EED 350

DIGITAL COMMUNICATION

ENDSEMESTER PROJECT REPORT

CRC Aided Decoding of Polar Codes

Author 1 :
Chaudhary RISHIKA
R.No. 1710110276

Author 2:
Kakkar PRANIKA
R.No.1710110253

Author 3:
Mittal ARJIT
R.No.1710110073

Instructor:
Vijay KUMAR CHAKKA

July 2, 2020

Abstract

CRC (cyclic redundancy check)-aided decoding schemes are proposed to improve the performance of polar codes. A unified description of successive cancellation decoding and its improved version with list or stack is provided and the CRC-aided successive cancellation list/stack (CA-SCL/SCS) decoding schemes are proposed. Simulation results in binary-input additive white Gaussian noise channel (BI-AWGNC) show that CA-SCL/SCS can provide significant gain of 0.5 dB over the turbo codes used in 3GPP standard with code rate 1/2 and code length 1024 at the block error probability (BLER) of 10 -4 . Moreover, the time complexity of CA-SCS decoder is much lower than that of turbo decoder and can be close to that of successive cancellation (SC) decoder in the high SNR regime.

Contents

List of Figures

1 Introduction

- **Polar Codes** :
  - Invented by Erdal Arikan in 2008
  - They are the first codes to have explicit proof for approaching capacity of the **Binary Discrete Memoryless channel**
  - They are included as the codes for the control channels in 5G standard
  - They are sequential in nature, i.e encoded in some order
  - Defined using a generator matrix in a recursive fashion
- **CRC** :A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents. On retrieval, the calculation is repeated and, in the event the check values do not match, corrective action can be taken against data corruption. CRCs can be used for error correction.
- **Channel Polarisation** : Combining multiple instances of a channel to get either ideally noiseless channels or very noisy channels, obtained from a single noisy channel through clever techniques, is channel polarisation.

2 Preliminary

2.1 Polar Transform

Shown below is an example of Polar transform using a Generator Matrix, where the matrix [2] $G_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$
Where

$$u_1, u_2$$

are the two inputs which are present at the leaf nodes and we get our output, which is a 2 length vector at the root node. The output is shown below below.

$$[u_1 u_2] G_2 = [u_1 \oplus u_2 u_2] \tag{1}$$

(N,K) Polar Code

$G_{2^n} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{\otimes n}$

- N = 2^n
- G_N = N x N matrix, Kronecker product of 2 x 2 kernel
- Binary tree representation
  - Depth n
  - $u^{(N)}$ = u G_N evaluated on tree with u at the bottom and $u^{(N)}$ at top
- A (N,K) Polar code -Message m of length K bits
- Polar Encoding - a vector u of length N bits as follows
  - Find N-K least reliable channels from the reliability sequence
  - Set u_i for those N-K channels to zero (frozen bits)
  - m : Remaining k bits of u (message bits)

2.2 Notations and the A Posteriori Probability

Assuming the communication over a B-DMC $\mathbf{W : X \rightarrow Y}$,where X and Y denote input and output alphabet respectively. The channel transition probabilities are defined as

$$W(y|x), \;\; x \epsilon X, \;\; y \epsilon Y \tag{2}$$

x is a uniform distribution on X , thus the channel a posteriori probabilities can be written as:-

$$P(x|y) = \frac{W(y|x)}{\sum_v W(y|v)} \tag{3}$$

After channel combining and splitting operation on N = 2^n, we get N successive uses of synthesised binary input channels $W_N^{(i)}$ with i = 1,2,..N . The information bits can be assigned to the channels with indices in the information set A, which are the more reliable subchannels .The complementary set A^c denotes the frozen bit set and the frozen bits uA^c which are the least reliable bits are therefore set to all zeroes, for the symmetric channels.

3 Polar Decoding Algorithms

3.1 Successive Cancellation Decoding

SC decoder is used for decoding of polar codes. This is also known as sequential decoder. Where, $u_1, u_2, ..u_N$ are the inputs to polarization kernel G_N Some of these inputs are frozen bits and the others are message bits. After the polarization is done, we have **Binary Phase Shift Keying** is done and noise is added. After which we get our received vector $r^{(N)}$.The goal is to get back our input for the received vector.

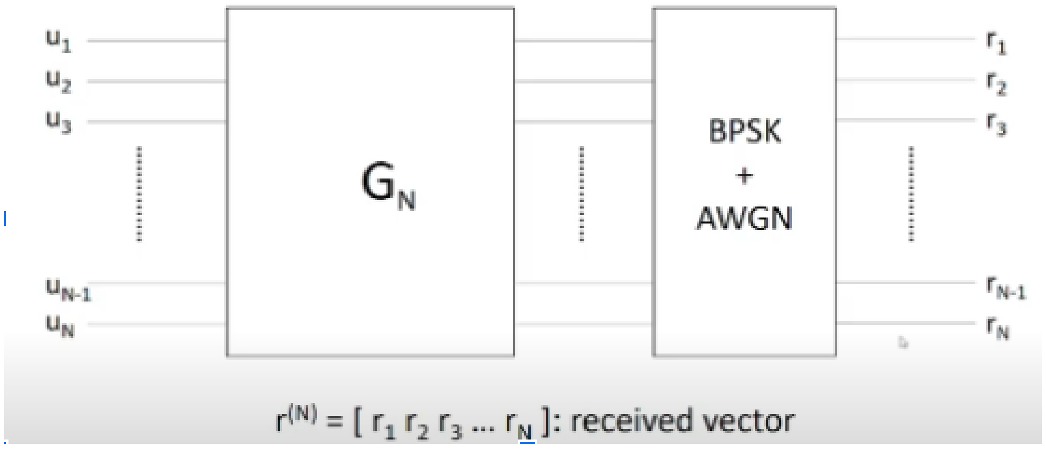


Figure 1: SC Decoder Representation

Sequence of SC decoder operations

- Start at root and go till the leaf node like in a binary tree Use the following methods for each node while traversing back from leaf to root node to get the final decision:
- If not leaf, do the following sequence :
  - Do step L and go to left child
  - When decision is received from left child, do step R and go to right child
  - When decision is received from right child, do step U and go to the parent
- If leaf, make a decision and go to parent

Steps L, R and U are explained in detail below

- **Step L (Left)**: As shown in the above figure, the parent node receives M incoming beliefs . Then min sum is carried out but in a certain way. The vector L is split into two vectors and then min sum is applied according to the coordinate resulting into a M/2 vector. Thus, M/2 values are sent to the child.
- **Step R (Right)**:Now, the M/2 bits are sent back from the left child to the parent as shown above (decession). Then the g operation is carried out, again splitting L vector into two parts along with the decisions acquired from the left child. Again the g operation is carried out coordinate wise as shown in the figure.
- **Step U (Back to parent node)**: For the last step , we get our decisions back from the right child also which are of M/2 bits. Together from the left and right child The parent node receives M bits and then the XOR addition is carried out to give the decision to the parent node.

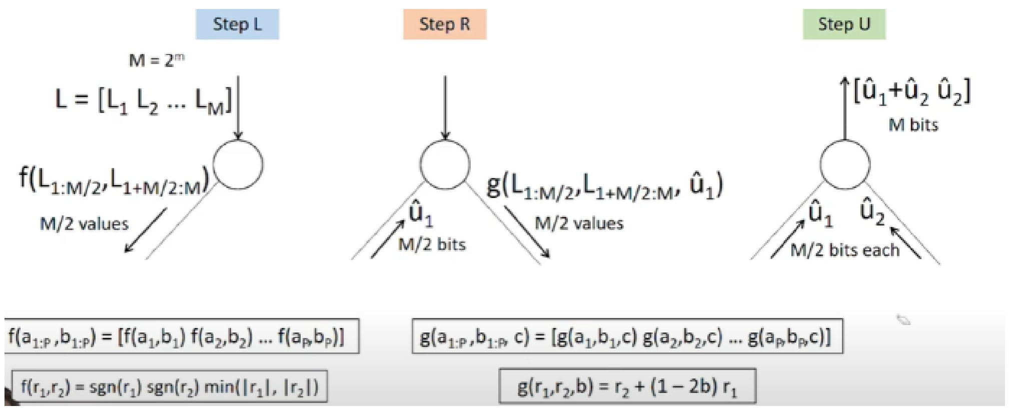

Step L
Step R
Step U
$M = 2^m$
$L = [L_1\ L_2 \dots L_M]$
$f(L_{1:M/2}, L_{1+M/2:M})$
M/2 values
$\hat{u}_1$
M/2 bits
$g(L_{1:M/2}, L_{1+M/2:M}, \hat{u}_1)$
M/2 values
$[\hat{u}_1 + \hat{u}_2\ \hat{u}_2]$
M bits
$\hat{u}_1$ $\hat{u}_2$
M/2 bits each
$f(a_{1:P}, b_{1:P}) = [f(a_1,b_1)\ f(a_2,b_2) \dots f(a_P,b_P)]$
$g(a_{1:P}, b_{1:P}, c) = [g(a_1,b_1,c)\ g(a_2,b_2,c) \dots g(a_P,b_P,c)]$
$f(r_1,r_2) = \text{sgn}(r_1)\,\text{sgn}(r_2)\,\min(|r_1|, |r_2|)$
$g(r_1,r_2,b) = r_2 + (1-2b)\,r_1$

Figure 2: The 3 decision making steps

Example of SC decoder using binary tree [2]

- This successive decoding can be represented as a path searching process on a code tree. For a polar code with code length N, the corresponding code tree T is a full binary tree
- More specifically, T can be represented as a 2-tuple (V, E) where V and E denote the set of nodes and the set of edges respectively.

- The depth of a node v $\in$ V is the length of the path from the root to the node. The set of all nodes at a given depth d is denoted by Vd, d = 0, 1, , N. The root node has a depth of zero.
- A i-length decoding path $e_1, e_2, ..e_i$ consists of i edges, with $e_i \in E_i$, i $\in$ 1, 2, $\cdots$, N. A vector v_1^i is used to depict the above decoding path, where vi is corresponding to the binary label of edge e_i.
- The reliability of a decoding path v_1^i can be measured using a posteriori probability .
- Time complexity is **O(N log N)** and space complexity is **O(N)**.

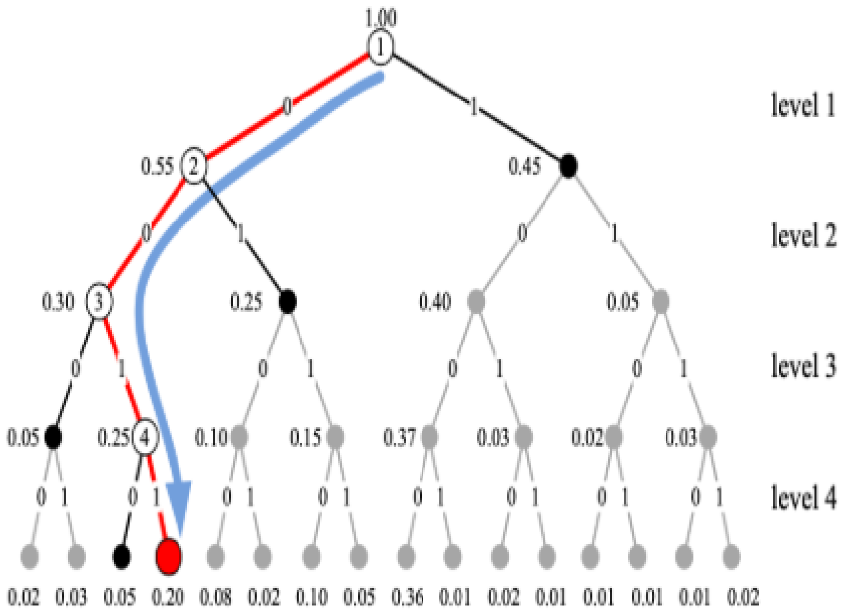


Figure 3: Example of a binary tree

Figure 3: An example of SC Decoding with code length N=4 and the path as 0011. But ,the path 1000 has the largest probability of all the N-length paths, but it failed in the competition at the first level.

3.2 Successive Cancellation List Decoding

SC decoding doesn't always provide with an optimum path, an enhancement was done in the scheme. The aim is to reduce the complexity and save the time for the iterations. To do so, an enhancement was done in the pre-existing SC decoding scheme and it was named as “SCL decoding” as List is being used in the enhanced scheme. The main reason behind designing a new scheme was that, in SC decoding one of the child that might give the optimum path was discarded at the very first level while traversing. So, In SCL, a List is used while traversing the tree. The length(L) of the list can be provided by the user. Level-by-level traversing is done so that an optimum path is not lost or ignored. This can also be called as breadth-first traversal, with searching width L.

Sequence of SC List decoder operations[2]

- **Initialization.** A null path is included in the initial list and its metric is set to zero, i.e.$L^{(0)} = \phi$, M $(\phi) = 0$
- **Expansion.** At the i-th level of the code tree, the number of candidate paths in the list are doubled by concatenating new bits v_i ,taking values of 0 and 1 respectively, that is

  for each $v_1^i \epsilon L^{(i)}$ the corresponding path metric(s) are updated.
- **Competition.**If the number of paths in the list after expansion is no more than L, just skip this step; otherwise, reserve the L paths with the largest metrics and delete the others.
- **Determination.** Repetition of Expansion and Competition is done until level-N is reached. Then, the decoder outputs the estimated source vector $u_1^N = v_1^N$, where v_1^N is the binary labels of the path with the largest metric in the list.

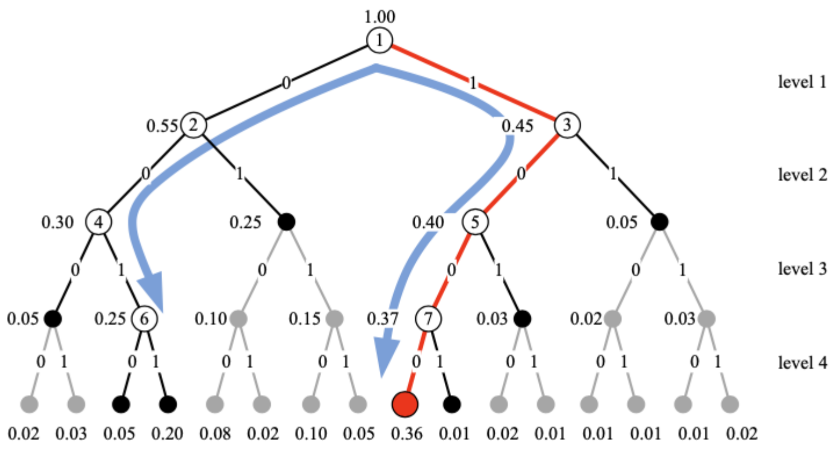


Figure 4: Successive Cancellation List Decoding

Figure 4: An example of SCL Decoding with L =2.It finds the most probable path as 1000.But the times of metric computations is increased.

Features of SC List decoder

- The successive cancellation list (SCL) decoder searches level-by-level on the code tree.
- However, unlike SC where only one path is reserved after processing at each level, SCL allows at most L candidate paths to be further explored at the next level.
- SCL can be regarded as a breadth-first search on the code tree T with a searching width L.
- During the SCL decoding, a bunch of candidate paths will be obtained and stored in a list. Since for every single candidate path, the metric calculations and bit determinations are still performed bit- by-bit successively.
- Space Complexity of SCL : *O(LN)*
- Computational Complexity of SCL :*(O(LN log(N))*

3.3 Successive Cancellation Stack Decoding

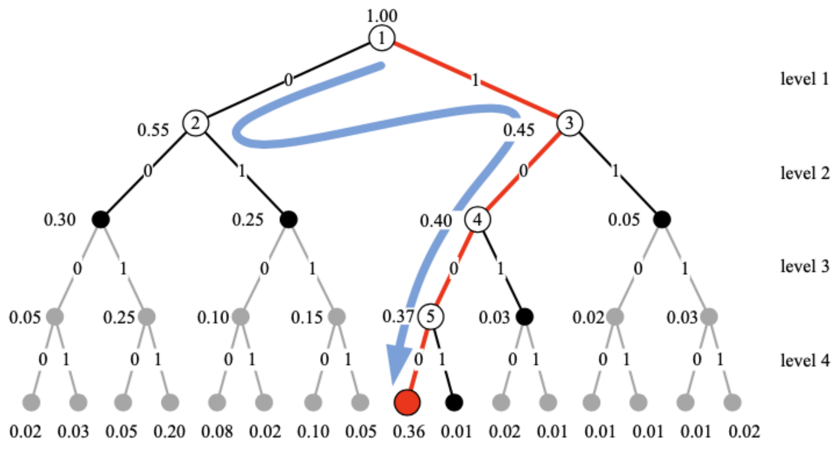


Figure 5: Successive Cancellation Stack Decoding

Figure 5: An example of SCS Decoding compared with SCL, SCS can also find the most probable path 1000 with fewer metric computations

Features of SC Stack decoder[2]

- **Initialization**: One null path is included in the initial list and its metric is set to zero, i.e. $L^{(0)} = \phi$, M $(\phi) = 0$
- Whenever the top path in the stack which has the largest path metric reaches length N, the decoding process stops and outputs this path.
- Unlike the candidate paths in the list of SCL which always have the same length, the candidates in the stack of SCS have difference lengths.
- Space Complexity of SCS : *O(DN)*
- Computational Complexity of SCS : *O(LN log(N)*

4 CRC-Aided Decoding of Polar Codes

After studying the Successive Cancellation Decoding example, we saw that the binary tree did not follow the optimal path , i.e. the path with the highest probability , therefore some improvement is required. So, we study List Decoding. The basic idea of a decoder is to take a received vector and then give out a possible codeword , now in list decoding instead of one possible codeword, a list of codewords is given. Now, the job is to choose from the list of possible codewords, for this purpose we are using cyclic redundancy checks. They have error detecting properties ,very simple and efficient to use.

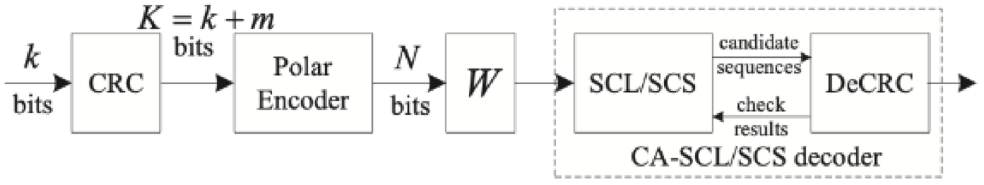


Figure 6: Polar Coding and CRC- aided Decoding Schemes

- Starting with k message bits, m length CRC bits are added to it. The total length is now K bits.
- K bits is the actual message for encoding not the raw k message bits. Then polar encoding is done.
- After polar encoding, we get N bits codeword which is subjected to BPSK AWGN. Then we get our received values.
- The next step is to decode our received values and thus SCL and SCS decoding is done, which gives us not one codeword but a list of codewords to choose from.
- For the last step, we check CRC , if the codeword passes the CRC, that will be our message estimate.

4.1 CRC aided Successive Cancellation List Decoding

The CRC-aided SCL decoding algorithm with the size of list L, denoted by CA-SCL (L), can be described as follows [3],[4]:

- **Initialization.** A null path is included in the initial list and its metric is set to zero, i.e.$L^{(0)} = \phi$, M $(\phi) = 0$
- At the i-th level of the code tree, double the number of candidate paths in the list by concatenating new bits, taking values of 0 and 1 and update the respective path metric.

$$ifL(u_i) \geq 0has(DM)_i = 0, \hat{u}_i = 1hasDM = |L(u_i)|$$

$$ifL(u_i) < 0has(DM)_i = 1, \hat{u}_i = 0hasDM = |L(u_i)|$$

- If the number of paths in the list exceeds L in the last step then keep the best L paths with largest metrics and discard the rest.
- How to choose path:
  - Repeat the last two steps until Level-N is reached.
  - The path that passes the CRC detection is estimated as the Optimum path by the decoder.
  - If none of the path passes the CRC-detection after traversing the whole L, then the algorithm declares an output failure.
- For the last step, we check CRC , if the codeword passes the CRC, that will be our message estimate.

4.2 CRC aided Successive Cancellation Stack Decoding

Let D and T denote the maximal and instantaneous depth of the stack in SCS decoder respectively. An additional parameter(from SCS decoding) Q is introduced to limit the extending paths that have same length. A counting vector $q_N^1 = q_1, q_2, ..q_N$ is used to record the number of the popping paths with length ,i.e. q_i means the number of popping paths with length-i during the decoding process. SCS decoder can successively output at most Q length-N candidate paths. Note that, in CRC-aided decoding scheme, SCS will not stop until CRC is passed or the limit Q is reached. The CA-SCS algorithm with the path counting limit Q and maximal stack depth D, denoted by CA-SCS (Q, D), is summarized as follows [2] :

- **Initialization** : Push the null path into stack and set the corresponding metric M $(\phi) = 0$. Initialize the counting vector q_N^1 with all-zeros and set $T = 1$.
- **Popping** : Path d_1^{i-1} popped from the stack, and rest variables are updated with$T = T - 1 and$ if the path is not null, set $q_{i-1} = q_{i-1} + 1$.
- **Competition** : If q_{i-1} reaches the limit Q then, delete all the paths with length equal to or less than $i - 1$.
- **Expansion** : For a frozen bit d_i , simply extend the path to $d_1^i = (d_1^{i-1}, 0)$ otherwise, if d_i is an information bit,extend current path to $(d_1^{i-1}, 0)$ and $(d_1^{i-1}, 1)$ (Here we extend for both possible decision). And calculate the path metric for the same.
- **Pruning** : For information bit d_i if $T > D - 2$ delete the path from the bottom of the stack and set $T = T - 1$. Then push the two extended paths into the stack and set $T = T + 2$. Otherwise, for frozen bit d_i , push the path $d_1^i = (d_1^{i-1}, 0)$, into the stack and set $T = T + 1$.
- **Sorting** : the path in descending order from top to bottom in the stack.
- **CRC-aided decision** : Now comes the most important decision (CRC)
  - If the top path in the stack reaches the leaf node(that has no Left or Right child) then pop that path and set $T = T - 1$ and $q_N = (q_N + 1)$
  - A CRC check is performed for the path d_1^N :
    - If passed, then this is the estimated path. Algorithm stops and d_1^N is taken as the decision sequence.
    - If not, then this top path is popped out and CRC is checked on the next path.
  - If the CRC checking is not passed on any path or the stack is empty then the algorithm stops and declares failure. Otherwise goes back to popping stage.

5 Performance and Complexity Comparison

CRC-aided decoding schemes are proposed to improve the performance of polar codes.Expected simulation results according to the paper shown in figure 7 and figure 8 corroborate that CA-SCL/SCS can provide significant gain of 0.5dB over the turbo codes.Moreover, the time complexity of CA-SCS decoder is much lower than that of turbo decoder and can be close to that of SC decoder in the high SNR regime . We would try and implement the same graph four our observations[1].

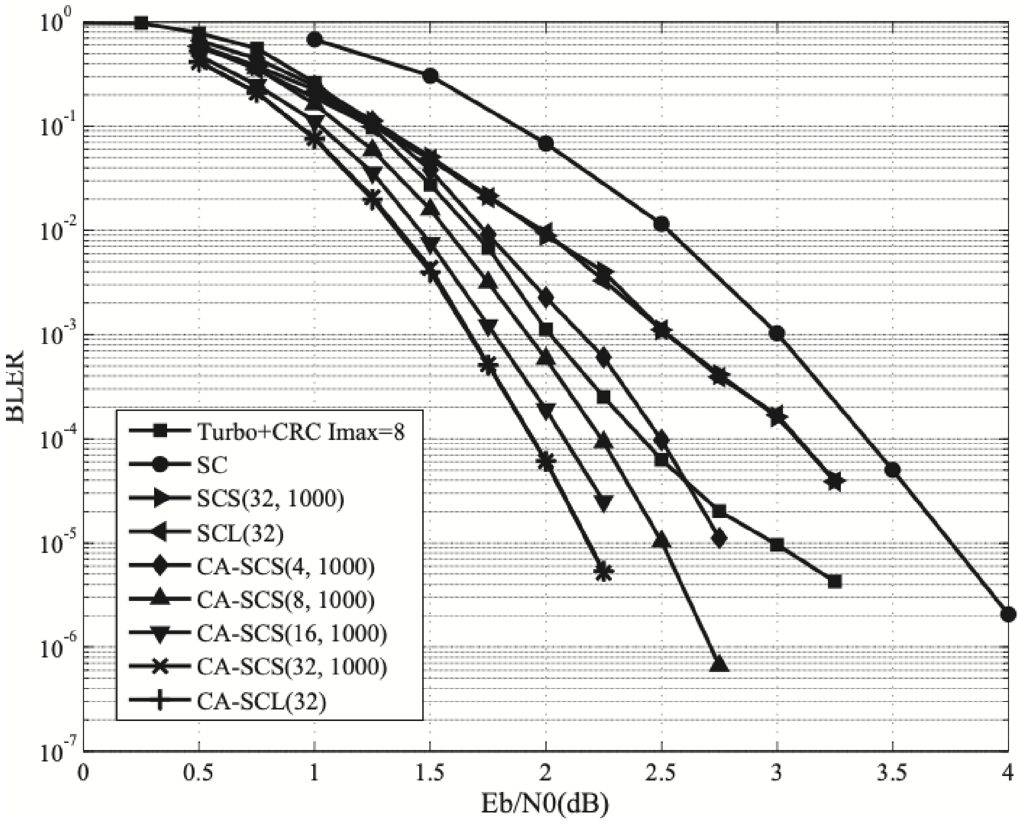


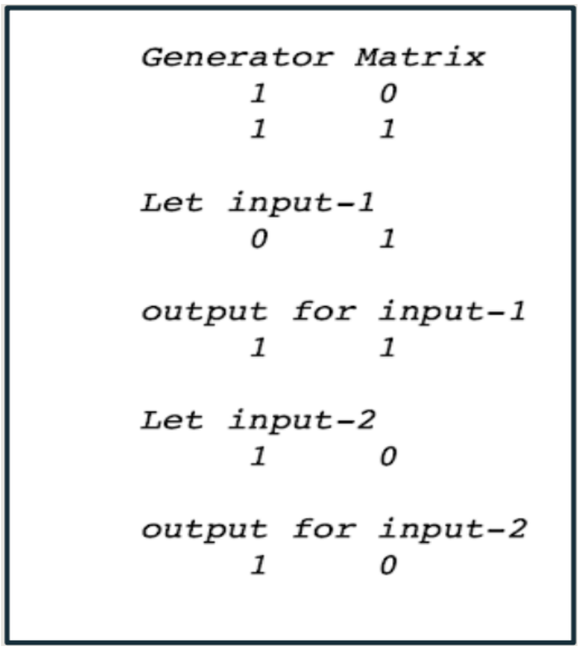
Figure 7: BLER performance comparisons for polar codes

Expected Results-BLER performance comparisons for polar codes and turbo codes with block length N=1024 and code rate R=1/2

6 Results and Conclusions

6.1 MATLAB code results

The following figure 8 shows the polar transform obtained for a 2 bit input. It is a MATLAB published screenshot. The result is accomplished using the generator matrix and the length the polar transformed output is 2 bits.



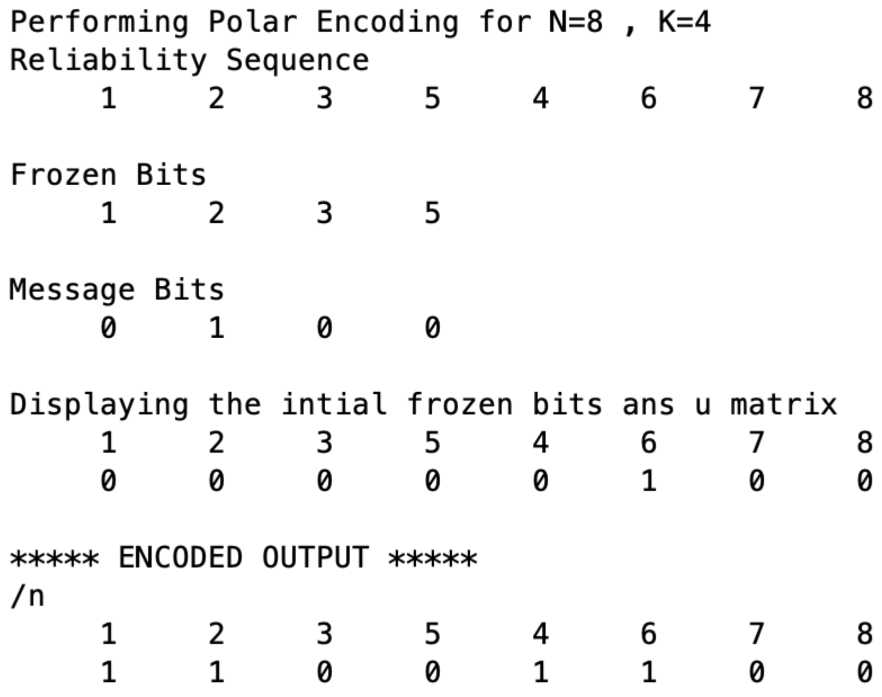
```
Generator Matrix
     1      0
     1      1

Let input-1
     0      1

output for input-1
     1      1

Let input-2
     1      0

output for input-2
     1      0
```

Figure 8: Matlab Implementation of Polar Transform (2 bits to 2 bits)

Polar encoding was implemented in MATLAB and following is the algorithm :

- Extracting the reliability sequence i.e Q1=Q(Q¡=N).
- Define the Frozen position-Q1(1:N-K).
- Defining the message positions Q1(N-K+1:end).
- Generating a random message
- Defining u as u=zeros(1,N) and the bits that are not frozen will be assigned the message.

- Next the computation of bits on each depth of the binary tree takes place ranging from (N-1 ,-1, 0) where 1 bit is combined at a time and stored in two parts a and b.



```
Performing Polar Encoding for N=8 , K=4
Reliability Sequence
     1     2     3     5     4     6     7     8

Frozen Bits
     1     2     3     5

Message Bits
     0     1     0     0

Displaying the intial frozen bits ans u matrix
     1     2     3     5     4     6     7     8
     0     0     0     0     0     1     0     0

***** ENCODED OUTPUT *****
/n
     1     2     3     5     4     6     7     8
     1     1     0     0     1     1     0     0
```

Figure 9: Matlab Implementation of Polar Encoding

Now, we are going to compare the performance and complexity of CA-SCL and SC via simulations over binary-input additive white Gaussian noise channels(BI-AWGN's). Figure 9, gives us the BLER performance comparisons for various coding and decoding schemes.All the schemes have the code length of $N = 1024$ and code rate $R = 1/2$.

- We can see that the performance of SCL4 and SCL8 is much better than that of SC.
- At almost BLER value of 10^{-3} , there is a difference of 0.7 dB performance gap between CA-SCL and SC.

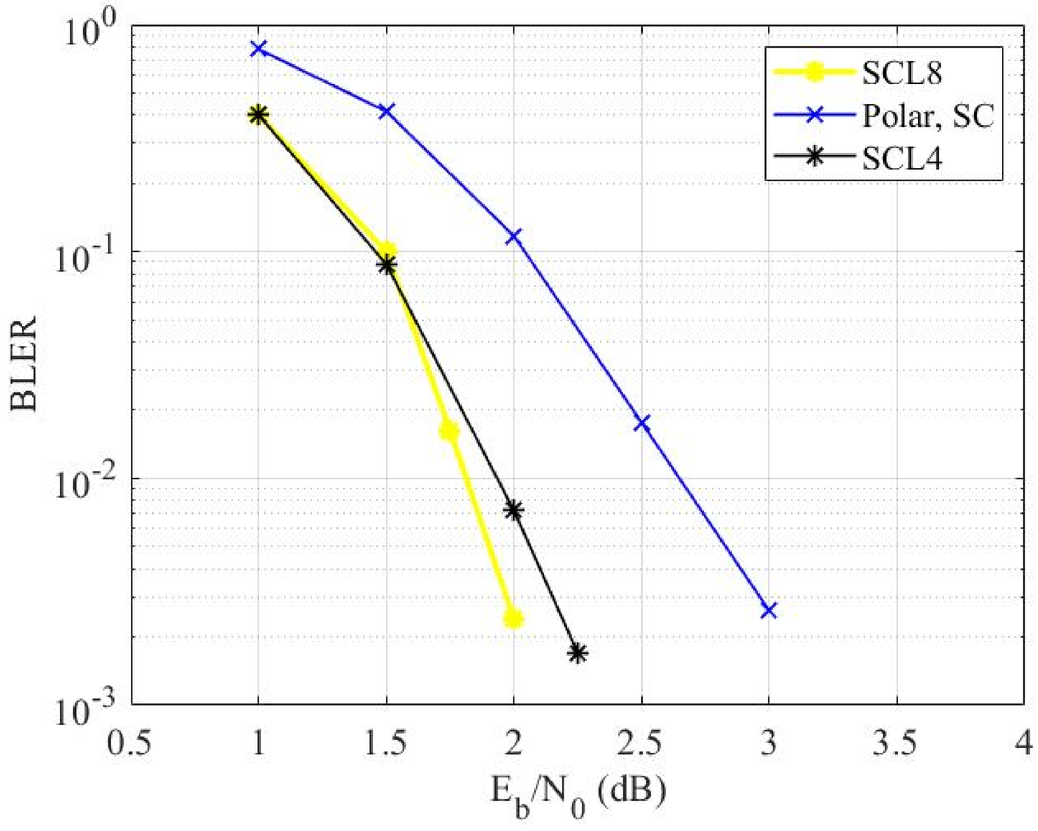


Figure 10: BLER performance comparisons for polar codes

6.2 Conclusions

Polar codes are the first capacity-achieve channel coding in history.The successive cancellation decoder is the first decoding algorithm for polar codes which can achieve binary memory-less symmetric channels' capacity. However, SC does not perform well.The SC decoder is known to have the least complexity but its corresponding performance is poor. We find that CRC aided SCL algorithm improves the effect of polar codes.SC List algorithm is proposed and is one of the best algorithms in terms of balance between bits error rate and computation complexity.CRC-aided decoding schemes are proposed to improve the performance of polar codes.They have increased complexity but their performance is better as compared to that of SC decoder which can provide a significant gain of around 0.7 dB.

7 References

[1] Bashar Tahir, Stefan Schwarz, and Markus Rupp,“ BER Comparison Between Convolutional, Turbo, LDPC, and Polar Codes”, Institute of Telecommunications,Technische Universitat (TU) Wien Vienna, Austria.

[2] Kai Niu and Kai Chen and Jia-Ru Lin ,“ Improved Successive Cancellation Decoding of Polar Codes”, arXiv:1208.3598v2 [cs.IT] 17 Jan 2013.

[3] I. Tal and A. Vardy, “List decoding of polar codes,” Inf. Theory Proc.,pp. 1-5, 2011

[4] K. Chen, K. Niu, and J. R. Lin, “List successive cancellation decoding of polar codes,” Electron. Lett., vol. 48, no. 9, pp. 500–501, 2012.

[5] I. Tal and A. Vardy, “List decoding of polar codes,” arXiv:1206.0050v1, May 2012.