CT-216 Introduction TO Communication Systems

LDPC Decoding For 5G NR

Lab group: 2
Project group: 7

Assigned by: Professor Yash Vasavada
Mentor TA : Vivek Patel

Honour Code

- The work we are presenting is our own work and we have not copied the work (Matlab code, results, etc) that someone else has done.

- Concepts, understanding and insights we will be describing are our own.

- Wherever we have relied on an existing work that is not our own, we have provided a proper reference citation. We make this pledge truthfully, knowing that violation of this solemn pledge can carry grave consequences.

Contents

INTRODUCTION

Introduction to LDPC Codes

What are LDPC Codes?

- Low-Density Parity-Check (LDPC) codes are a class of linear error-correcting codes defined by a sparse parity-check matrix, typically denoted by H.
- First introduced by Robert Gallager in 1962, they remained unused for decades due to computational complexity but were rediscovered by David MacKay in the 1990s, assisting in their practical use in modern systems.

Key Characteristics

- The matrix H has few 1s per row and column, ensuring efficient iterative decoding.
- LDPC codes offer near-capacity performance, approaching the Shannon limit in terms of error correction.
- Encoding and decoding can be efficiently implemented using Tanner graphs, a bipartite graphical representation of the code structure.

Why LDPC?

- LDPC codes are highly favored in communication systems due to:
  - High error correction performance
  - Scalability to long block lengths
  - Compatibility with iterative message-passing algorithms.(covered in later part of ppt.)

LDPC Structure & Applications

Structural Overview

- LDPC codes use an $m \times n$ parity-check matrix (H):
  - Each column has w_n ones (variable node degree).
  - Each row has w_r ones (check node degree).
  - These degrees are small compared to m and n (i.e., sparse).
  - Total ones in H: $w_n \times n = w_r \times m$
- Types:
  - Regular LDPC: Uniform w_n and w_r.
  - Irregular LDPC: Non-uniform degrees, optimized for performance.

Real-World Applications

- 5G NR (New Radio): Standardized for error correction in control and data channels.
- Wi-Fi 6/7, Satellite Communication, Deep-Space Telemetry
- Optical Storage Systems and Advanced Data Storage

LDPC Communication System

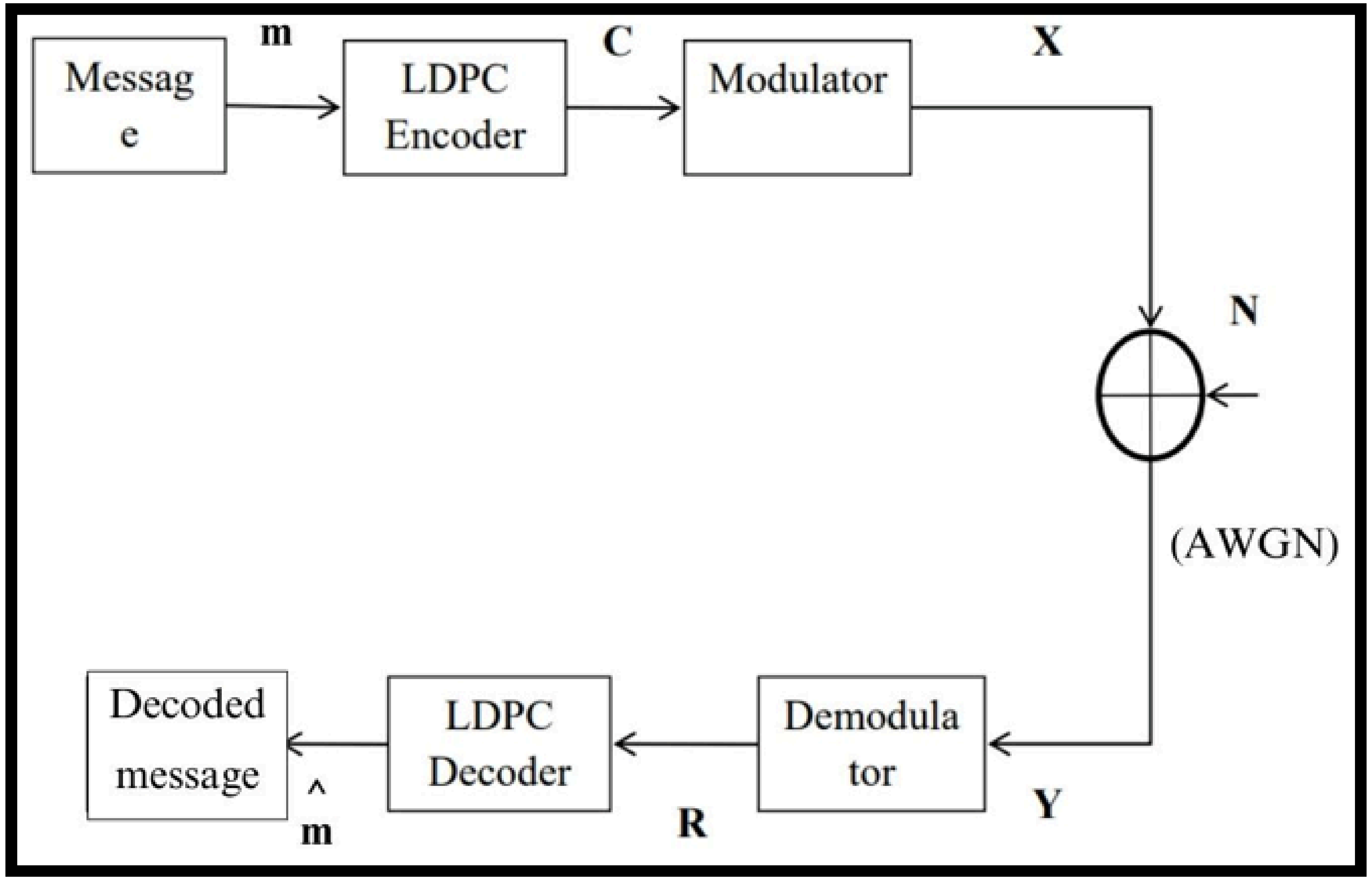


Fig. 1

Fig. 1 illustrates the end-to-end communication system using LDPC codes.
The message **m** is encoded using an LDPC encoder, modulated, and transmitted over an Additive White Gaussian Noise **(AWGN)** channel.
The received signal is demodulated and decoded to recover the original message **m̂.**
LDPC decoding helps correct errors introduced by channel noise, enabling reliable data recovery even in noisy environments.

Base Graphs in LDPC (5G NR)

- 5G NR employs two predefined base graphs (BGs) as compact representations of the full parity-check matrix.
- These base graphs are expanded using an expansion factor (Z) to generate the full LDPC matrix used in encoding and decoding.
  - Base graph entries consist of values from the set {–1, 0, 1, ..., Z–1}:
  - –1 indicates no connection in the graph. [All 0 matrix]
  - The choice of base graph (BG1 or BG2) is determined by the code rate and the payload size.

Base Graph	Size	Use Case	Max Message Bits
BG1	46Z × 68Z	Large transport blocks	22Z
BG2	42Z × 52Z	Small transport blocks	10Z

Fig. 2 Shows the two types of base graph used in LDPC for 5G NR

Types of base graph used in LDPC for 5G NR

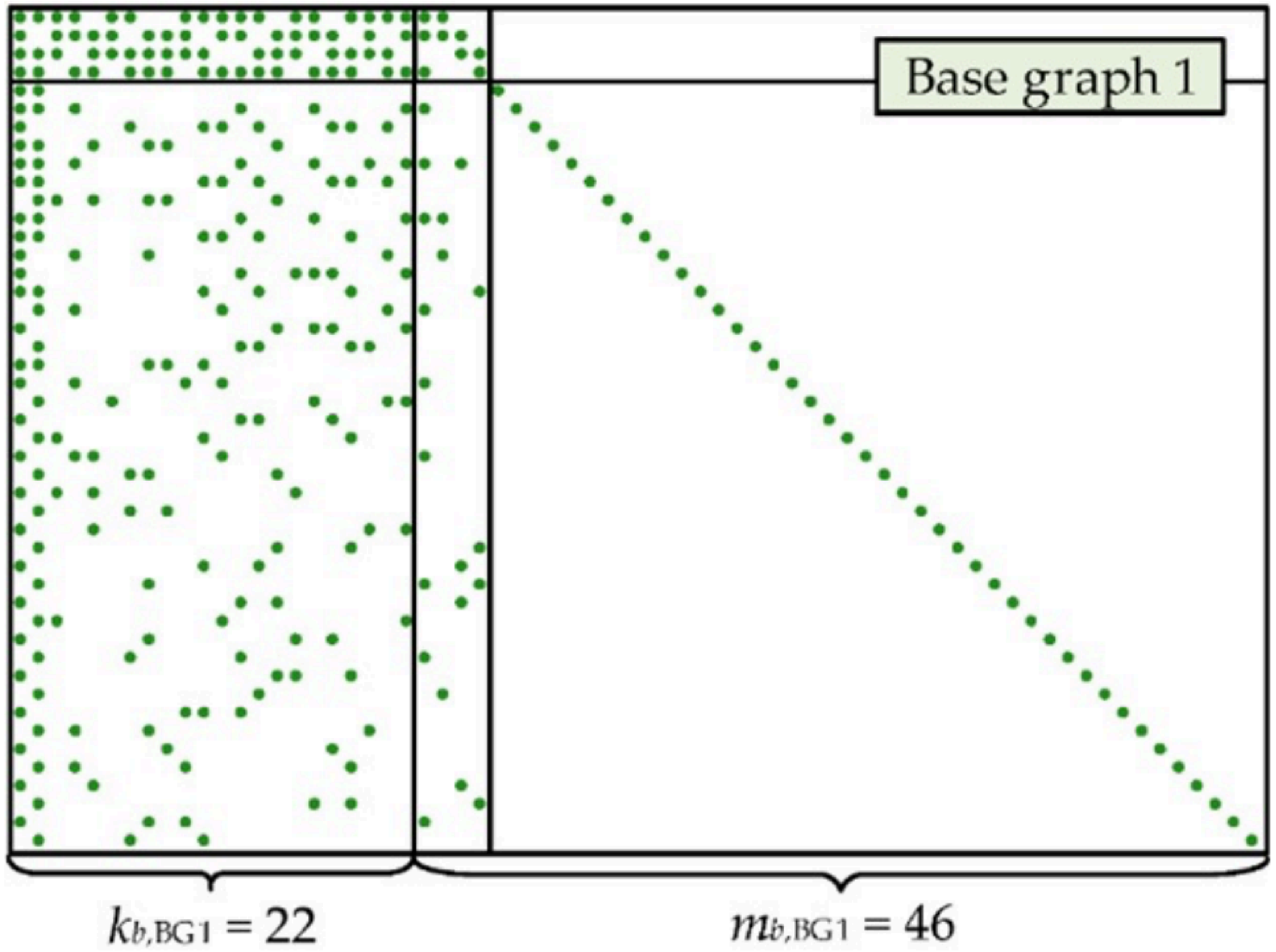


Fig. 3 : Represents BG1 (46Zx68Z)

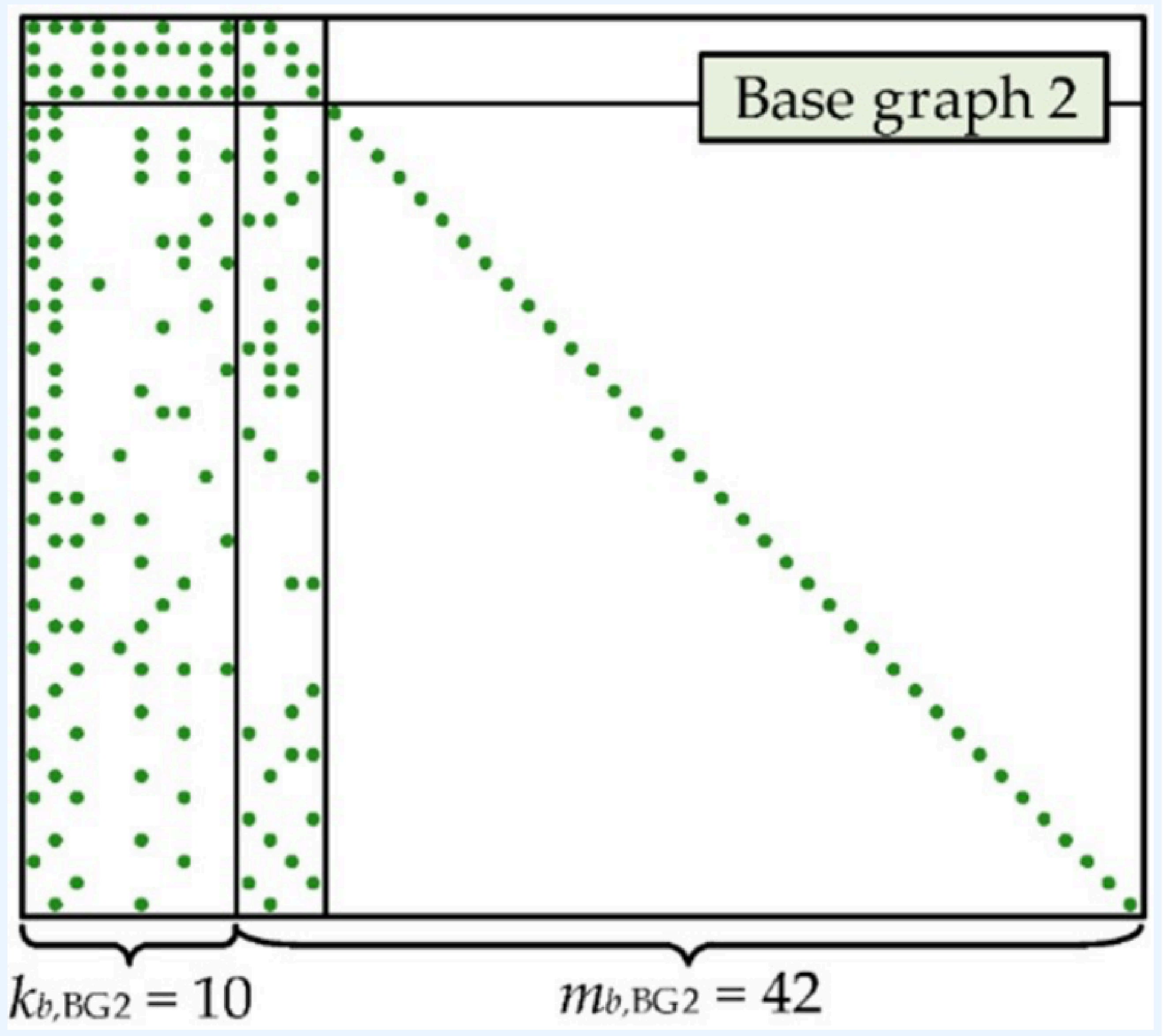


Fig. 4 : Represents BG2 (42Zx52Z)

[redacted]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	307	19	50	369	-1	181	216	-1	-1	317	288	109	17	357	-1	215	106	-1	242	180	330	346	1	0	-1	-1
2	76	-1	76	73	288	144	-1	331	331	178	-1	295	342	-1	217	99	354	114	-1	331	-1	112	0	0	0	-1
3	205	250	328	-1	332	256	161	267	160	63	129	-1	-1	200	88	53	-1	131	240	205	13	-1	-1	-1	0	0
4	276	87	-1	0	275	-1	199	153	56	-1	132	305	231	341	212	-1	304	300	271	-1	39	357	1	-1	-1	0
5	332	181	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	195	14	-1	115	-1	-1	-1	-1	-1	-1	-1	-1	166	-1	-1	-1	241	-1	-1	-1	-1	51	157	-1	-1	-1
7	278	-1	-1	-1	-1	-1	257	-1	-1	-1	1	351	-1	92	-1	-1	-1	253	18	-1	225	-1	-1	-1	-1	-1
8	9	62	-1	-1	316	-1	-1	333	290	-1	-1	-1	-1	-1	114	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
9	307	179	-1	165	-1	-1	-1	-1	-1	-1	-1	-1	18	-1	-1	-1	39	-1	-1	224	-1	368	67	-1	170	-1
10	366	232	-1	-1	-1	-1	-1	-1	-1	-1	321	133	-1	57	-1	-1	-1	303	63	-1	82	-1	-1	-1	-1	-1
11	-1	101	339	-1	274	-1	-1	111	383	-1	-1	-1	-1	-1	354	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
12	48	102	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	8	-1	-1	-1	47	-1	-1	-1	-1	188	334	115	-1	-1
13	77	186	-1	-1	-1	-1	-1	-1	-1	-1	174	232	-1	50	-1	-1	-1	-1	74	-1	-1	-1	-1	-1	-1	-1
14	313	-1	-1	177	-1	-1	-1	266	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	115	-1	-1	370	-1	-1
15	142	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	248	-1	-1	137	89	347	-1	-1	-1	12	-1	-1	-1	-1
16	241	2	-1	-1	-1	-1	-1	-1	-1	-1	210	-1	-1	318	-1	-1	-1	-1	55	-1	-1	-1	-1	-1	-1	269
17	13	-1	-1	338	-1	-1	-1	-1	-1	-1	-1	57	-1	-1	-1	-1	-1	-1	-1	-1	289	-1	57	-1	-1	-1
18	260	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	303	-1	81	358	-1	-1	-1	375	-1	-1	-1	-1
19	130	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	163	280	-1	-1	-1	-1	132	4	-1	-1	-1	-1	-1	-1
20	145	213	-1	-1	-1	-1	-1	344	242	-1	197	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
21	187	-1	-1	206	-1	-1	-1	-1	-1	264	-1	341	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	59	-1	-1	-1
22	205	-1	-1	-1	-1	102	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	328	-1	-1	-1	213	97	-1	-1	-1	-1
23	30	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	11	233	-1	-1	-1	22	-1	-1	-1	-1	-1	-1	-1	-1
24	-1	24	89	-1	-1	-1	-1	-1	-1	-1	61	-1	-1	-1	-1	-1	-1	-1	27	-1	-1	-1	-1	-1	-1	-1
25	298	-1	-1	158	235	-1	-1	-1	-1	-1	-1	339	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	234	-1	-1	-1
26	-1	72	-1	-1	-1	-1	17	383	-1	-1	-1	-1	-1	-1	312	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
27	71	-1	81	-1	76	-1	-1	-1	-1	-1	-1	-1	-1	-1		136	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
28	-1	194	-1	-1	-1	-1	194	-1	101	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
29	222	-1	-1	-1	19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	244	-1	274	-1	-1	-1	-1
30	-1	252	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	5	-1	-1	-1	147	-1	-1	-1	-1	-1	-1	78
31	159	-1	-1	-1	-1	-1	-1	-1	-1	-1	229	-1	-1	260	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	90	-1
32	-1	100	-1	-1	-1	-1	-1	215	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	258	-1	-1	256
33	102	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	201	-1	175	-1	-1	-1	-1	-1	-1	-1	-1	-1	287	-1
34	-1	323	8	-1	-1	-1	-1	-1	-1	-1	-1	361	-1	-1	-1	-1	-1	-1	-1	-1	-1	105	-1	-1	-1	-1
35	230	-1	-1	-1	-1	-1	-1	148	-1	-1	-1	-1	-1	-1	-1	202	-1	312	-1	-1	-1	-1	-1	-1	-1	-1
36	-1	320	-1	-1	-1	-1	335	-1	-1	-1	-1	-1	2	-1	-1	-1	-1	-1	-1	-1	-1	-1	266	-1	-1	-1
37	210	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	313	297	-1	-1	21	-1	-1	-1	-1	-1	-1	-1
38	-1	269	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	82	-1	-1	-1	-1	-1	-1	-1	-1	-1	115	-1	-1
39	185	-1	-1	-1	-1	-1	-1	-1	-1	177	289	-1	214	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
40	-1	258	-1	93	-1	-1	-1	346	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	297	-1	-1	-1	-1	-1	-1
41	175	-1	-1	-1	-1	-1	-1	-1	37	-1	-1	-1	-1	-1	-1	-1	-1	312	-1	-1	-1	-1	-1	-1	-1	-1
42	-1	52	-1	314	-1	-1	-1	-1	-1	139	-1	-1	-1	-1	-1	-1	-1	-1	288	-1	-1	-1	-1	-1	-1	-1
43	113	-1	-1	-1	14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	218	-1
44	-1	113	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	132		114	-1	-1	-1	-1	-1	-1	168
45	80	-1	-1	-1	-1	-1	-1	78	-1	163	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	274	-1	-1	-1
46	-1	135	-1	-1	-1	-1	149	-1	-1	-1	15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
11	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
17	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
21	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
22	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
23	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
24	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
25	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
26	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
27	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
28	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
29	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
30	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
31	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
32	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
33	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
34	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
35	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
36	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
37	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
38	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
39	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
40	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
41	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1
42	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1
43	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1
44	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1
45	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1
46	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Fig.Example of a BG1(46Zx68Z)

SOURCE:I. J. Computer Network and Information Security, 2023, 4, 1-12 Published Online on August 8, 2023 by MECS Press (http://www.mecs-press.org/)

Characteristics of Base graph

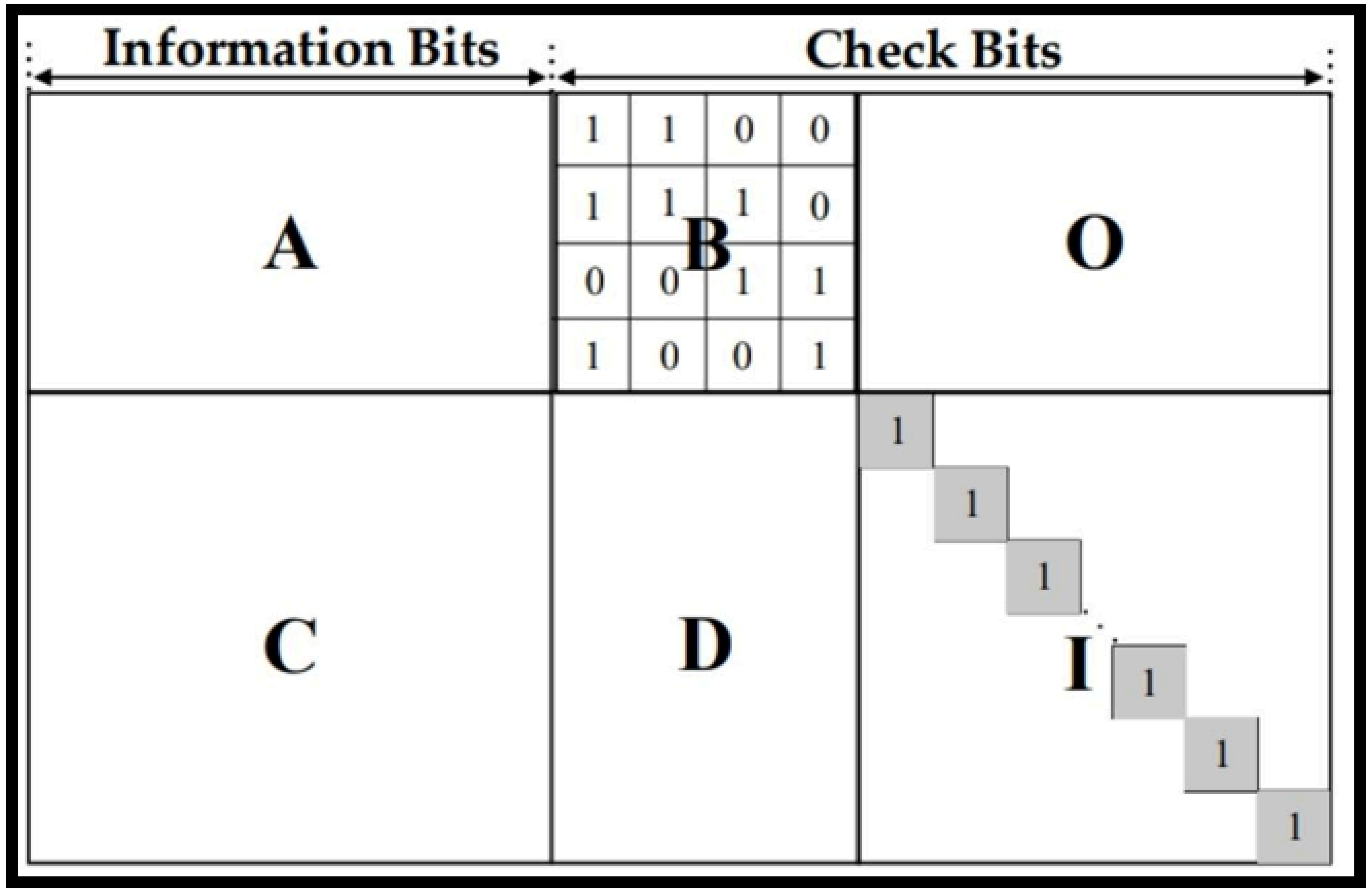


Fig. 5 : Different submatrix present in Base Graph

- **A** represents a high-density block, contributing to dense connectivity.
- **B** is a double diagonal matrix, ensuring structured redundancy.
- **O** is a zero matrix, indicating no connections.
- **C** and **D** are low-density blocks, used for sparse connections.
- **I** is an identity matrix, preserving original data alignment.
- **For BG1**, block sizes are:

A: 4×22, **B**: 4×4, **O**: 4×42, **C**: 42×22, **D**: 42×4, **I**: 42×42

- **For BG2**, block sizes are:

A: 4×10, **B**: 4×4, **O**: 4×38, C: 38×10, **D:** 38×4, **I**: 38×38

Characteristics of Base graph

To construct the parity-check matrix H from a base matrix B, we use the **lifting size** or **expansion factor** **z**, which transforms the small base matrix into a larger sparse matrix suitable for LDPC decoding.

Matrix Substitution Rules:

Let B(i,j) be the element at row i, column j in matrix B. Replace each element as follows:

- $B_{i,j} = -1$:
  → Replace it with a $z \times z$ zero matrix.
- $B_{i,j} = 0$:
  → Replace with a $z \times z$ identity matrix.
- $B_{i,j} \in \{1,2,\ldots,z-1\}$:
  → Replace with a $z \times z$ identity matrix that is circularly right-shifted by B(i,j) positions.

Resulting Matrix Dimensions

- The final matrix H has dimensions:

$$(m \cdot z) \times (n \cdot z)$$

Construction Of H from Base matrix

$$B = \begin{bmatrix} 1 & -1 & 3 & 1 & 0 & -1 \\ 2 & 0 & -1 & 0 & 0 & 0 \\ -1 & 4 & 2 & 1 & -1 & 0 \end{bmatrix}$$

Expansion factor: 5

$$H = \left[\begin{array}{ccccc|ccccc|ccccc|ccccc|ccccc|ccccc}
0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
\hline
0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0\\
0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
\hline
0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{array}\right]$$

Fig. 6: H is constructed from base matrix with expansion factor Z=5

The parity-check matrix H is constructed from a base matrix B[3X5] using an expansion factor (z=5).
Each entry in B is replaced with a z × z.
Submatrix based on the following rules:

- −1 → Replaced with a zero matrix.
- 0 → Replaced with an identity matrix.
- Positive integer k → Replaced with an identity matrix right-shifted by k positions.

In this example, with an expansion factor of 5, the 3×7 base matrix B is expanded into a 15×35 sparse matrix H.

Puncturing of H-Matrix

- In LDPC codes, a puncturing matrix determines which bits of the codeword are punctured (not transmitted) and which are retained for transmission.

- It is usually a binary matrix, where:
  - Each row represents a puncturing pattern.
  - Each column corresponds to a bit position in the codeword.

Uses:

- Puncturing provides a flexible way to balance code rate and error-correction performance.

- It allows LDPC codes to be adapted for various communication systems with different reliability and bandwidth requirements.

Puncturing of H-Matrix

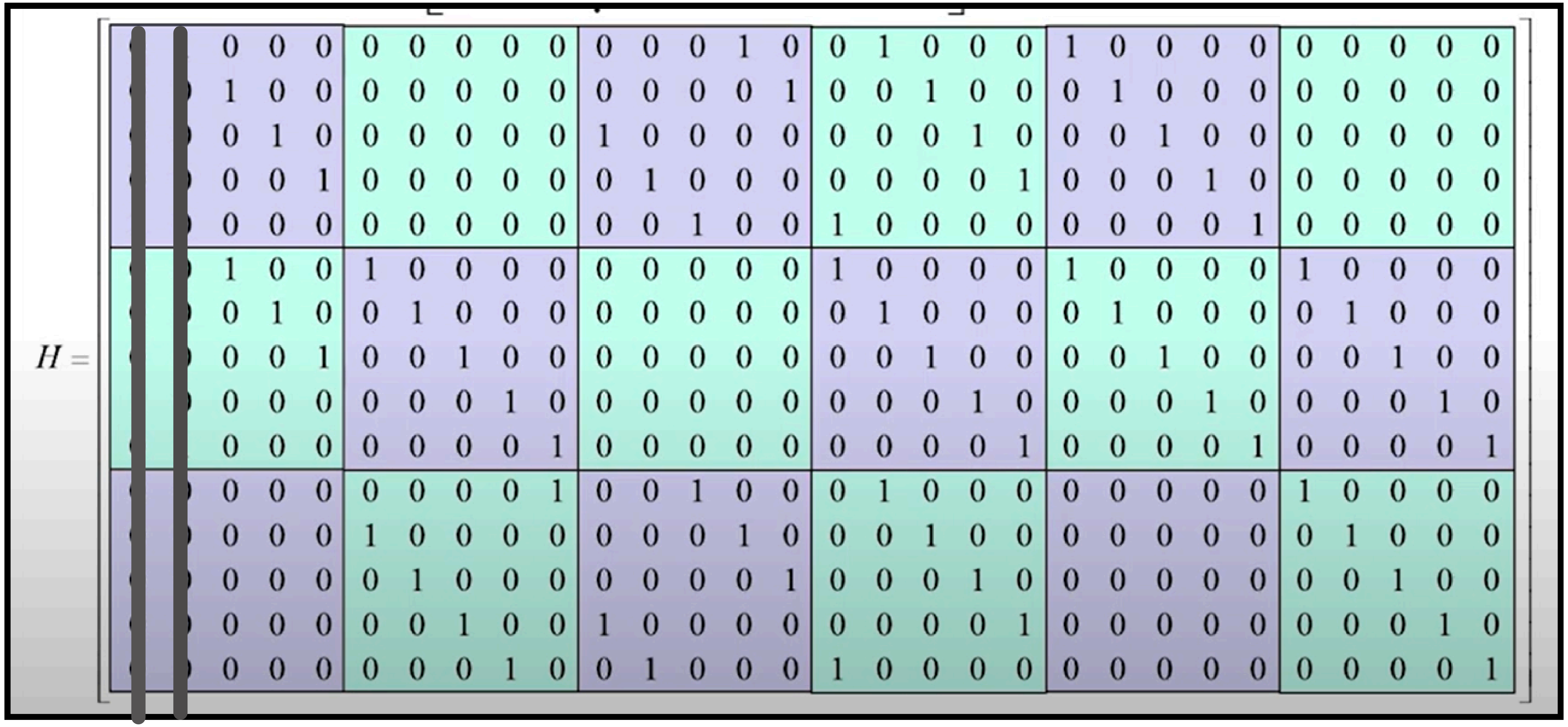
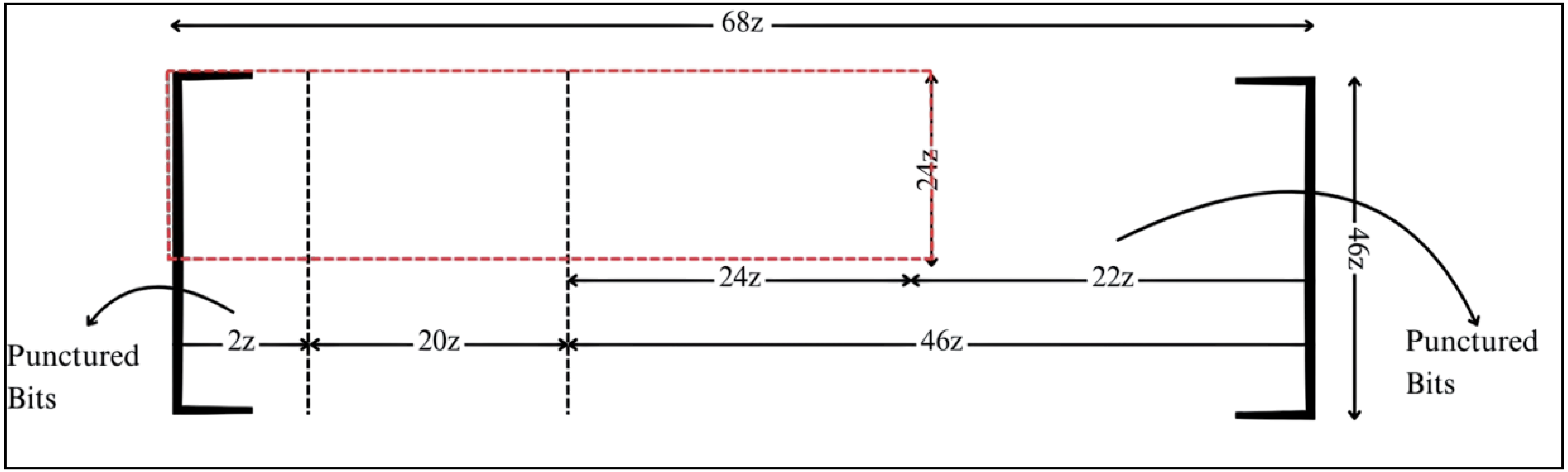


Fig. 7: First two column of H matrix are punctured.

Puncturing of H-Matrix

Toy Example

The red box represents the effective parity-check matrix of size **24z × 46z**, used for **encoding .**
 It includes **20z message bits** and **24z parity bits**, excluding the **punctured 2z bits** and any shortened bits. This submatrix defines the actual transmission structure after rate adjustment.



Rate Matching in LDPC Codes

Purpose :

- Rate matching in LDPC (Low-Density Parity-Check) codes is a technique used to adapt the number of transmitted bits to meet the desired code rate for a communication system, particularly in 5G NR (New Radio).
- This process ensures that despite using large parity-check matrices, the final codeword length aligns with system constraints, enabling flexible adaptation to varying channel conditions and throughput requirements.

Key Concepts :

- The base matrix of size $m \times n$ is expanded by a factor z, yielding a total codeword length of $n \cdot z$ bits.
- Out of these, the first 2z bits are typically punctured, leaving $(n-2) \cdot z$ usable bits.
- To achieve a target code rate of x/y, a subset of parity bits ($a \cdot z$ bits) is punctured from the available pool.
- The number of message bits is calculated as $(n - m) \cdot z$.

$$\frac{(n-m)\cdot z}{(n-2)\cdot z - a\cdot z} = \frac{x}{y}$$

Using this relation,we can determine the necessary amount of puncturing to match the desired rate.

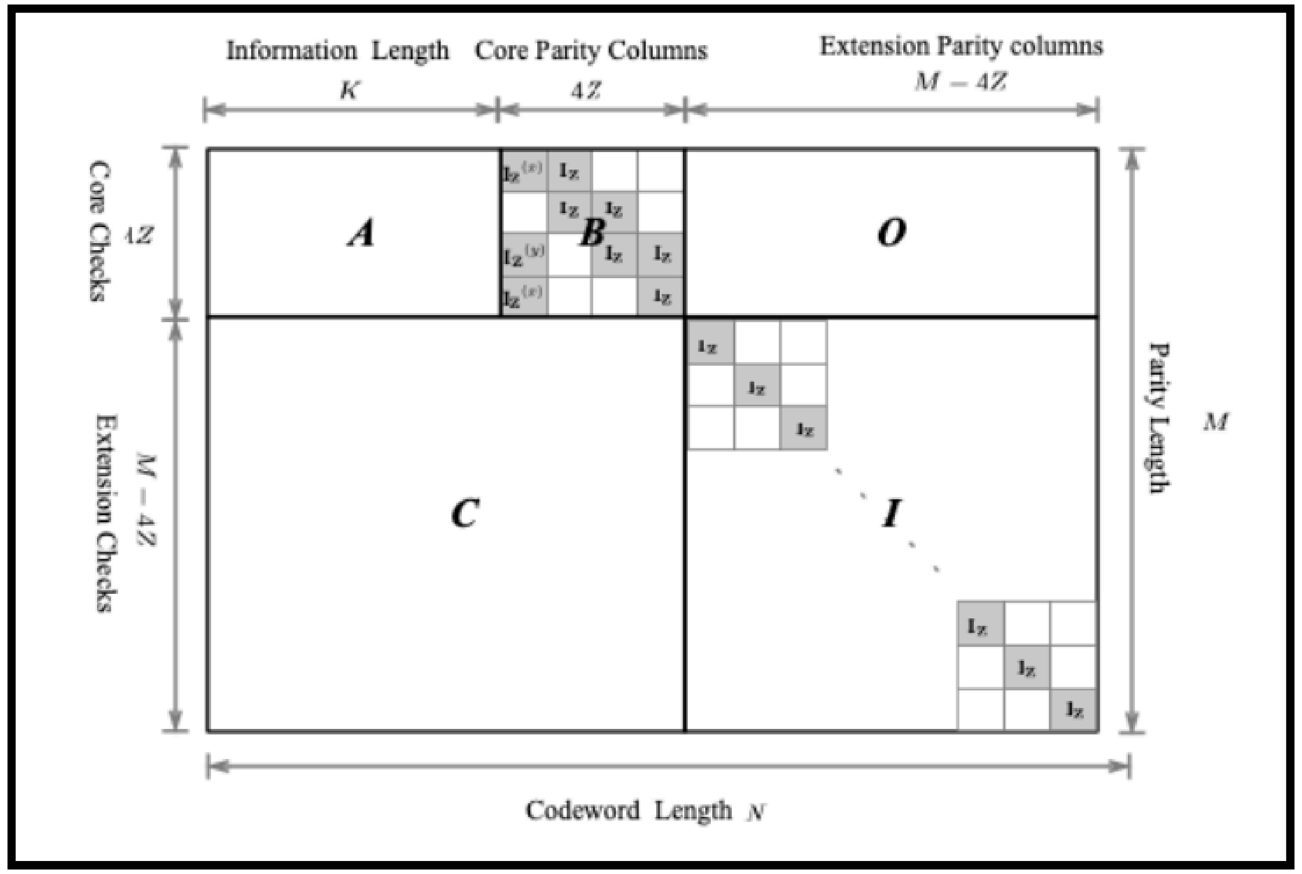
Encoding

Encoding

Encoding is to to compute Parity bits according the given message bits by solving the following checksum equation, where is the encoded bit sequence.

The structure of parity check matrix We can use the structure of parity check matrix of LDPC, is show in the picture beside c = [u’ ,p’]’

$$\mathbf{Hc} = \mathbf{0}, \quad \mathbf{H} \in \mathbb{B}^{M \times N}, \mathbf{c} \in \mathbb{B}^{N \times 1}$$



$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{O} \\ \mathbf{C}_1 & \mathbf{C}_2 & \mathbf{I} \end{bmatrix} \in \mathbb{B}^{M \times N}$$

$$\mathbf{A} \in \mathbb{B}^{4Z \times K} \quad \mathbf{B} \in \mathbb{B}^{4Z \times 4Z} \quad \mathbf{O} = O^{4Z \times M-4Z}$$

$$\mathbf{C} = [\mathbf{C}_1, \mathbf{C}_2] \in \mathbb{B}^{M-4Z \times K+4Z} \quad \mathbf{C}_1 \in \mathbb{B}^{M-4Z \times K} \quad \mathbf{C}_2 \in \mathbb{B}^{M-4Z \times 4Z} \quad \mathbf{I} = 1_{M-4Z}$$

$$\mathbf{c} = [\mathbf{u}^T, \mathbf{p}_c^T, \mathbf{p}_e^T]^T$$

$$\mathbf{u} \in \mathbb{B}^{K \times 1} \quad \mathbf{p}_a \in \mathbb{B}^{4Z \times 1} \quad \mathbf{p}_b \in \mathbb{B}^{M-4Z \times 1}$$

Encoding

$$\mathbf{B} = \begin{bmatrix} \mathbf{I}_Z^{(x)} & \mathbf{I}_Z & \mathbf{0}_Z & \mathbf{0}_Z \\ \mathbf{0}_Z & \mathbf{I}_Z & \mathbf{I}_Z & \mathbf{0}_Z \\ \mathbf{I}_Z^{(y)} & \mathbf{0}_Z & \mathbf{I}_Z & \mathbf{I}_Z \\ \mathbf{I}_Z^{(x)} & \mathbf{0}_Z & \mathbf{0}_Z & \mathbf{I}_Z \end{bmatrix}$$

t

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{a}_4 \end{bmatrix} \quad \mathbf{a}_i \in \mathbb{B}^{Z \times K}, \quad \mathbf{p}_c = \begin{bmatrix} \mathbf{p}_{\mathbf{c}1} \\ \mathbf{p}_{\mathbf{c}2} \\ \mathbf{p}_{\mathbf{a}3} \\ \mathbf{p}_{\mathbf{c}4} \end{bmatrix} \quad \mathbf{p}_{\mathbf{c}_i} \in \mathbb{B}^{Z \times 1},$$

e have

$$\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{a}_4 \end{bmatrix} \mathbf{u} + \begin{bmatrix} \mathbf{I}_Z^{(x)} & \mathbf{I}_Z & \mathbf{0}_Z & \mathbf{0}_Z \\ \mathbf{0}_Z & \mathbf{I}_Z & \mathbf{I}_Z & \mathbf{0}_Z \\ \mathbf{I}_Z^{(y)} & \mathbf{0}_Z & \mathbf{I}_Z & \mathbf{I}_Z \\ \mathbf{I}_Z^{(x)} & \mathbf{0}_Z & \mathbf{0}_Z & \mathbf{I}_Z \end{bmatrix} \begin{bmatrix} \mathbf{p}_{\mathbf{c}1} \\ \mathbf{p}_{\mathbf{c}2} \\ \mathbf{p}_{\mathbf{c}3} \\ \mathbf{p}_{\mathbf{c}4} \end{bmatrix} = \mathbf{0}$$

ıdding all rows of above equations , the parity nodes $\mathbf{p}_{c_2}, \mathbf{p}_{c_3}, \mathbf{p}_{c_4}$ are eliminated, obtains

$$\mathbf{A}\mathbf{u} + \mathbf{p}_{\mathbf{a}1}^{(y)} = 0 \Rightarrow \mathbf{p}_{\mathbf{c}1} = (\mathbf{A}\mathbf{u})^{(Z-y)} \quad (\because I_Z^{(y)}\mathbf{p}_{\mathbf{c}1} = \mathbf{p}_{\mathbf{c}1}^{(y)})$$

using the 1,2,3 row of above simultaneous equations, we extract parity check nodes in sequence

$$\begin{aligned} \mathbf{p}_{a2} &= \mathbf{a}_1\mathbf{u} + \mathbf{I}_z^{(x)}\mathbf{p}_{\mathbf{c}1} \\ \mathbf{p}_{c3} &= \mathbf{a}_2\mathbf{u} + \mathbf{p}_{\mathbf{c}2} \\ \mathbf{p}_{c4} &= \mathbf{a}_3\mathbf{u} + I_z^{(y)}\mathbf{p}_{\mathbf{c}1} + \mathbf{p}_{\mathbf{c}3} \end{aligned}$$

after solving $\mathbf{p}_c$, $\mathbf{p}_e$ can be easily solved

$$\mathbf{p}_e = \mathbf{C}_1\mathbf{u} + \mathbf{C}_2\mathbf{p}_c$$

Encoding (Example)

We are provided with the base matrix and Expansion factor(z) . With help of it , we need to create the H matrix.

Message m =[m1 m2 m3 m4].

Codeword c =[m1 m2 m3 m4 p1 p2 p3 p4]

TASK:
Here, p1,p2,p3,p4 are unkown. We need to find it.

$$
\left.\begin{array}{cccc|cccc}
I_1 & 0 & I_3 & I_1 & I_2 & I & 0 & 0 \\
I_2 & I & 0 & I_3 & 0 & I & I & 0 \\
0 & I_4 & I_2 & I & I_1 & 0 & I & I \\
I_4 & I_1 & I & 0 & I_2 & 0 & 0 & I
\end{array}\right]
$$

H matrix with Z_c: 5

Encoding (Example)

$$I_1m_1 + I_3m_3 + I_1m_4 + I_2p_1 + Ip_2 = \mathbf{0}$$ ....eq 1

$$I_2m_1 + \mathbf{I}m_2 + I_3m_4 + \quad \mathbf{I}p_2 + Ip_3 = \mathbf{0}$$ ....eq 2

$$I_1m_2 + I_2m_3 + \mathbf{I}m_4 + \quad I_1p_1 \quad + \quad \mathbf{I}p_3 + Ip_4 = \mathbf{0}$$ ....eq 3

$$I_3m_1 + I_1m_2 + \mathbf{I}m_3 + \quad I_2p_1 \quad + Ip_4 = \mathbf{0}$$ ....eq 4

Encoding (Example)

$$I_1m_1 + I_3m_3 + I_1m_4 + I_2p_1 + Ip_2 = \mathbf{0} \qquad \text{....eq 1}$$

$$I_2m_1 + \mathbf{I}m_2 + I_3m_4 + \mathbf{I}p_2 + Ip_3 = \mathbf{0} \qquad \text{....eq 2}$$

$$I_1m_2 + I_2m_3 + \mathbf{I}m_4 + I_1p_1 + \mathbf{I}p_3 + Ip_4 = \mathbf{0} \qquad \text{....eq 3}$$

$$I_3m_1 + I_1m_2 + \mathbf{I}m_3 + I_2p_1 + Ip_4 = \mathbf{0} \qquad \text{....eq 4}$$

Adding all 4 Eqn's (Mod 2 operation is used) we get

$$I_1p_1 = I_1m_1 + I_3m_3 + I_1m_4 + I_2m_1 + \mathbf{I}m_2 + I_3m_4$$
$$+I_1m_2 + I_2m_3 + \mathbf{I}m_4 + I_3m_1 + I_1m_2 + \mathbf{I}m_3$$

Encoding (Example)

Using Eq 1 find p_2

- 1: $I_1 m1 + I_3 m3 + I_1 m4 + I_2 p1 + I\, p2 = 0$

Using Eq 2 find p_3

- 2: $I_2 m1 + I\, m2 + I_3 m3 \qquad + I\, p2 + I\, p3 = 0$

Using Eq 4 find p_4

- 4: $I_4 m1 + I_1 m2 + I\, m3 + I_2 p1 \qquad + I\, p4 = 0$

IMPORTANCE OF DOUBLE DIAGONAL STRUCTURE :

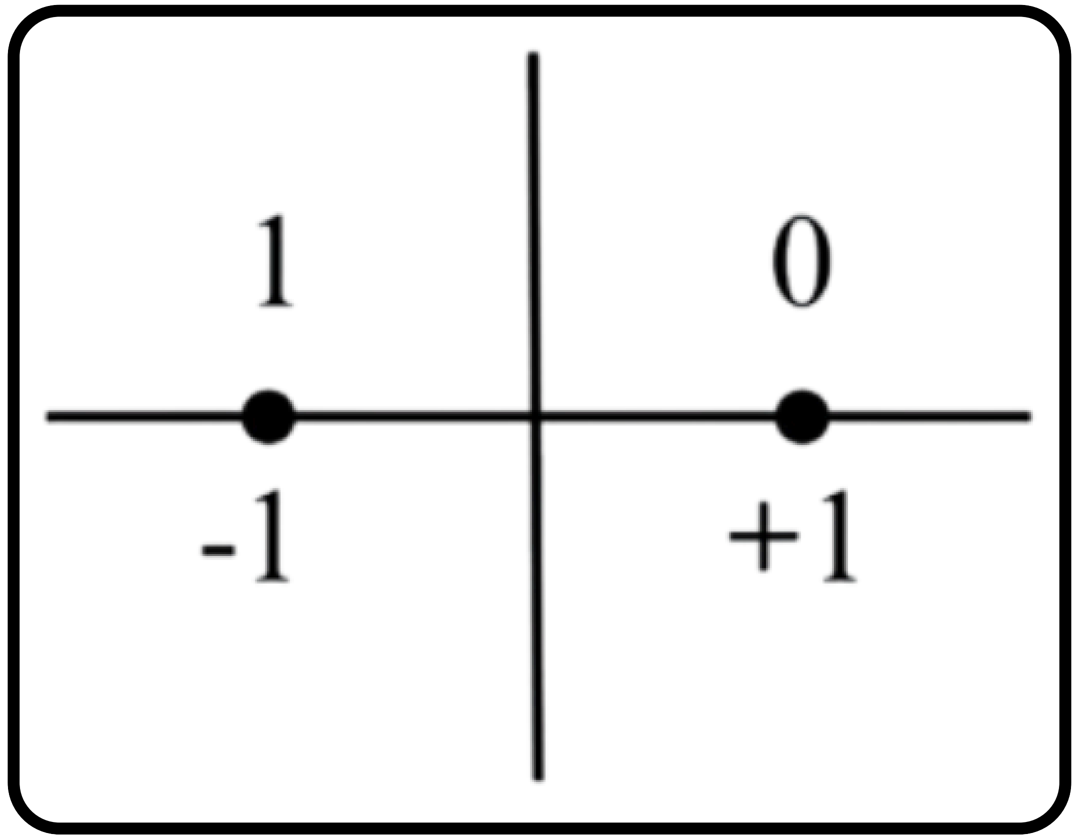
The double diagonal structure in the parity-check matrix simplifies encoding and decoding due to its predictable and sparse pattern. This structure allows easy elimination of unknown parity bits p2,p3,p4 through simple back-substitution, since each parity equation involves only a small number of known message and unknown parity bits in a cascading manner.

BPSK Modulation & AWGN

Modulation

What is BPSK?

- Binary Phase Shift Keying (BPSK) is a digital modulation technique.
- It maps binary data to phase shifts of a carrier wave.
- Especially useful in noisy environments due to its robustness.



BPSK Mapping

- Each bit $b \in \{0, 1\}$ is mapped to a signal symbol s.
- The mapping rule is:

$$s = 1-2b$$

  - If b = 0, then s = +1 volts
  - If b = 1, then s = -1 volts

AWGN– Additive White Gaussian Noise

What is AWGN?

- A noise model used in communication systems to mimic random processes affecting signal transmission.
- Represents thermal noise and other interferences in the channel.
- “White” implies the noise is spread across all frequencies.
- “Gaussian” refers to the normal distribution of the noise values.

Noise Characteristics

- Mean (μ): 0
- Variance (σ^2):

$$\sigma^2 = 1/\gamma = 1/SNR$$

or more generally with coding:

$$\sigma^2 = 1 / (2 \times Eb/N_0 \times r)$$

- where:
    - Eb/N_0: Energy per bit to noise power spectral density ratio
    - r: Code rate

AWGN– Additive White Gaussian Noise

AWGN Simulation Formula

$$u = \sigma_n \times u_s$$

- $u_s \sim N(0, 1)$ (standard normal distribution)
- Adds realistic distortion to the modulated signal
- Essential in testing and designing communication systems

Role in Transmission

- Noise is added during transmission or channel propagation:

$$y = s + u$$

- It affects signal clarity, potentially causing bit errors

Hard Decision Decoding

Hard Decision Decoding Process

1. **Initialization:**
   - Convert received symbols into hard bits (0 or 1) based on sign.
   - Initialize messages from variable nodes (VN) to check nodes (CN) using received bits.
2. **VN to CN Message Update:**
   - For each VN, send majority value (excluding one CN at a time) to connected CNs.
3. **CN to VN Message Update:**
   - For each CN, send XOR of all incoming VN messages (excluding one VN) to each VN.
4. **Estimate Codeword:**
   - Each VN makes a decision based on received message and majority of incoming CN messages.
5. **Check Syndrome:**
   - Compute syndrome H * c . If zero, decoding is successful.
6. **Repeat:**
   - Iterate steps 2–5 up to maxIteration or until convergence.

Hard Decision Decoding - Tanner Graph

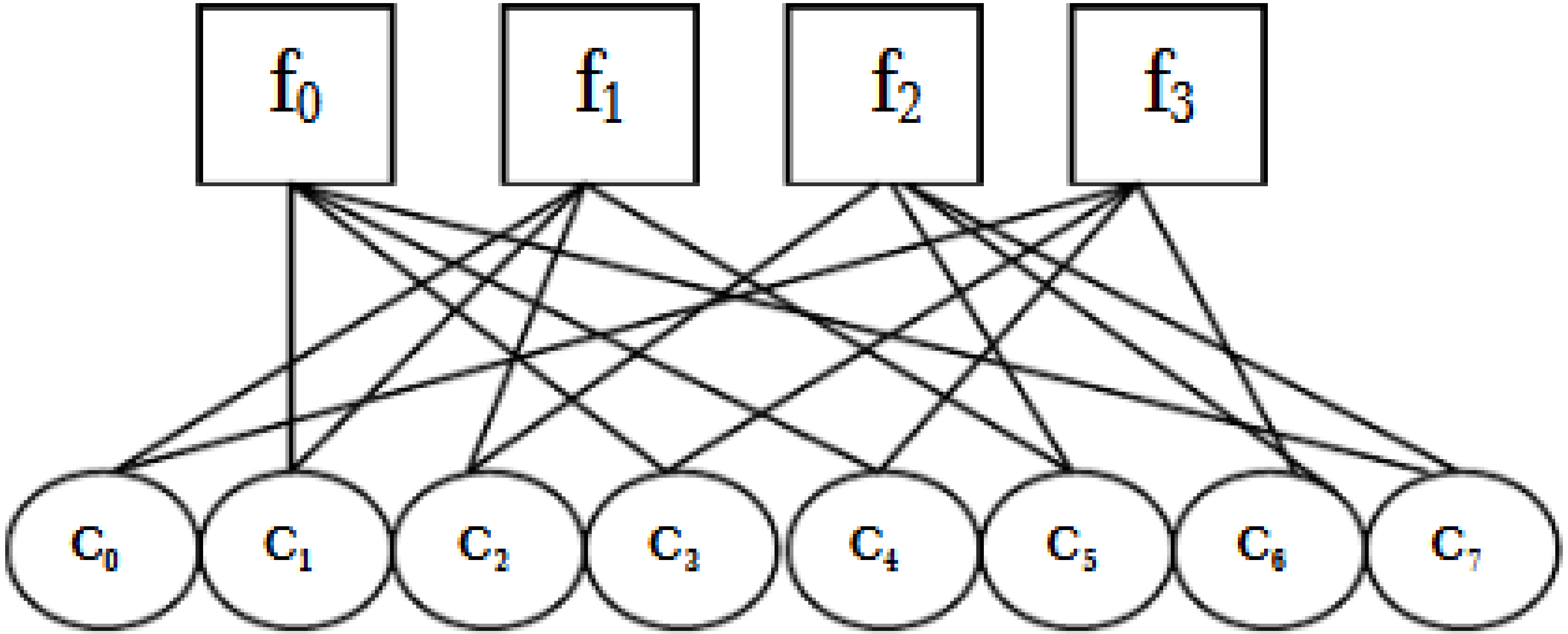


Fig 2: Tanner Graph corresponding to parity check matrix of 8 column (N) and 4 row(M)[4]

Hard Decision Decoding - VNs & CNs

- After demodulation, each Variable Node (VN) sends its bit to connected Check Nodes (CN).
- CNs then return updated parity-based bits back to the VNs.
- This process uses:
  - **Repetition Code** for VN → CN
  - **Single Parity-Check (SPC) Code** for CN → VN
- VNs update their bit using majority voting.
- Iteration continues until all parity checks are satisfied (syndrome = 0).

Hard Decision Decoding - VNs & CNs

Hard Decision Decoding Example – Step-by-Step

🔹 Given:

- Original Codeword: C = [00111100]
- Received Codeword: R = [00011100]

🛠️ **Decoding Steps:**

1. Initialization:

   Each variable node (VN) receives initial bit values from the received word.

2. VN → CN Message Passing:

   Each VN sends a message to its connected CNs using XOR of other CNs it's connected to.

   E.g., For f0: $c3 \oplus c5 \oplus c7 = 1 \oplus 1 \oplus 0 = 0$

3. CN → VN Feedback:

   Each CN checks messages from connected VNs.

   E.g., c0 is connected to f1 and f3: values = 1 and 0.

4. Majority Voting:

   VNs update their bit value based on majority of incoming messages (including initial bit).

Check Nodes and Variable Nodes

Table 1 Received values from c-nodes

	Received values from c-nodes
f_0	$c_1 \to 0, c_3 \to 1, c_5 \to 1, c_7 \to 0$
f_1	$c_1 \to 0, c_2 \to 0, c_4 \to 1, c_0 \to 0$
f_2	$c_2 \to 0, c_5 \to 1, c_6 \to 0, c_7 \to 0$
f_3	$c_0 \to 0, c_3 \to 1, c_4 \to 1, c_6 \to 0$

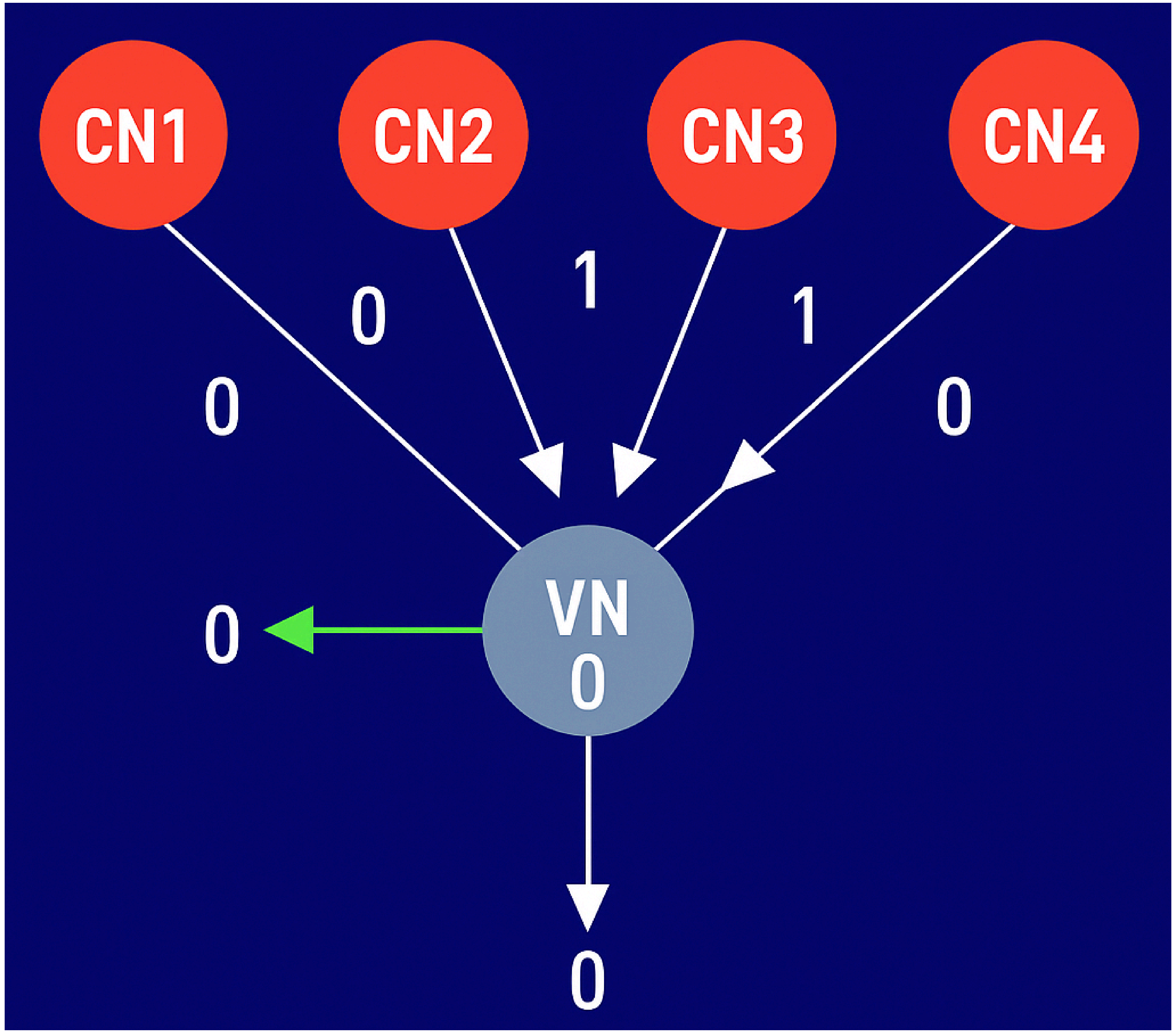
Table 2 Sent values from v-nodes

	sent values from v-nodes
f_0	$0 \to c_1, 1 \to c_3, 1 \to c_5, 0 \to c_7$
f_1	$1 \to c_0, 1 \to c_1, 1 \to c_2, 0 \to c_4$
f_2	$1 \to c_2, 0 \to c_5, 1 \to c_6, 1 \to c_7$
f_3	$0 \to c_0, 1 \to c_3, 1 \to c_4, 0 \to c_6$

Check Nodes and Variable Nodes

c-node	Connected to v-node	received R	values assigned to v-nodes	Majority vote
c_0	1,3	0	$f_1 \to 1,\ f_3 \to 0$	0
c_1	0,1	0	$f_0 \to 0,\ f_1 \to 1$	0
c_2	1,2	0	$f_1 \to 1,\ f_2 \to 1$	1
c_3	0,3	1	$f_0 \to 1,\ f_3 \to 1$	1
c_4	1,3	1	$f_1 \to 0,\ f_3 \to 1$	1
c_5	0,2	1	$f_0 \to 1,\ f_2 \to 0$	1
c_6	2,3	0	$f_1 \to 1,\ f_3 \to 0$	0
c_7	0,2	0	$f_0 \to 0,\ f_2 \to 1$	0

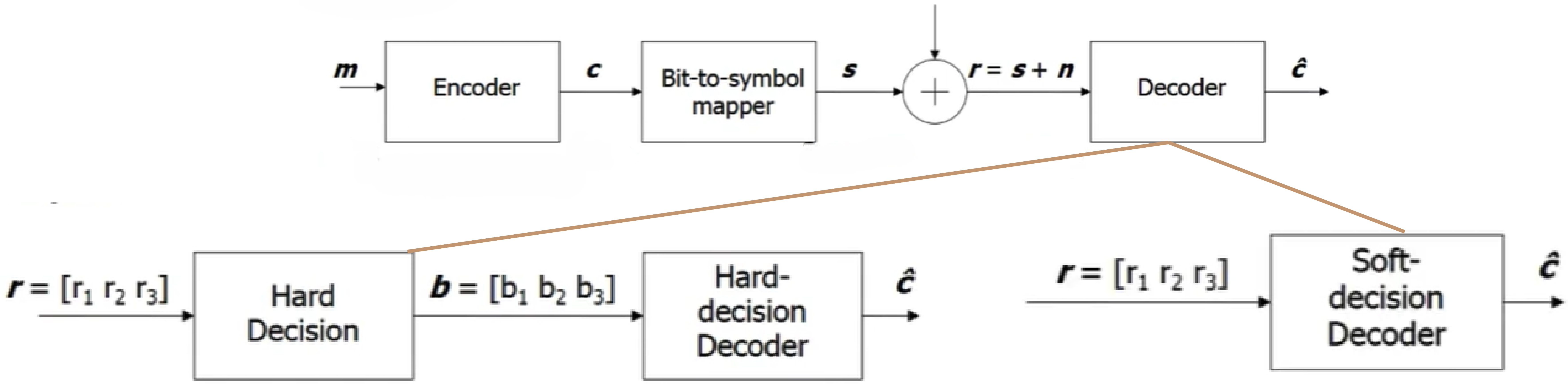
Hard Decision Decoding Output



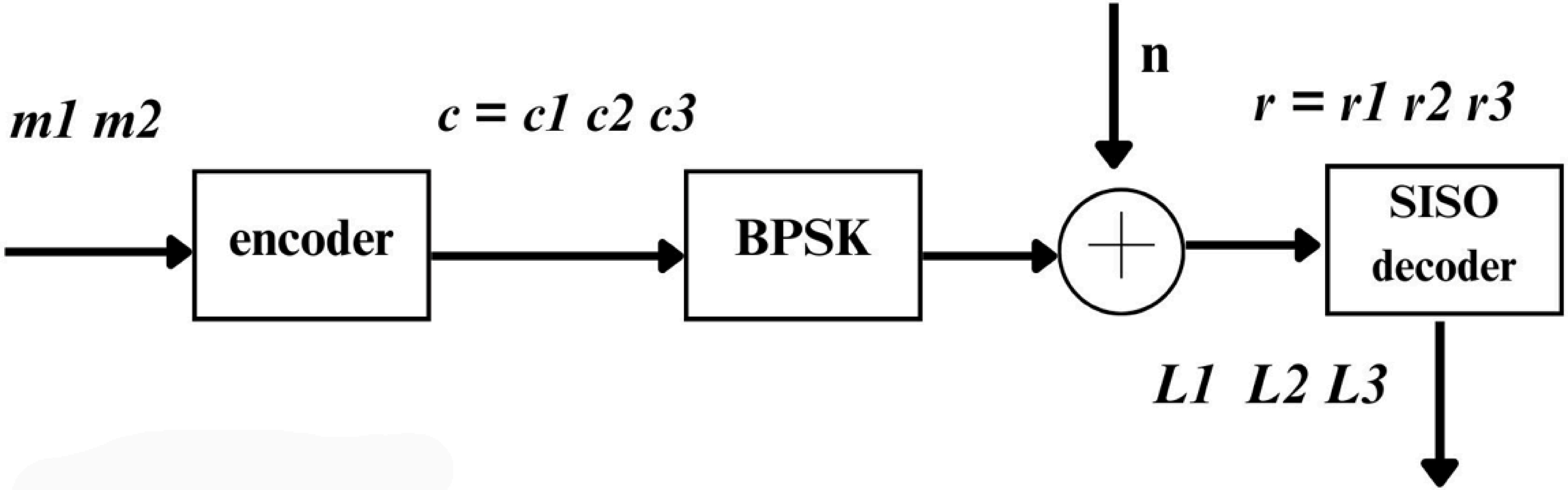
Soft Decision Decoding

Idea of Soft decision decoding

- Soft decision decoding is a technique used in error correction coding that uses the real-valued outputs from the channel (not just hard 0 or 1 decisions).
- Instead of making an early decision on each bit, the decoder processes probabilistic information—usually in the form of Log-Likelihood Ratios (LLRs).



SISO decoder for SPC Code



- Li = l1 + lext, 1
- LLR l1 = (2/(sigma)^2)*r1
- lext,1 = sign(l2)*sign(l3)*min(|l2|,|l3|))

Log likelyhood ratio

$$P(C_i = 1|r_i) = \frac{P(r_i|\, C_i = 1)P(C_i = 1)}{P(r_i)}$$

$$P(C_i = 0|r_i) = \frac{P(r_i|\, C_i = 0)P(C_i = 0)}{P(r_i)}$$

$$\lambda_i = \frac{P(C_i = 1|r_i)}{P(C_i = 0|r_i)} = \frac{P(r_i|\, C_i = 1)}{P(r_i|\, C_i = 0)}$$

$$\lambda_i = \frac{\left(\frac{1}{\sqrt{2\pi}\sigma}\right)\left(e^{-\frac{(r_i+1)^2}{2\,\sigma^2}}\right)}{\left(\frac{1}{\sqrt{2\pi}\sigma}\right)\left(e^{-\frac{(r_i-1)^2}{2\,\sigma^2}}\right)}$$

$$\lambda_i = e^{\frac{2r_i}{\sigma^2}}$$

$$\iota_i = \ln(\lambda_i) = \frac{2r_i}{\sigma^2}$$ intrinsic LLR

$where\ P(C_i = 1) = P(C_i = 0) = ½$

SISO decoder for SPC

Ex – (3, 2)

C1	C2	C3
0	0	0
0	1	1
1	0	1
1	1	0

$$c1 = c2 \oplus c3$$

Where Pi = (Ci=1)

$$P1 = P2(1 - P3) + P3(1 - P2) \text{ .... (1)}$$

$$(1 - P1) = P2P3 + (1 - P2)(1 - P3) \text{ .... (2)}$$

Now Subtract Equations (1-2)

$$P1 - (1 - P1) = P2(P3(1 - p3)) - (1 - P2)(P3(1 - p3))$$

$$P1 - (1 - P1) = (P2 - (1 - P2))(P3 - (1 - p3))$$

$$\frac{P1 - (1 - P1)}{P1 + (1 - P1)} = \frac{(P2 - (1 - P2))}{P2 + (1 - P2)} \frac{(P3 - (1 - p3))}{P3 + (1 - P3)}$$

$$\frac{1 - \frac{(1 - P1)}{P1}}{1 + \frac{(1 - P1)}{P1}} = \frac{1 - \frac{(1 - P2)}{P2}}{1 + \frac{(1 - P2)}{P2}} \frac{1 - \frac{(1 - P3)}{P3}}{1 + \frac{(1 - P3)}{P3}}$$

$$\frac{1 - e^{-l_{ext,1}}}{1 + e^{-l_{ext,1}}} = \frac{1 - e^{-l_2}}{1 + e^{-l_2}} \frac{1 - e^{-l_3}}{1 + e^{-l_3}}$$

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

$$\tanh\left(\frac{l_{ext,1}}{2}\right) = \tanh\left(\frac{l_2}{2}\right) \tanh\left(\frac{l_3}{2}\right)$$

SISO decoder for SPC

$l_{ext,1}$ can be written in two-part Magnitude and sign

$$sgn(l_{ext,1}) = sgn(l_2)\, sgn(l_3)$$

$$\tanh\left(\frac{|l_{ext,1}|}{2}\right) = \tanh\left(\frac{|l_2|}{2}\right) \tanh\left(\frac{|l_3|}{2}\right)$$

$$\log\left(\tanh\left(\frac{|l_{ext,1}|}{2}\right)\right) = \log\left(\tanh\left(\frac{|l_2|}{2}\right)\right) + \log\left(\tanh\left(\frac{|l_3|}{2}\right)\right)$$

Now $f(x)$ is,

$$f(x) = \left|\log\left(\tanh\left(\frac{|x|}{2}\right)\right)\right|$$

$$f(|l_{ext,1}|) = f(|l_2|) + f(|l_3|)$$

$$|l_{ext,1}| = f\left(f(|l_2|) + f(|l_3|)\right)$$

Because of the characteristics of the f function

$$f(|l_2|) + f(|l_3|) \approx f(min(|l_2|, |l_3|))$$

$$|l_{ext,1}| = f(f(min(|l_2|, |l_3|)))$$

Now, f is inverse of its own

$$|l_{ext,1}| = min(|l_2|, |l_3|)$$

$$l_{ext,1} = sgn(l_{ext,1}) * |l_{ext,1}|$$

Similarly for n

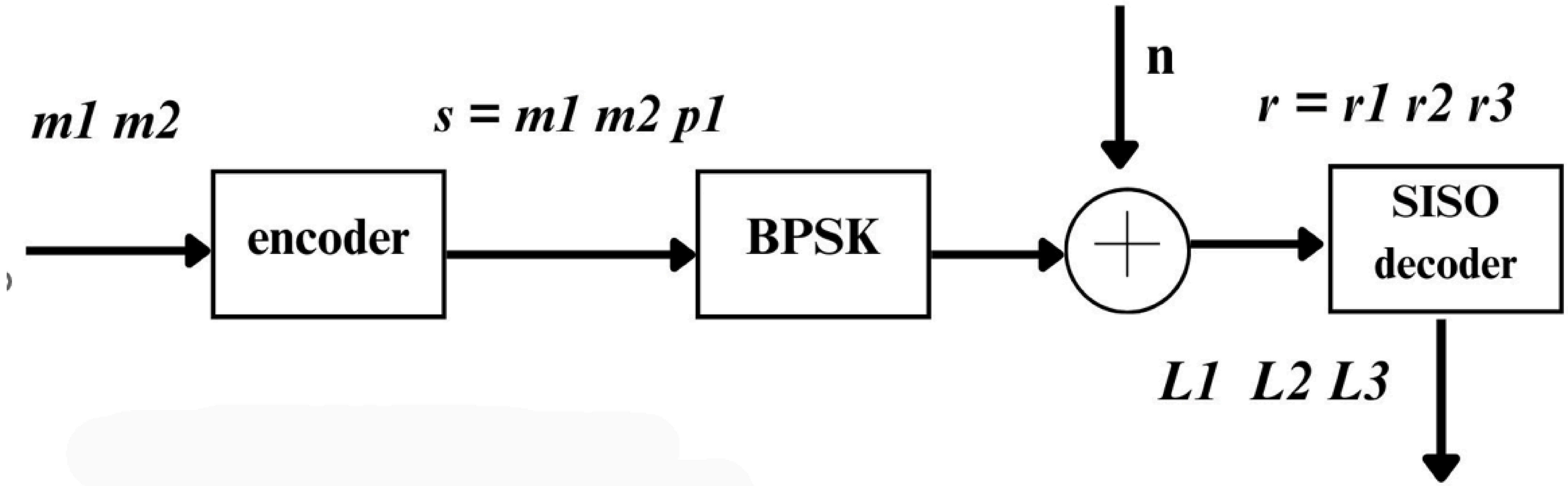
$$|l_{ext,1}| = min(|l_2|, |l_3|, \ldots\ldots, |l_n|)$$

$$sgn(l_{ext,1}) = sgn(l_2)\, sgn(l_3) \ldots\ldots sgn(l_n)$$

$$l_{ext,1} = sgn(l_{ext,1}) * |l_{ext,1}|$$

This is the return value after the mean operation.

SISO decoder for repetition Code



- Li = belief that bit is 0
- L1 = computed using r1,r2,r3
- L1 = r1+r2+r3
- for L1, r1 = intrinsic, r2+r3 = extrinsic

SISO decoder for Repetition Code

Calculation for L1

$$l_i = \frac{P(C_i = 1 \mid r_1, r_2, \ldots\ldots, r_n)}{P(C_i = 0 \mid r_1, r_2, \ldots\ldots, r_n)} = \frac{P(r_1, r_2, \ldots\ldots, r_n \mid C_i = 1)}{P(r_1, r_2, \ldots\ldots, r_n \mid C_i = 0)}$$

$$l_i = \frac{P(r_1 \mid C_1 = 1)\, P(r_2 \mid C_2 = 1) \ldots P(r_n \mid C_n = 1)}{P(r_1 \mid C_1 = 0)\, P(r_2 \mid C_2 = 0) \ldots P(r_n \mid C_n = 0)}$$ (Because all r 1 … independent from each other)

$$= \frac{\left(e^{-\frac{(r_1+1)^2}{2\sigma^2}}\right)}{\left(e^{-\frac{(r_1-1)^2}{2\sigma^2}}\right)} \frac{\left(e^{-\frac{(r_2+1)^2}{2\sigma^2}}\right)}{\left(e^{-\frac{(r_2-1)^2}{2\sigma^2}}\right)} \frac{\left(e^{-\frac{(r_3+1)^2}{2\sigma^2}}\right)}{\left(e^{-\frac{(r_3-1)^2}{2\sigma^2}}\right)} \qquad \left(\because \quad \lambda = \frac{\left(\frac{1}{\sqrt{2\pi\sigma}}\right)\left(e^{-\frac{(r_i-1)^2}{2\sigma^2}}\right)}{\left(\frac{1}{\sqrt{2\pi\sigma}}\right)\left(e^{-\frac{(r_i+1)^2}{2\sigma^2}}\right)} \right)$$

$$= e^{\frac{2r_1}{\sigma^2}}\, e^{\frac{2r_2}{\sigma^2}}\, e^{\frac{2r_3}{\sigma^2}}$$

$L_i = r_1 + r_2 + \ldots\ldots + r_n$ ignore $2/\sigma^2$ *factor*

Message Passing in Tanner Graph Using Log-Likelihood Ratios (LLRs)

Initialization (Intrinsic Information from Variable Nodes)

- Initially, each VN computes its intrinsic LLR based on the received channel information (e.g., from a noisy BPSK signal).
- Each VN sends this intrinsic LLR as a message to all connected Check Nodes (CNs)**.**

Check Node to Variable Node Message Passing (SISO SPC)

- Upon receiving LLR messages from all connected VNs, the CN computes extrinsic LLRs to send back to each VN.
- This computation at each CN can be viewed as a Soft-In Soft-Out (SISO) decoding of a Single Parity Check (SPC) code, using the incoming LLRs.
- The CN sends the computed extrinsic LLR to each VN.

Message Passing in Tanner Graph Using Log-Likelihood Ratios (LLRs)

Variable Node Update (SISO Repetition)

- After receiving extrinsic LLRs from all connected CNs, each VN combines them with its intrinsic LLR.
- This combination represents a Soft-In Soft-Out decoding of a repetition code, since each VN is connected to multiple CNs and thus receives multiple estimates of the same bit.
- These updated messages are sent back to each connected CN excluding the one from which the message was received (i.e., again, extrinsic information).

Iterative Process

- Steps 2 and 3 are repeated iteratively.
- The goal is for the LLRs at each VN to converge to a stable value representing a strong belief in the bit's value.

Soft Decision Decoding - MINSUM Algorithm

Storage Matrix L:

- L is a sparse matrix with the same dimensions as the parity check matrix H.
- An entry in Lis zero if the corresponding entry in H is zero.
- An entry in L is non-zero only if the corresponding entry in H is 1.
- Each non-zero entry in a row of L is initialized with the received value corresponding to that variable node (bit) from the received codeword.

$$\boldsymbol{r} = [\, r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5 \quad r_6 \quad r_7 \,]$$

$$L = \begin{bmatrix} r_1 & r_2 & r_3 & 0 & r_5 & 0 & 0 \\ 0 & r_2 & r_3 & r_4 & 0 & r_6 & 0 \\ r_1 & r_2 & 0 & r_4 & 0 & 0 & r_7 \\ r_1 & 0 & r_3 & 0 & r_5 & r_6 & r_7 \end{bmatrix}$$

Soft Decision Decoding - MINSUM algorithm

- In soft decision decoding, the received signal is used without hard demodulation, preserving the real-valued information.
- Each Variable Node (VN) sends a belief (soft information) to its connected Check Nodes (CNs).
- Check Nodes also send updated beliefs back to the VNs in a two-way iterative process.
- Message transfer:
    - VN to CN: Uses sum rule
    - CN to VN: Uses the Min-Sum algorithm.

MINSUM - Algorithm

- (Min-Sum SPC SISO)
- For each row:

Magnitude

- Min1 = Minimum absolute value of all nonzero entries in the row
- Min2 = Next higher absolute value
- Set magnitude of all values (except minimum) = Min1
- Set magnitude of minimum value = Min2

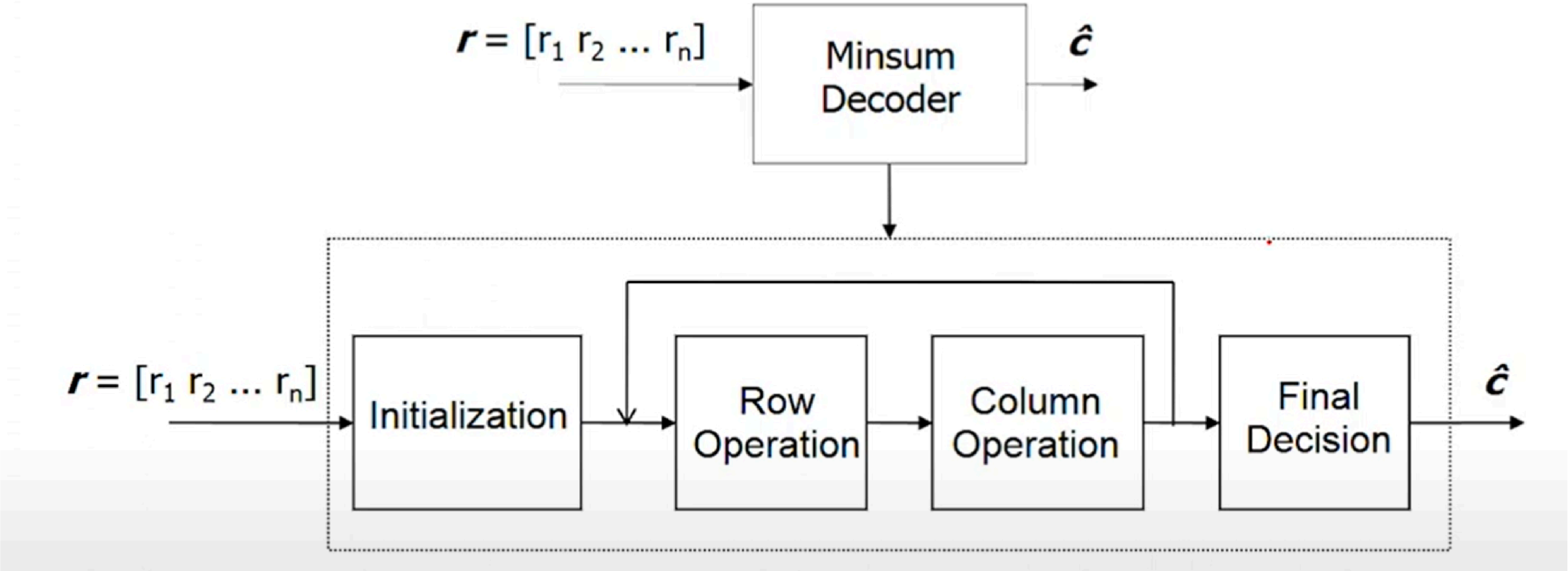
Sign

- Parity = Product of signs of entries in the row
- New sign of an entry = (Old Sign) × (Parity)

MINSUM - Algorithm

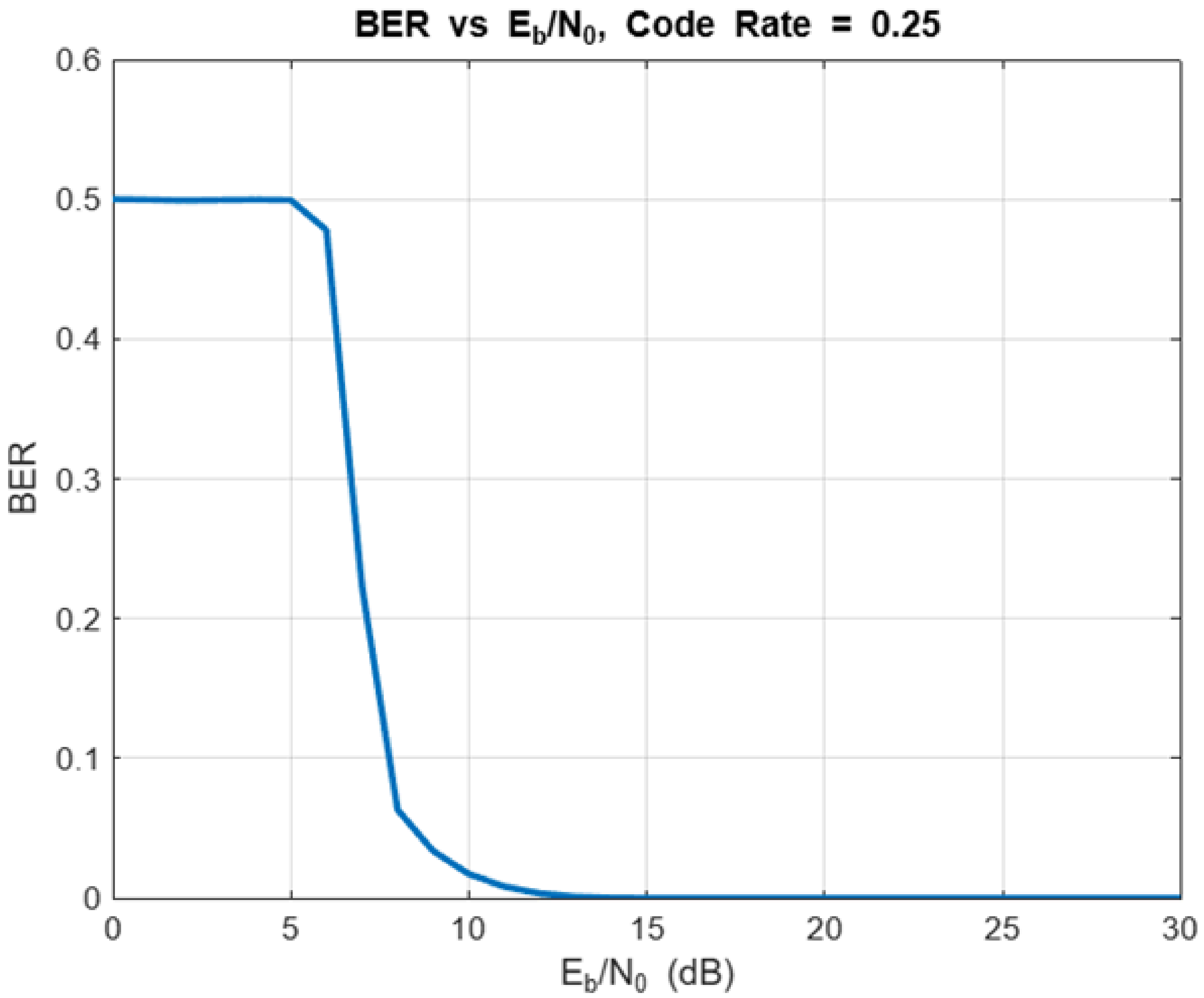
- (Min-Sum repetition SISO)
- For each column:

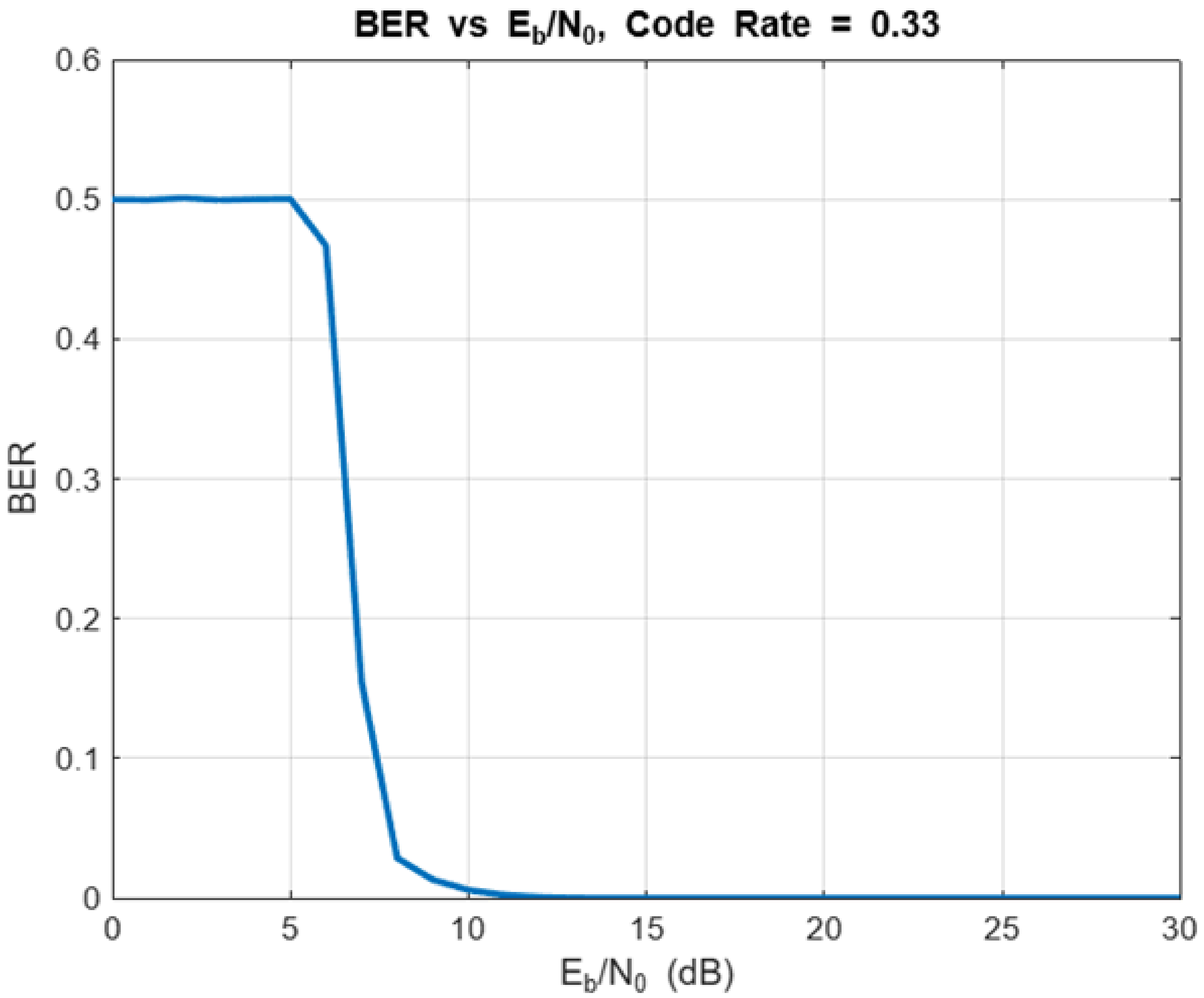
- sum_j = r_j + sum of all entries in column j
- new entry = sum - (old entry)



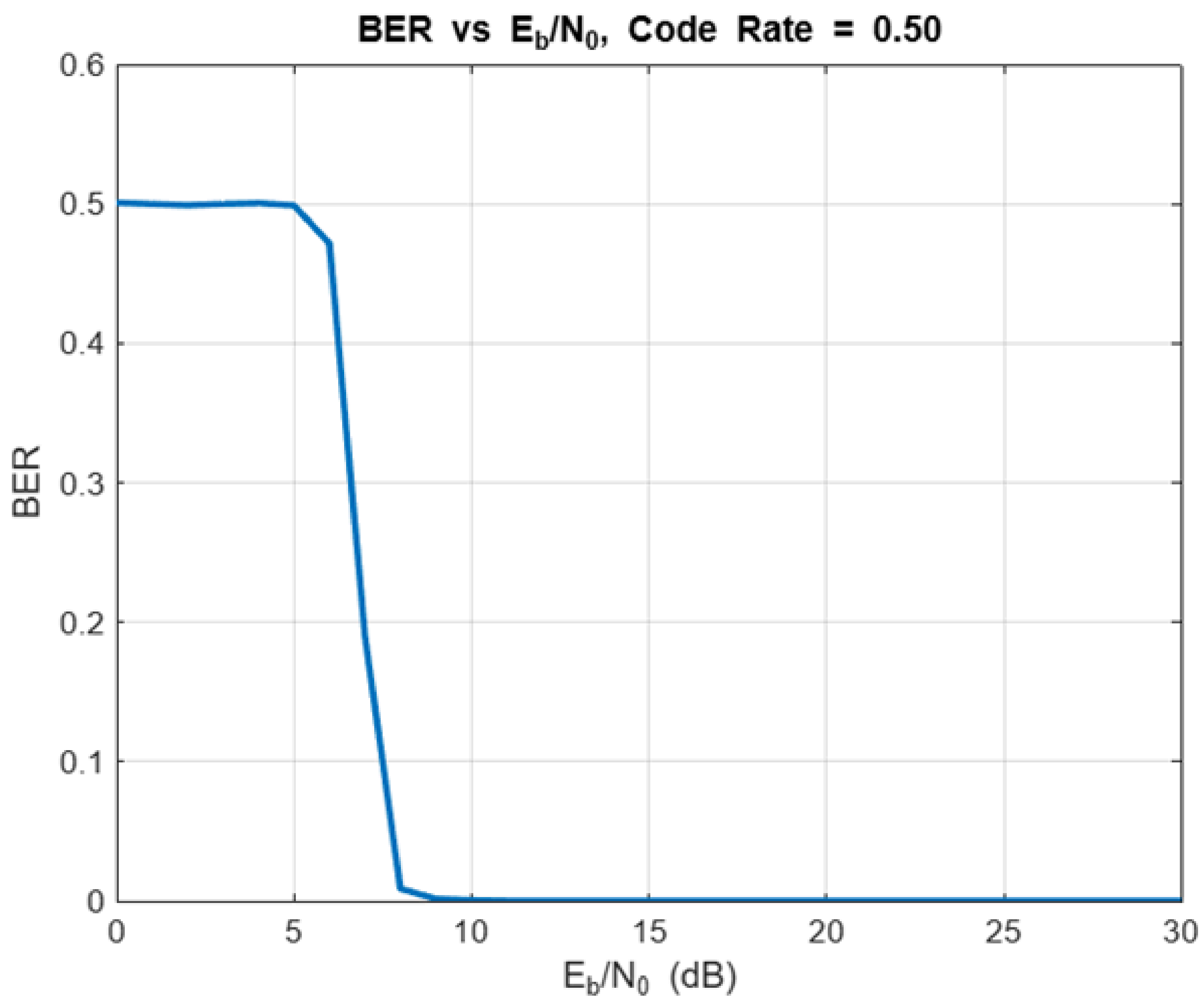
Results & Conclusions

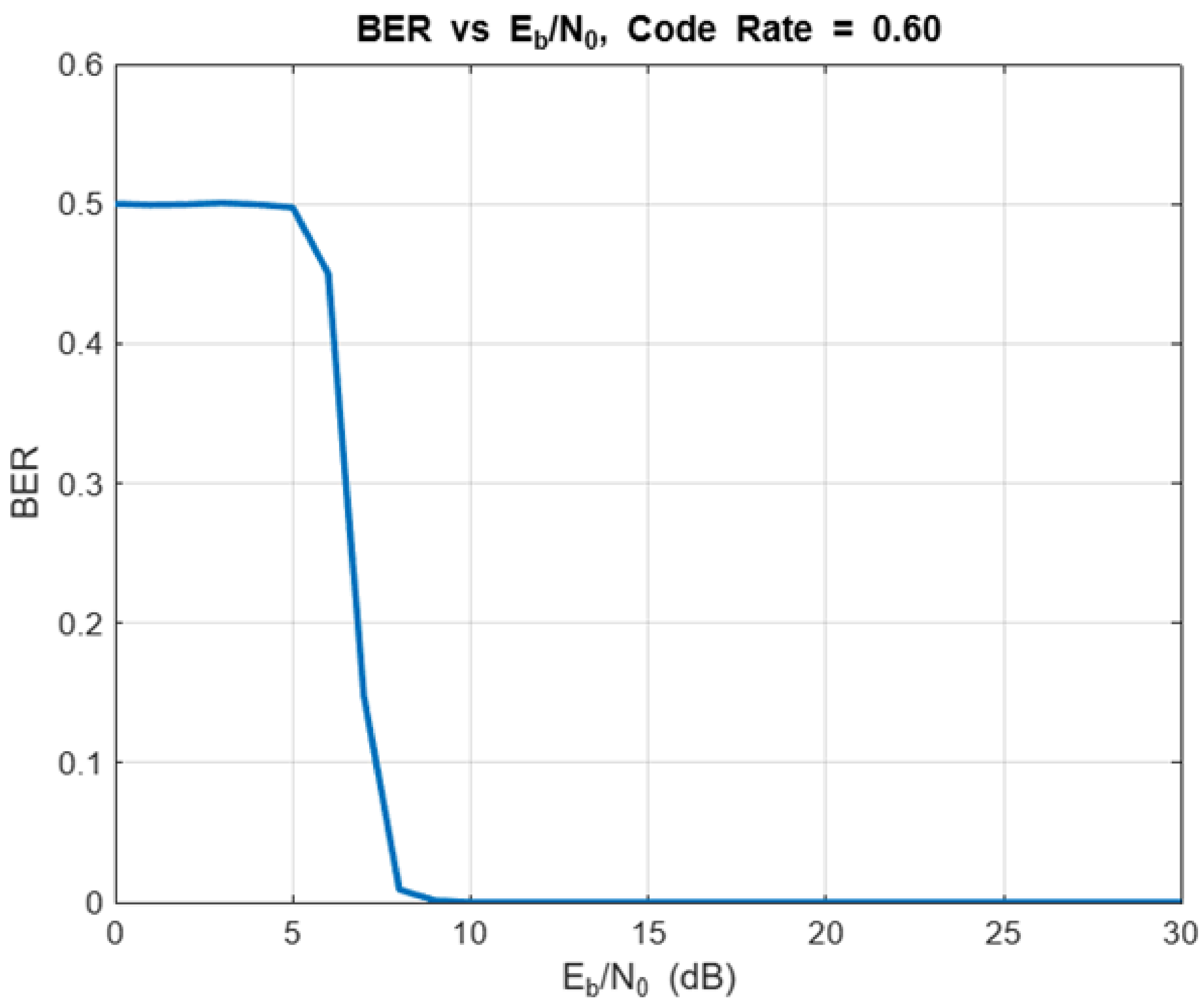
BER vs E_b/N_0 results for Hard decoding



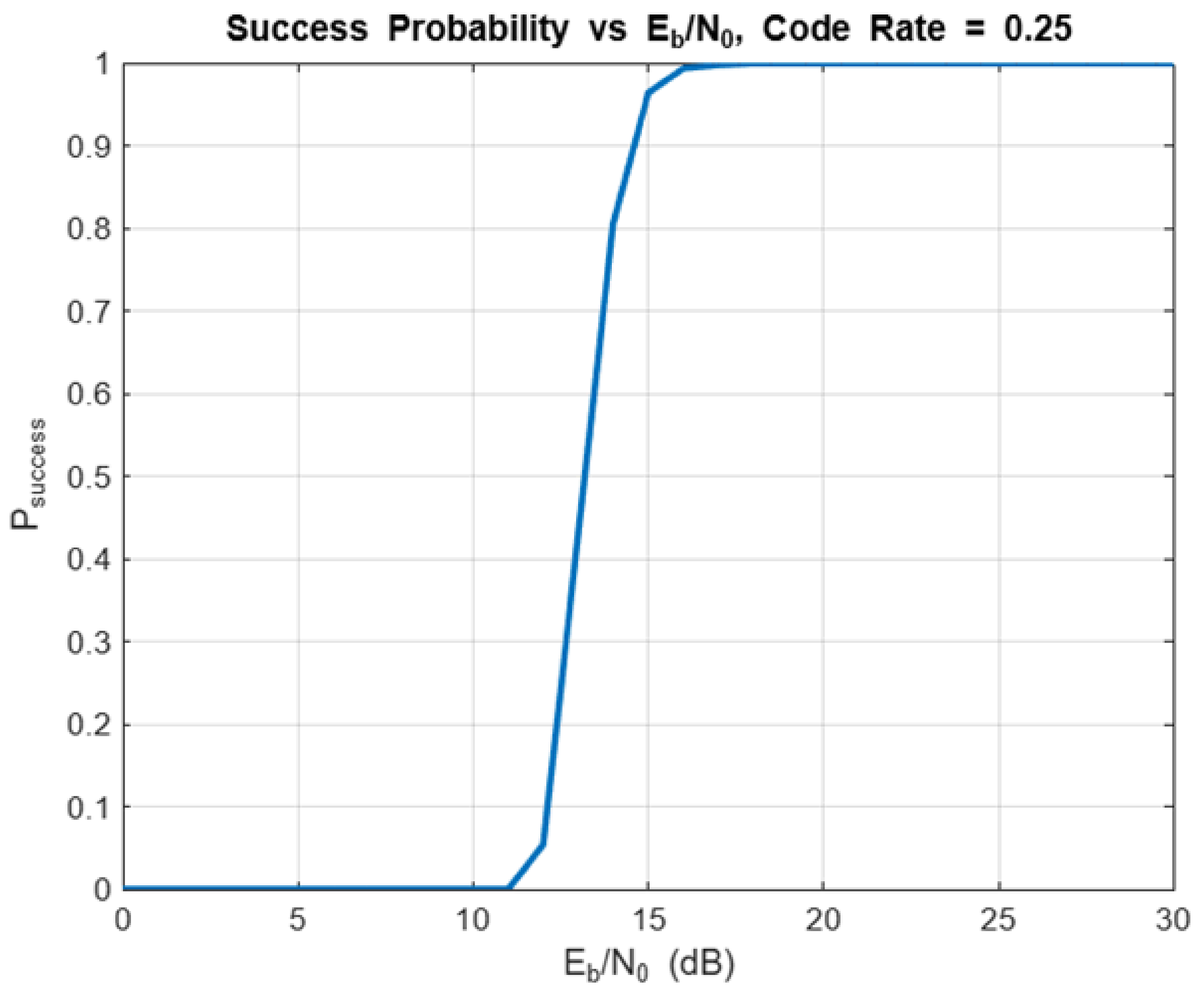


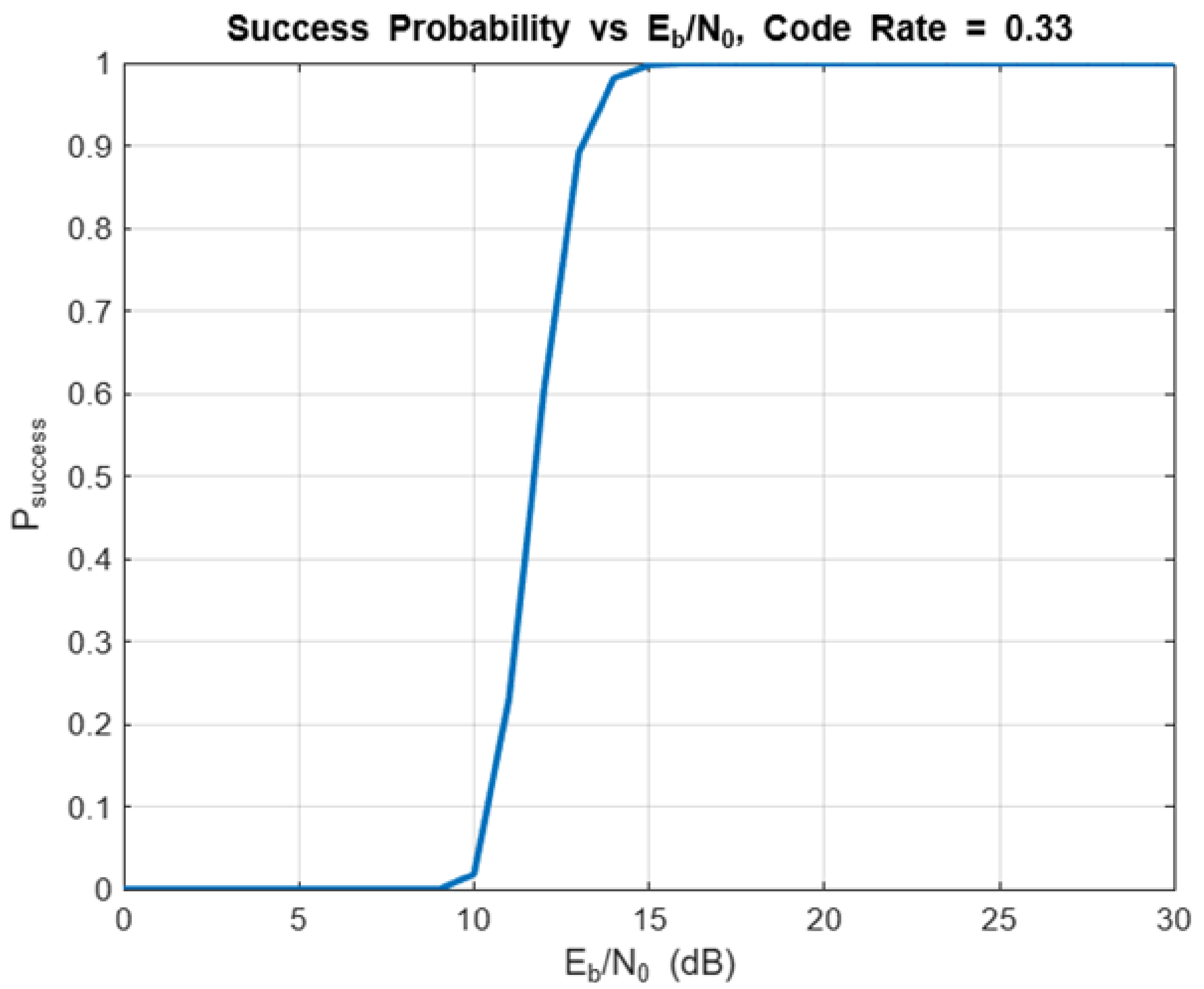
BER vs E_b/N_0 results for Hard decoding



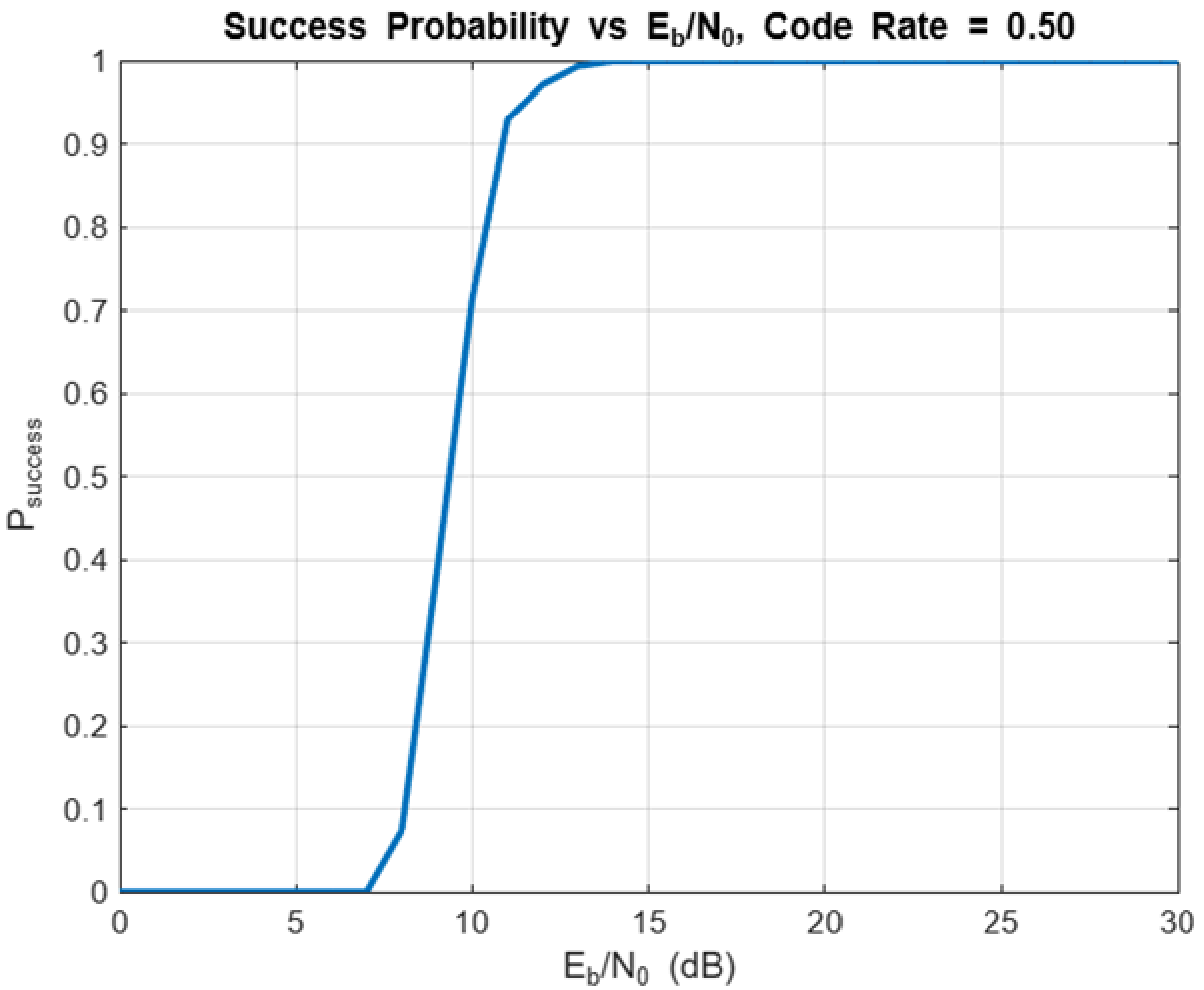


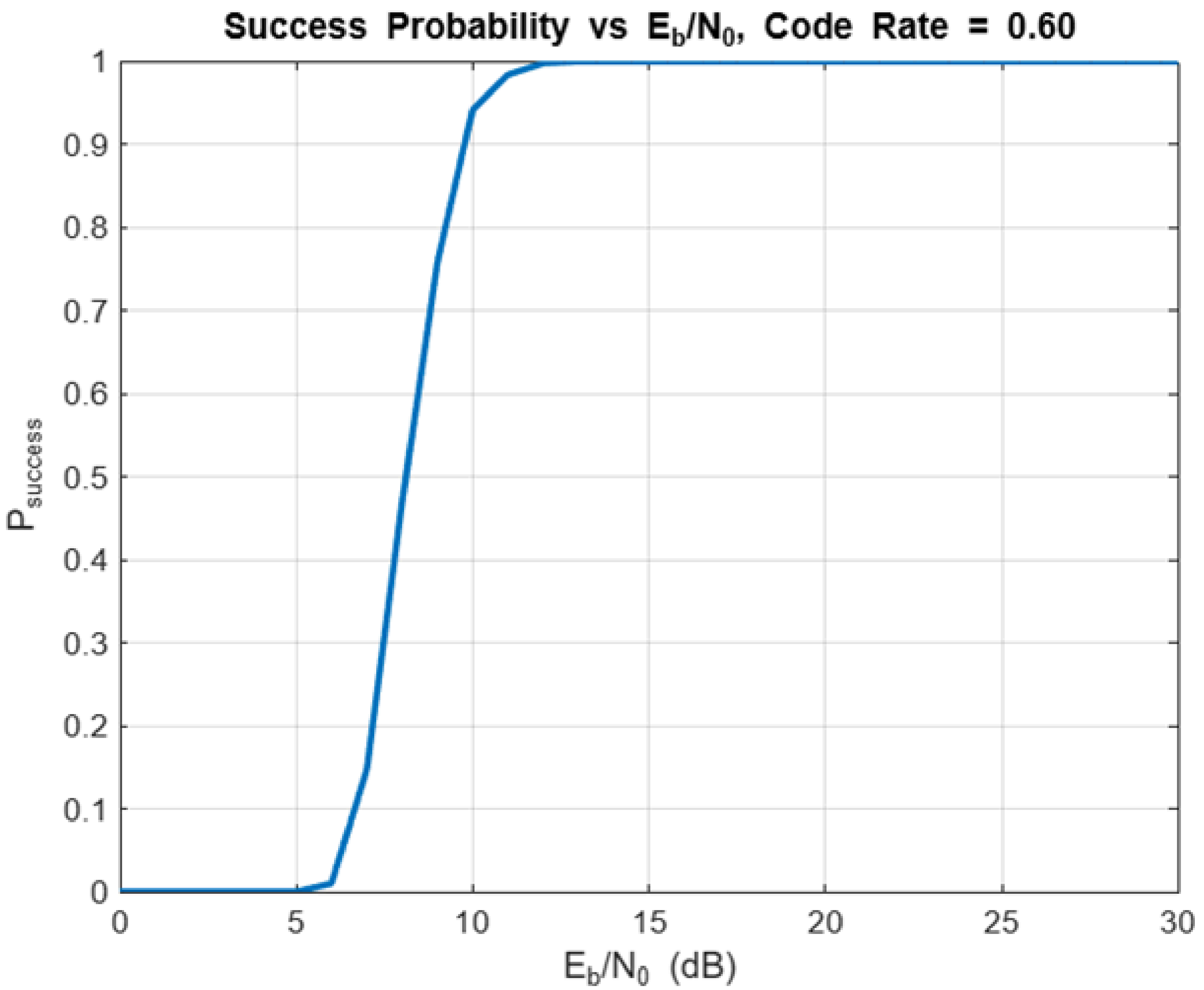
P_success vs E_b/N_0 results for Hard decoding



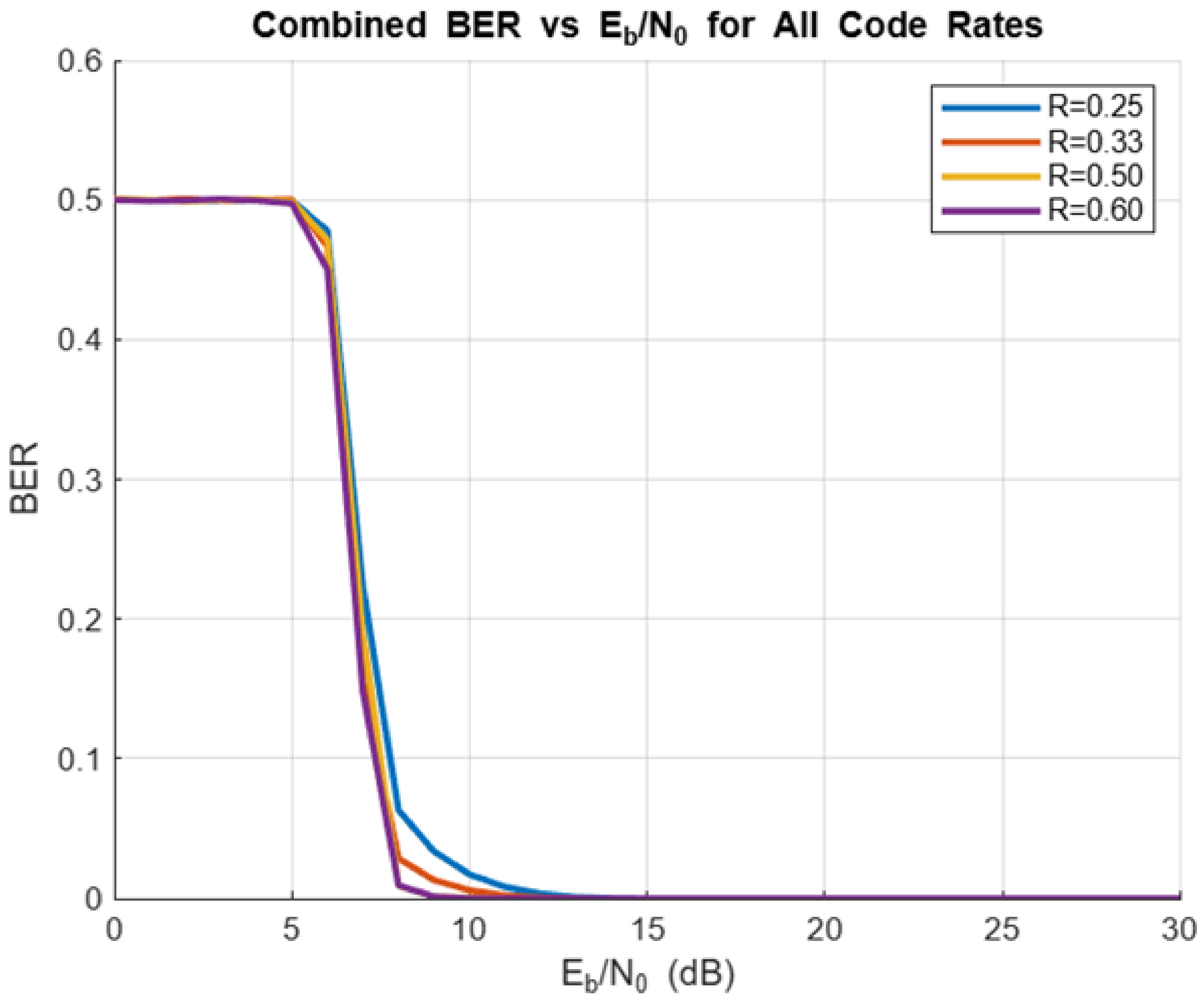


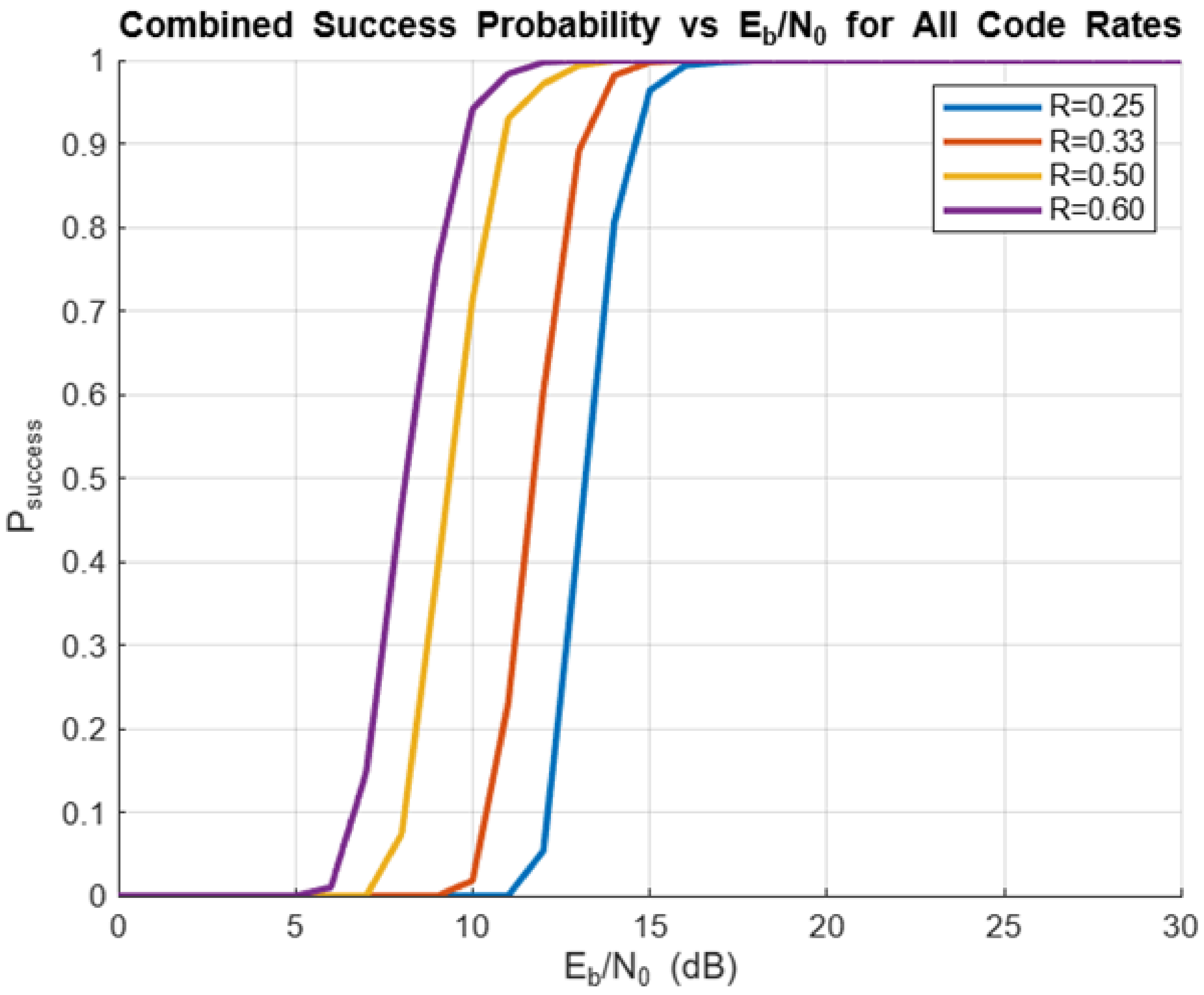
P_success vs E_b/N_0 results for Hard decoding



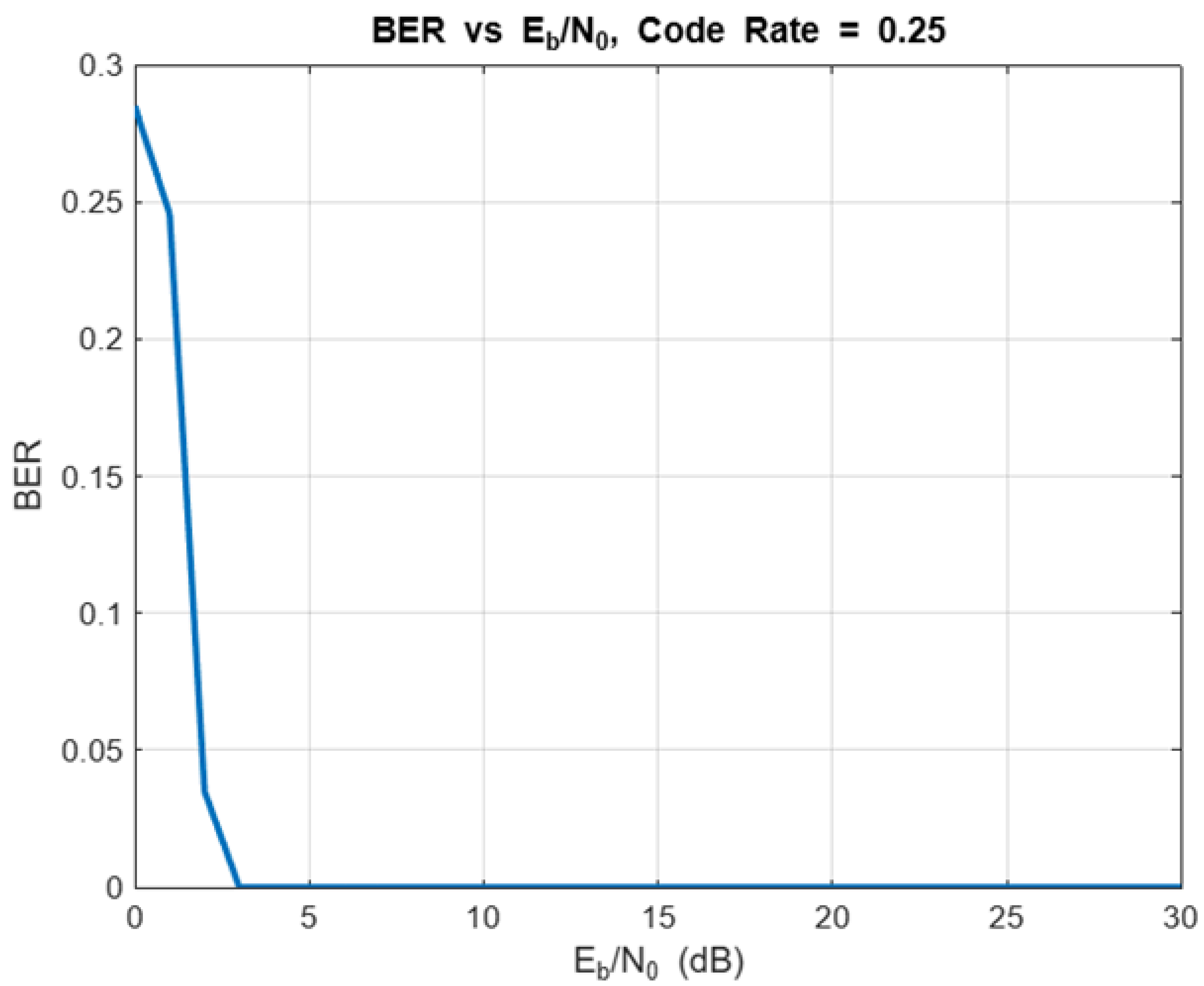


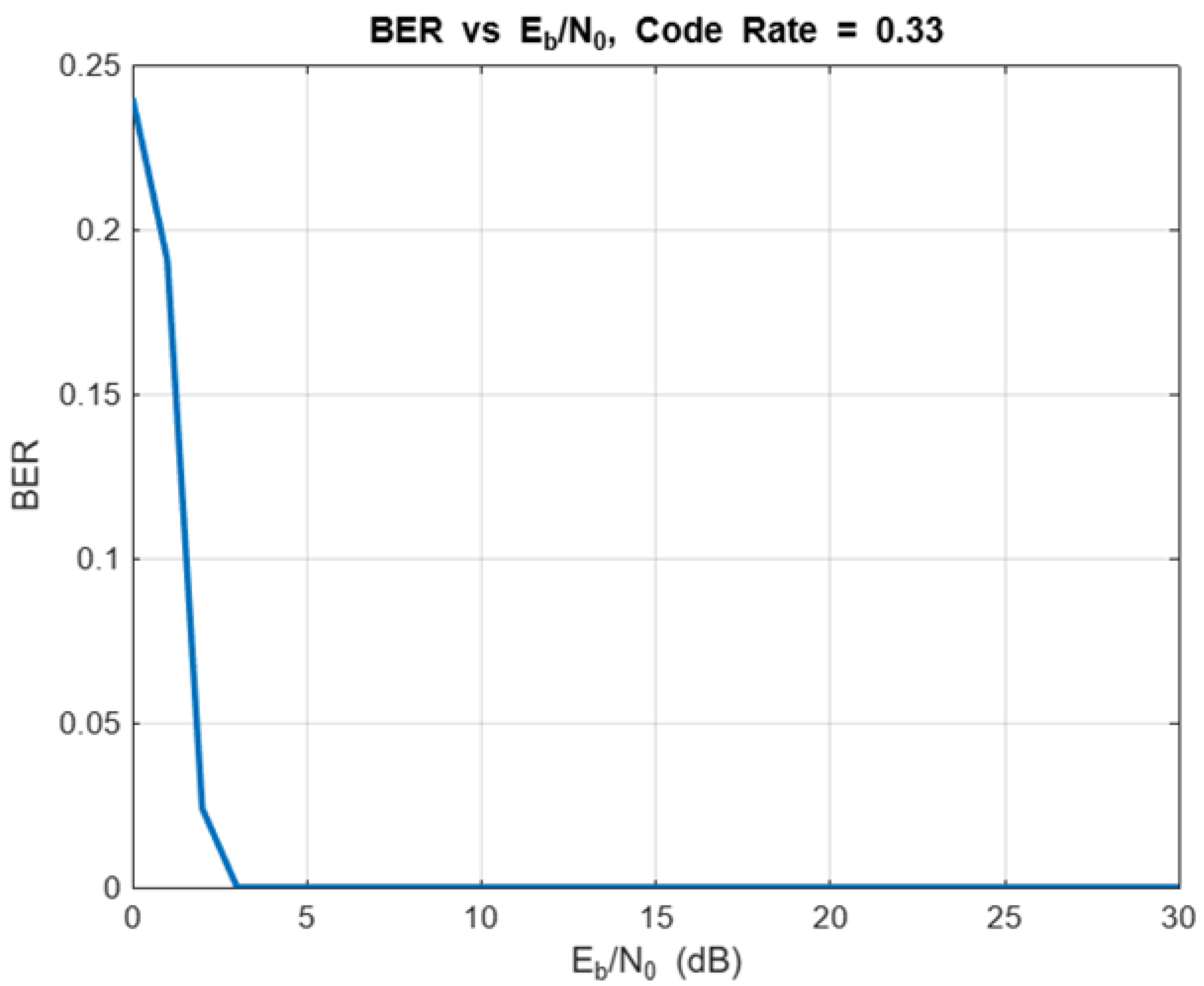
Combined graphs for all code rates(Hard decoding)



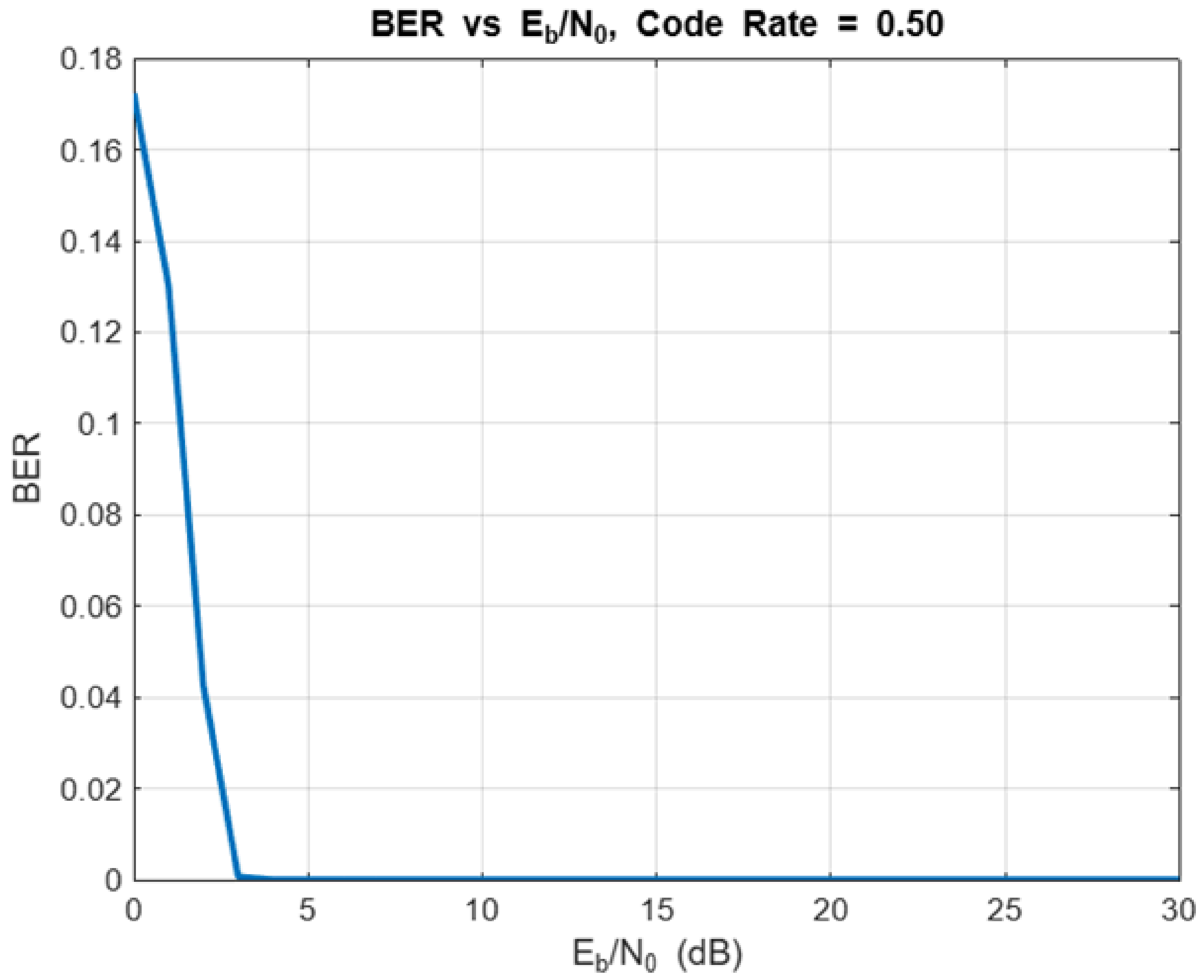


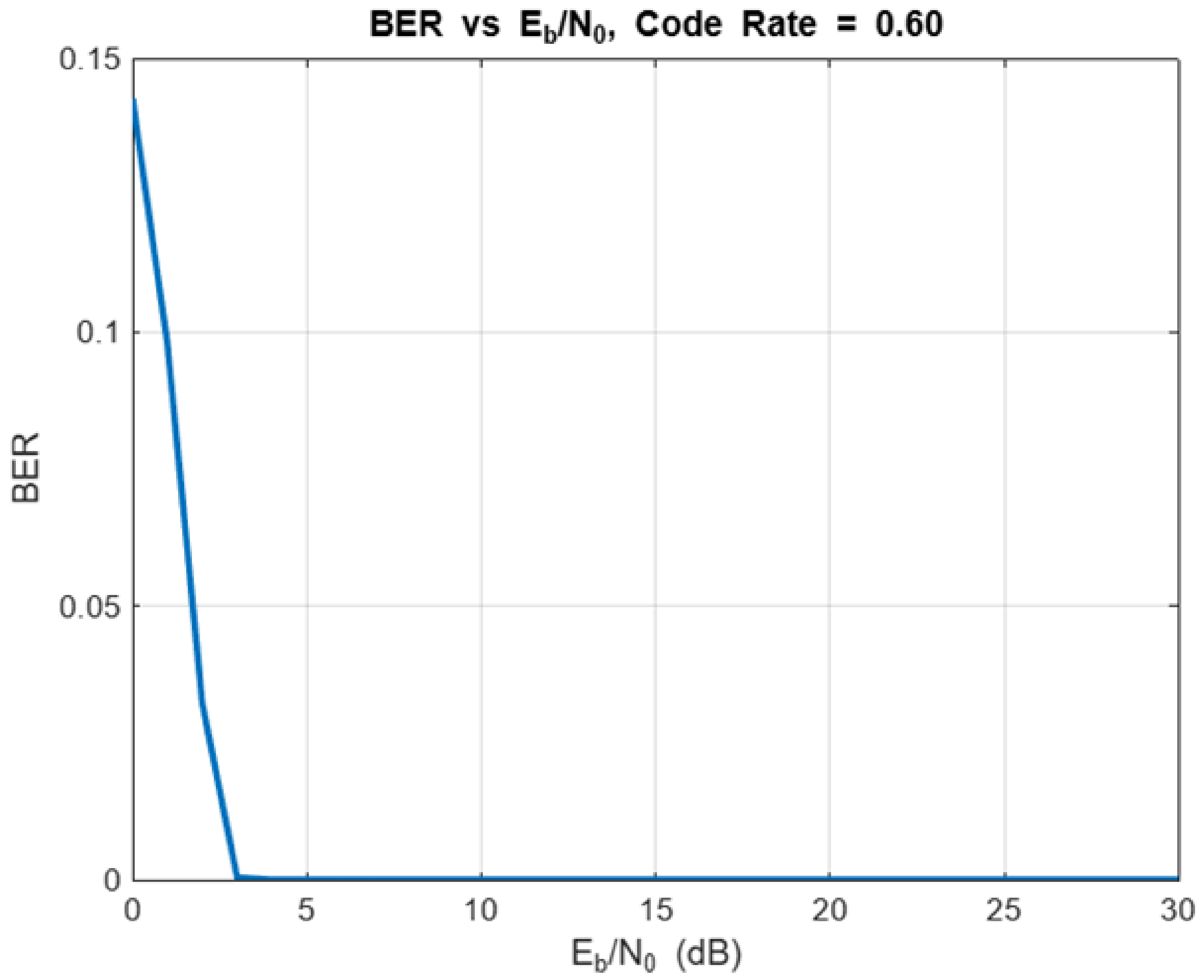
BER vs E_b/N_0 results for soft decoding



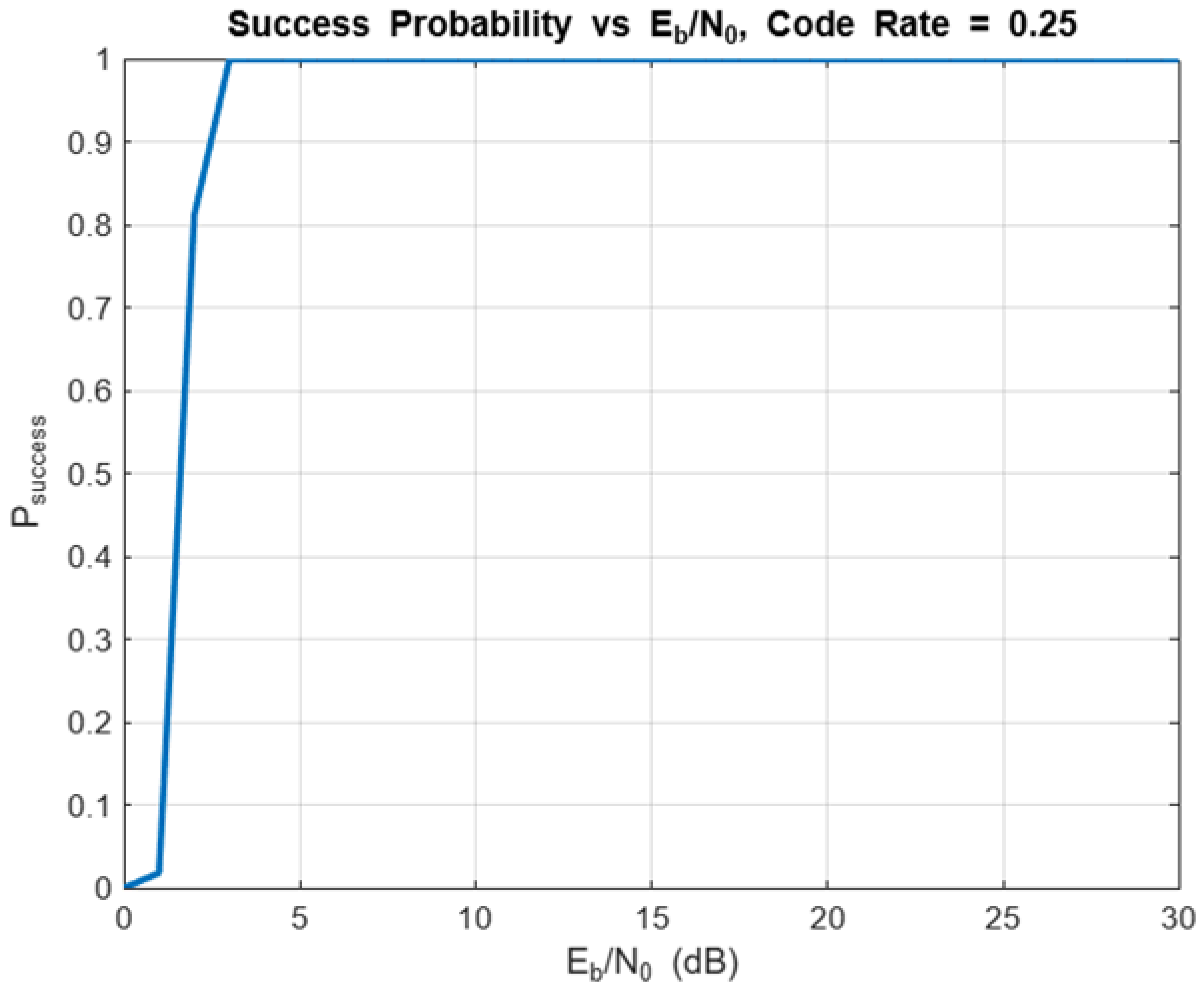


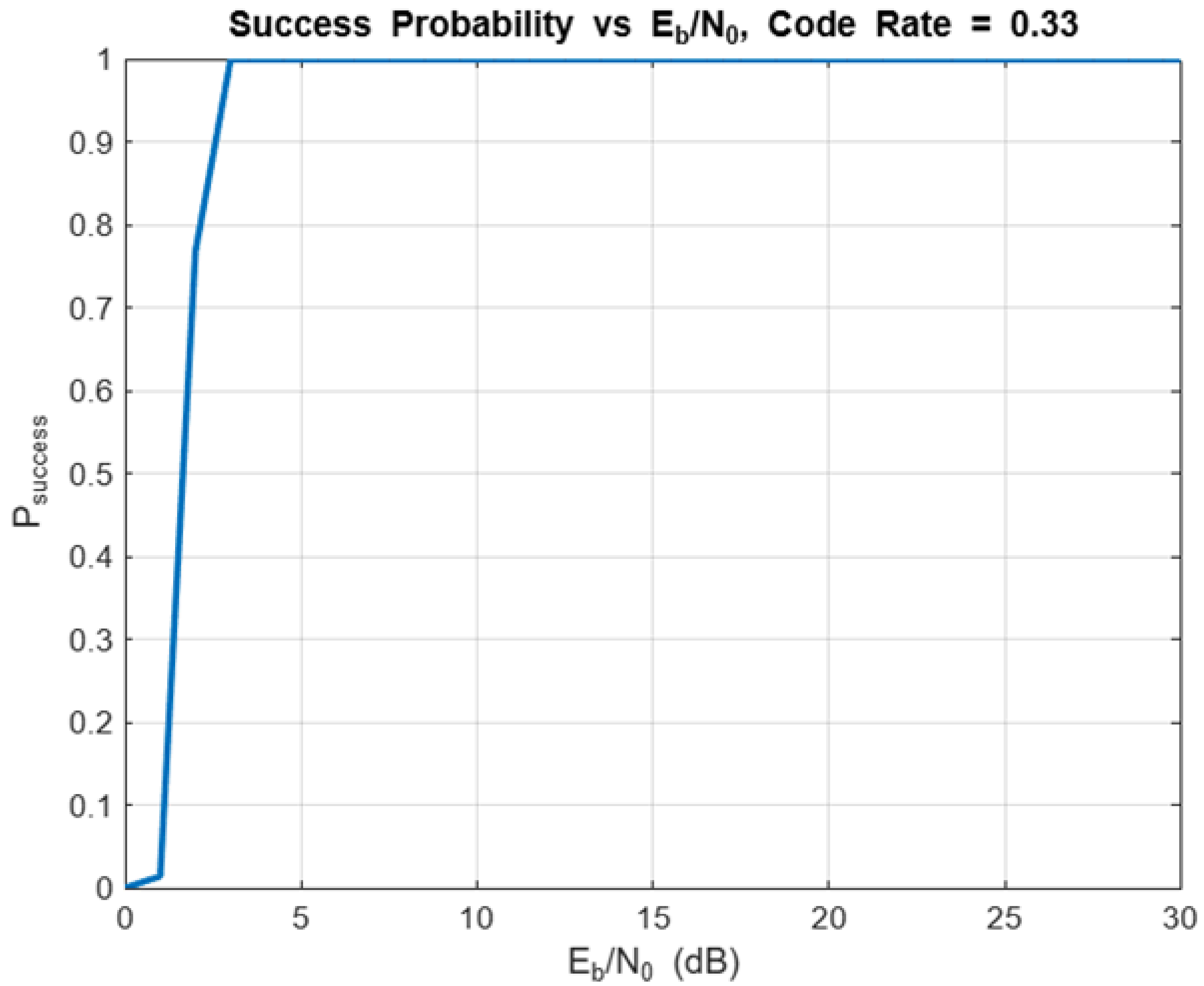
BER vs E_b/N_0 results for soft decoding



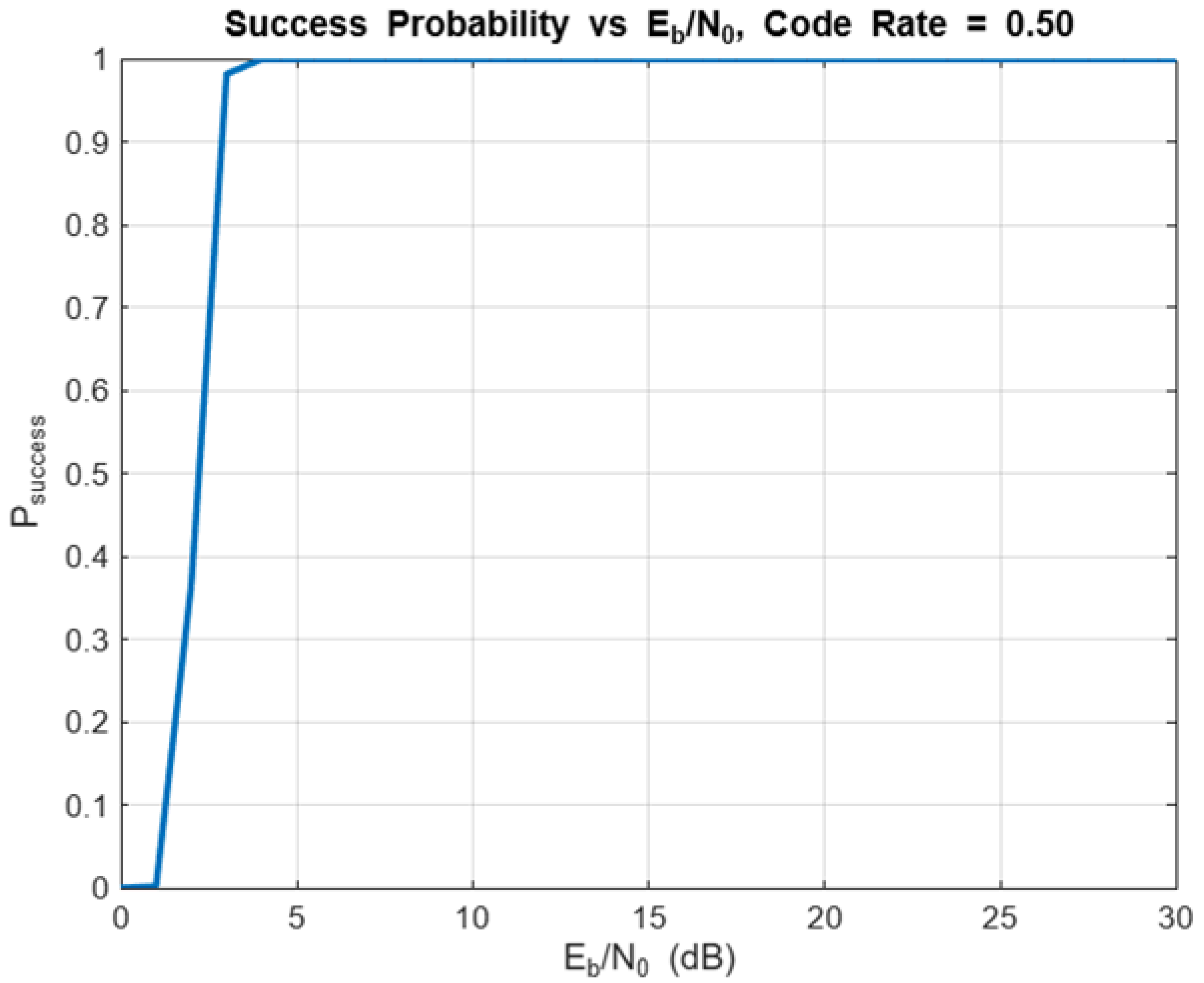


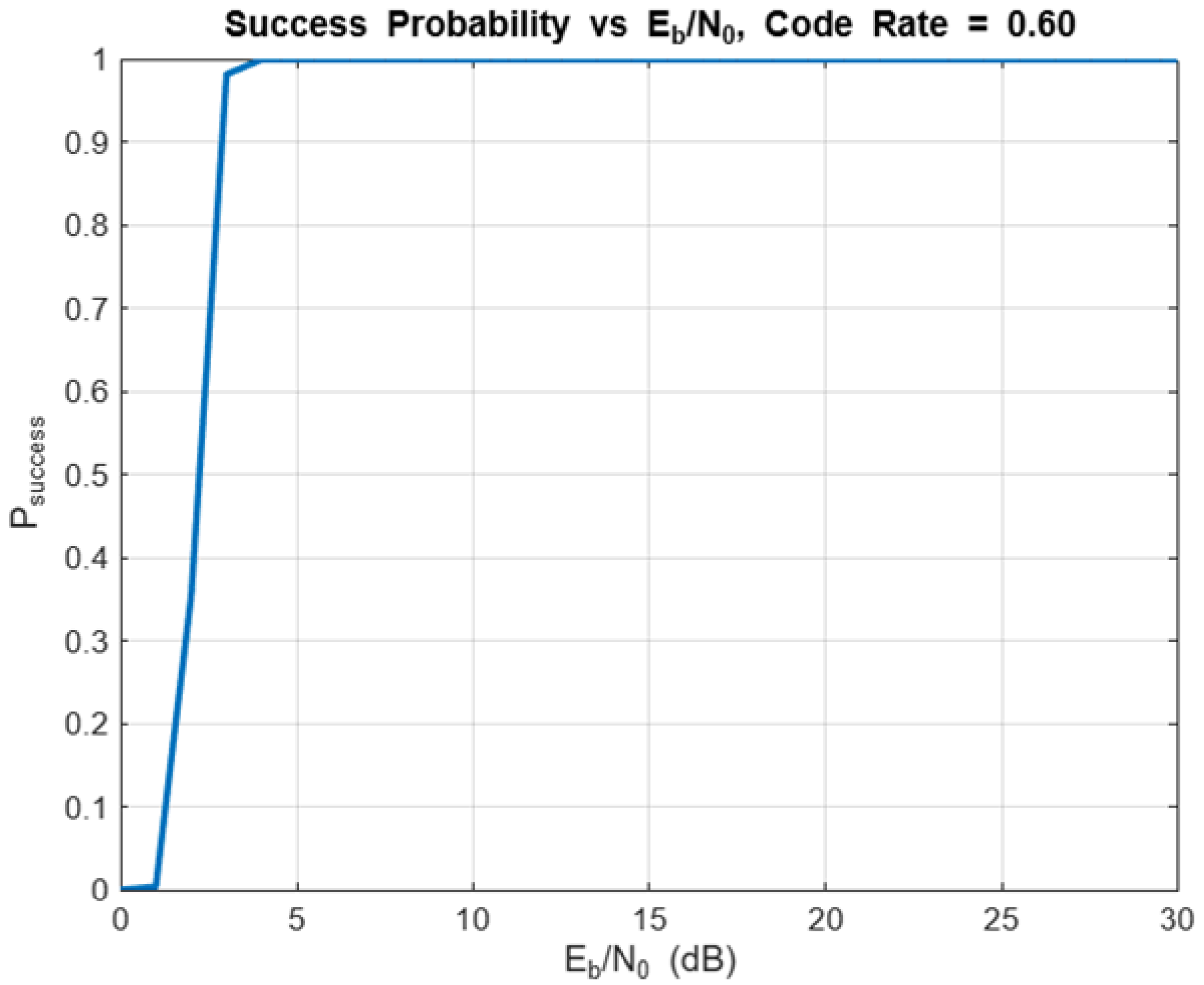
P_success vs E_b/N_0 results for soft decoding



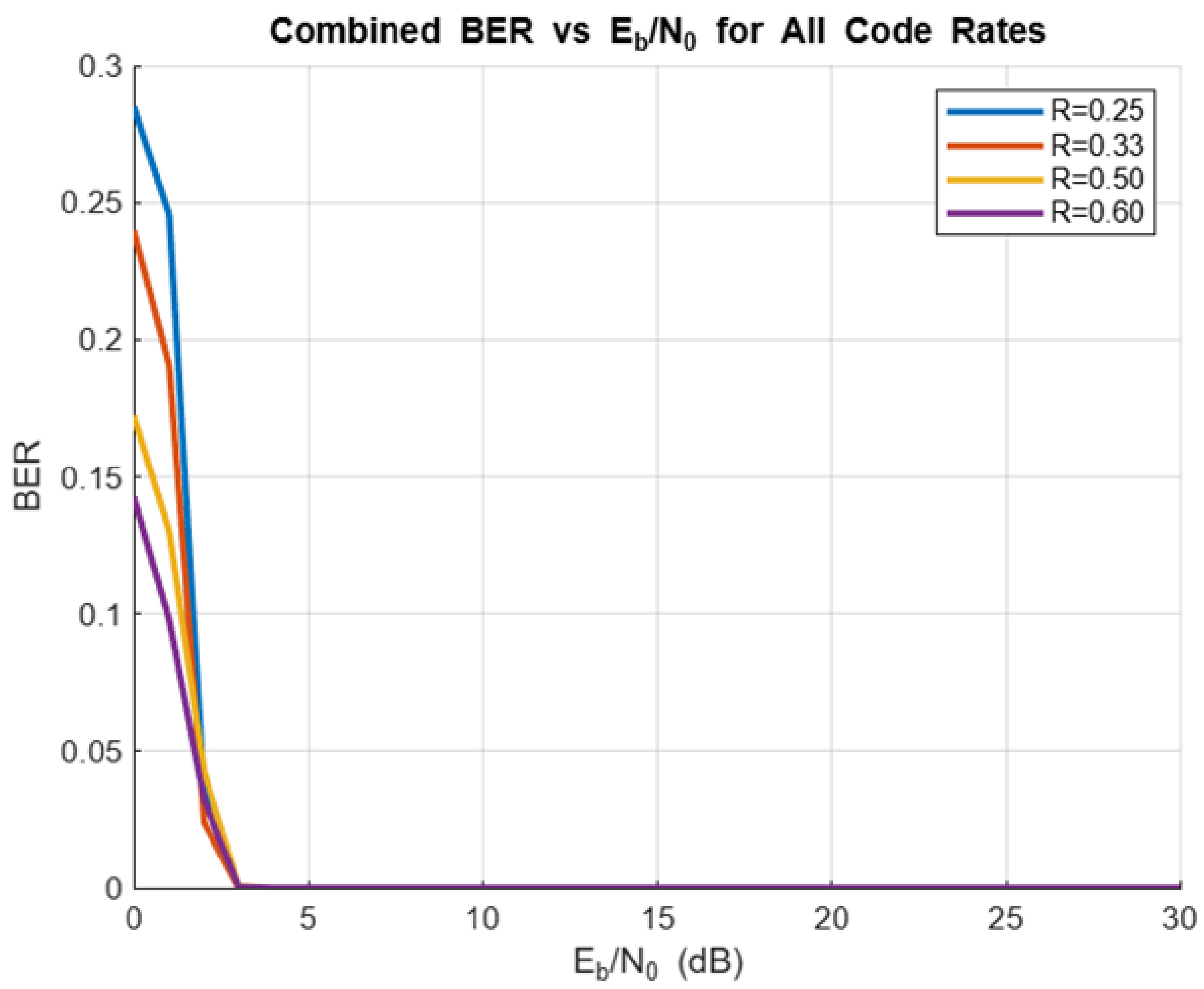


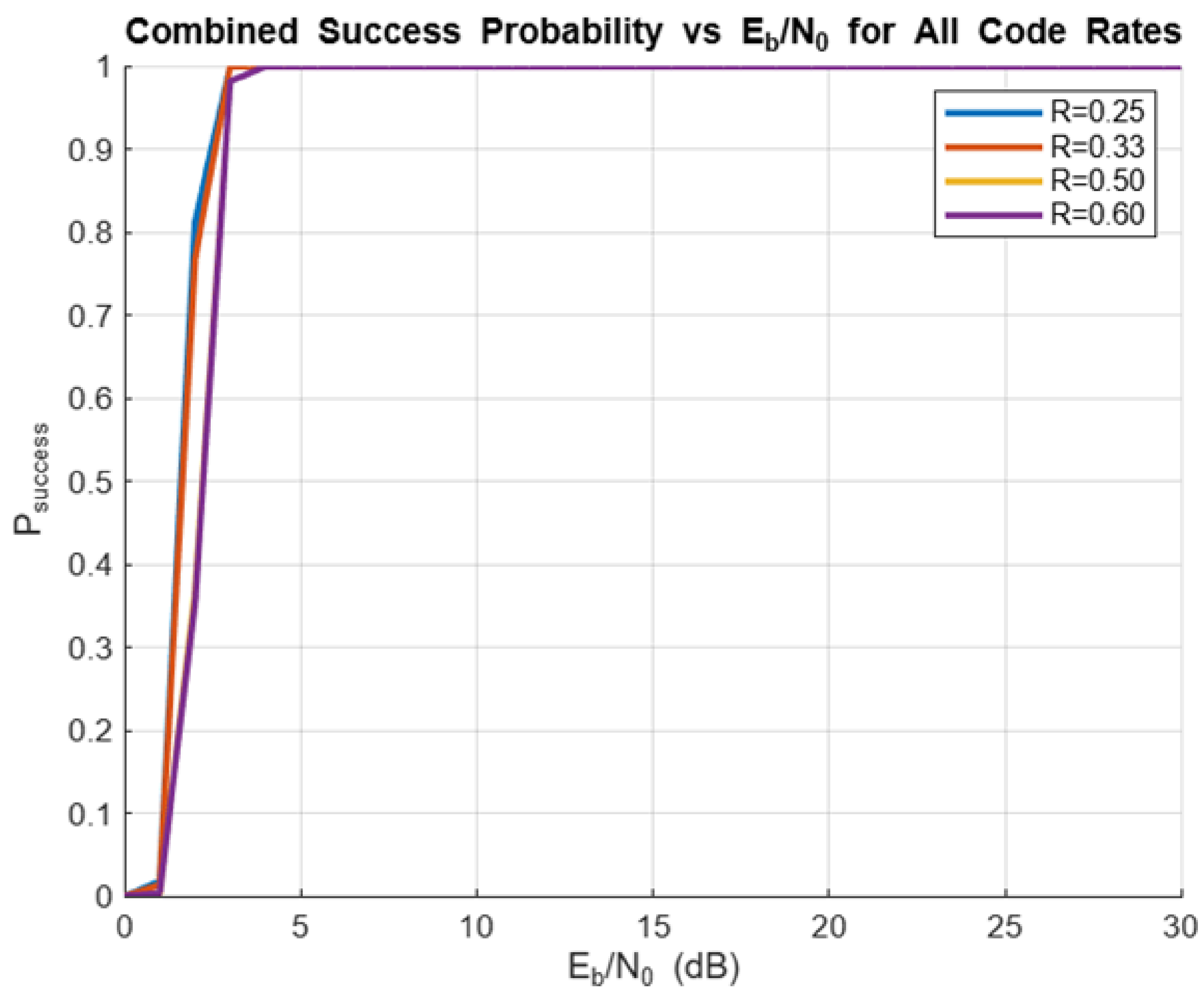
P_success vs E_b/N_0 results for soft decoding



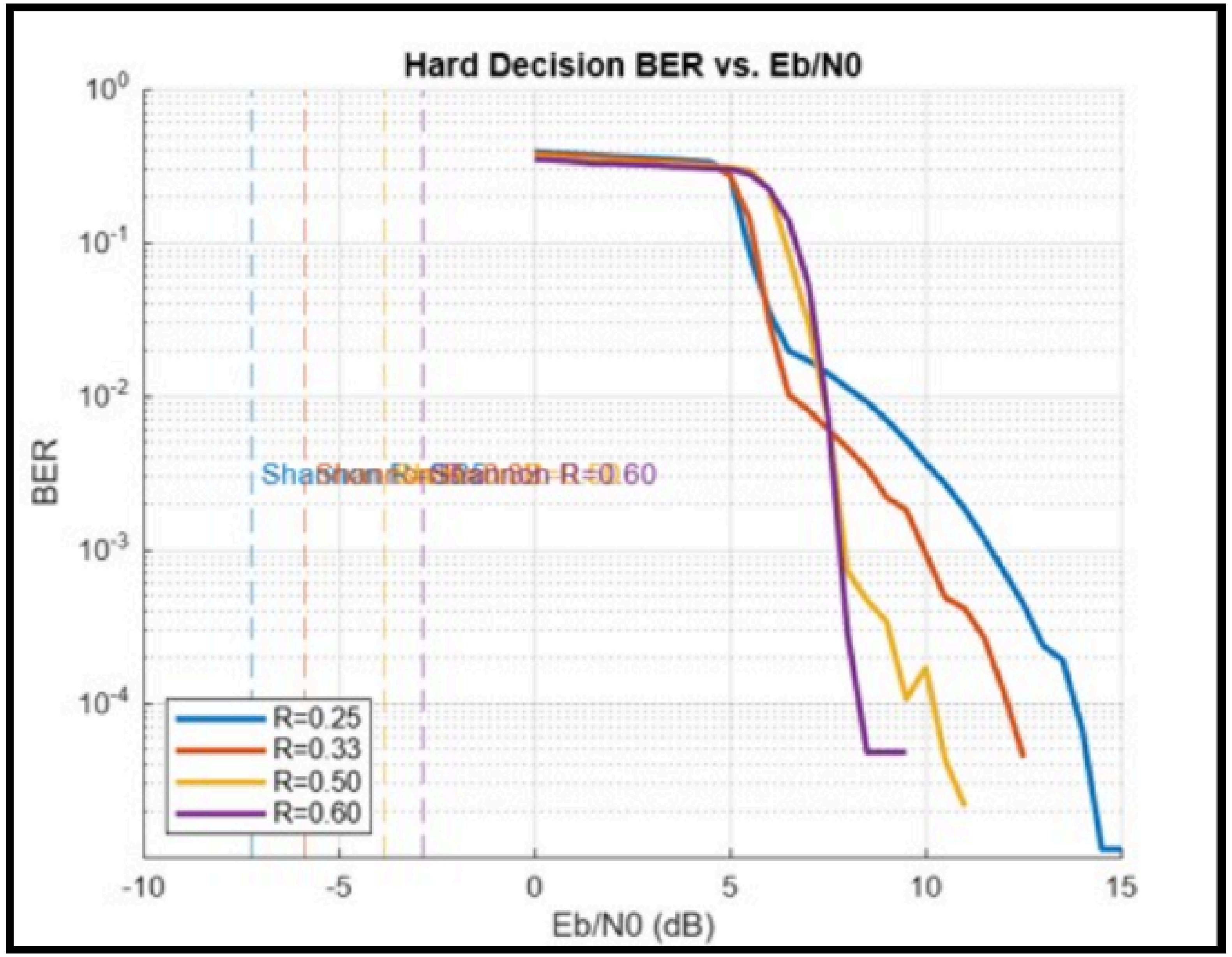


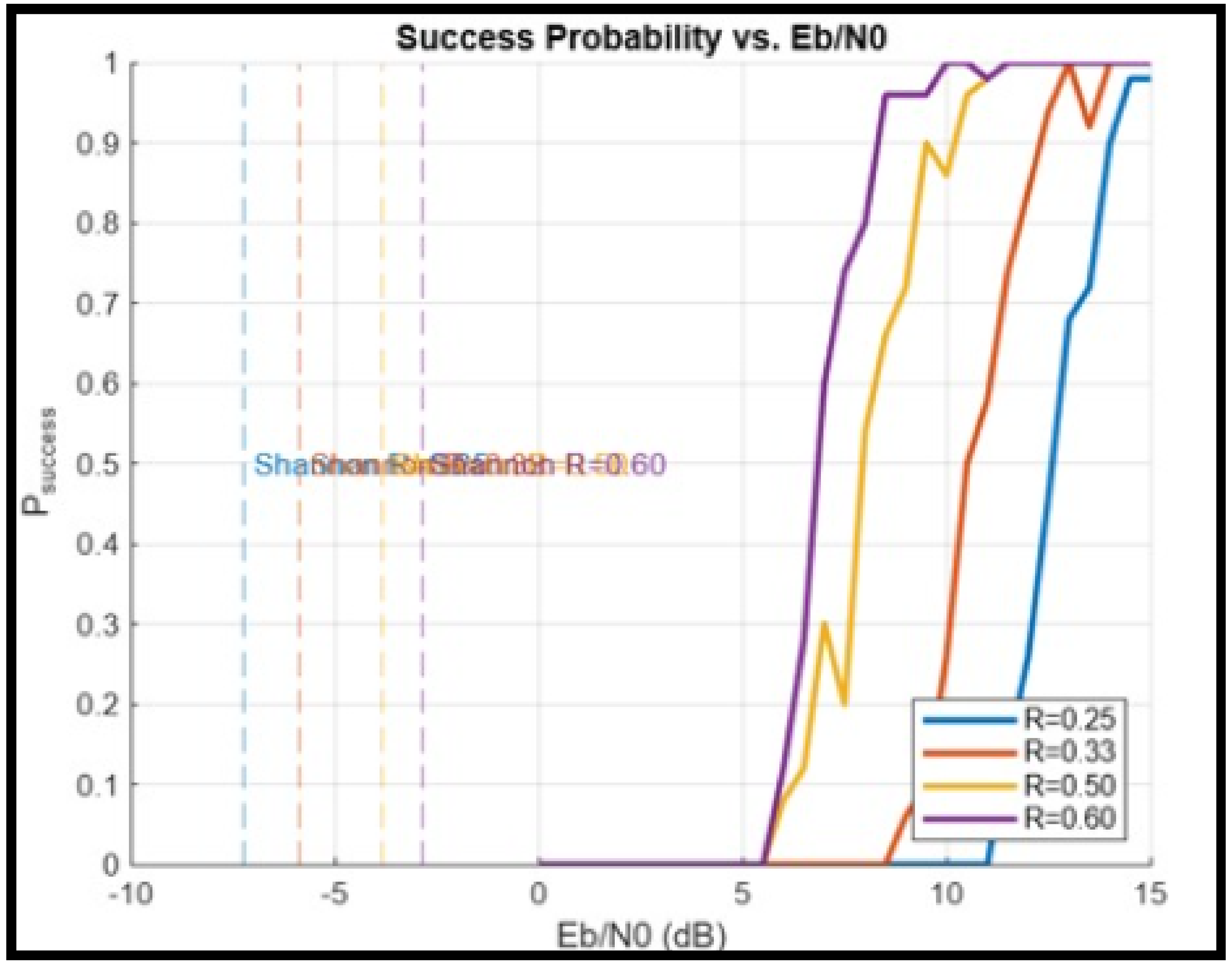
Combined graphs for all code rates(Soft decoding)





Shannon limit(Graph)





Team Members :

1. Prince Chovatiya	202301067	Prince
2. Rishank Dudhat	202301068	Rishank
3. Yashasvi Jadav	202301069	[illegible]
4. Heet Shah	202301070	[illegible]
5. Siddharth Rambhia	202301072	
6. Meet Jain	202301073	[illegible]
7. Kavish Patel	202301074	K. R. Patel
8. Tirth Kanani	202301075	Kanani
9. Aaditya Pala	202301076	A. V. Pala
10. Bhensadadia Happy	202301077	[illegible]

Citation

- CT216. Introduction to Communication Systems: Topic 1 Channel Coding. DA-IICT, Winter 2025: Prof. Yash Vasavada.

- Bhavsar, Namrata & Vala, Brijesh. (2014). Design of Hard and Soft Decision Decoding Algorithms of LDPC. International Journal of Computer Applications. 90. 10.5120/15803-4653.

- Reduced Complexity of LDPC Codes using Hard Decision Decoder Allu Swamy Naidu, Appala Naidu Tentu, and Ajeet Singh

- Andrew Thangaraj, NPTEL-IITM,2019 Lectures

- APA Style:Wiesmayr, R., Nonaca, D., Dick, C., & Studer, C. (2024). Optimizing Puncturing Patterns of 5G NR LDPC Codes for Few-Iteration Decoding. arXiv.

- Wiesmayr, R., Nonaca, D., Dick, C., & Studer, C. (2024). Optimizing Puncturing Patterns of 5G NR LDPC Codes for Few-Iteration Decoding

- LDPC — low density parity check code, EventHelix

- Implementation of Low-Density Parity-Check codes for 5G NR shared channels LIFANG WANG

- Andrew Thangaraj. Ldpc and polar codes in the 5g standard.