POLITECNICO DI TORINO

Master of Science in Communications Engineering

COMMUNICATION SYSTEMS

REPORT: ASSIGNMENT 2

November 30, 2022

Luca Nepote (s315234)

1 Introduction

In the second assignment of the Master of Science course "Communication Systems" we have analyzed Error Correction Hard with syndrome decoding, Error Correction Soft with minimum euclidian distance, randomizer (LSFR and m-sequence) and Sync Marker Detection.
The assignments' problems have been solved in the Matlab environment.

2 Exercise 1: Hard Correction vs. Soft Correction

In the first part of the assignment we compare two types of error correction, namely "Hard Correction" and "Soft Correction". The goal is to receive a codeword which is equal to the transmitted one, so $C_R = C_T$. This is an ideal goal, which is almost impossible to obtain: however we can make a simplification, such that we want to maximize $P(C_R = C_T)$. Practically, we look for

$$\mathbf{C_R} = argmin_{(\mathbf{c} \in C)} d_H(\mathbf{y}, \mathbf{c}).$$

Doing that, the complexity is too high, so it is better to divide the space in decision regions associated to the codewords

$$D(\mathbf{c}) = \{\mathbf{y} \in H^n : d_H(\mathbf{y}, \mathbf{c}) \leq d_H(\mathbf{y}, \mathbf{c}') \quad \forall \mathbf{c}' \in C\}.$$

Inside every decision region we have a sphere with radius $t = \frac{d_{min}-1}{2}$ which contains the error vectors that the code is able to correct with Hard Correction, defined as:

$$S(\mathbf{c}, t) = \{\mathbf{y} \in H^n : d_H(\mathbf{y}, \mathbf{c}) \leq t\}.$$

The condition for a perfect code able to correct all the errors is

$$2^k \cdot \sum_{i=0}^{t} \binom{n}{i} = 2^n.$$

Instead, non perfect codes are able to correct also $\mathbf{e}$ with $w(\mathbf{e}) > t$.
In our exercise we fixed the values of k and n as $k = 4$ and $n = 8$ and we considered the generator matrix

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

Initially, we considered the SNR ("Signal to Noise Ratio") E_b/N_0 varying between 0 and 10 dB and plotted the "Hard Correction Codeword Error Rate" analytic curve, given by the formula

$$P_w(e) = 1 - \sum_{i=0}^{t} \binom{n}{i} p^i (1-p)^{n-i}, \tag{1}$$

where t is the error correction capability defined as

$$t = \frac{d_{min}-1}{2}$$

and p is the bit error probability, given by

$$p = \frac{1}{2} erfc \sqrt{\frac{k}{n}\frac{E_b}{N_0}}$$

In this case $d_{min} = 3$, so $t = 1$.
Then, we built the LUT ("Look Up Table") corresponding to all the vectors with $w_H(e) \leq 1$. This is composed by sequences of bits in which the first k bits are the syndromes $\mathbf{s} = \mathbf{e}H$, and the others the error vectors themselves. We can consider the LUT as the sphere with radius t inside each decision region.
Later, we applied the "Syndrome Decoding" in order to simulate the Codeword Error Rate with $E_b/N_0 = 0 : 7dB$. The procedure followed was:

- create a random binary vector between 0 and $N = 2^k - 1$ and encode it with G;
- built the corresponding bipolar sequence, where all the zeros are replaced by "-1";
- transmit the sequence on an additive "White Gaussian Noise Channel", where each sample of Gaussian noise has zero mean and variance

  $$\sigma^2 = \frac{1}{2\frac{k}{n}\frac{E_b}{N_0}}.$$

  Done that we have the vector $\mathbf{r}$;
- from $\mathbf{r}$ we return to a binary sequence with 0 and 1 values;
- with the received vector we compute the syndrome $\mathbf{s} = \mathbf{e}H$ and look for the result in the LUT. If $\mathbf{s}$ is not present in the LUT the codeword received is wrong, whereas in the other case we estrapolate the corresponding error vector and add it to $\mathbf{y}$, obtaining the received codeword $\mathbf{C}_R$;
- compare the received codeword and the transmitted one and, in case they are different, increase the number of the wrong codewords. When we reach the value of $N_{cw} = 200$ we computed the "Word Error Probability" $P_w(e)$, dividing N_{cw} and N_{tc}.

After that, we compute and plot the Soft Correction Codeword Error Rate union bound and asymptotic approximation using the two relations

$$P_w(e) \leq \sum_{i=1}^{n} \frac{1}{2} A_i erfc \sqrt{\frac{k}{n} i \frac{E_b}{N_0}} \tag{2}$$

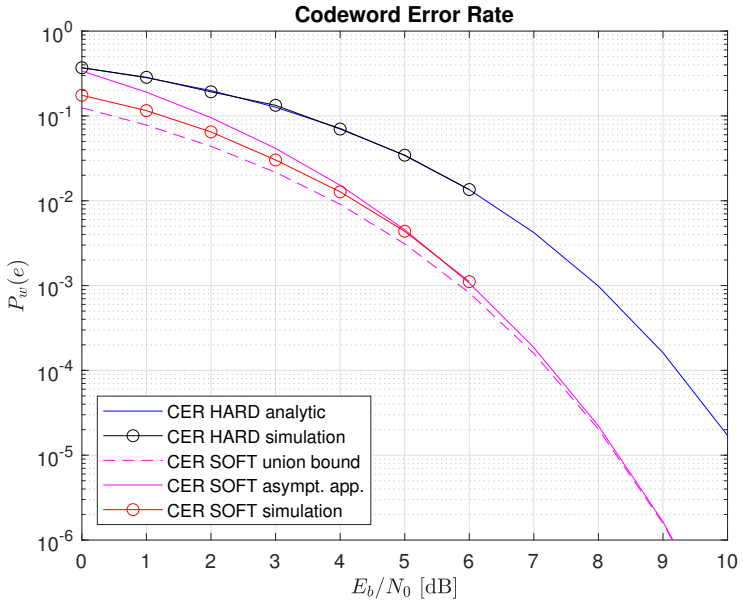
$$P_w(e) \approxeq \frac{1}{2} A_{min} erfc \sqrt{\frac{k}{n} d_{min} \frac{E_b}{N_0}} \tag{3}$$

where A_i is the number of codewords with Hamming weight equal to i and A_{min} the one with $W_H(\mathbf{c}) = d_{min}$.

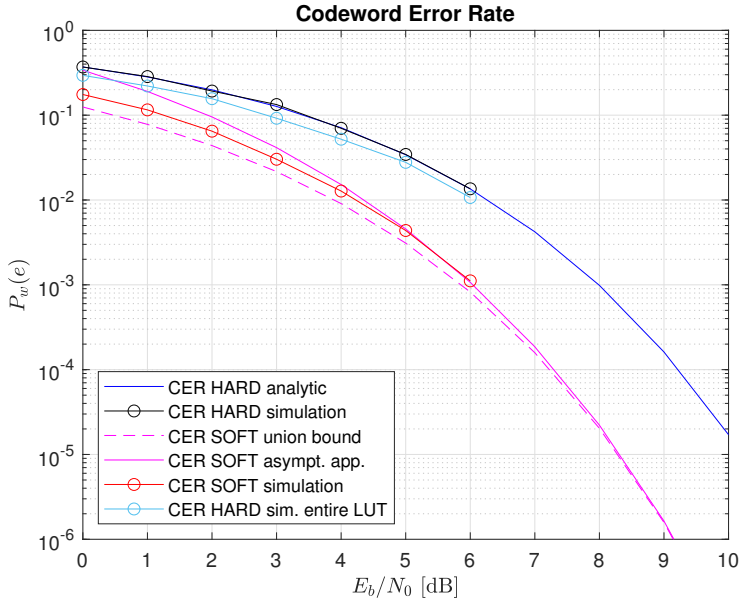
To simulate the Soft Correction, we construct the entire codebook and the corresponding bipolar one. The procedure is the same until we reach $\mathbf{r}$, then we calculate the Euclidean distance for each bipolar sequence of the codebook and take the codeword that gives the minimum distance.

$$\mathbf{C}'_R = argmin_{(\mathbf{c}' \in C)} d_E(\mathbf{r}, \mathbf{c}')$$

If $\mathbf{C}_R$ is different from $\mathbf{C}_T$ we increase the number of N_{cw} and reached the value $N_{cw} = 200$ we compute the Wrong Error Probability.

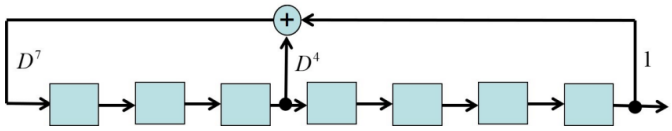


As we can see from the results, the union bound correctly reaches the asymptotic one for high SNR, whereas the simulation results agree with the analytical curves. Moreover, the Soft Correction returns an error probability which is smaller than the one given by the Hard Correction: the code is able to correct more errors because we exploit the all the characteristic of $\mathbf{r}$, considering both the amplitude and the sign. However, Soft Correction is computationally more complex because we have to generate all the codebook and compute the Euclidean distance for each codeword.
After that, we fill the complete LUT with size 2^r, adding the missing syndromes corresponding to the error vectors $w(\mathbf{e}) > t$ and repeat the whole simulation procedure obtaining that, for the same values of SNR, the error probability is smaller with respect to the previous LUT. This is not surprising, indeed the code is able to correct more errors.

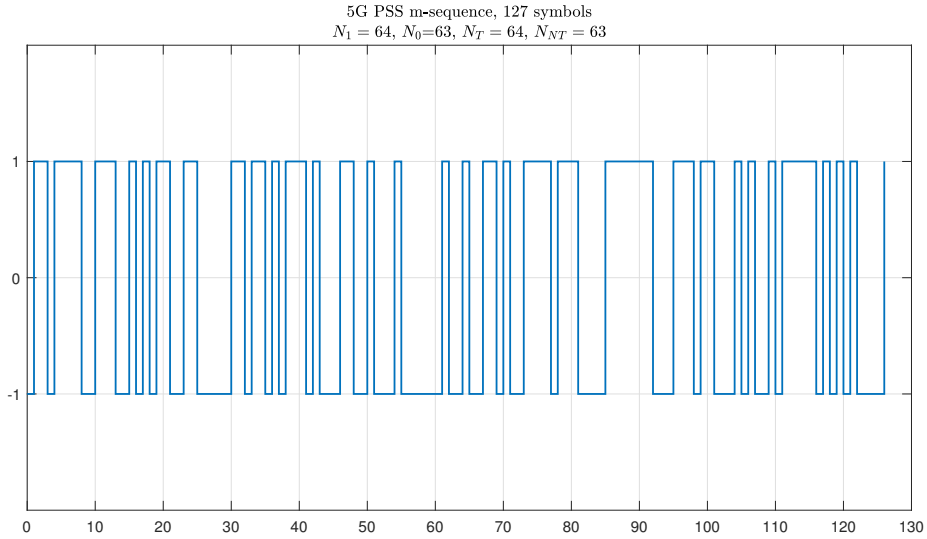


3 Exercise 2: LFSR m-sequences

In the second exercise we have dealt with a "Linear Feedback Shift Register" (LFSR) generating an m-sequence used in the "Primary Synchronization Signal" (PSS) of 5G New Radio.



The LFSR generates a pseudo-random binary sequence, shifting the content of every cell to the right, dropping the last bit and using as the input bit of the first cell a sum of some sequence bits. The sequence generated is the one composed by the $N = 2^m - 1$ dropped bits and it's called "m-sequence", where m is the number of cells.
In the considered case $m = 7$, the primitive polynomial describing the feedback connection is $D^7 + D^4 + 1$ and the starting seed is 1110110.
Using the code given in the assignment, we firstly generated the binary sequence and obtained the corresponding bipolar sequence, where all the zero bits are replaced by "-1" and the ones are mapped into "$+1$". Totally, we have an m-sequence of 127 symbols, which is represented in the following picture.



We then counted the number of 1s and 0s present in the sequence, such as the number of transitions and no-transitions in the signal. Notice that a transition occurs when a bit and its previous one have different values. The results obtained have been reported in the table below.

N0	N1	Nt	Nnt
63	64	64	63

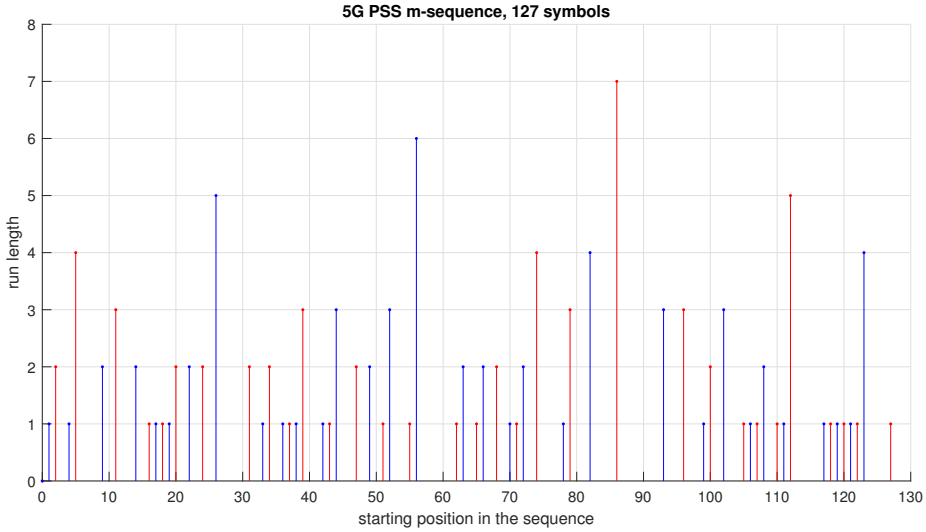
We can note that the properties of a pseudo-random binary sequence are satisfied: indeed

$$N_0 = N_1 - 1$$
$$N_T = N_{NT} + 1.$$

This is due to the fact that the all zero vector must be avoided by the possible sequences because it has no evolution.
We then create a table containing the $NR_0(i)$ and $NR_1(i)$, respectively the number of runs of i consecutive 0s or 1s present in the signal. Notice that a "run" with length i corresponds to a sequence of i consecutive equal bits. In our case we considered also the single bit, that have been treated as runs of length 1.

i	NR0	NR1
1	16	16
2	8	8
3	4	4
4	2	2
5	1	1
6	0	1
7	1	0

Moreover, we plotted each run length in its starting position in the sequence, using the blue color for the 0s and the red one for the 1s.
Considering the property for the runs in a ideal random binary sequence, we have that the number of runs of length i is defined as $N/2^i$, half for each bit value. The results obtained are almost ideal until, except for the the longest runs in the sequence, which is "6" for the ones and "7" for the zeros, due to the pseudo-random characteristic of the involved sequence.

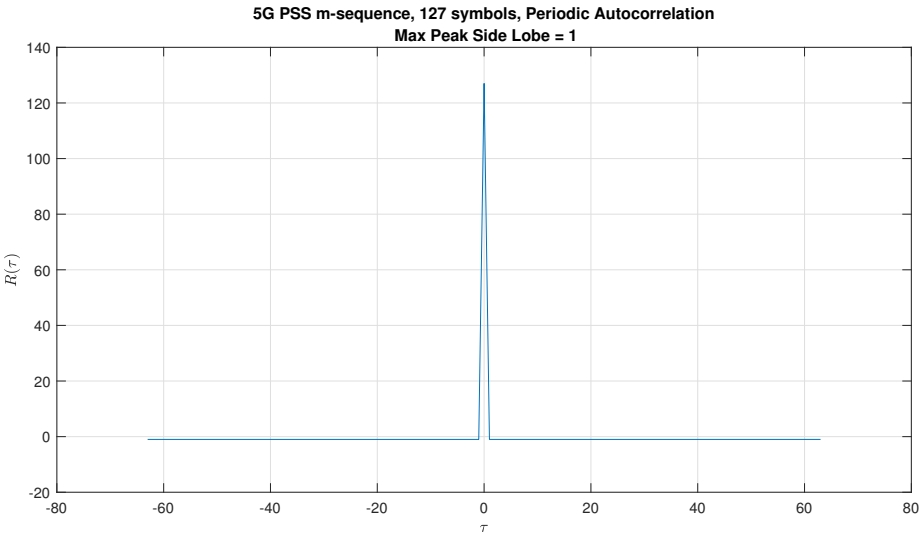


We then computed the autocorrelation function defined as

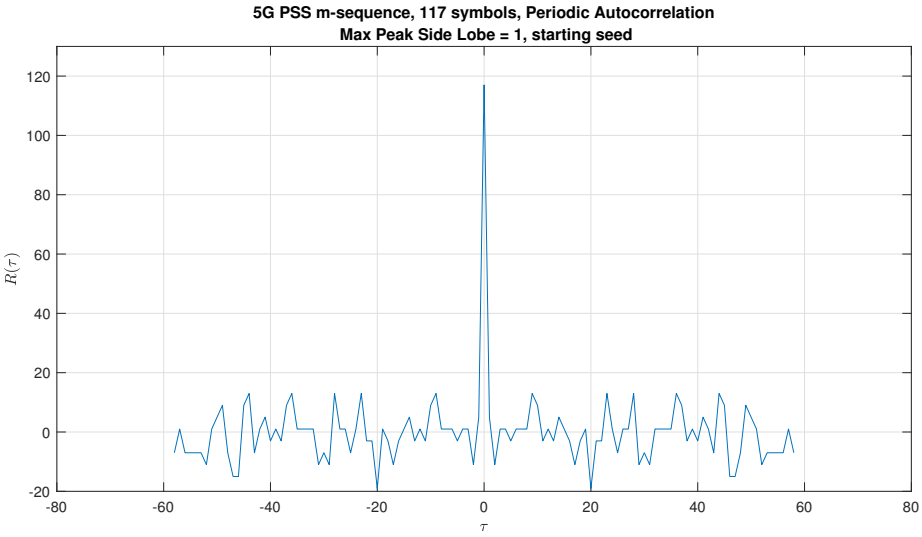
$$R(\tau) = \sum_{n=0}^{N-1} b(n)b(n-\tau) \qquad -(N-1) \leq \tau \leq N-1, \tag{4}$$

where b is the bipolar sequence. From the theory we know that, for a pseudo-random binary sequence, $R(\tau) = N$ for $\tau = 0$ and $R(\tau) = -1$ for $\tau \neq 0$.
The autocorrelation has been centered in the picture, where we can note that

the property is satisfied: in the origin $R(0)$ reaches the value 127, whereas for the others is equal to -1.



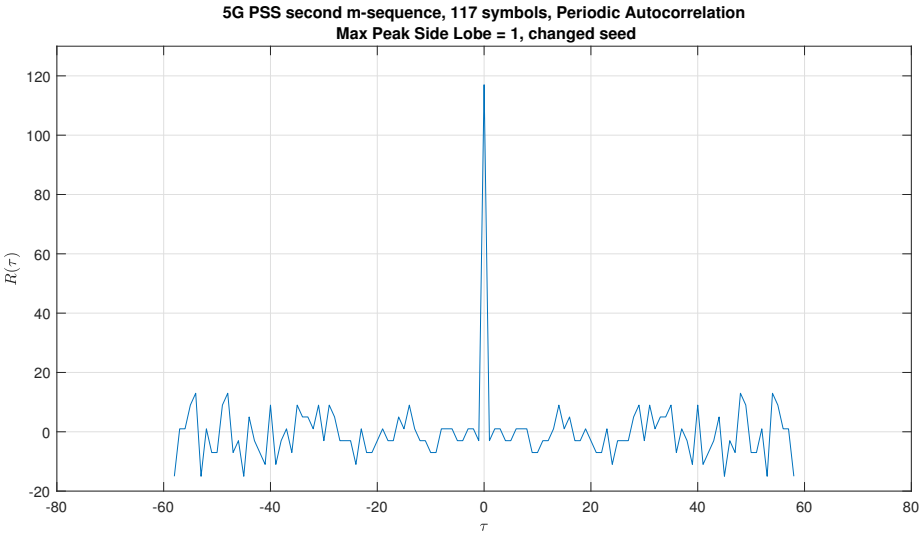
From now on, in the exercise we dealt with truncated m-sequences, where we cancel the last ten bits of the original m-sequence, so that $N = 117$. When truncation is applied, the nearly ideal properties of the m-sequence are lost: indeed, the autocorrelation becomes:



To compare the autocorrelation of the two sequences, we computed the "Maximum Peak Side Lobe", defined as

$$MPSL = max_{\tau \neq 0} |R(\tau)|.$$

For the ideal one we have $MPSL = 1$, whereas for the truncated one this does not hold anymore, becoming $MPSL = 19$. This is not unexpected: watching the picture representing $R(\tau)$ we can notice a lot of oscillations for $\tau \neq 0$, instead of a constant value "-1". From the theory we know indeed that, for a truncated m-sequence, $MPSL > 1$.
Finally, we changed the starting seed "0101000" and repeated again the procedure for the autocorrelation, obtaining a $MPSL = 15$.



We can conclude that, for any starting seed considered, if we deal with a truncated m-sequence, the value of the $MPSL$ is always greater than "1". Moreover, the autocorrelation $R(0)$, remains the same, whereas for the others not.

4 Exercise 3: Sync Marker detection

In the last exercise we dealt with the Sync Marker Detection, used to recognize a pattern or a marker at the beginning of the codewords during a transmission. In this case we fixed the sync marker length as $N = 16$ with the corresponding random binary pattern

$$\mathbf{p} = (p_1 \quad p_i \quad p_N), \text{ where } p_i \in \{0, 1\},$$

and the bipolar sequence

$$\mathbf{p}' = (p'_1 \quad p'_i \quad p'_N), \text{ with } p'_i \in \{-1, +1\},$$

so $\mathbf{p}' = \mathbf{2p} - 1$.
The $SNR = E_s/\sigma^2$ has been fixed to $E_s/\sigma^2 = -10dB$, with $E_s = 1$, the number of simulations $N_s = 100000$ and the minimum number of observed events $N_e = 20$.
During a transmission, at the receiver side we must distinguish between two

possible events: if we detect only noise (event defined as H_0, with $\mathbf{r} = \mathbf{n}$), or the bipolar pattern plus the noise (H_1, with $\mathbf{r} = \mathbf{p}' + \mathbf{n}$). Notice that $\mathbf{r}$ is the vector received. To distinguish between the two we used the "Likelihood Ratio Test" LRT

$$\Lambda(\mathbf{r}) = \frac{f_{\mathbf{r}/H_1}(\mathbf{r})}{f_{\mathbf{r}/H_0}(\mathbf{r})},$$

where we used the two PDF expressions

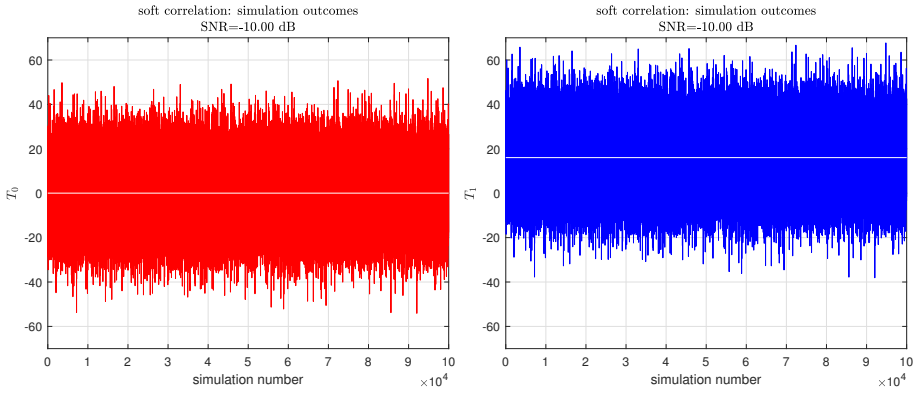
$$f_{\mathbf{r}/H_0}(\mathbf{r}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{r_i^2}{2\sigma^2}}$$

$$f_{\mathbf{r}/H_1}(\mathbf{r}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i - p_i')^2}{2\sigma^2}}.$$

Starting from the LRT, it is easier working with the its log, the "LogLikelihood Ratio Test"

$$T(\mathbf{r}) = \sum_{i=1}^{N} r_i p_i', \tag{5}$$

which is the soft correlation between the received vector and the bipolar pattern. This is done both for $\mathbf{r}_0$ and $\mathbf{r}_1$, obtaining



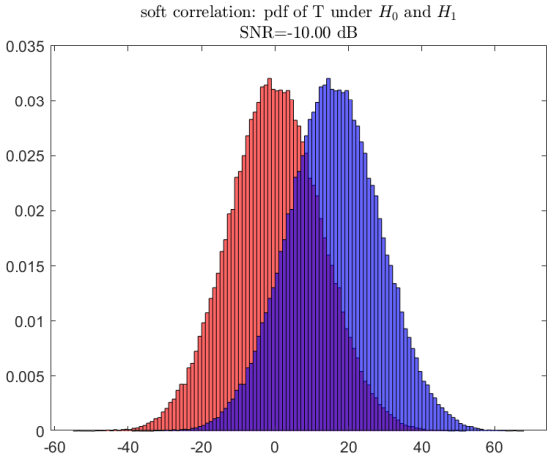
Notice that their mean value correspond to the white horizontal lines respectively around "0" and "16". Their values are correct, indeed, computed them analytically

$$T_{H_0}(\mathbf{r}) = \sum_{i=1}^{N} n_i p_i' = 0,$$

because we have only noise, and

$$T_{H_1}(\mathbf{r}) = \sum_{i=1}^{N} p_i' p_i' + \sum_{i=1}^{N} n_i p_i' = N + 0 = N,$$

that in this case is equal to $N = 16$.
Using the "histogram" function present in Matlab, we plotted on the same figure the two pdfs of T_0 an T_1.

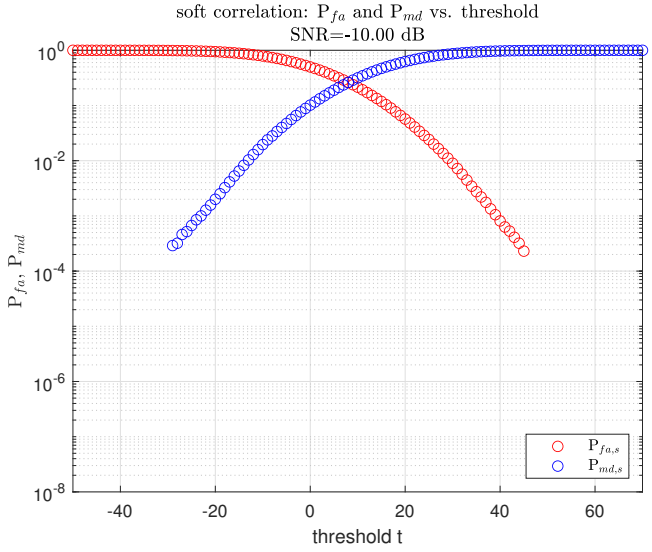


How we can see, the two pdfs present a gaussian distribution around their mean values, according with the theory. Moreover, they are normalized to "1" and the two distributions overlap for an interval of values. If we put the threshold with which compare the resulting T, we can obtain a "False Alarm" or a "Missed Detection", defined respectively as the probability that we declare that the pattern is present but it is not true and the case in which we declare the pattern absent but it was present.

$$P_{fa} = P(T_{H_0} > t)$$

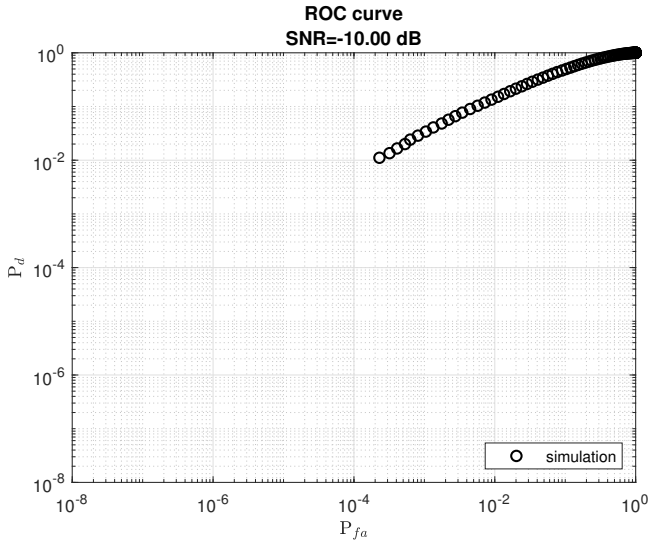
$$P_{md} = P(T_{H_1} < t)$$

The parameter t is the threshold with which we have to compare T, running for the whole interval of values.



We can note that for low value of t the P_{fa} is almost equal to "1", because the condition of False Alarm is always verified, whereas, increasing the threshold, the probability becomes smaller. Instead, P_{md} has the opposite behavior, increasing with the value of t. Notice that the point in which $P_{fa} = P_{md}$ is the middle point of the intersection between the two pdfs.
We then plotted the *ROC* curve defined as:

$$P_d = (1 - P_{md}) \text{ vs } P_{fa}$$



We then try to change the value of the SNR and if N, obtaining that (to see better the results, cfr Appendix A):

- decreasing the SNR, the two pdfs overlap more, so the probabilities of false alarm and missed detection increase. The *ROC* curve is linear but more compressed;
- increasing the SNR, the two pdfs move away, so the probabilities of false alarm and missed detection decrease. The *ROC* curve is always almost equal to 1, because of the low value of P_{fa};
- decreasing N, the two pdfs shift together to the left, so the two probabilities. The *ROC* curve is more extended;
- increasing N, the two pdfs shift together to the left, so the two probabilities. The *ROC* curve is closer to the value 1 with respect to the case with $N = 16$.

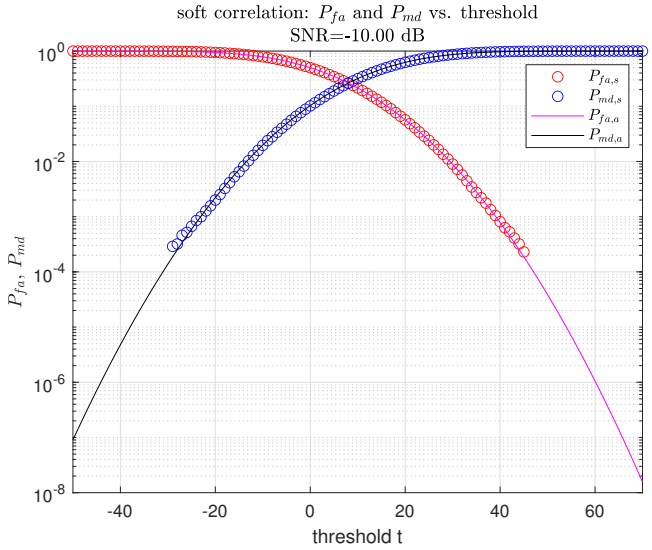
Moreover, changing the SNR, the variance changes too, according to:

$$\sigma^2 = \frac{1}{2\frac{k}{n}\frac{E_b}{N_0}}.$$

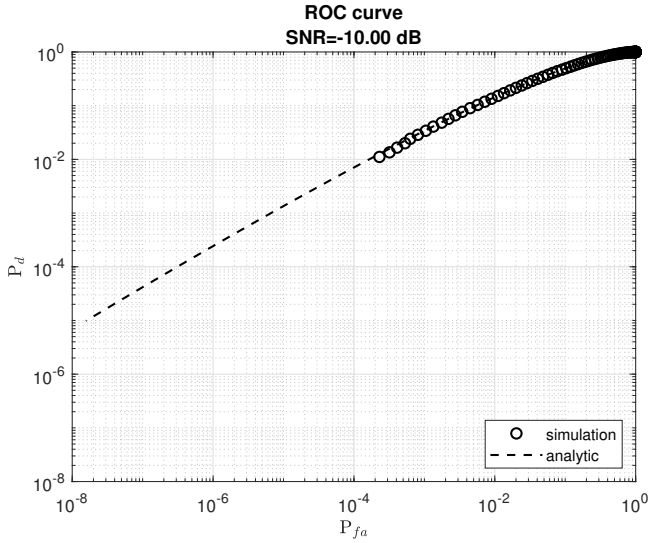
We then computed and added to the simulated curves the two analytical ones of P_{fa} and P_{md}, using the formulas:

$$T_{H_0}(t) = \frac{1}{\sqrt{2\pi N\sigma^2}} e^{-\frac{t^2}{2N\sigma^2}}$$

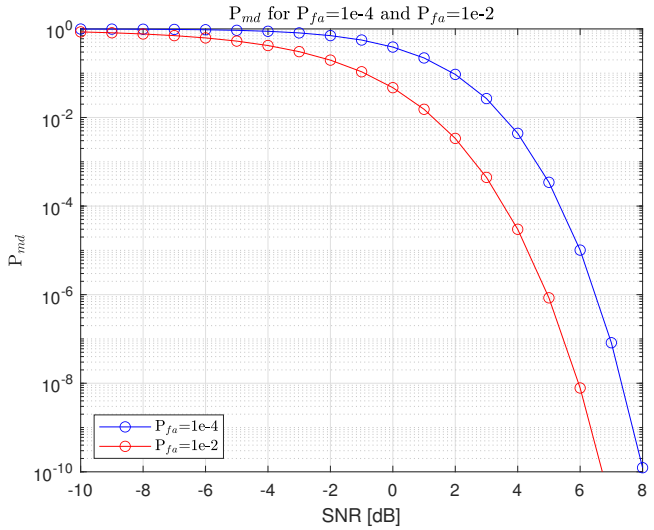
$$T_{H_1}(t) = \frac{1}{\sqrt{2\pi N\sigma^2}} e^{-\frac{(t-N)^2}{2N\sigma^2}}$$



Moreover, we did the same for the *ROC* curve.



Finally, considering $N = 16$ and $SNR_{dB} = -10 : 8$, we computed analytically the value of P_{md}, related to the two values of $P_{fa} = 10^{-2}$ and $P_{fa} = 10^{-4}$.



As we can see, to a smaller value of false alarm probability corresponds a smaller missed detection probability. It means that the two pdfs are more moved away for smaller values of P_{fa}, so the SNR is bigger in the first case.

5 Question 1: Extension, puncturing, shortening

For the first question of the assignment we dealt with extension, puncturing and shortening of a code $C(n,k)$ with $d_{min}(C) = d$.

- The first operation consists in "Extension": we obtain $C_1(n_1, k_1)$ adding an extra parity bit, in order to obtain all the codewords with even weight. In this case $k_1 = k$ and $n_1 = n + 1$. If the original code had an odd minimum distance, adding the new parity bit increase it by 1, otherwise it remains the same

$$d_{min}(C) \text{ odd} \longrightarrow d_{min}(C_1) = d_{min} + 1$$
$$d_{min}(C) \text{ even} \longrightarrow d_{min}(C_1) = d_{min}$$

- With the "Puncturing" we delete from the original code a parity bit, obtaining a new $C_2(n_2, k_2)$ with $k_2 = k$ and $n_2 = n-1$. In this case, if the d_{min} is associated to a codeword (whose distance is computed with respect to the all zero vector according to what we said in the precious assignment) with parity bit equal to "1", then the minimum distance decreases by 1, obtaining surely an odd d_{min}

$$d_{min}(C) \text{ even} \longrightarrow d_{min}(C_2) = d_{min} - 1 \text{ odd}$$

  Otherwise, it remains the same.

- "Shortening" consists on considering only the codewords that start with "0" and then delete the first bit. Here we have three possible cases:
  - if the d_{min} was associated to codewords that start with "1", then the d_{min} could increase till $d_{min} = n - 2$ (considering a code with parity bit at the end);
  - if the d_{min} was associated to codewords that start both with "1" and "0", then the d_{min} remains the same.
  - if the minimum distance was already associated to a codeword with "0" at the beginning, then it does not change.

  In this case the d_{min} could never decrease.

6 Question 2: Autocorrelation property of m-sequences

In this section we want to prove that for an entire m-sequence (not truncated), holds the following property:

- $R(\tau) = N$ for $\tau = 0$
- $R(\tau) = -1$ for $\tau \neq 0$

To do that, we considered a code with $k = m$ number of cells and we generate the entire codebook containing all the cyclic shifted m-sequence. Notice that for the generation of the sequence the feedback connection of the LFSR has to correspond to a primitive polynomial and that τ is the delay. Every sequence is then mapped into the corresponding bipolar one.
For τ the proof is trivial, indeed we have to multiply each bit for itself and then sum together all the results. Because we have only two possible values "-1" and "+1", each product makes "1" and the summation returns the number of bits of the m-sequence $N = 2^m - 1$. We also know that for a sequence of this type the number of "0" is $N_0 = N_1 - 1$, where N_1 is the number of ones. Obviously $N_0 + N_1 = N$, so $N_1 = \frac{N+1}{2}$ and $N_0 = \frac{N-1}{2}$.

$$R(0) = \sum_{n=0}^{\frac{N+1}{2}} (+1)^2 + \sum_{n=0}^{\frac{N-1}{2}} (-1)^2 = \frac{N+1}{2} + \frac{N-1}{2} = N$$

For the second part the proof is more tricky. We first note that for $\tau = 1$ $R(1) = -1$: considering the number of transition N_T and the number of no transition N_{NT}, for an m-sequence it holds $N_T = N_{NT} + 1$. For a single shift, where we have a transition in the original sequence, the value in the new position is the opposite of the first one ($+1 \to -1$) and ($-1 \to +1$): each product gives a value equal to "-1". Whereas, where we do not have a transition, the same value is present in that position in the m-sequence and in the shifted one: the product gives "1". Because we have N_{NT} and $N_T = N_{NT} + 1$ we obtain:

$$R(1) = N_{NT} \cdot (1) + N_T \cdot (-1) = N_{NT} - N_{NT} - 1 = -1$$

This holds for every m-sequence and its shifted version by $\tau = 1$ of the codebook. Then we use the property of the code, for which if we sum to a codeword another one of the codebook, we obtain again a codeword. So, to each couple of m-sequence, we can sum the same vector in order to obtain, for the first sequence, the original one. Doing that, we obtain couples with the starting m-sequence and the one shifted by $1 \leq \tau \leq N-1$, but the autocorrelation $R(\tau)$ remains always equal to "-1" because the summation of the same codeword to the two sequences does not modify the autocorrelation calculation.

7 Question 3: Autocorrelation and fft

In the last part of the assignment we have to prove that the autocorrelation

$$R(\tau) = \sum_{n=0}^{N-1} b(n)b(n-\tau)$$

can be written as

$$R(\tau) = \mathcal{F}^{-1}\left(\mathcal{F}(x_{1b}) \cdot \mathcal{F}^{*}(x_{1b})\right)$$

Considering the initial relation and making the change of variable $n - \tau = k$, we obtain:

$$R(\tau) = \sum_{k=0}^{N-1} b(k+\tau)b(k),$$

which corresponds to the convolution $b(n) * b(-n)$.

$$b(n) * b(-n) = \sum_{k=0}^{N-1} b(k)b(k-\tau) = \sum_{n=0}^{N-1} b(n+\tau)b(n)$$

which is the autocorrelation. Notice that we used : $n = k - \tau$, so $k = n + \tau$. Now we consider the second relation: we note that, thanks to the convolution theorem:

$$\mathcal{F}^{-1}\left(\mathcal{F}(x_{1b}) \cdot \mathcal{F}^{*}(x_{1b})\right) = \mathcal{F}^{-1}\left(X_k \cdot X_k^{*}\right) = \mathcal{F}^{-1}\left(X_k \cdot X_{\tau-k}\right) = b(k) * b(-k)$$

where we used the properties that:

- $X_k = X_{\tau-k}^{*}$
- $\mathcal{F}^{-1}\left(X(w) \cdot Y(w)\right) = x(t) * y(t)$

because we obtain from the two relations the same result, we can state that they are equivalent.

8 Appendix A

We reported the results obtaining for different values of SNR and N in Exercise 3.

Figure 2: SNR= -20 dB

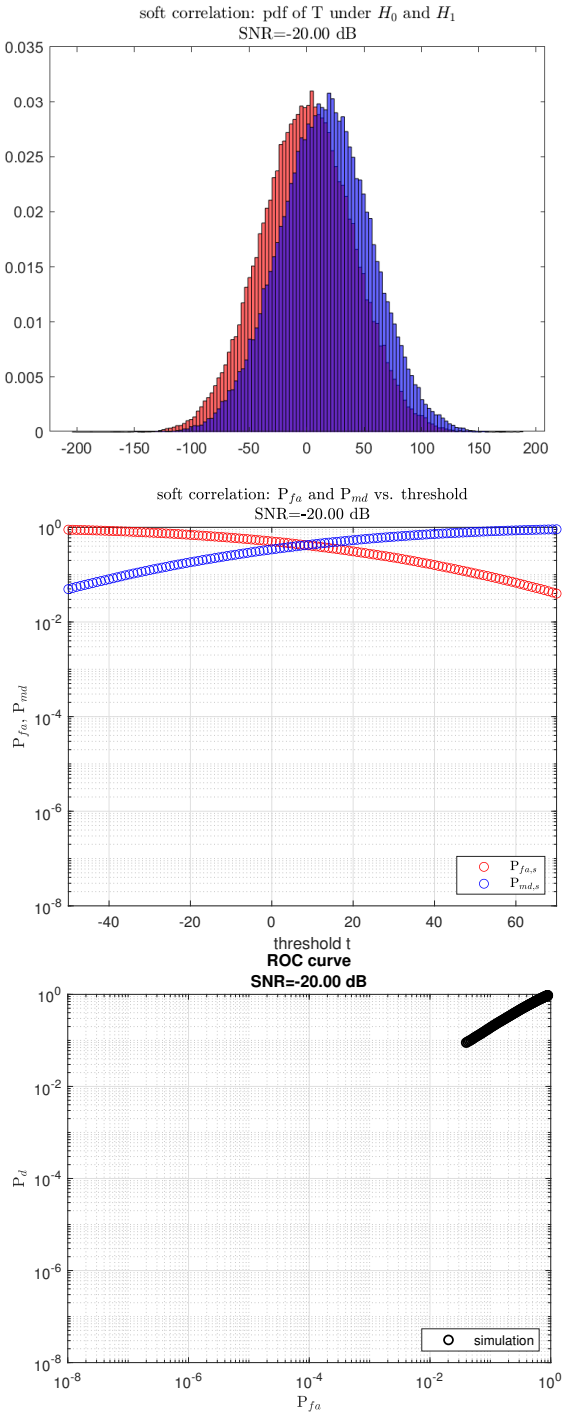


Figure 3: SNR= 0 dB

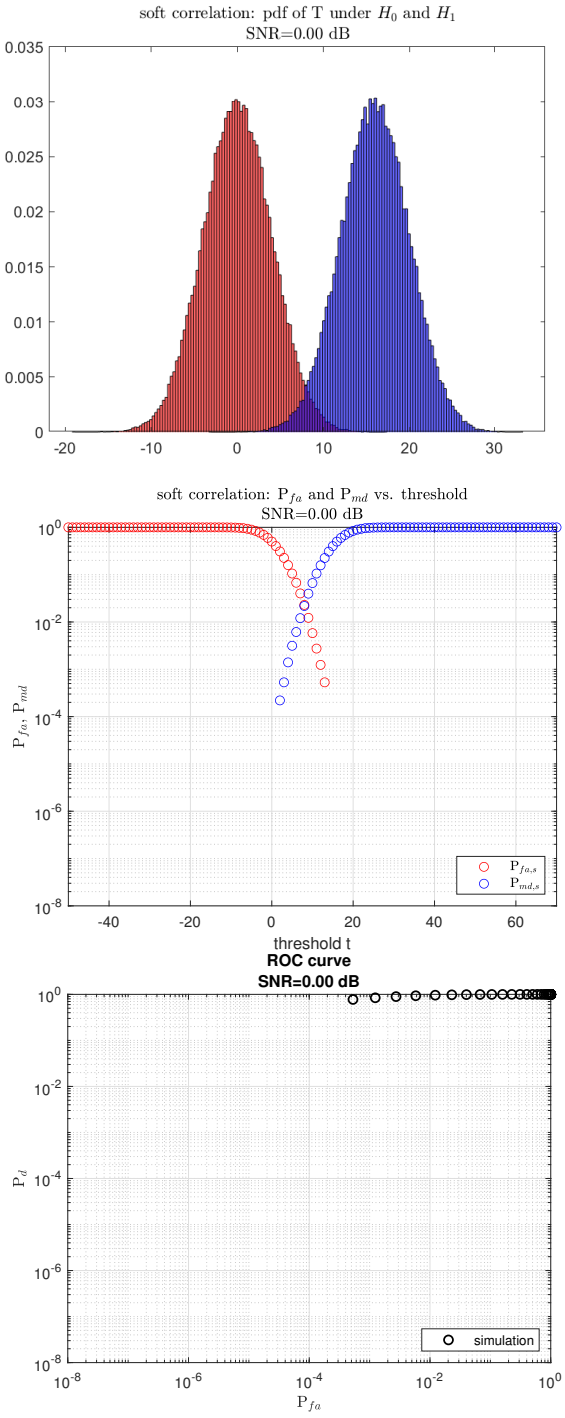


Figure 4: N = 8

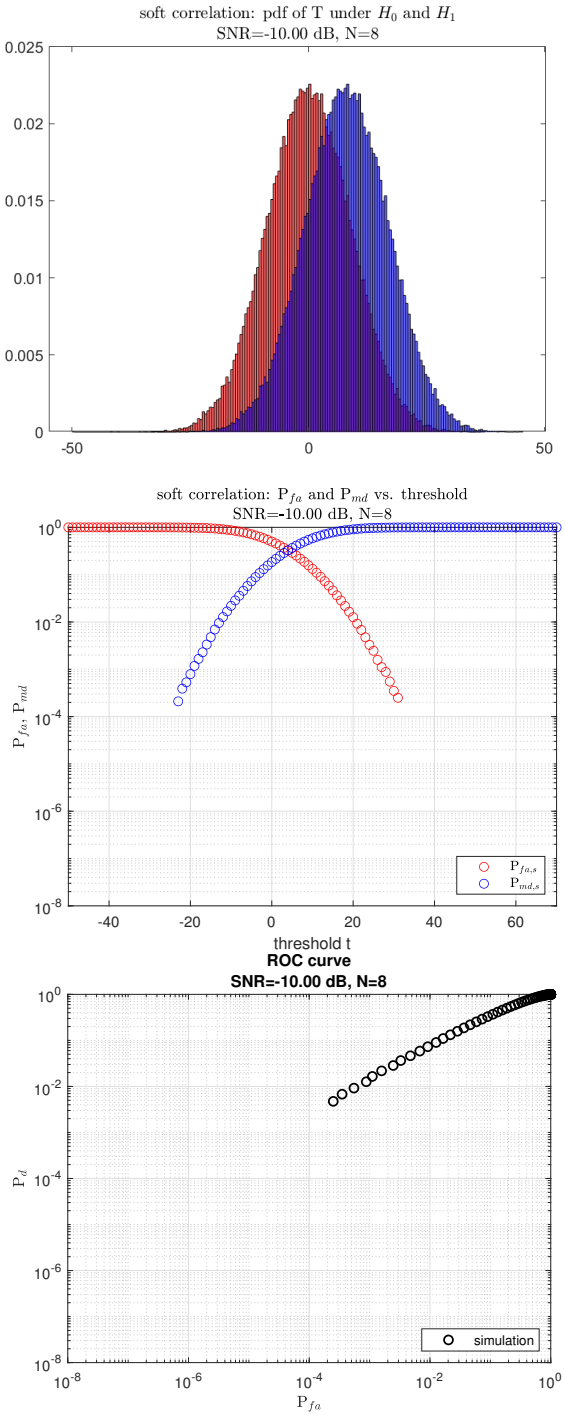
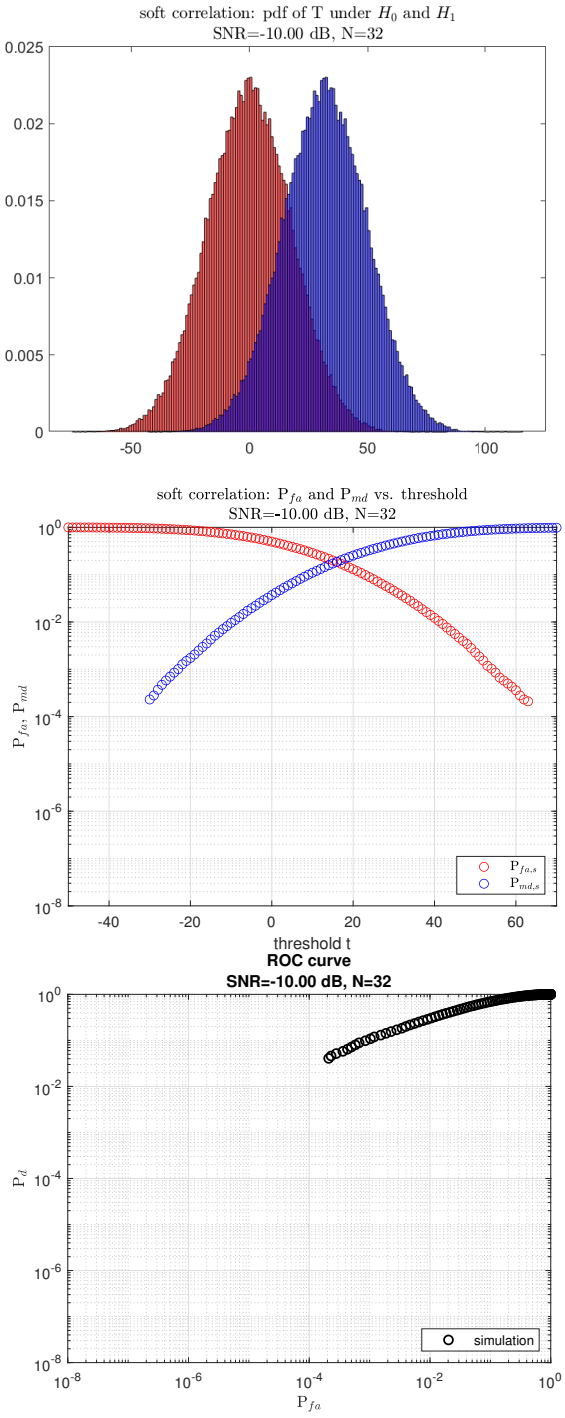

soft correlation: pdf of T under H_0 and H_1
SNR=-10.00 dB, N=8
0.025
0.02
0.015
0.01
0.005
0
-50
0
50
soft correlation: P_{fa} and P_{md} vs. threshold
SNR=-10.00 dB, N=8
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
P_{fa}, P_{md}
$P_{fa,s}$
$P_{md,s}$
-40
-20
0
20
40
60
threshold t
ROC curve
SNR=-10.00 dB, N=8
P_d
simulation
P_{fa}

Figure 5: N = 32


soft correlation: pdf of T under H_0 and H_1
SNR=-10.00 dB, N=32
0.025
0.02
0.015
0.01
0.005
0
-50
0
50
100
soft correlation: P_{fa} and P_{md} vs. threshold
SNR=-10.00 dB, N=32
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
P_{fa}, P_{md}
$P_{fa,s}$
$P_{md,s}$
-40
-20
0
20
40
60
threshold t
ROC curve
SNR=-10.00 dB, N=32
P_d
simulation
P_{fa}