CHANNEL CODING - FINAL PROJECT

UNIFIED HIGH-SPEED WIRELINE-BASE HOME NETWORKING TRANSCEIVERS

Access networks – In premises networks

Author:
Lorenzo Gasparollo

Professor:
Tomaso Erseghe

30th March 2017

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Index

- Standard overview
- Encoder
- Decoder
- Rationale Decoder
- BICM
- Results

An introduction

- The **G.hn ITU G9960** is a standard proposed by the ITU-T.
- **Designed for**: the transmission of data over premises' wiring.
- The standard **defines**:

1. the home network architecture and reference models.
2. the physical layer specification.

International Telecommunication Union

Transmission modes

- Each node of the home-network has a different **profile**.

- The standard refers to two different profiles:
  1. **Low-complexity profile** (L-CP).
  2. **Standard profile** (SP).

- A node of the network is required to support one profile, at minimum.

Profile name	Domain type	Valid bandplans
L-CP	Power-line baseband	25 MHz
SP	Power-line baseband	50 MHz, 100 MHz
	Telephone-line baseband	50 MHz, 100 MHz
	Coax baseband	50 MHz, 100 MHz
	Coax RF	50 MHz, 100 MHz, 200 MHz

Transmission modes

- The parity check matrix *H* of size *(N-K) x N* will have:

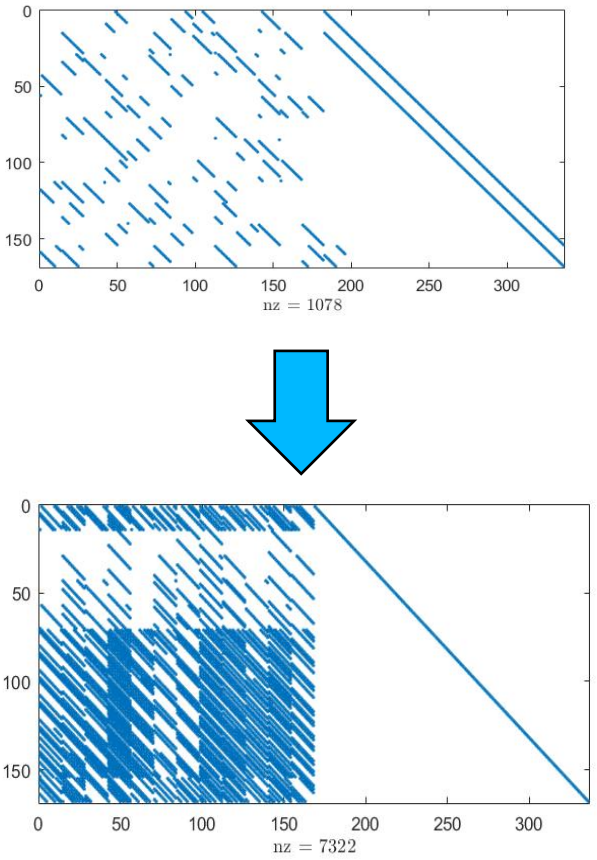
  - Rate $^1/_2$: *N* = 1920 (*LCP,SP*), 8640 (*SP*).
  - Rate $^2/_3$: *N* = 1440, 6480 (*SP*)
  - Rate $^5/_6$: *N* = 1152, 5184 (*SP*)

- Rate $^{16}/_{18}$, $^{20}/_{21}$ are obtained puncturing the code with rate $^5/_6$, through different puncturing patterns.

Profile name	FEC rate	FEC block size
L-CP	$^1/_2$	120 bytes (Payload)
SP	$^1/_2, ^2/_3, ^5/_6$ $^{16}/_{18}, ^{20}/_{21}$	120 and 540 bytes (Payload)

Encoder

1. Given H, get its systematic form H_{sys} .



Encoder

1. Given H, get its systematic form H_{sys} .

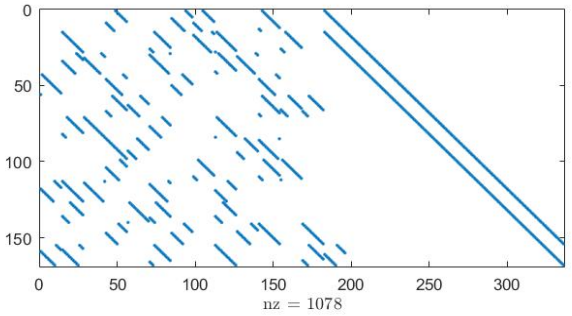
2. Encoding Procedure for a single word:

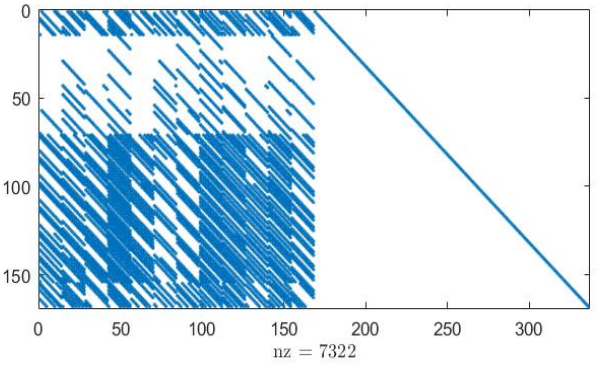
   ✓ The *K* information bits of *u* are directly copied to the codeword *v*.

   ✓ Let A be the submatrix obtained considering *N-K* rows and the first k columns of H_{sys} ; then, the *N–K* parity-check bits are computed as A * u.

   ✓ The final codeword is *v* = [u|A u]

- Preprocessing is done only once.





Decoder – a first (naif) approach

- All the minsum/ sumproduct algorithm have been written in c, and invoked via mex functions.

- There are at least three matrices as big as the H matrix:

Decoder – a first (naif) approach

- All the minsum/ sumproduct algorithm have been written in c, and invoked via mex functions.

- There are at least three matrices as big as the H matrix:

Unfeasible in terms of:
- Memory allocation.
- Computational efforts.

Decoder – a first (naif) approach

- All the minsum/ sumproduct algorithm have been written in c, and invoked via mex functions.

- There are at least three matrices as big as the H matrix:

Unfeasible in terms of:
- Memory allocation.
- Computational efforts.

Decoder – a first (naif) approach

- All the minsum/ sumproduct algorithm have been written in c, and invoked via mex functions.

- There are at least three matrices as big as the H matrix:

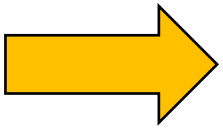
Unfeasible in terms of:
- Memory allocation.
- Computational efforts.

Decoder – Rationale

- Preprocess the matrix H in such a way to get all the **useful patterns** from it.

Decoder – Rationale

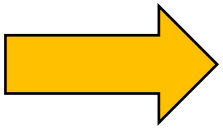
- Preprocess the matrix H in such a way to get all the **useful patterns** from it.



Do it only once.

Decoder – Rationale

- Preprocess the matrix H in such a way to get all the **useful patterns** from it.



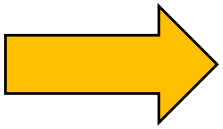
Do it only once.

- From H we get:

$$\underset{v\to c}{\Psi} \quad \underset{c\to v}{\Psi} \quad \underset{v\to c}{\prod} \quad \underset{c\to v}{\prod} \quad \{l_i\}_{i=1}^{N-K} \quad \{j_i\}_{i=1}^{N}$$

Decoder – Rationale

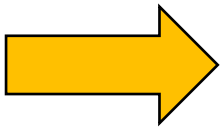
- Preprocess the matrix H in such a way to get all the patterns from it.



Do it only once.

- From H we get:

$$\underset{v\to c}{\Psi} \quad \underset{c\to v}{\Psi} \quad \underset{v\to c}{\prod} \quad \underset{c\to v}{\prod} \quad \{l_i\}_{i=1}^{N-K} \quad \{j_i\}_{i=1}^{N}$$



No need to evaluate directly H in the message passing algorithm for the computation of LLRs.

Updates as a linear map

- Let $\underset{c\to v}{\Psi}$ and $\underset{v\to c}{\Psi}$ be the check update matrix and the variable update matrix, respectively:

$$\underset{c\to v}{\Psi} = \begin{bmatrix} I^C_{l_1} & & 0 \\ & \ddots & \\ 0 & & I^C_{l_{N-K}} \end{bmatrix} \qquad \underset{v\to c}{\Psi} = \begin{bmatrix} I^C_{j_1} & & 0 \\ & \ddots & \\ 0 & & I^C_{j_N} \end{bmatrix}$$

- I^C_s denotes the complement of the Identity matrix of size s.
- $\{l_i\}_{i=1}^{N-K}$ and $\{j_i\}_{i=1}^{N}$ are the number of ones in each row and column respectively.

LLRs notation

- Let's define $\underset{c\to v}{LLR}, \underset{v\to c}{LLR} \in \mathbb{F}_2^U$ as the LLRs vectors, with U the total numbers of ones (edges) in the H matrix.

$$\underset{c\to v}{LLR} = \begin{bmatrix} \\ \vdots \\ \\ \\ \vdots \\ \end{bmatrix} \begin{matrix} \} j_1 \\ \\ \\ \\ \} j_N \end{matrix}$$

$$\underset{v\to c}{LLR} = \begin{bmatrix} \\ \vdots \\ \\ \\ \vdots \\ \end{bmatrix} \begin{matrix} \} l_1 \\ \\ \\ \\ \} l_{N-K} \end{matrix}$$

Updates in minSum and sumProd

- For both minsum and sumproduct decoder the **variable updates** will have the following expression:

$$\underset{v\to c}{LLR} = \underset{v\to c}{\Psi} \cdot \underset{c\to v}{LLR} + p$$

- In the sum product the **check updates** is:

$$\underset{c\to v}{LLR} = \left(\underset{c\to v}{\Psi} \cdot \Phi\left(\left|\underset{v\to c}{LLR}\right|\right)\right) * \operatorname{sgn}\left(\underset{v\to c}{LLR}\right)$$

- Where $\Phi(x) = \log\left(\frac{e^x+1}{e^x-1}\right)$, $\operatorname{sgn}(\cdot)$ is the function that maps each element of the vector to its corresponding sign and *p* is the vector of prior LLRs.

Permutation matrices

In order to properly compute the LLRs updates, vectors must be properly ordered…

HOW?

Permutation matrices

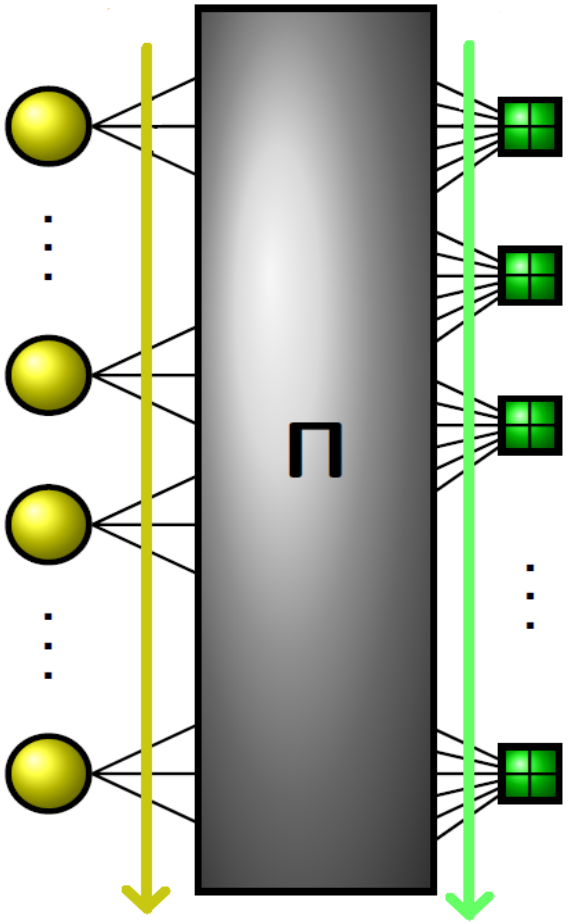
In order to properly compute the LLRs updates, vectors must be properly ordered…

HOW?

We will refer to two different LLR representation**:**

Check representation (CR)

Variable representation (VR)



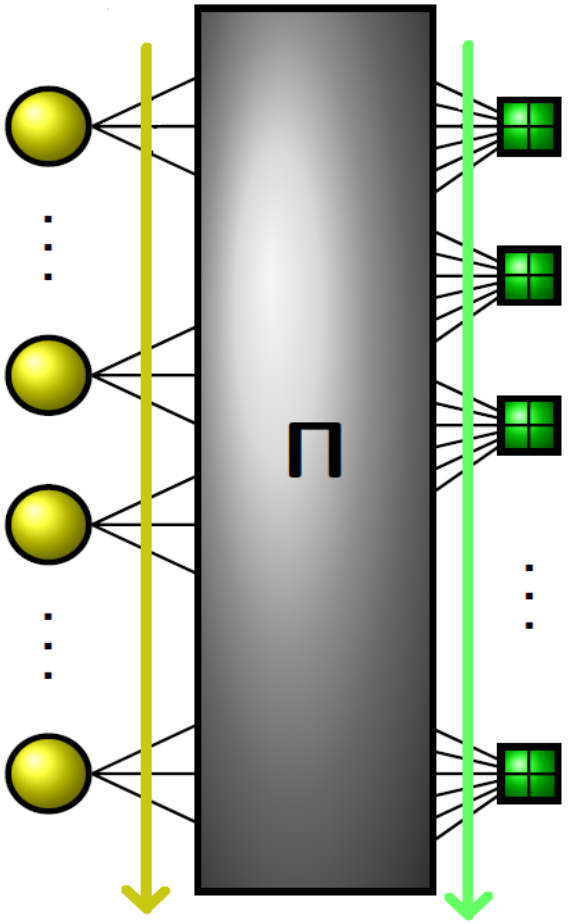
Permutation matrices

CHECK REPRESENTATION (CR)

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

VARIABLE REPRESENTATION (VR)

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$



Permutation matrices (1/2)

- Let $per : D_U \longrightarrow D_U$ be the permutation map that associates to each edge in expressed in the **CR** the corrisponding position in the **VR**.

- $D_U = \{1, \ldots, U\}$, U the number of ones (edges) in the matrix H.

$$\prod_{c \to v} = \begin{bmatrix} e_{per(1)}{}^{\top} \\ \hline \vdots \\ \hline e_{per(U)}{}^{\top} \end{bmatrix}$$

- Where $e_i \in \mathbb{F}_2^U$ denotes the i-th canonical vector in $\mathbb{F}_2^U$.

Permutation matrices (2/2)

- If we think of $\prod_{c \to v}$ as a linear application and we expoit the property of the permutation matrices:

$$\prod_{v \to c} = \prod_{c \to v}^{-1} = \prod_{c \to v}^{\top}$$

- Hence, there is no need to compute directly $\prod_{v \to c}$ since:

$$\prod_{c \to v} \longleftrightarrow \prod_{v \to c}$$

Sum product


```
for it=1:iter
     LLRvc = perMatrVC * LLRvc;
     curSign = sign(LLRvc);
      curSign(curSign == 0) = 1;
      signLLRcv = GetSign(curSign,nOR);
     tmpP = abs(LLRvc);
     tmpP(tmpP < minf) = 1e-10;
     temp = upMatrCV * PhiMap(tmpP);
     LLRcv =  PhiMap(temp).* signLLRcv;
     LLRcv = perMatrCV * LLRcv;
     LLRvc = upMatrVC * LLRcv + LLRprior;
     curIndex = 1;
    for j=1:length(nOC)
         if(vPLLRs(j) + sum(LLRcv(curIndex:nOC(j)+curIndex-1)) < 0)
            u_hat(j) = 1;
         else
            u_hat(j) = 0;
         end
         curIndex = curIndex + nOC(j);
    end
    if( mod(Hreal * u_hat,2) == 0)
        break;
    end
end
```

Get LLRvc in **CR**

Update

Get LLRcv in **VR**

Update

Practical considerations

- Product between sparse matrices is optimized in Matlab. (https://it.mathworks.com/help/matlab/math/sparse-matrix-operations.html)
- Only vectors proportional to the number of ones in the check matrix are used.

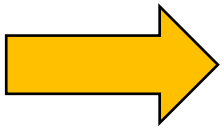
Mex function for specific tasks have been implemented:

- Arrayfunc command of matlab leads to slow computation.
- Mex function **PhiMap()** improves dramatically the performance.
- Search in the minsum is not a bottle neck since elements to be checked are few (consequence to the fact that H is sparse).

ENCODING

- Grouping sequences of b = $\log_2 M$ bits and then mapping them according to the suitable constellation mappers.

- Both **constellation mappers** (4 and 16 symbols) designed with mex function.



Faster than defining a matlab function and then using the arrayfunc function.

COMPUTATION OF THE LLRs FOR DECODER

b0	b1
0	0
0	1
1	0
1	1

ω

Re	Im
-1	-1
1	-1
-1	1
1	1

Given the rx symbol

$$LLR = \ln\left(\frac{P(0,0)+P(0,1)}{P(1,0)+P(1,1)}\right)$$

$b_0 \rightarrow v_0$

b0	b1
0	0
1	0
0	1
1	1

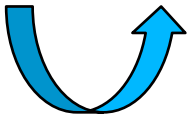
ω

Re	Im
-1	-1
-1	1
1	-1
1	1

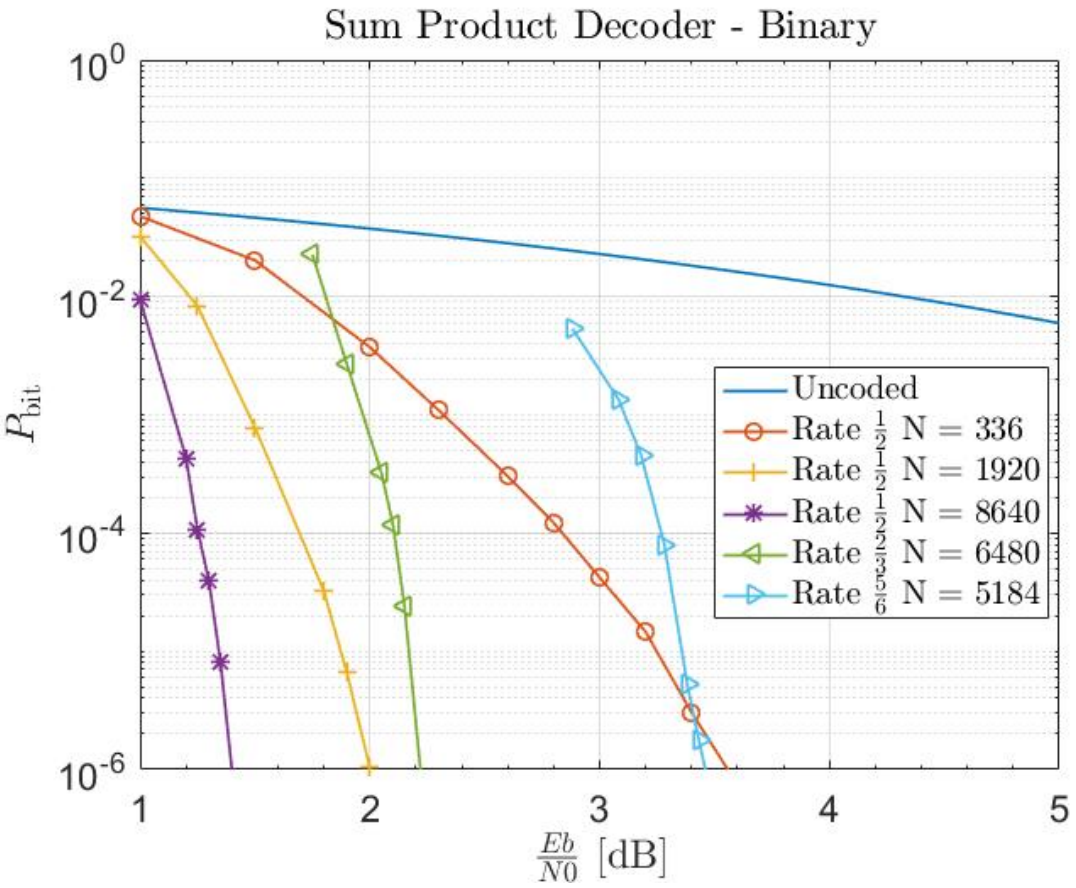
Given the rx symbol

$$LLR = \ln\left(\frac{P(0,0)+P(1,0)}{P(0,1)+P(1,1)}\right)$$

$b_1 \rightarrow v_1$

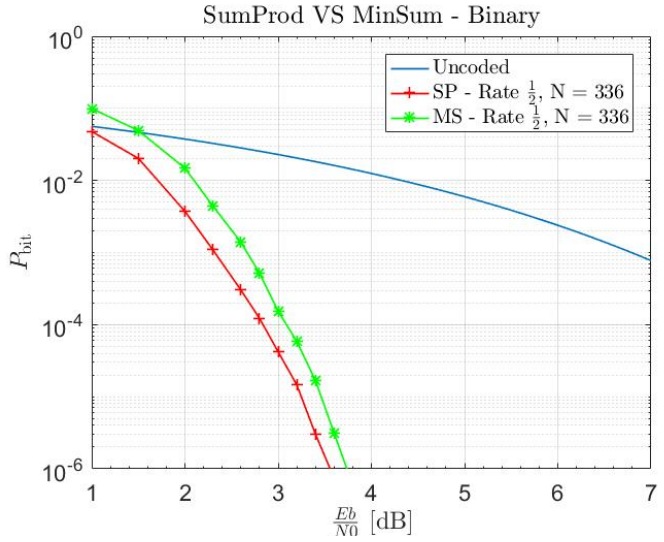


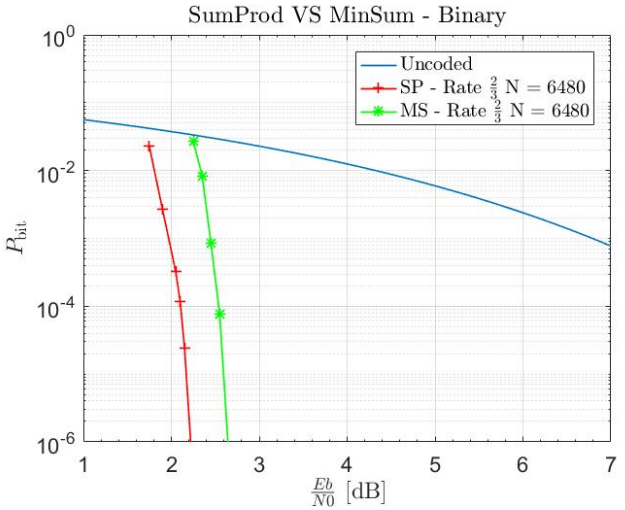
Results

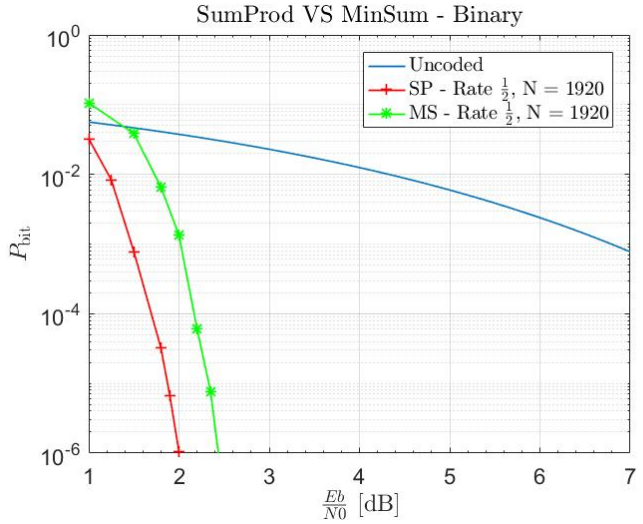


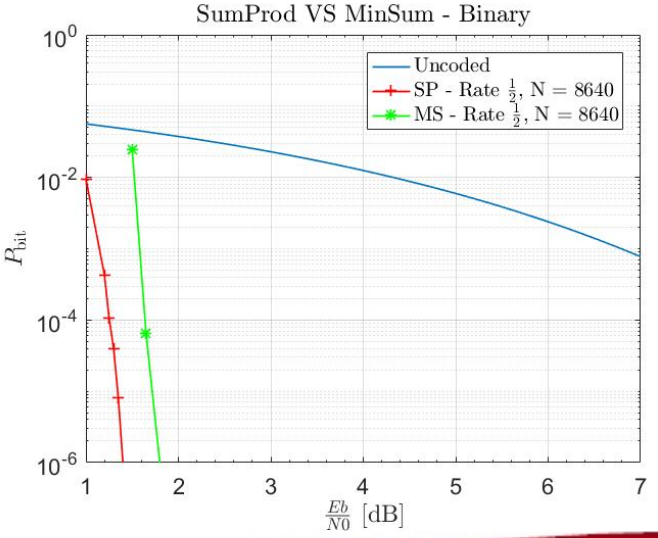
Uncoded case:
Pbit = 10^{-6}
for $\frac{Eb}{N0} \cong$ 10,5 dB

Results – MinSum vs SumProd



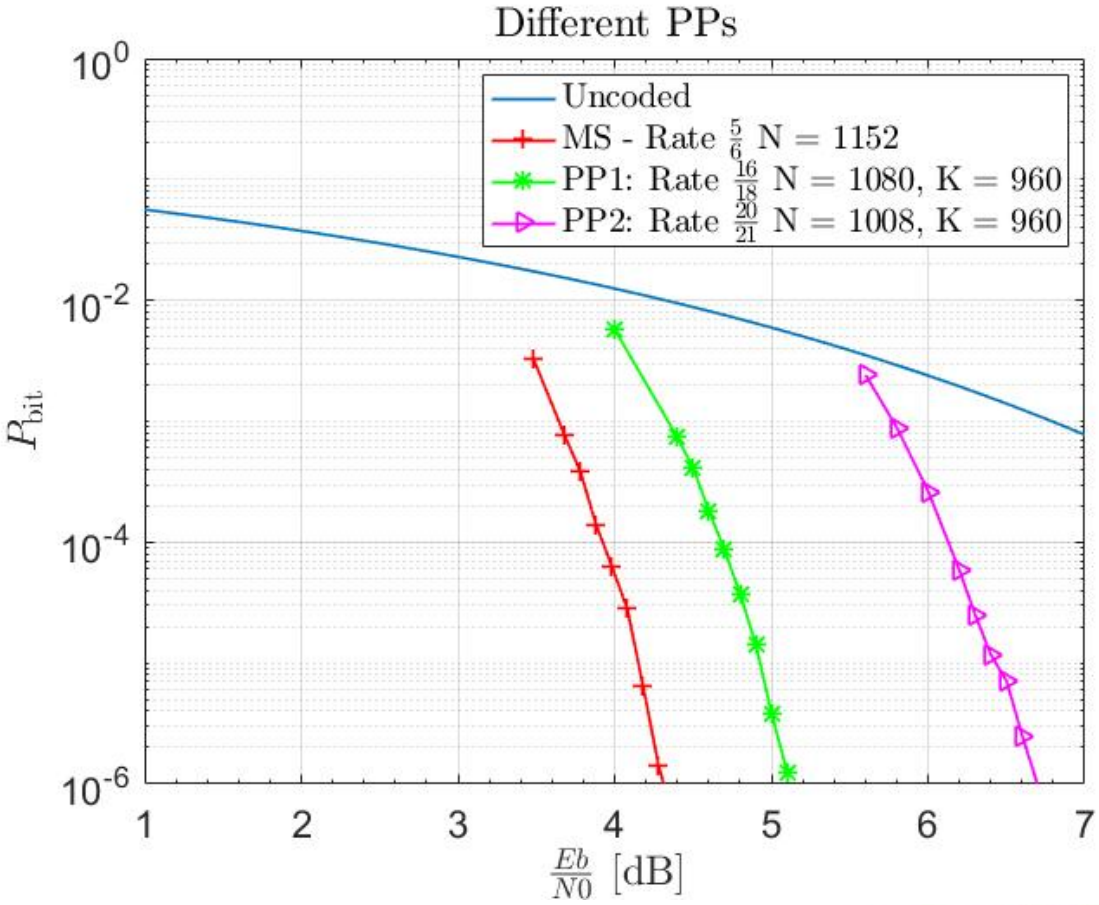






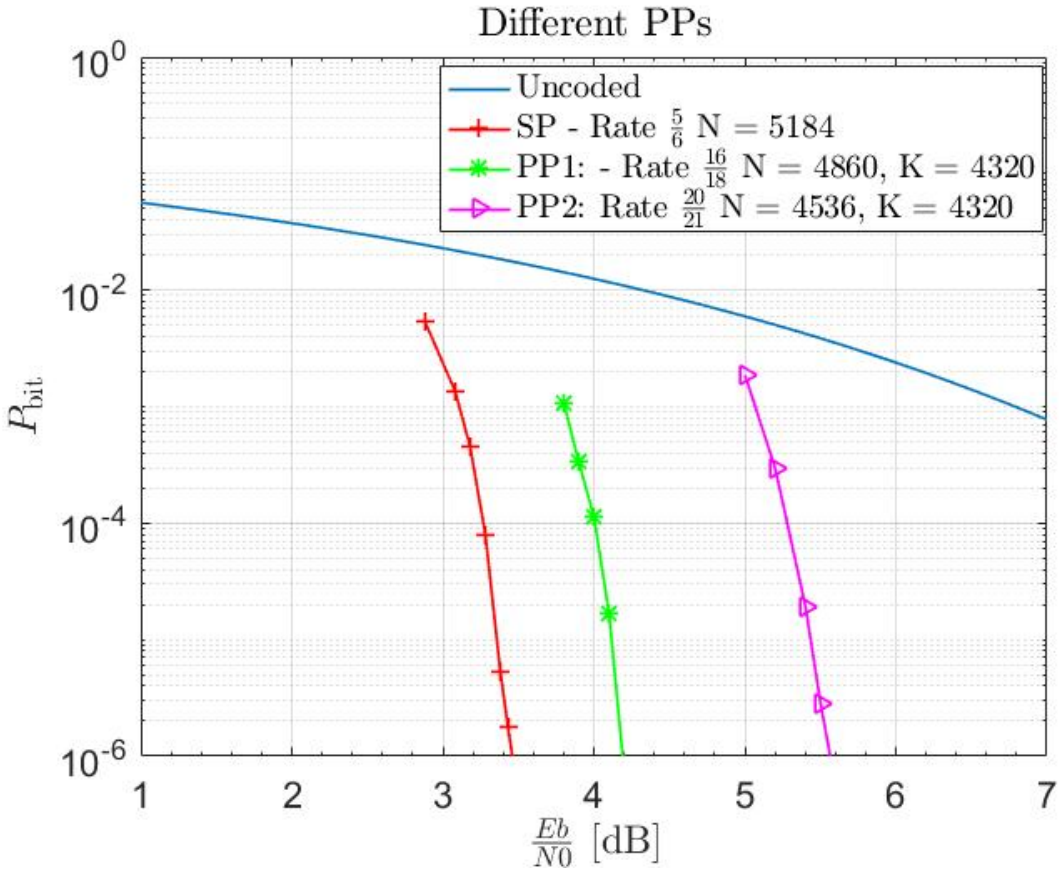
Results – Different Puncturing Patterns

For K = 960:

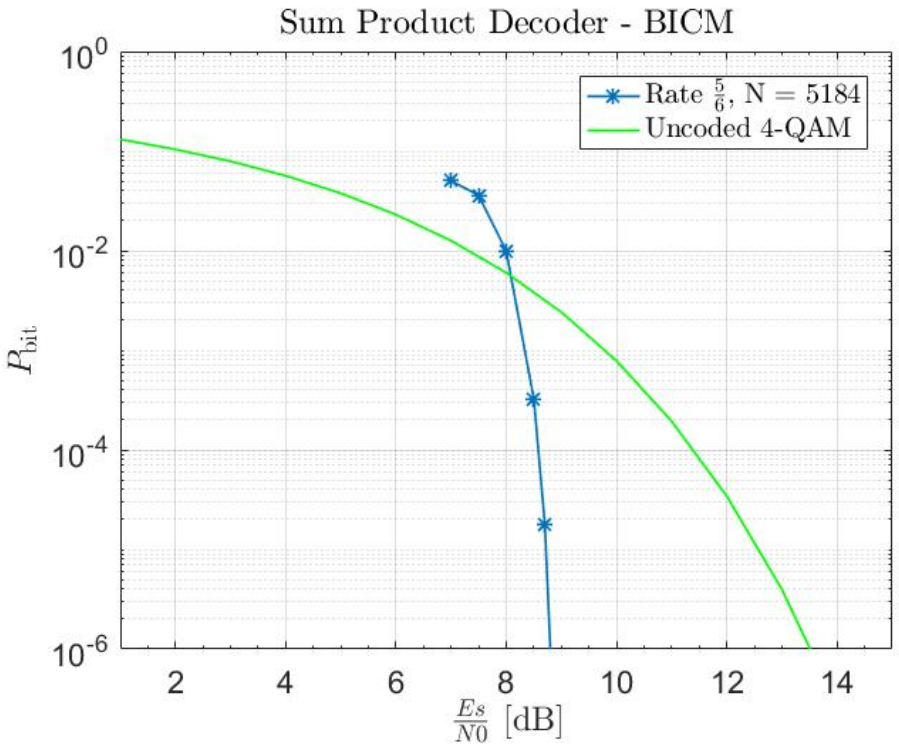


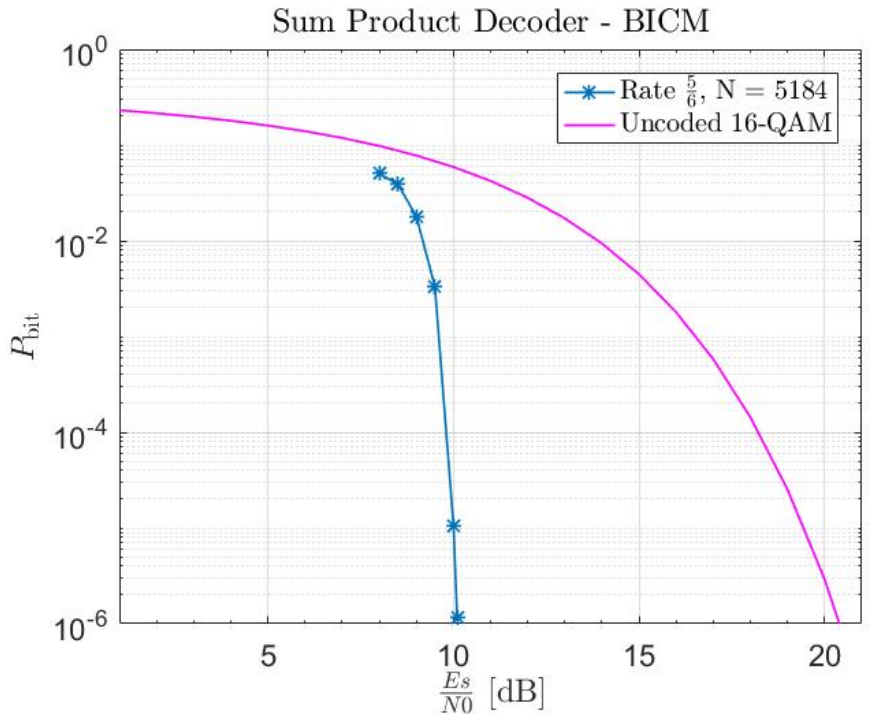
Results - Different Puncturing Patterns

For K = 4320:



Results - BICM





Conclusions and remarks

- Possible extensions/improvements.

Details

- All the images have been created with LaTeX or Matlab.

- The images reported in pag. 8 where taken from two superb books (According to Goodreads): The Pragmatic Programmer and Programming Pearls.

- The image of LDPC decoder have been selected from PDFs of Channel Coding Course (Tomaso Erseghe).