EESM STBC Link Simulations for 802.11n MISO Systems

Author

Maria Bantouraki

Thesis Committee

Prof. Karystinos Georgios (Supervisor)

Prof. Liavas Athanasios

Prof. Christopoulos Dionysios

**A thesis submitted in fulfillment
of the requirements for the degree of
Diploma in Electrical and Computer Engineering**

TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Technical University of Crete
Chania, Greece
August 1, 2024

Abstract

This thesis is on the IEEE 802.11n standard, also known as Wi-Fi 4, which is the first Wi-Fi standard to introduce multiple-input multiple-output systems. It is based on orthogonal frequency division multiplexing (OFDM) and supports space-time block coding (STBC). To meet the need for selecting in real-time the appropriate modulation and coding scheme from the available set of schemes that the standard offers, we examine the use of the exponential effective signal-to-noise-ratio (SNR) mapping method, known as the EESM method. The EESM method provides a function that maps the SNRs of the OFDM subcarriers into one single SNR value, which has the same system performance when evaluated at the single-carrier additive-white-Gaussian-noise channel. Specifically, in the thesis we develop the EESM method for a system with two transmit antennas and one receive antenna, employing STBC at the transmitter, and utilizing the IEEE TGn Model-B channel. We evaluate the performance of the EESM method in this system and show that EESM provides, in real-time and with low complexity, reliable estimates of the error rate of the modulation and coding schemes that are available by the IEEE 802.11n standard.

Acknowledgements

I would like to express my gratitude to my supervisor Prof. Georgios Karystinos for his constant guidance and support throughout the completion of this thesis.
I would also like to thank my thesis committee, Prof. Athanasios Liavas and Prof. Dionysios Christopoulos, for their contribution to assessing my thesis.
Lastly, I am deeply grateful to my family, my friends, and my partner for their unconditional support and encouragement.

Contents

List of Tables

List of Figures

List of Abbreviations

AWGN	Additive White Gaussian Noise
BCC	Binary Convolutional Coding
BPSK	Binary Phase Shift Keying
CFO	Carrier Frequency Offset
CSD	Cyclic Shift Delay
EESM	Exponential Effective SNR Mapping
ESM	Effective SNR Mapping
FEC	Forward Error Correction
FFT	Fast Fourier Transform
HT	High Throughput
HT-LTF	High Throughput Long Training Field
HT-SIG	High Throughput Signal Field
HT-STF	High Throughput Short Training Field
ICI	Inter-Carrier Interference
ISI	Inter-Symbol Interference
L-LTF	Legacy Long Training Field
L-SIG	Legacy Signal Field
L-STF	Legacy Short Training Field
LLR	Log Likelihood Ratio
MAC	Medium Access Control

MCS	Modulation Coding Scheme
MIMO	Multiple Input - Multiple Output
MISO	Multiple Input - Single Output
MMSE	Minimum Mean Square Error
NLOS	Non-line-of-sight
OFDM	Orthogonal Frequency-Division Multiplexing
PHY	Physical Layer
PPDU	Physical layer Protocol Data Unit
PSDU	Physical layer Service Data Unit
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
SISO	Single Input - Single Output
SNR	Signal to Noise Ratio
STBC	Space - Time Block Coding
WLAN	Wireless Local Area Networks

Chapter 1

Introduction

1.1 Motivation

A wireless local-area network (WLAN) is a computer network that provides a link among several devices based on radio transmissions instead of wired connections. The most broadly used computer networks are the WLANs based on the IEEE 802.11 Standard, which is commonly known as Wi-Fi. In recent years, reliable and high-performance Wi-Fi is a fundamental requirement, as it is becoming increasingly important in everyday life. This can be achieved by increasing the channel bandwidth, the data rate, the constellation size, and the number of antennas, as well as by employing different techniques. Space-time block coding (STBC) is a technique used to enhance the reliability of data transfer by transmitting a data stream across several antennas and utilizing the different received versions of the data.

Nonetheless, the increase in the performance of the network results in an increase in complexity. Consequently, link-level simulations and the evaluation of the system become significantly time-consuming. The exponential effective SNR mapping (EESM) method is a method used to considerably reduce the time complexity of such simulations for systems that employ the orthogonal frequency-division multiplexing (OFDM) technique. The objective of this thesis is to provide accelerated simulations for 802.11n systems with two transmit antennas and one receive antenna that support STBC with the use of EESM.

1.2 Thesis Outline

The rest of this thesis is organized as follows:

- **Chapter 2** contains a description of the IEEE 802.11n Standard.
- **Chapter 3** analyzes the 802.11n Transmitter and Receiver and their components in detail.
- **Chapter 4** provides a thorough description of Space-time block coding.
- **Chapter 5** presents the exponential effective SNR mapping (EESM) algorithm, along with the results from the simulations.
- **Chapter 6** summarizes the main findings of this thesis.

Chapter 2

Analysis of the IEEE 802.11n Standard

2.1 Background

The IEEE 802.11 standards define the protocols for implementing WLAN communications. The standard was first introduced in 1997 and has undergone several revisions and updates to improve performance, security, and interoperability. Key versions include 802.11a, 802.11b, 802.11g, 802.11n, and the latest 802.11ax, each introducing new features and enhancements to address the growing demands for wireless communication.

The IEEE 802.11 standard defines a protocol stack that includes the PHY and MAC layers. The PHY layer is responsible for the transmission and reception of data over the wireless medium, including modulation, encoding, and decoding of signals. The MAC layer manages access to the wireless medium, handling frame formatting, error control, and addressing, ensuring efficient communication without collisions. The interaction between these layers is crucial for achieving reliable and high-performance wireless communication.

IEEE 802.11n is an amendment to the IEEE 802.11 wireless networking standard which increased maximum data rates to 600 Mbps by using four spatial streams at a channel width of 40 MHz and improving the PHY and MAC layer efficiency. The evolution from previous standards such as IEEE 802.11a/b/g incorporated multiple new features including:

- **MIMO:** Enhances data throughput and range by using multiple transmit and receive antennas.
- **Channel Bonding:** Combines two adjacent 20 MHz channels to create a 40 MHz channel, doubling the maximum data rate.
- **Frame Aggregation:** Increases throughput by sending multiple data frames in a single transmission.

2.2 Orthogonal Frequency Division Multiplexing (OFDM)

The IEEE 802.11n standard employs OFDM to achieve high data rates. OFDM is a multi-carrier modulation technique that divides a high-rate data stream into multiple lower-rate streams, which are then transmitted over several orthogonal subcarriers. This approach decreases the impact of multipath fading and ISI by ensuring that each subcarrier experiences nearly flat fading, simplifying channel equalization. Here's an explanation of the key concepts:

- **Subcarriers:** OFDM divides the spectrum into subcarriers that are spaced apart at precise frequencies. Typically, 64 subcarriers for a 20 MHz channel. Of these, 4 are utilized as pilot subcarriers for synchronization, phase tracking, and channel estimation. Additionally, 52 subcarriers are used for data transmission. The remaining 8 subcarriers are mainly used as guard carriers or null subcarriers against interference from adjacent channels or sub-channels, thus minimizing ICI.
- **Orthogonality:** Ensures that subcarrier signals are orthogonal to each other, reducing interference. The total bandwidth is divided into multiple narrower subbands, each carrying a separate subcarrier signal. The subcarriers are orthogonal, meaning the integral of the product of any two different subcarriers over one symbol period is zero. Mathematically, this can be expressed as:

$$\int_0^T e^{j2\pi f_i t} \cdot e^{-j2\pi f_j t}\, dt = 0 \quad \text{for} \quad i \neq j$$

where f_i and f_j are the frequencies of the i-th and j-th subcarriers, respectively, and T is the OFDM symbol period. This leads to efficient spectrum usage because subcarriers can overlap in the frequency domain without interfering with each other.

- **FFT:** Transforms a signal from the time domain to the frequency domain. FFT is used to modulate and demodulate the data into and out of these subcarriers.

We should note that between OFDM symbols, a guard interval with a cyclic prefix is inserted to prevent ISI.

2.3 Physical Layer (PHY) and Setup

The IEEE 802.11n Standard features 32 different modulation and coding schemes (MCS), that define the modulation, the coding rate, and the number of spatial streams. The following table displays the features of the MCS that will be simulated in this thesis. Only one spatial stream is used, channel bandwidth is set to 20 MHz and guard interval (GI) duration is 800ns.

MCS	Modulation	Coding Rate	Data rate(Mb/s)
0	BPSK	1/2	6.5
1	QPSK	1/2	13.0
2	QPSK	3/4	19.5
3	16-QAM	1/2	26.0
4	16-QAM	3/4	39.0
5	64-QAM	2/3	52.0
6	64-QAM	3/4	58.5
7	64-QAM	5/6	65.0

Table 2.1: Modulation and Coding Schemes.

In 802.11n three different formats are supported: the Non-HT format, the HT-mixed format, and the HT-greenfield format. Figure 2.1 showcases the structure

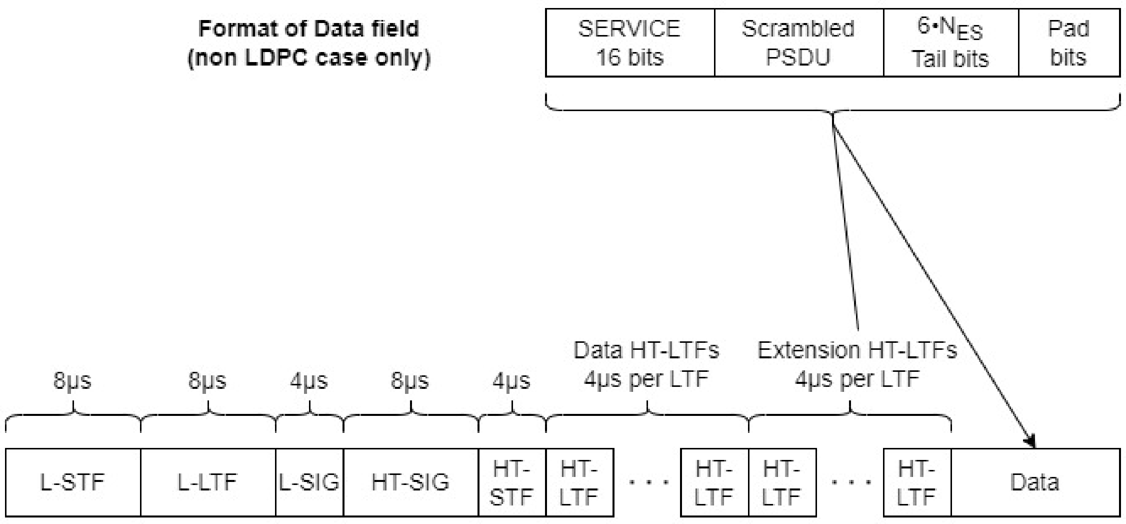

Figure 2.1: HT-mixed PPDU format.

of the HT-mixed physical layer protocol data unit (PPDU), which is the one used in this thesis.

The non-HT part of the preamble contains a legacy short training field (L-STF), a legacy long training field (L-LTF), and a legacy signal (L-SIG) field. The L-STF is used for carrier frequency offset (CFO) estimation and packet detection. The L-LTF is used for fine CFO estimation and the L-SIG field conveys information about rate and length.

The HT portion of the preamble consists of the HT-SIG, HT-STF, and HT-LTF fields. The HT-SIG field contains the information needed for the HT packet formats to be interpreted, for instance, information about the MCS or the channel bandwidth. The HT-STF improves automatic gain control estimation in MIMO systems. The HT-LTF is used for channel estimation at the receiver. More details about the fields of the preamble can be found in Sections 19.3.9.3 and 19.3.9.4 of [6].

Two different forward error correction code encoders are supported in the IEEE 802.11n Standard, the binary convolutional code (BCC) and the low-density parity-check (LDPC) code. In this thesis, only the BCC Encoder is discussed.

The IEEE 802.11n Standard was the first to introduce MIMO support with up to 4 transmit and receive antennas. The a$\times$b: c notation helps identify the capabilities of a system. The first number (a) is the number of transmit antennas, the second number (b) is the number of receive antennas and the third number (c) is the number of data spatial streams. In this thesis, a MISO 2x1 system with one spatial stream (2x1: 1) will be implemented.

Chapter 3

Analysis of IEEE 802.11n Transmitter

3.1 Transmitter

In this chapter, the 802.11n transmitter will be analyzed. The structure of the transmitter for a SISO system is displayed in Figure 3.1.

3.1.1 Information bits generation

According to Section 19.3.11.2 of [6], 16 service bits are generated and set to 0. To simulate the transmission of the information bits, random bits (PSDU) are generated. Service bits and information bits are combined to simulate the transmitted signal denoted as the DATA field in the standard.

3.1.2 Scrambling

A frame-synchronous scrambler is used to scramble and descramble the binary input signal. This function is commonly used in the transmitter and receiver of a Wi-Fi IEEE 802.11n system to add a certain level of randomness and enhance the security and reliability of the transmitted data. The scrambling method is described in 17.3.5.5 of [6].

function y = wlanScramble(x, scramInit)

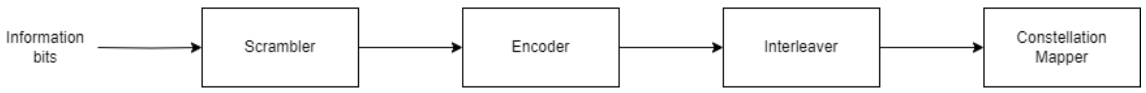

Figure 3.1: Transmitter structure for a SISO system.

`Input Parameters`

- `x`: The input binary data that needs to be scrambled. It can be a binary column vector or a matrix of type 'int8' or 'double'.
- `scramInit` : The initial state of the scrambler. It can be an integer between 1 and 127 (inclusive), or a corresponding 7-by-1 column vector of binary bits of type 'int8' or 'double'.

`Output`
The function returns the scrambled (or descrambled) binary data as a binary column vector or matrix of the same size and data type as the input x. The output y contains the scrambled data.
`Function Overview`
The function first determines the data type of the input x and performs validation to ensure that the provided x is binary data and has two dimensions (column vector or matrix). It also validates the scramInit to ensure it meets the requirements specified in the IEEE 802.11-2020 standard.
After that, it generates a scrambling sequence of length 127 using the generator polynomial defined in Section 17.3.5.5 of [6], based on the provided scramInit. This scrambling sequence is used to add randomness to the input data.
Then the scrambling process takes place, in which the input data is divided into frames of size 127 (or less if the input is smaller). For each frame, the function performs XOR operations with specific bits of the scrambling sequence to scramble the data, according to the polynomial $S(x) = x^7 + x^4 + 1$. The scrambling sequence is updated for each frame, creating a frame-synchronous scrambling process. We should note here that each column of the input is scrambled independently with the same initial state. The same scrambler structure is used for scrambling at the transmitter and descrambling at the receiver.

3.1.3 Encoding

Convolutional coding is a method of adding redundancy to the data before transmission to enhance the reliability of wireless communication. This redundancy enables the receiver to recover the original data even if some bits are corrupted during transmission. In [6], specific guidelines are outlined in sections 17.3.5.6, 19.3.11.4, and 19.3.11.6 for the BCC encoder used in the transmitter. Each input data bit is processed along with the previous bits in a shift register based on the generator polynomials. A specific function is used to convert a convolutional code polynomial representation to a trellis structure, as described below.

function y = wlanBCCEncode(x, rate)

`Input Parameters`

- `x`: The binary input data to be encoded. It can be a binary matrix of class 'int8' or 'double', where each column represents a separate stream to be encoded.
- `rate`: The coding rate is specified as a scalar, character vector, or string scalar. It can be 1/2, 2/3, 3/4, or 5/6.

`Output`

The function returns the binary matrix y, which contains the binary convolutionally encoded bits of the input data x. The number of rows of y is equal to the result of dividing the number of rows of input x by the specified rate, rounded to the next integer. The number of columns of y is equal to the number of columns of x.

`Function Overview`

The function begins by calling the `poly2trellis` function, which returns the state transition diagram of the decoder based on the specified parameters. According to IEEE Std 802.11-2020, the parameters are a constraint length (the number of previous input bits that affects the generation of each output bit), which is set to 7, and a code generator vector [133 171] representing the octal numbers of the generator polynomials.

Subsequently, the function performs various validations and assignments. Additionally, it utilizes puncturing patterns, if necessary, to achieve different rates of

convolutional encoding. These puncturing patterns are predefined according to the IEEE 802.11-2020 standard, the sections referenced earlier.
The encoding process is performed for each encoded stream in x using the `convenc function`. This function initializes the state of the trellis to zero and, following certain validations, it calls a built-in function (`convcore`) specifically designed for encoding.
function trellis = poly2trellis(constraintLength, codeGenerator)
`Input Parameters`

- `constraintLength` : A $1 \times k$ vector specifying the number of delays for each of the k input bit streams. This determines the "memory" or "order" of the convolutional encoder.
- `codeGenerator` : A $k \times n$ matrix of octal numbers: Specifies the n output connections for each of the k inputs. These numbers represent the coefficients of the polynomials used in the convolutional encoder.

`Output`
The output `trellis` is a structure containing the following fields:

- `numInputSymbols`: The number of input symbols, which is 2^k for a rate k/n code.
- `numOutputSymbols`: The number of output symbols, which is 2^n for a rate k/n code.
- `numStates`: The number of states in the trellis.
- `nextStates`: A matrix representing the next state transition from the current state for each possible input symbol. The rows represent the states and the columns represent the input bits.
- `outputs`: A matrix representing the output generated for each possible input symbol transition. Again the rows represent the states and the columns represent the input bits.

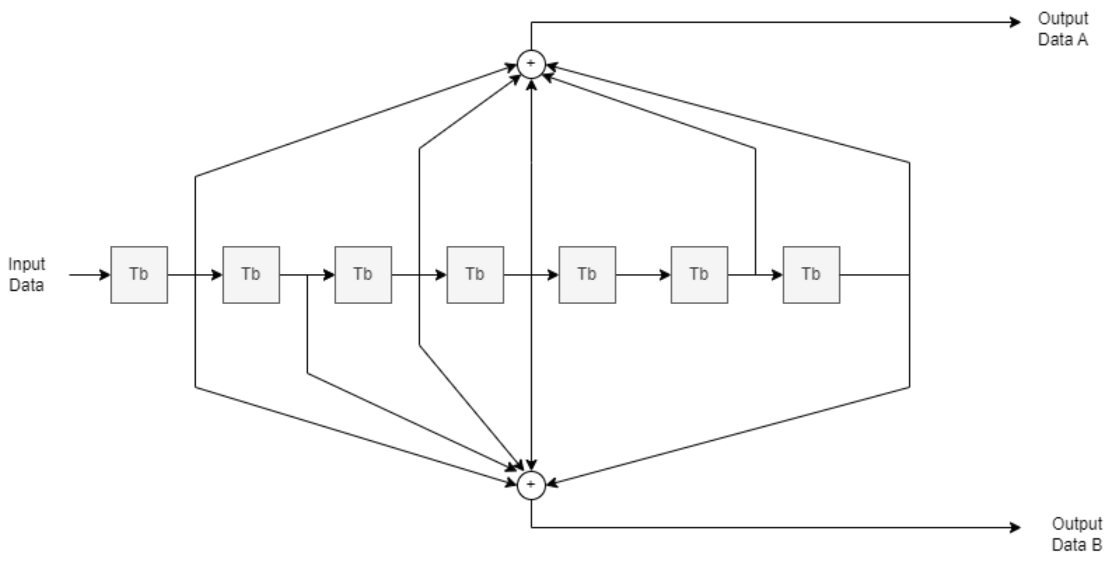

Figure 3.2: Example encoding procedure.

`Function Overview`

This function is responsible for converting the polynomial representation of the convolutional encoding to a trellis structure. As mentioned earlier, the call of this function is poly2trellis(7, [133 171]). The dimensions of the parameters indicate that, in the absence of a puncturing pattern, the encoding rate is $\frac{1}{2}$. The structure of the encoding process is shown in figure 3.2. It's important to note that the calculation of each output bit involves 7 bits: 6 previous bits and 1 input bit. For the first output bit, labeled as A, the polynomial $133_8 = 1011011_2$ is used, while for the second output bit, B, the polynomial $171_8 = 1111001_2$ is utilized.

3.1.4 Interleaving

Interleaving is a technique used to enhance the robustness of data transmission. By rearranging the order of transmitted symbols, it minimizes the impact of burst errors and fading effects, which are common in wireless channels. The permutations that take place are described in Sections 17.3.5.7 and 19.3.11.8 of [6]. These permutations help to shuffle the data bits creating a more uniform distribution of the errors.

The first permutation maps adjacent coded bits onto non-adjacent subcarriers.

$$i = N_{\mathrm{ROW}}(k \bmod N_{\mathrm{COL}}) + \left\lfloor \frac{k}{N_{\mathrm{COL}}} \right\rfloor, \qquad k = 0, ..., N_{\mathrm{CBPSS}}(i_{\mathrm{SS}}) - 1. \tag{3.1}$$

The second permutation maps adjacent coded bits alternately onto less and more significant bits of the constellation, to avoid long runs of low-reliability bits.

$$j = s(i_{\mathrm{SS}}) \cdot \left\lfloor \frac{i_{\mathrm{SS}}}{s(i_{\mathrm{SS}})} \right\rfloor + \Big(i + N_{\mathrm{CBPSS}}(i_{\mathrm{SS}}) - \left\lfloor N_{\mathrm{COL}} \frac{i}{N_{\mathrm{CBPSS}}(i_{\mathrm{SS}})} \right\rfloor \bmod s(i_{\mathrm{SS}}) \Big), \quad i = 0, ..., N_{\mathrm{CBPSS}}(i_{\mathrm{SS}}) - 1. \tag{3.2}$$

If there is more than one spatial stream, the third operation is applied and performs a frequency rotation to the additional spatial streams.

$$r = \Big(j - \Big((2(i_{\mathrm{SS}} - 1)) \bmod 3 + 3 \left\lfloor \frac{i_{\mathrm{SS}} - 1}{3} \right\rfloor \cdot N_{\mathrm{ROT}} \cdot N_{\mathrm{BPSCS}}(i_{\mathrm{SS}}) \Big) \Big) \bmod N_{\mathrm{CBPSCS}}(i_{\mathrm{SS}}), \quad j = 0, ..., N_{\mathrm{CBPSS}}(i_{\mathrm{SS}}) - 1. \tag{3.3}$$

In this thesis, only MCS with one spatial stream are used, so the third permutation is ignored.

Parameter	20MHz
$\mathrm{N_{COL}}$	13
$\mathrm{N_{ROW}}$	$4 \cdot \mathrm{N_{BPSCS}(i_{SS})}$
$\mathrm{N_{ROT}}$	11

Table 3.1: Block interleaver parameters

function y = wlanBCCInterleave(x, type, numCBPSSI)

`Input Parameters`

- `x`: The input binary data to be interleaved. It should be a 3D array of type ‘int8’ or ‘double’ with dimensions (Ncbpssi * Nsym) x Nss x Nseg. Ncbpssi is the number of coded bits per OFDM symbol per spatial stream per interleaver block, Nsym is the number of OFDM symbols, Nss is the number of spatial streams, and Nseg is the number of segments.

- `type`: A character vector or string specifying the type of interleaving to be performed. It can be either 'Non-HT' or 'VHT', in our case is set to 'VHT'.
- `numCBPSSI`: The number of coded bits per OFDM symbol per spatial stream per interleaver block. It is a positive integer scalar and depends on the specific standard used. In our case, it is set to 52, like the number of data subcarriers.
- `chanBW`: A character vector or string with the channel bandwidth. It must be one of: 'CBW1', 'CBW2', 'CBW4', 'CBW8', 'CBW16', 'CBW20', 'CBW40', 'CBW80', or 'CBW160' where the integer following CBW denotes the bandwidth in MHz. This parameter is optional because in 'Non-HT' interleaver type it is not required.

`Output`
The output variable y is a (Ncbpssi * Nsym) x Nss x Nseg array of the same class as the input x, containing the binary convolutionally interleaved data. (Nss and Nseg are equal to 1)
`Function Overview`
At first, the function validates the input arguments and determines the channel bandwidth (chanBW) and the number of coded bits per subcarrier per spatial stream (numBPSCS) based on the interleaver type.
The function then calculates the interleaving parameters (Ncol, Nrow, Nrot) based on the type, numCBPSSI, numBPSCS, and chanBW. It performs binary convolutional interleaving on the input data using the calculated parameters with the function `bccInterleaveCore`, which applies the three interleaving permutations stated in the standard.
In our case, type is set to 'VHT', and numCBPSSI is 52, like the number of data subcarriers. Inside `wlanBCCInterleave`, the internal function `bccInterleaveCore` is called, which applies the three interleaving permutations stated above.

3.1.5 Constellation Mapping

Constellation mapping is called the procedure of mapping the interleaved bits to constellation points (complex symbols) based on the MCS. The modulations that IEEE 802.11 standard support are BPSK, QPSK, and QAM, specifically 16-QAM and 64-QAM.

function y = wlanConstellationMap(x, numBPSCS)

`Input Parameters`

- `x`: X is a binary 'int8' or 'double' vector, matrix, or multidimensional array containing the input bits to map into symbols. The number of rows of X must be an integer multiple of NUMBPSCS.
- `numBPSCS`: The number of coded bits per subcarrier per spatial stream. It is an integer, whose values can be 1 (BPSK), 2 (QPSK), 4 (16-QAM), 6 (64-QAM) and is equal to $\log_2(M)$, where M is the modulation order.

`Output`

The output y is a complex vector, matrix, or multidimensional array containing the mapped symbols. It has the same size as the input x except for the number of rows, which is equal to the number of rows of x divided by numBPSCS.

`Function Overview`

The function performs constellation mapping, after doing some validations for the inputs, depending on the modulation scheme (numBPSCS). For BPSK modulation (numBPSCS=1), the constellation is [-1, 1], and the binary input bits are mapped to the corresponding complex symbols ($0 \rightarrow -1$ and $1 \rightarrow 1$). For other modulation schemes (numBPSCS>1), the function uses an internal function, 'comm.internal.wlanQAMSymbolMap', to get the symbol mapping. The symbol mapping is a vector containing the decimal representation of the modulation scheme as shown in figure 17-10 Section 17.3.5.8 of [6]. It utilizes MATLAB's 'comm.internal.qam.modulate' function to perform constellation mapping of the input x into complex symbols. If the input x has more than three dimensions, the function reshapes it into a column to support the 'comm.internal.qam.modulate' function, and then reshapes it back to the original size after the mapping. If the phase argument is provided as a third input, the function rotates the constellation points counter-clockwise by the specified amount in radians. The mapping is performed column-wise for the input x.

After constellation mapping, STBC can be applied if the number of space-time streams N_{STS} is greater than the number of spatial streams N_{SS}. This case is described in depth in Section 4.

3.1.6 OFDM Modulation

OFDM modulation is performed by MATLAB's **wlanOFDMModulate**, which is a function that performs OFDM modulation on the frequency domain signal and returns the time-domain signal.
function y = wlanOFDMModulate(x,cplen,varargin)
`Input Parameters`

- `x`: X is the Nc-by-Nsym-by-Nt frequency-domain signal, where Nc represents the number of frequency subcarriers, Nsym represents the number of OFDM symbols, and Nt represents the number of antennas in the spatial-domain.
- `cplen`: A non-negative, integer scalar representing the cyclic prefix length for all OFDM symbols.

`Output`
The output y is the modulated time-domain signal.
`Function Overview`
The function performs OFDM modulation, by applying the inverse Fourier transform to the input signal.

3.1.7 Example

Figure 3.3 represents a SISO system with a bandwidth of 20 MHz. The modulation used is 16-QAM (cfgHT.MCS = 4) with a coding rate of 3/4. To illustrate the information bits, 8000 random bits are generated in PSDU. After that, service, tail, and pad bits are added according to Section 17.3.5 of [6]. Data is then scrambled and encoded with a rate of 3/4 and the number of rows is the result of 8112/(3/4).

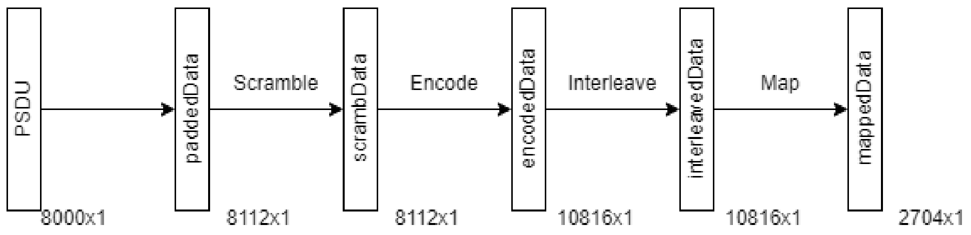

Figure 3.3: Example receiver procedure.

Consequently, the encoded data is interleaved as described above and then, the interleaved is mapped according to the modulation. The number of rows is 10816/4 where 4 represents the number of coded bits per subcarrier per spatial stream.

3.2 Signal Transmission

The channel generation process emulates the wireless communication channel to replicate real-world wireless propagation conditions.

3.2.1 Channel Generation

Firstly, a HT format configuration object, that contains the transmit parameters for the HT-Mixed Format, has to be created. This is accomplished by calling MATLAB's **wlanHTConfig** function.
cfgHT = wlanHTConfig; The following properties are set:

- The channel bandwidth is set to 20 MHz.
- The number of transmit antennas is set to the desired number.
- The number of space-time streams is set to the desired number.
- PSDU length in bytes, specified as an integer in the interval [0, 216 – 1].
- MCS is set to the desired MCS.
- Channel Coding method is set to BCC.
- Spatial Mapping Method is set to Fourier.

The next step is the configuration of the channel. IEEE TGn multipath fading channel is used, especially Model - B [7]. The distance between the transmitter and the receiver is set to 10 m., so that the model is NLOS. For this purpose, MATLAB's **wlanTGnChannel** is utilized and it creates a system object for the TGn Channel, as specified by the IEEE 802.11 WLAN Working group. The following parameters are set:
tgnChannel = wlanTGnChannel;
tgnChannel.DelayProfile = 'Model-B';
tgnChannel.NumTransmitAntennas = cfgHT.NumTransmitAntennas;
tgnChannel.NumReceiveAntennas = NumRxAnts;
tgnChannel.TransmitReceiveDistance = 10;
tgnChannel.LargeScaleFadingEffect = 'None';
tgnChannel.RandomStream = 'mt19937ar with seed';
tgnChannel.Seed = double(seeds(channelIndex,1));
tgnChannel.PathGainsOutputPort = 1;

3.2.2 Waveform Transmission

The generated waveform is passed through the TGn channel model (wlanTGnChannel) to simulate the effects of the wireless channel, including path loss, multipath fading, and noise. It is important to note that in this case, the received signal is noiseless. Noise can be added after the signal is received. It is important to recognize that the utilization of wlanTGnChannel yields a channel that exhibits temporal variation, a characteristic that can introduce complications in the simulation process. In order to avoid that, we use the first path gain generated by wlanTGnChannel across all time instances. Subsequently, the waveform is propagated through this static channel.

3.3 Receiver

3.3.1 Packet Detection

Packet detection is the initial step in the reception process, where the presence of a valid packet is identified within the received signal. According to the IEE802.11

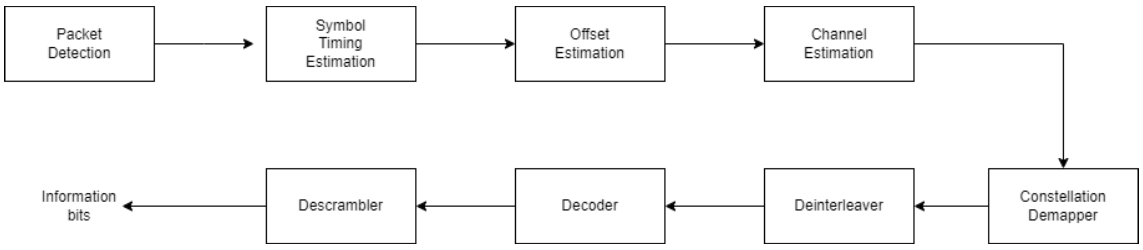

Figure 3.4: Receiver structure of a SISO system.

standard, preamble detection is used for this purpose. Specifically, the L-STF segment of the preamble is used to compute the offset from the beginning of the input waveform to the detected preamble's start using auto-correlation.

function startoffset= wlanPacketDetect(x, chanBW)

`Input Parameters`

- `x`: A single or double Ns-by-Nr matrix of real or complex samples, representing the received time-domain signal. Ns stands for the number of time domain samples and Nr for the number of receive antennas. In SISO systems, Nr is equal to 1.
- `chanBW`: A character vector or string describing the channel bandwidth. Its value can be 'CBW5', 'CBW10', 'CBW20', 'CBW40', 'CBW80', 'CBW160', or 'CBW320'. For our system 'CBW20' is used.

`Output`

The function returns an integer scalar indicating the location of the start of a detected packet as the offset from the start of the matrix X. If no packet is detected an empty value is returned.

`Function Overview`

At first, validations are performed and the length of the L-STF symbol is computed. Afterwards zeros are added to the input signal, to make its length equal to half of the length of the L-STF. The input waveform is then separated into blocks of half of the L-STF length so that it can be processed gradually. Packet detection is accomplished with the use of the function `correlateSamples`, which is also defined in wlanPacketDetect.m file. A loop is used, so that the input rxSig is each time the block of the original input signal X that we are processing.

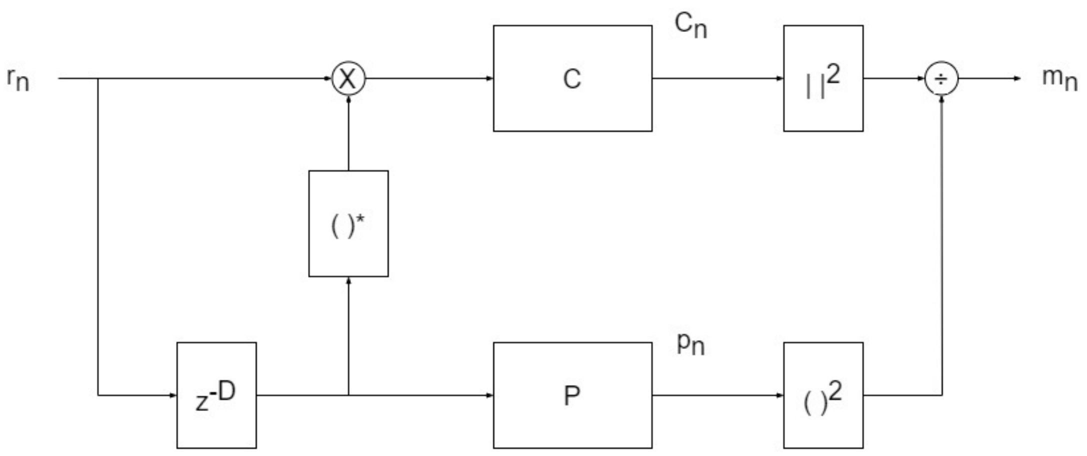

Figure 3.5: Packet detection algorithm implemented by MATLAB.

function [packetStart,Mn] = correlateSamples(rxSig, symbolLength, lenL-STF, threshold)

`Function Overview`

The received signal, rxSig, is delayed and then correlated in two sliding windows independently. The packet detection processing output provides decision statistics (m_n) of the received waveform. Figure 3.5, provided by MATLAB's Documentation for `wlanPacketDetect`, demonstrates how the function `correlateSamples` works.

Window C auto-correlates between the received signal and the delayed version, C_n.

$$C_n = \sum_{l=1}^{Nr} \sum_{K=0}^{D-1} r_{n+k,l} r^*_{n+k+D,l} \tag{3.4}$$

D is the period of the L-STF short training symbols.

Window P calculates the energy received in the auto-correlation window, P_n.

$$P_n = \sum_{l=1}^{Nr} \sum_{K=0}^{D-1} |r_{n+k+D,l}|^2 \tag{3.5}$$

The decision statistics, M_n, normalize the auto-correlation by P_n so that the decision statistic is not dependent on the absolute received power level.

$$m_n = \frac{|c_n|^2}{{p_n}^2} \tag{3.6}$$

The values of M_n that are not greater than the selected threshold, which in our case is the default threshold that is equal to 0.5, are discarded. Lastly, the relative distance between the first peak and subsequent peaks is checked, so that it does not exceed three times the symbol length. If it does, then no packet is detected. Each peak corresponds to the detection of a single packet. The output argument packetStart of the function `correlateSamples` is also the output argument startoffset of the function `wlanPacketDetect`.
It should be mentioned that the algorithm presented above is based on the fact that the L-STF sequences have good correlation properties and low peak-to-average power, as stated in Section 3.1.3. of [8]

3.3.2 Offset Estimation

For accurate data recovery, synchronization between the transmitter and the receiver is essential. In the IEEE 802.11 standard, offset estimation is responsible for aligning the received signal with the timing reference established by the transmitter. A correlation-based method and preamble fields are used for these timing offsets, and several functions are employed to achieve this. Firstly, the `wlanCoarseCFOEstimate` performs coarse CFO estimation using the L-STF of the preamble, as detailed in [9]. Following coarse CFO estimation, the function `helperFrequencyOffset`, a modified version of MATLAB's `FrequencyOffset` which is adjusted so that it can also be used for MIMO systems, is called. For our purposes, these adjustments can be disregarded. Subsequently, the function `wlanSymbolTimingEstimate` is utilized for the fine symbol timing estimation with the help of the L-LTF field of the preamble. Finally, fine CFO estimation is performed using `wlanFineCFOEstimate` function after symbol timing estimation.
function foffset = wlanCoarseCFOEstimate(in,chanBW)
`Input Parameters`

- `in`: A single or double complex matrix of size Ns-by-Nr. Ns is the number of time domain samples in the L-STF, and Nr is the number of receive antennas.

- `chanBW`: A character vector or string describing the channel bandwidth. Its value can be 'CBW5', 'CBW10', 'CBW20', 'CBW40', 'CBW80', 'CBW160', or 'CBW320'. For our system 'CBW20' is used.

`Output`
The function returns a double scalar representing the carrier frequency offset in Hertz.
`Function Overview`
After several validations are performed, the number of samples inside the L-STF is calculated. Samples are then extracted and the parameters for the estimation are set, assuming 4 repetitions per FFT period. The internal function `cfoEstimate` is called. CFO estimation is then performed and CFO is computed based on the phase of delay correlation of weighted signals.
function out = helperFrequencyOffset(in, fs, foffset)
`Input Parameters`

- **in**: A complex 2D array.
- **fs**: A double scalar representing the sampling rate in Hertz.
- **foffset**: A double scalar containing the frequency offset to be applied to the input in Hertz.

`Output`
The function returns the frequency-offset output, which is the same size as 'in'.
`Function Overview`
The function creates a vector t of time samples representing the time axis of the input signal. This time vector is used to modulate the input signal with a complex exponential, effectively applying the frequency offset. For each antenna (column) in the input signal, it applies the frequency offset correction using the complex exponential term $e^{i \cdot 2\pi \cdot f_{offset} \cdot t}$ which rotates the phase of each sample in the input signal by an amount proportional to the time index t and the frequency offset f_{offset}. This results in a frequency shift in the time domain, compensating for the carrier frequency offset.
function startOffset = wlanSymbolTimingEstimate(x, chanBW)
`Input Parameters`

- `x`: A single or double matrix of size Ns-by-Nr, containing real or complex values. This is the received time-domain signal.

- `chanBW`: A character vector or string describing the channel bandwidth. Its value can be 'CBW5', 'CBW10', 'CBW20', 'CBW40', 'CBW80', 'CBW160', or 'CBW320'. For our system 'CBW20' is used.

`Output`
The function returns an integer within the range [-L, Ns-2L], where L denotes the length of the L-LTF.
`Function Overview`
After the proper validations, the threshold is set to 1 by default, if not given as an input parameter. The threshold is used to determine the decision metric's level, indicating a successful symbol timing estimate. Then the L-LTF is computed based on the chanBW and the cross-correlation between x and L-LTF is computed. The decision metric M is computed based on the cross-correlation results. It measures the similarity between the received signal and the L-LTF, indicating potential symbol timing positions. Then, it takes into account the CSD, and startOffset is returned which is the timing offset between the start of the input waveform and the estimated start of L-STF.
function foffset = wlanFineCFOEstimate(in,chanBW)
`Input Parameters`

- `in`: A single or double complex matrix of size Ns-by-Nr, containing real or complex values. Ns is the number of time domain samples in the L-LTF of the preamble.

- `chanBW`: A character vector or string describing the channel bandwidth. Its value can be 'CBW5', 'CBW10', 'CBW20', 'CBW40', 'CBW80', 'CBW160', or 'CBW320'. For our system 'CBW20' is used.

`Output`
The function returns a double scalar representing the carrier frequency offset in Hertz.

`Function Overview`
This function operates the same calculations as `wlanCoarseCFOEstimate`, however, L-LTF is used, rather than L-STF. In this case, also 2 repetitions per FFT period are assumed. Fine CFO has to be applied to the input signal with the use of `helperFrequencyOffset`.

3.3.3 Channel Estimation

Channel estimation allows the receiver to accurately estimate the channel characteristics to minimize the effects of channel impairments, such as fading and interference. To perform channel estimation, the demodulated HT-LTF field is utilized, which is extracted from the received time-domain HT-LTF of the preamble with the help of `wlanHTLTFDemodulate` function. The demodulated signal is parsed as an input to the function `wlanHTLTFChannelEstimate` that performs channel estimation between all space-time, extension streams, and receive antennas.
function y = wlanHTLTFDemodulate(rx, cfgHT)
`Input Parameters`

- `rx`: A complex matrix containing the received time-domain HT-LTF signal. It is of size Ns-by-Nr.
- `cfgHT`: A packet format configuration object for the WLAN HT packet format.

`Output`
The function returns a complex matrix or 3-D array of size Nst-by-Nsym-by-Nr, that represents the frequency-domain signal corresponding to the HT-LTF. Nst is the number of data and pilot subcarriers and Nsym is the number of OFDM symbols in the HT-LTF.
`Function Overview`

First, some validations are performed. Afterwards, the HT-LTF sequence, as well as the OFDM configuration are obtained. After that, the internal function `legacyOFDMDemodulate` is called, which also calls another internal function, the `ofdmDemodulate`, which performs the demodulation by applying FFT to the input signal. Non-active subcarriers are then discarded and lastly, the remaining signal is denormalized and phase rotation is removed from the subcarriers. The demodulated signal can now be returned.

function est = wlanHTLTFChannelEstimate(rxSym,cfgHT)

`Input Parameters`

- `x`: A complex array containing demodulated HT-LTF OFDM symbols. Array size is Nst-by-Nsym-by-Nr.
- `cfgHT`: A packet format configuration object for the WLAN HT packet format.

`Output`

The function returns a complex array containing the estimated channel at data and pilot subcarriers. The size of the array is Nst-by-(Nsts+Ness)-by-Nr.

`Function Overview`

Inside the function, several validations are made and then after getting the OFDM configuration the internal function `htltfEstimate` is called. For SISO systems, Least Squares (LS) estimation is used. This means that we get the channel estimate by dividing the received signal by the input signal, in our case the HT-LTF part of the preamble.

3.3.4 Data Recovery

3.3.4.1 OFDM Demodulation

OFDM demodulation is performed by MATLAB's **wlanOFDMDemodulate**, which is a function that performs OFDM demodulation on the time-domain signal and returns the frequency domain signal, separating the data from the pilot subcarriers.

function [data, pilot] = wlanOFDMDemodulate(x, cfgOFDM, ofdmSymOffset)

`Input Parameters`

- `x`: X is the data part of the time-domain received signal.

- `cfgOFDM`: cfgOFDM is a struct containing the size of the FFT and the length of the cyclic prefix.
- `ofdmSymOffset`: ofdmSymOffset is a scalar containing the OFDM symbol offset.

`Output`
The output arguments of the function are two matrices `data` and `pilot` containing the information of the data and pilot subcarriers respectively.
`Function Overview`
The function performs OFDM demodulation, by removing the cyclic prefix and applying the Fourier transform to the input signal.

3.3.4.2 Equalizing

Equalization is a technique used to reverse the distortion that is incurred when a signal is transmitted through a channel. This is performed by MATLAB's internal function

function [y, CSI] = wlanEqualize(x, chanEst, eqMethod, varargin)
`Input Parameters`

- `x`: x is the input signal and is of size Nsd x Nsym x Nr, where Nsd represents the number of data subcarriers (frequency domain), Nsym represents the number of OFDM symbols (time domain), and Nr represents the number of receive antennas (spatial domain).
- `chanEst`: chanEst is the channel estimate and is of size Nsd x Nsts x Nr, where Nsts represents the number of space-time streams.
- `eqMethod`: eqMethod is the specified equalization method. In this thesis, the equalization method is set to MMSE.
- `varargin` : Additional optional parameters:
  - `noiseVar`: noiseVar is a single or double-precision, real, nonnegative scalar representing the noise variance.

`Output`
The single or double precision output `y` is of size Nsd x Nsym x Nsts and it contains the estimate of the transmitted signal. `y` is complex when either X or CHANEST is complex and is real otherwise. The single or double precision, real output CSI is of size Nsd x Nsts and contains the channel state information. `Function Overview`
For a SISO system, the function performs equalization by dividing the input signal by the norm of the channel and multiplying it by the conjugate of the channel estimate.
If STBC is applied, instead of equalizing, MATLAB performs STBC combining. This is described in Section 4.

3.3.4.3 Constellation Demapping

Constellation demapping is the process of converting the received equalized complex symbols back to interleaved bits or LLR based on the MCS. This function, **wlanConstellationDemap**, supports both hard-decision and soft-decision demodulation. Hard-decision demodulation directly maps the received symbols to the nearest constellation points, resulting in bit values. Soft-decision demodulation, on the other hand, computes LLRs, which are real numbers that provide information about the confidence of each bit's value, enhancing error correction performance. **function y = wlanConstellationDemap(x, noiseVar, numBPSCS, varargin)**
`Input Parameters`

- `x` : The received symbols. It should be a single or double-precision vector, matrix, or multidimensional array containing the received symbols.
- `noiseVar`: A single or double nonnegative scalar representing the noise variance estimate. For 'hard' demapping, this is ignored.
- `numBPSCS`: A scalar specifying the number of coded bits per subcarrier per spatial stream. It can be 1 (BPSK), 2 (QPSK), 4 (16-QAM), 6 (64-QAM) and is equal to $\log_2(M)$, where M is the modulation order.
- `varargin` : Additional optional parameters:

- `DEMAPTYPE`: A character vector or string scalar specifying the demapping type. It can be 'hard' for hard-decision demapping or 'soft' for soft-decision approximate LLR method. The default is 'soft'.
- `PHASE`: A scalar, matrix, or multidimensional array specifying the phase rotation in radians. The phase and mapped symbols must have compatible sizes.

`Output`
The output `y` is a matrix or multidimensional array containing the demapped symbols. It has the same data type as the input `x` and the same size except for the number of rows, which will be equal to the number of rows of `x` multiplied by `numBPSCS`.
`Function Overview`
The function performs constellation demapping of the input symbols `x`. It supports both hard-decision demapping, which maps the received symbols to the nearest constellation points to produce bit values, and soft-decision demapping, which computes LLRs. The noise variance `noiseVar` is used in the LLR computation for 'soft' demapping. LLRs are real numbers that represent the confidence of each bit's value, where a positive LLR indicates that the bit is more likely to be 0, with the magnitude indicating the certainty of this decision.

3.3.4.4 Deinterleaving

Deinterleaving is the process of inversing the three permutations that were performed in the interleaving process. The inverse of the third permutation is

$$j = \left(r + \left((2(i_{\mathrm{SS}}-1)) \bmod 3 + 3\left\lfloor \frac{i_{\mathrm{SS}}-1}{3} \right\rfloor \cdot N_{\mathrm{ROT}} \cdot N_{\mathrm{BPSCS}}(i_{\mathrm{SS}})\right)\right) \bmod N_{\mathrm{CBPSCS}}(i_{\mathrm{SS}}),$$
$$r = 0, 1, ..., N_{\mathrm{CBPSS}}(i_{\mathrm{SS}}) - 1. \tag{3.7}$$

The inverse of the second permutation is

$$i = s(i_{\text{SS}}) \cdot \left\lfloor \frac{j}{s(i_{\text{SS}})} \right\rfloor + \left(j + \left\lfloor N_{\text{COL}} \frac{j}{N_{\text{CBPSS}}(i_{\text{SS}})} \right\rfloor \bmod s(i_{\text{SS}}) \right), \; j = 0, 1, ..., N_{\text{CBPSS}}(i_{\text{SS}}) - 1. \tag{3.8}$$

The inverse of the first permutation is

$$k = N_{\text{COL}} \cdot i - (N_{\text{CBPSS}}(i_{\text{SS}}) - 1) \cdot \left\lfloor \frac{i}{N_{\text{ROW}}} \right\rfloor, \qquad i = 0, 1, ..., N_{\text{CBPSS}}(i_{\text{SS}}) - 1. \tag{3.9}$$

In MATLAB this process is executed by
function y = wlanBCCDeinterleave(x, type, numCBPSSI, varargin)
`Input Parameters`

- `x`: A single or double-precision array of size (Ncbpssi*Nsym)-by-Nss-by-Nseg containing binary convolutional interleaved data.
- `type`: A character vector or string specifying the type of deinterleaving to perform. It must be 'Non-HT' or 'VHT'.
- `numCBPSSI`: A positive integer scalar containing the number of coded bits per OFDM symbol per spatial stream per interleaver block. It is equal to Nsd*Nbpscs for 'Non-HT' TYPE, and equal to Nsd*Nbpscs/Nseg for 'VHT' TYPE, where Nsd is the number of data subcarriers, and Nbpscs is the number of coded bits per subcarrier per spatial stream.
- `varargin` : Additional optional parameters:
  - `CHANBW`: A character vector or string with the channel bandwidth. It must be one of 'CBW1', 'CBW2', 'CBW4', 'CBW8', 'CBW16', 'CBW20', 'CBW40', 'CBW80', or 'CBW160'. CHANBW is not a required parameter for the 'Non-HT' deinterleaver TYPE.

`Output`
The output `y` is an array of size (Ncbpssi*Nsym)-by-Nss-by-Nseg containing binary convolutionally deinterleaved data. Ncbpssi is the number of coded bits per OFDM

symbol per spatial stream per interleaver block, Nsym is the number of OFDM symbols, Nss is the number of spatial streams, and Nseg is the number of segments.
`Function Overview`
The function outputs binary convolutionally deinterleaved data for the specified interleaver type.

3.3.4.5 Decoding

To retrieve the decoded symbols encoded using a binary convolutional code, the function **wlanBCCDecode** is utilized which supports both hard-decision and soft-decision Viterbi algorithms [10]. Hard-decision decoding directly maps the received symbols to bit values, whereas soft-decision decoding computes LLRs to provide more nuanced information about each bit's confidence.
function y = wlanBCCDecode(x, rate, varargin)
`Input Parameters`

- `x` : The input symbols to decode. It should be a matrix of type single, double, or int8, containing encoded symbols. For 'soft' decoding, `x` should be a real matrix of log-likelihood ratios, where positive values represent logical 0 and negative values represent logical 1.
- `rate`: A scalar, character vector, or string scalar specifying the coding rate. Possible values are 1/2, 2/3, 3/4, or 5/6, or their corresponding string representations '1/2', '2/3', '3/4', or '5/6'.
- `varargin` : Additional optional parameters:
  - `DECTYPE`: A character vector or string scalar specifying the decoding type. It can be 'hard' for hard-decision decoding or 'soft' for soft-decision decoding without quantization. The default is 'soft'.
  - `TDEPTH`: A positive integer scalar specifying the traceback depth of the Viterbi decoding algorithm.

`Output`
The output `y` is a binary int8 matrix containing the decoded bits. The number of rows of `y` is equal to the number of rows of input `x` multiplied by the coding rate,

rounded to the next integer. The number of columns of `y` is equal to the number of columns of `x`.

`Function Overview`

The function performs binary convolutional decoding of the input symbols `x`. It supports both hard-decision decoding, which maps the received symbols to bit values, and soft-decision decoding, which computes LLRs. The optional traceback depth parameter `TDEPTH` controls the depth of the Viterbi algorithm's traceback, affecting decoding performance and complexity.

3.3.4.6 Descrambling

Descrambling is the process of reversing the scrambling applied to the transmitted data, restoring the original binary sequence. The `wlanScramble` function is used for both scrambling and descrambling, as it implements a frame-synchronous scrambler described in the IEEE 802.11n standard. The initial state must match the state used during scrambling, otherwise, if the initial state `scramInit` is all zeros, descrambling is bypassed, and the data is returned without modification.

Chapter 4

Space-Time Block Coding

Space-time block coding(STBC) is a technique used to improve the performance of a fading channel. Fading can be moderated by diversity, which means transmitting the information several times through independent channels and using the different received versions of the data to enhance the reliability of the transmission. Space-time block codes are a form of spatial diversity and can be exploited when the channel is unknown to the receiver.

4.1 Theoretical description of STBC

The following sections discuss a theoretical description of STBC for different systems with two transmit antennas.

4.1.1 2x1 MISO system

Let us assume a MISO system with 2 transmit antennas and 1 receive antenna, as depicted in figure 4.1.
Alamouti's scheme [11] is the simplest STBC and it was designed for two transmit antennas. Let us consider 2 symbols s_1 and s_2. At the first time slot, the first transmitting antenna (Tx_1) transmits s_1 and the second transmitting antenna (Tx_2) transmits s_2. At the second time slot, Tx_1 transmits $-s_2*$ and Tx_2 transmits s_1*, where * represents complex conjugate.

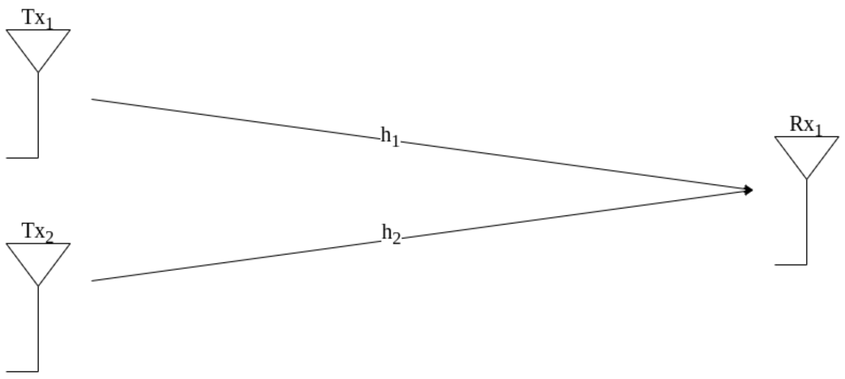

Figure 4.1: MISO 2x1 system.

So the received signal at time slot 1 is

$$y^{(1)} = \begin{bmatrix} h_1 & h_2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + w_1, \tag{4.1}$$

and the received signal at time slot 2 is

$$y^{(2)} = \begin{bmatrix} h_1 & h_2 \end{bmatrix} \begin{bmatrix} -s_2^* \\ s_1^* \end{bmatrix} + w_2. \tag{4.2}$$

The received signal at time slot 1 can be written as

$$y^{(1)} = h_1 s_1 + h_2 s_2 + w_1, \tag{4.3}$$

and the received signal at time slot 2 can be written as

$$y^{(2)} = h_1(-s_2^*) + h_2 s_1^* + w_2. \tag{4.4}$$

The complex conjugate of the received signal at time slot 2 is

$$y^{(2)*} = h_1^*(-s_2) + h_2^* s_1 + w_2^* = h_2^* s_1 + (-h_1^*) s_2 + w_2^*. \tag{4.5}$$

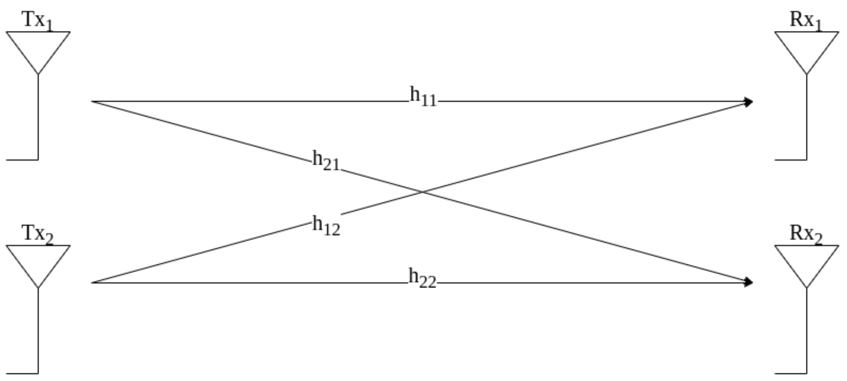

Figure 4.2: MIMO 2x2 system.

If we combine equations 4.3 and 4.5 we can obtain the following equation

$$\begin{bmatrix} y^{(1)} \\ y^{(2)*} \end{bmatrix} = \begin{bmatrix} h_1 & h_2 \\ h_2^* & -h_1^* \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2^* \end{bmatrix}. \tag{4.6}$$

To compute the estimate of the transmitted signal, the equivalent channel matrix has to be defined as

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 \\ h_2^* & -h_1^* \end{bmatrix}. \tag{4.7}$$

Hence, the estimate of the transmitted signal is

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H \begin{bmatrix} y^{(1)} \\ y^{(2)*} \end{bmatrix} = \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H\mathbf{H} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H \begin{bmatrix} w_1 \\ w_2^* \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} \tilde{w_1} \\ \tilde{w_2} \end{bmatrix}. \tag{4.8}$$

4.1.2 2x2 MIMO system

Let us assume a MIMO system with 2 transmit antennas and 2 receive antennas, as shown in figure 4.2.

In the case of two transmit and two receive antennas, the received signals at each of the receive antennas are

$$y_1^{(t)} = h_{11}x_1^{(t)} + h_{12}x_2^{(t)} + w_1^{(t)}, \tag{4.9}$$

$$y_2^{(t)} = h_{21}x_1^{(t)} + h_{22}x_2^{(t)} + w_2^{(t)}. \tag{4.10}$$

At the first time slot, the received signal at each of the antennas is

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} w_1^{(1)} \\ w_2^{(1)} \end{bmatrix}. \tag{4.11}$$

At the second time slot, the received signal at each of the antennas is

$$\begin{bmatrix} y_1^{(2)} \\ y_2^{(2)} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} -s_2^* \\ s_1^* \end{bmatrix} + \begin{bmatrix} w_1^{(2)} \\ w_2^{(2)} \end{bmatrix}. \tag{4.12}$$

So the combined equations are

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ y_1^{(2)*} \\ y_2^{(2)*} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ h_{12}^* & -h_{11}^* \\ h_{22}^* & -h_{21}^* \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} w_1^{(1)} \\ w_2^{(1)} \\ w_1^{(2)*} \\ w_2^{(2)*} \end{bmatrix}. \tag{4.13}$$

To compute the estimate of the transmitted signal, the equivalent channel matrix has to be defined as

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ h_{12}^* & -h_{11}^* \\ h_{22}^* & -h_{21}^* \end{bmatrix}. \tag{4.14}$$

Hence, the estimate of the transmitted signal is

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H \begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ y_1^{(2)*} \\ y_2^{(2)*} \end{bmatrix} = \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H\mathbf{H} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H \begin{bmatrix} w_1^{(1)} \\ w_2^{(1)} \\ w_1^{(2)*} \\ w_2^{(2)*} \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} \tilde{w_1} \\ \tilde{w_2} \end{bmatrix}. \tag{4.15}$$

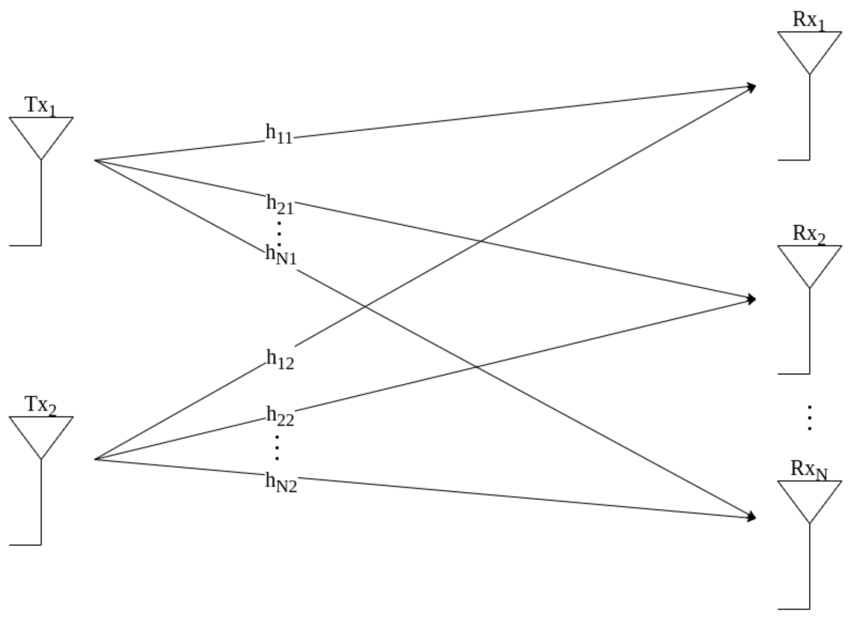

Figure 4.3: MIMO 2xN system.

4.1.3 2xN MIMO system

Let us assume a MIMO system with 2 transmit antennas and N receive antennas, as shown in figure 4.3.

In the case of two transmit and N receive antennas, the received signals at each of the receive antennas are

$$y_1^{(t)} = h_{11}x_1^{(t)} + h_{12}x_2^{(t)} + w_1^{(t)}, \tag{4.16}$$

$$y_2^{(t)} = h_{21}x_1^{(t)} + h_{22}x_2^{(t)} + w_2^{(t)}, \tag{4.17}$$

$$\vdots$$

$$y_{\mathrm{N}}^{(t)} = h_{\mathrm{N}1}x_1^{(t)} + h_{\mathrm{N}2}x_2^{(t)} + w_{\mathrm{N}}^{(t)}. \tag{4.18}$$

At the first time slot, the received signal at each of the antennas is

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_{\mathrm{N}}^{(1)} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ \vdots & \vdots \\ h_{\mathrm{N}1} & h_{\mathrm{N}2} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} w_1^{(1)} \\ w_2^{(1)} \\ \vdots \\ w_{\mathrm{N}}^{(1)} \end{bmatrix}. \tag{4.19}$$

At the second time slot, the received signal at each of the antennas is

$$\begin{bmatrix} y_1^{(2)} \\ y_2^{(2)} \\ \vdots \\ y_{\mathrm{N}}^{(2)} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ \vdots & \vdots \\ h_{\mathrm{N}1} & h_{\mathrm{N}2} \end{bmatrix} \begin{bmatrix} -s_2^* \\ s_1^* \end{bmatrix} + \begin{bmatrix} w_1^{(2)} \\ w_2^{(2)} \\ \vdots \\ w_{\mathrm{N}}^{(2)} \end{bmatrix}. \tag{4.20}$$

So the combined equations are

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_{\mathrm{N}}^{(1)} \\ y_1^{(2)*} \\ y_2^{(2)*} \\ \vdots \\ y_{\mathrm{N}}^{(2)*} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ \vdots & \vdots \\ h_{\mathrm{N}1} & h_{\mathrm{N}2} \\ h_{12}^* & -h_{11}^* \\ h_{22}^* & -h_{21}^* \\ \vdots & \vdots \\ h_{\mathrm{N}2}^* & -h_{\mathrm{N}1}^* \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} w_1^{(1)} \\ w_2^{(1)} \\ \vdots \\ w_{\mathrm{N}}^{(1)} \\ w_1^{(2)*} \\ w_2^{(2)*} \\ \vdots \\ w_{\mathrm{N}}^{(2)*} \end{bmatrix}. \tag{4.21}$$

To compute the estimate of the transmitted signal, the equivalent channel matrix has to be defined as

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ \vdots & \vdots \\ h_{\mathrm{N}1} & h_{\mathrm{N}2} \\ h_{12}^* & -h_{11}^* \\ h_{22}^* & -h_{21}^* \\ \vdots & \vdots \\ h_{\mathrm{N}2}^* & -h_{\mathrm{N}1}^* \end{bmatrix}. \tag{4.22}$$

Hence, the estimate of the transmitted signal is

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H \begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_\mathrm{N}^{(1)} \\ y_1^{(2)*} \\ y_2^{(2)*} \\ \vdots \\ y_\mathrm{N}^{(2)*} \end{bmatrix} = \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H\mathbf{H} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \frac{1}{\|\mathbf{H}\|_2}\mathbf{H}^H \begin{bmatrix} w_1^{(1)} \\ w_2^{(1)} \\ \vdots \\ w_\mathrm{N}^{(1)} \\ w_1^{(2)*} \\ w_2^{(2)*} \\ \vdots \\ w_\mathrm{N}^{(2)*} \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} \tilde{w_1} \\ \tilde{w_2} \\ \vdots \\ \tilde{w_\mathrm{N}} \end{bmatrix}. \tag{4.23}$$

4.2 Implementation of STBC in 802.11n

In 802.11n STBC is an optional feature, that is applied when the number of space-time streams N_{STS} is greater than the number of spatial streams N_{SS}. N_{SS} spatial streams are mapped to N_{STS} space-time streams, which are then mapped to N_{Tx} transmit chains. A space-time stream represents the time slot of the transmission and a spatial stream represents the space of the transmission, and this is why N_{SS} is limited by N_{Tx}. Additionally, the number of space-time streams has to be greater than or equal to the number of spatial streams and also less than or equal to twice the number of spatial streams.

4.2.1 MATLAB's functions for STBC Encoding and Combining

STBC Encoding is performed at the transmitter side after constellation mapping by MATLAB's internal function **wlanSTBCEncode**.
function y = wlanSTBCEncode(x, numSTS)
`Input Parameters`

- `x` : x is of size N_{SD} x N_{SYM} x N_{SS}. N_{SD} is the number of data subcarriers and N_{SYM} is the number of OFDM symbols. It contains the data after constellation mapping.
- `numSTS` : Number of space-time streams.

`Output`
The output is **y** and is of size N_{SD} x N_{SYM} x N_{STS} and has the same data type and complexity as the input **x**.
STBC combining is performed at the receiver side before constellation demapping by MATLAB's internal function **wlanSTBCCombine**.
function [y, CSI] = wlanSTBCCombine(x, chanEst, numSS, eqMethod)
`Input Parameters`

- `x` : A complex 3D array of size N_{SD} x N_{SYM} x N_{R} representing the received signal.
- `chanEst` : A complex 3D array of size N_{SD} x N_{STS} x N_{R}, representing the estimate of the channel.
- `numSS` : Number of spatial streams.
- `eqMethod` : Equalization method specified as 'MMSE' for minimum mean square error or 'ZF' for zero forcing. In this thesis, 'MMSE' equalization is used.

`Output`
The outputs are **y**, a complex array of size N_{SD} x N_{SYM} x N_{SS} containing the

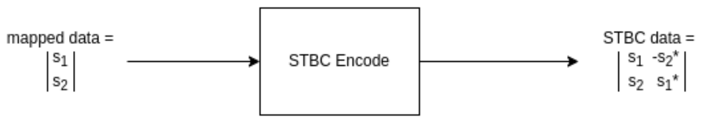

Figure 4.4: STBC encoding procedure.

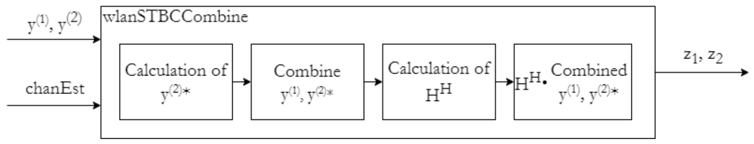

Figure 4.5: STBC combining procedure.

estimate of the transmitted signal and **CSI**, which is a real matrix of size Nsd-by-Nss representing the soft channel state information.
To understand their functionality, let us consider the Alamouti scheme described in Section 4.1.1. Figures 4.4 and 4.5 showcase the input and the output arguments of **wlanSTBCEncode** and **wlanSTBCCombine** respectively.

4.2.2 STBC Encoding in 802.11n

STBC encoding for different cases of N_{STS} and N_{SS} is described in Section 19.3.11.9.2 of [6].

- If there is one spatial stream and two space-time streams, the input and the output of the STBC Encoder is

$$\text{Input} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \ \text{Output} = \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix}. \tag{4.24}$$

- If there are two spatial streams and three space-time streams, the input and the output of the STBC Encoder is

$$\text{Input} = \begin{bmatrix} s_1 & s_3 \\ s_2 & s_4 \end{bmatrix}, \ \text{Output} = \begin{bmatrix} s_1 & -s_2^* & s_3 \\ s_2 & s_1^* & s_4 \end{bmatrix}. \tag{4.25}$$

- If there are two spatial streams and four space-time streams, the input and the output of the STBC Encoder is

$$\text{Input} = \begin{bmatrix} s_1 & s_3 \\ s_2 & s_4 \end{bmatrix}, \text{ Output} = \begin{bmatrix} s_1 & -s_2^* & s_3 & -s_4^* \\ s_2 & s_1^* & s_4 & s_3^* \end{bmatrix}. \tag{4.26}$$

- If there are three spatial streams and four space-time streams, the input and the output of the STBC Encoder is

$$\text{Input} = \begin{bmatrix} s_1 & s_3 & s_5 \\ s_2 & s_4 & s_6 \end{bmatrix}, \text{ Output} = \begin{bmatrix} s_1 & -s_2^* & s_3 & s_5 \\ s_2 & s_1^* & s_4 & s_6 \end{bmatrix}. \tag{4.27}$$

It should be noted that the columns of the STBC Encoder's output represent the space-time streams.

4.2.3 Spatial Mapping

After STBC is performed, the spatial mapper expands the number of space-time streams N_{STS} into the number of transmit antennas N_{Tx}. Spatial mapping can be performed through a Direct, a Hadamard, a Fourier, or a Custom Mapping Matrix. In this thesis, the Fourier Mapping Matrix is put into use. The mapping matrix is of size N_{Tx}*N_{STS}+N_{ESS}, where N_{ESS} stands for the number of extension spatial streams. In our case, this number is equal to 0. The element in the n^{th} row and m^{th} column of the Fourier matrix is calculated as follows

$$q_{\text{n,m}} = \frac{1}{\sqrt{N_{\text{Tx}}}} e^{\frac{-j2\pi(m-1)(n-1)}{N_{\text{Tx}}}}, n = 1, \ldots, N_{\text{Tx}}, m = 1, \ldots, N_{\text{STS}} + N_{\text{ESS}}. \tag{4.28}$$

Some examples of the Fourier mapping matrix are

- For the case of two transmit antennas and two space-time streams:

$$\mathbf{Q} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{4.29}$$

- For the case of three transmit antennas and two space-time streams:

$$\mathbf{Q} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 \\ 1 & e^{\frac{-j2\pi}{3}} \\ 1 & e^{\frac{-j4\pi}{3}} \end{bmatrix}. \tag{4.30}$$

- For the case of four transmit antennas and two space-time streams:

$$\mathbf{Q} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & e^{\frac{-j\pi}{2}} \\ 1 & e^{-j\pi} \\ 1 & e^{\frac{-j3\pi}{2}} \end{bmatrix}. \tag{4.31}$$

4.2.4 STBC and Spatial Mapping in MISO systems

Let us assume a N_{Tx}x1 MISO system with N_{SS} spatial streams and N_{STS} space-time streams. The mapping matrix $\mathbf{Q}$ for this system is of size N_{Tx}xN_{STS}. The symbols matrix $\mathbf{S}$ is of size N_{STS}xN_{SYM} where N_{SYM} is the number of symbols. The transmitted signal $\mathbf{X}$ is given by Equation 4.32 and is of size N_{Tx}xN_{SYM}.

$$\mathbf{X} = \mathbf{QS} \tag{4.32}$$

The following sections showcase the STBC encoding and spatial mapping process for different cases of 2x1, 3x1, and 4x1 MISO systems.

4.2.4.1 1 spatial stream and 1 space-time stream

The number of spatial streams is equal to the number of space-time streams, so STBC is not performed. The symbol matrix is $S = \begin{bmatrix} s \end{bmatrix}$ The transmitted signal is

$$\mathbf{X} = \mathbf{QS} = \mathbf{Q}s, \tag{4.33}$$

and the signal at the receive antenna is given by

$$y = \mathbf{HX} + w, \tag{4.34}$$

where $\mathbf{H}$ is the channel matrix, that is of size 1xN_{Tx}. Since $\mathbf{X} = \mathbf{QS}$ and $\mathbf{Q}$ is of size N_{Tx}x1, the equation of the received signal can be written as

$$y = cs + w, \tag{4.35}$$

where c is a scalar containing the result of $\mathbf{HQ}$. The estimate of the transmitted signal is

$$z = \frac{c^*}{\|c\|_2} y = s + \frac{c^*}{\|c\|_2} w = s + \tilde{w}. \tag{4.36}$$

4.2.4.2 1 spatial stream and 2 space-time streams

The number of space-time streams is greater than the number of spatial streams, so STBC is performed. The symbol matrix is

$$\mathbf{S} = \begin{bmatrix} s_1 & s_2 \\ -s_2^* & s_1^* \end{bmatrix}, \tag{4.37}$$

and the received signal is

$$\mathbf{y} = \mathbf{HX} + \mathbf{W}, \tag{4.38}$$

where $\mathbf{H}$ is the channel matrix, which is of size 1xN_{Tx}. Since $\mathbf{X} = \mathbf{QS}$ and $\mathbf{Q}$ is of size N_{Tx}x2, the equation of the received signal can be written as

$$\mathbf{y} = \mathbf{CS} + \mathbf{W}, \tag{4.39}$$

where $\mathbf{C} = \mathbf{HQ} = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$.
So the received signal at time slot 1 is

$$y^{(1)} = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + w_1. \tag{4.40}$$

The received signal at time slot 2 is

$$y^{(2)} = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} -s_2^* \\ s_1^* \end{bmatrix} + w_2. \tag{4.41}$$

The received signal at time slot 1 can be written as

$$y^{(1)} = c_1 s_1 + c_2 s_2 + w_1, \tag{4.42}$$

and the received signal at time slot 2 can be written as

$$y^{(2)} = c_1(-s_2^*) + c_2 s_1^* + w_2. \tag{4.43}$$

The complex conjugate of the received signal at time slot 2 is

$$y^{(2)*} = c_1^*(-s_2) + c_2^* s_1 + w_2^* = c_2^* s_1 + (-c_1^*) s_2 + w_2^*. \tag{4.44}$$

If we combine and $y^{(1)}$ and $y^{(2)*}$ we can obtain the following equation

$$\begin{bmatrix} y^{(1)} \\ y^{(2)*} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 \\ c_2^* & -c_1^* \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2^* \end{bmatrix}. \tag{4.45}$$

To compute the estimate of the transmitted signal, the equivalent channel matrix has to be defined as

$$\mathbf{C} = \begin{bmatrix} c_1 & c_2 \\ c_2^* & -c_1^* \end{bmatrix}, \tag{4.46}$$

and the estimate of the transmitted signal is

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \frac{1}{\|\mathbf{C}\|_2}\mathbf{C}^H \begin{bmatrix} y^{(1)} \\ y^{(2)*} \end{bmatrix} = \frac{1}{\|\mathbf{C}\|_2}\mathbf{C}^H\mathbf{C} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \frac{1}{\|\mathbf{C}\|_2}\mathbf{C}^H \begin{bmatrix} w_1 \\ w_2^* \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} \tilde{w_1} \\ \tilde{w_2} \end{bmatrix}. \tag{4.47}$$

4.2.4.3 2 spatial streams and 2 space-time streams

The number of space-time streams is equal to the number of spatial streams, so STBC is not performed. The symbol matrix is

$$\mathbf{S} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}. \tag{4.48}$$

The columns of the symbol matrix represent the space-time streams. The transmitted signal is

$$\mathbf{X} = \mathbf{QS}, \tag{4.49}$$

where $\mathbf{Q}$ is of size N_{Tx}x2. The received signal is

$$y = \mathbf{HX} + w = \mathbf{HQS} + w = \mathbf{CS} + w, \tag{4.50}$$

where $\mathbf{H}$ is of size 1xN_{Tx} and $\mathbf{C}$ is of size 1xN_{STS}=1x2.
To estimate the transmitted signal, MMSE equalization is performed. The goal is to find a coefficient $\mathbf{W}$ that minimizes the criterion

$$\mathbf{W}_{\mathrm{MMSE}} = argmin_W E[\|\mathbf{W}y - \mathbf{S}\|^2]. \tag{4.51}$$

By solving this equation we get

$$\mathbf{W} = \mathbf{C}^H(\mathbf{CC}^H + N_0\mathbf{I})^{-1}, \tag{4.52}$$

where N_0 is the noise variance and $\mathbf{I}$ is the identity matrix.
The estimate of the transmitted signal is

$$\mathbf{S} = \mathbf{W}y. \tag{4.53}$$

4.2.4.4 2 spatial streams and 3 space-time streams

This case only applies to systems with 3 or 4 transmit antennas, since the number of space-time streams cannot be greater than the number of transmit antennas. The number of space-time streams is greater than the number of spatial streams, so STBC is performed. The symbols matrix is

$$\mathbf{S} = \begin{bmatrix} s_1 & s_2 \\ -s_2^* & s_1^* \\ s_3 & s_4 \end{bmatrix}, \tag{4.54}$$

and the transmitted signal is $\mathbf{X} = \mathbf{Q}\ \mathbf{S}$. The received signal is

$$\mathbf{y} = \mathbf{HX} + \mathbf{W}. \tag{4.55}$$

4.2.4.5 2 spatial streams and 4 space-time streams

This case only applies to systems with 4 transmit antennas, since the number of space-time streams cannot be greater than the number of transmit antennas. The number of space-time streams is greater than the number of spatial streams, so STBC is performed. The symbols matrix is

$$\mathbf{S} = \begin{bmatrix} s_1 & s_2 \\ -s_2^* & s_1^* \\ s_3 & s_4 \\ -s_4^* & s_3^* \end{bmatrix}, \tag{4.56}$$

and the transmitted signal is $\mathbf{X} = \mathbf{Q}\ \mathbf{S}$. The received signal is

$$\mathbf{y} = \mathbf{HX} + \mathbf{W}. \tag{4.57}$$

4.2.4.6 3 spatial streams and 3 space-time streams

This case only applies to systems with 3 or 4 transmit antennas, since the number of space-time streams cannot be greater than the number of transmit antennas. The process is similar to Section 4.2.4.3 with the difference that in this case, the symbol matrix is

$$\mathbf{S} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}. \tag{4.58}$$

The channel matrix $\mathbf{H}$ for this case is of size 1xN_{Tx}, the mapping matrix $\mathbf{Q}$ is of size N_{Tx}x3 and the received signal is

$$y = \mathbf{HX} + w = \mathbf{HQS} + w = \mathbf{CS} + w, \tag{4.59}$$

where **C** is of size 1xN_{Tx}. To estimate the transmitted signal, MMSE equalization is performed, as described above.

4.2.4.7 3 spatial streams and 4 space-time streams

This case only applies to systems with 4 transmit antennas, since the number of space-time streams cannot be greater than the number of transmit antennas. The number of space-time streams is greater than the number of spatial streams, so STBC is performed. The symbols matrix is

$$\mathbf{S} = \begin{bmatrix} s_1 & s_2 \\ -s_2^* & s_1^* \\ s_3 & s_4 \\ s_5 & s_6 \end{bmatrix}, \tag{4.60}$$

and the transmitted signal is $\mathbf{X} = \mathbf{Q}\ \mathbf{S}$. The received signal is

$$\mathbf{y} = \mathbf{H}\mathbf{X} + \mathbf{W}. \tag{4.61}$$

Chapter 5

Exponential Effective SNR Mapping

5.1 Introduction

As stated before, modern-day wireless communication systems are of high complexity, in order to meet the needs of the ever-increasing users and provide high data rates. A technique used to improve the data rate is adaptive modulation and coding (AMC). The objective of this technique is to allow the transmitter to choose an appropriate MCS according to the channel characteristics. In the IEEE 802.11n Standard, link adaptation algorithms can be utilized to maintain reliable communication since the channel has frequency-selective qualities due to OFDM. This means that the channel gains of each subcarrier can vary a lot. However, a suitable MCS should be selected rapidly. Thus, mapping the per subcarrier SNRs to a single scalar value, which will in turn be mapped to a BER value, is critical. The importance of an effective SNR mapping (ESM) function in OFDM-based systems is highlighted in [12]. The purpose of the ESM is to map the instantaneous channel state into a single scalar value, called the effective SNR. This value is then used to find an estimate of the block-error probability (BLEP) for this specific channel state. For instance, the instantaneous SNR of each subcarrier (γ_{k}) can be mapped into a single equivalent SNR, the effective SNR (γ_{eff}), which will be used to find an estimate of the BLEP from an AWGN link-level simulation. An ESM has to satisfy the following equivalence

$$\mathrm{BLEP}(\{\gamma_\mathrm{k}\}) \approx \mathrm{BLEP}_\mathrm{AWGN}(\gamma_\mathrm{eff}), \tag{5.1}$$

where

- $\mathrm{BLEP}(\{\gamma_\mathrm{k}\})$ is the actual block-error probability for the instantaneous channel state (γ_k),
- $\mathrm{BLEP}_\mathrm{AWGN}(\gamma_\mathrm{eff})$ is the AWGN block-error probability.

In [13] the concept of exponential effective SNR mapping (EESM) is introduced and the following mapping function for the effective SNR γ_eff is proposed

$$\gamma_\mathrm{eff} = -\beta \log\Big(\frac{1}{N_\mathrm{SD}}\sum_{k=1}^{N_\mathrm{SD}} e^{-\frac{\gamma_\mathrm{k}}{\beta}}\Big), \tag{5.2}$$

- N_SD is the number of data subcarriers, in our case 52,
- γ_k is the per subcarrier SNR, and
- β is a parameter that is empirically calibrated and adjusted to each MCS separately.

The goal of EESM is to find a suitable γ_eff that satisfies the criterion

$$\mathrm{BER}_\mathrm{AWGN}(\gamma_\mathrm{eff}) \approx \mathrm{BER}(\gamma_1, \gamma_2, \dots, \gamma_{N_\mathrm{SD}}). \tag{5.3}$$

An optimal value for β of Equation 5.2 can be computed in various ways, however, two suggested optimization criteria are the following. The first criterion is

$$\beta_\mathrm{opt} = \arg\min_{\beta}\Big\{\frac{1}{N_\mathrm{C}}\sum_{i=1}^{N_\mathrm{C}} \Delta e_\mathrm{i}(\beta)\Big\}, \tag{5.4}$$

where

$$\Delta e_\mathrm{i}(\beta) = \frac{1}{L}\sum_{k=1}^{L}(\log_{10}(\mathrm{BER}_\mathrm{k,\,i}^\mathrm{meas}) - \log_{10}(\mathrm{BER}_\mathrm{k,\,i}^\mathrm{EESM}(\beta)))^2,\ \forall\ i = 1, 2, ..., N_\mathrm{C} \tag{5.5}$$

- N_C is the number of different channel realizations,
- L is the number of different instances of the noise variance for each realization,
- $\text{BER}^\text{meas}_\text{k, i}$ is the measured BER for the i^{th} channel realization and the k^{th} noise variance combination,
- $\text{BER}^\text{EESM}_\text{k, i}(\beta)$ is the estimated BER for the i^{th} channel realization and the k^{th} noise variance combination and for a given β.

The second criterion is

$$\beta_\text{opt} = \arg\min_{\beta}\left\{\frac{1}{N_\text{C}}\sum_{i=1}^{N_\text{C}}(\gamma_\text{AWGN}(i) - \gamma_\text{eff}(i,\beta))^2\right\}, \tag{5.6}$$

where

- N_C is the number of different channel realizations,
- $\gamma_\text{AWGN}(i)$ is the SNR value that is required to meet the target BER in an AWGN channel,
- $\gamma_\text{eff}(i,\beta)$ is the effective SNR value, that is calculated for the i^{th} channel realization and for a given β.

This means that the goal is to find an equivalent SNR, γ_eff, that is approximately equal to an SNR value, γ_AWGN, which in an AWGN channel produces a BER similar to the instantaneous BER that was produced in the frequency selective channel. This concept is described in Figure 5.1.

5.2 Derivation of EESM

As described in [13], the Union-Chernoff bound of error probabilities is the basis of EESM. Firstly, let us denote the symbol error rate approximation over an AWGN channel. Let us assume $d_\text{min} = 2A$ the minimum distance of the constellation. The

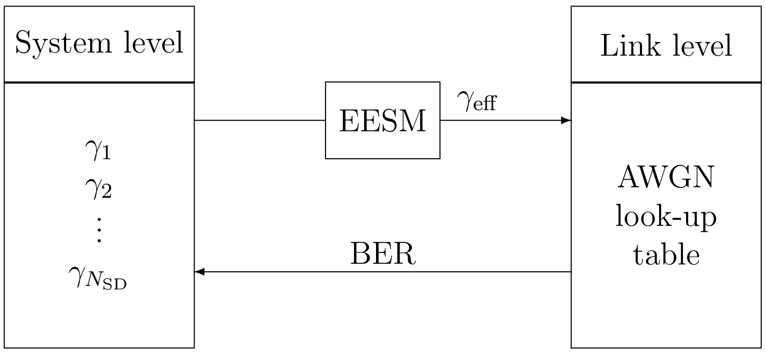

Figure 5.1: EESM block diagram.

symbol error rate approximation as stated in [14] is expressed as

$$\begin{aligned} P_{\mathrm{e}}(d_{\min}, \mathrm{SNR}) &\approx \bar{N}_{\mathrm{dmin}} Q\left(\sqrt{\frac{d_{\min}^2}{2N_0}}\right) \\ &= \bar{N}_{\mathrm{dmin}} Q\left(\sqrt{\frac{2A^2}{N_0}}\right) \\ &= \bar{N}_{\mathrm{dmin}} Q\left(\sqrt{\frac{2\mathrm{SNR}}{K}}\right), \end{aligned} \tag{5.7}$$

assuming that

- $\mathrm{SNR} = \frac{KA^2}{N_0}$,
- $\bar{N}_{\mathrm{dmin}}$ = average number of nearest neighbors for a signal point in the constellation,
- Q is the Q-function, $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-u^2}{2}} \mathrm{d}u$.

The bit error probability can be derived by dividing the symbol error rate approximation by the bits-to-symbol ratio of the MCS

$$
\begin{aligned}
P_{\mathrm{b}}(\mathrm{SNR}) &\approx \frac{P_{\mathrm{e}}(d_{\min}, \mathrm{SNR})}{\log_2(M)} \\
&= \frac{1}{\log_2(M)} N Q\left(\sqrt{\frac{2\mathrm{SNR}}{K}}\right) \\
&= \alpha Q\left(\sqrt{\frac{\mathrm{SNR}}{b}}\right),
\end{aligned}
\tag{5.8}
$$

where a and b are constellation scheme-dependent parameters. The Chernoff bound for the Q-function is

$$
Q(x) \leq e^{\frac{-x^2}{2}}, \; x > 0 \tag{5.9}
$$

and, by making use of that, an upper bound for the bit error probability can be defined as

$$
P_{\mathrm{b}}(\mathrm{SNR}) \leq \alpha e^{-\frac{\mathrm{SNR}}{2b}}. \tag{5.10}
$$

In a MISO system with one spatial stream, N_{SD} data subcarriers, and STBC, a vector γ containing all the per subcarrier SNRs can be defined as

$$
\gamma = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{N_{\mathrm{SD}}}. \end{bmatrix} \tag{5.11}
$$

As stated in [15] and [16], the per subcarrier SNR can be calculated by dividing the power of the received signal by the estimated noise variance as

$$
\gamma_{\mathrm{k}} = \frac{\|h_{\mathrm{k}}\|_2^2 \sigma_{\mathrm{x}}^2}{N_{\mathrm{Tx}} \hat{N}_0}. \tag{5.12}
$$

It should be mentioned that, in an OFDM system, three different SNR values can be defined, the nominal SNR, the post-FFT SNR, and the post-processing SNR. The nominal SNR is the SNR at the input of the receiver, the post-FFT SNR is the SNR after the OFDM modulation and the post-processing SNR is the SNR after

channel equalization. When calculating the post-processing SNR a smaller noise variance ($\hat{N}_0$) is used, than when calculating the nominal SNR. That is because when computing the post-processing SNR, only the 52 data subcarriers are taken into account. Thus, the power of the noise decreases, while the power of the signal remains the same. This new noise variance $\hat{N}_0$ is the one used for the calculation of the per-subcarrier SNRs. More details regarding these SNRs and how they are derived can be found in Chapter 4 of [8].
Since we consider each subcarrier to correspond to an equivalent Gaussian 1-state channel after channel equalization is performed, the bit error probability for N_{SD} independent Gaussian channels is

$$\begin{aligned}\mathrm{BER}(\gamma) &= \sum_{k=1}^{N_{\mathrm{SD}}} P_{\mathrm{b}}(\gamma_{\mathrm{k}}) \\ &\leq \sum_{k=1}^{N_{\mathrm{SD}}} \alpha e^{-\frac{\gamma_{\mathrm{k}}}{2b}}.\end{aligned} \tag{5.13}$$

As shown in 5.3, the objective is to find an equivalent SNR value γ_{eff} of an equivalent 1-state channel, that is approximately equal to an SNR value γ_{AWGN}, that in an AWGN channel would lead to the bit error probability of the multistate channel. A suitable γ_{eff} is found by matching the respective Chernoff-bounded bit-error probabilities

$$\begin{aligned}\mathrm{BER}(\gamma_{\mathrm{eff}}) &\approx \frac{1}{N_{\mathrm{SD}}} \sum_{k=1}^{N_{\mathrm{SD}}} \alpha e^{-\frac{\gamma_{\mathrm{k}}}{2b}} \Leftrightarrow \\ \alpha e^{-\frac{\gamma_{\mathrm{eff}}}{2b}} &= \frac{1}{N_{\mathrm{SD}}} \sum_{k=1}^{N_{\mathrm{SD}}} \alpha e^{-\frac{\gamma_{\mathrm{k}}}{2b}} \Leftrightarrow \\ -\frac{1}{2b}\gamma_{\mathrm{eff}} &= \log\left(\frac{1}{N_{\mathrm{SD}}} \sum_{k=1}^{N_{\mathrm{SD}}} e^{-\frac{\gamma_{\mathrm{k}}}{2b}}\right) \Leftrightarrow \\ \gamma_{\mathrm{eff}} &= -2b\log\left(\frac{1}{N_{\mathrm{SD}}} \sum_{k=1}^{N_{\mathrm{SD}}} e^{-\frac{\gamma_{\mathrm{k}}}{2b}}\right) \Leftrightarrow \\ \gamma_{\mathrm{eff}} &= -\beta\log\left(\frac{1}{N_{\mathrm{SD}}} \sum_{k=1}^{N_{\mathrm{SD}}} e^{-\frac{\gamma_{\mathrm{k}}}{\beta}}\right)\end{aligned} \tag{5.14}$$

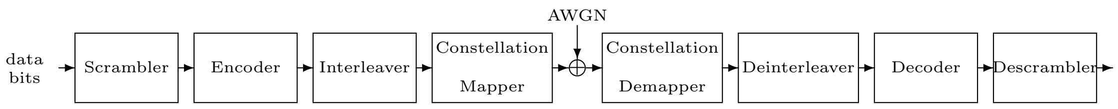

Figure 5.2: Block diagram for AWGN link simulation.

As stated in the previous section, β is a parameter that is different for each MCS and coding rate. We select a β, such that the estimated BER that is produced by EESM is as close as possible to the actual measured BER.

5.3 Implementation of EESM

The simulation environment for this thesis is MATLAB R2023b and WLAN Toolbox is put into use. Firstly the AWGN BER vs SNR curve for each MCS is obtained by a link simulation. This does not require the modeling of a channel and the process is described in Figure 5.2.
For the needed end-to-end link-level simulations for EESM, a 2x1 MISO system with one spatial stream and STBC was implemented for each MCS separately. For each SNR point, at least 200 realizations have to be simulated. The first 100 realizations over all SNR points will be used to find the optimal β parameter and the other 100 realizations will be used to validate the performance of EESM. All per subcarrier SNRs must be saved.
For the optimal β value, the criterion 5.4 was used at first. However, it was concluded that this criterion performs better if the BER values are not converted to dB. Thus, the optimization criterion used in this thesis is

$$\beta_{\text{opt}} = \arg\min_{\beta} \text{MSE}(\beta) = \arg\min_{\beta} \left\{ \frac{1}{N_{\text{points}}} \sum_{i=1}^{N_{\text{points}}} \Delta e_{\text{i}}(\beta) \right\}, \tag{5.15}$$

where

$$\Delta e_{\text{i}}(\beta) = \frac{1}{L} \sum_{k=1}^{L} (\text{BER}_{\text{k, i}}^{\text{meas}} - \text{BER}_{\text{k, i}}^{\text{EESM}}(\beta))^2, \ \forall\ i = 1, 2, ..., N_{\text{C}}, \tag{5.16}$$

N_{points} is the number of different combinations of channel realizations and noise variance.

The optimal β value is obtained via exhaustive search. The implemented algorithm, which is based on [17], is described here.

Algorithm 1 EESM Algorithm

```
**Input:** Vector b with $\beta$ values, vector $\text{BER}_{\text{AWGN}}$, vector $\text{SNR}_{\text{AWGN}}$, $\gamma_k(i)$, $\text{BER}_{meas}(i)$
**Output:** MSE, $\beta_{opt}$
for $\beta \in b$ do
    for $i = 1 : N_{points}$ do
        Calculate $\gamma_{eff}$ according to Equation 5.2
        if $\gamma_{eff} \in \text{SNR}_{\text{AWGN}}$ then
            Interpolate to find $\text{BER}_{EESM}(i)$ that satisfies Equivalence 5.3
        else
            Extrapolate to find $\text{BER}_{EESM}(i)$ that satisfies Equivalence 5.3
        end if
    end for
    Calculate MSE according to Equation 5.15
end for
Find minimum MSE
$\beta_{opt} \leftarrow \beta$ of minimum MSE
return $\beta_{opt}$
```

where

- N_{points} is the number of different combinations of channel realizations and noise variances,
- $\text{BER}_{meas}(i)$ is the measured BER for the i^{th} pair of channel realization and noise variance,
- $\gamma_k(i)$ is the vector containing the per subcarrier SNRs for the i^{th} pair of channel realization and noise variance.

5.4 Results

The BER vs SNR curves for a 2x1 MISO system with one spatial stream and STBC for static channel is shown in figure 5.3.

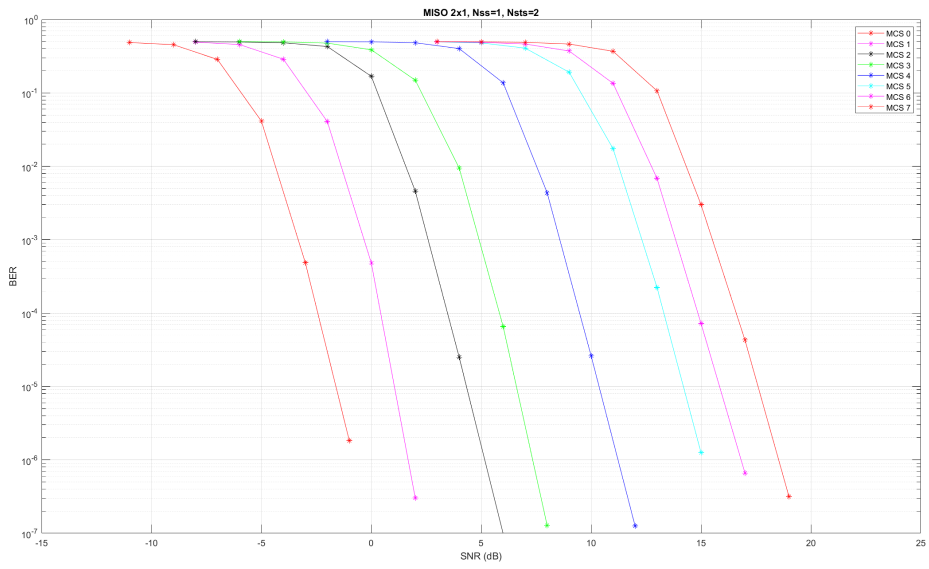

Figure 5.3: BER vs SNR curves for MCS 0-7.

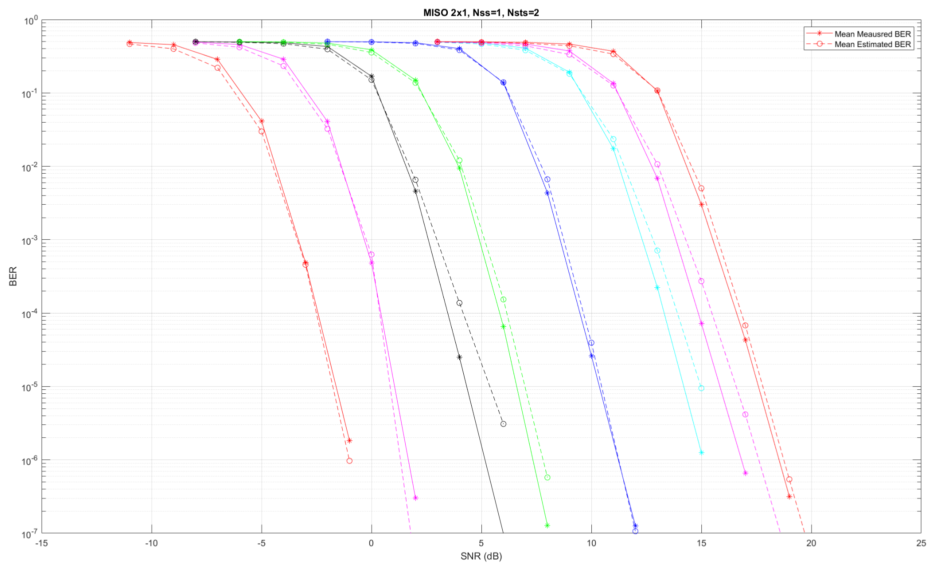

Figure 5.4: Measured and Estimated BER vs SNR curves for MCS 0-7.

Figures 5.4 and 5.5 showcase the performance of EESM.

This table contains the optimal β and the MSEs for all MCS. The first MSE is calculated by the criterion stated in 5.15. For the second MSE (MSE$_{\text{valid}}$), the last 100 realizations were used, in order to validate the performance of EESM.

MCS	MSE	MSE$_{\text{valid}}$	β_{opt}
0	0.0609	0.0594	0.5
1	0.0587	0.0763	1
2	0.0217	0.0229	1.32
3	0.0264	0.0330	2.88
4	0.0139	0.0084	5.99
5	0.0314	0.0246	14.07
6	0.0155	0.0167	19.68
7	0.0128	0.0186	25.25

Table 5.1: MSE values and β for MCS 0-7.

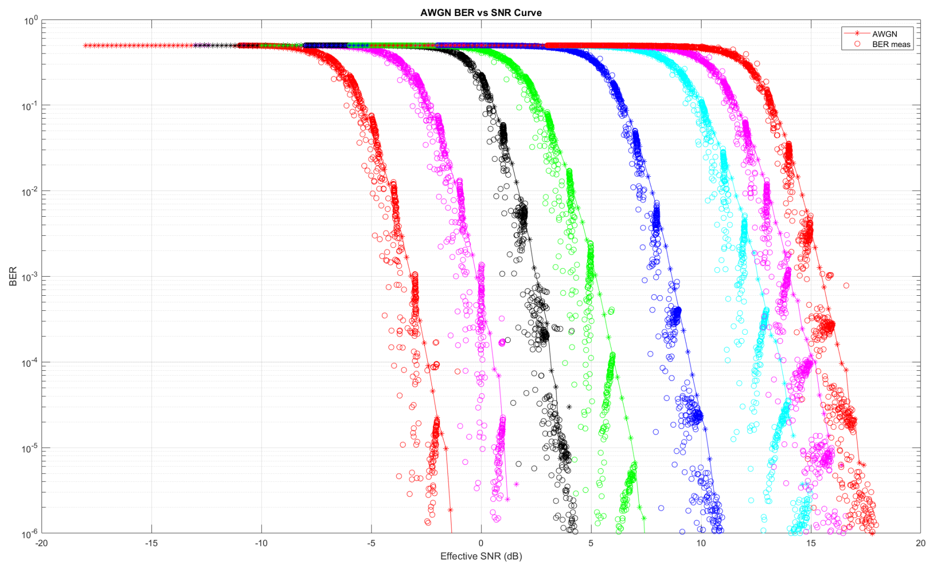

Figure 5.5: Validation of EESM for MCS 0-7.

In figure 5.6 there is an example of the MSE vs β graph for MCS = 0.
The MSE values presented in table 5.1 and figure 5.6 are multiplied by 10^2 for display reasons.
It can be observed that in some points there is a mismatch between the average measured BER and the average estimated BER. After examining the performance of EESM for individual measured BERs, we noticed that in some cases, different measured BER vs SNR curves corresponded to very similar estimated BER vs SNR curves. Even though the per subcarrier SNRs were different, γ_{eff} was approximately equal. That is because γ_{eff} acts as an exponential average, which means that it acts as a low-pass filter, giving more weight to the low SNR values. As a result, because in some cases, the EESM estimation was weak, the average estimated BERs are also affected.

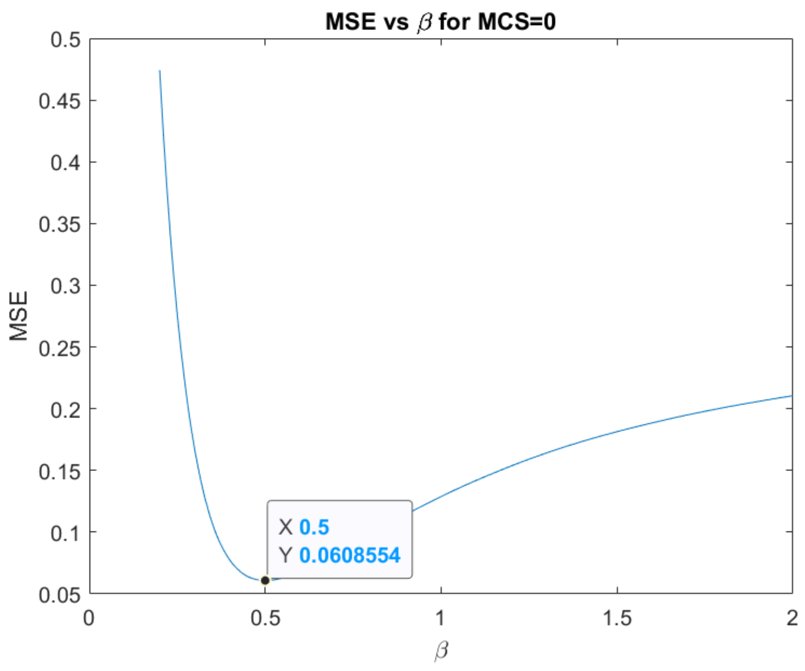

Figure 5.6: MSE vs β for MCS 0.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In conclusion, this thesis examines the performance of EESM in 802.11n systems with STBC. The implementation of the IEEE 802.11n Transmitter and Receiver was described in depth. A thorough analysis of the STBC technique was provided, emphasizing its applications in the IEEE 802.11n Standard. The EESM algorithm was introduced, providing its background theory and implementation. Several end-to-end simulations were conducted for a 2x1 MISO system with STBC for different MCS. The AWGN BER vs SNR curves that correspond to each of these MCS were simulated and the EESM method was applied. The results showcase that EESM is capable of providing an effective mapping function and therefore significantly decreasing the time complexity of the evaluation of the system, however, its performance is better for lower SNR values and decreases for higher SNR values.

6.2 Future Work

Future work regarding this topic can be concentrated on larger systems with more transmit and receive antennas and more spatial streams. Additionally, it would be interesting to examine the performance of EESM in more recent Wi-Fi Standards, such as Wi-Fi 6 (IEEE 802.11ax Standard), that support larger MCS and can achieve higher data speed and therefore higher performance. Another useful extension would be to employ the Signal-to-Interference-plus-Noise-Ratio (SINR) instead of the SNR in the mapping process since SINR is a stricter measure of a signal's quality as interference is also taken into consideration. Lastly, it would be of interest to attempt to use a tighter bound for the Q function and employ it to derive a mapping function for EESM and examine its performance.

References

[1] Pramod Viswanath David Tse. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005. ISBN: 9780511807213.

[2] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2005. ISBN: 9780471748823.

[3] Masoud Salehi John Proakis. *Digital Communications*. McGraw-Hill Education, 2007. ISBN: 9780072957167.

[4] Matthew S Gast. *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly Media, Inc., 2005. ISBN: 0596100523.

[5] Ishtiaq Ahmad and Kyunghi Chang. "Effective SNR Mapping and Link Adaptation Strategy for Next-Generation Underwater Acoustic Communications Networks: A Cross-Layer Approach". In: *IEEE Access* 7 (2019), pp. 44150–44164. DOI: 10.1109/ACCESS.2019.2908018.

[6] "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Corrigendum 1 – Correct IEEE 802.11ay Assignment of Protected Announce Support bit". In: *IEEE Std 802.11-2020/Cor 1-2022 (Corrigendum to IEEE Std 802.11-2020 as amended by IEEE Std 802.11ax-2021, IEEE Std 802.11ay-2021, and IEEE Std 802.11ba-2021)* (2022), pp. 1–18. DOI: 10.1109/IEEESTD.2022.9999411.

[7] P. Kyritsi Vinko Erceg V. L. Schumacher. *IEEE P802.11 Wireless LANs TGn Channel Models*. 2004.

[8] Emmanouil Logothetis. *Efficient WLAN Link Simulations by Means of EESM.* Oct. 2021.

[9] A. van Zelst and T.C.W. Schenk. “Implementation of a MIMO OFDM-based wireless LAN system”. In: *IEEE Transactions on Signal Processing* 52.2 (2004), pp. 483–494. DOI: 10.1109/TSP.2003.820989.

[10] A.J. Viterbi. “An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes”. In: *IEEE Journal on Selected Areas in Communications* 16.2 (1998), pp. 260–264. DOI: 10.1109/49.661114.

[11] S.M. Alamouti. “A simple transmit diversity technique for wireless communications”. In: *IEEE Journal on Selected Areas in Communications* 16.8 (1998), pp. 1451–1458. DOI: 10.1109/49.730453.

[12] Ericsson. *Considerations on the system-performance evaluation of HSDPA using OFDM modulation.* Oct. 2003.

[13] Ericsson. *System-level evaluation of OFDM – further considerations.* Nov. 2003.

[14] Upamanyu Madhow. *Fundamentals of Digital Communication.* Cambridge University Press, 2008. ISBN: 9780511807046.

[15] Athanasios Doukas and Grigorios Kalivas. “A Novel SNR per Subcarrier Estimation Scheme for OFDM Systems in Frequency Selective Channels”. In: *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications.* 2008, pp. 340–345. DOI: 10.1109/WiMob.2008.59.

[16] Sébastien Simoens et al. “Error prediction for adaptive modulation and coding in multiple-antenna OFDM systems”. In: *Signal Processing* 86.8 (2006). Special Section: Advances in Signal Processing-assisted Cross-layer Designs, pp. 1911–1919. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2005.09.033. URL: https://www.sciencedirect.com/science/article/pii/S0165168405003956.

[17] Sumit Roy Thomas R. Henderson Rohan Patidar. *Technical report on validation of error models for 802.11n.* 2017.