Adaptive Signal Processing

Project 2018Z:

Implementation and Study of different algorithms used for telecommunication receivers

---

Guillermo Pajares Martín

K-5123

Wydział Elektroniki
i Technik Informacyjnych

POLITECHNIKA WARSZAWSKA

Index Table

Figures Table

I. Summary

Since the telecommunication has existed, the problem of recovering the sent signal has been the most important one. For these aim, different technologies have been used along the years, from the analogic telecommunications to the digital ones used in the present.

Since some years ago, the capacity of developing adaptative filters in the receivers has been growing. This kind of filters try to minimize the error of the signal applying different algorithms to tune the filter in the receiver. This is fundamental in digital communications to recover the original signal sent by the channel. So, the principal schema in the digital communication is the next one:

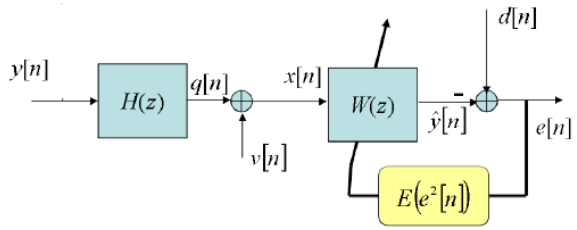


Figure 1: General Schema of a Telecommunications System

So, we can see that the filter for reception takes the error signal (difference between reference signal and the obtained after filtering the received one) and try to minimize the expected value of this for tuning the filter in the best way. For doing this, several algorithms have been developed during the years.

The main aim of this project was to analyse some of these algorithms and to obtain some conclusions about what the best under different situations is. These analysed algorithms are LMS and RLS, because these are some of the most used in the telecommunication systems.

For doing this analysis, all the system has been simulated using MATLAB. For this, y[n] is created as a random sequence of bits codified using 16-QAM; this signal is put in the channel (H[z]) which will follow the model of a FIR filter and also AWGN will be added in the channel, after this, x[n] is founded by receiver.

In the receiver, the necessities are to demodulate the signal and to recover the original sequence of bits, using the different adaptative algorithms for these, so the algorithms have been implemented for doing the needed experiments. The filter is tuned passing a training sequence to the algorithm. We'll obtain the output of the system and with the error signal the filter must be able to recover the original signal, using a reference signal (obtained from a decision function) for this aim.

So, with the different experiments, the next parameters can be obtained for each adaptative simulation: number of wrong recovered bits, BER, filter's coefficients and EVM. So, with these parameters, big conclusions about the different adaptative algorithms can be obtained, which is the aim of this project.

II. System definition

The system follows the general schema of a telecommunications system (Figure 1), so modules for the Transmitter, the Channel and the Receiver have been designed for the case of using a QAM modulation for the communications. In the next subsections, the design of this modules is described. So, the full system follows the next schema:

Figure 2: Developed System General Schema

i) Transmitter

For the transmitter, there are three necessaries sub modules: the signal generator, the encoder and the QAM modulator. In this way we obtain the final QAM-modulated signal (adapted to be sent through the channel) from the original bits.

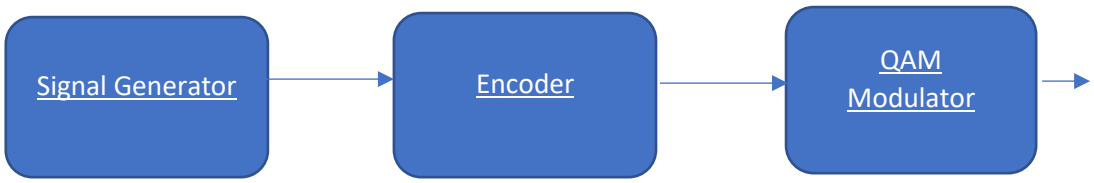


Figure 3: Developed Transmitter Schema

a. Signal Generator

The signal generator is the module that obtain the signal, in bits, to be sent through the channel. In this case, the Signal Generator obtains a simply random binary signal will should be recuperated later by the receiver.

b. Encoder

The encoder just obtains the needed code from the original bits, this code should be adapted converted into symbols of ***M levels***, using the appropriate number of bits per symbol for codification, following the next formula:

$$k = \log_2 M$$

Figure 4: Number of bits per symbol

c. QAM Modulator

In this project, is have been selected the QAM modulation for being used for the system. The Modulator of this project obtains the needed modulated signal from the code generated by the encoder. Before to be transmitted a training, sequence is concatenated with the modulated signal. This Training Sequence is used in the receiver to train the adaptive algorithm of the filter.

ii) Channel

The channel is modelled as a FIR filter plus the addition of AWGN to the sent signal. So, some coefficients should be used and the AWGN should be generated.

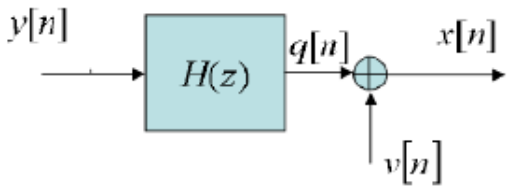


Figure 5: Channel schema

iii) Receiver

In the created system, the receiver is divided into three subsystems for obtain the full sent signal: the adaptive filter, the QAM demodulator and the Decoder. In this way, if all goes right, we can obtain again the original generated signal.

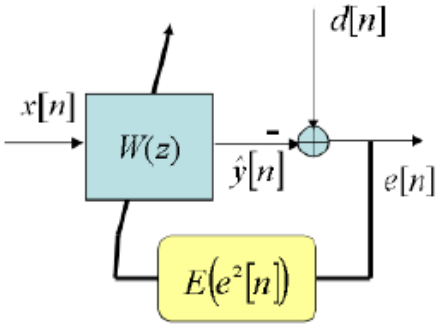


Figure 6: Receiver Schema

a. Adaptive filter

The adaptive filter allows us to obtain better quality in the recovery of the sent signal. There are different algorithms that allow us to tune the filter in the appropriate way, here there has been implemented the LMS and RLS ones for their study. This filter is actually not fully necessary, also there are some simulations made without filter, but the quality of the obtained signal is much better with it.

It uses the training signal transmitted to tune itself and reduce the possibility of error in the demodulation.

b. QAM demodulator

The QAM demodulator recovers the code from the symbols obtained after the adaptative filter.

c. Decoder

For obtaining the desired signal, the code obtained after the demodulation should be decoded into bits, doing the opposite operation of the encoder. After this module, the desired signal is obtained.

III. Developed Software Description

For the simulation of the described telecommunication system, several MATLAB files have been generated, the most important one is the script **EASP_Project_2018Z.m** which allow us to simulate all the system under the introduced parameters. This script also invokes anther additional functions: **lms.m, rls.m, decision.m and EVM.m**.

i) EASP_Project_2018Z.m

This script allows us to execute the full simulation of the system described in the previous section. For this, the desired parameters should be introduced in the first lines of the script:

```
%% System parameters

%Transmitter parameters
M = 16; %Bits per symbol
k = log2(M); %Coding levels
N = 10000; %Number of random generated bits

%Channel parameters
channel_SNR = 15; %Channel SNR for the addition of the White Noise
h = [1, 0.3]';%Channel coefficients (is supposed as a FIR filter)

%Receiver parameters
Alg = 'LMS'; %Algorithm to apply in the study ('NO', 'LMS' or
'RLS')
L = 2;
alpha = 0.01;
lambda = 0.8;
gamma = 150;
```

Figure 7: Example of System Parameters of the EASP_Project_2018Z.m script

The user can select the number of bits per symbol to apply to the QAM modulation, the number of random generated bits, the channel coefficients, the channel SNR, the Algorithm he desires to use for the tuning of the adaptive filter (it can be LMS, RLS or No filter), and the algorithm parameters

After making the desired modifications of the system parameters, only executing the script the user will obtain several graphs about how the signal changes in all the stages of the system. Also, the user will receive the original random sent bits and the recovered one and some measures trough the MATLAB's console. These measures are: Number of wrong bits, Bit Error Rate (BER) and Error Vector Magnitude (EVM).

ii) lms.m

This function obtains the LMS value of the training sequence, using as reference the decision function of the full received signal. Also, it uses the L and alpha parameters selected by the user:

```
function [e, y, ff] = lms(x, d, L, alpha)
```

This function is used for tuning the adaptive filter. This function follows the usual LMS algorithm in its version for complex numbers:

```
for n = 1:N

        x_n = [x(n); x_n(1:end-1,1)];

        y(n) = f_n.' * x_n;
        e(n) = d(n) - y(n);
        f_n = f_n + (alpha*e(n)*conj(x_n));

        ff(:,n) = f_n;

end
```

Figure 8: Implemented LMS algorithm

iii) rls.m

This function is also used for tuning the adaptive filter. This function obtains the RLS value of the training sequence, using as reference the decision function of the full received signal. Also, it uses the L and alpha parameters selected by the user:

```
[e, y, ff] = rls(x, d, L, lambda, gamma)
```

This function follows the usual RLS algorithm in its version for complex numbers:

```
    for n = 1:N

        x_n = [x(n); x_n(1:end-1,1)];

        y(n) =f_n'*x_n;
        e(n) = d(n)-y(n);
        alpha = P*x_n / (lambda+(x_n'*P*x_n));
        f_n = f_n + alpha*conj(e(n));
        P = (P - (alpha*x_n'*P))/lambda;


        ff(:,n) = f_n;

    end
```

Figure 9: Implemented RLS Algorithm

iv) EVM.m

It just calculates the Error Vector Magnitude of the desired signal; the result is printed in the MATLAB's console.

```
function [result] = EVM(x)
```

v) decision.m

This function returns the point from the 16 points of the training sequence which lies closest (in the Euclidean sense) to the point x given as argument, this have been developed following **[6]** and **[7]**. The output of this function is used as reference signal d(n) by the used adaptive algorithm and by the EVM function for its calculation.

```
function [z] = decision(x)
```

IV. Study of the Developed System

Executing the developed script (**EASP_Project_2018Z.m**, section III) enough number of times, we can obtain results about how the system works for each of the stages. These results will be analysed deeply in the next section.

i) Signal Transmission

The script generates a random bitstream each time, depending of the introduced parameters. In these experiments, the parameters selected have been: **M = 16** and **N = 10000**; all the repetitions of the script made under same conditions.

When we execute the script, the first result that we have is the random bitstream, the first 50 samples are plotted, like in the next example, taken from one of the executions.

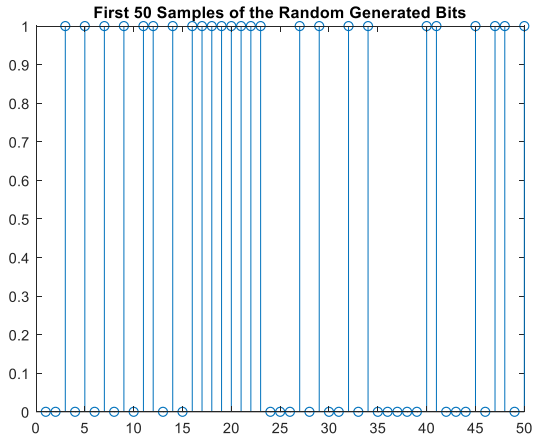


Figure 10: Example of the working of the random bit generator

After, these bits are converted into and encoded sequence of symbols by the encoder, as it was said in the section II, in the same example, we obtained the next sequence of symbols:

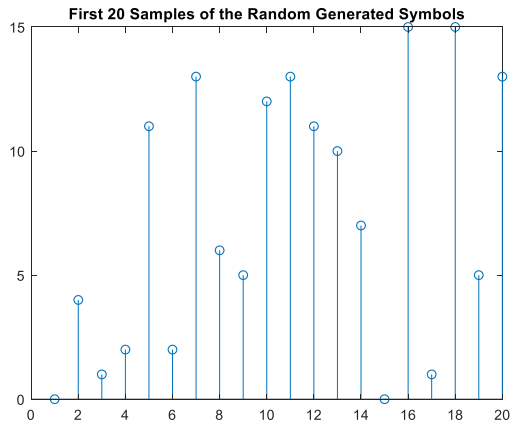


Figure 11: Example of the symbol's generation

So, now the signal is ready to be modulated, as explained in section II, and to be added the training sequence. Here, the modulation used is a 16-QAM, because the number of symbols is 16. The script also plots the generated constellation, with is the theorical one. This constellation is the one transmitted through the channel.

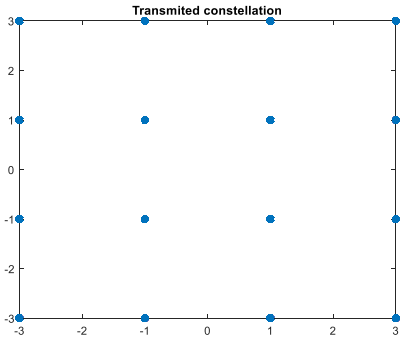


Figure 12: Constellation generated by the Transmitter

ii) Signal through the channel

The generated signal is sent by the transmitter through the channel, this is a channel modelled as FIR filter with the addition of noise (section II). In this case, the selected filter coefficients for the channel have been: $\mathbf{h = [1, 0.3]^T}$. The value of the channel SNR is variated through the experiments to obtain different results and can observe the influence of this parameter in the system. Some of the obtained results are the next ones:

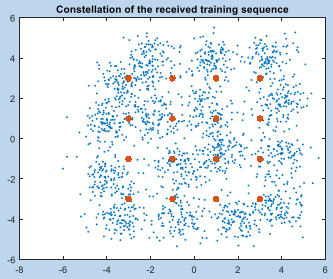
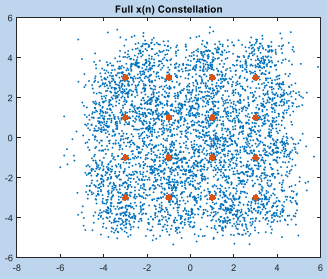
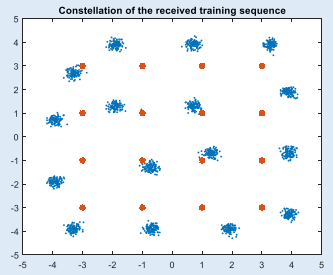
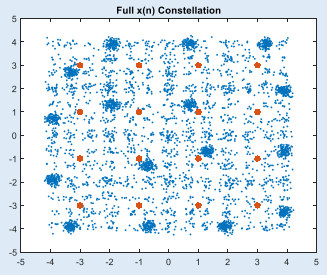
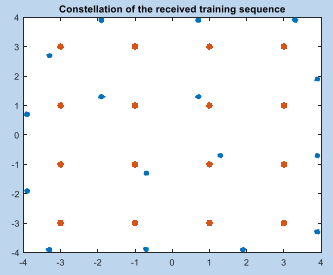
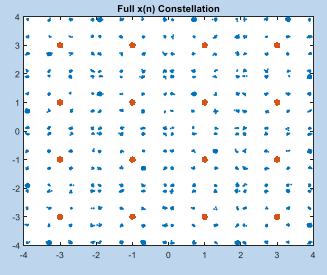
Channel SNR	Training sequence	Full signal
1		
15		
30		

Figure 13: Constellations of the signal after going channel filtration for different SNR values

It can be observed how the non-linearity of the channel and the addition of the AWGN affects to the original signal, these results will be analysed in the section V.

iii) Signal Reception

In this subsection, the obtained results are going to be described; but the detailed analysis and the obtained conclusions will be taken in the next section (section V).

There are three cases of analysis: when there is no adaptive filter in the receiver, when this adaptive filter uses the LMS algorithm and when it uses the RLS one. For these cases, several simulations of the system have been done for obtain these results, here some of them are going to be taken, but the general comportment will be described.

a. No Adaptive Filter

When there is no adaptive filter at the entrance of the receiver, the demodulated signal is just the one we obtain after the channel, with no treatment. So, some samples of the obtained codes are the next ones, in comparation with the original codes:

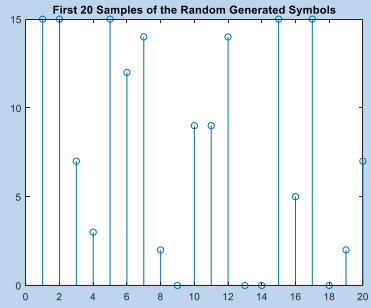
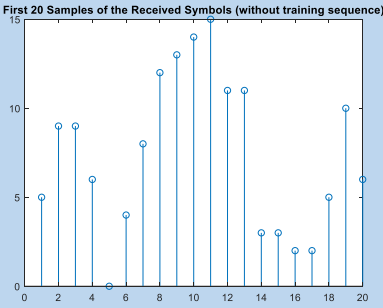
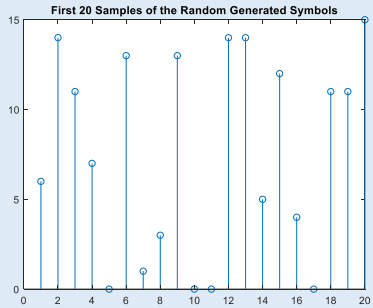
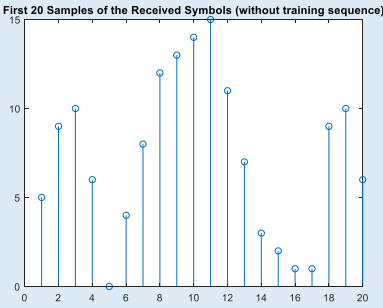
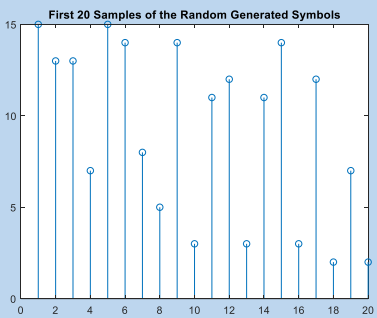
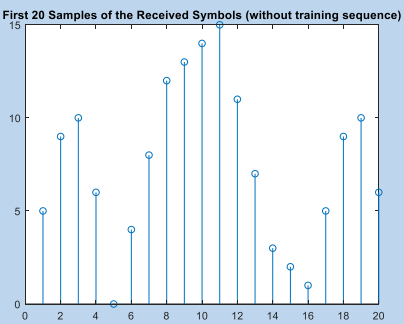
Channel SNR	Original code (in Transmitter)	Recovered Code (in Receiver)
1		
15		
30		

Figure 14: Recovered Code using no filter for different SNR values

After this demodulation, the decoding is applied, so the bitstreams are obtained. Also, here are given some examples:

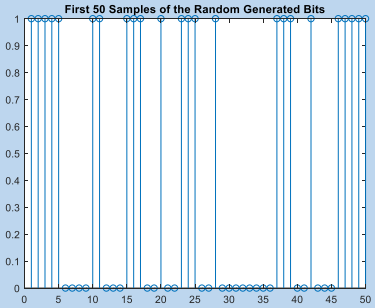
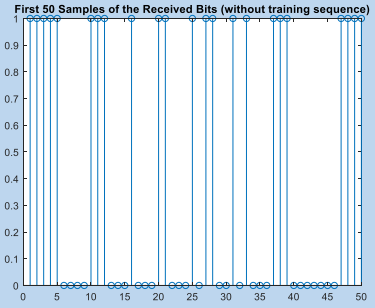
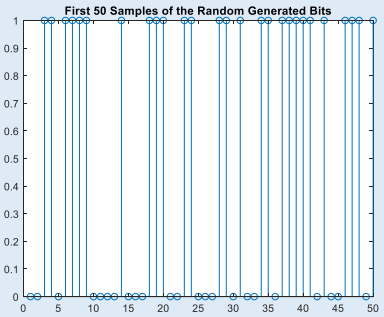
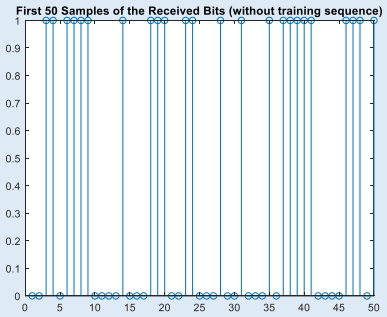
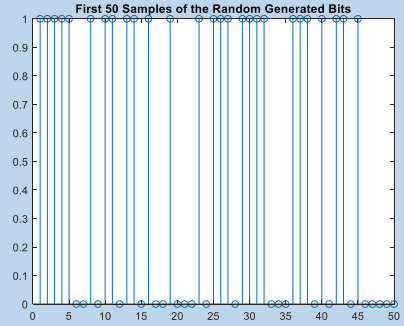
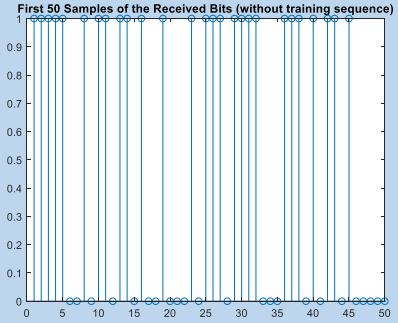
Channel SNR	Original signal (in Transmitter)	Recovered signal (in Receiver)
1		
15		
30		

Figure 15: Recovered Signal (Bitstream) using no filter for different SNR values

And about the measures of the full system, the average results after 5 executions of the full script for each SNR value (1, 15 and 30), we have the next results:

Channel SNR	Number of errors	BER	EVM
1	**1488.6**	**0.14886**	**0.5032**
15	**545.4**	**0.05454**	**0.49982**
30	**0**	**0**	**0.50258**

Figure 16: Measurements of the System using no filter for different SNR values

All these calculations were made using the auxiliary excel file **Measurements.xlsx** which is included in the PR_K5123.zip file.

b. Adaptive filter modelled by LMS algorithm

When we use the LMS algorithm to tune the input filter, the first thing with is necessary is to train this algorithm using the training filter; after the training is completed, we can use the resulting filter to obtain a better solution to our problem of recover the original transmitted signal. For these experiments, the parameters used for LMS algorithms are: **L = 2** and **α = 0.01.** For the training to be completed, is necessary that the |e(n)| signal goes near to zero. We can see that this happens for all the SNR used for the experiments.

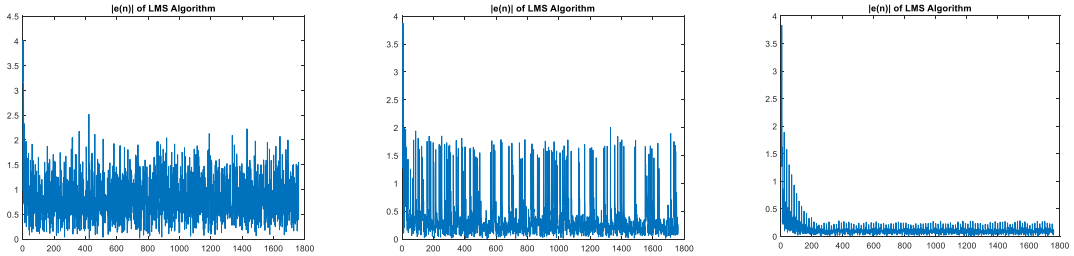


Figure 17: |e(n)| signal of LMS algorithm for different SNR values

The obtained constellations after the filter execution are the next ones:

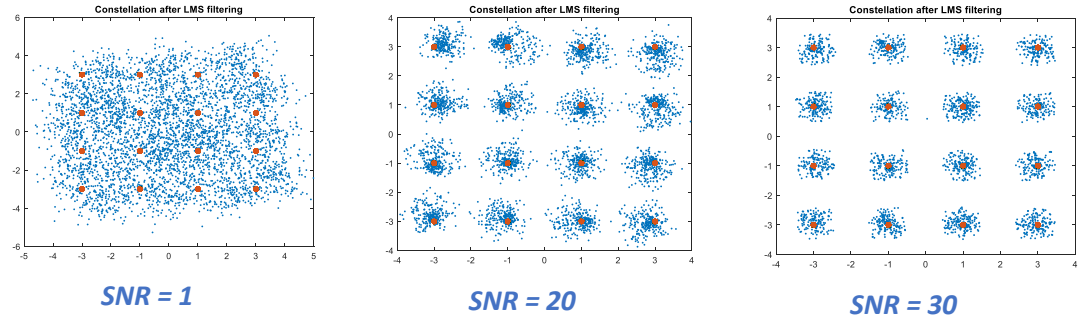
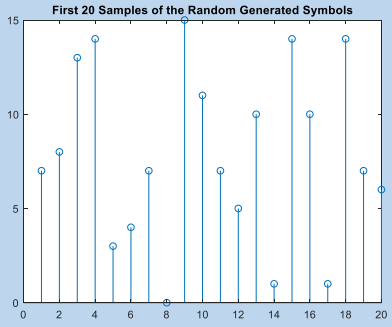
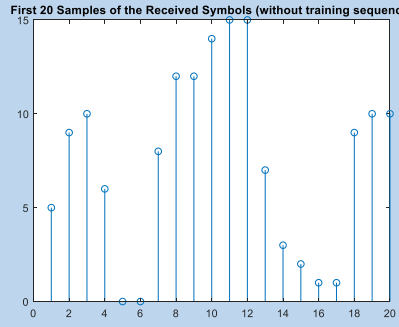
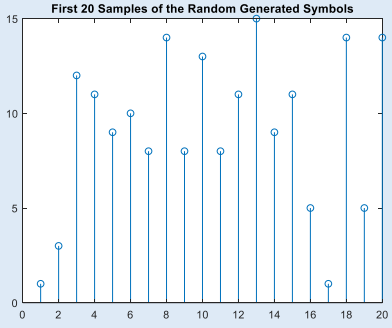
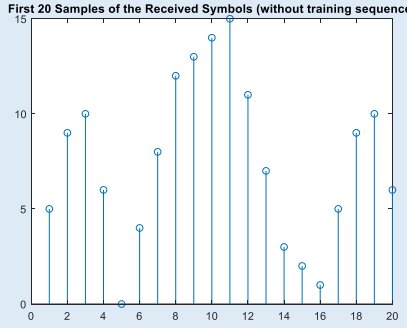


Figure 18: Constellations after LMS filter for different SNR values

So, obtaining the convolution of the received signal with the resulting filter coefficients and delaying the resulting signal the needed time (because of the delay produced by the convolution) we can recover the needed code:

Channel SNR	Original code (in Transmitter)	Recovered Code (in Receiver)
1		
15		

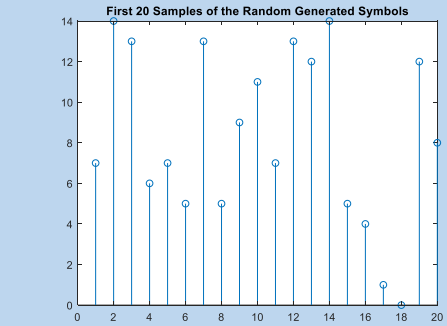
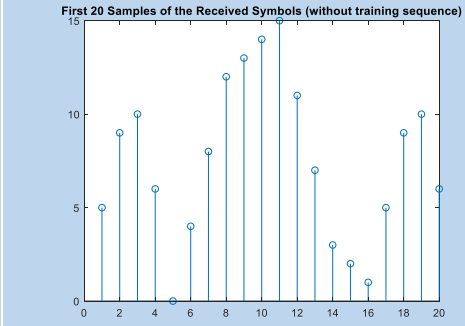
30		

Figure 19: Recovered Code using LMS algorithm for different SNR values

After this demodulation, the decoding is applied, so the bitstreams are obtained. Also, here are some results of the experiments:

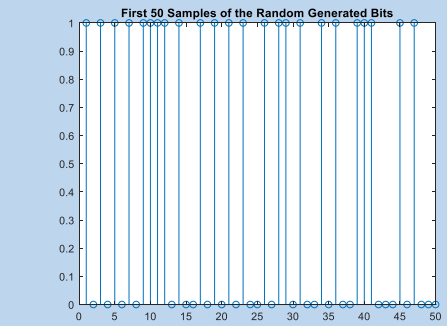
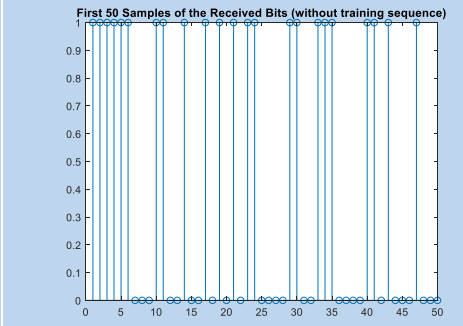
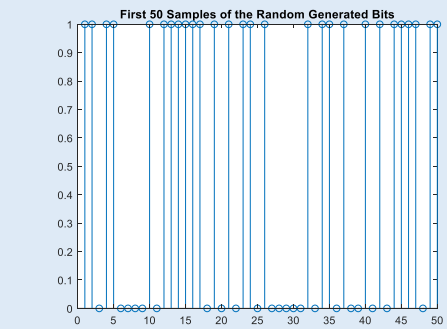
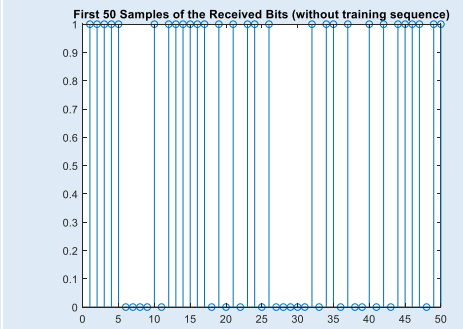
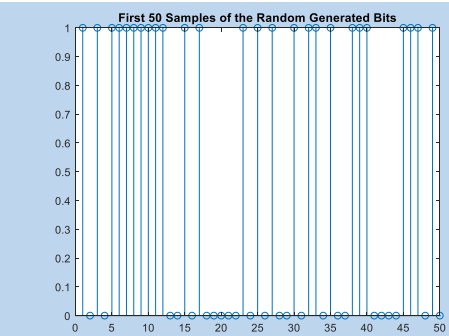
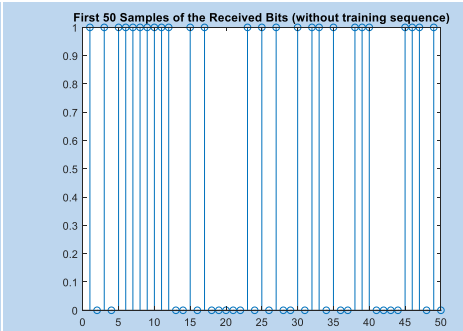
Channel SNR	Original signal (in Transmitter)	Recovered signal (in Receiver)
1		
15		
30		

Figure 20: Recovered Signal (Bitstream) using LMS algorithm for different SNR values

And about the measures of the full system, the average results after 5 executions of the full script for each SNR value (1, 15 and 30) using the LMS algorithm for tuning the receiver's filter, we have the next results:

Channel SNR	Number of errors	BER	EVM
1	1874.8	0.18748	0.50096
20	33.8	0.00338	0.50248
30	0	0	0.49988

Figure 21: Measurements of the System using LMS algorithm for different SNR values

All these calculations were made using the auxiliary excel file **Measurements.xlsx** which is included in the PR_K5123.zip file.

c. Adaptive filter modelled by RLS algorithm

When we use the RLS algorithm to tune the input filter, the first thing with is necessary is to train this algorithm using the training filter; after the training is completed, we can use the resulting filter to obtain a better solution to our problem of recover the original transmitted signal. For these experiments, the parameters used for RLS algorithms are: **L=2, α=0.01, γ=150 and λ=0.8.** For the training to be completed, is necessary that the |e(n)| signal goes to a value near to zero. We can see that this happens for all the SNR used for the experiments.

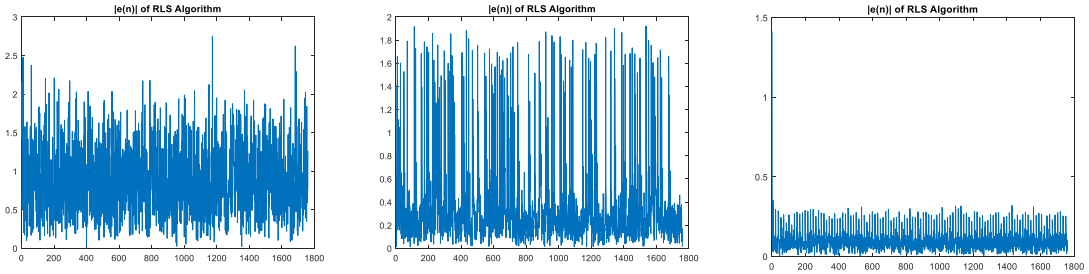


Figure 22: |e(n)| signal of RLS algorithm for different SNR values

The obtained constellations after the filter execution are the next ones:

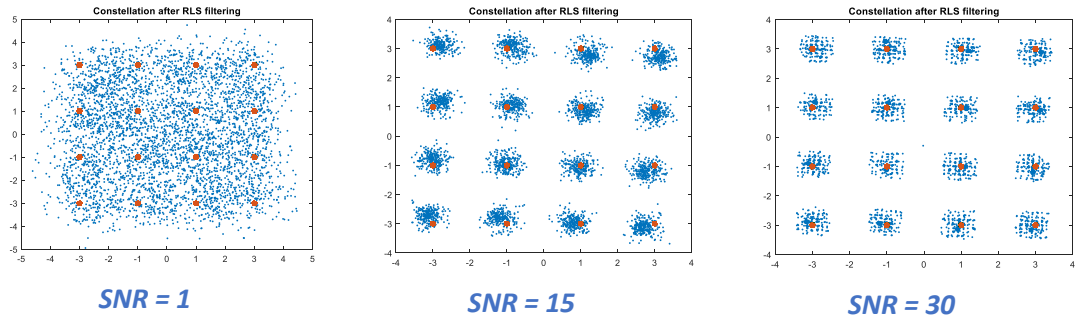


Figure 23: Constellations after RLS filter for different SNR values

So, obtaining the convolution of the received signal with the resulting filter coefficients and delaying the resulting signal the needed time (because of the delay produced by the convolution) we can recover the needed code:

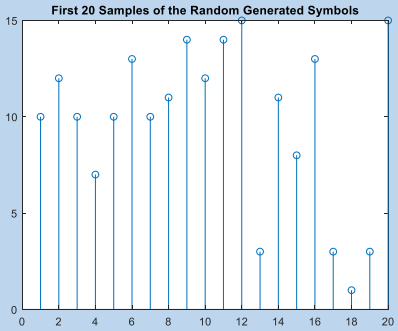
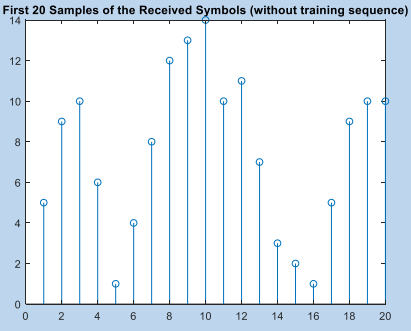
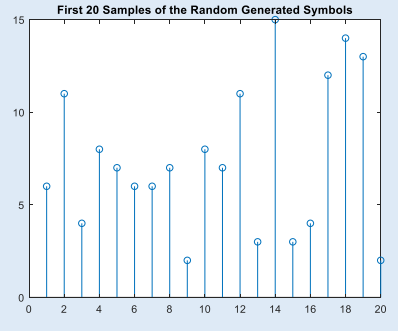
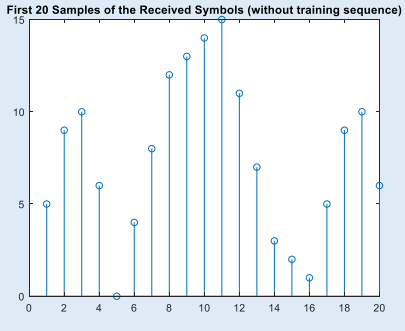
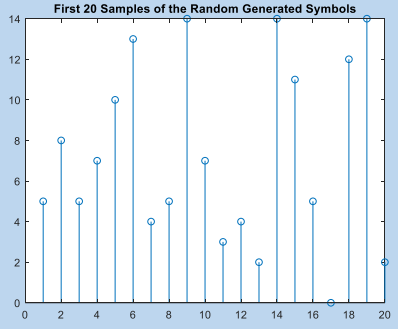
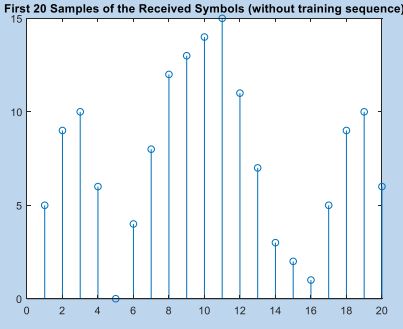
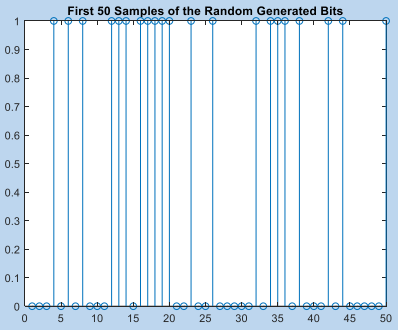
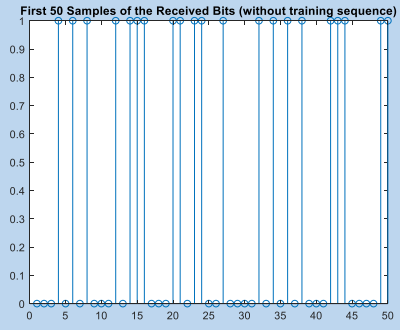
Channel SNR	Original code (in Transmitter)	Recovered Code (in Receiver)
1		
15		
30		

Figure 24: Recovered Code using RLS algorithm for different SNR values

After this demodulation, the decoding is applied, so the bitstreams are obtained. Also, here are some results of the experiments:

Channel SNR	Original signal (in Transmitter)	Recovered signal (in Receiver)
1		

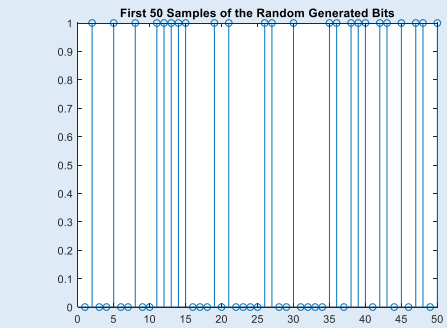
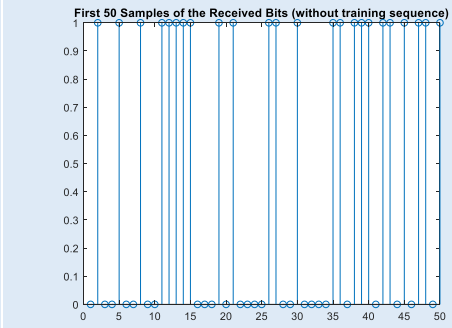
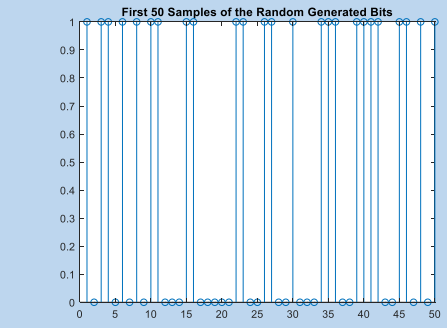
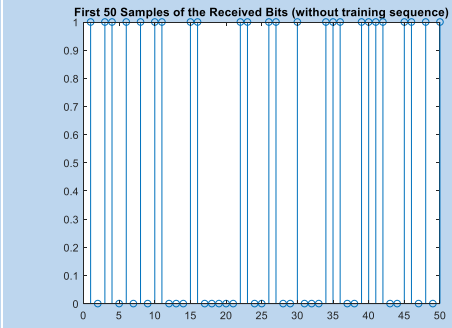
15		
30		

Figure 25: Recovered Signal (Bitstream) using RLS algorithm for different SNR values

And about the measures of the full system, the average results after 5 executions of the full script for each SNR value (1, 15 and 30) using the LMS algorithm for tuning the receiver's filter, we have the next results:

Channel SNR	Number of errors	BER	EVM
1	**2120.8**	**0.21208**	**0.50106**
15	**15**	**0.0015**	**0.5014**
30	**0**	**0**	**0.5018**

Figure 26: Measurements of the System using RLS algorithm for different SNR values

All these calculations were made using the auxiliary excel file **Measurements.xlsx** which is included in the PR_K5123.zip file.

d. Comparative graphics

If we plot all the measurements taken, we can observe the comparative of the results between the situations. All these plots have been made using the auxiliary excel file **Measurements.xlsx** which is included in the PR_K5123.zip file.

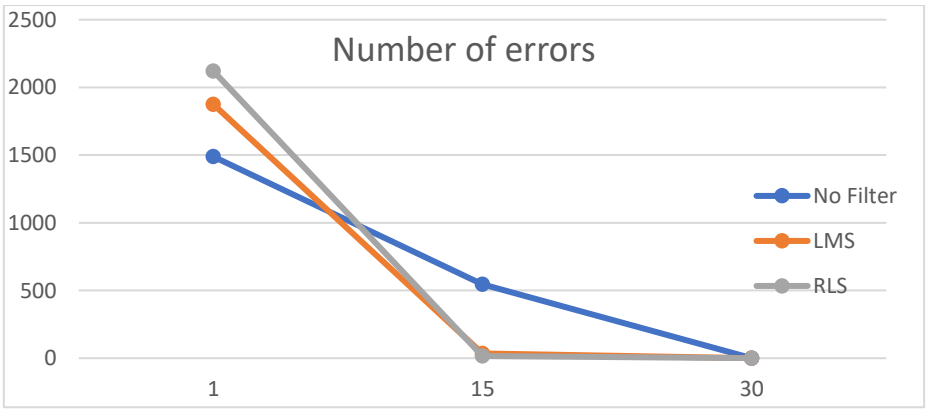


Figure 27: Number of errors comparation

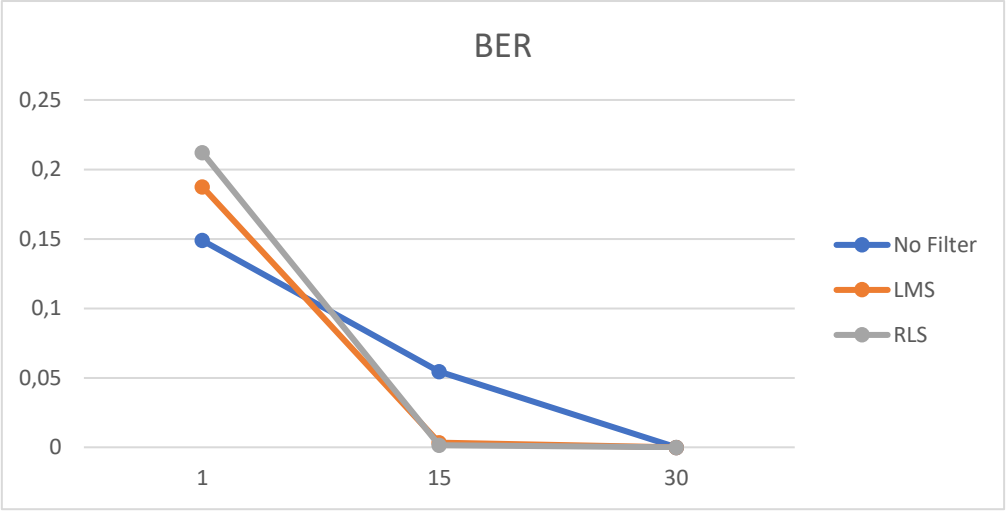


Figure 28: Bit Error Rate comparation

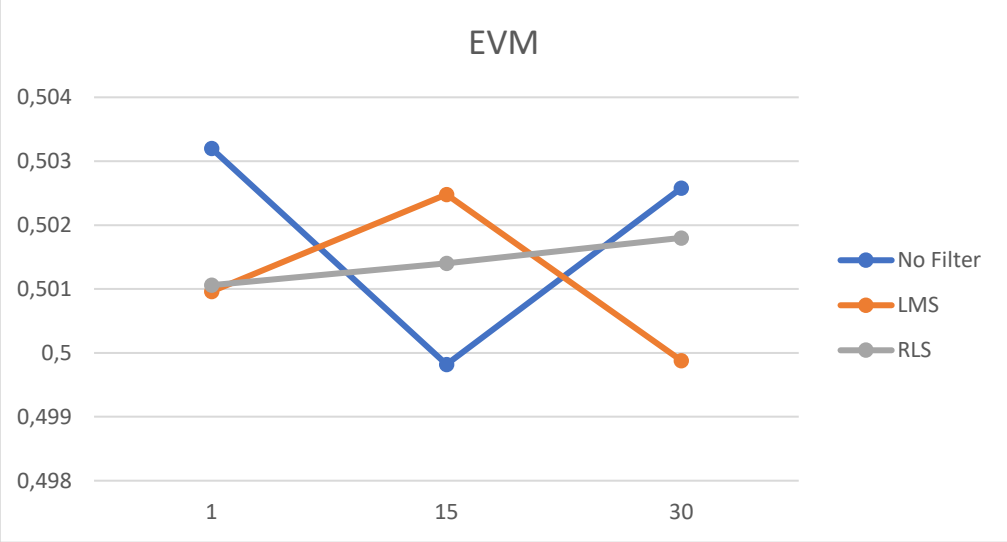


Figure 29: Error Vector Magnitude comparation

V. Analysis of Results and Conclusions

i) Signal Generation and Transmission

Analysing the Transmitter results, we can observe that all the aims of the project in this aspect have been reached. We can see that the system generates a random binary signal of N bits (for the experiments **N = 10000**), which is codified into the M symbols code (in this case **M = 16**) in a good way, as we can observe in the Figures 10 and 11. This obtained code is modulated using the appropriate M-QAM.

This modulated signal is concatenated with the training sequence and sent to the receptor through the channel; we can see that the sent signal's constellation is the correct 16-QAM as we pretended, as can be seen in the Figure 12.

ii) Channel Influence: Filter Coefficients and SNR

Observing the Figure 13, we can see how the constellation of the sent signal (blue dots) changes compared with theoretical symbols (orange dots) when the signal goes through the channel. These changes are produces by the influence of the Channel Filter's Coefficients and the added AWGN, whose power is defined by the selected SNR.

a. Filter Coefficients

In the Figure 13, we can observe that the points of the sent training sequence differ from the original signal. This is because of the effect of the selected coefficients: $\mathbf{h = [1, -0.3]^T}$. We can see this effect better for the situations of SNR = 30. Also, it can be seen the effect in the full sent signal in the same figure.

b. SNR

We can see also the influence of the addition of AWGN to the sent signal into the Figure 13. The noise produce that the sent symbols differs from the theoretical point, also with the effect of the channel's filter.

Observing the images, it can be seen the SNR influence on the original sent signal: the bigger is the value of the SNR parameter, the more similar are the sent and the received signals, this is because the level of the Signal Power value increases against the Noise Power value if we increase the SNR value. So, it is clear that the best situation is done for the case of SNR = 30, were the constellation dots are more aligned with the sent ones.

iii) Signal Reception and Recovery of the original Bitstream

After several investigation good results have been obtained. We can observe from Figure 27 that for the case of SNR = 1, the best situation is the case of no filter, this is because the big quantity of noise affects so much to the training sequence, so the filter is not well trained.

But nevertheless, we can see that the situation is clearly better if there is filter when the SNR = 15, for both cases, LMS and RLS, the number of errors is quite small in comparation to the case of no filter. The situation is better in the case of the RLS filter than in LMS, as it can be seen on the Figures 27 and 28. So, we can confirm that the use of adaptive filter allows us to recover the original signal in a better way under these conditions.

For the case of SNR = 30, the quantity of noise doesn't affect the signal so much, so in all situation the original signal is obtained, but even in this situation, we can see that the EVM is smaller for the cases of the filter than for the case of no filter (Figure 29). This is because the recovered

points of the constellation by the filter are closer to the theoretical ones, as can be seen in Figures 13, 18 and 23.

iv) General conclusions

The general conclusion of this project are the next ones:

- The aim of simulate a full telecommunication system that is able to make all the operations of the Transmitter and Receiver under some conditions of the channel has been reached.

- The result of the implementation of the adaptive filter in reception is the desired one, it takes the input samples and recovers the symbol transmitted (or really closer one, so the decoder is also able to recover the original)

- We can see that RLS algorithm works better than LMS one, but the difference is not so big as the computational cost, so there is a decision that the designer engineer should take between these two parameters (number of errors and computational cost). Also, the system parameters influence in this decision.

- Also, all the developed code can be used for obtaining solutions to other problems, the decision function and the training sequence should be changed, but not the rest.

- Also, from different experiments, we can confirm that for these cases, the algorithm has a big influence of the system characteristics so the input parameters should be elected in a very specific way to obtain good results, because if not, results can we even worse than using no adaptive filter.

VI. References

[1] Adaptive Signal Processing (EASP) lectures: https://studia.elka.pw.edu.pl/file/18Z/103A-TCTCM-MSA-EASP/priv/Lectures/

[2] P. M. Clarkson, Optimal and adaptive signal processing.

[3] MathWorks, MATLAB help: https://es.mathworks.com/help/matlab/index.html

[4] Digital Communication (EDICO) lectures: https://studia.elka.pw.edu.pl/file/18Z/103A-TCTCM-MSA-EDICO/lim/

[5] B. Sklar, Digital Communications.

[6] Fabio Pittalà, Fabian N. Hauske, Yabin Ye, Idelfonso T. Monroy, Josef A. Nossek; Training-based Channel Estimation for Signal Equalization and OPM in 16-QAM Optical Transmission Systems: https://mediatum.ub.tum.de/doc/1163451/1163451.pdf

[7] C. C. Do, A. V. Tran, C. Zhu, and E. Skafidas, Training Sequences in 16-QAM and QPSK Coherent Pol-Mux Single-Carrier Systems: file:///C:/Users/Guillermo%20P/Downloads/Trainingsequencesin16-QAMandQPSKcoherentpol-muxsingle-carriersystems.pdf