People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Saad Dahlab University – Blida 1

Institute of Aeronautics and Space Studies

Department of Space Studies

submitted for the degree of

Master of Space Telecommunications

Title:

AI-Based Filtering of Signals Affected by Random Noise

by

Mr Abdeldjalil, Mechmeche

Before the jury composed of:

Kheireddine Choutri	Assistant Professor (MCA)	President
Boussad Azmedroub	Lecturer (MCB)	Examiner
Sofiane Tahraoui	Assistant Professor (MCA)	Supervisor

Defense Date: 21/07/2025

بسم الله الرحمان الرحيم

نحمد الله اولا ونستعين به ونستهديه ونتوب اليه فالحمد لله الذي هدانا لهذا ومكنا لنهتدي لولا ان هدانا الله

Dedication

I dedicate this work primarily to my family, who have supported me at long academic journey. I extend my sincere thanks to my dear parents and siblings. Thank you very much, and I hope to live up to your expectations.

I dedicate this work in particular to all my professors at the Institute of Aeronautical and Space Studies, and especially to my academic supervisor Dr. sofiane tahraoui, who made every effort to ensure my success in this project. I would like to thank you all for all the education and support you have provided. I wish you all the best.

Finally, special thanks to all my friends at the telecommunication class and the staff of the Aeronautics Institute, from the administration and various employees to the transportation staff. Thank you as well.

Contents

العربية

تعد المعالجة الرقمية للإشارة جوهر التكنولوجيا الحديثة، وتُستخدم على نطاق واسع في مجالات مُتنوعة، بما في ذلك الأمن والاتصالات، الطب واستكشاف الفضاء، وغيرها من المجالات التي تعتمد على التقنيات الرقمية أو إشارات الراديو. اليوم أصبح من الممكن دمج تقنيات معالجة الإشارات مع الذكاء الاصطناعي، مما يُمكّن من الأنظمة من إطلاق كامل إمكاناتها واستغلالها بفعالية. يهدف هذا العمل إلى تطوير، خوارزمية تعلم الي تُسمى نظام النبضات التكيفية، تجمع بين أدوات معالجة الإشارات وقدرات تعلّم الذكاء الاصطناعي لبناء مُرشّحات رقمية وتعليمها واستخدامها بشكل الأمثل في حل إحدى أهم المشاكل في عالم الاتصالات اللاسلكية. تُعدّ التأثيرات الخارجية، مثل الضوضاء والتلاشي، من المشاكل الرئيسية في اتصالات الأقمار الصناعية اللاسلكية، حيث تؤثر سلبًا على جودة الإرسال. تُمثّل نسبة الموجة الحاملة الى الضوضاء عمومًا شدة هذه التأثيرات حيث تُمثّل النسبة الأقل من الصفر ظروف إشارة سيئة للغاية يصعب فيها بل يستحيل فيها استخراج البيانات بشكل موثوق، تهدف خوارزمية نظام النبضات التكيفية إلى بناء وتعليم أنظمة ترشيح رقمية ذكية قادرة على تعامل مع الحالات الصعبة ورفع نسب الموجة الحاملة إلى الضوضاء الأقل من الصفر الى ما يتجاوز 10 ديسيبل.

English

Digital signal processing (DSP) is at the core of modern technology and is widely used in diverse fields, including security, communications, medicine, space exploration, and other areas that rely on digital technologies or radio signals. Today, signal processing techniques can be combined with artificial intelligence (AI), enabling systems to unleash their full potential and effectively exploit them. This work aims to develop a machine learning algorithm, called Adaptive Pulses System (APS), that combines signal processing tools with AI learning capabilities to build, learn, and optimally utilize digital filters to solve one of the most significant problems in wireless communications. External influences such as noise and fading are major problems in wireless satellite communications, negatively impacting transmission quality. The carrier-to-noise ratio (C/N) generally represents the severity of these influences, with a C/N ratio below zero representing extremely poor signal conditions where reliable data extraction is difficult or impossible. The APS algorithm aims to build and learn high-performance DSP filters capable of raising the C/N ratio below zero and achieving a signal processing gain of more than 10 dB.

Français

Le traitement numérique du signal (DSP) est au cœur des technologies modernes et est largement utilisé dans divers domaines, notamment la sécurité, les communications, la médecine, l'exploration spatiale et d'autres domaines s'appuyant sur les technologies numériques ou les signaux radio. Aujourd'hui, les techniques de traitement du signal peuvent être combinées à l'intelligence artificielle (IA), permettant aux systèmes d'exploiter pleinement leur potentiel. Ce travail vise à développer un algorithme d'apprentissage automatique, appelé Adaptive Pulses System (APS), qui combine des outils de traitement du signal avec des capacités d'apprentissage de l'IA pour créer, apprendre et optimiser l'utilisation de filtres numériques afin de résoudre l'un des problèmes les plus importants des communications sans fil. Les influences externes telles que le bruit et l'évanouissement constituent des problèmes majeurs dans les communications sans fil par satellite, impactant négativement la qualité de transmission. Le rapport porteuse/bruit (C/N) reflète généralement la gravité de ces influences, un rapport C/N inférieur à zéro représentant des conditions de signal extrêmement mauvaises, rendant l'extraction de données fiable difficile, voire impossible. L'algorithme APS vise à créer et à apprendre des filtres DSP hautes performances capables d'élever le rapport C/N en dessous de zéro et d'atteindre un gain de traitement du signal supérieur à 10 dB.

General Introduction

In today's world, the demand for high-speed and reliable wireless communication continues to grow across a wide range of applications, from satellite broadcasting to mobile communications and deep-space exploration. One of the primary challenges hindering the performance and efficiency of communication systems is the persistent presence of noise, which can significantly degrade signal integrity and compromise data transmission quality.

The increasing complexity of communication environments, combined with the limitations of traditional signal processing techniques, has prompted researchers to explore more intelligent and adaptive solutions. Among these, the integration of artificial intelligence (AI) into signal processing has shown considerable promise. AI offers powerful tools capable of learning patterns, adapting to dynamic conditions, and effectively mitigating various types of interference, including random and colored noise, multipath fading, and thermal noise.

This thesis addresses the issue of noise in satellite communication links by proposing a novel framework based on Adaptive Pulse Systems (APS). These systems are designed using machine learning techniques to function as intelligent digital filters that adaptively reduce noise from modulated signals, particularly those affected by binary phase shift keying (BPSK) quadrature phase shift keying (QPSK), and other advanced modulation schemes. The work combines deep theoretical analysis with practical system modeling, digital modulation techniques, and Al-driven filter training and testing.

The research begins by examining the origins and types of noise in communication channels, including atmospheric noise, electronic thermal noise, and multipath propagation. It then delves into modulation methods and their sensitivity to noise, setting the foundation for designing effective denoising models. The proposed APS model is implemented using machine learning principles, leveraging convolutional filters trained on noisy and clean signal pairs.

Through simulation and evaluation, the developed system demonstrates its capability to improve signal quality, offering a scalable and intelligent approach to signal restoration. This work contributes to the advancement of robust, adaptive communication systems and paves the way for future integration of Al in satellite and wireless technologies.

1 The origin of radio noise

1.1 Introduction

The emergence of deterioration factors is both unexpected and inevitable in communication channels. These factors are numerous, such as noise and attenuation, and vary in their sources, types, and impact on each other. These factors pose a major problem in wireless communications, leading to weak or even nonexistent connectivity. Finding solutions to these factors requires a firm understanding of their origin and their evolution.

1.2 Satellite Communication Link

The channel propagation within the satellite link is a critical element. This is due to several influences such as atmospheric conditions, adverse effects of the transceiver, and the propagation area. The efficiency of the transmission link can be reduced or even eliminated in some cases. We can define the most important influences on the signal by following the path of a signal between a transmitter and receiver. Figure 1.1 shows the basic component in satellite link.

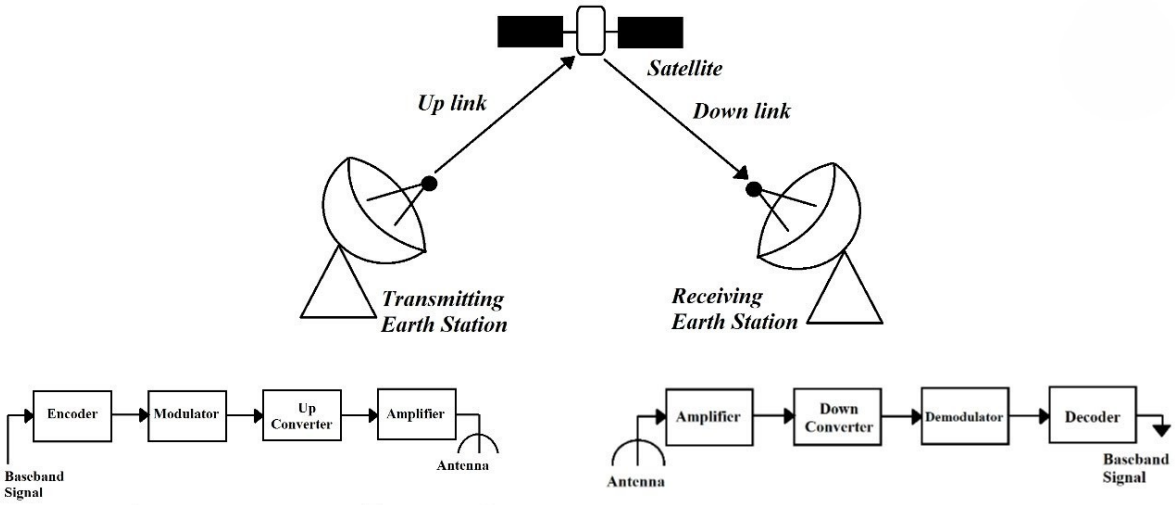

Figure 1.1: Signal path in satellite link

The Uplink Path

Uplink communication begins at the ground station by generating and modulating information, then converting it to the uplink frequency within the operating band. In satellite links, the frequency spectrum is divided into several bands. Each band has a specific use, and communications

satellites typically use the C, X, Ku and Ka bands[3]. Table 1.1 shows the uplink and downlink frequency ranges for each band.

Frequency Bands	Uplink (GHz)	Downlink (GHz)
C	5.925 - 6.425	3.7 - 4.2
X	7.9 - 8.4	7.25 - 7.75
Ku	14 - 14.5	10.9 - 12.75
Ka	26.5 - 40	18 - 20

Table 1.1: Frequency Bands and Their Uplink/Downlink Ranges

The signal is then amplified and sent to the antenna, which converts the electrical signal into electromagnetic waves. The antenna dish focuses all the waves into a powerful signal beam directed toward the satellite. Many factors affect the transmission beam, ranging from the thermal effects of transmitters, the interactions of radio waves with the atmosphere, and the thermal effects of the satellite receiver. All of these factors are different sources of radio noise. Radio noise consists of all unwanted contributions whose power is added to the desired carrier wave power, reducing the receiver's ability to correctly reproduce the information content of the received desired carrier wave.

The incoming signal is then amplified with a low-noise amplifier to reduce additional noise, then frequency-converted to the downlink frequency in the specified band, amplified, and broadcast to the receiver.

The Downlink path

There are two main factors that significantly impact the signal during descent. The first is multipath propagation, which occurs due to the nature of electromagnetic waves that reflect, refract, and scatter when interacting with terrestrial surfaces, such as mountains, buildings, and trees. This results in multiple signal paths reaching the receiver antenna, leading to fading, which is considered the superposition of all multipath components at the receiver level. The second factor is noise, introduced into the communication system at the receiving antenna and front-end circuitry. This noise is particularly critical in the downlink, where the desired signal level is typically at its weakest. Figure 1.1 shows a microwave receiver including the antenna and front end [5]. The front end of the receiver consists of a bandpass filter that select the receiving frequency range and a low-noise amplifier that minimize additive noise.

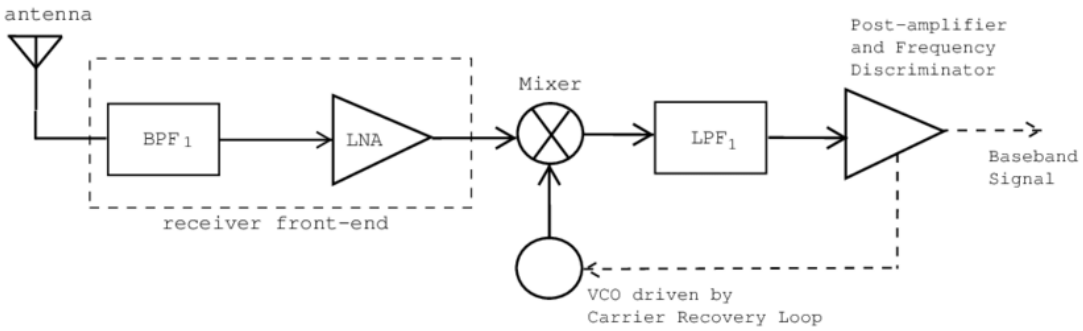

Figure 1.2: Microwave Receiver

1.3 Sources of Noise

The primary sources of noise in satellite communication links include:

- Natural noise originating from atmospheric interactions.

- Thermal noise generated by components within the receiving equipment.

In the following subsections, we examine each of these sources in detail, starting with the atmospheric contributions to noise.

Atmospheric Radio Noise

During radiative transfer, gas molecules such as oxygen and water vapor absorb the energy of radio waves. This process leads to a decrease in the amplitude of the signal (attenuation) and a corresponding increase in the energy of the gas molecules. To release this acquired energy, the molecules undergo random vibrational motion, generating thermal noise (also known as sky noise), the intensity of which depends on the degree of absorption.

Condensed water, in the form of rain or clouds, is a strong absorber of microwave signals and contributes significantly to atmospheric noise. This phenomenon is illustrated in Figure 1.2, which shows the measured sky noise temperature during rainfall in the KA-band signal propagation experiments for the ROCSAT-1 satellite [2].

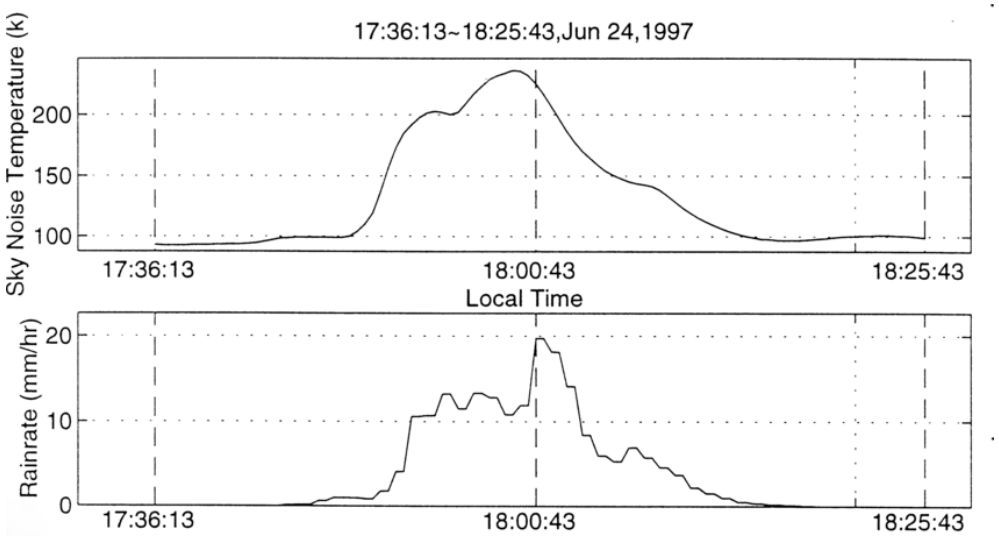

Figure 1.3: Sky noise temperature during a rain fall

The sky noise temperature contribution from atmospheric phenomena (rain, clouds, or clear-air absorption) can be derived from path attenuation using the following physical model:

$$T_n = T_m \left(1 - 10^{-A/10}\right) \quad [\text{K}] \tag{1.1}$$

Where:

- T_n is the equivalent sky noise temperature (K)
- A is the total path attenuation (dB)
- T_m is the effective mean path temperature (K)

The mean path temperature is empirically related to surface conditions by:

$$T_m = 1.12, T_s - 50 \quad [\text{K}] \tag{1.2}$$

Where :

- T_s is the ground surface temperature (K).

Key Implications:

The total noise produced in the channel is expressed by the system noise temperature.

$$T_{sys} = T_n + T_{rec} + T_{ant} \tag{1.3}$$

Where:

- T_{rec} is receiver noise temperature (k).
- T_{ant} is antenna noise (k).

For heavy rain ($A > 10$ dB), T_n dominates T_{sys} as $10^{-A/10} \to 0$.
The multiplier [1.12] in Eq. 1.2 accounts for:

- Atmospheric lapse rate (6.5 K/km)
- Water vapor density gradients

Example Calculation:

For Ts = 300 K (27∘C) and 20 dB rain attenuation:.

$$T_m = 1.12 \times 300 - 50 = 286 \text{ K} \qquad T_n = 286(1 - 10^{-2}) = 277.9 \text{ K}$$

Electronic thermal noise

The major contributor to noise at radio frequencies is thermal noise, which arises from the thermal motion of electrons in the receiver's components—both active and passive. Each component in the receiver system—amplifiers, mixers, upconverters, downconverters, switches, combiners, and multiplexers—generates noise power within the signal bandwidth. This internally generated noise must be accounted for in link performance calculations. The total system noise, therefore, includes both the hardware-induced thermal noise and the noise introduced along the radio wave transmission path by atmospheric conditions [6].

The Noise Power

The noise power N is defined by the **Nyquist formula**:

$$N = T_s \cdot B \cdot k \tag{1.4}$$

Where: - T_s is the source noise temperature in Kelvin $[K]$, - B is the noise bandwidth in Hertz $[Hz]$, - k is **Boltzmann's constant**, equal to:

$$k = 1.38 \times 10^{-23} \, \text{Joules}/K$$

The noise bandwidth B typically corresponds to the **RF bandwidth** of the information-bearing signal, usually defined by the system's filtering characteristics.

The Noise Figure (Active Devices)

Another convenient way to quantify the noise introduced by an amplifier or other active device in a communication system is through its **noise figure (NF)**.

The noise figure is defined as the ratio of the signal-to-noise ratio (SNR) at the input of the device to the SNR at the output:

$$NF = \frac{(S/N)_{\text{in}}}{(S/N)_{\text{out}}} \tag{1.5}$$

Figure 1.4 shows the equivalent noise circuit of an active device.

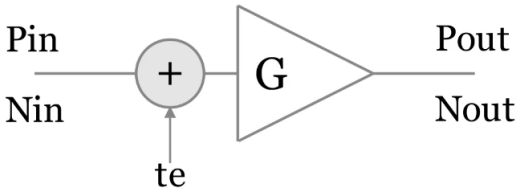

Figure 1.4: Equivalent noise circuit for an active device

Where:

- G is the gain of the device.
- t_e is the equivalent noise temperature added by the device.

The noise figure can be expressed as:

$$NF = \frac{P_{\text{in}} \cdot N_{\text{out}}}{P_{\text{out}} \cdot N_{\text{in}}} \tag{1.6}$$

Substituting the expressions for noise power:

$$NF = \frac{P_{\text{in}} \cdot [G \cdot B_s \cdot (T_s + t_e) \cdot k]}{P_{\text{in}} \cdot [G \cdot B_s \cdot T_s \cdot k]} \tag{1.7}$$

Simplifying:

$$NF = \frac{T_s + t_e}{T_s} = 1 + \frac{t_e}{T_s} \tag{1.8}$$

Where:

- T_s is the input reference temperature, usually taken as 290 K.

For systems composed of multiple devices, the equivalent noise temperature t_e of each device can be derived from the noise figure as:

$$t_e = 290 \cdot (NF - 1) \tag{1.9}$$

If the noise figure is expressed in decibels, the equivalent temperature becomes:

$$NF_{\text{dB}} = 10 \cdot \log_{10}\left(1 + \frac{t_e}{290}\right) \tag{1.10}$$

And equivalently:

$$t_e = 290 \cdot \left(10^{\frac{NF_{\text{dB}}}{10}} - 1\right) \tag{1.11}$$

The Noise Figure (Passive Devices)

Waveguides, cables, filters, and switches are examples of passive components that introduce signal attenuation. These devices also contribute to the overall system noise. Figure 1.5 shows the equivalent circuit of a passive device.

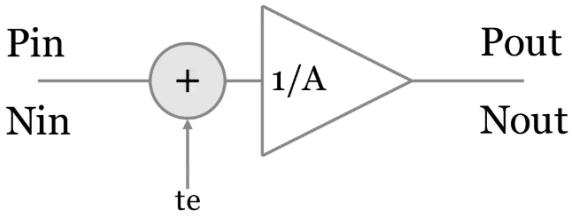

Figure 1.5: Equivalent circuit of a passive device

Let:

- A be the attenuation factor (linear scale),
- P_{in}, P_{out} be the input and output powers, respectively.

The attenuation is defined as:

$$A = \frac{P_{\text{in}}}{P_{\text{out}}} \tag{1.12}$$

The equivalent noise temperature added by a passive device is given by:

$$t_e = 290 \cdot (A - 1) \tag{1.13}$$

If attenuation is expressed in decibels:

$$t_e = 290 \cdot \left(10^{\frac{A_{\text{dB}}}{10}} - 1\right) \tag{1.14}$$

Receiver Antenna Noise

Noise can be introduced into the system at the receiver antenna through two main mechanisms: (1) losses within the physical antenna structure itself, and (2) noise originating from the radio propagation path.

Antenna losses, are absorptive losses caused by components such as the main reflector, subreflector, feed structures, etc. These losses reduce the effective power of the incoming radio wave and are typically expressed as an equivalent noise temperature. The equivalent antenna noise temperature usually ranges from a few to several tens of Kelvin, corresponding to approximately 0.5 to 1 dB of losses.

These losses are generally accounted in the antenna aperture efficiency η_{A}, and therefore do not need to be explicitly included in the system noise temperature calculation.

Noise from the radio path, on the other hand, directly adds to the system noise by increasing the effective antenna temperature. This form of noise—due to atmospheric conditions, cosmic background radiation, and ground reflections—can be a significant factor, especially in satellite communication systems where the received signal strength is often very low. As such, radio path noise at the receiver antenna can be a limiting factor in overall link performance and system design.

System Noise Temperature

The noise contributions of each device in the receiver chain—including atmospheric (sky) noise and hardware noise—combine to form the total **system noise temperature**, which is a key metric in evaluating the overall performance of a communication link.

Figure 1.6 shows a typical satellite receiver system with both active and passive components [14].

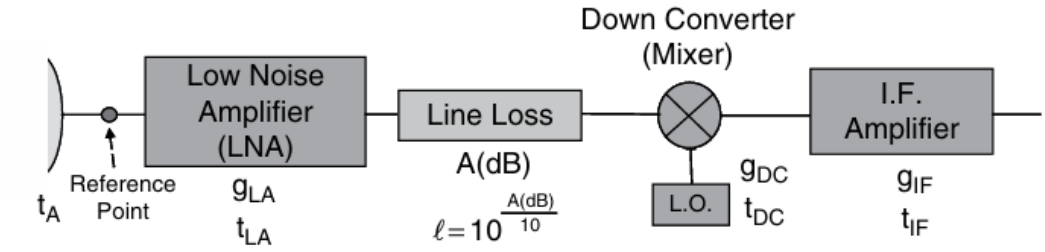

Figure 1.6: Receiver system components

The combined noise from the receiving antenna to all electronic receiving components can be modeled as a single equivalent noise figure NFs, which related to the system noise temperature (t_s).

$$NF_s = 1 + \frac{t_s}{290} \tag{1.15}$$

Where: - t_s is the system noise temperature in Kelvin, and 290 K is the standard reference temperature.

In general, (t_s) is calculated as the sum of the contributions of each noise source. To do this, we replace each component in the receiver with its equivalent noise circuit model. This is illustrated in Figure 1.7, where the reference point is at the leftmost side of the diagram. At this point, all noise sources are referred and combined to calculate the total system noise [22, 21].

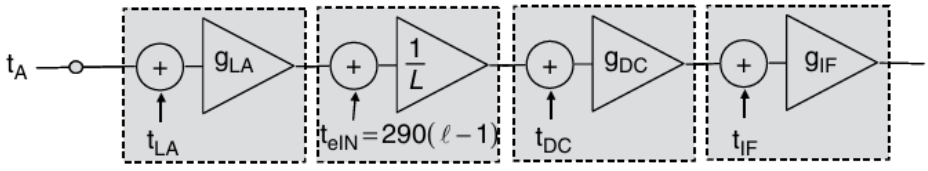

Figure 1.7: Equivalent circuit representation of a receiver system

The contributions are as follows:

- t_a: Antenna noise temperature — directly added. - t_{LA}: Low-Noise Amplifier (LNA) noise temperature — also directly added.

The line loss contribution, t_{eIN}, referred back to the input, is attenuated by the LNA gain, Where :

$$t_{eIN} = \frac{290(A-1)}{g_{LA}} \tag{1.16}$$

The Down Converter noise temperature, referred back to the reference point through the LNA and line loss, contributes:

$$\frac{t_{DC}}{g_{LA} \cdot \frac{1}{A}} = \frac{t_{DC} \cdot A}{g_{LA}} \tag{1.17}$$

The noise temperature of the intermediate Frequency (IF) amplifier is further attenuated by the previous stages, :

$$\frac{t_{IF}}{g_{LA} \cdot \frac{1}{A} \cdot g_{DC}} = \frac{t_{IF} \cdot A}{g_{LA} \cdot g_{DC}} \tag{1.18}$$

Summing all contributions gives the total system noise temperature:

$$t_s = t_a + t_{LA} + \frac{290(A-1)}{g_{LA}} + \frac{t_{DC} \cdot A}{g_{LA}} + \frac{t_{IF} \cdot A}{g_{LA} \cdot g_{DC}} \tag{1.19}$$

This value represents the cumulative noise seen by the system from the antenna up to the IF amplifier. In most practical scenarios, the dominant noise sources are the antenna and LNA; other contributions are often negligible, especially if the LNA has a high gain and is placed close to the antenna.

1.4 The Carrier-To-Noise Ratio

The ratio of the carrier power C to the noise power N, within the same bandwidth, is defined as the **carrier-to-noise ratio** (C/N). This ratio, denoted as $\frac{C}{N}$, considered an important parameter to determines the performance of a satellite communication link.

A higher $\frac{C}{N}$ value generally indicates better connection quality and reliability. Typical communications systems require a minimum $\frac{C}{N}$ value of 5 to 10 dB to achieve acceptable performance.

A lower carrier-to-noise ratio is due either to a decrease in carrier power C or an increase in noise power N. Both factors must be carefully considered when analyzing link budgets and designing the system to ensure satisfactory performance.[3, 14].

The Carrier Power C

The signal carrier in the transmission path is subject to various factors that cause power reduction. The carrier-to-noise ratio is determined by the ratio of carrier power to noise power at the receiver end. To evaluate the received carrier power C, all sources of attenuation along the transmission path must be considered. This is typically done using the **Friis transmission equation**:

$$P_r = P_t \cdot G_t \cdot G_r \cdot \left(\frac{1}{\ell_{FS} \cdot \ell_0} \right) \tag{1.20}$$

Where:

- P_r: received power [W],
- P_t: transmitted power [W],
- G_t, G_r: gains of the transmitting and receiving antennas, respectively (linear scale),
- ℓ_{FS}: free-space path loss,
- ℓ_0: additional loss factor accounting for other effects such as atmospheric attenuation, antenna feed losses, cable losses, etc.

The free-space path loss ℓ_{FS}, as a function of the transmission distance R and the signal wavelength λ, is given by:

$$\ell_{FS} = \left(\frac{4\pi R}{\lambda}\right)^2 \tag{1.21}$$

This formulation allows for accurate estimation of the received carrier power by incorporating both propagation-related and hardware-related losses.

The Noise Power N

As mentioned in the previous section, the noise power at the receiver can be calculated using the system noise temperature t_s through the **Nyquist formula** (1.4).

The carrier-to-noise ratio at the receiver terminal becomes:

$$\frac{C}{N} = \frac{P_t \cdot G_t \cdot G_r}{\ell_{FS} \cdot \ell_0 \cdot k \cdot t_s \cdot B} \tag{1.22}$$

This equation is fundamental for evaluating the performance of satellite communication links. It shows how the received signal quality depends on the transmitter power, antenna gains, propagation losses, system noise, and bandwidth.

1.5 Noise types

A noise signal is defined as a signal that exhibits random fluctuations over time. These fluctuations vary in nature and intensity depending on their source, such as electrical noise or natural sources such as thermal noise resulting from the random motion of electrons. This variation ultimately results in the production of different types of noise.

White Noise

White noise is a random signal that contains equal power across all frequency spectrum , as shown in Figure 1.8. The term "white" is analogous to white light [13], which contains all visible light frequencies.

Perfect white noise is a theoretical concept, requiring unlimited power over an unlimited bandwidth. In practice, additive white Gaussian noise (AWGN) is used to model a wireless communication channel in a particular frequency range. AWGN assumes that noise is linearly additive and has a mean amplitude of zero. Its probability density function (PDF) is given by:

$$P(n) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{n^2}{2\sigma^2}} \tag{1.23}$$

Where σ is the standard deviation of the noise, defined as:

$$\sigma = \sqrt{\frac{1}{N}\sum(x_i - \mu)^2} \tag{1.24}$$

Figure 1.9 shows the Gaussian distribution characteristic of white noise.

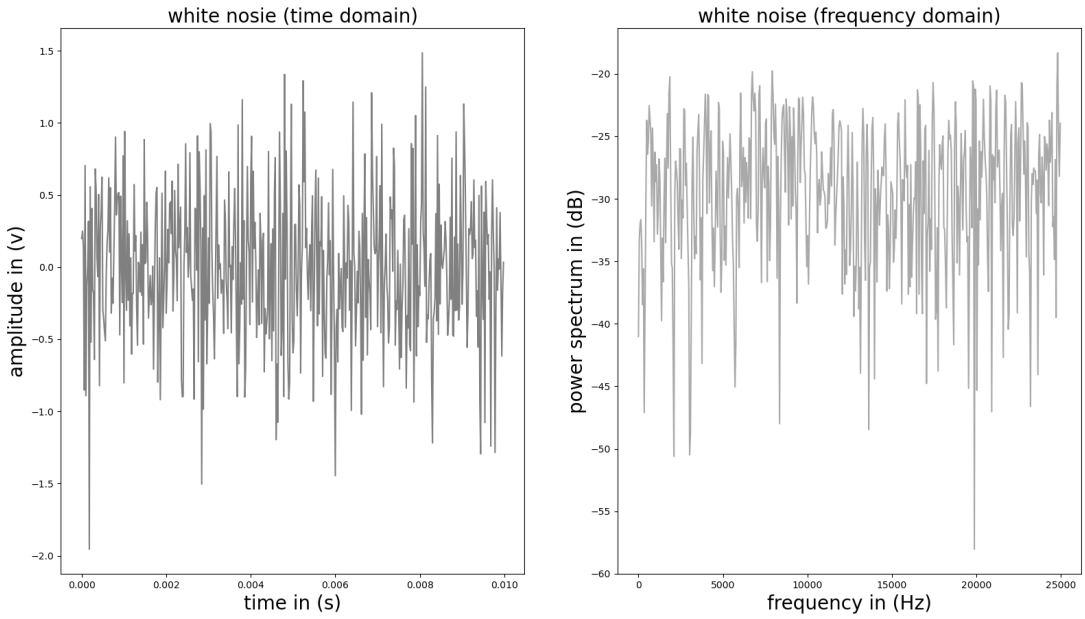

Figure 1.8: White noise in time and frequency domains

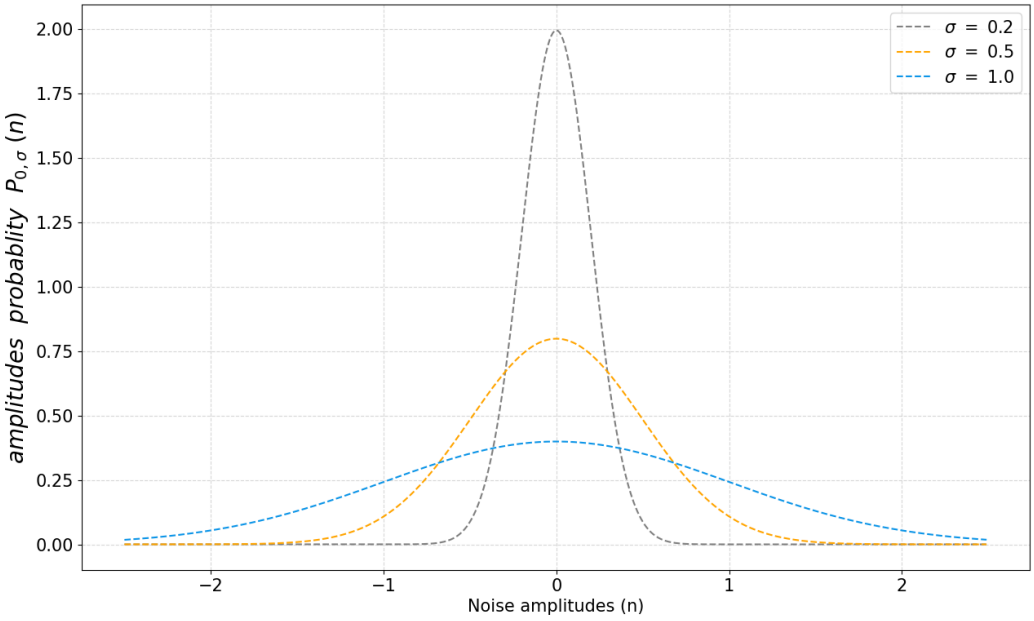

Figure 1.9: Gaussian distribution of white noise

Colored Noise

Pink Noise

Pink noise is a signal whose power spectral density (PSD) is inversely proportional to its frequency, meaning its power decreases by 3 dB for every octave of frequency. Pink noise can be generated by filtering white noise through a low-pass filter, which allows its power to be attenuated at high frequencies by 10 dB per decade, as shown Figure 1.10.

$$\text{PSD} \propto \frac{1}{f}$$

A decade refers to a tenfold increase in frequency (i.e., from 10^d to 10^{d+1}) [7].

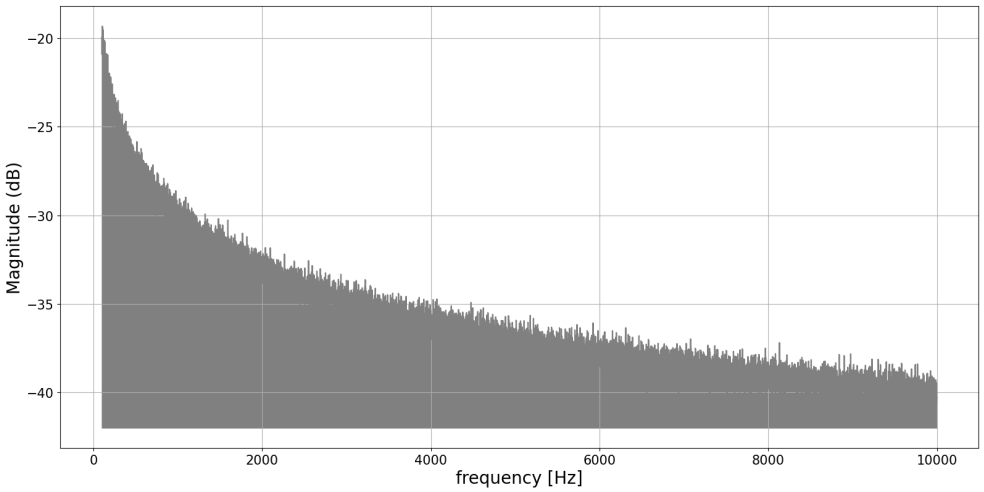

Figure 1.10: Pink noise Power density spectrum

Brownian Noise

Also called **brown noise** or **red noise**[17], Brownian noise refers to a signal whose power spectral density is inversely proportional to the square of the frequency:

$$\text{PSD} \propto \frac{1}{f^2}$$

This equates to a decrease of 20 decibels per decade. Brownian motion refers to the random movement of fluid particles (liquid or gas) due to collisions with the surrounding fluid. See Figure 1.11.

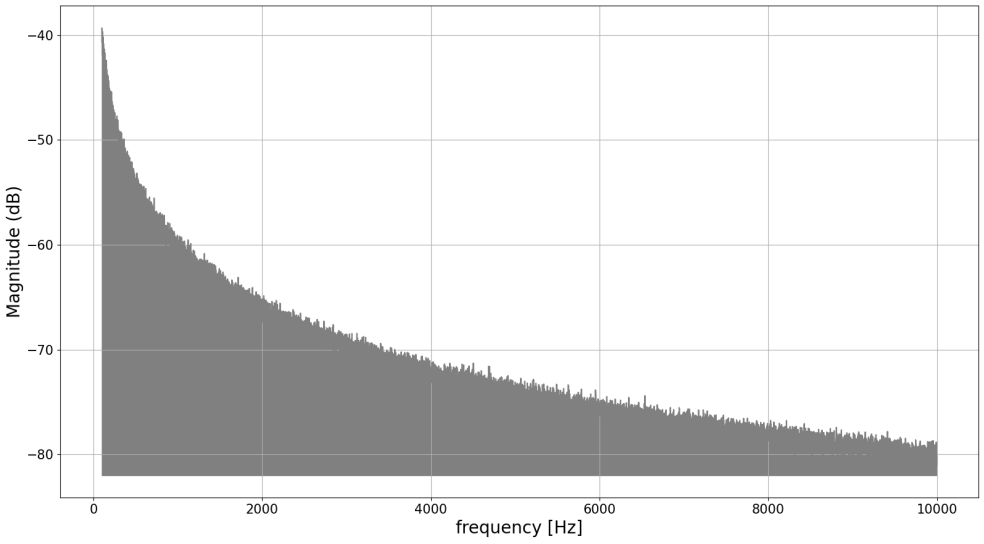

Figure 1.11: Brownian noise power density spectrum

Blue Noise

Blue noise, also known as **azure noise**, has a power spectrum that increases with frequency:

$$\text{PSD} \propto f$$

This results in a 10 dB increase per decade. One source of blue noise is **Cherenkov radiation**, generated when charged particles pass through a dielectric medium at speeds greater than the speed of light in that medium. Blue noise is relevant in high-frequency systems, as shown in Figure 1.12.

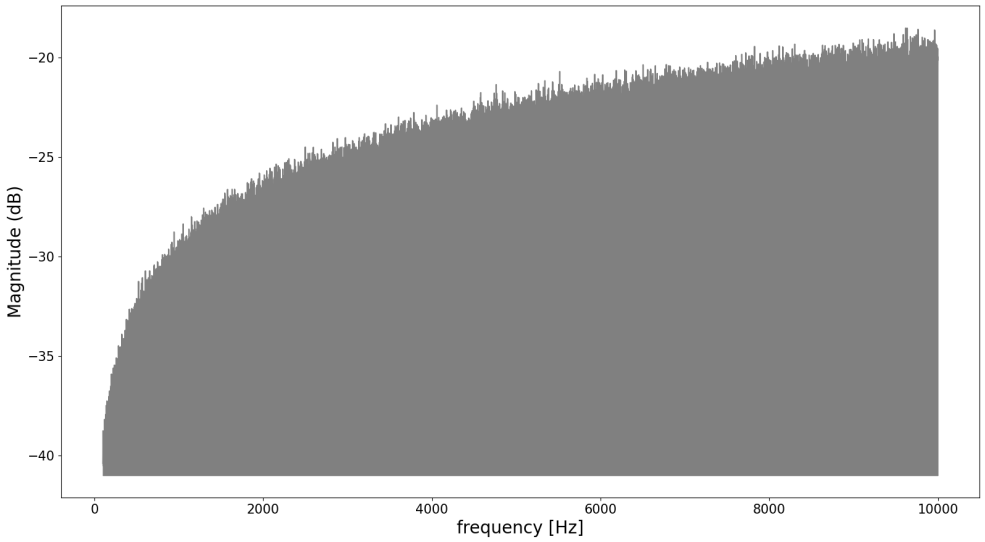

Figure 1.12: Blue noise power density spectrum

Purple Noise

Also known as **violet noise**, purple noise has a power spectrum proportional to f^2, meaning it increases by 20 dB per decade:

$$\text{PSD} \propto f^2$$

It is essentially the derivative of white noise (e.g., generated by high-pass filtering). Purple noise appears in the acceleration errors of GPS systems and other high-frequency signal analyses because its power density increases with frequency [6]. See Figure 1.13.

Relationship Between White and Colored Noise

Just as white light is the source of all visible colors, white noise is the base from which colored noise is derived. By applying filters to white noise, specific frequency components can be enhanced or attenuated, resulting in colored noise like pink, brown, or purple noise[24].

Multiplicative Noise

Multiplicative noise refers to a random signal that multiplies the desired signal, rather than being added. This means the noise amplitude is directly proportional to the signal's instantaneous intensity. It is commonly modeled as **multiplicative white Gaussian noise (MWGN)**, where the noise has a constant spectral power density (as in AWGN) but is multiplied by the signal instead of added. This type of noise can distort signal amplitude non-linearly, making it challenging to remove using standard linear filters.

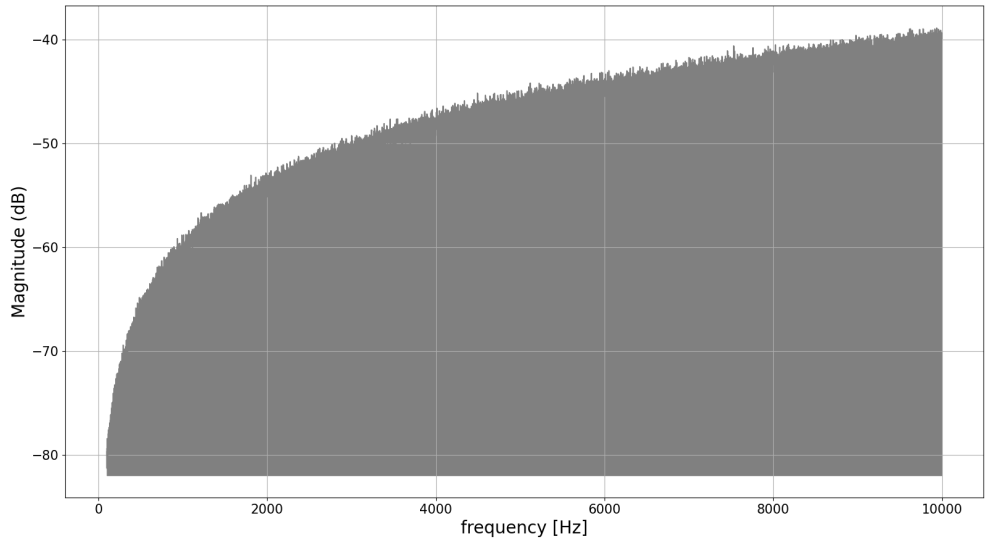

Figure 1.13: Purple noise power density spectrum

Shot Noise

Also called **quantum noise**, it is additive noise resulting from random fluctuations in the number of discrete particles (electrons or photons) in the signal. It becomes significant at low signal power levels and follows a Poisson distribution:

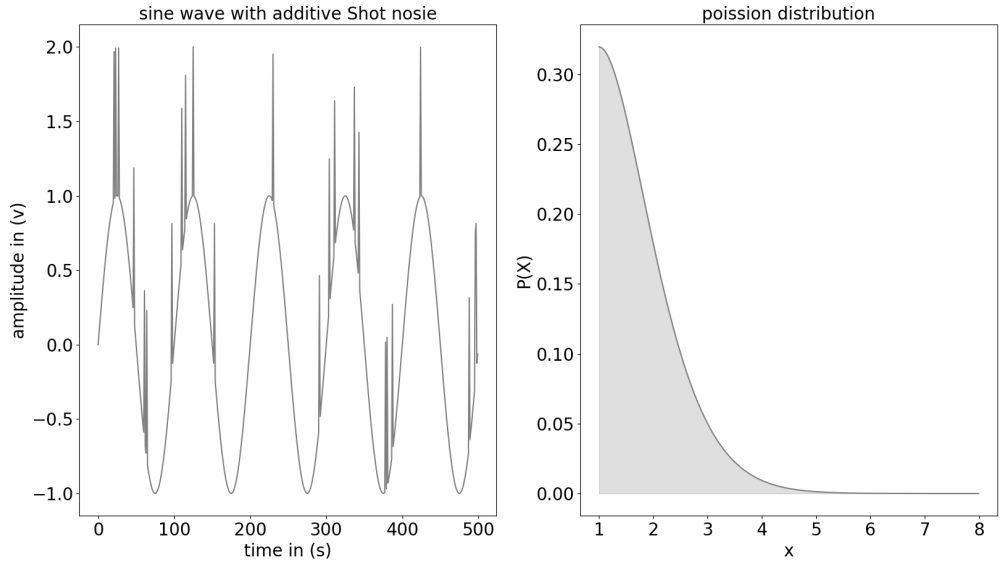

Figure 1.14: Sine wave with additive shot noise and corresponding distribution

$$P(x) = \frac{e^{-\lambda}\lambda^{x}}{x!} \tag{1.25}$$

where λ is the mean and variance of the noise. Figure 1.14 shows a sine wave affected by shot noise and its corresponding distribution.

Transient Noise

Transient noise refers to short-duration bursts or "clicks" caused by sudden interferences or signal corruption during storage or transmission. These typically include: - A sharp initial impulse (due to interference) - Followed by decaying oscillations (due to channel resonance)

This type of noise is commonly observed in digital communication systems and can lead to bit errors.

Phase Noise

Phase noise consists of random fluctuations in the phase of a signal.
It is primarily caused by:

- Thermal and shot noise,
- Pink and flicker noise,
- Hardware imperfections (e.g., oscillator instabilities and crystal resonator defects).

Phase noise is especially critical in systems relying on coherent detection or phase-locked loops, as it affects synchronization and demodulation performance.

1.6 Multipath Channel

Satellite services are characterized by their diversity and differences in capabilities and constraints. While Fixed Satellite Service (FSS) and Broadcast Satellite Service (BSS) links enable fixed stations to be connected via high-gain directional antennas that reduce the effects of local obstacles, enabling a stable and robust connection, this is not the case for Mobile Satellite Service (MSS) links, which rely on mobile stations.

In mobile satellite links, the receiver's movement requires a broadband electromagnetic transmission to ensure communication within a specific local area. Unfortunately, this broadband transmission introduces mechanisms such as "reflection", "diffraction", and "scattering", which can negatively impact the quality of communications links. These mechanisms produce multiple signals propagating along different paths toward the receiver, known as a "multipath channel." Interference between these paths at the receiver can be constructive or destructive, altering the signal's amplitude and phase.

Mechanism Producing Multipath

Reflection

Reflection occurs when a propagating radio wave encounters an object whose dimensions are significantly larger than the transmission wavelength. Typical sources of reflection include buildings, walls, and ground surfaces such as roads, bodies of water, and vegetation. If the reflecting surface is a perfect conductor, no energy is transmitted into the material, and the intensity of the reflected wave equals that of the incident wave[12]. In contrast, if the surface is a dielectric, a portion of the energy is transmitted into the material, thereby reducing the intensity of the reflected wave. In such cases, the receiver captures both the reflected wave and the direct Line-of-Sight (LOS) wave, if present. Figure 1.15 illustrates a reflection scenario involving the Earth's surface, resulting in multipath signal reception by the satellite.

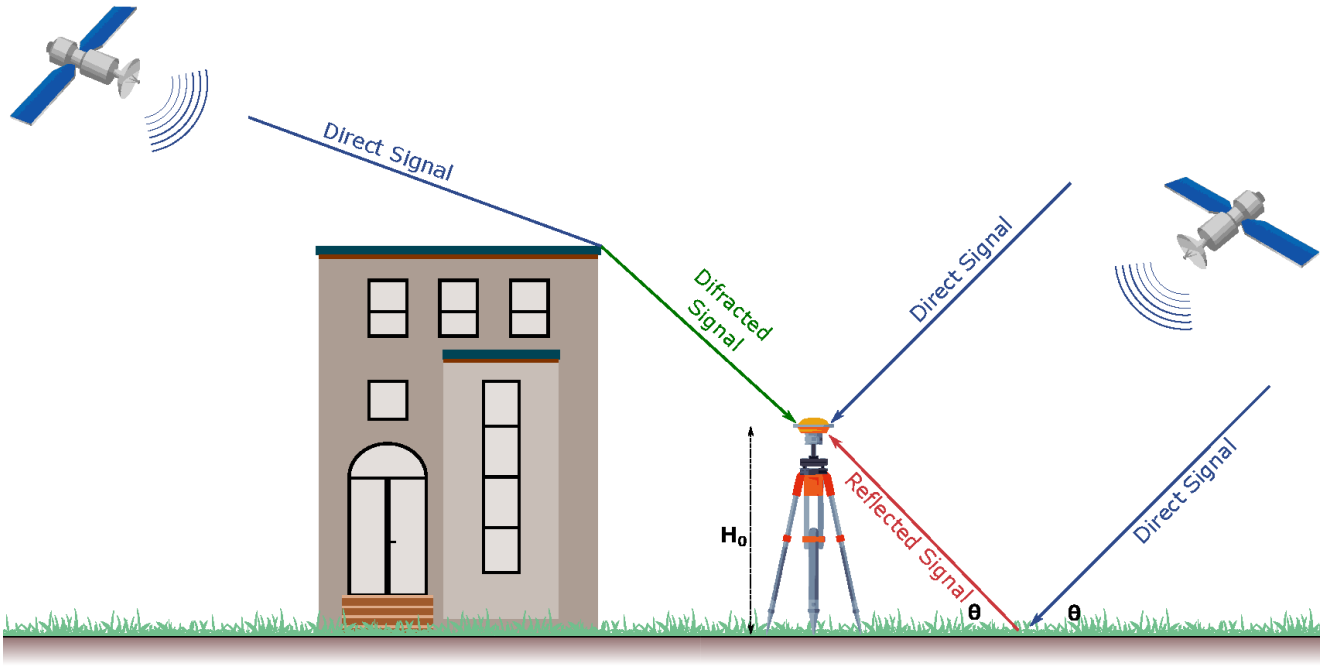

Figure 1.15: Multiple path satellite link

Diffraction

Diffraction occurs when the direct radio path between the transmitter and receiver is obstructed by a surface with sharp edges. In such cases, the wave does not vanish but instead generates secondary waves that propagate into the shadowed region behind the obstacle. This phenomenon gives the appearance that the wave "bends" around the obstacle. Typical sources of diffraction include the edges of tall buildings, towers, and mountain ridges[15].

Scattering

Scattering occurs when the medium through which a wave propagates contains numerous objects whose dimensions are small relative to the wavelength. These objects cause the wave to be redirected in multiple directions. Typical sources of scattering include rough surfaces, small discrete objects, foliage, street signs, and lampposts. The degree to which scattering impacts signal transmission is strongly influenced by the surface roughness and the density of scatterers within the medium.

Multipath Fading

Multipath fading when a transmitted radio signal reaches the receiver through multiple propagation paths, resulting from phenomena such as reflection, diffraction, and scattering in the transmission environment. These multiple paths often differ in length and phase, causing the superimposed signal at the receiver to fluctuate in amplitude and phase over time. This phenomenon, known as multipath fading, can significantly affect the quality and reliability of wireless communication links [4].

In the multipath channel model, the effects of different propagation paths on the transmitted signal $\mathbf{s}(\mathbf{t})$ — particularly in terms of amplitude and phase distortion — can be represented as a convolution with the channel's impulse response $\mathbf{h}(\mathbf{t})$. Thus, the received signal $\mathbf{y}(\mathbf{t})$ is given by:

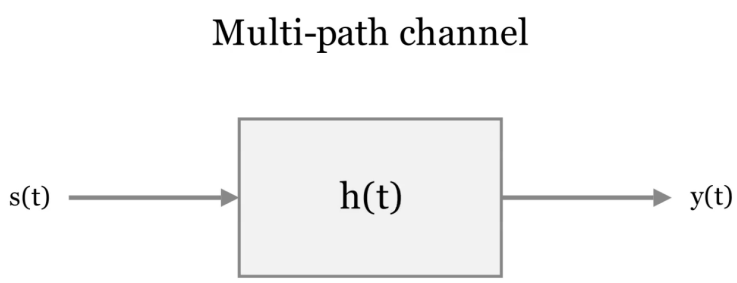

Figure 1.16: Multi-path model

$$y(t) = h(t) * s(t) \tag{1.26}$$

Each path, including the direct Line-of-Sight (LOS) and reflected paths,represent in $h(t)$ as a delayed and attenuated version of the signal. The impulse response can be expressed as:

$$h(t) = \sum_{i=0}^{n-1} \alpha_i \cdot \delta(t - \tau_i) \tag{1.27}$$

Where: - α_i is the complex amplitude (including phase shift) of the i^{th} path, - τ_i is the propagation delay for the i^{th} path, - n is the total number of multipath components.

In wireless communication, the baseband signal $x(t)$ is modulated onto a carrier frequency, yielding the passband transmitted signal:

$$s(t) = \text{Re}\left\{x(t) \cdot e^{jwt}\right\} \tag{1.28}$$

Applying the channel effect, the received signal $y(t)$ becomes:

$$y(t) = \sum_{i=0}^{n-1} \alpha_i \cdot \delta(t - \tau_i) * \text{Re}\left\{x(t) \cdot e^{jwt}\right\} \tag{1.29}$$

$$y(t) = \text{Re}\left\{\sum_{i=0}^{n-1} \alpha_i \cdot x(t - \tau_i) \cdot e^{jw(t-\tau_i)}\right\} \tag{1.30}$$

$$y(t) = \text{Re}\left\{\sum_{i=0}^{n-1} \alpha_i \cdot x(t - \tau_i) \cdot e^{-jw\tau_i} \cdot e^{jwt}\right\} \tag{1.31}$$

The process of demodulating the received carrier wave involves multiplying it by a locally generated carrier wave of the same frequency. This operation yields two components: one centered at twice the carrier frequency (high-frequency term), and another centered at baseband (low-frequency term). By applying a low-pass filter, the high-frequency component is removed, leaving only the baseband signal that contains the transmitted information.

After demodulation and filtering, the received signal can be expressed as:

$$y(t) = \sum_{i=0}^{n-1} \alpha_i \cdot x(t - \tau_i) \cdot e^{-j\omega\tau_i} \tag{1.32}$$

Narrowband Fading

In general, fading is classified as either narrowband (or flatband) or broadband (or frequency-selective). This classification is based on the relationship between the bandwidth of the transmitted signal and the coherence bandwidth of the channel. Narrowband (flatband) fading occurs when the signal bandwidth is smaller than the channel coherence bandwidth. In this case, the delay times of the rays arriving from nearby scatterers are small, due to the small path length differences between the rays. This means that the channel coherence bandwidth is large compared to the signal bandwidth, which is also the case for narrowband (flatband) fading.[18].

$$x(t - \tau) \approx x(t) \tag{1.33}$$

Thus, the received signal becomes:

$$y(t) = x(t) \cdot \sum_{i=0}^{n-1} \alpha_i \cdot e^{-j\omega\tau_i} \tag{1.34}$$

The multipath channel can then be represented by a single complex multiplicative fading coefficient h[19, 20]:

$$y(t) = h \cdot x(t) \qquad \text{where} \quad h = \sum_{i=0}^{n-1} \alpha_i \cdot e^{-j\omega\tau_i} \tag{1.35}$$

The fading coefficient h is a complex number that represents both the amplitude attenuation and the phase shift caused by the channel. The magnitude $\mid h \mid$ indicates the level of constructive or destructive interference between the incoming signals and it's depends on the phase shift caused by the channel. This phase is caused by a difference in path length (even a small one) relative to the carrier wavelength, and is typically a random variable evenly distributed between $-\pi, \pi$. The magnitude h (the intensity of the fading) is determined by the random phases of all converging multipath rays at the receiver. A small motion (such as $\lambda/4$ can flip the phase, converting constructive interference to destructive interference, and causing deep fading.

Due to variations in the radio propagation environment, the fading coefficient $|h|$ changes over time. This variation is commonly modeled using the **Rayleigh distribution**, especially when there is no dominant line-of-sight (LOS) path. The probability density function (PDF) of the Rayleigh distribution is given by [12, 15]:

$$P(|h|; \sigma) = \frac{|h|}{\sigma^2} \cdot \exp\left(-\frac{|h|^2}{2\sigma^2}\right) \tag{1.36}$$

Here, σ is the scale parameter, which determines the spread of the distribution. In Figure 1.17, the case with $\sigma = \frac{1}{2}$ shows that the maximum probability occurs when $|h| \in [0, 1]$, indicating that low values of the fading coefficient are more probable. This highlights the destructive nature of narrowband fading in many scenarios.

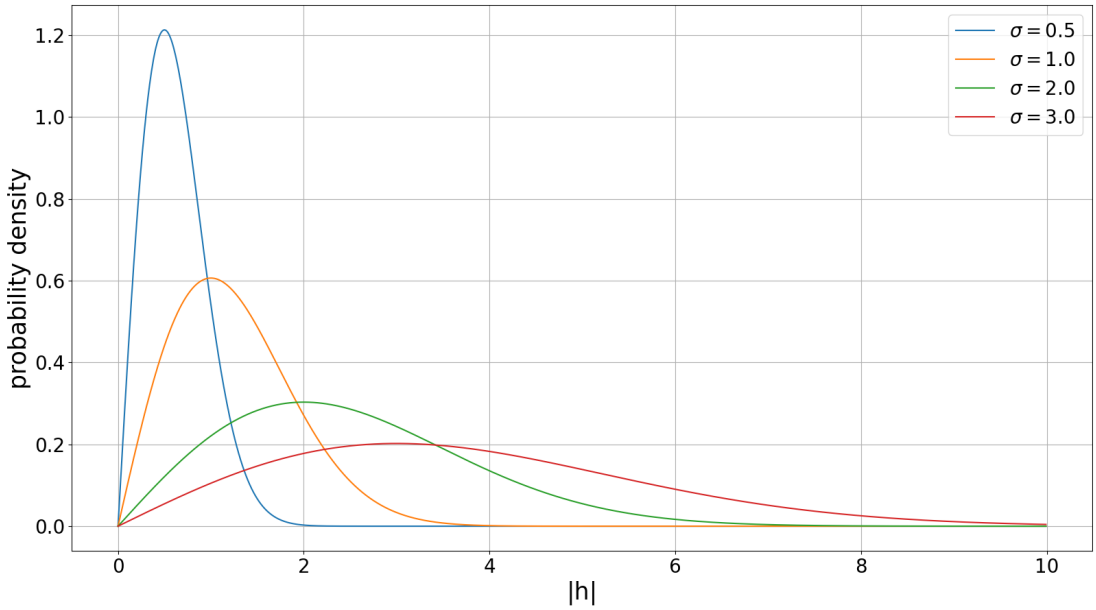

Figure 1.17: Rayleigh probability distribution

Figure 1.18 shows how the fading coefficient $|h|$ varies over time, further illustrating the dynamic nature of the fading process.

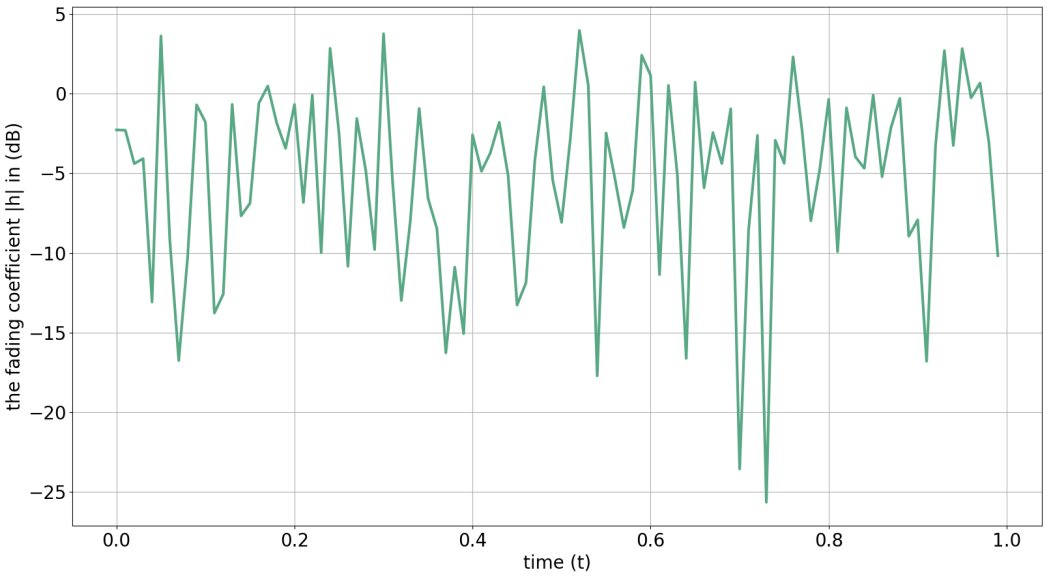

Figure 1.18: Fading coefficient variation over time

Wideband Fading

Wideband fading occurs when the bandwidth of the transmitted signal is greater than the bandwidth of the coherent channel. This can be considered a situation where the dispersed signals are far from the direct path between the satellite transmitter and the mobile receiver, resulting

in multiple copies of the transmitted signal arriving at the receiver with significantly different delays, which meaning a large phase shift, causing an inter-symbol interference (ISI). [19, 20].

Figure 1.19 illustrates this phenomenon, showing how multiple signals can reach a mobile receiver via distinct paths. Each received signal component has an associated amplitude $\mathbf{a_i}$, which depends on the specific path characteristics, and a delay $\tau_\mathbf{i}$, which represents the propagation time from the transmitter to the receiver through that path.

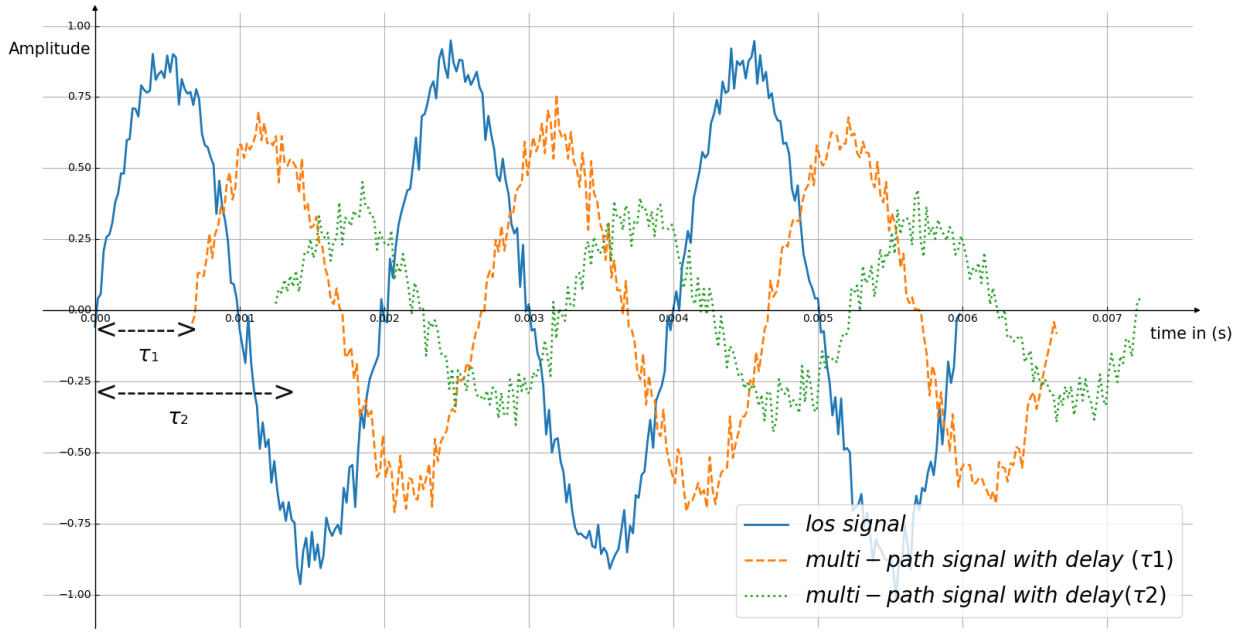

Figure 1.19: Signals arriving via multiple paths in a wideband fading channel

The **delay spread** $\tau_{\mathbf{DS}}$ is defined as the difference between the arrival time of the earliest component (typically the Line-of-Sight signal) and the latest arriving component. When the delay spread τ_{DS} is large compared to the symbol or bit duration of the transmitted signal, this causes **signal dispersion** in the time domain and introduces **inter-symbol interference (ISI)**.

ISI leads to overlapping symbols at the receiver, which results in signal distortion and degraded performance, especially in high-data-rate communication systems.

Furthermore, the signal components arriving later generally experience **greater attenuation**, as they travel through longer or more obstructed paths. As a result, the power of delayed signals tends to decrease with increasing delay, further complicating equalization and detection at the receiver.

Wideband fading is therefore a critical consideration in satellite and mobile communication system design, and it often necessitates the use of equalizers, RAKE receivers, or multicarrier modulation techniques like OFDM to mitigate its impact.

1.7 Conclusion

As demonstrated throughout this chapter, the effects of noise and channel impairments in communication systems are both diverse and significant. These disturbances originate from various sources, such as thermal fluctuations, multipath propagation, scattering, and Doppler shifts, each introducing unique challenges to signal integrity. Furthermore, these effects can vary in intensity, temporal behavior, and spectral characteristics depending on environmental conditions and system parameters.

This variability makes the communication channel one of the most complex and unpredictable components of a wireless system. Understanding the nature and impact of different noise types — including thermal noise, colored noise (e.g., pink, brown, blue, and purple), and multipath-induced fading — is essential for accurate channel modeling and effective system design.

In practice, these impairments can lead to signal degradation, reduced data rates, increased error probabilities, and loss of synchronization, all of which compromise the reliability and efficiency of communication. Therefore, developing robust mitigation techniques, such as adaptive filtering, diversity schemes, error correction coding, and equalization, is critical to ensuring system performance under such adverse conditions.

In conclusion, the study of noise and channel effects is foundational to the field of communications. It highlights the necessity for adaptive, intelligent systems capable of dynamically adjusting to channel variations in order to maintain reliable and high-quality transmission. This understanding serves as a critical basis for the design of advanced communication architectures and protocols addressed in the subsequent chapters.

2 Deep in Theory

2.1 Introduction

As we saw in the previous chapter, noise and fading are major problems in satellite communications systems. To address them, numerous techniques have been developed, including advanced digital signal processing techniques.However, given the varying impact of noise on the signal resulting from different types and intensity, as well as the attenuation and phase shifts caused by fading, applying signal processing techniques to preserve the original signal become a real challenge and requires a high level of expertise to develop a system that meets the minimum requirements. In this context we present the "Adaptive pulses systems" as an artificial intelligence algorithm for learn and developing processing systems able to adverse different levels of channel influences.

2.2 Adaptive pulses systems

In signal processing, digital filters are an essential component that allows for removing or enhancing certain parts of a signal, extracting or restoring its properties, and correcting them. However, developing digital filters based on specific mathematical functions limits their capabilities in certain situations, as it can be difficult to develop a digital filter system capable of handling multiple and complex tasks. Therefore, we develop digital filters using the "adaptive pulses systems" algorithm, which enables digital filters to learn main targets from the data, while ensuring that all capabilities are exploited and taken into account to achieve the desired goal, which is in our case, processing a communication signal from noise and multipath effects (fading).

2.3 Digital carrier modulation

In wireless communications, digital information must be modulated into a sine wave carrier wave, using ones of digital modulation technique, for can be transmitted over the external medium. The modulated carrier wave is then the part in the "transmission chain" that subjects to the external medium effects from noise and fading, which make the processing or reducing of channel effects in the wireless communication signal a processing process into the arriving modulated carrier. So from this point our object represent as developing an "Adaptive pulses systems" model geared toward processing modulated carrier from channel effects. But first we will see more in this section about different digital modulation techniques.

M-ary Phase Shift Keying

M-ary phase shift keying is one of the most widely used modulation techniques in satellite communication links [3]. It involves encoding a binary (or m-ary) bit stream by inducing a phase shift of the sinusoidal carrier wave.

Binary phase shift keying (BPSK)

The Binary phase shifting keying technique encodes two states of information (1 and 0) with shifting the phase of the carrier by predetermined phase (180 deg) for represent the change in information state. Consider $c(t)$ as the sinusoidal carrier formula.

$$c(t) = A \cdot \sin\left(2\pi f_c t + \phi\right) \tag{2.1}$$

Where A are the amplitude of the carrier, t is the time in second (s), f_c the frequency of the carrier in (Hz), and ϕ are the phase in (rad).
The shifting of phase by (180°) from a primary (0° or 180°) phase will produce two signal of opposite sign and the same amplitude,

$$c(t) = A \cdot \sin\left(2\pi f_c t + 0\right) \qquad s_1(t) = A \cdot \sin\left(2\pi f_c t\right) \tag{2.2}$$

$$c(t) = A \cdot \sin\left(2\pi f_c t + \pi\right) \qquad s_2(t) = -A \cdot \sin\left(2\pi f_c t\right) \tag{2.3}$$

The Binary Phase Shift Keying (BPSK) constellation diagram represents two points corresponding to the two possible phase shifts (0 and 180 degrees). These points are located along the I-axis (in-phase axis), with one point on the positive side (representing the first signal phase) and the other on the negative side (representing the second signal phase). The Q-axis (quadrature axis) represents the imaginary part of the complex plane, where there is no Q component in BPSK modulation. Figure 2.1 illustrates the BPSK modulation constellation diagram.

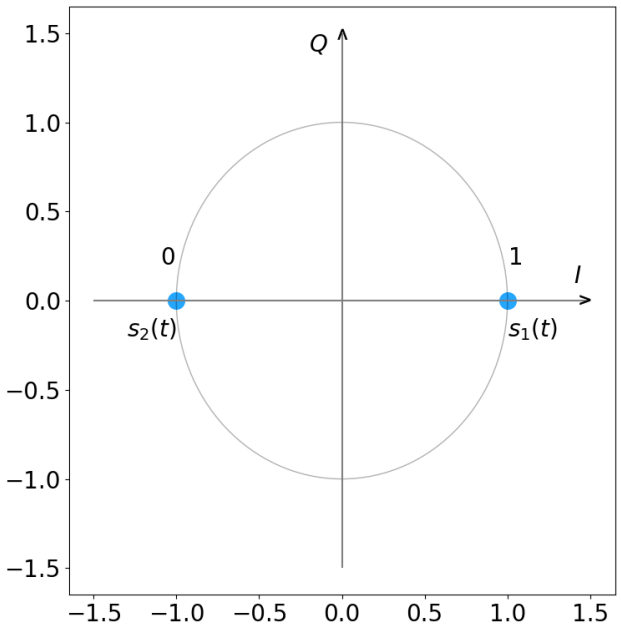

Figure 2.1: BPSK constellation diagram

Non-return-to-zero (NRZ) coding is widely preferred for binary phase shift key modulation because of its ability to modulate digital information directly on the carrier wave by combining the NRZ signal with a sinusoidal carrier wave.

Consider p(t) as the NRZ representation of digital information. Where:

$$p(t) = +1V \quad \text{When the digital state of bit take one (1)} \tag{2.4}$$

$$p(t) = -1V \quad \text{When the digital state of bit take zero (0)} \tag{2.5}$$

Figure 2.2 illustrates a non-return-to-zero (NRZ) encoded signal for an incoming bit stream.

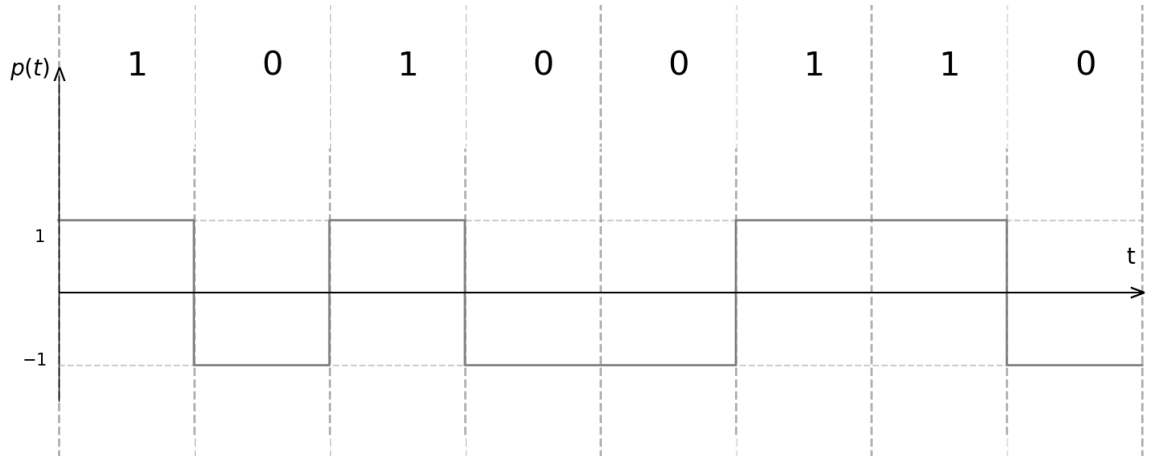

Figure 2.2: NRZ signal for an incoming bit stream

Combining an NRZ signal $p(t)$ with a sine wave carrier $c(t)$ directly results in a BPSK-modulated waveform $s(t)$. A phase shift (180°) enables the digital states (1 and 0) to be represented by two different signals$s_1(t)$ *and* $s_2(t)$. This can be embodied by combining a positive NRZ signal (representing bit 1) with the carrier wave to produce $s_1(t)$, while combining a negative NRZ signal (representing bit 0) with c(t) produces $s_2(t)$. Figure 2.3 shows a "binary phase shift key modulator" consisting of an NRZ encoder to encode the digital bits into an NRZ signal and an analog multiplier that combines the NRZ signal with the carrier signal. Then the binary phase

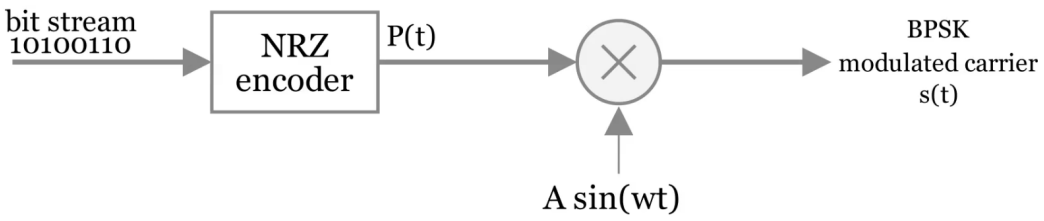

Figure 2.3: Binary phase shift keying modulator

shift keying modulated carrier $s(t)$ equal to.

$$s(t) = p(t) \cdot A \cdot \sin(2\pi f_c t) \tag{2.6}$$

When $p(t) = 1$ then the modulate carrier $s(t)$ equal

$$s(t) = s_1(t) = A \cdot \sin(2\pi f_c t) \tag{2.7}$$

When $p(t) = -1$ then the modulate carrier $s(t)$ equal

$$s(t) = s_2(t) = -A \cdot \sin(2\pi f_c t) \tag{2.8}$$

Figure 2.4 represent the combination process of NRZ signal with pure carrier wave $c(t)$ over time which result of the BPSK-modulated carrier.

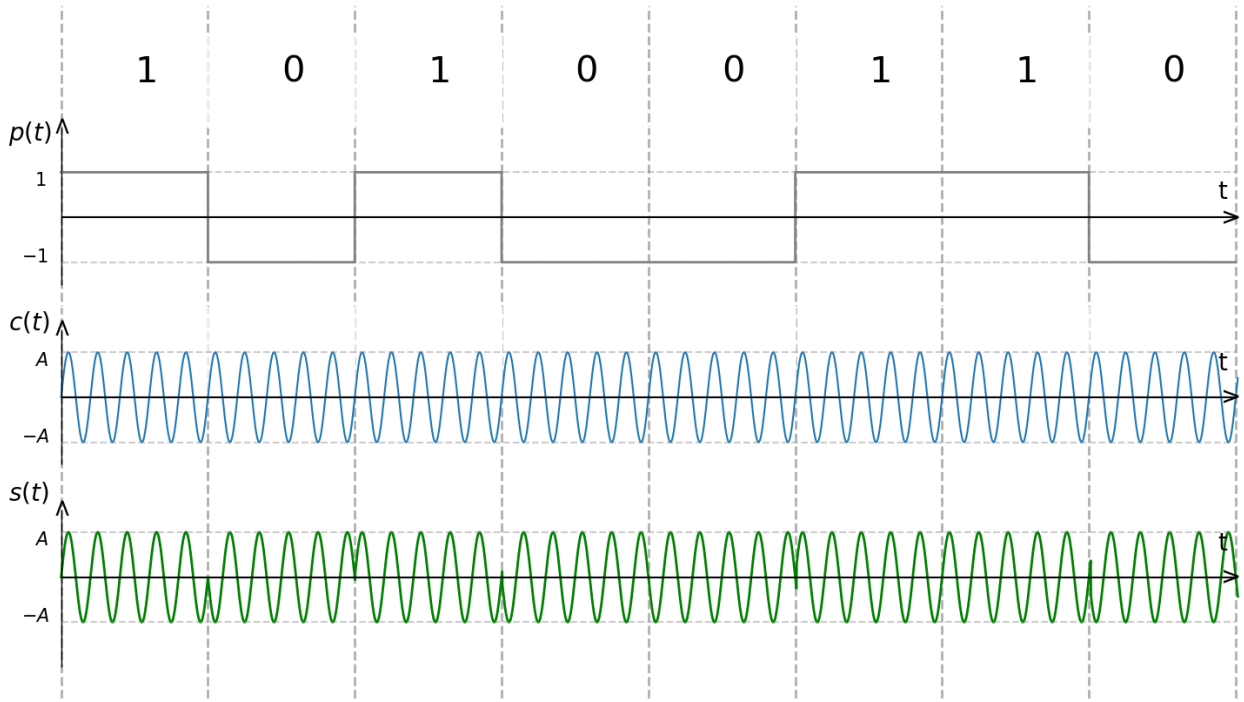

Figure 2.4: Binary phase shift key modulation over time

Quadrature phase shift keying (QPSK)

Binary phase shift keying modulation, as discussed above, transmits one bit per time period over a large bandwidth (compared to high-order phase shift modulation techniques), which does not meet the growing demand for high transmission rates and effective bandwidth utilization. With given available capacity, it is preferable to use more complex modulation schemes that provide high transmission rates and utilize the bandwidth efficiently. Quadrature phase shift keying, on the other hand, is phase shift key modulation scheme encode two bits per time period, twice the data rate of BPSK over the same bandwidth.

Representing two-bit over time period with two different states, that means 2^2 symbols that needs to be represented by a carrier phase shifting, which requires four predetermined phase angle for each symbol. Generally take $\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}$ as the carrier shifting phases. Consider $c(t)$ the carrier sinusoidal formula.

$$c(t) = A\cos\left(\omega_c t + \phi\right) \tag{2.9}$$

With ϕ take the predefined -phase shift signal, which is represent by ϕ_n

$$\phi_n = \frac{\pi}{4}(2n-1) = \frac{n\pi}{2} - \frac{\pi}{4} \tag{2.10}$$

With compensation ϕ_n in $c(t)$ we obtain.

$$\begin{aligned} c(t) &= A\cos\left(\omega_c t + \phi_n\right) && (2.11)\\ &= A[\cos(\omega_c t)\cos\left(\phi_n\right) - \sin\left(\omega_c t\right)\sin\left(\phi_n\right)] && (2.12)\\ &= A\left[\cos(\omega_c t)\cos\left(\frac{n\pi}{2} - \frac{\pi}{4}\right) - \sin\left(\omega_c t\right)\sin\left(\frac{n\pi}{2} - \frac{\pi}{4}\right)\right] && (2.13) \end{aligned}$$

$$s(t) = A[\cos(\omega_c t)\cdot I_n - \sin\left(\omega_c t\right)\cdot Q_n] \tag{2.14}$$

Where s(t) is the modulated carrier with I_n and Q_n respectively representing the In-phase and quadrature components that express the amplitude and phase of the fully modulated carrier in the complex plan.

With simplifying the I_n expression we get.

$$I_n = \cos\left(\frac{n\pi}{2} - \frac{\pi}{4}\right) \tag{2.15}$$
$$= \cos\left(\frac{n\pi}{2}\right)\cos\left(\frac{\pi}{4}\right) + \sin\left(\frac{n\pi}{2}\right)\sin\left(\frac{\pi}{4}\right) \tag{2.16}$$
$$= \frac{1}{\sqrt{2}}\left[\cos\left(\frac{n\pi}{2}\right) + \sin\left(\frac{n\pi}{2}\right)\right] \tag{2.17}$$

In the same way, the Quadrature component of the modulated signal becomes.

$$Q_n = \sin\left(\frac{n\pi}{2} - \frac{\pi}{4}\right) \tag{2.18}$$
$$= \sin\left(\frac{n\pi}{2}\right)\cos\left(\frac{\pi}{4}\right) - \cos\left(\frac{n\pi}{2}\right)\sin\left(\frac{\pi}{4}\right) \tag{2.19}$$
$$= \frac{1}{\sqrt{2}}\left[\sin\left(\frac{n\pi}{2}\right) - \cos\left(\frac{n\pi}{2}\right)\right] \tag{2.20}$$

By substituting different values of n into I_n and Q_n, we can obtain the constellation diagram components that expresses the amplitude and phase of the four signals, represent each two-bit symbol, as well as the digital state of the two bits. Table 2.1 show I_n and Q_n with corresponding phase and symbol.

n	I_n	Q_n	ϕ_n	Digital State
1	0.7071	0.7071	$\pi/4$	11
2	-0.7071	0.7071	$3\pi/4$	01
3	-0.7071	-0.7071	$5\pi/4$	00
4	0.7071	-0.7071	$7\pi/4$	10

Table 2.1: QPSK [I_n and Q_n] components with corresponding phase and symbol

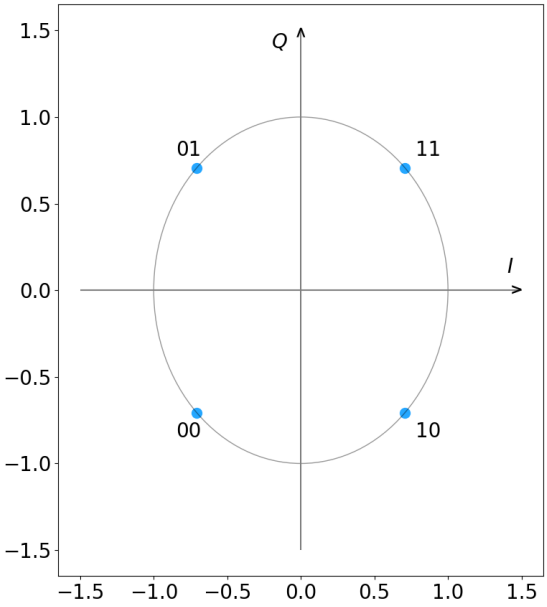

Figure 2.5: QPSK constellation diagram

As we can see, in a quadrature phase shift key modulation, the digital symbols are distributed over the four signals with following the I_n and Q_n component sign.

- When $I_n > 0$ the Most Significant Bit (MSB) take the digital state '1'.
- When $I_n < 0$ the Most Significant Bit take the digital state '0'.
- When $Q_n > 0$ the Least Significant Bit (LSB) take the digital state '1'.
- When $Q_n < 0$ the Least Significant Bit take the digital state '0'.

The sinusoidal carrier $c(t)$ for each time period varying in phase for represent incoming digital symbols which produce modulated carrier $s(t)$. Figure 2.6 represent the digital modulation of different symbols with sinusoidal carrier.

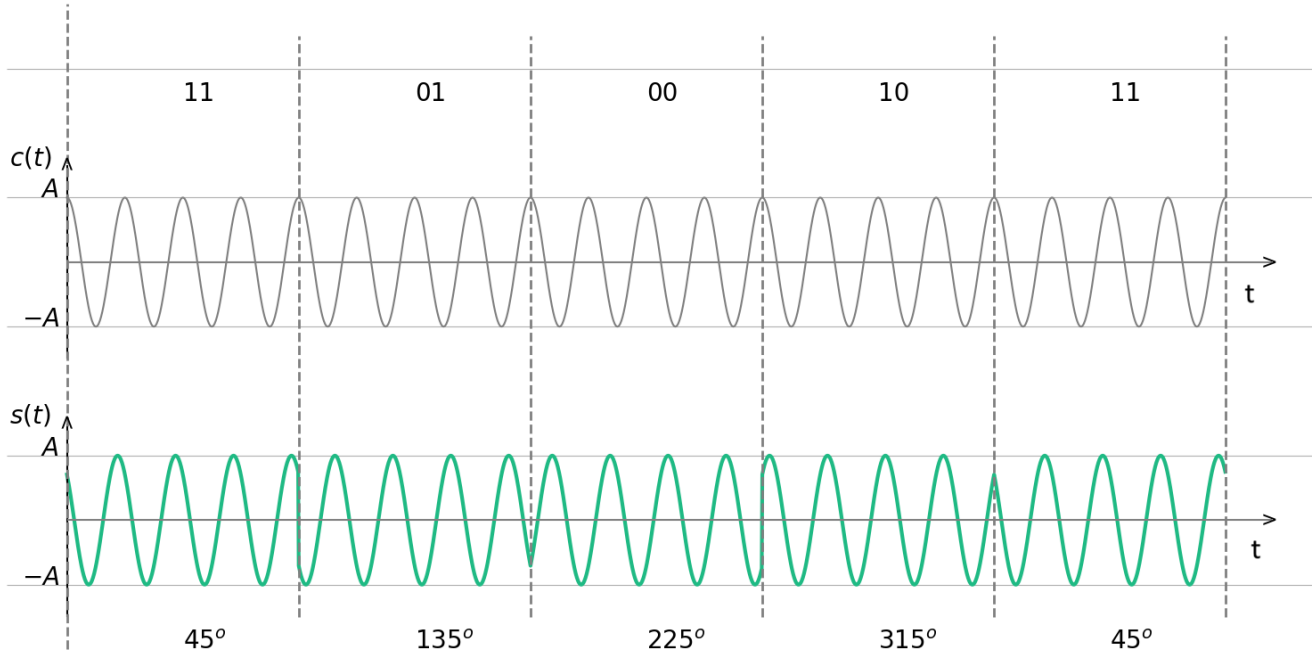

Figure 2.6: QPSK modulated carrier over time

Octal Phase Shift Keying (8PSK)

Octal phase shift keying (8PSK) is a phase shift key modulation scheme that encodes three bits of information into a carrier wave using eight different phase angles of the carrier signal. Consider $c(t)$ the sinusoidal carrier formula where.

$$c(t) = A\cos(\omega_c t + \phi) \tag{2.21}$$

With ϕ take eight predefined phase signal representing by ϕ_n.

$$\phi_n = (n-1)\frac{\pi}{4} = \frac{n\pi}{4} - \frac{\pi}{4} \tag{2.22}$$

With compensation ϕ_n in $c(t)$ we obtain.

$$\begin{aligned} c(t) &= A\cos(\omega_c t + \phi_n) && (2.23)\\ &= A[\cos(\omega_c t)\cos(\phi_n) - \sin(\omega_c t)\sin(\phi_n)] && (2.24)\\ &= A\left[\cos(\omega_c t)\cos\left(\frac{n\pi}{4} - \frac{\pi}{4}\right) - \sin(\omega_c t)\sin\left(\frac{n\pi}{4} - \frac{\pi}{4}\right)\right] && (2.25) \end{aligned}$$

$$s(t) = A[\cos(\omega_c t)\cdot I_n - \sin(\omega_c t)\cdot Q_n] \tag{2.26}$$

Where $s(t)$ are the modulated carrier with I_n and Q_n as the In-phase and the Quadrature components of the carrier signal in the complex plane.

With simplifying the In expression we get.

$$I_n = \cos\left(\frac{n\pi}{4} - \frac{\pi}{4}\right) \tag{2.27}$$
$$= \cos\left(\frac{n\pi}{4}\right)\cos\left(\frac{\pi}{4}\right) + \sin\left(\frac{n\pi}{4}\right)\sin\left(\frac{\pi}{4}\right) \tag{2.28}$$
$$= \frac{1}{\sqrt{2}}\left[\cos\left(\frac{n\pi}{4}\right) + \sin\left(\frac{n\pi}{4}\right)\right] \tag{2.29}$$

In the same way, the Quadrature component of the modulated signal is equal to.

$$Q_n = \sin\left(\frac{n\pi}{4} - \frac{\pi}{4}\right) \tag{2.30}$$
$$= \sin\left(\frac{n\pi}{4}\right)\cos\left(\frac{\pi}{4}\right) - \cos\left(\frac{n\pi}{4}\right)\sin\left(\frac{\pi}{4}\right) \tag{2.31}$$
$$= \frac{1}{\sqrt{2}}\left[\sin\left(\frac{n\pi}{4}\right) - \cos\left(\frac{n\pi}{4}\right)\right] \tag{2.32}$$

By replacing n with the ordered integers from one to eight, we obtain the components of the complex plane I_n and Q_n, and his corresponding phases.

n	I_n	Q_n	ϕ_n	Symbols
1	1	0	0	000
2	0.7071	0.7071	$\pi/4$	001
3	0	1	$\pi/2$	011
4	-0.7071	0.7071	$3\pi/4$	010
5	-1	0	π	110
6	-0.7071	-0.7071	$5\pi/4$	111
7	0	-1	$3\pi/2$	101
8	0.7071	-0.7071	$7\pi/4$	100

Table 2.2: Components of 8PSK constellation diagram with corresponding phases and symbols

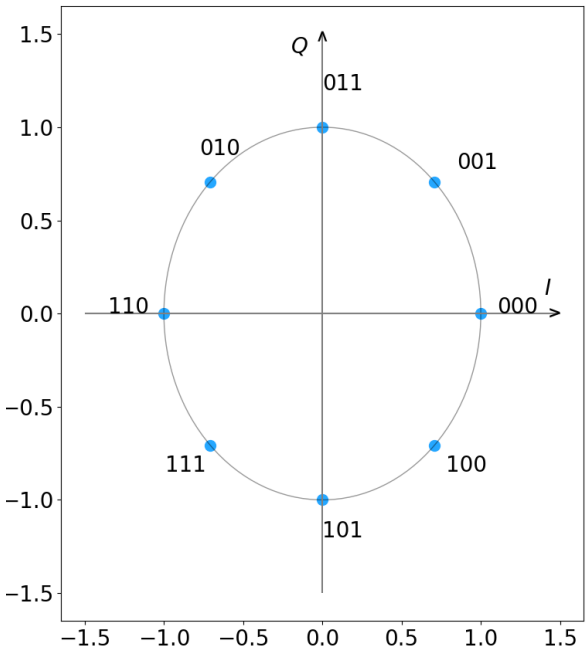

Figure 2.7: 8PSK constellation diagram

The distribution of symbols across the eight transition stages typically follows the Gray code method. This method helps reduce the bit error rate (BER) at reception, as the distance between adjacent symbols is short (about one bit), meaning that a classification error between two adjacent symbols results in a single bit error. Figure 2.7 shows an 8PSK constellation diagram with equivalent symbols, where we can see that the difference between two adjacent symbols is one bit.

Octal phase shift keying is a higher-order modulation scheme that achieves high data transfer rate and greater bandwidth efficiency. However, it is more affected by noise and interference, compared to QPSK or BPSK under the same channel conditions.

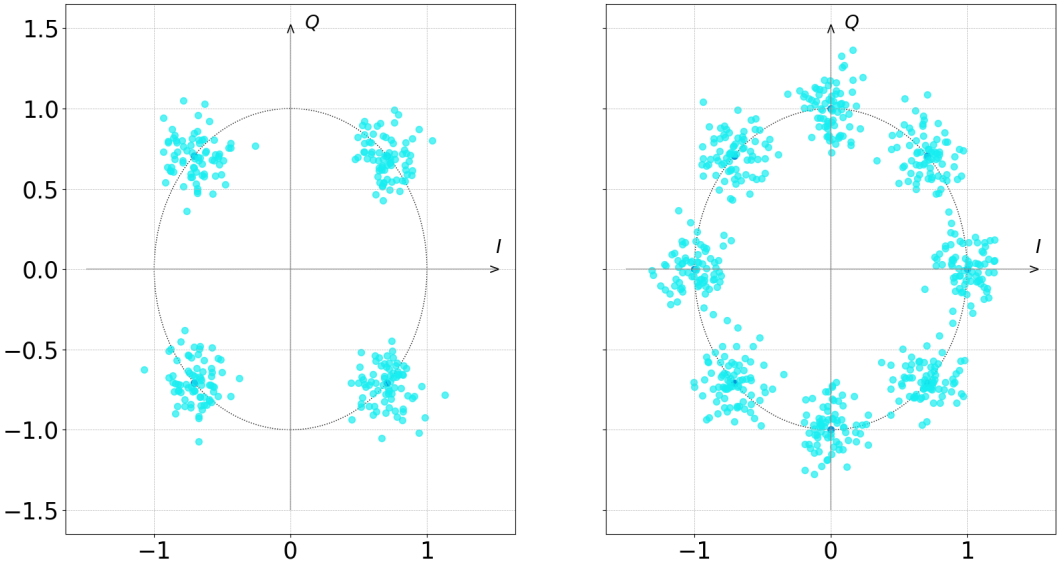

Figure 2.8: (a) QPSK constellation diagram (b) 8PSK constellation diagram

Figure 2.8 represents a comparison between (a) QPSK constellation diagram and (b) 8PSK constellation diagram over AWGN channel with 15 dB carrier-to-noise ratio[11]. Where in (a) different symbols are separate and distinguishable from each other, what is not happen in (b) the different symbols are very close and become more difficult to distinguish.

16-Quadrature Amplitude Modulation

16-Quadrature Amplitude Modulation (16-QAM) is a high-order digital modulation technique that transmits four-bit symbols per time period, allowing for high data rates and efficient bandwidth utilization. 16-QAM combines both amplitude and phase shifts in the carrier wave simultaneously to represent sixteen possible symbols [9].

All M-ary quadrature amplitude modulation (M-QAM) schemes use two orthogonal signals to carry information. Two carrier waves, typically sine and cosine waves, of the same frequency but 90 degrees out of phase (orthogonal), are multiplied by a specific component $I(t)$ and $Q(t)$ to convey a discrete symbol state. These two waves are then linearly summed to form the modulated carrier wave $s(t)$ [9].

$$s(t) = \cos\left(2\pi f_c \cdot t\right) \cdot I(t) - \sin\left(2\pi f_c \cdot t\right) \cdot Q(t) \tag{2.33}$$

The block diagram shown in Figure 2.9 illustrates the process of modulating a digital input signal into a 16-QAM modulated carrier. The serial-in, parallel-out shift register (SIPO) shifts 4 bits per time cycle, allowing the input data to be grouped into a 4-bit binary word (symbol). Each symbol must then be represented by its corresponding constellation components (I and Q). Therefore, for each component, a digital-to-analog converter (DAC) takes two bits and converts

them into a period time unit pulse with an amplitude corresponding to the component value in the constellation diagram. Table 2.3 and Table 2.4 show 2-bit symbols with their corresponding constellation components.

b1	b0	I
0	0	-3
0	1	-1
1	1	1
1	0	3

Table 2.3: Two-bit symbols with corresponding In-phase values

b3	b2	Q
0	0	3
0	1	1
1	1	-1
1	0	-3

Table 2.4: Two-bit symbols with corresponding Quadrature values

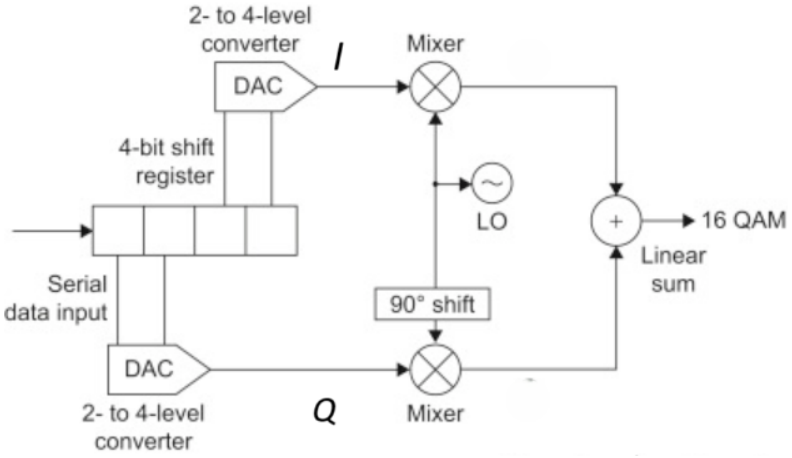

Figure 2.9: 16-QAM block diagram modulator

Generating a modulated carrier requires the use of a local oscillator to generate a cosine wave and a sine wave (phase-shifted by 90 degrees). The linear product of the cosine wave with the In-phase component and the sine wave with the Quadrature component produces an **In-phase carrier signal** and a **Quadrature carrier signal**. The linear summation of the two signals produces a 16-QAM modulated carrier.

This process satisfies the mathematical model of the 16-QAM modulated carrier (Equation 2.33), which expresses it as the sum of two orthogonal signals.

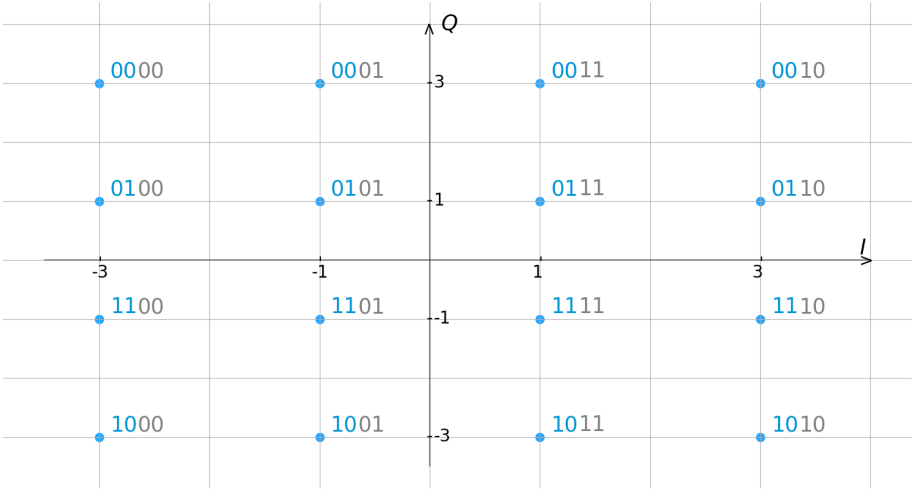

Figure 2.10: 16-QAM constellation diagram

16-Amplitude Phase Shift Keying

16-Amplitude Phase Shift Keying (16-APSK) is a digital modulation technique that combines amplitude and phase shifts to transmit 4-bit symbols. It uses a constellation diagram with 16 distinct points, arranged in a way that utilizes both amplitude and phase variations of the carrier signal to represent different data symbols. This allows for higher spectral efficiency compared to simpler modulation schemes like BPSK or QPSK, while also offering better resilience to noise compared to QAM.

In 16-APSK, the constellation points are arranged in equally spaced positions on concentric circles with increasing radii, known as 8+8-APSK, as shown in Figure 2.11. There are four parameters that define the 8+8-APSK constellation: the radii R_1 (inner circle) and R_2 (outer circle), and the angles θ_1 and θ_2. Due to the circular symmetry of the constellation, it is often defined by only two parameters: the ratio $[R_2/R_1]$ and the phase offset $[\theta_1 - \theta_2]$. These parameters may be chosen to maximize the distance between two circles.

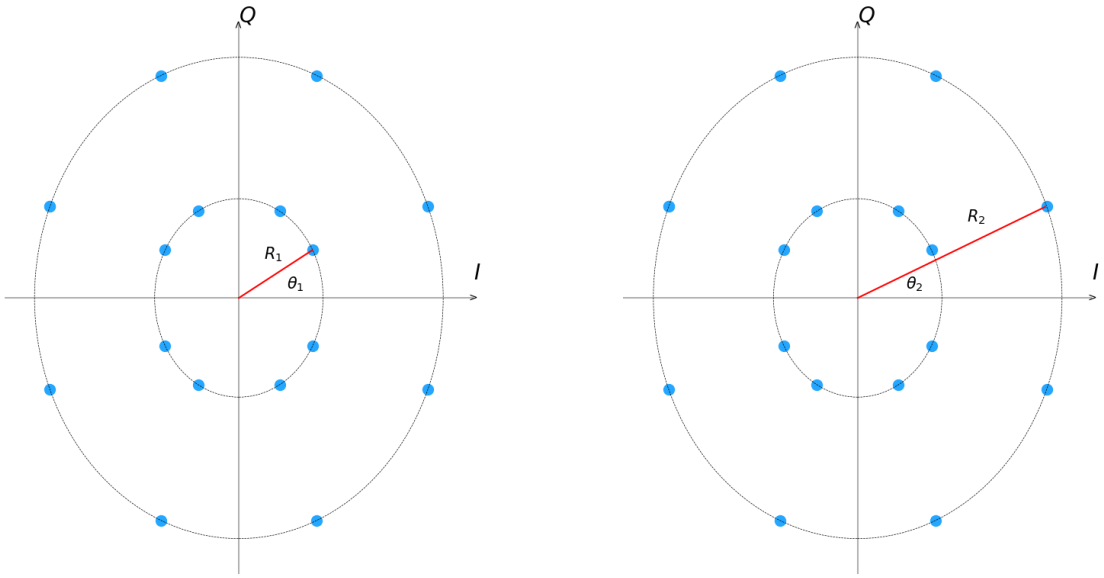

Figure 2.11: 16-APSK constellation diagram

For satellite links, the In-phase and Quadrature components of 16-APSK modulation are predefined by the **Second-Generation Satellite Digital Video Broadcasting (DVB-S2)** standard for global use [10]. Table 2.5 represents the defined complex components of the 16-APSK constellation diagram.

In the DVB-S2 standard, the symbol distribution across the constellation points in 16-APSK follows the Gray code, meaning that the maximum Hamming distance between two adjacent symbols is equal to one bit. This technique, as mentioned earlier, helps reduce errors caused by uncertainties in symbol identification at reception.

Symbols	In-phase component	Quadrature component
0000	0.4718	0.2606 i
0001	0.2606	0.4718 i
0010	-0.4718	0.2606 i
0011	-0.2606	0.4718 i
0100	0.4718	-0.2606 i
0101	0.2606	-0.4718 i
0110	-0.4718	-0.2606 i
0111	-0.2606	-0.4718 i
1000	1.2088	0.4984 i
1001	0.4984	1.2088 i
1010	-1.2088	0.4984 i
1011	-0.4984	1.2088 i
1100	1.2088	-0.4984 i
1101	0.4984	-1.2088 i
1110	-1.2088	-0.4984 i
1111	-0.4984	-1.2088 i

Table 2.5: Complex constellation points for LDPC code identifier 18/30 [10]

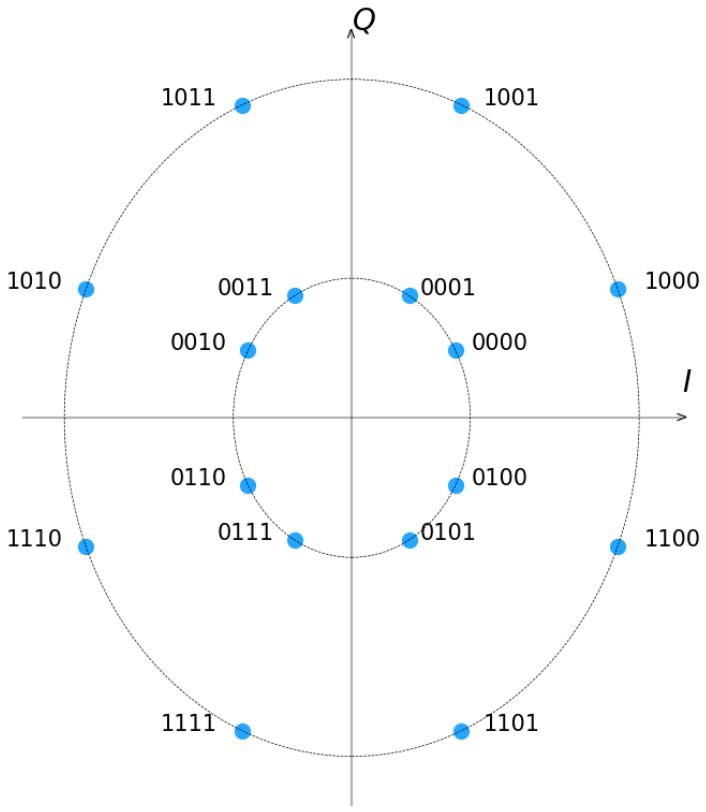

Figure 2.12: 16-APSK symbol distribution over constellation points

The Relationship Between Modulation and Carrier-to-Noise Ratio

Including channel bandwidth and transmission power, the carrier-to-noise ratio ($\frac{C}{N}$) is an important factor in determining the type of digital modulation used in the communication channel. Robust or simple modulation schemes, such as binary phase-shift keying (BPSK) or frequency-shift keying (FSK), are preferred with low $\frac{C}{N}$ (below 5 dB) due to their resistance to noise and interference. More sophisticated modulation schemes like Quadrature Phase-Shift Keying (QPSK) and 8 Phase Shift Keying (8PSK) can be used with carrier-to-noise ratios between 5 dB and 8.5 dB. Above 9 dB, high-order modulation schemes such as 16-QAM or 16-APSK become viable.

Figure 2.13 shows a comparison of the spectral efficiency of the DVB-S2 and DVB-S2X satellites relative to the carrier-to-noise ratio [8], illustrating how modulation order increases with the carrier to noise ratio [$\frac{C}{N}$].

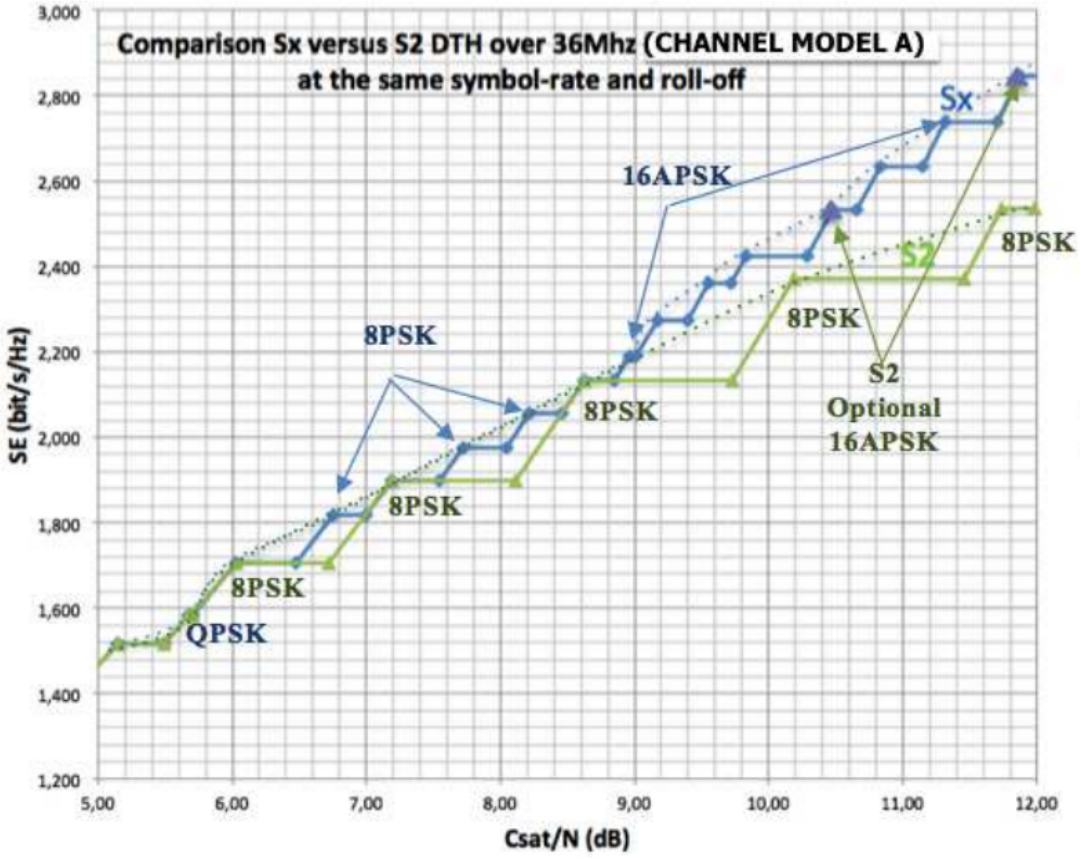

Figure 2.13: DVB-S2 and DVB-S2X spectral efficiency in the ($\frac{C}{N}$) range typical for broadcasting [8]

2.4 Signal Sampling Theory

Converting an incoming modulated carrier from an analog signal into a discrete digital signal is a crucial process to enable processing with digital systems such as adaptive impulse systems. Therefore, it is necessary to clarify the method of digitizing signals.

Pulse Code Modulation (PCM) is the most common method used to digitize an analog signal, involving three main processes:

- Sampling,

- Quantization,
- Source encoding.

Signal Sampling

Sampling consists of taking samples of the analog signal at regularly spaced intervals. Theoretically, sampling an analog signal involves multiplying it with an impulse train (sampling function) spaced by the sampling period T_s.

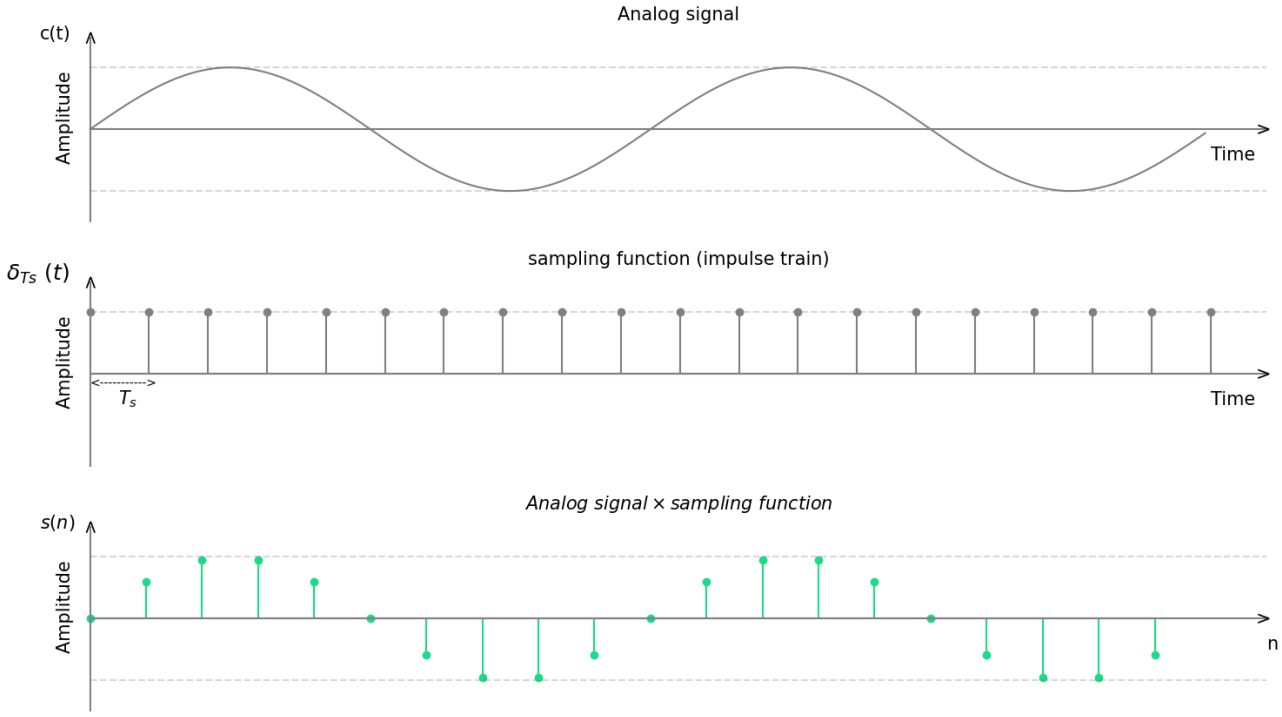

Figure 2.14: Sampling an analog signal

The time interval between two consecutive samples (sampling period) is a crucial factor that determines how accurately a continuous signal is represented by discrete samples. Choosing the appropriate sampling period—or its reciprocal, the sampling frequency F_s—depends on factors like the signal's characteristics and the system's requirements. This is where the Nyquist-Shannon sampling theorem applies:

$$s(t) = c(t) \times \delta_{T_s}(t) \tag{2.34}$$

$$s(t) = c(t) \times \sum_{n=-\infty}^{+\infty} \delta(t - n \cdot T_s) \tag{2.35}$$

Transforming $s(t)$ to the frequency domain converts the multiplicative product in the time domain into a convolutional product in the frequency domain:

$$\mathrm{TF}[s(t)] = \mathrm{TF}\left[c(t) \times \sum_{n=-\infty}^{+\infty} \delta(t - n \cdot T_s)\right] \tag{2.36}$$

$$S(f) = \frac{1}{T_s} \cdot \sum_{n=-\infty}^{+\infty} C(f) * \delta(f - n \cdot f_s) \tag{2.37}$$

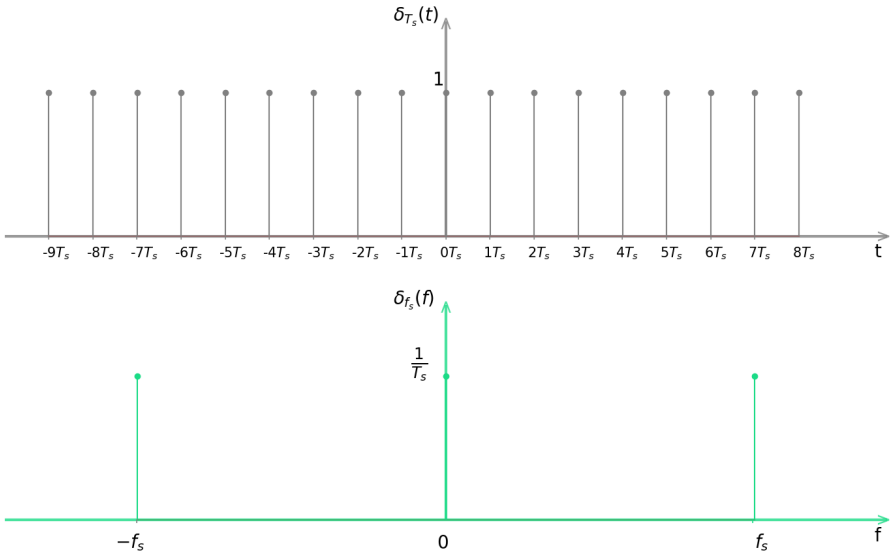

Figure 2.15: Representation of the sampling function in time and frequency domains

The convolutional product of the signal frequency spectrum $C(f)$ and $\delta(f - n \cdot f_s)$ generates a frequency spectrum train for different values of n, spaced by f_s:

$$S(f) = \frac{1}{T_s} \cdot \sum_{n=-\infty}^{+\infty} C(f - n \cdot f_s) \tag{2.38}$$

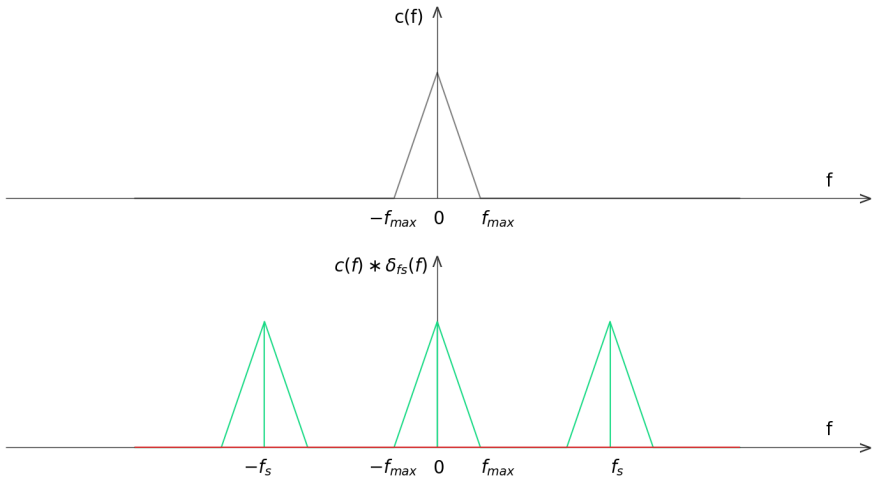

Figure 2.16: Signal frequency spectrum convolved with the sampling function in the frequency domain

Overlap occurs when the frequency spectrum of a signal $C(f)$ and its shifted version $C(f - f_s)$ interfere in the same frequency band. To avoid this, the Nyquist Theorem requires the sampling frequency to be at least twice the maximum frequency of the signal spectrum:

$$[h] f_s \geq 2 \cdot f_{\max} \tag{2.39}$$

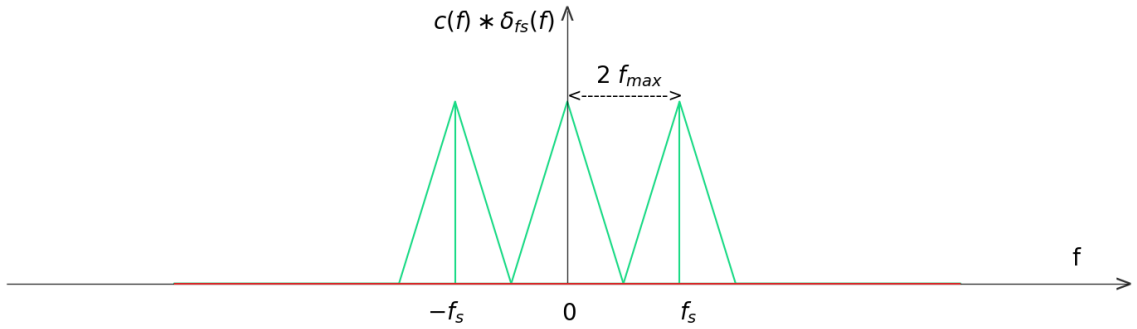

Figure 2.17: Signal frequency spectrum with a sampling frequency twice the maximum frequency

Signal Quantization

After the sampling process, quantization measures the amplitude intensity at each discrete point in time. Since it is impossible to represent all possible amplitude values, the amplitude range is divided into a finite number of discrete levels. The number of levels is typically 2^{nb}, where nb is the number of bits used for digital representation. Each sample's amplitude is assigned to the nearest available discrete level, which is then encoded as a binary number.

Signal Encoding

After sampling and quantization, the last process which is encoding converts the discrete amplitude values into a digital format (binary code) for storage, processing, or transmission.

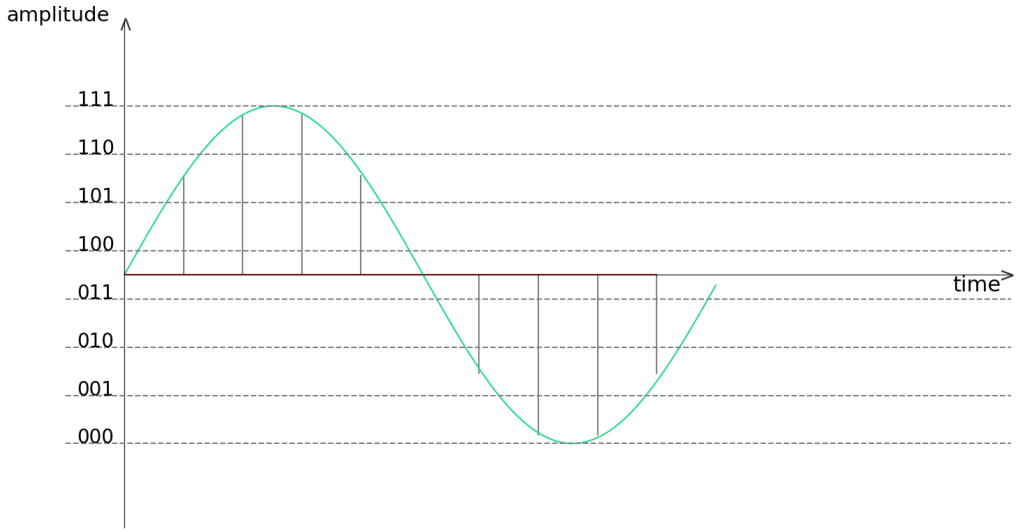

Figure 2.18: Signal samples encoding

2.5 APS signal processing model

The APS model for signal processing, represent in training a **digital filters** through a special **machine learning algorithm**, how to treat and maintain arriving modulated carrier waves that affected by noise and fading.

Digital filter impulse response

The *impulse response* often called *unit impulse* is the core of digital filter describe it's behavior in time domain, and embodying his effect at an input signal through convolution product. The impulse response of an filter usually denoted as $h(n)$.

$$h(n) = [h_0 \quad h_1 \quad h_2 \quad h_3..... \quad h_n] \tag{2.40}$$

Where $h_0 \quad h_1 \quad h_2...$ are the value of the impulse response at discrete time points.
The output signal of the convolutional product between $h(n)$ and input signal $x(n)$ express by $y(n)$, represents the effect of the filter at $x(n)$.

$$y(n) = h(n) \ * \ x(n) \tag{2.41}$$

$$y(n) = \sum_{m \ = \ -\infty}^{+\infty} h(n \ - \ m) \cdot x(n) \tag{2.42}$$

The pulses term in Adaptive pulses systems refer to the impulse response which include that a filter embodying in the model through its impulse response and apply his effect through convolutional product.

Machine learning model

Machine learning (ML) models, can be categorized into various types, each with its own strengths and applications. However, most of this models follow the same training process focusing on four key components. Figure 2.19 represent key components for create a machine learning model.

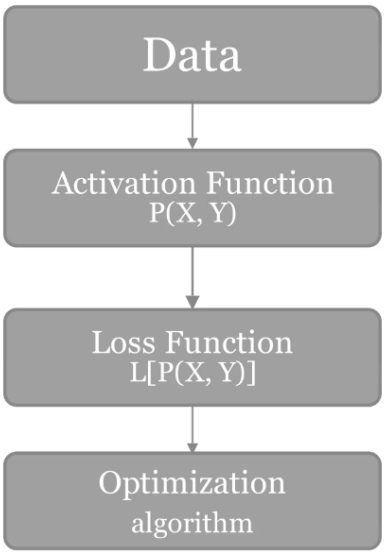

Figure 2.19: Simplified flow of a machine learning model training process

Single system model

Let's start by training a model of one filter with following the machine learning flow components.

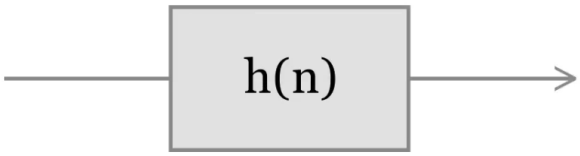

Figure 2.20: One filter block diagram

Data

The training process begins with data, which must be compatible with our model. For supervised machine learning models, which aim to achieve targets, the data is represented by two matrices. The input signal X—in our model, a signals for a modulated carrier wave affected by noise and fading—and the desired output signals Y—represents the original signals.

$$X_{(I\ \times\ J)} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & .....x_{1J} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & .....x_{2J} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & .....x_{3J} \\ . & . & . & . & . & \\ . & . & . & . & . & \\ x_{I1} & x_{I2} & x_{I3} & x_{I4} & x_{I5} & .....x_{Ij} \end{pmatrix} \qquad Y_{(I\ \times\ J)} = \begin{pmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} & .....y_{1J} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} & .....y_{2J} \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35} & .....y_{3J} \\ . & . & . & . & . & \\ . & . & . & . & . & \\ y_{I1} & y_{I2} & y_{I3} & y_{I4} & y_{I5} & .....y_{IJ} \end{pmatrix}$$

Each row i in X represents an input signal and must be accompanied in the same row with a desired output signal in Y. Both rows must be of the same length. This means that our model needs for any number of input samples J, give the same number J of output samples.

$$X_i = [x_{i1}\ \ x_{i2}\ \ x_{i3}\ \ x_{i4}\ \ .\ .\ .\ .\ \ x_{iJ}] \qquad Y_i = [y_{i1}\ \ y_{i2}\ \ y_{i3}\ \ y_{i4}\ \ .\ .\ .\ .\ \ y_{iJ}]$$

We're going to train a digital filter, which is actually an impulse response, so we represent our filter as a one dimensional matrix $(1\ \times\ N)$.

$$W_{(1\ \times\ N)} = [w_{11}\ \ w_{12}\ \ w_{13}\ \ .\ .\ .\ .\ \ w_{1N}] \tag{2.43}$$

The convolution product of W with the input signals X allows us to apply the filter effect to all our inputs. Let the output matrix A be the result of the convolution of W with X.

$$A_{(I\ \times\ N+J-1)} = W_{(1\ \times\ N)}\ *\ X_{(I\ \times\ J)} \tag{2.44}$$

$$A(i,\ \ k) = \sum_{m=1}^{k}\ W(1,\ \ k-m+1)\ \cdot\ X(i,\ \ m) \tag{2.45}$$

Where $1 \leq k \leq N+J-1 \quad \text{and} \quad 1 \leq i \leq I$.

$$A_{(I\ \times\ N+J-1)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & .....a_{1\ J+N-1} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & .....a_{2\ J+N-1} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & .....a_{3\ J+N-1} \\ . & . & . & . & . & \\ . & . & . & . & . & \\ a_{a1} & a_{a2} & a_{I3} & a_{I4} & a_{I5} & .....a_{I\ J+N-1} \end{pmatrix}$$

We notice in A that the number of columns is $N + J - 1$ which is definitely greater than J which poses a problem, as the number of columns in the convolutional output matrix A must be identical to the number of columns in the target matrix Y.

Another problem arises, which is that we need all filter impulse must interact in every sample on the output signal, which is not happen. For example, take a digi tal filter with four different impulses response W and an input signal X with same length.

$$W_{(1\ \times\ 4)} = [w_{11}\ \ w_{12}\ \ w_{13}\ \ w_{14}] \qquad X_{(1\ \times\ 4)} = [x_{11}\ \ x_{12}\ \ x_{13}\ \ x_{14}] \tag{2.46}$$

The convolutional product between W and X leads to the matrix $A_{(1\ \times\ 7)}$ containing the following samples.

$A_{(1,\ \ 1)} = w_{11} \cdot x_{11}$
$A_{(1,\ \ 2)} = w_{12} \cdot x_{11} + w_{11} \cdot x_{12}$
$A_{(1,\ \ 3)} = w_{13} \cdot x_{11} + w_{12} \cdot x_{12} + w_{11} \cdot x_{13}$
$A_{(1,\ \ 4)} = w_{14} \cdot x_{11} + w_{13} \cdot x_{12} + w_{12} \cdot x_{13} + w_{11} \cdot x_{14}$
$A_{(1,\ \ 5)} = w_{14} \cdot x_{12} + w_{13} \cdot x_{13} + w_{12} \cdot x_{14}$
$A_{(1,\ \ 6)} = w_{14} \cdot x_{13} + w_{13} \cdot x_{14}$
$A_{(1,\ \ 7)} = w_{14} \cdot x_{14}$

Note that for all samples of output signal A, only $A_{(1,\ \ 4)}$ have interaction with all filter impulse, where other sample like $A_{(1,\ \ 1)}$ have interaction only with w_{11} which represent a problem of non-balance at the training level.

Zero-phase filter

The phase shifting of the output signal is the key solution to this problems, begins with the need to train a zero-phase filter. A zero-phase filter is a special case of a linear-phase filter where the constant phase shift is zero, means that, the output signal aligns perfectly in time with the input signal. A zero-phase filter can be implemented by applying a successive forward and backward discrete time convolution to the input signal. To show this analytically, take the same example (2.46).

$$W_{(1\ \times\ 4)} = [w_{11}\ \ w_{12}\ \ w_{13}\ \ w_{14}] \qquad X_{(1\ \times\ 4)} = [x_{11}\ \ x_{12}\ \ x_{13}\ \ x_{14}]$$

Forward convolution

Forward convolution refers to the standard discrete-time convolution between W and X, which introduces a forward phase shift. It is very useful to present the convolution product as Matrix multiplication. Considering V the result of Matrix multiplication, between W^T and X.

$$V_{(4\ \times\ 4)} = W^T X \tag{2.47}$$

$$V_{(4\ \times\ \ 4)} = \begin{pmatrix} w_{11} \cdot x_{11} & w_{11} \cdot x_{12} & w_{11} \cdot x_{13} & w_{11} \cdot x_{14} \\ w_{12} \cdot x_{11} & w_{12} \cdot x_{12} & w_{12} \cdot x_{13} & w_{12} \cdot x_{14} \\ w_{13} \cdot x_{11} & w_{13} \cdot x_{12} & w_{13} \cdot x_{13} & w_{13} \cdot x_{14} \\ w_{14} \cdot x_{11} & w_{14} \cdot x_{12} & w_{14} \cdot x_{13} & w_{14} \cdot x_{14} \end{pmatrix} \tag{2.48}$$

Then the sum of each diagonal elements allow to obtain the convolutional product samples between W and X, or in other word The matrix A.
Considering The matrix A elements as.

$$A_{(1\ \times\ 7)} = [a_{11}\ \ a_{12}\ \ a_{13}\ \ a_{14}\ \ a_{15}\ \ a_{16}\ \ a_{17}] \tag{2.49}$$

Back convolution

The aim of the back convolution process is to shift the signal backwards to the same degree as the forward shift, and this can be achieved by flip the filter and then applying a discrete time convolution between A and $Flip\ (W)$.

$$Flip(W) \ = \ [w_{14} \ \ w_{13} \ \ w_{12} \ \ w_{11}] \tag{2.50}$$

Replacing the ordinary discrete time convolution with matrix multiplication allows for better study of the results. Considering B the result of matrix product of $Flip(W)$ and A.

$$B_{(4\ \times\ 7)} = Flip(W)^T\ A \tag{2.51}$$

$$B_{(4\ \times\ \ 7)} = \begin{pmatrix} w_{14}\cdot a_{11} & w_{14}\cdot a_{12} & w_{14}\cdot a_{13} & w_{14}\cdot a_{14} & w_{14}\cdot a_{15} & w_{14}\cdot a_{16} & w_{14}\cdot a_{17} \\ w_{13}\cdot a_{11} & w_{13}\cdot a_{12} & w_{13}\cdot a_{13} & w_{13}\cdot a_{14} & w_{13}\cdot a_{15} & w_{13}\cdot a_{16} & w_{13}\cdot a_{17} \\ w_{12}\cdot a_{11} & w_{12}\cdot a_{12} & w_{12}\cdot a_{13} & w_{12}\cdot a_{14} & w_{12}\cdot a_{15} & w_{12}\cdot a_{16} & w_{12}\cdot a_{17} \\ w_{11}\cdot a_{11} & w_{11}\cdot a_{12} & w_{11}\cdot a_{13} & w_{11}\cdot a_{14} & w_{11}\cdot a_{15} & w_{11}\cdot a_{16} & w_{11}\cdot a_{17} \end{pmatrix}$$

By representing final outputs of the forward-backward convolution between W and X, with the sum of the colored diagonal elements of the matrix B we can say that our problems are solved.

- First, we are absolutely sure that the output signal is exactly in time with the input signal a zero-phase shift filter.
- Second, all filter samples interact fully with each output signal sample.
- Third, note that the colored diagonals have the same length of elements and that the number of output samples Z equals the number of input samples X. Where Z samples are.

$$\begin{aligned} Z_{(1,\ \ 1)} &= w_{11}\cdot a_{11} \ + \ w_{12}\cdot a_{12} \ + \ w_{13}\cdot a_{13} \ + \ w_{14}\cdot a_{14} \\ Z_{(1,\ \ 2)} &= w_{11}\cdot a_{12} \ + \ w_{12}\cdot a_{13} \ + \ w_{13}\cdot a_{14} \ + \ w_{14}\cdot a_{15} \\ Z_{(1,\ \ 3)} &= w_{11}\cdot a_{13} \ + \ w_{12}\cdot a_{14} \ + \ w_{13}\cdot a_{15} \ + \ w_{14}\cdot a_{16} \\ Z_{(1,\ \ 4)} &= w_{11}\cdot a_{14} \ + \ w_{12}\cdot a_{15} \ + \ w_{13}\cdot a_{16} \ + \ w_{14}\cdot a_{17} \end{aligned}$$

One filter training

Now to train our filter W we need the (Data) X and Y. The forward backward convolution allow to apply the filter effect at the input data X which give the matrix Z. The training of the model lies in improving its effect on the input data X so that the output result Z matches the desired output Y as closely as possible.
Taking the forward Backward convolution between W and X.

$$Z_{(I\ \times\ J)} = W_{(1\ \times\ N)}\ **\ X_{(I\ \times\ J)} \tag{2.52}$$

$$Z_{(I\ \times\ J)} = \begin{pmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} & .....z_{1J} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} & .....z_{2J} \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} & .....z_{3J} \\ . & . & . & . & . & \\ . & . & . & . & . & \\ z_{a1} & z_{a2} & z_{I3} & z_{I4} & z_{I5} & .....z_{IJ} \end{pmatrix}$$

Now we need to know how much our filter oriented to its target, or in other words how close Z is to Y.

Loss Function

The efficiency of a filter can be represented by a likelihood function, which is a measure of the probability of seeing the target under different samples values of the filter.

The correlation coefficient

To estimate the likelihood we need to represent the correlation between each pair of signals in, Z and Y, as the probability of Z equal Y, $P(Z_i = Y_i)$. These can be reached by the correlation coefficient between Z_i and Y_i.
Considering P the result array of the correlation coefficient between Z and Y.

$$P_{(I\ \times\ 1)}(i,\ \ 1) = \frac{1}{J}\sum_{j=1}^{J}\ \ \left(\frac{z_{ij}\ -\ \overline{Z_i}}{\sigma Z_i}\right)\ \cdot\ \left(\frac{y_{ij}\ -\ \overline{Y_i}}{\sigma Y_i}\right) \tag{2.53}$$

$$\overline{Z_i} = \frac{1}{J}\sum_{j=1}^{J} z_{ij} \qquad \sigma Z_i = \sqrt{\frac{1}{J}\sum_{j=1}^{J}(z_{ij} - \overline{Z_i})^2} \tag{2.54}$$

$$\overline{Y_i} = \frac{1}{J}\sum_{j=1}^{J} y_{ij} \qquad \sigma Y_i = \sqrt{\frac{1}{J}\sum_{j=1}^{J}(y_{ij} - \overline{Y_i})^2} \tag{2.55}$$

Where

- $1 \leq i \leq I$
- $\overline{Z_i}$ and $\overline{Y_i}$ are both the average of the signal values in row i.
- σZ_i and σY_i are both the standard deviation of signal at row i.

Note that to exploit the correlation coefficient and train an effective model, the filter length, (N), must be proportional to the input signals X.

The representation of correlation coefficient as function of filter impulses depends on several factors, the most important of which is the number of impulses (length of filter). Where small length filter considered as finite impulse response (FIR) type has a weak response with the correlation coefficient and can not generate a function with it. Infinite impulse response filters (IIR) in the other hand represent a high length filters and give a better response with the correlation coefficient. means That the correlation coefficient are better for train filter with large samples (IIR) filters.

There is no way to visualizing the graph of correlation coefficient with a high length filter, but after some tests we can propose Figure 2.21 which represent the correlation coefficient response of output signal with desired output in function of the filter coefficient.

Note that in W axis the components $W0\ \ W1\ \ W2$ are not a filter samples. each component represent a different filter.

The Likelihood function

The likelihood function represents by the multiplication of probability of equality $P(Z = Y)$ which mean the multiplication of P array elements.

The correlation coefficient take a range of value from -1 to 1 where the negative values express the opposition between two entry. Where we cannot consider the negative correlation coefficient

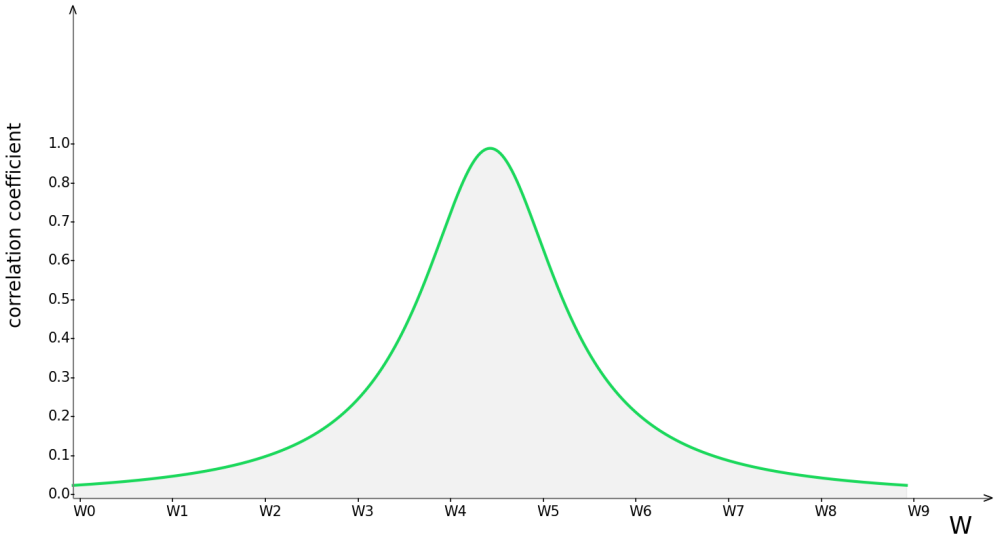

Figure 2.21: Proposed Correlation coefficient curve

as probability for the likelihood function, so we take the square of P array elements as probability of Z equal Y.

$$l = \prod_{i=1}^{I} P(i, \quad 1)^2 \tag{2.56}$$

$$l = \prod_{i=0}^{I} \left[\frac{1}{J} \sum_{j=1}^{J} \left(\frac{z_{ij} - \overline{Z_i}}{\sigma Z_i} \right) \cdot \left(\frac{y_{ij} - \overline{Y_i}}{\sigma Y_i} \right) \right]^2 \tag{2.57}$$

Practically we use the **maximum likelihood estimation (MLE)** which simplify calculations and provides a more numerically stable way to assess how well a model fits observe the data. The MLE represent by the natural logarithm, of the likelihood function.

$$LL = \ln \left(\prod_{i=0}^{I} \left[\frac{1}{J} \sum_{j=1}^{J} \left(\frac{z_{ij} - \overline{Z_i}}{\sigma Z_i} \right) \cdot \left(\frac{y_{ij} - \overline{Y_i}}{\sigma Y_i} \right) \right]^2 \right)$$

By simplifying this equation.

$$LL = 2 \cdot \sum_{i=1}^{I} \ln \left(\frac{1}{J} \sum_{j=1}^{J} \left(\frac{z_{ij} - \overline{Z_i}}{\sigma Z_i} \right) \cdot \left(\frac{y_{ij} - \overline{Y_i}}{\sigma Y_i} \right) \right)$$

$$LL = 2 \cdot \sum_{i=1}^{I} \ln \left(\sum_{j=1}^{J} \left(\frac{z_{ij} - \overline{Z_i}}{\sigma Z_i} \right) \cdot \left(\frac{y_{ij} - \overline{Y_i}}{\sigma Y_i} \right) \right) - 2 \cdot I \ln (J) \tag{2.58}$$

In the end obtaining the loss function (log loss) by the negative of the maximum Likelihood estimation (MLE).

$$L = -2 \cdot \sum_{i=1}^{I} \ln \left(\sum_{j=1}^{J} \left(\frac{z_{ij} - \overline{Z_i}}{\sigma Z_i} \right) \cdot \left(\frac{y_{ij} - \overline{Y_i}}{\sigma Y_i} \right) \right) + 2 \cdot I \ln (J) \tag{2.59}$$

The ability of the correlation coefficient to identify two signals as "perfectly correlated" even if they different in amplitude (amplification between them) is very useful in training with large

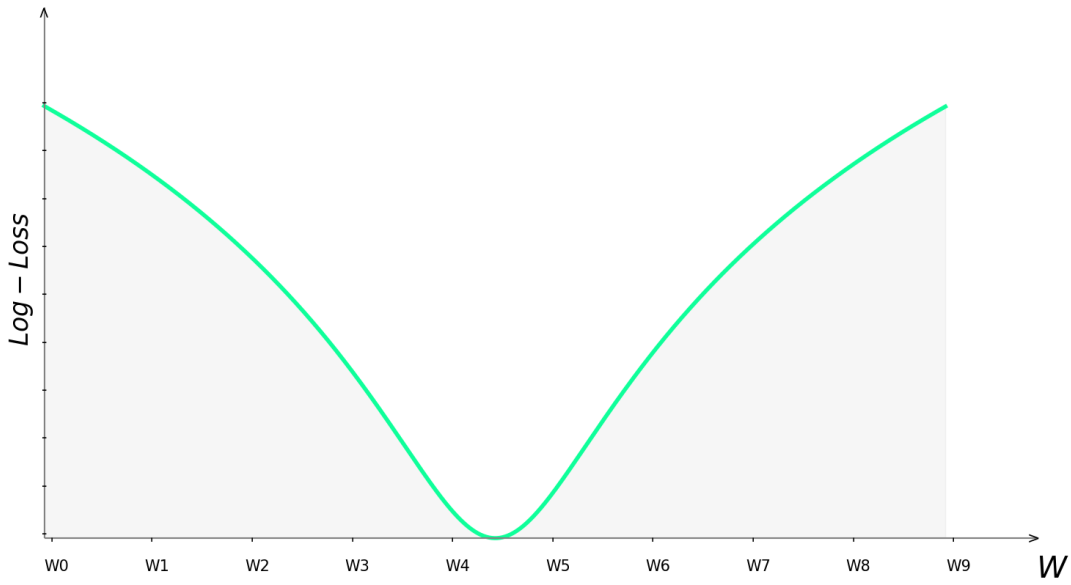

Figure 2.22: Proposed loss curve in function of filter samples

amounts of data, and helps reduce the second lobes in the learning curve. Figure 2.22 represent the proposed loss curve in function of different filters. The Log-loss, are in function of filter samples, which mean that the increase or decrease in its value are directly related to the filter samples. A decrease in the loss function value mean an improve in the filter ability to recognizing data, where Having a minimum loss value (preferably zero) means that the filter has full improvement (Adaption) to the target with it's full capacity.

Optimization algorithm

Minimizing the loss function and improve the filter capacity it's our main goal. An optimization algorithm is a mathematical procedure used (in our case) to minimize the loss function value as possible by the continues Adapting of model parameters (filter samples).

Gradient descent

The gradient descent is an optimization algorithm widely used in machine learning to solve optimization problems. This algorithm can be defined in four steps .

- Initialization, this step are just in beginning of training and never repeated again. It involves randomly assigning a starting value to different filter samples.

- Gradient calculation, which involves calculating the gradient of the loss function according to each sample.The gradient refers to the variation of the loss function according to a specific sample.

- Update filter, according to the gradient values each sample update his value for descent the loss function to the minimum possible value

- Repeat, in loop repeat the last two steps for a continues updating of filter.

Mathematically this steps can be represented by the next equations.
The gradient of loss function considering as.

$$\frac{\partial L}{\partial w_{1n}} = \sum_{i=1}^{I} \frac{\sum_{j=1}^{J} \frac{(z_{ij}-\overline{Z_i})' \cdot \sigma Z_i \;-\; (z_{ij}-\overline{Z_i}) \cdot \sigma Z_i'}{(\sigma Z i)^2} \;\cdot\; \frac{(y_{ij}-\overline{Y_i})}{\sigma Y i}}{\sum_{j=1}^{J} \left(\frac{(z_{ij}-\overline{Z_i})}{\sigma Z i} \cdot \frac{(y_{ij}-\overline{Y_i})}{\sigma Y i} \right)} \tag{2.60}$$

Where n take the range of $1 \leq n \leq N$, and the gradient of σZ_i equals to.

$$\sigma Z_i' = \frac{1}{J} \frac{\sum_{j=1}^{J} (z_{ij} - \overline{Z_i})' \;\cdot\; (z_{ij} - \overline{Z_i})}{\sqrt{\frac{1}{J} \sum_{j=1}^{J} (z_{ij} - \overline{Z_i})^2}} \qquad z_{ij}' = \frac{\partial z_{ij}}{\partial w_{1n}}$$

The update of samples can be represent by the next equation.

$$W_{t+1}(1,\;\; n) = W_t(1,\;\; n) \;-\; \alpha \, \frac{\partial L}{\partial w_{1n}} \tag{2.61}$$

The gradient will indicate the increase or decrease of loss function according to specific sample.

- If the loss is decreasing, the samples will experience an increase in value..
- If the loss is increasing, the samples will experience a decrease in value.

The term "filter training" represents the continuous convergence of filter samples to optimal values.

FIR filter model

As we discussed above, the correlation coefficient is not the best solution for training small length filters (finite impulse response filters), to compensate for this we suggest using the Lorentzian Mean Square Error (L-MSE).

Lorentzian Mean Square Error

The Lorentzian square error (L-MSE) has a better response than the correlation coefficient with small length filters, where the learning process can be expressed in the following steps.

- Calculate Z the forward-backward convolution between X and W
- Calculate the mean square error between the Z and Y signals. Consider the matrix M that represents the mean square error between Z and Y.

$$M_{(I \;\times\; 1)}(i,\;\; 1) = \frac{1}{J} \sum_{j=1}^{J} (Y(i,\;\; j) - Z(i,\;\; j))^2 \qquad m_{i1} = \frac{1}{J} \sum_{j=1}^{J} (y_{ij} - z_{ij})^2 \tag{2.62}$$

- Calculate the probability of Z equal Y, $P(Z = Y)$ through the Lorentzian function.

$$P_{(I \;\times\; 1)}(i,\;\; 1) = \frac{1}{m_{i1}^2 \;+\; 1} \tag{2.63}$$

- Estimation of the likelihood function, from the calculated Lorentz probability.

$$l = \prod_{i=1}^{I} P(i, \ \ 1) \qquad l = \prod_{i=1}^{I} \frac{1}{m_{i1}^2 \ + \ 1} \tag{2.64}$$

- Use the maximum likelihood estimation (MLE) for simplifying calculations and provide data seeing ability.

$$LL = \ln\left(\prod_{i=1}^{I} \frac{1}{m_{i1}^2 \ + \ 1}\right) \qquad LL = \sum_{i=1}^{I} \ln \frac{1}{m_{i1}^2 \ + \ 1} \tag{2.65}$$

- Obtain The loss function by taking the negative of the MLE.

$$L = -\sum_{i=1}^{I} \ln \frac{1}{m_{i1}^2 \ + \ 1} \qquad L = \sum_{i=1}^{I} \ln(\ m_{i1}^2 \ + \ 1) \tag{2.66}$$

- Calculate the gradient of loss function with different filter samples for the gradient descent algorithm.

$$\frac{\partial L}{\partial w_{1n}} = -4 \cdot \sum_{i=1}^{I} \cdot \frac{\sum_{j=1}^{J} z'_{ij} \ \cdot \ (y_{ij} - z_{ij})^3}{\sum_{j=1}^{J} (y_{ij} - z_{ij})^4} \qquad z'_{ij} = \frac{\partial z_{ij}}{\partial w_{1n}} \tag{2.67}$$

  Where $1 \leq n \leq N$ and N are the length of filter.

- Use the calculated gradient in the gradient descent algorithm equation to update the filter and repeat the process to continue training.

$$W_{t+1}(1, \ \ n) = W_t(1, \ \ n) - \alpha \cdot \frac{\partial L}{\partial w_{1n}} \tag{2.68}$$

The Concatenate Zero Algorithm

As we saw earlier, using the correlation coefficient or Lorentzian mean square error to train filters requires calculating the z elements gradient according to different filter samples. To achieve this, we use the concatenate-zero algorithm, which can generate Gz, a tensor containing the gradient of z elements (z') according to all filter samples. This algorithm can be expressed in the following steps.

- The algorithm first take the filter W and the input data X and create two matrix F and C where F equal.

$$F_{(1 \times N)} = flip \ (W) \tag{2.69}$$

- In a loop, the k-iterator takes the range of integers from zero to the length of the filter minus one.

$$0 \leq k \leq N - 1$$

- For each iteration, the algorithm starts by concatenating k zeros in matrix C with matrix F, then performs the convolutional product with X. The result is then stored in a matrix called Gf.

$$C_{(1 \ \times \ N+k)} = (k) \ Concatenate \ (F) \tag{2.70}$$

$$Gf_{(I \ \times \ f)} = C \ \ * \ \ X \tag{2.71}$$

  Where (f) equal $N + k + J - 1$.

- The algorithm next concatenate a number of zeros equal $(N-k-1)$ with W in matrix C then exercises the convolutional product with X. the result will be then saved in array called Gb.

$$C_{(1\ \times\ 2N-k-2)} = (N-k-1)\ Concatenate\ (W) \tag{2.72}$$

$$Gb_{(I\ \times\ b)} = C\ *\ X \tag{2.73}$$

  Where (b) equal $2N+J-K-2$.

- Calculating next (d) the difference between b and f.

$$d = b - f \tag{2.74}$$

- If (d) are greater or equal $'0'$ then concatenate Gf with an array D contain a number of zeros equal (d). .

$$Gf_{(I\ \times\ b)} = (Gf)\ Concatenate\ (D_{(I\ \times\ d)})$$

- Else concatenate Gb with an array D contain a number of zeros equal the absolute value of (d).

$$Gb_{(I\ \times\ f)} = (Gb)\ Concatenate\ (D_{(I\ \times\ d)})$$

- Save the summation of Gf and Gb in the tensor Gz where

$$Gz_{(I\ \times\ J\ \times\ N)}(:,\ :,\ k) = Gf\ +\ Gb \tag{2.75}$$

Note that for an iteration with a value of k correspond the gradient according to the sample $W(1,\ \ k+1)$ in the filter.

2.6 APS Multi-Systems Model

The primary goal of learning any AI model is to acquire the ability to handle various challenging situations. This learning requires massive data containing diverse scenarios and a flexible model capable of learning from them. In the field of signals, signals can take on different states, and no single system can handle them all.

A deep learning model

A multi-systems model is a deep learning model consisting of more than one system that follows a specific configurations. The multiple systems learn simultaneously to develop flexibility and cooperation in dealing with different situations.

Model configuration

A multi systems model can take on different configurations. Any multi systems model, based on the number of layers, determines, how deep the processing that will be performed on the input. Each layer can take a specific number of systems and length of samples , where adjacent layers with different length need to an **"element-wise operation"** to continuing the forward flow of input. Figure 2.23 illustrates a three-layer multi systems model, where first and second layers contain the same number of systems, but their lengths may vary. It's worth noting that layers two and three contain different numbers of systems, which requiring the **element-wise operation**, to follow the input flow.

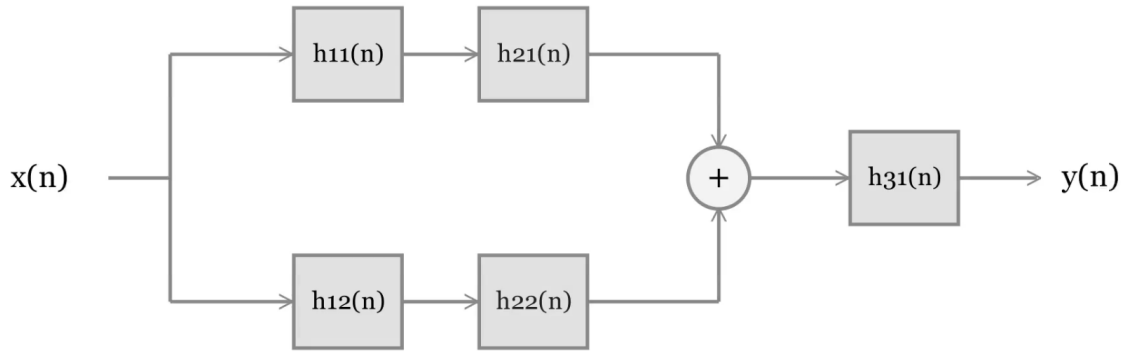

Figure 2.23: A multi systems block diagram

Model training

The training phase includes all systems simultaneously, i.e., integrated development between systems, where each unit complements the other. This learning method is very powerful in creating models that learn from a big amount of data and interact with diverse and different situations.

To train a multi-system model, we follow the flow component of machine learning Figure 2.19, which involves starting with the data. Training a flexible multi-systems requires first a useful data covering a variety of scenarios. Organizing the data into a **three-dimensional tensor** is the foundation of the model's operation, allowing it to work with multiple inputs and achieve input flow across layers of varying length. To illustrate this in practice, consider the learning process of the model, called the **H-model**, shown in Figure 2.24.

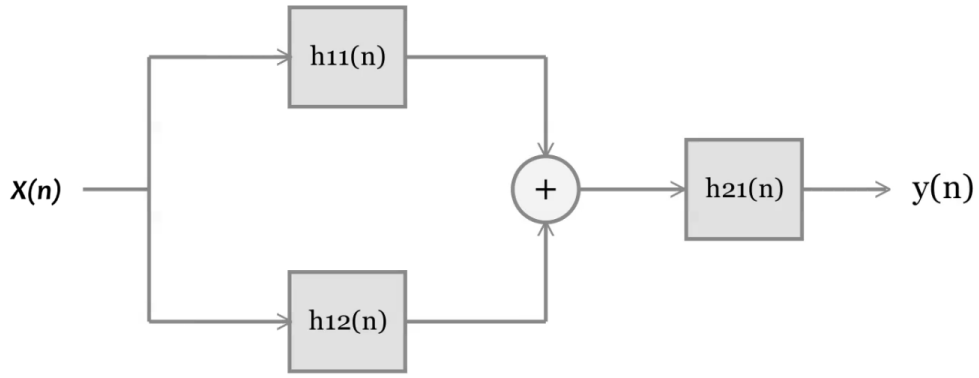

Figure 2.24: The blocks diagram of the H-model

The configuration of the **H-model** consist of two layers, where first layer contain two systems and the second layer which also the final layer have one system. The length of systems in the H-model are the same for three systems. The data as before splits into two part the input data X and desired target Y, where in this time X and Y are three dimensional tensors.

$$X_{(I \times J \times 1)} = \left(\begin{array}{cccccc} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & .....x_{1J} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & .....x_{2J} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & .....x_{3J} \\ . & . & . & . & . & \\ . & . & . & . & . & \\ x_{I1} & x_{I2} & x_{I3} & x_{I4} & x_{I5} & .....x_{Ij} \end{array} \right)$$

$$Y_{(I\ \times\ J\ \times\ 1)} = \left(\begin{pmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} & \ldots\ldots\ldots y_{1J} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} & \ldots\ldots\ldots y_{2J} \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35} & \ldots\ldots\ldots y_{3J} \\ . & . & . & . & . & \\ . & . & . & . & . & \\ y_{I1} & y_{I2} & y_{I3} & y_{I4} & y_{I5} & \ldots\ldots\ldots y_{IJ} \end{pmatrix} \right)$$

The learning algorithm focuses on two main operations, which facilitate the calculation of the loss function and the optimization of model systems.

The forward propagation

The forward propagation is the first process include the flow of input tensor through all systems. In first layer each system will perform a forward-backward convolution on its input, producing an output tensor of the same shape as the input. This output will be the input for the next layer, and so on, with following the model path to the final layer that produces the final output.

For **model** H with two systems in the first layer, the input tensor X flows through systems (h_{11} and h_{12}) resulting in an output tensor called (Z1). Where $Z1$ has a shape of ($I\ \times\ J\ \times\ 2$).

$$Z1(:,\ :,\ 1) = h_{11}\ **\ X \qquad\qquad Z1(:,\ :\ ,2) = h_{12}\ **\ X \tag{2.76}$$

A wise average in $Z1$ third axis is then required for able the flow through the last layer.

$$Z1_{(I\ \times\ J\ \times\ 1)} = \frac{Z1(:,\ :,\ 1) + Z1(:,\ :,\ 2)}{2} \tag{2.77}$$

The final layer will apply the filter h_{21} at $Z1$ and give the final output Z which equal.

$$Z_{(I\ \times\ J\ \times\ 1)} = h_{21}\ \ **\ \ Z1 \qquad\qquad Z = h_{21}\ **\ \left(\frac{h_{11}\ **\ X\ +\ h_{12}\ **\ X}{2} \right) \tag{2.78}$$

As we discussed above The calculation of Z allow the estimation of model likelihood function or more precisely the Maximum Likelihood estimation (MLE), by calculation the probability of $P(Z = Y)$.
This probability can be achieved by two different way which make two multi systems types, multi infinite systems (MIS) or multi finite systems (MFS).

Multi Infinite Systems (MIS)

The multi infinite systems designed for IIR filters take the correlation coefficient between Z and Y as the probability of equality $P(Z = Y)$, and obtain the loss function from the negative of the maximum likelihood estimation see Eq 2.42 which represent also the loss function of MIS model.

$$L = -2 \cdot \sum_{i=1}^{I} \ln \left(\sum_{j=1}^{J}\ \left(\frac{z_{ij}\ -\ \overline{Z_i}}{\sigma Z_i} \right)\ \cdot\ \left(\frac{y_{ij}\ -\ \overline{Y_i}}{\sigma Y_i} \right) \right)\ +\ 2 \cdot I \ln (J)$$

Since the third axis of Z and Y is equal to one, both of Z and Y considered as 2-dimension matrix. The variation of loss function is according now to all model systems and for minimizing the loss function it's necessary to calculate the gradient of loss according to each sample in the model.

$$\frac{\partial L}{\partial(h_{lk}(n))} = \sum_{i=1}^{I} \frac{\sum_{j=1}^{J} \frac{(z_{ij}-\overline{Z_i})'\cdot\sigma Z_i \;-\; (z_{ij}-\overline{Z_i})\cdot\sigma Z_i'}{(\sigma Zi)^2} \;\cdot\; \frac{(y_{ij}-\overline{Y_i})}{\sigma Yi}}{\sum_{j=1}^{J}\left(\frac{(z_{ij}-\overline{Z_i})}{\sigma Zi}\cdot\frac{(y_{ij}-\overline{Y_i})}{\sigma Yi}\right)} \tag{2.79}$$

$$\sigma Z_i' = \frac{1}{J}\frac{\sum_{j=1}^{J}(z_{ij}-\overline{Z_i})' \;\cdot\; (z_{ij}-\overline{Z_i})}{\sqrt{\frac{1}{J}\sum_{j=1}^{J}(z_{ij}-\overline{Z_i})^2}} \qquad z_{ij}' = \frac{\partial\, z_{ij}}{\partial(h_{lk}(n))}$$

Where l are the number of layer, k are the number of system in the layer and n refer to the according sample.

Multi Finite Systems (MFS)

On the other hand, multiple finite systems are designed for FIR filters, where the Lorentzian mean square error is taken to calculate the probability P(Z = Y) and obtain the loss function of the model through the negative of maximum likelihood estimation.

$$L = \sum_{i=1}^{I} \ln(\; M(i,\;\; 1)^2 \;+\; 1) \tag{2.80}$$

Where M are the mean square error of Y and Z, and i take the range between $1 \leq i \leq I$.

$$M_{(I\;\times\;1)}(i,\;\; 1) = \frac{1}{J}\sum_{j=1}^{J}(y_{ij} - z_{ij})^2 \tag{2.81}$$

Where the gradient of loss function according to each filter samples equal.

$$\frac{\partial L}{\partial(h_{lk}(n))} = -4\cdot\sum_{i=1}^{I}\cdot\frac{\sum_{j=1}^{J} z_{ij}' \;\cdot\; (y_{ij}-z_{ij})^3}{\sum_{j=1}^{J}(y_{ij}-z_{ij})^4} \qquad z_{ij}' = \frac{\partial z_{ij}}{\partial(h_{lk}(n))} \tag{2.82}$$

The path propagation

As we have seen, calculating Z' becomes necessary to optimize both models. In the single APS model, the algorithm concatenate zero by taking the input signals x and the filter W and returning Z'. In the multi systems model, the algorithm concatenate zero by taking the filter h and the corresponding partial tensor to calculate Z'.

The partial tensor corresponding to a filter include the flow of input through specifically the filter path. For example the path of filter h_{11} include just the input and the filter h_{21}. Figure 2.25 represent the path of filter h_{11}.

The inputs flow through this path without interacting with the filter h_{11}, which means that the only convolutional product is between the input data X and the filter h_{21}. this represented mathematically by

$$T_{(I\;\times\;J\;\times\;1)} = X \quad **\;\; h_{21} \tag{2.83}$$

Where T is the corresponding partial tensor of filter h_{11}. The algorithm concatenate zero is then take the filter h_{11} and the tensor T and calculate the gradient of the Z according to $h11$ samples. which give Z' which equal to.

$$Z'_{(I\;\times\;J\;\times\;N)} = \frac{\partial Z}{\partial\; h_{11}(n)} \tag{2.84}$$

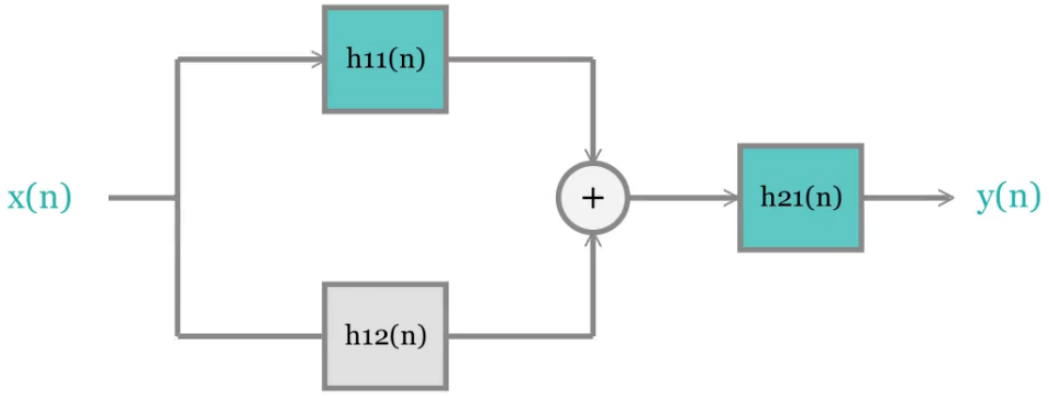

Figure 2.25: Blocks diagram of the (H) model with filter h_{11} path define

Where N are the number of samples in the filter.
The mathematic concept of the path propagation concept can expressed as next.

$$Z = h_{21} \ ** \left(\frac{h_{11} \ ** X \ + \ h_{12} \ ** X}{2} \right)$$

$$\frac{\partial Z}{\partial h_{11}(n)} = \frac{\partial}{\partial h_{11}(n)} \left(h_{21} \ ** \left(\frac{h_{11} \ ** X \ + \ h_{12} \ ** X}{2} \right) \right)$$

$$\frac{\partial Z}{\partial h_{11}(n)} = \frac{1}{2} \cdot \frac{\partial}{\partial h_{11}(n)} (h_{21} \ ** h_{11} \ ** X) \ + \ \frac{1}{2} \cdot \frac{\partial}{\partial h_{11}(n)} (h_{21} ** \ h_{12} \ ** X)$$

Where the gradient of the second part in the equation equal at h_{11} equal zero.

$$\frac{\partial Z}{\partial h_{11}(n)} = \frac{1}{2} \cdot \frac{\partial h_{11}}{\partial h_{11}(n)} \ ** (h_{21} \ ** X) \tag{2.85}$$

By repeating the same operation with other filter we find.

$$\frac{\partial Z}{\partial h_{12}(n)} = \frac{1}{2} \cdot \frac{\partial h_{12}}{\partial h_{12}(n)} \ ** (h_{11} \ ** X) \tag{2.86}$$

$$\frac{\partial Z}{\partial h_{21}(n)} = \frac{1}{2} \cdot \frac{\partial h_{21}}{\partial h_{21}(n)} \ ** (h_{11} \ ** X \ + \ h_{12} ** X) \tag{2.87}$$

The optimizing algorithm

After obtaining the gradient of X according to the different samples in each filter, the optimization of the systems become easy. We can use the gradient descent as optimization algorithm for the improving samples in each filter this represent by.

$$h_{lk}(n)_{(t+1)} = h_{lk}(n)_{(t)} - \alpha \cdot \frac{\partial L}{\partial h_{lk}(n)_{(t)}} \tag{2.88}$$

Continuous convergence of the algorithm allows for the true potential of deep learning, where systems can develop mechanisms to deal with complex situations that may be considered unsolvable by ordinary methods.

2.7 Conclusion

This chapter has established the fundamental mathematical framework for the Adaptive Pulse System, presenting the convolutional filter operations, forward-backward convolution methodology, and optimization algorithms that form the core of the model. The developed theoretical foundation demonstrates both practical applicability in signal processing and potential for further refinement through continued research into optimization techniques and alternative formulations. By rigorously deriving the impulse response characterization, loss functions, and training algorithms, this work provides a robust mathematical basis for the implementation and experimental validation discussed in subsequent chapters, while maintaining clear pathways for future enhancements to the system's performance and efficiency.

3 Systems Construction

3.1 Introduction

Training an APS model is a challenging task that requires significant computational effort and a deep understanding of learning algorithms. Due to the diversity of input data, which poses challenges of varying complexity, a single model configuration cannot handle them all. Therefore, designing models with different configurations becomes necessary, as the appropriate configuration gives the model greater data-handling capacity. In this chapter, we will discuss **Adaptive systems**, a powerful programming tool that enables APS models to be built with a variety of configurations and sizes.

3.2 Programming language

In **Adaptive systems** programming, we use **Python**, a high-level programming language used in diverse fields such as signal processing, machine learning, data analysis, and automation. Python supports object-oriented programming (OOP), which allows for a highly useful code organization for multi-part programming algorithms. Python contains numerous libraries that include modules and packages that provide pre-written code, making programming easier and more efficient.

Numpy library

Numpy are fundamental python library, provides support for large, multi-dimensional arrays and matrices with several functions to operate with them. An important advantage of Numpy that integrates well with other Python libraries.

Scipy library

Scipy is a scientific library in Python that provides numerous **mathematical algorithms** for integration, interpolation, optimization problems, signal processing, and image processing. Scipy is based on the numpy library, which provides high compatibility between matrices and mathematical operations. In our code, we will define working with Scipy in "Signal," a **submodule** of Scipy that provides signal processing tools, including filter design, spectral analysis, convolution, B-spline interpolation, and more.

3.3 Adaptive systems

Adaptive systems are programmed as a class on the concept of object-oriented programming, which consists of attributes and methods (functions) that complement each other to achieve the

goal of the class.

Code programming

The first main lines of our code involve importing the numpy and scipy libraries.

```
1 import numpy as np
2 from scipy import signal
```

In the link below we can find the complete code for the Adaptive Systems class, which allows to create and train any possible APS configuration. .

https://colab.research.google.com/drive/1mJDcK9-ix2NfFqHt1Tymi8dJ4Sncz3kQ

Model construction

Seven lines of codes are required for construct and train any (APS) possible model.

```
1 path = np.array([])
2 n_sample = np.array([])
3 configuration = np.array([])
4 Model = Adaptive_systems(configuration, n_sample, path)
5 Model.construct()
6 Model.fit(x, y, tdrb, learning_rate, systems)
7 Model.score()
```

The first Three lines are necessary for the specification of model form. Where **three arrays** can define any possible (APS) configuration.

- The line number one define the **path** array that allows specifying the connection between layers, where the "path" array take for two adjacent layer connect in cascade the value **'1'**, where take the value **'0'** for an "element-wise operation" between two layers.
- The line number two define the **n-sample** array that allow specifying the length of systems in each layer.
- The third line define the **configuration** array that allow to determine the number of systems in each layer.

The last four lines are then necessary for construct and train the predefined model.

- Lines number four and five allow the construction of an APS model, called a **"Model"** which has the form defined by the arrays ("path", "n-sample", "configuration").
- Line number six through the method **fit** allows the model to train on the input data **'x'** to give the desired target **'y'**.
- The parameter **"tdrb"** allow to determine the number of iteration on training loop, is usually define with the name epochs.

- The **Learning rate** parameter allow to determine the learning rate in the gradient decent algorithm.

- The **systems** parameter allows specifying the type of systems trained, where specifying *multi infinite systems* implies entering 'mis', or entering **'mfs'** for multi finite systems.

- The last line of code which includes the **score** method allows you to visualize the training result and the resulting loss.

Take the example of building and training a model we call Model (A) that has the configuration shown in Figure 3.1.

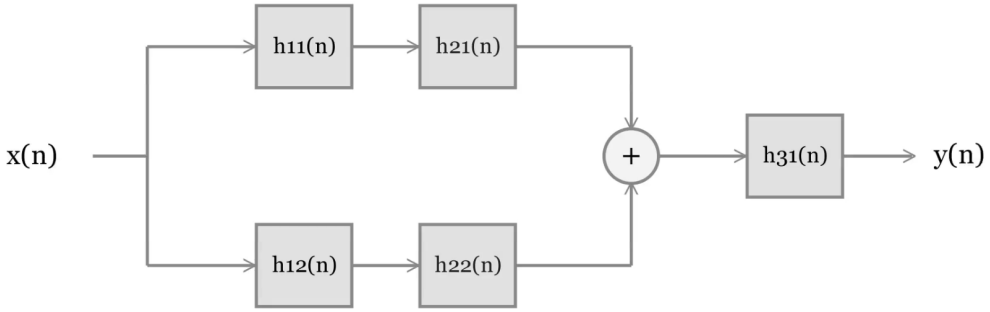

Figure 3.1: Model (A) blocks diagram

```
path = np.array([0, 1, 0])
n_sample = np.array([n1, n2, n3])
configuration = np.array([2, 2, 1])
model = Adaptive_systems(configuration, n_sample, path)
model.construct()
model.fit(x, y, tdrb = 100, learning_rate = 2e-4, systems = 'mis')
model.score()
```

Where $n1$, $n2$ and $n3$ are respectively the length of systems (number of samples) in layer one, layer two and layer three.

3.4 Binary phase shift keying noise reduction model

With back to our project on a noise reduction model geared towards digital modulation types.

Data

To train any model, we first need data. Our data consists of a set of clear binary phase shift modulation signals (the target) and a noisy version of the target (the input). The following points describe the characteristics of the data.

- - Signal frequency: We assume our signal has a frequency of 500 Hz and was sampled at 5000 Hz, ten times the signal frequency.

- We assume the signal length is 200 samples, and the bit flip period is ten times the signal period, meaning one phase shift in the modulated signal.

- Binary phase shift modulation includes two possible phase shifts, and with one bit flip in the signal, we have four possible signals.

- For each signal, we create fifty versions with white noise added to them, with a carrier-to-noise ratio of (-5 – 5) dB.

- Finally, for the four signals, each contains fifty distorted versions with increased carrier-to-noise ratios, which means we have data for 200 distorted and clear signals. In our project, we generate data using Python functions, which allow the signal to be modified with various types of digital modulation.

In our project we create the data by a programmed python functions, which able the modulation of signal with different digital modulation types.

```
1  def complex_signal(z):
2      #complex_signal empty array for stack complex symbols on it
3      complex_signals = np.full([0, 2], 0, dtype = np.complex64)
4      for it in range(z.size):
5          for ir in range(z.size):
6              Signal = np.array([z[it], z[ir]]).reshape(1, -1)
7
8              complex_signals = np.concatenate((complex_signals, Signal), axis = 0)
9      return complex_signals
10
11 def add_nosie(SR, ER, size, Signal):
12     # SR The start $\frac{C}{N}$ ratio,  ER the end $\frac{C}{N}$ ratio
13     Power = Signal.var() # calculate the power of signal
14     noise_stack = np.full([0, Signal.shape[1]], 0) #empty array for stack noise on it
15     for snr in range(SR, ER):
16         sttd = np.sqrt(Power * (10**(-snr/10))) # calculate the standard deviation of random signal
17         noise = np.random.normal(0, sttd, [size, Signal.shape[1]]) # use the numpy gaussian distrubtion function
18         noise_stack = np.concatenate((noise_stack, noise), axis = 0) # concatenate the noise levels
19
20     return noise_stack + Signal # add the signal to different noise level
21
22 def digital_Modulation(z, Tb, frequency, fs):
23     # z the complex representation of signal ,Tb the bit period, fs the sampling frequency
24     time = np.arange(0, Tb*z.size, 1/fs).reshape([z.size, -1//z.size])
25     carrier  = abs(z) * np.sin(2 * np.pi * frequency * time + np.angle(z))
26
27     return carrier.reshape(1, -1)
```

The work of this function is.

- First we have the complex signal function that maps all possible states of two values from a complex constellation of digital modulations.

- The add noise function create in order noise signals with a $(\frac{C}{N})$from the start ratio to the end ratio.

- Digital modulation create a modulated signal with following complex symbols in amplitude and phase.

We begin with Binary Phase Shift Keying (BPSK), a fundamental digital modulation scheme. The corresponding data is generated to train an Adaptive Pulses Systems (APS) model for BPSK signal processing.

```
z = np.array([1+0j, -1+0j]) #binary phase shift keying symbols
Z = complex_signal(z)

tb = 10/500 # the time period of bit
frequency = 500 # signal frequncy
fs = frequency * 10 # sampling frequency take five times

# create empty array for the noised signals
x = np.full([0, 200], 0, dtype = np.float64)# we make 200 columns for 200 signal sample
# create empty array for the clear signals
y = np.full([0, 200], 0, dtype = np.float64)

for it in range(Z.shape[0]):
    carrier = digital_Modulation(Z[it, :], Tb = tb, frequency = frequency, fs = fs)
    noised_signal = add_nosie(-5, 5, size = 5, Signal = carrier)

    x = np.concatenate((x, noised_signal), axis = 0)

    carrier = np.ones([noised_signal.shape[0], 1]).dot(carrier)
    y = np.concatenate((y, carrier), axis = 0)

```

fig. 3.2 represent two noised signal taken from x and his similar at row in y.

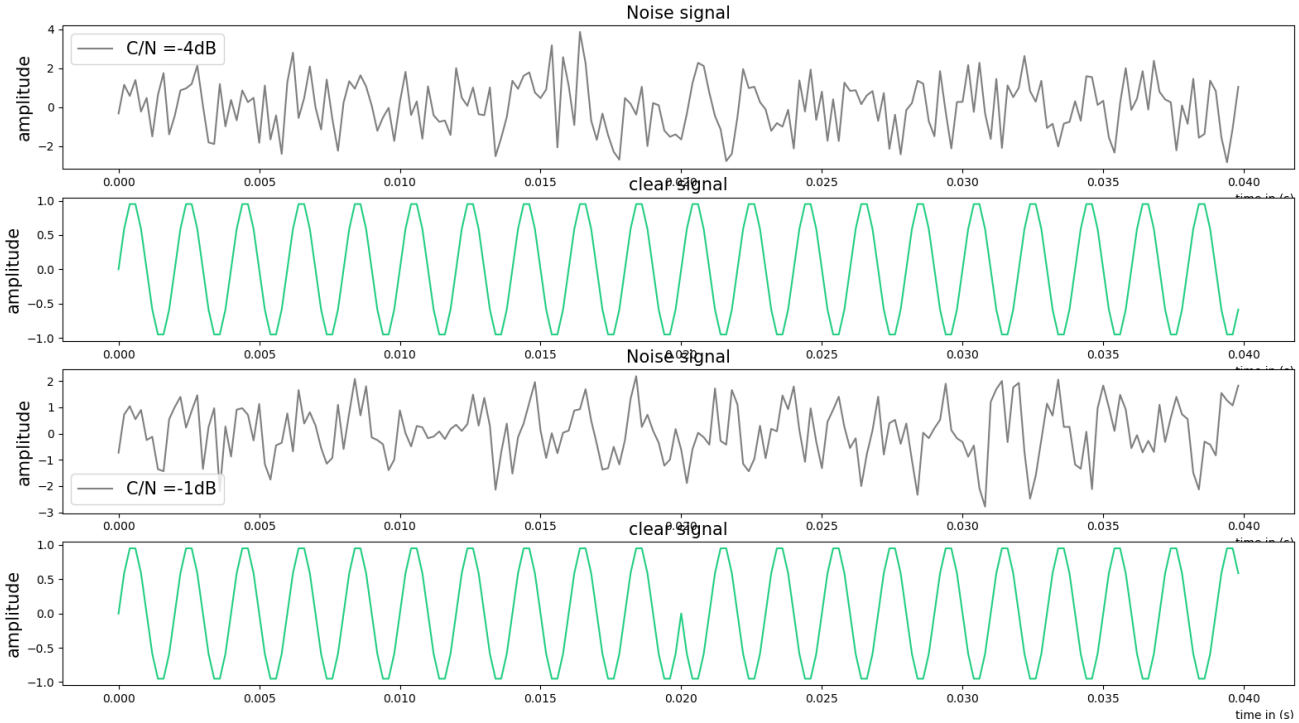

Figure 3.2: Two taken signals from x and y

Now we create two hundred signal. The preprocessing of data consist of.

- Splits the data into train set and test sets.
- Normalization of data.
- Transform it to a tensor with three dimension for the model.

```
# sklearn a library allow the splits of data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 66/x.shape[0],
                                                    shuffle = True)

# normalization of data
x_test = ((x_test)/1).reshape([x_test.shape[0], x_test.shape[1], 1])
x_train = ((x_train/1)).reshape([x_train.shape[0], x_train.shape[1], 1])
y_test = ((y_test )/1).reshape([y_test.shape[0], y_test.shape[1], 1])
y_train = ((y_train)/1).reshape([y_train.shape[0], y_train.shape[1], 1])

```

One system construction

Using the following code, we create and train a single system model, called Model (B), to process the binary phase shift key modulation signals. The model (B) system has length of 100 sample train as a 'multi infinite systems (mis)'.

```
path = np.array([0])
n_sampel = np.array([100])
configuration = np.array([1])
model = Adaptive_systems(configuration, n_sample, path)
model.construct()
#%%
model.fit(x_train, y_train, tdrb = 100, learning_rate = 2e-4, systems = 'mis')
loss = model.score()
system = model.System()
predicted_signal = model.predict(x_test)
```

After the training we can easy visualizing the loss curve Figure 3.3 represent the descent of loss during learning.

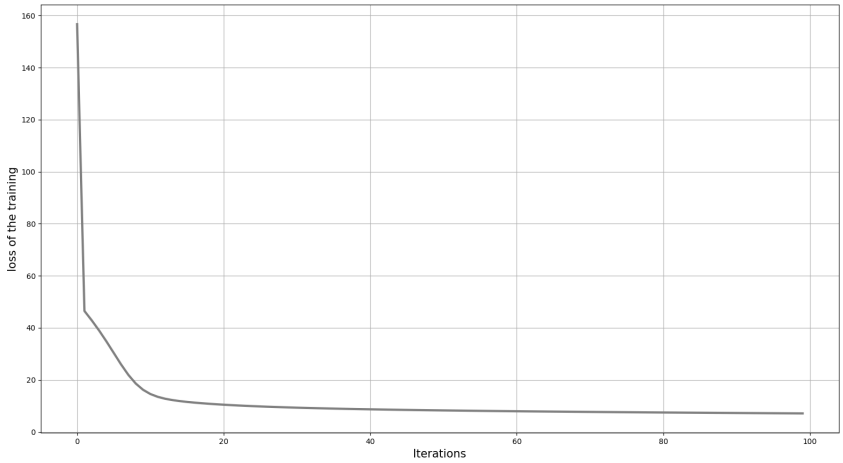

Figure 3.3: Loss evaluation during the learning

The next major step after training is model testing, which can be easily done using the "predict" method. The predicted (noise-free) data signals are shown in Figures 3.4 and 3.5, which represent the result of removing noise from the test data.

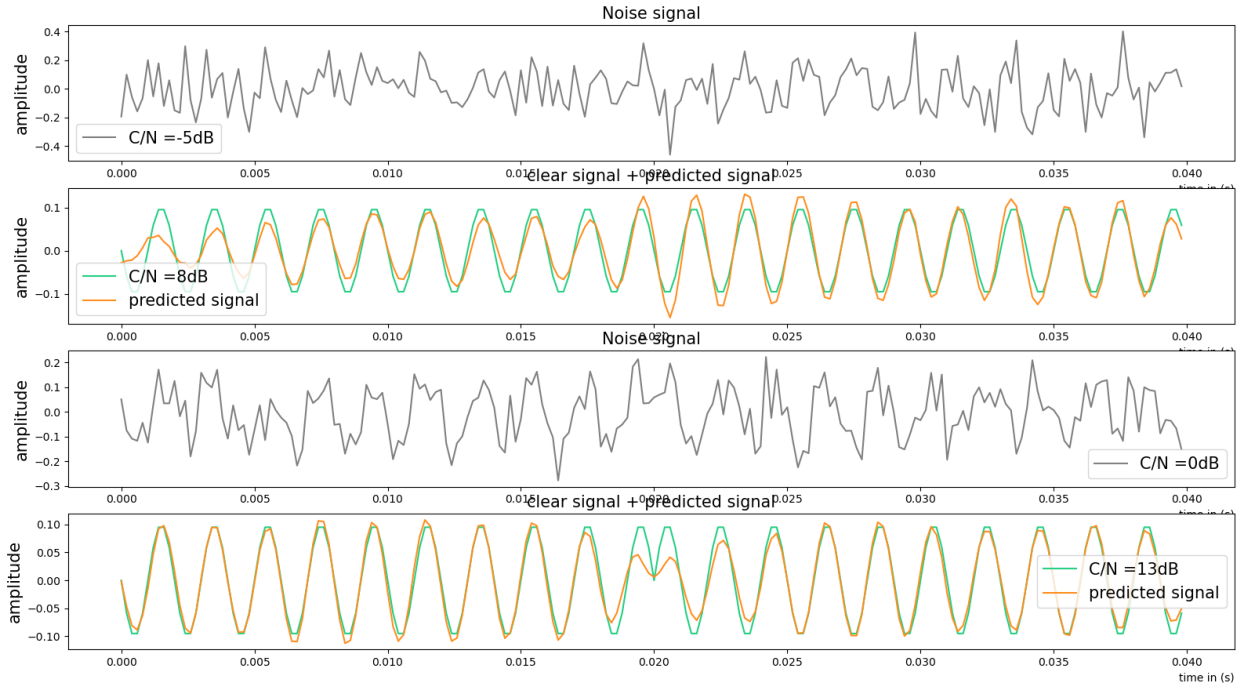

Figure 3.4: A comparison between the noised and the processed signals for the B-model

Note that for the first input noise signal in Figure 3.4, the carrier-to-noise ratio was -5 dB, meaning that the noise power was actually greater than the signal power. This is a very poor signal condition, where it is difficult or impossible to extract data reliably. The out processed signal in other hand, has a carrier to noise ratio of 8 dB, means the signal power is now about 6.3 times greater than the noise power. This is a significant improvement and usually sufficient for reliable performance.

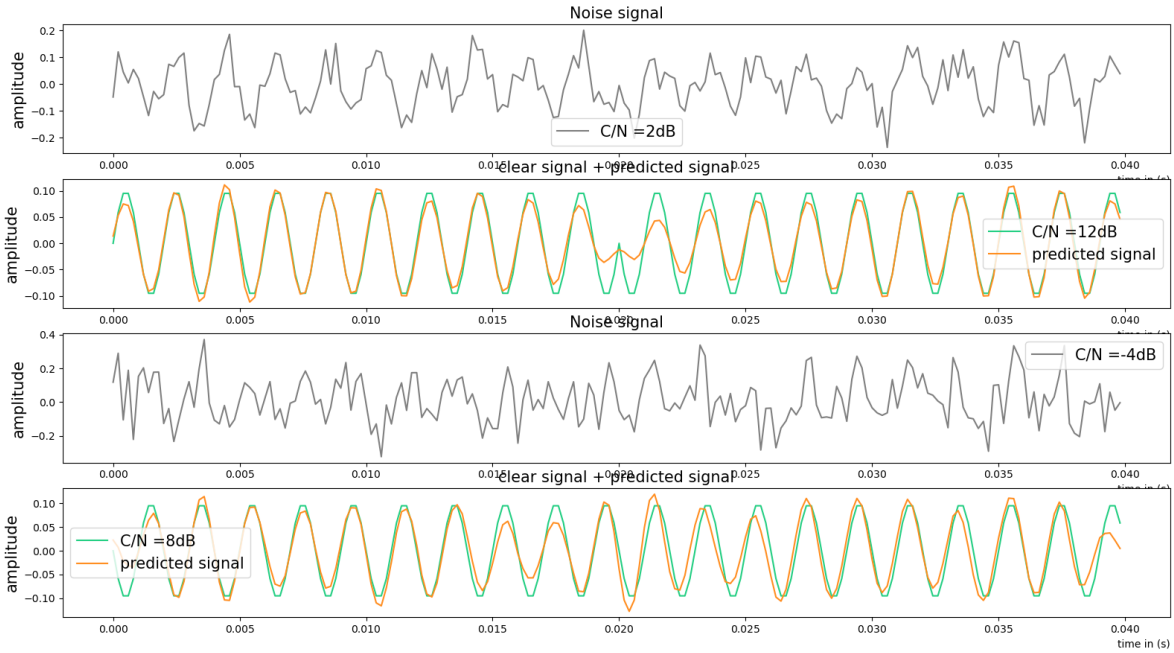

Figure 3.5: A comparison between the noised and the processed signals for the B-model

The table 3.1 combines the results obtained in Figures 3.4 and 3.5, the carrier to noise ratio of input signals and processed signals and the obtained gain.

Signal	Input signal (C/N)	processed signal (C/N)	gain
1	- 5 dB	8 dB	13 dB
2	0 dB	13 dB	13 dB
3	2 dB	12 dB	10 dB
4	- 4 dB	8 dB	12 dB

Table 3.1: A comparison in (c/n) between the input signals and processed signals

The results achieved by Model (B) are impressive, as it can distinguish between different cases. For example, in the first visualization, the model clearly distinguishes between the first signal without any phase shift and the second case with a phase shift. The model Gain is the difference between the input carrier to noise ratio and the output carrier to noise ratio. The average gain achieved by the model B from all the signals prepared for testing is [11 dB] which is an impressive result.

Multi systems construction

To increase profitability and achieve better results, it is necessary to train a multi-layer model, enabling the deep learning process. To achieve this, we will train a three-layer (G) model, with two systems in the first and second layers each having 100 samples, while one system in the third layer consists of a single sample.

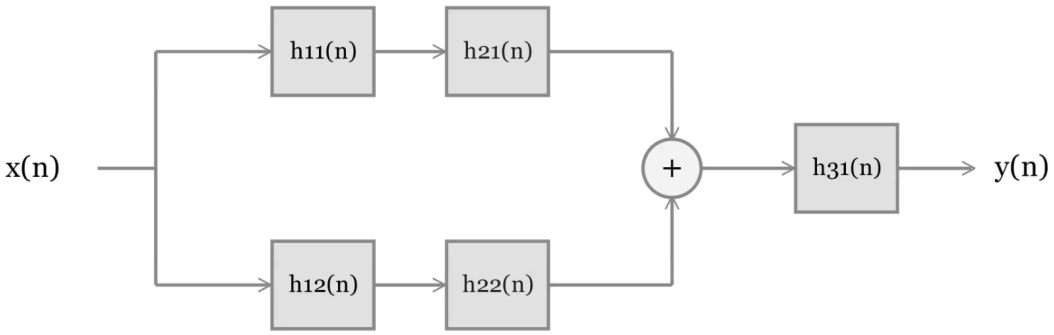

Figure 3.6: Model (G) blocks diagram

To create and train the **G-model**, the following lines of code are necessary.

```
path = np.array([0, 1, 0])
n_sample = np.array([100, 100, 1])
configuration = np.array([2, 2, 1])
model = Adaptive_systems(configuration, n_sample, path)
model.construct()
#%%
model.fit(x_train, y_train, tdrb = 400, learning_rate = 2e-4, systems = 'mis')
loss = model.score()
system = model.System()
predicted_signal = model.predict(x_test)
```

The learning of multi systems simultaneously require more time then one system for achieving best score. Figure 3.7 represent the loss descent on the training of the G-model.

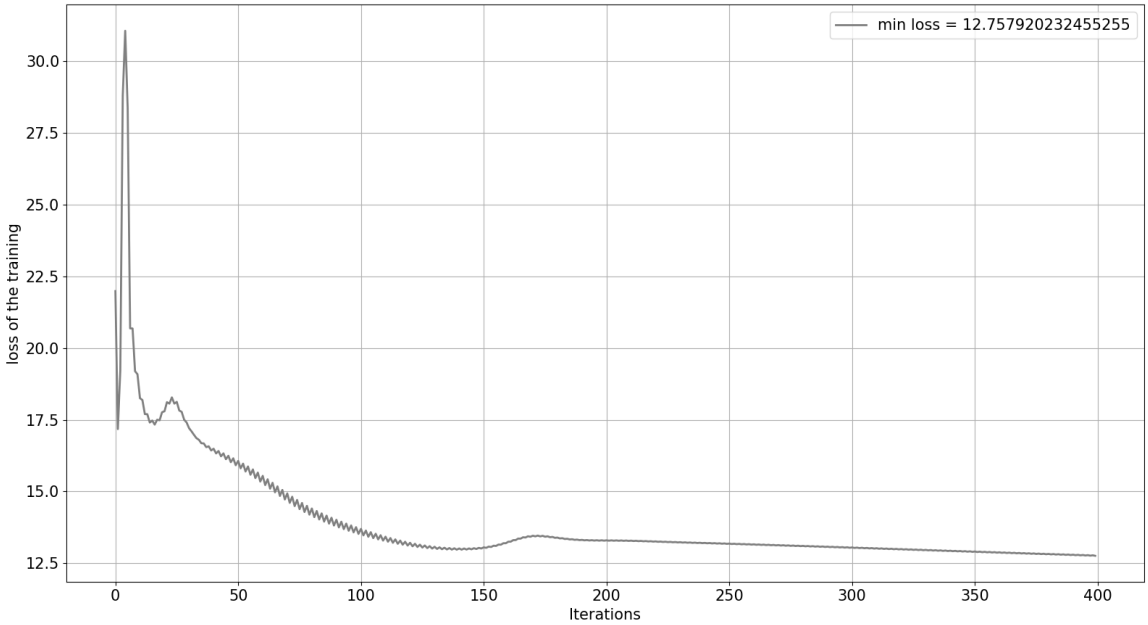

Figure 3.7: Model (G) loss evaluation during the learning

After the training phase, we can first visualize the impulse response of the model systems using the "Systems" method. Figure 3.8 shows the impulse response systems for the first and second layers, while the third layer system contains a single sample, which is considered a unit impulse.

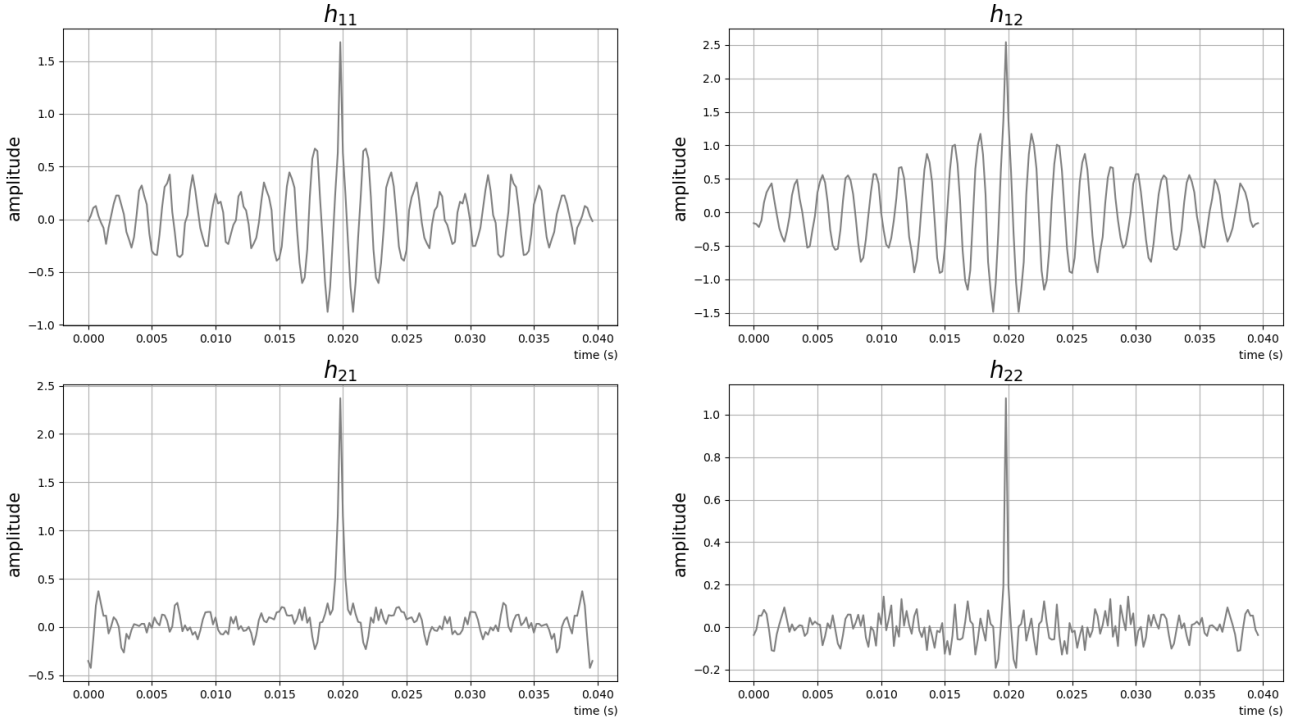

Figure 3.8: Impulse response of systems layer one and two in model (G)

It is necessary to test the model. This can be done by applying Model (G) to the test data using the "predict" method. Figures 3.9 and 3.10 show the results of processing the noise test signals using Model (G).

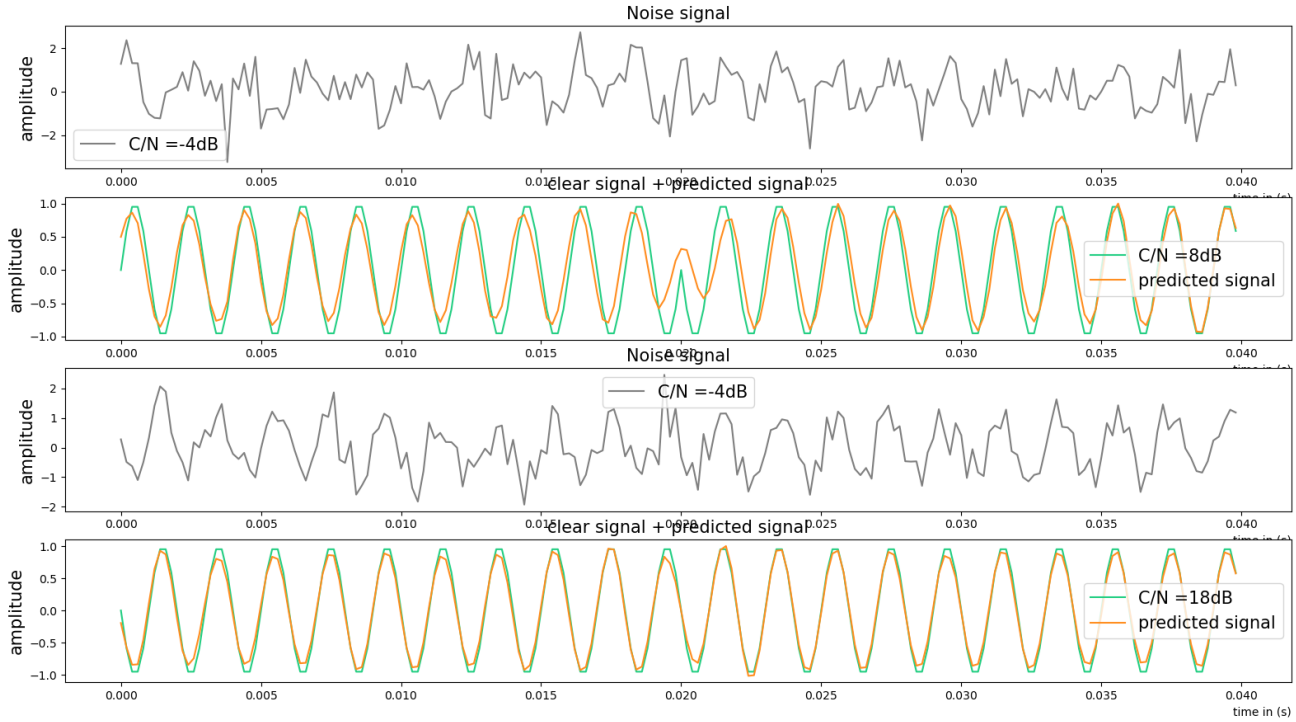

Figure 3.9: A comparison between the noised and the processed signals for the G-model

Note that Model (G) recognized the phase shift well in the first case and processed the signal efficiently, with the carrier-to-noise ratio ranging from -4 to 8 dB, achieving a gain of 12 dB. In the second case, the model recognized no phase shift and processed the signal. Consequently, the carrier-to-noise ratio ranged from -4 to 18 dB, achieving a gain of 22 dB. This results are impressive and live up the expectation from a multi systems model.

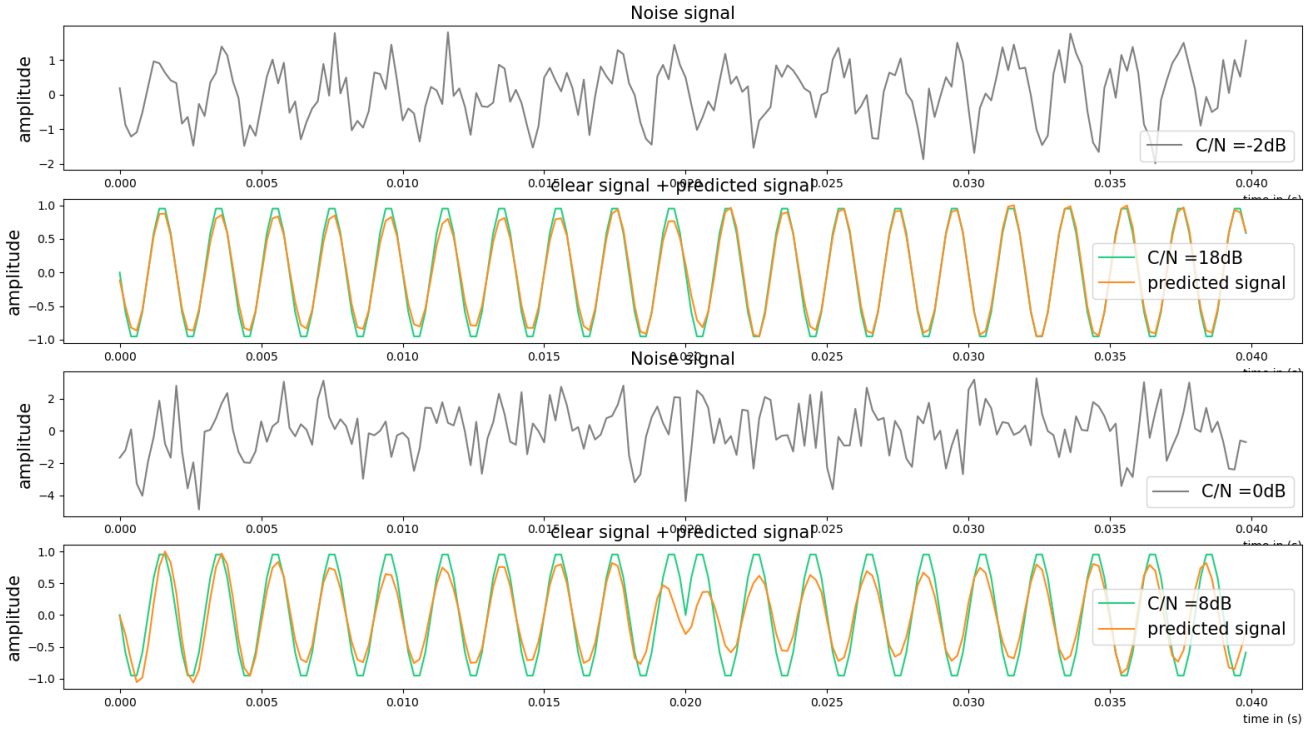

Figure 3.10: A comparison between the noised and the processed signals for the G-model

The table 3.2 show the carrier to noise ratio of the Input signal and processed signals and the obtained gain in each case.

Signal	Input signal (C/N)	processed signal (C/N)	gain
1	-4 dB	8 dB	12 dB
2	-4 dB	18 dB	22 dB
3	-2 dB	18 dB	20 dB
4	0 dB	8 dB	8 dB

Table 3.2: A comparison in (c/n) between the input signals and processed signals

The results achieved by Model G are impressive and reflect the high performance of the system, achieving a gain of 20 dB, a 100-fold increase in the required carrier signal power relative to noise, representing a significant enhancement in signal quality.

Models comparison

It is essential to compare the two models to determine which performs better. By taking the processed signals from both models and calculating the carrier-to-noise ratio for different scenarios, we obtain Table 3.3, a statistical comparison showing the results achieved by the two models.

Model	maximum $\frac{C}{N}$ (dB)	minimum $\frac{C}{N}$ (dB)	average $\frac{C}{N}$ (dB)	average gain (dB)
G	19.23	6.32	12.92	18.52
B	17.84	5.14	10.93	13.11

Table 3.3: Statical comparison between model G and model B

There is no disagreement that both models were able to identify the presence of phase shift or not in difficult cases and process the signal accordingly, but the difference between the two models appeared in the ability to reconstruct and maintain the original shape of the signal, as Model (G) excelled in the processing process by reaching a gain exceeding 20 dB in some cases, which was not achieved by Model (B). This level of gain reflects a high-performance system with a high ability to preserve the original signal shape and reduce noise levels.

3.5 Conclusion

This chapter introduced the complete implementation framework for Adaptive Pulse Systems (APS), detailing the Python-based architecture that enables flexible construction of different model configurations. Using object-oriented programming principles, we developed the Adaptive Systems core class, integrating: (1) system initialization parameters (configuration, n-sample, path), (2) forward convolution operations via forwardBackConv2D, (3) gradient computation via the novel Concatenate Zero algorithm, and (4) two different training modes (MFS for multi finite systems using Lorentzian MSE and MIS for multi infinte systems using correlation coefficient optimization). Experimental has demonstrated the effectiveness of the framework, with Model B (one system) achieving an average noise reduction gain of 13.11 dB for BPSK signals, while the more complex Model (G) show a better performance (an average gain of 12.92 dB) despite requiring significantly longer training. These results highlight several key insights: the importance of system architecture design (as evidenced by the role of the path parameter in cascaded communications), the computational trade-offs between model complexity and performance gains, and the framework's inherent scalability, demonstrated by successful operation across various configurations. The implementation specifically addresses real-world signal processing challenges through features such as automatic path detection for cascaded systems and memory-optimized gradient propagation. While the current implementation focuses on BPSK noise reduction, the modular design—particularly the separation of system construction (generate), training (fit), and evaluation (predict) methods—provides a versatile foundation for extension to other modulation schemes and signal processing applications. Future work could explore: (1) hybrid architectures that combine the current approach with deep learning techniques, (2) hardware acceleration for immediate deployment, and (3) theoretical analysis of convergence properties under different noise conditions, based on the practical implementation framework defined in this chapter.

4 Conclusion and Perspectives

This thesis has presented a comprehensive study on the design, implementation, and evaluation of Adaptive Pulse Systems (APS) for noise reduction in digital communication systems, with a particular focus on Binary Phase Shift Keying (BPSK) modulation. By bridging theoretical signal processing principles with modern machine learning methodologies, a flexible and modular framework was developed to enhance signal quality under challenging noise conditions.

At the theoretical level, the work introduced a forward-backward convolution model enabling zero-phase response, alongside custom loss functions tailored for both Finite and Infinite Impulse Response systems. Furthermore, the Concatenate Zero Algorithm was proposed to streamline gradient computation in multi-layer adaptive architectures.

From a practical standpoint, a Python-based object-oriented system was implemented, allowing for scalable experimentation with different APS topologies. The results demonstrated notable improvements, including an average 11 dB gain in carrier-to-noise ratio for BPSK signals, and validated the system's robustness in low-SNR environments. The analysis also highlighted important trade-offs between performance and computational cost, providing insights into the design of efficient filtering systems.

This work has also served as a formative experience, consolidating knowledge in digital signal processing, optimization, and software engineering. Through hands-on implementation, debugging, and iterative validation, the project fostered a deepened understanding of both theoretical and applied aspects of adaptive filtering.

Looking ahead, this work opens several promising avenues for further research and practical applications. These future directions aim to enhance the model's performance, adaptability, and real-world deployment potential:

- Integration with deep learning architectures
- Support for multi-modulation formats, such as:
  - Quadrature Amplitude Modulation (QAM)
  - Orthogonal Frequency Division Multiplexing (OFDM)
- Deployment in non-communication domains, including:
  - Biomedical signal processing
  - Seismic signal processing
- Real-time implementation and hardware acceleration, using:
  - Field-Programmable Gate Arrays (FPGA)
  - Compute Unified Device Architecture (CUDA)
  - Application-Specific Integrated Circuits (ASIC)

An exciting future direction is adapting the APS framework for drone sound detection. Like BPSK signals, drone acoustics exhibit harmonic patterns that can be treated as modulated signals. APS's 11 dB C/N improvement and phase-preserving convolution structure make it well-suited for isolating weak drone sounds from environmental noise. The multi-layered Model H could serve as a multi-band acoustic filter, while the Concatenate Zero Algorithm may assist in spectrogram-based feature extraction. With suitable adjustments—such as using Mel spectrograms, spectral contrast losses, and attention mechanisms—the framework could support real-time, GPU-accelerated drone detection for security and surveillance applications.

In conclusion, the APS framework developed in this thesis contributes a novel approach to adaptive signal filtering, demonstrating how rigorous theoretical development can be effectively translated into practical engineering solutions. This work lays a solid foundation for further exploration at the intersection of algorithm design and system-level implementation, and reflects the enduring value of learning through building—a principle that will continue to guide future academic and professional pursuits.

Bibliography

[1] Ippolito, L. J. (2008). *Satellite Communications Systems Engineering: Atmospheric Effects, Satellite Link Design and System Performance*. Wiley.

[2] Bhatia, V., & Jain, V. (2016). Study and Analysis of various Modulation Techniques and their Performance. ResearchGate. https://www.researchgate.net/publication/299166141

[3] Satellite Communication. WATElectronics.com. https://www.watelectronics.com/satellite-communication/

[4] Mohammed, A. A., & Abdul-Ghafour, H. F. (2018). Multi-Path Fading Mitigation Based on Optimal Channel Equalization and Diversity Techniques. http://article.sapub.org/10.5923.j.ijnc.20180805.01.html

[5] Colors of noise. Wikipedia, Wikimedia Foundation. https://en.wikipedia.org/wiki/Colors_of_noise

[6] Noise (signal processing). Wikipedia, Wikimedia Foundation. https://en.wikipedia.org/wiki/Noise_(signal_processing)

[7] Rafi, S. (2023, October 25). The Stellar Journey of Satellite-Based Navigation and GNSS. Medium. https://suhailsh7.medium.com/the-stellar-journey-of-satellite-based-navigation-and-gnss-2231694205c1

[8] DVB-S2X Highlights – White Paper. DVB, March 2015. https://dvb.org/wp-content/uploads/2020/01/a172_dvb-s2x

[9] Quadrature amplitude modulation. Wikipedia, Wikimedia Foundation. https://en.wikipedia.org/wiki/Quadrature_amplitude_modulation

[10] ETSI EN 302 307 V1.2.1. (2009-08). *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*. ETSI, Sophia Antipolis.

[11] Pratt, T., Bostian, C., and Allnutt, J. (2021). *Satellite Communications* (2nd ed.). Wiley.

[12] Rappaport, T. S. (2014). *Wireless Communications: Principles and Practice* (2nd ed.). Pearson Education.

[13] Sklar, B. (2021). *Digital Communications: Fundamentals and Applications* (2nd ed.). Pearson Education.

[14] Ippolito, L. J. (2008). *Satellite Communications Systems Engineering: Atmospheric Effects, Satellite Link Design and System Performance*. Wiley-Interscience.

[15] Goldsmith, A. (2005). *Wireless Communications*. Cambridge University Press.

[16] Madhow, U. (2008). *Fundamentals of Digital Communication*. Cambridge University Press.

[17] Proakis, J. G., and Salehi, M. (2007). *Digital Communications* (5th ed.). McGraw-Hill.

[18] Parsons, J. D. (2000). *The Mobile Radio Propagation Channel* (2nd ed.). Wiley.

[19] Hashemi, H. (1993). *The Indoor Radio Propagation Channel*. Proceedings of the IEEE, 81(7), 943–968.

[20] Stuber, G. L. (2017). *Principles of Mobile Communication* (4th ed.). Springer.

[21] Tsoulos, G. V. (2001). *Mobile Antenna Systems Handbook*. CRC Press.

[22] Dahlman, E., Parkvall, S., and Skold, J. (2021). *5G NR: The Next Generation Wireless Access Technology* (2nd ed.). Academic Press.

[23] Lutz, E., Werner, M., and Jahn, A. (1999). *Satellite Systems for Personal and Broadband Communications*. Springer.

[24] Eldar, Y. C. (2022). *Sampling Theory: Beyond Bandlimited Systems*. Cambridge University Press.