UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS
Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

EVALUATION, DESIGN AND IMPLEMENTATION OF FEC TECHNIQUES FOR SATELLITE PLATFORMS RESOURCE CONSTRAINED POCKETCUBE FOR IOT APPLICATIONS

Jacobo Díez Venegas

2024

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Título:	Evaluation, design and implementation of fec techniques for satellite platforms resource constrained PocketCube for IoT applications.
Autor:	D. Jacobo Díez Venegas
Tutor:	D. Ramón Martínez Rodríguez - Osorio y D. Miguel Alejandro Salas Natera
Ponente:	
Departamento:	Señales, Sistemas y Radiocomunicaciones (SSR)

MIEMBROS DEL TRIBUNAL

Presidente:	D. .....
Vocal:	D. .....
Secretario:	D. .....
Suplente:	D. .....

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de: .....

Madrid, a de de 20.....

This page has been intentionally left blank.

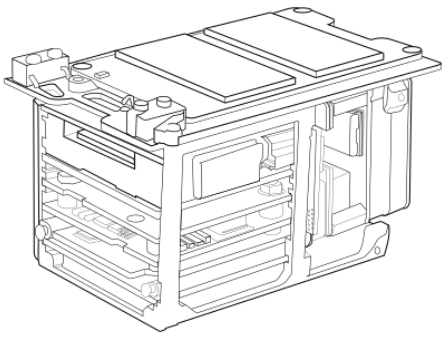
UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS
Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

EVALUATION, DESIGN AND IMPLEMENTATION OF FEC TECHNIQUES FOR SATELLITE PLATFORMS RESOURCE CONSTRAINED POCKETCUBE FOR IOT APPLICATIONS



Jacobo Díez Venegas

2024

RESUMEN

Las técnicas *Forward Error Correction* (FEC) pretenden corregir los errores que pueda ocasionar el canal, al contrario que el esquema *Automatic Response reQuest* (ARQ), que se basa en la retransmisión de los paquetes erróneos. Los códigos correctores *FEC* se convierten en una necesidad cuando la transmisión de datos consume demasiados recursos, algo a tener en cuenta en sistemas con prestaciones limitadas. Este trabajo pretende evaluar, diseñar e implementar un código corrector en un pocketCube con capacidades limitadas.

En primer lugar se procede a la evaluación del satélite, estudiando la potencia y capacidad computacional. Del mismo modo se analizan los requisitos y beneficios de cada código, así como, complejidad computacional y la ganancia de codificación obtenida.

Seguidamente, una vez se ha escogido la familia del FEC, se diseña cual es la mejor opción. Esta etapa del proyecto se basa en escoger cual es la redundancia necesaria para ofrecer la mejor protección posible sin perjudicar en demasía a la velocidad binaria. También es mencionable la elección del algoritmo de codificación y descodificación como medida del ahorro computacional.

Por último, se implementa la decisión final en el software del satélite. Se llevan a cabo mediciones finales de la complejidad computacional y se integra en el entorno software del satélite. Además se verifica con herramientas externas como *Octave* ©que efectivamente, tanto el proceso de codificación y descodificación sea correcto, incluyendo la capacidad correctora del FEC.

SUMMARY

The Forward Error Correction (FEC) techniques pretend to correct the errors that may be introduced by the channel, as opposed to the Automatic Response reQuest (ARQ), which attempts to retransmit again all the erroneous packets. The FEC correction becomes a necessity when the data transmission requires too many resources, it is mandatory to take into account because of the lack of capabilities. This work aims to evaluate, design and implement a corrector code on the pocketCube platform with limited resources.

First, the satellite is evaluated to determine the available power and computational capabilities. The requirements and benefits of each type of corrector, in particular computational complexity and coding gain, are assessed.

Once the FEC family has been selected, it is time to design which is the best option. This stage is based on the election of how many redundancy bits to allocate in order to provide the best protection without degrading the binary rate. It is worth mentioning the encoding and decoding algorithm as a measure of computational savings.

The final step is the implementation in the satellite software. New final measures of computational complexity are made and it is integrated with other systems. It will be verified using external tools such as *Octave* ©to check that the encoding and decoding procedures are correct, including the error correction rate.

PALABRAS CLAVE

FEC, tasa de error en bit (BER), BCH, Reed - Solomon, ganancia de codificación, recursos limitados, IoT, pocketCube, satélite, evaluación, diseño, implementación, verificación.

KEYWORDS

FEC, Bit Error Rate (BER), BCH, Reed - Solomon, coding gain, limited resources, IoT, pocketCube, satellite, evaluation, design, implementation, verification.

This page has been intentionally left blank.

A mi familia, amigos y profesores, en especial a mi abuela que me decía que estudiara siempre.

Contents

Glossary

ACK ACKnowledgment

ARA Accumulate Repeat Accumulate

AWGN Additive White Gaussian Noise

BCH r. Bose, d. ray-Chaudhuri a. Hocquenghem. The ECC take its name from the initials

BER Bit Error Rate

BPA Belief Propagation Algorithm

BSC Binary Simmetric Channel

CCSDS Consultative Committee for Space Data Systems

CD Compact Disc

COTS Commercial Off The Shelf

CRC Cyclic Redundance Codes

DSN Deep Space Network

DVB Digital Video Broadcasting

DVB-S2 Digital Video Broadcasting - Satellite 2nd version

ECC Error Correction Codes

FEC Forward Error Correction

FSK Frequency Shift Keying

GF Galois Field

HDD Hard Disk Drive

IoT Internet Of Things

IRA Irregular Reppeat - Accumulate

ITU International Telecommunication Union

LDPC Low Density Parity Check

LTE Long-Term Evolution

PCCC Parallel Concatenated Convolutional Codes

RA Reppeat - Accumulate

RS Reed - Solomon

SC Successive Cancellation

SCCC Serial Concatenated Convolutional Codes

SCL Successive Cancellation List

SDG Sustainable Development Goals

SGP4 Simplified General Perturbations

SPA Sum - Product Algorithm

List of Figures

List of Tables

1 Introduction and motivation

Since World War II, space exploration has achieved numerous milestones, including the launch of Sputnik and the Moon landing by Armstrong. Ensuring equitable access to space for research, especially for non-affluent societies, is imperative. The CubeSat standard, developed by Jordi Puig-Suari and Bob Twiggs, addresses this need with its low-cost satellite platform [1]. Based on 5 cm cubes weighing 1 kg or less, it enables diverse missions, from earth atmospheric studies [2] to Martian communication networks [3]. The standard's adaptability accommodates various satellite sizes and masses [4]. Figure [1] depicts a CubeSat.

Figure 1: Andrew Petro, NASA Small Satellite Program executive, holds a replica of the NASA Phonesat smartphone launched on the Antares test flight on April 21, 2013 [5].

The space improvements were accompanied by other possibilities, including IoT, among others. According to the ITU:

> IoT constitutes a global infrastructure for the information society, facilitating advanced services through the interconnectedness of physical and virtual entities, utilizing evolving interoperable information and communication technologies [6].

IoT tends to connect everything, even the smallest devices regardless of their location and condition. IoT enables data collection, analysis, and decision-making processes, thereby enhancing efficiency and fostering innovative applications.

Both are pivotal advancements for society, offering avenues for our species advancement. Yet, from a technical perspective, challenges persist. CubeSats may encounter difficulty in generating adequate power for system operation, constrained by limited surface area for solar panels and inconsistent solar orientation. Additionally, by definition, IoT encompasses devices which can be located in remote regions, like the middle of the Pacific Ocean. Establishing contact with them can be difficult due to the distance to the nearest antenna and limitations in transmission power.

Under these difficult circumstances, CubeSats are able to communicate even from Mars, as previously mentioned. How is it done? There are several methods to conserve energy and most of them are often combined to multiply their effectiveness. For example,

when flying over areas without service, CubeSats can be turned off. Among others, many power-saving techniques can involve surface equipment too, such as the use of ECC.

The ECC belong to FEC scheme. The FEC scheme pretends to avoid the retransmission of corrupted packets by attempting to correct them. ECC work by adding redundancy bits to the source word resulting in a codeword. The received codewords may contain errors, but the redundancy bits allow the receiver to correct them. Despite the binary rate penalty, the employment of the ECC allows for a reduction in the power consumption while reducing the necessary minimum energy to transmit and the number of retransmissions [1]. [7][8][9].

1.1 Introduction to Hydra´s satellite

Now that the FEC scheme has been introduced, it is time to get down to business. Hydra Space is a small company with a focus on "IoT connectivity" and an "affordable access to space" [10]. For now, they are refining how to connect Earth to the PocketCube, and the task is to add FEC as an additional layer to mitigate any kind of disturbance.

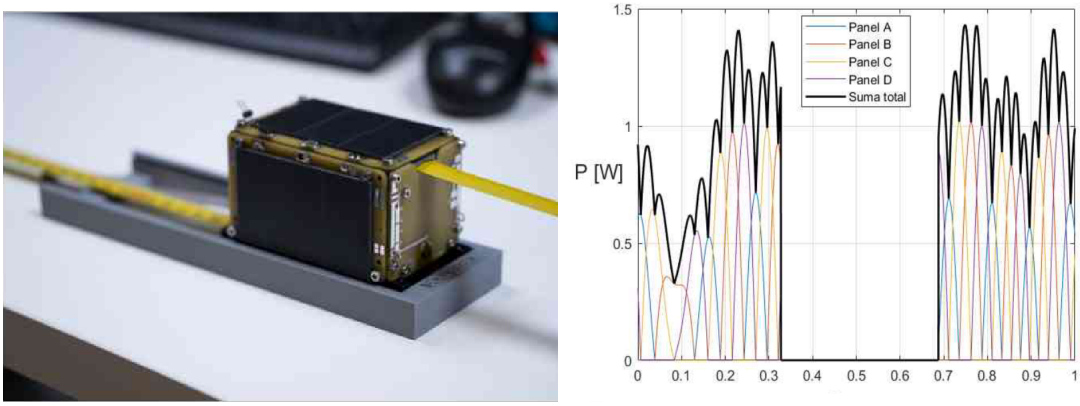
As mentioned above, CubeSats do not generate too much energy so proper power management is essential to maintain a reasonable quality of the link. This work is a collaboration with them to implement an ECC.

Firstly, an understanding of the working environment is necessary so the Genesis-U is presented as an example (see [2a]). It's size is 50x50x80 [mm], slightly smaller than a coke can. This satellite can produce 750 [mW] and store it in a 10 [Wh] battery (see [2b]). Assuming an orbit with a period of 95 minutes with 50 % sun exposure, it provides an average power of 500 [mW]. All the components are originally made for the automotive industry but they follow the COTS philosophy[2]. It's important to note that there is no room for redundant systems. This satellite is fullduplex thanks to the frequency deviation for each direction: UHF for transmitter and VHF for the receiver. The orbital altitude is 500 km, giving an expected lifetime of 24 to 36 months. There's also room for a customer payload with this sizes: $5 \times 5 \times 2$ [cm] or $5 \times 5 \times 8$ [cm] [11].

As a result, these satellites have many limitations. Efficiency is a must because the results of this work are supposed to be useful in most of satellites. The main ones are: available power, computing capacity and time. The first of these are previously mentioned. Small solar cells produce small amounts of energy. The second one, computing power, is a derivative of small space. The possible commercial processors that can be integrated are limited by the small size of the cube. Typically, small cores offer less computing power, so the ECC has to be computationally light. Finally, the time required for codification or decodification can not be very long. Typically, customers focus on countries that are over in a matter of minutes and the orbit may not pass over them again for hours. IoT is not a real-time application, but the value of the data degrades over time.

[1] The channel can modify random bits with probability *channel BER*. See [2.1] for more details.

[2] The components are designed for commercial purposes, not just for the space. This improves availability and the price of the component.



(a) Genesis - U

(b) Graph of power measures

Figure 2: Genesis - U and its power measures

Hydra also simulates the traffic and it can be considered as known. The packets can be consist of different numbers of bytes from 12 to 32 bytes. The majority of packets (50%) are 12 bytes sized and correspond to telemetry and ACK. The other half follows an exponential distribution, reaching the minimum in packets of 32 bytes. Also, the customers cannot send more than 4 packages per day.

1.2 Objectives

Thanks to the identification of the most important obstacles, it is possible to set some targets, however, it's important to listen to Hydra's needs.

The surface devices know when there's a satellite is overhead because they can calculate its position thanks to the SGP4 algorithm. SGP4 works by defining all the orbital parameters, and once the exact position has been determined, SGP4 can calculate its orbit and the passes over a given point on the Earth.

Synchronization poses a considerable challenge since all users can hear the satellite but cannot visually detect each other, leading to packet collisions. Moreover, the considerable distances between users ensure variations in received power, potentially corrupting many bits, albeit not all of them. It is also important to consider noise and interferences. At this point, ECC would kick in and correct the errors. Hydra estimates that this occurs in one in ten bits transmitted. For the most common message, Hydra requires an ECC that fails in less than 9.6 bits on average.

As has been repeated several times before, efficiency is mandatory in this project to keep the satellite and all of its systems alive. The ECC is incorporated into the software of a microcontroller. Hydra has various satellite models with different processors. The STM32F446 can be considered as a reference, mainly to establish values for an average power consumption and computing power but it will not be the final microcontroller. Hydra adapts to the needs of its customers and this means that many different configurations are possible. Although Hydra does the final verification, the ECC must be able to maintain a constant flow of information. As of today, the communication protocol is based on the "stop and wait" architecture. In other words, when a packet is sent, the transmitter does not continue sending more packets until it receives an acknowledgment (ACK) of the previous packet. The designed protocol is intended to be long and stable,

thus the transmission scheme considered will be the "selective repeat transmission", which is illustrated in figure [3].

Hydra claims to be able to transmit at 200 [bps], regardless of the direction of the link. This implies that the transmission time for the smaller packet is 0.48 [s], while for the largest packets this value increases to 1.28 [s]. From this point, the worst case is assumed, the transmission time is 0.48 [s].

To scale the system, the cyclomatic complexity will be a powerful tool. Big(O) notation suffers from the halting problem, so it's not practical for complex programs[3] [13]. Three tools are used: [14][15][16] [4].

Figure [3] shows, in blue, the maximum time to decode the packet (t_{max}) to keep the channel always busy. Note the red and green colours which denote the fullduplex nature of the link. As an IoT application, latency is not an issue and it is assumed that the packets are ready to be transmitted.

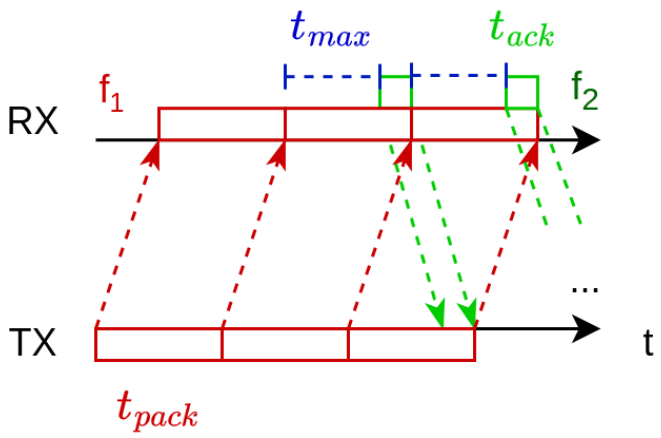


Figure 3: Selective repeat transmission

According to the figure [3]:

$$t_{pack} = t_{max} + t_{ack} \tag{1}$$

This transmission strategy is one of the most effective because it ensures that the channel is always occupied. The smallest packet size coincides with the size of the ACK ($t_{pack} = t_{ack}$) so the maximum time to encode and decode is the transmission time of the smallest packet, t_{pack}. In other words, the FEC layer has to complete its work in less than 0.48 [s]. Note that the image does not represent this case in order to add generality.

The objectives can be succinctly summarized in a single table.

OBJECTIVE	**DESCRIPTION**
Time	The decoding process must to be executed in less than 0.48 [s]
Error probability	The FEC layer must ensure a correct delivery of the smallests packets

[3]If the script is understood as a finite state machine this problem can be avoided but others arise, e.g., automated theorem proving [12]. This problem is related to Gödel's theorems.

[4]Estimating complexity can be difficult and there is no consensus. For example, [16] scales its results to be similar to [14]. [15] is also similar to [14]. [14] is pessimistic.

State of the art

1948 **Shannon** [17]
Information Theory

1949 **Golay** [18]
Golay codes

1950 **Hamming** [19]
Hamming codes

1954 **Peter Elias** [20]
First approach to product codes

1955 **Peter Elias** [21]
Convolutional codes

1956 **RAMAC I** [22]
First interleaver

1957 **Sputnik** [23]
Successful first satellite launched

1959 **A. Hocquenghem** [24]
BCH codes

1960 **Bose & Ray-Chaudhuri** [25]
Independant BCH codes

1960 **Reed & Solomon** [26]
RS codes

1961 **Peterson & Brown** [27]
CRC codes

1962 **R. Gallager** [28]
LDPC codes

1965 **Forney** [29]
Concatenated schemes

1968 **Abramson** [30]
Product codes

1977 **Voyager spacecraft** [31]
Convolutional + interleaver

1982 **CD** [32]
RS protection

1993 **Berrou et al.** [33][34]
Turbo codes

1999 **Jordi Puig-Suari, Bob Twiggs**
CubeSat standard [4]

2001 **Divsalar y Pollara** [35]
ARA codes

2003 **Eurockot** [36]
First launch of CubeSats

2003 **DVB Project** [37]
BCH + LDPC

2004 **4G** [38]
Turbo codes

2008 **Arikan** [39]
Polar codes

2018 **5G** [40]
Polar codes + LDPC

Once the motivation has been described, it is important to list the possible options. The main purpose of this state of the art is to present chronologically all the ECC that are going to be studied and to establish a first approach to the corrector families. Some standards and protocols are also referred to show how powerful and useful ECC are.

This timeline begins with Shannon's masterpiece, the Information Theory. In this paper, Shannon announced his popular theorem:

> Let a discrete channel have the capacity C and a discrete source the entropy per second H. If $H \leq C$, there exists a coding system such that the output of the source can be transmitted over the channel with an arbitrarily small frequency of errors (or an arbitrarily small equivocation).(...). [17, p. 411]

This theorem is of fundamental importance because of it's implications. In brief, for a given binary rate (see [1.2]) and bandwidth, it's possible to obtain a maximum value for the permissible noise level.
Other researchers have made important contributions. For example, Forney demonstrated that concatenated ECC exponentially decrease the probability error while the decoding complexity increases only algebraically [29]. Other worthwhile highlights include the study of product codes, which enhance the correction capability.

To name just a few of the ECC listed: BCH (1960) discovered by Hocquenghem and independently by Bose & Ray-Chaudhuri a year later, or RS by Reed and Solomon in 1960.

Even some additional strategies are included i.e. interleaving. An interleaver scrambles the bits to spread possible errors throughout the message, and make the error correction easier.

This was originally used by RAMAC I HDD to prevent magnetic contamination during the writing process. In addition to the interleaver topic, the Voyager spacecraft demonstrated the reliability of convolutional and interleaver protection even when implemented on primitive integrated circuit technology. Some modern standards are also mentioned, such as 5G.

1.3 Document structure

The project is divided into four distinct phases phases. The first one is a preliminary evaluation, which generates a conceptual context that will be useful to the subsequent stages. The evaluation phase encompasses the introduction of theoretical concepts and different ECCs. One of these concepts is the coding gain of an ECC. The Appendix [C] collects all different graphs and presents the distinctive performances of each code, while the Appendix [D] presents the considered codes. The objective of this stage is to determine the family to which the code belongs. The second step is to identify the optimal code within the previously selected family. In this case, no appendices are needed. The third stage is the implementation, which consists of the study of how the code will be incorporated into the Hydra's satellites. An example of how the code works is provided, whose mathematical principles are detailed in Appendix [E]. Finally, the code is submitted under test to verify its functionality. The objective of this verification stage is to demonstrate that the work is technically correct.

2 Preliminar evaluation

Once all the ECC families have been introduced, it is necessary to select the one that best fits into the system. The main points relevant for the design of the correctors are code rate, the error correction capabilities and the complexity. All of the proposals have many strengths and weaknesses so it is important to discard those that do not meet all the requirements. Criteria coding rate and error correction capability can be found at [7, p. 14].

The FEC layer can be thought of as a black box. This black box is an ECC C(n,k) with an input of k bits and an output of n bits ($n > k$). The redundancy bits are the difference between these values and are placed as a function of the input bits. This function can be more or less complex and can give a better or worse error correction capability. The more redundancy is introduced, the better the information is protected [9, p. 41].

It is of paramount importance to define mathematically the manner in which the noise is understood. Noise is defined as a random pattern of bits. If no noise is added, the i-th position of the error pattern is zero, otherwise the error pattern is 1 [8, p. 420]. When a "1" is added to a message bit, it negates itself by changing its value. This happens with probability *channel BER* but will be discussed later. The channel model used interprets the bits as an arbitrary signal. Furthermore, Hydra operates with binary FSK, so these arbitrary signals are two sine waves at different frequencies. When the transmitter sends the signal, AWGN is added. This noise can confuse the first stages of the receiver. At this point, the FEC decoder has a stream of potentially erroneous bits. While the right model for the received signal can be an AWGN channel the model used for the decoder input should be a BSC [41, p. 11] for such a rough approximation, or an Binary-Input AWGN Channel [41, p. 14] if more accuracy is required. From here on, BSC will be the model of choice, as it is simply mathematical.

2.1 Error Correction Codes Performance

The coding rate indicates how many redundant bits a code has [7, p. 13]. This parameter is useful when it is necessary to compare different codes from the same family. This value is obtained by the following this equation, using the notation mentioned above C(n,k).

$$Coding\ rate = \frac{k}{n} \tag{2}$$

By definition, $n > k$, so the coding rate is always greater than zero and less than or equal to one. Let's assume that there is always information ready to be transmitted, so $k > 0$. If the coding rate tends to 0, this means that the output is much greater than the input, so high redundancy is applied. Conversely, if the coding rate is close to one, it means that n and k are similar, so the level of protection is low. This is useful to see how much protection is achieved by adding more or fewer bits. However, it is not useful for comparing between family codes because some of them can correct so much, but their complexity is so much higher. For example, BCH(31,16) has a codification rate of 0.516 while a turbocode(4108,2408) has a codification rate of 0,498. The reader might think that they have similar error correction capabilities due to similar redundancy sets, but in fact one of them is much more powerful than the other. Let's examine why.

Error correction capabilities can be measured in two ways: BER [8, p. 469] or coding gain [7, p. 15]. The BER checks how many bits are wrong by comparing the original and the received message. In practice, it is impossible to calculate the error rate because the receiver doesn't have the original message, but since it is being simulated, it is possible to access to all the points of the sender and receiver words. It is also necessary to define two types of error rate: the first one is the source BER and the second one is the source BER. The channel BER measures how often does the channel changes the bits while source BER measures how often does the ECC returns an incorrect message. Obviously, if more bits are affected (high channel BER), the receiver will miss the message more often. The reduction of the channel BER is proportional to the error correction capabilities. The figure [5a] shows a scheme for identifying both concepts.

Apart from that, it is possible to compare between the same code and other families although it is more tricky. This is possible with the coding gain. "The coding gain is the required E_b/N_0 to achieve a specified error probability of the coded system over an uncoded system with the same modulation and channel characteristics" [8, p. 409]. In other words, this parameter compares the studied code without the FEC layer (uncoded) and it usually represents E_b/N_0 against the source BER. The E_b/N_0 (EbN0) is the ratio between the energy of a bit and the noise. Despite of the impossibility of working in a world without noise, it is possible to apply different energies to each bit. If FEC is applied and it is pretended that the binary rate is constant, the EbN0 will decrease because the period of the bit will also decrease.

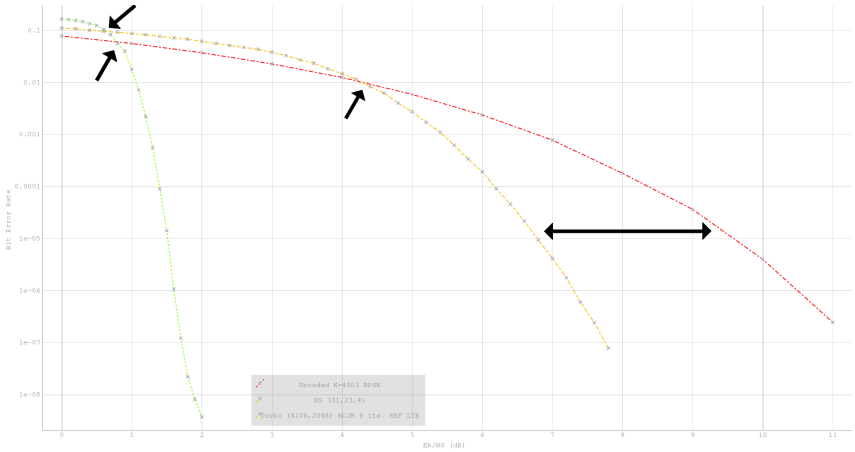
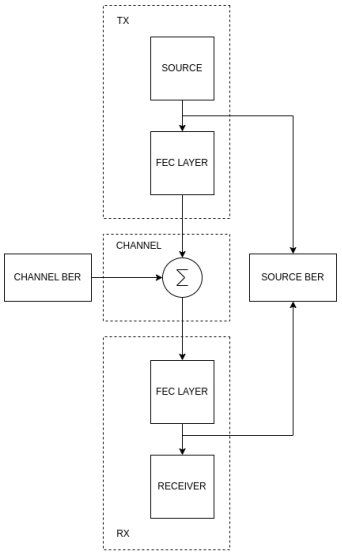


Figure 4: This figure shows many coding gain curves for three configurations: BCH(31,16), turbocoding(4108,2048) from LTE standard and uncoded [42]

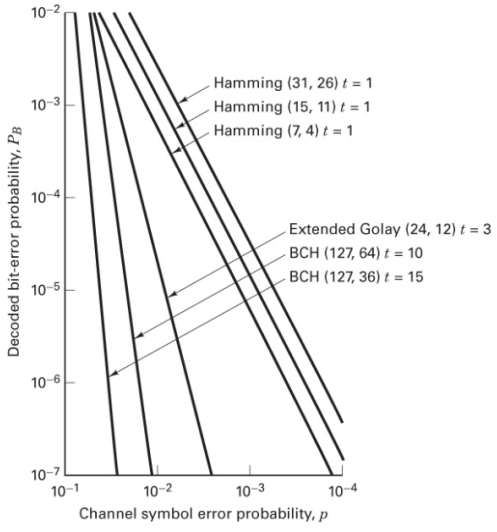
The graph has two interesting aspects: the intersection of the uncoded and coded cases and the slope[5] [7][8][9][41]. If EbN0 is too low, the messages will have too much noise to be decoded properly and the amount of errors will be greater than in the case where the FEC is not applied due to the errors in the redundancy. In this case, it is better not to apply any corrector, so that the bit period increases and becomes slightly higher. At what point are there fewer errors than in the uncoded case? The point of intersection. If the point of work is located on an EbN0 point higher than this point, it is better to apply ECC to reduce the error rate.

[5]The terms referred to in this paragraph are listed in the bibliography without explicit mention, but the terminology may change. This work follows [43] nomenclature.

Once it has confirmed that it is advantageous to correct errors, the main objective is to correct the maximum number of them. It is interesting to correct more and more errors as the EbN0 increases, in fact the increase in the ability to correct errors is not linear and depends on the ECC chosen. If we increase the EbN0 and a large number of errors are corrected, it is a great code because the slope of the graph is practically vertical. Typically, the ECCs with great slopes hides a high internal complexity. The figure [4] is an example of a coding gain graph of 3 cases, RS (yellow), turbo (green) and uncoded (red). The x-axis represents EbN0 and the vertical axis represents the source BER. The three simple arrows point to the intersection points. When EbN0 is less than 0.75 [dB] in the case of turbocoding, the probability of a bad correction is higher than the probability of no coding at all. This point is reached on RS at 4 [dB]. Even it is also possible to compare which is the best option between RS and turbocoding for low EbN0. This point is close to 0.5 [dB], but the number of errors is so considerable that it is not worth encoding, given the low probability of error by not encoding. It is important to note the high slope of the turbocoding curve. One may think that this decision comes down to seeing which code provides a better performance but, to illustrate the balance, the implementation of turbo coding is perceived to demand a considerable investment of computational resources when applied to the operational constraints of a small satellite. The double arrow is the graphical definition of the coding gain. It is usually defined against the unencoded case, but other schemes can be compared. Note how the slope becomes an important factor and a high value of it can obtain the same error ratio as the uncoded case with many fewer [dB].



(a) Channel and source BER scheme



(b) Example of real channel and source values. Note how the BER decrease after FEC [8, p. 470]

Figure 5: Channel BER and source BER

Another significant parameter is the Hamming distance between codewords. The Hamming distance, or simply distance, is defined as the number of bits that differ between two words. The minimum distance that a code is able to achieve is known as the minimum distance, which is usually related to the correction capabilities. The higher the minimum distance, the better the correction performance. The relationship between minimum distance and correction capabilities varies for each code, and thus this work will take the

error correction as the comparator element.

In addition to the concept of the minimum distance, another related concept is worthy of consideration. The probability of undetected error is the probability that the decoder is completely unable to detect the introduced errors. This occurs when the transmitter sends a codeword and the channel modifies it to seem like another codeword. The minimum distance is related to this probability because it dictates how many bits, at least, must be changed in order to obtain another codeword.

2.2 Possible enhancements of ECC

In addition to the ECC, there are other techniques which complement the capabilities of FEC. There are basically three: interleavers, concatenation and CRC.

2.2.1 Interleaving

The basics of interleaving were mentioned in [1.2], but will be discussed in more detail here. An interleaver is a device that scrambles the bits, symbols or blocks. Note that the interleaver does not add any redundancy and has no correction capabilities. Although it is tremendously simple, so why is it necessary? The main idea behind of interleavers is to protect against burst errors. Burst errors are a particular type of error pattern that corrupts many consecutive bits, while leaving others intact. Interleavers are useful against burst errors because they spread them across the codewords. When the receiver reorders the bits, the errors are distributed throughout the packet and each word has to correct itself. If the interleaver had not shuffled the bits, all the errors would be concentrated in a single word, exceeding correction capabilities. Figure [6] shows the commented effect. The borders of each rectangle represent the original position of each codeword, while the fill colours indicate the origin of the content. For example, in the packet "3 interleaved codewords", the first four elements are red, green, blue and red. However, it was originally the red codeword (see the red frame). The addition of noise, represented by black dots, results in the central elements to be corrupted, even so the corrupted blocks are distributed across the words: 1 damaged block for the red and blue codewords and two for the green codewords. Despite the same number of errors, without the interleaver, all the noise would have fallen on the green codeword, making it unrecoverable. The deinterleaver, which rearranges the bits, also moves the modified blocks making them correctable for the decoder.

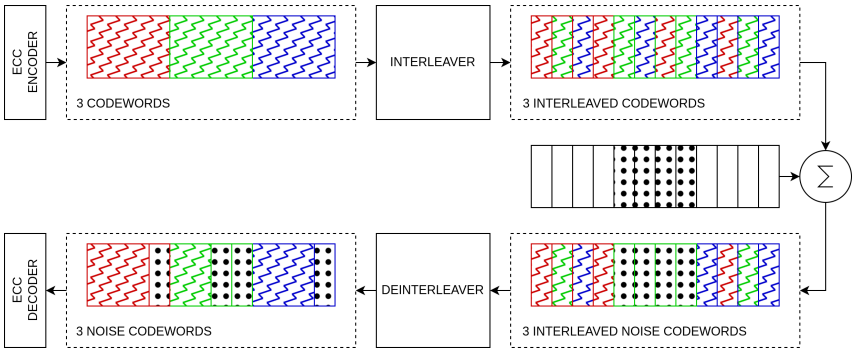


Figure 6: Illustrating interleavers

Note that the term block is used, not bits. This is because of the interleaver depth. This concept refers to the length of each block. The maximum depth is to reorder every

bit and the lowest depth is to take the codeword as a block and not change anything. The first case guarantees the highest level of protection, as opposed to the second, which does not protect the block. The price to pay of greater depth is the latency. All the bits must be present in each codeword; in order to be decoded, if the first bits are transmitted last, the receiver has to wait to deliver complete codewords to the decoder.

2.2.2 Concatenation

The concatenated schemes are a consequence of a simple principle: if it works, add more of them. There are two possible configurations, serial and parallel concatenation, which are typically referred to as single-level and multilevel concatenation, respectively.

Single-level concatenations were studied by Forney in 1966 and led him to the following conclusion:

> Concatenation of an arbitrarily large number of codes can yield a probability of error that decreases exponentially with the over-all block length, while the decoding complexity increases only algebraically [29, p. 3].

This means that the source BER can be reduced to any desired arbitrarily low value. What is more, the current BER decreases exponentially until it reaches the desired value, while the complexity is the result of adding each encoder complexity independently. This is due to the supercoder minimum distance, which is equal to the product of the minimum distances of the inner and outer coders. Figure [7] illustrates this structure.

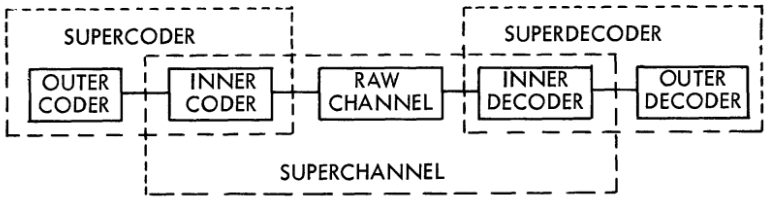


Fig. 2. Illustrating concatenation.

Figure 7: The presentation made by Forney of single-level schemes. Its terminology shall we respected [29, p. 10]

The superdecoder is reduced to an outer and an inner decoder, preserving the reflected structure of the superdecoder.

While this supercoder is simpler than multilever configuration (see [8]), the second arrangement provides "an even more powerful coding technique" and "more flexiblity in designing error-control systems" [7, p. 743]. The minimum distance of the output sequences is not well known yet. The decoding process is not as simple as for single-level codes, as it requires the codes to be decomposed and then decoded the individually. This increases latency and complexity. [7] proposes two decoders based on Trellis diagrams and iterative algorithms.

2.2.3 CRC

The CRC codes are a widely used error detection technique. It is really simple, the information is divided by a polynomial generator and the remainder is appended to the message. Consequently, if the generator is of degree n, the remainder will be constituted

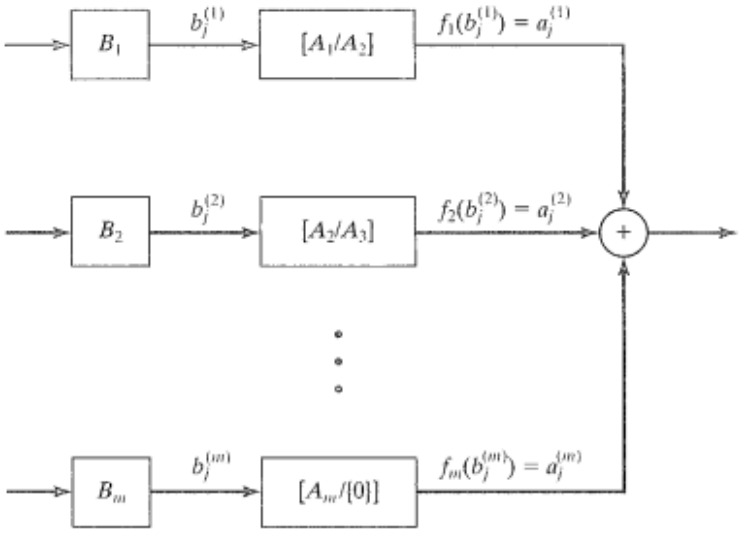


FIGURE 15.6: An encoder for an m-level concatenated code.

Figure 8: An encoder for an m-level concatenated code [7, p. 744]

by a maximum of n-1 output bits. Naturally, the greater the number of bits, the better the protection it provides.

There are several cryptographic systems which are essentially the same, a remainder, i.e. RSA. It is important to highlight that the CRC exclusively verifies the integrity of the data. The original input is still embedded in the final message and it can easily be modified to obtain the same CRC. This additional protection is useful against random errors. If the channel will hold sensitive information, then the information must be encrypted or hidden.

2.3 Error correction codes families

Previously, in [1.2], all the ECC of interest were presented. Since Shannon published his information theory, the ECC hasn't stopped growing. Prior to this, the only method of protecting information on the old telegraph lines was to repeat waveforms many times [44]. After 1948, mathematicians and engineers began to develop correction techniques, trying to reach the limit defined by Shannon's second theorem. The first approach was to use vector spaces over Galois fields, resulting in the family known as linear block codes. Other techniques were studied with the aim of complementing FEC capabilities like interleaving. Over time, other mathematical disciplines were applied, such as memory codes or iterative algorithms. The Appendix [C] presents various coding gain curves.

At this point, it is advisable to define what a systematic code is. Before the invention of the transistor, the electronic circuits were based on valves. These valves were slow and took up a lot of space. It was imperative to reduce the number of valves and electronic components in general, so engineers came up with an ingenious solution: embedding the word to be encoded within the coded word. When an ECC allows for systematic coding, it means that it is possible to embed the word in the output. Today, valves are completely obsolete, nevertheless systematic forms are still appreciated for the savings in computational complexity. With a bit of luck, it can even improve correction capabilities.

This happens when all the corrupted bits come from redundancy. Unless the packet is discarded, the data can be recovered without any problem. Other checks may be made to confirm the authenticity of the data, i.e. CRC. Typically the information is represented as polynomials. The idea is to represent binary data as the coefficients. If the i-th bit is "1", then the coefficient is "1" at that position. Systematic codes and data polynomials are detailed in figure [9].

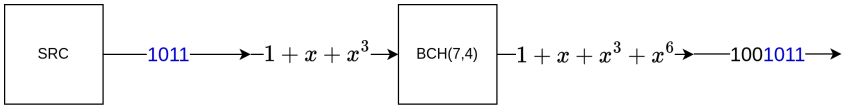


Figure 9: This figure explains how a polynomial can represent information

Let's define a binary string from the most significant bits to the least significant bits. The position of the ith bit (starting with 0) is the degree of the coefficient. It is highlighted in blue how data is embedded in a code word.

It is also worth mentioning the CCSDS standard. The Consultative Committee for Space Data Systems (CCSDS) aims to "enhance governmental & commercial interoperability & cross-support, while also reducing risk, development time & project costs" [45]. It is of the utmost importance to highlight that CCSDS is a recommendation, not a mandatory standard. In fact, they propose two independent codes, BCH (BCH(63,51)) and LDPC (LDPC(128,64) or LDPC(512,256), both of which have a code rate of 0.5)[46]. This standard will not be followed because the input size does not adapt to the system requirements.

2.3.1 Repetition

This is the simplest way to protect information. It consists of repeating the bits many times, the more times the better [44] [41]. It increases the binary rate in proportion to the number of repetitions. The decoder determines the transmitted bit by examining all the repetitions. The main advantage of this system is the simplicity of the system, however, the performance is too low in relation to the redundancy introduced.

2.3.2 Repeat - Accumulate and Irregular - Repeat - Accumulate codes

The repeat-accumulate (RA) codes are a modernised form of repetition codes [41, p. 142]. The information is divided into blocks of N bits and repeated q times. After this process, the result is scrambled by an interleaver and encoded by an accumulator. The accumulator works by adding the current bit to the previous output without modifying the first one.

The Irregular RA (IRA) codes replace the accumulator with a low density generator matrix code [47]. The reason for this change is the way the redundancy is built and to make it easier the design to adapt the code to the channel requirements. IRA codes repeat information many times, accumulate subsets and finally generate redundancy via a parity matrix. By omitting many redundant bits, it is possible to configure how much protection is needed. Iterative decoding is allowed on IRA codes. An algorithm is iterative if and only if the output returns to the input and converges to the theoretical performance bound defined by Shannon or the ECC mathematics.

These codes are very similar to convolutional and LDPC codes. They can be decoded with the same receivers. The difference lies in their simplicity, which is a case in point

in their class. In fact, there is an ongoing legal dispute as to whether DVB-S2 LDPC constitutes an IRA code [48].

Both of them are considered as easy to encode and have reasonable decoding complexity with very little coding gain. Both also have systematic forms.

2.3.3 BCH

BCH codes are constructed on a vector space, so they are considered to be a linear code block [7, p. 194]. In fact, this vector space is based on Galois fields, which gives these codes some special properties. They are characterised by a generator polynomial, $g(x)$, that allows a precise control of redundancy and the number of correctable errors.

The encoding process is quite simple. It can be reduced to the multiplication of binary data by the polynomial generator. If systematic coding is required, it is sufficient to shift the input bits $n - k$ positions and apply a few operations of equal complexity.

There are many decoding algorithms, but they all pretend to solve one equation: the error locator polynomial. First of all, the syndrome is calculated. The syndrome is a binary vector of $n - k$ bits. If the syndrome is 0, it means there are no errors and no further calculations are required. If it is not zero, the next step is to determine the error locator polynomial and its roots. Finally, the locations of the errors are related to the roots. The most expensive stage is the construction of the error polynomial. There are many ways to solve the error locator polynomial: extended Euclidean, standard array, Forney, etc. Some of them are better to run on hardware, while others show better performance when implemented in software.

Golay codes are included as linear block codes and are a specific case of BCH [18]. When standard array decoding is applied, this code is perfect, but this means that it fits perfectly into the standard matrix, not because Berkelamp described the paper as "the best single page published page" [49, p. 4]. The standard matrix is constructed using this decoding method to identify and rectify error patterns.

These codes offer a reasonably high coding gain with low to medium redundancy. Computational complexity is moderated.

2.3.4 Reed - Solomon

Reed-Solomon (RS) codes are a further development of BCH codes, as they are based on grouping bits on symbols [9, p. 115]. As a consequence, the input and output sizes are typically larger and they have better error correction. However, this enhanced performance comes at the cost of increased complexity.

Mathematically, BCH codes are constructed over a Galois field (GF) of two symbols, 1 and 0. In contrast, RS codes expand this Galois field to 2^m, where m is the number of bits per symbol. Despite of these differences, the operations are equal to BCH with the exception that instead of working with bits, they are done with symbols.

Decoding RS codes is similar to BCH. There are syndromes and many algorithms involved. The primary distinction between the two lies in the utilisation of two polynomials: an error locator and an error value polynomial. These mathematical differences can be seen in the decoding algorithms: Berkelamp, extended Euclidean or frequency domain decoding. RS codes offer a medium level of complexity and coding gain. Naturally, RS can be systematic too.

2.3.5 Convolutional codes

Convolutional codes diverge from the linear block codes and Galois fields [9, p. 157]. They are essentially memory codes, in other words, the output depends on the current input and the previous ones. The software may be difficult to implement due to the labyrinthic scheme of codecs. By comparison, they are really easy to implement with pure electronics. The encoder consists of in many registers where data is flowing among them. The greater the number of the registers, the greater the number of redundancy bits that are added. Furthermore, convolutional codes can be defined with a polynomial, which can be a useful characterisation. There are more useful ways to identify them, for example state diagrams, trellis diagrams or the transfer function. Since the input is a bit stream of variable size, it makes no sense to talk about output sizes. Instead, it is described with the code rate and the number of positions (one word) of the convolutional code, since for any given register state the redundancy size is fixed.

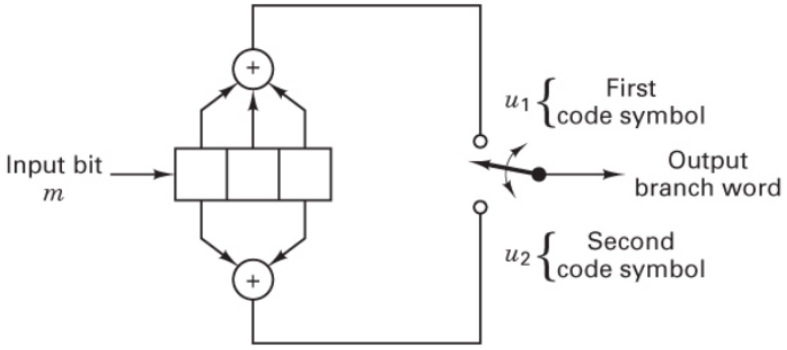


Figure 7.3 Convolutional encoder (rate $\frac{1}{2}$, $K = 3$).

Figure 10: Convolutional(1/2,$K = 3$). K always matches with the number of registers [8, p. 379]

There are many paths to decode these codes. One of the most common is the Viterbi algorithm. This decoding algorithm is closely related to graph theory, and there are versions that attempt to decrease the complexity, for example the soft output Viterbi algorithm. Other example of decoding algorithms is BCJR. Due to the fact that the decoding computations cost increase exponentially, there are also suboptimal decoders such as ZJ sequential decoding or Fano decoder.

Convolutional codes are very complex and can be difficult to implement, especially when systematic coding is required. The code rate of these codes is so high so high that they are usually punctured. Puncturing is the process of extracting some bits known by the transmitter and receiver to decrease the binary rate, adjusting by this way the coding rate. Puncturing supposes deliberate introduced errors which diminish correction capabilities.

2.3.6 Turbo codes and turboproduct codes

Turbo codes are fictitious codes in the sense that they are a concatenation of the previous ones [41, p. 298]. Originally, the design included many convolutional codecs with interleavers to differentiate their inputs in a multilevel concatenation scheme.

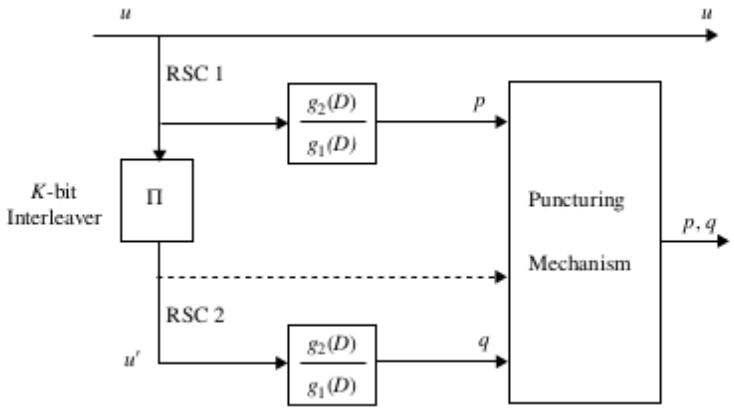


Figure 11: Most basic scheme of turbocodes. Π represents the interleaver [41, p. 299]

Initially, turbo codes were proposed with convolutional codes with a trellis decoder. The encoders, the interleavers and the puncturer must fulfil certain requirements which make the design difficult. The most extended decoding algorithms are the PCCC and SCCC. This is an iterative decoder with a high level of complexity. It is so complex that there are even some approaches with lower complexity and capabilities.

Turbo codes are high complexity codes with very high coding gain. As it can be seen in figure [11], they are typically systematic.

The turboproduct codes are the result of the multiplications between, typically, extended BCH codes. A code can be extended by adding parity bits, but this will never improve its performance. Instead of appending each data and redundancy string, the product of codes built a matrix, as it is illustrated in figure [12].

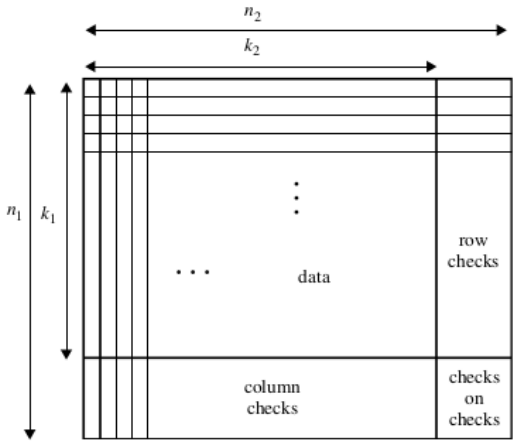


Figure 12: Turboproduct codes matrix [41, p. 329]

2.3.7 LDPC codes

LDPC codes are based on graph theory and can be fully characterised by a graph or a matrix. Tanner generalised LDPCs by introducing the Tanner graphs in 1981, after 35 years of complete neglect. MacKay also studied these codes using parity check matrices.

The main innovations of LDPC codes are the low-density matrices, which facilitate the iterative decoding. LDPC codes are close to be optimal, similar to turbo codes.

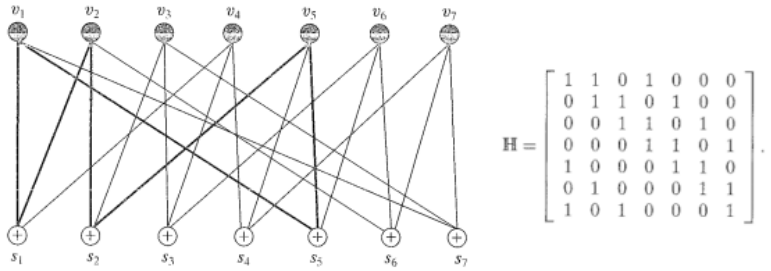


FIGURE 17.6: The Tanner graph of a (7, 3) linear block code.

Figure 13: Tanner graphs and its correspondent parity-matrix for LDPC(7,3) [7, p. 869]

The coding is easily done by multiplying the input bits and the parity matrix. The decoding process can be extremely complex, two algorithms are widely employed: the SPA or the BPA, which is iterative. LDPC are codes that are easy to implement in software through concurrency, but they are difficult to implement in hardware.

2.3.8 Polar codes

Polar codes divide the input bits into two blocks and apply a specific transformation to each part [41, p. 339]. This transformation selects many bits to be converted into a sequence with a lower error ratio, while the others are turned into the redundancy that helps in the correction.

The most popular decoding algorithms are Successive Cancellation (SC) and Successive Cancellation List (SCL). Their complexity is high, but, SC performs best when the error rate is low. The other algorithm is better in high error rate environments. Polar codes can be systematic.

2.4 Conclusions

As previously seen, this stage of the design process carries a heavy theoretical burden. The main concepts of this block are the characterisation of the ECC codes, their improvements and the ECC codes themselves. Figure [14] summarises these ideas.

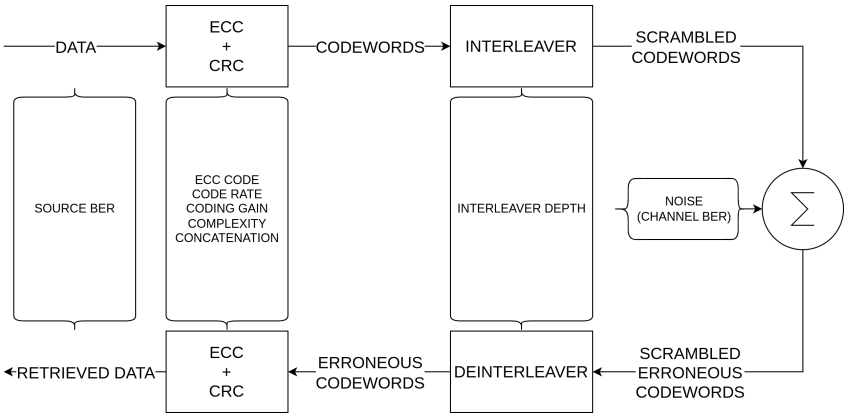


Figure 14: General FEC scheme. All concepts are listed here

The table [1] gives an overview of the evaluation. Red words discard the proposed codes. "CONV" means CONVolutional and "TPC" TurboProduct Codes.

Repetition codes are too simple to apply to CubeSats. Their performance is extremely low, which is the main reason why they are currently deprecated. Repetition codes are to ECC codes what fire is to a modern civilisation. It works, but something more sophisticated is needed. Their evolution, the RA and IRA codes, are being discarded for similar reasons.

BCHs are good candidates. Their balance between complexity and coding gain makes them a good choice, however, on their own they do not meet the requirements. It will be necessary to complement them with an interleaver or concatenated codes. Their large number of input sizes is variable, which reduces the problem of finding the right code.

RS codes are even better than BCH codes. By paying an addition to complexity, it is possible to guarantee an extra [dB]. While BCHs work with binary Galois fields, RS extend them to symbols of m bits. Their capacities are refereed to these symbols. This is the reason why if a burst pattern damages the information, the RS can resolve it. In other words, they can act as a false interleaver in the FEC layer scenario. Logically, the greater the number of bits per symbol, the greater the burst weight they can correct. Conversely, if the number of bits per symbol increases, so does the input size of the code. A code with too much padding is not efficient because the transmitter sends useless streams and the redundancy protects information that is automatically discarded instead of protecting the really important data.

Convolutional codes are great codes but Hydra's requirements reject them. On one hand, they have better timing performance on hardware devices, and the electronic boards are pre-designed. Building new electronic devices would mean a step backwards. There's also no room for new chips and it would suppose a reduction in payload for their customers. They have a high level of complexity compared to the available resources. On the other hand, there's another historical reason for avoiding the exclusive use of convolutional coding. In the 1970's, the NASA standard included an RS code in order to improve convolutional capabilities. NASA have been improving its FEC from the Reed - Muller codes (1969 - 1977, Mariner) to turbocodes and LDPC. This relieved the concatenated schemes. The Voyager spacecraft was the first one to concatenate a convolutional with RS, allowing the DSN to receive colour images of Uranus and Neptune taken by Voyager 2 [50] [51]. For its part, Voyager 1 took the famous picture "a pale blue dot" with these codecs too. It is well known that launching satellites is expensive and it is imperative to reduce the probability of error, so if an important and experienced organisation like NASA decided that convolutional codes weren't good enough. Hydra would then have reasons to suspect on convolutional codes.

Turbo and turboproduct codes employ concatenated simpler codes such as convolutional or BCH codes, proving effective in challenging channel conditions and nearing Shannon's limit. Despite their excellence, they demand significant computing resources, reflected in their integration into the 5G standard [52]. However, their decoding complexity and time per iteration exceed those of other codes, requiring at least 8 iterations for high rates, posing an ongoing challenge [53] [54] [55]. Research focuses on parallel computation and improved algorithms, while standards shift toward simpler ECC, adopting turbo codes in 4G and polar codes in 5G.

Table 1: Summary of the ECCs studied and the reason for discarding them

CODE	CODING RATE	CODING GAIN (SOURCE BER 10^{-5})	COMPLEXITY RELATED TO CODING GAIN	COMMENTS
RA	Variable	low (1)	low.	Simple codes with worst performance
IRA	Variable.	low (2.5)	High.	Specific case of LDPC
BCH	Variable (typically 0.5)	medium (1.75)	Low	Well known mathematical properties
RS	Variable	medium (2.75)	Low	RS works with symbols. This can helps on burst errors
CONV	Difficult to adjust	medium (2.5)	Medium - high	Best implemented on hardware
Turbo	Difficult to adjust	Very high (8)	Very high	Enhanced convolutional by multilevel concatenation
TPC	Difficult to adjust	Very high (6)	Very high	Not well known properties They preserve product codes properties
LDPC	Difficult to adjust	Very high (5.5)	Extremely high	Large inputs

LDPC codes are being discarded due to their complexity. These codes require an expensive computational decoder. There are also other considerations, such as the fact that this ECC works better at high input rates [56]. At the present time, there are just two codes that can be implemented: BCH and RS. Both of them have insufficient coding gain to be implemented alone, therefore some improvement is required.

The interleaver is designed to make the system against burst errors, but our channel is memoryless, in other words the apparition of one error does not supposes an increase in the probability of another error occurring. This does not imply that burst errors cannot arise; rather, the probability of such errors decreases because they have to appear in close proximity to the initial error. All of them have the same probability of being damaged, so it would be unusual to modify these specific bits. In conclusion, the interleaver does not improve the situation.

The only enhancement which can achieve a better coding gain is a concatenate scheme. The integration of multilevel schemes into the system would be unfeasible due to the necessity of incorporating post-decoding logic. Consequently, single-level schemes are the only available option then.

The next step is to determine which one should work as inner and outer code. This question is easily answered by studying RS codes. As previously mentioned, RS codes works with symbols, the larger the symbol size the better correction capabilities. Given that RS codes correct symbols and not bits, the correction is more effective when the

Table 2: Different ECC in space missions [57]

YEAR	EMPLOYED CODE	MISSION
1958–present	Uncoded	Explorer, Mariner, many others
1968–1978	convolutional codes (CC) (25, 1/2)	Pioneer, Venus
1969–1975	Reed-Muller code (32, 6)	Mariner, Viking
1977–present	Binary Golay code	Voyager
1977–present	RS(255, 223) + CC(7, 1/2)	Voyager, Galileo, many others
1989–2003	RS(255, 223) + CC(7, 1/3)	Voyager
1989–2003	RS(255, 223) + CC(14, 1/4)	Galileo
1996–present	RS + CC (15, 1/6)	Cassini, Mars Pathfinder, others
2004–present	Turbo codes	Messenger, Stereo, MRO, others
est. 2009	LDPC codes	Constellation, MSL

symbols are large. There are two significant drawbacks. Firstly, the inputs need to be larger. Secondly, a symbol can be damaged easier when the number of bits per symbol is higher. In light of these considerations, it can be posited that the RS code will serve as the inner code, while the BCH code will function as the outer code. The redundancy of the BCH can be large enough to be turned into the RS input so the first problem is solved. Subsequently, the design stage must identify a balance between symbol size and corrective capacity.

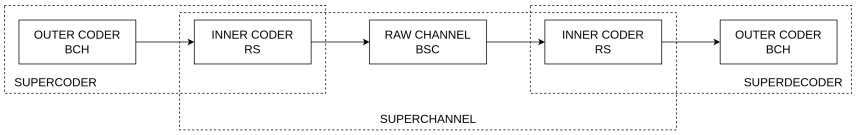


Figure 15: Proposed final scheme

3 Design

Figure [15] illustrates the final proposed scheme and in 1.1 the traffic was described: packets whose size encompasses from 12 bytes to 32 bytes. The smallest packet is the most probable with the odds decreasing almost exponentially to the largest packet. The aim of this stage is to determine the best size of outer and inner codes to fit properly with the input size and between the BCH and RS, while ensuring the minimum padding. The padding is regarded as information to the encoder when it is not. It is just 0 which are added to fit in the aforementioned situations. The considered BCH and RS are listed in the appendix [D].

The diagram [16] illustrates the scheme of the implementation.

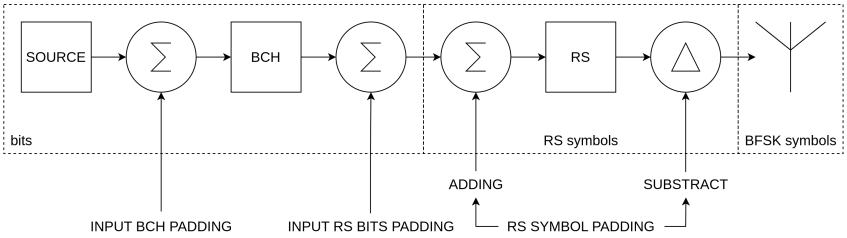


Figure 16: Padding scheme

In order to facilitate the subsequent analysis, it is necessary to define the BCH dimensions as BCH_N and BCH_K and the RS dimensions as RS_N and RS_K. The number of bits per symbol is m.The extensive range of BCH and RS couples makes it impossible to study each combination. There are also 21 different cases, one for each packet length. For that reason, the comparison between codes will be provided with statistical parameters. Specifically, the mean value and the standard deviation will be provided. The mean value represents the average of the data set, while the variance indicates the extent to which the values differ from that average.

It is reasonable to posit that no BCH and RS couple will be found that is perfectly suited to our system, due to discrepancies between data, BCH and RS's input sizes. This is the reason why, foreseeably, it will be necessary to add padding. The objective in this stage is to add the minimum amount of padding. In order to find the balance previously mentioned in 3.2, it will also be necessary to encode many times, which defines the number of necessary words. It is preferable to minimise the count of words as much as possible for the purpose of reduce the use of the encoder and, in particular, the decoder.

Firstly, the data must be adapted to a multiple of BCH_K with an input padding. If the input data already comply with that condition, the input padding will be 0. The number of words will be defined as the quotient between the input data size plus the padding and BCH_K. The output is then calculated as the number of words multiplied by BCH_N.

Secondly, it is necessary to adapt the BCH's output to the RS's input. As a consequence of the employment of symbols by RS, two substeps are required: the adjustment to symbol size and then the adaptation to the RS's input size. The first one introduces padding bits to convert the stream into a multiple of m, while the second one transforms the BCH-protected data into a compatible RS input. Once all the RS padding is added, the result shall be a stream of symbols multiple of RS_K. The number of words is obtained in a manner analogous to that employed for BCH. If a chain of bits is required,

it is straightforward to convert from decimal to binary. Furthermore, it is possible to regroup the information into the number of symbols dictated by the modulation.

Finally, it is possible to remove some padding in a an intelligent manner. All these codes are systematic, so data is embedded into the final string. This implies that, as long as the original input size is transmitted, it is possible to remove the padding. In theory, all the padding can be removed, nevertheless, transmitting the original data size may affect to other system layers. What is more, padding can be used as a sponge, if it results damaged by the channel, the receiver does not care because it will be discarded. Furthermore, it can even be utilised as a random source of information in order to enhance entropy in cryptographic systems [58]. Due to limitations in the extension of this document, this idea and the last Fermat's theorem will not be developed further.

3.1 BCH

Given the extensive range of potential values, the BCH will be defined initially. It should be noted that determining the inner code at this point will not be conclusive.

3.1.1 First criteria: bytes

The most critical criteria will be firstly applied to open the way to fine details. Given the byte-sized packets, it is logical to conclude that the BCHs whose inputs are multiple of 8 are simpler to implement than other codes. Adherence to this directive would result in the number of viable candidates being truncated from 206 to 5. The smallest code is BCH(7,4), which is not strictly a multiple of 8 but it can code a byte without padding. The largest code is BCH(127,8).

3.1.2 Second criteria: code rate

As has been said, the incorporation of redundancy provides enhanced protection and consequently drives demand for more binary rate. This conflicts with the Hydra's transmission time, therefore the reference code rate will be 0.5. In other words, a balance between redundancy and useful throughput. This criteria dismisses all the codes except the two smallest, BCH(7,4) and BCH(31,16).

3.1.3 Third criteria: number of words

Finally, the number of words is of significance, as it determines the number of times the decoder will be employed. Regardless of the chosen code, the decoder will be the same, so it is a waste of time decoding so many words. This implies that the only and remaining code is BCH(31,16).

3.2 RS

Once the outer code has been defined, the subsequent step now is to determine which RS will be concatenated. The order and first criteria are modified due to the nature of these codes.

3.2.1 First criteria: selection of Galois field

The size of the output bits is quite variable, however, it is important to remember that the most probable packet lengths are 96 bits, so a encoded packet has a load of 186 bits. The BCH has a strange property which is really useful: the padding is added if the number of input bytes is odd. Otherwise, no padding is added. This results in the output bits being equal in pairs. For example, the output rate of packets with 13 and 14 bytes is the same, 217 bits. Then one word is added, resulting in the next packets differing only in one word, 31 bits. Unfortunately, the output rates of the BCH codes are defined by the same equation that defines the set of the Marsenne numbers, $2^p - 1$. The Marsenne numbers have a high probability of being prime so unless other BCH is used, there will not be no m that will take of the BCH's output without the need to add padding.

Since the output always differs by 31 bits, it is logical to think in terms of the divisors of close numbers: 30 and 32. It is plausible to think in two values, 5 and 6 because they offer an obvious balance. The smallest RS uses 3 bits per symbol, which is mathematically possible but not practical due to the large number of small symbols produced. The largest RS would produce so few symbols that it would be translated into huge inputs, mathematically valuable but impractical.

3.2.2 Second criteria: number of words

The primary challenge in working with symbols is to regulate the output rates: for small inputs, the output will be small, but a larger input will result in a significant output. This does not completely rule out the RS(63,x) codes ($m = 6$), but they are not the preferred option. In this case, as Henry Fonda said, it is a reasonable doubt.

From here on, exclusively the codes with five bits per symbol will be taken into account. The number of input bits varies from 75 to 145, thus the range of the number of words is defined between 4 and 7. This is a reasonable range for these low-medium complexity codes.

3.2.3 Third criteria: code rate

Once the code has been identified as RS(31,X), it is necessary to determine which of these codes is the most optimal. Table [3] shows the output rate with the different RS codes.

As illustrated in figure [17], there is no clear preference in terms of coding gain. The inner code will be the RS(31,23). It is a balanced code with less redundancy than the BCH code, but the correction capabilities are better because it can correct 4 symbols of 5 bits each (the BCH can correct 3 bits). In addition, the number of padding symbols is lower for the smallest packets.

Table [3] demonstrates that there is no clear difference in terms of padding between the family RS(31,X). The columns headed by "BCH - RS COUPLE" specifies the studied codes. The next columns of the upper table represent the mean values of the added padding. Although, the details are widely explained on [3.3], it is possible to see how the "CROPPED OUTPUT" column values increases, more or less, constantly. The supercoder rate is the code rate of the concatenation and its defined as the product of each code rate. Furthermore, a measure of efficiency is provided. The efficiency is defined as the relation between the information and redundancy.

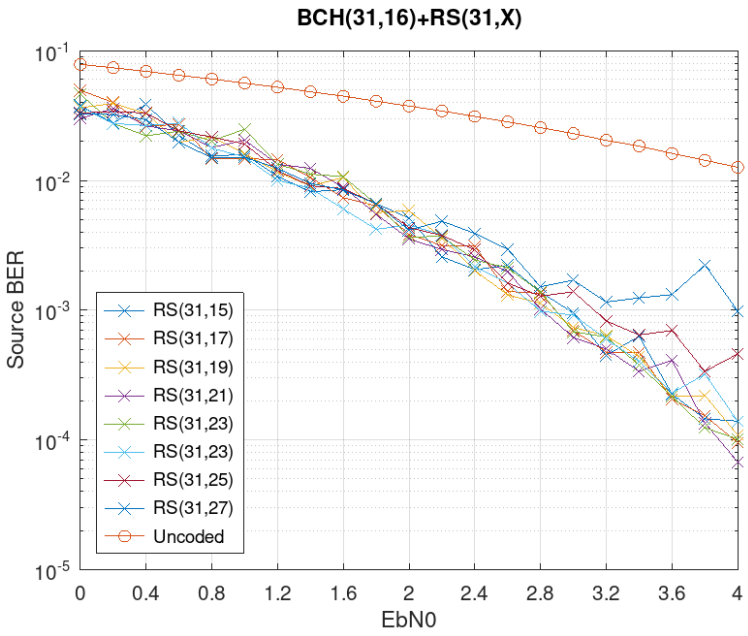


Figure 17: Comparative between BCH(31,16)+RS(31,X)

Table 3: Comparative between BCH(31,16)+RS(31,X). There is no code that stands out.

BCH – RS COUPLE									PADDING			
BCH (N,K)				RS (N,K)					BCH	RS		
N	K	T	CODE RATE	N	K	M	T	CODE RATE	COMPLETE BCH WORD	SYMBOL PADDING (b)	COMPLETE RS WORD (b)	TOTAL RS (b)
31	16	3	0,516	31	29	5	1	0,935	3,810	2,095	12,762	65,905
31	16	3	0,516	31	27	5	2	0,871	3,810	2,095	12,190	63,048
31	16	3	0,516	31	25	5	3	0,806	3,810	2,095	8,476	44,476
31	16	3	0,516	31	23	5	4	0,742	3,810	2,095	10,952	56,857
31	16	3	0,516	31	21	5	5	0,677	3,810	2,095	9,905	51,619
31	16	3	0,516	31	19	5	6	0,613	3,810	2,095	7,714	40,667
31	16	3	0,516	31	17	5	7	0,548	3,810	2,095	8,429	44,238
31	16	3	0,516	31	15	5	8	0,484	3,810	2,095	6,333	33,762

PADDING	BCH – RS COUPLE									OUTPUT			
TOTAL (b)	BCH (N,K)				RS					OUTPUT (b)	CROPPED OUTPUT (b)		
TOTAL (b)	N	K	T	CODE RATE	N	K	M	T	CODE RATE	OUTPUT (b)	CROPPED OUTPUT (b)	SUPERCODE RATE	$\frac{INFO}{REDUNDANCY}$
69,714	31	16	3	0,516	31	29	5	1	0,935	442,857	379,048	0,483	0,397
66,857	31	16	3	0,516	31	27	5	2	0,871	472,381	411,429	0,450	0,371
48,286	31	16	3	0,516	31	25	5	3	0,806	487,143	444,762	0,416	0,360
60,667	31	16	3	0,516	31	23	5	4	0,742	546,190	491,429	0,383	0,323
55,429	31	16	3	0,516	31	21	5	5	0,677	590,476	540,952	0,350	0,297
44,476	31	16	3	0,516	31	19	5	6	0,613	634,762	596,190	0,316	0,277
48,048	31	16	3	0,516	31	17	5	7	0,548	715,952	673,810	0,283	0,245
37,571	31	16	3	0,516	31	15	5	8	0,484	789,762	758,095	0,250	0,222

3.3 Padding in the selected code

The padding is constituted by the additional bits which are appended to fit the data to each codec. It does not carry any information and, in this context, does not confer any advantage; therefore, it should be avoided. 16 illustrates how the system handles the padding. To simulate different BCH and RS couples, an Excel sheet can be a very

valuable tool. The proposed sheet takes as input BCH_N, BCH_K, RS_N, RS_K and m.

Firstly, it is necessary to add some bits to fit the data into a multiple of BCH_K. The quotient between the information size plus the padding and BCH_K yields in the number of codewords. Then, the encoder incorporates the redundancy and the information has turned into a stream with a length equal to the product of the number of words and BCH_N. A distinctive property of this code is derived from the relation between to the packet size and BCH_K. If the original data length is an even multiple of 8, the byte size, then the padding is not necessary. As corollary, if the original data length is an odd multiple of 8, then the information fits if and only if a byte is added as padding. There are no exceptions to this rule.

Table [4] provides an overview of the input data. The first and the second columns indicate the input size in bytes and bits, respectively. The third column displays the number of codewords that will be produced. The next column illustrates the previously commented property: if the input is odd, then it is mandatory to add an extra padding byte. On average, the encoder and decoder are employed from 8 to 13 times. It is crucial to acknowledge that the presented data does not accurately reflect the reality, as the precise probability distribution of the packets remains uncertain. The provided values assume an uniform distribution.

Table 4: Selected code in ciphers: BCH input

INPUT (B)	**INPUT (b)**	**BCH WORDS**	**PADDING BCH (b)**	**INPUT TO THE BCH ENCODER (b)**
12	96	6	0	96
13	104	7	8	112
14	112	7	0	112
15	120	8	8	128
16	128	8	0	128
17	136	9	8	144
18	144	9	0	144
19	152	10	8	160
20	160	10	0	160
21	168	11	8	176
22	176	11	0	176
23	184	12	8	192
24	192	12	0	192
25	200	13	8	208
26	208	13	0	208
27	216	14	8	224
28	224	14	0	224
29	232	15	8	240
30	240	15	0	240
31	248	16	8	256
32	256	16	0	256
MEAN		11,238	3,809	179,809
VARIANCE		2,500	4,000	40,000

Table [5] provides a photograph of the manner in which the encoder handles with the padding. The first column represents the output of the BCH encoder. These values are always a multiple of BCH_N, although not necessarily of RS_K, in bits, not symbols. It is now necessary to add bits and symbols to fit the BCHs output to RS_K symbols. The RS(31,23) groups the bits into symbols of five bits. Consequently, it is necessary to

Table 5: Selected code in ciphers: RS input

	BCH OUT (b)	SYMBOL PAD (b)	INPUT TO RS (S)	RS WORDS (S)	COMPLETE WORD (S)	COMPLETE WORD (b)	TOTAL PAD RS (b)	RS IN (S)	RS IN (b)
RS IN	186	4	38	2	8	40	44	46	230
	217	3	44	2	2	10	13	46	230
	217	3	44	2	2	10	13	46	230
	248	2	50	3	19	95	97	69	345
	248	2	50	3	19	95	97	69	345
	279	1	56	3	13	65	66	69	345
	279	1	56	3	13	65	66	69	345
	310	0	62	3	7	35	35	69	345
	310	0	62	3	7	35	35	69	345
	341	4	69	3	0	0	4	69	345
	341	4	69	3	0	0	4	69	345
	372	3	75	4	17	85	88	92	460
	372	3	75	4	17	85	88	92	460
	403	2	81	4	11	55	57	92	460
	403	2	81	4	11	55	57	92	460
	434	1	87	4	5	25	26	92	460
	434	1	87	4	5	25	26	92	460
	465	0	93	5	22	110	110	115	575
	465	0	93	5	22	110	110	115	575
	496	4	100	5	15	75	79	115	575
	496	4	100	5	15	75	79	115	575
MEAN	348,381	2,095	70,095	3,524	10,952	54,762	56,857	81,048	405,238
VAR	77,500	1,200	15,540	0,795	6,075	30,375	30,175	18,285	91,425

add some bits to convert the BCHs output into a multiple of m. Once the number of bits indicated in column "SYMBOL PAD (b)", it is possible to work with symbols. The amount of symbols after the conversion is defined in the second column "INPUT TO RS (S)". Analogous to the BCH, the input, now in symbols, must be a multiple of RS_K, which it is not currently. It is important to highlight that the padding introduced here is significantly greater than that which was previously incorporated in previous instances, due to its symbol nature. Once the number of symbols is known, it is possible to obtain the number of words, which is indicated in column "RS WORDS (S)", by finding the next higher multiple of RS_K. It is also trivial to find how many symbols are needed to complete the word as indicated in the column "COMPLETE WORD (S)". To facilitate the comparison between the amount of previous padding bits and the padding introduced to fit symbols, it is useful to express this last column in terms of bits. Theoretically, the BCH requires less padding than 15 bits and to fit into symbols, less than 4 bits, while the RS requires less than 110 bits. In other words, the RS requires six times more padding. In practice, this value is even higher, as the BCH padding is 8 or 0 bits. The total amount of padding bits to fit the RS into the system is reflected on column "TOTAL PAD RS (b)". The penultimate and the last column express the input to the RS encoder as symbols and bits, respectively.

The final step is an idea that has been applied in the standard DVB-S2. As previously discussed, the systematic codes embed the information into the codeword. This implies that, despite of not being information, the padding is embedded into the supercodeword and can be easily punctured. The receiver should know how much padding has been removed in order to add it by its own. It is of paramount importance to determine how much padding had been removed because the redundancy depends on it. It is therefore necessary to send the original data size also. It is worth noting that the padding is not

useless; for example, if the receiver measures the signal, it can be useful to have a larger known string (including training sequence) to obtain BER values which are statistically more accurate. What is more, as previously indicated, it can be beneficial to generate entropy in cryptosystems. Such enhancements will be considered for future lines.

Table 6: Selected code in ciphers: RS output

	OUTPUT (S)	**OUTPUT (b)**	**TOTAL PADDING (b)**	**CROPPED OUTPUT (b)**	$\frac{INFO}{REDUNDANCY}$
RS OUTPUT	62	310	44	266	0,310
	62	310	21	289	0,335
	62	310	13	297	0,361
	93	465	105	360	0,258
	93	465	97	368	0,275
	93	465	74	391	0,292
	93	465	66	399	0,310
	93	465	43	422	0,327
	93	465	35	430	0,344
	93	465	12	453	0,361
	93	465	4	461	0,378
	124	620	96	524	0,297
	124	620	88	532	0,310
	124	620	65	555	0,323
	124	620	57	563	0,335
	124	620	34	586	0,348
	124	620	26	594	0,361
	155	775	118	657	0,299
	155	775	110	665	0,310
	155	775	87	688	0,320
	155	775	79	696	0,330
MEAN	109,238	546,190	60,667	485,524	0,323
VARIANCE	24,645	123,225	30,850	108,900	0,025

Table [6] represents the output of the RS encoder. The first and the second columns represent the output of the encoder in symbol and bits, respectively. The total amount of padding which has been integrated among the stages is reflected in the column "TOTAL PADDING (b)". Finally, the output is the difference between the original output and the padding. Additionally, a measure of efficiency is proportioned. The transmitter sends, on average, between 378 and 493 bits, however, this values assumes an uniform probability distribution of the packets.

3.4 Conclusion

Linear block codes offer a favourable relationship between efficiency and effectiveness. They typically exhibit a balanced coding rate, and are well known mathematically. These codes suppose a reliable option when the complexity and retransmissions incur a significant cost in terms of time and energy. Independently of the correction capabilities, this

ECC can detect any error which is not a codeword easily with the syndrome. In particular, the BCH(31,16) adapts very well to this application due to its inherent characteristics. Given that the input rate is two bytes, then the input padding is reduced, simplifying the implementation. Despite the impossibility of finding a RS which matches perfectly with this code, the surrounding possible ciphers have many divisors which facilitate the election of the RS family.

In contrast, the RS codes are the best in correcting burst errors. The substitution of symbols in place of bits enhances the correction capability and provides a robust initial defence against potential corruption. The satellites, which can rotate and lose the signal momentarily, introduce then a burst error that can be easily handled. Furthermore, thanks to the low binary rate, the number of damaged bits is low enough to be corrected completely. In this case, the election of the RS(31,23) is difficult because there is no benefit accentuated. Figure [17] illustrates the comparable coding gains of the concatenated scheme and Appendix [3.3] the padding that is introduced by each of them. Finally, the election was made in order to achieve balance as previously explained.

The concatenation of both codes yields the benefits exponentially, while the complexity follows a linear trend. On one side of the coin, the RS addresses with the majority of the errors, while on the other side the BCH corrects the few remaining damaged bits. Figure [18] represents the conclusion of this analysis.

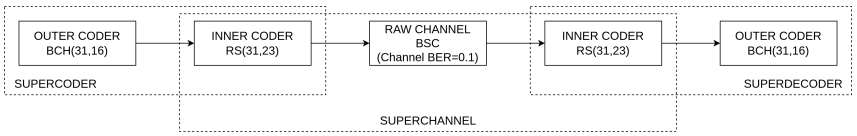


Figure 18: Proposed ECC

This structure is well designed because the RS code will try to correct all the erroneous symbols and if a corrupted symbol remains, the BCH will be able to correct it. As the foreseen channel BER is 0.1, it would be unusual for the RS to be overridden, but the BCH reinforces the structure and guarantees a error-free transmission.

4 Implementation

Each stage can be optimised to an infinite degree for example, by testing each ECC, studying possible improvements, and so on. However, this would be totally impractical. This stage is not an exception and it is fully in line with this directive. The implementation consists of landing all the theoretical concepts discussed in the real world. The hardware of Hydra is fully designed and implemented, therefore no changes are allowed.

It is feasible to develop all the required software, but this work would take a considerable amount of time, which exceeds the project timeline. Seconded to this reason, it is necessary to find libraries which provide a BCH and RS coder.

4.1 Why C?

The initial stage of any implementation is to determine the manner in which it will be executed. In this case, Hydra requested a software implementation. Given the limited space for electronic boards and the hostile environment with radiation, sudden temperature changes, etc., such as space it is preferable to use the least amount of hardware as possible. Figure [19] shows a catastrophic failure due to cosmic rays. Software is not immune to these extreme conditions, and, even on the surface, might have light problems. For example, as a curious aside, the fastest speed run on Super Mario 64 was achieved through to an unexpected bit flip. This involved a cosmic ray strike on the RAM memory, enabling the player to leap over a forbidden platform [59]. Other notable examples include the electoral results in Holanda where Maria Vindegovel obtained 4096 more votes [60], and the case of the flight QF72, which had to modify the original route to perform an emergency landing [61].

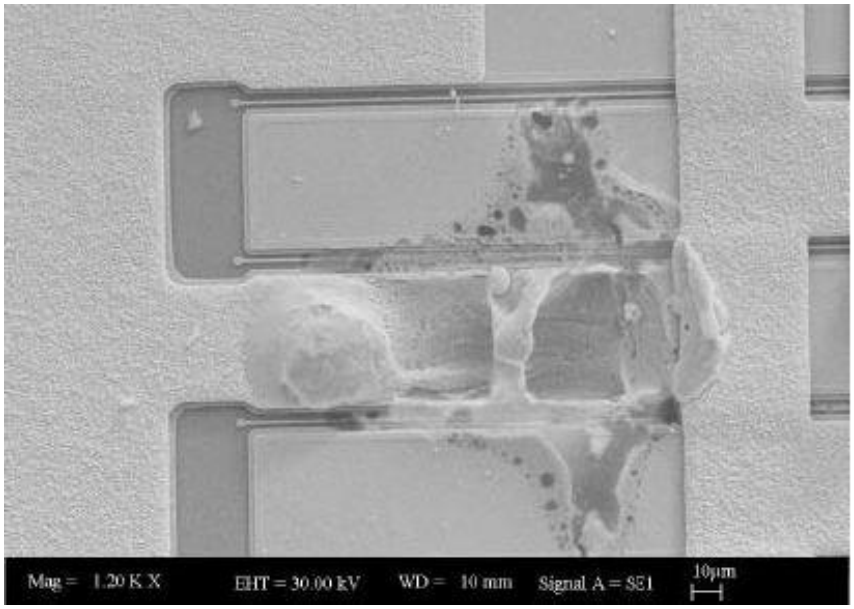


Figure 19: Catastrophic 'latch-up' due to heavy ion [62]

There are numerous languages that can be employed to implement this kind of software. A programming language is considered to be of a high level when it adds an abstraction layer which provides a more user-friendly interface. This allows the creation of programs that are easier to read and interpret. High level languages such as Python or

Javascript typically provide libraries of all types, including FEC libraries, but there are several drawbacks to them.

- **The lack of control over the hardware**
  In order to facilitate the programming process, the program makes certain assumptions that encroach upon the total control over the process. An analogy can be drawn between Windows and Linux. On one hand Windows is very easy to use, it provides a powerful generic environment and may support the majority of applications. However, there are folders and applications which cannot be deleted and some internal processes cannot be modified. On the other hand, Linux is more challenging to learn, there are numerous distributions that adapt to a wide variety of requirements. This leads to advanced users who have a total control of their computer but they must know how to manage that control.

- **Resource consumption**
  The high-level programs need an interpreter or a compiler. The first reduces the abstraction of the scripts and prepares the program to be executed, while the second translates the code to assembly language. In any case, there is an intermediate stage which requires time and valuable resources that a microcontroller may not have. This increases the execution time notably.

- **Portability**
  This type of languages often relies on libraries and may necessitate the use of execution environments such as the Java Virtual Machine. The surrounding complexity high level is such that they may require operating systems. Despite the fact that many of them are optimised for microcontrollers, it would be necessary to adapt all the communications software of Hydra.

It is therefore imperative to utilise low level languages such as hex, assembler or binary. Given the mathematical complexity of the FEC, it is not feasible to program this in the aforementioned languages. Consequently, the most common language which circumvents the aforementioned disadvantages is C.

4.2 BCH

The implementation of the outer code will be facilitated by the contributions of Morelos Zaragoza [63]. The author himself also mentions in his work two collaborators: Patrick Boyle and Simon Rockliff.

The encoder follows multiple steps to perform its labour. First of all, it is mandatory to define the framework, in other words, the Galois field where the encoder will work. In order to achieve this objective, it is mandatory to define the generator polynomial, which should be distinguished from the BCH generator polynomial. Once the polynomial has been calculated, the elements of the Galois field are computed. At this point, the mathematical framework is ready to proceed.

Now, it is turn to construct the encoder itself. Note the parallelism with the previous steps. First, the code must be defined by its generator polynomial. Once the generator polynomial has been obtained, the encoding process is straightforward: the bits in polynomial form are multiplied with the generator polynomial.

The decoder then follows. First, the decoder checks if there is any error by computing the syndrome. The syndrome is a polynomial of $n - k$ degree, which is zero if there is no

error. This polynomial is related to the position of the damaged bits. There are supposed to be many errors and the input needs to be corrected. The Berkelamp algorithm accepts the syndrome as input and find the error locator polynomial. If this polynomial has a degree greater than the error correction, the process is terminated and the data is returned, as there are so many errors that the system is unable to correct them. This is not an expected outcome and should occur in the minimum number of cases. In the event that this does not occur, the Chien search algorithm will identify the roots, which are directly related to the error locations, and correct them.
The original software is designed to operate with a generic BCH, so the implemented code is an advanced version of it specifically tailored to Hydra. In order to save time, the Galois field and the generator polynomial are precomputed. This optimisation achieves a higher saving of time. The reduction of the time is similar to the complete procedure with a BCH(7,4).

4.2.1 Example of coding and decoding

The following example illustrates the process of codification and decodification of the program. In order to facilitate the process, this example employs a BCH(7,4) code, despite the steps being completely identical. Some details of the original selected code are also provided. The first stages are precomputed but they are also explained in detail.

Although a brief introduction is provided, the arithmetic behind these codes is out of the subject of this work. The selected codes are linear block codes, which means that they are based on vector spaces and finite fields. Informally, a field is a "set of elements in which we can perform addition, subtraction, multiplication, and division without leaving the set" [7, p. 31]. The field in which FEC operates is that of Galois fields. In the binary case, this is represented by GF(2).

Table 7: Binary operations over Galois fields

+	**0**	**1**
0	0	1
1	1	0

×	**0**	**1**
0	0	0
1	0	1

In order to define mathematically a Galois field GF(2^m), it is necessary to define a generator polynomial that fulfils certain mathematical requirements. The generator polynomial for GF(2) is $p(x) = x + 1$. In order to successfully decode the codewords in question, it is necessary to construct a variation of the presented Galois fields. These are treated as extensions of the Galois fields. In the case of the BCH(7,4), it is necessary to build the GF(2^3), and for the BCH(31,16) the GF(2^4), which generator polynomials are respectively $p(x) = 1 + x + x^3$ (do not confuse with the polynomial generator, it is coincident.) and $p(x) = 1 + x^2 + x^5$. The first root of this polynomials are the generator element. If any operation results in an element with a higher degree than $2^m - 1$, then the result is congruent with the 2^m module. The following table shows the GF(2^3) elements [7, p. 1204]. It should be noted that the defined concepts are oversimplified.

The original Morelos's program always computes this polynomial and its roots, however, they will be precomputed.

Once, it is physically possible to perform complex operations over GF(2^m), the next step is to define which BCH will be employed. These codes are defined by a generator

Table 8: GF(2^3) elements generated by $p(x) = 1 + x + x^3$

Element	**Binary representation**	**Element**	**Binary representation**
-	000	α^3	110
α^0	100	α^4	011
α^1	010	α^5	111
α^2	001	α^6	101

polynomial, for this example is $g(x) = 1+x+x^3$. The generator polynomial of BCH(31,16) is:

$$g(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{15} \tag{3}$$

This is the last precomputed step.

Given this stream of bits $\vec{m} = (0110)$ and the BCH generator polynomial $g(x) = 1 + x + x^3$, the systematic codification process is presented:

1. Represent the information as a GF(2) polynomial. The right bit is the less significant.

$$\vec{m} = (0110) \Leftrightarrow m(x) = x + x^2 \tag{4}$$

2. Shift the bits $n - k$ positions.

$$m(x)x^{n-k} = (x + x^2)(x^{7-4}) = x^4 + x^5 \tag{5}$$

3. Divide $m(x)x^{n-k}$ between $g(x)$

$$m(x)x^{n-k} = q(x)g(x) + r(x) \Rightarrow \begin{cases} q(x) = 1 + x + x^2 \\ r(x) = 1 \end{cases} \tag{6}$$

4. Finally, compute the codeword, $v(x)$:

$$\begin{cases} v(x) = q(x)g(x) = (1 + x + x^2)(1 + x + x^3) = 1 + x^4 + x^5 \\ v(x) = m(x)x^{n-k} - r(x) = (x + x^2)(x^3) + 1 = 1 + x^4 + x^5 \end{cases}$$
$$v(x) = 1 + x^4 + x^5 \tag{7}$$

The obtained codeword is. Note how $\vec{m}$ is embedded.

$$v(x) = 1 + x^4 + x^5 \Leftrightarrow \vec{v} = 0110001 \tag{8}$$

The obtained codeword is then transmitted and received. The steps followed by the receiver are listed below. Initially, no errors are added to demonstrate that the syndrome is zero when this occurs. Then the received codeword will be $\vec{r} = 0110101$. The presented examples do not exceed the correction capabilities, which are $t = 1$ bits for BCH(7,4), and $t = 3$ bits for BCH(31,16).

1. Build the syndrome. This polynomial has $2t = 2$ coefficients over GF(2^3)

$$\vec{S} = (S_1, S_2) = (r(\alpha^1), r(\alpha^2)) = ((1+\alpha^4+\alpha^5), (1+\alpha+\alpha^3)) = (0, 0) \Rightarrow S(x) = 0 \tag{9}$$

If there were errors, $r(x) = 1 + x^2 + x^4 + x^5$.

$$\begin{cases} S_1 = r(\alpha) = \alpha^0 + \alpha^2 + \alpha^4 + \alpha^5 = \alpha^2 = 001 \\ S_2 = \alpha^0 + \alpha^4 + \alpha^8 + \alpha^{10} = \alpha^4 = 011 \end{cases} \quad \vec{S} = (\alpha^2, \alpha^4) \tag{10}$$

2. Application of the Berkelamp algorithm. [7] is not mathematically detailed, [64] is recommended instead. See [E.1] for more details.

Table 9: Application of Berkelamp to the received data.

Iteration	$\mathbf{\Delta_r}$	$\mathbf{B^r(x)}$	$\mathbf{\Lambda^r(x)}$	$\mathbf{L_r}$
0	-	1	1	0
1	α^2	α^6	$1+\alpha^2 x$	1
2	0	$\alpha^6 x$	$1+\alpha^2 x$	1

The error locator polynomial is then given by the expression: $\Lambda(x) = 1 + \alpha^2 x$. If this polynomial would were of a degree higher than the correction capabilities, it would be impossible to identify the position of the corrupted bits, and consequently, the damaged bits will not be fixed.

3. The roots of the error locator can be found through the Chien algorithm, which is essentially a brute force approach. Given that the polynomial has a degree of one, there is also just one root.

Table 10: Chien algorithm to find the roots of the error locator polynomial

TEST	**RESULT**	**TEST**	**RESULT**
α^0	α^6	α^3	α^4
α^1	α^1	α^4	α^2
α^2	α^5	α^5	0

4. α^5 is the root of the polynomial, so the error is located at its inverse: α^2. The error pattern in polynomial form is:

$$e(x) = x^2 \tag{11}$$

5. The error pattern is added to the received polynomial.

$$r'(x) = 1 + x^2 + x^4 + x^5 + x^2 = 1 + x^4 + x^5 \Leftrightarrow \vec{r'} = (0110001) \tag{12}$$

$\vec{r'}$ is equal to $\vec{v}$. The decoding algorithm has ended successfully.

4.3 Reed - Solomon

Although Linux distributed his code in the library "libfec0", the implemented RS code is provided by Phil Karn [65][66] [67]. Due to the fact that BCH and RS have the same mathematical origin, their codecs will be similar. The RS codes employ a variant of Galois fields called extended Galois fields, which means that, instead of transmitting just bits, symbols are used.

With regard to the encoder, the BCH may be considered as a reference because the procedure is identical. The only distinction is the coefficients of the RS generator polynomial, which are symbols instead of bits.

Despite its similarity, the decoder is modified slightly since two polynomials must be found: the error locator polynomial and the error value polynomial. Given that the BCH

is either 1 or 0, if an error is detected, it is sufficient to change the bit to correct it. In contrast, RS symbols are a set of bits which can be modified individually. This implies that it is necessary to identify the damaged symbols and the corrupted bits of each of these symbols. Once these two polynomials have been solved, the decoding process is complete. It is not surprising that the steps to build these polynomials are lightly modified or even equal BCH algorithms, in this case the Berkelamp - Massey - Forney algorithm (from now on, Berkelamp algorithm), which can be mathematically related to the Euclidean algorithm [9, p. 136]. Both polynomials are solved thanks to the Chien algorithm, like the BCH codes. Once the errors are located and its value have been determined, the correction is trivial. The example below illustrates the mathematics at the heart of the script. Due to simplicity, the example is based on a simpler code, RS(7,3), instead of the RS(31,23).

4.3.1 Example of coding and decoding

Firstly, it is mandatory to construct the mathematical framework that defines the Galois fields within which the RS operates. In this case, just one field is required because this ECC exclusively operates with symbols. The RS(31,23) works over GF(2^5), in contrast to RS(7,5) which works on GF(2^3). Each code is constructed with two generator polynomials which are respectively $1+x+x^3$ and $1+x^2+x^5$. The employed field is equal to the BCH of the previous example, and the table [8] illustrates GF(2^3). It should be noted that the number of bits per symbol defines the GF and the degree of the generator polynomial.

Once, the GF has been defined, it is necessary to determine the generator polynomial. Mathematically, obtaining it is straightforward. However, this is not the focus of this work, and thus only the results are presented. The generator polynomial for RS(7,3) is $3x+2$ in contrast, to the RS(31,23) which is:

$$g(x) = x^8 + \alpha^3 + x^7 + \alpha^{22}x^6 + \alpha^{23}x^5 + \alpha^{19}x^4 + \alpha x^3 + \alpha^9 x^2 + \alpha^{30}x + \alpha^5 \tag{13}$$

This is the last precomputed step.

1. In this case, the transmitter has the following message , which is expressed as a vector and polynomial.

$$\vec{m} = (101\ 111\ 011) \Leftrightarrow \vec{m} = (\alpha^6, \alpha^5, \alpha^4) \Leftrightarrow m(x) = \alpha^6 x^2 + \alpha^5 x + \alpha^4 \tag{14}$$

2. Shift the symbols $(n-k)$ positions:

$$m(x)x^{n-k} = (\alpha^6 x^2 + \alpha^5 x + \alpha^4)x^4 = \alpha^6 x^6 + \alpha^5 x^5 + \alpha^4 x^4 \tag{15}$$

3. Obtain the module of $m(x)x^{n-k}$ between $g(x)$. This is the redundancy.

$$m(x)x^{n-k}\ mod\ g(x) \ \equiv \ \alpha^3 x^3 + \alpha^2 x^2 + \alpha x + 1 \tag{16}$$

4. The codeword is the result of the concatenation of the message and the redundancy.

$$v(x) = m(x)x^{n-k} - (m(x)x^{n-k}\ mod\ g(x)) = \alpha^6 x^6 + \alpha^5 x^5 + \alpha^4 x^4 + \alpha^3 x^3 + \alpha^2 x^2 + \alpha x + 1 \tag{17}$$

5. If the codeword is required in bits to the modulator, $m(x)$ can be easily converted.

$$v(x) = \alpha^6x^6+\alpha^5x^5+\alpha^4x^4+\alpha^3x^3+\alpha^2x^2+\alpha x+1 \Leftrightarrow \vec{v} = (101\ 111\ 011\ 110\ 001\ 010\ 100) \tag{18}$$

Analogous to the BCH, the syndrome is calculated. In this case, the syndrome has four components because the ECC can correct upon two symbols. No errors are introduced so $r(x) = v(x)$

$$\begin{cases} S_1 = r(\alpha) = 0 \\ S_2 = r(\alpha^2) = 0 \\ S_3 = r(\alpha^3) = 0 \\ S_4 = r(\alpha^4) = 0 \end{cases} \tag{19}$$

In order to demonstrate the efficacy of the decoding algorithm, a burst error is introduced, resulting in:

$$r(x) = \alpha^6x^6 + \alpha^5x^5 + \alpha^4x^4 + \alpha^2x^3 + \alpha^3x^2 + \alpha x + 1 \tag{20}$$

1. The new syndrome is computed:

$$\begin{cases} S_1 = r(\alpha) = \alpha^3 \\ S_2 = r(\alpha^2) = \alpha \\ S_3 = r(\alpha^3) = \alpha^5 \\ S_4 = r(\alpha^4) = \alpha^4 \end{cases} \tag{21}$$

2. Berkelamp algorithm is applied. See [E.2] for more details.

Table 11: Application of Berkelamp to the received data

Iteration	$\mathbf{\Delta_r}$	$\mathbf{B^r(x)}$	$\mathbf{\Lambda^r(x)}$	$\mathbf{L_r}$
0	-	1	1	0
1	α^3	α^4	$1+\alpha^3x$	1
2	α^5	α^6x	$1+\alpha^5x$	1
3	α	α^2+x	$1+\alpha^5x+\alpha^5x^2$	2
4	0	$\alpha^6+\alpha^4x$	$1+\alpha^5x+\alpha^5x^2$	2

The error locator polynomial is $\Lambda^4(x) = 1 + \alpha^5x + \alpha^5x^2$. Its degree is two due to the number of errors, also there are two roots: $\beta_1^{-1} = \alpha^4$ and $\beta_2^{-1} = \alpha^5$. The table presents the Chien's outputs. At this point, the location of the errors are known: $\beta_1 = \alpha^3$ and $\beta_2 = \alpha^2$. Finally, the value of the errors can be determined by following the steps below.

Table 12: Chien's algorithm to find the roots of the error locator polynomial

TEST	**RESULT**	**TEST**	**RESULT**
α^0	α^0	α^3	α^6
α^1	α^6	α^4	0
α^2	α^2	α^5	0

3. Compute the error value polynomial, $Z(x)$:

$$Z(x) = 1 + (S_1 + \Lambda_1)x + (S_2 + \Lambda_1 S_1 + \Lambda_2)x^2 + \cdots + (S_\tau + \Lambda_1 S_{\tau-1} + \Lambda_2 S_{\tau-2} + \cdots + \Lambda_\tau)x^\tau$$
$$Z(x) = 1 + (S_1 + \Lambda_1)x + (S_2 + \Lambda_1 S_1 + \Lambda_2)x^2 = 1 + \alpha^2 x + \alpha^5 x^2 \tag{22}$$

Where τ is the number of errors.

4. Apply Chien to find the roots of the error value polynomial.

Table 13: Chien's algorithm to find the roots of the error value polynomial

TEST	**RESULT**	**TEST**	**RESULT**	**TEST**	**RESULT**
α^0	α	α^3	0	α^6	0
α^1	α^3	α^4	α^0	-	-
α^2	α^3	α^5	α	-	-

5. The error values can be obtained with the following expression:

$$e_{ji} = \frac{Z\left(\beta_1^{-1}\right)}{\prod\limits_{\substack{k=1 \\ k\neq i}}^{\tau} \left(1 + \beta_k \beta_i^{-1}\right)} \tag{23}$$

Then:

$$e_{j1} = \frac{Z\left(\beta_1^{-1}\right)}{\prod\limits_{\substack{k=1 \\ k\neq 1}}^{2} \left(1+\beta_k \beta_1^{-1}\right)} \Rightarrow e_{j1} = \frac{Z(\alpha^4)}{(1+\beta_2\beta_1^{-1})} = \frac{\alpha^0}{\alpha^2} = \alpha^5$$
$$e_{j2} = \frac{Z\left(\beta_2^{-1}\right)}{\prod\limits_{\substack{k=1 \\ k\neq 2}}^{2} \left(1+\beta_k \beta_2^{-1}\right)} \Rightarrow e_{j2} = \frac{Z\left(\alpha^5\right)}{\left(1+\beta_1\beta_2^{-1}\right)} = \frac{\alpha}{\alpha^3} = \alpha^5 \tag{24}$$

6. The error values are equal to α^5 for both cases and their positions are the second and third coefficients ($\beta_1 = \alpha^3$ and $\beta_2 = \alpha^2$), respectively. It should be reiterated that the positions are counted from the less significant coefficient. The error pattern is:

$$e(x) = \alpha^5 x^2 + \alpha^5 x^3 \tag{25}$$

The error pattern and the received data are added to recover the data:

$$r'(x) = r(x) + e(x) = (\alpha^6 x^6 + \alpha^5 x^5 + \alpha^4 x^4 + \alpha^2 x^3 + \alpha^3 x^2 + \alpha x + 1) + (\alpha^5 x^2 + \alpha^5 x^3) =$$
$$= \alpha^6 x^6 + \alpha^5 x^5 + \alpha^4 x^4 + \alpha^3 x^3 + \alpha^2 x^2 + \alpha x + 1 = v(x) \tag{26}$$

$\vec{r'}$ is equal to $\vec{v}$. The decoding algorithm has successfully terminated. If the degree of $\Lambda^4(x)$ exceeds the error correction capability of this code, $t = 2$ symbols, then there are more errors than the RS can correct, and the decoding would have failed.

4.4 Future lines

The efforts related to the maintenance of the software is minimal. It is sufficient to ensure that the layer remains being compatible with the adjacent layers. In order to expand this software and enhance its capabilities, several related ideas are presented here.

4.4.1 CRC

The CRC has the potential to increase the error detection, and, in certain instances, it may even circumvent the decoding process if the packet passes the CRC test. This improvement is relatively straightforward and can result in significant savings in terms of computational resources and time. The improvement can be demonstrated mathematically. Defining $r_{crc}(x)$ as the remainder of the division between the $r(x)$ and the CRC generator polynomial $g_{crc}(x)$:

$$r_{crc}(x) = r(x)\ mod(g_{crc}(x)) = (v(x) + e(x))\ mod(g_{crc}(x)) \tag{27}$$

A field is distributive by definition, and the division can be seen as a multiplication of inverses. Thus:

$$(v(x) + e(x))\ mod(g_{crc}(x)) = v(x)\ mod(g_{crc}(x)) + e(x)\ mod(g_{crc}(x)) \tag{28}$$

Finally, if the remainder of the message divided by $g_{crc}(x)$ without noise is considered, $r_v(x)$:

$$r_v(x) = r_{crc}(x) \Leftrightarrow e(x) mod(g_{crc}(x)) = 0 \tag{29}$$

The probability that the error pattern is a multiple of $G_{crc}(x)$ is minimal. In fact, it is so low that it is possible to rely on the CRC. Nevertheless, in the event that the test detects errors, the initial computation of the CRC could have been avoided. It supposes no additional information because the ECC can detect also errors. Then, a recalculation of the CRC is required in order to ensure that the FEC had worked properly.

In conclusion, it's a trade-off. If there are no corrupted bits, the message can be delivered correctly in practice. However, in other cases, the device loses time and computational resources.

4.4.2 Asymmetric FEC

This improvement is based on the fact that the IoT does not necessitate real-time connections and the encoding process is less demanding than decoding. While the satellite has a reduced amount of power and they have to answer to all customers, the ground devices have more time to compute and they can store more energy.

In these circumstances, it is possible that ground equipment may be able to exploit the additional time to decode more sophisticated codes, such as turbocodes, which offer greater coding gain. This would assist the satellite because the power demanded for transmission can be reduced even more while maintaining the source BER.

4.4.3 Dynamic FEC

Despite the fact that the link budget between satellite and receiver is within the line of sight, significant variations can occur. For example, consider the scenario in which there is a single client and the satellite is scheduled to fly directly over him. When the satellite appears up in the horizon, the distance between them is very high. The signal has to traverse across kilometres of atmosphere, where it is subject to a multitude of undesired phenomena, including refraction and noise, among others, occurring along the way. In a simplified manner, the signal has more time to be corrupted. In contrast to the case when the satellite is above the client position, where the distance is minimal and the signal does not have too much probability to be damaged.

The level of protection required when the satellite is positioned close to the horizon is likely to be greater than that required when the satellite is precisely above. Between these two cases, there is a spectrum of optimal degrees of protection. It is possible that the designed FEC is insufficient to cover the extreme long distances and that it adds too much redundancy to the best scenario. Adapting the ECC to each situation could result in a reduction in the binary rate and the time required for processing each packet.

In order to select the appropriate ECC, different measures can be taken into account. Two ideas from GSM/UMTS that can be useful in this regard. These include establishing a secondary channel dedicated to providing feedback to the other part of the quality of the link. In order to take statistically conclusive measures and adapt the link to fast changes in the channel, two loops are required. The larger loop nests the smaller one. The smaller loop detects fast changes in the channel, while the larger one measures magnitudes like source BER, which requires a higher amount of packets. By this method, the ECC can be adapted to each situation. In the event that the signal is clear but lacks sufficient power, it may be beneficial to implement an automatic power control. Conversely, if the receiver is able to listen to the transmitter properly, the power can be reduced.

4.4.4 Padding as an entropy source

The mathematical principles inherent in the cryptography are typically at odds with the "good properties" of linearity, and, in particular, bijectivity. A robust cryptographic system translates all of its inputs to very similar outputs. By this method, an attacker would have doubts if he attempts to obtain the foreign messages. The basic principles of modern cryptography are the based on the Kerchoff's principles. However, in order to guarantee the secrecy of the communications, Shannon proposed two techniques: confusion and diffusion [68]. Confusion pretends to conceal the relation between the key and the encrypted system. For example, if a message is to be encrypted and the output is obtained by substituting each letter with another arbitrary letter, this would not be a good confusion method because each letter corresponds to the same "key letter". On the other side, diffusion conceals the relation between the input and the output. For example, the DES protocol has 4 weak keys which should never be employed. This occurs because the output coincides with the input, thereby revealing the message.

A robust cryptosystem will substitute each character many times to create confusion and transpose and permute the input in order to generate diffusion. If the cryptosystem is fed with a low amount of bits, the message cannot be substituted and moved because there is no bits to do it. At this juncture, the padding could be a powerful tool if it is generated smartly. If the padding consisted of random bits or symbols that could be indistinguishable from actual information, the system would be less susceptible to any intrusive access.

5 Verification

5.1 Verification plan

Once, all the software has been implemented, it is then necessary to conduct several tests to ascertain that everything is working properly and is compliant with requirements. This final stage is designed to achieve this objective. In addition, some tools are needed to compare the obtained results so that their veracity have to be guaranteed. Although the popularity and power of the tools are considerable, every data set is going to be compared with two independent sources in order to insure credibility. The employed tools are listed here:

- **Octave:** Given that Octave offers FEC support, it can be considered a reliable tool. Since February 2000, Octave has been maintaining the communications package, extending its capabilities and fixing bugs in each update. Its most recent update was on May 2023.

- **Aff3ct:** Affect is a library designed to simulate the effects of different coding techniques. Its performance is noteworthy, particularly in terms of time, due to its multithreading programming. Despite its power, it has not yet achieved significant popularity, perhaps due to its relatively recent launch in 2019.

The process comprises three steps: the testing of BCH implementation, the testing of RS implementation, and the concatenation.

Respect to the first and second stage, the measures taken will be the execution time and complexity to verify compliance with the objectives of section [1.2]. Given the redundancy added by Octave, the encoder has to match that string entirely. The gain curve will be taken as the final guarantee of performance of each codec.

Finally, the concatenation tests will be conducted with the objective of achieving the final source BER, as specified by Hydra. The coding gain curve will be computed in Octave, supposed that BCH and RS will exhibit the expected behaviour.
It is also pertinent to highlight the high error probabilities used in the graphs. Due to the low computed probabilities, obtaining results may require a significant investment of time.

The verification plan can be summarized as follows:

Table 14: Verification plan

ECC UNDER TEST	STAGE UNDER TEST	TEST	OBJECTIVE
BCH	Design	Ascertain the program execution	The execution must be finished in less than 0.24 [s]
BCH	Implementation	Check if the redundancy is added correctly	The redundancy has to coincide to Octave's redundancy
BCH	Design	Check the complexity	The less complexity the better. Given that the software in question is in widespread use, this measure is designed to yield a specific outcome.
BCH	Implementation	Ascertain the coding gain	The coding gain must coincide with the obtained with Octave
RS	Design	Ascertain the program execution	The execution must be finished in less than 0.24 [s]
RS	Implementation	Check if the redundancy is added correctly	The redundancy has to coincide to Octave's redundancy
RS	Design	Check the complexity	The less complexity the better. Given that the software in question is in widespread use, this measure is designed to yield a specific outcome.
RS	Implementation	Ascertain the coding gain	The coding gain must coincide with the obtained with Octave
BCH+RS	Design	Given the channel BER, obtain the source BER	The source BER must be sufficiently low to ensure the delivery of a correct small packet on average

5.2 BCH

Firstly, the time will be quantified. The two provided values are the "real time" and the "user time". The first is the time as it is usually understood, namely the execution time of the program while the second is the time that the CPU has been busy with the program, outside the kernel layer. This is also an opportune moment to verify that the redundancy is added correctly. The timestamps are generated with the command "time" [69].

The output of the program is showed here.

```
1  Generated data:
2  0 1 1 1 0 1 0 0 1 0 1 1 1 1 1 1
3  Encoded word:
4  1 1 0 0 0 1 1 1 0 1 1 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 1 1 1 1 1
5  Noise:
6  1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0
7  Damaged message:
8  0 1 0 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 1
9  Decoded message:
10 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 1
11 Número total errores:
12 2
13 Posiciones:
14 0 14
15 Channel BER:
16 0.100000
17
18 Número de bits tx: 16
19 Número de errores: 2
20 Source BER: 1.25000e-01
21
22 real        0m0,010s
23 user        0m0,009s
```

The computing time is 0.010 [s] (line 22) which is considerably less than the upper limit defined in section [1.2]. This is even a highly satisfactory result particularly because this time includes the printing[6] and the encoding and decoding. The CPU, on the other hand has been computing for 0.009 [s] (line 23). These marks are, by far, encouraging. Ignore the fact the source BER is higher than channel BER, due to the limited number of messages, the obtained results are not statistically conclusive values.

As previously indicated, it is of the utmost importance to verify that the redundancy is perfectly calculated. The table presents data and codewords that have been computed by Octave and Morelo's software.

The complete set of codewords is consistent, which suggests that the coder is functioning as intended. Note that the data is embedded into the codeword. The fourth column means that the decoder has successfully recovered the same data.

[6]Printing is considered as a slow process due to the dependence on the operating system, applications (I/O buffering) and the screen.

DATA	OCTAVE	IMPLEMENTATION	MATCH
1100 0001 1110 1001	0101 0001 0010 1111 1000 0011 1101 001	0101 0001 0010 1111 1000 0011 1101 001	OK
1110 1111 1100 0111	0000 1110 1010 1111 1101 1111 1000 111	0000 1110 1010 1111 1101 1111 1000 111	OK
1101 0000 0100 1111	1110 0101 1110 1111 1010 0000 1001 111	1110 0101 1110 1111 1010 0000 1001 111	OK
1010 1111 0010 1101	1101 0000 1001 0101 0101 1110 0101 101	1101 0000 1001 0101 0101 1110 0101 101	OK
0101 1010 1010 1000	1001 1100 1110 0110 1011 0101 0101 000	1001 1100 1110 0110 1011 0101 0101 000	OK

Next, the complexity measures are presented. The cyclomatic complexity is governed by the following rules [16].

Table 15: Rules governing cyclomatic complexity

Range	Description
0 - 9	Easily maintained code
10 - 19	Maintained with little trouble.
20 - 29	Maintained with some effort.
30 - 39	Difficult to maintain code.
40 - 49	Hard to maintain code.
50 - 99	Unmaintainable code.
100 - 199	Crazy making difficult code.
200+	I only wish I were kidding.

Complexity obtained with [14]:

```
1  Modified McCabe Cyclomatic Complexity
2  |   Traditional McCabe Cyclomatic Complexity
3  |       |    # Statements in function
4  |       |        |   First line of function
5  |       |        |       |   # lines in function
6  |       |        |       |       |  filename(definition line number):function
7  |       |        |       |       |           |
8  9       9        36      2       40  bch.c(2): encode_bch
9  9       9        35      42      40  bch.c(42): char_encode_bch
10 37      37       128     87      181 bch.c(87): decode_bch
```

The column "modified McCabe Cyclomatic Complexity" is calculated by adding 1 to all the occurrenes of for, if, while, switch, &&, || and ?. The second column is the cyclomatic complexity obtained in the common way. Cyclomatic complexity is based on testing each linearly independent path through the program. In this case, the number of test cases will equal the cyclomatic complexity of the program. This approach enables the construction of graphs representing a program. The basic diagrams are illustrated in Figure [20]. The rest of columns are additional information like the number of lines, the line where the function starts and the lines occupied by the method. Other tools, like [15], also prints the number of lines without comments (column "NLOC"), tokens and the input parameters of a function. In this case there are two, the string to encode and a pointer to the output. The tokens are indivisible code blocks with special meaning to the compiler, i.e. for or if.

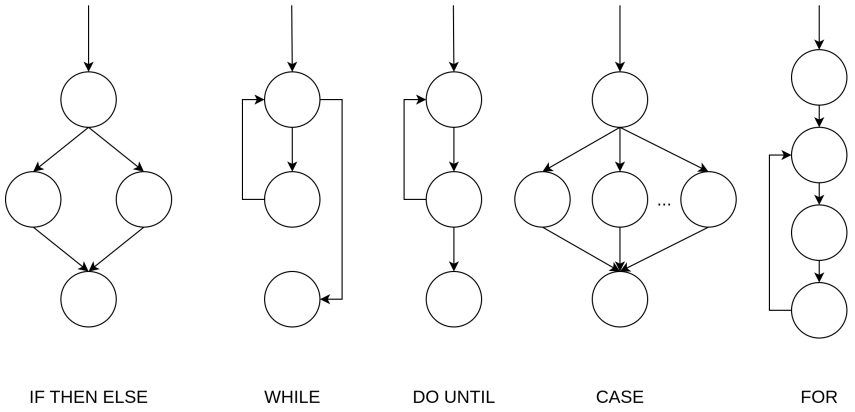


Figure 20: Basic flow diagrams

They are considered as irrelevant. [16] shows the complexity in the column "Score" and the number of lines with and without comments, ln-ct and nc-lns, respectively.

Complexity obtained with [15]:

```
================================================
NLOC    CCN   token  PARAM  length  location
------------------------------------------------
33      9     320    2      40      encode_bch@2-41@bch.c
32      9     317    2      40      char_encode_bch@42-81@bch.c
122     37    1338   2      181     decode_bch@87-267@bch.c
1 file analyzed.
==============================================================
NLOC    Avg.NLOC  AvgCCN  Avg.token  function_cnt    file
--------------------------------------------------------------
188      62.3     18.3    658.3      3               bch.c
==============================================================
```

Complexity obtained with [16]:

```
Score | ln-ct | nc-lns| file-name(line): proc-name
6      37      30       bch.c(41): char_encode_bch
6      37      31       bch.c(1): encode_bch
64     179     120      bch.c(86): decode_bch
total nc-lns      181
```

The decoder represents the most complex element in the system as it was foreseen in section [2.3.3], however, it is "easily maintained" in the complexity scale.

Finally, the graph [21] depicts the mean value of three measurements. Given that the data set exhibits minimal variation, this is not considered to be a significant statistic in this instance.

The graph illustrates the expected behaviour which may be considered as a proof that the system is functioning nominally. The values are also similar to figure [5b]. To generate the values, the two key values are the quantity of codewords transmitted and

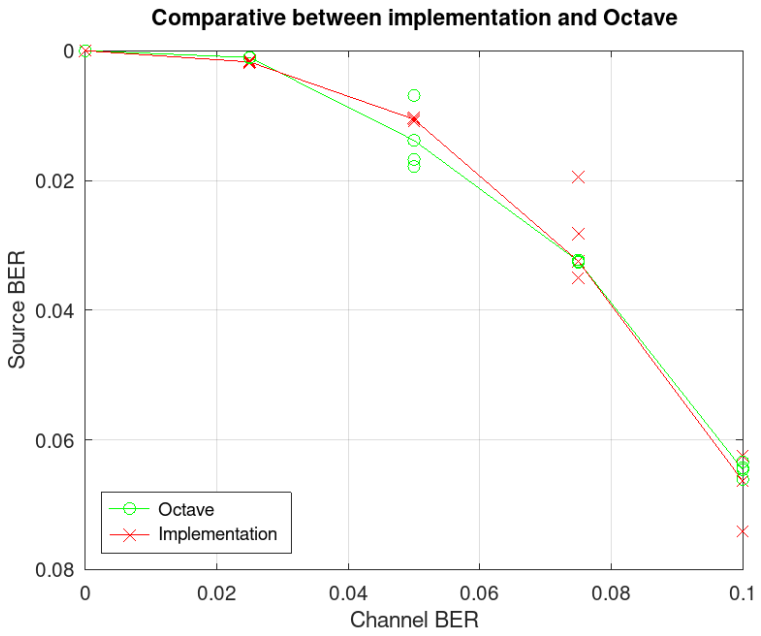


Figure 21: A comparative analysis of the Octave and Morelos software.

the maximum number of errors. When one of them is reached, the the code stops. In this case, the maximum number of words is 100,000,000 while the maximum amount of errors is 1,000,000. The requisite computational cost for higher values is significantly higher, which is the reason why the simulation took such low EbN0 values.

5.3 Reed - Solomon

Once the BCH has been successfully tested, the RS should be subjected to the same procedure. Here is an example of the output of the program.

```
DATA:
30 21 3 15 22 5 16 29 14 29 10 14 19 21 21 22 2 4 10 25 8 17 30
CODED:
30 21 3 15 22 5 16 29 14 29 10 14 19 21 21 22 2 4 10 25 8 17 30 1 10 7 2 4 7 27 2
AFTER NOISE:
30 21 3 15 18 1 22 29 14 5 10 28 23 29 21 22 3 4 11 25 8 17 30 1 10 7 18 4 7 11 18
DECODED:
30 21 3 15 18 1 22 29 14 5 10 28 23 29 21 22 3 4 11 25 8 17 30
Bits tx: 115
Error: 9
Source BER: 0.078261

real        0m0,004s
user        0m0,000s
```

With this test, it has become clear the system is capable of encoding and decoding at a rate that is fast enough. Once again, this time is the addition of the print and the encoding and decoding process. At this point, the objective of the time is completely fulfilled.

DATA	OCTAVE	IMPLEMENTATION	MATCH
8 3 31 11 20 14 22 2 16 12 16 13 10 21 3 3 8 7 0 22 10 13 1	8 3 31 11 20 14 22 2 16 12 16 13 10 21 3 3 8 7 0 22 10 13 1 17 3 1 14 22 19 6 7	8 3 31 11 20 14 22 2 16 12 16 13 10 21 3 3 8 7 0 22 10 13 1 17 3 1 14 22 19 6 7	OK
3 4 27 15 6 19 16 28 18 24 3 11 10 20 17 30 29 9 0 27 31 23 6	3 4 27 15 6 19 16 28 18 24 3 11 10 20 17 30 29 9 0 27 31 23 6 11 15 14 26 17 27 22 15	3 4 27 15 6 19 16 28 18 24 3 11 10 20 17 30 29 9 0 27 31 23 6 11 15 14 26 17 27 22 15	OK
7 9 12 30 26 24 3 30 24 24 29 2 19 25 24 9 26 19 0 10 23 24 2	7 9 12 30 26 24 3 30 24 24 29 2 19 25 24 9 26 19 0 10 23 24 2 11 17 23 20 6 11 23 27	7 9 12 30 26 24 3 30 24 24 29 2 19 25 24 9 26 19 0 10 23 24 2 11 17 23 20 6 11 23 27	OK
12 0 7 20 18 2 15 31 31 3 23 3 12 26 3 1 27 18 11 11 0 28 25	12 0 7 20 18 2 15 31 31 3 23 3 12 26 3 1 27 18 11 11 0 28 25 29 24 16 25 20 16 28 22	12 0 7 20 18 2 15 31 31 3 23 3 12 26 3 1 27 18 11 11 0 28 25 29 24 16 25 20 16 28 22	OK
11 18 1 7 31 8 10 30 30 25 7 1 26 2 5 11 9 19 15 31 14 6 30	11 18 1 7 31 8 10 30 30 25 7 1 26 2 5 11 9 19 15 31 14 6 30 12 1 31 6 22 22 4 30	11 18 1 7 31 8 10 30 30 25 7 1 26 2 5 11 9 19 15 31 14 6 30 12 1 31 6 22 22 4 30	OK

The following table compares different codewords to find any potential discrepancies.

Given 5 random words, there are no differences between them. The fourth column means that the decoder has successfully recovered the data.

The complexity measures are listed here.

The employed tool here is [14]:

```
Modified McCabe Cyclomatic Complexity
|   Traditional McCabe Cyclomatic Complexity
|       |    # Statements in function
|       |       |   First line of function
|       |       |       |   # lines in function
|       |       |       |       |  filename(definition line number):function
|       |       |       |       |           |
3       3       4       14      11      decode_rs_8.c(14): decode_rs_8
53      53      177     23      240     decode_rs.c(23): DECODE_RS
4       4       13      10      17      decode_rs_ccsds.c(10): decode_rs_ccsds
1       1       3       15      8       decode_rs_char.c(15): decode_rs_char
1       1       3       15      8       decode_rs_int.c(15): decode_rs_int
6       9       15      22      45      encode_rs_8.c(22): encode_rs_8
3       3       18      79      23      encode_rs_8.c(79): encode_rs_8_av
1       1       0       105     5       encode_rs_8.c(105): encode_rs_8_c
3       3       9       22      16      encode_rs_av.c(22): rs_init_av
3       3       18      39      23      encode_rs_av.c(39): encode_rs_av
7       7       19      15      38      encode_rs.c(15): ENCODE_RS
```

The employed tool is [15]:

```
================================================
NLOC    CCN   token  PARAM  length  location
------------------------------------------------
35      15    200      3      45  encode_rs_8@22-66@encode_rs_8.c
18      3     228      3      23  encode_rs_8_av@79-101@encode_rs_8.c
3       1     17       3      5   encode_rs_8_c@105-109@encode_rs_8.c
12      5     113      0      16  rs_init_av@22-37@encode_rs_av.c
18      3     230      3      23  encode_rs_av@39-61@encode_rs_av.c
9       3     91       3      14  encode_rs_ccsds@11-24@encode_rs_ccsds.c
4       1     30       3      6   encode_rs_char@10-15@encode_rs_char.c
4       1     30       3      6   encode_rs_int@10-15@encode_rs_int.c
8       3     42       4      11  decode_rs_8@14-24@decode_rs_8.c
12      4     109      4      17  decode_rs_ccsds@10-26@decode_rs_ccsds.c
```

```
6       1     39       4      8   decode_rs_char@15-22@decode_rs_char.c
6       1     39       4      8   decode_rs_int@15-22@decode_rs_int.c
11 file analyzed.
==============================================================
NLOC    Avg.NLOC  AvgCCN  Avg.token  function_cnt    file
--------------------------------------------------------------
66      18.7    6.3      148.3         3     encode_rs_8.c
39      15.0    4.0      171.5         2     encode_rs_av.c
27      0.0     0.0        0.0         0     encode_rs.c
11      9.0     3.0       91.0         1     encode_rs_ccsds.c
7       4.0     1.0       30.0         1     encode_rs_char.c
7       4.0     1.0       30.0         1     encode_rs_int.c
11      8.0     3.0       42.0         1     decode_rs_8.c
175     0.0     0.0        0.0         0     decode_rs.c
14      12.0    4.0      109.0         1     decode_rs_ccsds.c
10      6.0     1.0       39.0         1     decode_rs_char.c
10      6.0     1.0       39.0         1     decode_rs_int.c
```

The employed tool is [16]:

```
Complexity Scores
Score | ln-ct | nc-lns| file-name(line): proc-name
0       0       0       encode_rs_8.c(86): encode_rs_8_c
1       3       1       encode_rs_char.c(7): encode_rs_char
1       3       1       encode_rs_int.c(7): encode_rs_int
1       5       3       decode_rs_char.c(9): decode_rs_char
1       5       3       decode_rs_int.c(9): decode_rs_int
1       8       5       decode_rs_8.c(9): decode_rs_8
1       12      7       encode_rs_ccsds.c(9): encode_rs_ccsds
1       12      10      encode_rs_av.c(19): rs_init_av
1       15      10      decode_rs_ccsds.c(8): decode_rs_ccsds
2       21      16      encode_rs_8.c(61): encode_rs_8_av
2       21      16      encode_rs_av.c(34): encode_rs_av
4       35      33      encode_rs_8.c(13): encode_rs_8
total nc-lns      105
```

Although the library is precompiled, Linux provides all the source code, so it is possible to analyse it. Having regard the result, it is obvious to think that "DECODE_RS" is the main function and the others complement it. It has a high complexity to the point of "unmaintainable code" , so this is the weakest point of the system. In addition, it does not may appear in other tools due to the format of the programming, it must to be a generic return statement. There is also something that may shed some light on this and that its the Linux and open source guarantee. These libraries are under constant revision and they are widely used for a variety of tasks, therefore the possible bugs may be reported and corrected.

Finally, the curves are obtained:

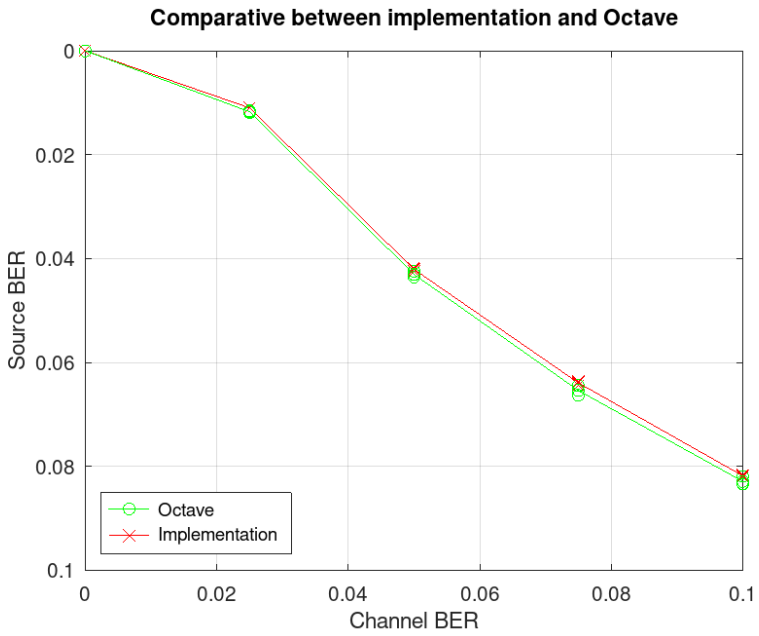


Figure 22: Channel BER vs source BER with RS(31,23)

A distinctive point of this graph in relation to the BCH is the variance of the data. This variation occurs when there are more data, so the probability converge to the theoretical value. Why are more data? The question is easy to answer, the symbols. Each symbol is 5 bits long, given the constant number of words, the transmitter sends 5 times more bits than the BCH. As far as the result is concerned, there is nothing to worry about, the two curves are very similar.

5.4 Concatenation scheme

It has been demonstrated that the FEC layer operates nominally, even despite the necessity of verifying that the system meets all the requirements. In other words, the system works, but does it work well enough? The trials conducted so far have tested the implementation, but not the design stage. The reason of this order is that there is no point in proving something that is technically impossible to achieve.

The graph [23] depicts the coding gain for the proposed concatenation scheme. The table quantifies the represented values.

Figure [23] represents the central element of this work, as it illustrates the behaviour of the system. Figures [30] and [31] are the coding gains of each single ECC. The obtained gain is significantly higher than each code separately (see [30] and [31]). It is important to note two key points: the intersection point and the slope.

On the one hand, the intersection point is analysed. These points are defined for each ECC are defined closely to four [dB], whereas in the concatenated scheme, it is not possible to measure this point. This represents a significant improvement because it means that it is always preferable to codify the information (section 3). On the other hand, the slope is moderately steeper. See also the exponential tendency of the curve, as Forney foresaw in 1965 [29]. One consequence of these simulation, when there is sufficient EbN0, the error

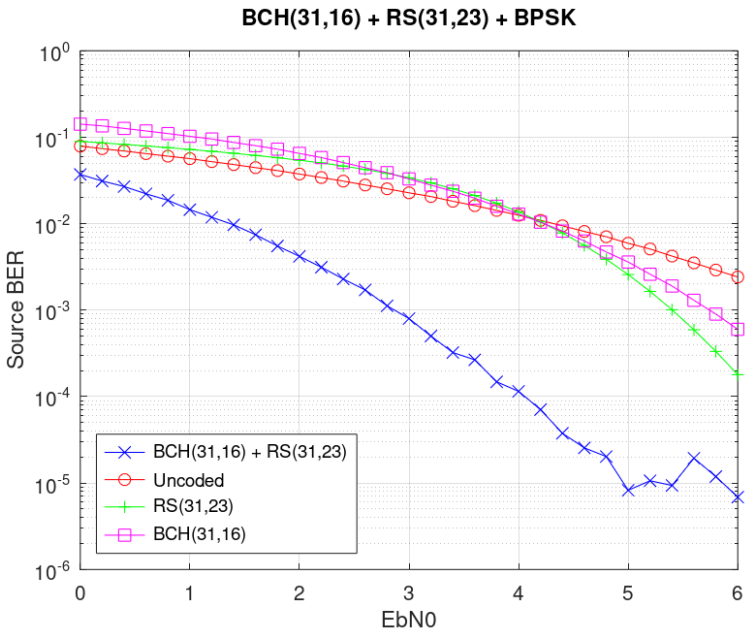


Figure 23: Coding gain of the final proposed scheme

Table 16: Graph [23] and a few examples of coding gains quantified

EbN0 value	Source BER				EbN0 value	Source BER			
	Uncoded	BCH(31,16)	RS(31,23)	BCH(31,16) + + RS(31,23)		Uncoded	BCH(31,16)	RS(31,23)	BCH(31,16) + + RS(31,23)
0	7,864E-02	2,043E-01	8,931E-02	3.7053E-02					
0.2	7,418E-02	1,940E-01	8,596E-02	3.1136E-02	2.2	3,415E-02	8,591E-02	5,036E-02	3,13E-03
0.4	6,938E-02	1,834E-01	8,258E-02	2.6883E-02	2.4	3,117E-02	7,610E-02	4,642E-02	2,29E-03
0.6	6,507E-02	1,727E-01	7,916E-02	2.2142E-02	2.6	2,826E-02	6,678E-02	4,236E-02	1,71E-03
0.8	6,043E-02	1,617E-01	7,571E-02	1.8635E-02	2,8	2,554E-02	5,802E-02	3,818E-02	1,12E-03
1	5,625E-02	1,507E-01	7,223E-02	1.4511E-02	3	2,283E-02	4,989E-02	3,391E-02	7,99E-04
1.2	5,238E-02	1,396E-01	6,872E-02	1.1858E-02	3,2	2,047E-02	4,242E-02	2,960E-02	5.0146E-04
1.4	4,848E-02	1,285E-01	6,517E-02	9.6868E-03	3,4	1,830E-02	3,565E-02	2,533E-02	3.2315E-04
1.6	4,463E-02	1,175E-01	6,157E-02	7.4437E-03	3,6	1,599E-02	2,959E-02	2,117E-02	2.6625E-04
1.8	4,080E-02	1,067E-01	5,792E-02	5.5119E-03	3,8	1,428E-02	2,424E-02	1,726E-02	1.4813E-04
2	3,767E-02	9,614E-02	5,419E-02	4.1862E-03	4	1,251E-02	1,959E-02	1,367E-02	1.1500E-04

EbN0 value	Source BER			
	Uncoded	BCH(31,16)	RS(31,23)	BCH(31,16) + + RS(31,23)
4,2	1,093E-02	1,560E-02	1,050E-02	7,06E-05
4,4	9,443E-03	1,225E-02	7,806E-03	3.763E-05
4,6	8,136E-03	9,460E-03	5,601E-03	2.550E-05
4,8	6,988E-03	7,190E-03	3,873E-03	2.025E-05
5	5,919E-03	5,372E-03	2,576E-03	8.250E-06
5,2	5,047E-03	3,942E-03	1,646E-03	1.063E-05
5,4	4,203E-03	2,840E-03	1,009E-03	9,38E-06
5,6	3,484E-03	2,008E-03	5,925E-04	1,94E-05
5,8	2,906E-03	1,391E-03	3,330E-04	1,19E-05
6	2,375E-03	9,437E-04	1,790E-04	6,88E-06

Source BER	Coding gain		
	BCH(31,16)	RS(31,23)	BCH(31,16) + + RS(31,23)
1,000E-01	-	-	-
5,000E-02	-	-	1.5
1,000E-02	0	0	2.75
5,000E-03	0.5	0.5	3.5

rate is extremely low. This makes computing the error rate is very expensive in terms of time, and the ECC is unable to correct any more errors. Consequently, the FEC is no longer an advancement and becomes an unnecessary element because there are no errors practically.

Mathematically, it is necessary to study if one packet is adequately protected against

the noise. It is a common practice to employ the EbN0 as a natural reference unit in these graphs. However, Hydra has specified its requirements regarding a source BER, rather than an EbN0. Given that Hydra works with FSK, their symbols are orthogonal, so the following expression can be applied [8, p. 125].

$$P_B = Q\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{30}$$

Where $Q(x)$ is the co-error function and P_B the channel BER.

$$Q(x) = \frac{1}{x\sqrt{2\pi}} \int_x^\infty \mathrm{e}^{-\frac{u^2}{2}} \approx \frac{1}{x\sqrt{2\pi}} \mathrm{e}^{-\frac{x^2}{2}} \tag{31}$$

According to different authors, this definition can vary. Given $P_B = 0.1$ and according to [8, p. 539]:

$$Q\left(\sqrt{\frac{E_b}{N_0}}\right) = 0.1 \Leftrightarrow \sqrt{\frac{E_b}{N_0}} = 1.28[dB] \Rightarrow \frac{E_b}{N_0} = 1,64[dB] \tag{32}$$

Figure [23] indicates the obtained gain at this point: three and a half [dB]. The source BER is $P_B = 7.4437 \times 10^{-3}$. Table [17] shows the average number of expected errors according to package size.

Table 17: Errors received on average

SIZE(B)	**ERROR**	**SIZE(B)**	**ERROR**	**SIZE (B)**	**ERROR**
12	0,7145952	19	1,1314424	26	1,5482896
13	0,7741448	20	1,190992	27	1,6078392
14	0,8336944	21	1,2505416	28	1,6673888
15	0,893244	22	1,3100912	29	1,7269384
16	0,9527936	23	1,3696408	30	1,786488
17	1,0123432	24	1,4291904	31	1,8460376
18	1,0718928	25	1,48874	32	1,9055872

It is mandatory to highlight that these values are obtained for the worst case, when channel BER is equal to 0.1. In the majority of the cases, it is expected that the link will exhibit superior performance in terms of EbN0. This ensures that the majority of the ACK and telemetry packets are received correctly without the need for any retransmission.

In conclusion, this designed FEC functions well enough to be present in a satellite.

6 Conclusions

The methodology employed in this work is deductive motivated by several purposes. Primarily, the deductive structure facilitates a clear and coherent structure among the sections and paragraphs. Other reasons for including this methodology are that it facilitates the introduction of concepts and the progression of these ideas. For example, when all the concepts were widely explained in section [2], it created a solid background to expose the results. The deductive structure also allows to follow for the logical order of the project. The software has now reached the stage where it is ready to be incorporated into the satellites. However, when this project commenced a year and a half ago, my knowledge of this field was limited. As the project progressed, each stage required different approaches, which gave me the opportunity to learn about all the steps of the process. Specially, the fact of being alone presented a personal challenge because this project was exclusively me against it. The inductive structure would have entailed the reduction of each chronological and nature step of the project to a list of arguments which would have distorted the work I have done. When an engineer encounters complex problems, he must face and solve them. This is the essence of the great engineering.

6.1 Lessons learnt

As previously stated, the project began one and a half years ago when Hydra came to the university requesting a FEC layer for their communication subsystem. At that juncture, I was interested in developing a substantial piece of work over the course of a year. Despite the considerable demanding time constraints, I would participate in an engineering project which could provide me with experience and a great content for an end-of-degree thesis.

6.1.1 Evaluation

This was the initial phase of the project, during which the current solutions were identified and presented as potential lines of investigation to Hydra. The main challenge for me was my lack of experience, which led me to invest a significant amount of time to explore these options. The fundamentals of this field are studied in a subject called "Teoría de la información", however, it was clearly insufficient to the scope of the project. While I had no difficulty studying the entropy of a source or the Hamming codes, the ECC protection codes, which are used in practice, represent an advanced evolution of these codes. In some cases, they even require knowledge of a distant field of mathematics, for example, the principles which rule the convolutional codes are diametrically opposed to the linear block codes. Understanding and realising all the mathematics and codes was a formidable task.

Other secondary problems were to find a reliable simulator to show Hydra all the possible codes. It was straightforward to find [42] so no comments are made in this regard.

6.1.2 Design

The primary objective of the design stage was to finally identify the codes that would be used. Up to this point, no numbers were practically provided. The main challenge of this stage was the subjectivity involved. Once the results are obtained, it is then time

to interpret them. Despite of the multiple paths that can be taken, it can be difficult to adhere to the numbers instead of hunches. The axis, which all this step was around, was the Excel sheet. While the padding statistics were available and reliable, the final coding gain remained an absolute mystery. Would it fulfil all the Hydra requirements?

This leads to the second major problem. The selected concatenation scheme was too specific to be provided by a paper or a book. It was necessary to test all the BCH(31,23) + RS(31,X) codes. This demanded the utilisation of untested codes and a significant computing load over a longer period of time. Thanks to the utilisation of the reliable Octave libraries and the Radiation Group's advanced computer, this problem was solved in a couple of weeks. The computation was completed within a week, and the results were obtained. And they were encouraging. As a fun fact, when I studied the obtained results I doubted their veracity. My concerns were alleviated when I compared the graphs with the turbo codes and read Forney's article, which I was pleased to quote because he is one of the founding fathers of the ECCs.

6.1.3 Implementation

The implementation phase was initiated, at last. It became evident that the project was nearing its conclusion. The main problem in this stage is the licenses. The FEC is a very specific aspect of telecommunications, so the libraries are not easy to find. Furthermore, the algorithms to decodify are not always optimal for the system. Fortunately, the libraries of C are free to use, even for commercial purposes. Nevertheless, Morelos protected his code for commercial use, thereby preventing Hydra from employing his code. This issue was identified after the code was written. From an academic perspective, the result is valid and sufficient to be considered an implementation, which is why it is reflected here. Legally, the implemented code is fully rewritten to be optimal for the BCH(31,16). Despite the extensive modifications and the prohibition on exploiting the code commercially, I will adapt the code to Hydra's requirements using the IT++ library, which employs the same decoding algorithms. This library is a C library so its use is completely free, even for commercial purposes. If I had been aware of this, I would have used directly the IT++ library. The changes are minimal thanks to the structure of the code and the performance will not be affected.

A Ethics

A.1 Introduction

The developed FEC layer is not directly related to a specific social problem or ethical dilemma. However, Hydra designs its satellites to meet the specific needs of the customer in question. As the previously stated in the mention, the space career took off after the Second World War and it flourished during the Cold War. Despite the military and political interest, there were civilian initiatives which were pioneers with low resources. For example, the Oscar - 1 was manufactured in a garage and it cost 68 American dollars (equivalent to 712.96 current American dollars) [70].

A.2 Description from impacts related to the project

Hydra has identified five objectives related to the Sustainable Development Goals (SDG), which are listed below [10]:

- **End poverty & zero hunger:** Satellite IoT for crop monitoring, food security and productivity
  This point is directly related to the first and second objectives: "end poverty in all its forms everywhere" and "zero hunger".

- **Affordable & clean energy:** Satellite IoT for energy monitoring and emissions reduction
  The SDG number seven pretends to guarantee an affordable access to energy.

- **Ocean, clean water & sanitation:** Satellite IoT for water monitoring, leak detection, waste reduction and quality
  Contributing to the objective number six: "clean water and sanitation".

- **Sustainable cities:** Satellite IoT for urban monitoring and efficiency
  The satellites helps "making cities and human settlements inclusive, safe, resilient and sustainable".

- **Protect and restore terrestrial ecosystems:** Satellite IoT for wildlife and ecosystem monitoring and protection
  SDG 15: "Protect, restore and promote sustainable use of terrestrial ecosystems, sustainably manage forests, combat desertification, and halt and reverse land degradation and halt biodiversity loss."

A.3 Satellites: pros and cons

The satellites play a pivotal role in the development of the urban areas. Their ability to obtain detailed data from the surface and offer communication services has earned the CubeSats a place in modern societies. For example, they achieve the broadcast the television signals among distant cities and remote regions without difficulty or location services via GPS.

Economically, they are usually a consequence of the high degree of development of a country. Satellites are even employed as military dissuasion and as a demonstration of power. Nevertheless, there are rays of hope to get better and useful goals, such as the

study of major migratory flows or the location of valuable resources like minerals.

Technologically, satellites represent a significant leap forward for humanity. Nowadays, it is possible to establish quasi-instantaneous intercontinental communications. Many users may receive the same signal despite their considerable separation. Also, the fact of being able of getting into orbit is an end in itself. This achievements allow us, as specie, to answer transcendental questions like "Who are we?".

While satellites represent a potential solution, they are often expensive and not always a viable option. They require to be launched and a few countries in the world are able to put objects in orbit. Contracting the launch services of one country or another may have political implications and can be a reason for disagreement. To illustrate this point, the countries generally conduct a great deal of research into the potential implications of purchasing jet fighters from a particular country or another [71].

Finally, there is another point which is inseparably related to the digital era, the privacy. In Google Maps can be found all the houses, addresses, etc. of a country. This sensible information, so much so that countries like China have forbidden the real location of the streets to be shown. It is evident that governments can auto-protect themselves with laws but it is less clear who protects the people. Even, the military forces can be exposed if they do not take action. The case of the film "*Top Gun: Maverick"* is a striking example of this. China diverted from its orbit a spy satellite to inspect the hypersonic plane that appears in the film [72]. It would not be surprising if resolutions good enough to track a person from space have been reached. Figure [24] shows an example of this. It is AMARG, the world's largest aircraft graveyard. The aircraft in the photograph is a KC-135, which has a wingspan of 40 metres, and the inscriptions on the wings are clearly visible. The satellite is 7000 [km] above (Geoeye-1) [73].

Figure 24: KC-135 on the AMARG

A.4 Conclusion

Urban development has become possible due to satellites providing great opportunities for modern communities. The ability of CubeSats to gather information and offer means for communication have revolutionized the way our cities look; we can now watch TV on long distances and get direction through GPS technology.

In terms of economy, having satellites is an indicator that a country has reached its technological heights, as they are often seen as a sign of developmental status. Although they do have military applications, their potential goes beyond this since they can be used in tracking migratory routes and locating mineral deposits.

Technically, humanity made a giant leap with the invention of satellites that allow us to communicate from one continent to another within seconds and make us see our planet from above. Their being in orbit raises profound metaphysical questions and provides humans with new insights into the universe.

Nevertheless, despite all their advantages, there are still complications with regard to satellites. Expensive price tags attached to these tools along with few states having capacity for sending them into space reveal the political intricacies inherent in satellite launching. Additionally, concerns about online privacy persist because images from spy satellites may infringe upon civil liberties or threaten national security.

B Budget

LABOUR COST

HOURS	€ / HOUR	TOTAL (€)
360	20	7200

COST OF MATERIAL RESOURCES (direct cost)

	PURCHASE PRICE (€)	USE (MONTHS)	AMORTIZATION (years)	TOTAL (€)
Computer	1500	10	5	250

TOTAL COST OF MATERIAL RESOURCES	250

GENERAL EXPENSES (indirect costs) (€)	15,00 %	over CD	1117.5
INDUSTRIAL PROFIT (€)	6,00 %	over CD+CI	514.05

CONSUMABLES

Print (€)	100
Binding (€)	300

SUBTOTAL BUDGET (€)	9481.55
VAT APPLICABLE (€)	1991.13
TOTAL BUDGET (€)	11472.68

C Error correction families: coding gain

This section documents the coding gains for comparison. See [2.2] and [2.3]

C.1 Possible enhancements in numbers

CRC just detects errors so no graph is needed.

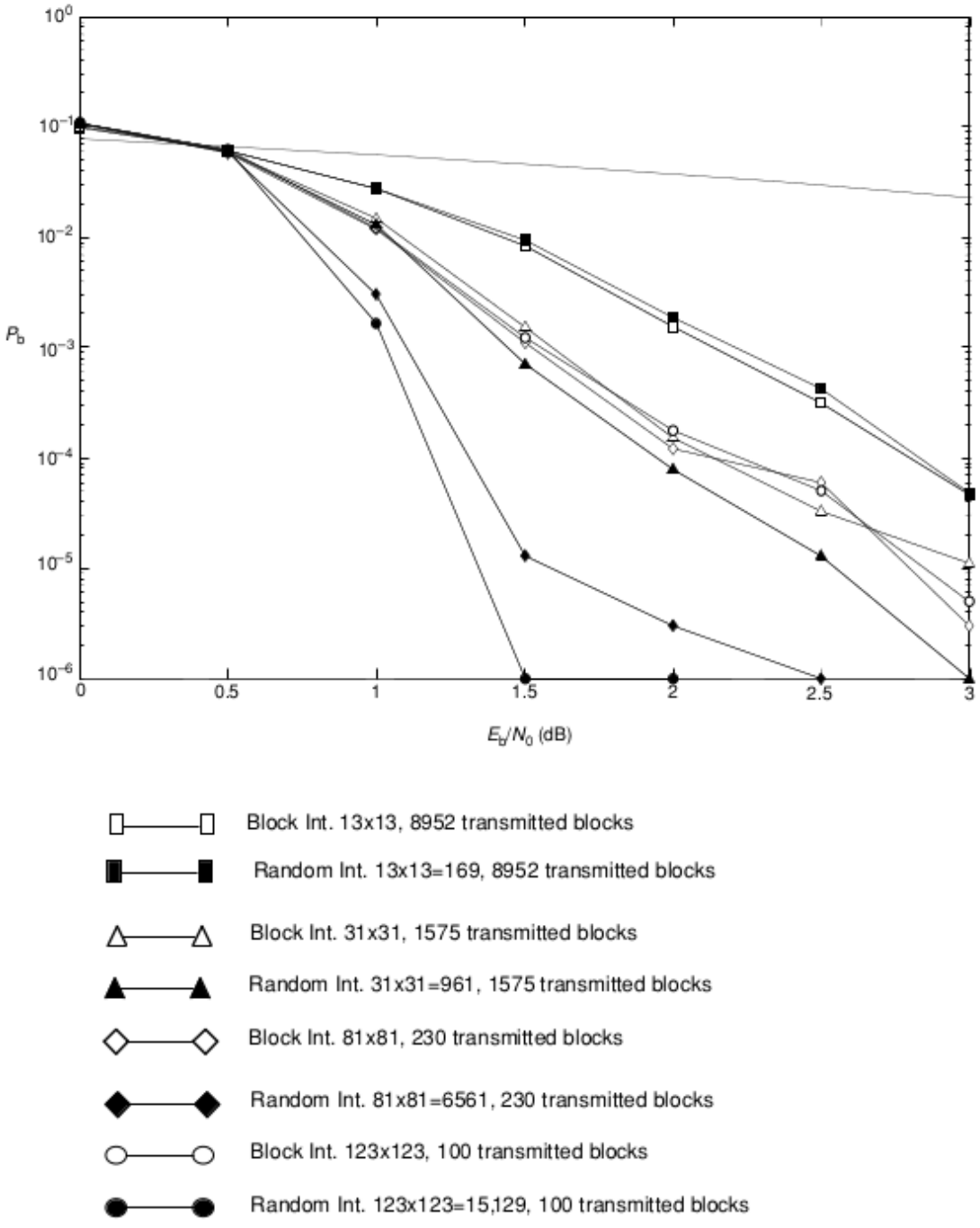


Figure 7.17 BER performance of a turbo code as a function of the type and size of the interleaver

Figure 25: Improvement achieved by interleavers [9, p. 252]

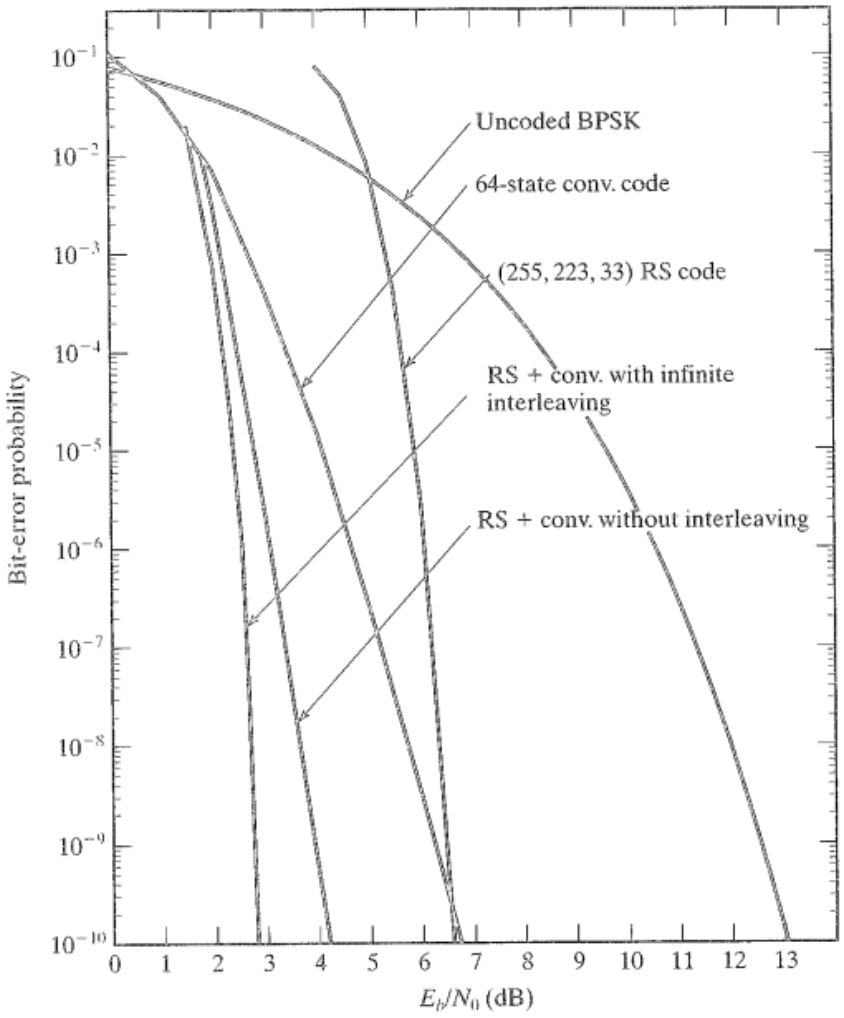


FIGURE 15.12: Error performance of the error-control coding scheme used in the NASA Tracking Data Relay Satellite System.

Figure 26: Improvement achieved by single-level concatenation [7, p. 761]. Interleaving effects can be seen here too

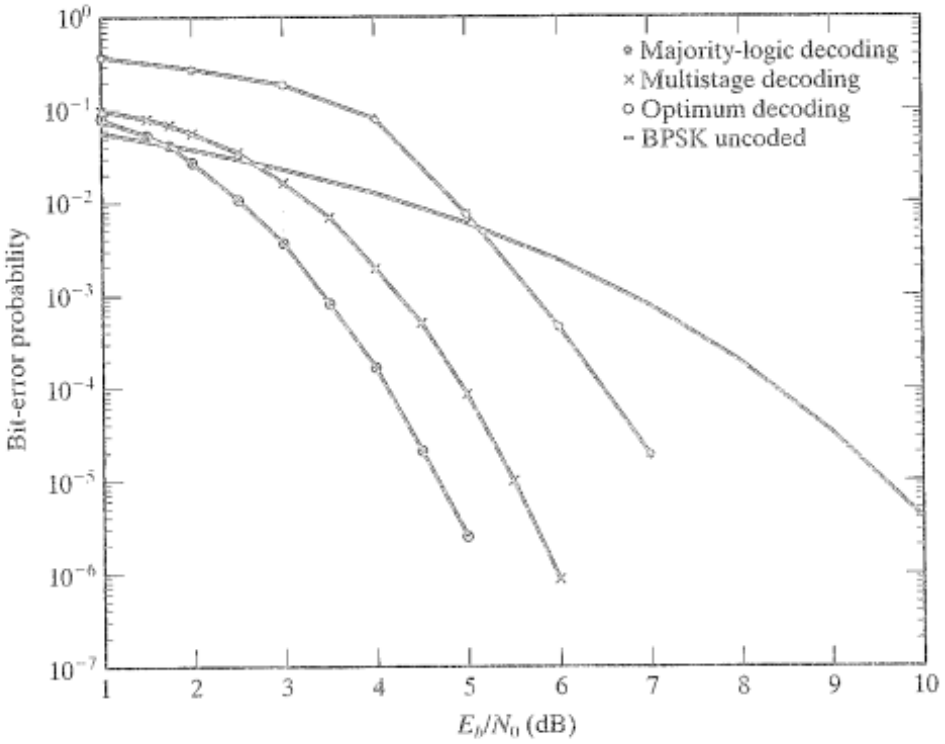


FIGURE 15.8: The error performance of the (128, 99) RM code with various decodings.

Figure 27: Improvement achieved by multilevel concatenated schemes. In this case, the same ECC but different decoders are concatenated [7, p. 753]

C.2 ECC families: coding gain

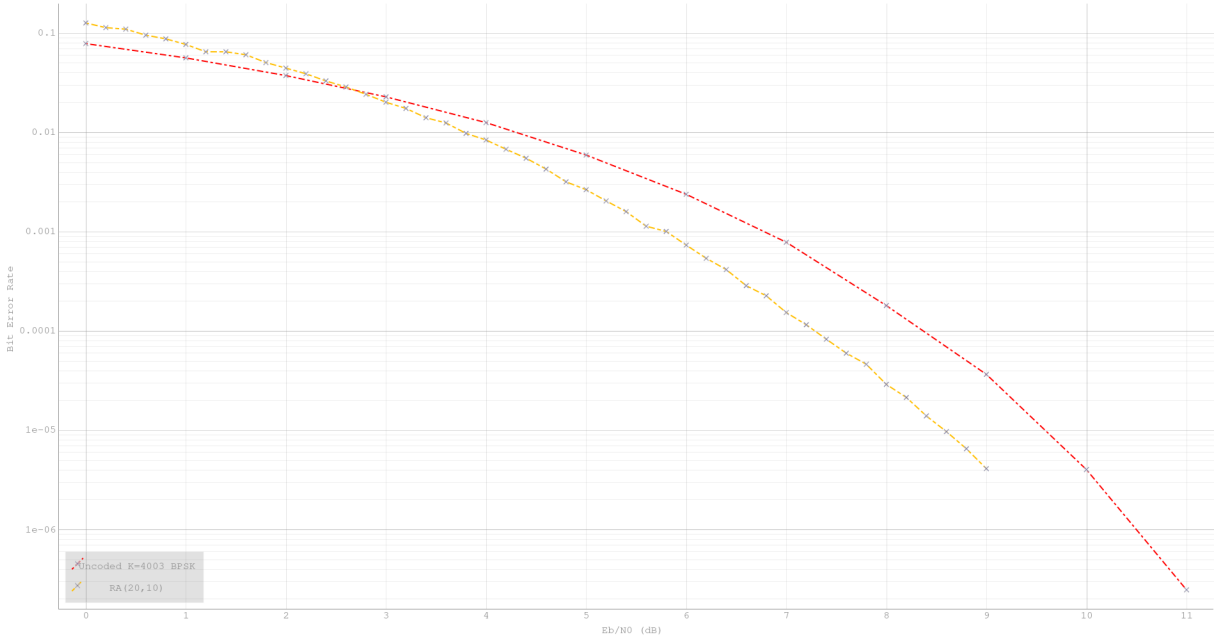


Figure 28: RA codes: coding gain [42]

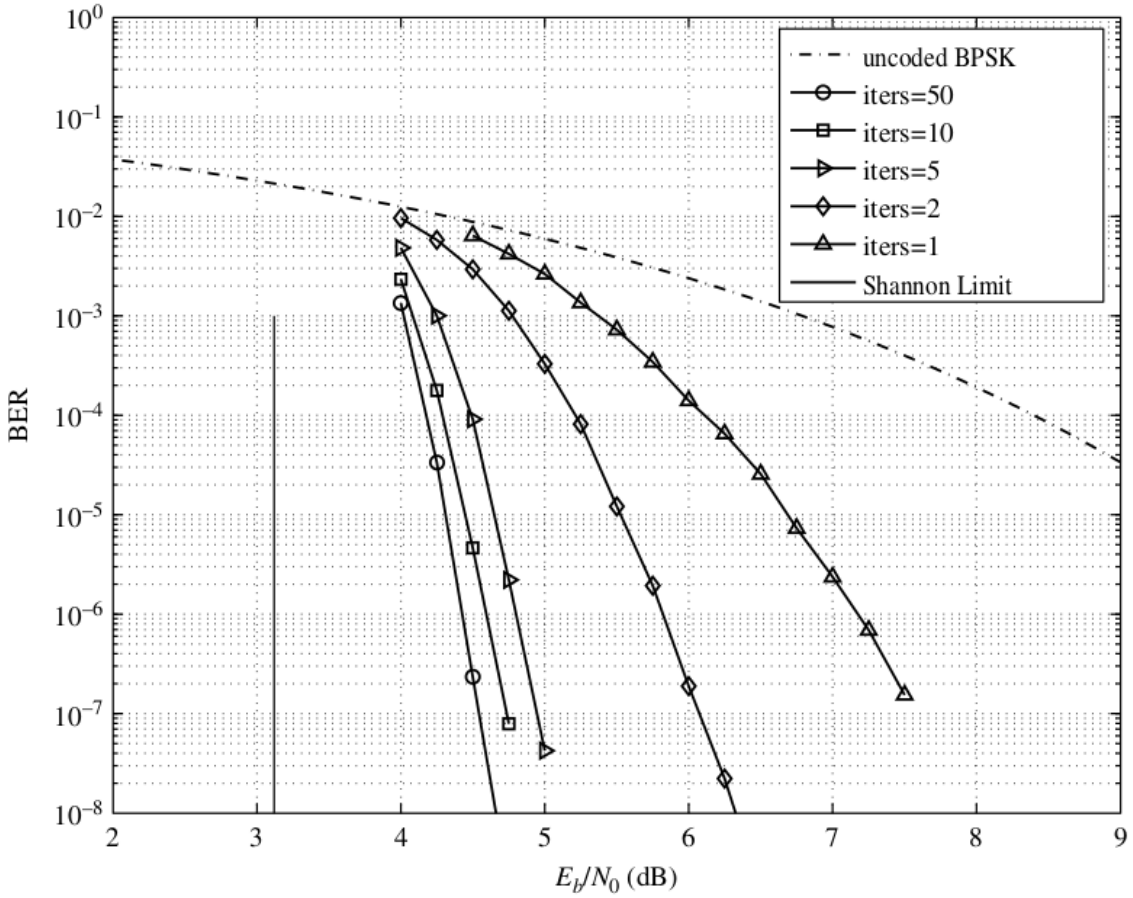


Figure 29: IRA codes: coding gain [41, p. 274]

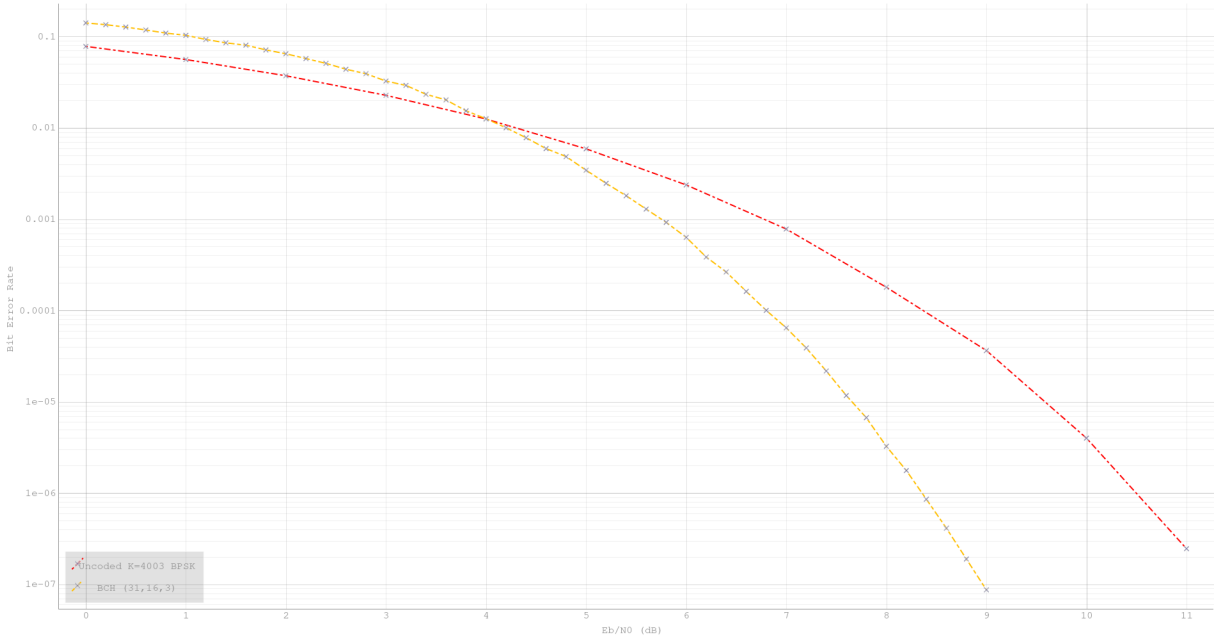


Figure 30: BCH codes: coding gain [42]

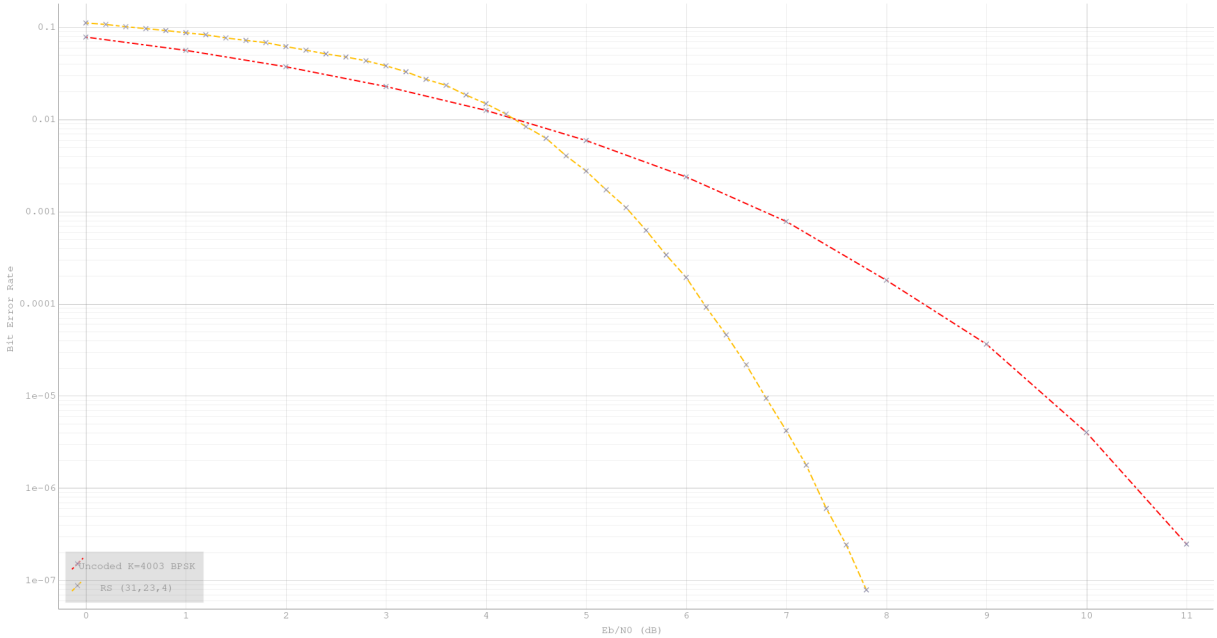


Figure 31: RS codes: coding gain [42]

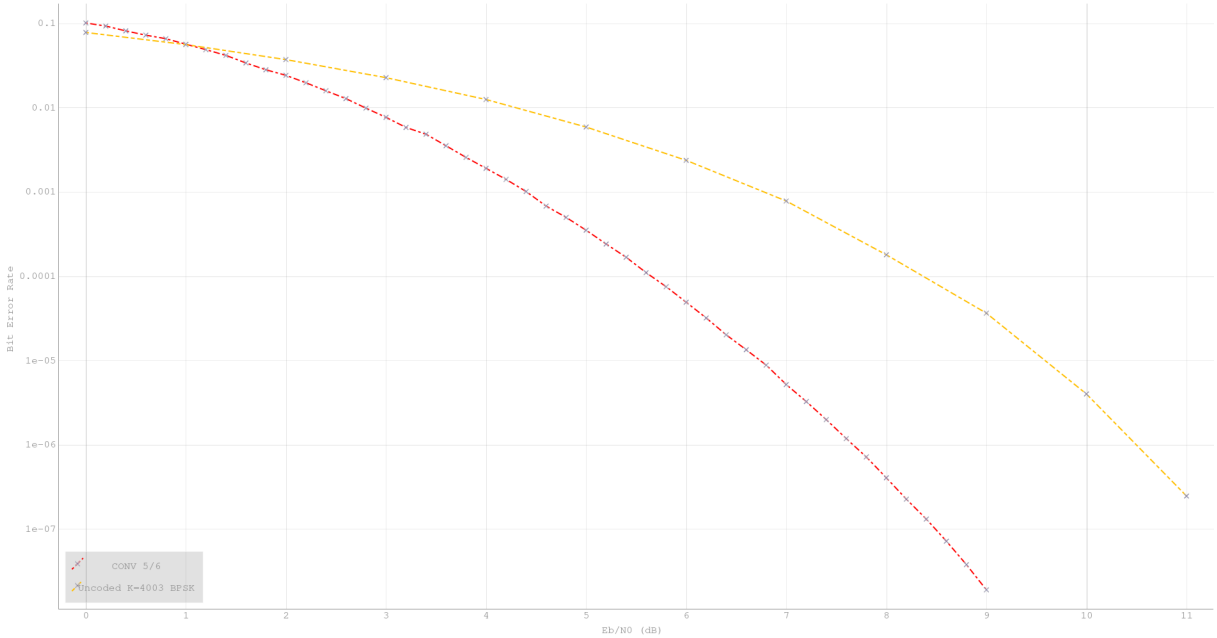


Figure 32: Convolutional codes: coding gain [42]

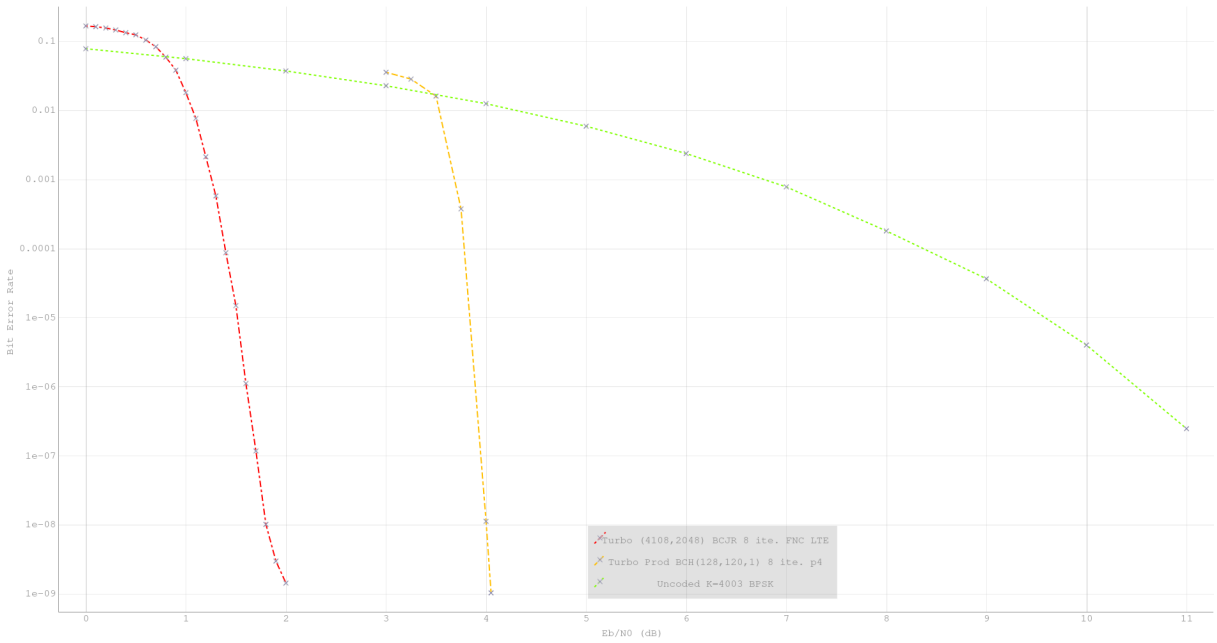


Figure 33: Turbo and turboproduct codes: coding gain [42]

;

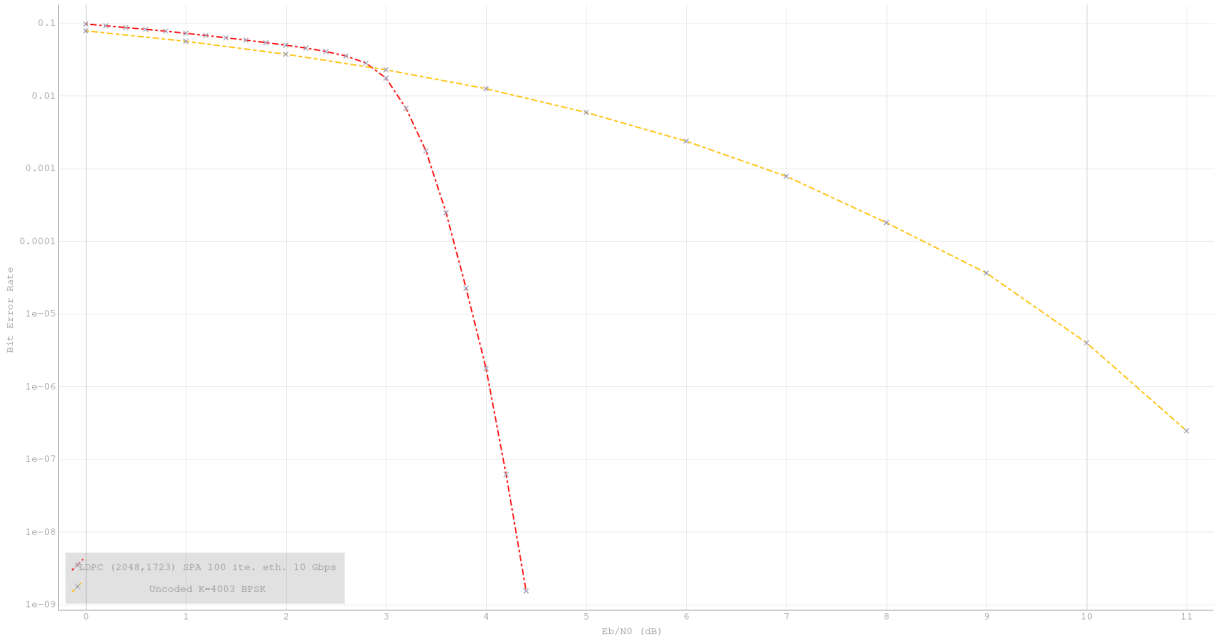


Figure 34: LDPC: coding gain [42]

D Considered BCH and RS

N, K is the notation used throughout the document. D is the minimum distance, T the correction capability in bits (in symbols when RS), M is the number of bits per symbol (when RS), D is the minimum distance and K/N is the code rate.

BCH (N,K,D,T,K/N) RS (N,K,T,M,D,K/N)

BCH	N	K	D	T	K/N
	7	4	3	1	0.571428571
	15	11	3	1	0.733333333
	15	7	5	2	0.466666667
	15	5	7	3	0.333333333
	31	26	3	1	0.838709677
	31	21	5	2	0.677419355
	31	16	7	3	0.516129032
	31	11	11	5	0.35483871
	31	6	15	7	0.193548387
	63	57	3	1	0.904761905
	63	51	5	2	0.80952381
	63	45	7	3	0.714285714
	63	39	9	4	0.619047619
	63	36	11	5	0.571428571
	63	30	13	6	0.476190476
	63	24	15	7	0.380952381
	63	18	21	10	0.285714286
	63	16	23	11	0.253968254
	63	10	27	13	0.158730159
	63	7	31	15	0.111111111
	127	120	3	1	0.94488189
	127	113	5	2	0.88976378
	127	106	7	3	0.834645669
	127	99	9	4	0.779527559
	127	92	11	5	0.724409449
	127	85	13	6	0.669291339
	127	78	15	7	0.614173228
	127	71	19	9	0.559055118
	127	64	21	10	0.503937008
	127	57	23	11	0.448818898
	127	50	27	13	0.393700787
	127	43	29	14	0.338582677
	127	36	31	15	0.283464567
	127	29	43	21	0.228346457
	127	22	47	23	0.173228346
	127	15	55	27	0.118110236
	127	8	63	31	0.062992126
	255	247	3	1	0.968627451
	255	239	5	2	0.937254902
	255	231	7	3	0.905882353
	255	223	9	4	0.874509804
	255	215	11	5	0.843137255
	255	207	13	6	0.811764706
	255	199	15	7	0.780392157
	255	191	17	8	0.749019608
	255	187	19	9	0.733333333
	255	179	21	10	0.701960784
	255	171	23	11	0.670588235

BCH (N,K,D,T,K/N)						RS (N,K,T,M,D,K/N)
BCH	255	163	25	12	0.639215686	
	255	155	27	13	0.607843137	
	255	147	29	14	0.576470588	
	255	139	31	15	0.545098039	
	255	131	37	18	0.51372549	
	255	123	39	19	0.482352941	
	255	115	43	21	0.450980392	
	255	107	45	22	0.419607843	
	255	99	47	23	0.388235294	
	255	91	51	25	0.356862745	
	255	87	53	26	0.341176471	
	255	79	55	27	0.309803922	
	255	71	59	29	0.278431373	
	255	63	61	30	0.247058824	
	255	55	63	31	0.215686275	
	255	47	85	42	0.184313725	
	255	45	87	43	0.176470588	
	255	37	91	45	0.145098039	
	255	29	95	47	0.11372549	
	255	21	111	55	0.082352941	
	255	13	119	59	0.050980392	
	255	9	127	63	0.035294118	
	511	502	3	1	0.982387476	
	511	493	5	2	0.964774951	
	511	484	7	3	0.947162427	
	511	475	9	4	0.929549902	
	511	466	11	5	0.911937378	
	511	457	13	6	0.894324853	
	511	448	15	7	0.876712329	
	511	439	17	8	0.859099804	
	511	430	19	9	0.84148728	
	511	421	21	10	0.823874755	
	511	412	23	11	0.806262231	
	511	403	25	12	0.788649706	
	511	394	27	13	0.771037182	
	511	385	29	14	0.753424658	
	511	376	31	15	0.735812133	
	511	367	35	17	0.718199609	
	511	358	37	18	0.700587084	
	511	349	39	19	0.68297456	
	511	340	41	20	0.665362035	
	511	331	43	21	0.647749511	
	511	322	45	22	0.630136986	
	511	313	47	23	0.612524462	
	511	304	51	25	0.594911937	
	511	295	53	26	0.577299413	
	511	286	55	27	0.559686888	
	511	277	57	28	0.542074364	
	511	268	59	29	0.52446184	

BCH (N,K,D,T,K/N)						RS (N,K,T,M,D,K/N)
BCH	511	259	61	30	0.506849315	
	511	250	63	31	0.489236791	
	511	241	73	36	0.471624266	
	511	238	75	37	0.465753425	
	511	229	77	38	0.4481409	
	511	220	79	39	0.430528376	
	511	211	83	41	0.412915851	
	511	202	85	42	0.395303327	
	511	193	87	43	0.377690802	
	511	184	91	45	0.360078278	
	511	175	93	46	0.342465753	
	511	166	95	47	0.324853229	
	511	157	103	51	0.307240705	
	511	148	107	53	0.28962818	
	511	139	109	54	0.272015656	
	511	130	111	55	0.254403131	
	511	121	117	58	0.236790607	
	511	112	119	59	0.219178082	
	511	103	123	61	0.201565558	
	511	94	125	62	0.183953033	
	511	85	127	63	0.166340509	
	511	76	171	85	0.148727984	
	511	67	175	87	0.13111546	
	511	58	183	91	0.113502935	
	511	49	187	93	0.095890411	
	511	40	191	95	0.078277886	
	511	31	219	109	0.060665362	
	511	28	223	111	0.054794521	
	511	19	239	119	0.037181996	
	511	10	255	127	0.019569472	
	1023	1013	3	1	0.990224829	
	1023	1003	5	2	0.980449658	
	1023	993	7	3	0.970674487	
	1023	983	9	4	0.960899316	
	1023	973	11	5	0.951124145	
	1023	963	13	6	0.941348974	
	1023	953	15	7	0.931573803	
	1023	943	17	8	0.921798631	
	1023	933	19	9	0.91202346	
	1023	923	21	10	0.902248289	
	1023	913	23	11	0.892473118	
	1023	903	25	12	0.882697947	
	1023	893	27	13	0.872922776	
	1023	883	29	14	0.863147605	
	1023	873	31	15	0.853372434	
	1023	863	33	16	0.843597263	
	1023	858	35	17	0.838709677	
	1023	848	37	18	0.828934506	
	1023	838	39	19	0.819159335	

BCH (N,K,D,T,K/N) RS (N,K,T,M,D,K/N)

BCH	1023	828	41	20	0.809384164
	1023	818	43	21	0.799608993
	1023	808	45	22	0.789833822
	1023	798	47	23	0.780058651
	1023	788	49	24	0.77028348
	1023	778	51	25	0.760508309
	1023	768	53	26	0.750733138
	1023	758	55	27	0.740957967
	1023	748	57	28	0.731182796
	1023	738	59	29	0.721407625
	1023	728	61	30	0.711632454
	1023	718	63	31	0.701857283
	1023	708	69	34	0.692082111
	1023	698	71	35	0.68230694
	1023	688	73	36	0.672531769
	1023	678	77	37	0.662756598
	1023	668	77	38	0.652981427
	1023	658	79	39	0.643206256
	1023	648	83	41	0.633431085
	1023	638	85	42	0.623655914
	1023	628	87	43	0.613880743
	1023	618	89	44	0.604105572
	1023	608	91	45	0.594330401
	1023	598	93	46	0.58455523
	1023	588	95	47	0.574780059
	1023	578	99	49	0.565004888
	1023	573	101	50	0.560117302
	1023	563	103	51	0.550342131
	1023	553	105	52	0.54056696
	1023	543	107	53	0.530791789
	1023	533	109	54	0.521016618
	1023	523	111	55	0.511241447
	1023	513	115	57	0.501466276
	1023	503	117	58	0.491691105
	1023	493	119	59	0.481915934
	1023	483	121	60	0.472140762
	1023	473	123	61	0.462365591
	1023	463	125	62	0.45259042
	1023	453	127	63	0.442815249
	1023	443	147	73	0.433040078
	1023	433	149	74	0.423264907
	1023	423	151	75	0.413489736
	1023	413	155	77	0.403714565
	1023	403	157	78	0.393939394
	1023	393	159	79	0.384164223
	1023	383	165	82	0.374389052
	1023	378	167	83	0.369501466
	1023	368	171	85	0.359726295
	1023	358	173	86	0.349951124

BCH (N,K,D,T,K/N)					RS (N,K,T,M,D,K/N)
1023	348	175	87	0.340175953	
1023	338	179	89	0.330400782	
1023	328	181	90	0.320625611	
1023	318	183	91	0.31085044	
1023	308	187	93	0.301075269	
1023	298	189	94	0.291300098	
1023	288	191	95	0.281524927	
1023	278	205	102	0.271749756	
1023	268	207	103	0.261974585	
1023	258	213	106	0.252199413	

BCH (N,K,D,T,K/N) RS (N,K,T,M,D,K/N)

RS	N	K	T	M	D	K/N
	7	3	2	3	5	0.428571429
	7	5	1	3	3	0.714285714
	15	3	6	4	13	0.2
	15	5	5	4	11	0.333333333
	15	7	4	4	9	0.466666667
	15	9	3	4	7	0.6
	15	11	2	4	5	0.733333333
	15	13	1	4	3	0.866666667
	31	15	8	5	17	0.483870968
	31	17	7	5	15	0.548387097
	31	19	6	5	13	0.612903226
	31	21	5	5	11	0.677419355
	31	23	4	5	9	0.741935484
	31	25	3	5	7	0.806451613
	31	27	2	5	5	0.870967742
	31	29	1	5	3	0.935483871
	63	31	16	6	33	0.492063492
	63	41	11	6	23	0.650793651
	63	43	10	6	21	0.682539683
	63	45	9	6	19	0.714285714
	63	47	8	6	17	0.746031746
	63	49	7	6	15	0.777777778
	63	51	6	6	13	0.80952381
	63	53	5	6	11	0.841269841
	63	55	4	6	9	0.873015873
	63	57	3	6	7	0.904761905
	63	59	2	6	5	0.936507937
	63	61	1	6	3	0.968253968
	127	67	30	7	61	0.527559055
	127	87	20	7	41	0.68503937
	127	99	14	7	29	0.779527559
	127	101	13	7	27	0.795275591
	127	103	12	7	25	0.811023622
	127	105	11	7	23	0.826771654
	127	107	10	7	21	0.842519685
	127	109	9	7	19	0.858267717
	127	111	8	7	17	0.874015748
	127	113	7	7	15	0.88976378
	127	115	6	7	13	0.905511811
	127	117	5	7	11	0.921259843
	127	119	4	7	9	0.937007874
	127	121	3	7	7	0.952755906
	127	123	2	7	5	0.968503937
	127	125	1	7	3	0.984251969
	255	155	50	8	101	0.607843137
	255	175	40	8	81	0.68627451
	255	191	32	8	65	0.749019608
	255	193	31	8	63	0.756862745

BCH (N,K,D,T,K/N)					RS (N,K,T,M,D,K/N)	
RS	255	195	30	8	61	0.764705882
	255	197	29	8	59	0.77254902
	255	199	28	8	57	0.780392157
	255	201	27	8	55	0.788235294
	255	203	26	8	53	0.796078431
	255	205	25	8	51	0.803921569
	255	207	24	8	49	0.811764706
	255	209	23	8	47	0.819607843
	255	211	22	8	45	0.82745098
	255	213	21	8	43	0.835294118
	255	215	20	8	41	0.843137255
	255	217	19	8	39	0.850980392
	255	219	18	8	37	0.858823529
	255	221	17	8	35	0.866666667
	255	223	16	8	33	0.874509804
	255	225	15	8	31	0.882352941
	255	227	14	8	29	0.890196078
	255	229	13	8	27	0.898039216
	255	231	12	8	25	0.905882353
	255	233	11	8	23	0.91372549
	255	235	10	8	21	0.921568627
	255	237	9	8	19	0.929411765
	255	239	8	8	17	0.937254902
	255	241	7	8	15	0.945098039
	255	243	6	8	13	0.952941176
	255	245	5	8	11	0.960784314
	255	247	4	8	9	0.968627451
	255	249	3	8	7	0.976470588
	255	251	2	8	5	0.984313725
	255	253	1	8	3	0.992156863
	511	311	100	9	201	0.608610568
	511	331	90	9	181	0.647749511
	511	351	80	9	161	0.686888454
	511	371	70	9	141	0.726027397
	511	391	60	9	121	0.765166341
	511	411	50	9	101	0.804305284
	511	431	40	9	81	0.843444227
	511	451	30	9	61	0.88258317
	511	453	29	9	59	0.886497065
	511	455	28	9	57	0.890410959
	511	457	27	9	55	0.894324853
	511	459	26	9	53	0.898238748
	511	461	25	9	51	0.902152642
	511	463	24	9	49	0.906066536
	511	465	23	9	47	0.909980431
	511	467	22	9	45	0.913894325
	511	469	21	9	43	0.917808219
	511	471	20	9	41	0.921722114
	511	473	19	9	39	0.925636008

BCH (N,K,D,T,K/N)						RS (N,K,T,M,D,K/N)
RS	511	475	18	9	37	0.929549902
	511	477	17	9	35	0.933463796
	511	479	16	9	33	0.937377691
	511	481	15	9	31	0.941291585
	511	483	14	9	29	0.945205479
	511	485	13	9	27	0.949119374
	511	487	12	9	25	0.953033268
	511	489	11	9	23	0.956947162
	511	491	10	9	21	0.960861057
	511	493	9	9	19	0.964774951
	511	495	8	9	17	0.968688845
	511	497	7	9	15	0.97260274
	511	499	6	9	13	0.976516634
	511	501	5	9	11	0.980430528
	511	503	4	9	9	0.984344423
	511	505	3	9	7	0.988258317
	511	507	2	9	5	0.992172211
	511	509	1	9	3	0.996086106

E Berkelamp algorithm

E.1 Berkelamp algorithm for BCH

This appendix presents the mathematical principles that constitute the fundamental basis of the decoder.

The Berkelamp algorithm iterates among two functions, $\Lambda_i^r(x)$ and and one value in GF(2^m). The most important polynomial is lambda, which, at the conclusion of the algorithm, becomes the error locator polynomial. B is another polynomial that is mathematically representative but merely auxiliary in this context.

The iteration variable is represented by r, and furthermore, r is constrained to the range $r \in [0, 2t]$. The employed notation states that the superindex variables represent the iteration, while the subindex variables signify the coefficients of a polynomial or the value at the iteration denoted. For example, Λ_j^r denotes the j-th coefficient at the r-th iteration and Δ_r denotes the value of Δ at the r-th iteration.

The formal description is as follows:

$$\begin{cases} \Delta_r = \sum_{j=0}^{L_{r-1}} \Lambda_j^{r-1} S_{r-j} \\ L_r = \delta_r(r - L_{r-1}) + L_{r-1}(1 - \delta_r) \\ \begin{pmatrix} \Lambda^r(x) \\ B^r(x) \end{pmatrix} = \begin{pmatrix} 1 & -\Delta_r x \\ \Delta_r^{-1}\delta_r & (1-\delta_r)x \end{pmatrix} \begin{pmatrix} \Lambda^{r-1}(x) \\ B^{r-1}(x) \end{pmatrix} \end{cases} \tag{33}$$

However it can be proven that:

$$\begin{cases} \Lambda^r(x) = \Lambda^{r-1}(x) - \Delta_m^{-1}\Delta_r x^{r-m}\Lambda^{m-1} \\ B^m(x) = \Delta_m^{-1}\Lambda^{m-1} \end{cases} \tag{34}$$

Where m is the last iteration which satisfies $L_r > L_{r-1}$ and $\delta_r = 1 \Leftrightarrow \Delta_r \neq 0$ and $2L_{r-1} \leq r - 1$. In other cases, $\delta_r = 0$. The initial values, the 0-th iteration is:

$$\Lambda^0(x) = 1 \qquad B^0(x) = 1 \qquad L_0 = 0 \tag{35}$$

The code of the example is a BCH(7,4), which means that its correction factor is $t = 1$ and there will be $2t = 2$ iterations. The syndrome is $\vec{S} = (\alpha^2, \alpha^4)$

1st iteration

$$\Delta_1 = \sum_{j=0}^{L_0} \Lambda_j^0 S_{1-j} = S_1 = \alpha^6 \tag{36}$$

$\Delta_1 \neq 0 \Rightarrow \delta_1 = 1$

$$L_1 = \delta_1(1 - L_0) + L_0(1 - \delta_1) = 1 \tag{37}$$

$$\Lambda^1(x) = \Lambda^0(x) - \Delta_m^{-1}\Delta_1 x^{1-m}\Lambda^{m-1} = 1 + \alpha^6 x \tag{38}$$

$L_1 > L_0 \Rightarrow m = 1$

$$B^1(x) = \Delta_1^{-1}\Lambda^{1-1}(x)\ B^1(x) = \alpha^9 \tag{39}$$

2nd iteration

$$\Delta_2 = \sum_{j=0}^{L_1} \Lambda_j^1 S_{1-j} = \Lambda_0^1 S_2 + \Lambda_1^1 S_1 = \alpha^{12} + \alpha^6 \alpha^6 = 0 \tag{40}$$

$2L_{2-1} > r - 1 \Rightarrow \delta_2 = 1$

$$L_2 = \delta_2(1 - L_0) + L_1(1 - \delta_2) = 1 \tag{41}$$

$$\Lambda^2(x) = \Lambda^1(x) - \Delta_1^{-1} \Delta_1 x^{1-1} \Lambda^{1-1} = 1 + \alpha^6 x \tag{42}$$

$L_2 = L_1 \Rightarrow m = 1$

$$B^2(x) = xB^1(x) \; B^2(x) = x\alpha^9 \tag{43}$$

The error locator polynomial is $\Lambda^2(x) = 1 + \alpha^6$. The maximum number of steps that are required is $2t$. However, on occasion, the error locator is found before this point.

E.2 Berkelamp algorithm for Reed - Solomon

The RS decoding algorithm is identical to the BCH previously described, including the initial values of $\Lambda(x)$, $B^0(x)$ and $L_0(x)$. The main difference is that, on this occasion, is required $2t = 4$ iterations instead of the two of the BCH.

1st iteration

$$\Delta_1 = \sum_{j=0}^{L_0} \Lambda_j^0 S_{1-j} = S_1 = \alpha^3 \tag{44}$$

$\Delta_1 \neq 0 \Rightarrow \delta_1 = 1$

$$L_1 = \delta_1(1 - L_0) + L_0(1 - \delta_1) = 1 \tag{45}$$

$$\Lambda^1(x) = \Lambda^0(x) - \Delta_m^{-1} \Delta_1 x^{1-m} \Lambda^{m-1} = 1 + \alpha^3 x \tag{46}$$

$L_1 > L_0 \Rightarrow m = 1$

$$B^1(x) = \Delta_1^{-1} \Lambda^{1-1}(x) \; B^1(x) = \alpha^4 \tag{47}$$

2nd iteration

$$\Delta_2 = \sum_{j=0}^{L_1} \Lambda_j^1 S_{1-j} = \Lambda_0^1 S_2 + \Lambda_1^1 S_1 = \alpha + \alpha^6 = \alpha^5 \tag{48}$$

$\Delta_2 \neq 0 \Rightarrow \delta_1 = 1$

$$L_2 = \delta_2(2 - L_1) + L_1(2 - \delta_2) = 1 \tag{49}$$

$$\Lambda^2(x) = \Lambda^1(x) - \Delta_m^{-1} \Delta_2 x^{2-m} \Lambda^{m-1} = 1 + \alpha^5 x \tag{50}$$

$L_2 = L_1 \Rightarrow m = 1$

$$B^2(x) = xB^1(x) \; B^2(x) = \alpha^4 x \tag{51}$$

3rd iteration

$$\Delta_3 = \sum_{j=0}^{L_2} \Lambda_j^0 S_{1-j} = \Lambda_0^2 S_3 + \Lambda_1^2 S_2 = \alpha^5 + \alpha^6 = \alpha \tag{52}$$

$\Delta_3 \neq 0 \Rightarrow \delta_3 = 1$

$$L_3 = \delta_3(3 - L_2) + L_2(3 - \delta_3) = 2 \tag{53}$$

$$\Lambda^3(x) = \Lambda^2(x) - \Delta_m^{-1} \Delta_3 x^{3-m} \Lambda^{1-1} = 1 + \alpha^5 x + \alpha^5 x^2 \tag{54}$$

$L_3 > L_2 \Rightarrow m = 3$

$$B^3(x) = \Delta_3^{-1} \Lambda^{3-1}(x) \; B^3(x) = \alpha^2 + x \tag{55}$$

4th iteration

$$\Delta_4 = \sum_{j=0}^{L_3} \Lambda_j^0 S_{1-j} = \Lambda_0^3 S_4 + \Lambda_1^3 S_3 + \Lambda_2^3 S_2 = \alpha^4 + \alpha^3 + \alpha^6 = 0 \tag{56}$$

$2L_{r-1} = 4 > 4 - 1 \Rightarrow \delta_4 = 1$

$$L_4 = \delta_4(4 - L_4) + L_4(4 - \delta_4) = 2 \tag{57}$$

$$\Lambda^4(x) = \Lambda^3(x) - \Delta_m^{-1}\Delta_4 x^{4-m}\Lambda^{m-1} = 1 + \alpha^5 x + \alpha^5 x^2 \tag{58}$$

$L_4 = L_3 \Rightarrow m = 3$

$$B^4(x) = xB^3(x)\ B^4(x) = \alpha^6 + \alpha^4 x \tag{59}$$

References

[1] "Search - NASA Technical Reports Server (NTRS)." [Online; accessed 25. Mar. 2024]. (Mar. 2024), [Online]. Available: https://ntrs.nasa.gov/search?q=NASA%2FTP-20210000201.

[2] "QBITO, el primer CubeSat de la UPM, camino de la Estación Espacial Internacional – E-Politécnica." [Online; accessed 26. Mar. 2024]. (Mar. 2024), [Online]. Available: https://www.upm.es/e-politecnica/qbito-el-primer-cubesat-de-la-upm-camino-de-la-estacion-espacial-internacional.

[3] https://www. jpl. nasa. gov/. "Mars Cube One (MarCO)." [Online; accessed 26. Mar. 2024]. (Mar. 2024), [Online]. Available: https://www.jpl.nasa.gov/missions/mars-cube-one-marco.

[4] C. P. Office, "Cubesat design specification, rev. 14," *CubeSat Design Specification*, vol. Rev. 14, no. 1, Feb. 2022. [Online]. Available: https://www.cubesat.org/s/CDS-REV14_1-2022-02-09.pdf.

[5] E. Howell, "This Hitchhiking Satellite – Launching Today – Is Small Enough To Fit In Your Hand," *Universe Today*, Dec. 2015. [Online]. Available: https://www.universetoday.com/106542/this-hitchhiking-satellite-launching-today-is-small-enough-to-fit-in-your-hand.

[6] "Internet of Things Global Standards Initiative." [Online; accessed 25. Mar. 2024]. (Mar. 2024), [Online]. Available: https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx.

[7] S. Lin and D. J. Costello, *Error control coding: Fundamentals and applications*, 2nd edition. Upper Saddle River, New Jersey: Pentice-Hall, 2002.

[8] Bernard Sklar and Fred Harris, *Digital Communications Fundamentals and Applications*, 2nd. Upper Saddle River, New Jersey: Prentice Hall, 2021, ISBN: 978-0-13-458856-8.

[9] J. C. Moreira and P. G. Farrell, *Essentials of Error-Control Coding*, Ilustred edition. Chichester, England: John Wiley & Sons, 2006, ISBN: 978-0-470-02920-6.

[10] "Hydra Space." [Online; accessed 27. Mar. 2024]. (Mar. 2024), [Online]. Available: https://hydra-space.com.

[11] E. Alonso, *Small satellite platform for inorbit testing and agile innovation*, 2024.

[12] Contributors to Wikimedia projects. "Automated theorem proving - wikipedia." [Online; accessed 18. Apr. 2024]. (Feb. 2024), [Online]. Available: https://en.wikipedia.org/w/index.php?title=Automated_theorem_proving&oldid=1205993341.

[13] Contributors to Wikimedia projects. "Halting problem - wikipedia." [Online; accessed 18. Apr. 2024]. (Apr. 2024), [Online]. Available: https://en.wikipedia.org/w/index.php?title=Halting_problem&oldid=1216903566.

[14] datacom-teracom. "Pmccabe." [Online; accessed 18. Apr. 2024]. (Apr. 2024), [Online]. Available: https://github.com/datacom-teracom/pmccabe.

[15] terryyin. "Lizard." [Online; accessed 18. Apr. 2024]. (Apr. 2024), [Online]. Available: https://github.com/terryyin/lizard.

[16] "Complexity - compute a complexity measure for c code." [Online; accessed 18. Apr. 2024]. (Apr. 2024), [Online]. Available: https://www.gnu.org/software/complexity.

[17] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[18] M. J. Golay, *Notes on digital coding*, 1949. [Online]. Available: https://pierre-hyvernat.apps.math.cnrs.fr/data/Enseignement/2223/info602/TP-Golay/golay_paper.pdf.

[19] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[20] P. Elias, "Error-free coding," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 29–37, 1954. DOI: 10.1109/TIT.1954.1057464.

[21] P. Elias, "Coding for two noisy channels," in 1955, ISBN: 978-1-4419-7969-8. DOI: 10.1007/978-1-4419-7970-4_14. [Online]. Available: http://web.mit.edu/6.441/www/reading/hd2.pdf.

[22] J. Allen-Robertson, "The materiality of digital media: The hard disk drive, phonograph, magnetic tape and optical media in technical close-up," *New Media and amp Society*, Sep. 2015. DOI: 10.1177/1461444815606368.

[23] "Sputnik." [Online; accessed 28. Mar. 2024]. (Mar. 2024), [Online]. Available: https://www.nasa.gov/history/sputnik/index.html.

[24] A. Hocquenghem, "Codes correcteurs d'erreurs," French, *Chiffres*, vol. 2, pp. 147–156, Sep. 1959.

[25] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, no. 1, pp. 68–79, 1960, ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(60)90287-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0019995860902874.

[26] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960, ISSN: 03684245. [Online]. Available: http://www.jstor.org/stable/2098968 (visited on 28/03/2024).

[27] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961. DOI: 10.1109/JRPROC.1961.287814.

[28] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962. DOI: 10.1109/TIT.1962.1057683.

[29] G. D. Forney, "Concatenated codes," Technical Report, Research Laboratory of Electronics, MIT, Cambridge, Massachusetts, 1965.

[30] N. Abramson, "Cascade decoding of cyclic product codes," *IEEE Transactions on Communication Technology*, vol. 16, no. 3, pp. 398–402, 1968. DOI: 10.1109/TCOM.1968.1089859.

[31] M. Perlman and J.-J. Lee, *The reed-solomon encoders: Conventional versus berlekamp's architecture*, NASA Technical Reports Server (NTRS), Disponible en: https://ntrs.nasa.gov/citations/19830008870, Dec. 1982.

[32] "Red book: Audio recording compact disc digital audio system, iec 60908, second edition, 1999 02 [isbn 2 8318 4638 2] : Free download, borrow, and streaming : Internet archive." [Online; accessed 28. Mar. 2024]. (Mar. 2024), [Online]. Available: https://tinyurl.com/ynjnn8ax.

[33] C. Berrou, A. Glavieux and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, 1993, 1064–1070 vol.2. DOI: 10.1109/ICC.1993.397441.

[34] C. Berrou, "Error-correction coding method with at least two systematic convolutional codings in parallel, corresponding iterative decoding method, decoding module and decoder," Patent US5446747A, 1995.

[35] A. Abbasfar, D. Divsalar and K. Yao, "Accumulate-repeat-accumulate codes," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 692–702, 2007. DOI: 10.1109/TCOMM.2007.894109.

[36] "Cubesat." [Online; accessed 28. Mar. 2024]. (Mar. 2024), [Online]. Available: https://www.cubesat.org/past-launches/2015/6/22/eurorockot-2003.

[37] "Etsi." [Online; accessed 29. Mar. 2024]. (Mar. 2024), [Online]. Available: https://www.etsi.org/technologies/dvb-s-s2.

[38] S. Dahmen-Lhuissier. "Etsi." [Online; accessed 29. Mar. 2024]. (Mar. 2024), [Online]. Available: https://www.etsi.org/technologies/mobile/4g.

[39] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes," in *2008 IEEE International Symposium on Information Theory*, 2008, pp. 1173–1177. DOI: 10.1109/ISIT.2008.4595172.

[40] S. Dahmen-Lhuissier. "Etsi." [Online; accessed 29. Mar. 2024]. (Mar. 2024), [Online]. Available: https://www.etsi.org/technologies/mobile/5g.

[41] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*, 1st. Cambridge University Press, 2009. [Online]. Available: https://www.researchgate.net/publication/265821157_Channel_Codes_Classical_and_Modern.

[42] A. Cassagne, O. Hartmann, M. Léonardon *et al.*, "Aff3ct: A fast forward error correction toolbox!" *Elsevier SoftwareX*, vol. 10, p. 100 345, Oct. 2019, ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2019.100345. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352711019300457.

[43] L. M. Tomás, "Comunicaciones móviles: Fundamentos de la interfaz radio," Comunicaciones Móviles (CMOV) 95000044 , Señales Sistemas y Radiocomunicaciones, Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT) - Universidad Politécnica de Madrid (UPM), 2024, 2024.

[44] A. Croisdale, "Error rates and error detection on telegraph circuits," *IRE Transactions on Communications Systems*, vol. 9, no. 1, pp. 28–37, 1961. DOI: 10.1109/TCOM.1961.1097653.

[45] Arsc. "Ccsds.org - the consultative committee for space data systems (ccsds)." [Online; accessed 27. Apr. 2024]. (Apr. 2024), [Online]. Available: https://public.ccsds.org/default.aspx.

[46] A. Modenini and K. Andrews, *Tc synchronization and channel coding*, Jul. 2021. [Online]. Available: https://public.ccsds.org/Pubs/231x0b2c1s.pdf.

[47] H. Jin, A. Kh and R. McEliece, "Irregular repeat accumulate codes," Nov. 2000.

[48] Contributors to Wikimedia projects. "Repeat-accumulate code - wikipedia." [Online; accessed 15. Apr. 2024]. (Dec. 2023), [Online]. Available: https://en.wikipedia.org/w/index.php?title=Repeat-accumulate_code&oldid=1188371837.

[49] E. R. Berlekamp, "Key papers in the development of coding theory," 1974. [Online]. Available: https://api.semanticscholar.org/CorpusID:60423350.

[50] N. Naimipour, H. Safavi and H. C. Shaw, "Document id 20190032244," in *International Astronautical Congress*, American Institute of Aeronautics and Astronautics (AIAA), International Astronautical Federation (IAF), Washington, D.C., United States, Oct. 2019.

[51] Colaboradores de los proyectos Wikimedia. "Detección y corrección de errores - wikipedia, la enciclopedia libre." [Online; accessed 22. Apr. 2024]. (Dec. 2023), [Online]. Available: https://es.wikipedia.org/w/index.php?title=Detecci%C3%B3n_y_correcci%C3%B3n_de_errores&oldid=156107193.

[52] M. Meenalakshmi, S. Chaturvedi and V. K. Dwivedi, "Deep learning-enabled polar code decoders for 5g networks and beyond," *AEU - International Journal of Electronics and Communications*, vol. 177, p. 155 220, 2024, ISSN: 1434-8411. DOI: https://doi.org/10.1016/j.aeue.2024.155220. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1434841124001055.

[53] B. Le Gal and C. Jego, "Low-latency and high-throughput software turbo decoders on multi-core architectures," *Annals of Telecommunications*, vol. 75, no. 1, pp. 27–42, 2020, ISSN: 1958-9395. DOI: 10.1007/s12243-019-00727-5. [Online]. Available: https://doi.org/10.1007/s12243-019-00727-5.

[54] S. Weithoffer, G. Aousaji, J. Nadal and C. A. Nour, "Iteration overlap for low-latency turbo decoding," in *2023 12th International Symposium on Topics in Coding (ISTC)*, 2023, pp. 1–5. DOI: 10.1109/ISTC57237.2023.10273532.

[55] H. Luo, Y. Zhang, W. Li *et al.*, "Low latency parallel turbo decoding implementation for future terrestrial broadcasting systems," *IEEE Transactions on Broadcasting*, vol. 64, no. 1, pp. 96–104, 2018. DOI: 10.1109/TBC.2017.2704425.

[56] M. Luby, M. Mitzenmacher, M. Shokrollahi and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 585–598, 2001. DOI: 10.1109/18.910576.

[57] K. S. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. R. Jones and F. Pollara, "The development of turbo and ldpc codes for deep-space applications," *Proceedings of the IEEE*, vol. 95, no. 11, pp. 2142–2156, 2007. DOI: 10.1109/JPROC.2007.905132.

[58] Contributors to Wikimedia projects. "Transport layer security - wikipedia." [Online; accessed 24. Apr. 2024]. (Apr. 2024), [Online]. Available: https://en.wikipedia.org/w/index.php?title=Transport_Layer_Security&oldid=1219026936.

[59] M. Donald, "How an ionizing particle from outer space helped a mario speedrunner save time," *TheGamer*, Sep. 2020. [Online]. Available: https://www.thegamer.com/how-ionizing-particle-outer-space-helped-super-mario-64-speedrunner-save-time.

[60] "Teknos." [Online; accessed 9. May 2024]. (May 2024), [Online]. Available: https://www.teknos.org/home/2022/8/17/corruption-and-correction-how-data-can-change-the-world.

[61] H. Ranter, *Accident airbus a330-303 vh-qpa,* Oct. 2008. [Online]. Available: https://aviation-safety.net/database/record.php?id=20081007-0.

[62] "Radiation: Satellites' unseen enemy." [Online; accessed 9. May 2024]. (May 2024), [Online]. Available: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Radiation_satellites_unseen_enemy.

[63] "The error correcting codes (ecc) page." [Online; accessed 27. Apr. 2024]. (Feb. 2023), [Online]. Available: https://www.eccpage.com.

[64] R. Meza Moreno, "El algoritmo de berlekamp-massey y decodificación de códigos bch sobre el anillo $\mathbb{Z}_{p^s}$," Spanish, M.S. thesis, Universidad Autónoma Metropolitana, Mexico, 4th May 2007. DOI: 10.24275/uami.2801pg402. [Online]. Available: https://doi.org/10.24275/uami.2801pg402 (visited on 29/04/2020).

[65] "Rs(3) — libfec-dev — debian unstable — debian manpages." [Online; accessed 28. Apr. 2024]. (Apr. 2024), [Online]. Available: https://manpages.debian.org/unstable/libfec-dev/rs.3.en.html.

[66] ka9q. "Libfec." [Online; accessed 28. Apr. 2024]. (Apr. 2024), [Online]. Available: https://github.com/ka9q/libfec.

[67] "Rs(3)." [Online; accessed 6. May 2024]. (May 2024), [Online]. Available: https://man.freebsd.org/cgi/man.cgi?query=rs&sektion=3&apropos=0&manpath=FreeBSD+14.0-RELEASE+and+Ports.

[68] C. E. Shannon, Sep. 1945. [Online]. Available: https://www.iacr.org/museum/shannon/shannon45.pdf.

[69] "Time(1) — time — debian bullseye — debian manpages." [Online; accessed 30. Apr. 2024]. (Apr. 2023), [Online]. Available: https://manpages.debian.org/bullseye/time/time.1.en.html.

[70] "Amsat ea - satélites oscar." [Online; accessed 7. May 2024]. (May 2024), [Online]. Available: https://www.amsat-ea.org/los-sat%C3%A9lites-oscar.

[71] G. Soriano, "La compra rumana de cazas f-35 deja a españa como único de sus cinco potenciales clientes que aún no los ha adquirido," *Infodefensa - Noticias de defensa, industria, seguridad, armamento, ejércitos y tecnología de la defensa*, Apr. 2023. [Online]. Available: https://www.infodefensa.com/texto-diario/mostrar/4248613/compra-rumana-cazas-f-35-deja-espana-como-unico-cinco-potenciales-clientes-no-anunciado.

[72] R. Morgan, "Chinese spy satellites tracked top gun 2 movie jet, producer says," *American Military News*, May 2022. [Online]. Available: https://americanmilitarynews.com/2022/05/chinese-spy-satellites-tracked-top-gun-2-movie-jet-producer-says.

[73] Contributors to Wikimedia projects. "Geoeye-1 - wikipedia." [Online; accessed 7. May 2024]. (Nov. 2023), [Online]. Available: https://en.wikipedia.org/w/index.php?title=GeoEye-1&oldid=1186121067.