COMS4105/7410 Communication Systems

Practical 2: Channel Coding, Bit Error Rate and Frequency Synchronisation

1 Introduction

This practical is the second of three practicals which will run in this course. Each practical will be run over a fortnight. Work for the practical can be either done at home or during the practical sessions. However you will need to attend at least one of the two sessions each fortnight to be marked off for both your preparation and lab work.

Practical work for this course will take place in the **78-212** laboratory. You should note that this location is classified as an engineering laboratory and workplace health and safety regulations apply. You should read the "Occupational Health and Safety in the Laboratory" guidelines and fill out the declaration. Failure to sign the declaration and/or abide by the guidelines will result in your exclusion from the laboratories. In particular, if you fail to wear covered footwear, you will be excluded from your prac session.

Bring your RTL-SDR device (including antenna) to each practical session. For each prac, you need to have attempted the pre-lab questions for each practical *before* attending the practical session. All code that is marked must be checked into your own private Git repository on GitHub. Also, see the *private* GitHub wiki for additional hints.

2 Learning Objectives

In this lab you will be evaluating different channel coding methods within a communication system which operates in a additive white Gaussian noise environment. In addition, due to the importance of frequency synchronisation, a simple method used to perform coarse frequency synchronisation will be introduced.

In the first part of this practical you will be simulating three different channel coding methods, Hamming coding, Convolutional coding and Reed-Solomon coding. As you will have direct control over the effects of the channel, you will be be able to adjust the signal-to-noise ratio (SNR) and find the bit error rate of the system as a whole.

The next two parts of the practical will experimentally test these coding techniques. For over-the-air (OTA) signals before applying any demodulation, there is a need to perform both time and frequency synchronisation. In practical 1 we studied time synchronisation in detail. This practical introduces a simple method to perform frequency synchronisation. The transmitted signal from the demonstration computer in the lab will be used to test the channel coding methods under varying signal to noise ratio.

The learning objectives are as follows:

1. Understand how a software defined radio (SDR) based communication system operates including its block diagram and process,
2. Be able to apply channel coding (and decoding) to various parts of data, implementing error detection and/or correction,
3. Be able to perform basic frequency synchronisation,
4. Read relevant sections of a communication standard (DAB – ETSI EN 300 401)

5. Using the knowledge obtained from the previous parts of this practical become more familiar with the DAB signal, by understanding where the different coding schemes apply in practice,
6. Experience designing a structured communication system by using modular system blocks which are connected with-in a *top-level design*.

3 Preparation

To help you test your code in the practical, the preparation questions will ask you to perform some channel coding and decoding by hand. Also you will need to research the available channel coding methods present in DAB.

Question 1 Design a Hamming (7,4) and Hamming (15,11) block code via a generator matrix **G**. Encode a message with a single '1' in the first bit position and the remaining bits set to '0's for both of these codes.

Find the corresponding parity check matrices **H** for both codes. After applying an error in the fourth bit of each of the code words, calculate the syndrome for this and explain which received bit the algorithm would correct.

What is the code rate for both of these codes? How many errors can be detected, and corrected?

Question 2 The cyclic-redundancy-check (CRC), is a code used to apply a checksum to a block of data. DAB uses a CRC-16 code to apply checksums on many of the packet structures. What is the polynomial used in DAB's CRC-16 calculations? Find the checksum for the data `1001 0000 0000 1001`. In DAB (ETSI EN 300 401) in the FIC – on how many bytes of data is the CRC-16 performed on?

Question 3 In the DAB standard specification (ETSI EN 300 401), find the structure of the Convolutional encoder used. Encode the bits `101`, and pad the message with '0's until the three original bits are no longer in the state variables.

As this convolutional code is quite large in terms of its state space, in this practical we will also consider a smaller code – so that it is feasible to check the code by hand. The convolutional code has the following output polynomials

$$\begin{aligned} x_0 &= 1 + p \\ x_1 &= 1 \\ x_2 &= 1 + p + p^2 \end{aligned}$$

Draw the trellis diagram for this simpler code, and decode the received bits '`101 001 110 011 001`'.

Question 4 In the DAB standard specification (ETSI EN 300 401), find the structure of the Reed-Solomon code, including both the code generator polynomial and field polynomial. Explain the process of encoding using this RS code.

Question 5 Consider a simple frequency synchroniser packet structure which repeats the preamble twice. If the received signal is a assumed to be a frequency shifted version of the transmitted signal, with an unknown complex channel, h, then the following expression holds.

$$y[t] = e^{j2\pi f_c t} h x[t] + n[t]$$

This means that if we compare the first preamble to the second, and ignore the noise terms, we notice the following

$$y[t + N_{\text{pre}}] = e^{j2\pi f_c N_{\text{pre}}} y[t]$$

or, each sample of the second preamble is a constant multiple of the corresponding sample in the first preamble. Hence, we devise a way to estimate the frequency offset via a 'best fit'.

This problem can be formulated by means of N_{pre} simultaneous equations, and one unknown. Being an overdetermined system of equations, it means that we should use the least squares (LS) method to solve it.

In Matlab/Octave, least square problems can be solved using the '\' operator, also known as the pseudo-inverse. This provides a "line of best fit" between two sets of variables. Whereas in Python you can use the `linalg.lstsq` function in numpy.

For example, consider the following vectors: x =(`1 3 4 9`)^T, and y =(`2 5.5 9 17`)^T, and we would like to map this to a function $y = mx$. We put the values into two column vectors, giving the equation $\mathbf{y} = \mathbf{x}m$, with $\mathbf{y} \in R^{4\times1}$, $\mathbf{x} \in R^{4\times1}$, and $m \in R^{1\times1}$. So $m = \mathbf{x}^{+}\mathbf{y}$.

After the value of best fit is found between the two copies of the preamble, find a way to convert this value to the frequency shift.

Unfortunately there are some limitations as to how large a frequency shift can be solved for. Find the maximum frequency shift which can be solved when the sample rate is f_s =2 MS/s and the training sequence has 8 symbols (16 after repetition). There are 8 samples per symbol.

4 Method

In this practical we will be using most significant bit (MSB) first ordering just like in Practical 1. The guide on Github (available at https://github.com/UQ-Communication-Systems/public/wiki/Guide-Converting-Data) shows you how to convert ASCII data to bits. Of course in the receiver you will need to reverse this process.

Next, to avoid non-linear distortion on your RTL-SDR, you should avoid using auto-gain, and instead capture signals with a 'gain' value set (via `-g`). If you are close to the transmitter use lower levels of gain.

```
rtl_sdr dump.bin -s 2e6 -f 110.9e6 -n 2e6 -g 35
```

4.1 Evaluating the Performance of Channel Coding

For channel coding there are a large number of different channel codes available to the designer. Each scheme has different characteristics in terms of code rate, error detection capability, error correction capability, and complexity.

In lectures we grouped channel coding into two main categories, block and continuous. The Hamming code, and polynomial (CRC) are block codes. Convolutional encoding is a continuous code. We also considered a multi-level block code known as the Reed-Solomon code. In this practical we will evaluate the bit error rate performance of the different channel coding methods as well as the improvement (if any) over not using channel coding at all.

The basic block diagram that you will need to implement is shown in Figure 1. You can interchange the various channel coding blocks for the different parts of the practical. As we are considering a single carrier communication system, you may assume that the baseband modulation occurs on a centre frequency $f_c = 0$ Hz. Also, implement your system to have a symbol duration of 8 samples. The dotted lines in the diagram are drawn to

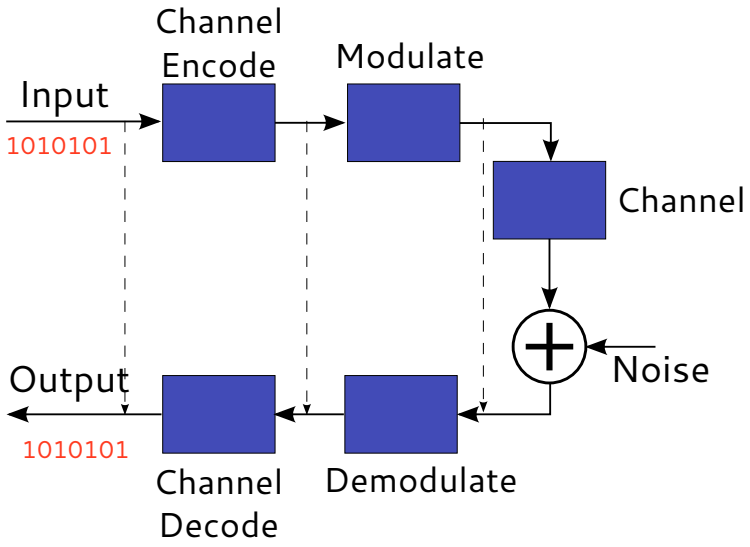


Figure 1: Recommended block diagram in Practical 2

remind you that you can test your system without the full implementation. For example, you can put the channel encoder output into the channel decoder input – this way you test your channel coding process without relying on the rest of your system. In addition, when testing under real data, you should look at the input and output of each block to see if the data looks correct – the I&Q values being demodulated should look like the constellation they are for (QPSK, BPSK, 16QAM etc). Note: in Gnuradio if you design your encoders and decoders correctly look at the `FEC` blocks and associated examples.

For each part you will need to send a significant number of bits through the communication system and then calculate the bit error rate. You will need to adjust the signal to noise ratio so that you can obtain a graph similar to the one below.

Exercise 4.1.1 In this exercise you will be using two different Hamming codes: the $(7, 4)$ and $(15, 11)$ hamming code. To do this you should write a number of functions:

1. Implement a block which takes one block of message bits and outputs one block of coded bits.
2. Implement the corresponding block which is used at the receiver. That is – a block which performs the syndrome calculation and corrects the bit error(s) (if they exist).
3. Put these two blocks into your BER testing system and generate the BER vs SNR graph for the two hamming codes. Compare the result to when no channel coding is used.

What are the improvements and penalties for using the channel code at various SNRs?

Hint: Your design should include a top-level design which is structured similarly to the figure presented at the beginning of this section. You do not need to process all the bits in one go, in fact it might make sense to send smaller batches of bits and accumulate the results. ■

Exercise 4.1.2 Another block based channel code is the cyclic redundancy check (CRC). We will be using the CRC-16 polynomial used in DAB (refer to the preparation ques-

tion). It will be used to add 16-bits of checksum to every block of 16 message bits.

This time, instead of performing error correction at the receiver, only error detection will be performed.

Using statistics of how many error frames occur and are detected for each value of SNR plot a graph of actual and un-detected errors per block as a function of SNR. ■

Exercise 4.1.3 Repeat 4.1.1 by using the simple Convolutional encoder presented in the preparation questions. ■

Exercise 4.1.4 Repeat 4.1.1 by using the RS encoder which is used in DAB.

Hint: Reed Solomon encoding is code which already implemented in Matlab / Octave (`rsenc`, `rsdec`). In Python check out GitHub for a compatible RS encoder. ■

4.2 FEC, Error detection and frequency synchronisation

In this part of the practical we testing the channel coding receivers using *over-the-air transmitted signals*.

With real received signals, we have noticed that in addition to time synchronisation, we must also obtain good frequency synchronisation. So before decoding the data we will first perform an exercise on frequency synchronisation.

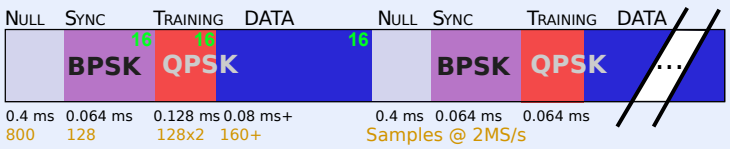
Like the previous practical, transmissions will occur on four frequencies (A-D) and these will be referred to in the corresponding exercises of the practical. The actual frequencies will be posted on the whiteboard during each practical.

You will need to use your signal detection methods from Practical 1 including both energy detection and preamble detection.

Exercise 4.2.1 To help with both frequency and time synchronisation the sequence introduced previously will be transmitted **twice** using QPSK modulation `11110011 10100000` (This is labelled as 'training' in the diagram below), and a new sync word will be used provide an even better way of detecting the frame, `10101011` – which is labelled as 'sync' in the diagram below. The same packet structure is used in most of the following examples. You can use your frequency offset estimator from the preparation question, and your packet detectors from practical 1.

Tune into **frequency A**, to see if your frequency estimator works correctly. Note, that if your frequency is out

The structure of the signal in frequency A is as follows:



The data and training are encoded using QPSK modulation. The data part of the transmission is a text message which will be decoded in the next exercise. ■

Exercise 4.2.2 The data transmitted in 4.2.1 is a text message which has been encoding using a Hamming (7,4) code.

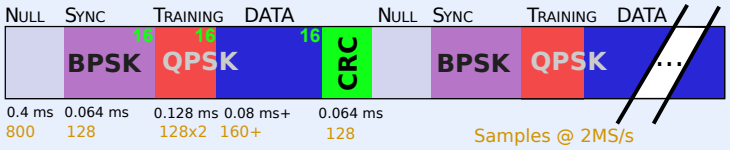
The same packet structure applies, but here we need to find the original text, rather than the coded message. The data part of the packet has been encoded using a Ham-

ming (7,4) code. Make sure your Hamming decoder can correct for the required number of errors.

Find the word being transmitted. ■

Exercise 4.2.3 If more errors occur than the channel code can correct or even detect we may not be able to receive our signal correctly.

In this exercise you will receive a signal encoded using a Hamming (15,11) code, but in addition each frame will have a CRC-16 checksum.



For this you will need to detect both the start and end of each frame. As you will need to detect where the CRC-16 finishes.

Tune into **frequency B**, and use 1 second recording (for offline processing)Find the words being transmitted in a few consecutive packets. ■

4.3 Convolutional, Reed Solomon and Concatenated Codes

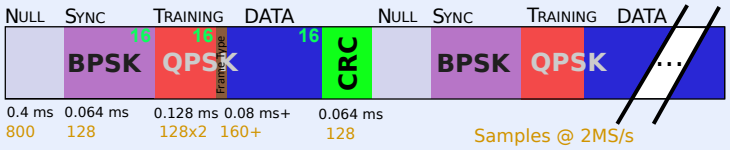
In a system such as DAB, two different channel coding methods are used. The first is convolutional coding, a continuous code, which is also known as an inner code. The second is Reed-Solomon coding, a block code, which is the outer code. The two codes are used together to produce a concatenated code which achieves superior channel coding gain.

Here we will study Convolutional and Reed-Solomon codes individually and then together as one concatenated code.

To study the error performance the signal power will be decreased for frequency C and D and hence the effect of using the channel codes will be visible – error correction should occur.

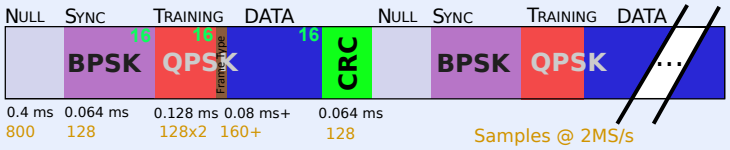
Exercise 4.3.1 Here we will have two different types of packets transmitted on the same frequency. The way you will distinguish them will be via different 'frame types'.

Tune into **frequency C**, find the messages prefixed with the same (double) training sequence as previously: `11110011 10100000`. Following this, the next 4 bits (or 2 symbols) are encoded with a frame type. For this exercise - frames with the frame type '`0 0 0 1`' are present. Next decode the data which has been encoded using the simpler convolutional code specified in the preparation questions.



Take a 1 second recording for offline processing. ■

Exercise 4.3.2 The second type of packet will be a Reed-Solomon encoded packet. The packet structure is similar to the previous example, and this time the frame type is '`0 0 1 0`'.



Tune into **frequency C**, find the messages with the stated frame type and decode the RS encoded data.

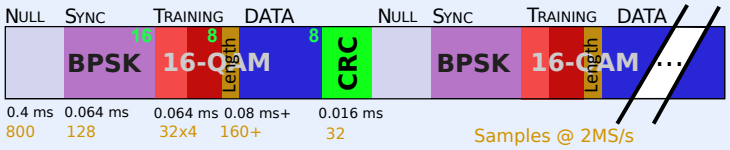
■

Exercise 4.3.3 In this exercise we will combine the different features from the various coding methods presented above. That is all data will be first encoded using Reed-Solomon coding followed by a convolutional code. This is the same encodings as used in the previous exercises, but this time together as a concatenated code.

Modulation will be done using a 16-QAM modulation in both the training and the data, with 8 samples bit symbol. The training sequence will be improved by creating two pairs of training sequences. Decoding should be done in the opposite order.

Training A will be the 16-QAM modulation of the previous sequence `11110011 10100000` (4 Symbols), and training B will be same symbols but with +1 on each symbol, giving `00000100 10110001`. Because of the reduced samples per bit, the total number of symbols of the training remains the same. However note that the symbol duration is half of the previous questions.

Finally to ease the detection of the end of the packet, the first 16 bits of each packet will contain the length of the remaining parts of the packet (coded data + CRC16).



Tune into **frequency D** to try to decode this message.

Challenge: How low can the power get at the transmitter before you are unable to receive the signal correctly.

■

References

[1] Template based on latextemplates.com/template/the-legrand-orange-book