AALTO UNIVERSITY
School of Electrical Engineering

Sergio Damian Lembo

MODELING BLER PERFORMANCE OF PUNCTURED TURBO CODES

Thesis submitted for examination for the degree of Master of Science in Technology

Espoo 26.05.2011

Thesis supervisor:

Prof. Olav Tirkkonen

Thesis instructor:

Lic. Kalle Ruttik

AALTO UNIVERSITY

ABSTRACT OF THE
MASTER'S THESIS

Author: Sergio Damian Lembo

Title: Modeling BLER Performance
of Punctured Turbo Codes

Date: 26.05.2011 Language: English Number of pages: 10+75

Faculty: School of Electrical Engineering

Professorship: Radio Communications Code: S-72

Supervisor: Prof. Olav Tirkkonen

Instructor: Lic. Kalle Ruttik

In this thesis the author constructed a mathematical model using a continuous and differentiable expression, that approximates the BLER (Block Error Rate) performance of punctured turbo codes for different (SNR) Signal to Noise ratios, block lengths and MCS (Modulation and Coding Schemes) rates.
The model can be used for two different purposes. First, the model can be used to replace look-up tables in system level simulators; bringing the advantage of introducing a continuous set of block lengths and MCS rates, not available in look-up tables. Second, with the help of this function communications systems can be optimized using analytical methods based on differentiation.
The construction of the mathematical model was developed for the particular case of the Turbo Code $(13, 15)_8$ and QPSK modulation; but the derivation of the model can be generalized to other Turbo Codes and modulation types. The model was also applied to a different link configuration using 16 QAM modulation.

Keywords: BLER Curves, Link level model, System level model, Punctured Turbo Codes, Link to System (L2S), Sigmoid function.

Preface

This thesis consists of research work that has been carried out at the School of Electrical Engineering, Department of Communications and Networking, of Aalto University (former Helsinki University of Technology).

I would like to express my gratitude to my thesis supervisor, Professor Olav Tirkkonen, and instructor, Lic. Kalle Ruttik for their guidance and support. The advices and ideas received were essential to elaborate on the different parts that comprise this research, and arrive to the results presented in this thesis.

I would like to thank to all the people, and in particular to my family, that gave me their support while studying at TKK.

I would like to acknowledge the work of all the people within the academic and secretarial environment of TKK that directly or indirectly facilitated the means to let me complete my studies and at the same time engage in several academic projects.

Otaniemi, 26.05.2011

Sergio D. Lembo

Contents

Symbols and Acronyms

Symbols

Chapter 2

$BLER$	Block Error Rate performance metric
c_i	Coefficients used to change the slope and position of a sigmoid function
dB	Decibel
$f_1()$	Aggregation function
$f_2()$	Mapping function
i	General purpose index
$I()$	Information measure function
N	Number of metrics γ_i
P_e	*Link-level* performance metric
R	Rate for a given Modulation and Coding Scheme
$s()$	Sigmoid function
$s_1()$	Modified Sigmoid function
x	Independent variable
x_i	Variables providing additional details to the mapping between link and system level models
α_i	parameters used to tune the aggregation of metrics in the ESM method
$\gamma_{\text{Aggr.}}$	Aggregated performance metric
γ_{avi}	SNR calculated by the Actual Value Interface method
γ_{AWGN}	SNR in a reference model using AWGN channel
γ_{eff}	SNR calculated by the Effective SNR Mapping method
γ_i	*System-level* performance metric
$\gamma_{\text{Ref.}}$	Reference metric

Chapter 3

a	Fading amplitude of the channel
dB	Decibel
E_b	Transmitted energy per bit
f_c	Correction function in the Jacobian logarithm
j	Reference index
k	Position index used to identify bit u_k
k	Number of parity bits at the output of the turbo encoder
l	Half length of the puncturing mask, actually the total length is $2l$
$L()$	Log Likelihood Ratio (LLR)
$L_e(u_k)$	Extrinsic LLR for the bit u_k
L_c	Channel reliability value
$L(u_k)$	A-priori information of the bit u_k
m	Number of bits input to the encoder to return the trellis to a pre-defined state

M	Memory size of the encoder
n	Number of information bits
N	Number of iterations in the decoder
N	Last stage in the trellis
$P()$	Probability function
R	Code rate
s	Present state
$\acute{s}$	Previous state
S_k	State k in the trellis
u_k	Data bit k
x	Independent variable
$\underline{y}$	Received sequence of data bits with noise
y_{ks}	Received systematic bit k
y_{kl}	Received parity bit k
$\alpha_{k-1}(\acute{s})$	Probability that being in state $\acute{s}$ at time $k-1$, the received sequence up to this state is $\underline{y}_{j<k}$
$\beta_k(s)$	Probability that being in state $\acute{s}$ at time k the sequence to be received in the future will be $\underline{y}_{j>k}$
$\gamma_k(\acute{s}, s)$	Probability that being in state $\acute{s}$ at time $k-1$, with a transition to state s at time k, the received sequence is $\underline{y}_k$
σ^2	Noise variance

Chapters 4, 5, 6, 7

a_j	Coefficients used to fit a surface in the SNR, B, R model
B	Block length
$BLER$	Block Error Rate
$BLER_{sim}()$	Collection of BLER data samples obtained by a link level simulator for different values of SNR, B and R
c_i	Coefficients used to change the slope and position of a sigmoid function
dB	Decibel
$F_i()$	Complementary functions in the model to approximate the relationship between γ, B and R
$floor()$	Floor function, $floor(x)$ is the largest integer not greater than x
$G()$	Transfer function of the convolutional code
$G_i()$	Complementary functions in the model to produce a shift in the SNR axis from a known SNR at constant BLER
i	General purpose index
j	General purpose index
k	Number of parity bits at the output of the turbo encoder
k_j	Coefficients used to fit the proposed function describing the model
l	General purpose index
m	Number of trellis termination bits in the turbo encoder
M	Number of bits per symbol in the modulation scheme adopted
n	Number of information bits input to the turbo encoder

$P_e()$	Function expressing the Block Error Rate obtained by the proposed model
R	Rate for a given Modulation and Coding Scheme
$s()$	Sigmoid function
$s_1()$	Modified Sigmoid function
$s_2()$	Modified Sigmoid function
$S()$	Modified sigmoid function consisting of additional variables to construct the model.
SNR	Signal to Noise Ratio
u_i	Variable used to indicate the presence or absence of a term in the functional form proposed for the model.
x	Independent variable
γ	Signal to Noise Ratio
γ_{BR}	SNR for a given B and R at constant BLER
$\Gamma()$	Function approximating SNR for a given set of B and R

Operators

$\binom{n}{k}$	Number of Combinations of n elements taken k at a time
D	Delay operator in Convolutional Code
$\sum$	Sum
$\%$	Modulo operation

Acronyms

16 QAM	QAM with constellation of 16 points (4 bits per symbol)
3GPP	Third Generation Partnership Project
AMC	Adaptive Modulation and Coding
ARQ	Automatic Repeat reQuest
AVI	Actual Value Interface
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BLER	Block Error Rate
C/I	Carrier to Interference Ratio
CBRM	Circular Buffer Rate Matching
CC	Convolutional Code
EbN0	Energy per bit to Noise power spectral density ratio
ESM	Effective SNR Mapping - Effective SINR Metric
H-ARQ	Hybrid ARQ
L2S	Link to System
LLR	Log Likelihood Ratio
log-MAP	logarithmic Maximum A Posteriori
LTE	Long Term Evolution

MAP	Maximum A-Posteriori
MCS	Modulation and Coding Scheme
MIESM	Mutual Information ESM
OFDM	Orthogonal Frequency Division Multiplexing
QAM	Quadrature Amplitude Modulation
QPP	Quadratic Permutation Polynomial
QPSK	Quadrature Phase-Shift Keying
RSC	Recursive Systematic Convolutional
SINR	Signal to Interference and Noise Ratio
SIR	Signal to Interference Ratio
SNR	Signal to Noise Ratio
SOVA	Soft Output Viterbi Algorithm
TC	Turbo Code

1 Introduction

The study and analysis of large communication systems is usually carried out by relying on models that represent the characteristic behavior of the system to study and analyze. Models considering every bit to be transmitted through the links between transmitters and receivers (bit level models), are complex, unpractical and computationally demanding. The complexity of such models can be simplified by separating the system to study into *link-level* and *system-level* models. Both models are usually related by representing a *link-level* performance metric, such as Bit Error Rate (BER) or Block Error Rate (BLER), as a function of a *system-level* performance metric, such as (SNR), in *system-level* studies. Such separation allows abstracting the *link-level* complexity of each transceiver behind a mapping function relating link and system level performance metrics; examples of these mapping functions are relationships between BER-SNR and BLER-SNR. As such a BER-SNR or BLER-SNR relationship represents a simple abstract *link-level* model [1] [2].

In packet radio systems the *link-level* performance metric BLER is preferred to BER. In these systems the information is sent in packets, each constituting a block of constant length. Therefore the link performance is measured in terms of BLER for a given block length.

A common approach to represent the BLER-SNR relationship is to tabulate *link-level* simulation results into a look-up table. Unfortunately look-up tables have limitations. The tables can capture the BLER-SNR relationship only for a finite amount of link configurations: modulation, block size, and data rate configurations. Furthermore tabular representations do not allow using optimization methods based on differentiation in the determination of optimum network level policies.

The turbo code used in modern communication systems such as LTE [3] allows system designers to select among many combinations of block sizes and data rates. By using code puncturing it is possible to create a nearly continuous selection of code rates. A table describing a large amount of block sizes and data rates becomes unmanageable. In this thesis, the limitations imposed by look-up tables are overcome by constructing a mathematical model consisting of a function that describes the performance curves relating SNR and BLER of all available block lengths and data rates, Fig. 1.1.

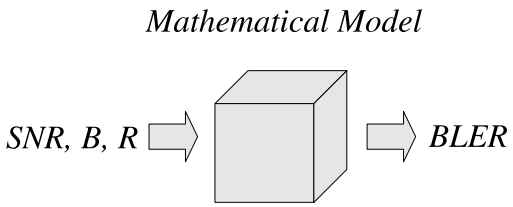

Figure 1.1: Graphical representation of the mathematical model to be constructed, depicted by a black box relating the input variables *SNR*, block length, *B*, and modulation and code rate, *R*; and the output variable *BLER*.

It can be noticed that the description of code performance here is essentially different from the code performance description used in channel code design. The coding community has developed detailed methods for characterizing code performance. In network level studies the complexity and precision of these methods is unnecessary. It is often sufficient just to have a function that approximates the BLER-SNR behavior; in network level investigations one of the most crucial design targets is the simplicity of the model. The biggest challenge of network modelers is to handle the complexity of large systems.

The application of the proposed model is twofold; in addition to replacing look-up tables, the proposed model can be used for performing optimizations using analytical methods based on differentiation. An example application is in the area of adaptive modulations and coding (AMC). In AMC the transmission rate selection is done by searching over all the possible modulation and coding scheme (MCS) rates. In [4] [5] this search has been facilitated by approximating the performance of a discrete set of MCS rates. In contrast, the model proposed in this thesis approximates performance of all the MCS rates by a continuous function and allows searching the optimum rate by differentiation. As illustrated in [6], optimization based on differentiation is an efficient way to find the switching points between several MCS.

When constructing mathematical models the problem that arises is about how to select suitable functions and the corresponding coefficients. With the increase of computing power the model selection can rely on some optimization process. The easiest approach for model selection is to provide a set of eligible functions to the computer and finding computationally the best fit to data over all combinations of those functions [7]. Unfortunately it is not know what the good initial functions are and for larger sets of functions the search space becomes unmanageable. In this thesis the construction of the desired model is performed by relying on exhaustive search with symbolic regression [8].

1.1 Objective and scope of the Research

In this thesis, the main objective it to construct a mathematical model that models the relationship between SNR and BLER for a continuous set of block lengths and data rates in a given communications system (Fig. 1.1).

It is desirable that the function describing the mathematical model fulfills certain properties that make the function suitable not only for mapping SNR and BLER, but also suitable to be applied in further analytical analysis. The function should be;

1. Continuous.
2. Differentiable.
3. Injective[1].
4. With asymptotic convergence toward $BLER = 1$, with $BLER \leq 1$, at low SNR values (i.e. in the region close to $BLER = 1$).

1.2 Author's contribution

In this thesis the author constructed a mathematical model using a continuous and differentiable expression, that approximates the BLER (Block Error Rate) performance of punctured turbo codes for different (SNR) Signal to Noise ratios, block lengths and MCS (Modulation and Coding Schemes) rates. In addition the author elaborated a systematic method to construct the model.

Furthermore the author reports different functions, each one providing different degrees of accuracy, as function of the number of terms in the model, for the particular cases of the Turbo Code $(13, 15)_8$, AWGN channel, and QPSK and 16 QAM modulations.

1.3 Applied Research Methods

The main research methods used in this thesis are based on system modeling and computer simulations. The table below summarizes the different techniques that were used in each stage of the construction and optimization of the mathematical model presented in this thesis.

- Generation of data samples by running simulations with a link-level simulator.
  - Monte Carlo methods.
- Construction of the model.
  - Symbolic regression.
  - Exhaustive search.
  - Non-linear fitting.
  - Observations of orthogonal projections from $\mathbb{R}^4$ to $\mathbb{R}^3$.
- Optimization of the model.
  - Symbolic regression.
  - Exhaustive search.
  - Non-linear fitting.

Data samples were obtained by running simulations with a link-level simulator. Monte Carlo method was used to generate the levels of noise in the channel, interleaving patterns and information bits input to the system.

[1] A function $f : A \rightarrow B$ is injective, or one-to-one, if whenever x_1, x_2 are distinct elements of the domain A, then $f(x_1), f(x_2)$ are distinct elements of the codomain B, i.e. different elements of the domain A are never mapped to the same element in the codomain B [9]. Thus the injective function property guarantees that for each SNR value (and for a given fixed block length and MCS rate) the function describing the model maps to only one $BLER$. Because the function describing the model should also be continuous, the same requirement can be stated by requiring that the function must be monotonically strictly decreasing.

Observations of orthogonal projections from $\mathbb{R}^4$ to $\mathbb{R}^3$ were carried out in order to select a preliminary set of functional forms eligible to approximate the data samples generated with the link level simulator.

Symbolic regression was used to find a mathematical expression, in symbolic form, that fits a given finite set of samples. Symbolic regression was combined with exhaustive search to evaluate all the combinations of the functional forms proposed.

Non-linear fitting was used to construct the mathematical model, consisting of non-linear functions, with the best fit to the data samples.

1.4 Structure and Organization of the Thesis

This thesis is organized as follows. Chapter 1 introduces the research topic, and presents the motivation, objectives and the author's contribution. Chapter 2 summarizes the basic theoretical background related to the different topics covered in the thesis, with the exception of Turbo Codes. Turbo Codes and puncturing of Turbo Codes are introduced in Chapter 3. Chapter 4 describes the link level system used for evaluation and development of the model throughout the thesis, using as example a system operated with QPSK modulation and in AWGN channel. Chapter 5 explains the steps followed to construct the mathematical model. Chapter 6 elaborates on the optimization of the proposed model, when it contains a reduced set of the terms. Chapter 7 applies the steps elaborated in previous chapters to a system operated with 16 QAM modulation. Finally, chapter 8 concludes the thesis. In addition, this thesis contains four appendices with complementary information.

2 Theoretical background

This chapter summarizes the basic theoretical background related to the different topics covered in the thesis, with the exception of Turbo Codes, that are covered in the next chapter. In the sections that follow, first the modeling of communications systems is introduced, and the relationship between *link-level* and *system-level* simulators is explained. Subsequently the Sigmoid function is introduced, the concept of symbolic regression is explained and SNR is defined.

2.1 Modeling of communications systems

The study and analysis of contemporary radio communication systems, as a complete system, is usually carried out by relying on simulations. In contrast, the study and analysis of a communications system can be also carried out by analytical methods. However analytical methods usually assume several simplifications to abstract the complexity of the system, and are not enough to understand and model the precise behavior of all the system as a whole entity.

In the simplest case a communication system consists of a transmitter, channel and receiver. In such scenario the transmitter and receiver can be thought as related together by a single link, and performance of the system can be determined at *bit level*. In other words, in a single link it is possible to analyze the effects of noise and conditions of the channel in every individual bit or symbol transmitted. Performance is then measured in terms of a suitable metric or indicator of the perceived quality of service by a user, like for example BER (Bit Error Rate) and/or BLER (Block Error Rate). This type of model is usually referred to as *link-level model*.

In more elaborated scenarios the number of transmitters and/or receivers is such that the number of links between transmitters and receivers increases significantly. In such scenario the analysis of a communications system at bit level becomes complex and unpractical; in particular when considering the management of resources among several base stations and mobile stations, interferences, traffic, mobility, interactions, events, channel models, etc. Simulation of these elaborated scenarios requires significant computational power and, in addition, simulations may require a high amount of time. Under these considerations, the analysis of the system is usually carried out using an abstract representation of each link at *signal level*. Performance is then measured in terms of a suitable metric or quality of service indicator, like for example SNR (Signal to Noise Ratio), SINR (Signal to Interference Noise Ratio), etc. This type of model is usually referred to as *system-level model*, and known also as *network-level model*.

A simple representation of the two models, *link-level model* and *system-level model*, is shown in Fig. 2.1. In the example depicted in the figure it can be observed that in the *link-level model* performance is measured in terms of BLER, and that in the *system-level model* performance is measured in terms of SNR.

When studying elaborated scenarios of communications systems, the *system-level model* is a convenient choice in terms of simplicity and computational cost. The *system-level*

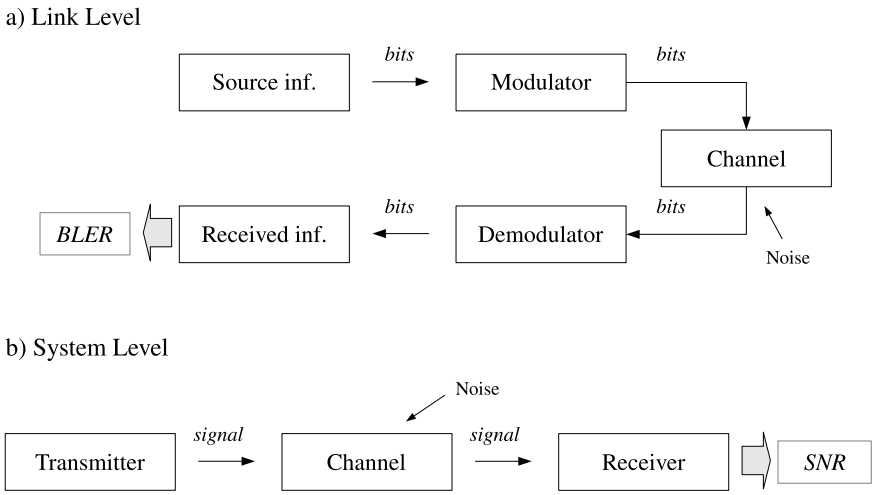

Figure 2.1: Differentiation between *link-level* and *system-level* models of a communications system.

model abstracts the bits transmitted in the links into a signal that can be easily used to add noise and interferences from other terminals in the network. But in the end, when looking at the performance of a terminal, the performance at *bit level* (BER or BLER, for example), is needed. When using *system-level* models it is thus necessary to find the correspondence between *system-level* performance metrics and *link-level* performance metrics. This thesis focuses particularly on finding a mathematical model that maps the *system-level* performance metric SNR to the *link-level* performance metric BLER (Fig. 2.2).

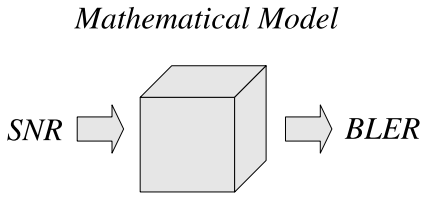

Figure 2.2: Mapping from *system-level* performance metric SNR to *link-level* performance metric BLER.

2.2 Relationship between *link-level* and *system-level* simulators

In the previous section it was explained that a *system-level* model is a convenient choice when studying elaborated scenarios of communications systems, and that it is necessary to find a correspondence between *system-level* performance metrics and *link-level* performance metrics. Such correspondence makes possible to abstract the bit level details of a communications link in a *system-level* model. The mapping between a *link-level* performance metric and a *system-level* performance metric can be established by means of a

tabular or functional relationship between the two metrics. Then the relationship between the two metrics is implemented inside a *system-level* simulator; thus allowing carrying out simulations at *system-level* and measuring performance at *link-level*. The interface between a *link-level* and *system-level* simulator is called *Link-to-System* (L2S) *interface* [10].

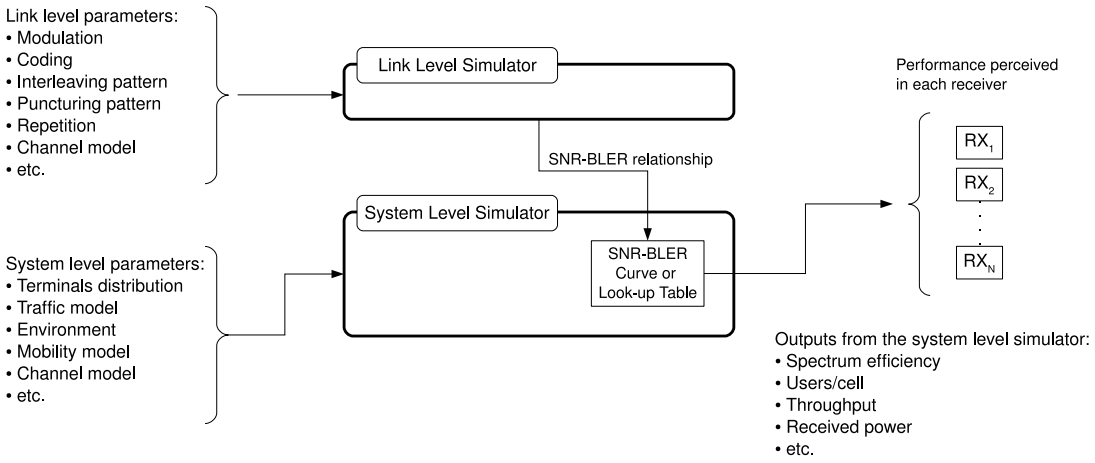

Figure 2.3: Relationship between Link and System level simulators (adapted from [1])

The L2S interface is shown on Fig. 2.3. The figure depicts a *link-level* simulator and a *system-level* simulator, both with the corresponding set of input parameters. In the example depicted in the figure, the *link-level* simulator is used to construct a function or look-up table relating two metrics, namely, a *system-level* metric (*SNR*) and a *link-level* metric (*BLER*). Then the *system-level* simulator relies on the mapping produced by the *link-level* simulator, and is able to abstract the details of each link at runtime; finding the mapping, for example, between *SNR* and *BLER*. In this way the *system-level* simulator is able to calculate the appropriate *link-level* metric for each receiver in the system.

The actual interface between a *link-level* performance metric and a *system-level* performance metric may require some pre-processing steps. Fig. 2.4 shows an example, that intends to be generic, showing how a collection of performance measurements at *system-level* γ_i are mapped to a final *link-level* performance metric P_e. The presence of a collection of performance measurements γ_i in a *system-level* simulator can be due to, for example, having multiple carriers inside a coding block. The notation presented on the top of the figure is used to present the general concepts, whereas the notation presented on the bottom of the figure corresponds to a particular example. The use and relative positioning of the L2S interface within a general context is explained in the following sub-sections with the aid of the steps presented on the figure.

In this thesis the main objective is to create a mapping between the *system-level* metric SNR and the *link-level* metric BLER, but in this section these metrics are generalized, and the explanations that follow are presented in terms of a *system-level* metric labeled γ and a *link-level* metric labeled P_e. Thus the goal stated in general terms is to map a

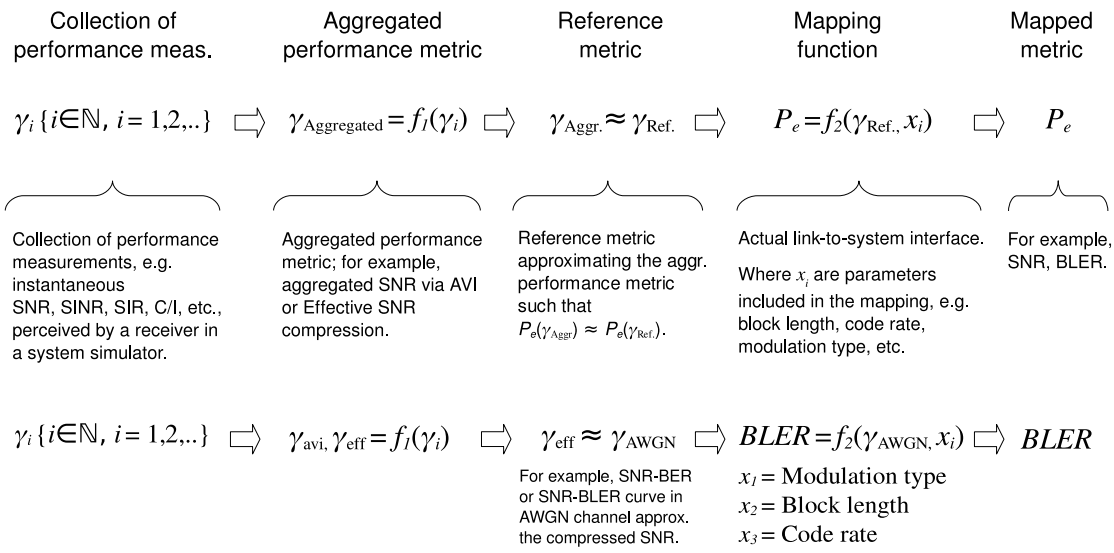

Figure 2.4: Graphical representation of the relative position of the interface between a *system-level* and *link-level* model implemented in a *system-level* simulator. In this generic example the *system-level* simulator provides a collection of *system-level* measurements γ_i, that subsequently are aggregated into a metric $\gamma_{Aggregated}$, and then approximated to a reference metric $\gamma_{\text{Ref.}}$. Finally the *system-level* performance metric $\gamma_{\text{Ref.}}$ is mapped to a *link-level* performance metric P_e. A particular example is shown at the bottom of the figure, aggregating a collection of instantaneous SNR measurements γ_i into an aggregated SNR γ_{avi} or γ_{eff}. The aggregated SNR corresponds to an equivalent SNR γ_{AWGN} that approximates a SNR-BLER curve in the AWGN channel. Finally γ_{AWGN} is mapped to $BLER$.

system-level metric γ to a *link-level* metric P_e.

$$f : \gamma \rightarrow P_e \tag{2.1}$$

2.2.1 Aggregation of performance measurements

In contemporary radio communication systems, e.g. wide-band digital communications using OFDM multi-carrier modulation, the metric measuring performance of the communication is calculated for each sub-carrier, resulting in a collection of performance measurements. In this case a measurement γ_i is used to indicate the *system-level* performance of the system in the i^{th} sub-carrier. Each γ_i is usually considered an estimate of instantaneous performance; hereafter the set of γ_i measurements it is referred to as *instantaneous measurements*. It is then desirable to aggregate, or compress, several measurements γ_i into a representative and scalar metric $\gamma_{\text{Aggr.}}$ suitable to be mapped to a *link-level* metric. There are different approaches to aggregate these measurements, γ_i, into a representative metric $\gamma_{\text{Aggr.}}$. Two of these approaches are AVI (Actual Value Interface) [11] and ESM (Effective SNR Mapping, also referred to as Effective SINR Metric) [12] [13].

AVI calculates a simple average of N measurements, e.g. corresponding to the instantaneous SNR from N sub-carriers,

$$\gamma_{\text{avi}} = \frac{1}{N}\sum_{i=1}^{N}\gamma_i. \tag{2.2}$$

ESM, instead, calculates a weighted average transformed with a function $I(\cdot)$ that characterizes the capacity of the channel, called *information measure*,

$$\gamma_{\text{eff}} = \alpha_1 I^{-1}\left(\frac{1}{N}\sum_{i=1}^{N} I\left(\frac{\gamma_i}{\alpha_2}\right)\right). \tag{2.3}$$

Parameters α_1 and α_2 are tuned considering the modulation type, coding, channel model, etc.

A proper selection and tuning of the parameters used in the compression of the measurements γ_i (e.g. parameters α_1 and α_2 in the case of ESM), will allow matching the *link-level* performance that would be obtained with the set of measurements γ_i, $P_e\left(\gamma_1, \gamma_2, ...\right)$, to an equivalent *link-level* performance, obtained now with the aggregated metric,

$$P_e\left(\gamma_{\text{Aggr.}}\right) \approx P_e\left(\gamma_1, \gamma_2, ...\right), \tag{2.4}$$

and in such a way that the mapping between the aggregated metric $\gamma_{\text{Aggr.}}$ and the *link-level* performance P_e (Fig. 2.4),

$$P_e = f_2\left(\gamma_{\text{Ref.}}, x_i\right), \tag{2.5}$$

with $\gamma_{\text{Ref.}} \approx \gamma_{\text{Aggr.}}$, is approximate to a reference curve; like for example a SNR-BLER curve calculated for AWGN channel (discussed in the next section).

In the example shown at the bottom of Fig. 2.4 it can be observed that a collection of instantaneous SNR measurements γ_i is aggregated to a SNR γ_{avi} or γ_{eff}.

2.2.2 Reference metric

After calculating the aggregated *system-level* metric, $\gamma_{\text{Aggr.}}$, the next step towards mapping system and link level metrics is to use this metric to estimate a *link-level* metric, P_e, but not only for a particular circumstance, like for a certain channel model and modulation type. Instead, it is desirable to have an aggregation method as generic as possible, such that, for example, several channel models and modulation types can be handled by the same compression method, and that the aggregated metric can be input to a reference function or look-up table containing a pre-computed relationship between *system-level* and *link-level* performance metrics for a reference model.

For example, in the case of ESM, by using suitable parameters α_1 and α_2, it is possible to obtain a SNR value γ_{eff} for different channel models, and different modulation types, such that it can be used to find an estimate of BLER from a basic SNR-BLER curve calculated for a reference model using an AWGN channel. In this case the parameters of γ_{eff} are tuned to obtain a γ_{eff} approximate to γ_{AWGN},

$$\gamma_{\text{eff}} \approx \gamma_{\text{AWGN}}, \tag{2.6}$$

such that [13],

$$\text{BLER}\left(\gamma_{\text{AWGN}}\right) \approx \text{BLER}\left(\gamma_{\text{eff}}\right) \approx \text{BLER}\left(\gamma_1, \gamma_2, ...\right). \tag{2.7}$$

Using a general notation the above reasoning is expressed as follows,

$$\gamma_{\text{Aggr.}} \approx \gamma_{\text{Ref.}}, \tag{2.8}$$

such that,

$$P_e\left(\gamma_{\text{Ref.}}\right) \approx P_e\left(\gamma_{\text{Aggr.}}\right) \approx P_e\left(\gamma_1, \gamma_2, ...\right), \tag{2.9}$$

where $\gamma_{\text{Ref.}}$ is a reference metric.

In the example shown at the bottom of Fig. 2.4 it can be observed that the aggregated SNR γ_{avi} or γ_{eff} corresponds to an equivalent SNR γ_{AWGN} that approximates a SNR-BLER curve in the AWGN channel.

2.2.3 Mapping between *system-level* metric and *link-level* metric

As discussed at the beginning of this chapter, a *system-level* simulator abstracts the link level details, and works with a set of metrics at *system-level*, but in the end, each receiver needs to evaluate the quality of service in terms of a *link-level* metric. Therefore it is necessary to create a suitable mapping between *system-level* metrics and *link-level* metrics. Typical *link-level* performance metrics are BER and BLER. Typical *system-level* performance metrics are SNR, SIR, SINR, C/I. The differentiation between *link-level* and *system-level* metrics is shown in Fig. 2.5.

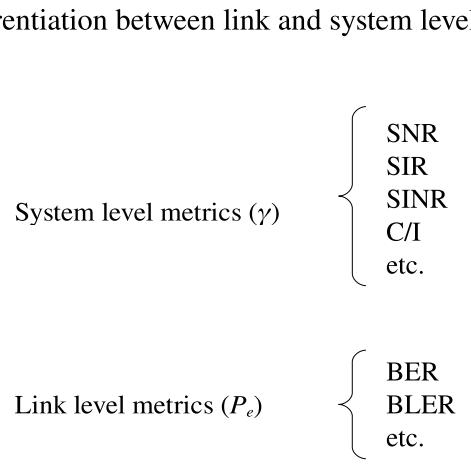

Figure 2.5: Differentiation between *link-level* and *system-level* metrics

The relationship between *link-level* and *system-level* metrics is usually established by performing simulations with a *link-level* simulator, and the mapping between the two metrics is either stored in a look-up table [14] [15] [16] or approximated with a function [4] [5];

Fig. 2.6 shows this differentiation. Alternative approaches to determine the mapping between the two metrics also exist, like for example generating the mapping directly with a *system-level* simulator and using a piecewise polynomial approximation in log. domain [17].

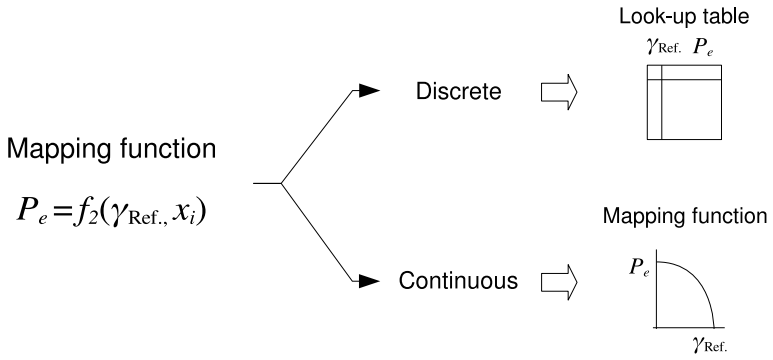

Figure 2.6: Differentiation of mapping approaches between a *system-level* metric $\gamma_{\text{Ref.}}$ and a *link-level* metric P_e; in the first case using a mapping table, and in the second case using an approximate curve given by a determined function.

Despite that, as mentioned in the previous sub-section, by tuning the parameters used in the ESM formula it is possible to approximate to a reference curve; models like the ESM do not take into consideration multiple variables, therefore different sets of look-up tables may be necessary for different modulation and coding schemes, block lengths, channel models, etc.

On the top of Fig. 2.4 it can be observed the mapping represented as a function f_2 that relates a target *link-level* metric P_e with a reference *system-level* metric $\gamma_{\text{Ref.}}$,

$$P_e = f_2\left(\gamma_{\text{Ref.}}, x_i\right). \tag{2.10}$$

The mapping is assisted with additional variables x_i. Variables x_i provide additional details to the mapping between the two metrics that were not included in the aggregation process, e.g. block length, modulation and coding rates [2], etc.

In the example shown at the bottom of Fig. 2.4 it can be observed the mapping between the *link-level* metric $BLER$ and the *system-level* metric γ_{AWGN} represented by a function f_2 constructed for AWGN channel,

$$BLER = f_2\left(\gamma_{\text{AWGN}}, x_i\right). \tag{2.11}$$

Where x_i are, for example, x_1 = modulation type, x_2 = block length, x_3 = code rate.

2.3 Sigmoid function

The sigmoid function (Fig. 2.7) is the main component in the mathematical model proposed in this thesis. It is defined as follows;

$$s(x) = \frac{1}{1 + e^{-x}}. \tag{2.12}$$

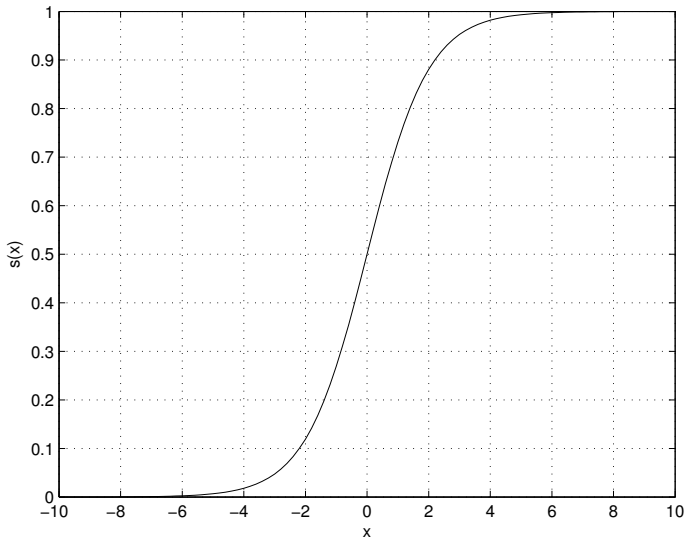

Figure 2.7: Sigmoid function

The sigmoid function fulfills the properties sought for the mathematical model to be constructed; it is continuous, differentiable, injective, and has an asymptotic convergence toward $s = 1$. The sigmoid is a function suitable to produce the desired model and with an appropriate transformation can approximate the mapping between SNR and BLER.

2.3.1 Sigmoid function observed in logarithmic domain

Throughout this thesis BLER is presented in logarithmic domain, therefore it is convenient to visualize the shape of the sigmoid function plotted in logarithmic domain (Fig. 2.8), in order to have it as reference when constructing a model that approximates SNR-BLER curves.

2.3.2 Modified sigmoid function

With suitable modifications to the sigmoid function it is possible to shift and change the curvature of the sigmoid while keeping its properties. Replacing the exponent $-x$ in equation (2.12) by the equation of a straight line $-c_1x - c_2$, the modified sigmoid function is written as follows,

$$s_1(x) = \frac{1}{1 + e^{-c_1x - c_2}}. \tag{2.13}$$

With this new function it is possible to alter the curvature of the sigmoid changing c_1 and shift the position of the sigmoid changing c_2. In Fig. 2.9 the modified sigmoid is plotted considering different values for c_1 and c_2.

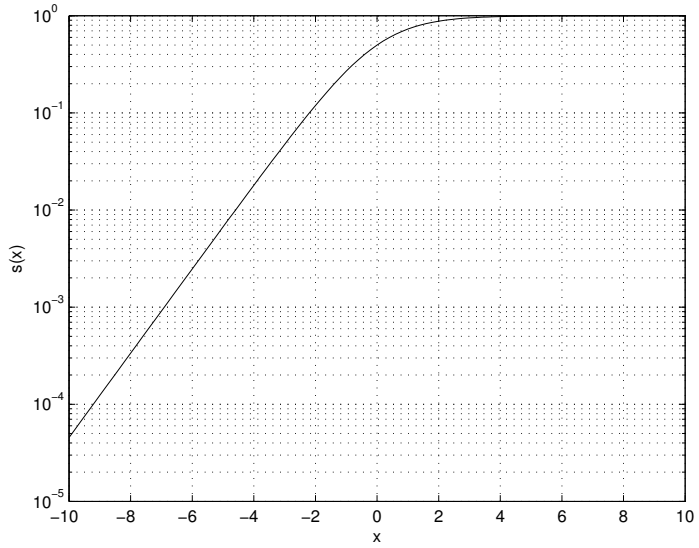

Figure 2.8: Sigmoid function observed in logarithmic domain

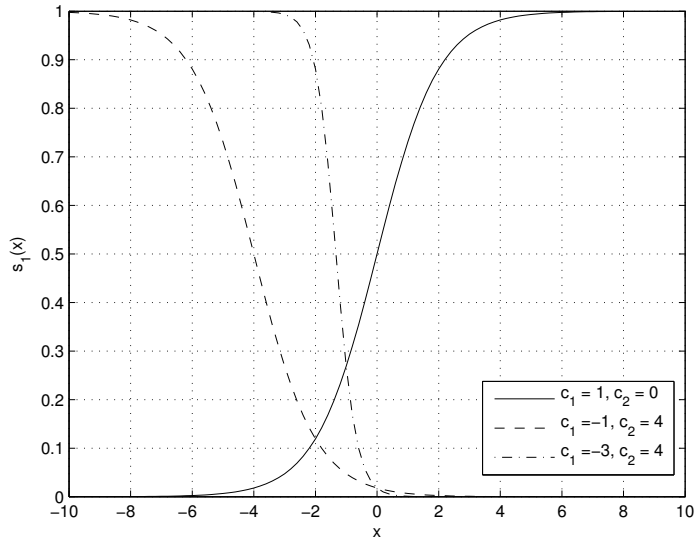

Figure 2.9: Modified sigmoid functions

2.4 Symbolic regression

Symbolic regression consists on finding a mathematical expression, in symbolic form, that fits a given finite set of sampled data points [8]. In general, symbolic regression involves also finding the appropriate numerical constants and coefficients that form part of the mathematical expression.

Symbolic regression differs from linear, quadratic or polynomial regression in that these methods just calculate the value of numeric coefficients in a given, fixed and pre-defined

mathematical expression.

Symbolic regression is usually implemented with genetic algorithms [8]; here, instead, the expression that describes the mathematical model is constructed performing an exhaustive search among all the combinations of a set of proposed functional forms.

2.4.1 Symbolic regression process

The process of symbolic regression consists on selecting random samples of the system to be modeled with respect to its independent variables, and then evaluating a proposed expression over the random set of samples. The performance of the proposed expression is evaluated against the results produced by a reference function describing the system, if known, or by relying on a fitting metric and making relative comparisons of performance among several expressions. The process is repeated several times, searching for a mathematical expression that provides an optimal fit for the given set of data.

2.5 Definition of SNR and relationship with EbN0

In the context of this thesis the relationship between EbN0 (energy per bit to noise power spectral density ratio) to SNR (Signal-to-Noise-Ratio) is defined as follows;

$$SNR_{\text{abs}} = EbN0_{\text{abs}} \cdot R, \tag{2.14}$$

where R is the modulation and coding rate in use; e.g. for QPSK modulation and coding rate $1/3$, R is equal to $2/3$. The subscript *abs* indicates that the units are in an absolute domain (i.e. without any limitation in the set of real numbers), and this notation is used to differentiate from units in the logarithmic domain. In this thesis, under the assumptions considered, there are no constraints regarding the maximum bandwidth available to the signal to be transmitted.

The same relationship in logarithmic domain, transformed to dB units[2], is as follows;

$$SNR_{\text{dB}} = EbN0_{\text{dB}} + 10\log_{10}(R). \tag{2.15}$$

In this thesis SNR units are preferred over EbN0 units. EbN0 units are often used in the field of coding theory to make a consistent comparison of performance between different codes, and the uncoded case, assuming that the energy allocated to each information bit is constant. Thus the comparison of performance between different codes is usually represented graphically with curves using an EbN0 scale against, for example, BER. These curves can be thought as being normalized in such a way that EbN0 is the energy allocated to each bit input to the encoder (assuming that noise is constant), regardless of how many bits are output from the encoder, and that this energy is common to all the codes being compared. On the other hand, performance curves plotted using a SNR scale can be thought as being normalized in such a way that SNR is the energy allocated to each bit

[2] In this thesis SNR is expressed in dB units, unless otherwise stated.

output from the encoder (assuming that noise is constant), regardless of how many bits are input to the encoder. SNR units are often used in studies of performance at *system-level*; in simple systems the power amplifier assigns a fixed transmission power to each bit regardless of the code rate in use to transmit the information.

3 Turbo Codes

Turbo codes [18] are a class of forward error correction codes that provide a high performance in error correction, close to the channel capacity.

Turbo codes encode the information by means of a turbo encoder that produces two encoded output streams of bits approximately statistically independent of each other. Some of the encoded bits output by a turbo code can be omitted (punctured) to obtain different code rates without changing the structure of the encoder. This feature allows trading off data rate by robustness of the transmitted data against noise in the channel. When the noise in the channel is significant all the encoded bits are transmitted, conversely, when the noise in the channel decreases, some encoded bits can be removed, provided that the decoder is able to recover the transmitted data, and thus increasing the transmitted data rate.

At the decoder, and in contrast to other type of decoders that utilize *hard* bits[3], a special decoding algorithm is used operating with *soft* bits. *Soft* bits provide not only an indication of whether a bit is 0 or 1 , but also the likelihood associated to the value of the bit.

The following sections provide a brief introduction to the main components and concepts related to turbo codes.

3.1 Turbo Encoder

The turbo encoder is a particular arrangement constituted by two convolutional codes (CC) encoders and an interleaver (Fig. 3.1). Often Recursive Systematic Convolutional (RSC) codes are used as the component codes due that these are non-catastrophic codes[4].

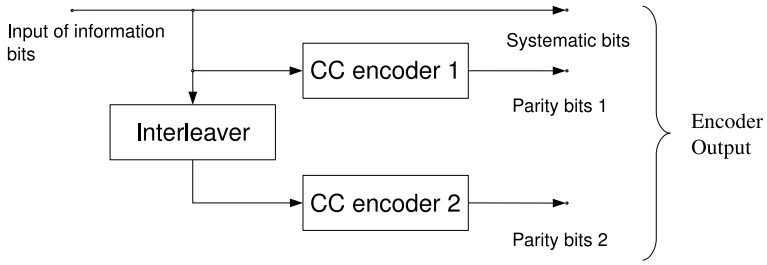

Figure 3.1: Turbo Encoder structure.

When the component convolutional encoders are RSC encoders (Fig. 3.2), the input stream of bits to the turbo encoder is output directly and receives the name of “systematic” bits. The two output streams encoded by the convolutional encoders receive the name of “parity” bits .

[3] Bit values quantized to 0 or 1.

[4] In catastrophic codes a finite number of channel errors produce infinite number of decoding errors.

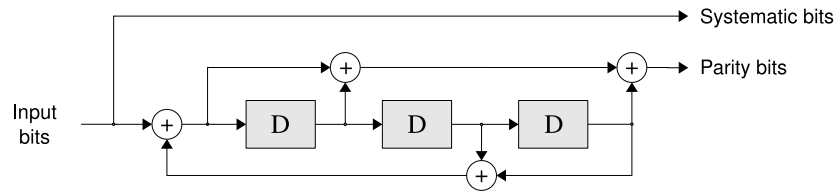

Figure 3.2: Recursive Systematic Convolutional Encoder $(13, 15)_8$.

The component convolutional encoders are described using a numeric notation that indicates the position of the taps between the shift registers for each output. Here the octal notation in the format *(recursive_output, forward_output)* is used; for example, the encoder shown in (Fig. 3.2) is described as a $(13, 15)_8$ encoder. Where *recursive_output* is the output in a RSC encoder that feedbacks the encoded bits to the input of the encoder, and *forward_output* is the output that directly outputs the encoded bits, without feedback, from the encoder.

3.2 Turbo Decoder

The general structure of a Turbo decoder consists of two component decoders linked by an interleaver at the input, and two additional interleavers at the output of the decoders (Fig. 3.3) [19]. Each decoder has three inputs; an input of systematic bits, an input of parity bits, and an input from the other component decoder. The systematic and parity inputs are the stream of bits received from the channel. The additional input from the other component decoder is referred also to as *a-priori* information, and it is extrinsic information provided by the other decoder about the likelihood of the bits being 0 or 1. Each component decoder outputs a stream of bits that are not quantized to binary values 0 or 1, denominated *soft* bits. *Soft* bits are real numbers that indicate by its sign the estimated value, 0 or 1 of each bit, and by its magnitude the numeric likelihood associated to the value of the bit (labeled as $L(u_k|\underline{y})$ in the figure).

In the turbo decoder, each component decoder provides an output containing extrinsic information. These outputs are feed as a-priori information to the complementary decoder. The extrinsic output indicates the likelihood of a bit to be 0 or 1 excluding the value of this bit received from the channel, and instead based on the information of the surrounding bits and the constraints imposed by the code being used.

The turbo decoder shown in (Fig. 3.3) operates iteratively. In the first iteration the first component decoder takes as input the stream of bits received from the channel corresponding to the bits encoded by the first encoder in the turbo encoder. Since it is the first iteration, the values of the a-priori information, not known at the beginning, are ignored or alternatively set to zero. The soft output provided by the first encoder is then used to calculate the extrinsic information, and it is then feed as a-priori information to the second component decoder. Then the second decoder estimates the bits relying now in the information received from the channel, associated to the stream encoded by the second

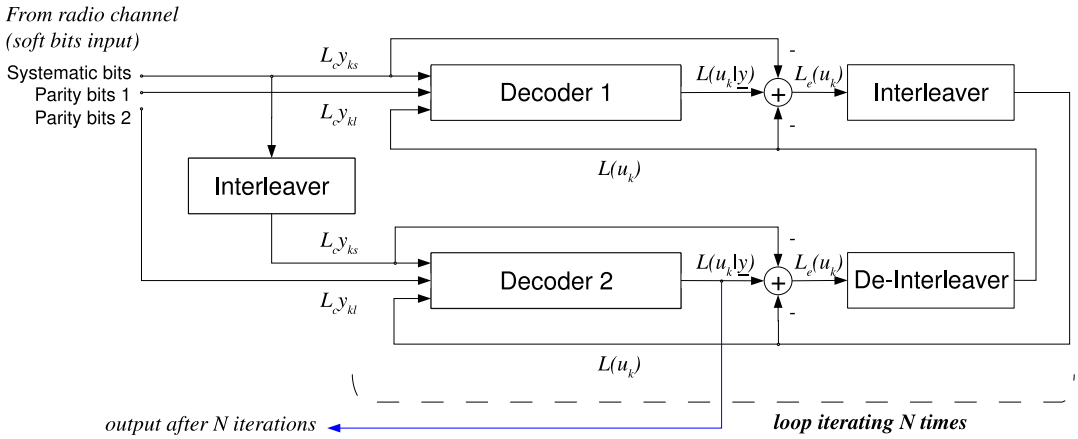

Figure 3.3: Turbo Decoder structure.

encoder in the turbo encoder, and the a-priori information supplied by the first decoder. Next the second iteration begins; the first decoder takes again as input the bits from the channel and in addition the a-priori information supplied by the second decoder in the previous iteration. As before, the second decoder takes as input the a-priori information from the first decoder and the received bits from the channel. This cycle is repeated until a given criteria is met, like for example after a given number of iterations. In each iteration the BER and BLER tends to decrease, nevertheless the gain in performance decreases as the number of iterations increases. When the iterations are completed the estimated sequence is output from the second component decoder.

The decoding algorithm used in each component decoder must output soft bits. Two suitable decoders for this purpose are the MAP (Maximum A-Posteriori) and SOVA (Soft Output Viterbi Algorithm) algorithms. In addition the MAP algorithm is simplified by the Log-MAP and Max-Log-MAP algorithms.

3.3 Log Likelihood Ratio (LLR)

A Log Likelihood Ratio (LLR) facilitates the passage of information between the different components that comprise the turbo decoder. The LLR is the logarithm of a proportion representing the ratio of the probability of a bit being 1 to the probability of the same bit being 0. The LLR for a data bit u_k, where k is a position index, is defined as follows:

$$L(u_k) \triangleq \ln\left(\frac{P(u_k = +1)}{P(u_k = -1)}\right), \tag{3.1}$$

where '+1' and '-1' are binary values equivalent to binary 1 and 0. In the literature, the adoption of these binary values helps to simplify posterior operations when deriving different mathematical expressions. The sign of the LLR is used to indicate the likelihood of the bit towards +1 or -1, and its magnitude is used to indicate the proportional likelihood associated to the value of the bit.

The probability of $u_k = +1$ or $u_k = -1$ can be derived from equation (3.1) ([19]) and is calculated as follows,

$$P(u_k = \pm 1) = \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}} \right) \cdot e^{\pm L(u_k)/2}. \tag{3.2}$$

The LLR based on the conditional probabilities of the bit u_k, given a received sequence of data bits with noise $\underline{y}$, is defined as,

$$L(u_k|\underline{y}) \triangleq \ln \left(\frac{P(u_k = +1|\underline{y})}{P(u_k = -1|\underline{y})} \right). \tag{3.3}$$

The conditional probabilities $P(u_k = \pm 1|\underline{y})$ are the probabilities that the soft decoder attempts to estimate. These probabilities are known as a-posteriori probabilities.

3.4 Maximum A-Posteriori (MAP) algorithm

The Maximum A-Posteriori (MAP) Algorithm [20], is an algorithm suitable to decode convolutional codes that operates with soft bits. The MAP algorithm is optimal in terms of minimizing the decoded bit error rate, it explores every possible path through the trellis and provides the estimated bit sequence, and also a probability for each bit stating the likelihood that the bit has been decoded correctly.

The MAP algorithm calculates the a-posteriori LLR $L(u_k|\underline{y})$,

$$L(u_k|\underline{y}) \triangleq \ln \left(\frac{P(u_k = +1|\underline{y})}{P(u_k = -1|\underline{y})} \right). \tag{3.4}$$

By using Baye's rule,

$$P(a \wedge b) = P(a|b) \cdot P(b), \tag{3.5}$$

equation (3.4) can be expressed as,

$$L(u_k|\underline{y}) \triangleq \ln \left(\frac{P(u_k = +1 \wedge \underline{y})}{P(u_k = -1 \wedge \underline{y})} \right). \tag{3.6}$$

As an aid to explain the derivation of the a-posteriori LLR $L(u_k|\underline{y})$, Fig. 3.4 shows a section of a trellis of a convolutional code with generators $(7, 5)_8$. In the figure an arbitrary present state in the trellis is labeled S_k, being S_{k-1}, S_{k-2} and S_{k-3} previous states and S_{k+1} a future state. The dashed lines connecting the different states of the convolutional code represent valid transitions when the bit is $u_k = +1$, and the continuous lines represent valid transitions when the bit is $u_k = -1$.

The probability $P(u_k = +1 \wedge \underline{y})$ in equation (3.6) is the joint probability between the bit $u_k = +1$ and the received sequence $\underline{y}$. The probability that $u_k = +1$ is equal to the probability of a transition in the trellis from a previous state S_{k-1} to a present state S_k,

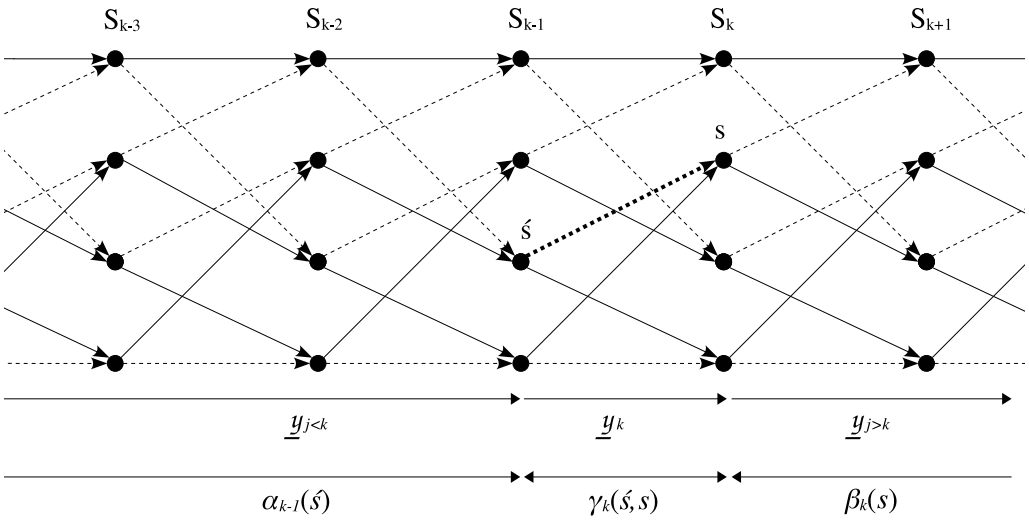

Figure 3.4: MAP Decoder Trellis for a RSC code with generators $(7, 5)_8$

where $u_k = +1$ is a valid transition in the finite state machine describing the convolutional code (dashed lines in the trellis section shown in Fig. 3.4). The set of transitions for $u_k = +1$ are mutually exclusive, therefore the probability that any of these transitions occur is equivalent to the sum of their individual probabilities. Therefore,

$$P(u_k = +1 \wedge \underline{y}) = \sum_{\substack{(\acute{s}, s) \Rightarrow \\ u_k = +1}} P(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y}), \tag{3.7}$$

where $(\acute{s}, s) \Rightarrow u_k = +1$ is the set of transitions from the previous state $S_{k-1} = \acute{s}$ to the present state $S_k = s$ when the bit $u_k = +1$.

Analogously when the bit $u_k = -1$ the joint probability $P(u_k = -1 \wedge \underline{y})$ can be expressed as the sum of the individual probabilities for the valid transitions when $u_k = -1$. Then equation (3.6) can be rewritten as,

$$L(u_k|\underline{y}) = \ln \left(\frac{\displaystyle\sum_{\substack{(\acute{s}, s) \Rightarrow \\ u_k = +1}} P(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y})}{\displaystyle\sum_{\substack{(\acute{s}, s) \Rightarrow \\ u_k = -1}} P(S_{k-1} = \acute{s} \wedge S_k = s \wedge \underline{y})} \right). \tag{3.8}$$

Considering the individual probabilities for the cases $u_k = +1$ and $u_k = -1$ in equation (3.8), the sequence of received bits $\underline{y}$ can be split into three sections; the received sequence before the present transition $\underline{y}_{j<k}$, the present transition $\underline{y}_k$ and the received sequence after the present transition $\underline{y}_{j>k}$ (refer to the example in Fig. 3.4); where j is a reference index. After some mathematical derivations [19], the probability $P(S_{k-1} = \acute{s} \wedge$

$S_k = s \wedge \underline{y}$) can be divided into the product of three probabilities, $\alpha_{k-1}(\acute{s}) \cdot \gamma_k(\acute{s}, s) \cdot \beta_k(s)$. The value $\alpha_{k-1}(\acute{s})$ represents the probability[5] that being in state $\acute{s}$ at time $k-1$, the received sequence up to this state is $\underline{y}_{j<k}$. The value $\gamma_k(\acute{s}, s)$ represents the probability[5] that being in state $\acute{s}$ at time $k-1$, with a transition to state s at time k, the received sequence is $\underline{y}_k$. The value $\beta_k(s)$ represents the probability[5] that being in state s at time k the sequence to be received in the future will be $\underline{y}_{j>k}$. Therefore (3.8) can be rewritten as follows,

$$L(u_k|\underline{y}) = \ln \left(\frac{\sum\limits_{\substack{(\acute{s}, s) \Rightarrow \\ u_k = +1}} \alpha_{k-1}(\acute{s}) \cdot \gamma_k(\acute{s}, s) \cdot \beta_k(s)}{\sum\limits_{\substack{(\acute{s}, s) \Rightarrow \\ u_k = -1}} \alpha_{k-1}(\acute{s}) \cdot \gamma_k(\acute{s}, s) \cdot \beta_k(s)} \right). \tag{3.9}$$

The values $\alpha_{k-1}(\acute{s})$ can be calculated by a forward recursion starting with known initial values, $\alpha_0(0) = 1$ and $\alpha_0(s) = 0$ for all $s \neq 0$. The values $\beta_k(s)$ can be calculated by a backward recursion starting with initial values $\beta_N(s) = 1$, for all s; where N is the last stage in the trellis. Due to trellis termination some transitions at the end of the trellis are not possible, corresponding to values $\gamma_k(\acute{s}, s)$ equal to zero for such transitions. Alternatively it is possible to set $\beta_N(0) = 1$ and $\beta_N(s) = 0$ for all $s \neq 0$ in the case of practical implementations where the trellis is terminated to the all zero state, and the decoding algorithm executes all the transitions up to the end of the trellis (including those that are not even possible due to trellis termination). In this case, setting the values $\beta_N(s) = 0$ for all $s \neq 0$ has an equivalent effect as avoiding the non existing transitions due to trellis termination, with the advantage that it simplifies the implementation of the algorithm. The values $\gamma_k(\acute{s}, s)$ are calculated from the input bits received from the channel, and from the a-priori LLRs $L(u_k)$ (obtained by the other component decoder). Then by performing a forward recursion, $\gamma_k(\acute{s}, s)$ values are used to compute values $\alpha_k(\acute{s}, s)$. Finally, when all the bits are received, a backward recursion takes place and computes all the values $\beta_k(\acute{s}, s)$.

3.5 Additional considerations on Turbo Decoding

With additional mathematical derivations, the a-posteriori LLR $L(u_k|\underline{y})$ can be rewritten as follows ([19]),

$$L(u_k|\underline{y}) = L(u_k) + L_c y_{ks} + L_e(u_k). \tag{3.10}$$

The first term in equation (3.10), $L(u_k)$, is the a-priori LLR of the k'th bit. In an iterative turbo decoder this LLR is information passed by the other decoder (see Fig. 3.3).

The second term in equation (3.10), $L_c y_{ks}$, is the systematic bit u_k sent through the channel, and received as y_{ks}. L_c is a parameter that indicates how reliable is the channel, named channel reliability value. L_c depends of the fading amplitude of the channel, a,

[5] Actually it is a scaled value related to the stated probability.

and the signal to noise ratio, $\frac{E_b}{2\sigma^2}$. When using BPSK modulation over a Gaussian channel, the channel reliability value is,

$$L_c = 4a\frac{E_b}{2\sigma^2}, \tag{3.11}$$

where E_b is the transmitted energy per bit, and σ^2 is the noise variance. The higher the signal to noise ratio, the higher the influence of the systematic bit in the a-posteriori LLR $L(u_k|\underline{y})$. Conversely, the lower the signal to noise ratio, the lower the impact in the a-posteriori LLR.

The third term in equation (3.10), $L_e(u_k)$, is the extrinsic LLR for the bit u_k. This LLR is derived using the constraints imposed by the adopted code, from the a-priori information $L(u_k)$, and the received information from the channel $\underline{y}$, excluding the received systematic bit y_{ks} and the a-priori information $L(u_k)$ (see Fig. 3.3).

3.6 Log-MAP algorithm

For the calculation of the a-posteriori LLR $L(u_k|\underline{y})$, the MAP algorithm requires in equation (3.9) the values $\alpha_{k-1}(\acute{s})$, calculated in a forward recursion, the values $\beta_k(s)$, calculated in a backward recursion, and the transition probabilities $\gamma_k(\acute{s}, s)$. The Log-MAP algorithm, [21], operates in the logarithm domain, working with the logarithms of the values $\alpha_{k-1}(\acute{s})$, $\beta_k(s)$ and $\gamma_k(\acute{s}, s)$.

The Log-MAP algorithm is equivalent in decoding performance to the MAP algorithm; in addition it reduces computational complexity and solves number representation problems. In the computation of values $\alpha_{k-1}(\acute{s})$ and $\beta_k(s)$ products are replaced by additions, and additions are replaced by $\max$ operations and a correction term. The Log-MAP algorithm leads to an improved numerical stability, in particular when it is implemented in hardware with limited representation of real numbers.

Operations that consist of calculating the logarithm of the addition between two exponentials, are simplified with the Jacobian logarithm [22];

$$\ln(e^{x_1}+e^{x_2}) = \max(x_1, x_2)+\ln\left(1 + e^{-|x_1-x_2|}\right) = \max(x_1, x_2)+f_c\left(|x_1 - x_2|\right), \tag{3.12}$$

where f_c is a correction function.

3.7 Rate matching and puncturing

Rate matching is related to the variation of the code rate by removing systematic or parity information output by the encoder. In Turbo Codes implemented with the structure presented in Fig. 3.1 the code rate of the mother code is (ideally) $1/3$. This means that for every information bit input to the encoder the turbo code outputs 3 bits; one systematic bit and two parity bits (usually labeled parity 1 and parity 2). Thus for n information bits the code rate is $n/(3n)$. The rate is ideally $1/3$ because in fact m bits in addition to the n information bits are input to the encoder to return the trellis termination to a pre-defined state (commonly inputting binary zeros and thus returning the state of the encoder to the

zero state)[6]; leading to a code rate of $n/(3n+m)$. The code rate of the mother code $n/(3n+m)$ can be increased by removing some of the systematic or parity bits; such removal of bits is called puncturing.

Traditionally, puncturing is performed only in parity bits, however in a turbo code with a well designed interleaver the puncturing of systematic bits is preferred due that the parity bits contain most of the Hamming weight in the minimum distance [23]. Nevertheless puncturing of systematic bits must be exercised with care due that certain puncturing patterns in the systematic bits can transform the code in catastrophic. At high coding rates, where excessive puncturing of parity bits can degrade the code performance, the optimal puncturing pattern is attained by puncturing a portion of systematic bits and a portion of parity bits [23]. When puncturing only parity bits the code rate is increased to $n/(n+m+k)$, where k is the number of parity bits (with $k < 2n$ when puncturing takes place).

The author found in the literature two different methods to perform the puncturing of bits. The first puncturing method receives the name of *circular buffer rate matching* (CBRM) method [23]. The second puncturing method utilizes a puncturing mask, containing a puncturing pattern indicated by zeros and ones [24] [25]. These methods are explained in detail in the next sub-sections.

3.7.1 Puncturing method using circular buffer rate matching (CBRM)

The method of puncturing using circular buffer rate matching (CBRM), consists on interleaving independently each one of the three output streams of bits from the turbo encoder: systematic, parity 1 and parity 2, with a sub-block interleaver. Then a buffer gathers the interleaved systematic bits (arranging these at the beginning of the buffer) and parity bits (at the end of the buffer), Fig. 3.5 [3] [23]. The parity bits (parity 1 and parity 2) are actually interlaced to each other bit-by-bit. This buffer is called *circular buffer* by the fact that it wraps around itself; bits can be taken from a starting position of the buffer, and if it is the case, the bits that are taken after the end of the buffer are the bits at the beginning of it. By relying on this buffer the rate matching is simplified in the sense that a desired rate for a given block length to transmit is composed by just taking a portion of systematic bits followed by a portion of parity bits. Furthermore the use of this buffer simplifies the selection of different sequences of bits to transmit as complementary data for H-ARQ.

3.7.2 Puncturing method using puncturing masks

The second method of puncturing utilizes a puncturing mask, containing a puncturing pattern indicated by zeros and ones [24] [25]. The presence of a 1 indicates that the bit is

[6] The turbo code considered here uses a trellis termination technique. In this technique bits are input to the encoder forcing it to return to the zero state. Another termination technique, referred to as *tail-biting*, avoids using extra termination bits to terminate the trellis. This technique has the advantage of not affecting the code rate, $n/(3n)$, but has the disadvantages of increasing the complexity at the receiver, and increasing latency due to the need of a training sequence to determine the initial state in the trellis.

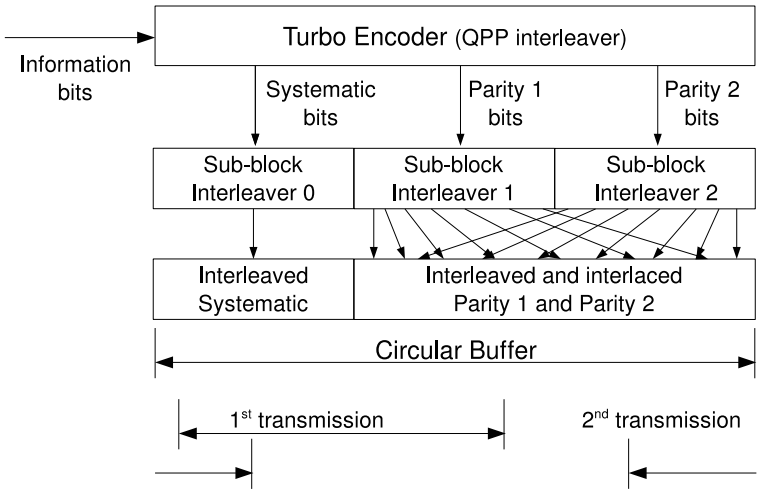

Figure 3.5: Circular buffer rate matching (CBRM) structure at the output of the Turbo Encoder (adapted from [23]).

retained, and the presence of a 0 indicates that the bit is punctured. The puncturing mask has a definite length and it is applied consecutively to the stream of systematic or parity bits output from the encoder.

The puncturing pattern should be selected by considering the interleaving pattern used in the interleaver; some puncturing patterns may lead to non optimal performance when combined with the interleaving pattern. However, when the interleaving pattern is random, the puncturing pattern does not have significant effect, in average, on the code performance for most of the code rates [25]. When the interleaving pattern is random, the puncturing pattern applied to the second stream of parity bits is irrelevant, due that the pattern will be randomly distributed when de-interleaving this stream of bits. The puncturing pattern applied to the first stream of parity bits may present poor performance for some code rates. The relationship between the puncturing patterns applied to the first and second stream of parity bits is also irrelevant, since there is almost no correlation between the bits in the first and second parity sequence.

According to [25], the puncturing pattern may affect with poor performance the code rates that retain bits always in the same position within the periodic pattern generated by the system response of the convolutional encoder. Detailed description of this phenomenon is given in the following sub-section.

3.7.3 Considerations on puncturing patterns

According to [25] some code rates may present poor performance for some puncturing patterns[7]. The cause of this poor performance is explained by considering the theory of

[7] The case considered here assumes that the bits output by the encoder are directly punctured. An alternative solution to the one presented in this section is to interleave the bits to be punctured with a

output weight distributions [26].

When an single nonzero bit is input to a recursive convolutional encoder, the impulse response at the output generates a periodic pattern due to the feedback connection in the encoder. When more nonzero bits are input, the output may present also periodic patterns in certain period. Since the feedback polynomial for the RSC code is usually selected to have maximal length, the maximum achievable period for an encoder with memory size M is a period with length $2^M - 1$ bits.

Often the puncturing mask used in each parity output has a definite length, that in practice is $2l$, with $2 \leq l \leq 16$. For such length the maximum achievable rate is $R = l/(l+1)$. This can be interpreted as inputting $2l$ bits to the encoder and retaining only one bit of each parity output ($R = 2l/(2l_{\text{systematic bits}} + 1_{\text{parity1 bit}} + 1_{\text{parity2 bit}}) = l/(l+1)$).

In the case that only one bit of each parity output is retained, a maximum of $2^M - 1$ positions in the period are available. Always considering a puncturing mask of length $2l$; for most of the rates, retaining one bit among $2l$ bits causes a cyclic positioning inside the $2^M - 1$ positions in the output periodic pattern. For example, for $M = 4$, the maximum length of the periodic pattern is 15; for $l = 2$ the code rate is $2/3$ and the length of the puncturing mask is 4. Then, without loss of generality, retaining the first bit in the puncturing mask causes a relative selection of bits 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15, 4, 8, 12 in the periodic pattern (Fig. 3.6).

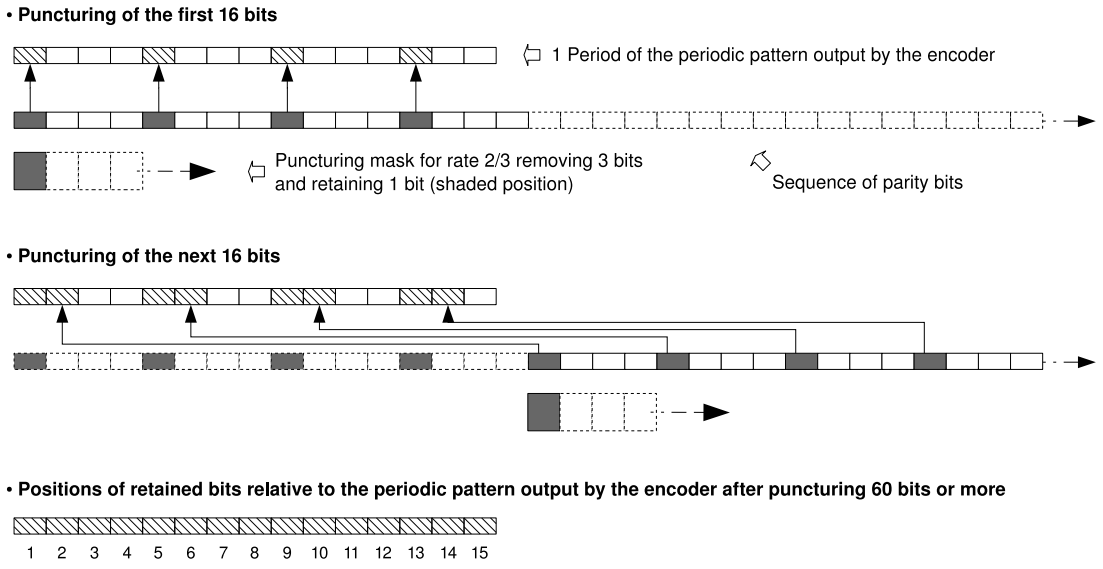

Figure 3.6: Retained bits after puncturing the sequence of parity bits, and positions of retained bits relative to the periodic pattern output by the encoder (convolutional encoder with memory size 4 and puncturing rate 2/3).

However, for a few code rates the length of the puncturing mask $2l$ and the length of the periodic pattern, $2^M - 1$, have the same least common multiple, causing a selection of a

properly designed interleaving pattern.

few positions in the periodic pattern. The length of the puncturing mask and the length of the periodic pattern should not have a common multiple, due that this leads to catastrophic (or semi-catastrophic) puncturing of the code [24]. For example, considering $M = 4$, and $l = 5$, the code rate is $5/6$ and the length of the puncturing mask is 10. Now the length of the periodic pattern, $2^M - 1 = 15$, and the length of the puncturing mask, $2l = 10$, have the same least common multiple 5. Then, without loss of generality, retaining the first bit in the puncturing mask causes a relative selection of bits 1, 11, 6 in the periodic pattern (Fig. 3.7). It can be observed that in this case the 15 positions available are not being used; if the selected positions happen to be all zeros, then the weight of the output sequence will be very low after puncturing.

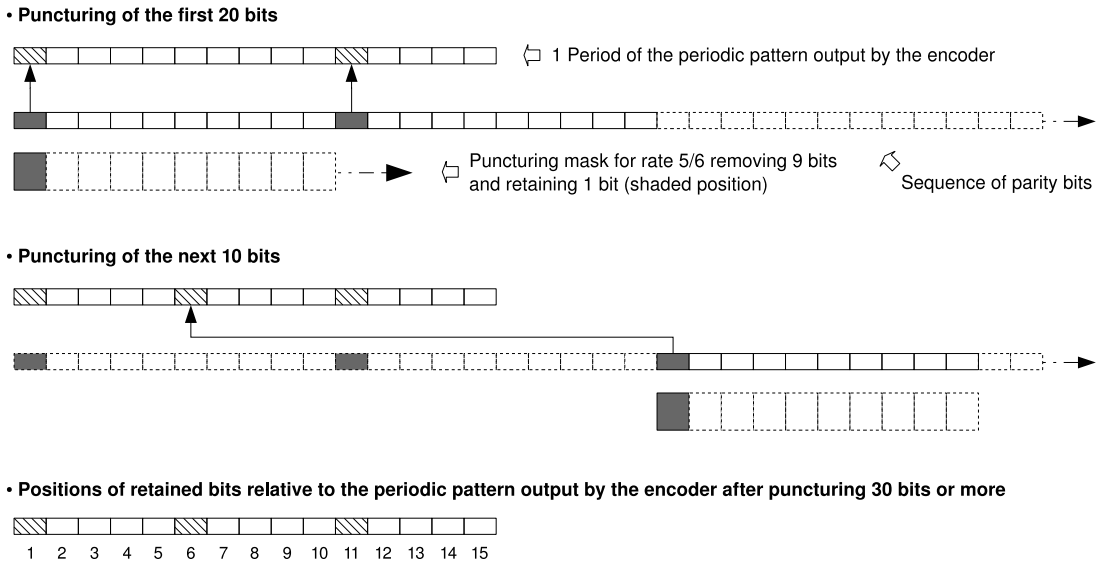

Figure 3.7: Retained bits after puncturing the sequence of parity bits, and positions of retained bits relative to the periodic pattern output by the encoder (convolutional encoder with memory size 4 and puncturing rate 5/6).

The turbo code $(13, 15)_8$, used later in this thesis, has a memory size $M = 3$, creating a periodic pattern of maximum of length $2^M - 1 = 7$. The puncturing patterns used to increase the rate of the code were selected by skipping those puncturing patterns that produce poor performance. Appendix A contains the list of puncturing patterns used to obtain different rates for the turbo code $(13, 15)_8$.

3.8 BLER curve

In the context of this thesis, a BLER curve is the curve that describes the performance of a Turbo Code in BLER units against SNR units, and for a given block length and MCS rate. An example of BLER curves is shown in Fig. 3.8. BLER curves have two characteristic regions, named waterfall region and error floor.

3.8.1 Waterfall region and error floor

Performance curves for Turbo Codes have a region in which the slope decreases steadily as the SNR increases; this region is called the waterfall region. After the waterfall region there is a slope transition region with reduced slope called the error floor region (Fig. 3.8) [27].

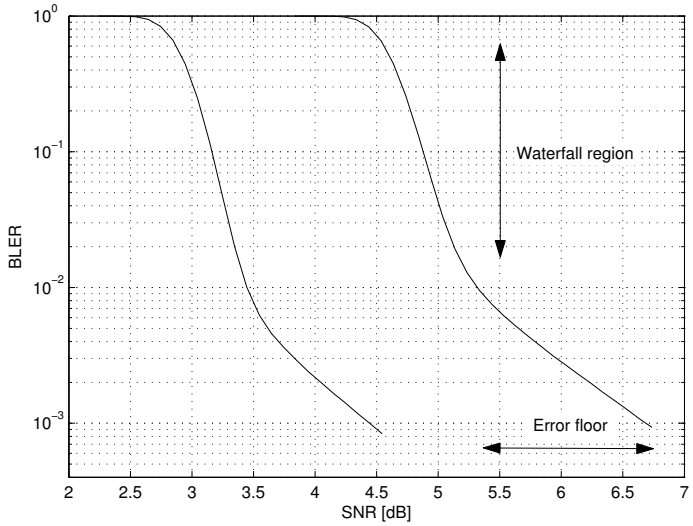

Figure 3.8: BLER curves

4 System and model description

In this thesis the main goal is to construct a mathematical model using a continuous and differentiable expression, that approximates the relationship between the *system-level* performance metric SNR and the *link-level* performance metric BLER of a *link-level* system composed by a turbo code. The turbo code considered operates with variable block lengths, B, and encodes information at different rates, obtained by puncturing, allowing transmitting information at different MCS rates, R. The construction of the mathematical model was developed for the particular case of the Turbo Code $(13, 15)_8$ and QPSK modulation; but the derivation of the model can be generalized to other turbo codes and modulation types.

In order to construct the model, a link level simulator was used to generate BLER data samples (hereafter referred to as *simulated BLER data samples*). The simulated BLER data samples are samples of BLER generated for different values of SNR, B and R.

At each signal to noise ratio (SNR, γ) level a given B and R combination provides a certain block error rate ($BLER$) performance. At *system-level* it is desired to have a function, $P_e()$, that abstracts the *link-level* details. Such function simply relates the performance of the QPSK modulated turbo code to the independent variables:

$$BLER = P_e(\gamma, B, R). \tag{4.1}$$

In the following chapters a model for (4.1) is derived by approximating a suitable function to the simulated BLER data samples. The simulated samples were obtained with the link level simulator for various (γ, B, R) combinations. Each BLER value for one of these combinations constitutes one SNR-BLER sample.

The model is constructed by fitting a suitable function in least-square sense to the simulated BLER data samples. The proposed function is selected by performing an exhaustive search among several functional forms, where each functional form is composed by a set of terms. Clearly by selecting more complex combinations of terms it is possible to have a better fit to the simulated BLER samples. The key question here is how to identify which set of terms produces the most suitable fit, i.e. the best combination with 4 terms, 5 terms, etc. This decision is circumvented by evaluating how close the proposed function approximates the simulated samples for all possible combinations of the terms. Such exhaustive search allows selecting the combination that produces the best match for a given number of terms.

The next sections describe the system under consideration and the performance metric used in the evaluation of the function describing the model.

4.1 General system description

A link level simulator was used to generate BLER data samples. The link was modeled using the parallel concatenated convolutional coded turbo code (TC) $(13, 15)_8$. The TC adopted is the TC standardized for 3GPP [3], although with a random interleaver and

different puncturing method. The TC is constructed from two recursive systematic convolutional codes concatenated with a random interleaver (Fig. 4.1). The transfer function of the constituent code is

$$G(D) = \left[1, \frac{1 + D + D^3}{1 + D^2 + D^3}\right], \tag{4.2}$$

where D is the delay operator.

Various MCS rates R are generated systematically by puncturing bits from the parity outputs of the TC. The puncturing method implemented is puncturing with puncturing masks. Puncturing masks were constructed by considering an even distribution of punctured bits. According to [25], when a pseudo-random interleaver is used, the selection of the puncturing pattern does not have significant effect in the code performance, as long as the puncturing pattern is not causing a cyclic effect in the bit locations inside the periodic pattern generated by the system response of the convolutional encoder (refer to sub-sections 3.7.2 and 3.7.3 for additional details).

The puncturing patterns were selected by skipping those puncturing patterns that produce poor performance. Appendix A contains the list of puncturing patterns used to obtain different rates. The final MCS rate is determined by combining n information bits, $m = 3$ trellis termination bits and k parity bits;

$$R = M\frac{n}{n + m + k}, \tag{4.3}$$

where M are the number of bits per symbol in the modulation scheme adopted. In the main mathematical model elaborated in this thesis the performance is evaluated for QPSK modulation (M = 2) and AWGN channel.

For different MCS rates the amount of information bits n is adjusted such that the amount of encoded bits fits into a transmission block of a given block length B;

$$B = n + m + k. \tag{4.4}$$

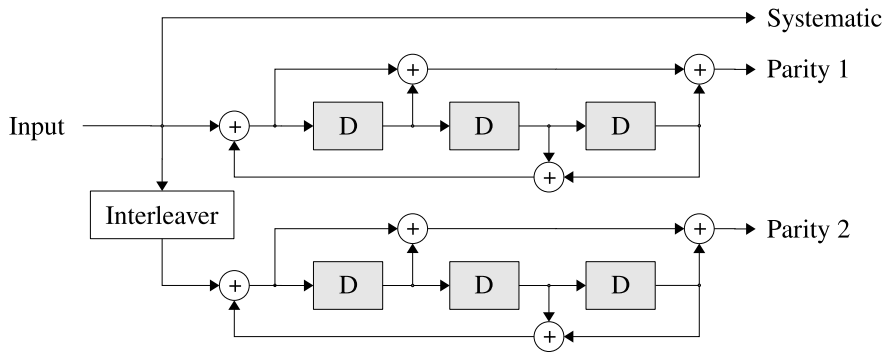

Figure 4.1: Turbo Encoder $(13, 15)_8$

4.2 Generation of BLER data samples with a link level simulator, and BLER curves

The performance of the TC is evaluated using a link level simulator. The simulator comprises the TC, QPSK modulator, AWGN channel and a log-MAP decoder with 8 iterations. The BLER samples at each simulated point $BLER_{sim}$, were obtained by averaging the BLER of 50 sets of 2000 blocks in error. BLER is considered in the region from $8 \cdot 10^{-4}$ to 1. Different levels of SNR, block lengths B and MCS rates R were considered. The selected set of block lengths B and MCS rates R is shown in Table 4.1. Block lengths were selected in such a way that when using a particular puncturing pattern (Appendix A), the amount of information bits and parity bits fits exactly into the selected block length. In the selection of block lengths it was also considered that the LTE standard defines a maximum code block size of 6144 bits [3], thus setting a maximum reference length for using the model in real applications.

BLER data samples, produced by simulations using the link level simulator, are presented in Fig. 4.2. In the figure data samples are connected together to form BLER curves for different combinations of B and R. The generated BLER data samples will be used in the following chapters to construct and optimize the proposed mathematical model.

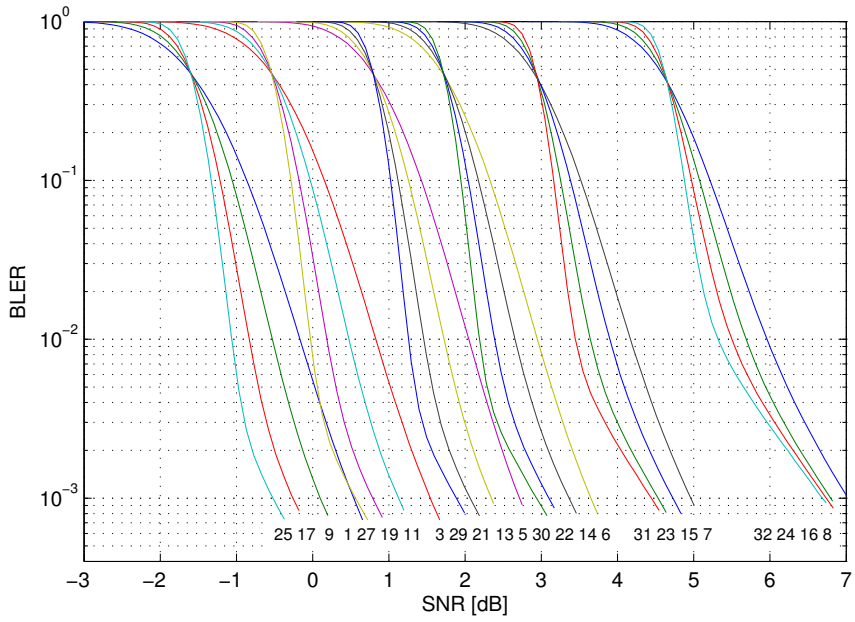

Figure 4.2: BLER performance of turbo code $(13, 15)_8$ for different block lengths and MCS rates, obtained by simulation and averaged over 50 sets of 2000 blocks in error. The system comprises: QPSK modulator, AWGN channel, random interleaver, log-MAP decoder with 8 iterations and puncturing patterns described in Appendix A. Numbers at the bottom of the curves indicate a curve number used for reference. Refer to Table 4.1 for details of block lengths and MCS rates.

Table 4.1: Set of block lengths B and MCS rates R used in the simulations

Curve number	Approx. MCS rate	MCS rate R	Block length B	Inf. bits n
1	2/3	0.6549020	510	167
2	8/11	0.7162426	511	183
3	4/5	0.7882352	510	201
4	8/9	0.8789062	512	225
5	1/1	0.9882812	512	253
6	8/7	1.1311154	511	289
7	4/3	1.3203124	512	338
8	8/5	1.5890410	511	406
9	2/3	0.6608016	1023	338
10	8/11	0.7214076	1023	369
11	4/5	0.7937438	1023	406
12	8/9	0.8836754	1023	452
13	1/1	0.9941406	1024	509
14	8/7	1.1378300	1023	582
15	4/3	1.3281250	1024	680
16	8/5	1.5933528	1023	815
17	2/3	0.6637342	2046	679
18	8/11	0.7243402	2046	741
19	4/5	0.7968750	2048	816
20	8/9	0.8861748	2047	907
21	1/1	0.9970704	2048	1021
22	8/7	1.1402052	2047	1167
23	4/3	1.3300782	2048	1362
24	8/5	1.5966796	2048	1635
25	2/3	0.6652014	4095	1362
26	8/11	0.7257632	4095	1486
27	4/5	0.7985348	4095	1635
28	8/9	0.8876404	4094	1817
29	1/1	0.9985352	4096	2045
30	8/7	1.1413920	4095	2337
31	4/3	1.3320312	4096	2728
32	8/5	1.5981446	4096	3273

4.3 Link Level simulator details

A Link level simulator was used to study the performance of the turbo code $(13, 15)_8$ in an AWGN channel. The Link level simulator consists of a transmitter, channel and receiver operating in baseband. More specifically the simulator contains the modules depicted in Fig. 4.3. The steps carried out in each one of the modules shown on the figure are explained in the sub-sections below.

4.3.1 Initial settings

Before initiating a simulation, the link level simulator (Fig. 4.3) must be configured with the proper initial parameters. The initial parameters comprise target SNR, block length, code rate, modulation type, and puncturing pattern. The values of SNR used in the simulations were spaced by 0.1 dB starting from a SNR corresponding to an averaged BLER equal[8] to 1 until a SNR corresponding to a BLER of approximately $8 \cdot 10^{-4}$. Block lengths and MCS rates used to generate the BLER data samples are listed on Table 4.1.

[8] An averaged BLER equal to 1 is the BLER resulting in value 1 from averaging the BLERs obtained from 50 simulations for a given SNR, B and R. In addition the BLER obtained from each simulation is

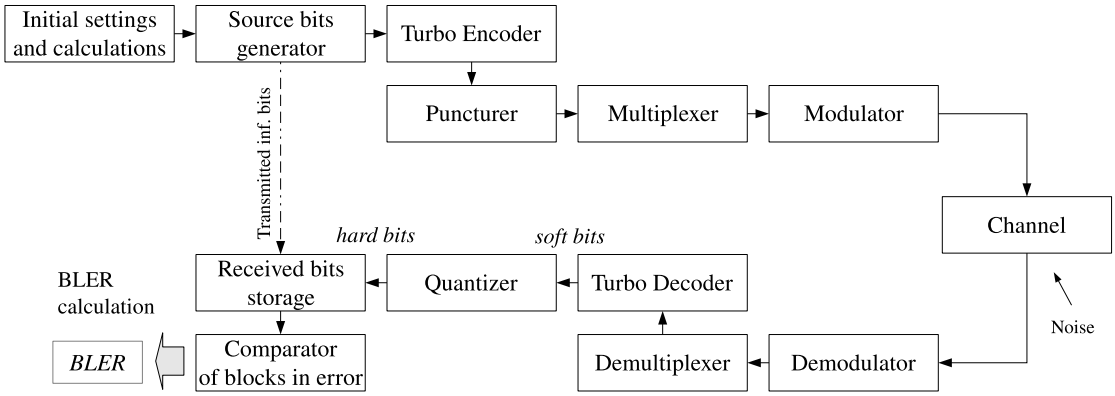

Figure 4.3: Basic structure of the Link Level simulator.

The simulations were performed by repeating a Monte Carlo experiment for the given initial settings, therefore it was necessary to specify also the number of experiments, or the stopping criterion, in the initial settings. In this thesis the author used 2000 blocks in error as stopping criterion in each simulation. Each simulation was repeated 50 times for the same values of SNR, B and R, and subsequently the collection of the resulting 50 BLERs was averaged and stored as BLER data samples for later processing.

4.3.2 Calculation of the final block length, number of information bits and rate

After configuring the initial parameters to be input to the simulator, a few preliminary calculations are necessary before initiating the simulation. Since the block length and number of information bits are quantities that belong to the set of natural numbers, a proper criterion must be devised to maintain a consistent size of the final block length and number of information bits in the link level simulator, and at the same time maintain the target code rate.

In the Link Level simulator used in this thesis, the puncturing method selected is puncturing with puncturing masks; a consequence of using this method results, in some cases, in keeping the proposed block length, B, unchanged, or resizing it to a length $B-1$ or $B-2$. Block lengths are selected in such a way that when using a particular puncturing pattern (Appendix A), the amount of systematic bits and parity bits fits exactly into the selected, or otherwise resized, block length. The reason behind the resizing of the block length lies on the fact that the puncturing is performed with puncturing masks of a definite length and definite pattern. Then in order to approximate the target code rate, 1 or 2 bits in the block may remain unused; on the contrary, using these 1 or 2 bits in the block will result in adding more systematic or parity bits and hence deviating from the target code rate.

In practical systems the block length may be fixed and the approach presented here may be not valid, however in this thesis the approach adopted is justified by arguing that the

averaged in 2000 blocks in error.

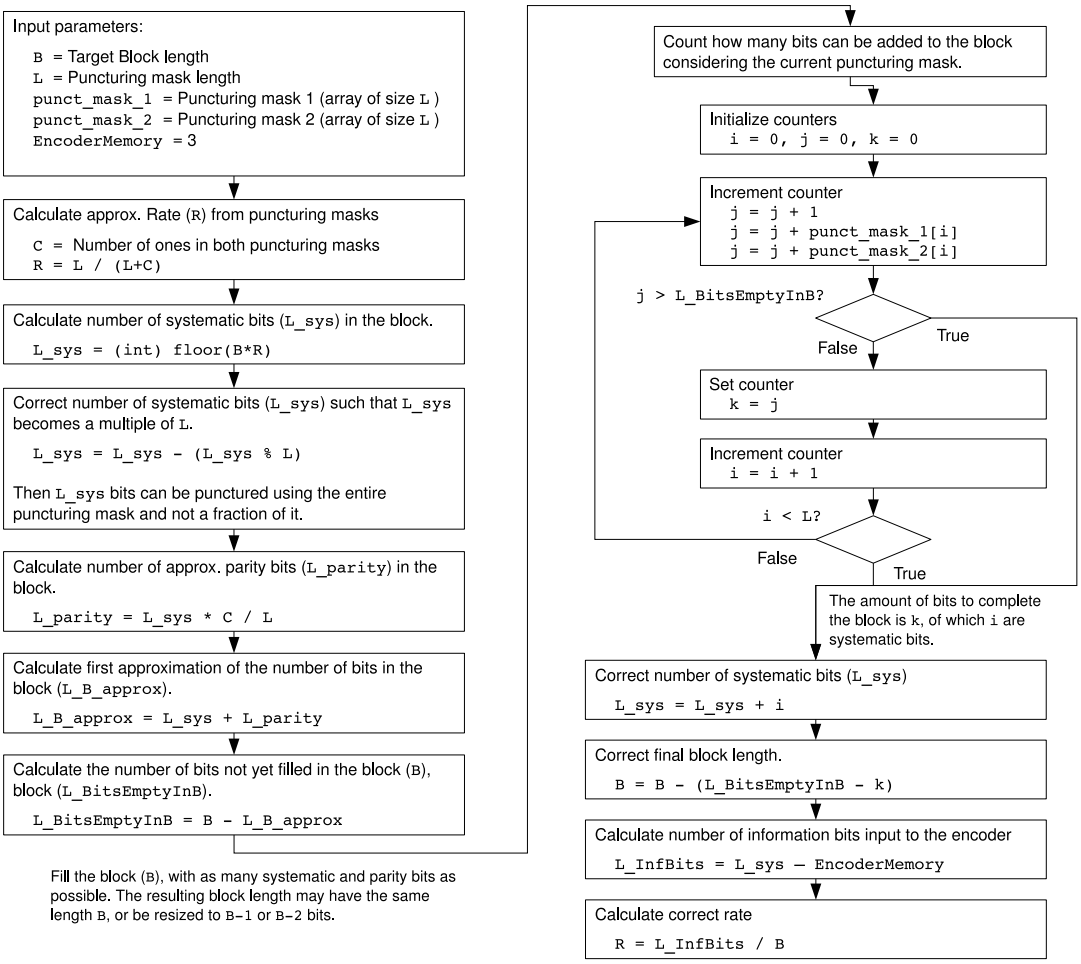

Figure 4.4: Algorithm adopted in the link level simulator to recalculate the final block length, number of information bits and rate in function of the target block length and rate. The binary operator % represents the modulo operation.

set of values B and R will be used as samples to construct a continuous function, being not imperative to have all the samples with a determined fixed value.

Fig. 4.4 shows the algorithm adopted in the link level simulator to re-calculate the final block length, number of information bits and rate in function of the target block length and rate.

In addition, it is not required to make the block length a multiple of M (the number of bits per symbol in the modulation scheme adopted). This decision aims to have a consistent criterion when working with different block lengths. The actual size of the block length in the channel of the link level simulator is the target (not corrected) block length (Fig. 4.4); but at the transmitter and receiver, as well as in the calculation of the rate (equation (4.3)), the corrected block length is used. The model to be created aims to target any block length and rate, regardless of the actual possibility of having all of these implemented in a real system. If instead, the block lengths are forced to be rounded to multiples of M,

the results obtained will be influenced by this rounding function. Therefore it is argued that the criterion adopted is consistent regarding the samples to be used to construct the continuous function of the model.

4.3.3 Generation of information bits, storage of received bits and calculation of BLER

Information bits are generated with a pseudo-random number generator (module *Source bits generator* in Fig. 4.3), and stored in memory. Received bits, after quantization, are also stored in memory (module *Received bits storage*), and then compared bit by bit (module *Comparator of blocks in error*) to the transmitted information bits. If there is a mismatch between one or more of the original information bits and the received bits, then the block is in error and a counter of blocks in error is increased.

The simulation stopping criterion is the number of blocks in error; when the number of blocks in error is 2000, the ratio between the number of blocks in error to the number of transmitted blocks is the averaged BLER for the given SNR, block length and MCS rate. In addition the simulation process was repeated 50 times per combination of SNR, block length, B, and MCS rate, R, and averaged the BLERs obtained.

4.3.4 Turbo Encoder, Turbo Decoder and quantization of bits

Turbo encoder and decoder (modules *Turbo encoder* and *Turbo decoder* in Fig. 4.3) were presented in the previous chapter. The Log-MAP algorithm was used in the turbo decoder with a fixed number of 8 iterations. The output of soft bits from the turbo decoder is quantized to hard bits zeros and ones in the module *Quantizer*.

4.3.5 Puncturing

Puncturing (module *Puncturer* in Fig. 4.3) is performed by using puncturing masks (listed in Appendix A). Puncturing is performed sequentially applying the puncturing masks to the parity bits output from the encoder. As explained in sub-section 4.3.2, the block length is corrected to be consistent with the resulting number of systematic and parity bits after puncturing.

4.3.6 Multiplexer and Demultiplexer

The multiplexer arranges the three streams of parallel bits, systematic, parity 1 and parity 2 bits, into a unique stream of bits. The demultiplexer performs the opposite action, and in addition it is combined with the operation opposite to puncturing, filling with zeros the places where the bits were punctured.

4.3.7 Modulator, Demodulator and Channel

The link level simulator was programmed to use QPSK and 16-QAM modulations. The numeric results used as example throughout the thesis, except were indicated, were generated using QPSK modulation. When 16-QAM modulation is used, the demodulator actually calculates the marginal probabilities of the bits in the symbol. Furthermore, an interleaver is added before the modulator, and a deinterleaver is added after the demodulator to avoid adding the same noise level to consecutive bits within a symbol.

The channel model used in the link level simulator is a simple AWGN channel.

4.4 Performance metric of the model

The quality of fitting, of the mathematical model to be constructed, is evaluated as a relative approximation error. It is calculated by summing the squares of differences between the simulated BLER samples, $BLER_{sim}\left(\gamma_i, B_j, R_l, \right)$, and the BLER obtained by the proposed model, $P_e\left(\gamma_i, B_j, R_l\right)$. The fitting is performed in the logarithmic domain ($\log_{10}(BLER_{sim})$, $\log_{10}(P_e)$). Such fitting allows introducing accuracy that is proportionally fair for samples at different SNR. The fitting metric is,

$$\sum_{i,j,l} \left[\log_{10}\left(BLER_{sim}(\gamma_i,B_j,R_l)\right) - \log_{10}\left(P_e(\gamma_i,B_j,R_l)\right)\right]^2, \tag{4.5}$$

where i, j and l are indexes, indexing a given number of SNR, block lengths and MCS rates, respectively.

5 Proposed model

The mathematical model that approximates the BLER performance of punctured turbo codes will be described by the function $P_e(\gamma, B, R)$. This function must have enough degrees of freedom to model the four-dimensional surface composed by γ, B, R and $BLER$. Models based directly on polynomials or exponential functions are not useful since these may not satisfy the desired characteristics of the function representing the model; these may not have an asymptotic convergence toward $BLER = 1$ at low SNR and may not present an injective function property relating SNR-$BLER$ at constant B and R. Furthermore these models may require several number of terms to approximate the samples with considerable accuracy and can oscillate between the fitting points, presenting non-consistent results when describing the performance of codes for any block length and rate.

For example, the surface depicted in Fig. 5.1 shows an early attempt to model the relationship between SNR, R and $BLER$ at constant B using a superquadric[9] surface.[10] In the figure it can be observed that the surface does not satisfy some of the desired properties; there is no asymptotic convergence to $BLER = 1$ at low SNR, and the relationship between SNR and $BLER$ is non-injective (some SNR values can be mapped to the same $BLER$ while keeping B and R constant).

Given that finding a four dimensional model for the system in hands is not trivial, the modeling task was divided in two parts, creating two partial models; firstly studying the relationship between SNR and $BLER$, secondly studying the relationship between SNR, B and R, and finally combining these models into a complete model relating SNR, B, R and $BLER$. Fig. 5.2 represents the steps carried out to construct the final model.

It is noted that a final model can be created directly without the need of creating partial models, but on the other hand requiring much more intensive computations. The partial models were created to understand the relationship between the variables and then propose a small number of eligible functional forms that are consistent with the relationship observed. In this way the computation effort to find an optimal model is reduced. In addition, having the independent relationship between the variables allowed looking for some notable relationship between these variables, like linearity, or a distinctive pattern that can be expressed with power series, etc. Such notable relationship was not found.

In the sections that follow, first a model relating SNR and $BLER$ without considering B and R is constructed (Section 5.1). Then a model relating SNR, B and R for fixed $BLER$ values is constructed (Section 5.2). Finally, in section 5.3 these models are combined into a complete model relating SNR, B, R and $BLER$.

[9] The general equation of a superquadric is $\left|\frac{x}{A}\right|^r + \left|\frac{y}{B}\right|^s + \left|\frac{z}{C}\right|^t \leq 1$. Where x, y, and z are independent variables; r, s, and t are positive real numbers that determine the main characteristics of the superquadric; and A, B, and C are scaling constants.

[10] The relation between variables in the figure is representative, and it is not directly related to the system under consideration. In the figure $EbN0$ was used instead of SNR. $BLER$ units were changed to $\log_{10}(BLER)$ for convenience in the plotting.

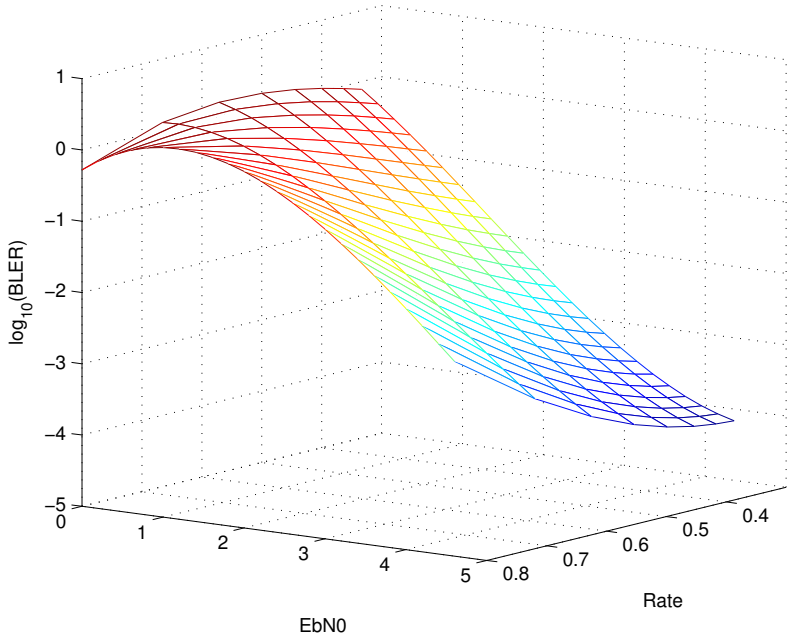

Figure 5.1: Attempt to model the relationship between SNR, R and $BLER$ at constant B using a superquadric function.

5.1 *SNR-BLER* Model

The relationship observed between BLER and SNR (Fig. 4.2) shows an asymptotic curvature towards 1 and 0. A continuous and differentiable function resembling BLER curves is the sigmoid function. In [28] a model based on the sigmoid function was used for approximating BLER curves of convolutional codes. Unfortunately the sigmoid function in the form

$$s_1(x) = \frac{1}{1 + e^{-c_1 x - c_2}} \tag{5.1}$$

is not capable to model the error floor present in turbo codes. A simple modification of the sigmoid

$$s_2(x) = 1 - \frac{1}{1 + e^{-c_1 x - c_2} + 2e^{-c_3 x - c_4}}, \tag{5.2}$$

provides a suitable function for characterizing the error floor of BLER in turbo codes. Furthermore (5.2) is a continuous and differentiable function, presenting an injective relationship between SNR and BLER. In Fig. 5.3, s_1 and s_2 are plotted.

For each combination of block lengths and MCS rates a different BLER curve is observed (Fig. 4.2). The location of a curve can be adjusted by selecting suitable values for the coefficients c_j in (5.2). In general form the relationship between SNR and $BLER$ is modeled by

$$P_e(\gamma) = 1 - \frac{1}{1 + e^{F_1(\gamma)} + 2e^{F_2(\gamma)}}, \tag{5.3}$$

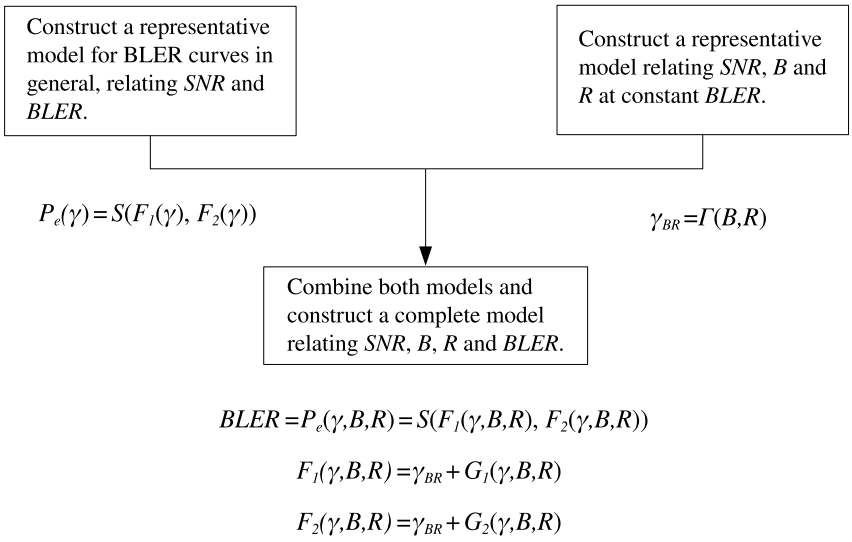

Figure 5.2: Representation of the steps carried out to construct the final model.

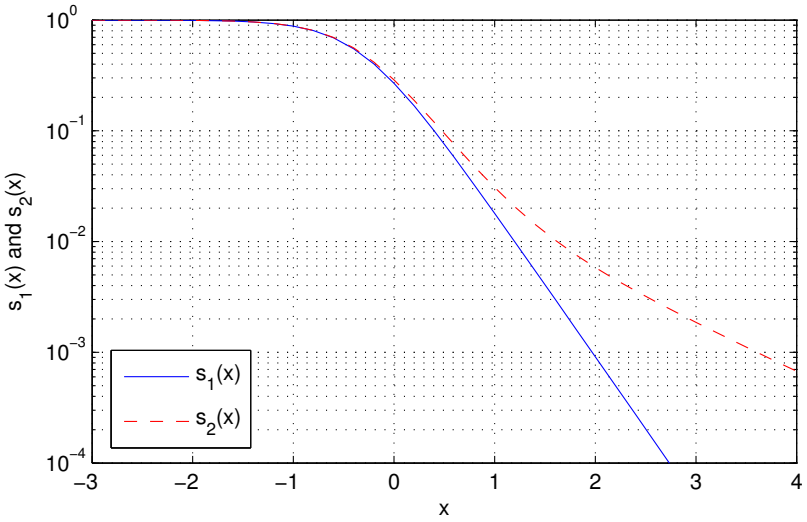

Figure 5.3: Shifted Sigmoid function s_1 and proposed function s_2

where F_1 and F_2 are polynomials or power series in γ that define the position and shape of the BLER curve[11].

In the figure representing the steps carried out to construct the final model, Fig. 5.2, equation 5.3 is expressed as follows,

$$P_e(\gamma) = S(F_1(\gamma), F_2(\gamma)); \tag{5.4}$$

where $S()$ is a modified sigmoid function.

[11] It is noted here that the multiplication by two in the term $2e^{F_2(\gamma)}$ it is not extrictly necesary. The

5.2 *SNR-Block_Length-Rate* Model

In the previous section a model relating SNR and BLER was proposed. In this section a model relating SNR, B and R, considering $BLER$ constant, is proposed. The model is expressed as follows,

$$\gamma_{BR} = \Gamma(B, R), \tag{5.5}$$

where $\Gamma()$ is a function approximating the SNR, γ_{BR}, for a given set of block lengths, B, and MCS rates, R.

In order to find the relationship between γ, B, R, the four dimensional space generated by $BLER$,γ,B,R, is reduced to three dimensions. The reduction from $\mathbb{R}^4$ to $\mathbb{R}^3$ is carried out projecting orthogonally γ, B, R, and setting the BLER, $BLER$, constant. Setting $BLER$ values as a constant can be visualized as making cuts in the BLER curves at fixed $BLER$ values (Fig. 5.4). Then the corresponding SNRs at the cut, γ_{BR}, along with B and R for each BLER curve, form a three-dimensional space. An example of the relationship between γ_{BR}, B and R at fixed $BLER = 8 \cdot 10^{-1}$ and $1 \cdot 10^{-3}$ is shown by the surfaces in Fig. 5.5.

In Fig. 5.5 it can be observed that a plane would be an initial choice for approximating a surface relating SNR, B and R. For a better approximation to this surface a more elaborated function consisting of 19 different terms was considered. The terms proposed were selected as eligible terms by observing the surface to model, and knowing the surfaces that can be constructed with these terms, independently, as part of a function. The main idea is to construct the model relating SNR, B and R by finding an optimal combination of the proposed terms; or to be more precise, by relying on symbolic regression. The proposed function is,

$$\begin{aligned}\Gamma(B,R) \;=\; & a_1 + a_2 B + a_3 R + \\ & + a_4 BR + a_5 B^2 + a_6 R^2 + a_7 B^3 + a_8 R^3 + \\ & + a_9 (B^2 - R^2) + a_{10} BR^2 + a_{11} B^2 R + a_{12} B^2 R^2 + \\ & + a_{13} B * R^3 + a_{14} B^3 R + a_{15} B^{a_{16}} + a_{17} R^{a_{18}} + \\ & + a_{19} B^{a_{20}} R + a_{21} B R^{a_{22}} + a_{23} B^{a_{24}} R^{a_{25}},\end{aligned} \tag{5.6}$$

where a_j, with $j \in \{1, 2, ..., 25\}$, are numeric coefficients. This function contains terms suitable to produce quadrics like elliptic paraboloid and hyperbolic paraboloid, superquadrics, the monkey saddle surface and terms suitable to displace and rotate these surfaces.

multiplication by two is fundamented by two reasons.

First, in the early stages of the research an attempt to express the model in terms of a series was made; and in particular a series that can be related to the code spectrum of the Turbo Code. In this case the multiplication by two states that the term belongs to the second term of a truncated series.

Second, the fitting algorithm used to optimize the model was inestable, not converging to an optimal solution, when it was presented with two identical terms in the function. It was observed that the algorithm ended presenting identical coefficients in the two identical terms when actually it was expected that the degrees of freedom provided by the second term helped to complement those provided by the first term. This behavior was not observed in any other case. Therefore the second exponential term was changed to a similar function, in this case two times an exponential function, to effectively introduce the desired degrees of freedom,

The multiplication by two introduces only a linear difference of $\ln(2)$. Therefore $2e^{F_2(\gamma)}$ can be easily transformed to the form $e^{F_2'(\gamma)}$ by making $F_2'(\gamma) = F_2(\gamma) + \ln(2)$ if it is necessary.

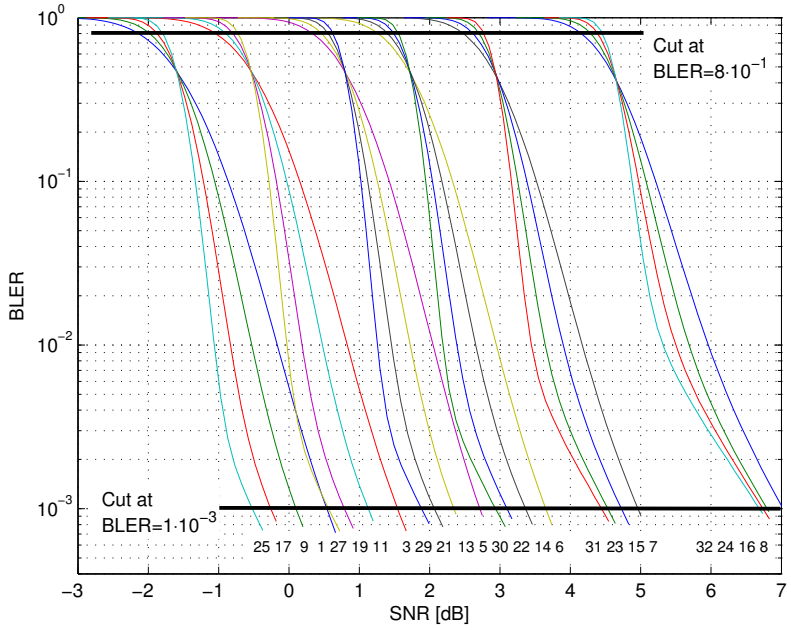

Figure 5.4: Detail of cuts in BLER curves at fixed $BLER = 8 \cdot 10^{-1}$ and $1 \cdot 10^{-3}$.

The terms in equation (5.6) are proposed terms to construct a suitable model approximating the relationship between SNR, B and R. In order to find an optimal function for the model, consisting of a subset of the proposed terms, the collection of terms in (5.6) is interpreted as a set of functional forms that will be selected by relying on symbolic regression. Symbolic regression is usually implemented with genetic algorithms [8]; here, instead, the functional form of the approximating function is selected by performing an exhaustive search among all the combinations of the terms. For this purpose equation (5.6) is rewritten as,

$$\begin{aligned}\Gamma(B,R) \;=\; & u_1 a_1 + u_2 a_2 B + u_3 a_3 R + \\ & + u_4 a_4 BR + u_5 a_5 B^2 + u_6 a_6 R^2 + u_7 a_7 B^3 + u_8 a_8 R^3 + \\ & + u_9 a_9 (B^2 - R^2) + u_{10} a_{10} B R^2 + u_{11} a_{11} B^2 R + u_{12} a_{12} B^2 R^2 + \\ & + u_{13} a_{13} B * R^3 + u_{14} a_{14} B^3 R + u_{15} a_{15} B^{a_{16}} + u_{16} a_{17} R^{a_{18}} + \\ & + u_{17} a_{19} B^{a_{20}} R + u_{18} a_{21} B R^{a_{22}} + u_{19} a_{23} B^{a_{24}} R^{a_{25}};\end{aligned} \tag{5.7}$$

where $u_i = \{0, 1\}$, with $i \in \{1, 2, ..., 19\}$, is a coefficient used to indicate the presence or absence of a term in the functional form proposed for modeling the desired relationship. Hereafter a term is defined as "active" when the term is present in the functional form describing the model ($u_i = 1$), and as "inactive" when the term is not present ($u_i = 0$).

The first three terms in equation (5.7) are considered always active ($u_1 = u_2 = u_3 = 1$). The remaining 16 terms (u_i with $i = 4...19$) are selected as eligible active terms, taking four terms at a time, such that the resulting model is composed by only seven terms; the first three fixed and the other four selected by symbolic regression. Deciding that the first three terms are directly part of the model is justified by the nature of these terms,

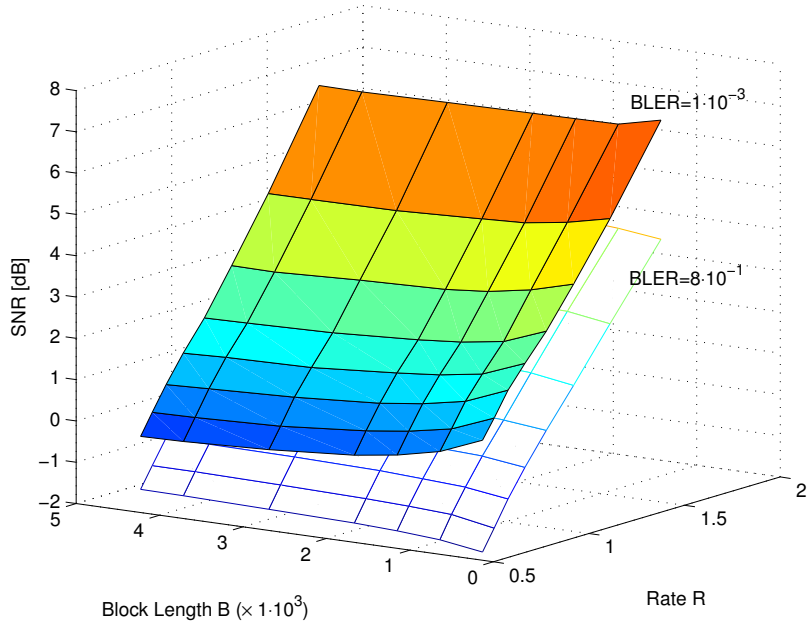

Figure 5.5: Relationship between SNR, B and R at fixed $BLER = 8 \cdot 10^{-1}$ and $1 \cdot 10^{-3}$

positioning a plane and more elaborated surfaces. The remaining 16 terms are taken in groups of four and give a total of $\binom{16}{4}$ = 1820 combinations to be explored by exhaustive search. The decision to construct the model with only seven terms is justified by the satisfactory degree of accuracy obtained in early models, using this amount of terms, and considering the constraints imposed by computing time.

The 1820 combinations of functional forms were numerically fitted in a least-square sense to the data samples of SNR, B and R with a non-linear curve-fitting algorithm [29] [30]. After the fitting process, the combinations of terms resulting in the smaller aggregated fitting metric were selected for the final model.

The fitting process for the 1820 combinations of functional forms was repeated 26 times, corresponding to 26 cuts of the BLER curves at constant BLERs between $1 \cdot 10^{-3}$ and $8 \cdot 10^{-1}$. For each cut a set of samples SNR, B, R was input to the non-linear fitting algorithm. This set of samples was obtained from the samples generated with the link level simulator. Actually, since the BLER samples were generated for a given SNR, notable values of BLER, proposed for the BLER cuts, like $8 \cdot 10^{-1}$, $7 \cdot 10^{-1}$, etc. were not available. To find the corresponding value of SNR for a given BLER cut, local interpolation between BLER samples was carried out. The corresponding SNRs for the given BLER cuts were stored along with the B and R values and input to the non-linear fitting algorithm.

Then non-linear fitting was carried out for each one of the BLER cuts, and considering several initial coefficients. The best fitting for each cut, measured by a fitting metric, was stored after completing the fitting process. After fitting the 26 BLER cuts an aggregated fitting metric was computed, calculated by adding the fitting metrics of the 26 BLER cuts.

The performance of each combination of terms was measured by means of the aggregated fitting metric. Table 5.1 lists the first 10 better functional forms describing the model in terms of the aggregated fitting metric. In the table, the pattern of active terms indicates which terms are active in equation (5.7). The same data is represented graphically in Fig. 5.6. Based on this analysis, it can be observed that in addition to the first three terms, terms 6, 8, 16 and 19 are in average the optimal ones for the proposed model; then a model relating SNR, B and R can be expressed by,

$$\Gamma(B,R) = a_1 + a_2 B + a_3 R + a_6 R^2 + a_8 R^3 + a_{17} R^{a_{18}} + a_{23} B^{a_{24}} R^{a_{25}}. \tag{5.8}$$

Table 5.1: Best 10 patterns of active terms and respective aggregated fitting metric

Pattern number	Pattern of active terms u_i $(i = 1..19)$	Fitting metric
1	11100 10100 00001 0001	0.071097
2	11100 10100 10000 0001	0.071046
3	11100 10000 00000 1011	0.070411
4	11100 00100 00000 1011	0.070187
5	11110 10100 00000 0001	0.069994
6	11100 10100 00000 0101	0.069691
7	11100 10100 00100 0001	0.068065
8	11100 00101 00000 1001	0.066601
9	11100 00100 00100 1001	0.066123
10	11100 10100 00000 1001	0.063945

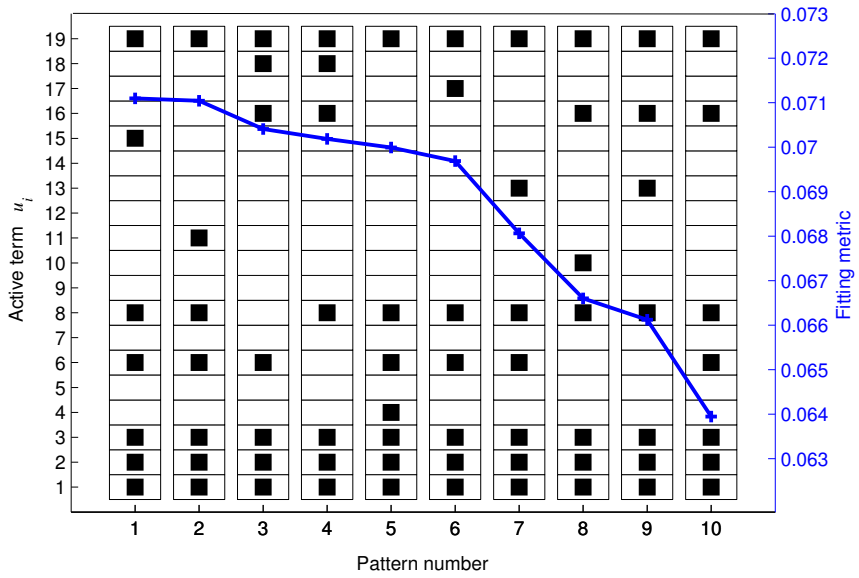

Figure 5.6: Best 10 patterns of active terms and respective aggregated fitting metric

Discussion: It can be argued that the criteria used to derive the best combination of terms among the 1820 combinations may be not optimal, in the sense

that the aggregation of independent fittings determines the best model to be adopted.

The fitting carried out for each BLER cut was independent of other BLER cuts (always considering the same combination of functional forms), therefore resulting each case in a set of optimized coefficients independent of other BLER cuts. In the process carried out, the fitting metric of each BLER cut was aggregated into an aggregated fitting metric by adding the fitting metrics of all the BLER cuts. This process was repeated for the 1820 combinations of terms; finally, the combination of terms with smaller aggregated fitting metric was selected as the best functional form.

In the actual BLER curves, BLER is not a fixed value (i.e. a cut at constant BLER like it was considered here), but it is a dependent variable; therefore a set of coefficients for a particular BLER cut may not necessarily be optimal for another BLER cut. However, when incorporating the BLER dimension into the final and complete model, additional degrees of freedom will help to overcome the constraint of constant BLER imposed by the SNR, B, R model. The additional degrees of freedom may be or may be not enough to justify that the approach followed here (aggregating the metrics) was optimal. Since the surfaces in the SNR, B, R model vary in each BLER cut, an optimized approach can rely on targeting a specific BLER value and produce a functional form for that specific BLER value. Also a range of BLER values can be targeted; in this case using a weighted fitting metric for the BLERs considered. In this thesis the approach to find the set of functional forms is justified by considering that here there is no specific target BLER but the goal is to approximate all the BLERs between $8 \cdot 10^{-1}$ and $1 \cdot 10^{-3}$ with a functional form than in average is fair to the given range of BLERs.

5.3 Complete Model

In the previous sections two models were constructed. First, a model relating SNR and $BLER$ without considering B and R (Section 5.1), and second, a model relating SNR, B and R for fixed $BLER$ values (Section 5.2). In this section these two models are combined into a complete model relating SNR, B, R and $BLER$.

The desired model

$$BLER = P_e(\gamma, B, R) \tag{5.9}$$

is first constructed by generalizing Equation (5.3) for B and R,

$$P_e(\gamma, B, R) = 1 - \frac{1}{1 + e^{F_1(\gamma,B,R)} + 2e^{F_2(\gamma,B,R)}}, \tag{5.10}$$

where F_1 and F_2 are composed from a known reference SNR γ_{BR}, for a given B and R at a constant $BLER$, and adding a shift in the SNR axis with a function $G()$ that produces the BLER corresponding to the target SNR γ, for the given B and R;

$$F_1(\gamma, B, R) = \gamma_{BR} + G_1(\gamma, B, R), \tag{5.11}$$

$$F_2(\gamma, B, R) = \gamma_{BR} + G_2(\gamma, B, R). \tag{5.12}$$

F_1 and F_2 can be thought simply as the set of suitable parameters to put in place and accommodate the shape of a BLER curve as explained in Section 5.1. The term γ_{BR} contributes to produce the necessary positioning along the SNR axis for a given B and R at certain constant $BLER$ (see dot marked in Fig. 5.7). But what the model should produce is the $BLER$ for the target SNR γ. Therefore in order to obtain the desired $BLER$, it is necessary to apply a correction to the model starting from the coordinates γ_{BR} and $P_e(\gamma_{BR}, B, R)$ in the SNR-BLER plot (Fig. 5.7). This correction (represented as $G(\gamma, B, R)$ in the figure) is a displacement in the SNR axis such that matches the target SNR γ and produces the corresponding BLER, $P_e(\gamma, B, R)$, for the given B and R.

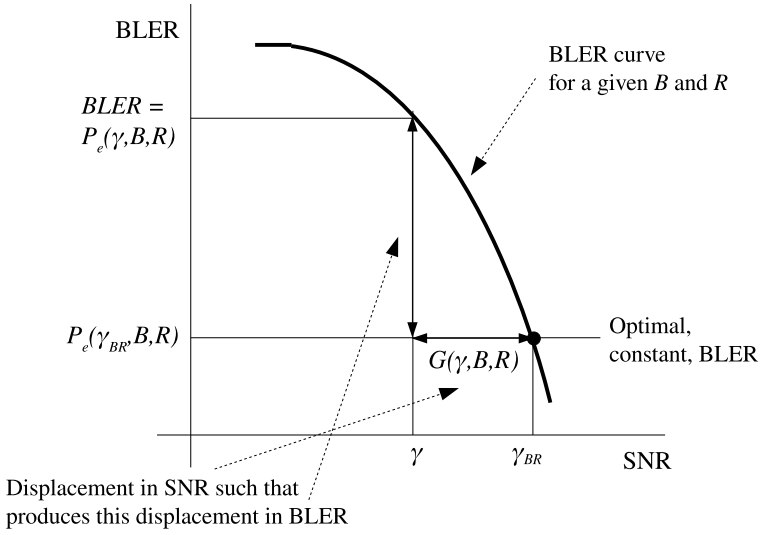

Figure 5.7: Visualization of shift in the SNR axis with a function $G()$ that produces the BLER corresponding to the target SNR γ, for the given B and R, and starting from a known reference SNR γ_{BR}, for a given B and R at a constant $BLER$.

Functions G_1 and G_2 complement the model proposed in Section 5.2, where $BLER$ was set constant to produce BLER cuts in the BLER curves. Functions G_1 and G_2 can be thought also as the necessary correction in BLER units, applied to the constant BLER, such that there is a displacement along the BLER axis proportional to the target SNR γ, and for the given B and R, releasing the constraint (of fixed BLER) imposed in γ_{BR}. In this case, since $BLER$ is the dependent variable in the model, in G_1 and G_2 the correction along the BLER axis is expressed in terms of SNR for the given SNR, B and R.

In the complete model the constant BLER (BLER cut) will be optimized to a BLER level resulting from the optimal fit of the model to the variables γ, B and R.

Introducing the model constructed in Section 5.2, F_1 and F_2 are rewritten as,

$$F_1(\gamma, B, R) = \Gamma(B, R) + G_1(\gamma, B, R), \tag{5.13}$$

$$F_2(\gamma, B, R) = \Gamma(B, R) + G_2(\gamma, B, R). \tag{5.14}$$

The proposed functions G_1 and G_2 contain a simple relationship between the variables;

$$G_1(\gamma, B, R) = q_1\gamma + q_2\gamma B + q_3\gamma R + q_4 BR, \tag{5.15}$$

$$G_2(\gamma, B, R) = q_5\gamma + q_6\gamma B + q_7\gamma R + q_8\gamma BR, \tag{5.16}$$

where q_i are coefficients to be determined. The functions were selected by their simplicity, allowing making a linear correction in each dimension and by observing that terms to the second and third power, other that the proposed, did not produced significant improvements in the final model.

The last terms proposed in (5.15) and (5.16) are different just to create two different final functions F_1 and F_2. The results found in [31] suggest that the term q_4BR is important for the model, but not two of them; i.e. in the model of [31] this term is not present at the same time in F_1 and F_2 when the number of terms in the model is limited to a reduced number of terms. Furthermore, having exactly the same functions F_1 and F_2 will lead later to "symmetric" results, in the sense that, when the number of terms in use is reduced, there will be a complementary pair of functional forms $F_1 = F_1$, $F_2 = F_2$, and $F_1 = F_2$, $F_2 = F_1$, leading to the same optimal model[12]. For this reason a variation of the last term in the form $q_8\gamma BR$ was proposed. If the term $q_8\gamma BR$ happens to be significant for the model, it will be present in the final function describing the model.

The final proposed model is constructed by replacing equations (5.8), (5.15) and (5.16) into (5.13) and (5.14), rearranging the terms and relabeling the coefficients. The proposed functions for the model are;

$$\begin{aligned} F_1(\gamma, B, R) \;=\; & k_1 + k_2\gamma + k_3 R + \\ & + k_4 B + k_5\gamma B + k_6\gamma R + k_7 BR + \\ & + k_8 R^2 + k_9 R^3 + k_{10} R^{k_{11}} + k_{12} B^{k_{13}} R^{k_{14}}, \end{aligned} \tag{5.17}$$

$$\begin{aligned} F_2(\gamma, B, R) \;=\; & k_{15} + k_{16}\gamma + k_{17} R + \\ & + k_{18} B + k_{19}\gamma B + k_{20}\gamma R + k_{21}\gamma BR + \\ & + k_{22} R^2 + k_{23} R^3 + k_{24} R^{k_{25}} + k_{26} B^{k_{27}} R^{k_{28}}, \end{aligned} \tag{5.18}$$

where k_j, with $j \in \{1, 2, ..., 28\}$, are numeric coefficients.

The terms in equations (5.17) and (5.18) are the proposed terms for the final and complete model approximating the relationship between SNR, B, R and $BLER$. In the next chapter an optimized version of this model, with a reduced set of terms, will be derived. For this purpose an additional variable u_i is introduced in (5.17) and (5.18), and these equations are rewritten as follows;

[12] Lets remember that F_1 and F_2 are indeed interchangeable despite of having two different exponential functions, namely $e^{F_1(\gamma,B,R)}$ and $2e^{F_2(\gamma,B,R)}$ in (5.10), by just introducing a linear difference of $\ln(2)$ in F_1 or F_2. See footnote 11 for additional comments.

$$
\begin{aligned}
F_1(\gamma, B, R) \;=\; & u_1 k_1 + u_2 k_2 \gamma + u_3 k_3 R + \\
& + u_4 k_4 B + u_5 k_5 \gamma B + u_6 k_6 \gamma R + u_7 k_7 B R + \\
& + u_8 k_8 R^2 + u_9 k_9 R^3 + u_{10} k_{10} R^{k_{11}} + u_{11} k_{12} B^{k_{13}} R^{k_{14}},
\end{aligned} \tag{5.19}
$$

$$
\begin{aligned}
F_2(\gamma, B, R) \;=\; & u_{12} k_{15} + u_{13} k_{16} \gamma + u_{14} k_{17} R + \\
& + u_{15} k_{18} B + u_{16} k_{19} \gamma B + u_{17} k_{20} \gamma R + u_{18} k_{21} \gamma B R + \\
& + u_{19} k_{22} R^2 + u_{20} k_{23} R^3 + u_{21} k_{24} R^{k_{25}} + u_{22} k_{26} B^{k_{27}} R^{k_{28}},
\end{aligned} \tag{5.20}
$$

where $u_i = \{0, 1\}$, with $i \in \{1, 2, ..., 22\}$, is a coefficient used to indicate the presence or absence of a term (active terms) in the functional form proposed for the model.

5.4 Summary of the model

The final and complete model is summarized on Table 5.2.

Table 5.2: Summary of the proposed model

$$P_e(\gamma, B, R) = 1 - \frac{1}{1 + e^{F_1(\gamma,B,R)} + 2e^{F_2(\gamma,B,R)}} \tag{5.21}$$

$$\begin{aligned} F_1(\gamma, B, R) \;=\; & u_1k_1 + u_2k_2\gamma + u_3k_3R + \\ & + u_4k_4B + u_5k_5\gamma B + u_6k_6\gamma R + u_7k_7BR + \\ & + u_8k_8R^2 + u_9k_9R^3 + u_{10}k_{10}R^{k_{11}} + u_{11}k_{12}B^{k_{13}}R^{k_{14}} \end{aligned} \tag{5.22}$$

$$\begin{aligned} F_2(\gamma, B, R) \;=\; & u_{12}k_{15} + u_{13}k_{16}\gamma + u_{14}k_{17}R + \\ & + u_{15}k_{18}B + u_{16}k_{19}\gamma B + u_{17}k_{20}\gamma R + u_{18}k_{21}\gamma BR + \\ & + u_{19}k_{22}R^2 + u_{20}k_{23}R^3 + u_{21}k_{24}R^{k_{25}} + u_{22}k_{26}B^{k_{27}}R^{k_{28}} \end{aligned} \tag{5.23}$$

where,

$P_e(\gamma, B, R)$	is the BLER for a given SNR, block length and MCS rate;
γ	is the SNR;
B	is the block length;
R	is the MCS rate;
$u_i = \{0, 1\}$	is a coefficient used to indicate the presence or absence of a term, with $i \in \{1, 2, ..., 22\}$, when $u_i = 0$ the term is inactive, when $u_i = 1$ the term is active;
k_j	are the coefficients for the active terms, with $j \in \{1, 2, ..., 28\}$, and $k_j = 0$ for inactive terms.

6 Optimization of the model

In the previous chapter a function describing the model was constructed. The function was summarized in equations (5.2), (5.22), and (5.23). Equations (5.22) and (5.23) have in total 22 terms that were selected by analyzing the relationship between $BLER$, SNR, B and R. This chapter presents the steps proposed to identify which of these terms are the most crucial for approximating the simulated SNR-BLER samples when the model contains a reduced set of the terms. Furthermore the numerical values of the coefficients for different functions describing the model with different number of terms are calculated.

6.1 Optimization process

In the proposed function for the model not all the terms are required for approximating the BLER curves. A suitable approximation can be found also if only few terms in Eq. (5.22) and (5.23) are used. In this context the following questions arise: which combination of the terms is the most suitable for approximating the SNR-BLER samples, and what performance can be reached with different amount of terms? These questions are answered by performing an exhaustive search over all combinations of the terms and measuring the accuracy of the model in terms of a fitting metric.

In order to find the optimal subset of terms in the proposed function for the model, the collection of terms in (5.22) and (5.23) are interpreted as a set of functional forms that will be selected by relying on symbolic regression. Symbolic regression is usually implemented with genetic algorithms [8]; here, instead, the functional forms of the approximating function were selected by performing an exhaustive search among the 2^{16} possible combinations of terms u_4 to u_{11} and u_{15} to u_{22}. Terms u_1, u_2, u_3, u_{12}, u_{13} and u_{14} are always present in the model, due that these are considered essential for positioning the curves. Hereafter a term is defined as "active" when the term is present in the functional form describing the model.

During the exhaustive search the values of the coefficients k_j that minimize a fitting metric for each combination of terms are calculated. To find the coefficients, the model was fitted in a least-square sense to the collection of BLER data samples $BLER_{sim}(\gamma, B, R)$ by using a non-linear curve-fitting algorithm [29] [30]. The optimization is done by minimizing Eq. (4.5). The non-linear optimization process could end up in a local minimum. For avoiding these minimums different sets of initial coefficients k_j were considered in each fitting iteration.

The algorithm used to find an optimized function for the model comprises the steps depicted in Fig. 6.1. In the figure a "reference step" was introduced as reference for the explanations that follow. In the first step in the algorithm a collection of SNR-BLER samples is loaded into memory. These samples were obtained previously by means of a link level simulator, and for different values of SNR, B and R. In the second step the initial number of active terms under consideration in the model, *NActiveTerms*, is initialized. Also a counter, *NActiveTermsIterCount*, is initialized. The counter is used later for indexing the best obtained fitting metric for each one of the *NActiveTerms* terms considered in

the model.

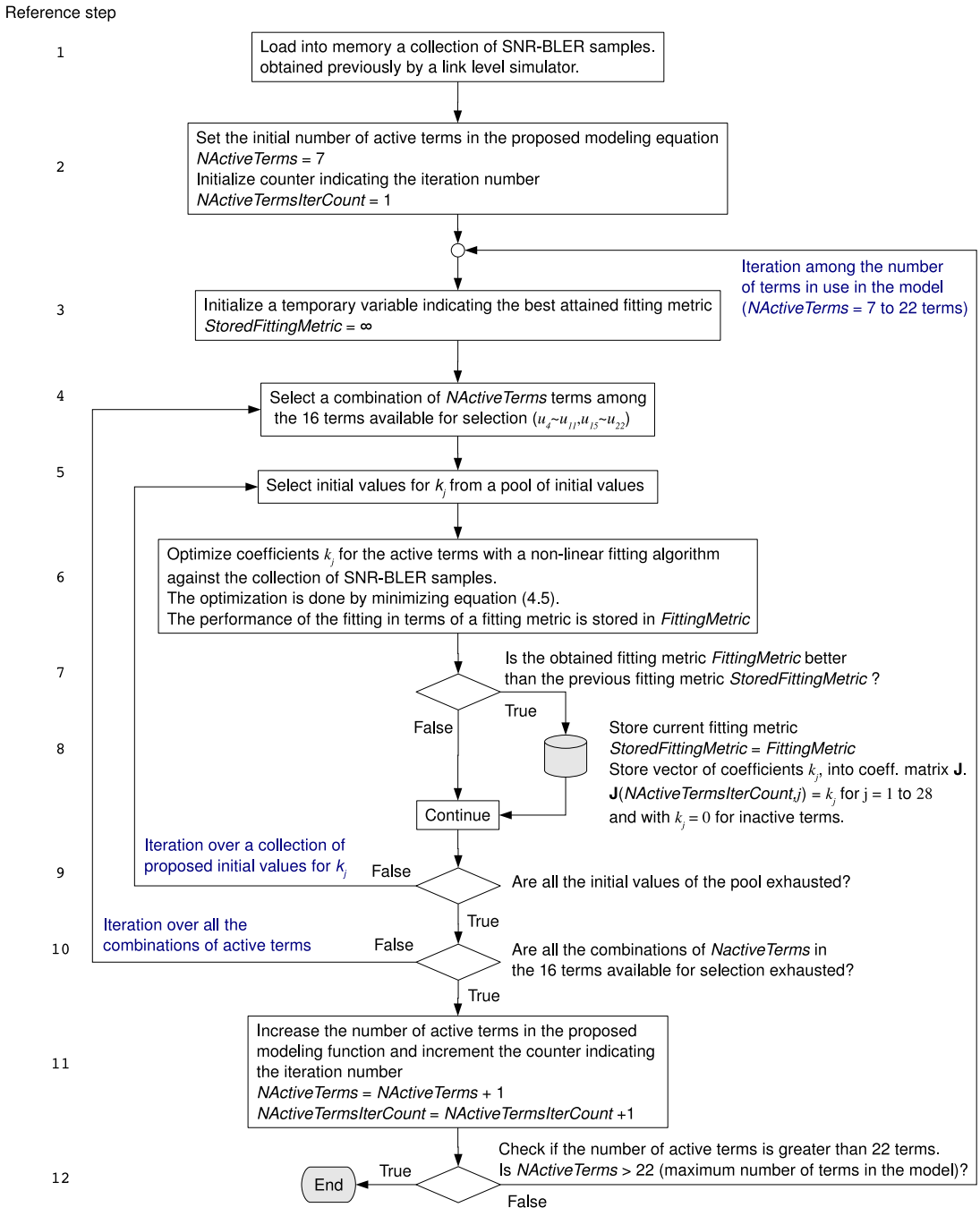

Figure 6.1: Optimization process

From step 2 onwards the algorithm iterates in three loops. The first loop is in charge of setting the total number of active terms in the model (variable *NActiveTerms*). The second loop iterates over all the possible combinations of the 16 eligible terms taken in groups of *NActiveTerms*, ${}^{16}\mathrm{C}_{NActiveTerms}$. The third loop iterates over a pool of proposed initial values for the coefficients k_j.

In step 3 the variable *StoredFittingMetric* is initialized to infinite. This variable is used to store the best attained fitting metric output by the fitting algorithm. In step 4 a loop is initiated, iterating among each one of the ${}^{16}\mathrm{C}_{NActiveTerms}$ combinations of terms. In each iteration one combination of terms is selected, and in step 10 the loop is closed. If all the combinations were exhausted the loop ends and the number of *NActiveTerms* is increased in step 11. In step 12 the number of active terms *NActiveTerms* is verified. If *NActiveTerms* is greater than 22 (maximum number of terms in the proposed model) the algorithm ends, otherwise the execution continues iterating again all the process for the new *NActiveTerms* terms active in the model.

In step 5 the initial values for the coefficients k_j are selected from a pool of initial values. Then the optimization of coefficients k_j for the given combination of *NActiveTerms* terms and initial values is performed in step 6. The optimization is performed with a non-linear fitting algorithm against the collection of SNR-BLER samples. The fitting algorithm returns a fitting metric (*FittingMetric*) along with the optimized values of coefficients k_j. Then in step 7 the fitting metric is compared against a previously stored fitting metric *StoredFittingMetric*; if the obtained fitting metric is better than the previous one, the metric is stored in *StoredFittingMetric* and the coefficients k_j are stored into the coefficients matrix **J**, indexed by the counter *NActiveTermsIterCount* (step 8). Finally, in step 9 it is checked whether all the initial values k_j of the pool are exhausted. If there are more entries in the pool the algorithm iterates again over the fitting with new initial values, otherwise the execution flow continues outside the loop.

6.2 Numerical illustration and evaluation of the model

This section illustrates how the proposed model approximates the simulated SNR-BLER data samples, generated with the link level simulator and settings mentioned in chapter 4.

The algorithm presented in the previous section outputs the matrix **J**. This matrix contains the best combination of terms when the model is composed of 7 to 22 terms, and the corresponding value of the coefficients k_j. The best combination of terms found by the algorithm, and for different number of terms in use, is represented in Fig. 6.2. In the figure it can be observed that some terms are more significant than others when the number of active terms change. By increasing the amount of available terms it can be observed which terms are more significant from an approximation point of view. For example, in addition to the fixed terms 1, 2, 3, 12, 13 and 14, it can be observed that term 11 ($k_{12}B^{k_{13}}R^{k_{14}}$), is always present, irrespective of the number of terms in use.

Furthermore it can be observed that term 7 (k_7BR) becomes inactive when the model switches from 14 to 15 active terms, appearing active at that time the term 18 ($k_{21}\gamma BR$). It is interesting to note that the additional degree of freedom given by γ in term 18 appears for a high number of terms; when there are additional degrees of freedom from other terms. Term 7 (k_7BR) is present at the same time than term 11 ($k_{12}B^{k_{13}}R^{k_{14}}$), from 9 to 14 active terms, suggesting that despite of term 11 having the freedom to use any power, it is not enough to provide the necessary shape and displacement to the surface approximating the relationship between SNR, B and R.

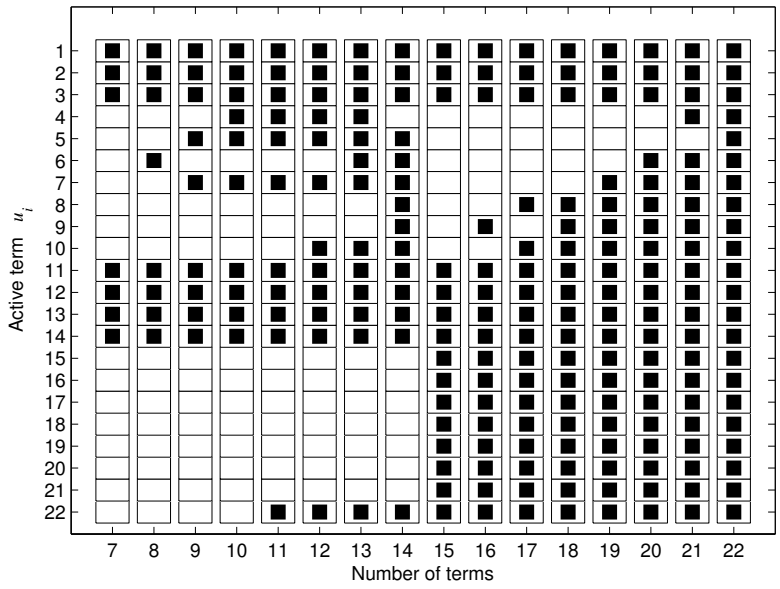

Figure 6.2: Best combination of terms against the number of terms in use

The degree of approximation attained by the model, measured by the fitting metric, against the number of terms in use can be visualized in Fig. 6.3. In the figure it can be observed that, gradually, increasing the amount of terms, the model becomes more accurate at the expense of increasing its complexity. It can be observed also that the fitting metric of the model composed of 22 terms is worse than the metric when using less number of terms; this result is attributed to overfitting.

Table 6.1 lists the values of coefficient k_j for the model when it is composed of 7 to 18 terms. Note that coefficients k_j in the table were obtained using B = B/1000, merely to increase stability in the fitting algorithm. Coefficients k_j with more precision are listed in Appendix B.

In Fig. 6.3 it can be observed that when the amount of terms in the model is large, the increment in accuracy by adding more terms is relatively less significant compared to the increments in accuracy when there were less number of terms in the model. By evaluating the desired degree of accuracy, traded off by the complexity of the model, it is possible to decide how many terms the model should have. In the examples that follow, the accuracy provided by the model when it is composed of 7, 10 and 16 terms is studied. In these cases accuracy is measured locally, in contrast of the aggregated accuracy measured by the fitting metric, by calculating the relative error between sample and modeled points,

$$\text{Relative error} = \frac{|BLER_{sim} - P_e|}{BLER_{sim}} \times 100 \tag{6.1}$$

The accuracy of the model when it is composed of 7 terms is illustrated in Fig. 6.4. In the upper part of the figure the continuous lines are data samples ($BLER_{sim}$) and dashed lines were obtained by using the proposed model (P_e). In the lower part of the figure it

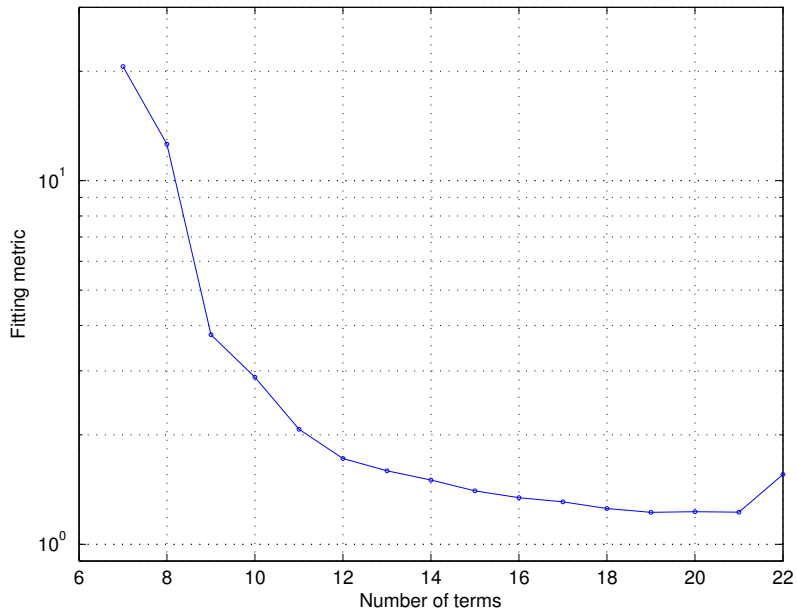

Figure 6.3: Fitting performance against the number of terms in use

can be observed that this model provides an approximation with a relative error of more than 100 % in certain cases.

The accuracy of the model when it is composed of 10 terms is illustrated in Fig. 6.5. Here it can be observed that in this case the model offers a better approximation of the BLER curves, when compared to the model composed of 7 terms, causing in the worst case a relative error of 54%. A model composed of 16 terms offers a better approximation (Fig. 6.6), but at the cost of increasing the complexity of the model.

Considering the premise on having a simple model, and considering also that several sources of errors may be present in real systems, it can be argued that the degree of approximation provided by a model with 10 terms is acceptable. Examples of errors are; errors estimating the SNR at the receiver, errors introduced by the quantization in the AMC schemes, errors introduced by the channel characteristics quantized by channel quality indicators, etc.; and in addition considering the random nature of the noise and presence of interferences.

As observed in Fig. 6.5, the prediction accuracy of the model composed of 10 terms for the selected set of BLER curves presents in general peaks with a relative error between 20 and 30 %, and in some cases the error rises to 54% . For low BLER, in the region between BLER 0.7 and 1.0, the prediction of the model is not very good. However, the discrepancy is big only in relative values. As observed in the figure, the difference in absolute values is still very small. In the following sub-section the model with 10 terms in summarized.

Table 6.1: Coefficients k_j for 7 to 18 terms in use and fitting metric (Note that some terms contain more than one coefficient)

coef. number k_j	term number (with coef. k_j)	7 terms	8 terms	9 terms	10 terms	11 terms	12 terms	13 terms	14 terms	15 terms	16 terms	17 terms	18 terms
1	1	-27.2489	6.5200	-13.8411	-13.1386	-12.8937	61.2652	-35.0093	-338.1510	-14.3270	-14.1643	-12.6511	-221.3426
2	2	-3.1761	-4.8296	-2.6482	-2.4077	-2.4091	-2.6134	-2.1779	-2.1160	-1.5349	-1.5577	-1.5763	-1.5804
3	3	23.6144	6.9671	16.6463	15.0537	14.8451	28.1919	25.9347	-480.8839	10.1519	9.7506	6.4604	-314.7467
4	4				-12.0028	-12.1513	-10.2578	-10.4460					
5	5			-1.8800	-2.1372	-2.1694	-1.9463	-1.9528	-1.9029				
6	6		1.4655					-0.4679	-0.5864				
7	7			9.9573	13.7740	13.9720	12.1109	12.2149	10.2084				
8	8								173.5752			2.3187	117.2328
9	9								-30.1222		0.5054		-21.1800
10	10						-86.1698	13.1045	678.4958			-0.0020	436.1255
11	10						0.1845	-1.1595	0.3546			-12.3952	0.3545
12	11	4.5688	-11.8273	-9.1753	-0.0161	-0.0344	-1.4578	-2.0768	-9.4391	0.3252	0.1214	0.1419	0.1386
13	11	-0.3474	0.1212	0.9506	0.7567	0.7177	0.2845	0.1552	0.9513	-0.1766	-1.3833	-1.3328	-1.3358
14	11	-0.4750	-0.9602	-0.3201	-9.6247	-8.2018	-2.9378	-3.2560	-0.2926	3.6810	1.8811	1.6635	1.6878
15	12	-54.6792	-54.4709	-15.7713	-16.1596	-15.6002	-14.5627	-15.2610	-15.3147	-321.9379	-56.5916	76.5570	-293.2443
16	13	-9.3045	-9.1430	-1.4052	-1.4962	-1.6437	-1.5037	-1.5841	-1.5857	-1.6613	-1.6017	-1.5910	-1.5902
17	14	60.8762	60.4027	11.0382	11.6328	11.0923	9.4164	10.4197	10.4572	-480.2253	219.3019	-125.1557	-436.8809
18	15									137.9717	96.0959	101.1347	117.3644
19	16									-2.5101	-2.5085	-2.5151	-2.5079
20	17									-1.0874	-1.1899	-1.1984	-1.1975
21	18									0.6509	0.6688	0.6678	0.6588
22	19									180.6604	-491.5097	88.1116	165.2861
23	20									-32.0649	-287.7169	-18.1556	-29.1069
24	21									655.8132	618.8034	-19.0818	596.2286
25	21									0.3554	2.6586	-1.1490	0.3557
26	22					0.1734	0.5084	0.3980	0.4135	-137.2219	-95.3494	-100.3769	-116.6041
27	22					-0.1483	-0.0629	-0.1462	-0.1467	0.9968	0.9956	0.9958	0.9964
28	22					4.6299	3.1352	3.3715	3.3058	-0.0834	-0.1184	-0.1129	-0.0974
Fitting metric		20.62	12.61	3.77	2.88	2.07	1.72	1.59	1.50	1.40	1.34	1.31	1.25

Table 6.2: Model composed of 10 terms

$$P_e(\gamma, B, R) = 1 - \frac{1}{1 + e^{F_1(\gamma,B,R)} + 2e^{F_2(\gamma,B,R)}} \tag{6.2}$$

$$\begin{aligned} F_1(\gamma, B, R) &= u_1 k_1 + u_2 k_2 \gamma + u_3 k_3 R + \\ &\quad + u_4 k_4 B + u_5 k_5 \gamma B + u_7 k_7 BR + \\ &\quad + u_{11} k_{12} B^{k_{13}} R^{k_{14}} \end{aligned} \tag{6.3}$$

$$F_2(\gamma, B, R) = u_{12} k_{15} + u_{13} k_{16} \gamma + u_{14} k_{17} R \tag{6.4}$$

where,

$P_e(\gamma, B, R)$	is the BLER for a given SNR, block length and MCS rate;
γ	is the SNR;
B	is the block length;
R	is the MCS rate;
$u_i = 1$	with $i \in \{1, 2, 3, 4, 5, 7, 11, 12, 13, 14\}$;
k_j	are the coefficients for the model composed of 10 terms, listed in Table 6.1;
$B = B/1000$	is a linear transformation used to increase the stability of the fitting algorithm.

6.2.1 Summary of the model composed of 10 terms

Analyzing the mathematical expression for the model when it is composed of 10 terms, it can be observed that the following terms are present; 1,2,3,4,5,7,11,12,13,14. Table 6.2 summarizes the model composed by 10 terms.

6.3 Discussion

The selection of the final number of terms in the model can be decided by evaluating the trade-off between desired accuracy and complexity of the model. It should be noted that a too accurate model may be not needed due that the system is subject to the random variation of noise, random interferences, the error introduced by the estimation of SNR at

the receiver, errors introduced by the quantization in the AMC schemes, errors introduced by the channel characteristics quantized by channel quality indicators, etc.; factors that may have a variance greater than the accuracy attained by the model. Furthermore it should be noted that one of the initial premises in the design of the model was simplicity. When replacing this model by a look-up table it may be desired to keep the number of computations to a minimum.

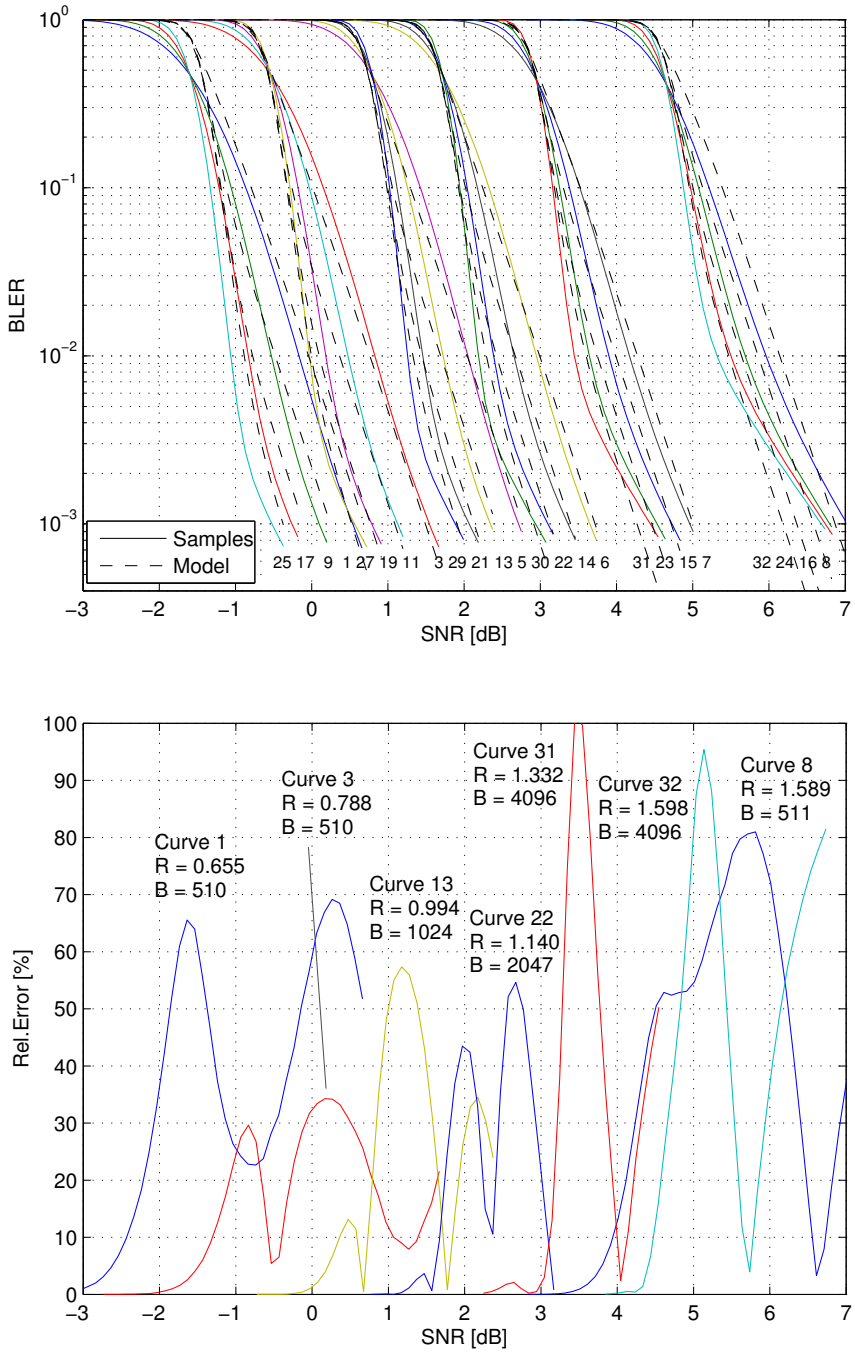

Figure 6.4: BLER curves from simulation and approximated by the model adopting 7 terms (top), and relative error introduced by the model (bottom).

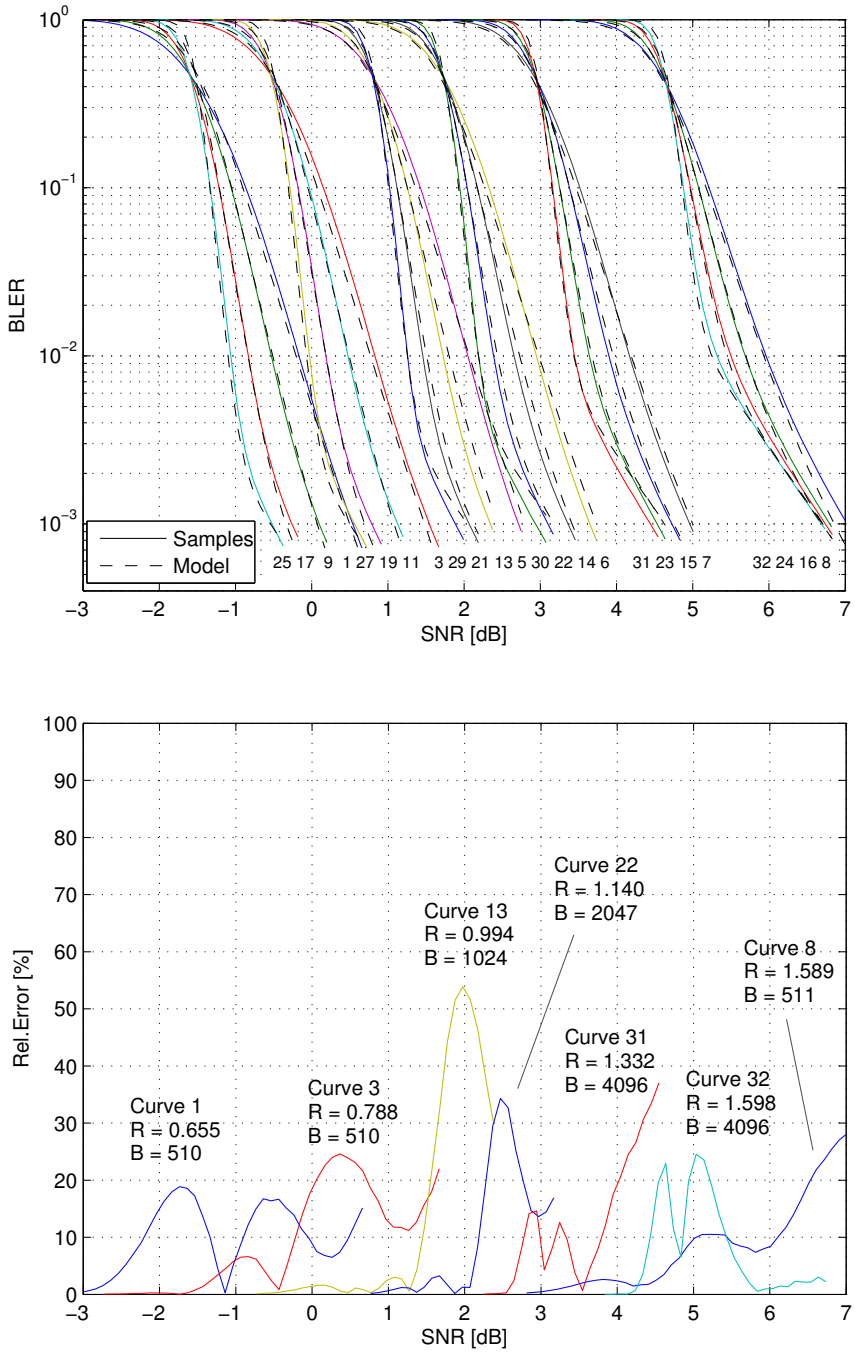

Figure 6.5: BLER curves from simulation and approximated by the model adopting 10 terms (top), and relative error introduced by the model (bottom).

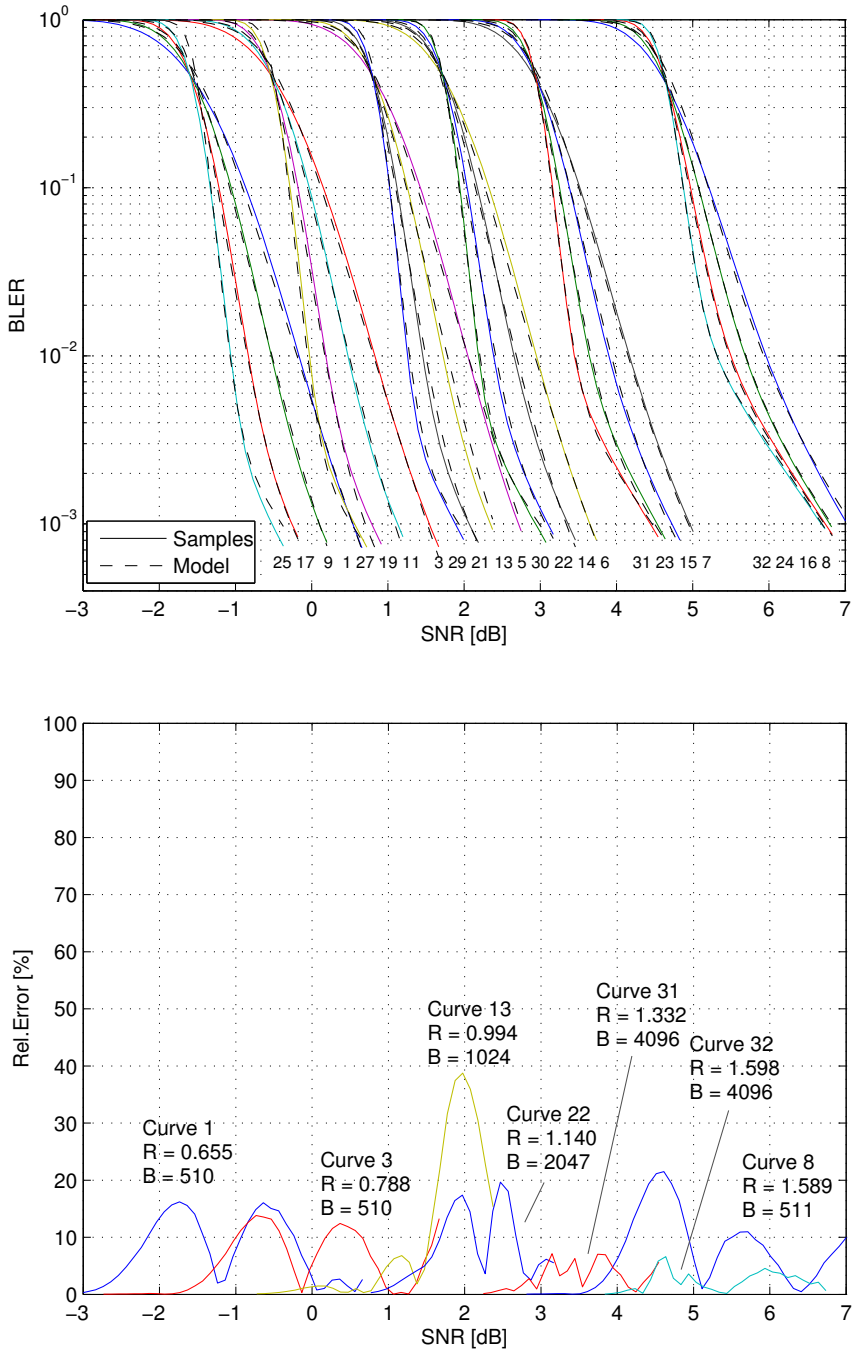

Figure 6.6: BLER curves from simulation and approximated by the model adopting 16 terms (top), and relative error introduced by the model (bottom).

7 Evaluation of the model using 16 QAM modulation

In this chapter the mathematical model approximating BLER curves is calculated for the same link configuration used in previous chapters, except that the modulation is changed from QPSK to 16 QAM. The model used in this case is the model proposed in Chapter 5, Section 5.4. The model presented in Chapter 5 was constructed from BLER data samples generated in a link level simulator transmitting with QPSK modulation. In this case, in presence of 16 QAM modulation, a correct approach would be to elaborate a new model with BLER samples created from a simulator implementing 16 QAM modulation. Nevertheless, and in character of observation, the model elaborated for QPSK will be used here to find an optimized version of the same model, with a reduced set of terms, using as input a collection of BLER samples generated with 16 QAM modulation. Even when the model was not constructed for this modulation type, the final result will let observe how reliable the approximation of the model is when it is presented to data samples from other link configurations.

The set of BLER data samples, produced by simulations using the link level simulator, are presented in Fig 7.1. The selected set of block lengths B and MCS rates R is shown in Table 7.1.

Table 7.1: Set of block lengths B and MCS rates R used in the simulations

Curve number	Approx. MCS rate	MCS rate R	Block length B	Inf. bits n
1	4/3	1.3216032	1023	338
2	16/11	1.4428152	1023	369
3	8/5	1.5874876	1023	406
4	16/9	1.7673508	1023	452
5	2/1	1.9882812	1024	509
6	16/7	2.2756600	1023	582
7	8/3	2.6562500	1024	680
8	16/5	3.1867056	1023	815
9	4/3	1.3274684	2046	679
10	16/11	1.4486804	2046	741
11	8/5	1.5937500	2048	816
12	16/9	1.7723496	2047	907
13	2/1	1.9941408	2048	1021
14	16/7	2.2804104	2047	1167
15	8/3	2.6601564	2048	1362
16	16/5	3.1933592	2048	1635
17	4/3	1.3294272	3072	1021
18	16/11	1.4505208	3072	1114
19	8/5	1.5960912	3070	1225
20	16/9	1.7740148	3071	1362
21	2/1	1.9960936	3072	1533
22	16/7	2.2819928	3071	1752
23	8/3	2.6627604	3072	2045
24	16/5	3.1963528	3071	2454
25	4/3	1.3304028	4095	1362
26	16/11	1.4515264	4095	1486
27	8/5	1.5970696	4095	1635
28	16/9	1.7752808	4094	1817
29	2/1	1.9970704	4096	2045
30	16/7	2.2827840	4095	2337
31	8/3	2.6640624	4096	2728
32	16/5	3.1962892	4096	3273

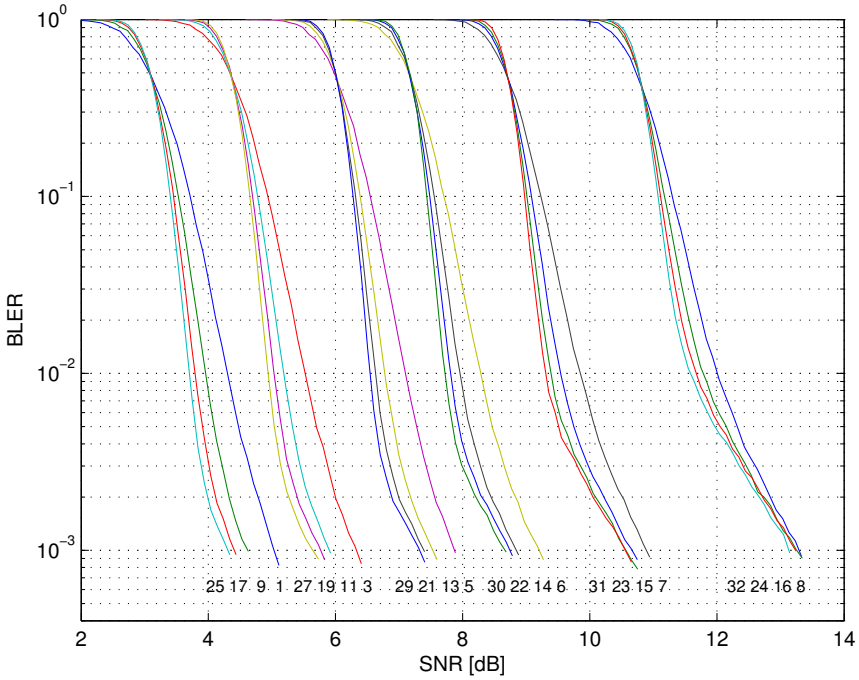

Figure 7.1: BLER performance of turbo code $(13, 15)_8$ for different block lengths and MCS rates, obtained by simulation and averaged over 2000 blocks in error. The system comprises: 16 QAM modulator, AWGN channel, random interleaver, log-MAP decoder with 8 iterations and puncturing patterns described in Appendix A. Numbers at the bottom of the curves indicate a curve number used for reference. Refer to Table 7.1 for details of block lengths and MCS rates.

The degree of approximation attained by the model, measured by a fitting metric, against the number of terms in use in the model can be visualized in Fig. 7.2. The figure actually shows the fitting metric for a few cases; 7, 8, 9, 10, 11, 19 and 20 active terms. The accuracy of the model when it is composed of 10 terms is illustrated in Fig. 7.3. The degree of approximation attained by the model with 10 terms is considered acceptable. Table 7.2 lists the values of coefficient k_j for this case. Note that coefficients k_j in the table were obtained using B = B/1000.

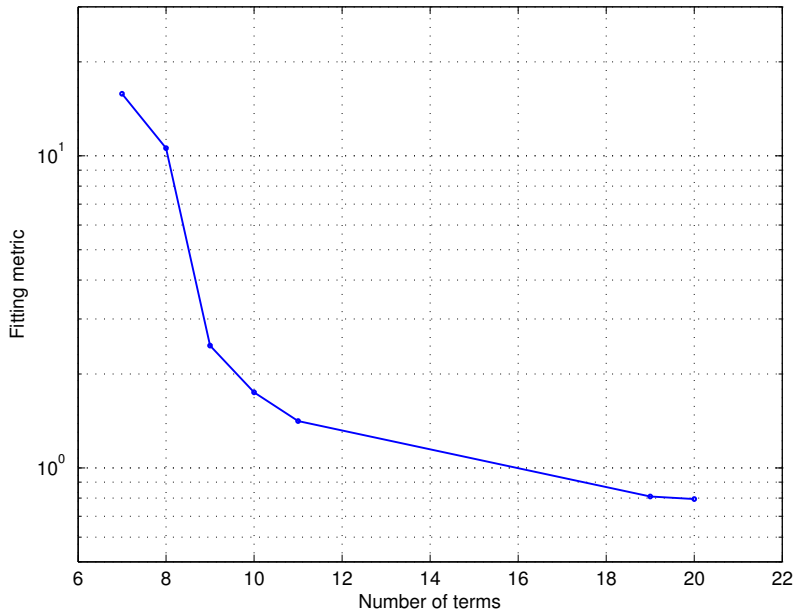

Figure 7.2: Fitting performance against the number of terms in use

Table 7.2: Coefficients k_j for 10 terms in use and fitting metric

coef. number k_j	term number (with coef. k_j)	10 terms
1	1	-4.396207229575736e+00
2	2	-2.346007889190979e+00
3	3	9.245518365509160e+00
4	4	-3.261262896260567e+00
5	5	-1.646349688507273e+00
6	6	
7	7	6.613941731979800e+00
8	8	
9	9	
10	10	
11	10	
12	11	-7.816786934376543e+00
13	11	6.981266939002828e-01
14	11	-8.258370275422028e+00
15	12	-1.012301623703804e+01
16	13	-1.180909132190926e+00
17	14	5.698631324446637e+00
18	15	
19	16	
20	17	
21	18	
22	19	
23	20	
24	21	
25	21	
26	22	
27	22	
28	22	
Fitting metric		1.75

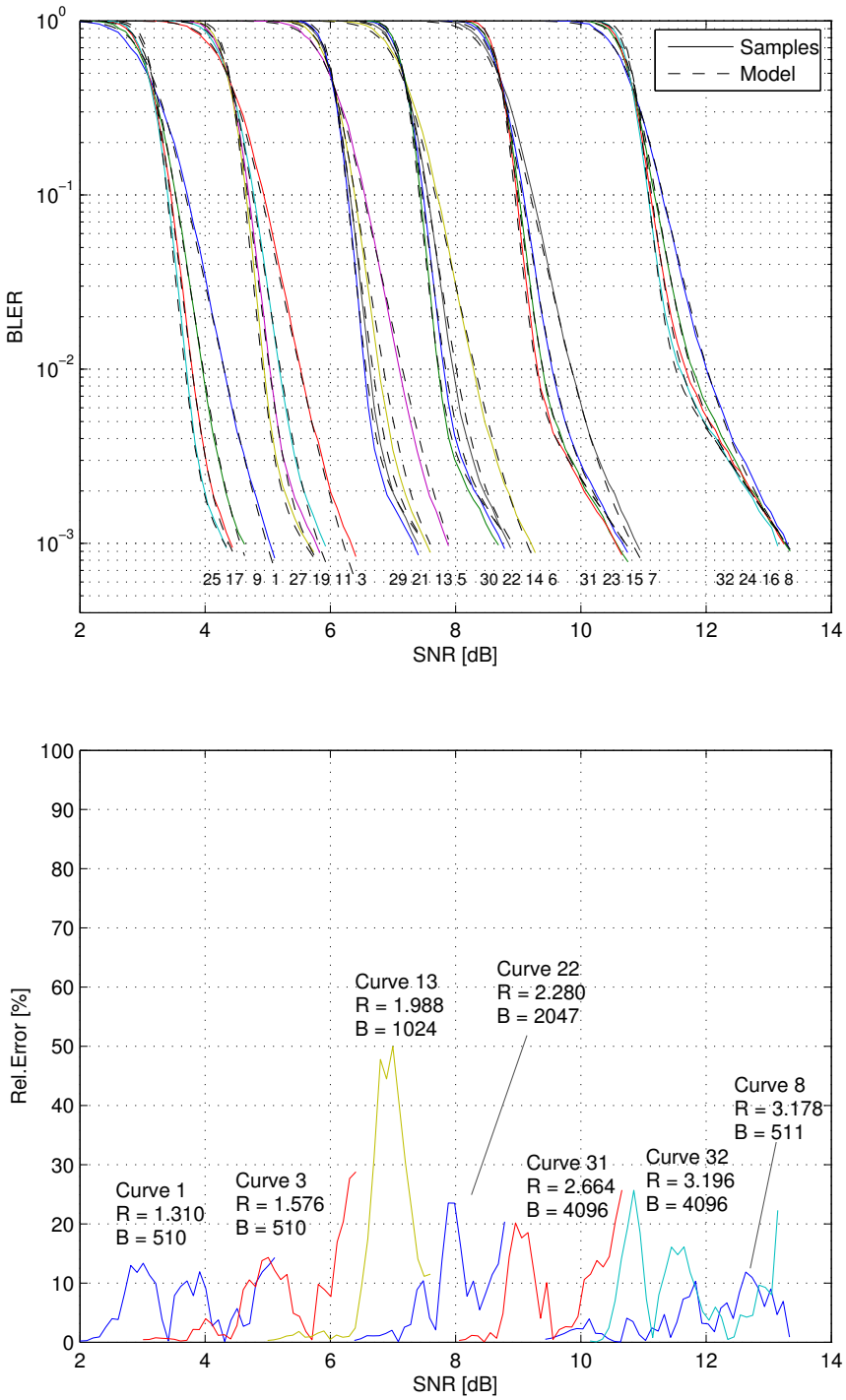

Figure 7.3: BLER curves from simulation and approximated by the model adopting 10 terms (top), and relative error introduced by the model (bottom).

8 Conclusion

In this thesis the author constructed a mathematical model that approximates the BLER performance of punctured turbo codes for different SNR, block lengths and MCS rates. In addition the author elaborated a systematic method to construct the model. The model consists of a continuous and differentiable function that can be used for two different purposes. First, the model can be used to replace look-up tables in system level simulators; bringing the advantage of introducing a continuous set of block lengths and MCS rates, not available in look-up tables. Second, with the help of this function communications systems can be optimized using analytical methods based on differentiation. For example, the function is a valuable tool for optimizing analytically MCSs in Adaptive Modulation and Coding.

The construction of the mathematical model was developed for the particular case of the Turbo Code $(13, 15)_8$ and QPSK modulation; but the method to construct the model can be generalized to other Turbo Codes and modulation types. For other Turbo Codes and modulation types the nature of the BLER curves is the same, changing only their positions and curvatures. The model was also tested with a different link configuration using 16 QAM modulation; in this case the degree of approximation of the model, fitting the BLER samples, is similar to the obtained for the QPSK modulation case.

For the link configuration using QPSK modulation with the TC $(13, 15)_8$, the author reported a mathematical model consisting of a set of functional forms, each providing different degrees of accuracy, as function of the number of terms in the model. The selection of the final number of terms in the model can be decided by evaluating the trade-off between desired accuracy and complexity of the model. The degree of approximation attained with a model composed of 10 terms was considered acceptable for the intended application, namely, modeling the link level behavior at system level.

The model constructed was tested replacing look up-tables and also an early version of the model proved to be useful optimizing analytically MCSs in Adaptive Modulation and Coding [6].

References

[1] S. Hämäläinen, "WCDMA Radio Network Performance," *Jyväskylä Studies in Computing*, no. 28, Feb., 2003.

[2] A. Oborina and M. Moisio, "Speed up of effective SINR mapping calculations for system level simulations," *EuCAP 2007*, Nov., 2007.

[3] 3GPP TS 36.212 v9.0.0 (2009-12), "Evolved Universal Terrestrial Radio Access (E-UTRA), Multiplexing and channel coding," 3GPP TS 36.212, version 9.0.0, Release 9, 2009.

[4] K. Hole, H. Holm, and G. Oien, "Adaptive multidimensional coded modulation over flat fading channels," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 7, pp. 1153 –1158, Jul., 2000.

[5] Q. Liu, S. Zhou, and G. Giannakis, "Cross-Layer combining of adaptive Modulation and coding with truncated ARQ over wireless links," *Wireless Communications, IEEE Transactions on*, vol. 3, no. 5, pp. 1746 – 1755, Sept., 2004.

[6] C. -H. Yu, S. Lembo, K. Ruttik, and O. Tirkkonen,, "Optimum Rate Quantization for Adaptive Modulation and Coding," *WPMC 2008*, Sept., 2008.

[7] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science mag.*, vol. 324, no. 5923, pp. 81–85, Apr., 2009.

[8] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1992.

[9] J. A. Green, *Sets and groups*. Routledge & Kegan Paul, 1965.

[10] M. Moisio and A. Oborina, "Comparison of Effective SINR Mapping with Traditional AVI Approach for Modeling Packet Error Rate in Multistate Channel," in *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, ser. Lecture Notes in Computer Science, Y. Koucheryavy, J. Harju, and V. Iversen, Eds. Springer Berlin / Heidelberg, 2006, vol. 4003, pp. 461–473.

[11] S. Hämäläinen, P. Salina, M. Hartman, A. Lappeteläinen, H. Hola, and O. Salonaho, "A novel interface between link and system level simulations," *Proc. ACTS Summit 1997, Aalborg, Denmark*, pp. 599–604, Oct., 1997.

[12] Ericsson, "Effective SNR mapping for modeling frame error rates in multiplestate channels," *3GPP2-C30-20030429-010*, Apr. 2003.

[13] Ericsson, "System level evaluation of OFDM further considerations," *3GPP TS-GRAN WG1 35, R1-031303, Lisbon, Portugal*, Nov. 2003.

[14] E. Malkamäki, F. de Ryck, C. Mourot, and A. Urie, "A method for combining radio link simulations and system simulations for a slow frequency hopped cellular system," in *Vehicular Technology Conference, 1994 IEEE 44th*, Jun. 1994, pp. 1145 –1149 vol.2.

[15] J. Wigard and P. Mogensen, "A simple mapping from C/I to FER and BER for a GSM type of air-interface," in *Personal, Indoor and Mobile Radio Communications, 1996. PIMRC'96., Seventh IEEE International Symposium on*, vol. 1, Oct. 1996, pp. 78 –82 vol.1.

[16] H. Olofsson, M. Almgren, C. Johansson, M. Höök, and F. Kronestedt, "Improved interface between link level and system level simulations applied to GSM," in *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, Oct. 1997, pp. 79 –83 vol.1.

[17] K. Brueninghaus, D. Astely, T. Sälzer, S. Visuri, A. Alexiou, S. Karger, and G.-A. Seraji, "Link performance models for system level simulations of broadband radio access systems," in *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, vol. 4, Sept., 2005, pp. 2306 –2311 Vol. 4.

[18] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *Communications, IEEE Transactions on*, vol. 44, no. 10, pp. 1261 –1271, Oct. 1996.

[19] L. Hanzo, T. H. Liew, and B. L. Yeap, *Turbo coding, turbo equalisation, and space-time coding*. Wiley, Jul., 2002.

[20] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *Information Theory, IEEE Transactions on*, vol. 20, no. 2, pp. 284 – 287, Mar. 1974.

[21] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *ETT*, vol. 8, pp. 119–125, 1997.

[22] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structure for ISI channels," *Communications, IEEE Transactions on*, vol. 42, no. 234, pp. 1661 –1671, Feb/Mar/Apr., 1994.

[23] J.-F. Cheng, A. Nimbalker, Y. Blankenship, B. Classon, and T. Blankenship, "Analysis of Circular Buffer Rate Matching for LTE Turbo Code," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, Sept., 2008, pp. 1 –5.

[24] S. Crozier, P. Guinand, and A. Hunt, "On Designing Turbo-Codes with Data Puncturing," in *Proc. of the 2005 Canadian Workshop on Information Theory (CWIT 2005), Montral, Quebec, Canada*, Jun., 2005.

[25] M. Fan, S. C. Kwatra, and K. Junghwan, "Analysis of puncturing pattern for high rate turbo codes," *Proc. MILCOM'99, USA*, pp. 547–500, Oct., 1999.

[26] S. Dolinar and D. Divsalar, "Weight distribution for turbo codes using random and nonrandom permutations," *TDA Progress Report 42-122*, pp. 56–65, Aug., 1995.

[27] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009.

[28] S. Lembo, K. Ruttik, and O. Tirkkonen, "Modeling of Convolutional code performance," *WPMC 2008*, Sept., 2008.

[29] T. F. Coleman and Y. Li, "On the Convergence of Reflective Newton Methods for Large-Scale Nonlinear Minimization Subject to Bounds," *Mathematical Programming*, vol. 67, no. 2, pp. 189–224, 1994.

[30] T. F. Coleman and Y. Li, "An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds," *SIAM Journal on Optimization*, vol. 6, pp. 418–445, 1996.

[31] S. Lembo, K. Ruttik, and O. Tirkkonen, "Modeling BLER Performance of Punctured Turbo Codes," *WPMC 2009*, Sept., 2009.

Appendix A - Puncturing patterns

Table A.1 lists the puncturing patterns used in the two parity outputs of the turbo encoder.

The puncturing patterns used in the two parity outputs of the turbo encoder are listed in Table A.1

Table A.1: Puncturing patterns used in the parity outputs of the turbo encoder

Code rate	Puncturing pattern
1/3	11111111 / 11111111
4/11	10111111 / 11111011
2/5	10111011 / 11101110
4/9	10101101 / 01011011
1/2	10101010 / 01010101
4/7	00101010 / 01010100
2/3	10001000 / 00010001
4/5	00100000 / 01000000

Appendix B - Coefficients k_j

Tables B.1, B.2 and B.3 list coefficients k_j with the maximum precision available for the model describing the system presented in Chapter 4. The block length used in the model was scaled to $B = B/1000$.

Table B.1: Coefficients k_j for 7 to 11 terms in use and fitting metric (Note that some terms contain more than one coefficient)

coef. num. k_j	term num.	7 terms	8 terms	9 terms	10 terms	11 terms
1	1	-2.724893038374986531e+01	6.52002567293930024 3e+00	-1.384110994204367273e+01	-1.313856934437080426e+01	-1.289372883584505480e+01
2	2	-3.176065036293461485e+00	-4.829555322248101135e+00	-2.648183496478103471e+00	-2.407747425874620539e+00	-2.409100875629760274e+00
3	3	2.361437788054879405e+01	6.967056223387426250e+00	1.664630438398532419e+01	1.505371559801515424e+01	1.484509183468462012e+01
4	4				-1.200276379511393010e+01	-1.215129521364400134e+01
5	5			-1.879965088718190680e+00	-2.137196656929706062e+00	-2.169351069183298453e+00
6	6		1.465522374004128103e+00			
7	7			9.957334999473932413e+00	1.377404680685739713e+01	1.397203586419015942e+01
8	8					
9	9					
10	10					
11	10					
12	11	4.568754968475158940e+00	-1.182727741593378923e+01	-9.175257638427975593e+00	-1.606408303966204432e-02	-3.441779436897857841e-02
13	11	-3.473783492963579689e-01	1.212122765035859445e-01	9.506416761434079987e-01	7.567481320202725792e-01	7.177372569264872659e-01
14	11	-4.749806400590025901e-01	-9.602451730800093399e-01	-3.201405588125300072e-01	-9.624744980098247638e+00	-8.201810142120949720e+00
15	12	-5.467921017732502520e+01	-5.447087675047405497e+01	-1.577131377220961639e+01	-1.615964056963249007e+01	-1.560019742617267546e+01
16	13	-9.304528354180211736e+00	-9.142981552941773060e+00	-1.405225684001734976e+00	-1.496210314177089939e+00	-1.643720705054497744e+00
17	14	6.087619060466391829e+01	6.040269928513832554e+01	1.103822734840225195e+01	1.163281730851091922e+01	1.109229709816489695e+01
18	15					
19	16					
20	17					
21	18					
22	19					
23	20					
24	21					
25	21					
26	22					1.733734579275171550e-01
27	22					-1.483325748423428014e-01
28	22					4.629873269519158008e+00

Table B.2: Coefficients k_j for 12 to 16 terms in use and fitting metric (Note that some terms contain more than one coefficient)

coef. num. k_j	term num.	12 terms	13 terms	14 terms	15 terms	16 terms
1	1	6.126519370114349528e+01	-3.500931165091812147e+01	-3.381510335536394791e+02	-1.432704896096129943e+01	-1.416428260211863410e+01
2	2	-2.613383803603530176e+00	-2.177877938624108278e+00	-2.115955041862420671e+00	-1.534933883713486935e+00	-1.557689604141395456e+00
3	3	2.819192587490612922e+01	2.593473402794493410e+01	-4.808839462766941892e+02	1.015189686476490927e+01	9.750621587903536280e+00
4	4	-1.025782895745832235e+01	-1.044597847433965043e+01			
5	5	-1.946288802945711494e+00	-1.952803099684746524e+00	-1.902866367232151967e+00		
6	6		-4.679113625643586682e-01	-5.863538469952406240e-01		
7	7	1.211086483364748112e+01	1.221491440796560646e+01	1.020843446194253801e+01		
8	8			1.735751946740110156e+02		
9	9			-3.012223111688305011e+01		5.054334148976140639e-01
10	10	-8.616978421208156647e+01	1.310452059385927548e+01	6.784957977732294694e+02		
11	10	1.844665831832030534e-01	-1.159485998167805842e+00	3.545904956146566644e-01		
12	11	-1.457824827458140460e+00	-2.076827234668995548e+00	-9.439103173890963205e+00	3.252098911862531527e-01	1.214201323351446343e-01
13	11	2.844528963630535978e-01	1.551861548076608754e-01	9.513214730361411187e-01	-1.766257666852290298e-01	-1.383312421075284737e+00
14	11	-2.937769609210428179e+00	-3.256011133469526087e+00	-2.926137405840018024e-01	3.681027887626905581e+00	1.881140376631312394e+00
15	12	-1.456265650147184409e+01	-1.526100960565360154e+01	-1.531470487517242773e+01	-3.219378870555775620e+02	-5.659158542682529713e+01
16	13	-1.503708280691083532e+00	-1.584108135559086028e+00	-1.585703283786378259e+00	-1.661313163035704310e+00	-1.601686368431889163e+00
17	14	9.416424645179645836e+00	1.041972301090886965e+01	1.045720517301280772e+01	-4.802252824907909599e+02	2.193019457921577668e+02
18	15				1.379717041374230746e+02	9.609587946351153676e+01
19	16				-2.510118182934120590e+00	-2.508490975007322987e+00
20	17				-1.087422799176952282e+00	-1.189937532278004140e+00
21	18				6.508830880120533013e-01	6.688055550456815102e-01
22	19				1.806604391762087971e+02	-4.915097402351492519e+02
23	20				-3.206490564598433934e+01	-2.877169342935067675e+02
24	21				6.558131844809007589e+02	6.188033865797203816e+02
25	21				3.553936186073162751e-01	2.658564172960051680e+00
26	22	5.083854487449678272e-01	3.980299926844327874e-01	4.134921679070399092e-01	-1.372219325648873394e+02	-9.534940101617856101e+01
27	22	-6.293620565114438348e-02	-1.469134199957299063e-01	-1.466673454301629653e-01	9.968451522880998894e-01	9.955866807630101567e-01
28	22	3.135155127559850818e+00	3.371485232044230784e+00	3.305845041430204301e+00	-8.335286209495036680e-02	-1.183773604838708193e-01

Table B.3: Coefficients k_j for 17 to 21 terms in use and fitting metric (Note that some terms contain more than one coefficient)

coef. num. k_j	term num.	17 terms	18 terms	19 terms	20 terms	21 terms
1	1	-1.26511467168999 6406e+01	-2.21342571927600 5452e+02	5.03461496066919 7359e+01	6.17743429924522 8999e+01	4.83505617001165 4331e+01
2	2	-1.57631779638244 4403e+00	-1.58035615048970 9899e+00	-1.61391361048750 2703e+00	-1.55904493923371 4899e+00	-1.56798740809159 0984e+00
3	3	6.46042334174843 0026e+00	-3.14746748143376 6352e+02	-9.88411702890828 1100e+01	-1.10500758135623 5988e+02	-9.76837743061132 9260e+01
4	4					-2.43529422059637 6843e-03
5	5					
6	6				-4.77052540119165 9520e-02	-3.39032316228282 6708e-02
7	7			-4.77213780844459 1039e-02	-4.91380914063686 5580e-02	-4.58969026921493 5503e-02
8	8	2.31874761704786 3141e+00	1.17232789426082 9169e+02	7.09817102366340 4355e+01	7.65633153350809 4367e+01	7.07454276314565 9308e+01
9	9		-2.11795672319265 665e+01	-1.57513874097969 7438e+01	-1.67232429146501 8931e+01	-1.57207731621599 4891e+01
10	10	-2.01860248184451 9996e-03	4.36125450781027 4390e+02	-1.04077503013137 8820e+01	-1.47925055816359 1656e+01	-9.38720173992307 6715e+00
11	10	-1.23952257718171 9321e+01	3.54549949360103 9775e-01	-1.46733606401737 5374e+00	-1.26622425795650 6838e+00	-1.54135470924256 3903e+00
12	11	1.41880371172969 4217e-01	1.38599429121273 4320e-01	2.94994896027877 1733e-06	2.00826709894452 7476e-06	3.68198812350290 4926e-06
13	11	-1.33282416501199 9851e+00	-1.33582671804571 0848e+00	-1.71977791759712 7725e+01	-1.77858793127775 9182e+01	-1.68507336171911 3194e+01
14	11	1.66346407058071 0441e+00	1.68780431158775 6123e+00	9.48219223242066 0893e-01	8.71526534026405 3417e-01	1.00228568083962 9212e+00
15	12	7.65570267803821 7357e+01	-2.93244335279539 7686e+02	-4.98999222921957 3261e+01	-2.81886381070900 3701e+02	-4.83240621051356 6691e+01
16	13	-1.59103894209153 6068e+00	-1.59015566374758 8877e+00	-1.59502804127957 3650e+00	-1.65515551014828 7545e+00	-1.59446222208405 8522e+00
17	14	-1.25155746597719 7268e+02	-4.36880932680975 0360e+02	1.87557870737794 1612e+02	-4.11400028604015 8016e+02	1.80144494886970 1932e+02
18	15	1.01134675972879 8759e+02	1.17364428342002 5403e+02	1.28625427115869 6838e+02	-2.00864505990508 2633e+02	1.20301658195888 8338e+02
19	16	-2.51506139426648 4118e+00	-2.50792974794685 1418e+00	-2.50387093317589 8372e+00	-2.41255770161576 5549e+00	-2.50453687070219 7771e+00
20	17	-1.19840503966504 8299e+00	-1.19745087815023 7763e+00	-1.20965200298125 7191e+00	-1.14750042711928 8791e+00	-1.21465842997670 0260e+00
21	18	6.67814276741157 5642e-01	6.58807809893064 9773e-01	6.53188243258294 3876e-01	5.55955827915158 6291e-01	6.56232576878766 1273e-01
22	19	8.81115552703030 1471e+01	1.65286082719928 2880e+02	-4.15672713779483 6153e+02	1.54126425553918 9492e+02	-3.98230303882907 5490e+02
23	20	-1.81556040204867 0596e+01	-2.91068614789448 9650e+01	-2.42610747075497 7840e+02	-2.67464720822340 9223e+01	-2.31712999009249 3301e+02
24	21	-1.90818340294958 8370e+01	5.96228610441376 9863e+02	5.22943557092268 2928e+02	5.68232604917623 1209e+02	5.00452752302757 9150e+02
25	21	-1.14897035795368 3186e+00	3.55701252014854 9184e-01	2.65828966668823 3410e+00	3.55307932924256 6087e-01	2.65766616599497 2296e+00
26	22	-1.00376926262063 7619e+02	-1.16604078599511 4533e+02	-1.27870571977940 2301e+02	2.01612846501397 9110e+02	-1.19552303038467 0875e+02
27	22	9.95832891143652 2546e-01	9.96420012328574 2450e-01	9.96708594033986 9277e-01	1.00210816486425 7800e+00	9.96463626405019 6202e-01
28	22	-1.12919881212482 6284e-01	-9.73779519500504 4535e-02	-8.89095046805028 9046e-02	5.70428881016406 4027e-02	-9.49246610897605 6749e-02

Appendix C - Example code (QPSK case)

The code listed below shows an example, in MATLAB programming language, of the model with 10 terms for the QPSK case.

```
% Plotting QPSK BLER curves with fitting function using 10 terms
% Sergio - 2011

% xdata contains 3 the independent variables,
% arranged in 3 columns (SNR, B and coding R)

xdata = [
2.6176e-01  510  3.274510e-001
2.4698e+00 2047  5.701026e-001
6.0362e+00 4096  7.990723e-001
]

% Calculate MCS rate from code rate and current modulation (QPSK)
xdata(:,3) = xdata(:,3) * 2;

% Divide by 1000 block length
xdata(:,2) = xdata(:,2) / 1000;

% Arrange data
xdata = xdata';

% Coefficients kj for the model adopting 10 terms
x1 = [
  -1.313856934437080426e+01
  -2.407747425874620539e+00
   1.505371559801515424e+01
  -1.200276379511393010e+01
  -2.137196656929706062e+00
                          0
   1.377404680685739713e+01
                          0
                          0
                          0
                          0
  -1.606408303966204432e-02
   7.567481320202725792e-01
  -9.624744980098247638e+00
  -1.615964056963249007e+01
  -1.496210314177089939e+00
   1.163281730851091922e+01
                          0
                          0
                          0
                          0
                          0
                          0
                          0
                          0
                          0
                          0
]

x1 = x1';

% Arrange data
S = xdata(1,:); % SNR
B = xdata(2,:); % B
R = xdata(3,:); % R
```

```
% Function from the model
F = @(x, S, B, R) (...
       1 - ( (1./ (1 + ...
         exp(...
           x(1) + x( 2).*S + x(3).*R + ...
           x( 4).* B    + ...
           x( 5).* S.*B + ...
           x( 6).* S.*R + ...
           x( 7).* B.*R + ...
           x( 8).* R.^2 + ...
           x( 9).* R.^3 + ...
           x(10).* R.^x(11) + ...
           x(12).* B.^x(13) .* R.^x(14) ...
         )...
         + 2 * exp(...
           x(15) + x(16).*S + x(17).*R + ...
           x(18).* B    + ...
           x(19).* S.*B + ...
           x(20).* S.*R + ...
           x(21).* S.*B.*R + ...
           x(22).* R.^2 + ...
           x(23).* R.^3 + ...
           x(24).* R.^x(25) + ...
           x(26).* B.^x(27) .* R.^x(28) ...
         )...
       )))...
      );

% The result of the function
BLER = F(x1, S, B, R)

%Figure
figure
semilogy(S,BLER,'o')
grid
xlabel('SNR')
ylabel('BLER')
```

Appendix D - Example code (16 QAM case)

The code listed below shows an example, in MATLAB programming language, of the model with 10 terms for the 16 QAM case.

```
% Plotting 16 QAM BLER curves with fitting function using 10 terms
% Sergio - 2011

% xdata contains 3 the independent variables,
% arranged in 3 columns (SNR, B and coding R)

xdata = [
4.0110e+00 1023  3.304008e-01
7.0976e+00 2048  4.985352e-01
1.1446e+01 4096  7.990723e-01
]

% Calculate MCS rate from code rate and current modulation (16 QAM)
xdata(:,3) = xdata(:,3) * 4;

% Divide by 1000 block length
xdata(:,2) = xdata(:,2) / 1000;

% Arrange data
xdata = xdata';

% Coefficients kj for the model adopting 10 terms
x1 = [
    -4.396207229575736e+000
    -2.346007889190979e+000
     9.245518365509160e+000
    -3.261262896260567e+000
    -1.646349688507273e+000
                         0
     6.613941731979800e+000
                         0
                         0
                         0
                         0
    -7.816786934376543e+000
     6.981266939002828e-001
    -8.258370275422028e+000
    -1.012301623703804e+001
    -1.180909132190926e+000
     5.698631324446637e+000
                         0
                         0
                         0
                         0
                         0
                         0
                         0
                         0
                         0
                         0
                         0
]

x1 = x1';

% Arrange data
S = xdata(1,:); % SNR
B = xdata(2,:); % B
R = xdata(3,:); % R
```

```
% Function from the model
F = @(x, S, B, R) (...
       1 - ( (1./ (1 + ...
         exp(...
           x(1) + x( 2).*S + x(3).*R + ...
           x( 4).* B    + ...
           x( 5).* S.*B + ...
           x( 6).* S.*R + ...
           x( 7).* B.*R + ...
           x( 8).* R.^2 + ...
           x( 9).* R.^3 + ...
           x(10).* R.^x(11) + ...
           x(12).* B.^x(13) .* R.^x(14) ...
         )...
         + 2 * exp(...
           x(15) + x(16).*S + x(17).*R + ...
           x(18).* B    + ...
           x(19).* S.*B + ...
           x(20).* S.*R + ...
           x(21).* S.*B.*R + ...
           x(22).* R.^2 + ...
           x(23).* R.^3 + ...
           x(24).* R.^x(25) + ...
           x(26).* B.^x(27) .* R.^x(28) ...
         )...
       )))...
      );

% The result of the function
BLER = F(x1, S, B, R)

%Figure
figure
semilogy(S,BLER,'o')
grid
xlabel('SNR')
ylabel('BLER')
```

MATLAB® is a registered trademark of The MathWorks, Inc.