POLITECNICO DI TORINO

Master's Degree in Degree Course

Master's Degree Thesis

Analyses and upgrades of the open source project GNPy

Supervisors

Prof. Vittorio CURRI

Dr. Andrea D'AMICO

Candidate

Vittorio GATTO

December 2023

Abstract

In the rapidly evolving landscape of the digital age, the demand for Internet bandwidth shows no signs of slowing down. Most of today's Internet traffic travels through fiber optics. The remarkable properties of optical fiber, such as low attenuation and high bandwidth capacity, make it an ideal choice for long-distance, high-capacity data transmission. As the global demand for high-speed Internet continues to grow, the continued advancement of optical technology remains critical to ensuring that the digital infrastructure can handle the ever-increasing traffic and reduce energy consumption through increased efficiency. A better understanding of the physics of each element needed to realize its great potential. The goal of this thesis is to provide a model that accurately characterizes the physical effect that light suffers as it traverses different elements of the optical network. The first part of the research focused on finding a model to characterize the optical amplifier and test it in different real-world scenarios. A dataset with information about different optical amplifiers has been provided. The result that allowed the development of the model is that the gain profile depends only on the gain and the tilt of the amplifier. A step-by-step procedure has been created to best characterize each amplifier, resulting in 2 unique profiles that are specific to the amplifier. From the amplifier characterization, it is possible to recreate all possible gain profiles derived from the gain and tilt pair. The final step has been to provide an efficient way to insert this model into the open source software GNPy (a digital twin for the optical network developed by a consortium of companies). The second part of the research has been the aim to verify the behavior of GNPy with the new model of amplifier in multi-band scenarios. Validation of GNPy for L-band and C-band has been performed. The topology used for data acquisition consists of L-band and C-band optical amplifiers and 5 fiber spans. Optical fiber parameters and connector losses have been measured. Connector losses and some fiber parameters have been post-processed to obtain consistent data. An experimental measured campaign has been carry out with different launch power level. An optical spectrum analyzer has been placed before and after each amplifier to measure power and noise. This data has been compared with the simulated data from GNPy to evaluate the accuracy of the overall model in a multi-band scenario. In summary, the relentless

growth in demand for Internet bandwidth and network efficiency is a driving force for ongoing research in the field of telecommunications, and particularly in the optical field. Implementing more accurate models that account for how the network infrastructure actually operates is the path that needs to be taken. At the same time, the continued development of innovative solutions to fully utilize the capacity of existing infrastructure ensures that the Internet backbone can meet the ever-increasing demand for high-speed data transmission.

Acknowledgements

Grazie alla mia famiglia, ai miei amici e al gruppo Planet.

Grazie per aver contribuito a plasmare la persona che sono oggi.

Grazie per avermi aiutato a raggiungere questo traguardo.

Grazie per il tempo che abbiamo condiviso.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Optical networks have become increasingly important due to the growth of digital technologies and the internet. As more people use high-bandwidth applications like streaming video, online gaming, and cloud computing, the demand for faster and more reliable network connections has grown. Optical networks are able to meet this demand, making them an essential part of modern telecommunications infrastructure. They are also used in a variety of other applications, including medical equipment, industrial automation, and scientific research. The increase of bandwidth request and the energy price growth, due to geopolitical and social factors, have accelerated research into methods to optimize existing infrastructure.

In this context GNPy, an open source software, has been developed. This software provide the tools to planning and optimize real-word mesh optical network.

My contribution was to develop a model and a way to characterize the gain profile of the optical amplifier. This was possible thanks to experimental measurements of different models of amplifiers and a dataset collected from an experimental setup composed of a real dimensional experimental setup. In the second part of my work, I focused on the validation of GNPy for the multi-band scenario, in particular for the combination of C and L band. The use of different bands increases the efficiency of the already installed network. This is possible thanks to the multiplexing performed by wavelength division multiplexing.

In the first part of Chapter 2 of this thesis is presented the analysis of the existing work on the characterization of the gain profile of the EDFA amplifier. Then is provided the new analytical model for the characterization of the EDFA with all the validation that has been carried out to verify the accuracy. A description of how to implement this model in the GNPy environment is given at the end of this chapter.

Chapter 3 describes the tuning process used to obtain the consistent data needed for the multiband validation scenario. A description of the obtained result is given.

The analysis of the GNPy simulation of the C+L band is present in the Chapter

4. Three different simulations have been performed, the simulations are incremental to respect the information provided. A final comparison is made between the simulation and the actual measurements, using different information for the simulation.

The research carried out in this thesis aims to be able to use the results obtained in a real scenario, the result obtained has an immediate application for improving the accuracy of GNPy.

Chapter 2

Theoretical background

2.1 History of the optical communication

Light has been used to transmit information since the beginning of civilization. Fire beacons, mirrors, and smoke signals were a means of communication among various ancient peoples. The use of light in free space was one of the first means of exchanging information between people in different places. As in modern communication, the rule of communication should be established before communication takes place. The need to share more data led to the development of other strategies, such as the use of colored smoke [1], similar strategy are now used in communications technology to increase the capacity of the network.

Over the past 60 years, the evolution of telecommunications has been tied to the technological evolution of fiber optics. In the 1960s, the first steps in optical communications technology began with the invention of the laser. But the optical fibers available at the time had high losses, limiting their use to short-range data transmission, such as in gastroscopes.

In 1970, a significant advancement took place when a research paper scientists from Corning industry revealed their success in reducing fiber losses to under $20\,\mathrm{dB\,km^{-1}}$ at approximately 630 nm wavelength [2]. Further improvement came in 1979 when a Japanese research team successfully reduced optical fiber loss to nearly $0.2\,\mathrm{dB\,km^{-1}}$ in the infrared wavelength range around 1.55 µm[3].

The 1980s saw the first commercial deployment of fiber optic communications systems, primarily in long-haul telecommunications networks.

In the 1990s Wavelength Division Multiplexing technology emerged, allowing multiple data streams to be transmitted simultaneously over a single optical fiber. This dramatically increased the capacity of fiber optic networks.

In the early 2000s the deployment of optical amplifiers, such as Erbium-Doped Fiber Amplifiers (EDFAs), became common. Optical amplification helped overcome signal attenuation, enabling longer-distance transmission without the need for frequent signal regeneration. In the Mid to Late 2000s Dense Wavelength Division Multiplexing technology further increased the capacity of optical fibers by densely packing more channels into the available spectrum.

2.2 Optical network elements

At the heart of optical networks are various elements, often developed in a closed environment. Modeling this element is a key component of optical research. The phenomena act on the signal and propagate their effect through all the network elements it crosses. To fully understand these complex phenomena, a general view of the optical system is required.

The models of the elements that make up the optical infrastructure must be accurate and, more importantly, validated to understand how reliable they are.

2.2.1 Transceiver

The transceiver serves as the interface between fiber optics and electronic devices. Transceivers provide full-duplex communication, meaning they can transmit and receive data simultaneously because the receive and transmit circuits are independent. Today, they are plug-in devices that are inserted into a standardized transponder. In the Figure 2.1 the physical object that it is insert the transponder.

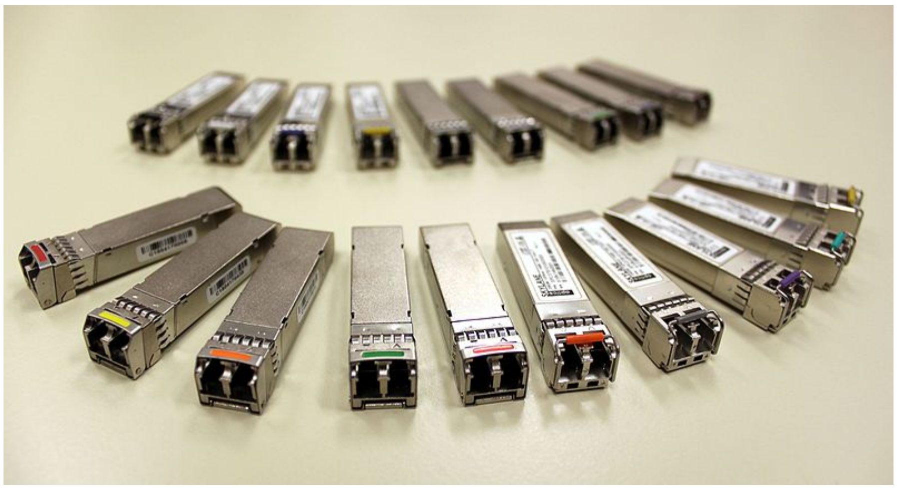


Figure 2.1: Picture showing standard transceivers. Source: [4]

2.2.2 ROADM

ROADM stands for Reconfigurable Optical Add-Drop Multiplexer. It is a key component in fiber optic networks, especially in wavelength division multiplexing (WDM) systems. There is a ROADM for each direction a node has on the network. ROADM can add new data streams to the existing fiber by assigning them specific wavelengths. It can remove specific wavelengths from the optical signal and direct them to a specific target. It can also allow certain wavelengths to pass through the node unaltered.[5]

2.2.3 Optical amplifier

Optical amplifiers increase the power of optical signals without converting them into electrical signals. This allows for the transmission of signals over longer distances without the need for frequent regeneration Optical amplifiers have specific gain spectra, indicating the range of wavelengths over which they can effectively amplify signals.

The noise figure of an optical amplifier measures the amount of additional noise introduced during the amplification process. There are different type of optical amplifier that exploit different physical effects.

Erbium-Doped Fiber Amplifiers (EDFA)s are the most common type of optical amplifiers used in long-haul and metro optical networks. They use erbium-doped optical fibers to amplify signals in the 1550 nm wavelength window, which is the low-loss region of optical fibers.

Semiconductor Optical Amplifiers (SOA)s use semiconductor materials to amplify optical signals. Compact and versatile, they are often used in short-range applications such as optical switches and wavelength converters.

Raman amplifiers use the Raman effect, in which photons interact with the vibrational modes of the material, to amplify signals. [6] They can provide amplification across a wide range of wavelengths and are often used in long-distance and undersea optical communication systems.

ASE

The Amplified Spontaneous emission (ASE) is generate in the Optical amplifiers. ASE noise is generated by the random emission of photons from the gain medium of an optical amplifier. These photons are amplified along with the signal light, and they add to the noise floor of the system. The gain of the amplifier and the bandwidth of the signal determine the amount of ASE noise generated.

The Figure 2.2 shown the schema of the optical amplifier. A gain and tilt are applied to the input signal, and the ASE noise, which is a function of the gain and tilt, is added to the result.

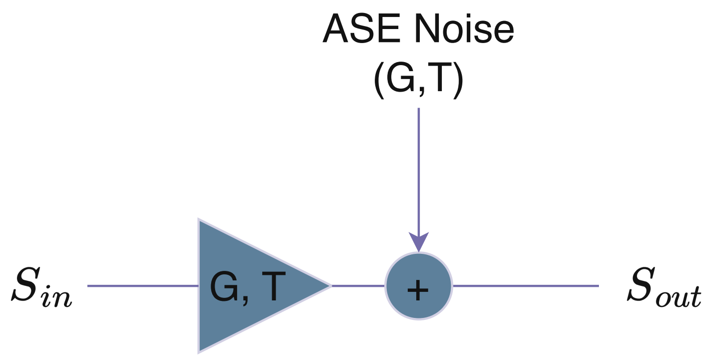


Figure 2.2: Operating diagram of an optical amplifier

2.2.4 Fiber

An optical fiber, also known simply as a fiber optic cable, is a thin, flexible, transparent fiber made of high-quality glass (silica) or plastic, through which light pulses are transmitted. The core is the central part of the optical fiber where light travels. It is made of pure glass or other transparent materials. In the case of glass fibers, the core is usually made of silica. Surrounding the core is the cladding, which is also made of glass or plastic. The cladding has a slightly lower refractive index than the core. This difference in refractive indices enables the phenomenon of total internal reflection, which keeps the light signals within the core by reflecting them back into the core. The core and cladding are protected by a thin layer of coating material, often made of acrylate or other polymers. This coating provides mechanical protection to the fragile glass fibers. The Figure 2.3 shows all the elements that make up the optical fiber cable.

Optical fibers can support multiple modes of light propagation. Single-mode fibers allow only one mode of light to travel, enabling higher bandwidth and longer transmission distances. Multi-mode fibers allow multiple modes, suitable for shorter distances.

Effective area

The effective area of an optical fiber refers to the cross-sectional area through which light is effectively transmitted within the core of the fiber. It is a crucial parameter in optical fiber design, particularly in the context of high-power transmission and nonlinear effects. The effective area is a parameter that depends on several factors. It can be roughly approximate as $A_{eff} = \pi r^2$

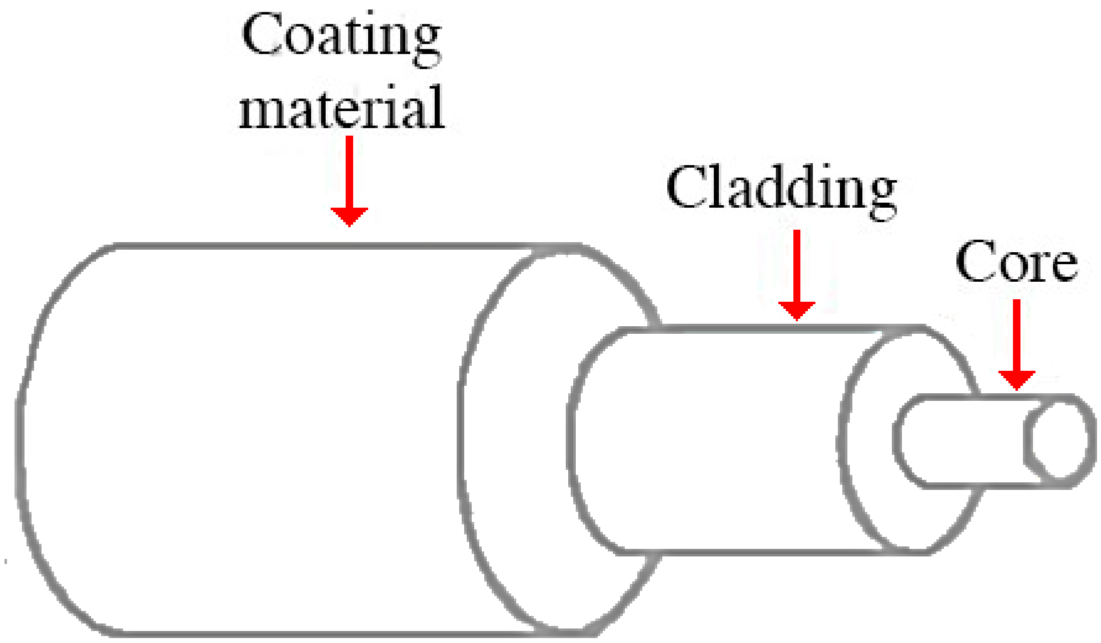


Figure 2.3: Scheme of the composition of an optical fiber cable

Kerr Effect

The Kerr effect, specifically the Kerr non-linearities, play a significant role in the field of fiber optics. Named after the physicist John Kerr, this effect refers to the phenomenon where the refractive index of a material changes in response to the field power $P(z,t)$. [7] In the context of optical fibers, the Kerr effect is a type of nonlinear optical effect that becomes prominent at high optical power levels. It induces a change in the refractive index of the glass:

$$n(z,t) = n_L + n_2 \frac{P(z,t)}{A_{eff}} \tag{2.1}$$

Chromatic dispersion

In general, chromatic dispersion CD is an effect due to the dispersive nature of the medium. Chromatic dispersion distorts the pulse by causing different delays for each spectral component based on their local group delay, and by introducing a phase mismatch between individual spectral components. The accumulated chromatic dispersion is completely compensated by the Digital Signal Processing (DSP) unit inside the receivers.

Loss coefficient function

In optical communication, the loss coefficient function, represents the attenuation or loss of optical power as a function of frequency. This coefficient characterizes how much optical power is lost per unit length at a specific frequency. Losses can occur due to various factors, such as absorption, scattering, and bending of optical fibers. The loss coefficient function is highly frequency-dependent. Different types of optical fibers (single-mode, multimode, specialty fibers) have varying loss coefficients due to their core and cladding materials, dopants, and manufacturing processes. It may be affected by environmental conditions such as temperature, pressure, and humidity.

Stimulated Raman Scattering

Stimulated Raman Scattering (SRS) is a nonlinear optical process in which incident photons interact with the vibrational modes of a material (such as a crystal or a gas) and transfer energy to these vibrational modes. This interaction leads to the generation of new photons with lower energy (longer wavelength) and the creation of a Stokes shift. The process is called stimulated because it requires the presence of an external photon source to initiate the scattering process. The frequency shift between the incident photon and the generated Stokes photon corresponds to the vibrational frequency of the material. This frequency shift provides valuable information about the molecular structure of the material being studied. SRS is widely used in Raman spectroscopy to analyze the vibrational and rotational modes of molecules. It provides detailed information about the chemical composition and molecular structure of materials. SRS can occur in optical fibers, leading to signal degradation. Analytical formulas are available to predict the behavior of this effect in optical fibers. [8]

2.3 Improvement of optical networks

The infrastructure improvement is composed of different elements: increase reliability, reduce fault recovery time, allow building piece of infrastructure with components from different vendors and reduce the energy consumption of the system.

By automating network control, the process of recovering from a failure can be accelerated, increasing the resilience of the system. To implement this, the Software Defined Network (SDN) paradigm must be used. In this way, the data plane is decoupled from the control plane, allowing the implementation of automatic fault detection and recovery procedures. [9]

The ability to create a vendor-neutral system is necessary to be vendor-agnostic and allow the operator to make the most of the hardware available. To achieve this, a model for each network element is required. The YANG model provides a standardized way to model all components of the optical network. It describes the configuration parameters of the optical devices, topology and connectivity information [10].

Reducing operational expenditure (OPEX) is a key objective for telecom operators. To achieve this, better utilization of the existing infrastructure is required. From a telecommunications point of view, reducing the margin that the operator has to guarantee the reliability of the communication can generate an increase in the capacity of the existing infrastructure. This is possible if the operators have the tools that provides information related the quality of the channel starting from the information of the existent infrastructure. To do so a model and a way to characterize all the component of the network it necessary. GNPy open source software that implement a digital twin of the optical network. It provides the model of all the component of the optical network that must feed with the parameters of the real infrastructure and it return the estimation of the quality of transmission Generalized Signal-to-Noise Ratio (GSNR).

2.4 Software defined network

Software-Defined Networking (SDN) is a new paradigm for designing, deploying, and managing computer networks. It decouples the control plane from the data plane in network devices (such as switches and routers), allowing network administrators to dynamically control and manage network traffic flows through software. In traditional networking, the control plane is embedded within the networking hardware, making it inflexible and limiting the ability to adapt to changing network requirements. In SDN, the control plane is separated from the data plane and moved into a centralized controller. The data plane, also known as the forwarding plane, is responsible for packet forwarding. In SDN, it remains in the networking hardware, which is responsible for processing and forwarding data packets based on the instructions received from the centralized controller. The SDN controller is the centralized brain of the SDN architecture. It acts as the control plane and communicates with networking devices through open protocols like OpenFlow. The controller makes decisions about how network traffic should be forwarded and communicates those instructions to the networking hardware. With a centralized controller, network administrators have a complete view of the network and can make global policy changes. SDN allows the development of network applications that can interact with the SDN controller to implement specific network services and policies. These applications can be created to address various networking needs, such as load balancing, security, and traffic optimization. SDN abstracts the underlying network infrastructure, making it easier to manage and configure networks. Network administrators can define policies and rules at a high level without needing to deal with the complexities of individual network devices.

SDN makes it easier to adapt to changing network requirements by dynamically adjusting network configurations through software. It enables better resource allocation and traffic optimization, leading to improved network performance. SDN can automate many network management tasks, reducing human error and operational overhead. The use of a standard environment and the transition from hardware to software implementation of the function allows the use of components provided by different vendors, reducing CapEX and OpEX costs [11]. SDN has found applications in data centers, wide-area networks, cloud computing, and network virtualization, among other areas, and it continues to evolve as a foundational technology in modern networking. In the optical field the paradigm of SDN is usefull to exploit the desegregation of the network [12].

2.5 Optical Transmission

2.5.1 Wavelength Division Multiplexing

WDM transmission is often referred to as colored transmission due to the analogy between wavelengths used in the optical spectrum and colors visible to the human eye. Each data stream is assigned a unique wavelength within the optical spectrum. Lasers are used to generate optical signals at different wavelengths [13]. Each laser corresponds to a specific data channel. These optical signals, each at a different wavelength, are combined into a single optical fiber using a device called an optical multiplexer. The multiplexer combines the individual optical signals into a composite signal for transmission. At the receiving end, an optical demultiplexer is used to separate the combined optical signal back into its individual wavelengths. In Figure 2.4 the schema is displayed: Coarse Wavelength Division Multiplexing

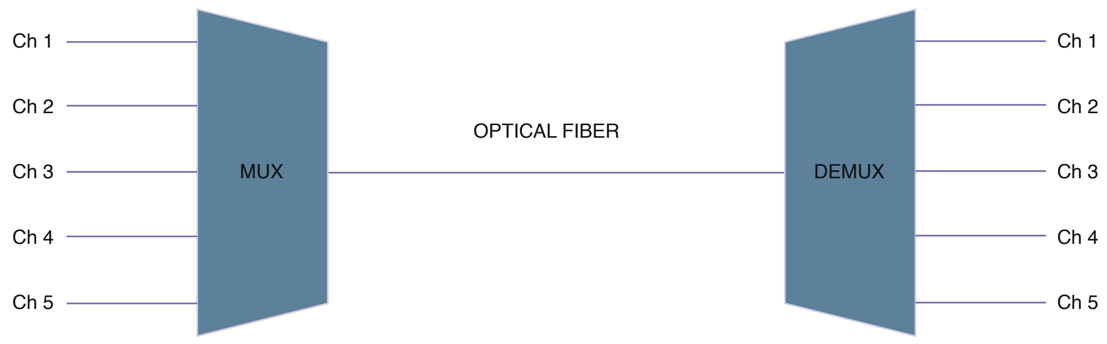


Figure 2.4: WDM schema

(CWDM) involves a limited number of channels, ranging from 2 to 16, widely spaced apart with a spacing of 20 nm. In contrast, Dense Wavelength Division Multiplexing (DWDM) accommodates a higher number of channels, reaching up to 80-100, closely packed together. In commercial optical systems, it's common to combine up to 80 wavelengths on a single fiber. Each wavelength operates at $10\,\mathrm{Gbit\,s^{-1}}$, resulting in a total capacity of $800\,\mathrm{Gbit\,s^{-1}}$ per fiber. The Wavelength Division Multiplexing (WDM) grid, indicating the channel spacing, is standardized by the ITU-T in multiples of 12.5 GHz.

2.5.2 Modulation format

The use of a coherent receiver allows the use of a multilevel modulation format. Unlike direct detection intensity modulation (IMDD), multilevel modulation formats use more than two levels to represent more than one bit per symbol. This enables higher data rates and increased spectral efficiency. Another improvement that increases spectral efficiency is the use of dual polarization of the modulation format. The used modulation format are the DP-QPSK, DP-16QAM and the the DP-64QAM. In Figure 2.5 the I/Q graph for 16 QAM modulation with Gray codes is shown. Gray codes are binary sequences in which neighboring symbols differ by only one bit. These codes are particularly useful in applications where it is important to avoid multiple bit changes, such as mechanical encoders, digital communication systems, and error detection.

2.6 Digital twin

A digital twin is a virtual replica or simulation of a physical object, system, or process. It is created using data and digital models to represent the object or system in a virtual environment. This technology enables engineers and designers to test and optimize the performance of physical objects, systems, and processes without the need to create physical prototypes or perform physical testing. Digital twins can be used in a wide range of applications, from manufacturing and construction to healthcare and transportation. For example, in manufacturing, a digital twin can be used to simulate the performance of a machine or production line before it is built, allowing engineers to identify and fix potential issues before they arise. In healthcare, a digital twin can be used to create a virtual model of a patient, which can be used to test and optimize treatments and procedures. Overall, digital twin technology offers significant benefits in terms of efficiency, cost savings, and risk reduction [14].

The digital twin must implement the model that characterizes the physical behavior of the real system. This model must be tested with the real measurements to check if it has an accuracy that respects the needs of the particular application.

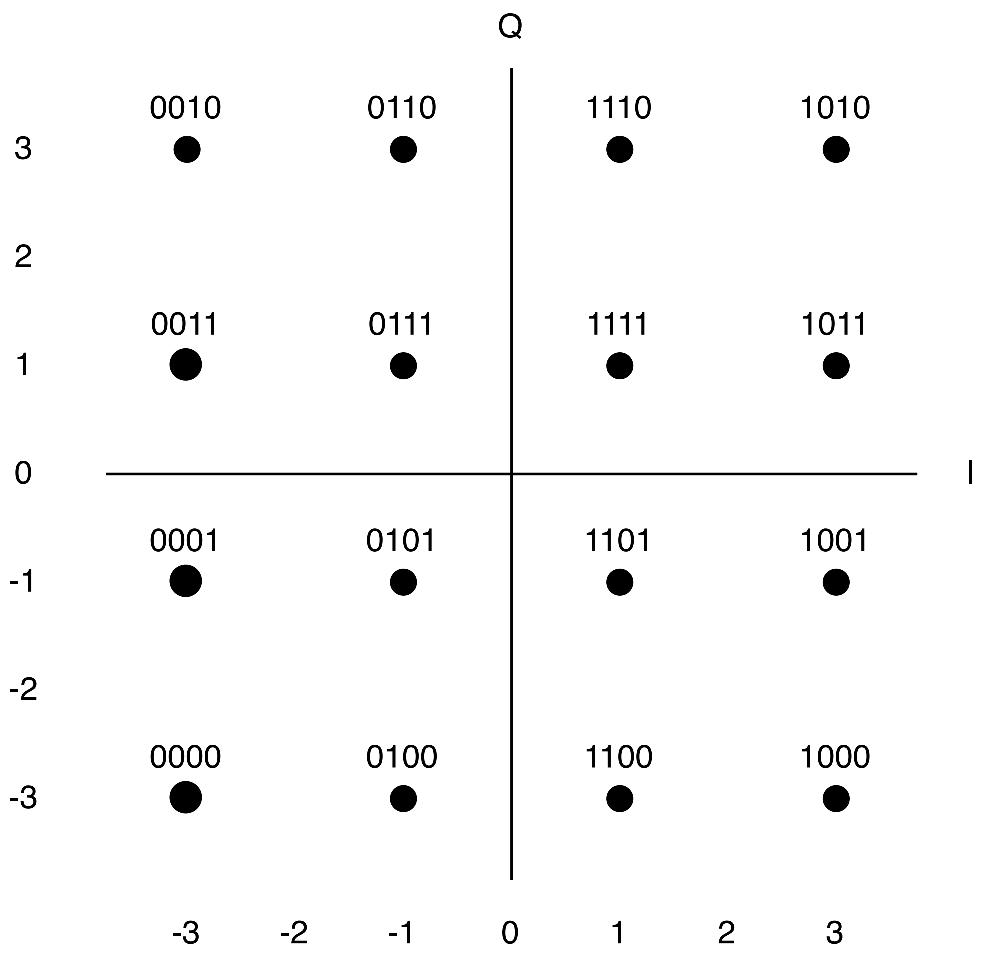


Figure 2.5: I/Q graph for 16-QAM with Gray code

At the same time, in the operational use of the digital twin, the parameters that need to be used in the model must be easy to find.

2.6.1 Optical network digital twin: GNPy

GNPy is an open source library that implement models to characterize all the phenomenons at the physical layer of an optical network. The physical behaviour that has to be considered in the optical field are: the amplified spontaneous emission (ASE), the Non-Linear Interference (NLI), stimulated Raman scattering (SRS), loss coefficient function and the chromatic dispersion (CD). The aim is to obtain a value of GSNR similar to the value of the real network or at least smaller to respect the real value. In this way is possible to exploit all the capacity of the network. It is possible to select the correct modulation format by knowing the exact value of GSNR. The efficiency can be increased because relay on an accurate evaluation of the GSNR can lead to a reduction of the safety margin on the choice of modulation format. GNPy returns the GSNR value by entering the network element parameters.

In general, there are 2 different json input files for GNPy configuration: the equipment configuration file contains the description for each variant of each element, and the topology file contains the topology of the network with the element present in the equipment configuration. The elements are the Amplifier(EDFA), the Fiber, the RamanFiber, the Span, the ROADM, the Fused and the Transceiver. For each of them different variants are defined. For a specific element, you can define a file with parameters that are specific to that element [15].

2.7 SNR in the optical world

In a communication system the Signal to Noise Ratio SNR is the ratio of the power of the signal to the power of the noise present in the communication channel (evaluated on a specific bandwidth). The output of the digital twin is a parameter called GSNR. An accurate evaluation of this parameter allows to the operators to increase the efficiency of the network. The $GSNR$ is:

$$GSNR = \frac{P_{ch}}{P_{ASE} + P_{NLI}} \tag{2.2}$$

Where P_{ch} is the signal power, P_{ASE} is the Amplified Spontaneous Emission power and the P_{NLI} is the power due to the Non-linear Interference (NLI). The P_{NLI} is $P_{NLI} = N_s \eta P_{ch}^3$, which means that an increase in signal power does not correspond to an increase in $GSNR$. There is an optimal value of power that produces an optimal value of $GSNR$. In the Figure 2.6 is shown the behaviour of the function:

$$GSNR = \frac{P_{ch}}{P_{ASE} + N_s \eta P_{ch}^3} \tag{2.3}$$

The Optical Signal to Noise Ratio ($OSNR$) is a parameter that can be written as $OSNR = P_{ch}/P_{ASE}$, instead the (SNR_{NLI}) is the Signal to Noise Ration due to the NLI. The $GSNR$ can also be written:

$$GSNR = (OSNR^{-1} + SNR_{NLI}^{-1})^{-1} \tag{2.4}$$

The total SNR in the optical communication field is defined as:

$$\frac{1}{SNR} = \frac{1}{GSNR} + \frac{1}{SNR_{RX}} + \frac{1}{SNR_{TX}} \tag{2.5}$$

where ${SNR_{RX}}^{-1}$ and SNR_{TX}^{-1} are the electrical SNR of the receiver and transmitter, respectively.

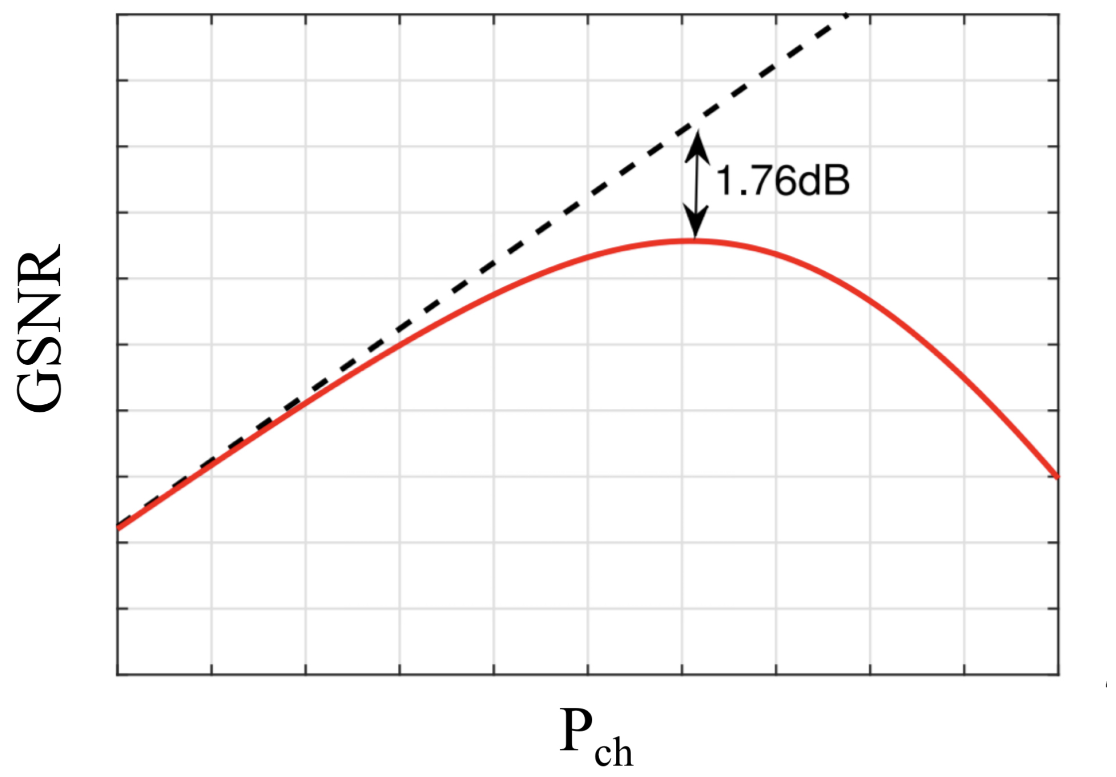


Figure 2.6: The red curve shows the trend of the GSNR function as a function of channel power, while the dotted line shows the linear behavior of the OSNR component.

2.8 Evaluation of the noise figure

The noise figure of an amplifier is a measure of how much the amplifier degrades the SNR of the input signal. It quantifies the amount of additional noise introduced by the amplifier. A lower noise figure indicates better performance because it means the amplifier is adding less noise to the signal.

Mathematically, noise figure (NF) is defined as the ratio of SNR at the input to SNR at the output of the amplifier:

$$NF(f) = \frac{SNR_{out}(f)}{SNR_{in}(f)} \tag{2.6}$$

where $SNR_{out}(f)$ is the SNR at the output of the amplifier and $SNR_{in}(f)$ is the SNR at the input of the amplifier. In practice, the noise figure can also be defined using noise temperatures.

$$NF(f) = 1 + \frac{T_{out}(f)}{T_{in}(f)} \tag{2.7}$$

where T_{in} is the input noise temperature (in Kelvin) and T_{out} is the output noise temperature. For the optical amplifier, the noise figure can be evaluated in this way:

$$NF(f) = \frac{P_{ase}^{out}}{hfB(G(f)-1)} \tag{2.8}$$

where P_{ase}^{out} is the output noise of the amplifier, h is the Plank constant, B is the bandwidth of the amplifier, and $G(f)$ is the gain as a function of frequency.

Chapter 3

Erbium-Doped Fiber Amplifier

An Erbium-Doped Fiber Amplifier, EDFA is a device used in optical communication system to amplify optical signals. It is one of the most widely used types of optical amplifiers due to its high gain, low noise figure, and compatibility with a wide range of wavelengths. The basic principle behind an EDFA is the phenomenon of stimulated emission. The erbium ions, which are embedded in the fiber core, can be excited by pumping them with light energy of a specific wavelength. Typically, pump lasers operating at 980 nm or 1480 nm wavelengths are used for this purpose. When the pump laser light is coupled into the erbium-doped fiber, it energizes the erbium ions, causing them to transition to higher energy states. When an input signal passes through the erbium-doped fiber, the energized erbium ions can interact with the signal and release additional photons that are coherent with the input signal. This process amplifies the input signal. In this work has been studied the two-stage configuration of the EDFA, in particular has been analysed 2 family of amplifiers and for each family different devices. The two-stage configuration offers several advantages over a single-stage EDFA. Firstly, it allows for higher overall gain. Each stage contributes its individual gain, and the gains of the two stages are combined, resulting in a higher net gain for the system. This is particularly beneficial when amplifying signals over long distances or compensating for significant losses in the optical fiber link. Secondly, the two-stage configuration helps reduce the noise figure of the amplifier system. The noise figure represents the amount of additional noise added to the signal during amplification. In a cascaded configuration, the noise contribution from each stage is reduced due to the gain provided by the preceding stage. As a result, the overall noise figure of the two-stage EDFA is lower compared to a single-stage EDFA. Another advantage of the two-stage EDFA is that it allows for greater flexibility in controlling the gain and power levels. By adjusting the gain of each stage independently, it becomes easier to optimize the amplification for different signal power requirements and link conditions.

3.1 Amplifier model

EDFA characterization refers to the process of evaluating and analyzing the performance characteristics of the devices. The characterization process typically involves different parameter: gain profile, noise figure, polarization dependence, input/output power dependence, wavelength dependence and temperature dependence. In this work, the characterisation of the gain profile has been analysed in order to obtain an accurate model for simulating this parameter. The gain profile is used to quantify the amplification of the amplifier. The term gain is used because it is defined as the ratio of output power to input power. The term profile indicates that it is a frequency dependent parameter. The gain profile is a transfer function, so multiplying the input power by the gain profile is possible to obtain the output power of the amplifier. The decibel scale is used for the gain profile in all of the following treatments. The theoretical characterization of an ideal amplifier gain profile can be done using this equation:

$$G(f;G,T) = G + \frac{T}{B}(f - f_c) \tag{3.1}$$

where: $G(f;G,T)$ is the gain profile expressed in dB as a function of frequency; G is a scalar value representing the gain target expressed in dB; T is the tilt target expressed in dB; B is the operational amplifier bandwidth expressed in Hz; f_c is the center frequency in Hz, the pivot point of the tilt profile; f is a variable representing the frequency in Hz. In Figure 3.1 the model of the ideal amplifier is shown. The continuous line represents the gain profile as a function of frequency. The centre frequency is the pivot point of the slope (T), at which point the value of the gain profile is the gain (G). The real model of the amplifier can be written as the ideal model with the addition of a ripple that depends on the frequency:

$$g(f;G,T) = G + \frac{T}{B}(f - f_c) + r_T(f) \tag{3.2}$$

where $r_T(f)$ is the frequency dependent ripple expressed in dB that characterize the specific devices. To obtain an accurate characterization of a specific amplifier $r_T(f)$ has to been evaluated for all specific amplifier.

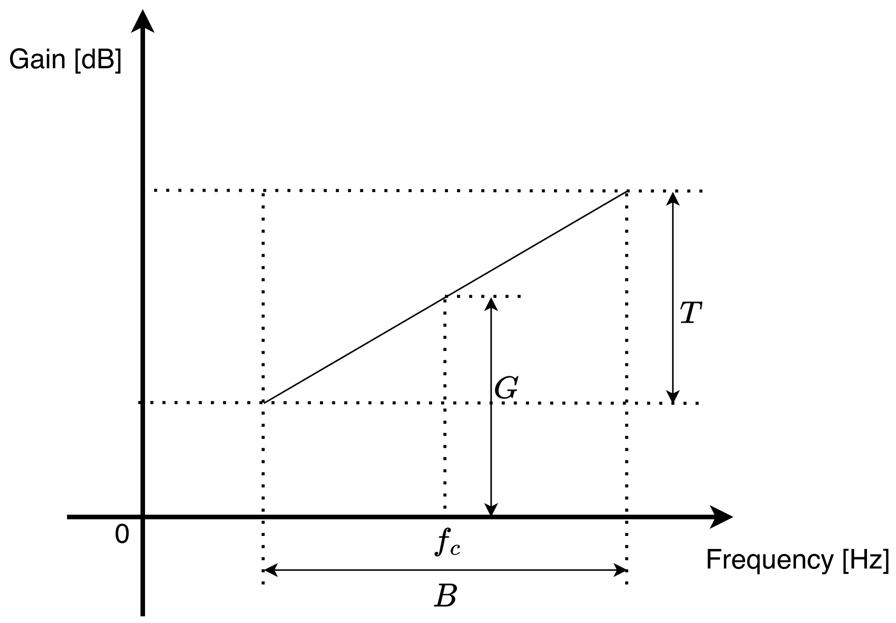


Figure 3.1: Ideal amplifier gain profile model

3.2 State of the art EDFA characterization

3.2.1 Machine learning characterization

Various machine learning techniques have been developed to characterise the gain profile parameters. In [16], a deep neural network has been used to predict the gain profile for each individual EDFA based on different channel loads of the input spectra. The characterisation provides a frequency dependent gain model which has been compared with the analytical centre of mass (CM) model. The CM model has been described in [17]. The Equation 3.3 describes a gain model of the EDFA for multiple wavelength input:

$$\hat{g}(\lambda_i) = g(\lambda_i) + \frac{\Sigma_{j=1}^{n}\{g_s(\lambda_j) - g(\lambda_j)\}}{n} \tag{3.3}$$

where $g_s(\lambda_i)$ is the gain when only λ_i is the input to the EDFA, $g(\lambda_i)$ is the gain in the full spectral load scenario. A neural network (NN) has been used with ninety features, one for each channel power level. The training process has been aimed at minimizing the Mean Square Error (MSE). The result shows a reduction in the RMSE between the analytical model and the machine model from 51.71 % to 9.44 %. The problem with this approach is the large amount of data required for the machine learning training and prediction process. A hybrid approach has been used in the [18] to solve the same problem, trying to create an accurate model to predict the power output changing the input power of the EDFA. The CM model is used as input to the machine learning model. This reduces the size of the training sample. Another limitation of these analyses is the fact that in the real case scenario the channels are always allocated and there is no dynamic allocation of spectrum. In general, the system operates at full spectral load.

The [19] proposes a supervised machine learning model to characterize the gain ripple and filter penalties. The main idea of this model is to use a monitoring system for the network that provides information to the machine learning algorithm through a control plane. To provide this information, Optical Channel Monitors (OCM) are installed in the network to provide filter penalty and power profile information. This approach gives you near real-time information that takes into account how device behavior changes over time. However, this means that a monitoring infrastructure must be put in place.

3.3 Dataset pre-processing

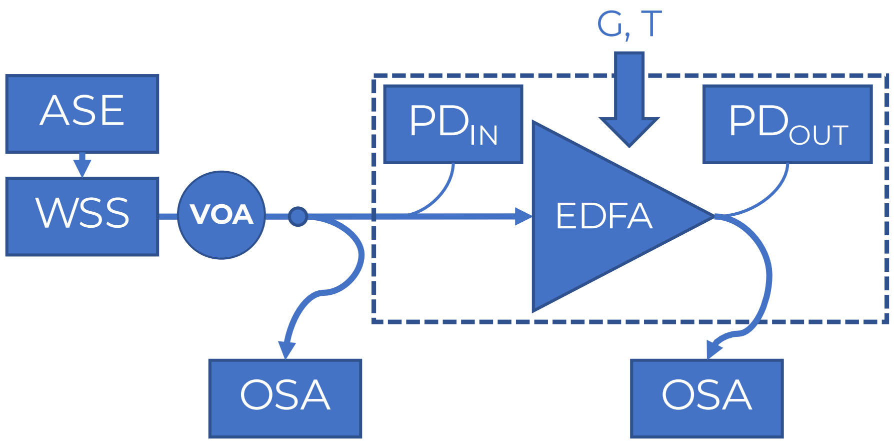


Figure 3.2: Schematic block of the device used in the measurement configuration

In order to validate all the models to be used in the next step, a complete experimental characterization of four different EDFA families (hereafter referred to as EDFA 1, EDFA 2, EDFA 3 and EDFA 4) is performed. Each family consists of different elements, for each of which experimental data-sets have been collected containing different gain profiles corresponding to specific combinations of total input power, target gain, and tilt parameters. The configuration setup for the acquisition of the dataset is depicted in Figure 3.2. To shape the ASE noise and create a Wavelength Division Multiplexed (WDM) comb, a commercial Wavelength Selective Switch (WSS) is programmed. The WSS is configured to generate either a 40-channel or 48-channel WDM) comb, with channels spaced at 100 GHz. Each channel is modulated at a rate of 32 GBaud. This programming of the WSS allows for the provision of two different spectral loads at the input of the EDFAs, based on the specifications of each device. Specifically, the spectral load provided is 4 THz for EDFA1 and 4.8 THz for EDFA2. The Secure Shell (SSH) protocol is used to control each amplifier, allowing secure remote access to the amplifiers. SSH can be used to adjust the gain and tilt parameters of each amplifier. In addition, the optical input power of each amplifier can be changed by manipulating the Variable Optical Attenuator (VOA) located in front of the amplifier. The optical spectrum at both the input and output of the EDFA is measured using an Optical Spectrum Analyser (OSA). This device has an absolute power uncertainty of ± 0.1 dB The Table 3.1 shows all the values used to collect the data set. The datasat has been collected by varying the 3 parameters: input power, target gain and Tilt. The differences from one family to another are due to the different operating range of the item.

	Input power [dBm]	Target gain [dB]	Tilt [dB]
EDFA 1	[−10:−4:+2]	[12:27:+1]	[−5:+3:+1]
EDFA 2	[−4:−1:+1]	[19:25:+1]	[−5:+2:+1]
EDFA 3	[−10:+0:+2]	[10:20:+1]	[−3:+3:+1]
EDFA 4	−10	[17:30:+1]	[−3:+3:+1]

Table 3.1: [X:Y:Z], where X is the start value, Y is the stop value, and Z is the step

Additional information is added externally to respect the data set. This prevents the measurement from being affected by saturation due to tilt. This information is related to the operating range of the EDFA. It is used to discard profiles that are outside this range.

3.4 Bandwidth and center frequency

The center frequency and the bandwidth are 2 characteristic parameters of the amplifier. These parameters must be the same for the characterization of the amplifier and for the evaluation of the generic gain profile. The values used are evaluated from the channel assignment standardized by ITU. The bandwidth value for L-band and C-band has been evaluated as the difference between the lower frequency channel and the higher frequency channel. For both bands, the bandwidth is 4.9 THz. The center frequency for the L band is 188.6 THz instead for the C band is 193.6 THz. This value is the center frequency between the value of the lower frequency channel and the higher frequency channel.

3.5 Dynamic Gain Tilt

The DGT [20] is a parameter that can be used to characterise an amplifier with a set of measurements. The formula for evaluating the DGT is:

$$dgt(f) = \frac{\Delta g(f; G, T)}{\Delta g(fe; G, T)} \tag{3.4}$$

The numerator is a vector representing the difference between two specific gain profiles, the denominator is a scalar representing the difference between the value of the same gain profile at a specific frequency. In order to obtain a value of DGT that includes the information related to the specific ripple of the EDFA, one of the gain profiles must have a tilt of 0 dB and both profiles must be measured at the same gain value. Using the model described in the Equation 3.2, the DGT is:

$$dgt(f) = \frac{\frac{T_j}{B}(f - f_c) + r_j(f) - \frac{T_i}{B}(f - f_c) - r_i(f)}{\frac{T_j}{B}(fe - f_c) + r_j(fe) - \frac{T_i}{B}(fe - f_c) - r_i(fe)} \tag{3.5}$$

If T_j is 0 and both profile has the same gain:

$$dgt(f) = \frac{r_j(f) - \frac{T_i}{B}(f - f_c) - r_i(f)}{r_j(fe) - \frac{T_i}{B}(fe - f_c) - r_i(fe)} \tag{3.6}$$

It is not possible to evaluate the generic ripple that characterises the amplifier using the DGT definition and the real gain model. But with a different amplifier model, where the ripple parameter includes the dependence of B and f_c:

$$g(f; G, \tilde{T}_i) = (r(f) \cdot \tilde{T}_i) + G \tag{3.7}$$

where $\tilde{T}_i$ is a scalar value. The DGT can be written as:

$$dgt(f) = \frac{g(f; G, \tilde{T}_0) - g(f; G, \tilde{T}_i)}{g(fe; G, \tilde{T}_0) - g(fe; G, \tilde{T}_i)} = \frac{(r(f) \cdot \tilde{T}_0) - (r(f) \cdot \tilde{T}_1)}{r(fe) \cdot \tilde{T}_0 - r(fe) \cdot \tilde{T}_1} \tag{3.8}$$

If $\tilde{T}_0$ is 0 and both profile has the same G:

$$dgt(f) = \frac{-r(f) \cdot \tilde{T}_1}{-r(fe) \cdot \tilde{T}_1} = \frac{r(f)}{r(fe)} \tag{3.9}$$

The difference between the DGT and the ripple profile is due to the normalization factor $r(fe)$, which depends on the frequency at which the DGT has been evaluated. Knowing the DGT and the value of the ripple at the reference frequency fe:

$$r(f) = dgt(f)r(fe) \tag{3.10}$$

3.6 Dinamic Gain Tilt GNPy validation

The aim of this section is to check how the EDFA implemented in GNPy handles the DGT. For this, the `high_detail_model_example` variant of the EDFA has been used.

The first part of the analysis tested the DGT, Equation 3.4. A standard DGT profile has been used from GNPy to evaluate the gain profile. Then 2 different procedures has been carried out from the evaluated GNPy gain profile: the evaluation of the DGT and the evaluation of the DGT from the ripple profile. The workflow is shown in Figure 3.3 and the script used is shown in Section A.1. The

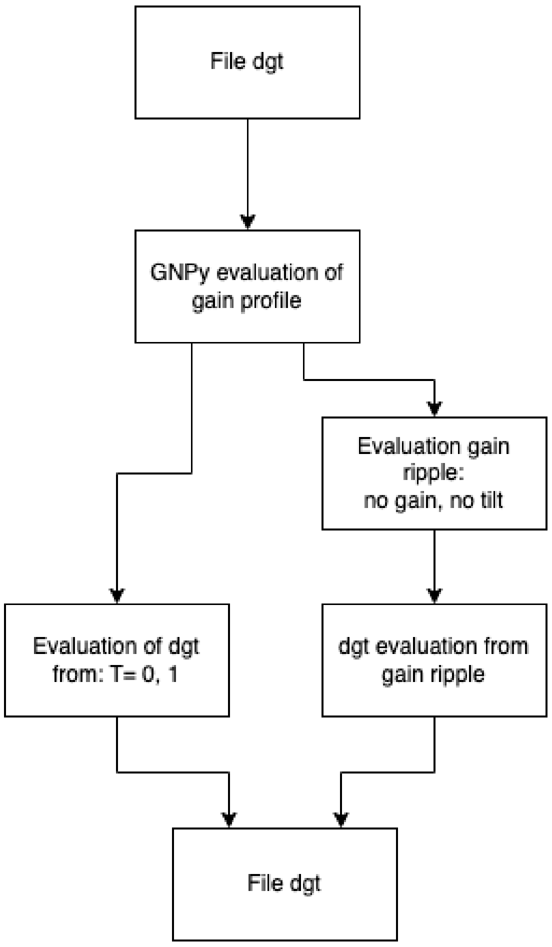

File dgt
GNPy evaluation of gain profile
Evaluation gain ripple: no gain, no tilt
Evaluation of dgt from: T= 0, 1
dgt evaluation from gain ripple
File dgt

Figure 3.3: Work flow of DGT evaluation

first procedure produces a DGT profile that is exactly the same as the default one. This can be seen in: Figure 3.4 The reference frequency fe used for the evaluation has been the smallest one, as can be seen from the fact that the smallest frequency

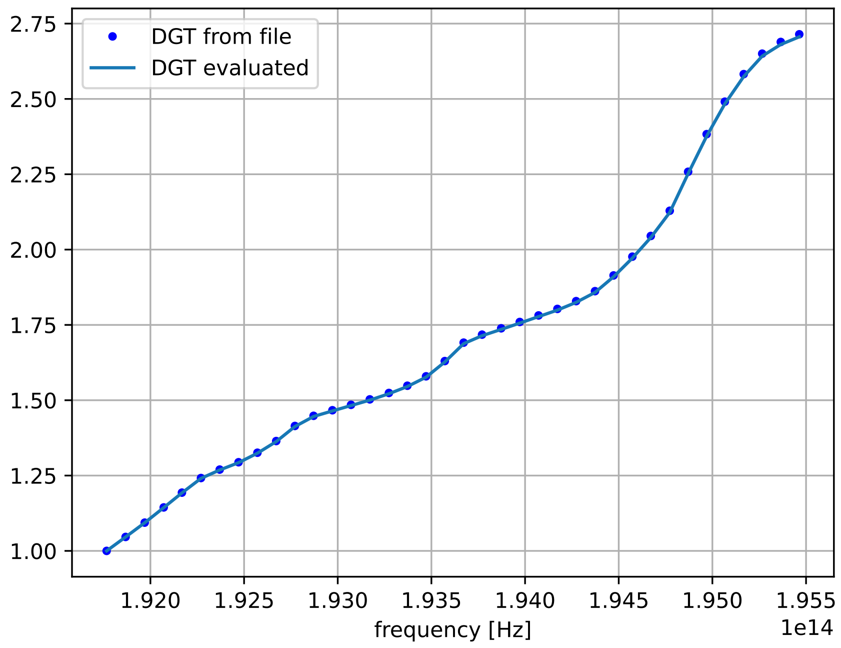


Figure 3.4: Comparison between default file and GNPy evaluated gain profile

has a value of 1. The 2 gain profiles used for the evaluation have the same G and the value of T is 0 dB and 1 dB.

The second procedure has been used to test the Equation 3.10. The Figure 3.5 contains the default DGT, the normalized ripple profile and the DGT evaluated from the ripple profile. The ripple of the gain profile has been evaluated with Section 3.7.2. The ripple has been normalized to compare with the DGT standard profile (the shape and tilt are the main parameters). The Figure 3.5 shows the difference between the ripple profile and the DGT, where the inverse of the Equation 3.10 is used to evaluate the DGT from the ripple: $dgt(f) = \frac{r(f)}{r(fe)}$

In the second part of the analysis has been compare the gain profile evaluation from the GNPy and the real measurement from the dataset. The data file has been pre-processed and saved for comparison. The data has been filtered to exclude the value where the EDFA is outside the operational range. From the measurement has been evaluate the DGT parameter considering the 2 gain profile with tilt 0 dB and 1 dB. The value of DGT required by GNPy is interpolated at the frequency of the amplifier. For these reasons, the obtained DGT profile has been subjected to an interpolation operation before use. This value of DGT has been used to evaluate the gain profile by GNPy. The parameters that GNPy require for the evaluation

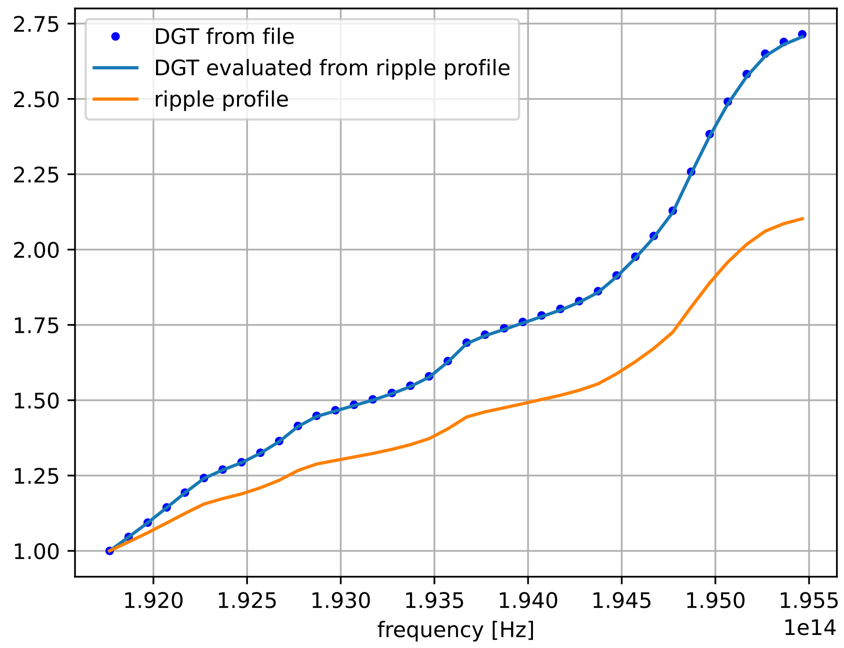


Figure 3.5: Comparison between standard DGT, ripple profile and DGT evaluated from ripple profile

of the gain profile are the gain G and the tilt T. For each value of gain, tilt and input power a gain profile has been evaluated and saved by GNPy. The function that generated the gain profile is `_gain_profile()` in `elements.py`. To evaluate the gain profile starting from the DGT, GNPy calculates the DGT slope and the tilt target slope. The DGT slope is the angular coefficient of the fitted straight line, the tilt target slope is the angular coefficient of the straight line where the x-axis is the bandwidth of the amplifier and the y-axis is the tilt target value. For a fixed value of Gain G and Tilt T, but changing the input power, the same value of Gain Profile has been obtained, meaning that the evaluation of the gain profile is independent of the input power. This can be seen in the Figure 3.6. The 2 curves match perfectly because the model implemented in GNPy does not take the input power into account. This behaviour can be considered correct, as the measurement shows that the gain profile does not depend on the input power. In Figure 3.7 it is possible to observe the gain profile measured with the same value of gain and tilt but with different input power. All the curves are within a range of 0.1 dB, which is in the range of the error measurement.

If the tilt target of the EDFA is set to 0 dB, the shape of the DGT is not

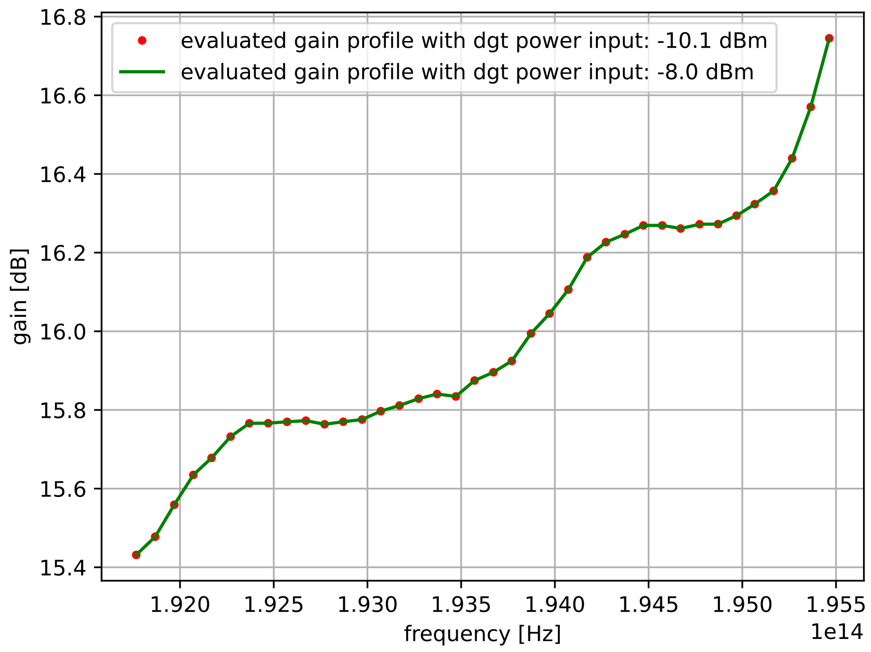


Figure 3.6: GNPy power comparison for 2 gain profile with different power and Gain 18 dB and Tilt −2 dB

applied to the gain profile, the only shape added is the gain ripple. To better show the behaviour of the gain profile evaluated by GNPy, a dynamic figure has been developed where it is possible to change the value of the tilt and the gain with 2 sliders, Figure 3.8. The python code developed is in Section A.2. In the figure above is show the comparison of the profile evaluate from GNPy and the profile measured. It is possible to observe a mismatch in the tilt of this profile. In particular in Figure 3.9 and Figure 3.11 it is possible to observe a slope of 0.5 dB for the profile evaluated by GNPy, instead the measured profile correctly has a tilt of 0 dB. There is the same behaviour in the Figure 3.10 and in the Figure 3.12. The measured gain profile and the evaluated one do not match.

This model mismatch can be explained by the equation that compares the DGT and the gain ripple, Equation 3.10. For this reason, a new methodology for characterizing and generating the gain profile has been developed.

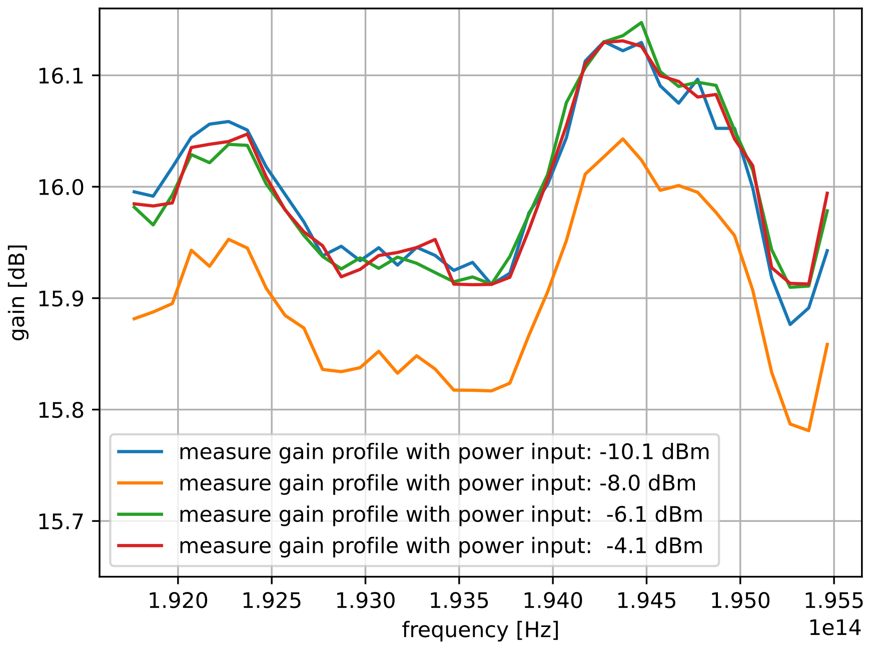


Figure 3.7: Measured power comparison for 4 gain profiles with different input power

3.7 New evaluation of gain profile

A semi-analytical EDFA gain profile model based on a two-measurement characterization phase for a two-stage EDFA has been described. Validation is performed in a full spectral load transmission scenario. This method provides an equation for generating a gain profile with a specific gain and tilt, taking into account the real amplifier model. The aim of this method is to obtain a value of the ripple profile, $r_i(f)$, of the Equation 3.2 that depends only on the tilt of the gain profile of the specific amplifier.

3.7.1 Characterization parameters

The $K(f)$ parameter has been defined has:

$$K(f) = \frac{r_0(f) - r_T(f)}{T} \tag{3.11}$$

Where: T is the tilt of one of the gain profiles, $r_i(f)$ is the ripple of the profile with tilt T and $r_0(f)$ is the ripple of the profile with 0 tilt. This parameter is tilt

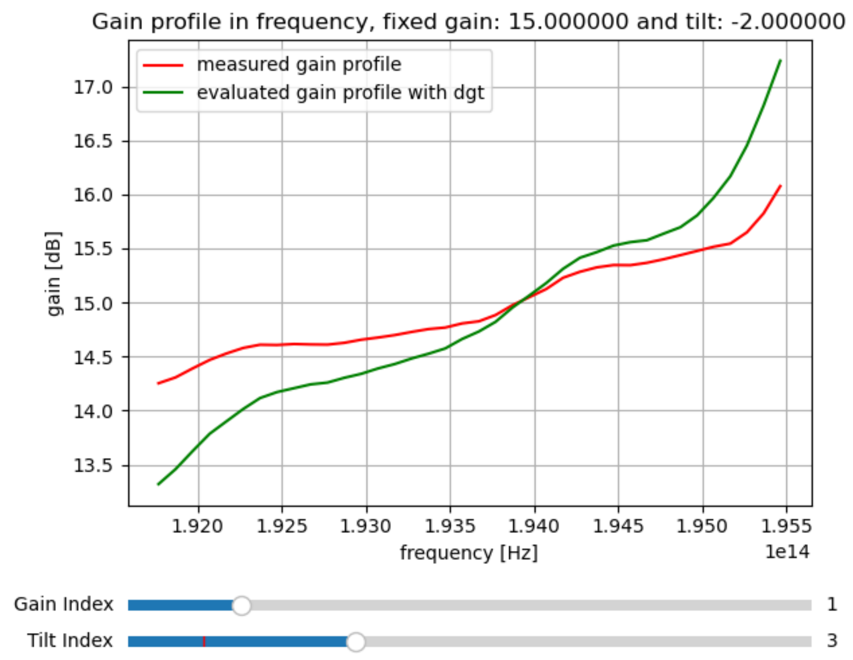


Figure 3.8: Dynamic comparison between gain profile from measurement and from GNPy

independent, the same $K(f)$ value is obtained if the profile used for the evaluation comes from the same EDFA and if the tilt T and $r_T(f)$ change accordingly. $r_0(f)$ is a specific parameter of the amplifier, it is the difference to respect the flattering filter applied on the EDFA. Considering a field scenario where it is not possible to perform a measurement for all the EDFAs. This relationship described in Equation 3.11 has been analysed by evaluating the $K(f)$ parameters for different gain and tilt values and then finding the differences. The script used is in Section A.3. In Figure 3.13 is shown $K(f)$ which has been evaluated for different values of T and for a fixed value of G, 14 dB. The difference between all the curves it is at most 0.05 which is in the range of the error measurement, showing that the parameter $K(f)$ is independent of the tilt.

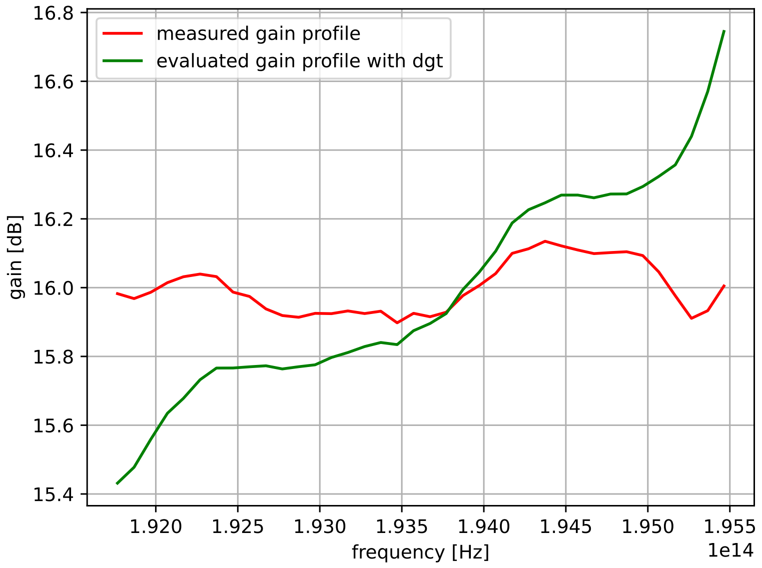


Figure 3.9: Comparison between the profile measured and the profile evaluated with DGT model in GNPy, gain: 16 dB, tilt: 0 dB

3.7.2 Evaluation of the characterization parameters

First method

It is possible to define a generic ripple profile from Equation 3.2 as:

$$r_i(f) = g(f, G, T_i) - G - \frac{T_i}{B}(f - f_c) \tag{3.12}$$

$K(f)$ and $r_0(f)$ parameters are needed to fully characterize an EDFA amplifier. The parameters f_c and B are fixed for a specific amplifier. The parameters G and T_i are evaluate from the measure of the gain profile $g(f, G, T_i)$ G is the mean of the profile. Considering a discrete gain profile (for each frequency correspond a value of gain) the mean can be evaluate as:

$$G = \frac{\sum_{f=0}^{n} g(f; G, T_i)}{n} \tag{3.13}$$

where n is the index representing the frequency of the discrete gain profile. The tilt T_i is the angular coefficient of the interpolating line of the gain profile. For the parameters G and T, the value inserted in the EDFA for the evaluation of the profile can be used. This approach is less accurate to respect the evaluation from the measured value.

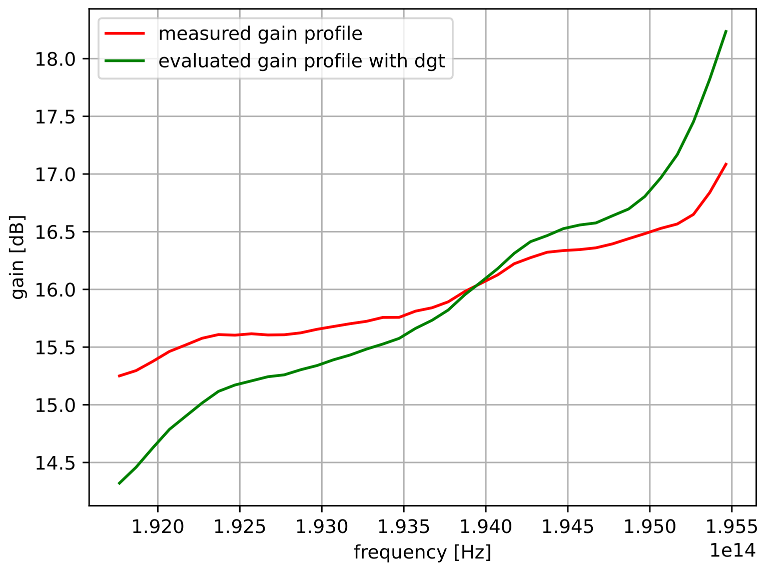


Figure 3.10: Comparison between the profile measured and the profile evaluated with DGT model in GNPy, gain: 16 dB, tilt: −2 dB

Starting with a two measure gain profile where T is set to 0 dB and a second profile where T is set to an arbitrary value, it is possible to obtain two ripple profiles, r_0 and r_T, thought the Equation 3.12. Knowing this parameter, it is possible to evaluate the parameter $K(f)$ using Equation 3.11, and $r_0(f)$ is:

$$r_0(f) = g(f; G, T_0) - G \tag{3.14}$$

The choice of the second tilt of the gain profile is arbitrary, but the experimental evidence has shown that a larger tilt results in a smaller relative measurement errors.

Second method

The $K(f)$ parameter can be evaluated in another way. If both gain profiles are measured at the same gain G and if T_0 is 0 dB:

$$g(f; G, T_0) - g(f; G, T_i) = r_0(f) - \frac{T_i}{B}(f - f_c) - r_{T_i}(f)] \tag{3.15}$$

Dividing Equation 3.15 for the T_i is possible to obtain an expression that contain the $K(f)$ parameters:

$$\frac{r_0(f) - \frac{T_i}{B}(f - f_c) - r_{T_i}(f)}{T_i} = \frac{r_0(f) - r_{T_i}(f)}{T_i} - \frac{1}{B}(f - f_c) \tag{3.16}$$

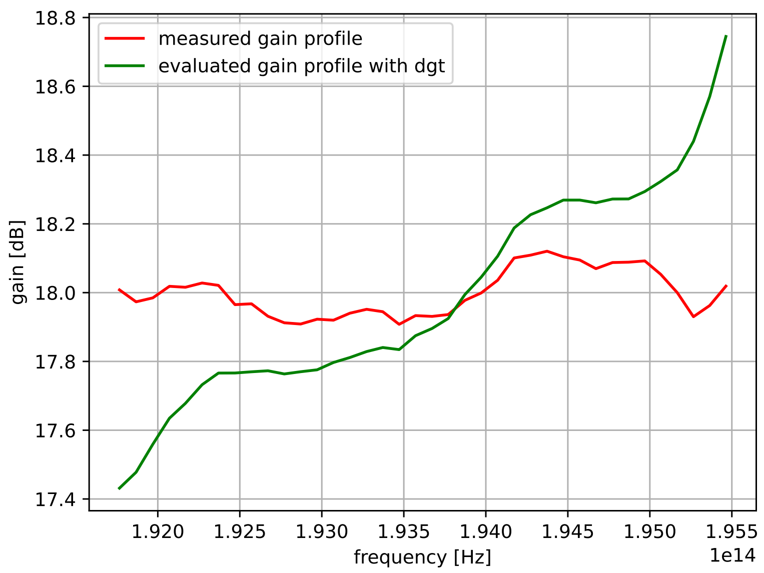


Figure 3.11: Comparison between the profile measured and the profile evaluated with DGT model in GNPy, gain: 18 dB, tilt: 0 dB

Inverting the equation gives $K(f)$ as a function of the 2 gain profile, the value of the tilt of the second profile and the parameters B and f_c:

$$K(f) = \frac{g(f; G, T_0) - g(f; G, T_i)}{T_i} + \frac{1}{B}(f - f_c) \tag{3.17}$$

This method avoids evaluating the ripple of the second profile and reduces the overall computational cost. The r_0 parameter is evaluated as in the previous method. The 2 gain profiles must be the same, otherwise the Equation 3.15 is no longer valid.

3.7.3 Generation of the generic gain profile

By inverting the Equation 3.11, it is possible to obtain a generic tilt profile that depends only on the tilt T_i:

$$r_{T_i}(f) = r_o(f) - K(f)T_i \tag{3.18}$$

From the Equation 3.2 it is possible to replace the specific ripple of the gain profile with the specific ripple of the amplifier, r_0 and the ripple depending on the slope $-K(f)T_i$. Considering the Equation 3.18, the generic gain profile is:

$$g(f; G, T_i) = G + r_0(f) - T_i K(f) + \frac{T_i}{B}(f - f_0) \tag{3.19}$$

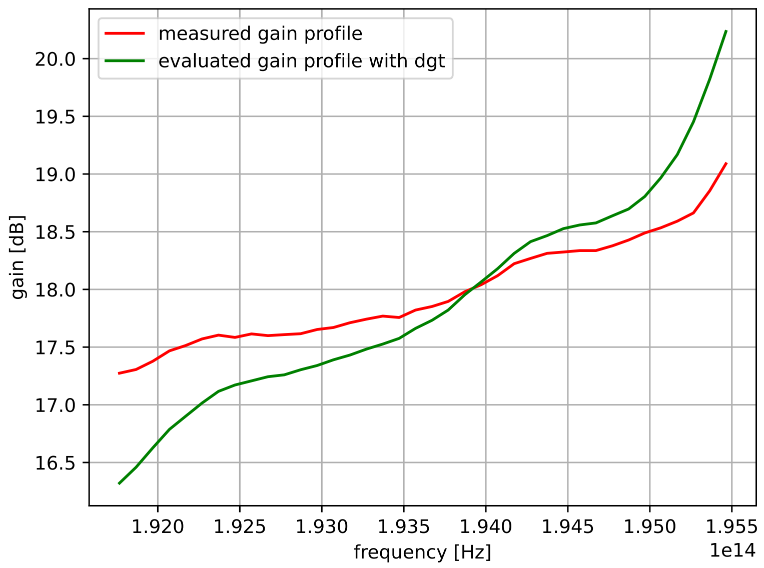


Figure 3.12: Comparison between the profile measured and the profile evaluated with DGT model in GNPy, gain: 18 dB, tilt: −2 dB

By splitting the specific ripple of the amplifier into 2 components, r_0 and $-K(f)T_i$, it is possible to generalize the Equation 3.18 for the real case where it is not possible to characterize each EDFA. It is possible to characterize an EDFA as described in Subsection 3.7.1 and use the same $K(f)$ parameter for all amplifiers of the same family. In this case, the generation of a gain profile from another EDFA gain profile characterization will be the Equation 3.18:

$$g(f; G, T_i) = G - T_i K(f) + \frac{T_i}{B}(f - f_0) \tag{3.20}$$

where is not present the parameter $r_0(f)$ because it is specific of the EDFA. This model does not take into account problems due to amplifier saturation or due to an impossible tilt value that the amplifier cannot reach. To obtain a consistent result the model input (gain G and tilt T) must be within the working range of the EDFA.

3.7.4 Resuls

The aim of this work has been to verify the accuracy of the new model used to characterize a series of EDFA amplifiers. A comparison has been made between the experimental data and an implementation of the model. The implemented

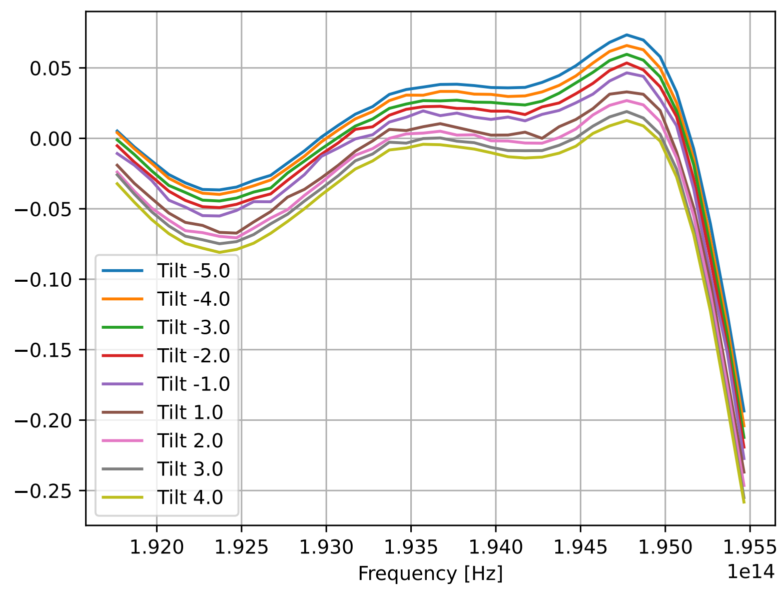


Figure 3.13: $K(f)$ parameter evaluated for different value of tilt and same gain

script consists of 3 main parts: there is the pre-processing of the data set, the evaluation of the parameters $K(f)$ and $r_0(f)$ and the plotting of the result. The pre-processing activity served to import and organize the data to make it easily accessible. The data is insert into a matrix where the dimensions represent the value of gain, tilt and input power. In this phase, the values outside the operating range of the amplifier are discharged.

In the second part of the script, the 2 characterization parameters ($K(f)$ and $r_0(f)$) has been evaluated for each amplifier. These parameters are stored in a list. A gain profile has been evaluated for each amplifier characterization parameters and for all possible input parameters, gain, tilt. The 2 version of the model have been analyzed, the case with and without $r_0(f)$. The Equation 3.19 and Equation 3.20 have been used to evaluate it. These gain profiles are stored in a matrix organized in the same way as the matrix used to store the dataset. The input power is not used in the model but has been included in the matrix to maintain the same organization of the dataset. For all possible combinations of gain profile of the dataset and gain profile evaluated with the model, the error has been evaluated, defined as the difference for each frequency between the evaluated gain profile and the measured gain profile.

In the last part, 2 different types of plots have been made, the comparison between the evaluated and the measured profile for a given gain and tilt and the boxplot of the error. In the first type of plot, for a given value of gain and tilt, all possible gain profiles are shown for different input powers. It is possible to confirm, as before, that the real measurement has a slightly different gain profile for different input powers. In Figure 3.14 and Figure 3.16 it is possible to observe the measured gain profile and the evaluated gain profile for 2 different amplifiers with tilt -5, whereas in Figure 3.15 and Figure 3.17 the profile for tilt -2 is shown.

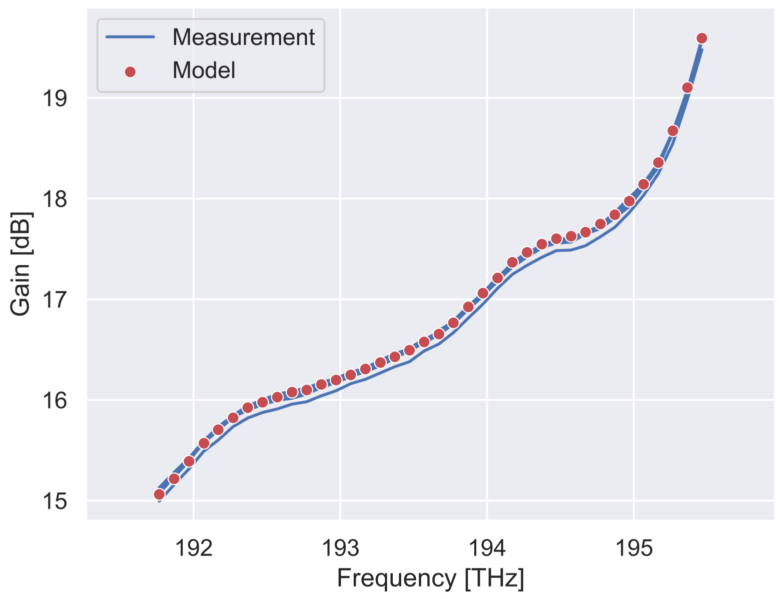


Figure 3.14: Gain profiles of EDFA 17 with gain 17 dB and tilt −5 dB

Each graph shows the gain profiles for all input powers. The measured profiles do not match perfectly due to imperfections in the real amplifier and measurement errors. There is only one gain profile of the model for all the input powers because the model is independent of the input powers. In Figure 3.18 and Figure 3.19 it is possible to observe the errors between the real gain profile and the evaluated gain profile for all the possible combinations of input power, gain and tilt. In both cases the error is centered around 0 dB.

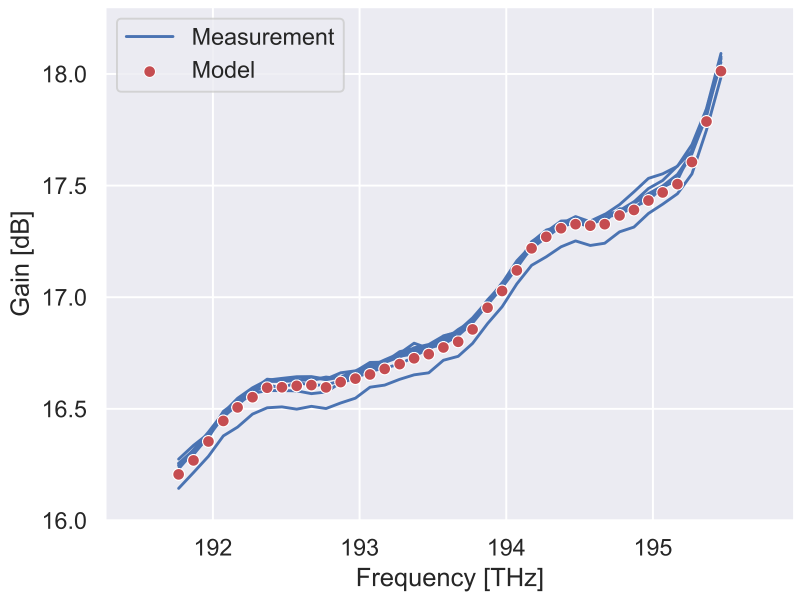


Figure 3.15: Gain profile of EDFA 17 with gain 17 dB and tilt −2 dB

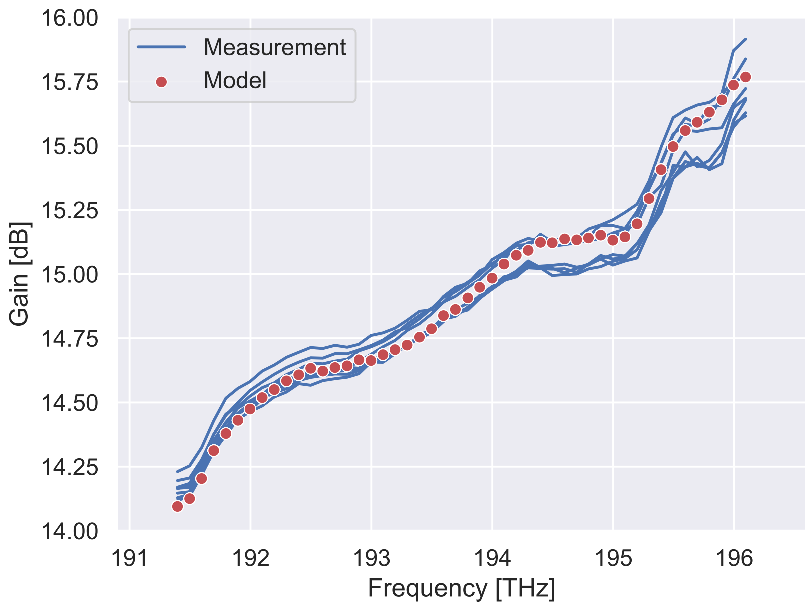


Figure 3.16: Gain profile of EDFA 35 with gain 17 dB and tilt −2 dB

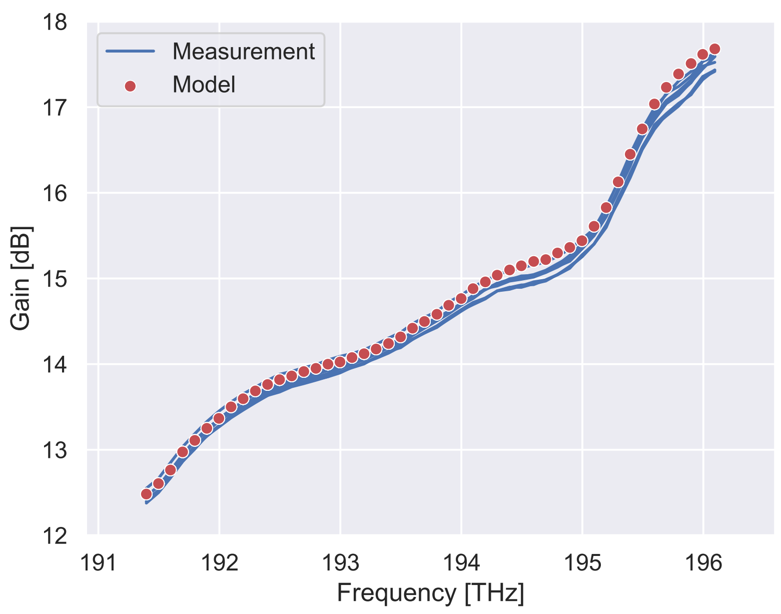


Figure 3.17: Gain profile of EDFA 35 with gain 17 dB and tilt −5 dB

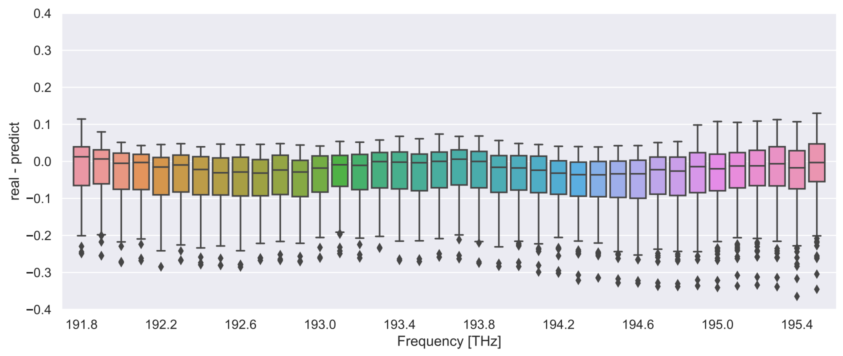


Figure 3.18: Boxplot of the error for the EDFA 17

3.8 Difference between DGT and new implementation

In order to maintain backward compatibility, the following equations are proposed for the transition from the DGT model to the new model:

$$dgt(f) = \frac{gp_0(f) - gp_i(f)}{gp_0(\bar{f} - gp_i(\bar{f})} =$$

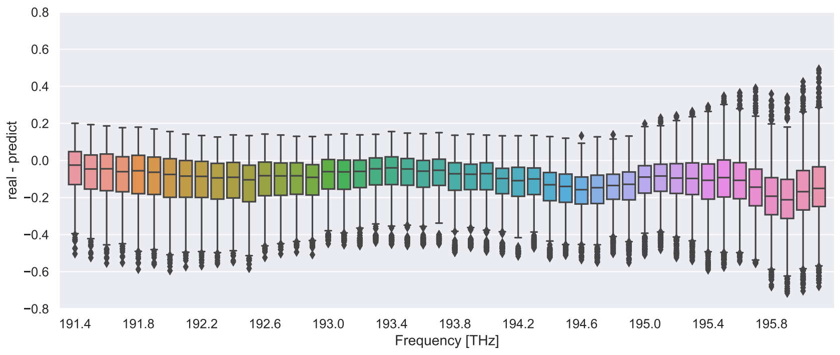


Figure 3.19: Boxplot of the error for the EDFA 35

$$= \frac{G + T_0\frac{f-f_c}{B} + r_0(f) - G - T_i\frac{f-f_c}{B} - r_i(f)}{G + T_0\frac{\bar{f}-f_c}{B} + r_0(\bar{f}) - G - T_i\frac{\bar{f}-f_c}{B} - r_i(\bar{f})}$$

$T_0 = 0$ and so:

$$dgt(f) = \frac{r_0(f) - T_i\frac{f-f_c}{B} - r_i(f)}{r_0(\bar{f}) - T_i\frac{\bar{f}-f_c}{B} - r_i(\bar{f})} = \frac{\frac{r_0(f)-r_i(f)}{T_i} - \frac{f-f_c}{B}}{\frac{r_0(\bar{f})-r_i(\bar{f})}{T_i} - \frac{\bar{f}-f_0}{B}}$$

$$= \frac{k(f) - \frac{f-f_c}{B}}{k(\bar{f}) - \frac{\bar{f}-f_c}{B}}$$

and so:

$$(k(\bar{f}) - \frac{\bar{f}-f_0}{B})dgt(f) = k(f) - \frac{f-f_c}{B}$$

$$k(f) = (k(\bar{f}) - \frac{\bar{f}-f_c}{B})dgt(f) + \frac{f-f_0}{B}$$

For evaluate $k(f)$ you have to know $dgt(f)$, f_0, $r_0(f)$, B and $k(\bar{f})$. The $dgt(f)$ must be evaluate in $\bar{f}$.

$$k(f) = (k(\bar{f}) - \frac{\bar{f}-f_c}{B})\frac{k(f) - \frac{f-f_0}{B}}{k(\bar{f}) - \frac{\bar{f}-f_c}{B}} + \frac{f-f_0}{B}$$

3.9 GNPy implementation

The result obtained showed a better approximation of this new model to respect the DGT model implemented in GNPy. The new function `gain_profile()` can be used in GNPy to evaluate the gain profile, replacing the method `_gain_profile()` of the EDFA element. The function has to implement a saturation control system to shift the gain profile to an acceptable power level if the gain insert is not compatible with the operating range of the EDFA model. The code update requires a strict protocol to be followed. The modification of the GNPy code has to be done gradually, so the first update developed is the insertion of the parameter and the insertion of the default value for the EDFA. Then the new gain profile implementation and saturation control can be implemented. All patches must pass the tests to verify that the changes produce a reasonable result. The Pytest framework is used to perform this test. The test scripts are already in the repository, the code changes made have modified the EDFA element, the specific test for the EDFA has been tested.

Chapter 4

Tuning of C+L band losses

4.1 Introduction

The objective of this task is to perform a tuning to obtain an accurate simulation of the behavior of a line using C band and L band. Some measured parameters have been tuned for this purpose. A configuration with C and L EDFA amplifiers has been set up. The Figure 4.1 shows the configuration. The setup consists of 12 EDFA (6 for L-band and 6 for C-band), followed by a single span of fiber for the pair of amplifiers. The Table 4.1 describes the length and the loss of the fiber measured at 2 different frequencies (representing the center frequency for C and L band). The value of the measured loss for both fiber include the loss due to the mux (the devices that permit to join the signal coming from the C and L EDFA), the input and output connector loss and the loss of the connector placed in the center of the fiber (each span is composed by 2 different fiber that join in approximately half the length). The value of the losses is independent of the frequency, but the C and L band losses are different because of the connector loss. The value of the the fiber loss is the same in both cases. The connector is the device that physically connects the amplifier to the fiber, changing the loss characteristic by managing it. Disconnection and reconnection can lead to a large variation in the loss parameters. The measurement of the characteristic of the fiber is performed by disconnecting and reconnecting each fiber, for this reason the loss of the connector can have variation. Before and after each amplifier is present an OSA.

The losses are separated for C and L band to account for the different value due to the different connector. For each amplifier, the input and output signal power and the input and output noise power are given. The characterization of each span of the fiber has been performed and the fiber CD and losses have been provided. The BER has been measured and from this value the GSNR has been evaluate, which has been used to check the effective accuracy of the simulation.

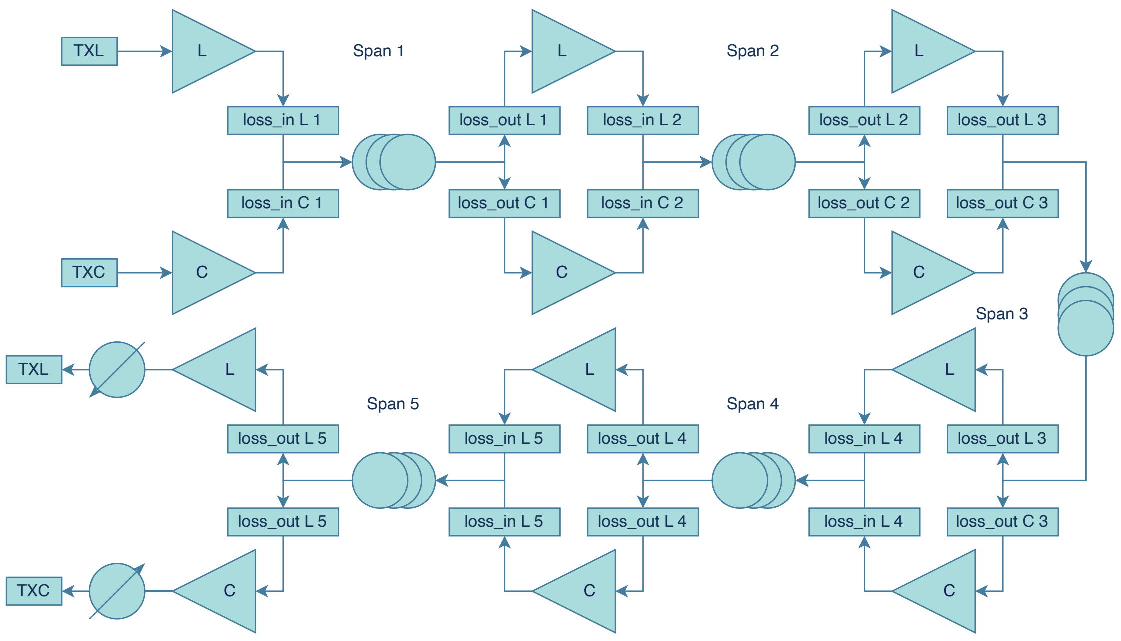


Figure 4.1: Experimental scenario topology for C+L band configuration

	Span-1	Span-2	Span-3	Span-4	Span-5
Length [km]	85.48	85.56	87.85	85.63	87.83
Total loss @193.4 THz [dB]	16.40	15.54	15.98	15.76	15.95
Total loss @188.65 THz [dB]	15.580	15.020	15.540	15.300	15.530

Table 4.1: Fiber characteristics for all the span of the C+L band experiment

4.2 Theory recap

4.2.1 General evaluation of Bit Error Probability (BER)

In electrical communication is possible to define a relation between the SNR and the bit error probability (BER) of the modulation format that has been used. In general the value of BER is very small for this reason is used the logarithmic conversion:

$$BER_{log} = log(BER_{lin}) \tag{4.1}$$

The probability of error in an M-QAM modulation is

$$P(e) = 2\frac{\sqrt{M}-1}{\sqrt{M}}erfc\left(\sqrt{\frac{3log_2\sqrt{M}}{M-1}\frac{E_b}{N_0}}\right) - \left(\frac{\sqrt{M}-1}{\sqrt{M}}\right)^2 erfc^2\left(\sqrt{\frac{3log_2\sqrt{M}}{M-1}\frac{E_b}{N_0}}\right) \tag{4.2}$$

where M is the the number of symbols in the dictionary, E_b is the energy per bit, and N_0 is the noise power spectral density. In the case of a 16-QAM modulation format with Gray code, an adapt filter and considering only the first factor of the Equation 4.2, the equation relating BER to E_b/N_0 is:

$$BER = \frac{P(e)}{4} = \frac{3}{8}erfc\left(\sqrt{\frac{2}{5}\frac{E_b}{N_0}}\right) \tag{4.3}$$

E_b/N_0 can be written as follows:

$$SNR = \frac{E_b}{N_0}R_c\, log_2(M) \tag{4.4}$$

$R_c = k/n$ where: k is the number of useful bits of information and n is the number of total bits. Consider the Equation 4.3 and Equation 4.4 considering $M = 16$ then $log_2(M) = 4$:

$$BER = \frac{P(e)}{4} = \frac{3}{8}erfc\left(\sqrt{\frac{1}{10}\frac{SNR}{R_c}}\right) \tag{4.5}$$

4.2.2 Back2back characterization

For the evaluation of back to back has been used a file that contains the information related to the BER and the GSNR measured in the back2back configuration. The setup configuration is shown in Figure 4.2. For a fixed frequency are present different value of BER and for each value of BER is present the GSNR measured. The goal of this task has been to validate the model and the code for evaluating the GSNR from the BER, and to analyze the behavior of the SNR_{TRX}. From the equation Equation 4.5 and without considering Forward Error Correction (FEC ($k = n$ then $R_c = 1$):

$$BER = \frac{3}{8}erfc\left(\sqrt{\frac{1}{10}SNR}\right) \tag{4.6}$$

The ideal equation for finding the SNR knowing the BER and the modulation format has been improved adding the SNR_{TRX}. The SNR_{TRX} is the electrical noise that the transceiver insert in the transmission. From the datasheet of the

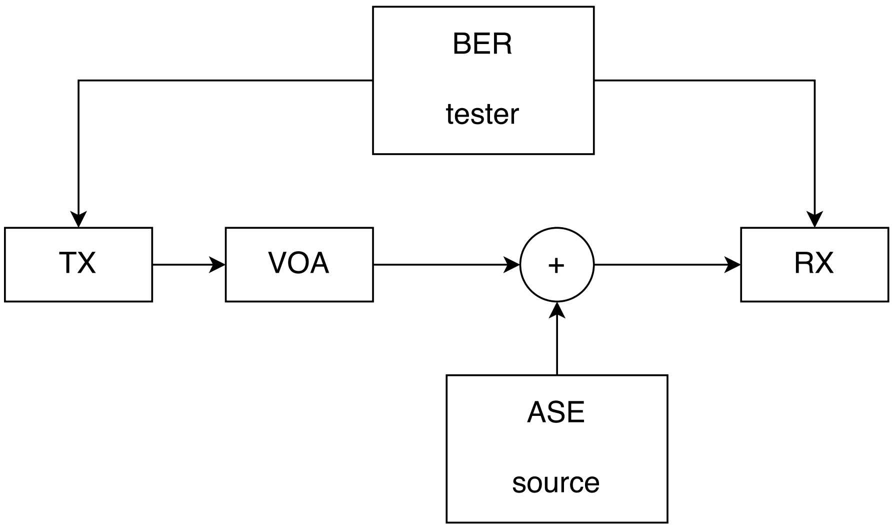


Figure 4.2: Setup configuration for measuring of the BER and GSNR

component it is a value around 20 dB. The SNR is defined as the parallel between the $GSNR$ and the SNR_{TRX}:

$$SNR = \left(\frac{1}{GSNR} + \frac{1}{SNR_{TRX}}\right)^{-1} \tag{4.7}$$

From the Equation 4.6 and Equation 4.7 it is possible to find the $GSNR$:

$$GSNR = \left(\frac{1}{10 ercinv^2\left(\frac{8}{3}BER\right)} - \frac{1}{SNR_{TRX}}\right)^{-1} \tag{4.8}$$

The validation of the model has been performed by comparing the measured value of GSNR with the ideal curve generated with Equation 4.6. An optimization process has been used to find the SNR_{TRX} value that best fit the ideal curve. The RMSE has been evaluated for each frequency between the measured and the ideal BERvsGSNR curve. The value of the SNR_{TRX} has been changed in a range from 18 dB to 22 dB with steps of 0.05 dB. If the RMSE error is less than the threshold value the optimization is completed.

The result are shown in Figure 4.3. There is a dependence on the frequency of the measurement, but it can be associated with an uncertainty in the measurement. So the mean value of the SNR_{TRX} has been used: 20 dB.

In the Figure 4.4 and Figure 4.5 the BER vs. GSNR are evaluated and measured with the mean value of SNR_{TRX}. The 2 curves overlap and this means that the model is correct.

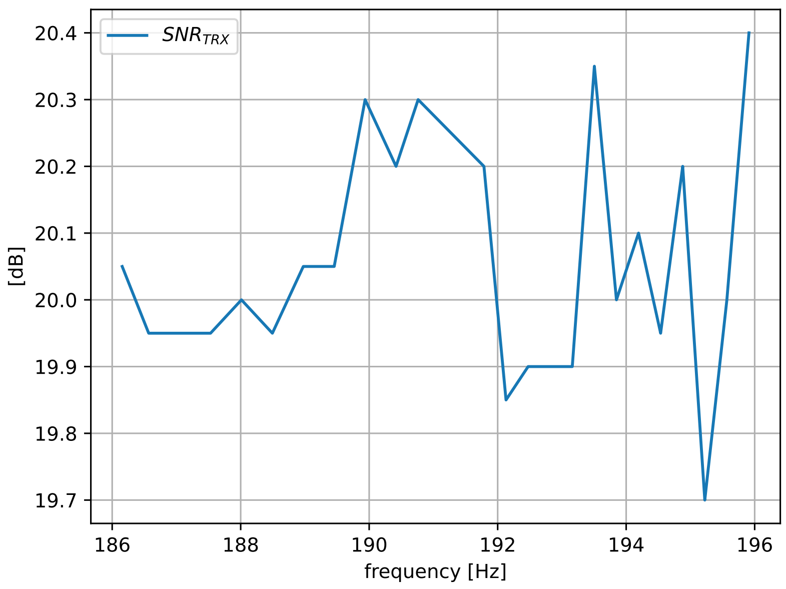


Figure 4.3: Frequency dependence of the SNR_{TRX}

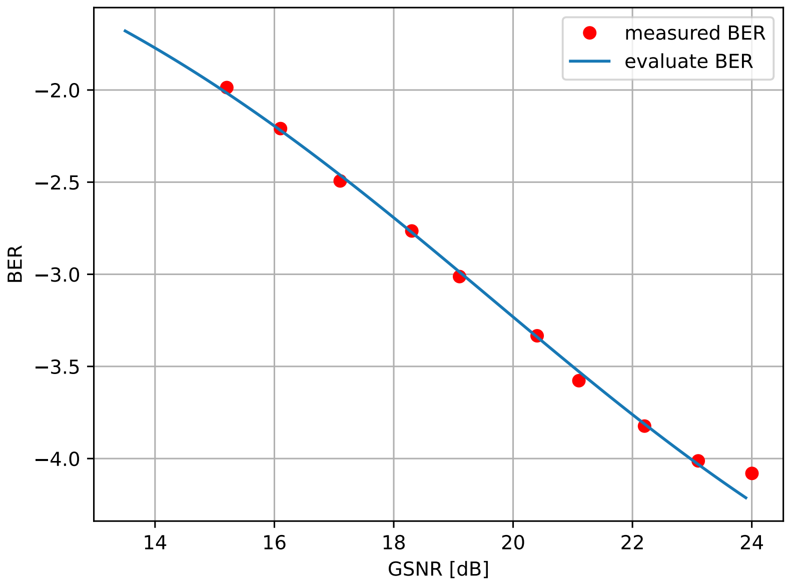


Figure 4.4: BER vs GSNR curve evaluated at 187.53 THz

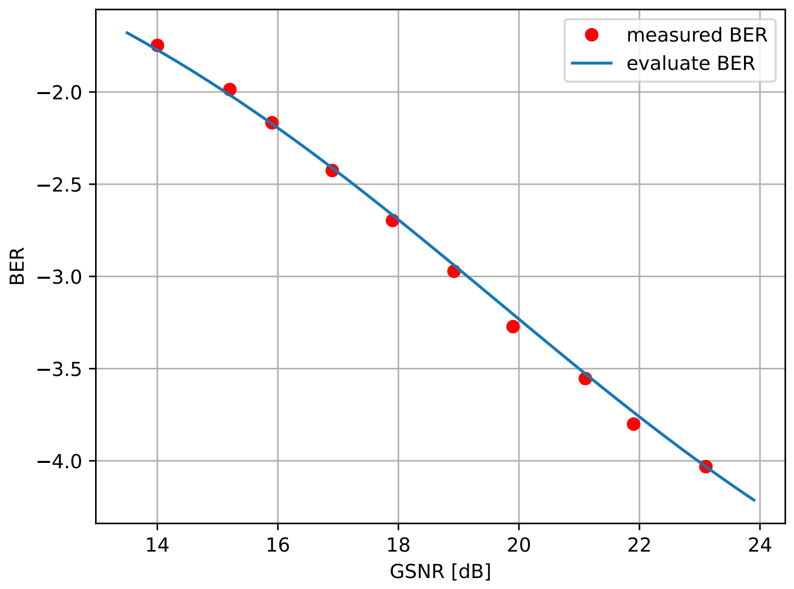


Figure 4.5: BER vs GSNR curve evaluated at 194.19 THz

4.3 Experiment setup

The topology file has been created following the schema that has been proposed in Figure 4.1. The experiment is composed by 4 transceiver, 6 EDFA amplifier and 5 Fiber span. In the implementation of the topology for GNPy before and after the EDFA has been added one fused elements. This elements allows to separate the input and output losses of the fiber for C and L band. The data from the dataset for the EDFA are: the input and the output power of the signal, the input and the output power of the noise. From this value is possible to evaluate the gain profile and the noise figure ripple for each amplifier.

The gain profile is evaluated subtracting the signal output of the amplifier to the signal input of the amplifier. The noise figure ripple is evaluated thanks to Equation 4.9.

$$NF(f) = \frac{(P_{ase}^{out}(f)/G) - P_{ase}^{in}(f)}{h\,f\,R_b} \tag{4.9}$$

where: P_{ase}^{out} is the noise at the output of the amplifier, P_{ase}^{in} is the noise at the input of the amplifier and R_b is the baud rate. The dataset information of the fiber is related to the loss coefficient and the chromatic dispersion. Both values are characterized in the frequency domain. This information is loaded into the topology file, the generic topology is read by the script and the 2 parameters are inserted according to the file format.

The dataset is composed of 9 measurement campaigns carried out on the same experimental setup and, in particular, without physically plugging and unplugging any fiber cable. This has been done to avoid possible differences in connector losses in the different campaigns. In each campaign there is a different shape of launch power and a different level of launch power.

4.4 Tuning of the parameters

The parameters that have been tuned to find consistent data are the fused connector losses. In order to obtain a more accurate result, a tuning of some specific parameters of the fiber has been carried out, but it leads to solutions that are physically infeasible. The tuning process consists in comparing the measured signal profile with the value propagated on the GNPy simulator. The signal measured at the launch has been propagate thought the characterized fiber and the EDFA. The EDFA simulation is performed by multiplying the input signal with the measured gain profile. In this way only the phenomena due to the fiber propagation and the attenuation due to the loss are considered. This process has been done for each span, 2 different case has been developed. In the first case, the signal is fully propagated from the beginning to the end of the line; in the second case, the signal is regenerated after each amplifier. These 2 different cases have been developed to avoid the accumulation error that the losses, the EDFA model and the fiber model can introduce.

4.5 Fused element tuning

Two scripts have been developed, one for tuning the signal for each span and a second for propagating the signal along the whole network to simulate the whole process. The tuning code is an already implemented Python class that uses Evolution Strategies (ES). ES is a family of optimization algorithms inspired by the process of natural evolution. It does not require the computation of gradients, making it suitable for optimizing complex, high-dimensional, and non-convex functions.

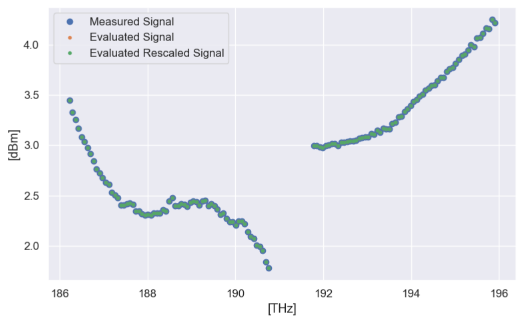
Each fused element has a loss value for the C band and for the L band. There are 2 fused elements for each EDFA, one before and one after. The tuning is different for these 2 types of fused elements. The evaluation of the losses for the FUSED before the EDFA is done by subtracting the expected value of the power at the output of the element (the measured power at the input of the EDFA) from the input power of the EDFA (all values are in dB). For the fused element after the EDFA the optimization has been performed using the ES optimization. The signal has been propagate through the fused element and the fiber, the result has been compared with the measured value. The estimator that has been used to compare the 2 profiles is the RMSE. A variation of the loss in one band changes the value of the power in the line, and for this reason there is a variation of the Raman effect, this can lead to a variation of the optimal loss in the other band. The L and C losses are changed separately and propagation is performed after each change.

All of the graphs described here show the L band on the left and the C band on the right. The graphs in Figure 4.6 show the propagation of the signal along the line. There are 6 graphs, one for each amplifier, showing the output signal profile. It can be seen that the blue dot and the green dot, representing the measured signal power and the evaluated signal power from the rescaled spectrum, respectively, are in perfect agreement. This is because the signal is regenerated at each output of the EDFA. The error between the measured profile and the profile evaluated from the propagation of the spectrum increases with the increase of the propagation element crossed. This lead in the last span to a maximum error of 0.5 dB. The amplification of the amplifier is considered correct due to the observation that has been done before, the error is do to the optimization process for the fused losses.

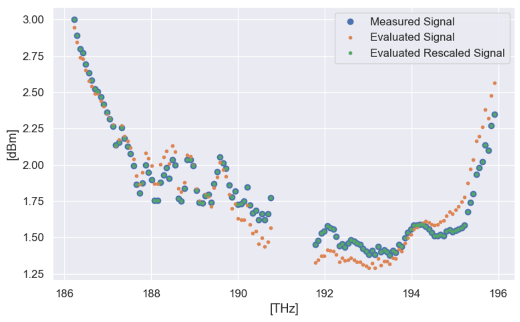
In the Figure 4.7 are shown the sequential plot of the ASE noise. Unlike the signal, there is no match between the measured curve and the value of ASE taken from the rescaled spectrum. This is because the rescaling of the spectrum is done equal to the signal, the ASE is changed accordingly with the signal and not with the measured value. The ASE evaluated from the rescaled spectrum is much closer to the ASE evaluated from the fully propagated spectrum. This can be explained by the fact that the losses affect the signal and the noise in the same way. The noise figure insert in the model of the EDFA is evaluated directly from the measurement, the error of the evaluated ASE is little affected by the noise figure inaccuracy.

The Figure 4.8 shows the last span of all different cases with different launch power. Optimization is done separately for each strategy, resulting in different losses. The maximum error between the measurement and the propagated signal is in all cases less than 0.5 dB.

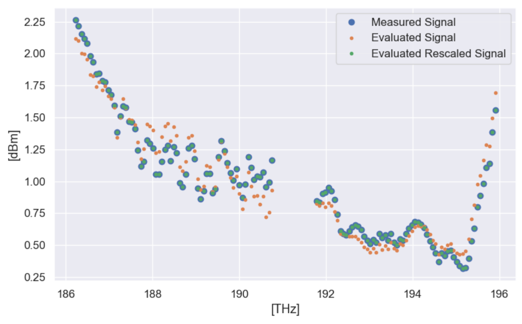
This last set of plots, Figure 4.9, shows the ASE after the last EDFA in the chain. The accuracy of the propagated ASE is high due to the fact that the noise figure of the amplifier is evaluated from the measurement and no model of this parameter has been applied.



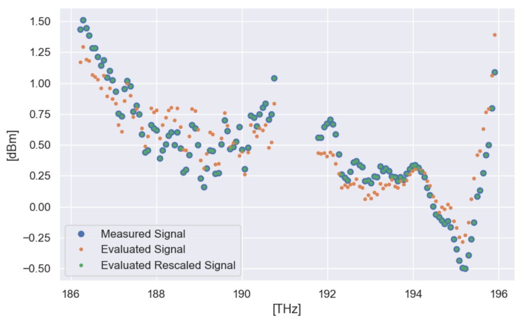
(a) Span 0



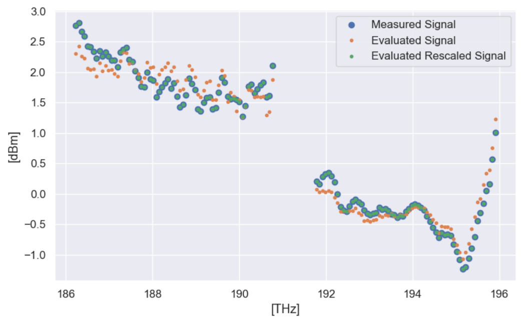
(b) Span 1



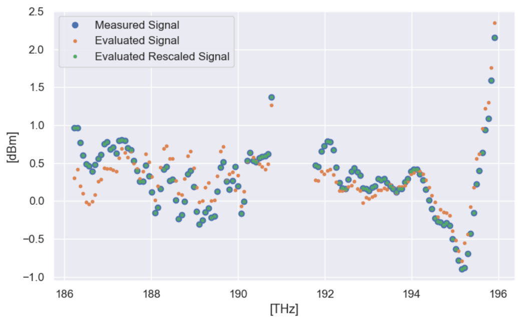
(c) Span 2



(d) Span 3

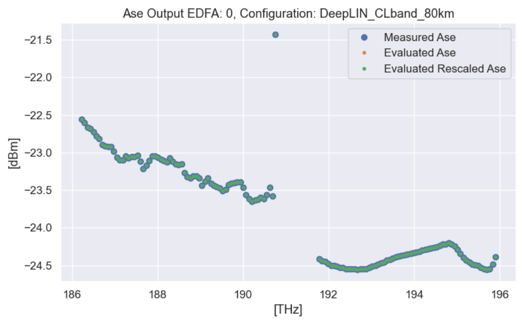


(e) Span 4

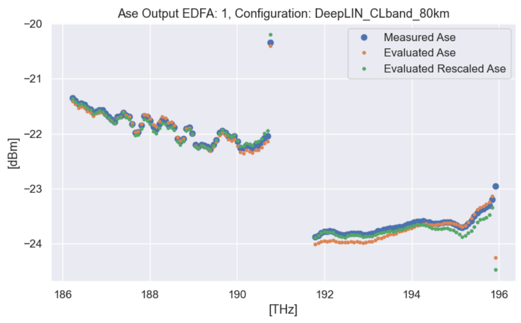


(f) Span 5

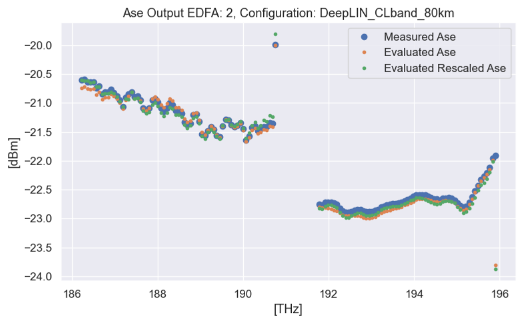
Figure 4.6: Deeplin strategy, signal profile at the output of each EDFA measured by oSA



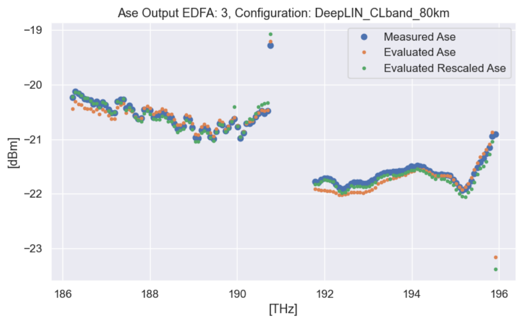
(a) Span 0



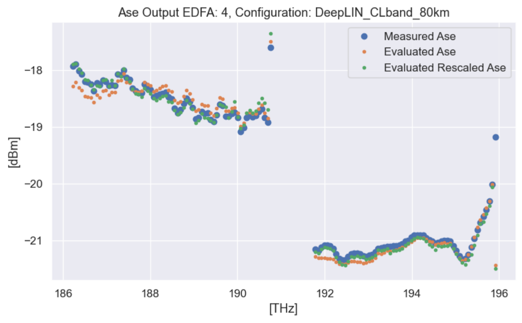
(b) Span 1



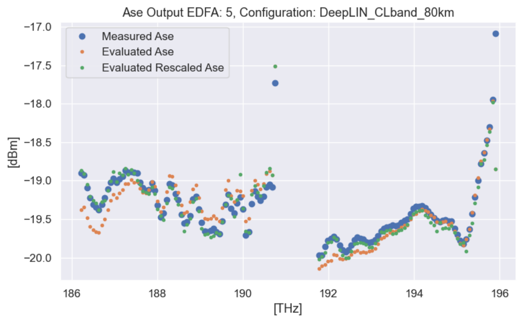
(c) Span 2



(d) Span 3

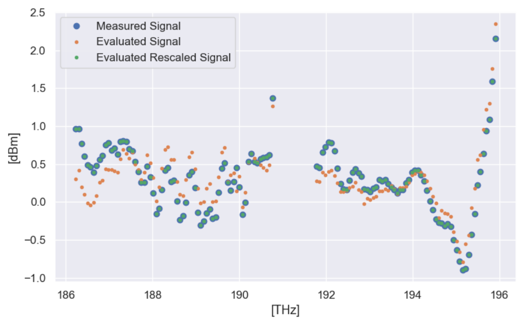


(e) Span 4

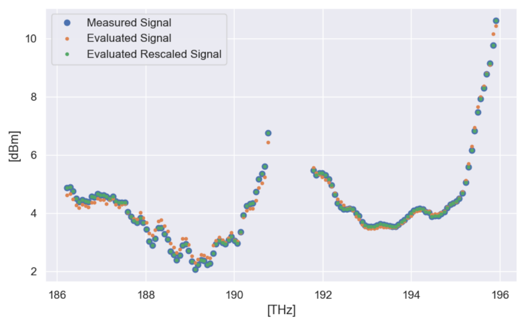


(f) Span 5

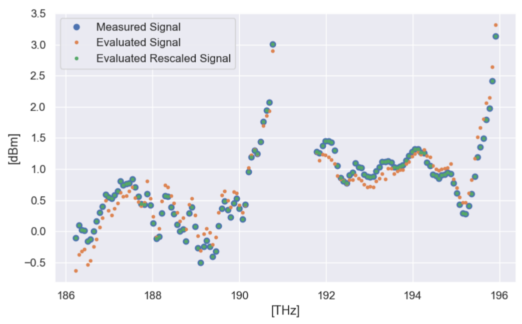
Figure 4.7: Deeplin strategy, ASE profile at the output of each EDFA measured by OSA



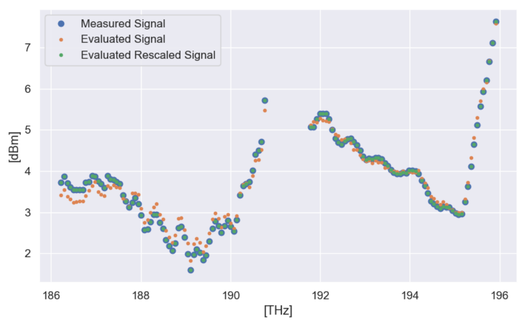
(a) DeepLIN span 5, signal



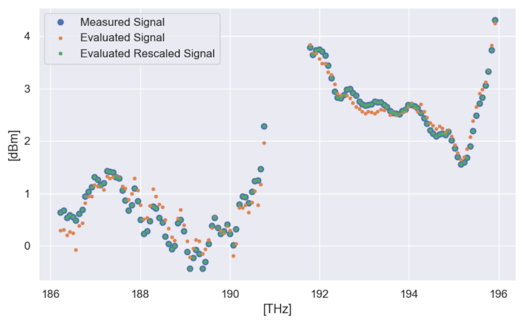
(b) DeepNLI span 5, signal



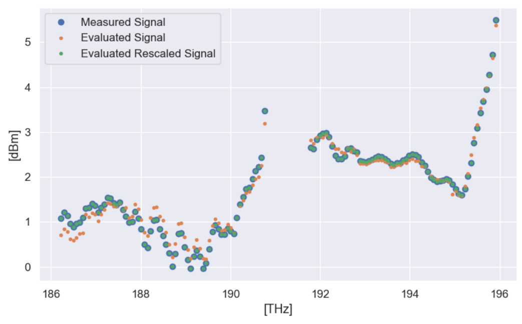
(c) FlatGSNR span 5, signal



(d) FlatProfile span 5, signal

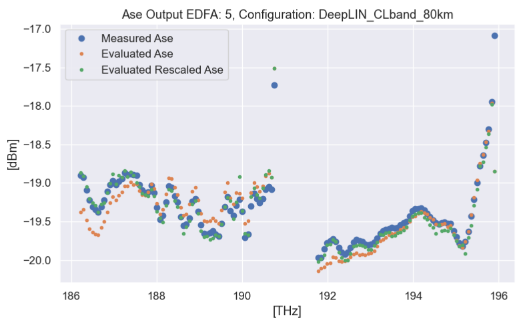


(e) MaxGMI span 5, signal

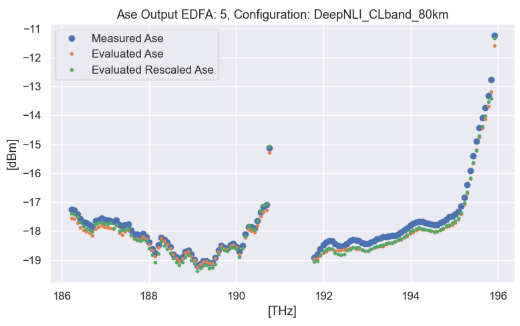


(f) OptNLI span 5, signal

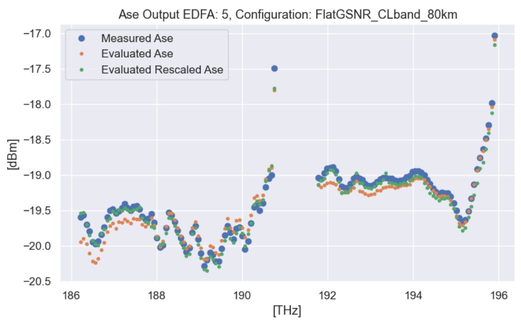
Figure 4.8: Signal last span of all the power level



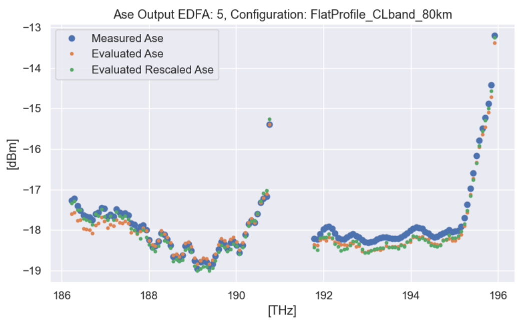
(a) DeepLIN span 5, ASE



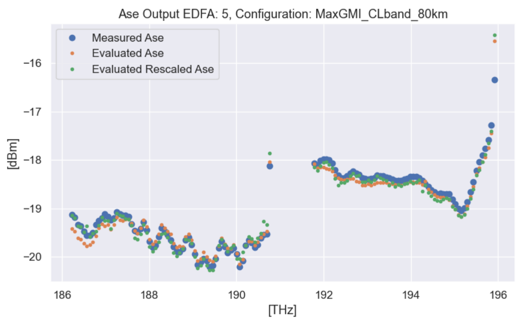
(b) DeepNLI span 5, ASE



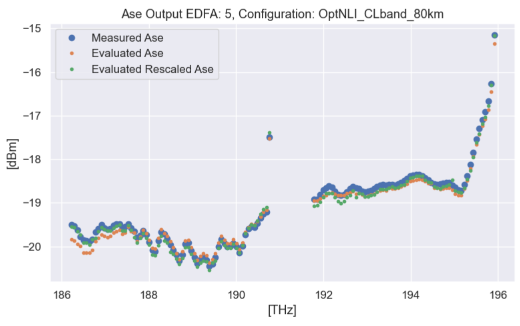
(c) FlatGSNR span 5, ASE



(d) FlatProfile span 5, ASE



(e) MaxGMI span 5, ASE



(f) OptNLI span 5, ASE

Figure 4.9: ASE last span of all the power level

4.6 General fiber tuning

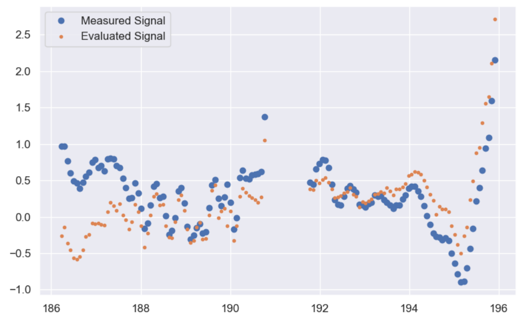
The loss function of the fiber and the loss of the C and L band connector are parameters that are independent of the power that passes through them. The tuning of the fused element and thus the tuning of the connector loss between different measurements performed at different power levels has resulted in a difference loss value. An analysis of the difference between the same connector loss for all the different levels of measured power has been performed. The results in Table 4.2 show that the dispersion of the tuning loss parameter is acceptable in the majority of the fused element. The propagation of the spectrum has been performed using

	C band		L band	
	std [dB]	mean [dB]	std [dB]	mean [dB]
Fused_1_in	0.17	0.39	0.24	1.07
Fused_1_out	0.14	2.13	0.24	1.43
Fused_2_in	0.26	1.55	0.3	0.75
Fused_2_out	0.24	2.15	0.28	2.09
Fused_3_in	0.26	0.47	0.21	0.26
Fused_3_out	0.19	1.99	0.21	1.79
Fused_4_in	0.41	0.41	0.27	0.1
Fused_4_out	0.3	2.14	0.36	1.74
Fused_5_in	0.42	0.6	0.1	0.04
Fused_5_out	0.32	3.12	0.16	2.45

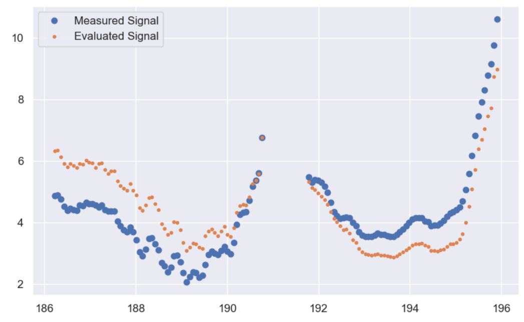
Table 4.2: Analysis of the loss dispersion of the tuned fused element

the same loss value for all power levels, the mean value of the Table 4.2 has been used. The signal and the ASE in the last span are shown in Figure 4.10 and Figure 4.11. The maximum error between the signal power measurement and the propagated signal is 1 dB. There is an increase of the error to respect the case in which the losses are tuned for each specific power level. The error for the ASE evaluation from the not re-scaled spectrum has the same behavior as the error of the signal. This is because the loss affects both the ASE and the signal power in the same way. The ASE evaluated from the re-scalded spectrum is more accurate because the error is only due to the last span.

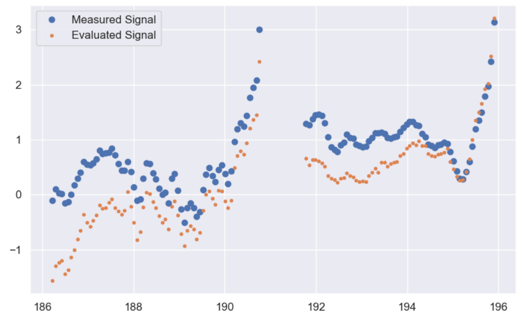
The error for the signal and the ASE in the last span is the accumulative error for each span, for this reason the value obtained in this processing is acceptable.



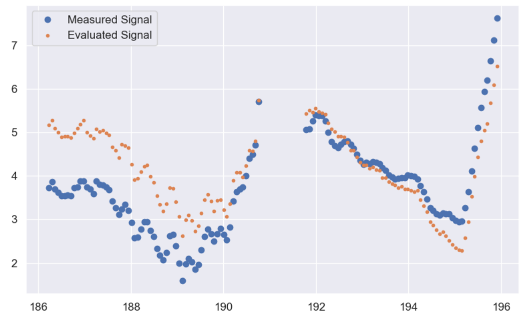
(a) DeepLIN span 5, signal



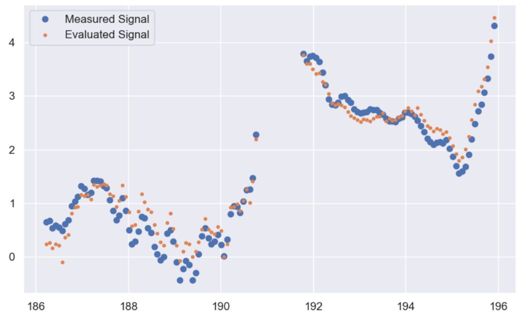
(b) DeepNLI span 5, signal



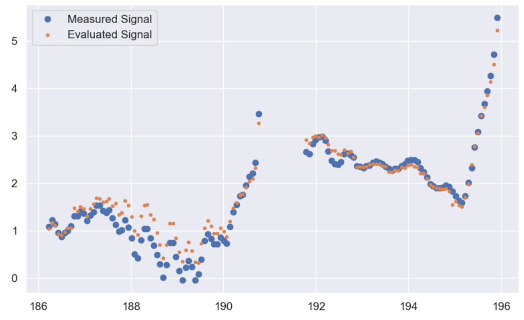
(c) FlatGSNR span 5, signal



(d) FlatProfile span 5, signal

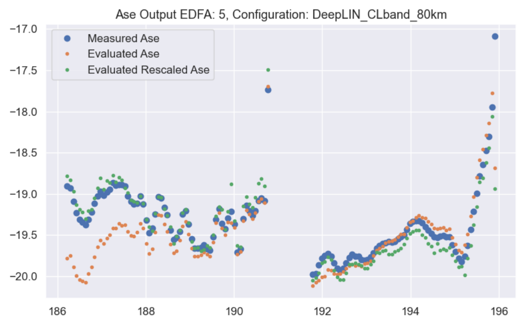


(e) MaxGMI span 5, signal

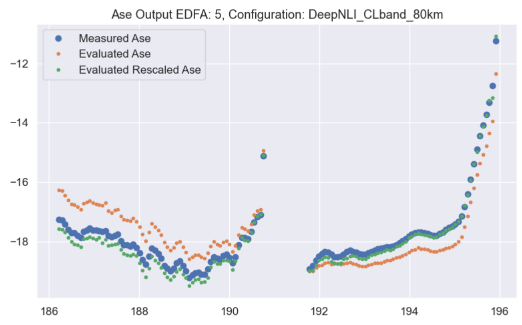


(f) OptNLI span 5, signal

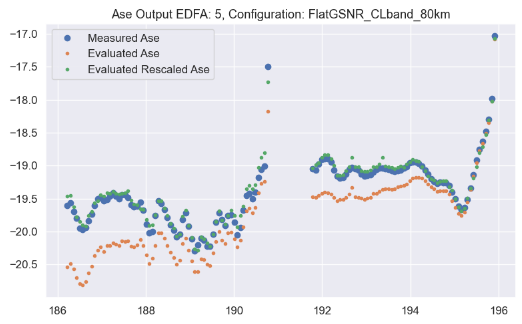
Figure 4.10: Signal last span of all the power level, same losses for all the cases



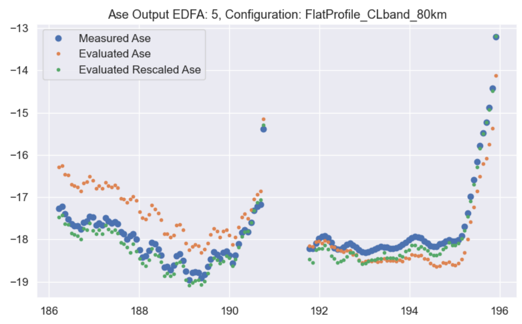
(a) DeepLIN span 5, ASE



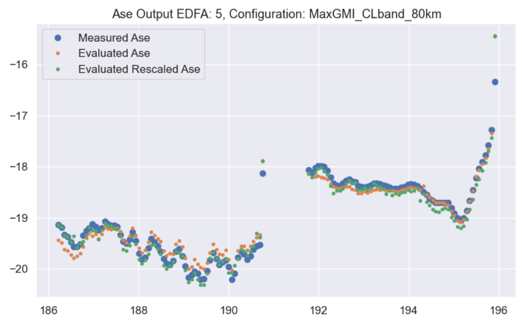
(b) DeepNLI span 5, ASE



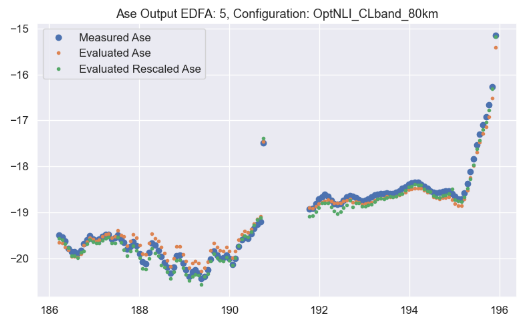
(c) FlatGSNR span 5, ASE



(d) FlatProfile span 5, ASE



(e) MaxGMI span 5, ASE



(f) OptNLI span 5, ASE

Figure 4.11: ASE last span of all the power levele, same losses for all cases

Chapter 5

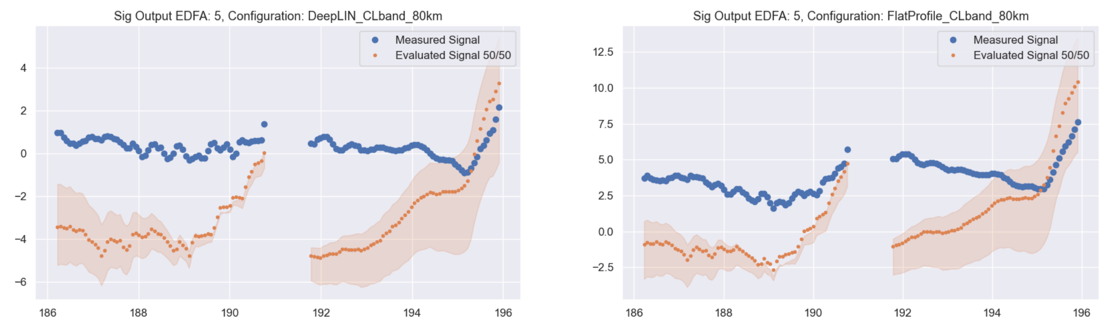
Multiband Amplifier GNPy implementation

5.1 Introduction

There has been an analysis of the spectrum division implemented in a patch of GNPy. The analysis has been done for the C and L bands based on the dataset used in the previous section. The experimental setup is the same as the previous setup. The analysis consists of 2 incremental steps to determine which element contributes most to the accuracy of the model. The focus of the research is to understand if the model that characterizes the specific network element in GNPy, work correctly in a C+L band scenario. The analysis has been carried out considering the result on the tuning of the losses of the fused element that has been done in the previous section and the model that has been formalized in the Chapter 3. The signal component is the main concern of this research, the optimization of the losses has been done by comparing the propagated signal with the measured signal. The measured ASE have been used for the simulation, this because it is not found a valid model to characterize the noise component of the EDFA. [21][22]

5.2 Using GNPy with EDFA's new model and flat noise figure

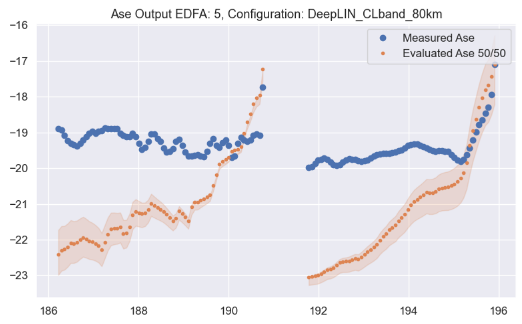
In the first scenario the connector losses has been evaluated from the difference between the measured signal power between 2 consecutive amplifier and the evaluated loss of the fiber. The loss of the fiber has been evaluated multiplying the length of the fiber for the alpha coefficient. The alpha coefficient is only a value and it is not dependent on the frequency. The default values of GNPy are used for this evaluation. No other information is added to the element that simulates the fiber. The connector loss has been divided by 2 and the value is inserted at the input of the fiber (input connector loss) and at the output of the fiber (output connector loss). The limit cases where all the losses are at the input connectors or at the output connectors have been evaluated. As this value changes, the NLI is the component that changes the most because it is directly proportional to the power passing through the fiber. New model of the amplifier is implemented, to fully experiment a real scenario, the characterization of the amplifier has been done with a different dataset (considering the same model). From the measurement performed, it has been possible to evaluate the noise figure as a function of frequency. However, to simulate a real-world scenario, the average of the evaluated noise figure has been inserted into each EDFA. In the simulation, the spectrum is split before each pair of amplifiers, there are 2 amplifiers, one for the C band and one for the L band. After the amplifier, the inverse operation is performed, merging the C and L spectrum to enter the fiber. Before the splitting operation and after the merging operation of the spectrum, the spectrum passes through the fused element where the losses are applied separately to the C and L spectrum. The Figure 5.1 shows the comparison between the evaluated and the measured signal power for 2 different level powers in the last span. The same comparison has been done for the ASE



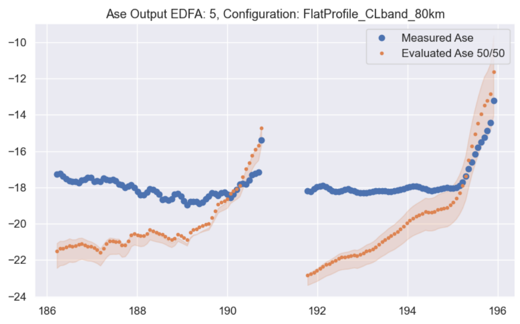
(a) DeepLIN span 5, signal   **(b)** Flatprofile span 5, signal

Figure 5.1: Signal last span of deeplin and flatprofile power level, connector loss estimated

noise, in Figure 5.2. It is possible to observe the same type of error in this plot. The reasons of this error are due to the connector losses, they act in the same wrong way on the signal and on the ASE. And the propagation of the signal error accumulates, the correction on the model is not enough to achieve a reasonable error. The resulting GSNR is not accurate because the signal error is not accurate, and so the evolution of the NLI leads to large errors. The same is true for OSNR evaluation, the losses are not accurate enough and affect the evaluation in a wrong



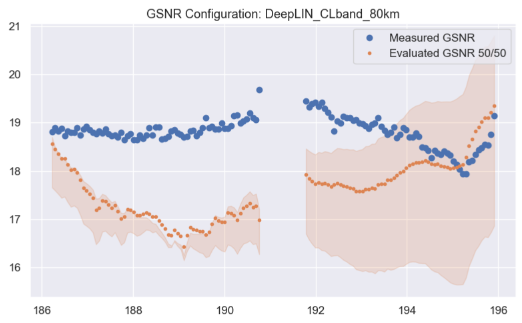
(a) DeepLIN span 5, ASE



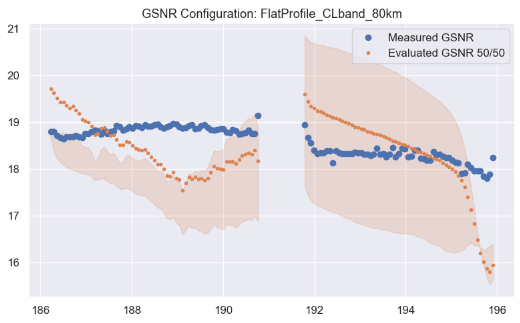
(b) Flatprofile span 5, ASE

Figure 5.2: ASE last span of deeplin and flatprofile power level, noise figure flat

way. In Figure 5.3 this comparison is plotted. The orange band represents the



(a) DeepLIN GSNR



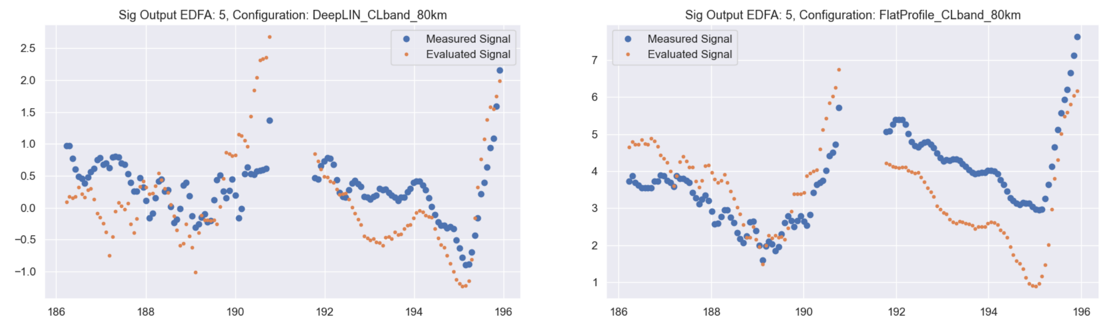
(b) Flatprofile GSNR

Figure 5.3: GSNR of deeplin and flatprofile power level, connector loss estimated and noise figure flat

variation in the result when the estimated connector loss is placed at the input or output of the fiber. The output signal power will change if the connector loss is placed at the beginning or end of the fiber because the fiber behavior depends on the input power.

5.3 Using GNPy with the new EDFA model and tuned losses

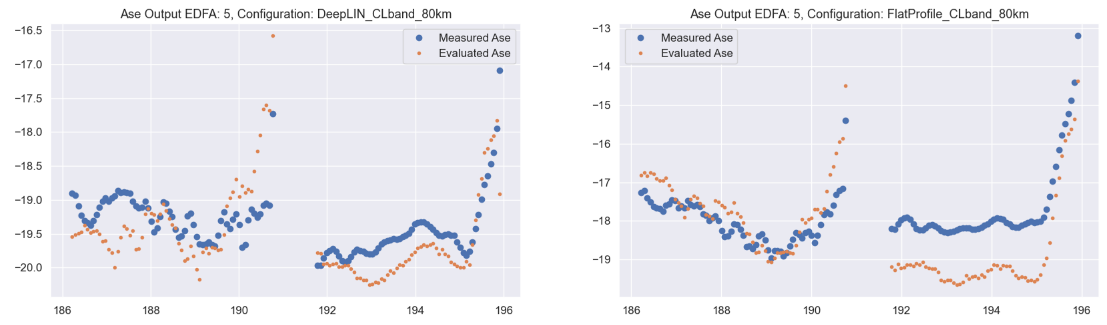
In this case, all the information related to the fiber is inserted, the value of the connector loss are tuned to obtain at the output of the fiber the measured value of the power (The results of the previous section has been used). The noise figure insert in the EDFA element is the measured one. In the Figure 5.4 the signal power comparison has been performed. In the Figure 5.5 the curves of the measured ASE



(a) DeepLIN span 5, signal **(b)** Flatprofile span 5, signal

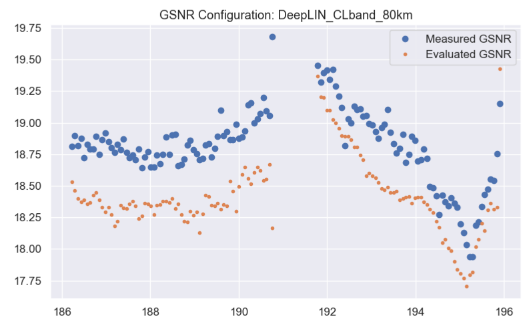
Figure 5.4: Signal last span of deeplin and flatprofile power level, connector loss tuned

and the evaluated ASE are plotted. The GSNr is plotted in Figure 5.6

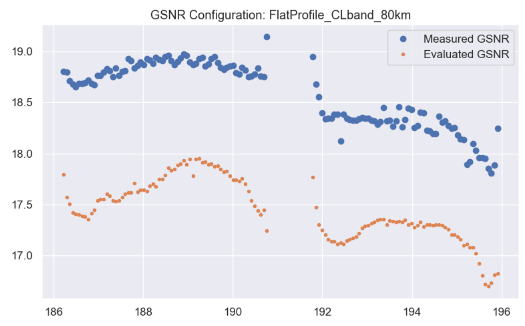


(a) DeepLIN span 5, ASE **(b)** Flatprofile span 5, ASE

Figure 5.5: ASE last span of deeplin and flatprofile power level, noise figure measured



(a) DeepLIN GSNR



(b) Flatprofile GSNR

Figure 5.6: GSNR of deeplin and flatprofile power level, connector loss tuned and noise figure measured

The errors that are present in the signal are due to the inaccuracy of the losses tuning process and in large part to the uncertainty of the model of the amplifier.

5.4 GSNR comparisons

In this last section, it has been carried out an analysis on the GSNR evaluated from different spectrum propagation. 3 different spectrum propagation has been compared. In the first case, the EDFA does not use a model to apply the gain, but uses the measure. The gain profile is evaluated as the difference between the measured signal power after the EDFA and the measured signal power at the input of the EDFA. The losses of the fused element are tuned for each power level. In the second case, the same implementation of EDFA is used, but in this case the same value of losses is used for the fused. The same value of Section 4.6 has been used. The Figure 5.7 shows the OSNR for the 3 cases described above and the measured value. All the curves are comparable, this because all the information related to the ASE noise used in the simulation is measured.

The Figure 5.8 shows the SNR due to the NLI. The evaluation of this parameter has been performed with the Generalized Gaussian Noise (GGN). The measured non-linear SNR is evaluated from the GSNR evaluated from the BER and the measured OSNR with the Equation 5.1:

$$SNR_{NL} = \left(\frac{1}{GSNR} - \frac{1}{OSNR}\right)^{-1} \tag{5.1}$$

The GSNR and OSNR are expressed in linear units.

The Figure 5.9 shows the GSNR for all spectrum propagation. The difference between the curve using the spectrum propagated through the amplifier model and

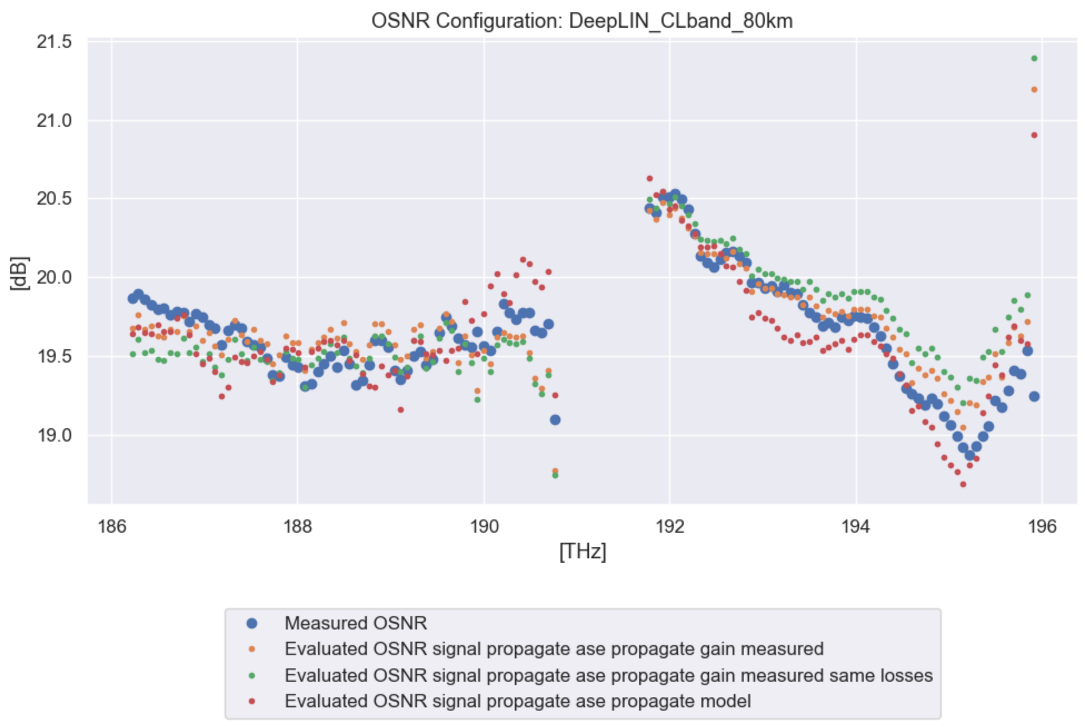


Figure 5.7: OSNR for Deeplin Power Level Evaluated with Different Spectrum Propagation

the curve using the measurement is very small. This is an additional motivation that demonstrates the goodness of the model. The estimated curve is always lower than the measured one, because the nonlinear SNR model is conservative. This is positive because the simulation estimates a result that is always below the real measure. From a communication point of view, this means that the analytical evaluation of the BER always leads to a value that is the worst to respect the real capacity of the network. Choosing a modulation format that respects the BER constraint evaluated from the simulated GSNR is a solution that already includes a margin of safety.

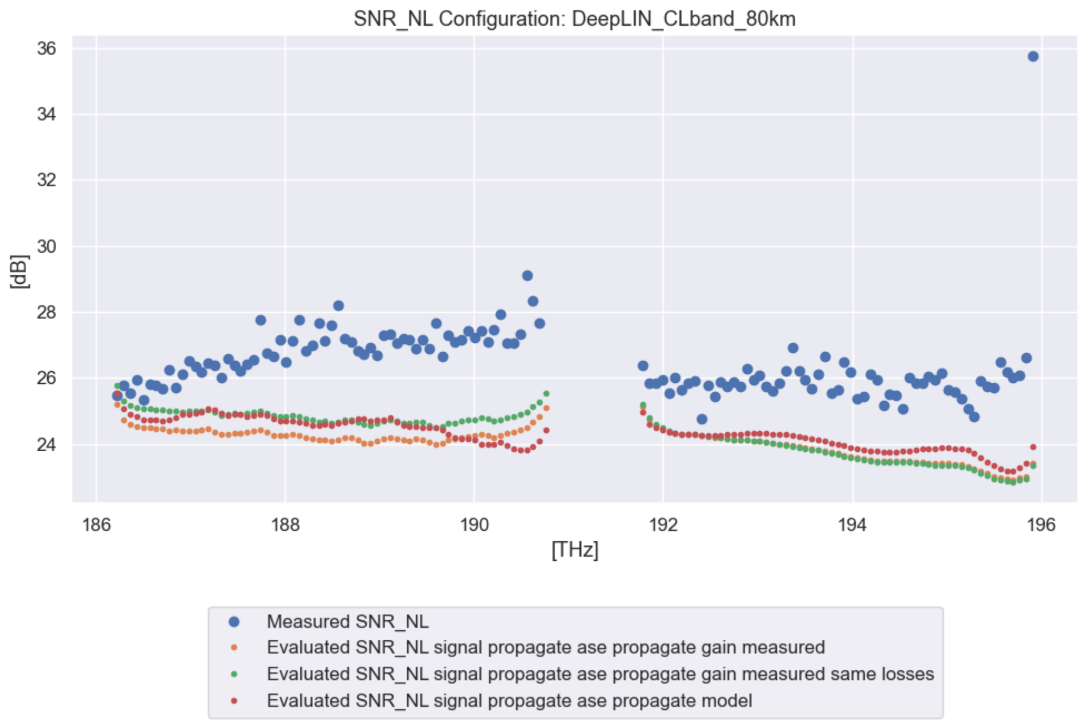


Figure 5.8: non-linear SNR for Deeplin Power Level Evaluated with Different Spectrum Propagation

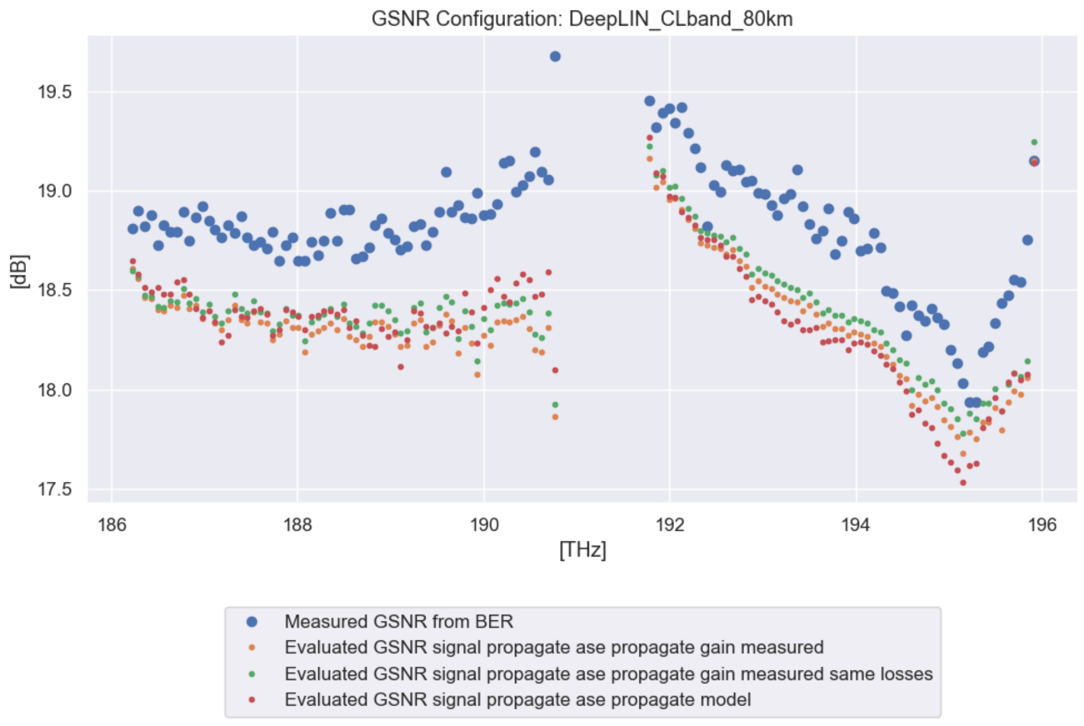


Figure 5.9: GSNR for Deeplin Power Level Evaluated with Different Spectrum Propagation

Chapter 6

Conclusion

The main focus has been to find and validate the model of the network element of the optical network and to understand if the GNPy models are accurate in a C+L band configuration. The results that has been obtained demonstrate that GNPy can be used to accurately simulate GSNR values even in the case of multi-band. The increase of information related to the fiber significantly increase the accuracy of the whole process, but the use of default value also lead to reasonable result. The model of the amplifier works well in the C+L scenario, it is important to have information related to the signal before and after the amplifier to apply a tuning of the connector losses, which is not possible to measure in an effective way. A field data collection system must be developed to achieve a level of accuracy that will significantly increase network efficiency over the next decade.

The evolution of the work can move to in-depth study of the noise component of EDFA. This type of analysis is difficult for several reasons: the high variability of the noise due to environmental interactions and the difficulty of measuring such a small value. Machine learning techniques can be used to characterize the noise figure of the amplifier, but this requires a lot of data from different EDFAs. An automatic system for measuring a large number of amplifiers must be created to generate a large enough data set.

Another possible future work can concern the analysis of the degradation of the performance of the EDFA in a long time range. Understanding if the functionality of the amplification of the EDFA can change due to the aging of the component is essential for the future network that will operate in a high efficiency context.

Appendix A

Appendix

A.1 DGT analysis

```
from pathlib import Path

import numpy as np
from numpy import array
from numpy import polyfit

from gnpy.core.elements import Edfa
from gnpy.core.equipment import trx_mode_params
from gnpy.core.utils import dbm2watt
from gnpy.tools.cli_examples import load_common_data
from gnpy.topology.request import PathRequest
from gnpy.core.info import create_arbitrary_spectral_information,
    Pref
from gnpy.topology.request import ref_carrier

import matplotlib.pyplot as plt


def load_params(equipment):
    source = None
    destination = None
    nb_channels = None
    nodes_list = None
    loose_list = None
    params = {'request_id': 0, 'trx_type': '', 'trx_mode': '', '
    source': source, 'destination': destination,
              'bidir': False, 'nodes_list': nodes_list, 'loose_list':
     loose_list, 'format': '', 'path_bandwidth': 0,
              'effective_freq_slot': None, 'nb_channel': nb_channels}

    trx_params = trx_mode_params(equipment, equipment['Transceiver'][
   'Lab_links'],
                                     equipment['Transceiver']['Lab_links'
   ].mode[0])
    params.update(trx_params)

    return params


def tilt_evaluation(p_in_array, gain_array, tilt_array, gain,
   frequency):
    tilt_evaluated = []
    for p, power_in in enumerate(p_in_array):
        gain_row = []
        for g, gain_ob in enumerate(gain_array):
            tilt_row = []
            for t, tilt_ob in enumerate(tilt_array):
                tilt_row.append(polyfit(frequency, gain[p, g, t], 1)
   [0] *
                                (max(frequency) - min(frequency)))
            gain_row.append(tilt_row)
        tilt_evaluated.append(gain_row)

    return array(tilt_evaluated)


def gain_gnpy_evaluation(info):
    equipment_path = ROOT / 'eqpt_config.json'
    topology_path = ROOT / 'my_test_04.json'
    sim_params_path = ROOT / '../../tasks/sim_params.json'
    save_network_before_autodesign_path = None

    (equipment, network) = load_common_data(equipment_path,
   topology_path, sim_params_path,

   save_network_before_autodesign_path)

    params = load_params(equipment)
    req = PathRequest(**params)

    edfas = {n.uid: n for n in network.nodes() if isinstance(n, Edfa)
   }
    test_edfa = edfas['EdfaAC-1']

    carrier = ref_carrier(equipment)

    gain_measure = []
    for p, pin_real in enumerate(info['p_in_array']):
```

```python
        pin_row = []
        for g, gain in enumerate(info['gain_array']):
            gain_row = []
            for t, tilt in enumerate(info['tilt_array']):
                test_edfa.effective_gain = gain
                test_edfa.params.f_min = min(info['frequency'])
                test_edfa.params.f_max = max(info['frequency'])
                test_edfa.tilt_target = info['tilt_array'][t]

                p_span0 = -60
                p_spani = -60

                si = create_arbitrary_spectral_information(info['frequency'], baud_rate=req.baud_rate,
                                                           signal=dbm2watt(pin_real), tx_osnr=req.tx_osnr,
                                                           ref_power=Pref(p_span0, p_spani, carrier))

                # propagate
                si_new = test_edfa(si)

                gain_row.append(test_edfa.gprofile)
            pin_row.append(array(gain_row))
        gain_measure.append(array(pin_row))

    return array(gain_measure), test_edfa.interpol_dgt


if __name__ == "__main__":
    ROOT = Path(__file__).parents[1]
    ROOT = ROOT / 'gnpy/example-data'

    my_info = {'p_in_array': array([-10.1, -8., -6.1, -4.1, -2.1, -0.1, 1.9, 3.9, 6.]),
               'gain_array': array([14., 15., 16., 17., 18., 19., 20.]),
               'tilt_array': array([-4, -3, -2, -1, 0, 1, 3, 5]),
               'frequency': array([1.91766289e+14, 1.91867542e+14, 1.91969516e+14, 1.92070983e+14,
                                   1.92168247e+14, 1.92269925e+14, 1.92369859e+14, 1.92470514e+14,
                                   1.92571275e+14, 1.92671523e+14, 1.92771255e+14, 1.92870966e+14,
                                   1.92971526e+14, 1.93070947e+14, 1.93171093e+14, 1.93273213e+14,
                                   1.93371698e+14, 1.93471532e+14, 1.93571469e+14, 1.93672135e+14,
                                            1.93770402e+14, 1.93872529e+14,
    1.93971626e+14, 1.94072080e+14,
                                            1.94172639e+14, 1.94272673e+14,
    1.94373440e+14, 1.94472419e+14,
                                            1.94572762e+14, 1.94671186e+14,
    1.94772367e+14, 1.94870486e+14,
                                            1.94969338e+14, 1.95066387e+14,
    1.95167344e+14, 1.95265861e+14,
                                            1.95365115e+14, 1.95463833e+14])}

    gain_gnpy, dgt_gnpy_test = gain_gnpy_evaluation(my_info)

    # dgt evaluation
    p_in = 0
    g = 0
    t1 = 5  # tilt = +1
    t2 = 4  # tilt = 0
    dgt = (gain_gnpy[p_in, g, t1, :] - gain_gnpy[p_in, g, t2, :]) / \
          (gain_gnpy[p_in, g, t1, -1] - gain_gnpy[p_in, g, t2, -1])
    dgt = (dgt + 1 - dgt[0])

    # real tilt clean
    np.seterr(invalid='ignore')
    gain_ripple_no_gain = (my_info['gain_array'][np.newaxis, :, np.
    newaxis, np.newaxis] - gain_gnpy)
    tilt_real = tilt_evaluation(my_info['p_in_array'], my_info['
    gain_array'], my_info['tilt_array'],
                                        gain_ripple_no_gain, my_info['
    frequency'])
    gain_ripple_no_tilt = np.divide(gain_ripple_no_gain, tilt_real[:,
     :, :, np.newaxis])
    gain_ripple_no_tilt_mean = np.nanmean(gain_ripple_no_tilt, axis
    =(0, 1, 2))

    rfe = (gain_gnpy[p_in, g, t2, -1] - gain_gnpy[p_in, g, t1, -1])

    ripple_profile = gain_ripple_no_tilt_mean + (1 -
    gain_ripple_no_tilt_mean[0])
    gain_ripple_no_tilt_mean = gain_ripple_no_tilt_mean / rfe
    gain_ripple_no_tilt_mean = gain_ripple_no_tilt_mean + (1 -
    gain_ripple_no_tilt_mean[0])

    plt.figure()
    plt.plot(my_info['frequency'], dgt_gnpy_test, '.b', label='DGT
    from file')
    plt.plot(my_info['frequency'], dgt, label='DGT evaluated')
    plt.grid()
    plt.xlabel('frequency [Hz]')
    plt.legend()

    plt.savefig("dgt_comparison.pdf", format="pdf")

    plt.figure()
    plt.plot(my_info['frequency'], dgt_gnpy_test, '.b', label='DGT
   from file')
    plt.plot(my_info['frequency'], gain_ripple_no_tilt_mean, label='
  DGT evaluated from ripple profile')
    plt.plot(my_info['frequency'], ripple_profile, label='ripple
  profile')
    plt.grid()
    plt.xlabel('frequency [Hz]')
    plt.legend()

    plt.savefig("dgt_comparison_ripple_profile.pdf", format="pdf")
    plt.show()
```

A.2 Comparison between measured gain profile and GNPY dgt gain profile

```
from pathlib import Path

import numpy as np
import pandas as pd
from numpy import array, sqrt, mean
from numpy import unique, isnan, polyfit

from gnpy.core.elements import Edfa
from gnpy.core.equipment import trx_mode_params
from gnpy.core.utils import dbm2watt
from gnpy.tools.cli_examples import load_common_data
from gnpy.topology.request import PathRequest
from gnpy.core.info import create_arbitrary_spectral_information,
    Pref
from gnpy.topology.request import ref_carrier

import matplotlib.pyplot as plt
from matplotlib.widgets import Slider


def filtering_nan(arr):
    return arr[~ isnan(arr)]


```

```python
def get_all_sub_folder(ROOT, value):
    amp_path = ROOT / 'amplifiers/'
    amplifiers = [x for x in amp_path.iterdir() if x.is_dir()]
    amplifier_gain = []
    for amplifier in amplifiers:
        sub_folders = [x for x in amplifier.iterdir() if x.is_dir()]
        for sub in sub_folders:
            amplifier_gain.append(sub / value)
    return amplifier_gain


def load_params(equipment):
    source = None
    destination = None
    nb_channels = None
    nodes_list = None
    loose_list = None
    params = {'request_id': 0, 'trx_type': '', 'trx_mode': '', 'source': source, 'destination': destination,
              'bidir': False, 'nodes_list': nodes_list, 'loose_list': loose_list, 'format': '', 'path_bandwidth': 0,
              'effective_freq_slot': None, 'nb_channel': nb_channels}

    trx_params = trx_mode_params(equipment, equipment['Transceiver']['Lab_links'],
                                 equipment['Transceiver']['Lab_links'].mode[0])
    params.update(trx_params)

    return params


def tilt_evaluation(p_in_array, gain_array, tilt_array, gain, frequency):
    tilt_evaluated = []
    for p, power_in in enumerate(p_in_array):
        gain_row = []
        for g, gain_ob in enumerate(gain_array):
            tilt_row = []
            for t, tilt_ob in enumerate(tilt_array):
                tilt_row.append(polyfit(frequency, gain[p, g, t], 1)[0] *
                                (max(frequency) - min(frequency)))
            gain_row.append(tilt_row)
        tilt_evaluated.append(gain_row)

    return array(tilt_evaluated)


```

```python
def info_from_dataframe(amp):
    # find f_min, f_max and n_channel for the first set of parameters (first row)
    frequency = amp[[col for col in amp.columns if 'freq' in col]].values[0, :] * 1e12
    frequency = array(frequency, dtype='float64')

    p_in_array = filtering_nan(unique(amp.pin_real.values))
    gain_array = filtering_nan(unique(amp.gain_target.values))
    tilt_array = filtering_nan(unique(amp.tilt_target.values))

    gain = amp[[col for col in amp.columns if 'gain_profile' in col]].values
    gain = gain.reshape([p_in_array.size, gain_array.size, tilt_array.size, frequency.size])

    gain_ripple_real = amp[[col for col in amp.columns if 'gain_ripple' in col]].values
    gain_ripple_real = gain_ripple_real.reshape([p_in_array.size, gain_array.size, tilt_array.size, frequency.size])

    tilt_evaluated = tilt_evaluation(p_in_array, gain_array, tilt_array, gain_ripple_real, frequency)

    gain_ripple_no_tilt = gain_ripple_real / np.repeat(
        tilt_evaluated[:, :, :, np.newaxis],
        gain_ripple_real.shape[3], axis=3)

    gain_ripple_no_tilt_mean = np.mean(gain_ripple_no_tilt, axis=(0, 1, 2))

    tilt_real = amp.tilt_real.values.reshape([p_in_array.size, gain_array.size, tilt_array.size])
    tilt_target = amp.tilt_target.values.reshape([p_in_array.size, gain_array.size, tilt_array.size])

    # safe check
    p_out_max = max(amp.loc[:, 'pout_real'])
    tilt_th = 0.21
    pout_check = amp.pout_target.values.reshape([p_in_array.size, gain_array.size, tilt_array.size]) <= p_out_max
    tilt_check = abs(tilt_target - tilt_real) <= tilt_th
    safe_zone = pout_check & tilt_check

    temp = {'frequency': frequency, 'p_in_array': p_in_array, 'gain_array': gain_array, 'tilt_array': tilt_array,
            'gain': gain, 'gain_ripple': gain_ripple_real, 'gain_ripple_no_tilt': gain_ripple_no_tilt,
            'gain_ripple_no_tilt_mean': gain_ripple_no_tilt_mean,
            'tilt_real': tilt_real, 'tilt_target': tilt_target, 'tilt_evaluated': tilt_evaluated,
            'safe_zone': safe_zone}

    return temp


def gain_gnpy_evaluation(info):
    equipment_path = ROOT / 'eqpt_config.json'
    topology_path = ROOT / 'my_test_04.json'
    sim_params_path = ROOT / '../../tasks/sim_params.json'
    save_network_before_autodesign_path = None

    (equipment, network) = load_common_data(equipment_path, topology_path, sim_params_path,
                                            save_network_before_autodesign_path)

    params = load_params(equipment)
    req = PathRequest(**params)

    edfas = {n.uid: n for n in network.nodes() if isinstance(n, Edfa)}
    test_edfa = edfas['EdfaAC-1']

    carrier = ref_carrier(equipment)

    gain_measure = []
    for p, pin_real in enumerate(info['p_in_array']):
        pin_row = []
        for g, gain in enumerate(info['gain_array']):
            gain_row = []
            for t, tilt in enumerate(info['tilt_array']):
                test_edfa.effective_gain = gain
                test_edfa.params.f_min = min(info['frequency'])
                test_edfa.params.f_max = max(info['frequency'])

                test_edfa.tilt_target = info['tilt_target'][p, g, t]
                tilt_0 = int(np.where(info['tilt_array'] == 0)[0])
                test_edfa.params.gain_ripple = info['gain_ripple_no_tilt_mean']
                test_edfa.params.dgt = dgt_evaluation(info['gain'], tilt_0)

                p_span0 = -60
                p_spani = -60

                si = create_arbitrary_spectral_information(info['frequency'], baud_rate=req.baud_rate,
                                                                          signal=
    dbm2watt(pin_real), tx_osnr=req.tx_osnr,
                                                                          ref_power=
    Pref(p_span0, p_spani, carrier))

                # propagate
                si_new = test_edfa(si)

                gain_row.append(test_edfa.gprofile)

            pin_row.append(array(gain_row))
        gain_measure.append(array(pin_row))
    return array(gain_measure)


def dgt_evaluation(gains, tilt_0):
    gains_for_dgt = gains[0, 0, :, :]  # all the input power and all
    the frequency

    f0_ind = -1  # int(frequency.size / 2)
    g1 = gains_for_dgt[tilt_0, :]
    g2 = gains_for_dgt[tilt_0 + 1, :]

    dgt = (g2 - g1) / (g2[f0_ind] - g1[f0_ind])
    return dgt


if __name__ == "__main__":
    ROOT = Path(__file__).parents[1]
    ROOT = ROOT / 'gnpy/example-data'

    # import configuration
    paths = get_all_sub_folder(ROOT, 'post_processing/
    characterization.csv')
    for i, path in enumerate(paths):
        print(i, ': ', path)

    value = input('Insert number of the line (if -1 all line): ')
    amps = pd.read_csv(paths[int(value)])

    info = info_from_dataframe(amps)
    gain_gnpy = gain_gnpy_evaluation(info)

    # Plot 4
    fig, ax = plt.subplots()
    plt.subplots_adjust(left=0.25, bottom=0.25)

    index_gain = 1
    index_tilt = 1

    gain_slider_ax = plt.axes([0.25, 0.1, 0.65, 0.03])
    gain_slider = Slider(gain_slider_ax, 'Gain Index', 0, len(info['gain_array']) - 1, valinit=index_gain,
                                 valstep=1)

    tilt_slider_ax = plt.axes([0.25, 0.05, 0.65, 0.03])
    tilt_slider = Slider(tilt_slider_ax, 'Tilt Index', 0, len(info['tilt_array']) - 1, valinit=index_tilt,
                                 valstep=1)


    def update(val):
        index_gain = int(gain_slider.val)
        index_tilt = int(tilt_slider.val)
        ax.cla()  # clear the previous plot
        ax.set_title('Gain profile in frequency, fixed gain: %f and tilt: %f' % (info['gain_array'][index_gain],
                     info['tilt_array'][index_tilt]))

        p_in = 5
        ax.plot(info['frequency'], info['gain'][p_in, index_gain, index_tilt, :], 'r',
                     label='measured gain profile')
        ax.plot(info['frequency'], gain_gnpy[p_in, index_gain, index_tilt, :], 'g',
                     label='evaluated gain profile with dgt')

        ax.set_xlabel('frequency [Hz]')
        ax.set_ylabel('gain [dB]')

        ax.legend()
        ax.grid()
        fig.canvas.draw_idle()


    update(1)
    gain_slider.on_changed(update)
    tilt_slider.on_changed(update)
#################################################################
    index_gain = 2
    index_tilt = 5
    plt.figure()
    plt.plot(info['frequency'], gain_gnpy[0, index_gain, index_tilt, :], '.r',
             label=f'evaluated gain profile with dgt power input: {info["p_in_array"][0]} dBm')
    plt.plot(info['frequency'], gain_gnpy[1, index_gain, index_tilt,
   :], 'g',
             label=f'evaluated gain profile with dgt power input: {
   info["p_in_array"][1]} dBm')

    plt.legend()
    plt.xlabel('frequency [Hz]')
    plt.ylabel('gain [dB]')
    plt.grid()
    plt.savefig(f'GNPy_power_comparison.pdf', format="pdf")
################################################################
    index_gain = 2
    index_tilt = 5
    plt.figure()
    plt.plot(info['frequency'],  info['gain'][0, index_gain,
   index_tilt, :],
             label=f'evaluated gain profile with dgt power input: {
   info["p_in_array"][0]} dBm')
    plt.plot(info['frequency'],  info['gain'][1, index_gain,
   index_tilt, :],
             label=f'evaluated gain profile with dgt power input: {
   info["p_in_array"][1]} dBm')
    plt.plot(info['frequency'], info['gain'][2, index_gain,
   index_tilt, :],
             label=f'evaluated gain profile with dgt power input: {
   info["p_in_array"][2]} dBm')
    plt.plot(info['frequency'], info['gain'][3, index_gain,
   index_tilt, :],
             label=f'measured gain profile with dgt power input: {
   info["p_in_array"][3]} dBm')

    plt.legend()
    plt.xlabel('frequency [Hz]')
    plt.ylabel('gain [dB]')
    plt.grid()
    plt.savefig(f'measured_power_comparison.pdf', format="pdf")
################################################################
    p_in = 5
    index_gain = 2
    index_tilt = 5
    plt.figure()
    plt.plot(info['frequency'], info['gain'][p_in, index_gain,
   index_tilt, :], 'r',
             label='measured gain profile')
    plt.plot(info['frequency'], gain_gnpy[p_in, index_gain,
   index_tilt, :], 'g',
             label='evaluated gain profile with dgt')
    plt.legend()
    plt.xlabel('frequency [Hz]')
    plt.ylabel('gain [dB]')
    plt.grid()
    plt.savefig(f'GNPy_dgt_vs_measured_gain_{info["gain_array"][index_gain]}_tilt_{info["tilt_array"][index_tilt]}.pdf',
                format="pdf")

    plt.show()
```

A.3 Evaluation of K(f) parameter for different tilt value

```
from pathlib import Path

import numpy as np
import pandas as pd
import json
from numpy import array
from numpy import unique, isnan, polyfit

from gnpy.core.equipment import trx_mode_params
import matplotlib.pyplot as plt


def filtering_nan(arr):
    return arr[~ isnan(arr)]


def get_all_sub_folder(ROOT, value):
    amp_path = ROOT / 'amplifiers/'
    amplifiers = [x for x in amp_path.iterdir() if x.is_dir()]
    amplifier_gain = []
    for amplifier in amplifiers:
        sub_folders = [x for x in amplifier.iterdir() if x.is_dir()]
        for sub in sub_folders:
            amplifier_gain.append(sub / value)
    return amplifier_gain


def load_params(equipment):
    source = None
    destination = None
    nb_channels = None
    nodes_list = None
    loose_list = None
    params = {'request_id': 0, 'trx_type': '', 'trx_mode': '', 'source': source, 'destination': destination,
              'bidir': False, 'nodes_list': nodes_list, 'loose_list': loose_list, 'format': '', 'path_bandwidth': 0,
              'effective_freq_slot': None, 'nb_channel': nb_channels}

    trx_params = trx_mode_params(equipment, equipment['Transceiver']['Lab_links'],
                                 equipment['Transceiver']['Lab_links'].mode[0])
    params.update(trx_params)

    return params


def tilt_evaluation(p_in_array, gain_array, tilt_array, gain, frequency):
    tilt_evaluated = []
    for p, power_in in enumerate(p_in_array):
        gain_row = []
        for g, gain_ob in enumerate(gain_array):
            tilt_row = []
            for t, tilt_ob in enumerate(tilt_array):
                tilt_row.append(polyfit(frequency, gain[p, g, t], 1)[0] * (max(frequency) - min(frequency)))
            gain_row.append(tilt_row)
        tilt_evaluated.append(gain_row)

    return array(tilt_evaluated)


def tilt_vector(info, band, f0):
    tilts = \
        np.squeeze([[[- info['tilt_target'][p, g, t] / band * array([f - f0 for f in info['frequency'] / 1e12])
                      for t in range(info['tilt_array'].size)]
                     for g in range(info['gain_array'].size)]
                    for p in range(info['p_in_array'].size)])
    return tilts


def info_from_dataframe(amp):
    # find f_min, f_max and n_channel for the first set of parameters (first row)
    frequency = amp[[col for col in amp.columns if 'freq' in col]].values[0, :] * 1e12
    frequency = array(frequency, dtype='float64')

    p_in_array = filtering_nan(unique(amp.pin_real.values))
```

```python
    gain_array = filtering_nan(unique(amp.gain_target.values))
    tilt_array = filtering_nan(unique(amp.tilt_target.values))

    gain = amp[[col for col in amp.columns if 'gain_profile' in col]].values
    gain = gain.reshape([p_in_array.size, gain_array.size, tilt_array.size, frequency.size])

    gain_ripple_real = amp[[col for col in amp.columns if 'gain_ripple' in col]].values
    gain_ripple_real = gain_ripple_real.reshape([p_in_array.size, gain_array.size, tilt_array.size, frequency.size])

    tilt_evaluated = tilt_evaluation(p_in_array, gain_array, tilt_array, gain_ripple_real, frequency)

    gain_ripple_no_tilt = gain_ripple_real / np.repeat(
        tilt_evaluated[:, :, :, np.newaxis],
        gain_ripple_real.shape[3], axis=3)

    gain_ripple_no_tilt_mean = np.mean(gain_ripple_no_tilt, axis=(0, 1, 2))

    tilt_real = amp.tilt_real.values.reshape([p_in_array.size, gain_array.size, tilt_array.size])
    tilt_target = amp.tilt_target.values.reshape([p_in_array.size, gain_array.size, tilt_array.size])

    # safe check
    p_out_max = max(amp.loc[:, 'pout_real'])
    tilt_th = 0.21
    pout_check = amp.pout_target.values.reshape([p_in_array.size, gain_array.size, tilt_array.size]) <= p_out_max
    tilt_check = abs(tilt_target - tilt_real) <= tilt_th

    safe_zone = pout_check & tilt_check

    temp = {'frequency': frequency, 'p_in_array': p_in_array, 'gain_array': gain_array, 'tilt_array': tilt_array,
            'gain': gain, 'gain_ripple': gain_ripple_real, 'gain_ripple_no_tilt': gain_ripple_no_tilt,
            'gain_ripple_no_tilt_mean': gain_ripple_no_tilt_mean,
            'tilt_real': tilt_real, 'tilt_target': tilt_target, 'tilt_evaluated': tilt_evaluated,
            'safe_zone': safe_zone}

    return temp


```

```
def k_evaluation(info, tilt_0, tilt_1, fe=-1):
    gains_for_dgt = info['gain'][0, 0, :, :]  # all the input power
    and all the frequency
    g1 = gains_for_dgt[tilt_0, :]
    g2 = gains_for_dgt[tilt_1, :]
    k = (g2[fe] - g1[fe])

    return k


def diff_evaluation(info, gain_no_gain_no_tilt):
    gain_measure = []
    tilt_0 = int(np.where(info['tilt_array'] == 0)[0])

    for p, pin_real in enumerate(info['p_in_array']):
        pin_row = []
        for g, gain in enumerate(info['gain_array']):
            gain_row = []
            for t, tilt in enumerate(info['tilt_array']):
                tilt_row = gain_no_gain_no_tilt[p, g, t] -
    gain_no_gain_no_tilt[p, g, tilt_0]
                gain_row.append(tilt_row)
            pin_row.append(array(gain_row))
        gain_measure.append(array(pin_row))
    return array(gain_measure)


if __name__ == "__main__":
    ROOT = Path(__file__).parents[1]
    ROOT = ROOT / 'gnpy/example-data'

    # import configuration
    paths = get_all_sub_folder(ROOT, 'post_processing/
    characterization.csv')
    paths1 = get_all_sub_folder(ROOT, 'post_processing/amp_info.json'
    )
    for i, path in enumerate(paths):
        print(i, ': ', path)

    value = input('Insert number of the line (if -1 all line): ')
    amps = pd.read_csv(paths[int(value)])

    info = info_from_dataframe(amps)

    with open(paths1[int(value)]) as f:
        info1 = json.load(f)

    gain_offset = info['gain_array'][np.newaxis, :, np.newaxis, np.
    newaxis]
    tilt_offset = tilt_vector(info, info1['band'], info1['central_frequency'])
    ripple = info['gain'] - gain_offset - tilt_offset
    gain_no_0_profile = diff_evaluation(info, ripple)

    plt.figure()
    for t, tilt in enumerate(info['tilt_target'][0][0]):
        if tilt != 0:
            k = gain_no_0_profile[0, 0, t, :] / info['tilt_array'][t]  # (r0 -r_max)/tilt_max
            plt.plot(info['frequency'], k, label=f"Tilt {tilt} dB")

    plt.legend()
    plt.xlabel('Frequency [Hz]')
    plt.savefig(f'K_changing_tilt_cisco_edfa17.pdf', format="pdf")

    plt.grid()
    plt.show()
```

Acronyms

ASE Amplified Spontaneous Emission

SNR Signal to Noise Ratio

GSNR Generalized Signal to Noise Ratio

NLI Non-Linear Interference

OSNR Optical Signal to Noise Ratio

RMSE Root Mean Square Error

SSMF Standard Single-mode Fiber
SRS Stimulated Raman Scattering

CD Chromatic Dispersion

VOA Variable Optical Attenuator

QoT Quality of Transmission

AGC Automatic Gain Control

NN Neural Network

EDFA Erbium Doped Fiber Amplifier

CM Center of Mass

MSE Mean Square Error

DGT Dynamic Gain Tilt

OCM Optical Channel Monitor

SSH Secure Shell

WDM	Wavelength Division Multiplexed
WSS	Wavelength Selective Switch
OSA	Optical Spectrum Analyser
BER	Bit Error Probability
SDN	Software Defined Network
OLS	Optical Line System
OPEX	OPerational EXpenditure
CWDM	Coarse Wavelength Division Multiplexing
DWDM	Dense Wavelength Division Multiplexing
IMDD	Intensity Modulation with Direct Detection
ITU	International Telecommunication Union
DSP	Digital Signal Procesing
ROADM	Reconfigurable Optical Add-Drop Multiplexer
FEC	Forward Error Correction
SOA	Semiconductor Optical Amplifiers
ES	Evolution Strategies
GGN	Generalized Gaussian Noise

Bibliography

[1] G. P. Agrawal. «Optical Communication: Its History and Recent Progress». In: *Optics in Our Time*. Ed. by M. D. Al-Amri et al. Cham: Springer International Publishing, 2016, pp. 177–199. ISBN: 978-3-319-31903-2. DOI: `10.1007/978-3-319-31903-2_8`. URL: `https://doi.org/10.1007/978-3-319-31903-2_8` (cit. on p. 3).

[2] F. P. Kapron et al. «RADIATION LOSSES IN GLASS OPTICAL WAVEGUIDES». In: *Applied Physics Letters* 17.10 (Oct. 2003), pp. 423–425. ISSN: 0003-6951. DOI: `10.1063/1.1653255`. eprint: `https://pubs.aip.org/aip/apl/article-pdf/17/10/423/7728944/423\_1\_online.pdf`. URL: `https://doi.org/10.1063/1.1653255` (cit. on p. 3).

[3] T. Miya et al. «Ultimate low-loss single-mode fibre at 1.55 ?m». Undetermined. In: *Electronics Letters* 15.4 (1979), pp. 106–108. DOI: `10.1049/el:19790077` (cit. on p. 3).

[4] CC BY-SA 3.0. Christophe Finot. URL: `https://creativecommons.org/licenses/by-sa/3.0` (cit. on p. 4).

[5] B. Keyworth. «ROADM subsystems and technologies». In: *OFC/NFOEC Technical Digest. Optical Fiber Communication Conference, 2005.* Vol. 3. 2005, pp. 4–3. DOI: `10.1109/OFC.2005.192706` (cit. on p. 5).

[6] M. Islam. «Raman amplifiers for telecommunications». In: *IEEE Journal of Selected Topics in Quantum Electronics* 8.3 (2002), pp. 548–559. DOI: `10.1109/JSTQE.2002.1016358` (cit. on p. 5).

[7] R. Stolen et al. «Optical Kerr effect in glass waveguide». In: *Applied Physics Letters* 22.6 (Oct. 2003), pp. 294–296. ISSN: 0003-6951. DOI: `10.1063/1.1654644`. eprint: `https://pubs.aip.org/aip/apl/article-pdf/22/6/294/7607253/294\_1\_online.pdf`. URL: `https://doi.org/10.1063/1.1654644` (cit. on p. 7).

[8] R. H. Stolen et al. «Raman gain in glass optical waveguides». In: *Applied Physics Letters* 22.6 (Oct. 2003), pp. 276–278. ISSN: 0003-6951. DOI: `10.1063/1.1654637`. eprint: `https://pubs.aip.org/aip/apl/article-pdf/22/6/276/7607319/276\_1\_online.pdf`. URL: `https://doi.org/10.1063/1.1654637` (cit. on p. 8).

[9] W. H. Xu Zhang Lei Guo et al. «Failure recovery solutions using cognitive mechanisms based on software-defined optical network platform». In: *Optical Engineering*. 2017. DOI: `https://doi.org/10.1117/1.OE.56.1.016107` (cit. on p. 8).

[10] H. Zheng et al. *A YANG Data Model for Optical Transport Network Topology*. Internet-Draft draft-ietf-ccamp-otn-topo-yang-17. Work in Progress. Internet Engineering Task Force, July 2023. 80 pp. URL: `https://datatracker.ietf.org/doc/draft-ietf-ccamp-otn-topo-yang/17/` (cit. on p. 9).

[11] A. E. E. Kamal Benzekki Abdeslam El Fergougui. «Software-defined networking (SDN): a survey». In: (2017), pp. 1–3. DOI: `https://doi.org/10.1002/sec.1737` (cit. on p. 10).

[12] K. Benzekki et al. «Software-defined networking (SDN): a survey». In: *Security and Communication Networks* 9.18 (2016), pp. 5803–5833. DOI: `https://doi.org/10.1002/sec.1737`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.1737`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1737` (cit. on p. 10).

[13] B. Mukherjee. «WDM optical communication networks: progress and challenges». In: *IEEE Journal on Selected Areas in Communications* 18.10 (2000), pp. 1810–1824. DOI: `10.1109/49.887904` (cit. on p. 10).

[14] E. VanDerHorn et al. «Digital Twin: Generalization, characterization and implementation». In: *Decision Support Systems* 145 (2021), p. 113524. ISSN: 0167-9236. DOI: `https://doi.org/10.1016/j.dss.2021.113524`. URL: `https://www.sciencedirect.com/science/article/pii/S0167923621000348` (cit. on p. 11).

[15] V. Curri. «GNPy model of the physical layer for open and disaggregated optical networking [Invited]». In: *Journal of Optical Communications and Networking* 14.6 (2022), pp. C92–C104. DOI: `10.1364/JOCN.452868` (cit. on p. 13).

[16] S. Zhu et al. «Machine Learning Based Prediction of Erbium-Doped Fiber WDM Line Amplifier Gain Spectra». In: *2018 European Conference on Optical Communication (ECOC)*. IEEE. 2018, pp. 1–3. DOI: `10.1109/ECOC.2018.8535323` (cit. on p. 18).

[17] K. Ishii et al. «Wavelength assignment dependency of AGC EDFA gain offset under dynamic optical circuit switching». In: (2014), pp. 1–3. DOI: 10.1364/OFC.2014.W3E.4 (cit. on p. 18).

[18] S. Zhu et al. «Hybrid machine learning EDFA model». In: *Optical Fiber Communication Conference*. Optical Society of America. 2020, T4B–4. DOI: 10.1364/OFC.2020.T4B.4 (cit. on p. 19).

[19] A. Mahajan et al. «Modeling EDFA Gain Ripple and Filter Penalties With Machine Learning for Accurate QoT Estimation». In: *Journal of Lightwave Technology* 38.9 (2020), pp. 2616–2629. DOI: 10.1109/JLT.2020.2975081 (cit. on p. 19).

[20] R. D. Muro. «The Er3+-Fiber Gain Coefficient Derived from a Dynamic Gain Tilt Technique». In: *Journal of Lightwave Technology* 18.3 (2000), pp. 343–347. DOI: 10.1109/50.827506 (cit. on p. 21).

[21] A. D'Amico et al. «Experimental validation of GNPy in a multi-vendor flex-grid flex-rate WDM optical transport scenario». In: *J. Opt. Commun. Netw.* 14.3 (Mar. 2022), pp. 79–88. DOI: 10.1364/JOCN.442208. URL: https://opg.optica.org/jocn/abstract.cfm?URI=jocn-14-3-79 (cit. on p. 53).

[22] A. D'Amico et al. «GNPy Experimental Validation for Nyquist Subcarriers Flexible Transmission up to 800 G». In: *2022 Optical Fiber Communications Conference and Exhibition (OFC)*. 2022, pp. 1–3 (cit. on p. 53).