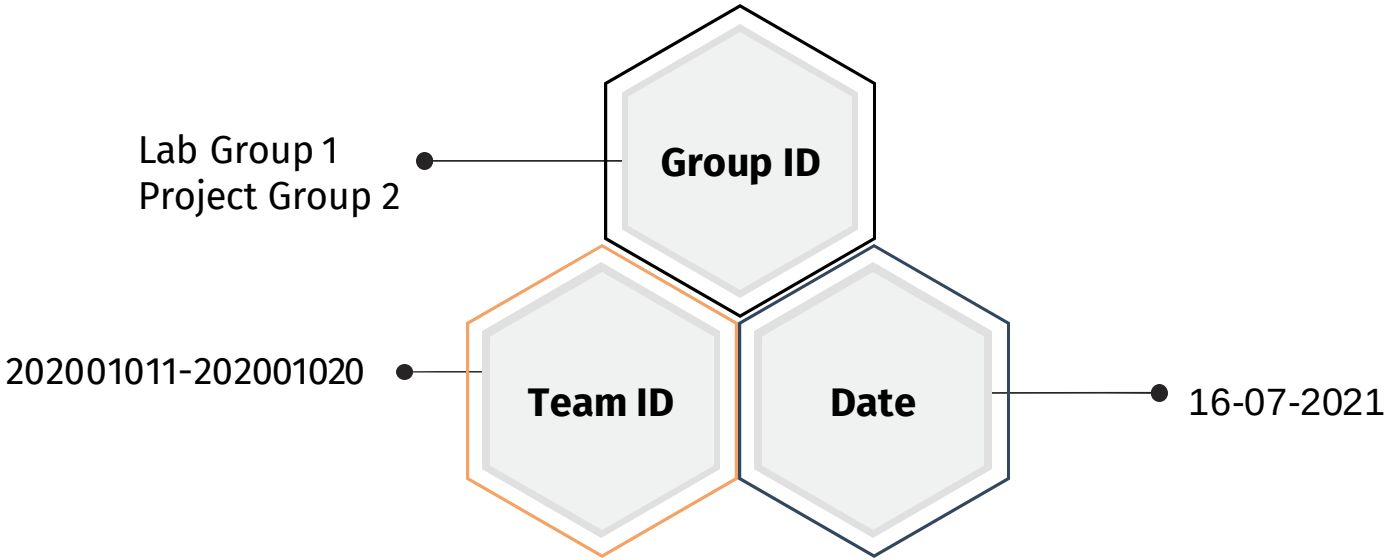
CT111

Introduction to communication Systems

End Semester Project

Professor : Yash Vasavada

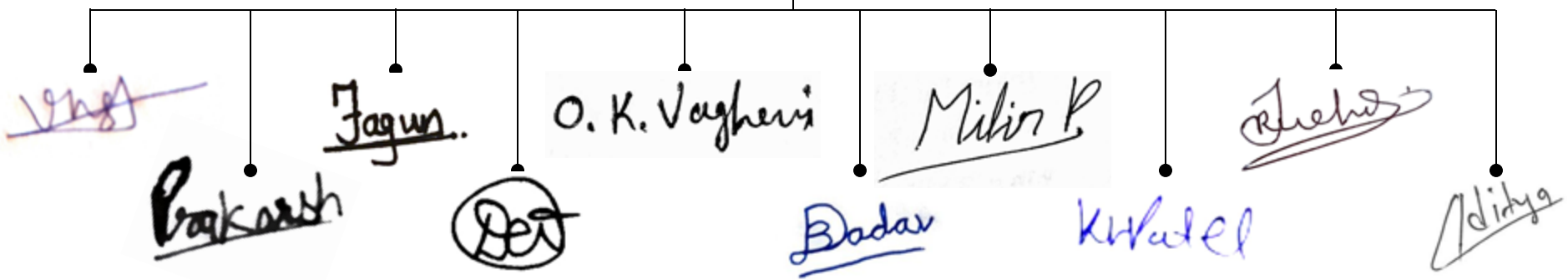

Lab Group 1
Project Group 2
Group ID
202001011-202001020
Team ID
Date
16-07-2021

Honor Code

We declare that,

- The work we are presenting is our own work, we have not copied work from any source
- The concepts, understanding, and insights we will be describing are our own
- Whatever we have relied on an existing work that is not our own, we have provided a proper reference citation
- We make this pledge truthfully. we know that violation of this solemn pledge can carry grave consequences

Signatures



Contribution

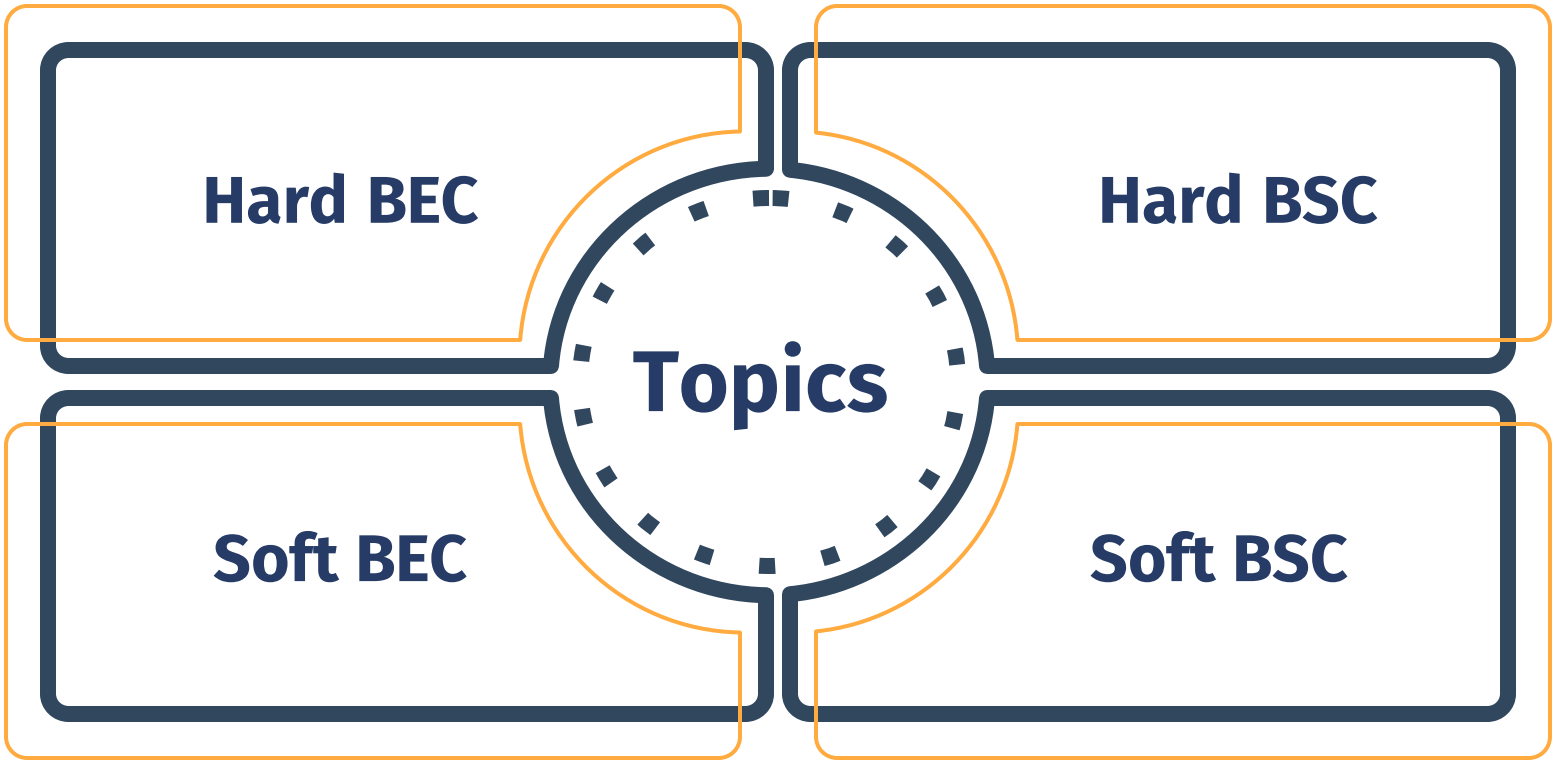
Topics	**Student Names**	**Student ID**	**Specific Contribution of the students**
Channel Implementation (BSC & BEC)	• Prakarsh Mathur • Fagun Vora • Ruchi Detroja	• 202001012 • 202001013 • 202001019	Created codes for noise generation of each channel.
Tanner Graph Implementation for message passing.	• Vansh Parikh • Dev Patel • Mihirkumar Patel • Kamal Patel	• 202001011 • 202001014 • 202001017 • 202001018	Logic was discussed and implemented in C++ by Kamal, Mihir and Dev. Vansh implemented same logic in python with minor changes.

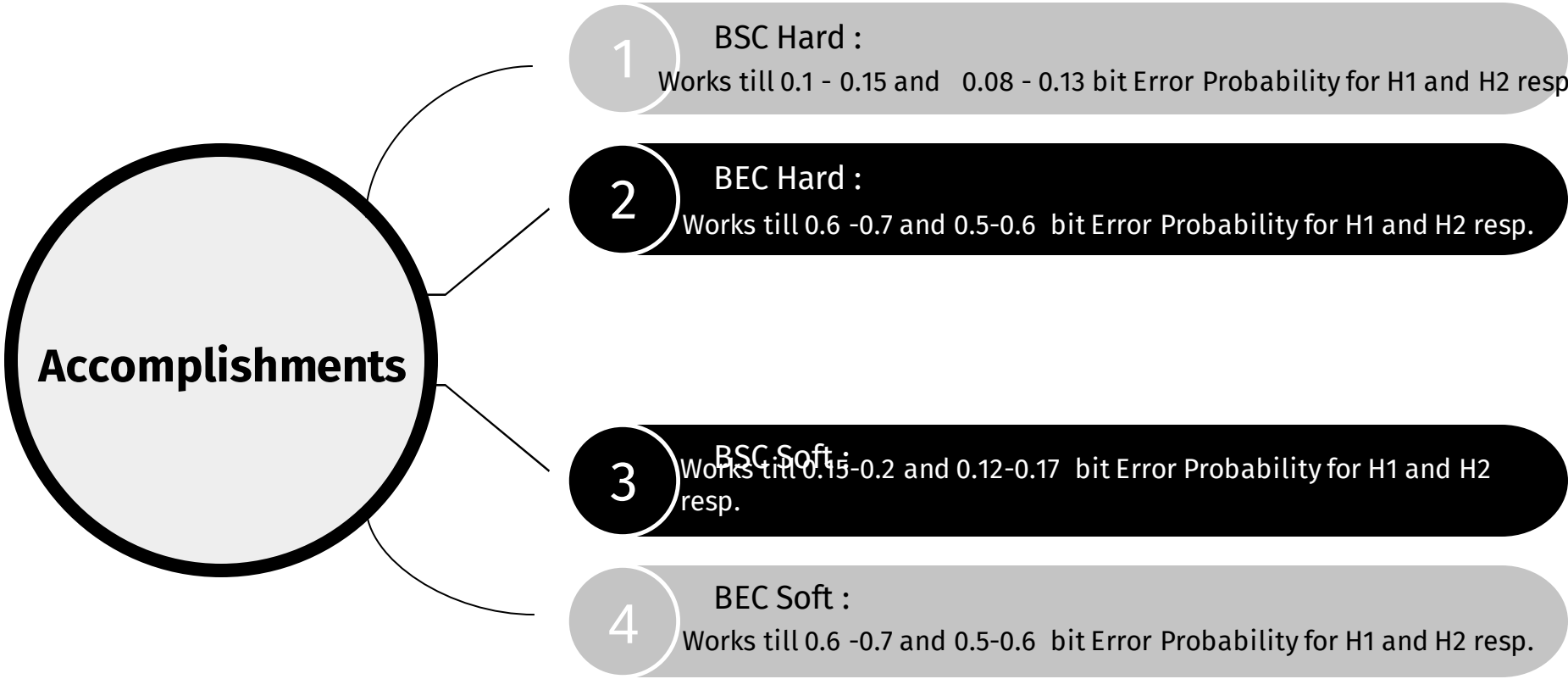
Contribution

BSC Hard Channel BEC Hard Channel BSC Soft Channel	• Vansh Parikh • Dev Patel • Mihirkumar Patel • Kamal Patel	• 202001011 • 202001014 • 202001017 • 202001018	C++ implementation of the logic was done by Kamal , Dev and Mihir. Python implementation was done by Vansh.
BEC Soft	• Vansh Parikh • Kamal Patel • Ruchi Detroja	• 202001012 • 202001018 • 202001019	C++ implementation of the logic was done by Kamal and Ruchi. Python implementation was done by Vansh.
Monte-Carlo Simulation: Probability of Successful Decoding (checking the probability for a range of probabilities)	• Vansh Parikh • Dev Patel • Mihirkumar Patel • Kamal Patel	• 202001011 • 202001014 • 202001017 • 202001018	C++ implementation of the logic was done by Kamal , Dev and Mihir. Python implementation was done by Vansh.

Contribution

Monte-Carlo Simulation: Algorithm Convergence	• Vansh Parikh	• 202001011	Implemented the code and plotted the graph in python.
Graph and Comments	• Om Vaghani • Aditya Patel	• 202001015 • 202001020	Om plotted the graph in c++ but later on we used matlab graphs Aditya did commenting in the codes
Presentation	• Whole Group		Design Part was done by Dev Jadav and Fagun vora. Content Writing was done by whole group.


Hard BEC
Hard BSC
Topics
Soft BEC
Soft BSC


Accomplishments
1
BSC Hard :
Works till 0.1 - 0.15 and 0.08 - 0.13 bit Error Probability for H1 and H2 resp
2
BEC Hard :
Works till 0.6 -0.7 and 0.5-0.6 bit Error Probability for H1 and H2 resp.
3
BSC Soft :
Works till 0.15-0.2 and 0.12-0.17 bit Error Probability for H1 and H2 resp.
4
BEC Soft :
Works till 0.6 -0.7 and 0.5-0.6 bit Error Probability for H1 and H2 resp.

Problems and solutions

Likelihood ratio going to infinity due to data type limitations

Gave a particular high value to it

Time complexity using Linked List

Used Array

Less decoding is occurring due to addition of feedback

We removed feedback.

Monte-Carlo Simulations

First, we made the decoders for all the four schemes. We took an all zero-codeword

For each simulation, we generated a new binary noise string of length equal to the original codeword i.e. all zero-codeword

For BSC channel, when the noise value is equal to 1, we flipped the bit.

For BEC channel, when the noise value is equal to 1, we erased the bit.

For each simulation, we give the noisy codeword to our decoder and check the result with the all zero-codeword

Improving Time and Space Complexity

We implemented Hard decision decoder for BSC channel using Linked List Data structure

Advantage
Works For different Degree of check nodes or variable nodes

Disadvantage
Traversing Linked list is a time taking process

After implementing our code (using Arrays), we still had time problems for Big H-matrices

To overcome this problem, we used a temp variable for storing messages at each CN and this helped us in reducing the number of loops we used previously over all the CNs.

Hard BEC

Decoder

Algorithm for Decoding

Hard BEC:

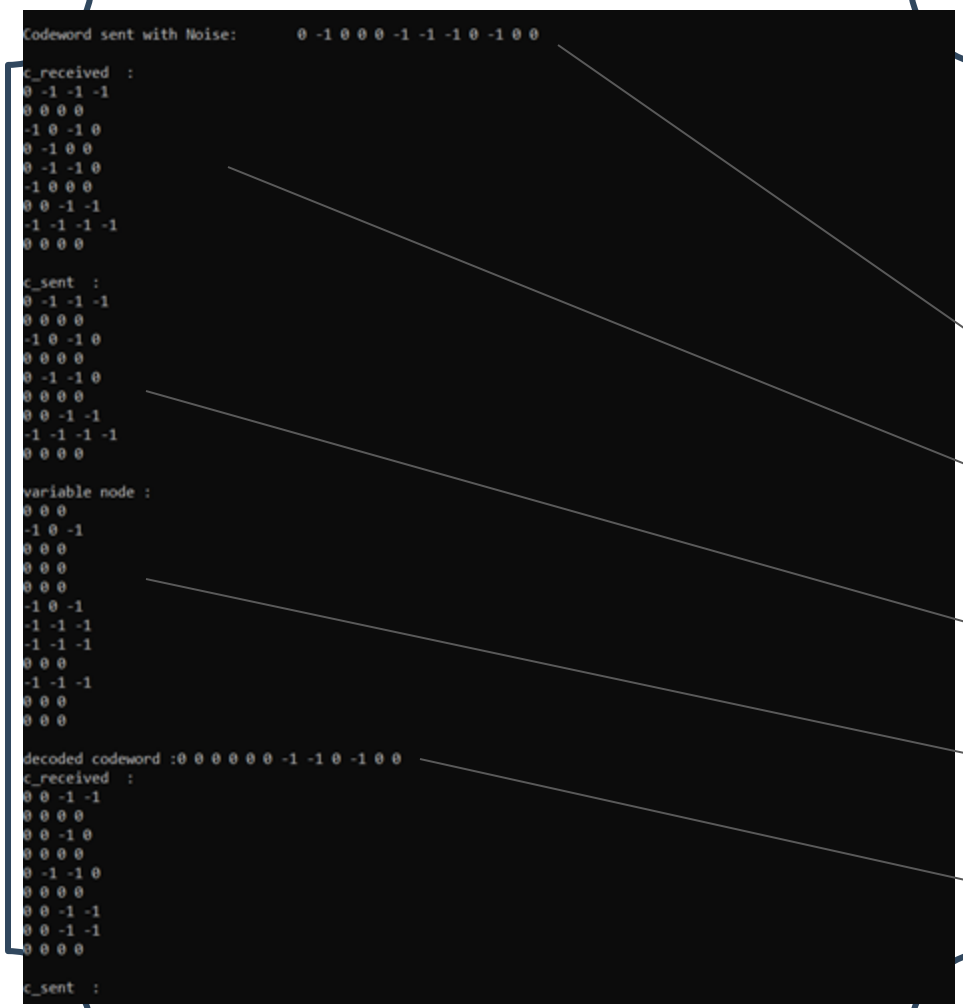
- In the first iteration, we will load the variable nodes in the array respectively. I.e there is no bit error we will send 0 else we will send -1.

- Then we will iterate through check nodes, if there is any erasure in variable nodes connected to that check nodes, then
  - If there is one erasure then it is solvable. So,
    - The value of that erased node will be **XOR** of all other variable nodes' values.
  - If there is more than one erasure then it is not solvable.
- Then we will return the relative value to the variable node.

- If there is no erasure in all variable nodes or if there is no change in e^{th} codeword and e^{th+1} codeword then we will break the loop.

- Then we will check if transmitted and received codeword are same or not. If both are not same then we will consider that the codeword is not decoded and error count increases.

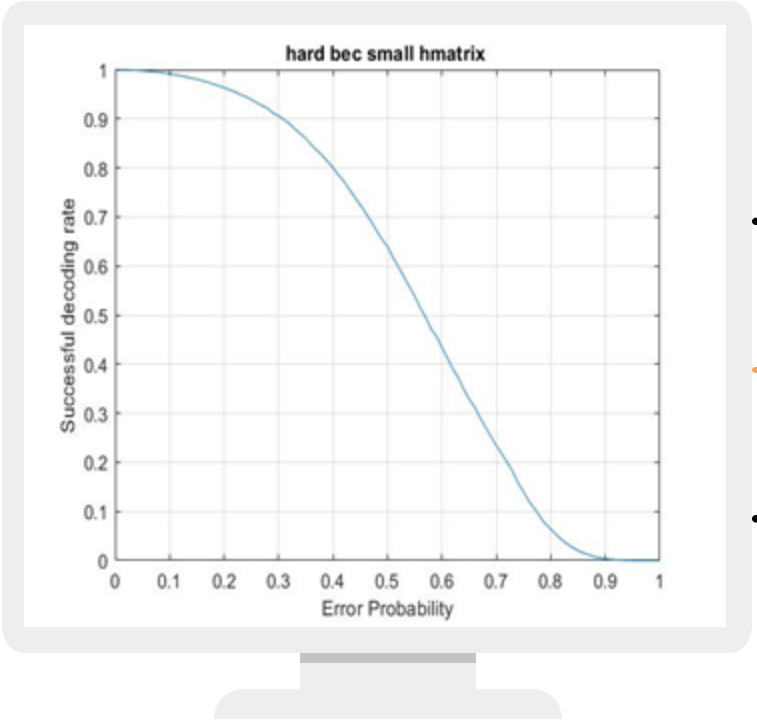
Snippet representing a short implementation of above discussed Hard BEC algorithm



Output shows Algorithm:

- Noise added Received message
- c_received contains values at CN's received from connected VN's
- c_sent is updated value after SPC decoding
- variable node contains values at each VNs received from connected CNs
- Decoded message is Majority voting at each VN

For small H matrix (9,12)

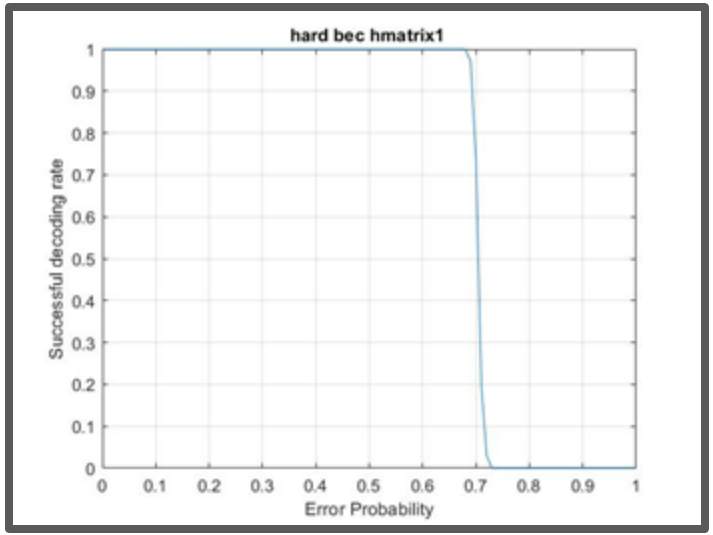


This graph shows Success Rate Vs Error Probability(p) for 10^5 Monte-Carlo Simulations.

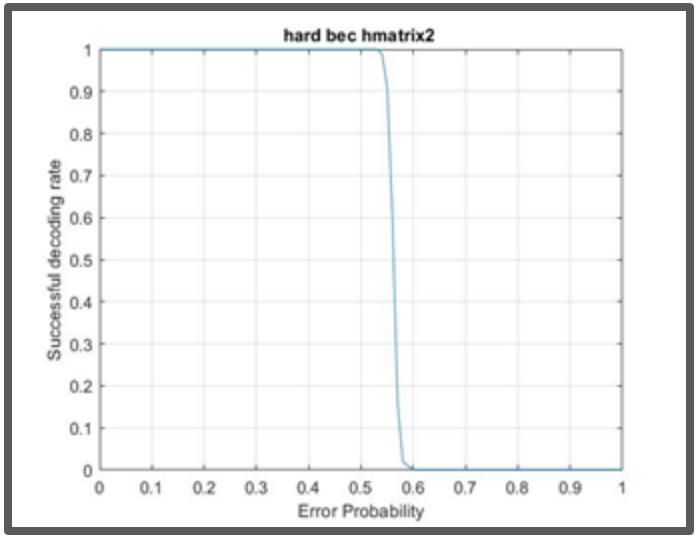
For Bit erasing probability / BEC(p) greater than 0.8 or 0.9 shows that it is hard / almost impossible to recover the original bit message.

The probability of success is monotonically decreasing because as probability of erasing bits increases, it is hard for decoder to decode the codeword.

H matrix 1



H matrix 2



- When the error probability is low the success rate is high for big H-matrix
- When the error probability is high the success rate is high for small H-matrix.

Soft BEC

Decoder

Algorithm for Decoding

Soft BEC:

- In the first iteration, we will load the variable nodes in the array respectively. I.e there is no bit error we will send 0 else we will send 0.5.
- We have made spc function that will take the probability of variable nodes and return likelihood ratio(odds in favor of variable node=1) for all check nodes.
- Then we will multiply all the likelihood ratios of check nodes for that variable node.
- If for all the variable nodes likelihood ratio is other than 1 then we will consider that the codeword is successfully decoded.

Snippet representing a short implementation of above discussed Soft BEC algorithm

Output shows Algorithm:

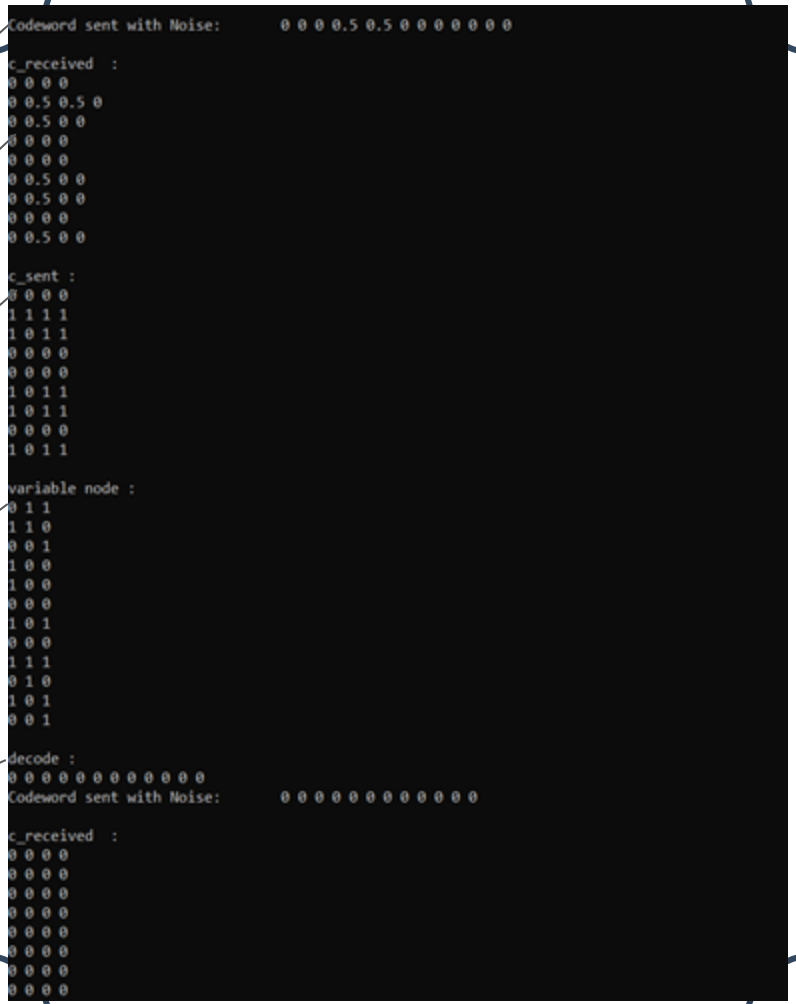
Noise added Received message/ probabilities

c_received contains values at CN's received from connected VN's

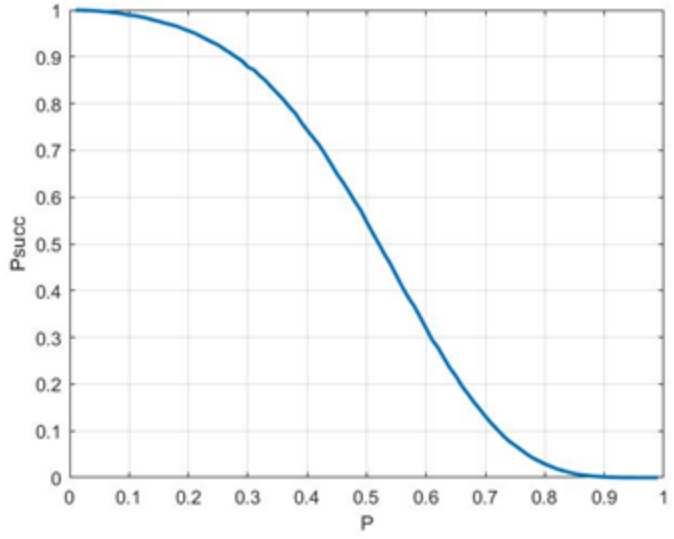
c_sent are updated values after SPC decoding

variable node contains values at each VNs received from connected CNs

Decoded message is after Majority voting at each VN



For small H matrix (9,12)

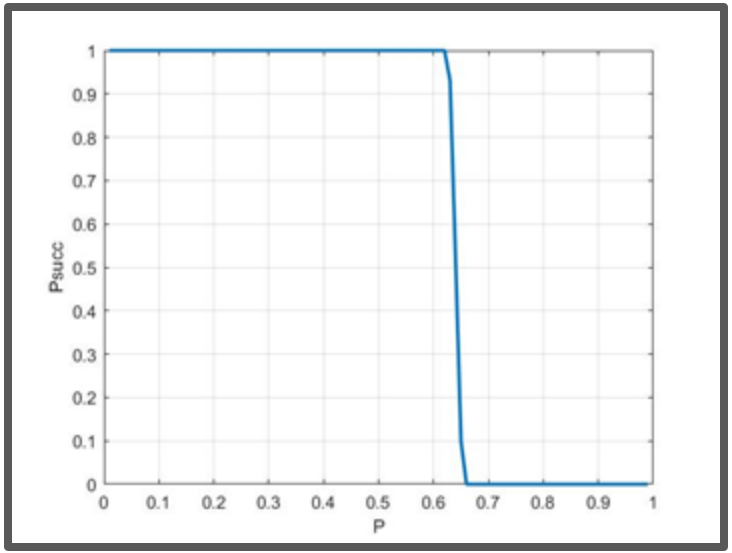


This graph shows Success Rate Vs Error Probability(p) for 10^4 Monte-Carlo Simulations.

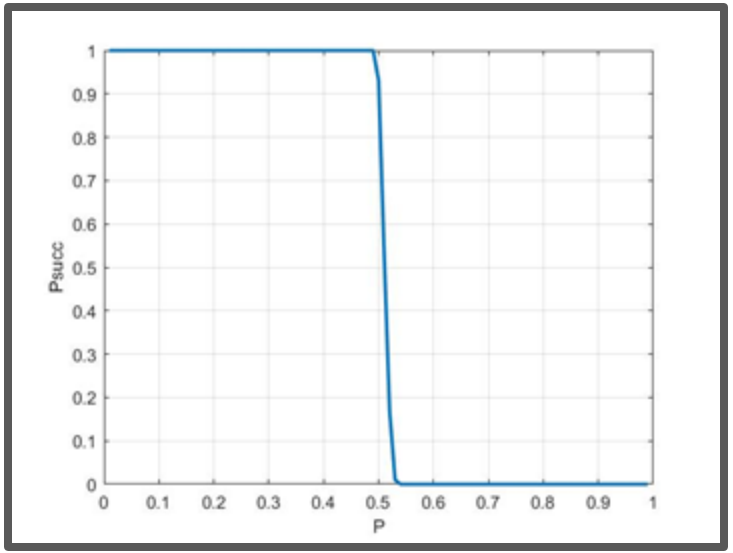
For Bit erasing probability / BEC(p) greater than 0.8 or 0.9 shows that it is hard / almost impossible to recover the original bit message.

The probability of success is monotonically decreasing because as probability of erasing bits increases, it is hard for decoder to decode the codeword.

H matrix 1



H matrix 2



- For BEC channel we know whatever bit we will have is either confirmed or eraser.

Hard BSC

Decoder

Algorithm for decoding

Hard BSC:

- In the 0^{th} iteration, we will load the VNs in the array with the values of the corresponding received bits.
- Then we will iterate through the CNs, update all the VNs connected to it with SPC decoding of remaining bits of that check node .
- Then we will return the decoded values to all the respective variable nodes.
- If there is no update in e^{th} codeword and e^{th+1} codeword, then we will break the loop. Otherwise in next iteration we will again load the check nodes with updated values of variable nodes .
- After breaking of loop, we will check if the transmitted and received codeword are same or not, If both are same, then we will consider that the codeword is correctly decoded and the success count increases .

Snippet representing a short implementation of above discussed Hard BSC algorithm

Output shows Algorithm:

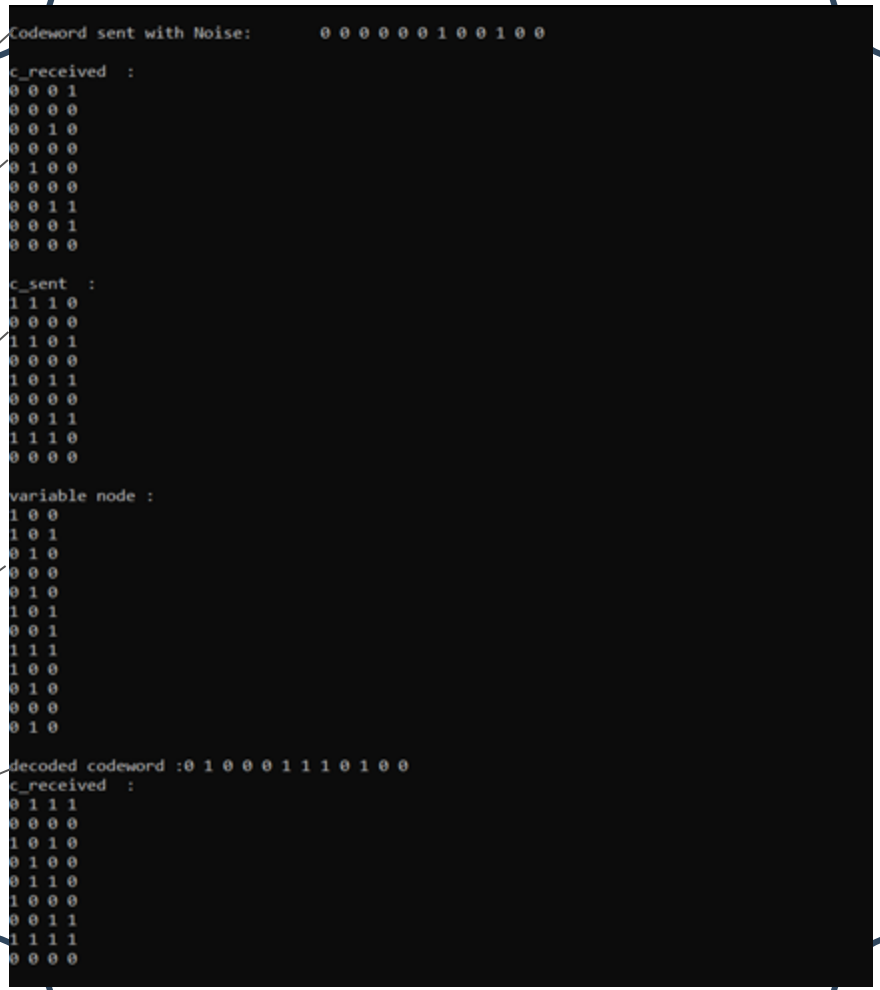
Noise added Received message

c_received contains values at CN's received from connected VN's

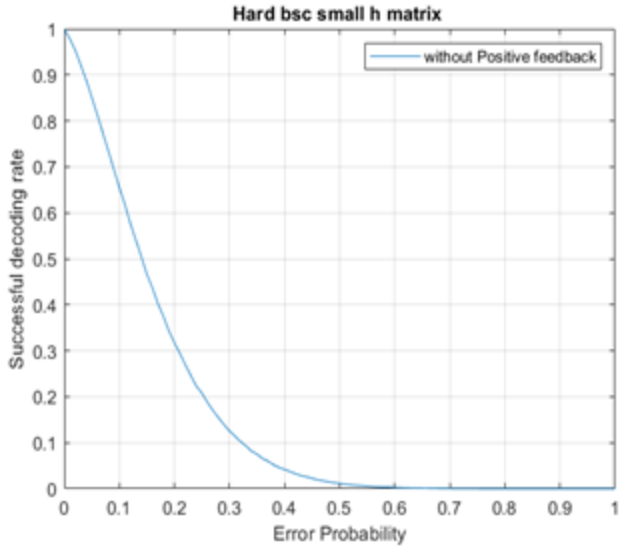
c_sent are updated values after SPC decoding

variable node contains values at each VNs received from connected CNs

Decoded message is after Majority voting at each VN



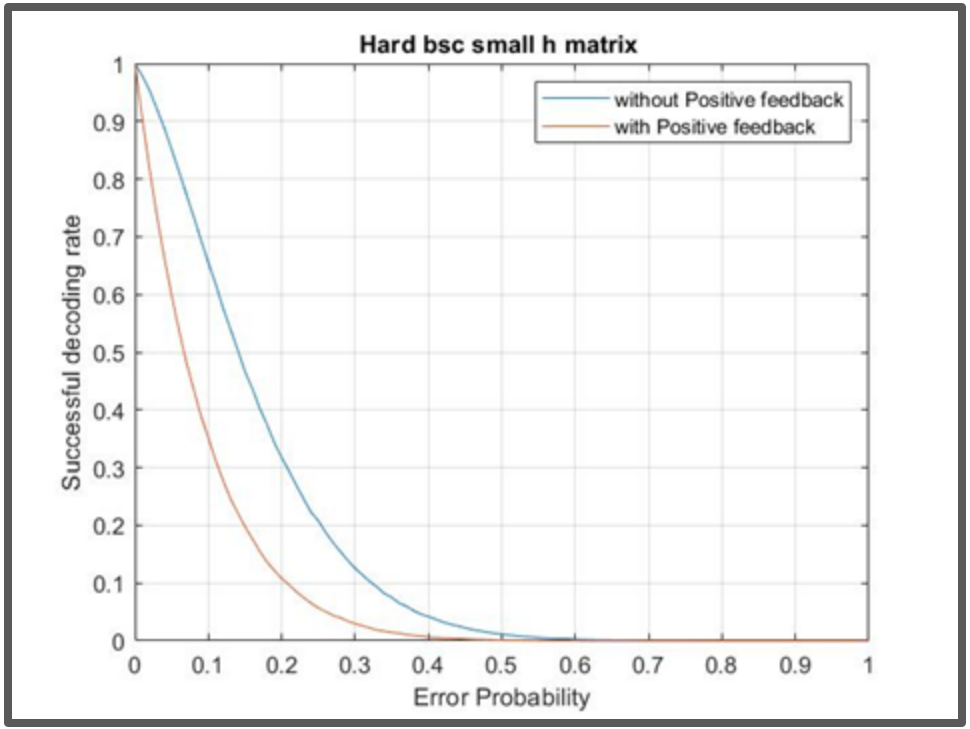
For small H matrix (9,12)



This graph shows Success Rate Vs Error Probability(p) for 1 lakh Monte-Carlo Simulations

Bit flipping probability / BSC(p) greater than 0.5 or 0.6 shows that it is hard / almost impossible to recover the original bit message

In the next slide, we will see that why we have not considered the value of node itself (Why we have not considered positive feedback)

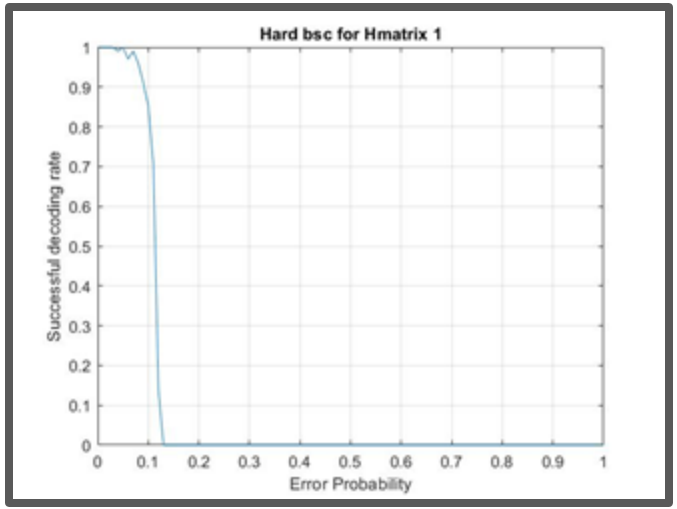


Positive feedback is like asking the bit about its own value, which is not desirable when we do SPC decoding, as the message generated at SPC / CN node for a particular VN depends only on the values received from other members of the SPC

Conclusion : In both VN → CN and CN → VN message passing, the message sent to a node will never depend on the message received from that node.

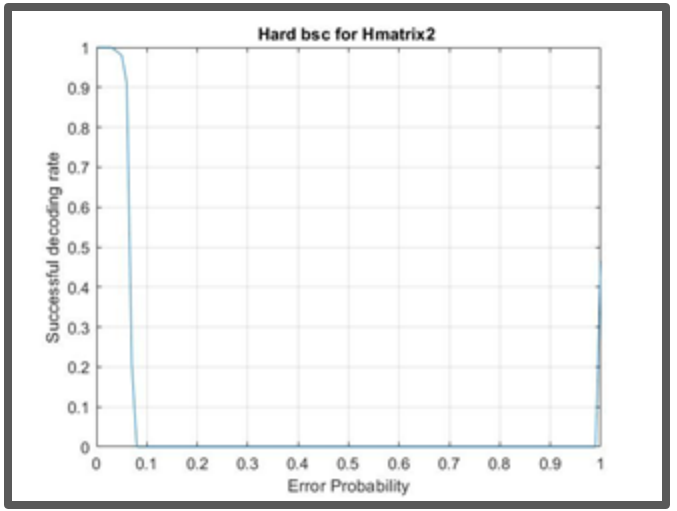
Here the graph clearly describes how the performance of the decoder gets better when we remove the positive feedback from the decoder.

Hmatrix 1 (3972, 5056)

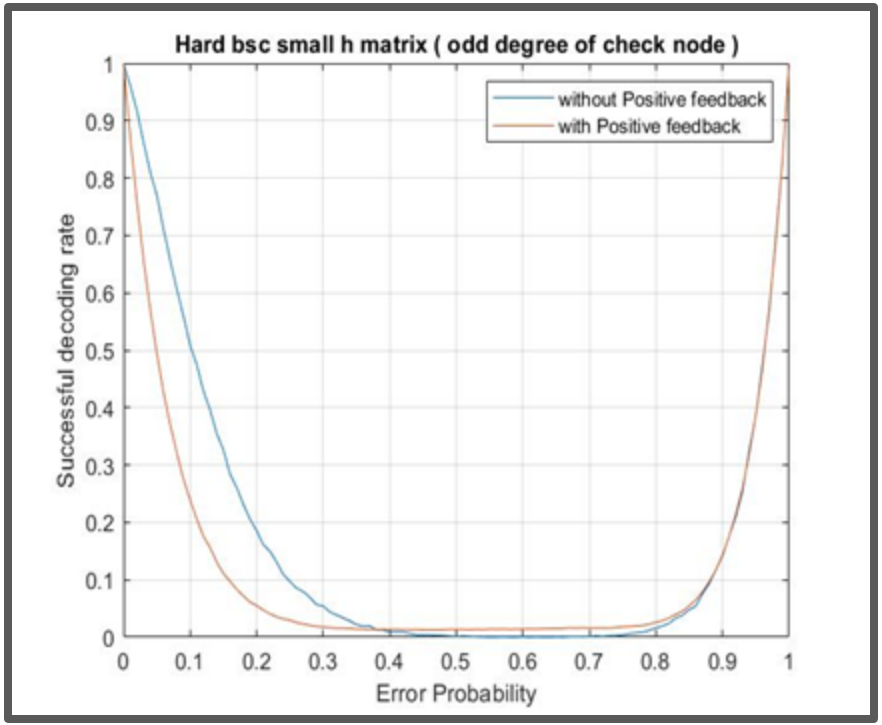


- Here, we notice a quick downfall in the success rate of the decoder at earlier/lesser error probability (p~0.13), as compared to the BEC decoder which decodes at erasure probabilities as high as 0.6.

Hmatrix 2 (3000, 5000)



- Here, for one more time, we notice an increment in success rate for Higher error probability(p) for an odd degree of Check node i.e. 5, in the above H matrix.



We can observe the increment in success rate again at higher error probability which makes the shape of graph parabolic.

Reason : As we have odd degree of check node, at every CN while SPC decoding we have remaining even number of nodes for generating the message, so when the error probability(p) is large(>0.8) i.e. when majority of the bits are flipped, the SPC/even parity check will satisfy for majority of CNs.

Again, we can see result of deoder with positive feedback is still less as compared to without positive feedback.

Soft BSC

Decoder

Algorithm for Decoding

Soft BSC:

- In the 0^{th} iteration, we will load the VNs in the array with the probability values of the corresponding received bits.
- Then we will iterate through the CNs, update all the VNs connected to it with SPC decoding of likelihoods of the remaining bits connected to that check node.
- Then we will return the decoded likelihood to all the respective variable nodes.
- We will perform a majority vote decoding at each VN based on the likelihoods received from each CN connected to it & its value we received over the BSC(p) channel and thereby updating the values at each VN with the bit decision obtained by majority voting.
- If there is no update in e^{th} codeword and e^{th+1} codeword, then we will break the loop. Otherwise in next iteration we will again load the check nodes with updated values of variable nodes.
- After breaking of loop, we will check if the transmitted and received codeword are same or not, If both are same, then we will consider that the codeword is correctly decoded and the success count increases.

Snippet representing a short implementation of above discussed Soft BSC algorithm

Output shows Algorithm:

Noise added Received message/ probabilities

c_received contains probabilities at CN's received from connected VN's

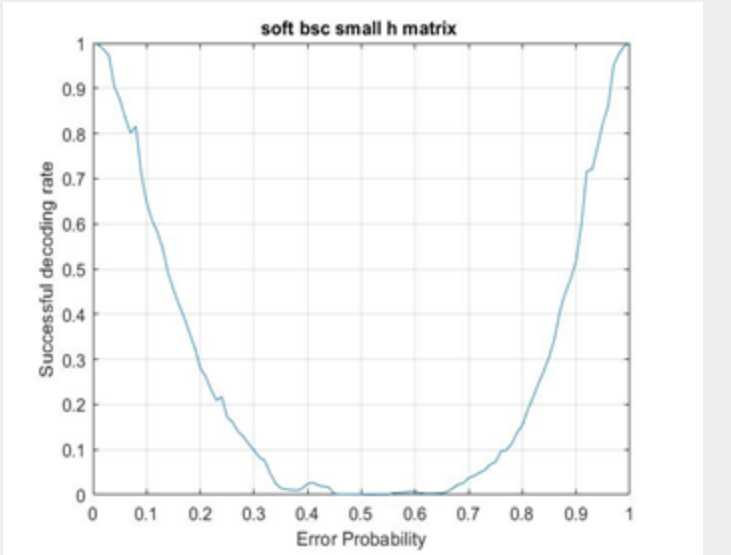
c_sent are updated likelihoods after SPC decoding

variable node contains likelihoods at each VNs received from connected CNs

Decoded message is after likelihood analysis at each VN

```
Codeword sent with Noise:       0.4 0.4 0.6 0.6 0.6 0.6 0.6 0.4 0.6 0.6 0.6 0.6

c_received  :
0.4 0.6 0.4 0.6
0.4 0.6 0.6 0.6
0.4 0.6 0.6 0.6
0.6 0.6 0.6 0.6
0.6 0.6 0.4 0.6
0.4 0.6 0.6 0.6
0.4 0.6 0.6 0.6
0.4 0.6 0.4 0.6
0.6 0.6 0.6 0.6

c_sent :
0.984127 1.01613 0.984127 1.01613
1.01613 0.984127 0.984127 0.984127
1.01613 0.984127 0.984127 0.984127
1.01613 1.01613 1.01613 1.01613
0.984127 0.984127 1.01613 0.984127
1.01613 0.984127 0.984127 0.984127
1.01613 0.984127 0.984127 0.984127
0.984127 1.01613 0.984127 1.01613
1.01613 1.01613 1.01613 1.01613

variable node :
0.984127 1.01613 1.01613
1.01613 1.01613 0.984127
1.01613 0.984127 1.01613
0.984127 0.984127 1.01613
0.984127 0.984127 0.984127
1.01613 1.01613 1.01613
0.984127 0.984127 0.984127
0.984127 1.01613 0.984127
0.984127 0.984127 1.01613
1.01613 0.984127 1.01613
0.984127 1.01613 0.984127
1.01613 0.984127 1.01613

decode :
0.677419 0.677419 1.52419 1.47619 1.4297 1.57376 1.4297 0.656085 1.47619 1.52419 1.47619 1.52419
c_received  :
0.403846 0.611463 0.396166 0.603834
0.403846 0.596154 0.588426 0.596154
0.403846 0.588426 0.588426 0.596154
0.603834 0.611463 0.596154 0.603834
0.603834 0.588426 0.396166 0.603834
```

For small H matrix (9,12)

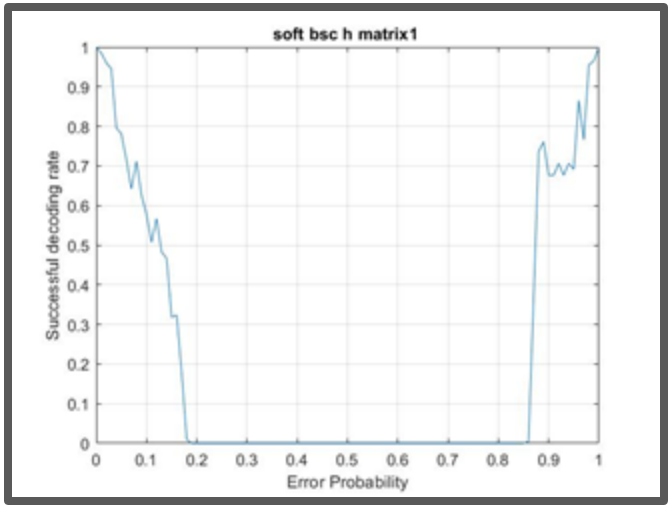


Soft decision decoding is not blind So our decoder also knows bit flip probability(p of BSC) which allows the decoder to determine how many bits are flipped in the received message

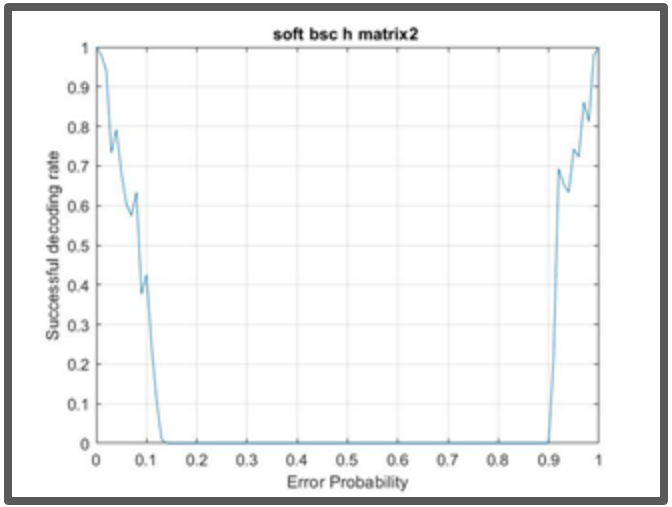
For any message that has higher error probability(>0.7), decoder can consider that message as highly flipped message at the time of decoding

So overall considering ideal condition, message with bit flip probability 0.3 and 0.7 are treated same

H matrix 1



H matrix 2



- Again Success rate falls at very low Pe But the thing is we got quite better rate than Hard.

Algorithm

Convergence

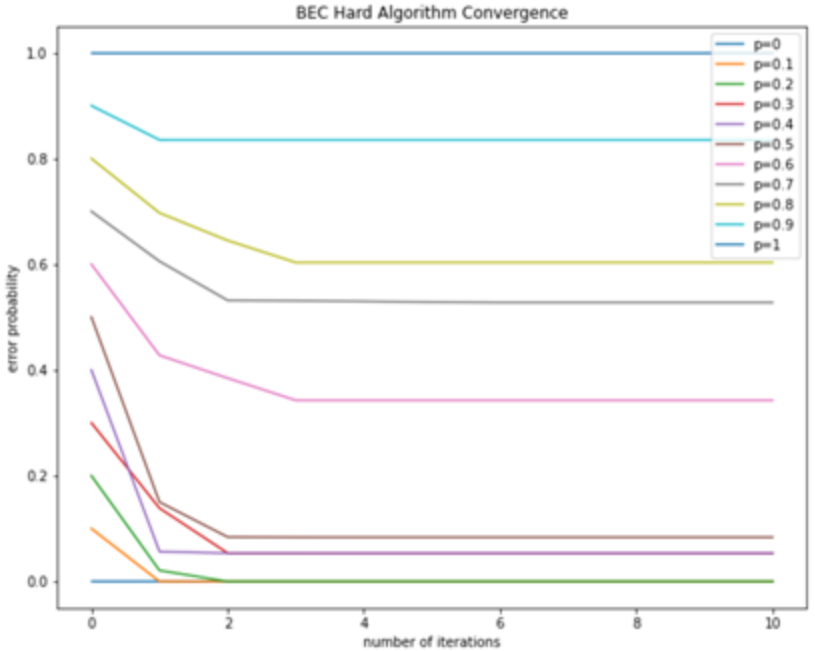
Convergence graph for BEC

- Convergence graph is the graph between probability of error and the number of iterations .

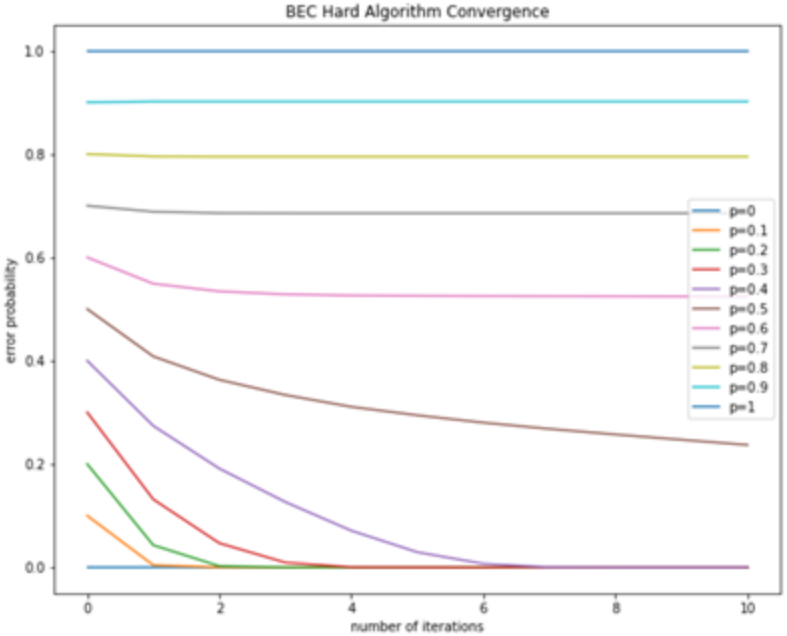
- As iterations increases, number of erasure decreases which is obvious. So, graph will be asymptotic to to 0.

- But when probability of erasure is greater than 0.6 then graph will not be asymptotic to 0 but it is asymptotic to some constant value.

Convergence graph for BEC Hard

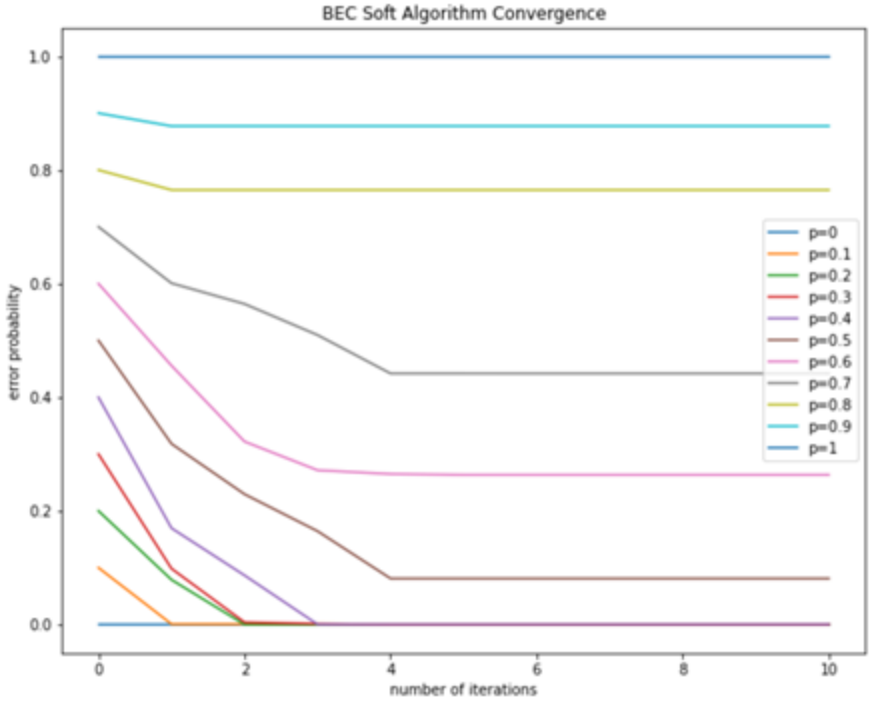


(9,12)
dc= 4 , dv = 3

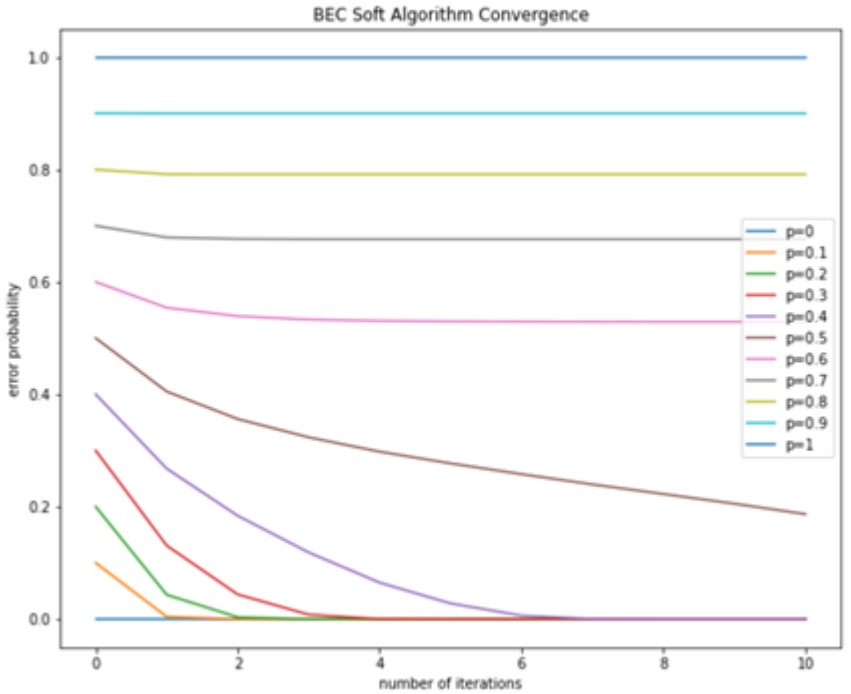


(3000,5000)
dc =5 , dv =3

Convergence graph for BEC Soft



(9,12)
dc= 4 , dv = 3

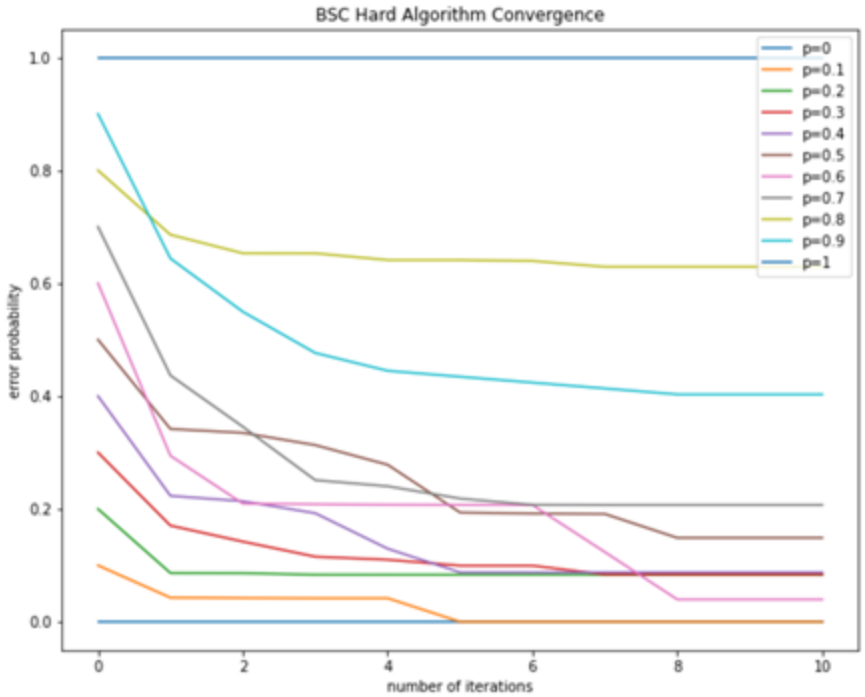


(3000,5000)
dc =5 , dv =3

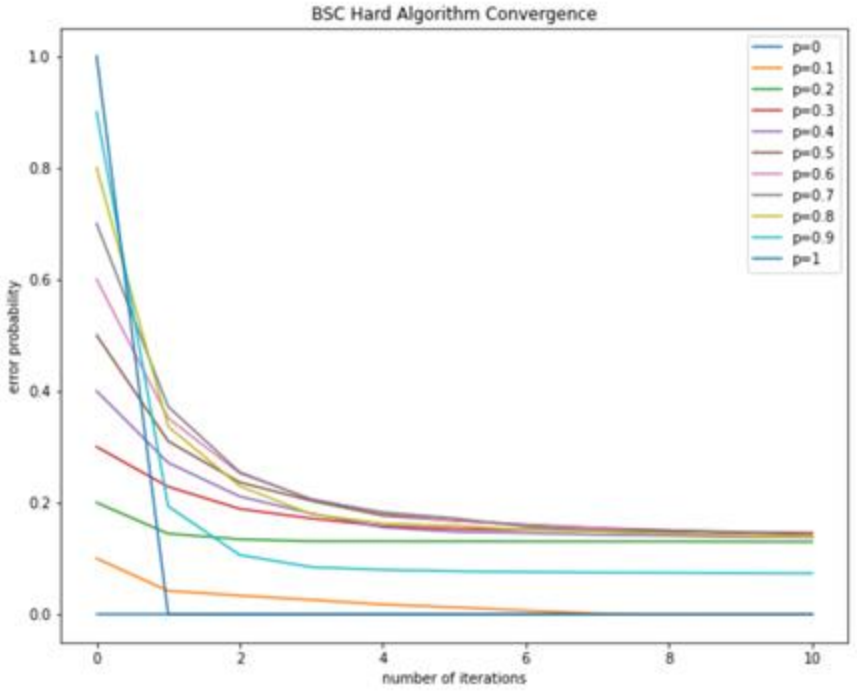
Convergence graph for BSC

- In the BSC channel,as iterations increases, number of flipped bits decreases.
- After some iterations, the graph will be asymptomatic to any constant line.
- In BSC,for any value of bit flipping probability the graph will not approaches to 0 but it will approach any constant value.

Convergence graph for BSC Hard

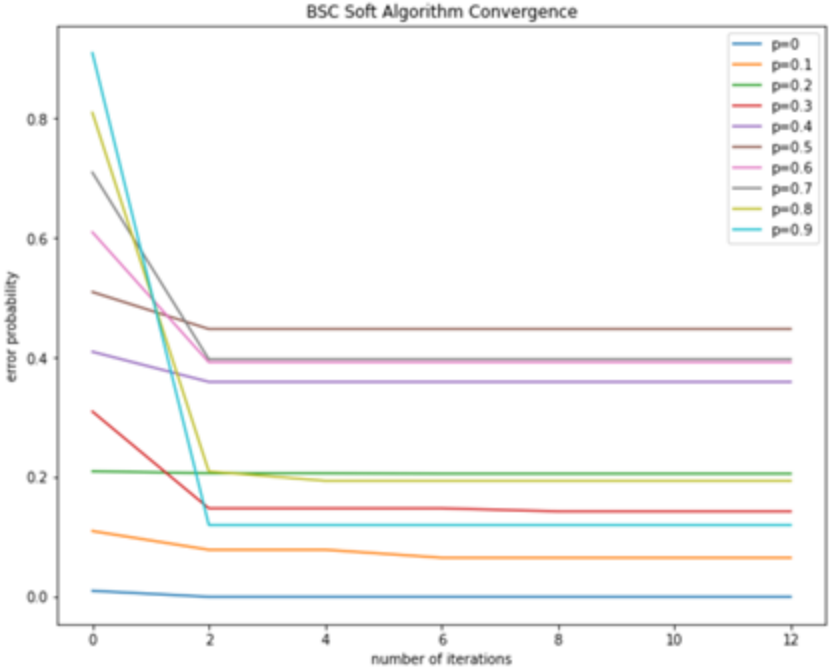


(9,12)
dc= 4 , dv = 3

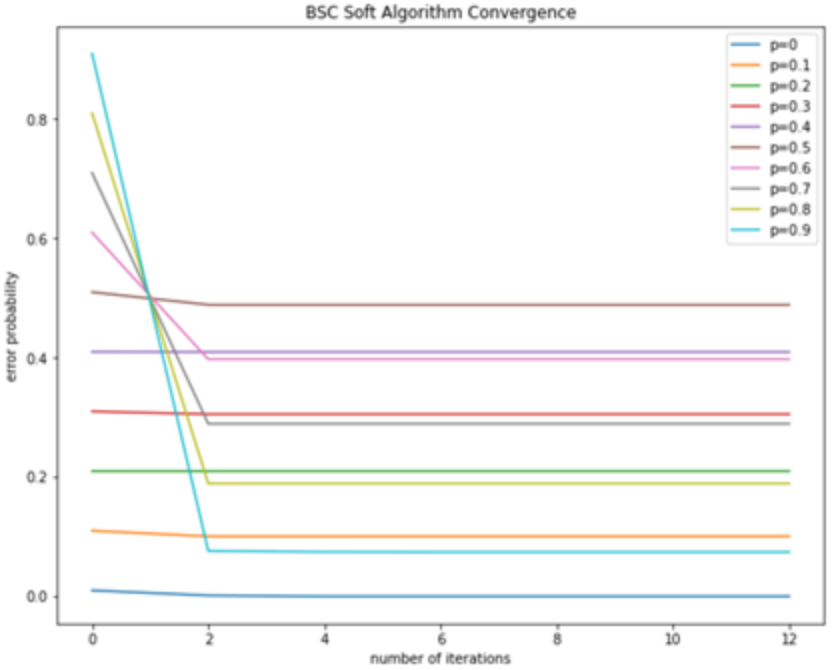


(3000,5000)
dc =5 , dv =3

Convergence graph for BSC Soft

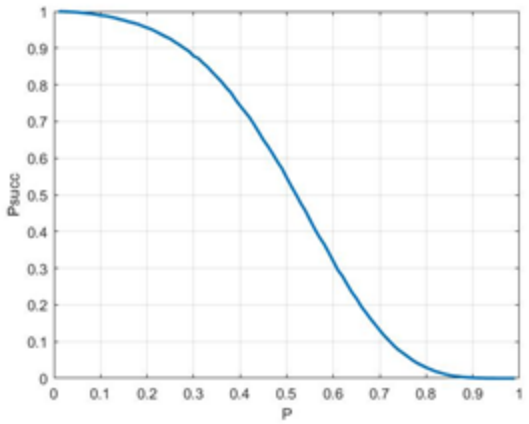


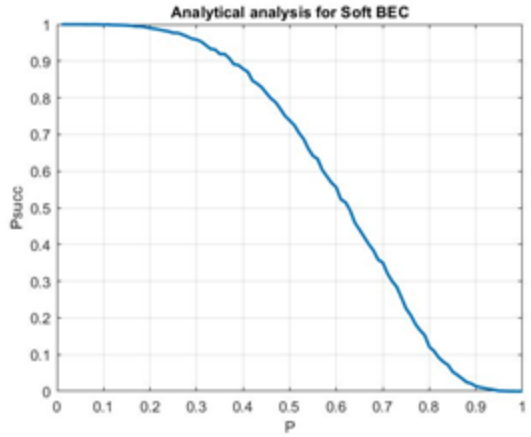
(9,12)
dc= 4 , dv = 3



(3000,5000)
dc =5 , dv =3

Comparison between Monte-Carlo and Analytical analysis for Soft BEC





Monte-Carlo

Analytical analysis

We have done analytical process by density evaluation method , which confirms our implemented / practical results.

Displacement method (Advancement over previous methods)

How to make more advantage of received reliability at receiver ??????

eg. bit 1 =0 bit 2 =1

0--|----|----------1
--0.1__0.2----------

=> impact of 0.1 > 0.2 and towards 0

0----------|-----|--1
----------0.7___0.9--

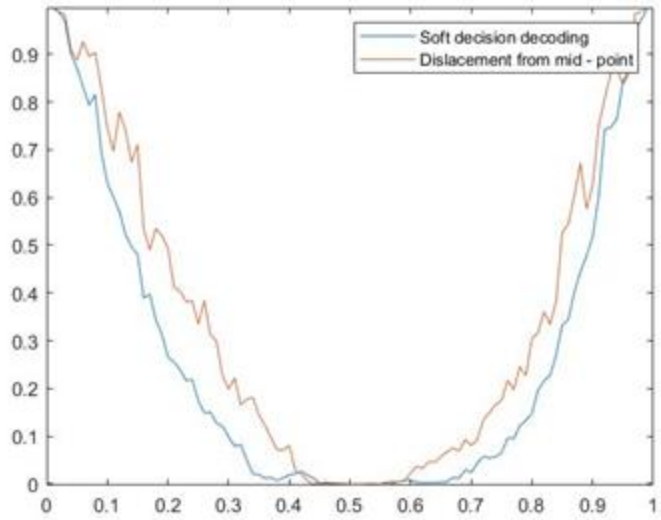
=> impact of 0.9 >0.7 and towards 1

So the method is

step-1 --> |D|= distance from mid point
observation : D greater more than impact or reliability more

step-2 --> D = Displacement or Sign of distance
observation : sign gives us idea of bit's impact is towards which given bit

Receiver makes more advantage of received reliability probability than soft in this ...



Summary

- Overall, the project is acceptably hard but by helping each other out and solving each other’s difficulties, we completed the project.

References

- ‘LDPC codes-a brief tutorial’ -paper by Bernhard M.J. Leiner
- Slides by Prof. Yash Vasavada

Thank You

By Group 1 Team 2

Thanks to our TAs, Aarushi Dhami & Abhishek Pandya

Mention : Prof. Yash Vasavada