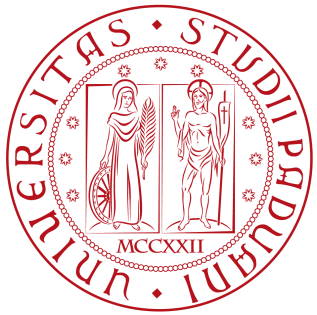
Turbo Codes for Deep Space Communications: CCSDS 131.0-B-2 standard implementation

Final project for the Channel Coding course

Gianluca Marcon
University of Padova
gianluca.marcon.1@studenti.unipd.it

July 27, 2017



DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

Standard specifications

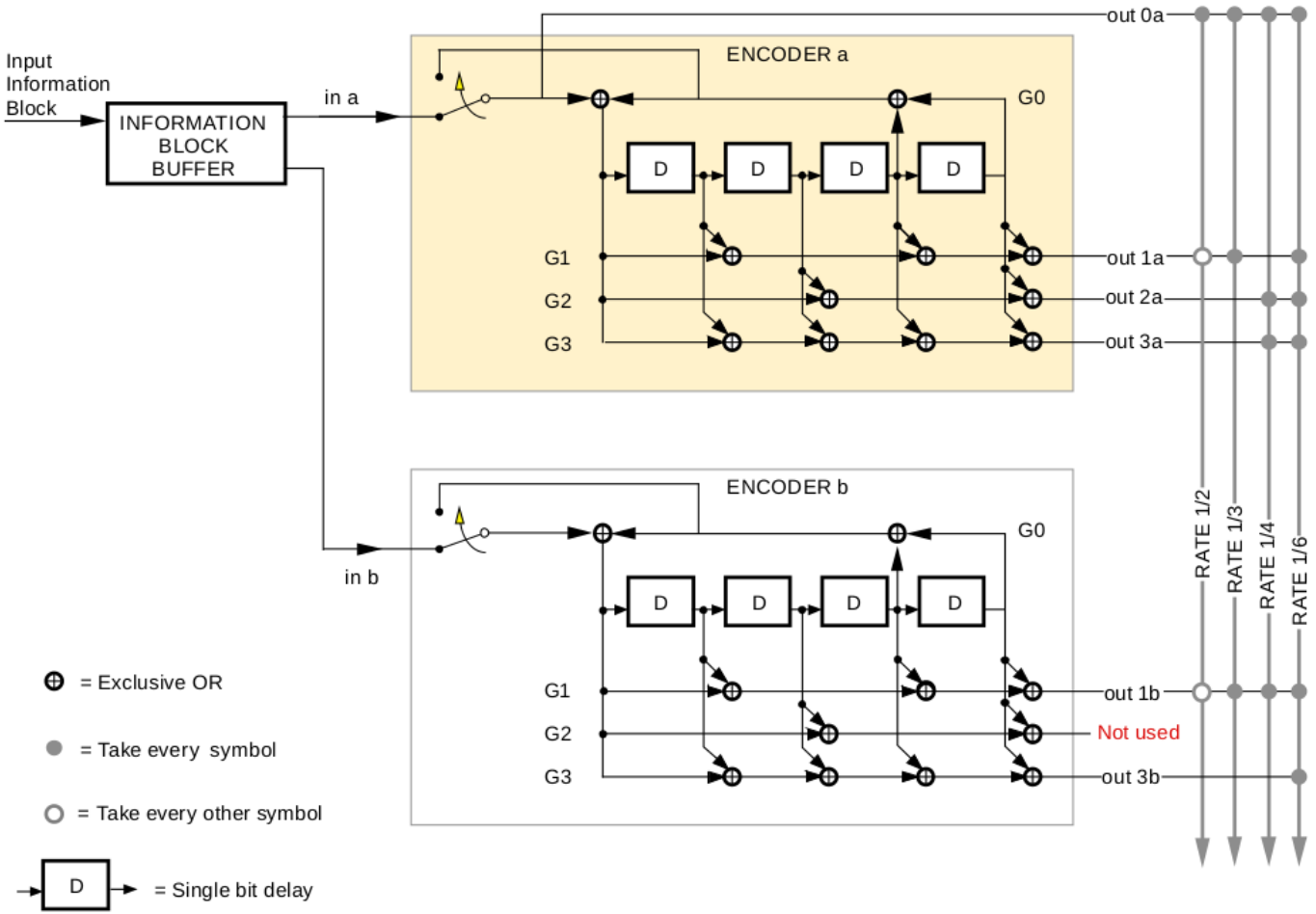
The standard specifies different input packet lengths k

- 1784
- 3568
- 7136
- 8920

...and different code rates R

- $1/2$
- $1/3$
- $1/4$
- $1/6$

Encoder structure

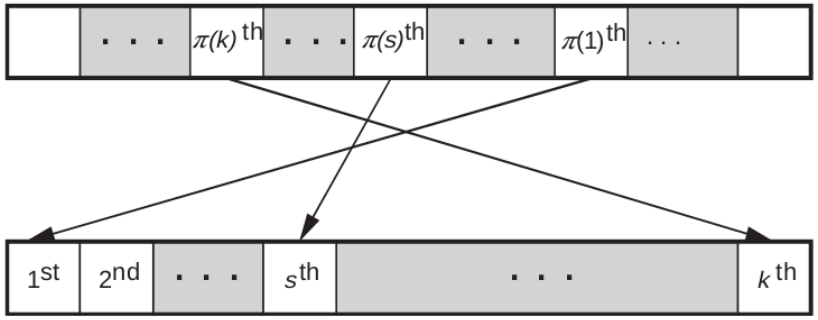


Convolutional codes defined through forward and backward connection vectors

Example: defining a code in C

```
// define first code
int N_components = 2;
char *forward[N_components];
forward[0] = "10011";
forward[1] = "10101";

char *backward = "0011";

t_convcode code = convcode_initialize(forward, backward,
                                      N_components);

t_turbocode turbo = turbo_initialize(code, code, pi,
                                     info_length);
```

Interleaver



i-th bit of the interleaved packet is the $\pi(i)$-th bit of the original packet

Input length k	$k_1 \times k_2 \times k_3$
1784	$8 \times 223 \times 1$
3568	$8 \times 223 \times 2$
7136	$8 \times 223 \times 4$
8920	$8 \times 223 \times 5$

Building the interleaver

$$p = \begin{bmatrix} 31 & 37 & 43 & 47 & 53 & 59 & 61 & 67 \end{bmatrix}$$

for $s = 1$ to k do

$$m = (s-1) \mod 2$$

$$i = \text{floor}\,[(s-1)/(2k_2)]$$

$$j = \text{floor}\,[(s-1)/2] - ik_2$$

$$t = (19i+1) \mod (k_1/2)$$

$$q = t \mod 8 + 1$$

$$c = (p_q j + 21m) \mod k_2$$

$$\pi(s) = 2(t + ck_1/2 + 1) - m$$

end for

Decoding

- BCJR (in log domain) on upper and lower code
- scheduling as seen in class
- number of iterations is tuned accordingly
- puncturing is applied at reception for an easier implementation $\hat{r}[i] = r[i] \cdot p[i], 1 \leq i \leq (k+4)/R$

insert picture of MP decoding (from slides) and explain why puncturing is done like that

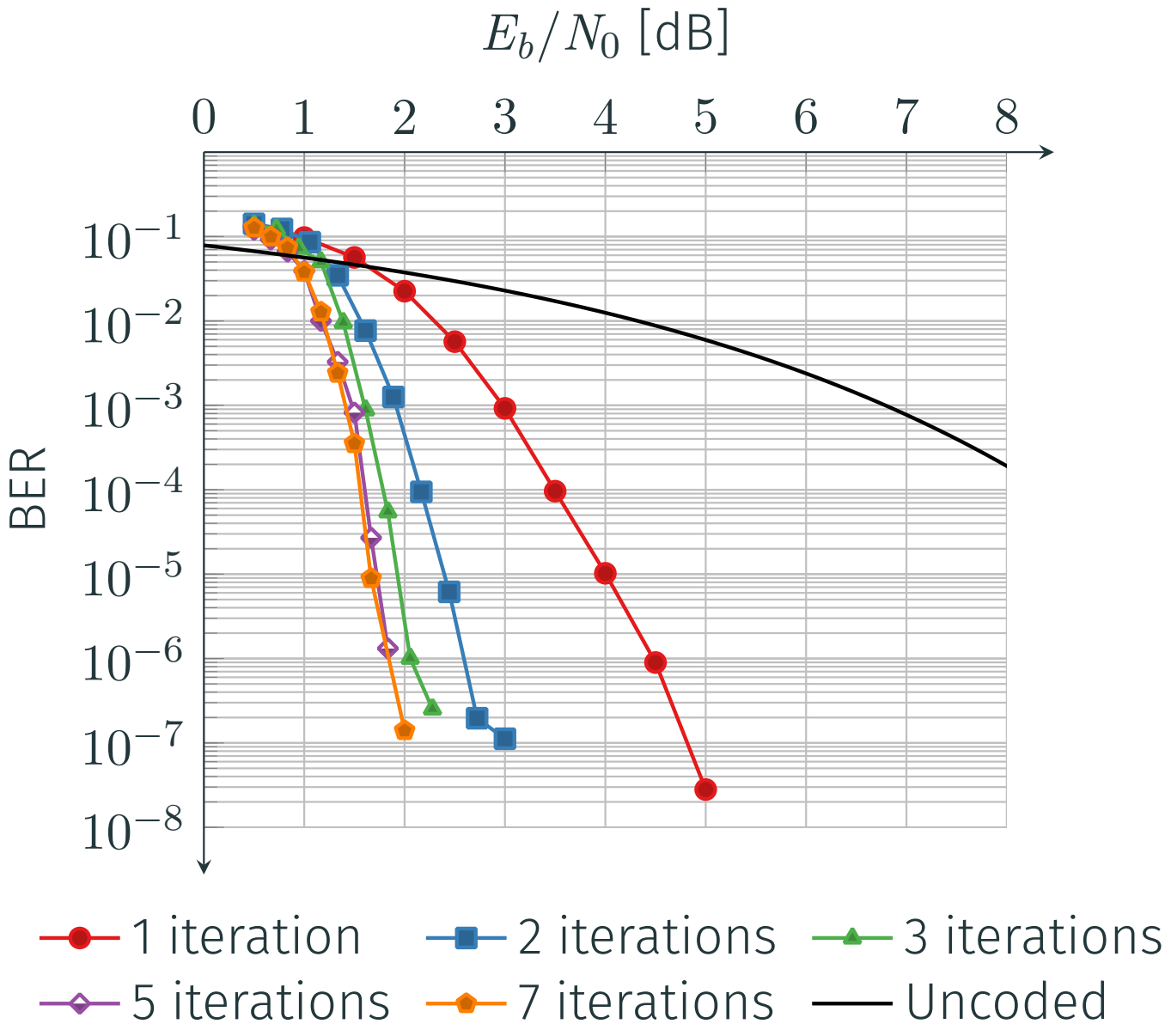
Example: defining a code in C

```
int *decoded = NULL;
for (int i = 0; i < iterations; i++) {

    // run BCJR on upper code
    convcode_extrinsic(streams[0], lengths[0],
                       &messages, code.upper_code,
                       noise_variance, 0);
    // apply interleaver
    message_interleave(&messages, code);

    // run BCJR on lower code
    decoded = convcode_extrinsic(streams[1], lengths[1],
                       &messages, code.lower_code,
                       noise_variance,
                       i == (iterations - 1));
    // deinterleave
    message_deinterleave(&messages, code);
}
```

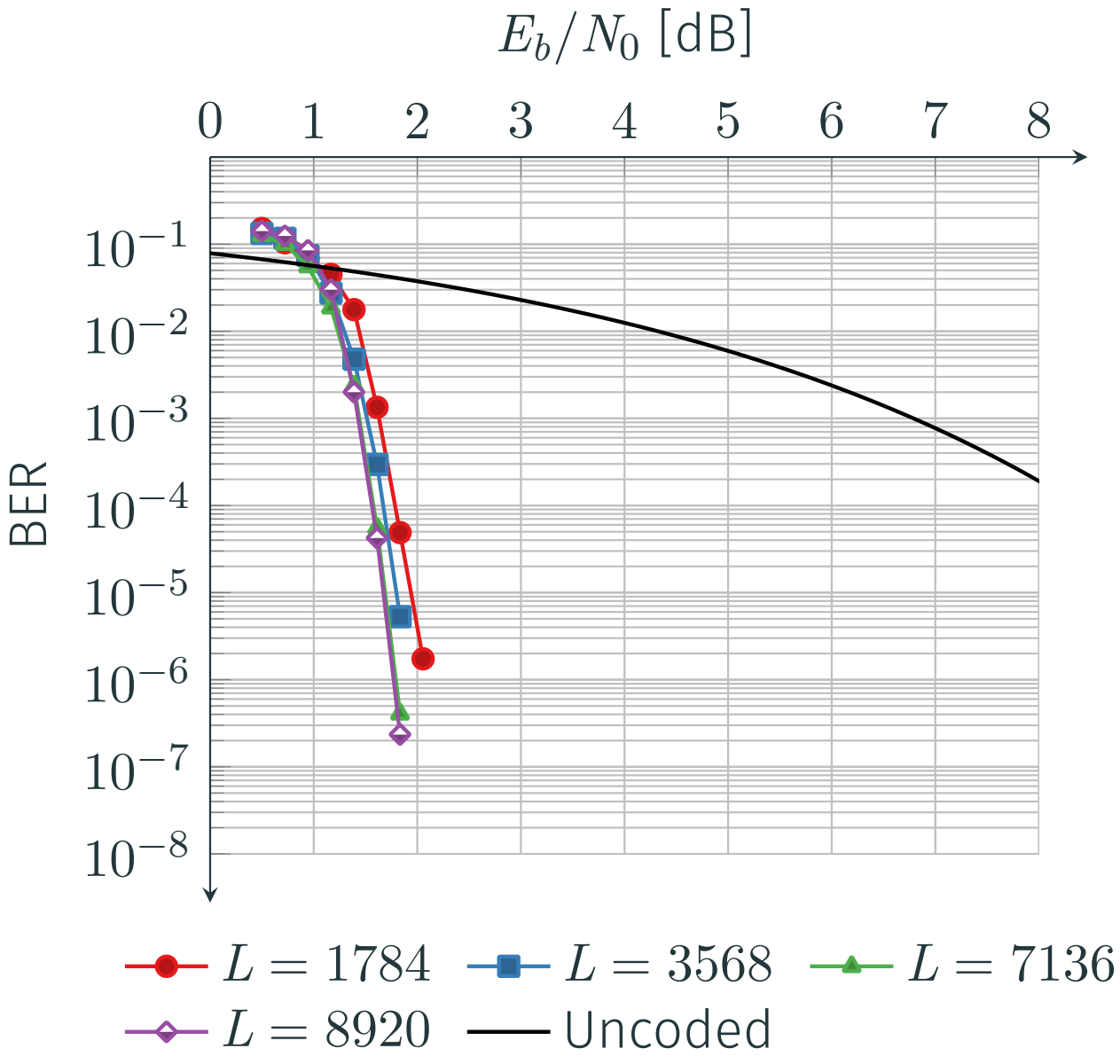
Simulator options

```
gianluca@ursula ~/deepspace-turbo/bin  master ● ./deepspace_turbo --help
-y / --skip-confirm
       skip confirmation dialog after summarizing simulation parameters. Useful when automating simulations.

        -h / --help
       print this help dialog.

-o / --output FILENAME
       save results in a comma-separated format in FILENAME. If this argument is not used, the results will be saved in a file named with the current date and time.

-c / --packet-count INTEGER
       set the number of packets to encode/decode. INTEGER can be given in exponential notation i.e. 1e4 for 10000 packets.

-C / --cores INTEGER
       set the number of CPU cores to use.

-k / --multiplier INT
       set the input packet length through the following  formula: packet-length = 223 * 8 * multiplier

-l / --packet-length INTEGER
       set the number of information bits in a packet. Exponential notation can be used.

-m / --min-SNR FLOAT
       set the lower extreme of the SNR range to test.

-M / --max-SNR FLOAT
       set the upper extreme of the SNR range to test.

-n / --SNR-points INTEGER
       set the number of linearly spaced points inside the interval [min-SNR, max-SNR]

   -i / --iterations
       set the number of iterations for the turbo decoding algorithm.
```

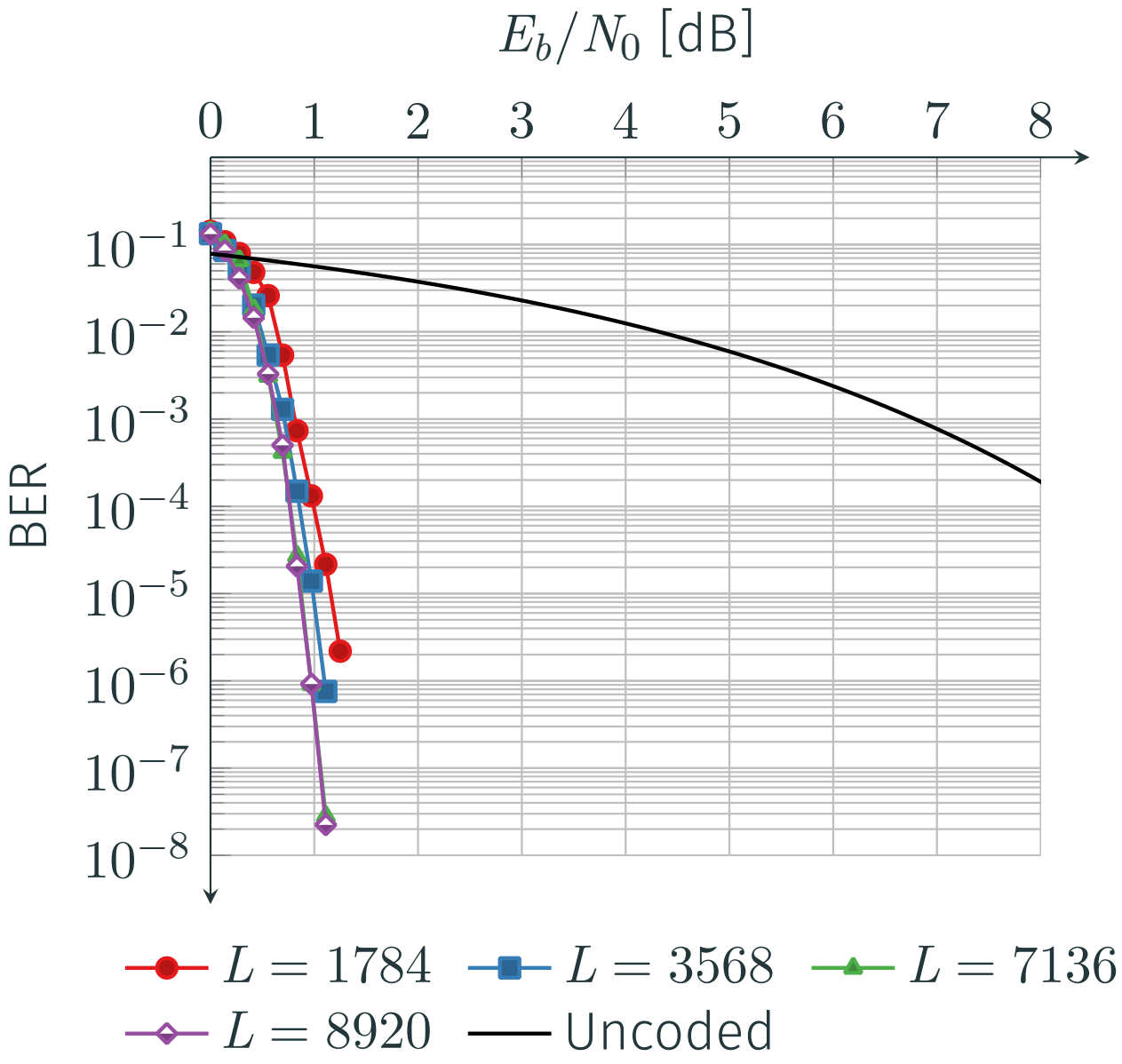
Number of MP iterations



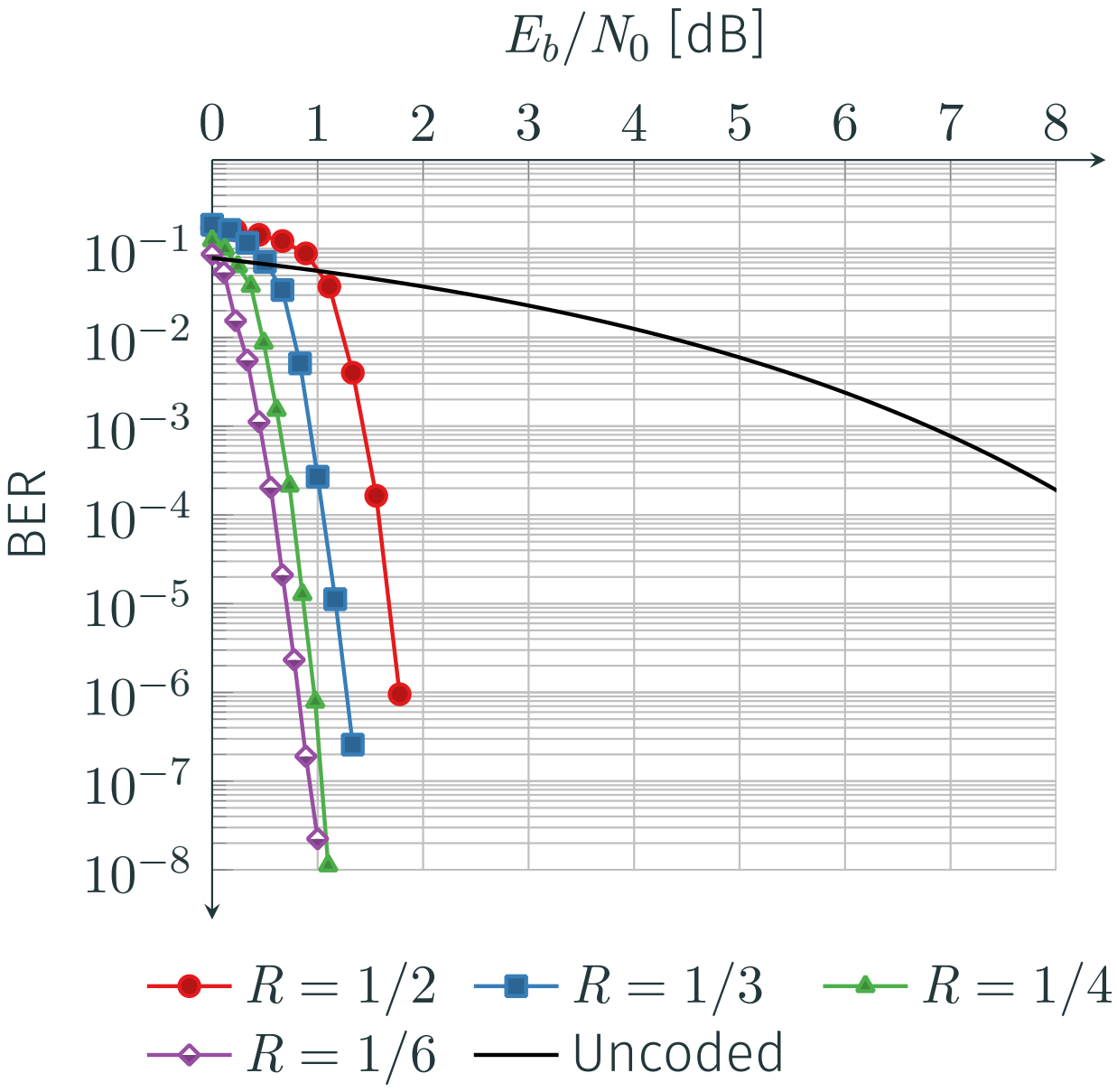
Different packet sizes: $R = 1/2$



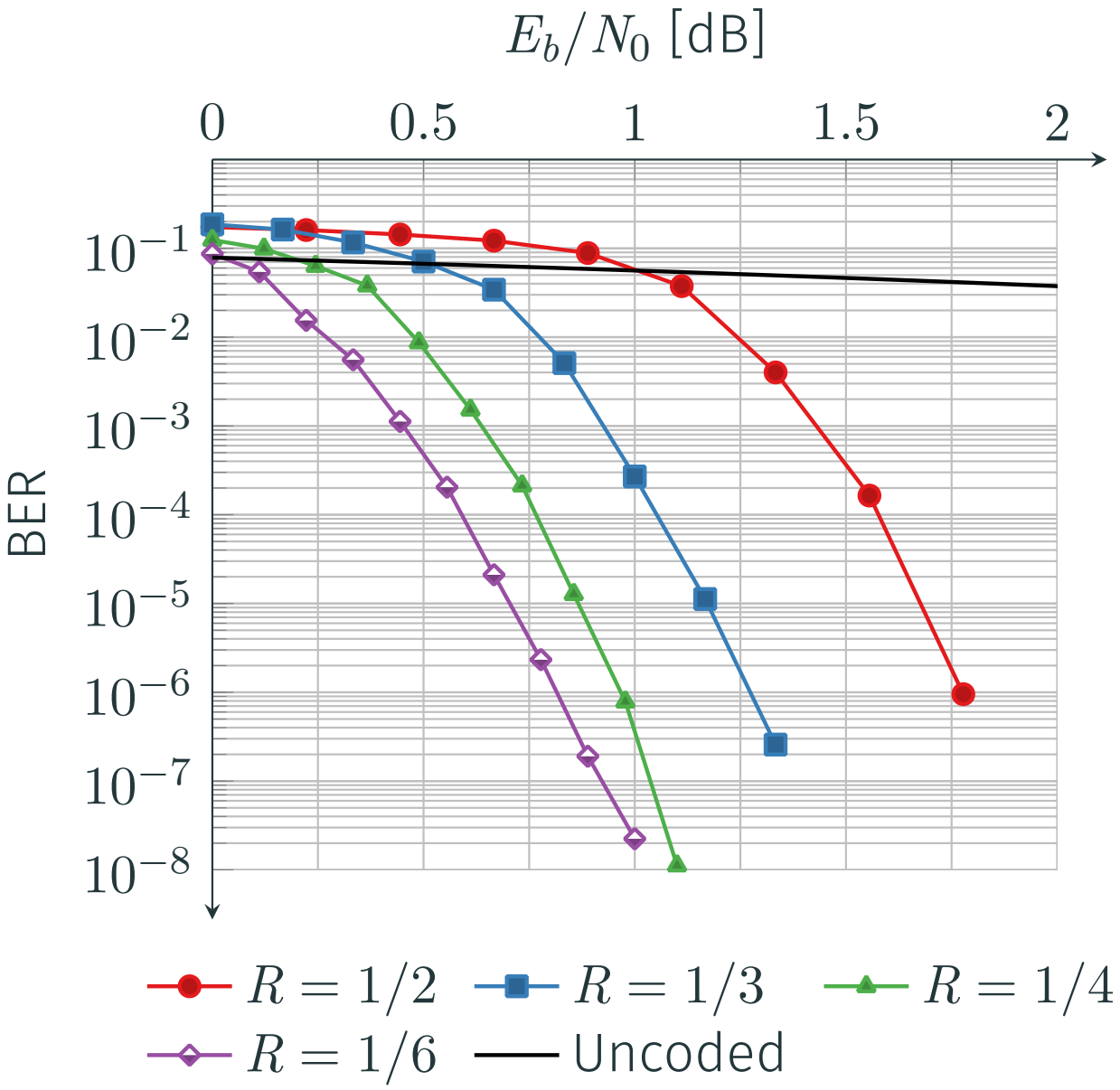
Different packet sizes: $R = 1/4$



Codes comparison: $L = 8920$



Codes comparison: a closer look



Thank you!