ALEXANDRIA UNIVERSITY
FACULTY OF ENGINEERING
Department of Communication & Electronics Engineering

5G PDSCH Simulator and DRL Approach For Designing The Optimal Precoder Vector and Power Allocation in MU-MIMO

By
Ahmed Hany
Ahmed Al-Alwani
Abulrahman Muhammad Itman
Muhammad Emad
Momen Ashraf
Khalid Diab
Muhammad Abu-Elmagd
Muhammad Nagy

Supervised By
Dr. Muhammad Karmoose

A Graduation Project submitted to the Department of Communication & Electronics in partial fulfillment of the requirements for the degree of B.Sc. in Communication Engineering.

Alexandria
July 2023

Contents

List of Figures

Chapter 1

Overview

1.1 The mobile industry in numbers

Mobile penetration is approaching saturation in most markets around the world, especially among adult and urban populations. In every region, the majority of new subscribers will be young consumers and rural dwellers. Despite increasing saturation in developed regions, there is still room for growth in many large, underpenetrated markets in developing regions. For example, India and Sub-Saharan Africa will account for around half of new mobile subscribers globally over the 2022–2030 period "Globally, there were 4.4 billion mobile internet users in 2022, equivalent to 55The mobile internet usage gap has narrowed markedly in the last five years – from 50% in 2017 to 41% in 2022 on average – as more people around the world rely on the internet for many daily activities, especially in the wake of the Covid-19 pandemic.

1.1.1 5G in the mobile market

"5G will overtake 4G in 2029 to become the dominant mobile technology by the end of this decade" 5G adoption continues to rise due to new network deployments and cheaper devices. As of January 2023, there were 229 commercial 5G networks around the world and over 700 5G smartphone models had been launched, including more than 200 in 2022. The number of connections on legacy networks (2G and 3G) will continue to decline in the coming years as users migrate to 4G and 5G, resulting in more network shutdowns. To date, operators have announced plans to shut down 96 2G networks and 107 3G networks around the world.

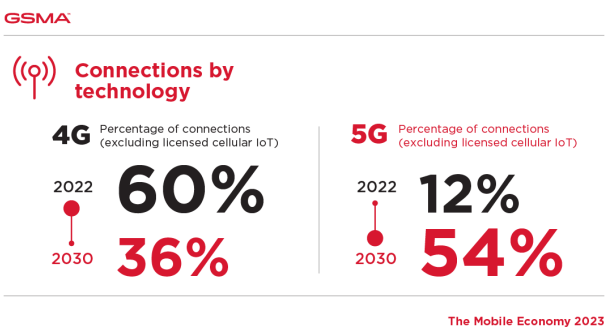


Figure 1.1: 4G vs 5G connections

1.2 The next generation – 5G NR

Despite LTE being a very capable technology, there are requirements not possible to meet with LTE or its evolution. Furthermore, technology development over the more than 10 years that have passed since the work on LTE was initiated allows for more advanced technical solutions. To meet these requirements and to exploit the potential of new technologies, 3GPP initiated the development of a new radio access technology known as NR (New Radio). A workshop setting the scope was held in the fall of 2015 and technical work began in the spring of 2016. The first version of the NR specifications was available by the end of 2017 to meet commercial requirements on early 5G deployments already in 2018. NR reuses many of the structures and features of LTE. However, being a new radio-access technology means that NR, unlike the LTE evolution, is not restricted by a need to retain backwards compatibility. The requirements on NR are also broader than what was the case for LTE, motivating a partly different set of technical solutions.

1.2.1 5G Use Cases

In the context of 5G, one is often talking about three distinctive classes of use cases: enhanced mobile broadband (eMBB), massive machine-type communication (mMTC), and ultra-reliable and low-latency communication (URLLC)

- eMBB corresponds to a more or less straight forward evolution of the mobile broadband services of today, enabling even larger data volumes and further enhanced user experience, for example, by supporting even higher end-user data rates.
- mMTC corresponds to services that are characterized by a massive number of devices, for example, remote sensors, actuators, and monitoring of various equipment. Key requirements for such services include very low device cost and very low device energy consumption, allowing for very long device battery life of up to at least several years. Typically, each device consumes and generates only a relatively small amount of data, that is, support for high data rates is of less importance.
- URLLC type-of-services are envisioned to require very low latency and extremely high reliability. Examples hereof are traffic safety, automatic control, and factory automation.

It is important to understand that the classification of 5G use cases into these three distinctive classes is somewhat artificial, primarily aiming to simplify the definition of requirements for the technology specification. There will be many use cases that do not fit exactly into one of these classes. Just as an example, there may be services that require very high reliability but for which the latency requirements are not that critical. Similarly, there may be use cases requiring devices of very low cost but where the possibility for very long device battery life may be less important.

1.2.2 Waveform, Numerology, and Frame Structure

The choice of radio waveform is the core physical layer decision for any wireless access technology. After assessments of all the waveform proposals, 3GPP agreed to adopt orthogonal frequency division multiplexing (OFDM) with a cyclic-prefix (CP) for both DL and UL

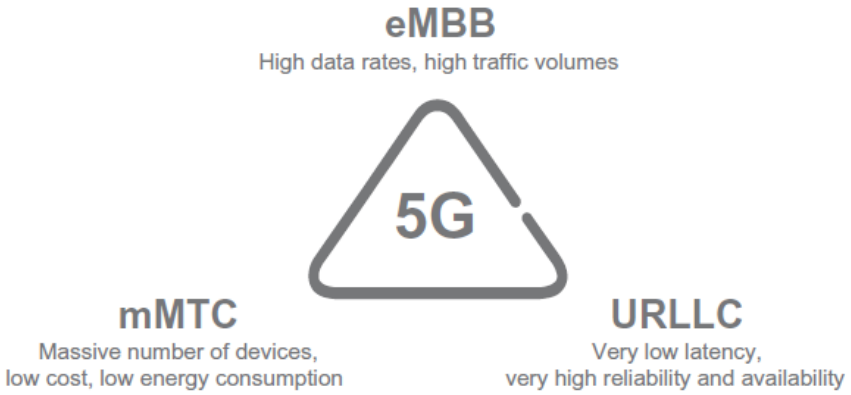


Figure 1.2: 5G Use cases classification

transmissions. CP-OFDM can enable low implementation complexity and low cost for wide bandwidth operations and multiple-input multiple-output (MIMO) technologies. NR supports operation in the spectrum ranging from sub-1 GHz to millimeter wave bands. Two frequency ranges (FR) are defined in Release 15:

- FR1: 450 MHz – 6 GHz, commonly referred to as sub-6 GHz.
- FR2: 24.25 GHz – 52.6 GHz, commonly referred to as millimeter wave.

Scalable numerologies are key to support NR deployment in such a wide range of spectrum. NR adopts flexible sub-carrier spacing of $2^{\alpha} \times 15$ kHz ($\alpha = 0, 1, ..., 4$) scaled from the basic 15 kHz sub-carrier spacing in LTE. Accordingly, the CP is scaled down by a factor of 2^{α} from the LTE CP length of 4.7 μs. This scalable design allows support for a wide range of deployment scenarios and carrier frequencies. At lower frequencies, below 6 GHz, cells can be larger and sub-carrier spacings of 15 kHz and 30 kHz are suitable. At higher frequencies, cells and delay spread are typically smaller and the CP lengths provided by the 60 and 120 kHz numerologies are sufficient.

A frame has a duration of 10ms and consists of 10 subframes. This is the same as in LTE, facilitating NR and LTE coexistence. Each subframe consists of 2^{α} slots of 14 OFDM symbols each. Although a slot is a typical unit for transmission upon which scheduling operates, NR enables transmission to start at any OFDM symbol and last only as many symbols as needed for the communication. This type of “mini-slot” transmission can thus facilitate very low latency for critical data as well as minimize interference to other links per the lean carrier design principle that aims at minimizing transmissions. Latency optimization has been an important consideration in NR. Many other tools besides “mini-slot” transmission have been introduced in NR to reduce latency.

1.3 5G Data Channels: Logical, Transport & Physical

In order to be able to carry the data across the 5G radio access network, the data and information is organized into a number of data channels. By organizing the data into various channels the 5G communications system is able to manage the data transfers in an orderly fashion and the system is able to understand what data is arriving and hence it is able to process it in the

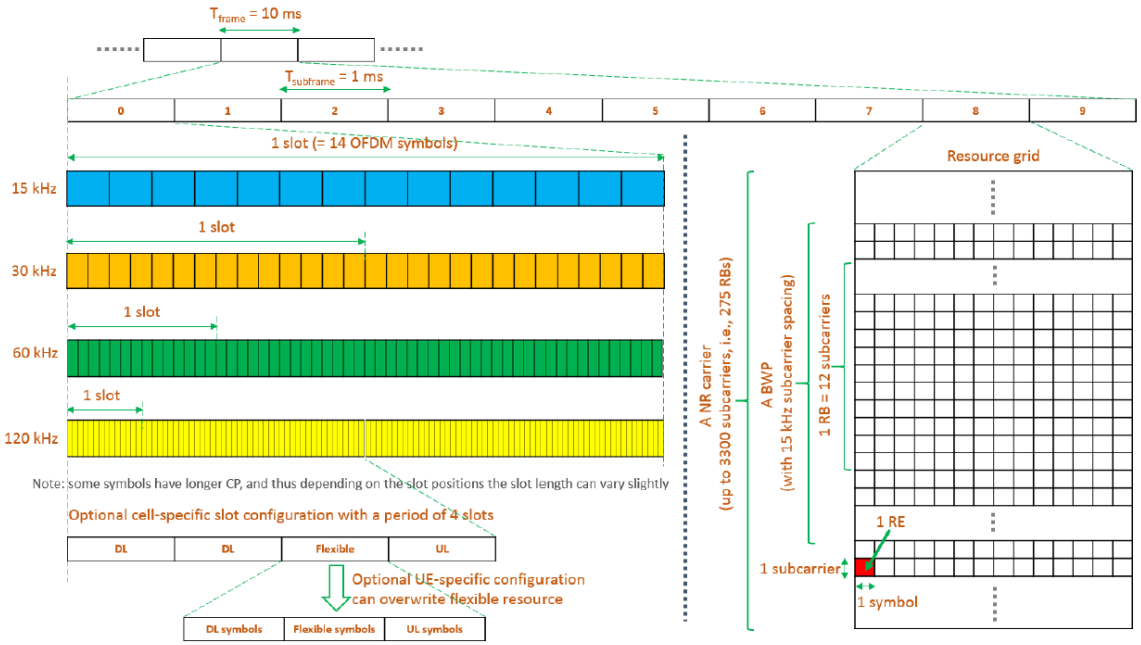


Figure 1.3: 5G Frame structure

required fashion. As there are many different types of data that need to be transferred - user data obviously needs to be transferred, but so does control information to manage the radio communications link, as well as data to provide synchronization, access and the like. All of these functions are essential and require the transfer of data over the radio access network. The 5G mobile or wireless communications system uses a similar access stratum to that used by 4G LTE. Although there are two protocol stacks: user plane and control plane, they still adopt the familiar OSI reference model. As a result there are various protocol layers and accordingly there are several data channel layers that are defined for the radio communications. In order to group the data to be sent over the 5G NR radio access network, the data is organised in a very logical way. As there are many different functions for the date being sent over the radio communications link, they need to be clearly marked and have defined positions and formats.To ensure this happens, there are several different forms of data "channel" that are used. The higher level ones are "mapped" or contained within others until finally at the physical level, the channel contains data from higher level channels. In this way there is a logical and manageable flow of data from the higher levels of the protocol stack down to the physical layer. There are three main types of data channels that are used within mobile communications systems. This is true for 5G systems, and accordingly the hierarchy is given below.

- Logical channel: Logical channels can be one of two groups: control channels and traffic channels:
  - Control channels: The control channels are used for the transfer of data from the control plane
  - Traffic channels: The traffic logical channels are used for the transfer of user plane data.
- Transport channel: Is the multiplexing of the logical data to be transported by the physical layer and its channels over the radio interface.
- Physical channel : The physical channels are those which are closest to the actual transmission of the data over the radio access network / 5G RF signal. They are used to

carry the data over the radio interface.

The physical channels often have higher level channels mapped onto them of provide a specific service. Additionally, the physical channels carry payload data or details of specific data transmission characteristics like modulation, reference signal multiplexing, transmit power, RF resources, etc.

1.3.1 5G NR logical channels

There are several different logical channels that are used within the 5G NR radio access network. Some of them will be familiar names from the 4G LTE system as the names have been carried over.

- **Broadcast Control Channel (BCCH)**: The BCCH is used within the downlink, and it is used for sending broadcast style information to the UE within that cell. The system information transmitted by the 5G NR BCCH is divided into different blocks:
  - Master Information Block (MIB): There is one MIB and this is mapped onto the BCH transport channel and then to the PBCH physical channel.
  - System Information Block (SIB): There are several system information blocks, SIBs. These are mapped onto the DL-SCH transport channel and then onto the PDSCH physical channel.
- **Paging Control Channel (PCCH)**: This is a Downlink channel. It is used to page the UEs whose location at cell level is not known to the network. As a result the paging message needs to be transmitted in multiple cells. The PCCH is mapped to the PCH transport channel and then to the PDSCH physical channel.
- **Common Control Channel (CCCH)**: This 5G channel is used on both the downlink and uplink for transmitting control information to and from the user equipments or mobiles. The channel is used for initial access, i.e. those mobiles that do not have a radio resource control, RRC connection.
- **Dedicated Control Channel (DCCH)**: The DCCH is used within the uplink and downlink to carry dedicated control information between the UE or mobile and the network. It is used by the UE and the network after a radio resource control, RRC connection has been established.
- **Dedicated Traffic Channel (DTCH)**: This 5G channel is present in both the uplink and downlink. It is dedicated to one UE and is used for carrying user information to and from a specific UE and the network.

1.3.2 5G NR transport channels

There are *five different transport channels.* Some are used on the uplink, others on the downlink, and some can be used on both.

- **Broadcast Channel (BCH)**: The BCH 5G channel is used in the downlink only for transmitting the BCCH system information and specifically the Master Information Block, MIB, information. In order that the data can be utilised, it has a specific format.

- **Paging Channel (PCH)**: The PCH is used for carrying paging information from the PCCH logical channel. The PCH supports discontinuous reception, DRX, to enable the UE to save battery power by waking up at a specific time to receive the PCH.
  In order that the PCH is received by all mobiles / UEs in the cell, the PCH must be broadcast over the entire cell as a single message, or where beam forming is used, this can be done using several different PCH instances.

- **Downlink Shared Channel (DL-SCH)**: As the name indicates, this is a downlink only channel. It is the main transport channel used for transmitting downlink data and it supports all the key 5G NR features. These include: dynamic rate adaptation; HARQ, channel aware scheduling, and spatial multiplexing.
  The DL-SCH is also used for transmitting some parts of the BCCH system information, specifically the SIB. Each UE has a DL-SCH for each cell it is connected to.

- **Uplink Shared Channel (UL-SCH)**: This is the uplink counterpart to the DLSCH that is, the uplink transport channel used for transmission of uplink data.

- **Random-Access Channel (RACH)**: The RACH is a transport channel, which carries the random access preamble which is used to overcome the message collisions that can occur when UEs access the system simultaneously.

1.3.3 5G NR physical channels

The 5G physical channels are used to transport information over the actual radio interface. They have the transport channels mapped into them, but they also include various physical layer data required for the maintenance and optimization of the radio communications link between the UE and the base station. The 5G mobile communications physical layer channels resemble those of 4G LTE, but PHICH and PCIICH have been removed. The HARQ operation has also been updated to be more flexible. Also the downlink control channel PDCCH is now administered by layer 3 procedures. There are *three physical channels* for each of the uplink and downlink

5G NR Downlink Physical Channels

Physical downlink shared channel (PDSCH): 5G NR physical downlink shared channel, PDSCH carries data sharing the capacity on a time and frequency basis. The PDSCH physical channel carries a variety of items of data: user data; UE-specific higher layer control messages mapped down from higher channels; system information blocks (SIBs) & paging.
The PDSCH uses an adaptive modulation format dependent upon the link conditions, i.e. signal to noise ratio. It also uses a flexible coding scheme. The combination of these means that there is a flexible coding and data rate.

Physical downlink control channel (PDCCH): As the name implies, 5G physical downlink control channel carries downlink control data. Its primary function is scheduling the downlink transmissions on the PDSCH and also the uplink data transmissions on the PUSCH.
The PDCCH uses QPSK as its modulation format and polar coding as the coding scheme, except for small packets of data.

Physical broadcast channel (PBCH): This 5G channel forms part of the synchronization signal block. Its function is to provide UEs with the Master Information Block, MIB. A further function of the PBCH in conjunction with the control channel is to support the synchronization of time and frequency. This aids with cell acquisition, selection and re-selection.
The PBCH uses a fixed data format and there is one block that extends over a TTI of 80ms, uses QPSK modulation and it transmits a cell specific demodulation reference signal, DMRS pattern that can be used aid with beam-forming.

5G NR Uplink Physical Channels

Physical random access channel (PRACH): This 5G channel is used for channel access. It transmits an initial random access pre-amble consisting of sequences which may be of two different lengths:

- A long sequence is 839 which is applied to the subcarrier spacings of 1.25kHz and 5 kHz
- Short sequence lengths of 139 are applied to subcarrier spacings of 15 kHz and 30 kHz (FR1 bands) and 60 kHz and 120 kHz (FR2 bands).

Physical uplink shared channel (PUSCH): The counterpart of the PDSCH. It is used to carry data from the UL-SCH and its higher mapped channels on a frequency and time-shared basis.
Like the PDSCH, The PUSCH also has a very flexible format. The allocation of frequency resources is undertaken using resource blocks along with a flexible modulation and coding scheme dependent upon the link signal to noise ratio.
To support the channel link estimation and demodulation, the PUSCH contains DMRS signals.

Physical uplink control channel (PUCCH): This carries the uplink control data. It is also possible that dependent upon the resource allocation the uplink control information or data may also be sent on the PUSCH, even though in the downlink direction, control information is always sent on the PDCCH.

The use of these 5G channels provide a method for organizing the flow of data over the radio interface of the 5G communications network. Using channels enables the communications system to recognize the type of data that is being sent, and to deal with it accordingly. The format used is very similar to that employed on 4G LTE and it built on the technology of previous mobile communications or mobile phone generations.

Figures 1.4 & 1.5 shows the mapping between logical, transport and physical channels in both DL and UL

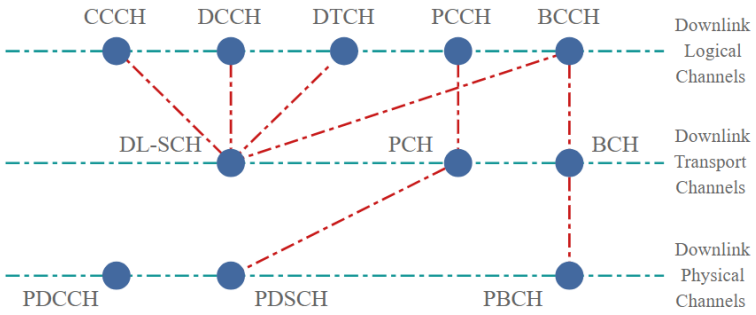


Figure 1.4: 5G Downlink logical, transport and physical channel mapping

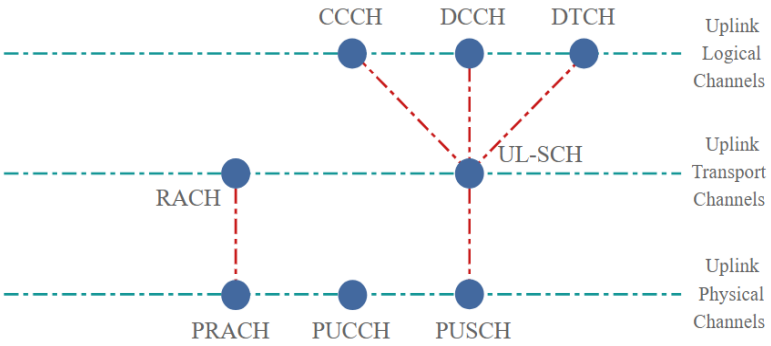


Figure 1.5: 5G Uplink logical, transport and physical channel mapping

Part I

5G PDSCH Simulator

PDSCH overview & Motivation

The Physical Downlink Shared Channel (PDSCH) is a crucial component of 5G wireless communication systems that enables high-speed data transmission with low latency. As a team, we were intrigued by the potential of 5G technology to revolutionize the way we communicate and connect with each other. We were particularly interested in the PDSCH channel, as it plays a critical role in achieving the high data rates and low latency that are essential for applications such as virtual reality, autonomous driving, and the Internet of Things (IoT).
We were also motivated by the technical challenges involved in simulating the PDSCH channel accurately. 5G is a complex and evolving technology that requires a deep understanding of wireless communication systems, signal processing, and coding theory. Our project aimed to develop a MATLAB based simulation model that could capture the unique characteristics of PDSCH and provide insights into its performance under different conditions.

In this Part, we will delve into the technical details of PDSCH and explain how our simulation model was designed and implemented. We will also discuss the results of our simulation experiments and highlight the insights we gained from them. Our hope is that this work will contribute to the ongoing efforts to improve the performance and reliability of 5G networks.
Designing and implementing a simulation model for PDSCH requires a detailed understanding of the 5G standard specifications, as well as knowledge of wireless communication systems, signal processing, and coding theory. Here is an overview of the steps we took to design and implement our simulation model:
Our simulation model for PDSCH in 5G wireless communication systems consists of several key blocks that are critical to achieving high-speed data transmission with low latency. These blocks include channel coding, MIMO setups, reference signals, and channel estimation.

The channel coding block is responsible for protecting the data from errors that may be introduced during wireless transmission. We simulated both linear block codes and LDPC coding to evaluate their performance under different conditions.
The MIMO setups block allows for multiple antennas to be used at both the transmitter and receiver, which can improve the quality and reliability of wireless communication. We simulated several MIMO setups, including SISO, SIMO, and MIMO, to gain insights into the performance of PDSCH under different conditions.
The reference signals block provides important information about the wireless channel, such as channel state information and data demodulation and decoding. We focused on two types of reference signals: CSI-RS and DMRS, which are used for channel estimation, synchronization, and data demodulation.

Finally, the channel estimation block is responsible for accurately estimating the wireless channel conditions and compensating for any distortion or interference that may be present. We spent a significant amount of time on channel estimation, which is essential for accurately simulating the performance of PDSCH under realistic conditions.
In the following sections, we will discuss each of these blocks in detail, including the technical details of how they were implemented and the results of our simulation experiments.

Chapter 2

Channel Coding

2.1 Overview

This part deals with linear block codes covering their fundamental concepts, generator and parity check matrices, error-correcting capabilities, encoding and decoding, and performance analysis. The linear block code discussed in this part is Hamming code.

2.1.1 Shannon's Noisy Channel Coding Theorem

Any channel affected by noise possesses a specific "channel capacity" C a rate of conveying information that can never be exceeded without error, but in principle, an error-correcting code always exists such that information can be transmitted at rates less than C with an arbitrarily low BER.

2.1.2 Channel Coding Principle

The channel coding principle is to add redundancy to minimize error rate as illustrated in Figure 2.1.

Figure 2.1: Illustration of the channel coding principle.

2.1.3 Channel Coding Gain

The bit error rate (BER) is the probability that a binary digit transmitted from the source received erroneously by the user. For required BER, the difference between the powers required for without and with coding is called the coding gain. A typical plot of BER versus Eb/No (bit energy to noise spectral density ratio) with and without channel coding is shown in Figure 2.2. It can be seen that coding can arrive at the same value of the BER at lower Eb/No than without coding. Thus, the channel coding yields coding gain which is usually measured in dB. Also, the coding gain usually increases with a decrease in BER.

Figure 2.2: Coding gain.

2.2 Block Codes

The data stream is broken into blocks of k bits and each k-bit block is encoded into a block of n bits with $n > k$ bits as illustrated in Figure 2.3. The n-bit block of the channel block encoder is called the code word. The code word is formed by adding $(n - k)$ parity check bits derived from the k message bits.

Figure 2.3: Coded data stream.

Properties of Block Codes

Block code rate

The block code rate (R) is defined as the ratio of k message bits and length of the code word n.

$$R = \frac{k}{n}$$

Code word weight

The weight of a code word or error pattern is the number of nonzero bits in the code word or error pattern.
For example, the weight of a code word $c = (1, 0, 0, 1, 1, 0, 1, 0)$ is 4.

Hamming distance

The Hamming distance between two blocks v and w is the number of coordinates in which the two blocks differ.

$$d_{hamming}(v, w) = d(v, w) = |\{i | v_i \neq w_i, i = 0, 1, \ldots, n-1\}|$$

Example: Consider the code words $v = (00100)$ and $w = (10010)$, then the Hamming distance $d_{hamming}(v, w) = 3$.
Hamming distance allows for a useful characterization of the error detection and error correction capabilities of a block code as a function of the code's minimum distance.

The Minimum Distance of a Block Code

The minimum distance of a block code C is the minimum Hamming distance between all distinct pairs of code words in C.

A code with minimum distance d_{min} can:

- *detect* all error patterns of weight less than or equal to $(d_{min} - 1)$.
- *correct* all error patterns of weight less than or equal to $(d_{min} - 1)/2$.

Example: Consider the binary code C composed of the following four code words.

$$C = \{(00100), (10010), (01001), (11111)\}$$

Hamming distance of (00100) and (10010) = 3
Hamming distance of (10010) and (01001) = 4
Hamming distance of (00100) and (01001) = 3
Hamming distance of (10010) and (11111) = 3
Hamming distance of (00100) and (11111) = 4
Hamming distance of (01001) and (11111) = 3
Therefore, the minimum distance $d_{min} = 3$.

2.2.1 Linear Block Codes

A block code C consisting of n-tuples $\{(c_0, c_1, ..., c_{n-1})\}$ of symbols from $GF(2)$ is said to be binary linear block code if and only if C forms a vector subspace over $GF(2)$.

Note: finite fields are also called Galois fields (GF).

The code word is said to be systematic linear code word if each of the 2^k code words is represented as linear combination of k linearly independent code words.

Linear Block Codes Properties

There are *two important properties* of linear block codes which are:

Property 1: The linear combination of any set of code words is a code word.

Property 2: The minimum distance of a linear block code is equal to the minimum weight of any nonzero word in the code.

Also, there are 2 well-known bounds on the minimum distance which are

Singleton Bound: The minimum distance of an (n, k) linear code is bounded by

$$d_{min} \leq n - k + 1 \tag{2.1}$$

Hamming Bound: An (n, k) block code can *detect* t_{ed} errors per code word and can correct up to t_{ec} errors per code word, provided that n and k satisfy the Hamming bound.

$$2^{n-k} \geq \sum_{i=0}^{t_{ec}} \binom{n}{i} \quad \text{where} \quad \binom{n}{i} = \frac{n!}{(n-1)!i!} \tag{2.2}$$

$$t_{ec} = \frac{(d_{min} - 1)}{2} \quad , \quad t_{ed} = d_{min} - 1$$

The relation is the upper bound on d_{min} and is known as *the Hamming bound.*

2.2.2 Generator & Parity Check Matrices

Let $\{g_0, g_1, \ldots, g_{k-1}\}$ be a basis of code words for the (n, k) linear block code C and $m = [m_0, m_1, \ldots, m_{k-1}]$ the message to be encoded. The Theorem that says the code word $c = (c_0, c_1, \ldots, c_{n-1})$ for the message is uniquely represented by the following linear combination of $g_0, g_1, \ldots, g_{k-1}$

$$c = m_0 g_0 + \ldots + m_{k-1} g_{k-1}$$

for every code word $c \in C$.
Since every linear combination of the basis elements must also be a code word, there is a one-to-one mapping between the set of k-bit blocks $(a_0, a_1, \ldots, a_{k-1})$ over $GF(2)$ and the code words

in C. A matrix G is constructed by taking the vectors in the basis as its rows.

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix}$$

This matrix G is a generator matrix for the code C. It can be used to directly encode k-bit blocks in the following manner:

$$mG = [m_0, m_1, \ldots, m_{k-1}] \cdot \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = m_0 g_0 + m_1 g_1 + \ldots + m_{k-1} g_{k-1} = c$$

The dual space of a linear block code C is the dual code of C and a basis $\{h_0, h_1, \ldots, h_{n-k-1}\}$ can be found for dual code of C, and the following parity check matrix can be constructed:

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{0,1} & \ldots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \ldots & h_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k-1,0} & g_{n-k-1,1} & \cdots & g_{n-k-1,n-1} \end{bmatrix}$$

In a systematic linear block code, the last k bits of the codeword are *the message bits*, that is

$$c_i = m_{i-(n-k)} \quad , \quad i = n-k, \ldots, n-1$$

While the first $n-k$ bits in the codeword are check bits generated from the k message bits according to

$$\begin{aligned} c_0 &= p_{0,0} m_0 + p_{1,0} m_1 + \ldots + p_{k-1,0} m_{k-1} \\ c_1 &= p_{0,1} m_0 + p_{1,1} m_1 + \ldots + p_{k-1,1} m_{k-1} \\ &\vdots \\ c_{n-k-1} &= p_{0,n-k-1} m_0 + p_{1,n-k-1} m_1 + \ldots + p_{k-1,n-k-1} m_{k-1} \end{aligned}$$

The above equation can be written in a matrix form as:

$$[c_0, c_1, \ldots, c_{n-1}] = [m_0, m_1, \ldots, m_{k-1}] \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} & 1000 & \cdots & 0 \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} & 0100 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0000 & \cdots & 1 \end{bmatrix}_{k \times n} \tag{2.3}$$

or

$$c = mG$$

where G is the matrix on the right hand side of the equation 2.3.
The $k \times n$ matrix G is called a generator matrix of the code and it has the following form:

$$G = [P | I_k]_{k \times n} \tag{2.4}$$

The matrix I_k is the identity matrix of order k, and P is an arbitrary $k \times n-k$ matrix. When P is specified, it defines the (n,k) block code completely. The parity check matrix H corresponding to the above generator matrix G can be obtained as

$$H = \left[I_{n-k}|P^T\right]$$

$$H = \begin{bmatrix} 1000 & \cdots & 0 & p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} \\ 0100 & \cdots & 0 & p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0000 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \tag{2.5}$$

Theorem 2.2.1: Parity Check Theorem

For any (n,k) linear block code C with $(n-k) \times n$ parity check matrix H, a code word $c \in C$ is a valid code word ***if and only if*** $cH^T = 0$.

Example:

For the following generator matrix of (7,4) block code. Find the code vector for the message vector m = (1110) and check the validity of code vector generated.

$$G = \left[\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Solution:

The code vector for the message block $m = (1110)$ is given by

$$c = mG = [1110] \cdot \left[\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] = [0101110]$$

$$H = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$$

$$cH^T = [0101110] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = [000]$$

Hence, the generated code vector is valid.

2.3 Hamming Codes

Hamming code is a linear block code capable of correcting single errors having a minimum distance $d_{min} = 3$. It is very easy to construct Hamming codes. The parity check matrix H

(obtained in equation 2.5) must be chosen so that no row in H^T is zero and the first $(n-k)$ rows of H^T form an identity matrix and all the rows are distinct. We can select $2^{n-k}-1$ distinct rows of H^T.
Since the matrix H^T has n rows, for all of them to be distinct, the following inequality should be satisfied.

$$2^{n-k} - 1 \geq n$$

Implying that

$$(n-k) \geq \log_2(n+1)$$
$$n \geq k + \log_2(n+1) \tag{2.6}$$

Hence, the minimum size n for the code words can be determined from equation 2.6.

Example: Design a Hamming code with message block size of eleven bits.
Solution: It follows from equation 2.6 that

$$n \geq 11 + \log_2(n+1)$$

The smallest n that satisfies the above inequality is 15, and hence, we need a $(15, 11)$ block code. Thus, the transpose of the parity check matrix H will be 4×15 matrix. The first four rows of H^T will be I_4 matrix. The last eleven rows are arbitrarily chosen, with the restrictions that no row is zero, and all the rows are distinct.

$$H^T = \left[\begin{array}{cccc} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \hline 0&1&0&1\\ 0&1&1&0\\ 0&1&1&1\\ 0&0&1&1\\ 1&0&0&1\\ 1&0&1&0\\ 1&0&1&1\\ 1&1&0&0\\ 1&1&0&1\\ 1&1&1&0\\ 1&1&1&1 \end{array}\right] = \left[\frac{I_{n-k}}{P^T}\right] \tag{2.7}$$

Then generator matrix G equals

$$G = \left[\begin{array}{cccccccccccccc} 0&1&0&1&1&0&0&0&0&0&0&0&0&0&0\\ 0&1&1&0&0&1&0&0&0&0&0&0&0&0&0\\ 0&1&1&1&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&1&0&0&0&0&0&0&0\\ 1&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\ 1&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\ 1&0&1&1&0&0&0&0&0&0&1&0&0&0&0\\ 1&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\ 1&1&0&1&0&0&0&0&0&0&0&0&1&0&0\\ 1&1&1&0&0&0&0&0&0&0&0&0&0&1&0\\ 1&1&1&1&0&0&0&0&0&0&0&0&0&0&1 \end{array}\right]$$

Example: Construct parity check and generator matrices for a $(7, 4)$ Hamming code.
Solution: The parity check matrix H and generator matrix G for a $(7, 4)$ Hamming code are

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3.1 Syndrome Table Decoding

Consider a valid code word c for transmission and let e be an error pattern introduced by the channel during transmission. Then, the received vector r can be written as

$$r = c + e$$

Multiplying the r by the transpose of the parity check matrix gives the syndrome S which can be expressed as

$$\begin{aligned} S &= r \cdot H^T \\ &= (c + e) \cdot H^T \\ &= cH^T + eH^T \\ &= 0 + eH^T \\ &= eH^T \end{aligned} \tag{2.8}$$

Thus, the syndrome vector is independent of the transmitted code word c and is only a function of the error pattern e. Decoding is performed by computing the syndrome of a received vector, looking up the corresponding error pattern, and subtracting the error pattern from the received word.

Example: Construct a syndrome decoding table for a $(7, 4)$ Hamming code
Solution: For a $(7, 4)$ Hamming code, there are $2^{(7-4)}$ error patterns e as in Table 2.8

Table 2.1: Syndrome decoding table for a $(7, 4)$ Hamming code

Error Pattern e	Syndrome
0000000	000
1000000	100
0100000	010
0010000	001
0001000	110
0000100	011
0000010	111
0000001	101

The syndrome for $(7, 4)$ Hamming code is computed using the parity check matrix H (as given in the solution 2.7) as follows

$$S = e \cdot H^T$$

Thus, the syndrome decoding table for a $(7, 4)$ Hamming code is as in Table 2.1

2.3.2 Hamming Codes Decoding

Syndrome table is used to decode the Hamming codes. The syndrome table gives the syndrome value based on the simple relationship with parity check matrix. The single-error-correcting codes (i.e., Hamming codes), are decoded by using syndrome value. Consider a code word c corrupted by e, an error pattern with a single one in the j^{th} coordinate position results a received vector r. Let $h_0, h_1, \ldots, h_{n-1}$ be the set of columns of the parity check matrix H. When the syndrome is computed, we obtain the transposition of the j^{th} column of H.

$$s = eH^T = [0, \ldots, 0, 1, 0, \ldots, 0] \cdot \begin{bmatrix} h_0^T \\ h_1^T \\ \vdots \\ h_{n-1}^T \end{bmatrix} = h_j^T \tag{2.9}$$

The above-mentioned process in equation 2.9 can be implemented using the following algorithm:

1. Compute the syndrome s for the received word. If $s = 0$, the received code word is the correct code word.
2. Find the position j of the column of H that is the transposition of the syndrome.
3. Complement the j^{th} bit in the received codeword to obtain the corrected code word.

Example: Decode the received vector $r = [001100011100000]$ using the $(15, 11)$ parity check matrix.
Solution:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

The received vector is $r = [001100011100000]$
The corresponding syndrome $s = r \cdot H^T$ is

$$s = [0011]$$

The syndrome is the transposition of 7^{th} column of H. Inverting the 7^{th} coordinate of r, the following code word is obtained

$$c = [001100001100000]$$

2.4 LDPC Coding

Chapter 3

Input/Output Setup

3.1 Overview

In this chapter, we will discuss various input-output setups (depending on *how many antennas are used in both the transmitter and the receiver*) that were implemented in the simulator. The availability of multiple antennas at the transmitter and/or the receiver can be utilized in different ways to achieve different aims:

- Multiple antennas at the transmitter and/or the receiver can be used to provide additional diversity against fading on the radio channel. In this case, the channels experienced by the different antennas should have low mutual correlation, implying the need for a sufficiently large inter-antenna distance (spatial diversity), alternatively the use of different antenna polarization directions (polarization diversity).
- Multiple antennas at the transmitter and/or the receiver can be used to "shape" the overall antenna beam (transmit beam and receive beam, respectively) in a certain way, for example, to maximize the overall antenna gain in the direction of the target receiver/-transmitter or to suppress specific dominant interfering signals.
- The simultaneous availability of multiple antennas at the transmitter and the receiver can be used to create what can be seen as multiple parallel communication "channels" over the radio interface. This provides the possibility for very high bandwidth utilization without a corresponding reduction in power efficiency or, in other words, the possibility for very high data rates within a limited bandwidth without an un-proportionally large degradation in terms of coverage. This feature is highly present in the MIMO setup.

The following table explains the different setups implemented in the simulator.

Table 3.1: Transmitter & Receiver for different setups

Setup	Transmitter	Receiver
SISO (Single-Input Single-Output)	One antenna	One antenna
SIMO (Single-Input Multi-Output)	One antenna	Many antennas
MISO (Multi-Input Single-Output)	Many antennas	One antenna
MIMO (Multi-Input Multi-Output)	Many antennas	Many antennas

Later in the chapter we will discuss how each setup works, its encoding and decoding techniques and benifits. We will also zoom in as if we were to send only one symbol. This will help us explain the techniques used in each setup better.

Before we discuss each setup in great detail, some information about the channel model must be cleared out:

- We are using an OFDM based system.
- Each sub-carrier carries only one modulated symbol.
- We let the channel model be a block fading channel where the channel stays the same for some time (Coherence time) and for some subcarriers (Coherence bandwidth).

The following assumptions were also made:

- We will ignore the modulation of the input signal for better understanding of how each setup works.
- We will also assume perfect knowledge of the channel at both the transmitter and the receiver.

3.1.1 Narrow-band wireless fading channel

One of the most common wireless channel model and the one that we will be using is the following

$$y[m] = h[m]x[m] + n[m]$$

Where the index m is a time index, $x[m]$ is the transmitted complex symbol at time m, $y[m]$ is the corresponding receives signal, and $n[m]$ is the AWGN at time m.
The different component compared to the AWGN channel is $h[m]$. This is referred to as the "fading" coefficient. It is a random value which captures the changes that happen in the transmitted electromagnetic waves due to the environment.
$h[m]$ is modelled as a complex number, where

$$h[m] = \mathrm{a} + j\mathrm{b} = |h[m]|e^{j\theta[m]} \tag{3.1}$$

where the magnitude of the channel $|h[m]|$ is a Rayleigh random variable

$$|h[m]|\ f_{\sigma^2}(x) = \frac{x}{\sigma^2} e^{\frac{-x^2}{2\sigma^2}}$$

The phase $\theta[m]$ is a uniform random variable. The figure shows a Rayleigh distribution for different values of σ. This shows the impact of fading on communications. Specifically, the value of $|h[m]|$ can be small with a considerable probability, and that effectively reduces the received power of the transmitted signal. When $|h[m]|$ is too small, the channel is said to be in deep fading.

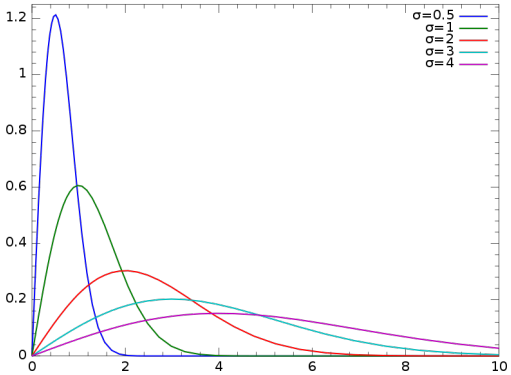


Figure 3.1: Rayleigh random variable

3.2 SISO

SISO stands for **"Single Input, Single Output"**. In communication engineering, SISO is the simplest way to describe a communication link between a transmitter and a receiver. It is used to describe the case where both the transmitter and receiver have single antennas. Pre-coding is not typically used in SISO communication systems.
For a SISO channel (we drop the time index for now)

$$y = hx + n$$

The noise power is N_o. The minimum distance between Constellation points without fading is $2a$, where a is the smallest transmitted amplitude per constellation points. Now, with fading coefficient h, minimum distance becomes $d_{min} = 2|h|a$. There, for this CSI value, we can upper bound the probability of error as

$$P_e\left(\bar{h}\right) \leq \; k\, Q\left(\frac{d_{min}}{\sqrt{2N_o}}\right) = \; k\, Q\left(\sqrt{\frac{4\left|h\right|^2 a^2}{N_o}}\right) = k\, Q\left(\sqrt{2\left|h\right|^2 \mathrm{SNR}}\right) \tag{3.2}$$

We assume that h is Rayleigh (as described in equation 3.1) with $\sigma^2 = 1$. Then, we want to compute the "average" P_e over CSI realization. This turns out to be

$$P_e \; = \mathbb{E}_{|h|^2}\{P_e\left(h\right)\} \leq k\left(\frac{1}{2} - \frac{1}{2}\sqrt{\frac{\mathrm{SNR}}{1+\mathrm{SNR}}}\right) = \frac{k}{2}\left(1-\mu\right) \tag{3.3}$$

How does this compare with AWGN?
Recall: P_e for AWGN is bounded by $kQ(\sqrt{2\mathrm{SNR}})$.
The following figure compares Pe for both AWGN and fading far BPSK.

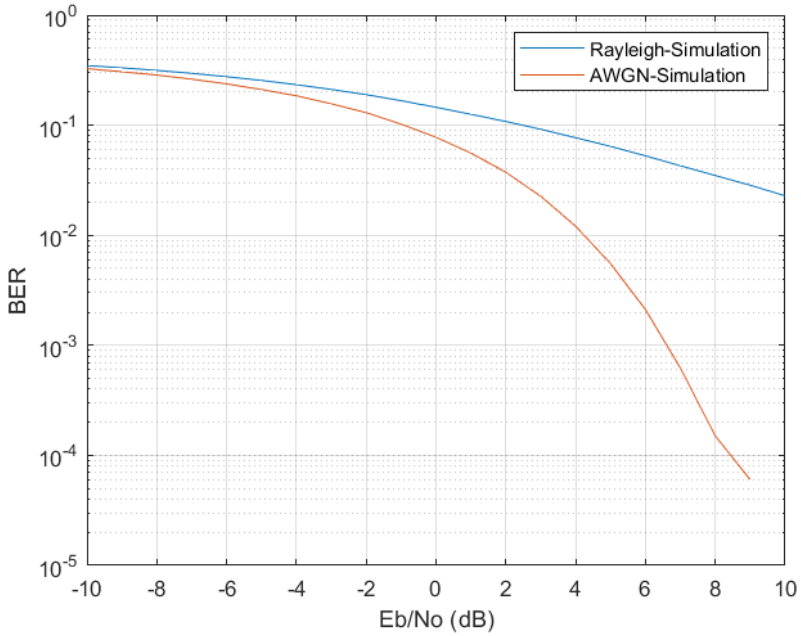


Figure 3.2: AWGN vs Rayleigh fading BER for SISO

At high SNR:
Probability of error in AWGN has a water fall behavior which indicates very good performance. However, it decays linearly in case of fading and that is too slow. This is because $|h|$ is many times too small and therefore the received power $|h|^2a^2$ is not enough to make decoding performance good. we say that a channel in *"deep fade"* if received signal power is less than or equal to the noise power.

$$|h|^2 a^2 \leq N_o$$
$$|h|^2 \leq \frac{N_o}{a^2} = \frac{1}{\text{SNR}}$$

What is the probability of this happening?
This probability is found to can be approximated to

$$10\log\left(\mathbb{P}\{|h|^2 \leq \frac{1}{\text{SNR}}\}\right) = \text{const} - 10\log(\text{SNR}) = \text{const} - \text{SNR}_{dB} \tag{3.4}$$

This looks like the linear decrease behavior we saw above in Figure 3.2.

3.3 SIMO

3.4 MISO

3.5 MIMO

3.5.1 Diversity

3.5.2 Multiplexing

Chapter 4

Reference Signals

4.1 Overview

Reference signals are predefined signals occupying specific resource elements within the down-link time-frequency grid. The NR specification includes several types of reference signals transmitted in different ways and intended to be used for different purposes by a receiving device. Unlike LTE, which relies heavily on always-on, cell-specific reference signals in the down-link for coherent demodulation, channel quality estimation for CSI reporting, and general time-frequency tracking, NR uses different down-link reference signals for different purposes. This allows for optimizing each of the reference signals for their specific purpose. It is also in line with the overall principle of ultra-lean transmission as the different reference signals can be transmitted only when needed. Later release of LTE took some steps in this direction, but NR can exploit this to a much larger degree as there are no legacy NR devices to cater for.
The NR reference signals include:

- Demodulation reference signals (DMRS) for PDSCH are intended for channel estimation at the device as part of coherent demodulation. They are present only in the resource blocks used for PDSCH transmission. Similarly, the DMRS for PUSCH allows the gNB to coherently demodulate the PUSCH.
- Phase-tracking reference signals (PTRS) can be seen as an extension to DMRS for PDSCH/PUSCH and are intended for phase-noise compensation. The PT-RS is denser in time but sparser in frequency than the DM-RS, and, if configured, occurs only in combination with DMRS.
- CSI reference signals (CSI-RS) are downlink reference signals intended to be used by devices to acquire down-link channel-state information (CSI). Specific instances of CSI reference signals can be configured for time/frequency tracking and mobility measurements.
- Tracking reference signals (TRS) are sparse reference signals intended to assist the device in time and frequency tracking
- Sounding reference signals (SRS) are uplink reference signals transmitted by the devices and used for uplink channel-state estimation at the base stations.

The previous section assumed full knowledge of the channel characteristics at both transmitter and receiver sides, this section discusses how such knowledge is gained by focusing on two important reference signals : CSI-RS and DMRS.
The propagation channel depends on the transmit frequency. Therefore, if up-link and down-link operate on two different frequencies as is the case in FDD, there is no choice but relying on the receiver to communicate information about the channel back to the transmitter. This is the case at the bottom right.
In the case of TDD, where the up-link and down-link share the same transmit frequency, it is possible, on the other hand, to estimate the down-link channel based on measurements on up-link transmission (or the opposite). In this section we focus on the FDD transmission case.

4.2 Demodulation Reference signal (DMRS)

The DM-RS in NR provides quite some flexibility to cater for different deployment scenarios and use cases: a front-loaded design to enable low latency, support for up to 12 orthogonal antenna ports for MIMO, transmissions duration from 2 to 14 symbols, and up to four reference-signal instances per slot to support very high-speed scenarios. To achieve low latency, it is beneficial to locate the demodulation reference signals early in the transmission, sometimes known as front-loaded reference signals. This allows the receiver to obtain a channel estimate early and, once the channel estimate is obtained, process the received symbols on the fly without having to buffer a complete slot prior to data processing. This is essentially the same motivation as for the frequency-first mapping of data to the resource elements. Two main time-domain structures are supported, differing in the location of the first DM-RS symbol:

- Mapping type A, where the first DMRS is located in symbol 2 or 3 of the slot and the DMRS is mapped relative to the start of the slot boundary, regardless of where in the slot the actual data transmission starts. This mapping type is primarily intended for the case where the data occupy (most of) a slot. The reason for symbol 2 or 3 in the down-link is to locate the first DMRS occasion after a CORESET located at the beginning of a slot.
- Mapping type B, where the first DMRS is located in the first symbol of the data allocation, that is, the DMRS location is not given relative to the slot boundary but rather relative to where the data are located. This mapping is originally motivated by transmissions over a small fraction of the slot to support very low latency and other transmissions that benefit from not waiting until a slot boundary starts but can be used regardless of the transmission duration.

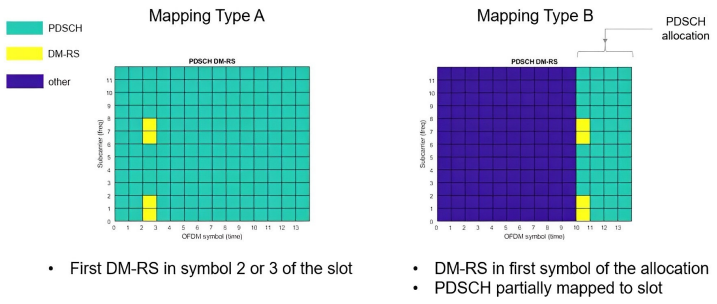


Figure 4.1: DMRS Mapping type A & type B

Although front-loaded reference signals are beneficial from a latency perspective, they may not be sufficiently dense in the time domain in the case of rapid channel variations. To support high-speed scenarios, it is possible to configure up to three additional DM-RS occasions in a slot. The channel estimator in the receiver can use these additional occasions for more accurate channel estimation, for example, to use interpolation between the occasions within a slot.

4.2.1 Pilot based DMRS channel estimation

Assume a single antenna port at both Tx and Rx.

$$y = h_{11}x$$

In order to know $h_1 1$, we need to send a symbol x which is known at the receiver, then $h_1 1$ can be calculated as:

$$h_{11} = \frac{y}{x}$$

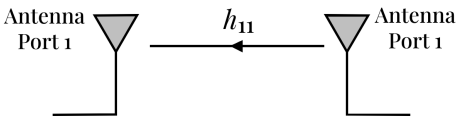


Figure 4.2: Single Tx Single Rx

This known symbol x is called a pilot symbol. The problem with this approach is that there exists a channel coefficient need to be known for each RE in the PDSCH which requires sending pilots in all REs, hence there will not be any place for data bits. The solution to this problem is to send pilot symbols in some REs as shown in the figure to determine the channel coefficients in those REs.

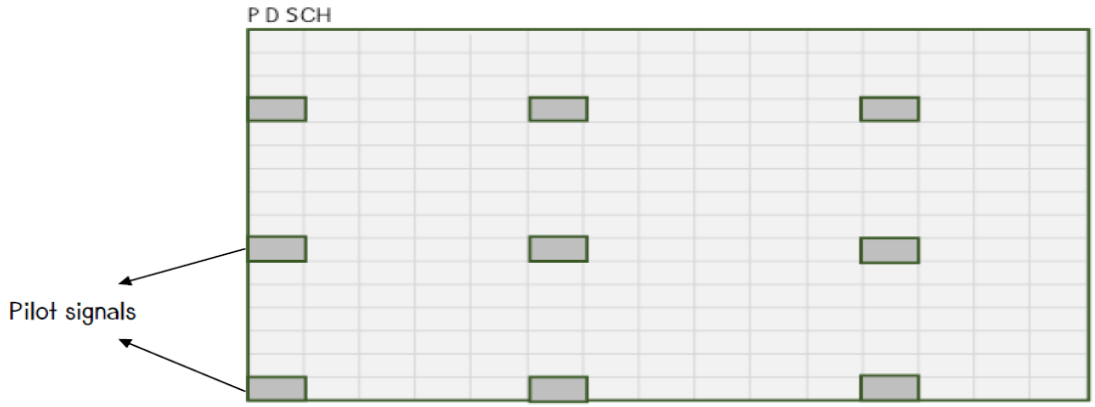


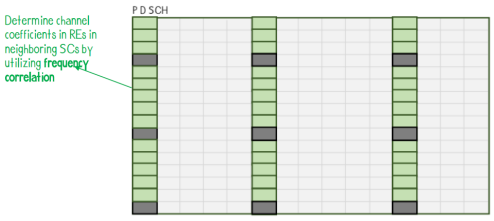
Figure 4.3: Pilot signals in some REs

Use time/frequency correlation to determine channel coefficients in other REs

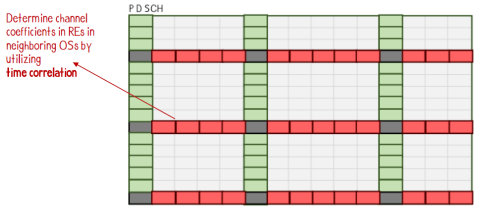
For the above property to be used time and frequency correlations must be determined, these are known using coherence time and coherence bandwidth.

Coherence bandwidth is a statistical measurement of the range of frequencies over which the channel can be considered "flat", or in other words the approximate maximum bandwidth or frequency interval over which two frequencies of a signal are likely to experience comparable or correlated amplitude fading. If the multi-path time delay spread equals D seconds, then the coherence bandwidth B_c is given approximately by the equation

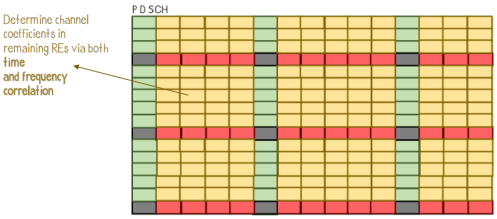
$$B_c \approx \frac{1}{D} \tag{4.1}$$



(a) Frequency correlation



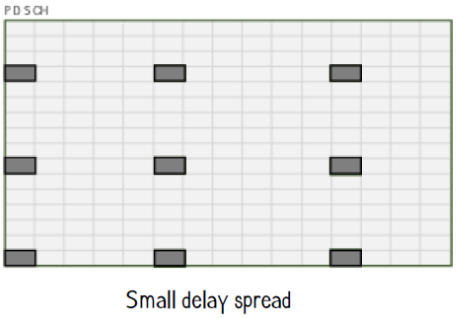
(b) Time correlation



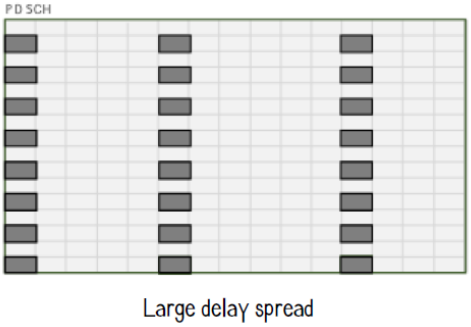
(c) Time / frequency correlation

Figure 4.4: Correlation in time & frequency

Hence if the delay spread is small, frequency correlation is high which means we need less pilots density in the frequency domain.



(a) Small delay spread



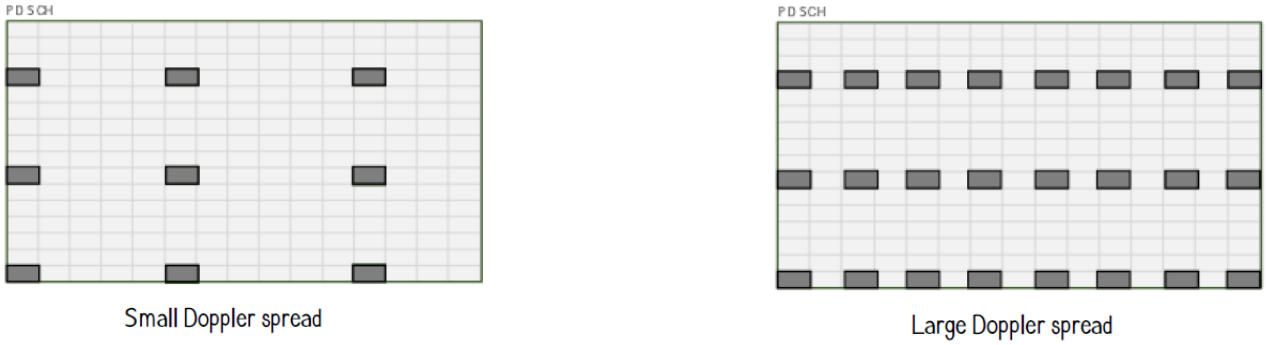
(b) Large delay spread

Figure 4.5: Delay spread effect

Coherence time is the time duration over which the channel impulse response is considered to be not varying. Such channel variation is much more significant in wireless communications systems, due to Doppler effects. If the maximum doppler spread equals f_m Hz, then the coherence time T_c is given approximately by the equation

$$T_c \approx \frac{1}{f_m} \tag{4.2}$$

Hence if the doppler spread is small, time correlation is high which means we need less pilots density in the time domain.



(a) Small Doppler spread

(b) Large Doppler spread

Figure 4.6: Doppler spread effect

4.3 Channel State Information Reference Signal (CSI-RS)

In the first release of LTE (release 8), channel knowledge for the downlink transmission direction was solely acquired by means of device measurements on the so-called cell-specific reference signals (CRS). The LTE CRS are transmitted over the entire carrier bandwidth within every LTE sub-frame of length 1 ms, and can be assumed to be transmitted over the entire cell area. Thus, a device accessing an LTE network can assume that CRS are always present and can be measured on.
In LTE release 10 the CRS were complemented by so-called CSI-RS. In contrast to CRS, the LTE CSI-RS are not necessarily transmitted continuously. Rather, an LTE device is explicitly configured to measure on a set of CSI-RS and does not make any assumptions regarding the presence of a CSI-RS unless it is explicitly configured for the device. The origin for the introduction of CSI-RS was the extension of LTE to support spatial multiplexing with more than four layers, something which was not possible with the release-8 CRS. However, the use of CSI-RS was soon found to be an, in general, more flexible and efficient tool for channel sounding, compared to CRS. In later releases of LTE, the CSI-RS concept was further extended to also support, for example, interference estimation and multi-point transmission.
As already described, a key design principle for the development of NR has been to as much as possible avoid "always on" signals. For this reason, there are no CRS-like signals in NR. Rather, the only "always-on" NR signal is the so called SS block which is transmitted over a limited bandwidth and with a much larger periodicity compared to the LTE CRS. The SS block can be used for power measurements to estimate, for example, path loss and average channel quality. However, due to the limited bandwidth and low duty cycle, the SS block is not suitable for more detailed channel sounding aimed at tracking channel properties that vary rapidly in time and/or frequency.
Instead the concept of CSI-RS is reused in NR and further extended to, for example, provide support for beam management and mobility as a complement to SS block. CSI-RS is a DL signal that can be used by the UE to measure some parameters and reports it back to the gNB to take actions based on that parameters. Some of this parameters can be :

- Channel quality indicator (CQI)
- Precoding matrix indicator (PMI)
- Rank indicator (RI)

4.3.1 Basic CSI-RS structure

A configured CSI-RS may correspond to up to 32 different antenna ports, each corresponding to a channel to be sounded. In NR, a CSI-RS is always configured on a per-device basis. It is important to understand though that configuration on a per-device basis does not necessarily mean that a transmitted CSI-RS can only be used by a single device. Nothing prevents identical CSI-RS using the same set of resource elements to be separately configured for multiple devices, in practice implying that a single CS-RS is shared between the devices.
As illustrated in Fig. 2.7, a single-port CSI-RS occupies a single resource element within a block corresponding to one resource block in the frequency domain and one slot in the time domain. In principle, the CSI-RS can be configured to occur anywhere within this block although in practice there are some restrictions to avoid collisions with other downlink physical channels and signals. Especially, a device can assume that transmission of a configured CSI-RS will not collide with:

- Any CORESET configured for the device.
- Demodulation reference signals associated with PDSCH transmissions scheduled for the device.
- Transmitted SS blocks.

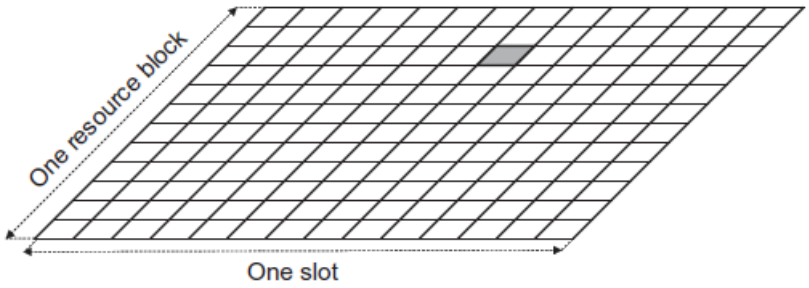


Figure 4.7: Single-port CSI-RS structure consisting of a single resource element within an RB/slot block.

4.3.2 Frequency-Domain structure of CSI-RS configurations

A CSI-RS is configured for a given downlink bandwidth part and is then assumed to be confined within that bandwidth part and use the numerology of the bandwidth part.
The CSI-RS can be configured to cover the full bandwidth of the bandwidth part or just a fraction of the bandwidth. In the latter case, the CSI-RS bandwidth and frequency-domain starting position are provided as part of the CSI-RS configuration.
Within the configured CSI-RS bandwidth, a CSI-RS may be configured for transmission in every resource block, referred to as CSI-RS density equal to one.
However, a CSI-RS may also be configured for transmission only in every second resource block, referred to as CSI-RS density equal to 1/2. In the latter case, the CSI-RS configuration includes information about the set of resource blocks (odd resource blocks or even resource blocks) within which the CSI-RS will be transmitted. CSI-RS density equal to 1/2 is not supported for CSI-RS with 4, 8, and 12 antenna ports.
There is also a possibility to configure a single-port CSI-RS with a density of 3 in which case

the CSI-RS occupies three subcarriers within each resource block. This CSI-RS structure is used as part of a so-called Tracking Reference signal (TRS).

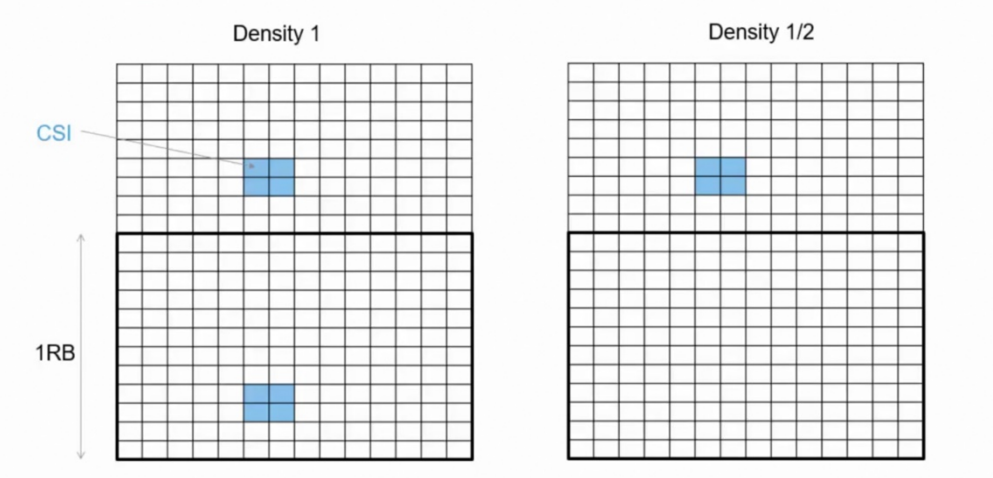


Figure 4.8: CSI-RS Frequency density

4.3.3 Time-Domain structure of CSI-RS configurations

The per-resource-block CSI-RS structure outlined above describes the structure of a CSI-RS transmission, assuming the CSI-RS is actually transmitted in a given slot. In general, a CSI-RS can be configured for periodic, semi-persistent, or aperiodic transmission.
In the case of periodic CSI-RS transmission, a device can assume that a configured CSI-RS transmission occurs every Nth slot, where N ranges from as low as four, that is, CSI-RS transmissions every fourth slot, to as high as 640, that is, CSI-RS transmission only every 640th slot. In addition to the periodicity, the device is also configured with a specific slot offset for the CSI-RS transmission
In the case of semi-persistent CSI-RS transmission, a certain CSI-RS periodicity and corresponding slot offset are configured in the same way as for periodic CSI-RS transmission. However, actual CSI-RS transmission can be activated/ deactivated based on MAC control elements (MAC CE). Once the CSI-RS transmission has been activated, the device can assume that the CSIRS transmission will continue according to the configured periodicity until it is explicitly deactivated. Similarly, once the CSI-RS transmission has been deactivated, the device can assume that there will be no CSI-RS transmissions according to the configuration until it is explicitly re-activated.
In the case of aperiodic CSI-RS, no periodicity is configured. Rather, a device is explicitly informed ("triggered") about each CSI-RS transmission instant by means of signaling in the DCI.

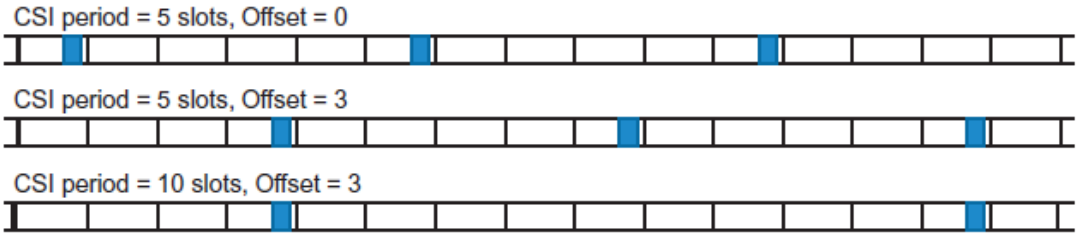


Figure 4.9: CSI-RS Periodicity and slot offset

4.3.4 CSI-RS and DMRS for full channel estimation

DMRS is used to help the receiver determine the effective channel, i.e., after taking into account the effect of the precoding matrix.
The question is how to determine the best precoding matrix ? CSI-RS is the answer. Channel estimation process steps are as follows:

- CSI-RS is transmitted to the UE to make initial channel estimation.
- A CSI feedback is sent to the gNB to be used to choose the necessary precoding, keep in mind that it is not necessary that the gNB uses the precoding suggested by the UE.
- PDSCH and its DMRS are transmitted after applying the chosen precoding.
- DMRS is used by the UE to estimate the effective channel after taking into account the effect of the precoding.

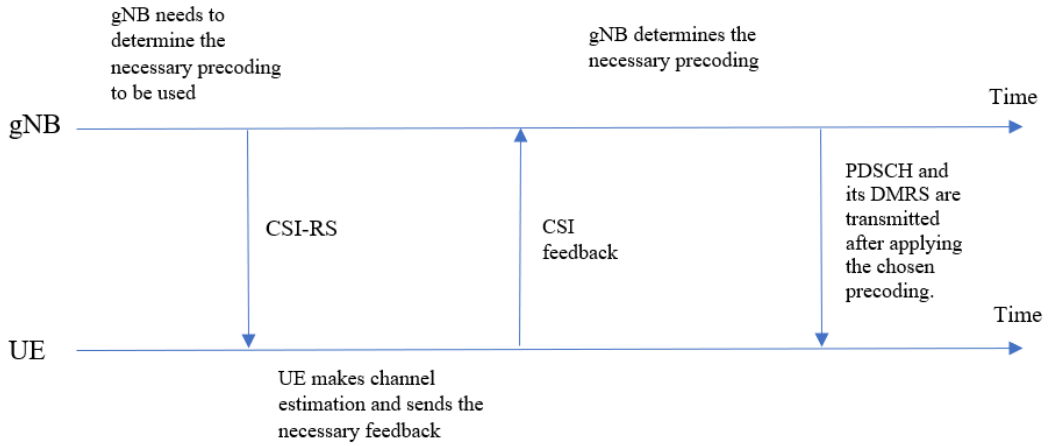


Figure 4.10: Channel estimation process

4.4 Channel Estimation

In general, there are two types of MIMO channel estimation methods:

Training based approach: which uses known training symbols.

Blind-based approach: which perform CE without the benefit of known training symbols.

In training-based CE, known training symbols are transmitted at certain prescribed times and frequencies that are known by the receiver. Since the receiver knows the training symbols, as well as when and where (i.e., at which frequencies) they are transmitted, it uses that information to estimate the gain and phase rotation imparted by the channel at each point in time and frequency based on the characteristics of the received training symbols. Although blind-based methods have higher bandwidth efficiencies because they do not use any resources for transmitting training symbols, they tend to have lower speed and poorer performance than

training-based methods. For this reason, training-based CE is used more than blind-estimation, and it is the method we focus on in our project.

The placement of training symbols in time, frequency, and space (i.e., the transmit antenna's) dimensions is a key part of the design of a MIMO communication system. In general, training symbols should be spaced as far apart as possible to reduce training overhead, while still maintaining a required performance level. For example, in a high Doppler, fast fading environment, training symbols need to be placed relatively often in time. Similarly, in a highly frequency-selective channel, training symbols need to be placed close together in the frequency dimension. In this section we discuss two channel estimation methods :

- Simple pilot based estimation
- MMSE estimation

4.4.1 Simple pilot based estimation

Assume a single antenna port at both Tx and Rx

$$y = h_{11}x$$

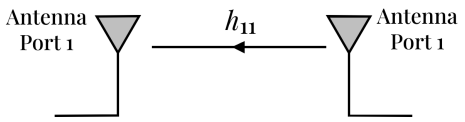


Figure 4.11: Single Tx Single Rx

In order to know h_{11}, we need to send a symbol x Which is known at the Rx, then h_{11} can be known as

$$h_{11} = \frac{y}{x}$$

This known symbol x is called a *pilot symbol.*

What about a MIMO channel ?
Assume a 2x2 MIMO setup as shown in Figure 4.12
The received signal y at the receiver side is

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{21} \\ h_{12} & h_{22} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

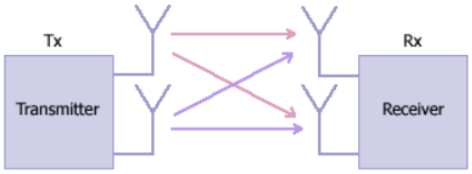


Figure 4.12: 2x2 MIMO setup

Estimating the 4 channel coefficients is not as simple as the SISO case, to do so one pilot symbol won't be enough, a number of pilot symbols equal the number of transmit antennas will be needed.

On the grey REs, pilots from the first layer only are sent

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{21} \\ h_{12} & h_{22} \end{bmatrix} \times \begin{bmatrix} p_1 \\ 0 \end{bmatrix} = \begin{bmatrix} h_{11} \\ h_{12} \end{bmatrix} p_1$$

Knowing the value of the pilot symbol at the receiver side the channel coefficients corresponding to the first Tx layer can be estimated.
The same is done on the red REs with pilots from the second layer only are send, hence, the channel coefficients corresponding to the second Tx layer can be estimated.
This means that a number of pilot symbols equals the number of Tx layers is needed to fully

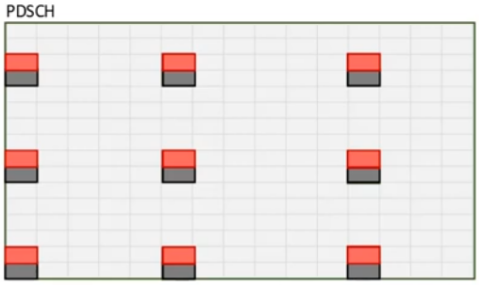


Figure 4.13: pilot symbols for 2x2 MIMO channel

estimate a MIMO channel.

The problem with this estimation method is that in the presence of noise the estimated channel coefficients are not exactly the real coefficients which results in errors in both precoding and decoding operations at the Tx and Rx. This problem appears clearly specially when estimation is done for two reference signals (i.e., CSI-RS and DMRS) where CSI-RS is used to choose the necessary precoding then DMRS is used to estimate the effective channel at the receiver side for decoding operation, the accumulated error in both estimations becomes big which results in a poor BER performance.

4.4.2 Minimum mean square error (MMSE) estimation

In MMSE the same approach as simple pilot based estimation (as mentioned in subsection 4.4.1) is done but the difference is that MMSE estimation takes into account the noise effect to get a better estimate of the channel coefficients.
For the channel model $y = Hx + n$ the channel estimate $\hat{\mathrm{H}}$ is given as

$$\hat{H} = R_{hy} R_{yy} (\bar{y} - \bar{\mu_y}) + \mu_n \tag{4.3}$$

$$\text{where} \quad R_{hy} = \mathbb{E}\{(h - \mu_h)(\bar{y} - \bar{\mu_y})^T\} \quad \text{is the cross covariance of } H, y$$
$$\text{and} \quad R_{yy} = \mathbb{E}\{(\bar{y} - \bar{\mu_y})(\bar{y} - \bar{\mu_y})^T\} \quad \text{is the covariance matrix of } y$$
$$\bar{\mu_y} \quad , \quad \mu_n \quad \text{are the expected value of } y \text{ and } n \text{ respectively.}$$

Using some mathematical manipulation we get that the estimate value of the channel can be given by this simplified expression

$$\hat{H} = \frac{\frac{\bar{x}^H \bar{y}}{N_o} + \frac{\mu_h}{\sigma_h^2}}{\frac{\|\bar{x}\|^2}{N_o} + \frac{1}{\sigma_h^2}} \tag{4.4}$$

Although MMSE estimation is more computationally complex than simple pilot estimations it gives a better estimate of the channel coefficients and hence a better BER performance.

Chapter 5

Simulations

Part II

Deep Reinforcement Learning Model

Chapter 6

Introduction

The idea that we learn by interacting with our environment is probably the first to occur. to us when we think about the nature of learning. When an infant plays, waves its arms, or looks about, it has no explicit teacher, but it does have a direct sensorimotor connection. to its environment. Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about what to do to achieve goals. Throughout our lives, such interactions are undoubtedly a major source. of knowledge about our environment and ourselves. Whether we are learning to drive a car or to hold a conversation, we are acutely aware of how our environment responds. to what we do, and we seek to influence what happens through our behavior. Learning from interaction is a foundational idea underlying nearly all theories of learning and intelligence.

6.1 Reinforcement Learning

6.1.1 Meaning

Reinforcement learning is learning what to do think in other way how to behave in such situations to make some reaction or as in literature action to certain state you are in as to maximize a numerical reward signal. The learner is not told how to behave or which action he must take but he discovers it with trial and error and decide which action achieve the highest reward which map to the best action reward but also the next situation and, through that, all subsequent rewards. These two characteristics—trial-and-error search and delayed reward—are the two most important distinguishing features of reinforcement learning. The basic idea is simply to capture the most important aspects of the real problem facing a learning agent interacting over time with its environment to achieve a goal. A learning agent must be able to sense the state of its environment to some extent and must be able to take actions that affect the state. The agent also must have a goal or goals relating to the state of the environment. Markov decision processes are intended to include just. these three aspects—sensation, action, and goal—in their simplest possible forms without trivializing any of them. Any method that is well suited to solving such problems we consider to be a reinforcement learning method.

6.1.2 RL & Machine Learning

Reinforcement learning is also different from what machine learning researchers call. Unsupervised learning, which is typically about finding structure hidden in collections of unlabelled data. The terms supervised learning and unsupervised learning would seem. to exhaustively classify machine learning paradigms, but they do not. Although one might be tempted to think of reinforcement learning as a kind of unsupervised learning. because it does not rely on examples of correct behavior, reinforcement learning is trying. to maximize a reward signal instead of trying to find hidden structure.
Uncovering structure in an agent's experience can certainly be useful in reinforcement learning, but by itself does not address the reinforcement learning problem of maximizing a reward signal. We therefore consider reinforcement learning to be a third machine learning paradigm, alongside supervised learning and unsupervised learning and perhaps other paradigms.

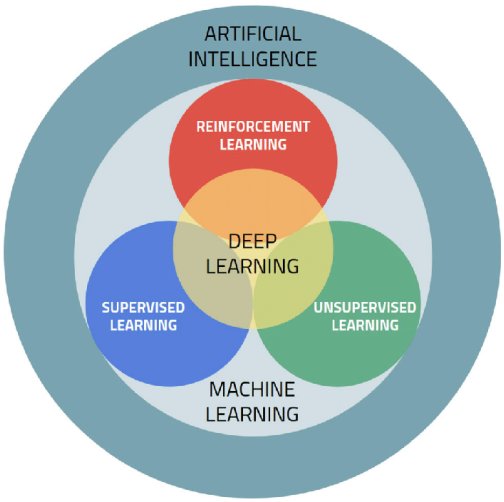


Figure 6.1: Artificial Intelligence Map

One of the challenges that arise in reinforcement learning, and not in other kinds. of learning, is the trade-of between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past. and found to be effective in producing reward. But to discover such actions, it must try actions that it has not selected before. The agent must exploit what it has already. experienced in order to obtain reward, but it also has to explore in order to make better. action selections in the future. The dilemma is that neither exploration nor exploitation. can be pursued exclusively without failing at the task. The agent must try a variety of actions and progressively Favor those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward. The exploration-exploitation dilemma has been intensively studied by mathematicians for many decades yet remains unresolved. For now, we simply note that the entire issue of balancing exploration and exploitation do not even arise in supervised and unsupervised. learning, at least in the purest forms of these paradigms. Another key feature of reinforcement learning is that it explicitly considers the whole. problem of a goal-directed agent interacting with an uncertain environment. This is in contrast to many approaches that consider sub-problems without addressing how they might fit into a larger picture.

Reinforcement learning takes the opposite tack, starting with a complete, interactive, goal-seeking agent. All reinforcement learning agents have explicit goals, can sense. aspects of their environments and can choose actions to influence their environments. Moreover, it is usually assumed from the beginning that the agent must operate despite significant uncertainty about the environment it faces. When reinforcement learning involves planning, it must address the interplay between planning and real-time action. selection, as well as the question of how environment models are acquired and improved. When reinforcement learning involves supervised learning, it does so for specific reasons. that determine which capabilities are critical and which are not. For learning research to make progress, important sub-problems must be isolated and studied, but they should. be sub-problems that play clear roles in complete, interactive, goal-seeking agents, even if all the details of the complete agent cannot yet be filled in. By a complete, interactive, goal-seeking agent we do not always mean something like a complete organism or robot. These are clearly examples, but a complete, interactive, goal-

seeking agent can also be a component of a larger behaving system. In this case, the agent directly interacts with the rest of the larger system and indirectly interacts. with the larger system's environment.

One of the most exciting aspects of modern reinforcement learning is its substantive. and fruitful interactions with other engineering and scientific disciplines. Reinforcement learning is part of a decades-long trend within artificial intelligence and machine learning toward greater integration with statistics, optimization, and other mathematical subjects. Of all the forms of machine learning, reinforcement learning is the closest to the kind of learning that humans and other animals do, and many of the core algorithms of reinforcement learning was originally inspired by biological learning systems. Reinforcement learning has also given back, both through a psychological model of animal learning that better matches some of the empirical data, and through an influential model of parts of the brain's reward system. The body of this book develops the ideas of reinforcement learning. that pertain to engineering and artificial intelligence, with connections to psychology and neuroscience.

6.1.3 Vocabulary of Reinforcement Learning

Beyond the agent and the environment, one can identify *four main sub-elements* of any reinforcement learning system:

A policy: A policy defines *the learning agent's way of behaving at a given time.* Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states.

Reward signal (the goal of RL): A reward signal defines *the goal* of a reinforcement learning problem. On each time step, the environment sends to the reinforcement learning agent a single number called the reward. The agent's sole objective is to maximize the total reward it receives over the long run. The reward signal thus defines what are the good and bad events for the agent.

Value function: A value function specifies *what is good in the long run.* Roughly speaking, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state to the end. A state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards.

Model of the environment: This is *something that mimics the behavior of the environment*, or more generally, that allows inferences to be made about how the environment will behave.

6.2 An Example: Tic-Tac-Toe

Consider the familiar child's game of tic-tac-toe (Figure 6.2). Two players take turns playing on a three-by-three board. One player plays Xs and the other Os until one player wins by placing three marks in a row, horizontally, vertically, or diagonally. If the board fills up with neither player getting three in a row, then the game is a draw like illustrated in Figure 6.2. Because a skilled player can play so as never to lose, let us assume that we are playing against an imperfect

player, one whose play is sometimes incorrect and allows us to win. For the moment, in fact, let us consider draws and losses to be equally bad for us.
Now the question is: **How might we construct a player that will find the imperfections in its opponent's play and learn to maximize its chances of winning?**

Classical optimization methods for sequential decision problems, such as dynamic programming, can compute an optimal solution for any opponent, but require as input a complete specification of that opponent, including the probabilities with which he opponent makes each move in each board state. Let us assume that this information is not available a priori for this problem, as it is not for most problems of practical interest. On the other hand, such information can be estimated from experience, in this case by playing many games against the opponent. About the best one can do on this problem is first to learn a model of the opponent's behavior, up to some level of confidence, and then apply dynamic programming to compute an optimal solution given the approximate opponent model.

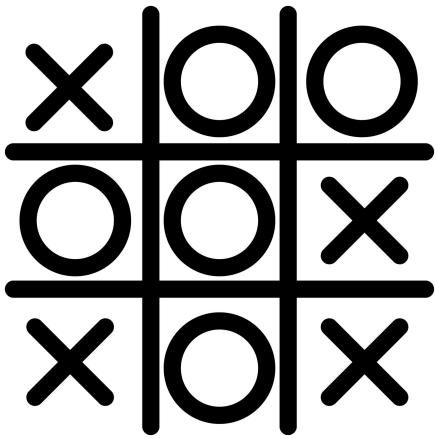


Figure 6.2: Tic-Tac-Toe

An evolutionary method applied to this problem would directly search the space of possible policies for one with a high probability of winning against the opponent. Here, a policy is a rule that tells the player what move to make for every state of the game—every possible configuration of Xs and Os on the three-by-three board. For each policy considered, an estimate of its winning probability would be obtained by playing some number of games against the opponent. This evaluation would then direct which policy or policies were considered next. A typical evolutionary method would hill-climb in policy space, successively generating and evaluating policies in an attempt to obtain incremental improvements. Or, perhaps, a genetic-style algorithm could be used that would maintain and evaluate a population of policies. Literally hundreds of different optimization methods could be applied.

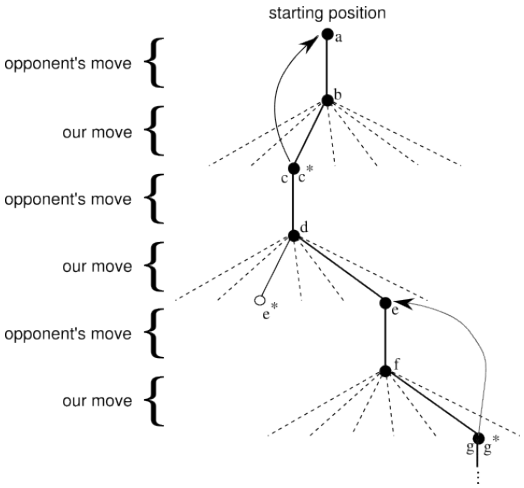


Figure 6.3: A sequence of tic-tac-toe moves.

Here is how the tic-tac-toe problem would be approached with a method making use of a value function:
First, we would set up a table of numbers, one for each possible state of the game. Each number will be the latest estimate of the probability of our winning. from that state. We treat this estimate as the state's value, and the whole table is the learned value function. State A has higher value than state B or is considered "better". than state B, if the current estimate of the probability of our winning from A is higher. than it is from B. Assuming we always play Xs, then for all states with three Xs in a row the probability of winning is 1, because we have already won. Similarly, for all states with three Os in a row, or that are filled up, the correct probability is 0, as we cannot. win from them. We set the initial values of all the other states to 0.5, representing a guess that we have a 50% chance of winning.

Then, we play many games against the opponent. To select our moves, we examine the states that would result from each of our possible moves (one for each blank space on the board) and look up their current values in the table. Most of the time we move greedily, selecting the move that leads to the state with greatest value, that is, with the highest. estimated probability of winning. Occasionally, however, we select randomly from among the other moves instead. These are called exploratory moves because they cause us to experience states that we might otherwise never see. A sequence of moves made and considered during a game can be diagrammed as in the Figure 6.3.
While we are playing, we change the values of the states in which we find ourselves. during the game. We attempt to make them more accurate estimates of the probabilities. of winning. To do this, we "back up" the value of the state after each greedy move to the state before the move, as suggested by the arrows in Figure 6.3. More precisely, the current value of the earlier state is updated to be closer to the value of the later state. This can be done by moving the earlier state's value a fraction of the way toward the value of the later state. If we let $\mathbf{S}_t$ denote the state before the greedy move, and $\mathbf{S}_{t+1}$ the state after the move, then the update to the estimated value of $\mathbf{S}_t$, denoted $\mathbf{V}(\mathbf{S}_t)$, can be written as

$$\mathbf{V}(\mathbf{S}_t) \leftarrow \mathbf{V}(\mathbf{S}_t) + \alpha\left[\mathbf{V}(\mathbf{S}_{t+1}) - \mathbf{V}(\mathbf{S}_t)\right]$$

here α is a small positive fraction called *the step-size parameter*, which influences the rate of learning. This update rule is an example of a temporal-difference learning. method, so called because its changes are based on a difference, $\mathbf{V}(\mathbf{S}_{t+1}) - \mathbf{V}(\mathbf{S}_t)$ between estimates at two successive times.

6.3 Problem Formulation

6.3.1 Reinforcement Learning Taxonomy

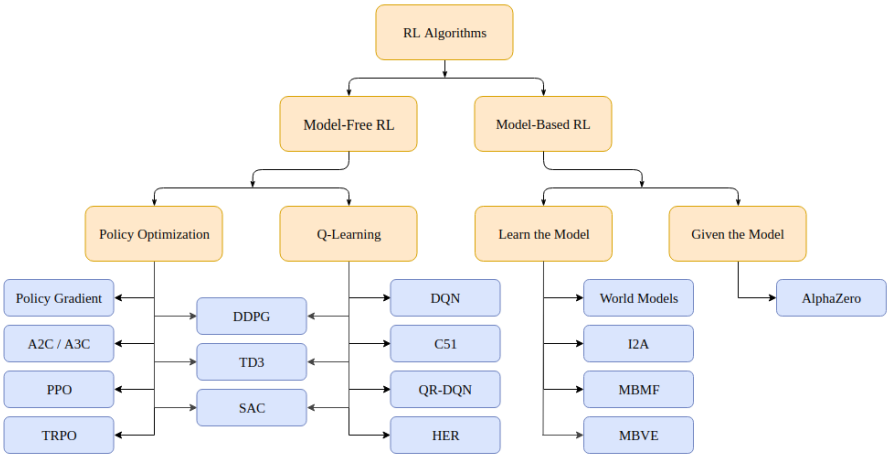


Figure 6.4: RL Taxonomy

6.3.2 Problem Nature

Continuous or Discrete

We have found that our problem has a *Continuous action and state space* since in a real-world situation, the states space is mostly continuous, with uncountable states and actions combinations for an agent to explore and in our problem the channel has infinite changes so it can have infinite number of pre-coders for each of these states. So we have excluded algorithms which depend on discrete spaces.

Deterministic or Stochastic

In AI and Reinforcement Learning (RL), policy refers to an agent's strategy to interact with an environment. Policies define the behavior of an agent. A policy determines the next action an agent takes in response to the current state of the environment. In other words, A policy is a function that maps a state to an action. Depending on the context and problem at hand, policies can be deterministic or stochastic. In this tutorial, we explain the difference between these two policy types.

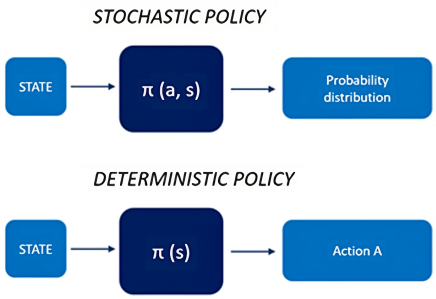


Figure 6.5: Deterministic vs. Stochastic Policies

Deterministic Policy: A deterministic policy is a policy that maps each state to a single action with certainty. In other words, the agent will always take the same action given a state. This policy is represented by a function $\pi : \mathbf{S} \rightarrow \mathbf{A}$ where $\mathbf{S}$ is the state space, and $\mathbf{A}$ is the action space. The deterministic policy function maps each state $s \in \mathbf{S}$ to a single action $a \in \mathbf{A}$. The advantage of a deterministic policy is that it is easy to interpret and implement. It is also suitable for tasks where the same action should be taken for the same state every time. *For example*, in a chess game, the best move for a given board configuration is always the same. A deterministic policy can be the best choice to play the game optimally in such cases.

Stochastic Policy: A stochastic policy is a policy that maps each state to a probability distribution over actions. In other words, given a state, the agent will choose an action randomly based on the probability distribution. We represent this policy by a function $\pi : \mathbf{S} \times \mathbf{A} \rightarrow [0, 1]$ where $\mathbf{S}$ is the state space, $\mathbf{A}$ is the action space, and $\pi \in (s, a)$ is the probability of taking an action a in a state s. The advantage of a stochastic policy is that it can capture the uncertainty in the environment. *For example,* in a poker game, the agent may not always take the same action in response to the same hand since there is a

probability of winning or losing depending on the opponent's hand and how the betting has proceeded. In such cases, a stochastic policy learns the best strategy based on the probability of winning.

Comparison
The primary difference between a deterministic and stochastic policy is the way in which they choose actions. A deterministic policy chooses a single action for each state, while a stochastic policy chooses from a probability distribution over actions for each state. This means that a deterministic policy always chooses the same action for the same state, while a stochastic policy may choose different actions for the same state. So depending on our problem nature where we need an exact pre-coder which will be optimal for a channel space so we need to have deterministic policy which leads to exclude some algorithms.

6.3.3 Model Nature

We interact with the environment all the time. Every decision we make influences our next ones in some unknown way. This behavior is the core of Reinforcement Learning (RL), where instead the rules of interaction and influence are not unknown, but predefined. RL algorithms can be either Model-free (MF) or Model-based (MB). If the agent can learn by making predictions about the consequences of its actions, then it is MB. If it can only learn through experience, then it is MF. In Reinforcement Learning, we have an agent which can take action in an environment. Additionally, there are probabilities associated with transitioning from one environment state to another. These transitions can be deterministic or stochastic. Ultimately, the goal of RL is for the agent to learn how to navigate the environment to maximize a cumulative reward metric.

Model-Free RL

Simply put, model-free algorithms refine their policy based on the consequences of their actions.as an example!
Consider this 5×5 environment (Figure 6.6)
In this example, we want the agent (in green) to avoid the red squares and reach the blue one in as few steps as possible. To achieve this, we need to define an appropriate reward function. Here's one way:

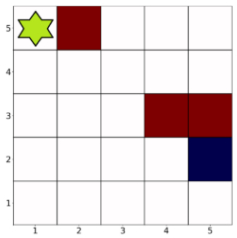


Figure 6.6: 5x5 environment

- Landing on an empty square: -1 point
- Landing on a red square: -100 points
- Landing on the blue square: +100 points

The agent has 4 possible actions: left, right, up, and down. On the edges, it has only 2 or 3 of these choices.

Model-Based RL

In a way, we could argue that Q-learning is model-based. After all, we are building a Q-table, which can be seen as a model of the environment. However, this is not how

the term model-based is used in the field. To classify as model-based, the agent must go beyond implementing a model of the environment. That is, the agent needs to make predictions of the possible rewards associated with certain actions. This provides many benefits. For example, the agent interacts with the environment a few times. Then, the model uses this information to simulate subsequent iterations without needing to interact with the environment. Using supervised learning, we can optimize the model to determine which trajectories are most likely to generate the biggest rewards.

6.3.4 Policy Learning Type

ON Policy

In on policy learning the agent learns about the policy used to generate the data. It means to learn about a policy. We simply mean learning the value function and in control We mean learning the optimal policy.

OFF Policy

In off policy learning the agent learns about a policy from data generated by following a different policy. That is the policy that we are learning is off the policy. They we are using for Action selection.

For example, you could learn the optimal policy while following a totally random policy we call the policy that the agent is learning the target policy because it is the target of the agents learning the target policy is usually denoted by π. The value function that the agent is learning is based on the target policy one example of a Target policy is the optimal policy we call the policy that the agent is using to select actions the behavior policy because it defines our agents behavior. The behavior policy is usually denoted by b. The behavior policy is in charge of selecting actions for the agent, so we are coupling the behavior from the target policy Because it provides another strategy for continual exploration. If our agent behaves according to the Target policy, it might only experience a small number of states. If our agent can behave according to a policy that favors exploration. It can experience a much larger number of states. Another key rule of off policy learning is that the behavior policy must cover the target policy In other words, if the target policy says the probability of selecting an action a given State s is greater than zero then the behavior policy must say the probability of selecting that action in that state is greater than 0 .It's worth noting that off policy learning is a strict generalization of on policy learning on policies the specific case where the target policy is equal to the behavior policy.

6.3.5 Types of RL Gradients

Value-Based: learn the state or state-action value. Act by choosing the best action in the state. Exploration is necessary.

Policy-Based: Directly learn the stochastic policy function that maps state to action. Act by sampling policy.

Model-Based: learn the model of the world, then plan using the model. Update and re-plan the model often.

We now focus on the policy gradient algorithms which learn a stochastic policy to maximize a cumulative reward. However, they can suffer from high variance, slow convergence, and poor exploration but we need add judgment or criticism to our action to be sure it is the best which need to add value-based gradient to determine if the action is the best.

Eventually this led us to ACTOR-CRITIC algorithms. Actor-critic algorithms are a variant of policy gradient methods that use an additional value function, called the critic, to reduce the variance of the policy gradient and improve the learning efficiency, and to add the determinism to our model we have especially chosen DDPG algorithm (discussed in the next chapter).

Chapter 7

Deep Deterministic Policy Gradient (DDPG)

As we stated in the last chapter, we have chosen the DDPG algorithm based on the nature of our problem so we now introduce the algorithm as the basic idea then we will go deeper to formulate our problem in the algorithm later this chapter.
To fully understand DDPG we must first walk through Actor-Critic algorithms family.

7.1 ACTOR-CRITIC Methods

Figure 7.1 depicts the main concept of Actor-Critic methods. Actor-Critic methods combine the **value-based methods** such as DQN and **policy-based methods** such as Reinforce.
DQN Agent which learns to approximate the optimal action value function. If the Agent learns sufficiently well so deriving a good policy for the Agent, it is straightforward. On the other side the Reinforce Agent parametrizes the policy and learns to optimize it directly. Here, the policy is usually stochastic, as we receive the distribution probability.

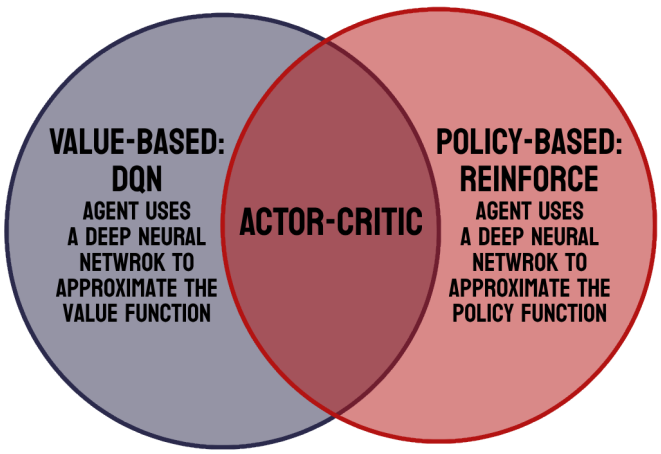


Figure 7.1: ACTOR-CRITIC Methods

Right now, we will investigate deterministic policies, which take a state and return the single action (no stochasticity, the policy will be deterministic).

For Reinforce Algorithm: it has to complete the episode before we can start training. For the environments, where every episode can last hundreds or even thousands of frames (like Atari games). It can be wasteful for the training perspective, where we have to interact with the environment in long perspective, only just to perform a single training step (in

order to estimate $\mathbf{Q}$ as accurate as possible). In this case, the training batch becomes very large.

For DQN: it is possible to replace the exact value for a discounted reward with our estimation using the one-step Bellman equation:

$$\mathbf{Q}(s,a) = r + \gamma \mathbf{V}(\mathbf{S}')$$

When we consider the Policy Gradient method (as we discussed above), we contemplated that the values $\mathbf{V}(\mathbf{S})$ or $\mathbf{Q}(s,a)$ exist anymore. In this case we apply the Actor-Critic method instead, where we use neural network to estimate $\mathbf{V}(\mathbf{S})$ and use this estimation to obtain $\mathbf{Q}$.

Estimated gradient in Policy Gradient method is proportional to the discounted reward from the given state. However, the range of this reward is highly environment dependent (it can happen that the Agent plays only short game the Agent lose very quick (low value of reward) or the Agent is smart enough and plays the game for the longer time, while collecting the rewards). Large difference between rewards collection can seriously affect the training dynamics, as one lucky episode will dominate in the final gradient. In such occurrences, the policy gradient method has high variance, which can influence the training process can become unstable.

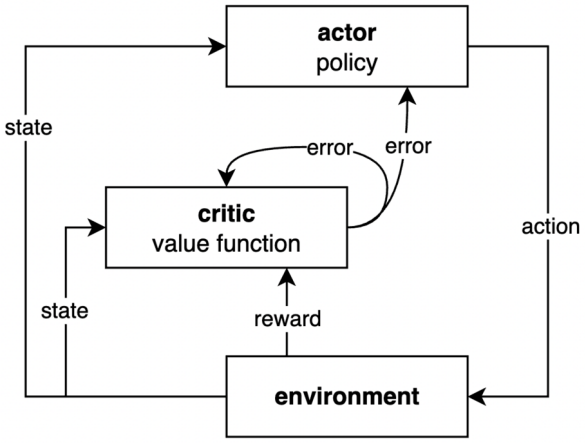


Figure 7.2: ACTOR-CRITIC

7.2 DDPG Algorithm

In Dynamic Programming, the Markov Decision Process (MDP) is solved by using value iteration and policy iteration. Both techniques require transition and reward probabilities to find the optimal policy. When the transition and reward probabilities are unknown, we use the Monte Carlo method to solve MDP. The Monte Carlo method requires only sample sequences of states, actions, and rewards. Monte Carlo methods are applied only to episodic tasks.

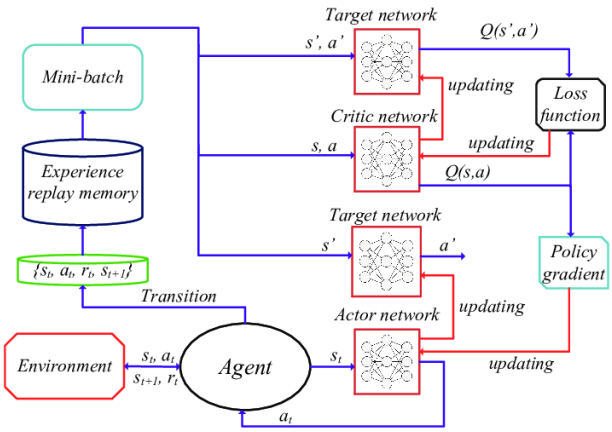


Figure 7.3: DDPG Algorithm

We can approach the Monte-Carlo estimate by considering that the Agent play in episode A. We start in state $\mathbf{S}_t$ and act $\mathbf{A}_t$. Based on the process the Agent transits to state $\mathbf{S}_{t+1}$. From environment, the Agent receives the reward $\mathbf{R}_{t+1}$. This process can be continued until the Agent reaches the end of the episode. The Agent can also take part in other episodes like B, C, and D. Some of those episodes will

have trajectories that go through the same states, which influences that the value function is computed as average of estimates. Estimates for a state can vary across episodes so the Monte-Carlo estimates will have high variance.
On the other side, we can apply the Temporal Difference estimate. Here, the **TD** approximates the current estimate based on the previously learned estimate, which is also called bootstrapping (we try to predict the state values).

$$\begin{aligned} \mathbf{V}(\mathbf{S}) = \mathbf{V}(\mathbf{S}) + \alpha\left(r + \gamma\mathbf{V}(\mathbf{S}') - \mathbf{V}(\mathbf{S})\right) \\ \mathbf{V}(\mathbf{S}) \text{ is the } \textit{expected reward}. \\ (r + \gamma\mathbf{V}(\mathbf{S}')) \text{ is the } \textit{actual reward}. \end{aligned} \tag{7.1}$$

Equation 7.1 is the difference (**TD** error) between the actual reward and the expected reward multiplied by the learning rate α (the learning rate, also called step size, used for convergence reason). **TD** estimates are low variance because you are only compounding a single time step of randomness instead of a full rollout like in Monte-Carlo estimate. However, due to applying bootstrapping (dynamic programming) the next state is only estimated. Estimated values introduce bias into our calculations. The agent will learn faster, but converging problems can occur. Deriving the Actor -Critic concept requires to consider first the policy-based approach *(performed by AGENT)*.
As we discussed before, the Agent playing the game increases the probability of actions that lead to a win and decreases the probability of actions that lead to losses. However, such a process is cumbersome due to a lot of data to approach the optimal policy. On the other hand, we can evaluate the value-based approach *(performed by CRITIC)*, where the guesses are performed on-the-fly, throughout all the episodes. At the beginning our guesses will be misaligned (not correct). But over time, when we capture more experience, we will be able to make solid guesses. Though this is not a perfect approach either, guesses introduce a bias because they will sometimes be wrong, particularly because of a lack of experience.

We can summarize that:

- The Agent using policy-based approach is learning to act (agent learns by interacting with environment and adjusts the probabilities of good and bad actions, while in a value-based approach, the agent is learning to estimate states and actions.).
- The Critic is used to evaluate the quality of actions more quickly (proper action or not) and speed up learning.

As a result of the merger of Actor-Critic we utilize *two separate neural networks*. The role of the Actor network is to determine the best actions (from probability distribution) in the state by tuning the parameter θ (weights). The Critic, by computing the temporal difference error **TD** (estimating expected returns), evaluates the action generated by the actor.

(DQN) is working in discrete environments where the number of action which could be performed by the Agent was limited. However, very often we operate with a continuous environment, securing continuous motion. The number of actions then can be unlimited (huge). This is one of the problems DDPG solves. DDPG algorithm uses Agent- Critic concept, where we use two deep neural networks.
In DDPG, the Actor is used to approximate the optimal policy deterministically. That means we want always to generate the best believed action for any given state.

The Actor follows the policy-based approach and learns how to act by directly estimating the optimal policy and maximizing reward through gradient ascent. The Critic, however, utilizes the value-based approach and learns how to estimate the value of different state-action pairs.

7.2.1 Algorithm Steps

The DDPG algorithm can be presented as follows:

1. The Actor and the Critic use *separate* neural networks.

2. Define the Actor network with
$$a = \mu(s, \theta^{\mu}) \tag{7.2}$$
which takes input as a state s and results in the action a where θ^{μ} is the Actor network learning weights. The actor here is used to approximate the optimal policy deterministically. That means that the output is the best believed action for any given state. This is unlike a stochastic policy (probability distribution) in which we want the policy to learn a probability distribution over the actions.
In DDPG, we want the believed best action every single time we query the actor network. The actor is basically learning the argmax of $\mathbf{Q}(s, a)$, which is the best action.

3. Define the Critic network
$$\mathbf{Q}\left(s; a, \theta^{Q}\right) \Rightarrow \mathbf{Q}\left(s; \mu(s; \theta^{\mu}), \theta^{Q}\right) \tag{7.3}$$
which takes an input as a state s and action a and returns the $\mathbf{Q}$ value where θ^{Q} is the Critic network weights.
The critic learns to evaluate the optimal action value function by using the actors best believed action.

4. We define a target networks $\mu(s; \theta^{\mu'})$, $\mathbf{Q}\left(s; a, \theta^{Q'}\right)$ for both the Actor network and Critic network respectively where $\theta^{\mu'}$, $\theta^{Q'}$ are the weights of the target Actor and Critic network.

5. Next, we perform the update of Actor network weights with policy gradients and the Critic network weight with the gradients calculated from the **TD** error.

6. In order, to select correct action, first we have to add an exploration noise N to the action produced by Actor $\mu(s; \theta^{\mu'}) + N$. The noise is added to encourage exploration since the policy is deterministic.

7. Selected action in a state s, receive a reward r and move to a new state s'.

8. We store this transition information in an experience replay buffer.

9. As it is performed while we use DQN algorithm, we sample transitions from the replay buffer and train the network, and then we calculate the target $\mathbf{Q}$ value
$$y_i = r_i + \gamma \mathbf{Q}'\left(s_{i+1}, \mu'\left(\theta^{\mu'} | \theta^{Q'}\right)\right) \tag{7.4}$$

10. Then, we can compute the **TD** error as
$$L = \frac{1}{M} \sum_{i} \left(y_i - \mathbf{Q}\left(s_i, a_i | \theta^{Q}\right)^2\right) \tag{7.5}$$

11. Subsequently, we perform the update of the Critic networks weights with gradients calculated from this loss L.

12. Update our policy network weights using a policy gradient.

13. Next, we update the weights of Actor and Critic network in the target network. In DDPG algorithm topology consist of two copies of network weights for each network, (Actor: regular and target) and (Critic: regular and target). In DDPG, the target networks are updated using a soft updates strategy. A soft update strategy consists of slowly blending regular network weights with target network weights. In means that every time step we make our target network be 99.99 percent of target network weights and only a 0.01 percent of regular network weights (slowly mix of regular network weights into target network weights).

14. We update the weights of the target networks (as show in equation 7.6) slowly, which promotes greater stability (soft updates strategy).

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{aligned} \tag{7.6}$$

7.2.2 DDPG Algorithm Pseudocode

The algorithm can be expressed in the form of a pseudocode as shown in Algorithm 1.

Algorithm 1 DDPG Algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$.
Initialize replay buffer R.
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration.
  Receive initial observation state s_1.
  for t=1 , T **do**
    Select action $a_t = \mu\left(s_t|\theta^\mu\right) + \mathcal{N}_t$ according to the current policy and exploration noise.
    Execute action a_t and observe reward r_t and new state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in R.
    Sample a random mini-batch of N transitions (s_t, a_t, r_t, s_{t+1}) from R.
    Set

$$y_i = r_i + \gamma \mathbf{Q}'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right)$$

    Update critic by minimizing the loss

$$L = \frac{1}{N}\sum_i \left(y_i - \mathbf{Q}\left(s_i, a_i|\theta^Q\right)^2\right)$$

    Update the actor policy using the sampled policy gradient

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \left(\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \cdot \nabla_{\theta^\mu}\mu\left(s|\theta^\mu\right)\right)$$

    Update the target networks

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for