

EE5350: Programming Assignment 2

5th November 2025

1 General Instructions

In this assignment you are expected to decode

1. Reed Muller codes using three different decoding algorithms (a) Maximum Likelihood (ML) decoder, (b) successive-cancellation decoder (SCD) and (c) majority logic decoder
2. Polar codes using two different decoders (a) ML and (b) successive cancellation decoder

You can assume the channel is BSC(p) with $p < 0.5$. Therefore the ML decoder will be the same as minimum distance decoder i.e., finding a codeword closest to the received vector.

You are expected to submit the following files within a .zip file. The zip filename needs to be of the form Group_No.firstname1.firstname2.zip

1. RMdecode.m that implements: function RMdecode(r, m, decoder, p, filename) that takes as input five arguments (1) value of r (2) value of m (3) decoder type (1 corresponds to ML decoder, 2 to SCD and 3 to majority logic decoder), (4) p : bit flipping probability and (5) filename. The last input will path of the file containing N corrupted codewords i.e., $N * 2^m$ bits.

Few reference files are provided with different n, k values where $n = 2^m$,

$$k = \sum_{i=0}^r \binom{m}{i} \text{ all corresponding to different error patterns seen for all-zero}$$

codeword. **Note that the matlab scripts that you submit will be tested with arbitrary corrupted codewords.** The expected output

is a file containing recovered codewords (not the message vectors) i.e., there need to be a file of size $N * 2^m$. If the input file name is

“abcdefg_m.8_r.1_N.100.in.txt” then the output file names need to be either “abcdefg_m.8_r.1_N.100.out_mld.txt”, “abcdefg_m.8_r.1_N.100.out_ml.txt”

or

“abcdefg_m.8_r.1_N.100.out_scd.txt” depending on the decoder type that is set in the input.

2. PolarDecode(m, Afile, decoder, p, filename) that takes as input four arguments (1) value of m (2) Afile is path to .mat file with A set defined in it (3) decoder type (1 corresponds to ML decoder, 2 to SCD) (4) p : bit flipping probability and (5) filename. The last input will path of the file containing N corrupted codewords i.e., $N * 2^m$ bits.

Few reference files are provided with different $(n = 2^m, k, A)$ values all corresponding to different error patterns seen for all-zero codeword. **Note that the matlab scripts that you submit will be tested with arbitrary corrupted codewords.** The expected output is a file containing recovered codewords (not the message vectors) i.e., there need to be a file of size $N * 2^m$. If the input file name is “abcdefg_m.8_k.9_N.100_in.txt” then the output file names need to be either “abcdefg_m.8_k.9_N.100_out_ml.txt” or “abcdefg_m.8_k.9_N.100_out_scd.txt” depending on the decoder type that is set in the input.

3. MATLAB .fig files that plots the probability of codeword errors for RM, Polar codes for the following cases:
 - (a) $(r = 2, m = 4)$ RM (decoder types 1, 2, 3), $(n = 16, k = 11, A)$ Polar code (decoder types 1, 2)
 - (b) $(r = 1, m = 8)$ RM (decoder types 1, 2, 3), $(n = 256, k = 9, A)$ Polar code (decoder types 1, 2)
 - (c) $(r = 2, m = 8)$ RM (decoder types 2, 3), $(n = 256, k = 37, A)$ Polar code (decoder types 2),
 - (d) $(r = 3, m = 8)$ RM (decoder types 2, 3), $(n = 256, k = 93, A)$ Polar code (decoder type 2)

Required input corrupted codeword files are provided in the folder and mat files that have A sets defined for each of the above cases are also made available.

2 Encoding for RM and Polar codes

Encoder implementation is made available in MATLAB files shared. We first make the following observation. Rows of \hat{G}_m defined below correspond to monomial evaluations.

$$\hat{G}_m = P_m \begin{bmatrix} \hat{G}_{m-1} & 0 \\ \hat{G}_{m-1} & \hat{G}_{m-1} \end{bmatrix}$$

where P_m is a permutation matrix such that $\underline{u}P_m = (\underline{u}_o \ \underline{u}_e)$. Lets look at example matrices:

$$\hat{G}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

The rows of this matrix can be thought of as evaluations of monomials x_1 and 1 respectively. The columns can be thought of as indexed by $x_1 = 1$ and $x_1 = 0$. Similarly:

$$\hat{G}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

The columns can be thought of as indexed by $(x_2, x_1) = (1, 1), (1, 0), (0, 1), (0, 0)$ respectively. The rows of above matrix can be thought of as evaluations of monomials $x_2x_1, x_1, x_2, 1$. You can check that for \hat{G}_3 the columns are indexed by binary representation of numbers 7, 6, 5, \dots , 0 and rows by monomials $x_3x_2x_1, x_2x_1, x_3x_1, x_1, x_3x_2, x_2, x_3, 1$.

The encoding proceeds using the encoding algorithm we have seen for the polar code. Let us define $(n = 2^m, k, A)$ polar code \mathcal{C} such that:

$$A = \{i \in [1 : n] \mid w(G(i, :)) \geq 2^r\}.$$

Then \mathcal{C} is exactly the same as the RM(r, m) code $\mathcal{C}_{r,m}$. Encoding is described this way to make the successive cancellation decoding implementation simple.

For the polar code, the parameter A will be inputted to the decode function and for the test cases made available, A sets are described in corresponding text files.

3 Decoding

3.1 Majority Logic decoding of RM codes

Here the highest order monomial coefficients are decoded first and then their effect is cancelled from the received vector followed by which next order monomial coefficients are identified and so on so forth.

A monomial can be written of the form $x_m^{a_1} x_{m-1}^{a_2} \dots x_1^{a_m}$ for some $(a_1, a_2, \dots, a_m) \in \{0, 1\}^m$. The evaluation of this monomial is represented by row $a = 1 + \sum_{i=0}^{m-1} (1 - a_{i+1})2^i$. From the previous examples discussed in encoding section it can be verified that for $m = 3$ the monomial $x_3x_2x_1$ appears in first row whereas the monomial x_2 appears in row 6. Therefore if the coefficient of a monomial is decoded as one then the row corresponding to it is subtracted from the received vector before the lower order monomial coefficients are decoded.

3.2 Maximum Likelihood decoding

For ML decoding the expectation is that you will implement the minimum distance decoder where you'll enumerate all the codewords in a table and for each of the received vectors, you will look for codeword that is at minimal distance from it. You can use the example encoding files provided to generate the list of codewords.

3.3 Successive Cancellation Decoder

This algorithm is exactly as described in the class. Please refer to the following paper (Pseudo algorithms 2, 3, 4) for a more detail.

- List Decoding of Polar Codes, Ido Tal and Alex Vardy (<https://arxiv.org/pdf/1206.0050>)