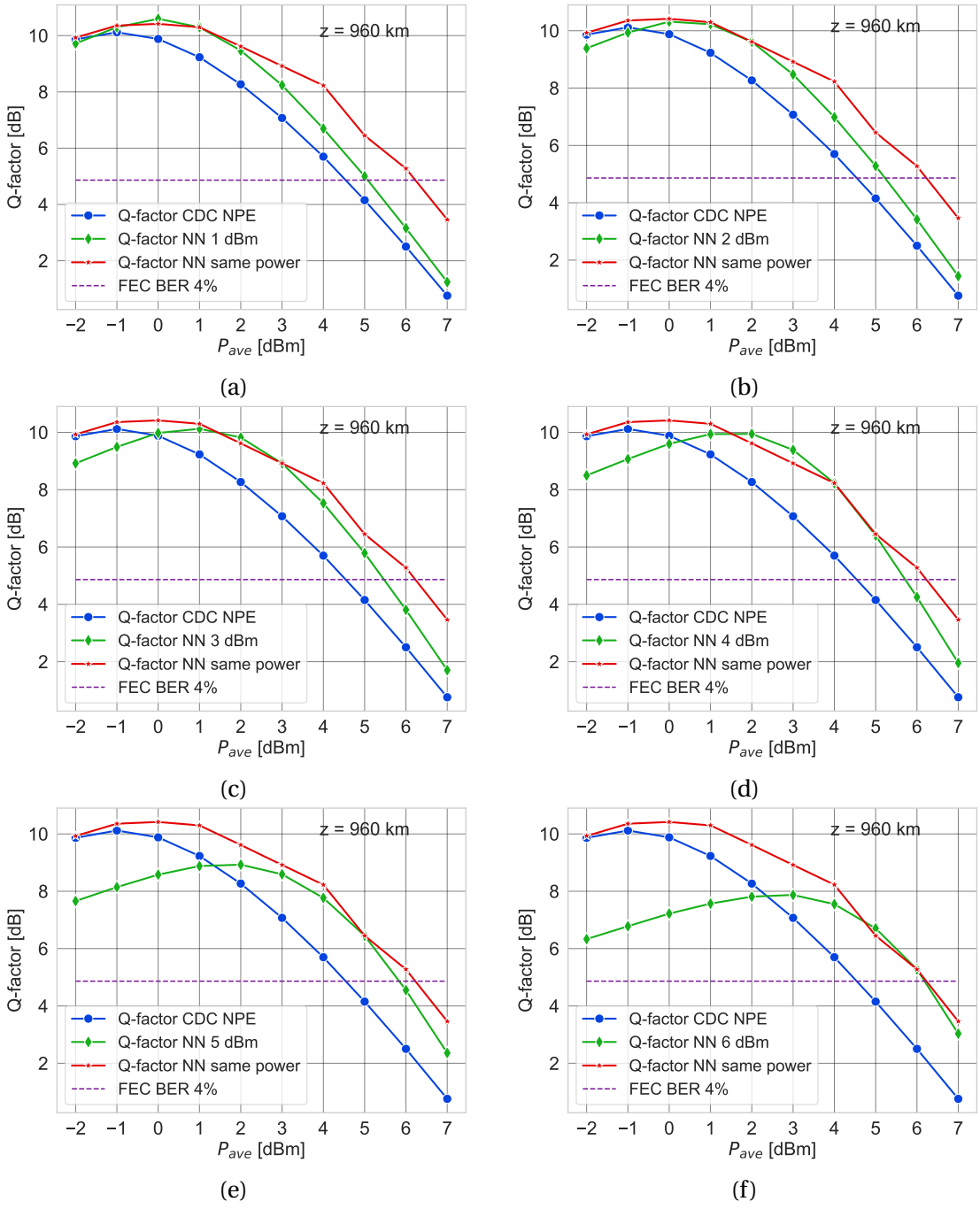


Figure 6.6: Representation of dependance of Q-factor vs signal average power for different NNs trained on different signal power. It starts with 1 dBm (upper left) and subsequently increment up to 6 dBm (lower right). Propagation distance is 960 km.

their performance did not exceed that of NNs trained at the same power levels (red line). Furthermore, the NN trained at 7 dBm did not perform well across the power range, except precisely at the trained power level.

Figure 6.6 illustrates that neural networks (NNs) can markedly enhance signal quality post-equalization. In various conditions, NNs have achieved more than a 2 dB improvement in performance, as seen in Fig. 6.6d for 4 dBm signals. Impressively, they can even push system performance beyond the forward error correction (FEC) threshold, as depicted for 5 and 6 dBm in Fig. 6.6f.

An additional noteworthy finding is the versatility of NNs across different power levels. For instance, an NN trained on a 3 dBm signal can effectively improve performance for 1 and 2 dBm test power levels, as shown in Fig. 6.6c. Remarkably, an NN trained on 4 dBm outperforms an NN trained at the same power for test powers of 2 and 3 dBm, as observed in Fig. 6.6d. This suggests that NNs can generalize the impact of nonlinearity and accurately predict the necessary corrections η_k. The significance of this is that NNs are not confined to a single power level and can be adapted to various system types and operating conditions without a new training phase.

A future research direction is to implement transfer learning, which is a method where a model developed for one task is reused as the starting point for a model on a second task. This approach could broaden the range of scenarios for NN training, enhancing their adaptability and efficiency.

In Figure 6.7, we see a pattern similar to the one in Figure 6.6. Neural networks (NNs) effectively perform nonlinear equalization, enhancing system performance. For all power levels examined, NNs managed to raise the Q-factor to levels suitable for forward error correction (FEC), as shown by surpassing the upper dashed line in Fig. 6.7. However, a noticeable difference is that NNs specialized for a certain power level generally do not exceed the performance of NNs trained and tested at the same power level. This implies that with the added complexity from three extra spans of fiber, our current method needs to be improved to create NNs that are versatile across various power conditions (although Fig. 6.7e shows promising results where the green line for 3 and 4 dBm nearly matches the red line).

To conclude, the results presented here are an example of how NNs can be leveraged in future research. They demonstrate that a data-driven approach can be highly effective when sufficient data is available. Thanks to the HpCom simulator, data generation for different types of fibers and power levels is now quicker and easier. These findings will contribute to an upcoming review article on the application of machine learning in nonlinear equalization, which is currently in progress and will extend beyond the scope of this thesis.

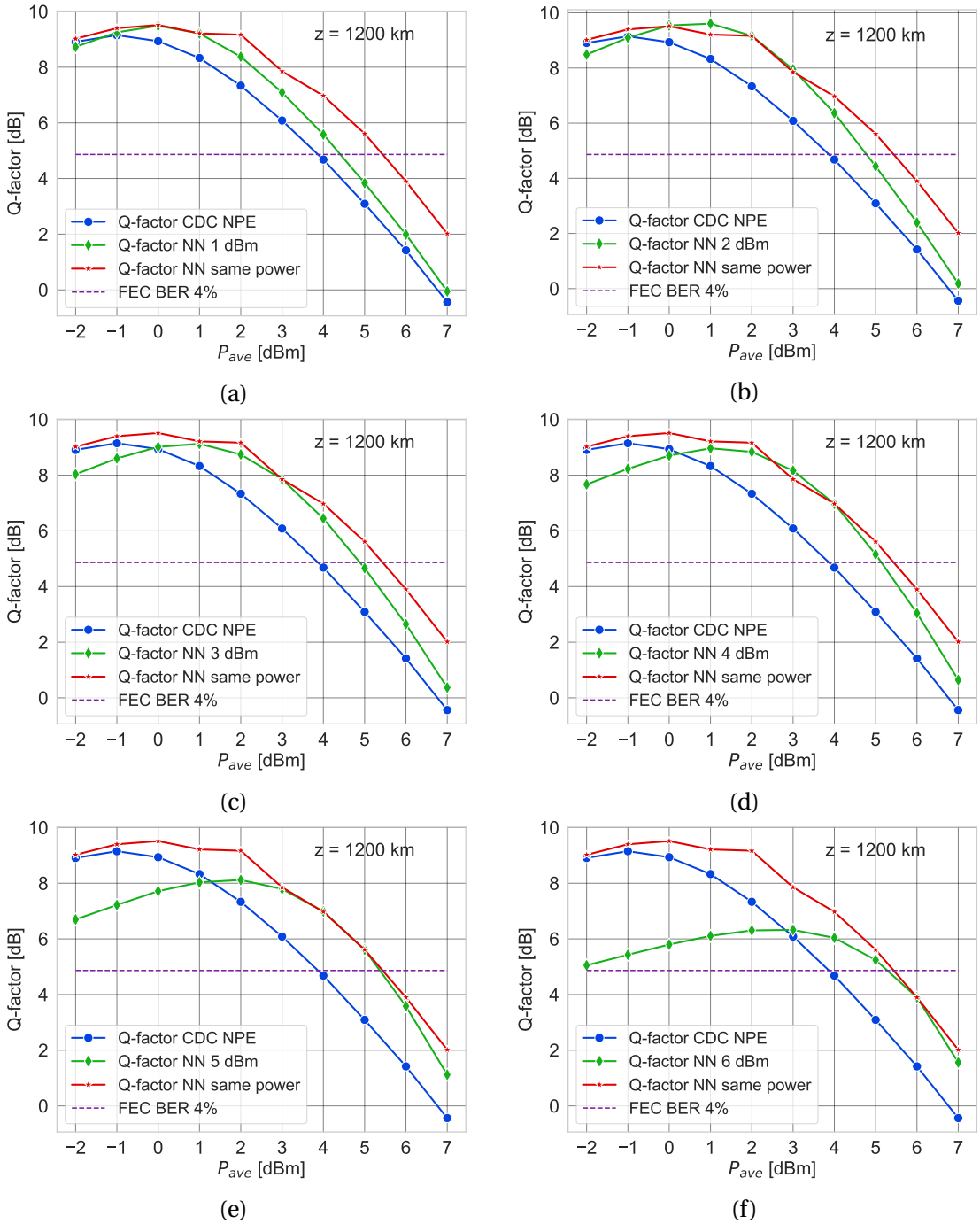


Figure 6.7: Representation of dependance of Q-factor vs signal average power for different NNs trained on different signal power. It starts with 1 dBm (upper left) and subsequently increment up to 6 dBm (lower right). Propagation distance is 1200 km.

7 Data Structures Shaped by Nonlinearity

7.1 Introduction

Optical communication channels often exhibit nonlinear effects that can significantly impact the integrity and recoverability of transmitted signals. Understanding of these effects is essential for designing robust communication systems capable of maintaining high levels of performance. With a deterministic model such as the NLSE that closely corresponds with real systems, we can study this behavior in depth. If we consider that we have precise knowledge of the final distribution of constellation points at the receiver for any given input at the transmitter, we can leverage this information to our advantage. As new symbols arrive at the receiver, we might initially be uncertain about their identity. However, with comprehensive knowledge about the ultimate distribution of all constellation points, we can ascertain how likely it is that a new point corresponds to a specific symbol. This approach allows us to accurately map each new point to the correct constellation symbol, enhancing the accuracy of the signal interpretation.

In pursuit of a deeper understanding of these effects, we turn to computational simulations. Specifically, we use a GPU-based simulation package, HpCom [183], which allows us to generate a large amount of data. By analyzing the internal structure of the received constellation points, we aim to demonstrate that the nonlinear effects induce distinctive patterns in the data. These patterns are not merely noise; they hold valuable information about the behavior of the signal within the fiber. Through detailed analysis, we can begin to reveal these patterns and improve our ability to predict and mitigate the impact of nonlinearity on our communications, pushing the boundaries of what's possible in optical data transmission.

In this study of optical communication systems, we initially ignore Amplified spontaneous emission (ASE) noise to focus on how signals change due to nonlinear effects alone. This choice is not by mistake but a planned decision to look closely at how nonlinear effects work together without noise interference. By considering signal changes as a predictable process due to nonlinear effects, we can better understand these effects without the randomness of noise getting in the way.

We know this approach has its downsides. It's important to point out that leaving out noise is a first step in our analysis. Our goal is to clearly see how pure nonlinear effects shape the signals we receive. This makes our analysis simpler and helps us get to the basics of how nonlinear effects influence signal changes. This basic understanding is crucial for coming up with new ideas and algorithms to handle these nonlinearities better. Yet, we accept that this simplification is just the beginning. The way noise and nonlinear effects interact is important for a complete picture of optical systems, but this thesis does not cover that.

To sum up, this thesis looks at the predictable nonlinear effects in optical communication systems. Future studies will build on this work to also include how noise affects these systems.

7.2 Methodology

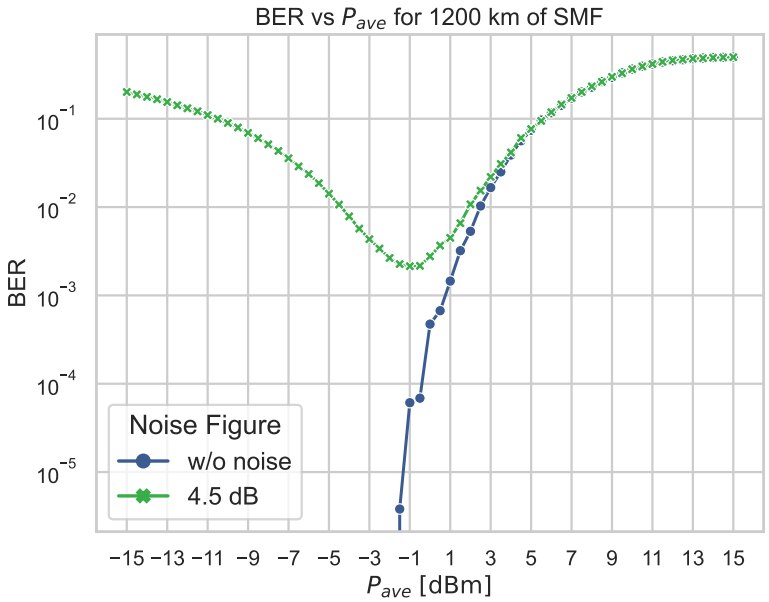


Figure 7.1: BER vs P_{ave} [dBm] for studied system. The green line marked with crosses represents the reference system, including the impact of additional EDFA noise characterized by a 4.5 dB Noise Figure. The blue line illustrates the ideal case, free from ASE noise, which is the focus of our study.

This study focuses on the analysis of a 16-QAM WDM signal, specifically investigating the behavior of received constellation points post-CDC and NPE. We employed an optical channel model of Standard single-mode fiber (SSMF) with Erbium-Doped Fiber Amplifiers (EDFAs). The signal format under consideration is a 16-QAM WDM with single polarization and a symbol rate of 34.4 GBd. Pulse shaping was realized using a digital Root Raised Cosine (RRC) filter with a roll-off factor of 0.1. The total transmission distance spanned 15 links of 80 km each. The EDFA was assumed to be ideal, introducing no noise. The average signal power varied from −2 to 8 dBm. Signal propagation through the fiber was modeled by the nonlinear

Schrödinger equation (NLSE), which was solved using the GPU-accelerated split-step Fourier method[183]. The fiber's parameters were set to a wavelength of $\lambda = 1550$ nm, a dispersion coefficient of $D = 16.8$ ps/nm·km, and a nonlinear coefficient of $\gamma = 1.2\ \mathrm{W}^{-1}\cdot\mathrm{km}^{-1}$. For the study, we generated 2^{24} data points for each specified average power level.

Figure 7.1 illustrates the dependence of the BER on the average signal power for the system parameters described previously. It depicts two scenarios: one with additional noise from EDFA and one without. The power range under investigation is selected to align with the optimal levels for real-world transmission systems. As indicated by the graph (represented by the green line), the BER reaches its minimum around −1 dBm. It is important to note that, for the purposes of our study, we utilize data from the channel without EDFA noise. This approach allows us to specifically examine the impact of nonlinearity without the confounding effects of random noise.

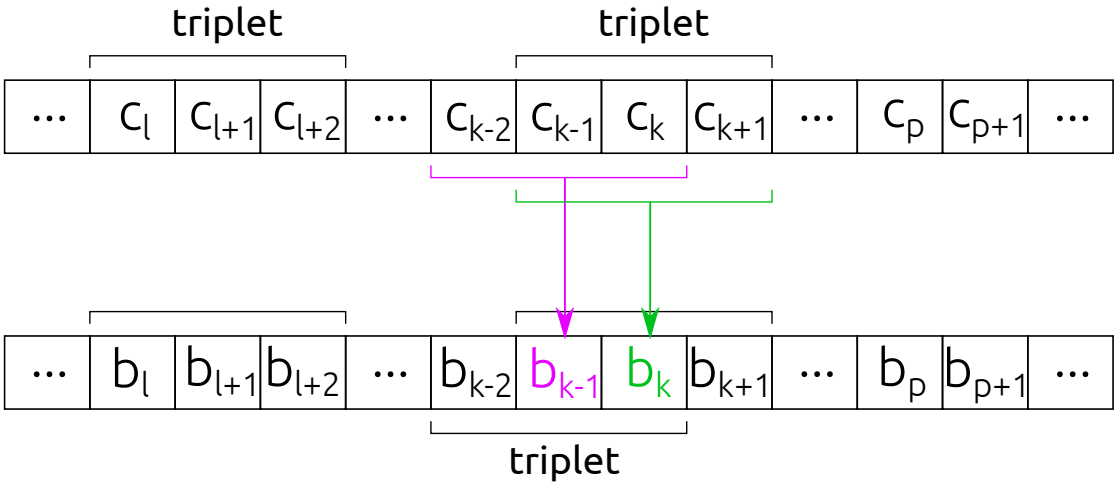


Figure 7.2: Schematic representation of the "triplet" concept, illustrating the extraction of data b_k corresponding to each transmitted triplet (c_{k-1}, c_k, c_{k+1}).

In this work, we introduce the concept of a "triplet", which is defined as a set of three consecutive points consisting of a left, central, and right point (Fig. 7.2). On the transmitter side, there is a sequence of transmitted symbols. Unlike real systems where the sequence is continuous and lacks a definitive beginning or end, our simulation employs a cyclic version: $\{c_0, c_1, \ldots, c_{k-1}, c_k, c_{k+1}, \ldots, c_{K-1}, c_K\}$, where K represents the total number of transmitted symbols. In the continuum of transmitted constellation symbols c_k, a triplet is any set of three points (c_{k-1}, c_k, c_{k+1}) for the transmitted sequence. Similarly, for received symbols b_k, we define a triplet as (b_{k-1}, b_k, b_{k+1}). Our focus is on the distributions of the received symbol b_k for a specific transmitted triplet (c_{k-1}, c_k, c_{k+1}). In essence, we aim to categorize our dataset and analyze the distributions for b_k based on the transmitted symbol and its adjacent neighbors.

In our simulation setup, we maintain a fixed average signal power for the transmitted signal (we conduct separate analyses for each signal power level, ranging from −2 dBm up to 8 dBm).

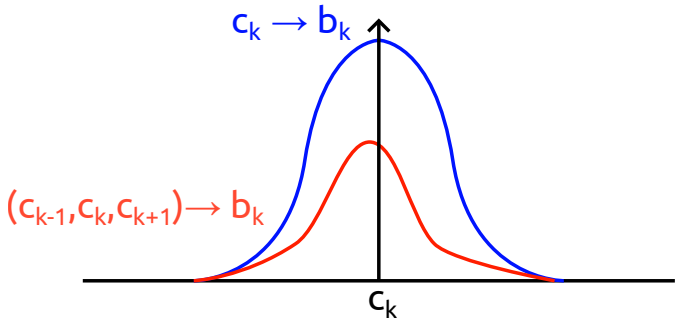


Figure 7.3: Schematic illustration of received symbol distributions based on transmitted triplets. The blue line represents the distribution of received symbols b_k originating from the transmission of a single symbol c_k. The red line shows the modified distribution of b_k when considering the entire transmitted triplet (c_{k-1}, c_k, c_{k+1}).

This setup provides us with complete information, including the sequence of transmitted symbols $\{c_k\}$ and the sequence of received symbols $\{b_k\}$. It is important to note that for this analysis, the received symbols $\{b_k\}$ have already been compensated for dispersion and phase shift, allowing us to concentrate solely on the deviations in the complex constellation plane caused by nonlinear interactions.

To analyze the impact of these nonlinear interactions, we examine the distributions of the received symbols $\{b_k\}$ based on their originating "triplet" (c_{k-1}, c_k, c_{k+1}). This process involves iteratively selecting indices k corresponding to each specific triplet configuration (for example, (c_{k-1}, c_k, c_{k+1}) is $(1+1\mathrm{j}, 1+3\mathrm{j}, -3-1\mathrm{j})$ or any other triplet), and gathering all corresponding b_k. For a 16-QAM system, there exist $16 \times 16 \times 16 = 2^{12}$ potential triplet combinations. With 2^{24} data points in our dataset, this results in approximately 2^{12} data points per triplet, available for further analysis. Through this method, we create distributions of received symbols $D(b_k)$ for each of the 2^{12} possible triplet combinations, thereby associating each distribution $D(b_k)$ with its respective sent triplet (c_{k-1}, c_k, c_{k+1}).

To simplify the analysis, we normalize the distributions of b_k for each triplet by subtracting the original transmission point c_k, shifting all distribution centers towards the origin $(0,0)$. This adjustment aims to reduce the influence of the original transmission point's location on the analysis, allowing us to more clearly observe the shifts in distribution centers caused by nonlinear effects, which will be further explored in subsequent analysis steps.

Figure 7.3 provides a schematic representation of this process: transmitting c_k results in a distribution of received symbols b_k (depicted by the blue line). When we limit the analysis to received symbols b_k that correspond not only to the transmitted symbol c_k but to the sent triplet (c_{k-1}, c_k, c_{k+1}), we observe a modified distribution (indicated by the red line).

The primary model utilized for distribution analysis is the GMM, which will facilitate the understanding of the underlying structure within the received signal constellations.

7.2.1 Gaussian Mixture Model and Likelihood

A GMM is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. The formula for a GMM can be written as:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{7.1}$$

where $\mathbf{x}$ is a data point, K is the number of Gaussian components, π_k is the mixing coefficient for the kth Gaussian component (with the constraint $\sum_{k=1}^{K} \pi_k = 1$), $\boldsymbol{\mu}_k$ is the mean vector of the kth Gaussian component, $\boldsymbol{\Sigma}_k$ is the covariance matrix of the kth Gaussian component and $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the Gaussian distribution.

In this study we will use likelihood function to compare different models for the same data. The model with the higher likelihood (or higher log-likelihood) is generally considered to have a better fit to the data. For a given statistical model and observed data, the likelihood function $L(\theta|x)$ (where θ represents the parameters of the model and x represents the observed data) quantifies how well the model with specific parameter values explains the data.

Let's consider a simple example for likelihood for single one-dimensional Gaussian distribution. The likelihood function L for a Gaussian distribution gives us a measure of how likely it is to observe a given set of data points $(x_1, x_2, \ldots, x_n)$ given the parameters of the Gaussian distribution, specifically the mean μ and variance σ^2. For a set of independent and identically distributed observations from a Gaussian distribution, the likelihood function is:

$$L(x_1, x_2, \ldots, x_n|\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \tag{7.2}$$

The log-likelihood is the natural logarithm of the likelihood function, which turns the product of probabilities into a sum of log probabilities, making it easier to work with especially for computations:

$$\log L(x_1, x_2, \ldots, x_n|\mu, \sigma^2) = \sum_{i=1}^{n} \left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x_i - \mu)^2}{2\sigma^2}\right] \tag{7.3}$$

If we return to two-dimensional case, for GMM (Eq. (7.1)) the log-likelihood is given by:

$$\log p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right), \tag{7.4}$$

where single two-dimensional datapoint $\mathbf{x_n} \in \mathbf{X}$ and k is the number of corresponding Gaussian component.

In our scenario, the two-dimensional data points $\mathbf{x}_n$ correspond to the received symbols $\{b_k\}$ associated with the selected triplet.

7.2.2 Model fitting

As previously discussed, within the dataset, we isolate all points b_k corresponding to any particular transmitted triplet (c_{k-1}, c_k, c_{k+1}). For each triplet in the 16-QAM system, this results in approximately 2^{12} data points. Given the 2^{12} combinations of triplets, we conduct a comprehensive analysis in the following manner. Initially, we assign to each triplet a unique identifier (ID) for ease of representation. Several IDs that will be referenced in this text are listed in Table 7.1: For these triplets, we gather all corresponding points b_k and create a

Table 7.1: Triplet unique identifiers (IDs)

Left point	Central point	Right point	identifier (ID)
$1+3i$	$1+3i$	$3+1i$	552
$-1-1i$	$1+1i$	$-1-1i$	1285
$3+3i$	$3+3i$	$3+3i$	2730
$3-3i$	$3-3i$	$1-1i$	2993

centered distribution $D(b_k - c_k)_{\text{ID}}$ of $b_k - c_k$. Although fitting the distribution is possible without this centering process, shifting the received points to align with the transmitted points simplifies the representation and interpretation of the data. Subsequently, we employ the BRMLtoolbox [203] to fit a GMM distribution to the centered data.

The Expectation-Maximization (EM) algorithm is a powerful iterative method used to find the maximum likelihood estimates of parameters in statistical models, especially when the model depends on unobserved latent variables. In the context of GMMs, the EM algorithm is used to estimate the parameters of the Gaussian distributions (means, variances, and mixture coefficients) that best fit the data. The EM algorithm for GMMs involves two main steps repeated iteratively: the Expectation step (E-step) and the Maximization step (M-step) [204, 205].

E-Step

In the E-step, the algorithm estimates the posterior probabilities that a given data point belongs to each of the Gaussian distributions. This is done using the current estimates of the parameters. These probabilities are known as "responsibilities" and are calculated based on Bayes' theorem.

Given K Gaussian distributions, the responsibility of the k-th distribution for the i-th data point, x_i, is given by:

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

where:

- $\gamma(z_{ik})$ is the responsibility of the k-th Gaussian for x_i,
- π_k is the mixture coefficient for the k-th Gaussian (the prior probability that a randomly chosen data point comes from the k-th Gaussian),
- $\mathcal{N}(x_i|\mu_k, \Sigma_k)$ is the probability density of x_i under the k-th Gaussian distribution with mean μ_k and covariance matrix Σ_k.

M-Step

In the M-step, the algorithm updates the parameters of the Gaussian distributions (means, covariance matrices, and mixture coefficients) based on the responsibilities computed in the E-step. The updates are calculated to maximize the expected log-likelihood of the data.

The updated parameters are calculated as follows:

- Updated mean for the k-th Gaussian:

$$\mu_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) x_i$$

- Updated covariance matrix for the k-th Gaussian:

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik})(x_i - \mu_k^{new})(x_i - \mu_k^{new})^T$$

- Updated mixture coefficient for the k-th Gaussian:

$$\pi_k^{new} = \frac{N_k}{N}$$

where $N_k = \sum_{i=1}^{N} \gamma(z_{ik})$ is the effective number of data points assigned to the k-th Gaussian, and N is the total number of data points.

In the two-dimensional case, each data point x_i is a vector in $\mathbb{R}^2$, and the covariance matrices Σ_k of the Gaussian distributions are 2×2 matrices. The formulas for updating the means, covariance matrices, and mixture coefficients remain the same. However, the computation of the probability density $\mathcal{N}(x_i|\mu_k, \Sigma_k)$ for each data point involves the two-dimensional Gaussian density function, which takes into account the covariance between the two dimensions for each Gaussian component. This model allows for capturing more complex shapes in the data distribution by considering the correlations between dimensions, making GMMs a powerful tool for clustering and density estimation in multivariate data.

The analysis includes fitting GMM models with a number of Gaussian components ranging from one to five. For instance, a GMM with two components yields the following properties from the fitting process:

1. **Log-likelihood.** Log-likelihood of the data for the particular GMM parameters. The log-likelihood is given by Eq. (7.4).
2. **Mixing coefficients.** π_1 and π_2 are the mixing coefficient for the first and second Gaussian components respectively ($\pi_1 + \pi_2 = 1$).
3. **Mean Vectors.** For the first and second Gaussian component:
$$\boldsymbol{\mu}_1 = \begin{pmatrix} \mu_{1,1} \\ \mu_{2,1} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\mu}_2 = \begin{pmatrix} \mu_{1,2} \\ \mu_{2,2} \end{pmatrix}.$$
4. **Covariance Matrices.** For the first and second Gaussian component:
$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} \sigma_{11,1} & \sigma_{12,1} \\ \sigma_{21,1} & \sigma_{22,1} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_2 = \begin{pmatrix} \sigma_{11,2} & \sigma_{12,2} \\ \sigma_{21,2} & \sigma_{22,2} \end{pmatrix}.$$
5. **Eigenvalues.** The eigenvalues of the covariance matrices could be obtained by solving the characteristic equation $\det(\boldsymbol{\Sigma} - \lambda \boldsymbol{I}) = 0$, where $\boldsymbol{I}$ is the identity matrix. For the first and second Gaussian components, the eigenvalues could be represented as follows:
$$\boldsymbol{\lambda}_1 = \begin{pmatrix} \lambda_{1,1} \\ \lambda_{2,1} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\lambda}_2 = \begin{pmatrix} \lambda_{1,2} \\ \lambda_{2,2} \end{pmatrix}.$$

For other numbers of Gaussian components, the data is processed in a similar manner. All fitting results are stored for further analysis and are accessible from my GitHub repository or via email. Detailed information and data sets can be obtained through the following repository link or by contacting me directly[1]. In the subsequent section, specific results and insights gleaned from the data will be presented.

7.3 Results

After the comprehensive data collection and fitting of distributions for all conceivable triplets, we begin our discussion with visual illustrations.

Figures 7.4 and 7.5 depict the distributions of the received constellation points b_k, centered with respect to the transmitted symbol c_k (after CDC and NPE). Essentially, these figures represent the received symbols after subtracting the original transmitted symbol, corresponding to specific "triplets". Figure 7.4 presents the distribution for triplet 552, whereas Figure 7.5

[1] GitHub Repository: https://github.com/esf0. Email: egor.sedoff@gmail.com

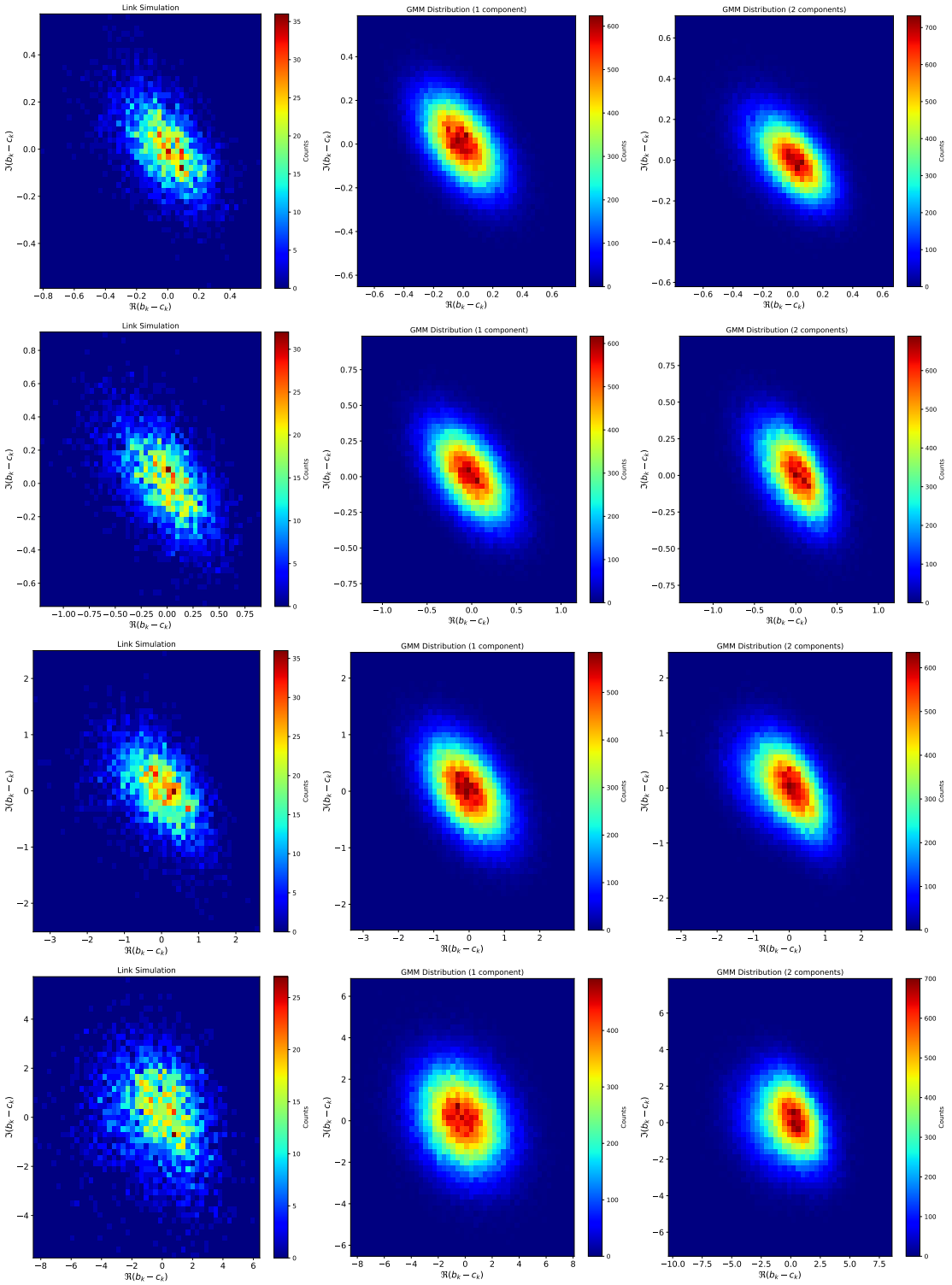


Figure 7.4: The distributions of the received constellation points b_k, centered with respect to the transmitted symbol c_k for triplet 552 $[1+3\mathrm{i}, 1+3\mathrm{i}, 3+1\mathrm{i}]$. The first column illustrates the distribution of simulation data for a real communication system. The second column represents the distribution fitted with a single two-dimensional Gaussian distribution, while the third column visualizes a mixture of two Gaussians.

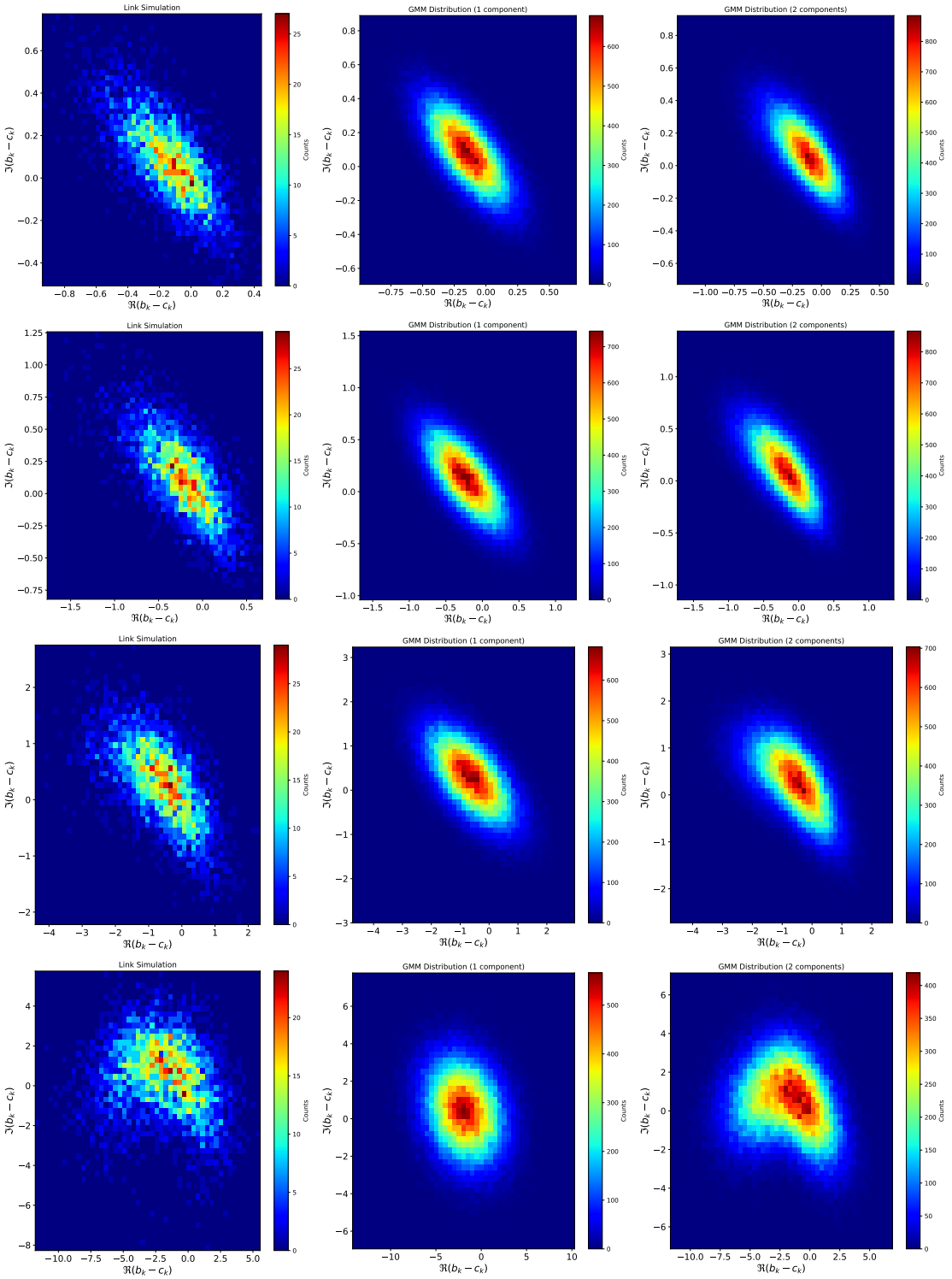


Figure 7.5: The distributions of the received constellation points b_k, centered with respect to the transmitted symbol c_k for triplet 2730 $[3+3\mathrm{i}, 3+3\mathrm{i}, 3+3\mathrm{i}]$. The first column illustrates the distribution of simulation data for a real communication system. The second column represents the distribution fitted with a single two-dimensional Gaussian distribution, while the third column visualizes a mixture of two Gaussians.

displays the distribution for triplet 2730. Notably, these distributions arise from nonlinear effects during the propagation of the WDM signal through the optical fiber, as the system excludes additional noise from EDFA amplifiers.

Both Figures 7.4 and 7.5 are structured as follows: rows correspond to different average signal power levels (first row at −2 dBm, second at 0 dBm, third at 4 dBm, and fourth at 8 dBm). The first column illustrates the distribution of simulation data for a real communication system with the aforementioned parameters. These distributions are then fitted with a GMM, and the derived parameters are used to generate the visualizations in the second and third columns. The second column represents the distribution fitted with a single two-dimensional Gaussian distribution, while the third column visualizes a mixture of two Gaussians, each column containing 100000 randomly generated points.

Examining Figure 7.4, at low average signal power, the GMMs with one and two components appear similar; however, the distribution for a single-component GMM is broader, indicating discernible differences. At 0 dBm average power, the distinction remains, though both distributions seem to fit the simulation reasonably well. The disparity becomes more visible at higher power levels, where the two-component GMM notably deviates from a Gaussian shape.

A comparable behaviour we can see for triplet 2730 in Figure 7.5. At lower average signal powers, the distributions vary but are distinct. A striking example is at the highest power level (last row), where both the simulation distribution and the GMM with two components form a distinctive shape of a German pretzel. It is evident that a single-component GMM fails to accurately represent this distribution.

These figures are merely exemplars out of 4096 different triplet distributions, yet they offer a compelling demonstration that the internal structure of distributions can be intricate and extend well beyond simple Gaussian forms.

Having observed that the shape of distributions for certain triplets can markedly differ when using single or multi-component GMM, we now proceed to compare the likelihoods of these distributions. To this end, we employ the log-likelihood measure described in the previous section (see Eq. (7.4)) to test the hypothesis that our simulation distribution corresponds to a specific GMM. Figure 7.6 presents violin plots[2] that illustrate the log-likelihood distribution for different average signal powers and numbers of components in the GMM. The x-axis of each plot denotes the number of components in the GMM, while the y-axis shows the distribution of log-likelihood values across all possible triplets. This visualization method is particularly useful given the 2^{12} possible triplet combinations, which would be exceedingly challenging to depict using line plots (as demonstrated in Fig. 7.8). The average power in the graphs on Fig. 7.6 ranges from -2 dBm (upper left) to 8 dBm (lower right).

[2] A violin plot is a method of plotting numeric data and can be understood as a combination of a box plot and a kernel density plot. It includes a marker for the median of the data and a box indicating the interquartile range, as seen in standard box plots, overlaid with a kernel density estimation.

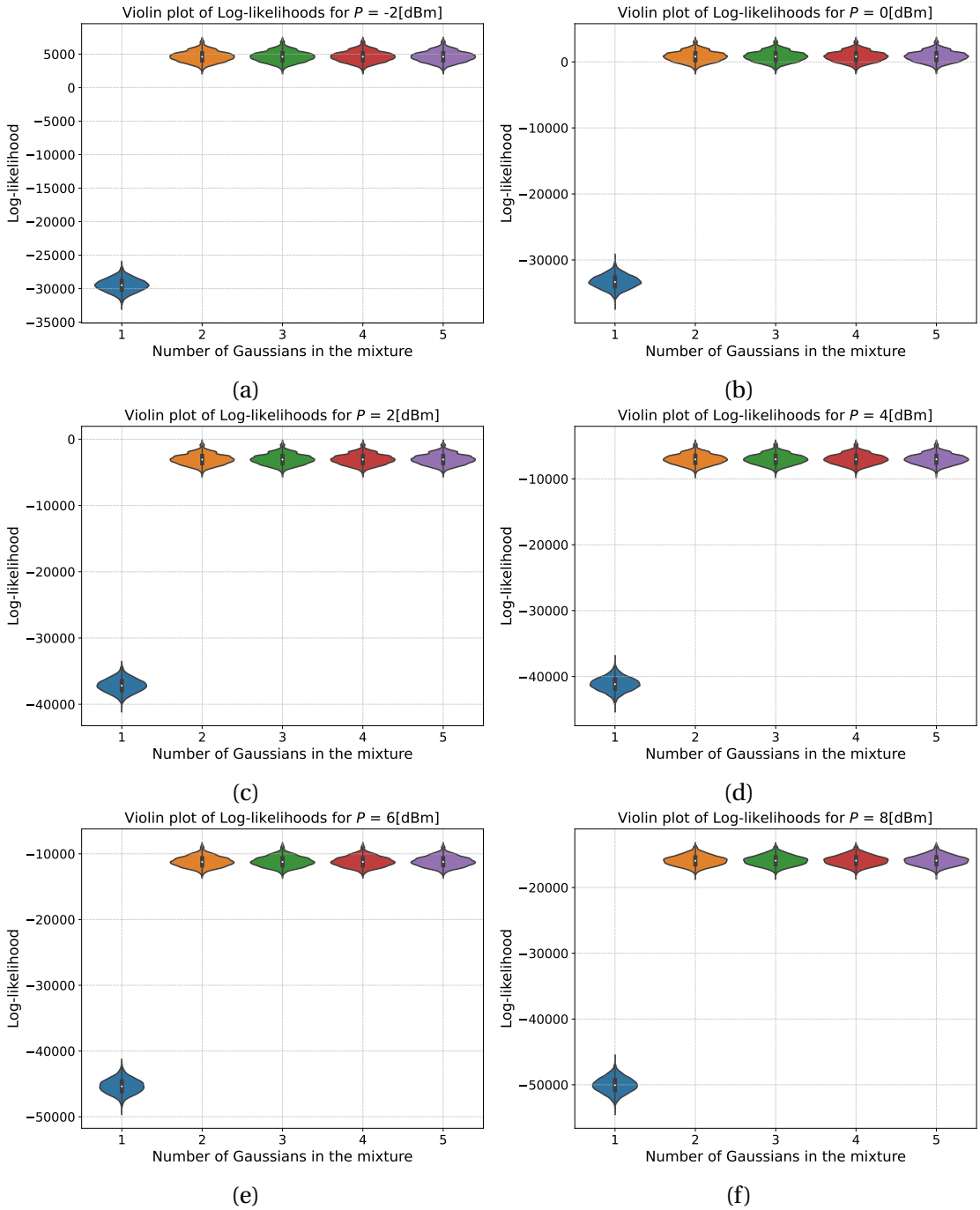
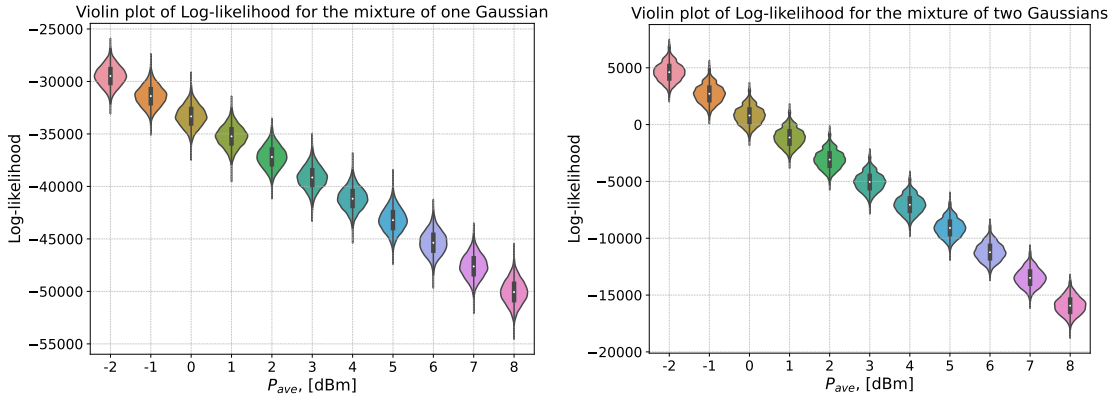


Figure 7.6: Violin plot of log-likelihood for triplet distribution for different number of components in GMM. **(a)** $P_{ave} = -2$ [dBm], **(b)** $P_{ave} = 0$ [dBm], **(c)** $P_{ave} = 2$ [dBm], **(d)** $P_{ave} = 4$ [dBm], **(e)** $P_{ave} = 6$ [dBm], **(f)** $P_{ave} = 8$ [dBm]



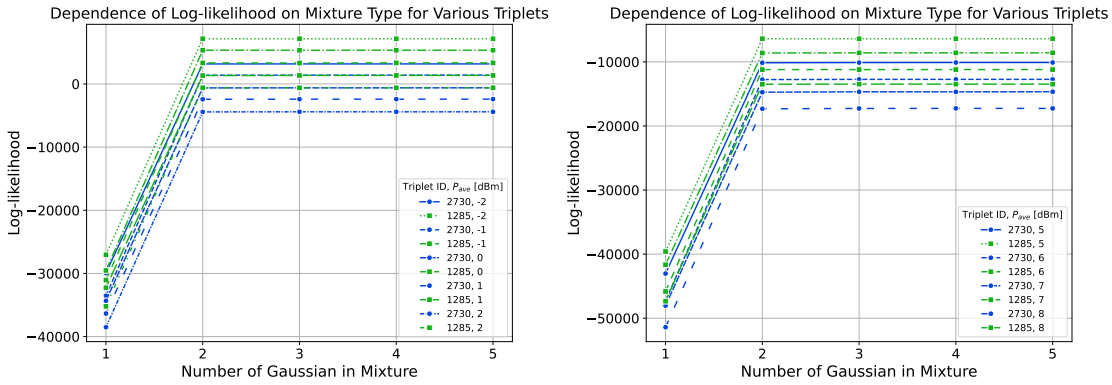
(a) (b)

Figure 7.7: Violin plot of log-likelihood dependence for triplet distribution on average signal power for different number of components in GMM. One Gaussian in the mixture **(left)** and mixture of two Gaussians **(right)**.

Regardless of the log-likelihood values, all graphs consistently reveal that the likelihood for a single-component GMM is significantly lower compared to a GMM with 2, 3, 4, or 5 components, which all exhibit approximately the same log-likelihood values. This pattern holds across all levels of average signal power. However, what varies with the power is the actual value of the log-likelihood. The principal finding, however, is as follows: the likelihood that our experimental data are drawn from a single-component GMM is exceedingly low compared to that from a multi-component GMM. This observation aligns with our earlier discussions: a standard Gaussian distribution does not adequately fit the experimental data, necessitating the use of a more complex model comprising multiple Gaussian components.

Figure 7.7 presents a violin plot of the log-likelihood for triplets across various power levels, contrasting a mixture with one component (left panel) against two components (right panel). In both instances, the average log-likelihood value decreases as the power increases, suggesting the potential for more intricate structural characteristics within the distribution. Notably, even at higher power levels, the two-component GMM provides a markedly superior fit to the experimental data; for instance, at 8 dBm, the average log-likelihood value for a single component is approximately -50000, whereas for two components it is around -16000. Although these values alone do not conclusively indicate that the experimental data correspond to a particular type of distribution, they do offer insights that a more suitable model might exist to describe such distributions. Furthermore, as illustrated in Fig. 7.6, the decline in average log-likelihood is consistent and nearly linear with increasing average power. This observation, coupled with insights from Fig. 7.7, indicates that adding more components to the mixture does not enhance the likelihood significantly. Therefore, for our simulations, a two-component mixture is deemed sufficient for an initial approximation of data representation.

Figures 7.8 and 7.9 offer a different view of the same data depicted by the violin plots in Figures 7.7 and 7.6, focusing on specific triplets. This perspective is valuable because violin



(a) (b)

Figure 7.8: Log-likelihood dependence for triplet distribution for different number of components in GMM. Graph shows triplets with ID 2730 $[3+3i, 3+3i, 3+3i]$ and ID 1285 $[-1-1i, 1+1i, -1-1i)]$.

plots can obscure the behavior of log-likelihood for individual triplets. These figures confirm a consistent finding: there is a substantial difference in the log-likelihood values between the single Gaussian model and a mixture of several Gaussians. The log-likelihood stabilizes with an increasing number of components in the GMM, indicating that a mixture of two Gaussians is typically adequate. The same trend is observed with the log-likelihood's response to signal power: it decreases linearly as the average power increases. Additionally, for a GMM with two components, the lines representing different triplets do not cross, affirming the consistency of this pattern.

7.4 Discussion

In digital communications understanding the impact of nonlinear effects is crucial for improving signal integrity and system performance. This research delves into the analysis of these effects through the lens of Gaussian Mixture Model, focusing on a constrained set of data points – triplets – to manage computational complexity and provide meaningful insights into the behavior of the system under nonlinear influences.

7.4.1 Computational Feasibility

Given the constraints of high-speed data generation tools, both time and storage space are limiting factors in our research. The choice to analyze triplets, resulting in 2^{12} possible combinations from a 16-QAM constellation, allows for a manageable dataset of 2^{24} data points, ensuring an average of 2^{12} points per combination for robust analysis. Expanding the neighborhood analysis to include pairs, thereby increasing the combinations to 2^{20}, would necessitate 2^{32} data points – a 256-fold increase – rendering the analysis impractical with current resources. For instance, fitting a GMM to one power level of 4096 triplets took approximately

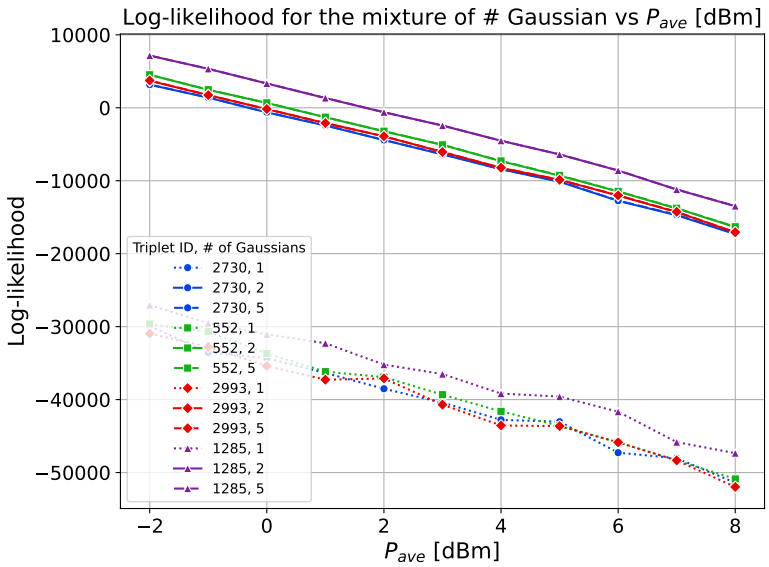


Figure 7.9: Dependence of log-likelihood for triplet distribution on average signal power for different number of components in GMM. Graph shows triplets with ID 2730 $[3+3i, 3+3i, 3+3i]$, ID 1285 $[-1-1i, 1+1i, -1-1i]$, ID 2993 $[3-3i, 3-3i, 1-1i]$ and ID 552 $[1+3i, 1+3i, 3+1i]$.

two hours on a standard workstation; extending this to five-point combinations would extend the analysis time to an impractical duration, approximately one month for a single power level, highlighting a significant gap in computational efficiency that is beyond the scope of this study.

7.4.2 Approach

The study of neighbor influences within signal transmission represents an iterative evolution in understanding how information is transformed and perceived. Initially, the focus was on the direct transformation of a transmitted point through signal propagation to its received counterpart, leading to rigid decision boundaries devoid of contextual data. Subsequently, the exploration expanded to include neighboring symbols at the receiver, aiming to enhance equalization accuracy through this supplementary information. For instance, as discussed in Section 6.1, the gradient boosting approach leverages data about received points and their neighbors to infer the original transmitted point.

Progressing beyond this, our research introduces an alternative perspective by examining the impact of sending sequence neighbors (rather than those received) on the configuration of received points. This method does not specify the propagation distance; instead, it investigates how neighboring points from the transmitter influence the received signal. The core objective of this work is to demonstrate how varying combinations of triplets lead to unique structures in the received points, driven by nonlinear effects. This phenomenon is attributed to the complex interplay within the nonlinear system, where the subsequent neighbors' influence is

also embedded in the resulting distribution. By increasing the dataset size, we can generalize the effect of different sequences of subsequent neighbors, providing a clearer understanding of the distribution patterns that emerge. This approach not only deepens our comprehension of signal propagation dynamics but also lays the groundwork for future enhancements in signal processing techniques.

7.4.3 Future Directions

In this phase of our analysis, the focus is intentionally not placed on the neighbors at the receiver, as the primary objective is to lay the groundwork for developing a message passing (MP) algorithm. This algorithm is envisaged to adopt a comprehensive approach by incorporating information about neighboring points, thereby facilitating more informed decision-making regarding the received points based on the sequence of neighbors. The algorithm will dynamically determine the optimal number of neighbors to consider, a parameter that will be fine-tuned in future studies to enhance performance. Utilizing the distribution information for triplets, the MP algorithm aims to refine solution accuracy. While the potential exists to extend this analysis to more complex structures involving five or more points, the current focus on triplets and their probability distributions is deemed sufficient for significantly improving decoding performance. Consequently, there is no immediate requirement to analyze longer initial sequences; this research marks a preliminary step towards the development of a sophisticated algorithm.

7.4.4 Conclusion

In summary, this research marks a pioneering step towards understanding how nonlinearities influence signal structures in 16-QAM systems, revealing that these structures deviate significantly from Gaussian models. Our investigation underscores the complexity inherent in the distribution of received signals, where Gaussian Mixture Model enhance fit but reach a saturation point in effectiveness with additional components. While the validation of these findings through error rate calculations has not been undertaken within the scope of this thesis, it forms a critical component of our ongoing research agenda.

Future directions will explore beyond the GMM to discover if alternative or more intricate models could more accurately capture the nuances of signal distributions. The immediate next step involves developing an algorithm that utilizes the GMM framework to predict the distribution category of received signals. By calculating the likelihood across different models, we can identify the most fitting distribution for a given signal point. Integrating message passing algorithms could further refine this prediction process, offering a robust tool for decoding signal distributions. Preliminary work is already underway, with the aim of substantiating the practical benefits of the proposed GMM configuration through rigorous error rate analysis in future publications.

Moreover, the analytical insights into signal distribution patterns hold promise for channel monitoring applications. Notably, discrepancies from anticipated distributions may signal underlying system issues, providing a novel diagnostic approach to maintain the operational integrity of optical communication networks.

Expanding this analysis to encompass various modulation formats and system configurations represents an exciting avenue for further research. The use of the HpCom [183] package will be instrumental in generating relevant statistics, facilitating a broad-based exploration of optical communication systems. Embracing these avenues for future investigation will not only deepen our understanding of the dynamics within optical communication systems but also contribute to enhancing their efficiency and reliability.

8 Conclusion and Outlook

This thesis aims to create new techniques which can be used to improve optical communication systems, vital for global connectivity, in the face of increasing data demands from technologies like 5G/6G and cloud computing. It particularly focuses on enhancing Nonlinear Fourier Transform (NFT) techniques and incorporating machine learning to optimize these systems.

A significant contribution is the development of a novel method for applying NFT to continuous signals, using Chromatic dispersion compensation (CDC) for pre-processing. This approach surpassed traditional Digital back-propagation (DBP) methods in single and dual polarization signal processing. The research further explores the presence of solitons in WDM and OFDM symbols, finding that soliton components carry a majority of the signal energy.

A breakthrough in this work is the use of Neural networks (NNs) for forward and inverse NFT, which showed promising results, especially in noisy signal environments where standard NFT methods falter. This innovative use of NNs demonstrates their potential in addressing nonlinearity issues in optical channels, suggesting that NFT could be a key to unlocking greater channel capacities and enhancing the performance of optical communication systems.

In the second part of the research, machine learning algorithms are explored for their utility in enhancing optical communication systems. The introduction of High-Performance COMmunication library (HpCom), a GPU-accelerated framework, marks a significant advancement, enabling efficient simulations of diverse optical fiber channels. This tool aids in data collection for further investigation, with examples and results included for reference.

Gradient Boosting (GB) is highlighted as an effective method for addressing the nonlinearity in optical channels, indicating its potential for broader application. Similarly, the adaptability and efficacy of NNs are demonstrated across various channel and signal parameters, affirming their role in system optimization.

A novel data-driven approach is applied to reveal intricate structures within received symbol constellations, a direct consequence of fiber nonlinearity. This discovery of non-Gaussian

patterns across different signal point groupings adds a new dimension to the understanding of optical fiber communication.

Overall, the integration of machine learning into optical communication presents a leap forward in both theory and practical application, promising to influence future technological developments in the industry.

A Appendix to Part I: Nonlinear Fourier Transform

A.1 Methods for numerical calculations

A.1.1 Boffetta-Osborne method for determining scattering data

This method was proposed in 1992 by Boffetta and Osborne and described in detail in their work [206]. The main idea of the method is that the ZS system can be rewritten as:

$$\begin{cases} \partial_t \psi_1 = -\xi \psi_1 + q\psi_2 \\ \partial_t \psi_2 = \sigma q^* \psi_1 + \xi \psi_2 \end{cases} \tag{A.1}$$

which, after passing to a discrete grid for t variable, gives an evolution of the spectral eigenfunction Ψ on each interval Δt:

$$\Psi(t_n + \Delta t) = U(q_n)\Psi(t_n) \tag{A.2}$$

where $U(q_n, \Delta t)$ is represented as an exponential function of the matrix $Q(\xi)$:

$$U(q) = \exp[\Delta t Q(\xi)] = \exp\left(\Delta t \begin{pmatrix} -i\xi & q \\ \sigma q^* & i\xi \end{pmatrix}\right) \tag{A.3}$$

If we expand the above exponent into a Taylor series and sum up the matrices, and then, assuming that each element of the matrix is a Taylor series, we summarize these rows for each element, then we get the following expression for the propagator:

$$\begin{aligned} U(q) &= \exp[\Delta t Q(\xi)] = \\ &= \begin{pmatrix} \cosh(k\Delta x) - \frac{i\xi}{k}\sinh(k\Delta x) & \frac{q}{k}\sinh(k\Delta x) \\ \frac{\sigma q^*}{k}\sinh(k\Delta x) & \cosh(k\Delta x) + \frac{i\xi}{k}\sinh(k\Delta x) \end{pmatrix} \end{aligned} \tag{A.4}$$

where $k^2 = \sigma|q|^2 - \xi^2$ is a constant on the interval Δt. Using the above method, we can solve the scattering problem — determine the coefficients $a(\xi)$ and $b(\xi)$. However, for a discrete

spectrum, this is not sufficient: since the coefficient $a(\xi)$ is zero, it is necessary to calculate its derivative $a'(\xi_n) = \frac{\partial a(\xi)}{\partial \xi}|_{\xi=\xi_n}$. Now we introduce a four-component field containing both the vector function Ψ and its derivative with respect to ξ:

$$\Xi(t,\xi) = \begin{pmatrix} \Psi \\ \Psi' \end{pmatrix} \tag{A.5}$$

where $\Psi' = \partial\Psi/\partial\xi$. Similar to the expressions (A.3) and (A.4) for the Ξ field, one can write a recursive relation by differentiating the expression (A.2):

$$\Xi(t_n + \Delta t) = T(q_n)\Xi(t_n) \tag{A.6}$$

here

$$T(q_n) = \begin{pmatrix} U(q_n) & 0 \\ U'(q_n) & U(q_n) \end{pmatrix} \tag{A.7}$$

4×4 matrix, $U'(q_n) = \partial U(q_n)/\partial\xi$ and can be written as:

$$\begin{aligned} U'_{11} &= i\Delta t\frac{\xi^2}{k^2}\cosh(k\Delta t) - \left(\xi\Delta t + i + i\frac{\xi^2}{k^2}\right)\frac{\sinh(k\Delta t)}{k} \\ U'_{12} &= -\frac{q\xi}{k^2}\left(\Delta t\cosh(k\Delta t) - \frac{\sinh(k\Delta t)}{k}\right) \\ U'_{21} &= -\frac{\sigma q\xi}{k^2}\left(\Delta t\cosh(k\Delta t) - \frac{\sinh(k\Delta t)}{k}\right) \\ U'_{22} &= -i\Delta t\frac{\xi^2}{k^2}\cosh(k\Delta t) - \left(\xi\Delta t - i - i\frac{\xi^2}{k^2}\right)\frac{\sinh(k\Delta t)}{k} \end{aligned} \tag{A.8}$$

For the numerical calculation, we consider the initial potential of $q(t,0)$ on the $M+1$ segments, assuming that

$$q(t,0) = q_n \quad \text{when} \quad t \in \left(t_n - \frac{\Delta t}{2};\ t_n + \frac{\Delta t}{2}\right]. \tag{A.9}$$

In this case, the discrete solution of the scattering problem is the following:

$$\Xi(t_n) = \prod_{j=n-1}^{-M/2} T(q_j)\Xi(t_{-M/2}). \tag{A.10}$$

To obtain the scattering data, it is necessary to calculate the scattering matrix S

$$S(\xi) = \prod_{j=M/2-1}^{-M/2} T(q_j) = \begin{pmatrix} \Sigma(\xi) & 0 \\ \Sigma^{'}(\xi) & \Sigma(\xi) \end{pmatrix}, \tag{A.11}$$

where matrix Σ is defined as

$$\Sigma = \prod_{j=M/2-1}^{-M/2} U(q_j), \quad \Sigma^{'} = \partial\Sigma/\partial\xi. \tag{A.12}$$

Initially, in the problem (1.15), the vector function Ψ extended from $-\infty$ to $+\infty$. In numerical simulation, this interval is replaced by the interval from $-\frac{T}{2}$ to $+\frac{T}{2}$. In this case, we get the following result:

$$\begin{aligned} a(\xi) &= S_{11}(\xi)e^{i\xi T}, \\ b(\xi) &= S_{21}(\xi), \\ \frac{\partial a(\xi)}{\partial \xi} &= [S_{31} + i\frac{T}{2}(S_{11} + S_{33})]e^{i\xi T}, \\ \frac{\partial b(\xi)}{\partial \xi} &= S_{41} + i\frac{T}{2}(S_{43} - S_{21}). \end{aligned} \tag{A.13}$$

This method allows us to find all the scattering data for an arbitrary ξ. Although the method does not directly calculate the spectrum of the problem (1.15), it can help us find it, because we know that at the points of the discrete spectrum on the complex plane the scattering coefficient $a(\xi) \equiv 0$.

It should be noted that for $t \ll 1$ the matrix $U(q)$ A.4 reduces to

$$U(q) \approx \begin{pmatrix} 1 - i\xi\Delta t & q\Delta t \\ \sigma q^* \Delta t & 1 + i\xi\Delta t \end{pmatrix}. \tag{A.14}$$

This form was found by Ablowitz and Ladik [207, 208] and can also be used to solve the Zakharov-Shabat problem.

A.1.2 Töplitz inner bordering method

The formation of technologies for creating Fiber Bragg Gratings (FBGs) [209, 210] was accompanied by the development of numerical methods for their modeling and analysis. Recovery of the refractive index from a given dependence of the scattering coefficient on the frequency is the inverse scattering problem, which is solved in mathematical physics using a pair of Gelfand-Levitan-Marchenko [53] equations.

The work [211] was proposed a method for solving the GLM equations (1.33) and (1.34). It consists in the fact that the GLM equations can be reduced to equations with the Toeplitz kernel by replacing $u(t,x) = A_1(t, t-x)$ and $v(t,y) = -A_2^*(t, y-t)$:

$$\begin{aligned} u(t,x) + \int_x^{2t} \Omega^*(y-x)v(t,y)dy = 0 \\ v(t,y) + \int_0^y \Omega(y-x)u(t,x)dx + \Omega(y) = 0 \end{aligned} \tag{A.15}$$

where $\Omega = \Omega_{sol} + \Omega_{rad} = \sum_n c_n e^{-i\xi_n x} + \frac{1}{2\pi}\int_{-\infty}^{+\infty} d\xi\, r(\xi)e^{-i\xi x}$ — integral kernel, the parameters x and y are in the range $0 \le x, y < 2t$, the variable t is in the range $0 \le t < T$, and the signal is restored using the formula $q(t) = 2v(t, 2t-0)$.

Further, the equations (A.15) are discretized on a uniform grid, and the integrals are represented as sums. The resulting equations have a block structure and are written in matrix form:

$$\mathbf{G}\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix} = \begin{pmatrix}\mathbf{a}\\ \mathbf{b}\end{pmatrix} \tag{A.16}$$

where matrix $\mathbf{G}$ and vectors $\mathbf{u}$, $\mathbf{v}$, $\mathbf{a}$ and $\mathbf{b}$ are defined by:

$$\mathbf{G} = \begin{pmatrix}\mathbf{E} & \pm\Omega^{\dagger}\\ \Omega & \mathbf{E}\end{pmatrix}, \quad \Omega = h\begin{pmatrix}\frac{1}{2}\Omega_0 & 0 & \dots & 0\\ \Omega_1 & \frac{1}{2}\Omega_0 & \dots & 0\\ & & \ddots & \\ \Omega_{m-1} & \Omega_{m-2} & \dots & \frac{1}{2}\Omega_0\end{pmatrix}$$
$$\mathbf{a} = \pm\frac{h}{2}v_m^{(m)}\tilde{\rho^*}, \mathbf{b} = -\left(1+\frac{h}{2}u_1^{(m)}\rho\right),$$
$$\rho = \begin{pmatrix}\Omega_1\\ \vdots\\ \Omega_m\end{pmatrix}, \mathbf{u} = \begin{pmatrix}u_1^{(m)}\\ \vdots\\ u_m^{(m)}\end{pmatrix}, \mathbf{v} = \begin{pmatrix}v_1^{(m)}\\ \vdots\\ v_m^{(m)}\end{pmatrix}. \tag{A.17}$$

$\mathbf{E}$ — identity matrix.

The first and last elements of the solution vector are calculated by the formulas:

$$u_1^{(m)} = \pm\frac{h}{2}v_m^{(m)}\langle\mathbf{y}^*|\tilde{\rho}^*\rangle \mp \left(1+\frac{h}{2}u_1^{(m)}\right)\langle\mathbf{z}^*|\rho\rangle,$$
$$u_1^{(m)} = \pm\frac{h}{2}v_m^{(m)}\langle\tilde{\mathbf{z}}|\tilde{\rho}^*\rangle - \left(1+\frac{h}{2}u_1^{(m)}\right)\langle\tilde{\mathbf{y}}|\rho\rangle. \tag{A.18}$$

Parentheses denote scalar product $\langle\mathbf{x}|\mathbf{y}\rangle = \mathbf{x}^T\cdot\mathbf{y}$. We can introduce the following parameters

$$\alpha_m = h\langle\mathbf{z}^*|\rho\rangle, \quad \beta_m = h\langle\tilde{\mathbf{y}}|\rho\rangle. \tag{A.19}$$

Then equation (A.18) allows calculate $v_m^{(m)}$:

$$v_m^{(m)} = \frac{-\beta_m/h}{1\pm\mathrm{Im}\ \alpha_m - \frac{1}{4}(|\alpha_m|^2 \mp |\beta_m|^2)}. \tag{A.20}$$

We assume that with the required accuracy $\alpha_m = -u(x_m,0)h + O(h^2)$, which leads us to the final solution:

$$q_m = 2v_m^{(m)} = -2\beta_m/h + O(h^2). \tag{A.21}$$

This procedure can be described as an algorithm that can be reversed. Next, we present two algorithms of direct and inverse transformations, as a result of which a signal can be reconstructed from the core (Algorithm 1) and vice versa: the integral core is obtained from the signal (Algorithm 2).

Algorithm 1. Inverse TIB algorithm for signal recovery from the kernel

1: $m = 1\ q_0 = -2\Omega_0\ y_0^{(1)} = \frac{1}{1\mp h^2|\Omega_0|^2/4}\ z_0^{(1)} = -\frac{y_0^1 h\Omega_0}{2}$
2: $\beta_m = h\sum_{j=0}^{m-1}\Omega_{m-j}y_j^{(m)}$
3: $q_m = -2\beta_m/h$
4: $c_m = \frac{1}{1\mp|\beta_m|^2}, d_m = -\beta_m c_m$
5: $\mathbf{y}^{(m+1)} = c_m\begin{pmatrix}\mathbf{y}^{(m)}\\0\end{pmatrix} + d_m\begin{pmatrix}0\\\pm\tilde{\mathbf{z}}^{*(m)}\end{pmatrix}$
6: $\mathbf{z}^{(m+1)} = c_m\begin{pmatrix}\mathbf{z}^{(m)}\\0\end{pmatrix} + d_m\begin{pmatrix}0\\\tilde{\mathbf{y}}^{*(m)}\end{pmatrix}$
7: Increment m, go to the step 2.

Algorithm 2. TIB direct algorithm for calculating the kernel from the signal

1: $\Omega_0 = -q_0/2$
2: $\beta_m = -hq_m/2$
3: $\Omega_m = \left(\beta_m - h\sum_{j=1}^{m-1}R_{m-j}y_j^{(m)}\right)/hy_0^{(m)}$
4: $c_m = \frac{1}{1\mp|\beta_m|^2}, d_m = -\beta_m c_m$
5: $\mathbf{y}^{(m+1)} = c_m\begin{pmatrix}\mathbf{y}^{(m)}\\0\end{pmatrix} + d_m\begin{pmatrix}0\\\pm\tilde{\mathbf{z}}^{*(m)}\end{pmatrix}$
6: $\mathbf{z}^{(m+1)} = c_m\begin{pmatrix}\mathbf{z}^{(m)}\\0\end{pmatrix} + d_m\begin{pmatrix}0\\\tilde{\mathbf{y}}^{*(m)}\end{pmatrix}$
7: Increment m, go to the step 2.

A.1.3 N-soliton solution

Factorization of GLM equations

For a discrete spectrum, factorization of the kernel of the GLM equations leads to a system of algebraic equations (a detailed description can be found in [212]). Then the N-soliton solution can be found using the following exact expression:

$$q^{(N)}(t,z=0) = -2\langle\Psi(t)|(\widehat{\mathbf{E}}+\widehat{\mathbf{M}}^*(t)\widehat{\mathbf{M}}(t))^{-1}|\Phi(t)\rangle \tag{A.22}$$

$\widehat{\mathbf{E}}$ is $N\times N$ identity matrix,

$$\begin{aligned}\langle\Psi(t)| &= \langle r_1 e^{-i\xi_1 t},...,r_N e^{-i\xi_N t}|,\\ \langle\Phi(t)| &= \langle e^{-i\xi_1 t},...,e^{-i\xi_N t}|,\\ \widehat{\mathbf{M}}_{k,j}(t) &= r_j\frac{e^{i(\xi_k^*-\xi_j)t}}{\xi_k^*-\xi_j},\end{aligned} \tag{A.23}$$

r_j were defined in (1.24). For large N, numerical algorithms for this formula are unstable, but for small N this method is very convenient. For the numerical inversion of matrices, $O(n^4)$ operations are required in the general case. Special algorithms require $O(n^3)$ operations. The process accumulates a calculation error and we cannot find a solution for large N ($N > 100$).

Recursive Darboux method

There is another method to find the N-soliton solution, which is called the dressing method. The main idea of the method is to use the Darboux transformation, which is used to construct an iterative scheme for finding the N-soliton solution. Consider this method in more detail.

Let us introduce the following system:

$$\begin{cases} \Psi_t = P(\xi, q)\Psi, \\ \Psi_z = M(\xi, q)\Psi, \end{cases}, \tag{A.24}$$

where operator M is defined in (1.13), and P is calculated by the formula

$$P = \begin{pmatrix} -i\xi & q(t,z) \\ -q^*(t,z) & i\xi \end{pmatrix}. \tag{A.25}$$

This is the same $Q(t,z)$ matrix from the expression (1.14), but with $\sigma = 1$, since we study soliton solutions. Also, we use the notation ξ – a complex number, but not necessarily an eigenvalue for q, which is the solution underlying the system (A.24).

The algorithm begins with a trivial solution $q^{(0)} = 0$, and the initial eigenvectors are chosen as $v(t,\xi_j;0) = [A_j e^{-i\xi_j t}, B_j e^{i\xi_j t}]$. The coefficients A_j and B_j determine the spectral amplitude and shape of the pulse. In order to get new eigenvectors, one needs to use the following formulas:

$$\begin{aligned} v_1(t,\xi_j;q^{(k+1)}) = &\frac{1}{\|v(t,\xi_{k+1};q^{(k)})\|^2} \\ &\{\{(\xi_j - \xi_{k+1})|v_1(t,\xi_{k+1};q^{(k)})|^2 + \\ &+ (\xi_j - \xi^*_{k+1})|v_2(t,\xi_{k+1};q^{(k)})|^2\} v_1(t,\xi_j;q^{(k)}) \\ &(\xi^*_{k+1} - \xi_{k+1}) v_1(t,\xi_{k+1};q^{(k)}) v_2^*(t,\xi_{k+1};q^{(k)}) v_2(t,\xi_j;q^{(k)})\}, \end{aligned} \tag{A.26}$$

$$\begin{aligned} v_2(t,\xi_j;q^{(k+1)}) = &\frac{1}{\|v(t,\xi_{k+1};q^{(k)})\|^2} \\ &\{\{(\xi_j - \xi^*_{k+1})|v_1(t,\xi_{k+1};q^{(k)})|^2 + \\ &+ (\xi_j - \xi_{k+1})|v_2(t,\xi_{k+1};q^{(k)})|^2\} v_2(t,\xi_j;q^{(k)}) \\ &(\xi^*_{k+1} - \xi_{k+1}) v_1^*(t,\xi_{k+1};q^{(k)}) v_2(t,\xi_{k+1};q^{(k)}) v_1(t,\xi_j;q^{(k)})\}, \end{aligned} \tag{A.27}$$

for $k = 0, ..., N-2$ and $j = k+2, ..., N$. A signal addition occurs according to the formula:

$$q^{(k+1)} = q^{(k)} + 2i(\xi^*_{k+1} - \xi_{k+1}) \frac{v_1(t,\xi_{k+1};q^{(k)})}{\|v(t,\xi_{k+1};q^{(k)})\|^2} \tag{A.28}$$

Such a method is convenient for obtaining N-soliton solutions with a large value of N, since matrices are not required to be inverted.

A.1.4 Fourier collocation method for finding a nonlinear spectrum

For most of the initial potentials, the nonlinear spectrum cannot be calculated analytically. In such cases, it is necessary to use numerical algorithms.

The equation (1.15) is a standard problem of finding the eigenvalues of the operator L and is represented as a system:

$$\begin{cases} -\partial_t \psi_1 + q(t,0)\psi_2 = \xi\psi_1 \\ \partial_t \psi_2 - \sigma q^*(t,0)\psi_1 = \xi\psi_2. \end{cases} \tag{A.29}$$

One way to solve this problem is to move to finite difference schemes on a discrete grid. For this, a finite interval is selected from the infinite interval along the t axis and is divided into finite intervals, in accordance with the sampling grid. Derivatives operators are replaced by their finite-difference analogues. After these operations, the problem (A.29) becomes the standard task of finding eigenvalues and eigenfunctions for matrices, which can be solved by standard algorithms: LR, QR, the Cholesky method and so on. It should be noted that the QR-algorithm is more stable than the LR, which is now almost not used. Unfortunately, the accuracy of approximation by means of the finite-difference method is rather low and can lead to the appearance of "spurious" eigenvalues.

We can assume that if we could rewrite the original system (A.29) without differential operators, then we could use this method to find the nonlinear spectrum. In this question, the classical Fourier transform helps us, in which differential operators are replaced with multiplication by the appropriate factor:

$$\begin{cases} -\partial_t \psi_1 + q(t,0)\psi_2 = \xi\psi_1 \\ \partial_t \psi_2 - \sigma q^*(t,0)\psi_1 = \xi\psi_2 \end{cases} \rightarrow \begin{cases} -ik\phi_1 + u\phi_2 = \xi\phi_1 \\ ik\phi_2 - \sigma u^*\phi_1 = \xi\phi_2 \end{cases} \tag{A.30}$$

In this transform, we replace the initial potential and eigenfunctions with the following discrete sums:

$$\psi_1(t) = \sum_{-N/2}^{N/2} \phi_{1,n} e^{ink_0 t}, \psi_2(t) = \sum_{-N/2}^{N/2} \phi_{2,n} e^{ink_0 t}, q(t,0) = \sum_{-N/2}^{N/2} u_n e^{ink_0 t}, \tag{A.31}$$

where $k_0 = 2\pi/T$, T is a time slot and $N+1$ is a number of discretization points (number of Fourier modes) in the Fourier transform of the initial signal $q(t,0)$ (N is even). With this transform, the initial problem (A.29) is replaced by the problem of finding eigenvalues for a block matrix:

$$\begin{pmatrix} -H_1 & H_2 \\ H_2^\dagger & H_1 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = i\xi \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}, \tag{A.32}$$

where

$$\begin{aligned} \phi_1 &= (\phi_{1,-N/2}, \phi_{1,-N/2+1}, ..., \phi_{1,N/2-1}, \phi_{1,N/2})^T \\ \phi_2 &= (\phi_{2,-N/2}, \phi_{2,-N/2+1}, ..., \phi_{2,N/2-1}, \phi_{2,N/2})^T \end{aligned} \tag{A.33}$$

The blocks of the matrix are as follows:

$$H_1 = ik_0 \begin{pmatrix} -\frac{N}{2} & & & & \\ & -\frac{N}{2}+1 & & & \\ & & \ddots & & \\ & & & \frac{N}{2}-1 & \\ & & & & \frac{N}{2} \end{pmatrix} \tag{A.34}$$

$$H_2 = \begin{pmatrix} u_0 & u_{-1} & \dots & u_{-N} & & & \\ u_1 & u_0 & u_{-1} & \ddots & \ddots & & \\ \vdots & u_1 & u_0 & \ddots & \ddots & \ddots & \\ u_N & \ddots & \ddots & \ddots & \ddots & \ddots & u_{-N} \\ & u_N & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & \ddots & \ddots & u_{-1} \\ & & & u_N & \dots & u_{-1} & u_0 \end{pmatrix}. \tag{A.35}$$

The accuracy of finding the eigenvalues for this system is much higher than for the original equation (A.29), and the "fake" eigenvalues do not appear in the nonlinear spectrum.

A.1.5 Cauchy Integral

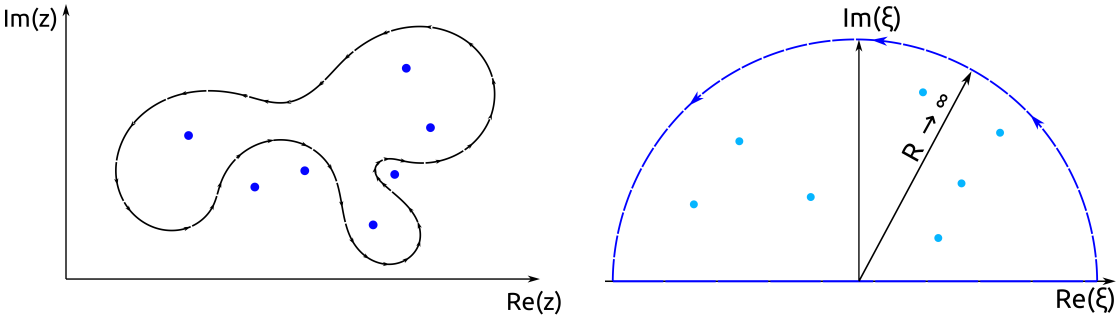
The problem of finding a discrete spectrum, as we have seen earlier, can be reduced to finding the zeros of the coefficient $a(\xi)$ in the upper complex half-plane. In this interpretation, the dependence of the coefficient a on the complex variable ξ can be understood as a function defined on the set of complex numbers. Therefore, we can apply the Cauchy integral formula to solve the above problem:

$$f(z_0) = \frac{1}{2\pi i}\int_\Gamma \frac{f(z)}{z-z_0}dz, \tag{A.36}$$

where D is a region in the complex plane with the boundary $\Gamma = \partial D$, $f(z)$ is a holomorphic function in $\bar{D}$, z_0 is a point in D. We consider one turn, i.e. one bypass along the border of the region D. To find the eigenvalues, we use the method described in [213, 214], which uses the set of contour integrals $\{s_p\}_{p=1}^P$:

$$s_p = \frac{1}{2\pi i}\int_\Gamma \xi^p \frac{a'(\xi)}{a(\xi)}d\xi = \sum_{j=1}^{P}\xi_j^p, \quad p = 1...P. \tag{A.37}$$

where P is the number of zeros of the coefficient $a(\xi)$ in the upper half-plane. Each contour integral gives us the sum to the right in the expression that is used to find all the eigenvalues (Fig. A.1a). When $p = 0$, the expression gives the total number of zeros inside the contour Γ. This fact can be used to count the number of discrete eigenvalues for a given signal. To do this, we rewrite the formula (A.37) for a contour that passes along the real axe and closes in the



(a) (b)

Figure A.1: **(a)** Traversing several zeros on the complex plane. **(b)** Calculation the number of zeros of a complex function.

upper half-plane through an infinitely distant point (Fig. A.1b):

$$N = \frac{1}{2\pi} Arg(a(\xi))|_{-\infty}^{+\infty}, \tag{A.38}$$

where N is the total number of solitons in the signal, the spectral parameter ξ takes values from $-\infty$ to $+\infty$ on the real axe. In the framework of this task $N = P$. This formula is convenient for numerical calculation: choosing a sufficiently large interval of the real straight line and calculating the phase change of the coefficient $a(\xi)$, we will know exactly the number of discrete eigenvalues.

To find the individual eigenvalues, the so-called Newton identities $\{\sigma\}_{p=1}^{P}$ are used:

$$\begin{aligned} -\sum_{j=1}^{P} \xi_j &= \sigma_1, \\ \xi_1\xi_2 + \xi_2\xi_3 + ... + \xi_{P-1}\xi_P &= \sigma_2, \\ &... \\ (-1)^P \xi_1\xi_2...\xi_P &= \sigma_P. \end{aligned} \tag{A.39}$$

which are associated with the value of the integrals:

$$\begin{aligned} s_1 + \sigma_1 &= 0, \\ s_2 + s_1\sigma_1 + 2\sigma_2 &= 0, \\ &... \\ s_P + s_{P-1}\sigma_1 + ... + s_1\sigma_{P-1} + P\sigma_P &= 0 \end{aligned} \tag{A.40}$$

Solving this system, we obtain the values of each Newton identity, calculated by the recurrence formula:

$$\sigma_p = -\frac{1}{p}\left(s_p + \sum_{j=1}^{p-1} s_j\sigma_{p-j}\right), \quad p = 1...P. \tag{A.41}$$

As a result, σ_p (up to a sign) are Viète's formulas for the polynomial

$$M(z) = z^P + \sigma_1 z^{P-1} + \sigma_2 z^{P-2} + ... + \sigma_{P-1} z + \sigma_P, \tag{A.42}$$

which zeros coincide with the zeros of the original coefficient $a(\xi)$. Using any algorithm to find the zeros of a polynomial, we thus calculate the discrete spectrum $\{\xi_j\}_{j=1}^P$.

One of the ways to find the zeros of the polynomial (A.42) is to go to the problem of finding eigenvalues of the so-called companion matrix $C(M)$:

$$C(M(z)) = \begin{bmatrix} 0 & 0 & \dots & 0 & -\sigma_P \\ 1 & 0 & \dots & 0 & -\sigma_{P-1} \\ 0 & 1 & \dots & 0 & -\sigma_{P-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -\sigma_1 \end{bmatrix}, \tag{A.43}$$

for which the polynomial (A.42) is a characteristic polynomial. There exist effective algorithms of finding eigenvalues of such a matrix, for example, the QR-algorithm [215–217].

A.2 Complexity of Nonlinear Fourier Transform

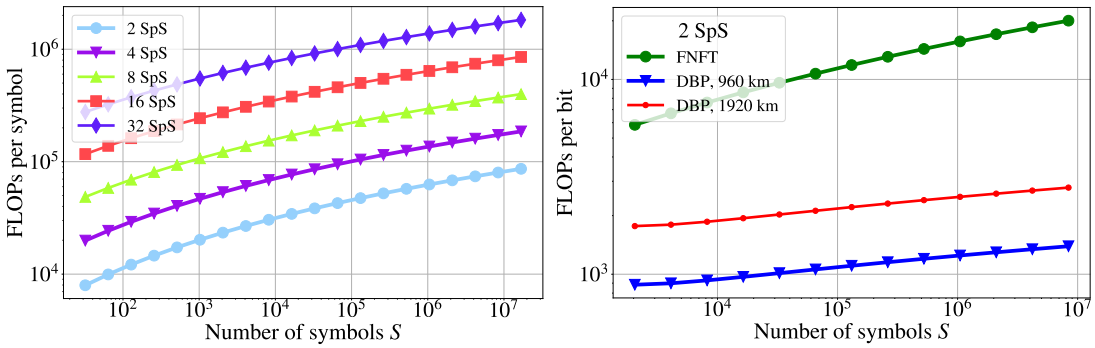
Computational complexity is one of the most important parameters of processing systems. Number of floating point operations (FLOP) is used to evaluate this parameter. To assess complexity in telecommunications, the FLOPs per symbol or per bit of transmitted information is used. Fig. A.2a shows the value of FLOPs per symbol for the FNFT method depending on the number of samples per symbol (SpS). As expected, the complexity increases with the number of symbols transmitted, as the overall signal length grows. As SpS increases, the graph shifts upward, since the number of points in the signal increases proportionally. Fig. A.2b shows a comparison of the complexity of FNFT and DBP. This graph shows the value of FLOPs per bit. The graph shows the FNFT method without additional estimation of the complexity for the PJT method and the complexity of calculating the normalization factors for the discrete spectrum. The graph shows that the complexity of the NFT is several times greater than the complexity of the DBP. However, the complexity of the NFT is independent of the fiber length.

The peculiarity of estimating the number of operations for searching for discrete eigenvalues is that it depends on the signal power. At higher power, there are more points in the continuous spectrum, therefore the algorithm works longer. For PJT, the number of operations can be estimated as follows: we can multiply the number of discrete eigenvalues by the average of the number of operations to calculate one eigenvalue. The minimum number of operations can be achieved using the Ablowitz-Ladik method, which gives us $49N$ operations to calculate one coefficient $a(\xi)$.

For example, our research shows that for 2048 symbols, with an average power of 0 dBm,

about 10^5 additional calculations of the coefficients $a(\xi)$ are required. The number of discrete eigenvalues is about 10^3. As a result, $10^5 \cdot 49N$ operations will be added to the FLOPs for calculating the continuous spectrum. Then the number of operations for the bi-directional algorithm for calculating the phase coefficients is added to this number. It is equal to the number of eigenvalues multiplied by the number of operations to calculate one coefficient: $10^3 \cdot 152N$ FLOPs. With 2048 symbols and SpS = 2, the number of points in the signal is $N = 2 \cdot 2^{11}$, which means in addition to the FLOPs per symbol shown in Fig. A.2, $10^5 \cdot 50.5 \cdot 2 \approx 10^7$ operations are added. This shows that the main computational costs are spent on calculating the discrete spectrum and phase coefficients. As a solution to this problem, adaptive and fast algorithms can be used to reduce the number of operations. For this it is necessary to develop a fast bi-directional algorithm scheme.

We assume the number of operations for the inverse problem to be equal to the number of operations for the direct problem, up to a coefficient. The main costs are required for the Darboux method, the number of operations for which is proportional to N^2.



(a) **(b)**

Figure A.2: Complexity comparison. **(a)** Number of FLOPs per transmitted symbol for different number of samples per symbol (SpS) for the FNFT method. **(b)** Number of FLOPs per transmitted bit for DBP and FNFT methods (SpS = 2).

A.3 Bayesian Optimization of Neural Network Parameters

```
import os
import csv
import numpy as np
from tqdm import tqdm

from os.path import dirname, join as pjoin
import scipy as sp
import scipy.io as sio

import tensorflow as tf
from keras import callbacks
from keras.models import Sequential
```

```
from keras.layers import Activation, Dense, Reshape, Flatten,
  BatchNormalization, Conv1D

from skopt import gbrt_minimize, gp_minimize, load
from skopt.utils import use_named_args

from skopt.space import Real, Categorical, Integer
from skopt.callbacks import CheckpointSaver

job_name = "polycoef_a_norm_1"
# DRIVE_DIR = "/home/esf0/data_for_nn/data_nfdm/"
DRIVE_DIR = "/work/ec180/ec180/esedov/data_for_polycoef/"
checkpoint_filepath = DRIVE_DIR + 'model_' + job_name + '_checkpoint'
skopt_checkpoint_file = "b_checkpoint_" + job_name + ".pkl"
skopt_param_file = "bo_param_" + job_name + ".txt"
load_flag = True  # previous bayesian optimiser state

input_shape = (1024, 2)
output_shape = (1024, 2)

patience = 500
n_epochs = 30000
n_params_max = 20_000_000
initial_batch_size = 15000

# tf.keras.losses.MeanSquaredError()
# tf.keras.losses.MeanSquaredLogarithmicError()
loss_function = tf.keras.losses.MeanSquaredError()
learning_rate = 1e-4

f_data_load = True
if f_data_load:
    train_x = np.load(DRIVE_DIR + "data_polycoef_train_x_p_0_50_run_0.npy
  ", allow_pickle=True)
    train_y = np.load(DRIVE_DIR + "
  data_polycoef_train_a_normalised_p_0_50_run_0.npy", allow_pickle=True)

    test_x = np.load(DRIVE_DIR + "data_polycoef_test_x_p_0_50_run_0.npy",
   allow_pickle=True)
    test_y = np.load(DRIVE_DIR + "
  data_polycoef_test_a_normalised_p_0_50_run_0.npy", allow_pickle=True)

    print('data loaded')


n_conv_layers = Integer(low=2, high=4, name='n_conv')

n_filters_1 = Integer(low=10, high=128, name='filters1')
n_filters_2 = Integer(low=10, high=128, name='filters2')
n_filters_3 = Integer(low=10, high=256, name='filters3')
n_filters_4 = Integer(low=10, high=256, name='filters4')

```

```
s_kernel_1 = Integer(low=1, high=20, name='kernel1')
s_kernel_2 = Integer(low=1, high=20, name='kernel2')
s_kernel_3 = Integer(low=1, high=20, name='kernel3')
s_kernel_4 = Integer(low=1, high=20, name='kernel4')

stride_1 = Integer(low=1, high=3, name='stride1')
stride_2 = Integer(low=1, high=3, name='stride2')
stride_3 = Integer(low=1, high=3, name='stride3')
stride_4 = Integer(low=1, high=3, name='stride4')

dilation_1 = Integer(low=1, high=3, name='dilation1')
dilation_2 = Integer(low=1, high=3, name='dilation2')
dilation_3 = Integer(low=1, high=3, name='dilation3')
dilation_4 = Integer(low=1, high=3, name='dilation4')

n_dense_layers = Integer(low=1, high=3, name='n_dense')
n_dense_cell_1 = Integer(low=128, high=4096, name='cell1')
n_dense_cell_2 = Integer(low=128, high=4096, name='cell2')
n_dense_cell_3 = Integer(low=128, high=4096, name='cell3')

act_func_1 = Categorical(name='activation1', categories=['relu', 'sigmoid
  ', 'tanh'])
act_func_2 = Categorical(name='activation2', categories=['relu', 'sigmoid
  ', 'tanh'])
act_func_3 = Categorical(name='activation3', categories=['relu', 'sigmoid
  ', 'tanh'])
act_func_4 = Categorical(name='activation4', categories=['relu', 'sigmoid
  ', 'tanh'])
act_func_5 = Categorical(name='activation5', categories=['relu', 'sigmoid
  ', 'tanh'])
act_func_6 = Categorical(name='activation6', categories=['relu', 'sigmoid
  ', 'tanh'])
act_func_7 = Categorical(name='activation7', categories=['relu', 'sigmoid
  ', 'tanh'])


dimensions = [n_conv_layers,
              n_filters_1, n_filters_2, n_filters_3, n_filters_4,
              s_kernel_1, s_kernel_2, s_kernel_3, s_kernel_4,
              stride_1, stride_2, stride_3, stride_4,
              dilation_1, dilation_2, dilation_3, dilation_4,
              n_dense_layers,
              n_dense_cell_1, n_dense_cell_2, n_dense_cell_3,
              act_func_1, act_func_2, act_func_3, act_func_4, act_func_5,
   act_func_6, act_func_7]


def parse_arguments(kwargs):
    # print('kwargs: ', kwargs)
    # parse arguments for convolutional layers
    # print(kwargs['n_conv'])
    n_conv = kwargs['n_conv']
```

```
104
105     n_filters = []
106     s_kernel = []
107     stride = []
108     dilation = []
109     for k in range(n_conv):
110         # print(kwargs['filters' + str(k + 1)])
111         n_filters.append(kwargs['filters' + str(k + 1)])
112         dilation.append(tuple([kwargs['dilation' + str(k + 1)]]))
113         s_kernel.append(tuple([kwargs['kernel' + str(k + 1)]]))
114         if kwargs['dilation' + str(k + 1)] > 1:
115             stride.append(tuple([1]))  # TF doesn't support stride > 1
    for dilation > 1
116         else:
117             stride.append(tuple([kwargs['stride' + str(k + 1)]]))
118
119     # print(n_filters, s_kernel, stride, dilation)
120
121     # parse arguments for dense layers
122     n_dense = kwargs['n_dense']
123     # print(n_dense)
124
125     n_cell = []
126     for k in range(n_dense):
127         n_cell.append(kwargs['cell' + str(k + 1)])
128
129     activation = []
130     for k in range(n_dense + n_conv):
131         activation.append(kwargs['activation' + str(k + 1)])
132
133     # print(n_cell, activation)
134
135     return n_conv, n_dense, n_filters, s_kernel, stride, dilation, n_cell
    , activation
136
137
138 def create_model(n_conv, n_filters, s_kernel, stride, dilation, n_dense,
    n_cell, activation, padding='valid',
139                  input_shape=(1024, 2), output_shape=1024):
140     # build convolutional neural network
141     # first layer has to have input shape
142     list_of_layers = [
143         Conv1D(filters=n_filters[0], kernel_size=s_kernel[0], strides=
    stride[0], dilation_rate=dilation[0],
144                activation=activation[0], padding=padding, input_shape=
    input_shape)]
145     for k in range(1, n_conv):
146         list_of_layers.append(
147             Conv1D(filters=n_filters[k], kernel_size=s_kernel[k], strides
    =stride[k], dilation_rate=dilation[k],
148                    activation=activation[k], padding=padding))
149     list_of_layers.append(Flatten())
```

```
150     for k in range(n_dense):
151         list_of_layers.append(Dense(units=n_cell[k], activation=
    activation[k + n_conv]))
152 
153     if len(np.shape(output_shape)) == 0:
154         list_of_layers.append(Dense(units=output_shape))  # output layer
    has to have output shape
155     else:
156         list_of_layers.append(Dense(units=(output_shape[0] * output_shape
    [1])))
157         list_of_layers.append(Reshape(output_shape))
158     model = Sequential(list_of_layers)
159 
160     return model
161 
162 
163 @use_named_args(dimensions=dimensions)
164 def fitness(**kwargs):
165     # this function is designed to make an arbitrary convolution neural
    network
166     # so we have to parse argument in the beginning
167 
168     n_conv, n_dense, n_filters, s_kernel, stride, dilation, n_cell,
    activation = parse_arguments(kwargs)
169     # print(n_conv, n_filters, s_kernel, stride, dilation)
170     # print(n_dense, n_cell, activation)
171 
172     losses = np.array([999.0])
173     if load_flag:
174         try:
175             losses = np.load(DRIVE_DIR + 'losses_' + job_name + '.npy',
    allow_pickle=True)
176         except Exception as e:
177             print("Problem with loading losses", e)
178             losses = np.array([999.0])
179 
180     padding = 'valid'
181 
182     loss = 100.0
183 
184     model_build_flag = False
185 
186     # build convolutional neural network
187     try:
188         model = create_model(n_conv, n_filters, s_kernel, stride,
    dilation, n_dense, n_cell, activation,
189                              padding=padding, input_shape=input_shape,
    output_shape=output_shape)
190         model_build_flag = True
191 
192     except Exception as e:
193         # print("Probably too big batch size", e)
```

```
194         print("Cannot create model")
195 
196     if model_build_flag:
197 
198         model.compile(loss=loss_function,  # keras.losses.
    mean_squared_error
199                       optimizer=tf.keras.optimizers.Adam(learning_rate=
    learning_rate),  # keras.optimizers.Adam(lr=0.001)
200                       metrics=['accuracy'])
201 
202         if model.count_params() > n_params_max:
203 
204             loss = 10.0
205 
206         else:
207 
208             batch_size = initial_batch_size
209             flag_success = False
210 
211             while not flag_success:
212 
213                 earlystopping = callbacks.EarlyStopping(monitor="loss",
214                                                         mode="auto",
215                                                         patience=patience
    ,
216 
    restore_best_weights=True)
217 
218                 checkpoint = callbacks.ModelCheckpoint(filepath=
    checkpoint_filepath,
219                                                        save_weights_only=
    False,
220                                                        save_best_only=
    True,
221                                                        monitor='val_loss'
    ,
222                                                        mode='min')
223 
224                 try:
225                     history_w_aug = model.fit(train_x, train_y,
226                                               validation_data=(test_x,
    test_y),
227                                               epochs=n_epochs,
228                                               batch_size=batch_size,
    verbose=0,
229                                               callbacks=[earlystopping])
230                 except Exception as e:
231                     print("Probably too big batch size", e)
232                     batch_size = int(batch_size / 2)
233                 else:
234                     flag_success = True
235 
```

```
            predictions = model.predict(test_x)
            mse = tf.keras.losses.MeanSquaredError()
            loss = mse(test_y, predictions).numpy()

            # check if loss is less then other and save
            if loss < np.min(np.array(losses)):
                # save model
                model.save(DRIVE_DIR + 'model_' + job_name + '.h5')

            del model

    # save model

    # Print all the hyperparameters
    print('#################################################################')
    print('loss MSE:', loss)
    print(f"{loss} {n_conv} " +
          f"{' '.join([str(l) for l in n_filters])} " +
          f"{' '.join([str(l) for l in s_kernel])} " +
          f"{' '.join([str(l) for l in stride])} " +
          f"{' '.join([str(l) for l in dilation])} " +
          f"{n_dense} " +
          f"{' '.join([str(l) for l in n_cell])} " +
          f"{' '.join(activation)}")
    print('#################################################################')
    print(' ')

    f_performance = open(DRIVE_DIR + skopt_param_file, 'a')
    f_performance.write(f"{loss} {n_conv} " +
                        f"{' '.join([str(l) for l in n_filters])} " +
                        f"{' '.join([str(l) for l in s_kernel])} " +
                        f"{' '.join([str(l) for l in stride])} " +
                        f"{' '.join([str(l) for l in dilation])} " +
                        f"{n_dense} " +
                        f"{' '.join([str(l) for l in n_cell])} " +
                        f"{' '.join(activation)}\n")
    f_performance.close()

    # k_back.clear_session()
    losses = np.append(losses, loss)
    np.save(DRIVE_DIR + 'losses_' + job_name + '.npy', losses)

    return loss


checkpoint_saver = CheckpointSaver(DRIVE_DIR + skopt_checkpoint_file,
  compress=9)

if load_flag:
    res = load(DRIVE_DIR + skopt_checkpoint_file)
    x0 = res.x_iters
    y0 = res.func_vals
```

```
else:
    # x0 = default_parameters
    x0 = None
    y0 = None

gp_result = gp_minimize(func=fitness,
                        dimensions=dimensions,
                        n_calls=10000,
                        noise=0.01,
                        n_jobs=-1,
                        kappa=5,
                        callback=[checkpoint_saver],
                        x0=x0,
                        y0=y0)
```

Listing A.1: Bayesian Optimisation of NN Parameters

A.4 NFT performance for different modulation and baudrate

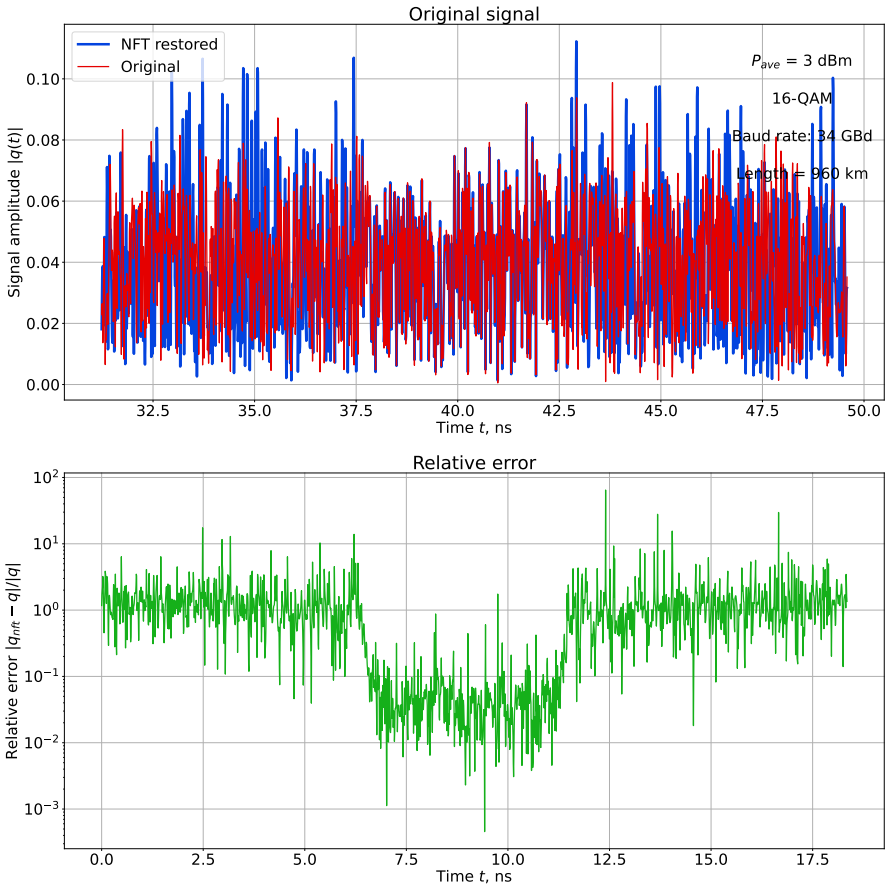


Figure A.3: Example of NFT performance for 16-QAM WDM with 3 dBm average power, 34 GBd symbol frequency and 960 km of SMF. Upper pane shows the signal amplitude while lower pane shows the relative error for NFT restoration.

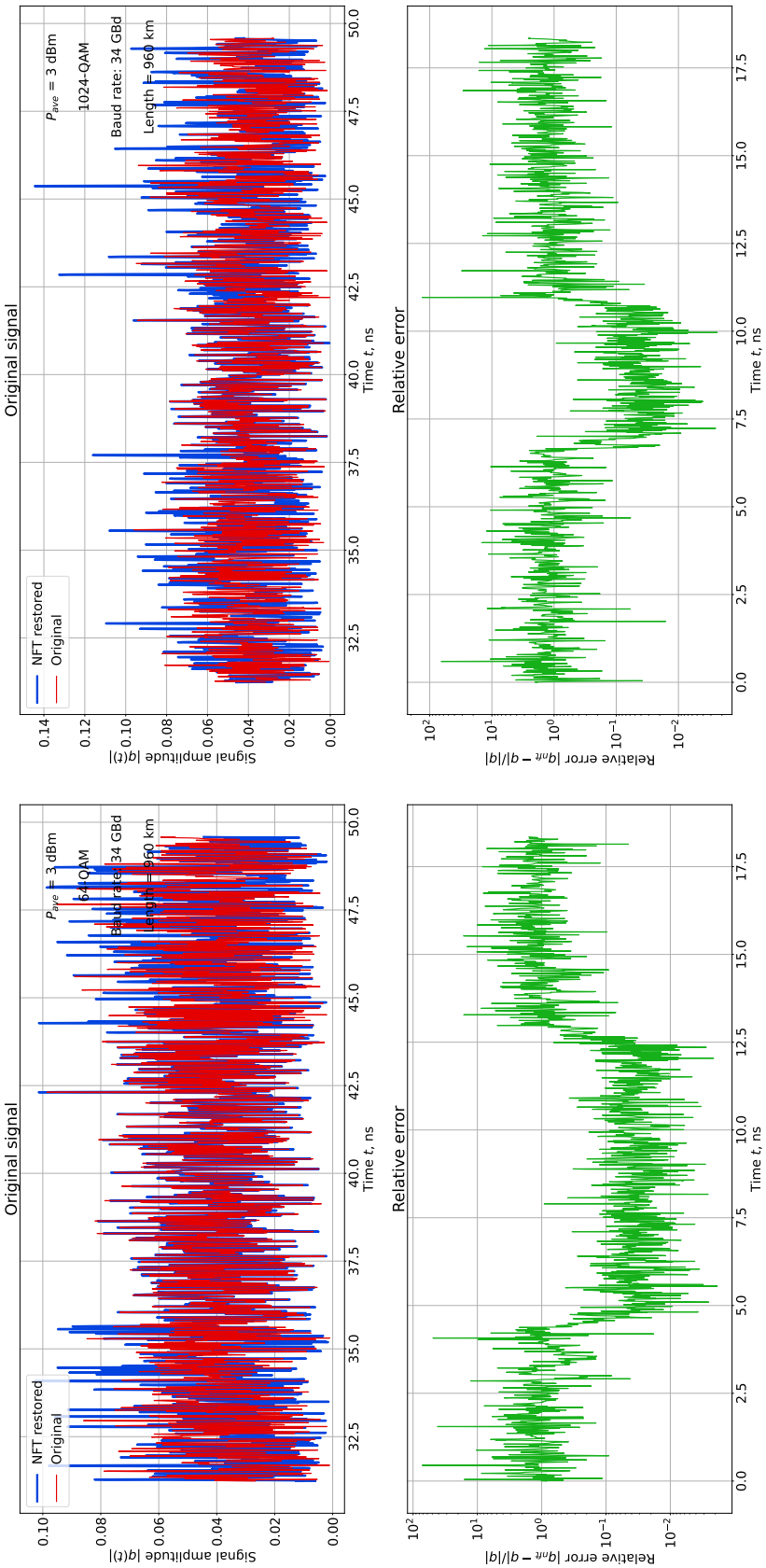


Figure A.4: Same as Fig. A.3 for 64-QAM (left) and 1024-QAM (right).

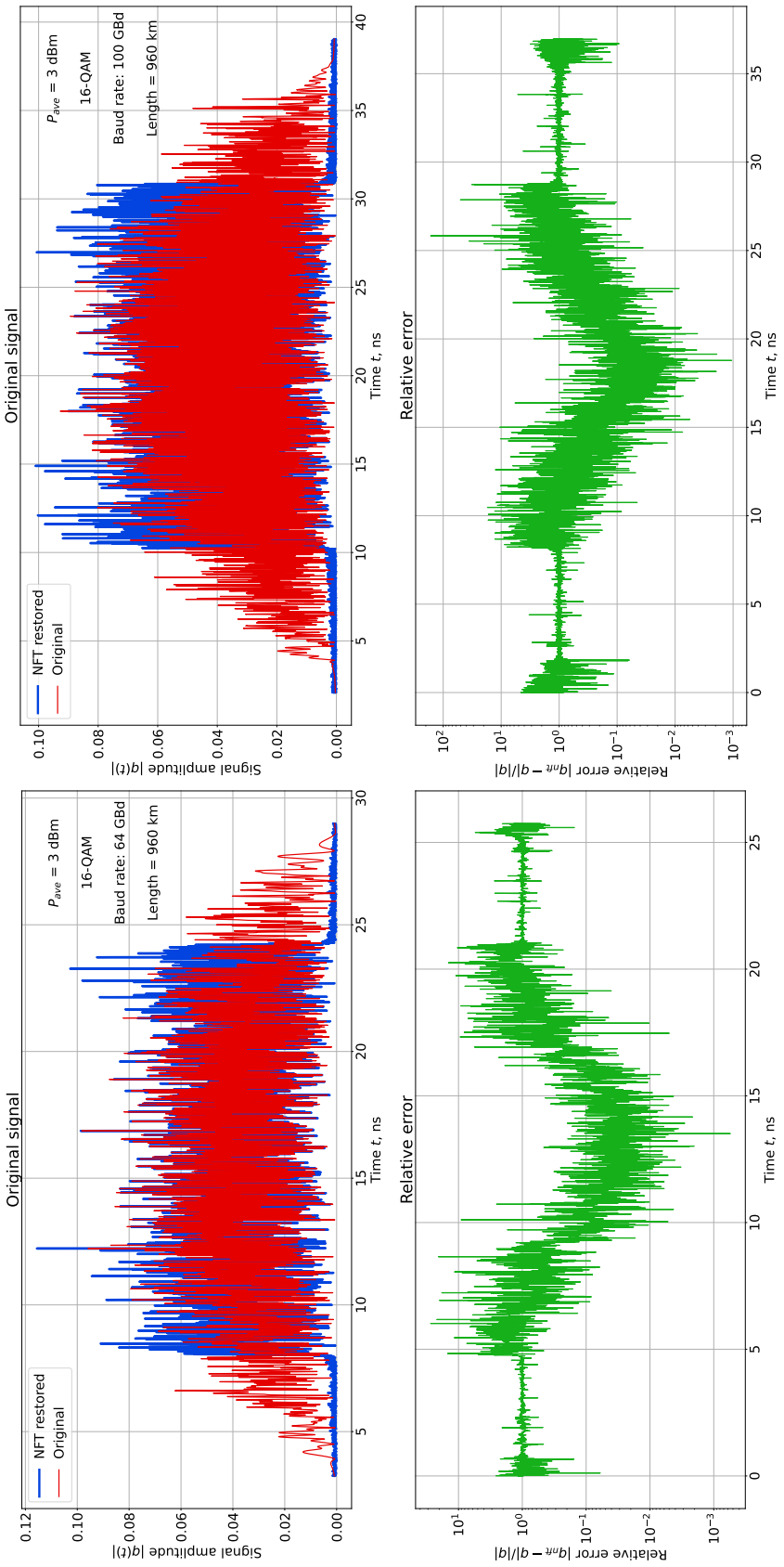


Figure A.5: Same as Fig. A.3 for 16-QAM and 64 GBd (left) and 100 GBd (right).

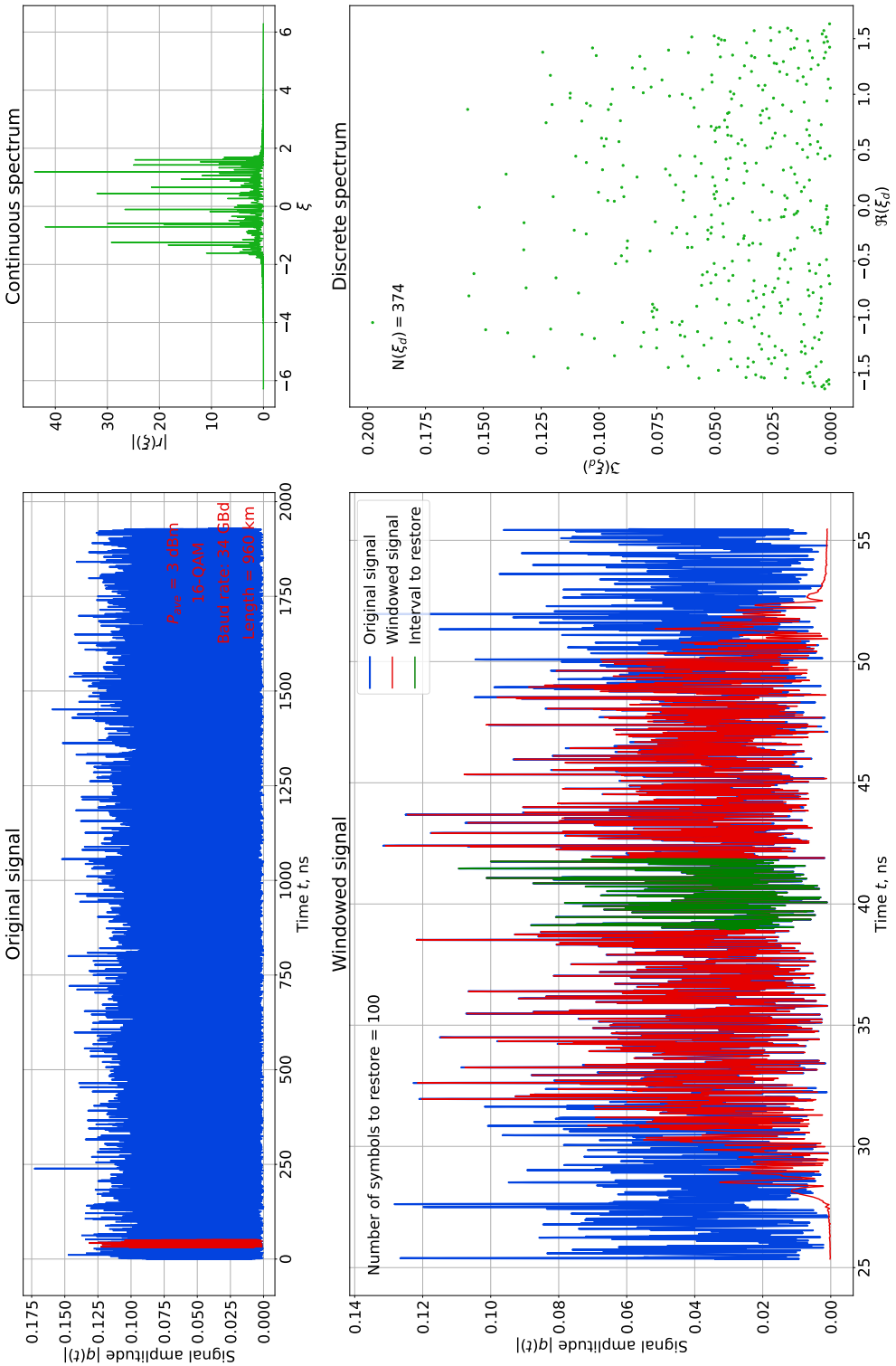


Figure A.6: NF spectrum for WDM signal with 34 GBd and 16-QAM.

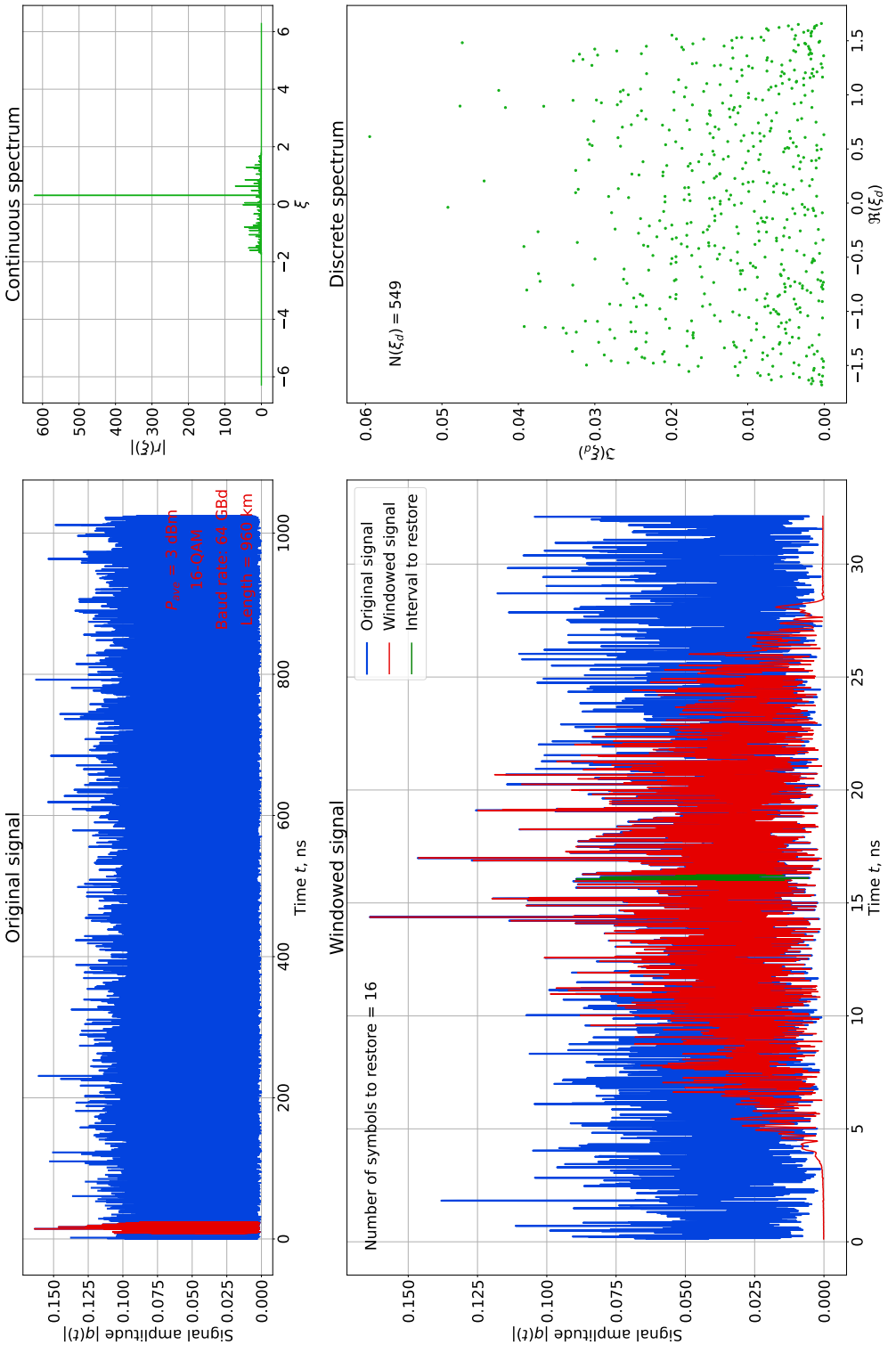


Figure A.7: NF spectrum for WDM signal with 64 GBd and 16-QAM.

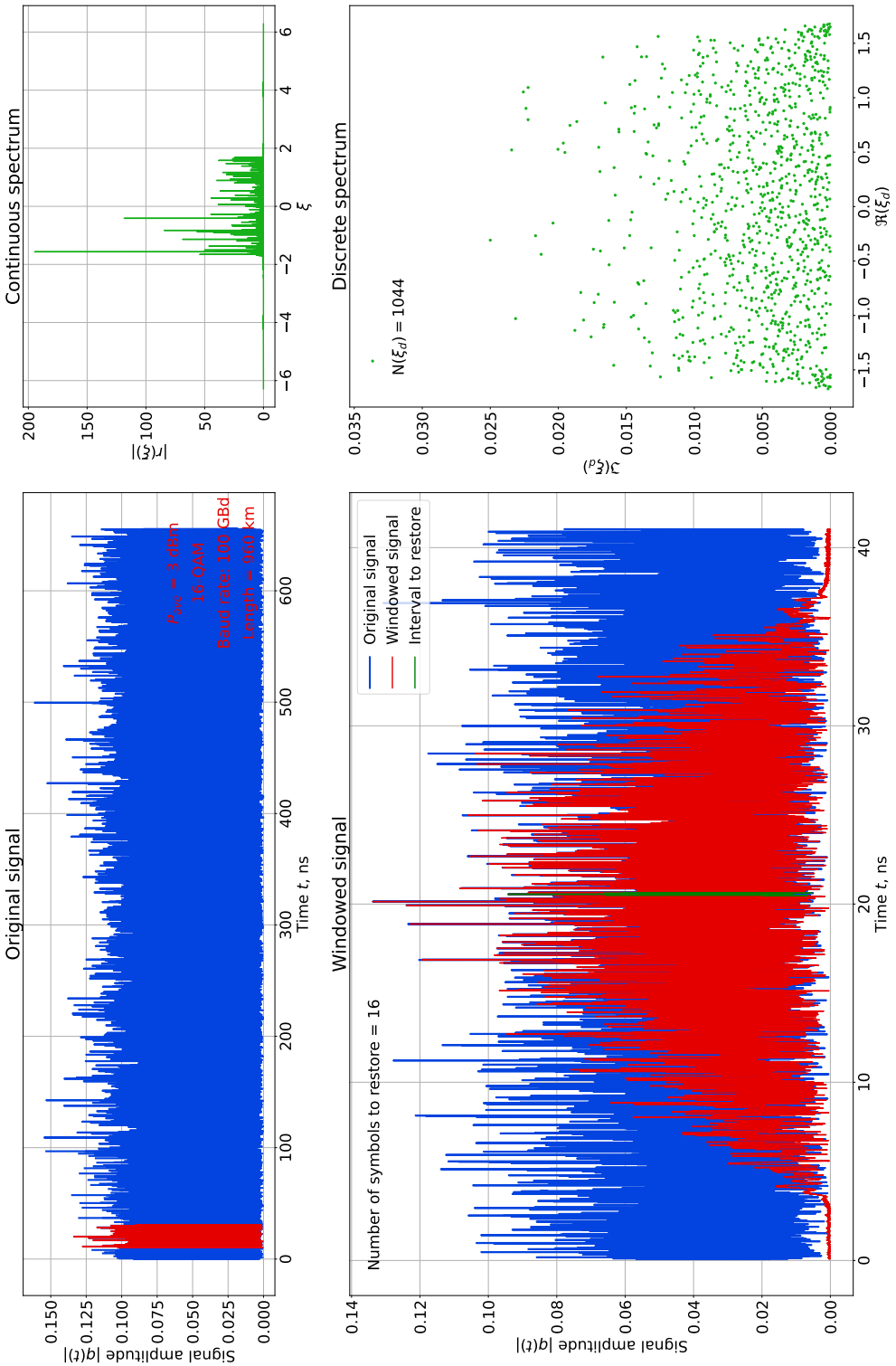


Figure A.8: NF spectrum for WDM signal with 100 GBd and 16-QAM.

B Appendix: Real-Time Demonstration of Optical Communication System Using HpCom Simulator

This interactive demonstration offers a unique, real-time exploration of optical communication system simulations, powered by the High-Performance COMmunication (HpCom) library (Fig. B.1). Participants can witness first-hand the effects of system parameters and signal processing techniques on image quality, transforming a photograph taken at our stand into a vivid illustration of optical communication complexities.

In the rapidly advancing domain of optical communication systems, efficient, high-performance simulation tools are of critical importance. Recognizing the need for a software tailored to the demands of this field, we introduce our GPU-accelerated HpCom library[183]. This framework enhances simulation capabilities, fostering innovation and accelerating research.

The use of GPUs in our library provides significant advantages, such as parallel computations, scalability, and power efficiency. These aspects enable efficient handling of large-scale computations, an essential feature for researchers tackling the computational demands of emerging optical communication systems.

Our demo serves as an educational platform, illustrating how GPU-based simulations allow researchers to explore varied scenarios quickly and efficiently, leading to faster discoveries and more effective resource utilization. As participants interact with the demo, they will gain a tangible understanding of the impacts of system parameter changes and advanced techniques on signal quality.

B.1 Procedure

The primary objective of this demonstration is to showcase the speed of our numerical simulations. To emphasize this, we propose a demonstration procedure in which each stage takes no more than a few dozen seconds, contingent on the specific system parameters.

1. Image Capture and Conversion: A visitor to the stand has their photograph taken. This digital image, composed of a matrix of pixels, is then transformed into binary data. Every

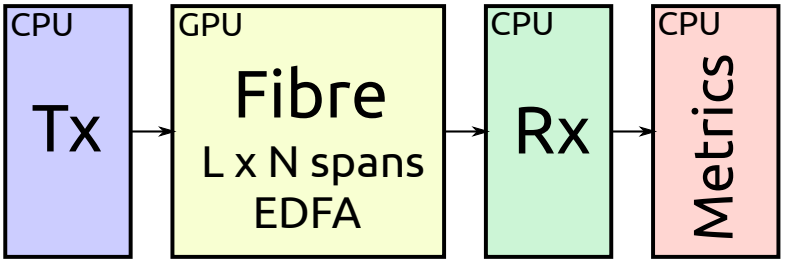


Figure B.1: Framework architecture for optical communication system simulation, featuring optimized transceiver (Tx) design, GPU-accelerated SSFM-based channel model (with N spans of length L), receiver implementation (Rx), and performance metrics evaluation including BER, EVM, and MI.

pixel, composed of red, green, and blue (RGB) colour components, is broken down into its RGB values. These values are then converted into binary format, generating a long bit stream that serves as the payload for the ensuing simulation.

2. Setting Simulation Parameters: The visitor is invited to interact with our intuitive web interface, designed to provide control over various simulation parameters. These parameters include:

`Average power`	The mean optical power launched into the fibre in dBm. Default $P_{ave} = 0$ [dBm].
`Modulation format`	The optical modulation scheme used to encode the data. Possible values are 4-, 16-, 64-, 256- and 1024-QAM.
`Symbol frequency`	Symbol frequency in GHz. Default is 34 [GHz].
`Number of spans`	The number of optical fibre spans. Default is 12.
`Length of spans`	The length of each fibre span. Default is 80 [km].
`Attenuation`	The fibre's attenuation parameter. For SMF $\alpha = 0.2$ [dB/km]
`Nonlinearity`	The fibre's nonlinearity parameter. For SMF $\gamma = 1.2$ $[\mathrm{W}\cdot\mathrm{km}]^{-1}$.
`Noise`	The amplified spontaneous emission (ASE) noise from the erbium-doped fibre amplifiers (EDFAs). Default value is 4.5 [dB]. For system without noise big negative value is used, e.g. -200.
`Dispersion`	The fibre's chromatic dispersion parameter. For SMF $D = 16.8$ ps/[nm$\cdot$ km].
`Simulation step`	Length of simulation step in km. Default is 5 [km].

3. Signal Generation and Propagation: Once the parameters are set, the HpCom library generates the signal according to the selected modulation format and propagates it through the simulated optical fibre. The library employs the Split-Step Fourier method to accurately simulate the combined effects of chromatic dispersion, nonlinearity, and noise.

4. Decoding and Visualization: After the signal propagation, the signal is received on the receiver, where a corresponding matched filter is applied, followed by equalisation and demodulation. The resulting bit stream is transformed back into the RGB image data, which is rendered on the screen. The image is likely to show distortions due to the impairments introduced during signal propagation, as no Forward Error Correction (FEC) is applied.

5. Application of Advanced Techniques: To demonstrate how advanced signal processing techniques can improve the system performance, visitors are given the option to enable these strategies. This includes Machine Learning techniques for impairment mitigation and Digital Back Propagation (DBP) to compensate for the combined effects of chromatic dispersion and nonlinearity. The number of DBP steps per span can also be adjusted for optimal performance.

Throughout the demo, the impact of various parameters and techniques on the resulting image quality is clearly observable, providing visitors with a tangible understanding of optical communication systems' complexity. By comparing the initial and final images, they can directly see the effects of their chosen settings. The demo's real-time and interactive nature makes it a unique educational tool for those interested in optical communications.

B.2 Interface

The practical setup is structured as follows: The React frontend manages the web interface, parameter input, log display, and selection of images. The backend comprises a Telegram bot that receives and stores images sent from a mobile camera, a Flask backend to manage frontend changes and activate the optical channel simulation, and a simulation channel that utilizes the HpCom package for its simulations.

Figure B.3 displays an instance of the interface with specific images and settings. The interface is primarily divided into two main sections: a **visualization section** on the left and an **information and control section** on the right.

Within the interface, the Visualization Section prominently features the "Original Image" at the top-left. This image, captured with a mobile camera, represents the initial input prior to any optical channel simulations. As example, Fig. B.3 captures a scene from the ECOC 2023 GLASGOW conference, featuring two attendees. To maintain their privacy, their faces have been tastefully blurred. Adjacent to and below this original capture are three other images, each denoting a distinct processing technique, notably "prop," "cdc," and "dbp10." The "prop" image consistently displays the aftermath of channel simulation without any equalization applied. Meanwhile, the other images have undergone various equalization

procedures. Specifically, "cdc" reveals the outcome post-chromatic dispersion compensation and nonlinear phase equalization. The labels "dbp2", "dbp3", and "dbp10" depict the results of DBP with 2, 3, and 10 steps per fiber span, respectively. The "nn" label showcases additional nonlinear distortion equalization, employing different neural networks. The precise neural network deployed aligns with the specific fiber parameters. The labels "view1" through "view4" offer alternate perspectives of the constellation post-CDC and NPE.

Transitioning to the Information and Control Section, it encompasses details about both the utilized library and the aforementioned conference. Furthermore, it comprises a simulation parameters segment, a methods dropdown menu allowing users to select the visual representation, and a log console to track operations.

B.3 Example Demonstration

Figure B.2: The left image is the original picture to be transmitted. The central image showcases how the picture appears after signal propagation through an optical system with parameters: $P_{ave} = 0$ [dBm], 12×80 [km] spans of SSFM. The fiber was characterized by an attenuation coefficient of $\alpha = 0$ [dB/km], EDFA noise figure 4.5 [dB], a dispersion coefficient of $D = 16.8$ ps/[nm$\cdot$km], and a nonlinear coefficient of $\gamma = 1.2$ [W$\cdot$km]$^{-1}$. The image on the right demonstrates the when $P_{ave} = 5$ [dBm].

As illustrated in Fig. B.2, we demonstrate the process of image corruption due to signal propagation with specific system parameters. The original image of a parrot (left) undergoes signal propagation with parameters: 12×80 [km] spans of standard single-mode fiber (SSFM), attenuation coefficient of $\alpha = 0$ [dB/km], EDFA noise figure 4.5 [dB], a dispersion coefficient of $D = 16.8$ ps/[nm$\cdot$km], and a nonlinear coefficient of $\gamma = 1.2$ [W$\cdot$km]$^{-1}$. The resultant image (centre) exhibits the distortions introduced by these system parameters. Increasing the P_{ave} to 5 [dBm] (right image) further alters the image quality, showing how different power levels can impact the quality of the received signal.

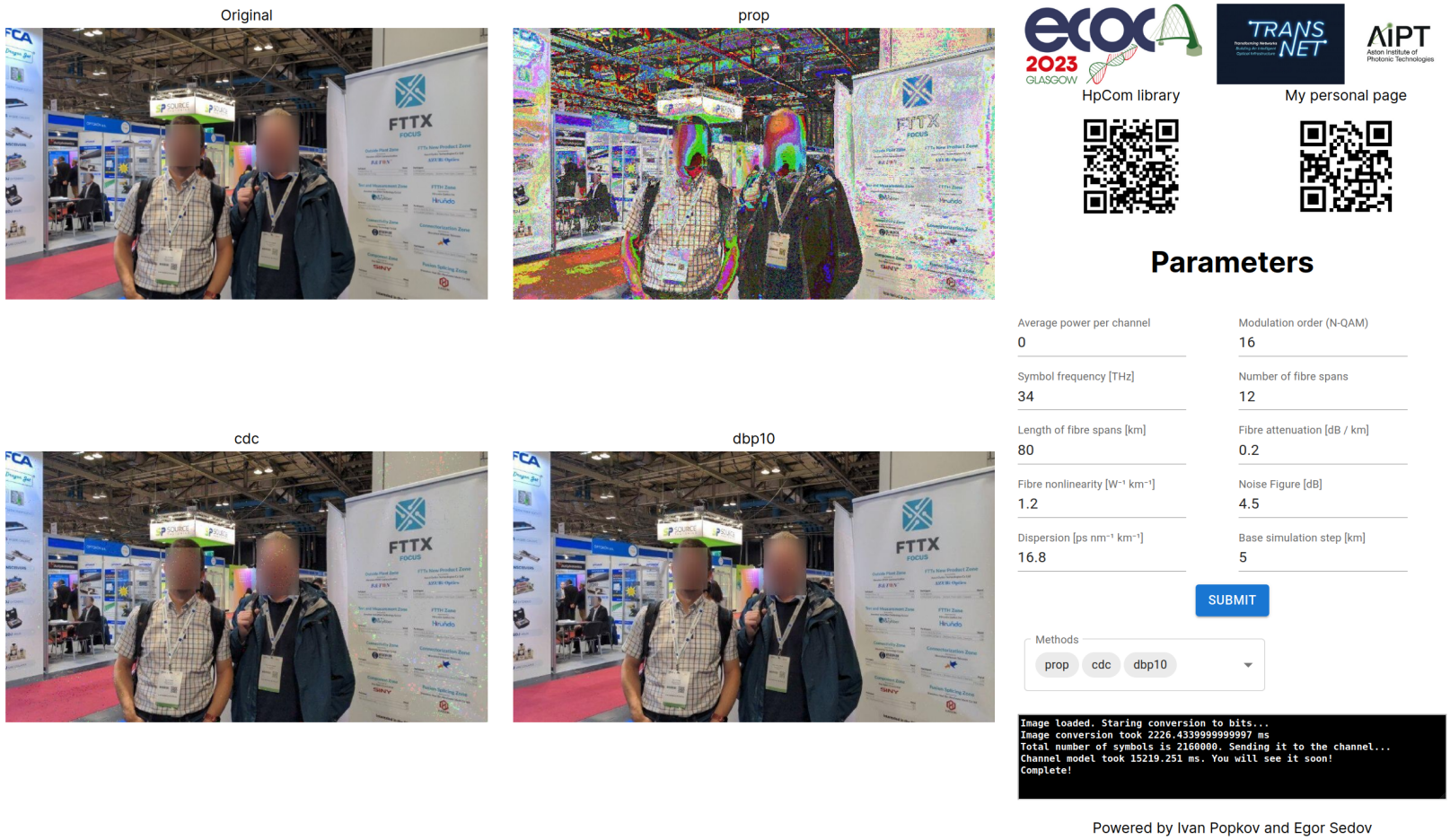


Figure B.3: ECOC2023 GLASGOW Demo Interface: Left side displays 4 spans for image selection, while the right side offers sections for simulation parameter adjustments, mode selection, and a log console.

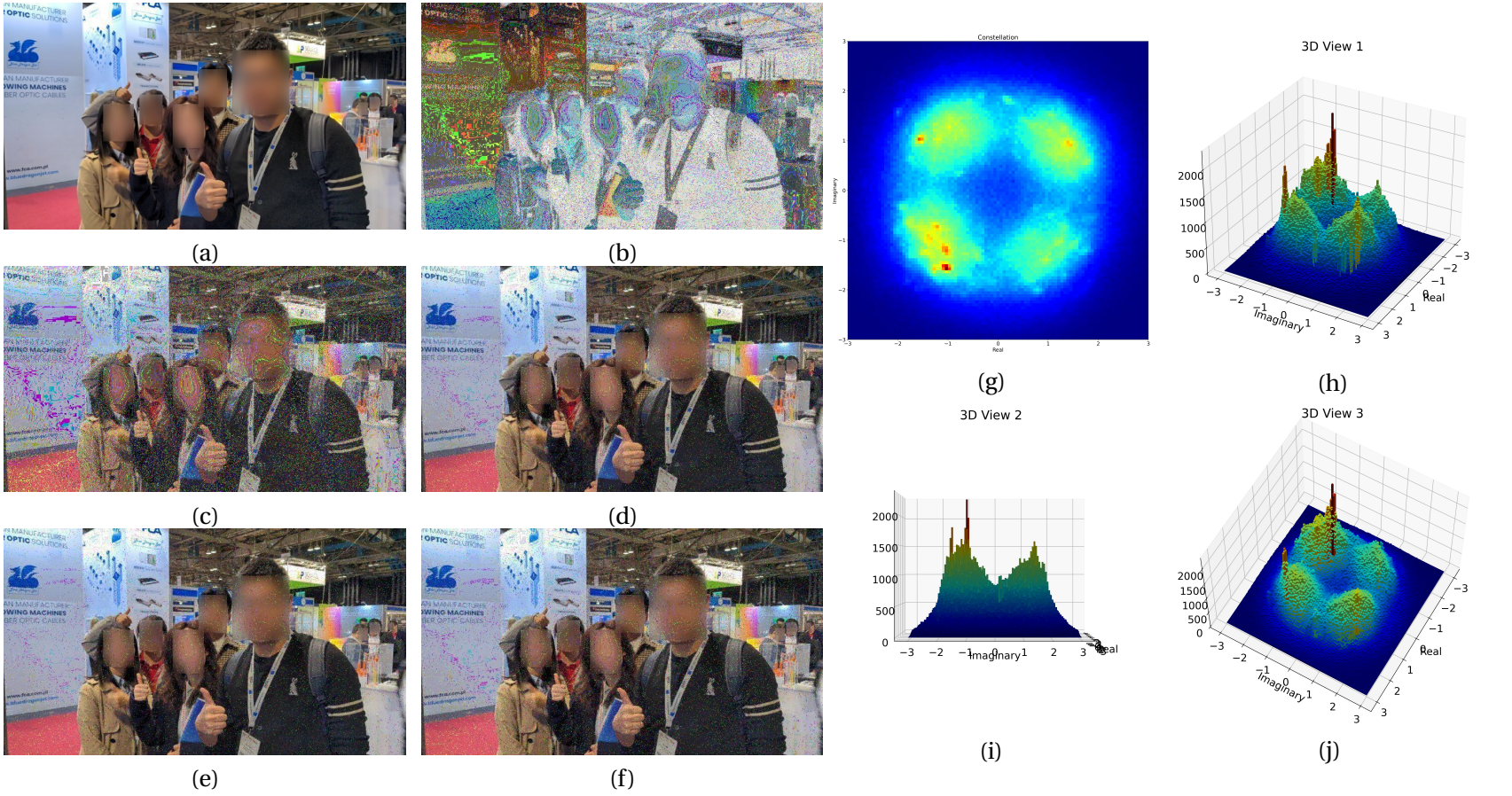


Figure B.4: Signal propagation with parameters: 10×50 [km] spans of TWC fibre, attenuation coefficient of $\alpha = 0.2$ [dB/km], EDFA noise figure 4.5 [dB], a dispersion coefficient of $D = 2.8$ ps/[nm$\cdot$km], and a nonlinear coefficient of $\gamma = 2.5$ [W$\cdot$km]$^{-1}$, average signal power $P_{ave} = 5$ dBm, QPSK format. **(a)** Original image. **(b)** Image post-propagation without equalisation. **(c)** Image after CDC and NPE. **(d)** and **(e)** DBP with 2 and 3 steps per span, respectively. **(f)** Image after NN equalisation for received symbols. **(g)** to **(j)** Various constellation diagram representations.

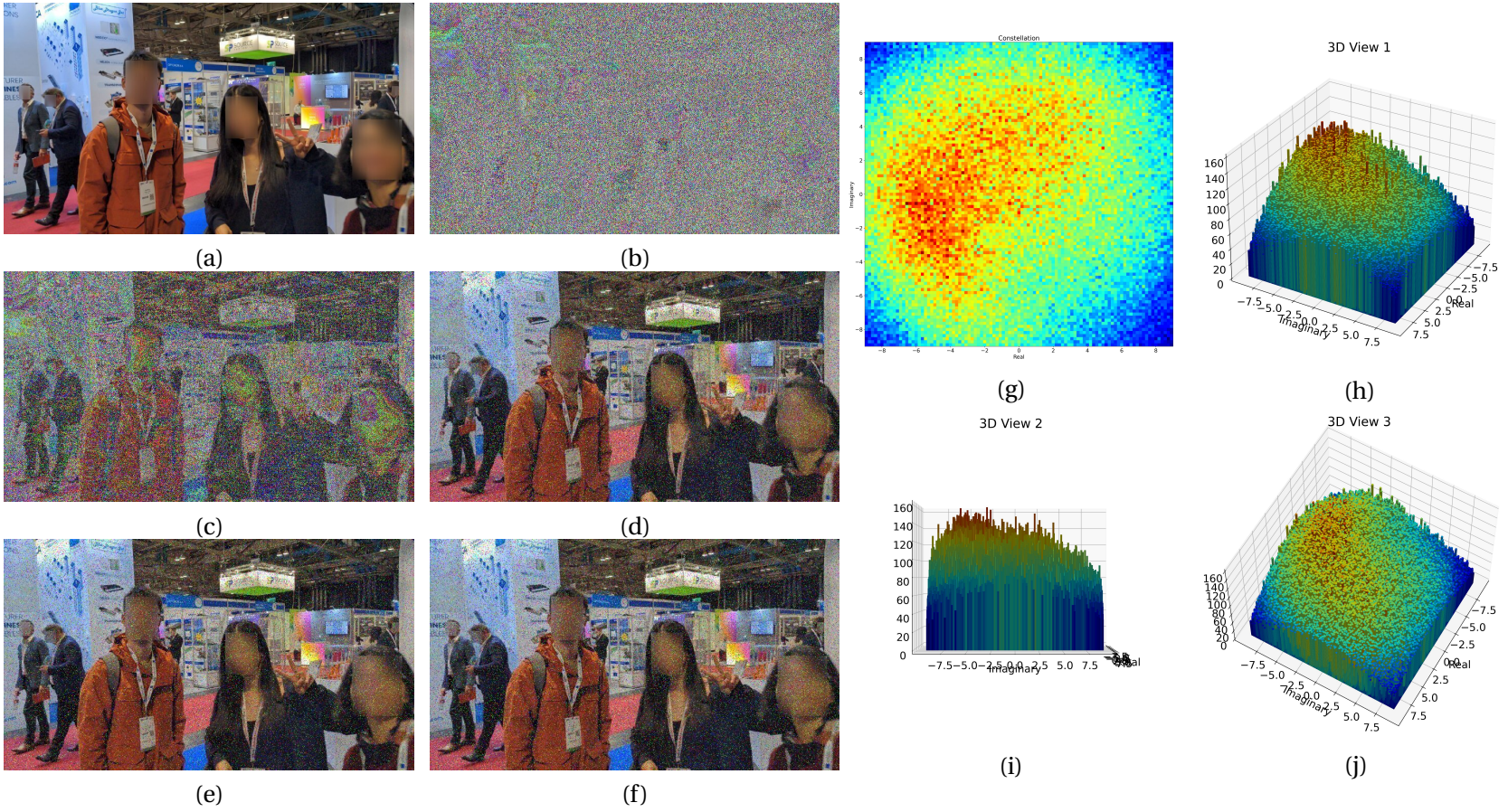


Figure B.5: Signal propagation with parameters: 6×80 [km] spans of SMF, attenuation coefficient of $\alpha = 0.2$ [dB/km], EDFA noise figure 4.5 [dB], a dispersion coefficient of $D = 16.8$ ps/[nm$\cdot$km], and a nonlinear coefficient of $\gamma = 1.2$ [W$\cdot$km]$^{-1}$, average signal power $P_{ave} = 10$ dBm, 64-QAM format. **(a)** Original image. **(b)** Image post-propagation without equalisation. **(c)** Image after CDC and NPE. **(d)** and **(e)** DBP with 2 and 3 steps per span, respectively. **(f)** Image after NN equalisation for received symbols. **(g)** to **(j)** Various constellation diagram representations.

Figure B.4 illustrates signal propagation through a system with the following parameters: spans of 10×50 [km] using TWC fibre, an attenuation coefficient $\alpha = 0.2$ [dB/km], an EDFA noise figure of 4.5 [dB], a dispersion coefficient $D = 2.8$ ps/[nm$\cdot$km], and a nonlinear coefficient $\gamma = 2.5$ [W$\cdot$km]$^{-1}$. The average signal power is set at $P_{ave} = 5$ dBm, and the modulation format is QPSK. The figure showcases various stages of the signal, from the original image to different processing methods, including propagation without equalisation, CDC and NPE application, DBP with different steps per span, and equalisation using NN. Additionally, it features different constellation diagram representations.

Figure B.5 presents signal propagation in a setup characterized by: spans of 6×80 [km] using SMF, an attenuation coefficient $\alpha = 0.2$ [dB/km], an EDFA noise figure of 4.5 [dB], a dispersion coefficient $D = 16.8$ ps/[nm$\cdot$km], and a nonlinear coefficient $\gamma = 1.2$ [W$\cdot$km]$^{-1}$. The average signal power stands at $P_{ave} = 10$ dBm, utilizing a 64-QAM format. The visualizations in the figure encompass the original image and its evolutions after several processing techniques, such as post-propagation without equalisation, the effects of CDC and NPE, DBP at varying steps per span, and NN equalisation. The figure concludes with different views of the constellation diagram.

B.4 Conclusion

Through this live demonstration, participants can explore the capabilities and benefits of the HpCom library and better understand the impact of various parameters and techniques on signal quality in optical communication systems. By directly observing the results of their choices, participants will gain a better understanding of the complexities of optical signal transmission and the opportunities for improvement.

C Appendix: HpCom Code Examples

C.1 Defining the Parameters

The following list details the elements of the WDM parameters dictionary created by the `create_wdm_parameters` function:

`n_channels`	Number of Wavelength Division Multiplexing channels.
`channel_spacing`	Channel spacing in Hz.
`n_polarisations`	Number of polarizations.
`p_ave_dbm`	Average power in dBm.
`n_symbols`	Number of symbols.
`m_order`	Modulation order.
`modulation_type`	Type of modulation, derived from modulation order.
`n_bits_symbol`	Number of bits per symbol, derived from modulation type.
`roll_off`	Roll-off factor for Root Raised Cosine filter.
`upsampling`	Upsampling factor.
`downsampling_rate`	Downsampling rate.
`symb_freq`	Symbol frequency in Hz.
`sample_freq`	Sampling frequency in Hz, calculated as the product of symbol frequency and upsampling factor.
`np_filter`	Nyquist pulse filtering.
`p_ave`	Average power in Watts, calculated from `p_ave_dbm`.

`seed`	Seed for random number generator.
`scale_coef`	Scale coefficient for constellation, derived based on modulation type, average power, and number of polarizations.

Below is an example of how the `create_wdm_parameters` function can be used with some specific parameters:

```
# Import the function
from hpcom.signal import create_wdm_parameters

# Define the parameters
n_channels = 4
p_ave_dbm = -3
n_symbols = 2**15
m_order = 16
roll_off = 0.2
upsampling = 4
downsampling_rate = 2
symb_freq = 32e9
channel_spacing = 50e9

# Call the function
wdm_params = create_wdm_parameters(
    n_channels, p_ave_dbm, n_symbols, m_order,
    roll_off, upsampling, downsampling_rate,
    symb_freq, channel_spacing
)

# Now wdm_params contains the WDM parameters dictionary
```

Listing C.1: Usage of create_wdm_parameters function

In example C.1, the `create_wdm_parameters` function is called with specific values for the parameters, and the resulting dictionary is stored in the `wdm_params` variable.

The functions `get_default_wdm_parameters` and `get_default_channel_parameters` furnish the capability to generate default parameters for WDM signal creation and SSMF channel with EDFA amplification, respectively.

```
def get_default_wdm_parameters():

 wdm = {}
 wdm['n_channels'] = 1  # Single WDM channel
 wdm['channel_spacing'] = 75e9  # 75 GHz
 wdm['n_polarisations'] = 2  # 2 polarisations - Manakov equation
 wdm['p_ave_dbm'] = 0  # dBm
 wdm['n_symbols'] = 2 ** 15
 wdm['m_order'] = 16  # 16-QAM
 wdm['roll_off'] = 0.1  # 0.1 roll-off for RRC filter
 wdm['upsampling'] = 8  # 8 times oversampling
```

```
    wdm['downsampling_rate'] = 1
    wdm['symb_freq'] = 64e9  # 64 GHz
    wdm['sample_freq'] = int(wdm['symb_freq'] * wdm['upsampling'])  #
     sampling frequency
    wdm['p_ave'] = (10 ** (wdm['p_ave_dbm'] / 10)) / 1000  # Average power
     in Watts
    wdm['modulation_type'] = get_modulation_type_from_order(wdm['m_order'])
      # 16-QAM
    wdm['n_bits_symbol'] = get_n_bits(wdm['modulation_type'])  # 4 bits per
     symbol
    wdm['seed'] = 'fixed'  # fixed seed for random number generator
    wdm['scale_coef'] = get_scale_coef_constellation(wdm['modulation_type'])
      / np.sqrt(wdm['p_ave'] / wdm['n_polarisations'])  # scale coefficient
     for constellation

    return wdm
```

Listing C.2: Default WDM parameters

```
def get_default_channel_parameters():

    channel = {}
    channel['n_spans'] = 12  # Number of spans
    channel['z_span'] = 80  # Span Length [km]
    channel['alpha_db'] = 0.225  # Attenuation coefficient [dB km^-1]
    channel['alpha'] = channel['alpha_db'] / (10 * np.log10(np.exp(1)))
    channel['gamma'] = 1.2  # Non-linear Coefficient [W^-1 km^-1]. Default =
      1.2
    channel['noise_figure_db'] = 4.5  # Noise Figure [dB]. Default = 4.5
    channel['noise_figure'] = 10 ** (channel['noise_figure_db'] / 10)
    channel['gain'] = np.exp(channel['alpha'] * channel['z_span']) # gain
     for one span
    channel['dispersion_parameter'] = 16.8 #  [ps nm^-1 km^-1]  dispersion
     parameter
    channel['beta2'] = -(1550e-9 ** 2) * (channel['dispersion_parameter'] *
     1e-3) / (2 * np.pi * 3e8)  # conversion to beta2 - Chromatic Dispersion
      Coefficient [s^2 km^-1]
    channel['beta3'] = 0
    channel['h_planck'] = 6.62607015e-34  # Planck's constant [J/s]
    channel['fc'] = 299792458 / 1550e-9  # carrier frequency
    channel['dz'] = 1.0  # length of the step for SSFM [km]
    channel['nz'] = int(channel['z_span'] / channel['dz'])  # number of
     steps per each span
    channel['noise_density'] = channel['h_planck'] * channel['fc'] * (
     channel['gain'] - 1) * channel['noise_figure']
    channel['seed'] = 'fixed'

    return channel
```

Listing C.3: Default channel parameters

C.2 Typical fibre parameters

Table C.1: Typical parameters for LEAF, TWC, and SMF fibers at 1550 nm.

Fiber Type	Dispersion ps/[nm·km]	Attenuation [dB/km]	Nonlinearity $[W \cdot km]^{-1}$
LEAF (1550 nm)	4 – 8	0.2 – 0.25	0.8 – 1.2
TWC (1550 nm)	2.5 – 3.5	0.2 – 0.25	2 – 3
SMF (1550 nm)	16.5 – 17.5	0.2 – 0.25	1.1 – 1.5

C.3 Appropriate scaling for constellation points

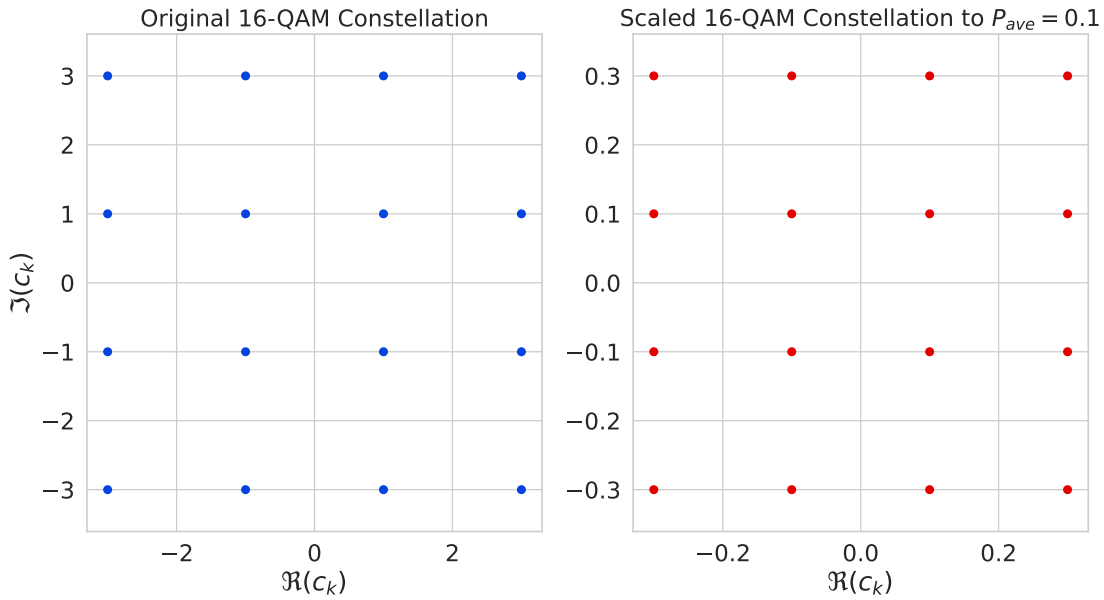


Figure C.1: A visualization displaying two sets of 16-QAM constellations on a complex plane. The left plot shows the original constellation, marked in blue, where each point represents a unique symbol. The right plot illustrates the scaled constellation, colored in red, which has been normalized to achieve a specified average power level of 0.1.

Constellation points normalization is a technique used in digital signal processing, specifically in the context of modulation in communications, to ensure that the average power of the constellation points matches a desired level. This process is particularly important in systems like Wavelength Division Multiplexing (WDM), where each signal's power needs to be controlled to avoid interference and optimize the transmission.

The average power of a constellation is computed as the mean of the squared magnitudes of

all the constellation points. In the Python code provided, the constellation points are obtained and then scaled to match the desired average power, P_{ave}, which is an attribute of the WDM system.

The scaling factor is derived by dividing the square root of P_{ave} by the square root of the actual average power of the unscaled constellation. This ensures that after scaling, the constellation will have the required average power. The functions `get_scale_coef_constellation` and `get_sq_of_average_power` are used to compute the scaling factor based on the modulation type.

Let c_k represent the constellation points for a given modulation scheme, and P_{ave} denote the desired average power for the WDM system. The scaling process can be expressed as:

$$c_{k,\text{scaled}} = c_k \cdot \frac{\sqrt{P_{\text{ave}}}}{\sigma_c}$$

where σ_c is the scaling factor for constellation and defined as:

$$\sigma_c = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |c_i|^2}$$

Here, N is the number of constellation points, and $|c_i|$ is the magnitude of the i-th constellation point. This normalization ensures that the average power of the constellation is equal to P_{ave}.

The value

$$\frac{\sqrt{P_{\text{ave}}}}{\sigma_c}$$

is the scaling coefficient that is used to adjust the constellation from a standard scale (as shown in Fig. C.1) to the desired power level.

C.4 Simulation Script for Data Mining

The script C.4 serves as a detailed example of simulating and analyzing optical communication systems. It begins by importing necessary libraries such as TensorFlow for GPU memory management, Pandas for data handling, and HpCom for signal generation and channel modeling. By specifying a directory for data storage and a job name, it ensures organized handling of different simulation batches. The script outlines a variety of system, signal, and channel parameters, setting the stage for a range of simulation scenarios. Through nested loops, it iterates over various configurations of these parameters, each time setting up the signal and channel conditions, running the simulation, and collecting key metrics like bit error rate and Q-factor, along with the transmitted and received points. This data is compiled into a Pandas DataFrame, which is then saved to a Pickle file for each set of channel count

and average power values, ensuring organized storage of simulation outcomes. The script also demonstrates efficient management of GPU memory allocation, which is crucial in multi-GPU setups. By conducting multiple runs with identical parameters, it accumulates a significant amount of data, addressing the limitations posed by restricted GPU memory. A 'done' message at the end signifies the successful completion of the script, exemplifying a structured approach to exploring and analyzing various configurations in optical communication systems. This script can serve as a robust framework for those looking to conduct thorough analyses of optical communication systems under varying conditions.

```
# Import of necessary packages
import tensorflow as tf  # tensorflow used for GPU memory allocation
import pandas as pd  # pandas used for data storage
import hpcom  # hpcom used for signal generation and channel modelling
from tqdm import tqdm  # tqdm used for progress bar

# Directory with data files for Linux and Windows
data_dir = "/home/username/data/"
# data_dir = 'C:/Users/username/data/'

# Name of the job to store data for different parameters
job_name = 'example'

# System parameters
GPU_MEM_LIMIT = 1024 * 6  # 6 GB of GPU memory is allocated

# Signal parameters
n_polarisations = 1  # number of polarisations. Can be 1 for NLSE and 2
  for Manakov
n_symbols = 2 ** 18  # number of symbols to be transmitted for each run
m_order = 16  # modulation order (16-QAM, 64-QAM, etc.)
symb_freq = 34e9  # symbol frequency
channel_spacing = 75e9  # channel spacing
roll_off = 0.1  # roll-off factor for RRC filter
upsampling = 16  # upsampling factor
downsampling_rate = 1  # downsampling rate
p_ave_dbm_list = [-2, -1, 0, 1, 2]  # list of average power values in dBm

# Channel parameters
z_span = 80  # span length in km
n_channels_list = [1, 7, 15]  # number of WDM channels
n_span_list = [6, 8, 10, 12, 14]  # 480, 640, 800, 960, 1120
noise_figure_db_list = [-200, 4.5]  # list of noise figure values in dB.
  -200 means no noise
alpha_db = 0.2  # attenuation coefficient in dB/km
gamma = 1.2  # nonlinearity coefficient
dispersion_parameter = 16.8  # dispersion parameter in ps/nm/km
dz = 1  # step size in km

# Simulation parameters
n_runs = 64  # number of runs for each parameter set
verbose = 0  # verbose level. 0 - no print, 3 - print all system logs
```

```
seed = 'time'  # seed for random number generator. 'time' - use current time, 'fixed' - fixed seed
channels_type = 'middle'  # type for which of WDM channels all metrics will be calculated. 'middle' - middle channel, 'all' - all channels

# GPU memory allocation
gpus = tf.config.list_physical_devices('GPU')
print("Num GPUs Available: ", len(gpus), gpus)
if gpus:
 # Restrict TensorFlow to only allocate 4GB of memory on the first GPU
 try:
  tf.config.set_logical_device_configuration(
   gpus[0],
   [tf.config.LogicalDeviceConfiguration(memory_limit=GPU_MEM_LIMIT)])
  logical_gpus = tf.config.list_logical_devices('GPU')
  print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
 except RuntimeError as e:
  # Virtual devices must be set before GPUs have been initialized
  print(e)

for n_channels in n_channels_list:
 for p_ave_dbm in p_ave_dbm_list:

  # Create an empty dataframe
  df = pd.DataFrame()
  for noise_figure_db in noise_figure_db_list:

   for n_span in n_span_list:
    for run in tqdm(range(n_runs)):
     print(f'run = {run} / n_channels = {n_channels} / '
        f'p_dbm = {p_ave_dbm} / n_span = {n_span} / '
        f'noise = {noise_figure_db}')

     # Signal parameters
     wdm_full = hpcom.signal.create_wdm_parameters(n_channels=n_channels, p_ave_dbm=p_ave_dbm,
                 n_symbols=n_symbols, m_order=m_order,
                 roll_off=roll_off, upsampling=upsampling,
                 downsampling_rate=downsampling_rate,
                 symb_freq=symb_freq,
                 channel_spacing=channel_spacing,
                 n_polarisations=n_polarisations, seed=seed)

     # Channel parameters
     channel_full = hpcom.channel.create_channel_parameters(n_spans=n_span,
                    z_span=z_span,
                    alpha_db=alpha_db,
                    gamma=gamma,
                    noise_figure_db=noise_figure_db,
                    dispersion_parameter=dispersion_parameter,
                    dz=dz)
```

```
89
90      # Run the simulation
91      result_channel = hpcom.channel.full_line_model_wdm_new(channel_full,
     wdm_full,
92                        channels_type=channels_type,
93                        verbose=verbose)
94
95      # Store the results
96      result_dict = {}
97
98      result_dict['run'] = run  # run number
99      result_dict['n_channels'] = n_channels  # number of WDM channels
100     result_dict['n_polarisations'] = n_polarisations  # number of
    polarisations
101     result_dict['n_symbols'] = n_symbols  # number of symbols
102     result_dict['p_ave_dbm'] = p_ave_dbm  # average power in dBm
103     result_dict['z_km'] = n_span * 80  # total distance in km
104     result_dict['scale_coef'] = wdm_full['scale_coef']  # scale
    coefficient for signal generation
105
106     result_dict['noise_figure_db'] = channel_full['noise_figure_db']  #
    noise figure in dB
107     result_dict['gamma'] = channel_full['gamma']  # nonlinearity
    coefficient
108     result_dict['z_span'] = channel_full['z_span']  # span length in km
109     result_dict['dispersion_parameter'] = channel_full['
    dispersion_parameter']  # dispersion parameter in ps/nm/km
110     result_dict['dz'] = channel_full['dz']  # step size in km
111
112     result_dict['points_orig'] = result_channel['points_orig']  #
    original points
113     result_dict['points'] = result_channel['points']  # points after
    transmission
114     result_dict['points_shifted'] = result_channel['points_shifted']  #
    points after transmission, cromatic dispersion compensation and phase
    shifting
115
116     result_dict['ber'] = result_channel['ber']  # bit error rate
117     result_dict['q'] = result_channel['q']  # Q-factor
118     result_dict['evm'] = result_channel['evm']  # error vector magnitude
119     result_dict['mi'] = result_channel['mi']  # mutual information
120
121     # Store the results in a dataframe
122     df = pd.concat([df, pd.DataFrame(result_dict)], ignore_index=True)
123
124 # Store the dataframe in a pickle file
125 df.to_pickle(data_dir + 'data_collected_' + job_name + '_nch_' + str(
    n_channels) +
126      '_pavedbm_' + str(p_ave_dbm) + '.pkl')
127
128 # Finish the script
```

```
129 print('done')
```

Listing C.4: Example of data generation for range of parameters

Bibliography

[1] Tarsem Lal Singal. *Optical fiber communications: principles and applications*. Cambridge University Press, 2017.

[2] Peter J. Winzer, David T. Neilson, and Andrew R. Chraplyvy. "Fiber-optic transmission and networking: the previous 20 and the next 20 years". In: *Opt. Express* 26.18 (Sept. 2018), pp. 24190–24239. DOI: 10.1364/OE.26.024190. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-26-18-24190.

[3] Huawei. *Nine Key Challenges Facing Optical Communications in the Next 10 Years*. 2023. URL: https://www.huawei.com/ch-en/huaweitech/publication/93/nine-challgenges-optical-communications-next-decade.

[4] Sergei K Turitsyn et al. "Nonlinear Fourier transform for optical data processing and transmission: advances and perspectives". In: *Optica* 4.3 (2017), pp. 307–322. DOI: 10.1364/OPTICA.4.000307.

[5] Supreet Kaur et al. "Recent trends in wireless and optical fiber communication". In: *Global Transitions Proceedings* 3.1 (2022), pp. 343–348.

[6] A. Hasegawa and F. Tappert. "Transmission of stationary nonlinear optical pulses in dispersive dielectric fibers. I. Anomalous dispersion". In: *Appl. Phys. Lett.* 23 (1973).

[7] Natanael Karjanto. "The nonlinear Schr\" odinger equation: A mathematical model with its wide-ranging applications". In: *arXiv preprint arXiv:1912.10683* (2019).

[8] Victor M Perez-Garcia et al. "Low energy excitations of a Bose-Einstein condensate: A time-dependent variational analysis". In: *Physical review letters* 77.27 (1996), p. 5320.

[9] Victor M Perez-Garcia et al. "Dynamics of Bose-Einstein condensates: Variational solutions of the Gross-Pitaevskii equations". In: *Physical Review A* 56.2 (1997), p. 1424.

[10] Embrecht van Groesen, N Karjanto, et al. "Displaced phase-amplitude variables for waves on finite background". In: *Physics letters A* 354.4 (2006), pp. 312–319.

[11] Natanael Karjanto. "Mathematical Aspects of Extreme Water Waves". In: (2006).

[12] N Karjanto, E Van Groesen, et al. "Extreme wave phenomena in down-stream running modulated waves". In: *Applied mathematical modelling* 31.7 (2007), pp. 1425–1443.

[13] Karen L Henderson, D Howell Peregrine, and John W Dold. "Unsteady water wave modulations: fully nonlinear solutions and comparison with the nonlinear Schrödinger equation". In: *Wave motion* 29.4 (1999), pp. 341–361.

[14] Efim Pelinovsky, Christian Kharif, et al. *Extreme ocean waves*. Vol. 1495. Springer, 2008.

[15] Christian Kharif, Efim Pelinovsky, and Alexey Slunyaev. *Rogue waves in the ocean*. Springer Science & Business Media, 2008.

[16] Roger K Dodd et al. "Solitons and nonlinear wave equations". In: (1982).

[17] Philip G Drazin and Robin Stanley Johnson. *Solitons: an introduction*. Vol. 2. Cambridge university press, 1989.

[18] J Frenkel. "On the theory of plastic deformation and twinning". In: *J. Phys.* 1 (1939), pp. 137–149.

[19] JK Perring and THR138393 Skyrme. "A model unified field equation". In: *Nuclear Physics* 31 (1962), pp. 550–555.

[20] JD Gibbon, IN James, and Irene M Moroz. "An example of soliton behaviour in a rotating baroclinic fluid". In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 367.1729 (1979), pp. 219–237.

[21] Brian David Josephson. "Possible new effects in superconductive tunnelling". In: *Physics letters* 1.7 (1962), pp. 251–253.

[22] BRIAN D Josephson. "The discovery of tunnelling supercurrents". In: *Reviews of Modern Physics* 46.2 (1974), p. 251.

[23] RD Parmentier. "Solitons and long Josephson junctions". In: *The new superconducting electronics*. Springer, 1993, pp. 221–248.

[24] David James Kaup and Alan C Newell. "Theory of nonlinear oscillating dipolar excitations in one-dimensional condensates". In: *Physical Review B* 18.10 (1978), p. 5162.

[25] Niels Grobech-Jensen, Yuri S Kivshar, Mogens R Samuelsen, et al. "Nonlinear dynamics of a parametrically driven sine-Gordon system". In: *Physical Review B* 47.9 (1993), p. 5013.

[26] Sven Gnutzmann and Uzy Smilansky. "Quantum graphs: Applications to quantum chaos and universal spectral statistics". In: *Advances in Physics* 55.5-6 (2006), pp. 527–625.

[27] Peter Kuchment. "Quantum graphs: an introduction and a brief survey". In: *arXiv preprint arXiv:0802.3442* (2008).

[28] Gregory Berkolaiko and Peter Kuchment. *Introduction to quantum graphs*. 186. American Mathematical Soc., 2013.

[29] Hadi Susanto et al. "Soliton and breather splitting on star graphs from tricrystal Josephson junctions". In: *Symmetry* 11.2 (2019), p. 271.

[30] Daniel R Solli et al. "Optical rogue waves". In: *nature* 450.7172 (2007), pp. 1054–1057.

[31] Nail Akhmediev et al. “Recent progress in investigating optical rogue waves”. In: *Journal of optics* 15.6 (2013), p. 060201.

[32] John M Dudley, Goëry Genty, and Benjamin J Eggleton. “Harnessing and control of optical rogue waves in supercontinuum generation”. In: *Optics Express* 16.6 (2008), pp. 3644–3651.

[33] Cristian Bonatto et al. “Deterministic optical rogue waves”. In: *Physical review letters* 107.5 (2011), p. 053901.

[34] VI Karpman and EM Maslov. “Perturbation theory for solitons”. In: *Zh. Eksp. Teor. Fiz* 73.2 (1977), pp. 537–559.

[35] Adrien E Kraych et al. “Statistical properties of the nonlinear stage of modulation instability in fiber optics”. In: *Physical review letters* 123.9 (2019), p. 093902.

[36] Andrey Gelash et al. “Bound state soliton gas dynamics underlying the spontaneous modulational instability”. In: *Physical review letters* 123.23 (2019), p. 234102.

[37] Amdad Chowdury et al. “Modulation instability in higher-order nonlinear Schrödinger equations”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.12 (2018).

[38] Yuri S Kivshar and Boris A Malomed. “Dynamics of solitons in nearly integrable systems”. In: *Reviews of Modern Physics* 61.4 (1989), p. 763.

[39] DJ Kaup. “A perturbation expansion for the Zakharov–Shabat inverse scattering transform”. In: *SIAM Journal on Applied Mathematics* 31.1 (1976), pp. 121–133.

[40] Yuji Kodama and Akira Hasegawa. “Amplification and reshaping of optical solitons in glass fiber—II”. In: *Optics Letters* 7.7 (1982), pp. 339–341.

[41] EM Dianov et al. “Nonlinear dynamics of the amplification of solitons in the presence of stimulated Raman scattering in fiber-optic communication lines”. In: *Soviet Physics Doklady*. Vol. 20. 1985, p. 689.

[42] David J Kaup and Alan C Newell. “Solitons as particles, oscillators, and in slowly changing media: a singular perturbation theory”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 361.1707 (1978), pp. 413–446.

[43] Avner Peleg and Debananda Chakraborty. “Radiation dynamics in fast soliton collisions in the presence of cubic loss”. In: *Physica D: Nonlinear Phenomena* 406 (2020), p. 132397.

[44] Alexander Hause, Christoph Mahnke, and Fedor Mitschke. “Impact of fiber loss on two-soliton states: Substantial changes in eigenvalue spectrum”. In: *Physical Review A* 98.3 (2018), p. 033814.

[45] Yang Jianke. *Nonlinear waves in integrable and nonintegrable systems*. Vol. 16. Society for Industrial and applied Mathematics, Philadelphia, 2010.

[46] G Agrawal. *Applications of Nonlinear Fiber Optics*. Elsevier Science, 2010. ISBN: 9780080568768. URL: http://books.google.ru/books?id=1HZ6ROsqz2gC.

[47] Arnold Markovich Kosevich, BA Ivanov, and AS Kovalev. "Magnetic solitons". In: *Physics Reports* 194.3-4 (1990), pp. 117–238.

[48] Alfred Osborne. *Nonlinear Ocean Waves and the Inverse Scattering Transform*. Academic press, 2010.

[49] Govind P Agrawal. *Fiber-optic communication systems*. Vol. 222. John Wiley & Sons, 2012.

[50] Linn F Mollenauer and James P Gordon. *Solitons in optical fibers: fundamentals and applications*. Elsevier, 2006.

[51] Clifford S Gardner et al. "Method for Solving the Korteweg-deVries Equation". In: *Physical Review Letters* 19.19 (Nov. 1967), pp. 1095–1097. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.19.1095. URL: https://link.aps.org/doi/10.1103/PhysRevLett.19.1095.

[52] M. Yousefi and F. Kschischang. "Information transmission using the nonlinear Fourier transform, Part I: Mathematical tools". In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 4312–4328.

[53] V. E. Zakharov and A. B. Shabat. "Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media". In: *Soviet Physics JETP* 34.1 (1972), p. 62.

[54] Mark J Ablowitz et al. "The inverse scattering transform-Fourier analysis for nonlinear problems". In: *Studies in Applied Mathematics* 53.4 (1974), pp. 249–315.

[55] S Novikov et al. *Theory of solitons: the inverse scattering method*. Springer Science & Business Media, 1984.

[56] A. Hasegawa and T. Nyu. "Eigenvalue communication". In: *Journal of Lightwave Technology* 11.3 (1993), pp. 395–399.

[57] Xianhe Yangzhang et al. "Dual-Polarization Non-Linear Frequency-Division Multiplexed Transmission With b-Modulation". In: *Journal of Lightwave Technology* 37.6 (2019), pp. 1570–1578.

[58] R. Essiambre et al. "Capacity limits of optical fiber networks". In: *Journal of Lightwave Technology* 28.4 (2010), pp. 662–701.

[59] Jaroslaw E Prilepsky, Stanislav A Derevyanko, and Sergei K Turitsyn. "Nonlinear spectral management: Linearization of the lossless fiber channel". In: *Optics Express* 21.20 (2013), pp. 24344–24367.

[60] V. Aref. "Control and detection of discrete spectral amplitudes in nonlinear fourier spectrum". In: *arXiv preprint arXiv:1605.06328* (2016).

[61] Jaroslaw E Prilepsky et al. "Nonlinear inverse synthesis and eigenvalue division multiplexing in optical fiber channels". In: *Physical Review Letters* 113.1 (2014), p. 013901.

[62] S. Le, Jaroslaw E Prilepsky, and Sergei K Turitsyn. "Nonlinear inverse synthesis for high spectral efficiency transmission in optical fibers". In: *Optics Express* 22.22 (2014), pp. 26720–26741.

[63] S. T. Le, J. E. Prilepsky, and S. K. Turitsyn. “Nonlinear inverse synthesis technique for optical links with lumped amplification”. In: *Opt. Express* 23 (2015), pp. 8317–8328.

[64] Son Thai Le et al. “Nonlinear inverse synthesis for optical links with distributed Raman amplification”. In: *Journal of Lightwave Technology* 34.8 (2015), pp. 1778–1786.

[65] S. Le et al. “Demonstration of nonlinear inverse synthesis transmission over transoceanic distances”. In: *Journal of Lightwave Technology* 34.10 (2016), pp. 2459–2466.

[66] S. Le, V. Aref, and H. Buelow. “Nonlinear signal multiplexing for communication beyond the Kerr nonlinearity limit”. In: *Nature Photonics* 11.9 (2017), p. 570.

[67] M. Kamalian et al. “On the Design of NFT-Based Communication Systems With Lumped Amplification”. In: *Journal of Lightwave Technology* 35.24 (2017), pp. 5464–5472.

[68] Mansoor Yousefi and Xianhe Yangzhang. “Linear and nonlinear frequency-division multiplexing”. In: *IEEE Transactions on Information Theory* 66.1 (2019), pp. 478–495.

[69] Sander Wahls. “Generation of time-limited signals in the nonlinear Fourier domain via b-modulation”. In: *2017 European Conference on Optical Communication (ECOC)*. IEEE. 2017, pp. 1–3.

[70] Tao Gui et al. “Nonlinear frequency division multiplexing with b-modulation: shifting the energy barrier”. In: *Optics Express* 26.21 (2018), pp. 27978–27990.

[71] Dmitry Shepelsky et al. “Nonlinear Fourier spectrum characterization of time-limited signals”. In: *IEEE Transactions on Communications* 68.5 (2020), pp. 3024–3032.

[72] Shrinivas Chimmalgi and Sander Wahls. “Bounds on the Transmit Power of b-Modulated NFDM Systems in Anomalous Dispersion Fiber”. In: *Entropy* 22.6 (2020), p. 639.

[73] Xianhe Yangzhang et al. “Experimental Demonstration of Dual-Polarization NFDM Transmission With b-Modulation”. In: *IEEE Photonics Technology Letters* 31.11 (2019), pp. 885–888.

[74] Siddarth Hari, Mansoor I Yousefi, and Frank R Kschischang. “Multieigenvalue communication”. In: *Journal of Lightwave Technology* 34.13 (2016), pp. 3110–3117.

[75] H. Buelow, V. Aref, and W. Idler. “Transmission of waveforms determined by 7 eigenvalues with PSK-modulated spectral amplitudes”. In: *ECOC 2016; 42nd European Conference on Optical Communication; Proceedings of*. VDE. 2016, pp. 1–3.

[76] Yue Wu et al. “Robust neural network receiver for multiple-eigenvalue modulated nonlinear frequency division multiplexing system”. In: *Optics Express* 28.12 (2020), pp. 18304–18316.

[77] S. Derevyanko, J. Prilepsky, and S. Turitsyn. “Capacity estimates for optical transmission based on the nonlinear Fourier transform”. In: *Nature Communications* 7 (2016), p. 12710.

[78] Maryna Pankratova et al. “Signal-Noise Interaction in Optical-Fiber Communication Systems Employing Nonlinear Frequency-Division Multiplexing”. In: *Physical Review Applied* 13.5 (2020), p. 054021.

[79] Xianhe Yangzhang et al. “Impact of perturbations on nonlinear frequency-division multiplexing”. In: *Journal of Lightwave Technology* 36.2 (2018), pp. 485–494.

[80] Iman Tavakkolnia and Majid Safari. “The impact of PMD on single-polarization nonlinear frequency division multiplexing”. In: *Journal of Lightwave Technology* 37.4 (2019), pp. 1264–1272.

[81] Francesco Musumeci et al. “An overview on application of machine learning techniques in optical networks”. In: *IEEE Communications Surveys & Tutorials* 21.2 (2018), pp. 1383–1408.

[82] Faisal Nadeem Khan et al. “An optical communication’s perspective on machine learning and its applications”. In: *Journal of Lightwave Technology* 37.2 (2019), pp. 493–516.

[83] Stéphane Randoux et al. “Nonlinear spectral analysis of Peregrine solitons observed in optics and in hydrodynamic experiments”. In: *Physical Review E* 98.2 (2018), p. 022219. URL: https://doi.org/10.1103/PhysRevE.98.022219.

[84] J. M. Soto-Crespo, N. Devine, and N. Akhmediev. “Integrable Turbulence and Rogue Waves: Breathers or Solitons?” In: *Phys. Rev. Lett.* 116 (10 Mar. 2016), p. 103901.

[85] Sergei K Turitsyn, Igor S Chekhovskoy, and Mikhail P Fedoruk. “Nonlinear Fourier transform for characterization of the coherent structures in optical microresonators”. In: *Optics Letters* 45.11 (2020), pp. 3059–3062.

[86] Jia Wang et al. “Eigenvalue spectrum analysis for temporal signals of Kerr optical frequency combs based on nonlinear Fourier transform”. In: *Chinese Physics B* 29.3 (2020), p. 034207.

[87] P. Ryczkowski et al. “Real-time full-field characterization of transient dissipative soliton dynamics in a mode-locked laser”. In: *Nature Photonics* 12.4 (2018), p. 221.

[88] Srikanth Sugavanam et al. “Analysis of laser radiation using the Nonlinear Fourier transform”. In: *Nature Communications* 10.1 (2019), p. 5663.

[89] I. S. Chekhovskoy et al. “Nonlinear Fourier Transform for Analysis of Coherent Structures in Dissipative Systems”. In: *Physical Review Letters* 122.15 (Apr. 2019), p. 153901. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.122.153901. URL: https://link.aps.org/doi/10.1103/PhysRevLett.122.153901.

[90] J. Skaar, Ligang Wang, and T. Erdogan. “On the synthesis of fiber Bragg gratings by layer peeling”. In: *IEEE Journal of Quantum Electronics* 37.2 (2001), pp. 165–173.

[91] G. E. Turitsyna et al. “Novel design of FBG-based composite double notch VSB filter for DWDM systems”. In: *Journal of Lightwave Technology* 24.9 (2006), pp. 3547–3552.

[92] Samuel H Rudy et al. “Data-driven discovery of partial differential equations”. In: *Science Advances* 3.4 (2017), e1602614.

[93] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. “Deep learning for universal linear embeddings of nonlinear dynamics”. In: *Nature Communications* 9.1 (2018), p. 4950.

[94] Zongyi Li et al. “Fourier Neural Operator for Parametric Partial Differential Equations”. In: *arXiv preprint arXiv:2010.08895* (2020).

[95] Morteza Kamalian et al. “Periodic nonlinear Fourier transform for fiber-optic communications, Part I: theory and numerical methods”. In: *Optics Express* 24.16 (2016), pp. 18353–18369.

[96] Jan-Willem Goossens, Hartmut Hafermann, and Yves Jaouën. “Data transmission based on exact inverse periodic nonlinear Fourier transform, Part I: Theory”. In: *Journal of Lightwave Technology* 38.23 (2020), pp. 6499–6519. DOI: 10.1109/JLT.2020.3013148.

[97] Jan-Willem Goossens, Yves Jaouën, and Hartmut Hafermann. “Experimental demonstration of data transmission based on the exact inverse periodic nonlinear Fourier transform”. In: *Optical Fiber Communication Conference*. Optica Publishing Group. 2019, pp. M1I–6. DOI: 10.1364/OFC.2019.M1I.6.

[98] Morteza Kamalian et al. “Signal modulation and processing in nonlinear fibre channels by employing the Riemann–Hilbert problem”. In: *Journal of Lightwave Technology* 36.24 (2018), pp. 5714–5727.

[99] S Novikov et al. *Theory of solitons: the inverse scattering method*. Springer Science & Business Media, 1984.

[100] Thomas Trogdon and Sheehan Olver. *Riemann–Hilbert problems, their numerical solution, and the computation of nonlinear special functions*. SIAM, 2015.

[101] Alfred Osborne. *Nonlinear Ocean Waves and the Inverse Scattering Transform*. Academic Press, 2010.

[102] Alfred R Osborne. “Nonlinear fourier analysis: Rogue waves in numerical modeling and data analysis”. In: *Journal of Marine Science and Engineering* 8.12 (2020), p. 1005.

[103] Vladimir B Matveev. “30 years of finite-gap integration theory”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366.1867 (2008), pp. 837–875.

[104] Eugene D Belokolos et al. *Algebro-geometric approach to nonlinear integrable equations*. Vol. 550. Springer, 1994.

[105] Vladimir Kotlyarov and Dmitry Shepelsky. “Planar unimodular Baker-Akhiezer function for the nonlinear Schrödinger equation”. In: *Annals of Mathematical Sciences and Applications* 2.2 (2017), pp. 343–384. DOI: 10.4310/AMSA.2017.v2.n2.a6.

[106] Morteza Kamalian-Kopae et al. “Full-spectrum periodic nonlinear Fourier transform optical communication through solving the Riemann-Hilbert problem”. In: *Journal of Lightwave Technology* 38.14 (2020), pp. 3602–3615. DOI: 10.1109/JLT.2020.2979322.

[107] Stepan Bogdanov et al. “Phase computation for the finite-genus solutions to the focusing nonlinear Schrödinger equation using convolutional neural networks”. In: *Communications in Nonlinear Science and Numerical Simulation* 125 (2023), p. 107311.

[108] Mark J. Ablowitz and Harvey Segur. *Solitons and the Inverse Scattering Transform*. Philadelphia: Society for Industrial and Applied Mathematics, Jan. 1981, p. 425. ISBN: 978-0-89871-174-5. DOI: 10.1137/1.9781611970883. URL: http://epubs.siam.org/doi/book/10.1137/1.9781611970883.

[109] Scott R. J. “Report on waves”. In: *Rep. 14th Meeting of the British Assoc. for the Advancement of Science* (1844), pp. 311–390.

[110] N. J. Zabusky and M. D. Kruskal. “Interaction of “Solitons” in a Collisionless Plasma and the Recurrence of Initial States”. In: *Phys. Rev. Lett.* 15 (1965), pp. 240–243.

[111] Michael Faraday. “XVII. On a peculiar class of acoustical figures; and on certain forms assumed by groups of particles upon vibrating elastic surfaces”. In: 121 (1831).

[112] A. V. Vasyliev, Y. M. Romanovskii, and V. G. Yahno. *Autowave processes*. Moscow: Science, 1987.

[113] B. S. Kerner and V. V. Osipov. *Autosolitons*. Moscow: Science, 1991.

[114] Y.R. Shen. *The Principles of Nonlinear Optics*. Pure & Applied Optics Series: 1-349. Wiley, 1984. ISBN: 9780471889984.

[115] P.N. Butcher and D.N. Cotter. *The Elements of Nonlinear Optics*. 1990.

[116] R. W. Boyd. “Nonlinear Optics”. In: *San Diego: Academic Press* (1992).

[117] Stéphane Randoux, Pierre Suret, and Gennady El. “Inverse scattering transform analysis of rogue waves using local periodization procedure”. In: *Scientific Reports* 6 (2016), p. 29238.

[118] G. A. Askaryan. “Prediction of the existence of a waveguide mode with the propagation of an electromagnetic beam”. In: *JETP* 42 (1962), p. 1567.

[119] Rene Schmogrow et al. “Error Vector Magnitude as a Performance Measure for Advanced Modulation Formats”. In: *IEEE Photonics Technology Letters* 24.1 (2012), pp. 61–63.

[120] Morteza Kamalian et al. “Signal Modulation and Processing in Nonlinear Fibre Channels by Employing the Riemann–Hilbert Problem”. In: *Journal of Lightwave Technology* 36.24 (Dec. 2018), pp. 5714–5727. ISSN: 0733-8724. DOI: 10.1109/JLT.2018.2877103. URL: https://ieeexplore.ieee.org/document/8500234/.

[121] Andrey Gelash et al. “Bound State Soliton Gas Dynamics Underlying the Spontaneous Modulational Instability”. In: *Physical Review Letters* 123.23 (Dec. 2019), p. 234102. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.123.234102. URL: https://link.aps.org/doi/10.1103/PhysRevLett.123.234102.

[122] Rustam Mullyadzhanov and Andrey Gelash. “Direct scattering transform of large wave packets”. In: *Optics Letters* 44.21 (Nov. 2019), p. 5298. ISSN: 0146-9592. DOI: 10.1364/ol.44.005298. URL: https://www.osapublishing.org/abstract.cfm?URI=ol-44-21-5298.

[123] Mansoor I. Yousefi and Frank R Kschischang. “Information Transmission Using the Nonlinear Fourier Transform, Part I: Mathematical Tools”. In: *IEEE Transactions on Information Theory* 60.7 (July 2014), pp. 4312–4328. ISSN: 0018-9448. DOI: 10.1109/TIT.2014.2321143. URL: http://ieeexplore.ieee.org/document/6808480/.

[124] G. Boffetta and A.R Osborne. “Computation of the direct scattering transform for the nonlinear Schroedinger equation”. In: *Journal of Computational Physics* 102.2 (Oct. 1992), pp. 252–264. ISSN: 00219991. DOI: 10.1016/0021-9991(92)90370-E. URL: http://linkinghub.elsevier.com/retrieve/pii/002199919290370E.

[125] O. V. Belai et al. “Efficient numerical method of the fiber Bragg grating synthesis”. In: *Journal of the Optical Society of America B* 24.7 (July 2007), p. 1451. ISSN: 0740-3224. DOI: 10.1364/JOSAB.24.001451. URL: http://arxiv.org/abs/physics/0611039%0Ahttp://dx.doi.org/10.1364/JOSAB.24.001451%20https://www.osapublishing.org/abstract.cfm?URI=josab-24-7-1451.

[126] Leonid L. Frumin et al. “Efficient numerical method for solving the direct Zakharov-Shabat scattering problem”. In: *Journal of the Optical Society of America B* 32.2 (2015), p. 290. ISSN: 0740-3224. DOI: 10.1364/JOSAB.32.000290. URL: https://www.osapublishing.org/abstract.cfm?URI=josab-32-2-290%20http://josab.osa.org/abstract.cfm?URI=josab-32-2-290.

[127] Xulun Zhang et al. “Full-Spectrum INFT Algorithm for Dual-Polarization NFDM Transmission”. In: *Journal of Lightwave Technology* 41.10 (May 2023), pp. 3017–3025. ISSN: 0733-8724. DOI: 10.1109/JLT.2023.3239867. URL: https://ieeexplore.ieee.org/document/10026447/.

[128] Mansoor I Yousefi and Frank R Kschischang. “Information Transmission Using the Nonlinear Fourier Transform, Part II: Numerical Methods”. In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 4329–4345. ISSN: 0018-9448. DOI: 10.1109/TIT.2014.2321151. URL: http://ieeexplore.ieee.org/document/6808508/.

[129] A Vasylchenkova et al. “Direct nonlinear Fourier transform algorithms for the computation of solitonic spectra in focusing nonlinear Schrödinger equation”. In: *Communications in Nonlinear Science and Numerical Simulation* 68 (Mar. 2019), pp. 347–371. ISSN: 10075704. DOI: 10.1016/j.cnsns.2018.09.005. URL: http://arxiv.org/abs/1708.01144%2010.1016/j.cnsns.2018.09.005%20https://linkinghub.elsevier.com/retrieve/pii/S1007570418302855.

[130] Sander Wahls and H. Vincent Poor. “Introducing the fast nonlinear Fourier transform”. In: *International Conference on Acoustics, Speech and Signal Processing*. Vancouver: IEEE, 2013, pp. 5780–5784. ISBN: 978-1-4799-0356-6. DOI: 10.1109/ICASSP.2013.6638772. URL: http://ieeexplore.ieee.org/document/6638772/.

[131] Sander Wahls and H. Vincent Poor. “Fast inverse nonlinear Fourier transform for generating multi-solitons in optical fiber”. In: *International Symposium on Information Theory (ISIT)*. Hong Kong: IEEE, 2015, pp. 1676–1680. ISBN: 9781467377041. DOI: 10.1109/ISIT.2015.7282741. URL: https://ieeexplore.ieee.org/document/7282741.

[132] Sander Wahls and H. Vincent Poor. “Fast Numerical Nonlinear Fourier Transforms”. In: *IEEE Transactions on Information Theory* 61.12 (Dec. 2015), pp. 6957–6974. ISSN: 00189448. DOI: 10.1109/TIT.2015.2485944. URL: http://ieeexplore.ieee.org/document/7286836/.

[133] Sander Wahls and Vishal Vaibhav. “Fast Inverse Nonlinear Fourier Transforms for Continuous Spectra of Zakharov-Shabat Type”. In: *arXiv preprint arXiv:1607.01305* (July 2016). URL: http://arxiv.org/abs/1607.01305.

[134] Vishal Vaibhav. “Higher Order Convergent Fast Nonlinear Fourier Transform”. In: *IEEE Photonics Technology Letters* 30.8 (Apr. 2018), pp. 700–703. ISSN: 1041-1135. DOI: 10.1109/LPT.2018.2812808. URL: http://ieeexplore.ieee.org/document/8307084/.

[135] Sander Wahls, Shrinivas Chimmalgi, and Peter J Prins. “FNFT: A Software Library for Computing Nonlinear Fourier Transforms”. In: *Journal of Open Source Software* 3.23 (Mar. 2018), p. 597. ISSN: 2475-9066. DOI: 10.21105/joss.00597. URL: http://joss.theoj.org/papers/10.21105/joss.00597.

[136] Shrinivas Chimmalgi, Peter J. Prins, and Sander Wahls. “Fast Nonlinear Fourier Transform Algorithms Using Higher Order Exponential Integrators”. In: *IEEE Access* 7 (2019), pp. 145161–145176. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2945480. URL: https://ieeexplore.ieee.org/document/8856211/.

[137] Son Thai Le, Jaroslaw E. Prilepsky, and Sergei K. Turitsyn. “Nonlinear inverse synthesis for high spectral efficiency transmission in optical fibers”. In: *Optics Express* 22.22 (Nov. 2014), p. 26720. ISSN: 1094-4087. DOI: 10.1364/OE.22.026720. URL: https://www.osapublishing.org/oe/abstract.cfm?uri=oe-22-22-26720.

[138] Jaroslaw E. Prilepsky et al. “Nonlinear inverse synthesis and eigenvalue division multiplexing in optical fiber channels”. In: *Physical Review Letters* 113.1 (2014), pp. 12–14. ISSN: 10797114. DOI: 10.1103/PhysRevLett.113.013901.

[139] René-Jean Essiambre and Robert W Tkach. “Capacity trends and limits of optical communication networks”. In: *Proceedings of the IEEE* 100.5 (2012), pp. 1035–1055.

[140] Stella Civelli et al. “Polarization-multiplexed nonlinear inverse synthesis with standard and reduced-complexity NFT processing”. In: *Optics Express* 26.13 (2018), p. 17360. ISSN: 1094-4087. DOI: 10.1364/OE.26.017360. URL: https://www.osapublishing.org/abstract.cfm?URI=oe-27-3-3617%20https://www.osapublishing.org/abstract.cfm?URI=oe-26-13-17360.

[141] Egor V. Sedov et al. “Soliton content in the standard optical OFDM signal”. In: *Opt. Lett.* 43.24 (Dec. 2018), pp. 5985–5988. DOI: 10.1364/OL.43.005985. URL: http://ol.osa.org/abstract.cfm?URI=ol-43-24-5985.

[142] L. L. Frumin, A. A. Gelash, and S. K. Turitsyn. "New Approaches to Coding Information using Inverse Scattering Transform". In: *Physical Review Letters* 118.22 (May 2017), p. 223901. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.118.223901. URL: http://link.aps.org/doi/10.1103/PhysRevLett.118.223901.

[143] Tao Gui et al. "High-order modulation on a single discrete eigenvalue for optical communications based on nonlinear Fourier transform". In: *Optics Express* 25.17 (Aug. 2017), p. 20286. ISSN: 1094-4087. DOI: 10.1364/OE.25.020286. URL: https://www.osapublishing.org/abstract.cfm?URI=oe-25-17-20286.

[144] Vahid Aref, Son Thai Le, and Henning Buelow. "Modulation Over Nonlinear Fourier Spectrum: Continuous and Discrete Spectrum". In: *Journal of Lightwave Technology* 36.6 (Mar. 2018), pp. 1289–1295. ISSN: 0733-8724. DOI: 10.1109/JLT.2018.2794475. URL: http://ieeexplore.ieee.org/document/8260868/.

[145] Alexey Slunyaev. "Nonlinear analysis and simulations of measured freak wave time series". In: *European Journal of Mechanics-B/Fluids* 25.5 (2006), pp. 621–635.

[146] AV Slunyaev. "Analysis of the nonlinear spectrum of intense sea wave with the purpose of extreme wave prediction". In: *Radiophysics and Quantum Electronics* 61 (2018), pp. 1–21.

[147] Alexey Slunyaev. "Persistence of hydrodynamic envelope solitons: Detection and rogue wave occurrence". In: *Physics of Fluids* 33.3 (2021).

[148] Oleg Sidelnikov et al. "Advanced Convolutional Neural Networks for Nonlinearity Mitigation in Long-Haul WDM Transmission Systems". In: *Journal of Lightwave Technology* 39.8 (2021), pp. 2397–2406. DOI: 10.1109/JLT.2021.3051609.

[149] Oleg Sidelnikov, Alexey Redyuk, and Stylianos Sygletos. "Nonlinear Equalization in Long Haul Transmission Systems Using Dynamic Multi-Layer Perceptron Networks". In: *2018 European Conference on Optical Communication (ECOC)*. 2018, pp. 1–3. DOI: 10.1109/ECOC.2018.8535144.

[150] Pedro J Freire et al. "Performance versus complexity study of neural network equalizers in coherent optical systems". In: *Journal of Lightwave Technology* 39.19 (2021), pp. 6085–6096.

[151] Darko Zibar et al. "Machine learning techniques in optical communication". In: *Journal of Lightwave Technology* 34.6 (2015), pp. 1442–1452.

[152] John C Cartledge et al. "Digital signal processing for fiber nonlinearities". In: *Optics express* 25.3 (2017), pp. 1916–1936.

[153] Igor Chekhovskoy et al. *Introducing phase jump tracking – a fast method for eigenvalue evaluation of the direct Zakharov-Shabat problem*. 2020.

[154] I S Chekhovskoy et al. “Fast Eigenvalue Evaluation of the Direct Zakharov-Shabat Problem in Telecommunication Signals Using Adaptive Phase Jump Tracking”. In: *2021 Conference on Lasers and Electro-Optics Europe & European Quantum Electronics Conference (CLEO/Europe-EQEC)*. IEEE, June 2021, pp. 1–1. ISBN: 978-1-6654-1876-8. DOI: 10.1109/CLEO/Europe-EQEC52157.2021.9542265. URL: https://ieeexplore.ieee.org/document/9542265/.

[155] Sergey Medvedev et al. “Fast sixth-order algorithm based on the generalized Cayley transform for the Zakharov-Shabat system associated with nonlinear Schrodinger equation”. In: *Journal of Computational Physics* 448 (2022), p. 110764. ISSN: 10902716. DOI: 10.1016/j.jcp.2021.110764. URL: https://doi.org/10.1016/j.jcp.2021.110764.

[156] Sergey Medvedev et al. “Exponential fourth order schemes for direct Zakharov-Shabat problem”. In: *Optics Express* 28.1 (Jan. 2020), p. 20. ISSN: 1094-4087. DOI: 10.1364/OE.377140. URL: http://arxiv.org/abs/1908.11725%20https://www.osapublishing.org/oe/abstract.cfm?doi=10.1364/OE.377140%20https://www.osapublishing.org/abstract.cfm?URI=oe-28-1-20.

[157] Peter J Prins and Sander Wahls. “Higher Order Exponential Splittings for the Fast Non-Linear Fourier Transform of the Korteweg-De Vries Equation”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 4. IEEE, 2018, pp. 4524–4528. ISBN: 9781538646588. DOI: 10.1109/ICASSP.2018.8461708. URL: https://ieeexplore.ieee.org/document/8461708/.

[158] M Klaus and J. K. Shaw. “On the eigenvalues of Zakharov-Shabat system”. In: *J. Math. Anal.* 34.4 (2003), pp. 759–773.

[159] Simone Gaiarin et al. “Dual-polarization NFDM transmission using distributed Raman amplification and NFT-domain equalization”. In: *IEEE Photonics Technology Letters* 30.22 (2018), pp. 1983–1986.

[160] Jonas Koch, Rebekka Weixer, and Stephan Pachnicke. “Equalization of Soliton Transmission Based on Nonlinear Fourier Transform using Neural Networks”. In: *45th European Conference on Optical Communication (ECOC)*. 2019, pp. 1–3.

[161] Oleksandr Kotlyar et al. “Machine learning for performance improvement of periodic NFT-based communication system”. In: *2019 European Conference on Optical Communications*. 2019.

[162] Oleksandr Kotlyar et al. “Combining nonlinear Fourier transform and neural network-based processing in optical communications”. In: *Optics Letters* 45.13 (2020), pp. 3462–3465.

[163] Oleksandr Kotlyar et al. “Convolutional long short-term memory neural network equalizer for nonlinear Fourier transform-based optical transmission systems”. In: *Optics Express* 29.7 (2021), pp. 11254–11267.

[164] Shohei Yamamoto, Ken Mishina, and Akihiro Maruta. “Demodulation of optical eigenvalue modulated signal using neural network”. In: *IEICE Communications Express* 8.12 (2019), pp. 507–512.

[165] Rasmus T Jones et al. “Time-domain neural network receiver for nonlinear frequency division multiplexed systems”. In: *IEEE Photonics Technology Letters* 30.12 (2018), pp. 1079–1082.

[166] Wen Qi Zhang, Terence H Chan, and Shahraam Afshar. “Direct decoding of nonlinear OFDM-QAM signals using convolutional neural network”. In: *Optics Express* 29.8 (2021), pp. 11591–11604.

[167] Egor Valentinovich Sedov et al. “Application of neural networks to determine the discrete spectrum of the direct Zakharov–Shabat problem”. In: *Quantum Electronics* 50.12 (2020), p. 1105.

[168] Egor V Sedov et al. “Neural networks for computing and denoising the continuous nonlinear Fourier spectrum in focusing nonlinear Schrödinger equation”. In: *Scientific Reports* 11.1 (2021), p. 22857.

[169] Egor Valentinovich Sedov, Igor Sergeevich Chekhovskoy, and Jaroslav Evgen’evich Prilepsky. “Neural network for calculating direct and inverse nonlinear Fourier transform”. In: *Quantum Electronics* 51.12 (2021), p. 1118.

[170] Makoto Matsumoto and Takuji Nishimura. “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator”. In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30.

[171] S. Wahls, S. Chimmalgi, and P.J. Prins. “FNFT: A Software Library for Computing Nonlinear Fourier Transforms”. In: *The Journal of Open Source Software* 3 (23 2018), p. 597. ISSN: 2475-9066.

[172] Sergey Turitsyn et al. “Nonlinear Spectrum of Conventional OFDM and WDM Return-to-Zero Signals in Nonlinear Channel”. In: *Journal of Lightwave Technology* 38.2 (2019), pp. 352–358.

[173] Martin Klaus and JK Shaw. “On the eigenvalues of Zakharov–Shabat systems”. In: *SIAM Journal on Mathematical Analysis* 34.4 (2003), pp. 759–773.

[174] Sergei K Turitsyn and SA Derevyanko. “Soliton-based discriminator of noncoherent optical pulses”. In: *Physical Review A* 78.6 (2008), p. 063819.

[175] Stanislav A Derevyanko and Jaroslaw E Prilepsky. “Soliton generation from randomly modulated return-to-zero pulses”. In: *Optics Communications* 281.21 (2008), pp. 5439–5443.

[176] Sergey Medvedev et al. “Exponential fourth order schemes for direct Zakharov-Shabat problem”. In: *Optics Express* 28.1 (Jan. 2020), pp. 20–39.

[177] Yaniv Taigman et al. “Deepface: Closing the gap to human-level performance in face verification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1701–1708.

[178] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).

[179] Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. "BOA: The Bayesian optimization algorithm". In: *Proceedings of the genetic and evolutionary computation conference GECCO-99*. Vol. 1. Citeseer. 1999, pp. 525–532.

[180] Jonas Močkus. "On Bayesian methods for seeking the extremum". In: *Optimization techniques IFIP technical conference*. Springer. 1975, pp. 400–404.

[181] James C Spall. "Adaptive stochastic approximation by the simultaneous perturbation method". In: *IEEE Transactions on Automatic Control* 45.10 (2000), pp. 1839–1853.

[182] Ken Mishina et al. "Eigenvalue-domain Neural Network Demodulator for Eigenvalue-modulated Signal". In: *Journal of Lightwave Technology* (2021). DOI: 10.1109/JLT.2021.3074744.

[183] Egor Sedov. *HpCom - High Performance Communication Library for Python*. Version v0.1.4. Apr. 2023. DOI: 10.5281/zenodo.7880552. URL: https://doi.org/10.5281/zenodo.7880552.

[184] Marius Brehler et al. "A GPU-accelerated fourth-order Runge–Kutta in the interaction picture method for the simulation of nonlinear signal propagation in multimode fibers". In: *Journal of Lightwave Technology* 35.17 (2017), pp. 3622–3628.

[185] Sasipim Srivallapanondh et al. "Knowledge Distillation Applied to Optical Channel Equalization: Solving the Parallelization Problem of Recurrent Connection". In: *Optical Fiber Communication Conference*. Optica Publishing Group. 2023.

[186] Goëry Genty et al. "Machine learning and applications in ultrafast photonics". In: *Nature Photonics* 15.2 (2021), pp. 91–101.

[187] AV Reznichenko et al. "Optimal input signal distribution for a nonlinear optical fiber channel with small Kerr nonlinearity". In: *JOSA B* 39.3 (2022), pp. 810–820.

[188] Peter J Winzer. "Optical networking beyond WDM". In: *IEEE Photonics Journal* 4.2 (2012), pp. 647–651.

[189] G P Agrawal. *Nonlinear Fiber Optics*. 5th ed. Boston: Academic Press, 2013. ISBN: 9780123973078. URL: https://www.elsevier.com/books/nonlinear-fiber-optics/agrawal/978-0-12-397023-7.

[190] Francesco Poletti and Peter Horak. "Description of ultrashort pulse propagation in multimode optical fibers". In: *JOSA B* 25.10 (2008), pp. 1645–1654.

[191] Sami Mumtaz, René-Jean Essiambre, and Govind P Agrawal. "Nonlinear propagation in multimode and multicore fibers: generalization of the Manakov equations". In: *Journal of Lightwave Technology* 31.3 (2012), pp. 398–406.

[192] René-Jean Essiambre et al. "Capacity limits of optical fiber networks". In: *Journal of Lightwave Technology* 28.4 (2010), pp. 662–701.

[193] MG Taylor et al. "Application of dynamic gain equalisation for ultra long haul transmission". In: *IEEE/LEOS Summer Topi All-Optical Networking: Existing and Emerging Architecture and Applications/Dynamic Enablers of Next-Generation Optical Communications Systems/Fast Optical Processing in Optical*. IEEE. 2002, TuF1–TuF1.

[194] Gabriel Charlet et al. "72× 100Gb/s transmission over transoceanic distance, using large effective area fiber, hybrid Raman-Erbium amplification and coherent detection". In: *Optical Fiber Communication Conference*. Optica Publishing Group. 2009, PDPB6.

[195] Sotiris B Kotsiantis. "Decision trees: a recent overview". In: *Artificial Intelligence Review* 39 (2013), pp. 261–283.

[196] Yan-Yan Song and LU Ying. "Decision tree methods: applications for classification and prediction". In: *Shanghai archives of psychiatry* 27.2 (2015), p. 130.

[197] Ibomoiye Domor Mienye, Yanxia Sun, and Zenghui Wang. "Prediction performance of improved decision tree-based algorithms: a review". In: *Procedia Manufacturing* 35 (2019), pp. 698–703.

[198] Alexey Natekin and Alois Knoll. "Gradient boosting machines, a tutorial". In: *Frontiers in neurorobotics* 7 (2013), p. 21.

[199] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. "A comparative analysis of gradient boosting algorithms". In: *Artificial Intelligence Review* 54 (2021), pp. 1937–1967.

[200] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[201] Jerome H Friedman. "Stochastic gradient boosting". In: *Computational statistics & data analysis* 38.4 (2002), pp. 367–378.

[202] Adrián Alcolea and Javier Resano. "FPGA accelerator for gradient boosting decision trees". In: *Electronics* 10.3 (2021), p. 314.

[203] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[204] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[205] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[206] G. Boffetta and A. R. Osborne. "Computation of the Direct Scattering Transform for the Nonlinear Schroedinger Equation". In: *Journal of computational physics* 102.2 (1992), pp. 252–264.

[207] M. J. Ablowitz and J. Ladik. "Nonlinear differential-difference equations". In: *Math. Phys.* 16 (1975), pp. 598–603.

[208] M. J. Ablowitz and J. Ladik. "Nonlinear differential–difference equations and fourier-analysis". In: *Math. Phys.* 17 (1976), pp. 1011–1018.

[209] G.P. Agrawal. *Fiber-optic communication systems*. Wiley series in microwave and optical engineering 1. Wiley-Interscience, 2002. ISBN: 9780471215714.

[210] R. Kashyap. *Fiber Bragg Gratings*. Electronics & Electrical. Elsevier Science, 1999. ISBN: 9780124005600.

[211] O. V. Belai et al. "Efficient numerical method of the fiber Bragg grating synthesis". In: *J. Opt. Soc. Am. B* 24.7 (July 2007), pp. 1451–1457.

[212] G.L. Lamb. *Elements of Soliton Theory*. A Wiley-Interscience publication. Wiley, 1980. ISBN: 9780471045595.

[213] Anastasiia Vasylchenkova, Jaroslaw E. Prilepsky, and Sergei K. Turitsyn. "Contour integrals for numerical computation of discrete eigenvalues in the Zakharov–Shabat problem". In: *Optics Lett.* 43.15 (2018), pp. 3690–3693.

[214] L. M. Delves and J. N. Lyness. "A numerical method for locating the zeros of an analytic function". In: *Math. Comp.* 21 (1967), pp. 543–560.

[215] Vera N. Kublanovskaya. "On some algorithms for the solution of the complete eigenvalue problem". In: *USSR Computational Mathematics and Mathematical Physics* 1.3 (1963), pp. 637–657.

[216] J.G.F. Francis. "The QR Transformation, I". In: *The Computer Journal* (1961), pp. 265–271.

[217] J.G.F. Francis. "The QR Transformation, II". In: *The Computer Journal* (1962), pp. 332–345.

Acronyms

ASE Amplified spontaneous emission. xiii, 129, 130

BER Bit Error Rate. ix, x, xii, xiii, 30, 32, 47–50, 95–97, 99, 100, 105, 106, 121, 122, 131

BPSK Binary Phase-Shift Keying. 29

CDC Chromatic dispersion compensation. ix, xv, 7, 8, 39, 43, 96, 101, 124, 125, 130, 136, 146, 174, 176–178

CNN Convolution Neural network. 8, 72, 73, 80, 81, 84

CPU Central Processing Unit. xii, 94, 104–106, 110

DBP Digital back-propagation. xiv, xv, 7, 48, 99, 100, 123, 146, 157, 158, 174, 176–178

DP Dual-polarization. 48

DPSK Differential Phase-Shift Keying. 29

DQPSK Differential Quadrature Phase-Shift Keying. 29

DSP Digital signal processing. 96

EDFA Erbium-Doped Fiber Amplifier. xii, xiii, 12, 96, 104, 106–108, 115, 130, 131, 139, 180

EM Expectation-Maximization. 134

EVM Error Vector Magnitude. xii, 30, 32, 48, 95–97, 99, 100, 105

FBG Fiber Bragg Grating. 150

FEC Forward error correction. 1, 125

FFT Fast Fourier Transform. 36, 37, 99

FLOP floating point operations. xiv, 157, 158

FNFT Fast Nonlinear Fourier Transform. xi, xiv, 38, 46, 78, 79, 85, 157, 158

FNN Feed-forward Neural Network. 73

FPGA Field Programmable Gate Arrays. 110, 123

FT Fourier Transform. 13, 14, 29

GB Gradient Boosting. iv, xiii, 7–9, 110, 112–123, 146

GLM Gelfand-Levitan-Marchenko. 16, 21, 150, 152

GMM Gaussian Mixture Model. iv, xiv, 9, 132–136, 139–144

GPU Graphics Processing Unit. xii, 94–96, 102, 104–106, 110, 115, 129, 131, 146, 183, 184

GVD Group velocity dispersion. 11

HpCom High-Performance COMmunication library. 7, 8, 96, 146

IFFT Inverse Fast Fourier Transform. 37, 103

IGR Information Gain Ratio. 109

ISI Inter-Symbol Interference. 35–37

IST Inverse Scattering Transform. 16, 17, 39

ME Manakov Equation. 11–13, 17, 21, 22, 39, 41, 45, 48, 49

MI Mutual information. xii, 30, 33, 95–97, 99, 100, 105

ML Machine learning. iv, 90, 96

MLP Multi-Layer Perceptron. 123

MSE Mean squared error. xi, 81, 83, 87, 113, 114

MZS Manakov-Zakharov-Shabat. 22, 24

NF Nonlinear Fourier. vi, ix, xi, 14, 15, 18, 23, 28, 44, 45, 64, 66, 71, 76, 78–80, 83, 84, 87, 90–92

NFDM Nonlinear Frequency Division Multiplexing. 71, 87

NFT Nonlinear Fourier Transform. iv, ix, xi, xii, 1–8, 12–18, 21, 23–25, 35, 38, 39, 41–43, 45, 46, 48–50, 71–74, 77–81, 83–92, 146, 157

NLSE Nonlinear Schrödinger Equation. 3–5, 11–13, 15–18, 20, 22, 25, 41, 45, 48, 49, 54, 91, 124, 129

NN Neural network. x, xv, 3, 4, 7–9, 15, 71–74, 77–81, 83, 84, 86–88, 90–92, 123, 146, 176–178

NPE Nonlinear Phase Equalisation. xv, 101, 125, 130, 136, 174, 176–178

OFDM Orthogonal Frequency Division Multiplexing. ix, x, xii, 7, 8, 36, 37, 53–60, 97–99, 101, 103, 146

OOK On-Off Keying. 29

PAM Pulse Amplitude Modulation. 29

PDE partial differential equation. 13, 17, 92

PNFT Periodic Nonlinear Fourier Transform. 14–16, 28

PSK Phase-Shift Keying. ix, 29–31, 37

QAM Quadrature amplitude modulation. ix, x, xii, xv, xvi, 1, 29–32, 36, 37, 45, 55–57, 60, 61, 63, 75, 96, 97, 102, 104, 106–108, 112, 115, 130, 132, 134, 142, 144, 177, 178, 182

QPSK Quadrature Phase-Shift Keying. ix, x, xv, 29, 31, 36, 54–57, 60, 61, 74, 75, 78, 176, 178

RC Raised Cosine. ix, 34–36

RF Random Forest. 110

RH Riemann-Hilbert. 16

RHP Riemann-Hilbert Problem. 16

RNN Recurrent neural network. 80

RRC Root Raised Cosine. ix, 34–36, 63, 99, 101, 102, 105, 115, 130

RZ return-to-zero. 35, 71

SDM Space Division Multiplexing. 2

SMF Single-mode fiber. xiv, xv, 105, 165, 172, 177, 178

SNR Signal-to-noise ratio. xi, xii, 34, 78, 79, 83–87, 89

SSFM Split-step Fourier method. 63, 95, 123

SSMF Standard single-mode fiber. 63, 96, 104, 115, 130, 180

TWC TrueWave classic. xv, 176, 178

WDM Wavelength Division Multiplexing. ix–xii, xiv, 7, 8, 33–36, 43, 45, 50, 53, 55, 60–63, 66, 69, 71, 72, 75, 76, 78, 96–102, 104–106, 130, 139, 146, 165, 179, 180, 182, 183

ZS Zakharov-Shabat. 22, 80

ZSP Zakharov-Shabat problem. 18, 19, 26

ZSSP Zakharov-Shabat spectral problem. 18