HW#4

SIMULATION REPORT

Fateme Noorzad | AFT | Azar 1398

Problem 1:

First of all, the *4-QPSK* symbols need to be simulated. For this goal the signal size is set 1000. Meaning there are 4000 samples of sent symbols.

As been defined in the problem, all sent symbols are i.i.d. and have the same probability of happening. With that is mind, a random number is chosen between 0 and 1 uniformly. If the generated number is less than 0.25, the first symbols is sent. If the generated number is between 0.25 and 0.5, the second number is sent, and so on. In this way 1000 samples of sent symbols are generated.

Second step is to create vector $\underline{u}(i)$. This vector is the output of channel having 4000 columns (the number of samples of sent symbols). Based on these and impulse response of channel, $\underline{u}(i)$ is simulated. (The z-transform and finding one of the elements of this vector is written in HW.)

Third step is to create vector $\underline{x}(i)$. This vector is the result of adding a Gaussian complex noise to $\underline{u}(i)$. Real and imaginary parts of noise has Gaussian distribution with zero mean and a variance found based on the given SNR. As been calculated in paper solution, its variance is 0.0494. So a noise with 4000 samples is created and added to $\underline{u}(i)$ in order to make $\underline{x}(i)$.

Here if the loop is in training mode, training data is used. Meaning data coming to find error, is actually a shift of $s(i)$. The optimum shift (Δ) is found is the first part of the problem. Based on its value the desired signal is created. For samples with index less than Δ, desired signal is zero and for more than that is s(i).

Now let's continue with problem:

PART 2: SIMULATING LMS AND SD AND COMPARING MISSADJUSTMENTS:

$\underline{u}(i)$ and $\underline{x}(i)$ are found as has been explained. In this part $\Delta = 19$, so based on this desired signal (d) is found and simulated.

LMS simulation:

LMS algorithm uses the bottom formula to update the weights of filter (here equalizer):

$$W(i+1) = W(i) + \mu e^*(i)x(i)$$

In the above formula, $\underline{W}(i)$ is a vector with size M*number of samples, containing all the founded weights. $e(i)$ is error which is :

$$e(i) = d(i) - y(i)$$

$$y(i) = W^H(i)x(i)$$

So based on the above formulas, y is going to be a number for each sample i making e be a number for each i^{th} sample. As a result W is a 35*1 vector.

To implement this algorithm $\underline{x}$ is cut in each iteration and then the above formula is used.

In this algorithm, in each iteration J, the expected value of error, is found. J is calculated using:

$$J\big(W(i)\big) = e * conj(e)$$

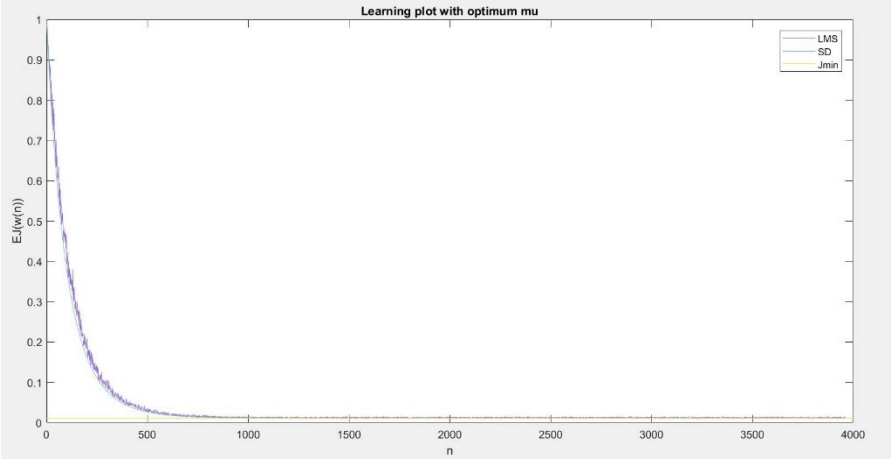
In this part the number of re-dos of iteration is said to be set 100. So *J(W(i))* are found for each i then all of the are summed up together and divided by 100. So actually *E{J(W)}* is calculated and shown as the output of algorithm.

For this part *M* is 10% :

$$\mu = \frac{٢ * ٠.١}{١٧٤.٦٢٩ * (١ + ٠.١)} = ١.٠٤١١٦٨٣١٦ * ١٠^{-٣}$$

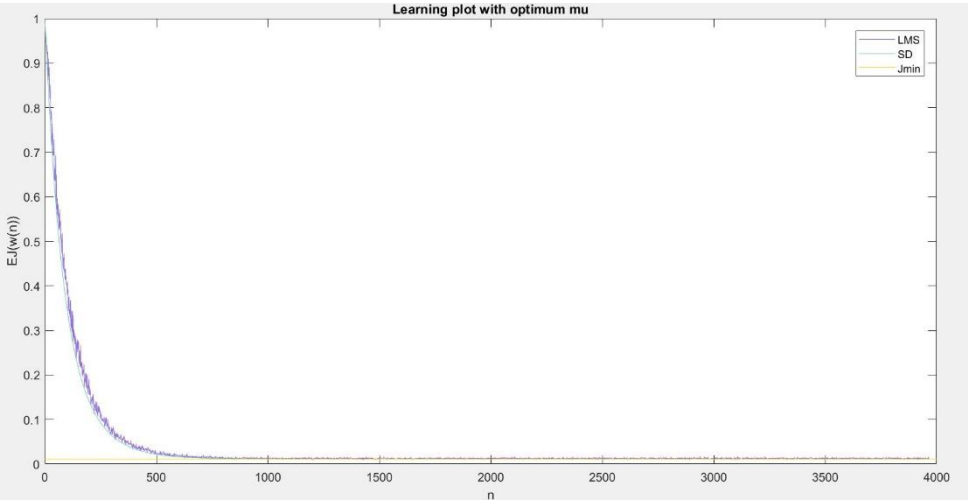
$$\mu \cong \frac{٢ * ٠.١}{١٧٤.٦٢٩} = ١.١٤٥٢٨٥١٤٧ * ١٠^{-٣}$$

For both of these values, the simulation is done. The results are:



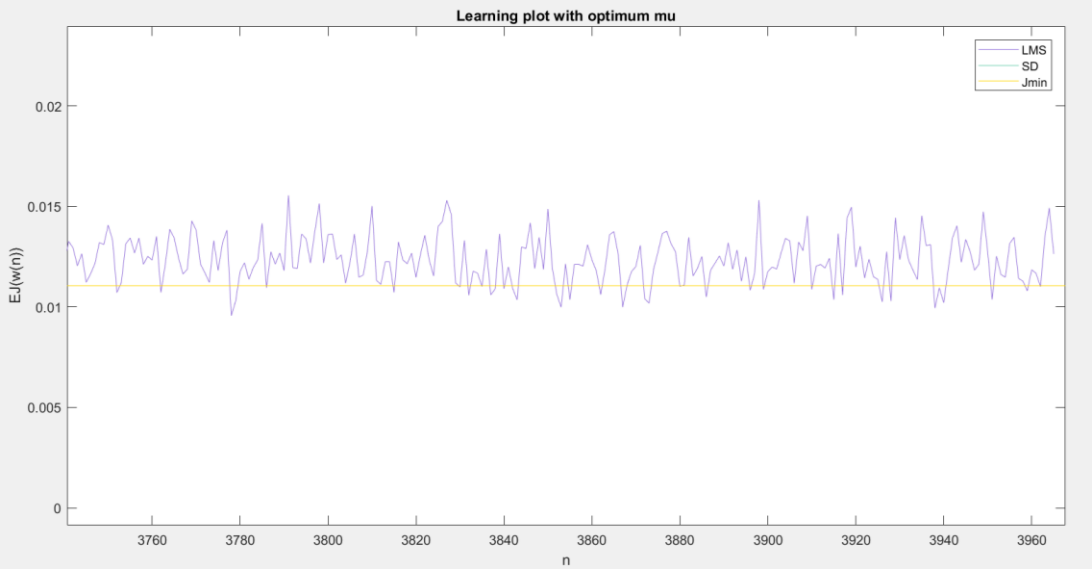
Plot1. Using the exact value of μ

```
With 4000 number of iterations (signal size) and 100 number of re-dos, misadjustment is :10.103114
```



Plot2. Using the approximated value of μ

```
With 4000 number of iterations (signal size) and 100 number of re-dos, misadjustment is :11.371465
```

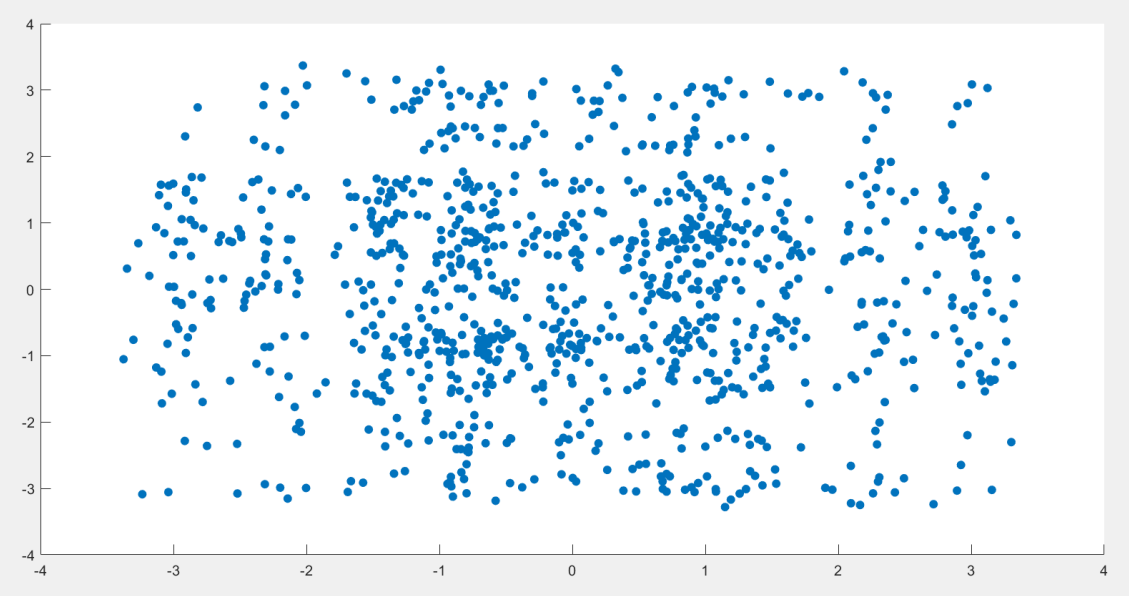


Plot3. Using the approximated value of μ (a closer look)

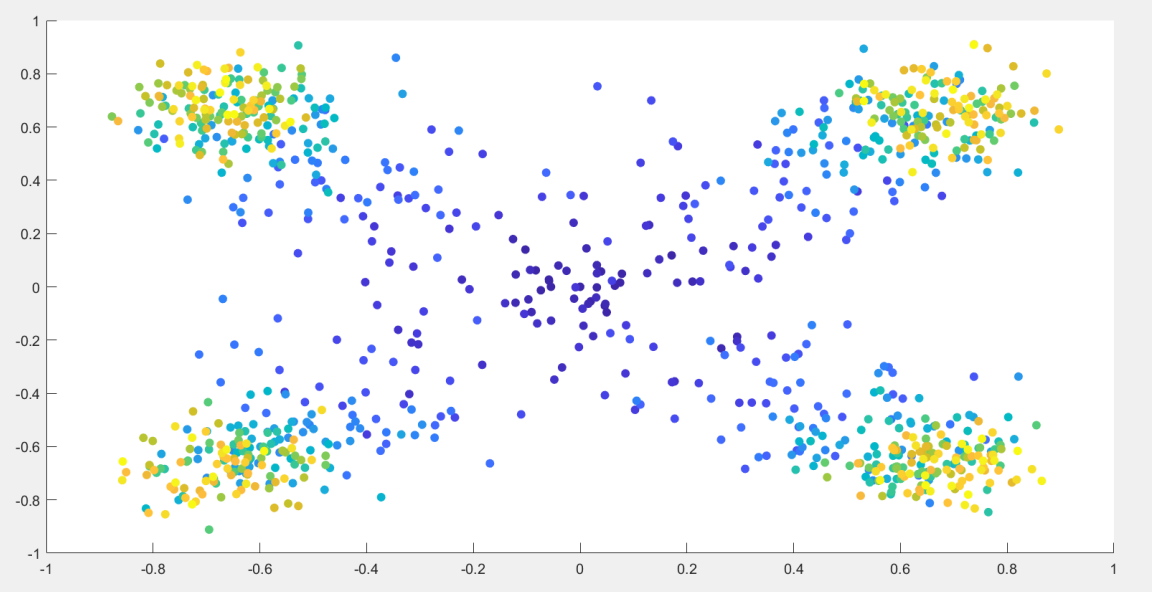
Because of what can be seen in plot 3, the average of last 10 data is used to calculate miss-adjustment.

PART3: SCATTERING PLOT FOR 1000 SAMPLES:

Using the implemented code above the scattering plot is drawn as below:



Plot4. Scattering plot of the equalizer input

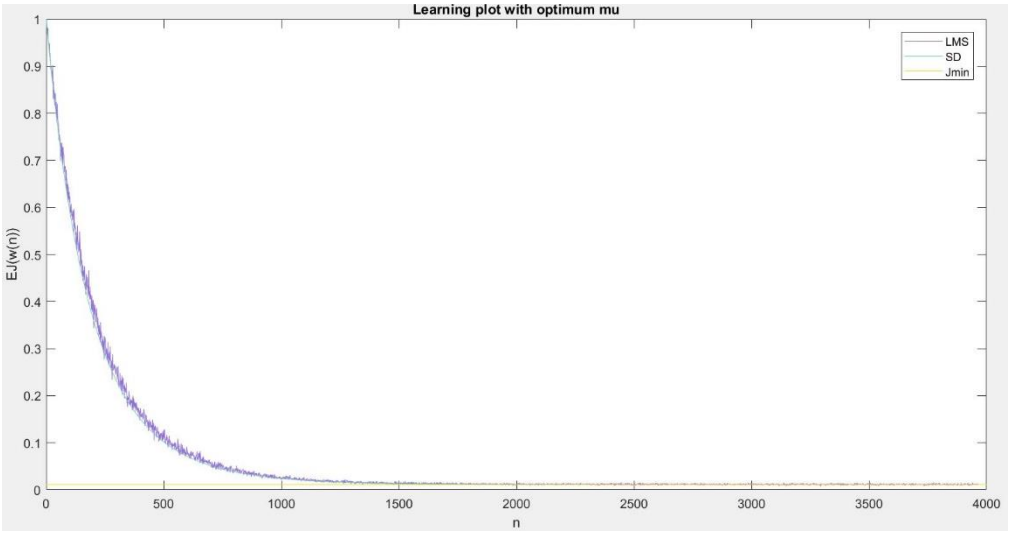


Plot5. Scattering plot of the equalizer output

As can be seen after passing the equalizer, symbols are in the place where they should be. Although there are some error and still some symbols are not in their exact place, more iterations can result in a better result.

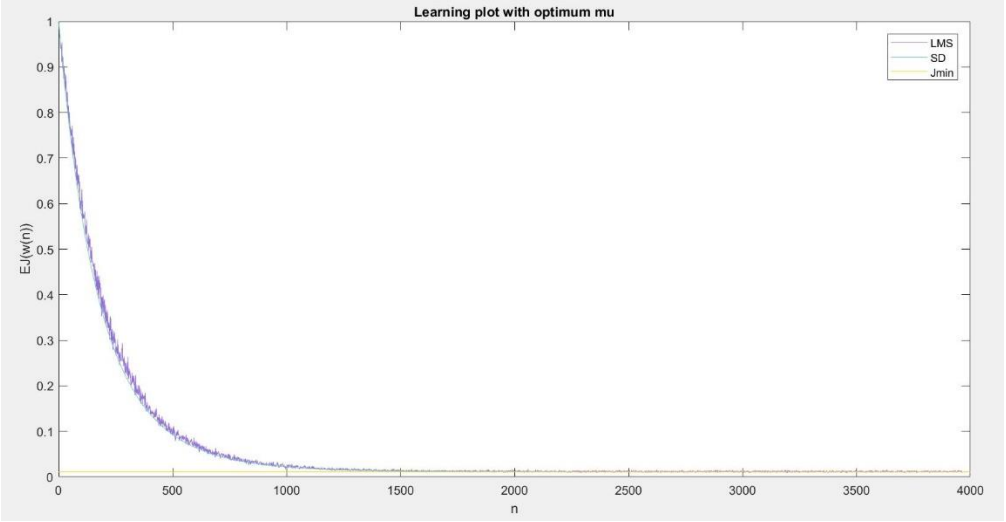
PART4: SIMULATION OF PART 2 WITH *M*=5% AND 1%:

For a 5% and 1% miss-adjustment the exact value and approximated of μ is found and then the simulation is done. First the results for *M*=5% is shown:



Plot6. Using the exact value of μ

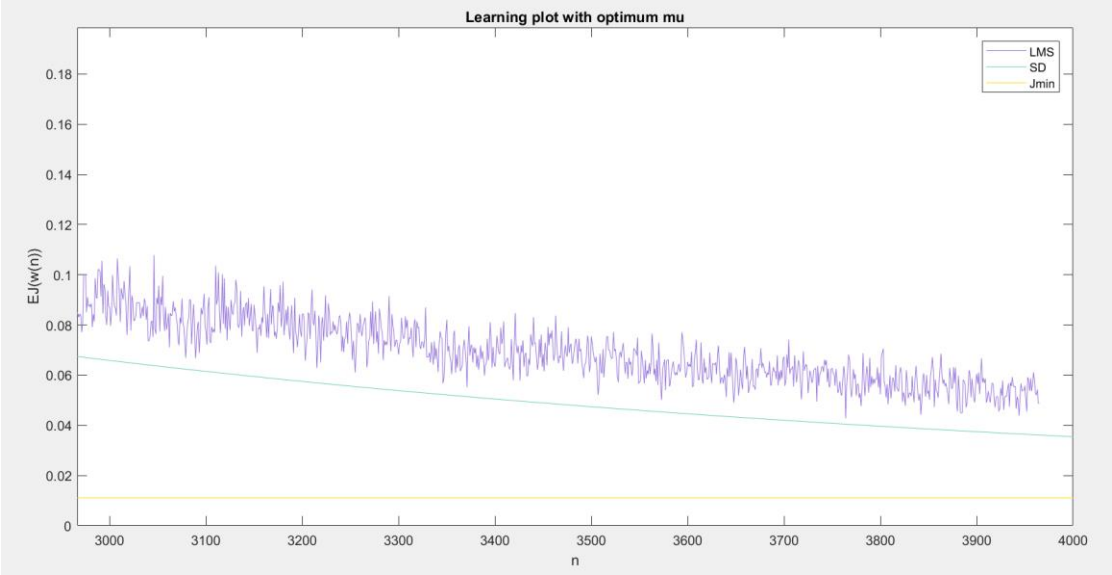
```
With 4000 number of iterations (signal size) and 100 number of re-dos, misadjustment is :4.456212
```



Plot7. Using the approximated value of μ

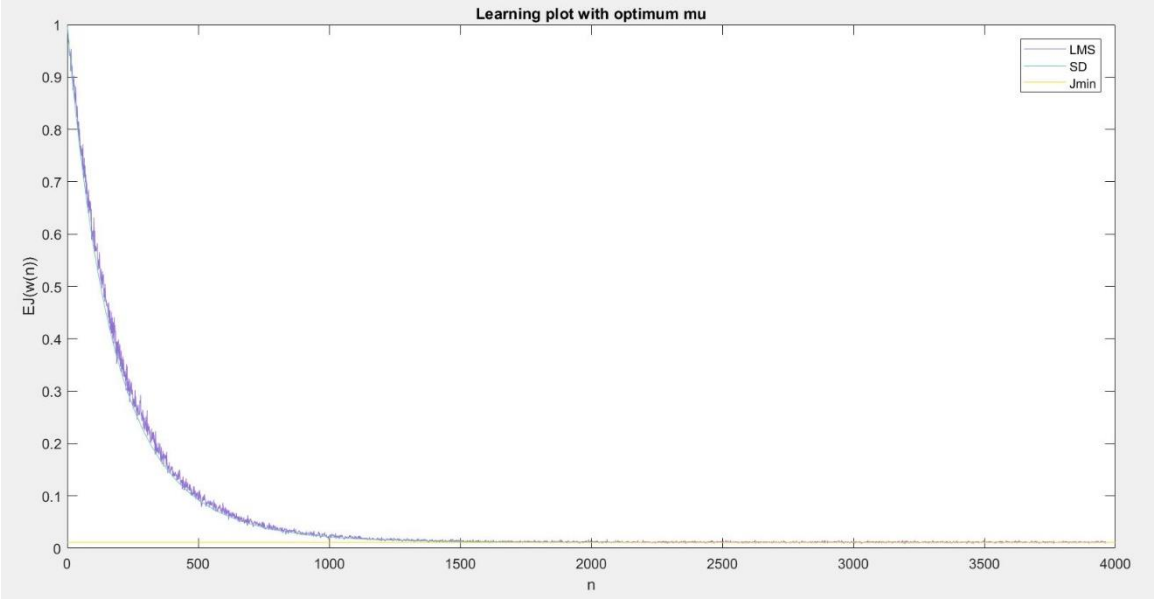
```
With 4000 number of iterations (signal size) and 100 number of re-dos, misadjustment is :4.918280
```

For a miss-adjustment this small since τ will be very big, the signal size which is the number of iterations in one loop must be larger to have a logical convergence. With the same data as before none of algorithms converge to Jmin. It can be seen in below plot:



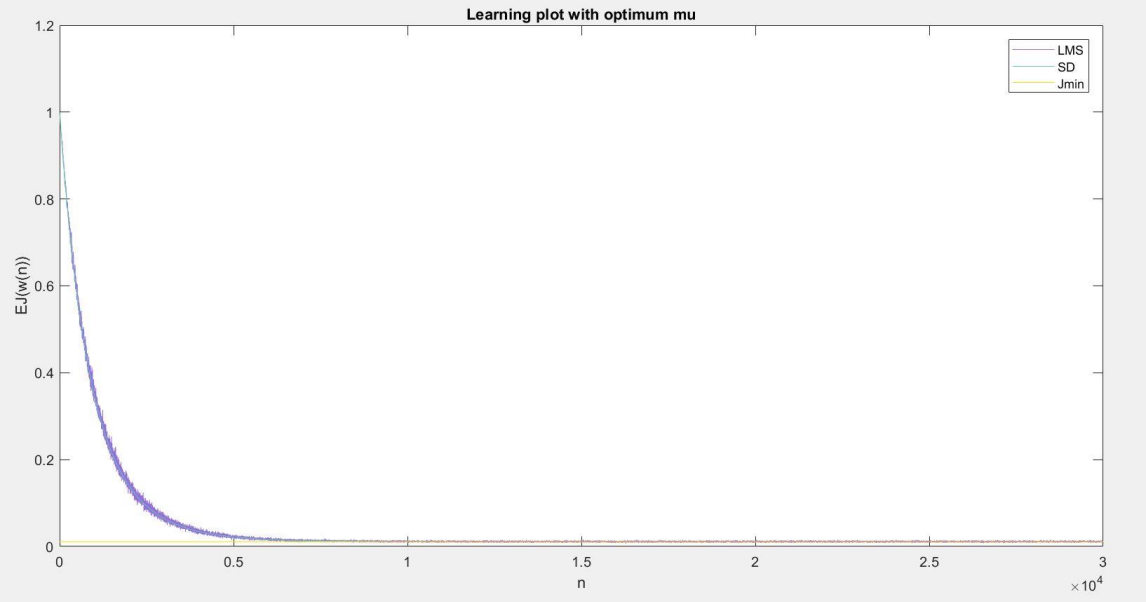
Plot8. No convergence

So signal size is changed and the results are:



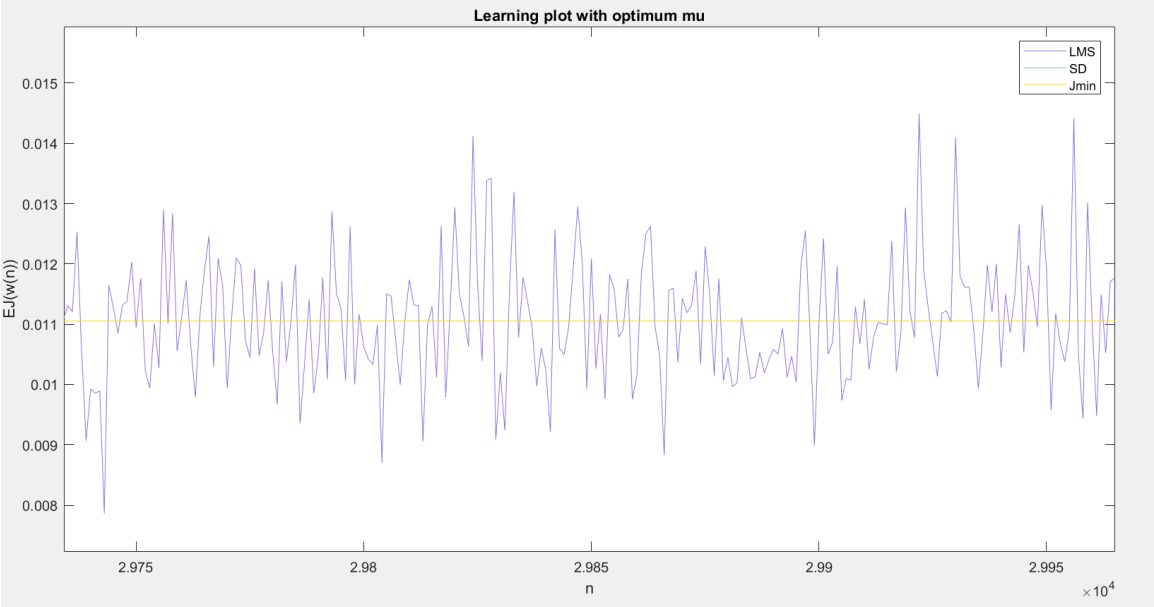
Plot9. Using the exact value of μ

```
With 30000 number of iterations (signal size) and 100 number of re-dos, misadjustment is :1.419610
```



Plot10. Using the approximated value of μ

```
With 30000 number of iterations (signal size) and 100 number of re-dos, misadjustment is :1.817964
```



Plot11. Using the approximated value of μ (a closer look)

Because of what can be seen in plot 9, the average of last 10 data is used to calculate miss-adjustment.

PART5: DECISION DIRECTED MODE

In this part, the system uses 2 modes, first mode is the training mode and the second one is decision directed. For about 100 iterations, the actual data is used meaning *d(i)*. After 100 iterations, the learnt data is used.

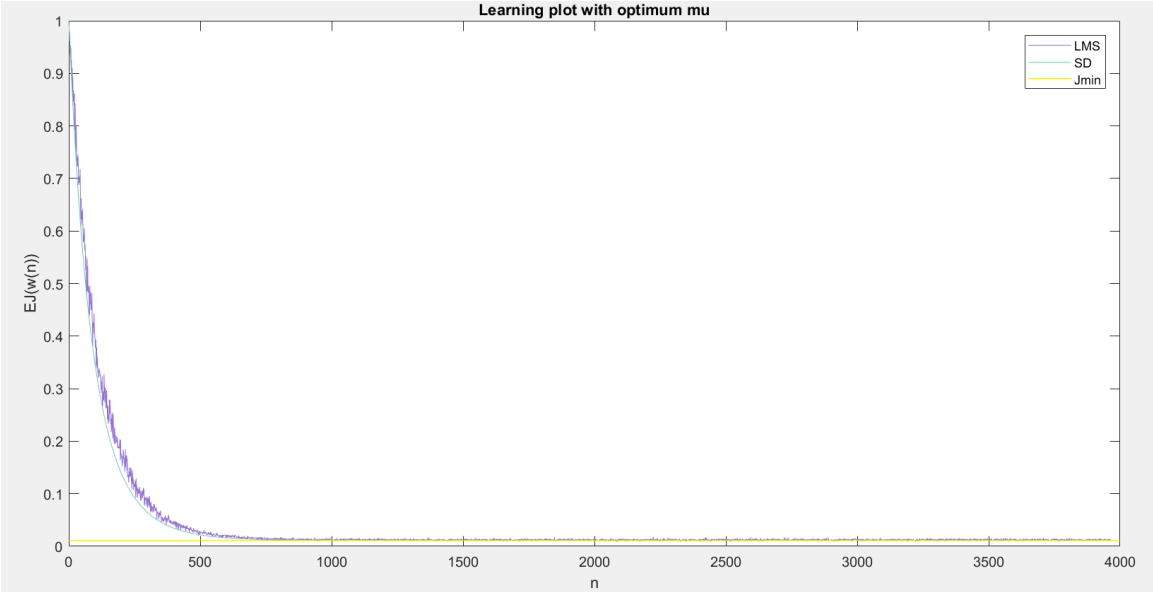
Making learnt data:

Here if the output of the equalizer has a positive real and imaginary parts, it means the first symbol is sent. If real part is positive but the imaginary part is negative, the second symbol is sent. If the real part is negative but the imaginary part is positive, the third one is sent and if none of the above conditions are true, the last symbol is sent. Based on this in the first 100 iterations 'dLearnt" is made.

Using learnt data:

After these iterations, instead of using d to find error, dLearnt is used. In the meantime, dLearnt is updated with the algorithm stated above.

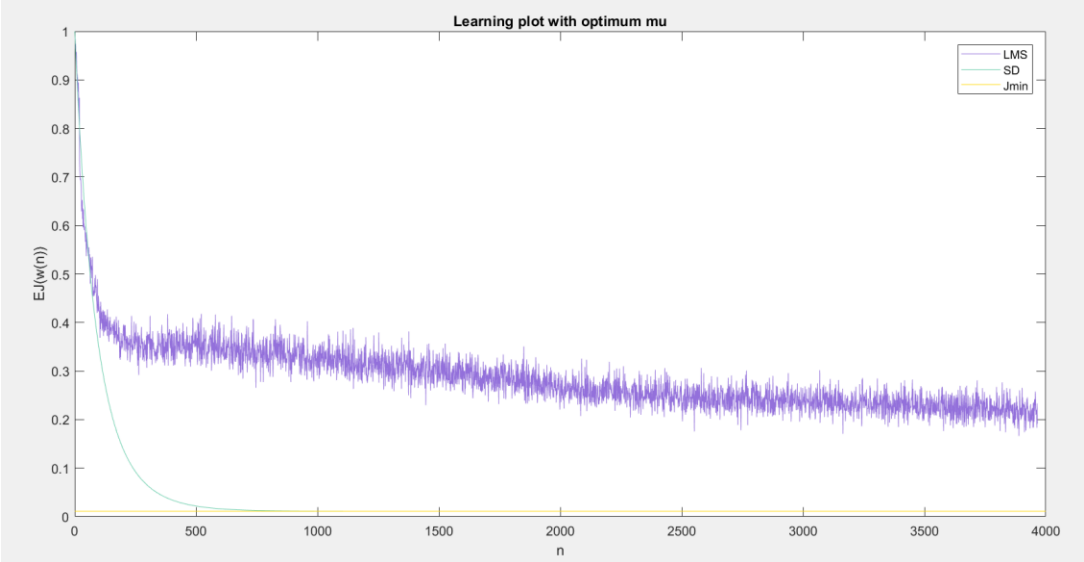
The result of its expected value of J can be seen in the plot below:



Plot12. Working in 2 modes.

```
With 4000 number of iterations (signal size) and 100 number of re-dos, misadjustment is :12.869552
```

As can be seen, its miss-adjustment is not as accurate as the last parts, because the learnt data is not as accurate as d. If the number of iterations were more, a better result could be found. Based on what been said if the same work is done for 20 data the result will be:



Plot13. Working in 2 modes

As above plot shows, for 20 data the convergence rate is too slow. Because the data we used to guess *d(i)* was not accurate because weights of the equalizer were not accurate. With more iterations it may lead to a better miss-adjustment, but this shows it is important when to stop using the actual data and use the trained data.

Problem 2:

PART2: USING LCMV TO DETECT ANGLE OF RADIATION:

In this question, like last ca, two complex normal signals are simulated. These signals model the sent signal from sources. They are independent and have power as 10 and 20. This state means their expected value is zero and their variance is 10 and 20.

Using "normrand" command in matlab, s1 and s2 signal which are the sent signals are created. Signal size is set as 1000 in this question.

For simulating noise the same story holds. "Normrand" command is used. But for noise variance is 1.

For both sent signals and noise, variance is divided by 2 for real and imaginary part of each of them.

Because of the distance between antennas which is $\frac{\lambda}{2}$, there'll be a delay between received signals in different antennas. If we assume the first antenna from left receives the signal with no delay, the second antenna will receive signal with a delay equal to $\exp(j2\pi cos\theta)$ and the M^{th} one's delay will be $\exp(j2(M-1)\pi cos\theta)$. (θ is the angle between the earth and signal beam.)

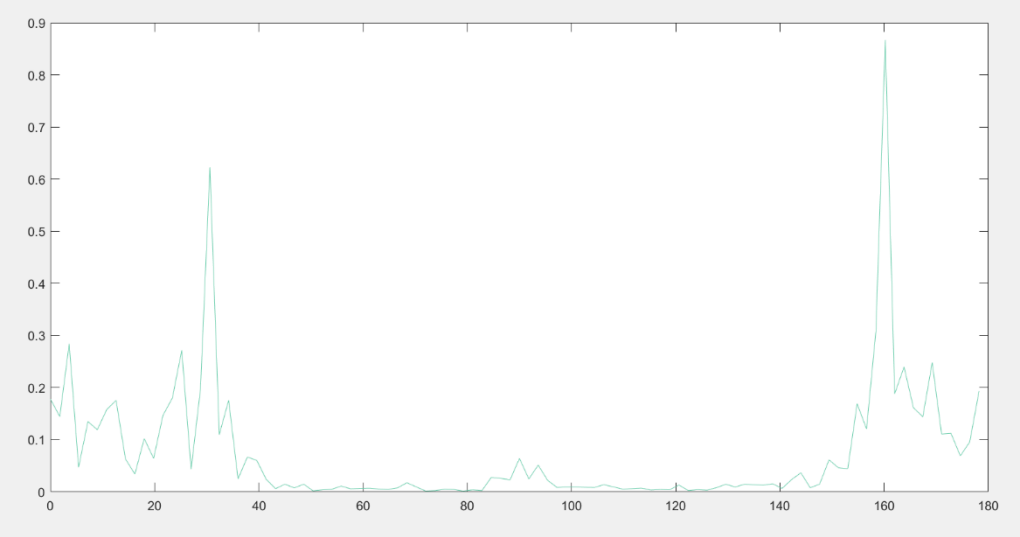
Based on this approach a matrix is needed to be defined containing these delay parameters. In this question two different values for θ is given. So 2 different matrix are defined. One for angle equal to 30 degrees and the other one 160 degrees.

The received signal will receive s1 and s2 and noise. But s1 and s2 will have a coefficient explained above, the delays. All the received signals are collected in a vector called u. Its components are the received signal of each antenna. Since here M is 6 and signal size is assumed as 100, u 'll be a matrix of size 100*6.

With the summation given, R_u is then calculated. It'll be a 6*6 matrix.

Then θ is defined to be between 0 and 180. Using θ now $c(\theta)$ is found the same way as $a(\theta)$ for 30 and 160 degrees.

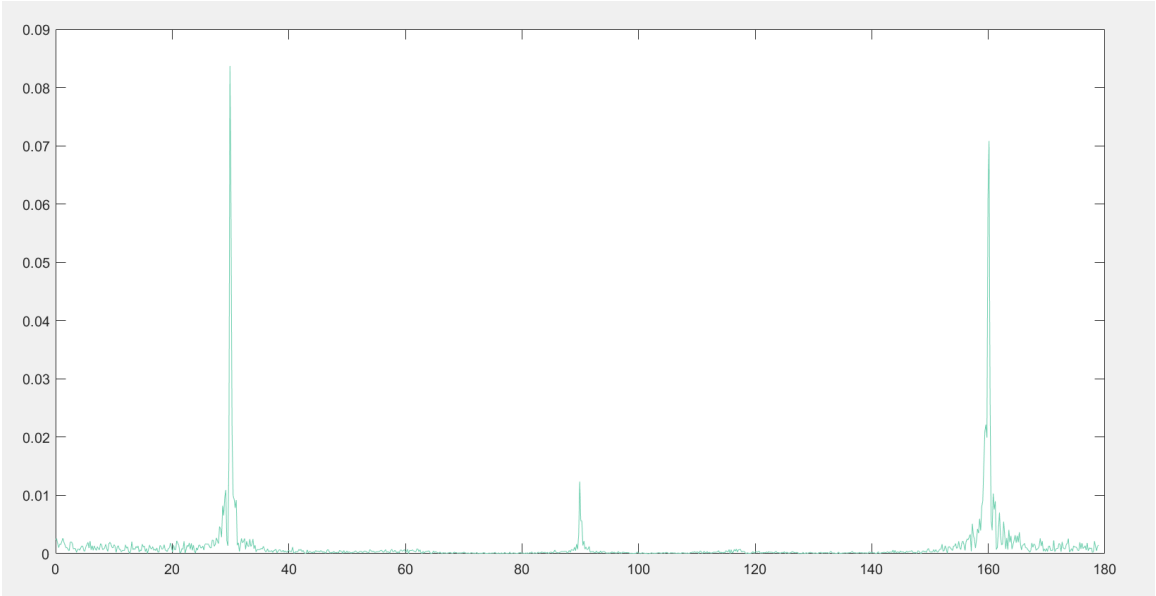
With $c(\theta)$ and the approximation of R_u, J_{min} is calculated. Now we need to plot the result. It'll be like this :



Plot14. Result of beamforming for N = 100

So this plot shows clearly, there are 2 peaks in 30 and 160. This shows that we can identify the angle of the beams. Since the angles are far different, peaks are clear and can surly state that these 2 are the angle of beams.

If we simulate for N = 1000 the result would be:



Plot15. Result of beamforming for N = 1000

Same as last part two peaks at 30 and 160 degrees happened.