TIINCO

Mini Project

Study Decoding Performances of Convolutional Codes by MATLAB Simulation

March 24th 2017

Group 20

Michał Kapiczyński - 201700096

Benjamin Schlie - 201370902

Lasse Have Nielsen - 201370981

TIINCO, Q3 2017

Department of Engineering, Aarhus University

Version 1

Contents

1 Introduction

The purpose of this mini project is to implement an end-to-end simulation from convolutional codes encoding, transmission through a BSC channel and decode it with a Viterbi hard decoding. The simulation will be used in order to analyse the impact of constraint length and code rate on the decoding performance.

The descriptions and comparisons of the codes will be performed, as a matter of input information for further analysis.

2 Background

In this chapter, the three main methods used in the simulation process is described.

2.1 Convolutional Codes

Convolutional codes is a encoding method for error-control coding. Where a traditional linear block code has an output in block form, this type of code outputs a sequence, which is dependent upon present and previous message input elements.

A convolutional code is typically defined by: $C_{Conv}(n, k, K)$, where n is the amount of generated symbols (output), k is the amount of input symbols and K is the memory order. $K + 1$ is called the constraint length.

The constraint length is measured in time units, and is the maximum number of time units that a given bit of the input sequence can influence the output.

The code rate is the quotient between the number of input symbols, k, and the number of output symbols, n, and is given by: $R_c = \frac{k}{n}$. The code rate depends upon how many output symbols, which is define by the amount of generator polynomials [2].

Convolutional codes can be represented in the D-domain (delay), but for simplification, it will not be covered in this report.

2.1.1 Connection representation

In this mini project all codes are represented in vector form. This means that each generator polynomial can be represented like a vector, where a '1' is a connection, and a '0' is not a connection.

Each output branch has a vector defining how the generator polynomials correspond [2].

2.1.2 State diagrams

All convolutional codes can be represented by a state diagram. The state diagram represents the different states the code is in, and how the state changes. It provides a good overview of which states that can be changed to. In the mini project there were made state diagrams for the codes. The diagram for C_{Conv3} is shown in chapter 3 on page 4. As Matlab does all the calculations the diagrams are only used for understanding.

2.1.3 Trellis diagrams

Along with state diagrams the convolutional codes can be shown by a trellis diagram. The trellis diagram can be used to describe the evolution of the code, and used for viterbi decoding, which is described in section 2.3.

As there are 2^K possible states, this diagram is only shown for C_{Conv3} in chapter 3 on the following page.As Matlab does all the calculations the diagrams are only used for understanding.

2.2 The Binary Symmetric Channel

The binary symmetric channel (BSC) is a medium through which it is possible to transmit one symbol per time unit. This channel is unreliable, and is characterised by the error probability p: $(0 \leq p \leq 0.5)$ meaning that there is a probability p for the channel to "flip" the incoming bit. The symmetry of this channel comes from the fact that the error probability p is the same for both of the symbols involved. All probabilities should eventually sum up to 1.

A channel having an error probability of $p = 0.2$ will eventually have the following outcomes:

Input	**Probability of output = 0**	**Probability of output = 1**
0	0.8	0.2
1	0.2	0.8

Table 1
Probabilities of a BSC with error probability = 0.2

2.3 Viterbi Decoding

Viterbi decoding is a method of decoding a bitstream that has been encoded using convolutional code or trellis code with use of Viterbi algorithm.

A Viterbi algorithm finds the most likely valid convolutional code from a noisy received signal. A Viterbi algorithm looks at the current received state of the signal and the series of previous states to decide what the most likely true value of the current state is.

When using Viterbi decoding, one method is the 'hard' decoding, which should also be used for this mini project. The 'hard' decoding uses the trellis diagrams, and 'finds' the shortest path through the trellis diagram as shown in fig. 1, which means the minimum hamming distance [1].

The minimum hamming free distance here is $4 + 1 + 1 + 1 = 7$, and represents the code C_{Conv3}, described in chapter 3 on the following page.

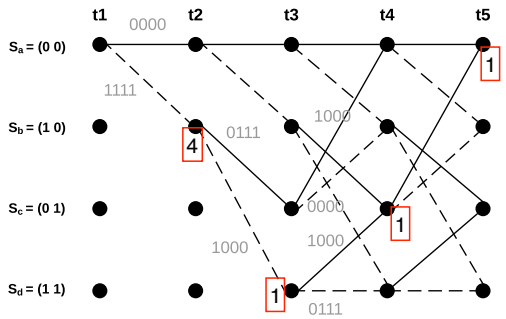

Figure 1: Viterbi hard decoding

3 Codes

The following chapter will contain a description of the different convolutional codes to be examined in this mini-project. The three convolutional codes will be shown in form of a circuit diagram. One of the codes will be described with a state- and trelli diagram as well.

In section 2.3 is a calculation of the free distance from the Cconv3 trelli diagram, and the result was 7. As stated in the assignment description, the free distance is the same for all codes described in this chapter.

3.1 Cconv1

Cconv1 has memory order of 6 and thereby a constraint length of 7. A circuit diagram can be seen on Figure 2. The convolutional code is described with the following generator sequences:

$g^{(1)} = (1111001)$, $g^{(2)} = (1011011)$

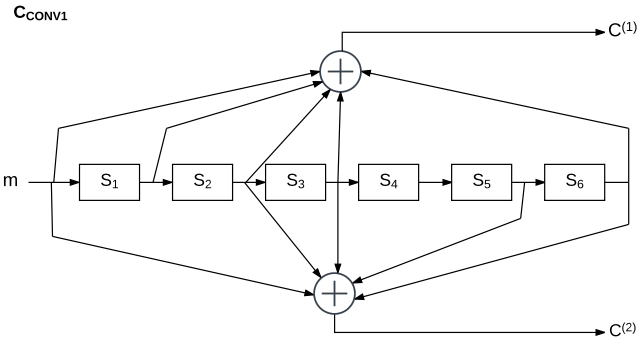

Figure 2: Circuit diagram of Cconv1(2,1,6)

3.2 Cconv2

Cconv2 has memory order of 3 and thereby a constraint length of 4. A circuit diagram can be seen on Figure 3. The convolutional code is described with the following generator sequences:

$g^{(1)} = (1011)$, $g^{(2)} = (1101)$, $g^{(3)} = (1111)$

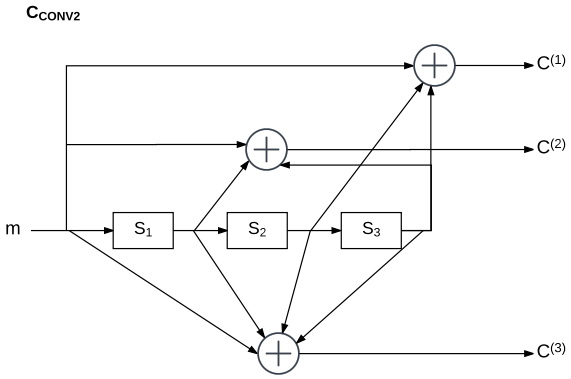

Figure 3: Circuit diagram of Cconv2(3,1,3)

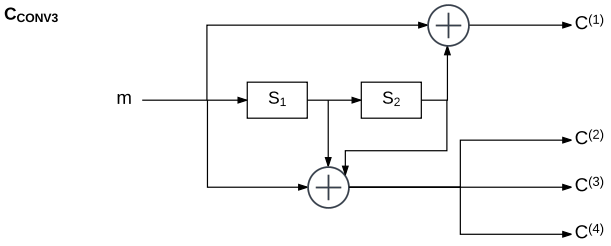

Figure 4: Circuit diagram of Cconv3(4,1,2)

3.3 Cconv3

Cconv3 has memory order of 2 and thereby a constraint length of 3. A circuit diagram can be seen on Figure 4. The convolutional code is described with the following generator sequences:

$g^{(1)} = (101)$, $g^{(2)} = (111)$, $g^{(3)} = (111)$, $g^{(4)} = (111)$

Figure 5 shows the state diagram of Cconv3. This describes the output vector, when receiving a single input bit, when being in the various states. Figure 6 shows the trellis diagram of Cconv3, which is created from the state diagram. This is used in the Viterbi decoding, in order to find the path with minimum hamming distance. When receiving a new codeword from the BSC, it compares the codeword with the possible paths in the trelli diagram in order to find the weight (difference in received bits), hence finding the minimum hamming distance.

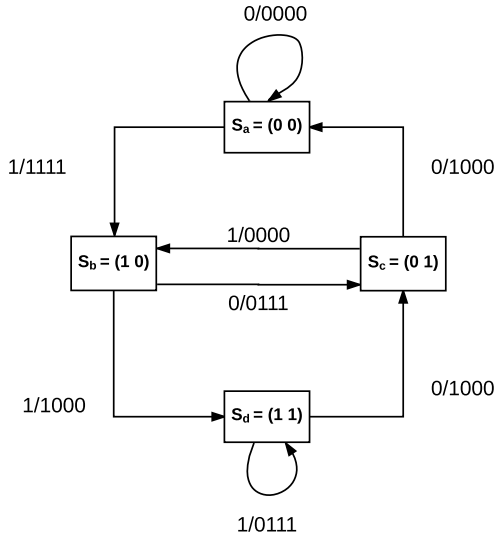

Figure 5: State diagram of C_{conv3}

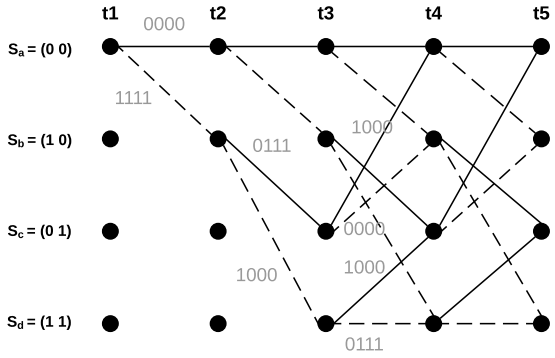

Figure 6: Trelli diagram of C_{conv3}

4 Simulation

The case is a evaluation of a three step simulation. The simulation follows fig. 7.

Primarily the vector of uniformly distributed pseudorandom binary data of size 10^6 is generated. The vector is then encoded using the convolutional encoder, and the result binary input signal data is sent through a binary symmetric channel with error probability p. The transferred data is then decoded using hard decision Viterbi decoder.The final output is then compared to the original input, to calculate the Bit Error Rate (BER).

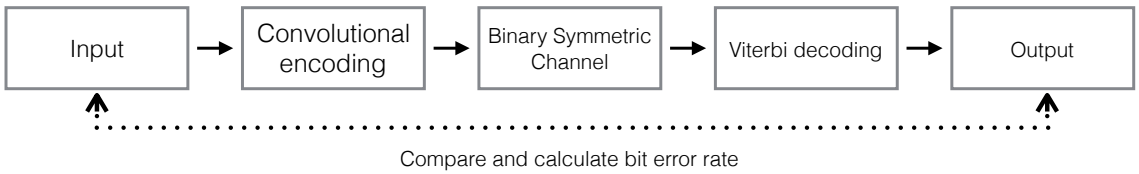

Figure 7: Simulation setup

All decoding performances will be evaluated by the BER, time and Bit-Error pr. second with the following variables taken into account:

1. Code-Rate
2. Code-Length
3. Burst-Error
4. Time
5. Corrected bits pr. second

4.1 Scenarios

The given codes C_{conv1}, C_{conv2}, C_{conv3} differ in the constraint length, and their code rates.

To assess the impact of the constraint length we need to evaluate the simulation results with the same code rate for every code, and to assess the impact of code rate on the decoding performance we need codes with the same constraint length but different code rates, thereby some generator polynomials were made up for C_{conv1} and C_{conv2}.

The simulations has been carried out for codes described in table 2:

	C_{conv1}	C_{conv2}	C_{conv3}
Constraint Length	7	4	3
Code rate $\frac{1}{2}$	$g^{(1)} = (1111001)$, $g^{(2)} = (1011011)$	$g^{(1)} = (1011)$, $g^{(2)} = (1101)$	$g^{(1)} = (101)$, $g^{(2)} = (111)$
Code rate $\frac{1}{3}$	$g^{(1)} = (1111001)$, $g^{(2)} = (1011011)$, $g^{(3)} = (0111011)$	$g^{(1)} = (1011)$, $g^{(2)} = (1101)$, $g^{(3)} = (1111)$	$g^{(1)} = (101)$, $g^{(2)} = (111)$, $g^{(3)} = (111)$
Code rate $\frac{1}{4}$	$g^{(1)} = (1111001)$, $g^{(2)} = (1011011)$, $g^{(3)} = (0111011)$, $g^{(4)} = (0011111)$	$g^{(1)} = (1011)$, $g^{(2)} = (1111)$, $g^{(3)} = (1101)$, $g^{(4)} = (1110)$	$g^{(1)} = (101)$, $g^{(2)} = (111)$, $g^{(3)} = (111)$, $g^{(4)} = (111)$

Table 2
Derived codes for simulation

4.2 Results

The results of the simulations are provided in the form of diagrams. There are three types of diagrams, which can be met through this section:

- **Bit Error Rate Plot (BER)** showing the bit error probability from the BSC on the x-axis, and the bit error rate on y-axis.
- **Time Plot** depicting BER on the x-axis and the time of encoding, transmitting and decoding one message sequence on the y-axis.
- **Efficiency Plot** describing the connection between time and errors corrected. It has BER on the x-axis and bits corrected pr second x 10^6 on the y-axis.

All simulations were carried out for different error probabilities from the range of 0.01 to 0.15 with interval equal to 0.01. The results allow to distinguish the differences between different characteristics of the given codes.

4.2.1 Original codes

Primarily, the simulation for the original codes given in the project description was conducted. Their performance can be seen in the figures below.

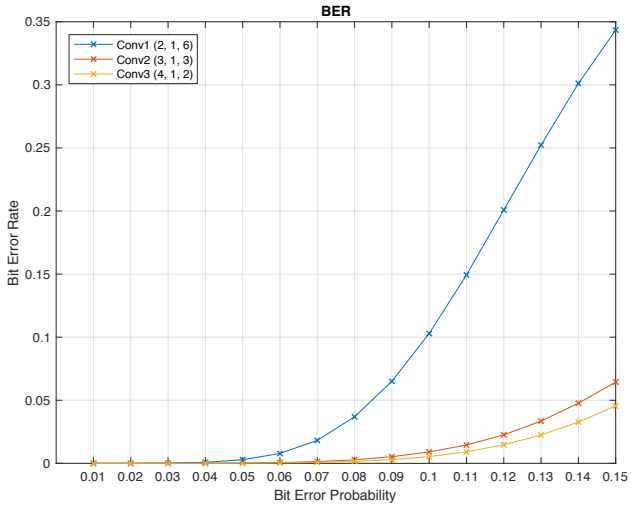

Figure 8: Original Codes - BER

Figure 8 indicates that the code $C_{conv1}(2, 1, 6)$ has the biggest BER of all the codes, meaning that the code with the smallest code rate, and biggest constraint length operates the worst.

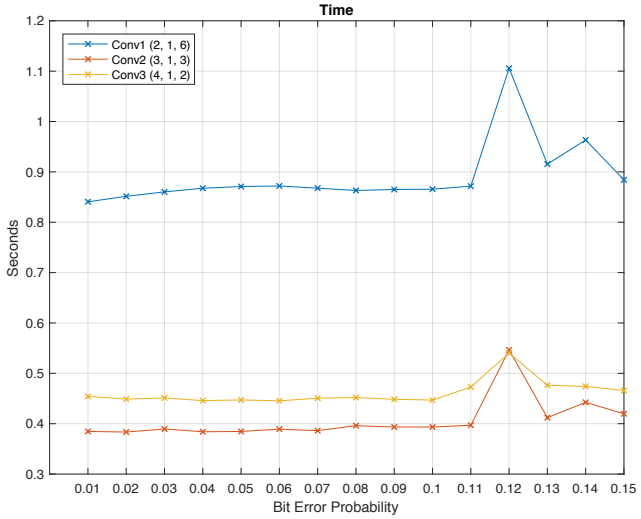

Figure 9: Original Codes - Time

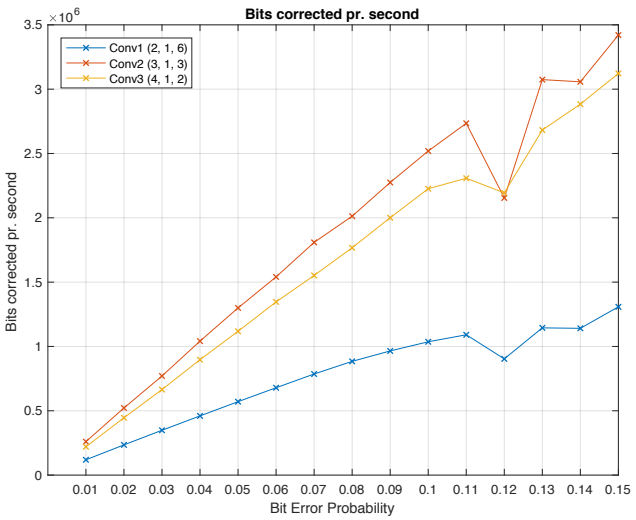

Figure 10: Original Codes - Efficiency

Figure 9 shows that processing through $C_{conv1}(2, 1, 6)$ is much more slower compared to $C_{conv2}(3, 1, 3)$, and $C_{conv3}(4,1,2)$. The time processing the message sequence through C_{conv1} is approx. 188% of the time processing C_{conv3}, whereas the time of processing with C_{conv2} is 70% of C_{conv3} and thereby the fastest. It is vivid that the bigger the constraint length (the memory of the code) the slower the transmission.

Figure 10 shows that the amount of bits corrected pr. second is approx twice as much for C_{conv2} and C_{conv3} compared to C_{conv1}, whearas C_{conv2} and C_{conv3} are closer to each other.

4.2.2 Original codes - Burst Errors On

For this simulation all codes used were the ones given from the hand-in, and burst error simulation were enabled. Their performance can be seen in the figures below. The length of the burst error was set to 8.

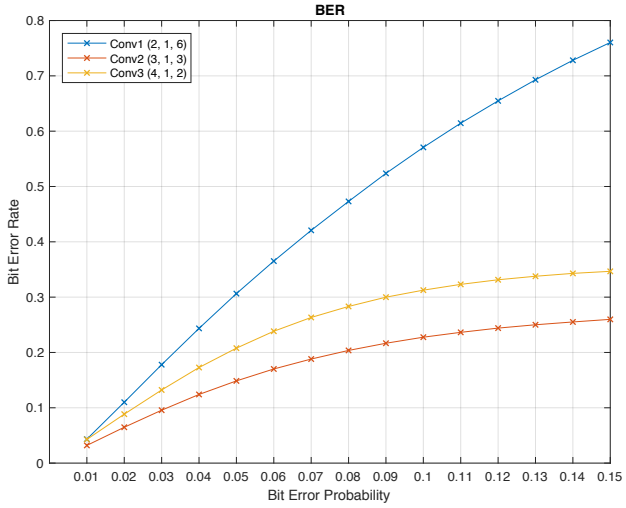

Figure 11: Original Codes (Burst Errors On) - BER

Figure 11 shows that the bit error rate increases rapidly for C_{conv1}, whereas C_{conv2} and C_{conv3} is "flattend" more quickly. C_{conv1} is having a BER approx. 0.75 when the error probability is 0.15. With the same error probability, C_{conv2} has a BER of approx. 0.25, and C_{conv3} approx. 0.35.

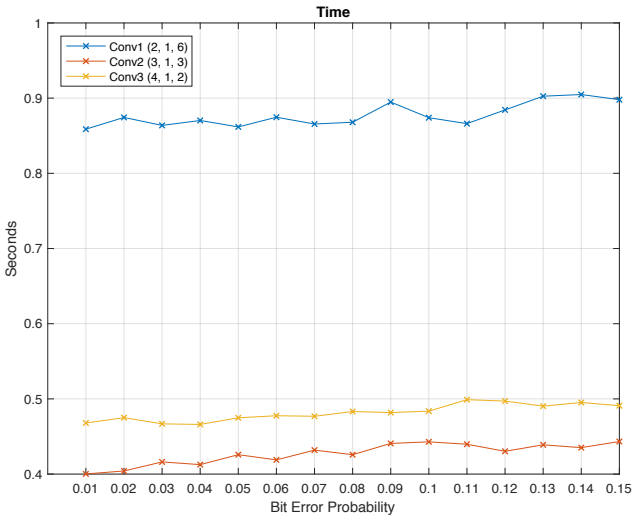

Figure 12: Original Codes (Burst Errors On) - Time

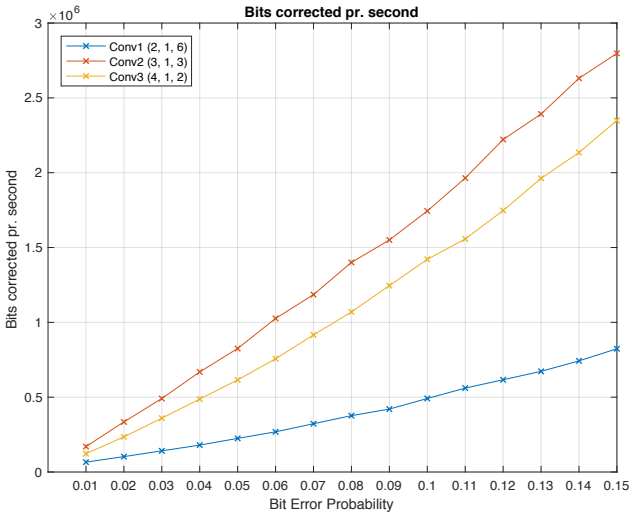

Figure 13: Original Codes (Burst Errors On) - Efficiency

Figure 12 indicates that the time processing the messages sequence for the three convolutional codes, are very similar as without burst errors. Same similarity can be seen on Figure 13 when comparing the 3 codes, although the bit errors corrected pr. second are lower when burst errors are introduced.

4.2.3 Code Rate impact on original codes

4.2.3.1 Modified Codes - Code Rate $R_C = \frac{1}{2}$

For this simulation the original codes has been modified so that all of them have the same code rate equal to $R_C = \frac{1}{2}$. Their performance can be seen in the figures below.

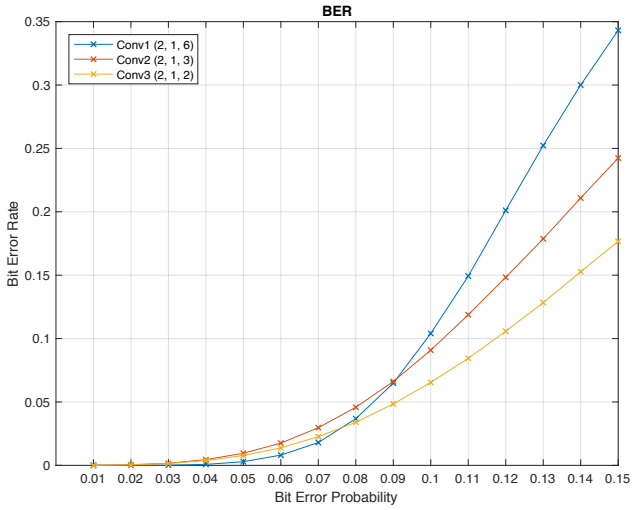

Figure 14: Modified Codes (Code rate $R_C = \frac{1}{2}$) - BER

The figure 14 shows that for the same code rate equal to 1/2 and bit error probability smaller than 0.08 the C_{conv1} with the biggest constraint length operates slightly better than the other codes. However, when the bit error probability increases above 0.8 the BER of C_{conv1} rises the fastest, and its characteristic is worst than the ones of the other codes. All in all, with the increase of the bit error rate the negative impact of the constraint length can be registered.

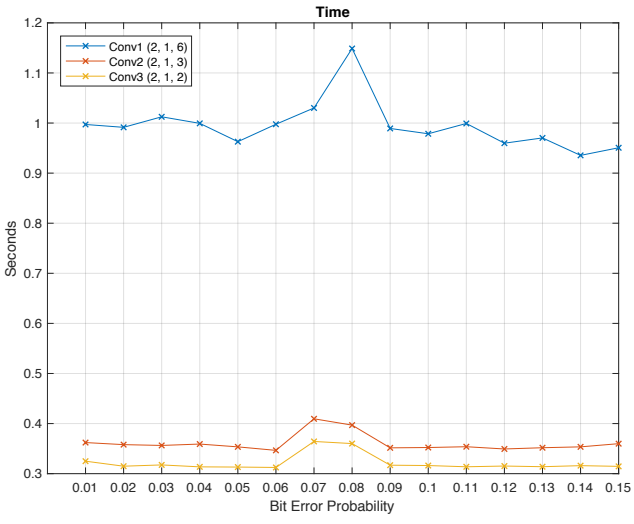

Figure 15: Modified Codes (Code rate $R_C = \frac{1}{2}$) - Time

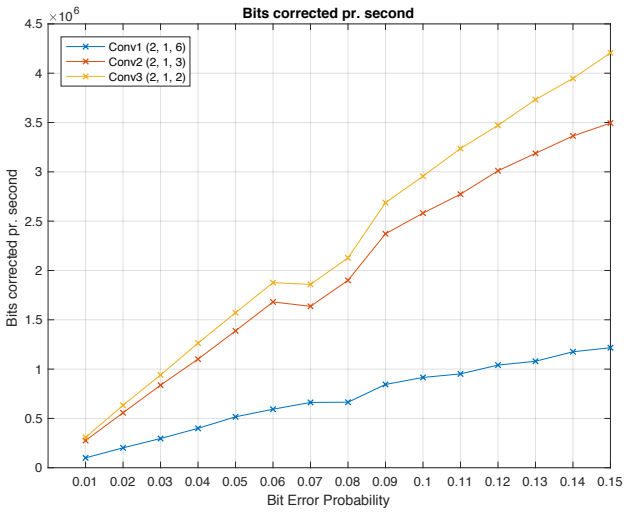

Figure 16: Modified Codes ($R_C = \frac{1}{2}$) - Efficiency

The result from Figure 15 is that the bigger the constraint length (the code delay) the longer the transmission takes when having the same code rate. This also have an impact on the efficiency as seen on Figure 16.

4.2.3.2 Modified Codes - Code Rate $R_C = \frac{1}{3}$

For this simulation, the original codes has been modified so that all of them have the same code rate equal to $R_C = \frac{1}{3}$. Their performance can be seen in the figures below.

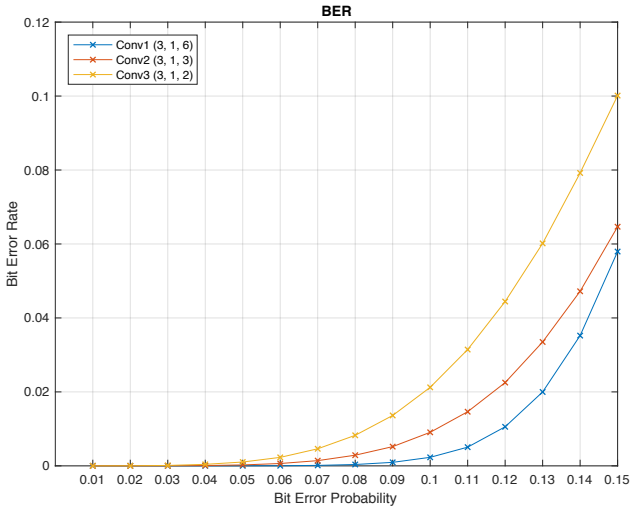

Figure 17: Modified Codes (Code rate $R_C = \frac{1}{3}$) - BER

In the case described by Figure 17, having all the codes with the same code rate equal to 1/3 it is possible to register that the code C_{conv1} with the biggest constraint length has the smallest BER. It is still vivid that with the increase of the bit error probability the angle of the characteristics with bigger constraint length increases more rapidly, but up until 0.15 the constraint length impact on the BER stays positive.

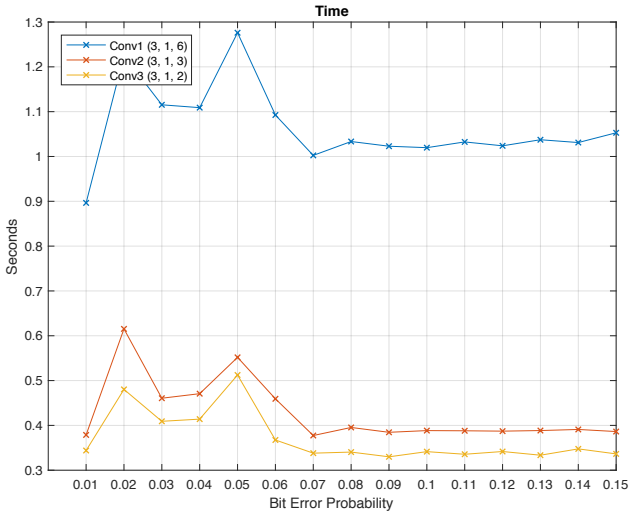

Figure 18: Modified Codes ($R_C = \frac{1}{3}$) - Time

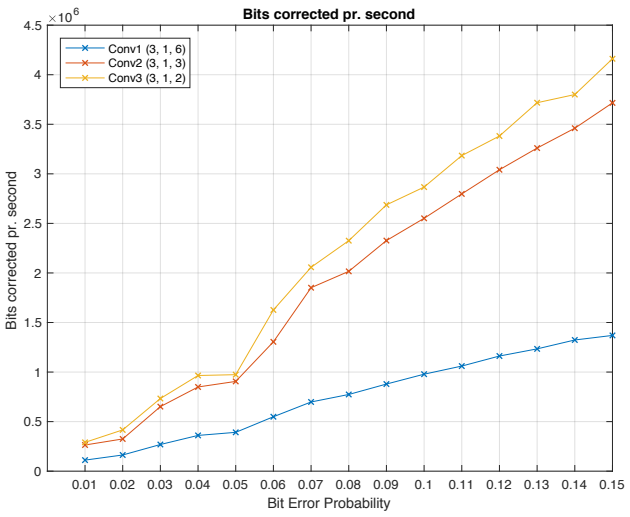

Figure 19: Modified Codes ($R_C = \frac{1}{3}$) - Efficiency

Figure 18 and Figure 19 shows the time and efficiency. As seen before, the time of performing C_{conv1} is slower than the other two, which has an inpact on the efficiency. This means that even though C_{conv1} performs the best, it corrects less bits pr. second than C_{conv2} and C_{conv3}.

4.2.4 Code rate impact on single code

In this case the simulation was carried out for the same code with modified code rate. The overall results has been the same for all of the codes, wherefore only the results for the C_{conv1} will be described.

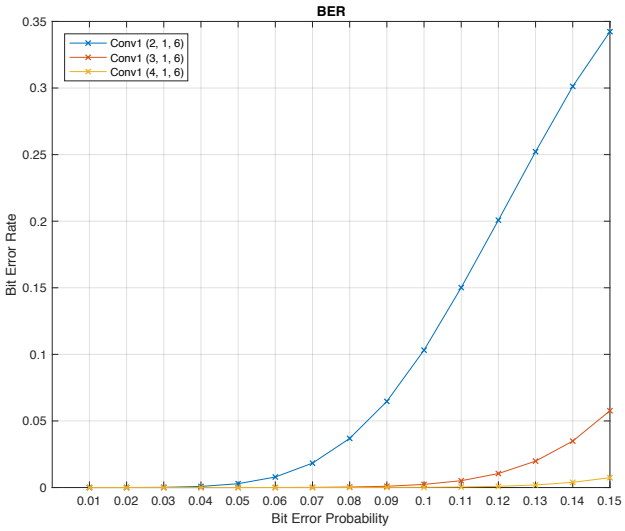

Figure 20: Cconv1 with different code rates - BER

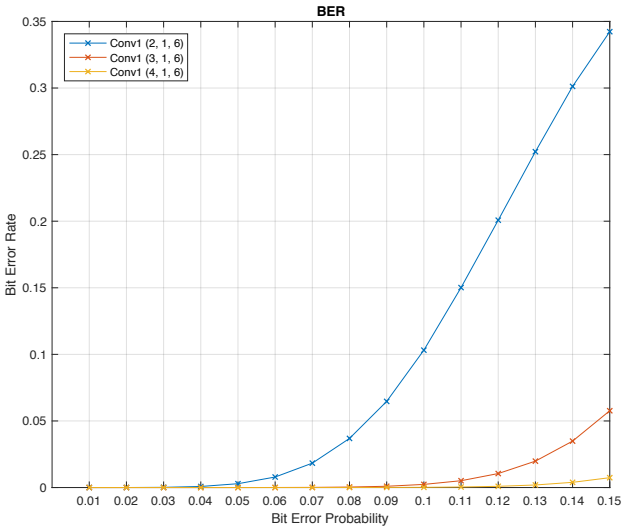

Figure 21: Cconv1 with different code rates - Time

As seen on Figure 20, decreasing the code rate results in a decreasing the Bit Error Rate, however the time increases as the code rate decreases, as seen on Figure 21. The efficiency can be seen in Figure 22, and shows that all codes almost perform the same measured by bits corrected pr. second. Thereby will a doubled time result in a doubled amount of errors corrected pr. second.

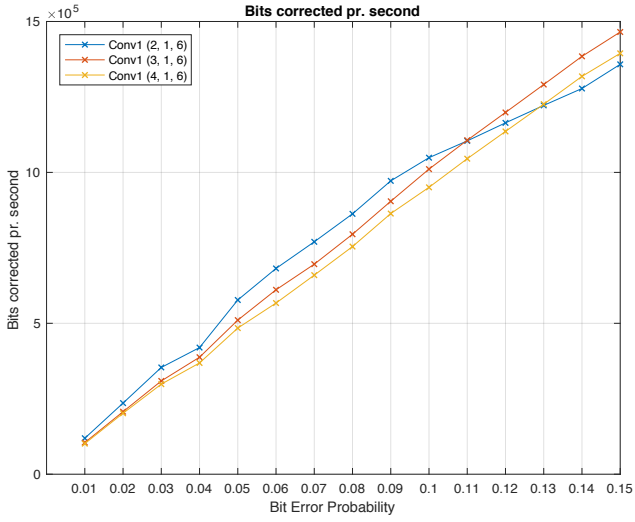

Figure 22: Cconv1 with different code rates - (Efficiency)

4.3 Comparison

After the simulation was made, the results were compared on the 5 parameters from the simulation.

The general observation is that a decrease in code rate leads to reduction of Bit Error Rate. What is more, for codes with the same code rates, but different constraint lengths, it is possible to notice the positive impact of

the constraint length on the code performance for small bit error probabilities (p $<$ 0.8 with $R_C = \frac{1}{2}$), seen in Figure 14.

When the bit error probability starts to increase (p $>$ 0.8) the code with high constraint length (in this case K = 6) eventually starts to perform worse than the codes with K = 3, K = 4. This indicates that for a channel with a higher error probability than 0.8 a larger constraint length will have a negative impact on the performance.

We assume that the reason why, is that the larger constraint length will cause more errors "stored" in the memory, and there by provide a larger probability of a wrong decoding, because it will take longer time for the error to get through the code with 6 memory states, instead of 3 or 4. It is an important observation, which should be taken into account when choosing an appropriate code for a given channel.

In all scenarios, the code memory order related to constraint rate vividly impacts the overall transmission time. As a result, it shall be noted that the increase of constraint length leads to the increase of transmission time (See Figures 9, 15 and 18.

Comparison of the results for original codes with burst errors shows that the BER values are significantly higher than values for simulation without burst errors.The performance gap between C_{conv1} and the rest of the codes was narrowed down. Another observation is, that the occurrence of burst errors transforms the Bit Error Rate characteristics of the codes from exponential to logarithmic form.

It is hard to evaluate the optimum trade-off between the code rate and constraint length because their ratio is highly dependent on the value of channel bit error probability. The optimal implementation is that the code rate and constraint length should be chosen on a basis of the bit error probability.

In general the decrease of code rate will lead to better performance for a larger constraint length (smaller BER), but with the negative impact on transmission time. This can also be seen in 20, where C_{conv1} was chosen, an had significant better results with a code rate of $\frac{1}{4}$, compared to $\frac{1}{2}$. The time however showed that a decrease in code rate increased the time by almost 50% (Figure 21).

5 Conclusion

This mini-project has carried out an end-to-end simulation of convolutional codes encoding, transmission through a BSC channel and decoding using Viterby hard algorithm. The purpose was to analyse the impact of code rate and constraint lengths on codes performance. Three convolutional codes were given, with different constraint- and code lengths. The authors added derived codes in order to compare the codes with same constraint lengths and different code rates and vise verse.

The analysis was focusing on 5 parameters: code rate, constraint length, burst error (on - burst length 8 / off), time (from start encoding to finished decoding) and corrected bits pr. second. It was discovered that C_{conv1} was generally slower, almost twice the time compared to the other two given codes. Also the BER of C_{conv1} was remarkably higher than the other two. When adding burst errors the percent wise difference between C_{conv1} and the other two was smaller but C_{conv1} still performed quite bad as error probability of 0.15, was resulting a BER of 0.75.

C_{conv3} performed slightly better than C_{conv2} when comparing BER with burst error disabled, but on the other hand C_{conv2} was better when burst error was enabled. In both cases C_{conv2} was faster, hence if burst error is possible C_{conv2} would be the best choice.

When comparing codes with equal constraint length and different code rates, it was verified that increased code rate had an significant impact on the BER. Opposite, when having different constraint length with same code rate, it was discovered that with a code rate of 1/2 a longer constraint length performed better at low bit error bit error probability, and when decreasing the the code rate, the time before the long constraint ($K > 5$) becomes worse than the other codes. This is at error probability $p > 0.15$.

In general, the optimal throughput would be possible to achieve with a decoder that knows the error probability of the channel, and thereby can be adjustable on runtime, so the constraint length and code rate can be modified depending on the current error probability of the channel.

Bibliography

[1] MIT 6.02. Viterbi Decoding of Convolutional Codes. `web.mit.edu/6.02/www/f2010/handouts/lectures/L9.pdf`, 2010. [Online; accessed 20-March-2017].

[2] Moreira et. al. Jorge Castineira. *Essentials of Error-Control Coding*. John Wiley & Sons, Ltd, 2006. ISBN 0-470-02920-X.