University of Ottawa

Assignment 4 - Steepest descent, Newton, LMS and Newton-LMS algorithm

Jana Rusrus (300205310)

ELG5377 - Adaptive Signal Processing March 26, 2021

Question 1

R and P

The theoretical correlations functions of **R** and **P** were reused from the second assignment .

P vector

$P = [\phi_{xd}(0) \ldots \phi_{xd}(-N+1), \;\; \phi_{xd}(0) \ldots \phi_{xd}(-N+1)]^T$

R Matrix

$$R = \begin{pmatrix} \phi_{xx}(0) & \cdots & \phi_{xx}(N-1) \\ \vdots & \ddots & \vdots \\ \phi_{xx}(-N+1) & \ldots & \phi_{xx}(0) \end{pmatrix}$$

Power

$\sigma^2{}_d = 7249.25$

Eigenvalue Spread

$$\frac{\lambda_{max}}{\lambda_{min}} = \frac{1.4445 \times 10^3}{0.2551} = 5.6617 \times 10^3$$

Question 2

Algorithms

Steepest Descent

μ *Range* = $\left[0, \frac{2}{\lambda_{max}}\right]$ = $[0, 1.4 \times 10^{-3}]$

$\mu_{SD} = 1 \times 10^{-3}$

$W(k+1) = w(k) - \mu_{SD}\,(R\,W(k) - P)$

Newton Algorithm

μ *Range* = [0 , 2], not dependent on eigenvalues

$\mu_{Newton} = 1$

$W(k+1) = w(k) - \mu_{Newton}\,(R^{-1}\,(R\,W(k) - P))$

LMS Algorithm

μ *Range* = $\left[0, \frac{2}{3\,\mathrm{Tr}(R)}\right]$ = $[0, 2.64 \times 10^{-4}]$

$\mu_{LMS} = 1 \times 10^{-4}$

$W(n+1) = w(n) - \mu_{LMS}\,(X(n)\,e^{*}(n))$

Newton - LMS Algorithm

μ *Range* = $\left[0, \frac{2}{N}\right]$ = $[0, 0.02]$

$\mu_{NewtonLMS} = 0.012$

$W(n+1) = w(n) + \mu_{NewtonLMS}\,(\boldsymbol{R}^{-1}\,X(n)\,e^{H}(n))$

Errors

Normalized MSE learning curve

$\frac{\zeta(w(n))}{\sigma^2{}_d} = \frac{\sigma^2{}_d - w^H(n)\,P - w^T(n)\,P^{*} + w^H(n)\,\mathrm{R}w(n)}{\sigma^2{}_d}$; (this represents subplot 3)

Squared Error magnitude

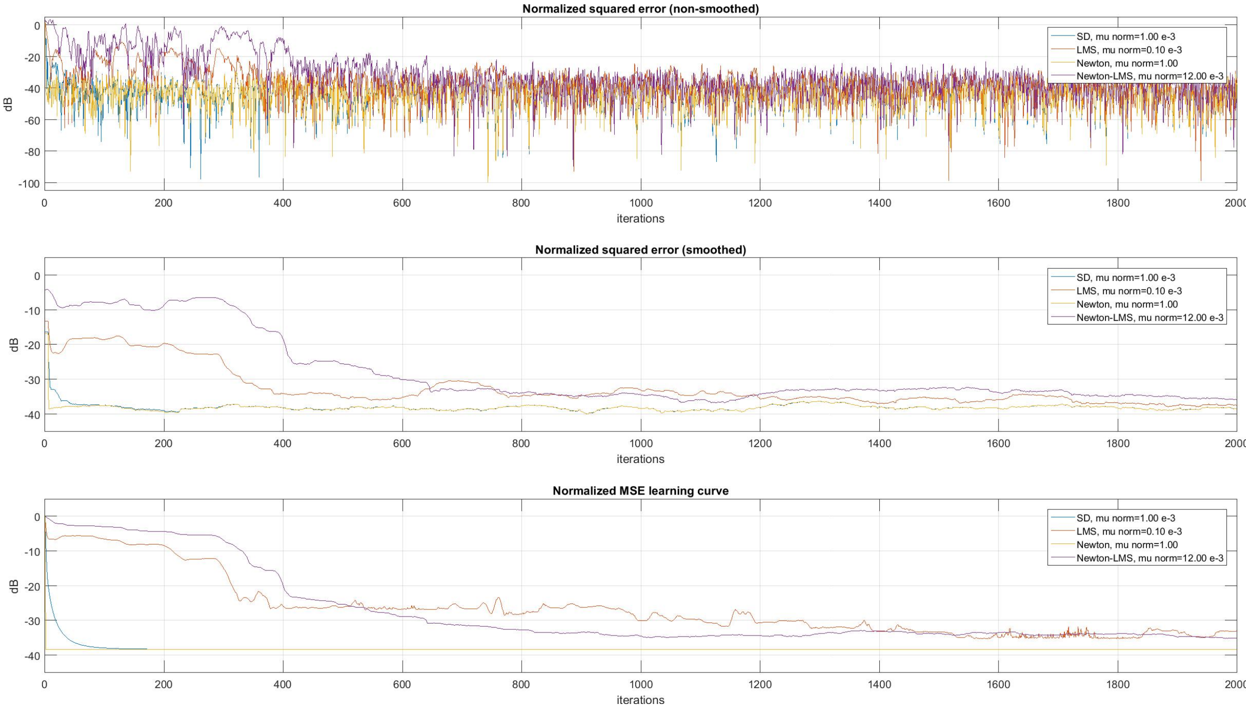
$e(n) = d(n) - w(n)^H\,x(n)$; (this represents subplot 1)

MMSE

$\frac{\zeta(w_{opt}(n))}{\sigma^2{}_d} = \frac{\sigma^2{}_d - P^H\,R^{-1}\,P}{\sigma^2{}_d} = -38.352$; (this represents reference-line marked in black)

Algorithms step size set to have fastest initial convergence

Out[•]=



Note

- We note that the Newton’s method seems to converge more rapidly than the steepest descent method. It converges in one iteration.
- Steepest Descent is pretty fast, because it uses the ideal **R** and **P.** It can use a much larger step size than other algorithms.
- In this case, since Steepest Descent uses a larger step size than LMS, it converges faster .
- Since the error in the second subplot is computed from one realization and the error in the third one computed over million of simulations, there is a difference between these two subplots. If the two figures computed over million of samples there would be no difference between them.
- Newton’s method does indeed generally converge more rapidly, because it uses information of **R** matrix to find the direct path on the error surface to $\zeta_{\min}$.
- The LMS is much simpler and easier to compute because it does not require offline gradient estimate or repetitions of data.
- The adaptive process of the LMS is noisy and is not following the true line of the steepest descent on the performance surface, since its weights changes at each iteration based on imperfect gradient estimates.
- LMS without averaging the gradient components does contain a large component of noise, but the noise is attenuated with time by the adaptive process, which acts as a low pass filter in this respect.

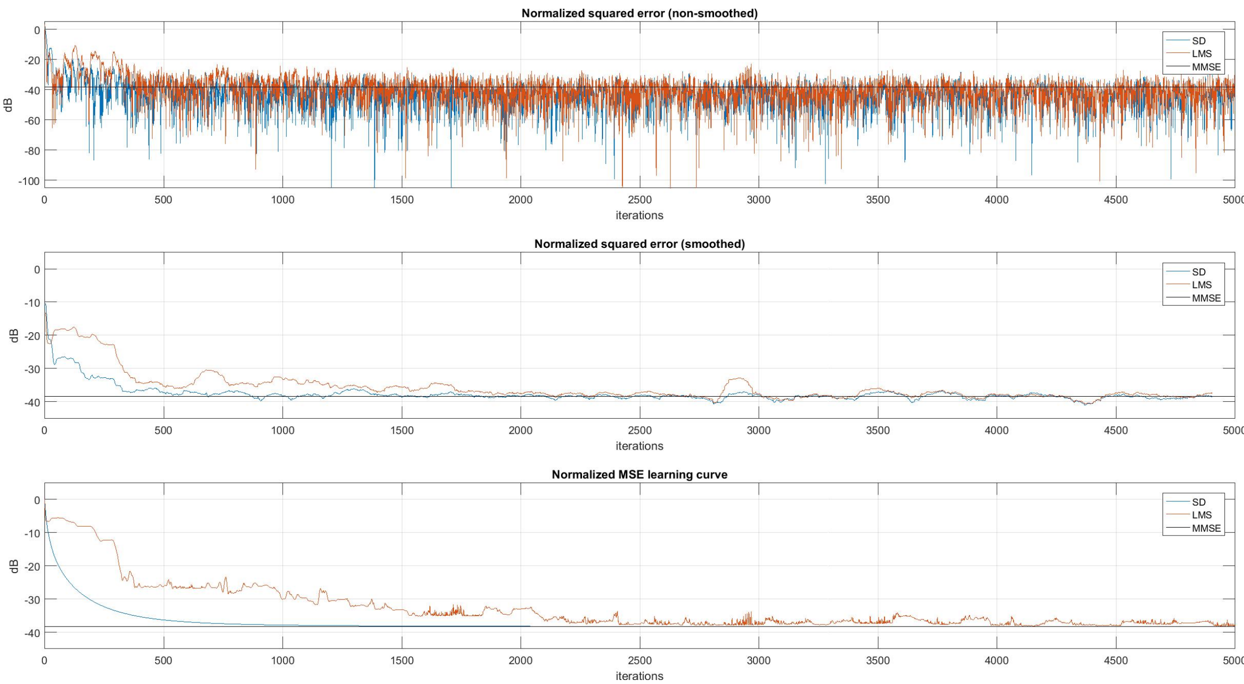
- The Range of LMS μ is more restrictive bound but much easier to apply because the elements of **R** and the signal power can generally be estimated more easily than the eigenvalues of **R.**
- The efficiency of LMS algorithm approach a theoretical limit for adaptive algorithms when the eigenvalues of **R** matrix are equal or nearly equal.

Question 3

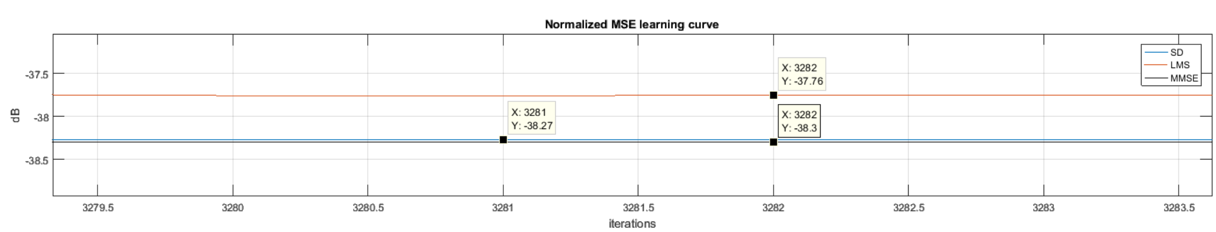
Comparing the convergence of the Steepest Descent and LMS algorithm using common μ value

Using $\mu_{\text{LMS}} = 1 \times 10^{-4}$ for both algorithms.

Out[∘]=



And we zoom on the two algorithm after steady state to compare with the Optimal solution



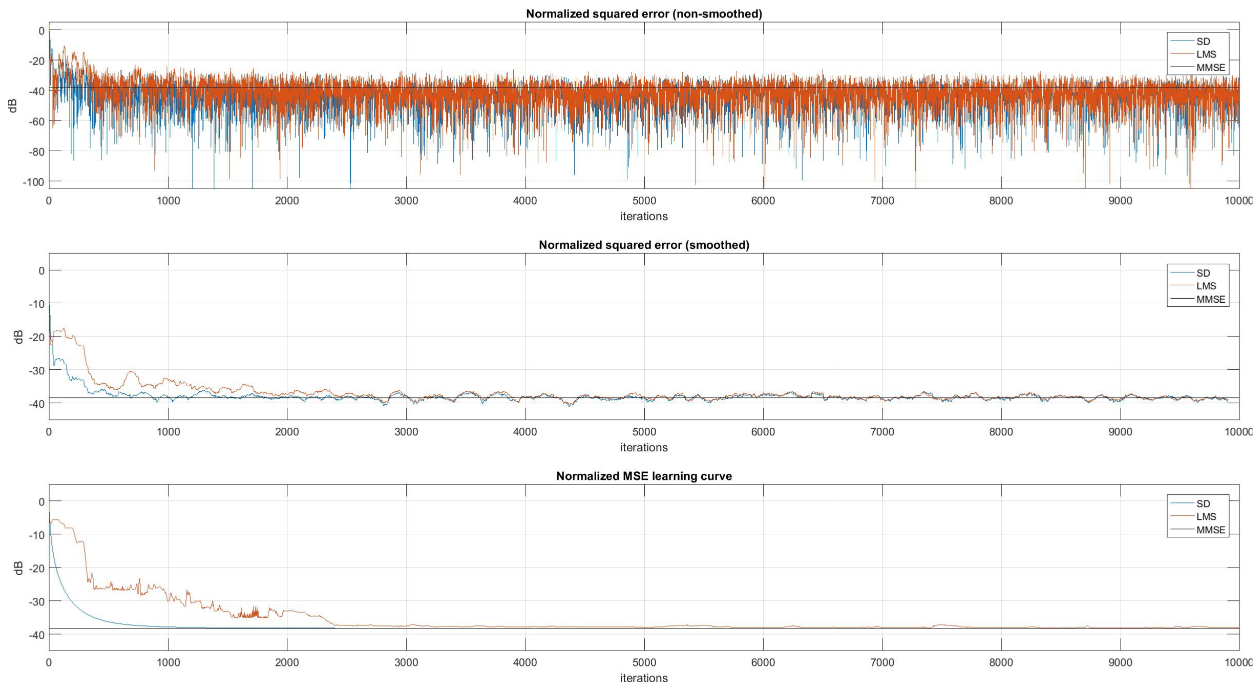
Note

- We note that the Steepest Descent method and LMS Algorithm roughly follow the same trajectory when they both use a common μ.

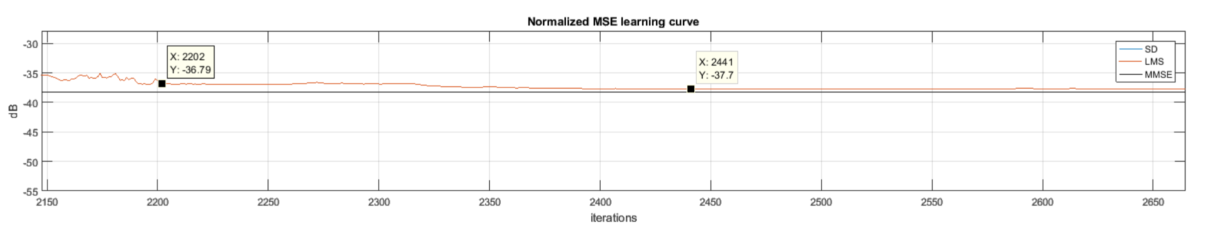
- Both algorithms are very similar but not the same because μ is large. This results in making SD outperform the LMS.
- If μ is small, they both will be the same.
- Looking at the zoomed figure, we note that the SD has no misdjustment. However, LMS has misadjustment. [I.e; when the LMS reaches the optimum it jumps around the optimum and never reaches it].
- LMS misadjustment = 0.2312. However, the theoretical value for LMS = μ Tr(R) / 2 = 0.1263
- The difference between the theoretical misadjustment and the practical one, can be caused by number of reasons. For example, the randomness of the signals and the estimation of **R** and **P**, and the numerical error.

Reducing the Misadjustment

Out[ ]=



Zooming over the steady state section, μ was reduced at iteration 2200 (i.e. at steady state) by a factor of 10.



Note

- Reducing μ allows the algorithm to improve a little bit (by 1dB). This shows that it is possible to squeeze a little of of a performance.
- However, the improvement was not immediate, it took some iterations for the algorithm to reach a new better steady state performance. This means that the original learning rate was good enough to reach close enough to the optimal solution.
- There is a trade off between the misadjustment and the rate of adaptation.