ADAPTIVE FILTERS
Solutions of Computer Projects

Ali H. Sayed

Electrical Engineering Department
University of California at Los Angeles

Readers are welcome to bring to the attention of the author at sayed@ee.ucla.edu any typos or suggestions for improvements. The author is thankful for any feedback.

CONTENTS

PART I: OPTIMAL ESTIMATION

COMPUTER PROJECT

Project .1 (Comparing optimal and suboptimal estimators) The programs that solve this project are the following.

1. psk.m This function generates a BPSK signal $\boldsymbol{x}$ that assumes the value $+1$ with probability p and the value -1 with probability $1-p$.
2. partB.m This program generates four plots of $\hat{x}_N$, as a function of N, one for each value of p — see Fig. 1.

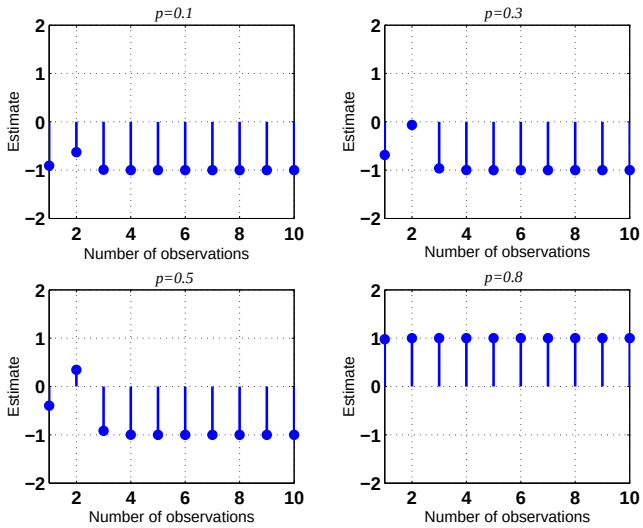

Figure I.1. The plots show the values of the optimal estimates $\hat{x}_N$ for different choices of N (the number of observations) and for different values of p (which determines the probability distribution of $\boldsymbol{x}$).

3. partC.m This program generates a plot that shows $\{\hat{x}_N, \hat{x}_{N,av}\}$ for four different values of p over the interval $1 \leq N \leq 300$ — see Fig. 2.
4. partD.m This program estimates the variances of $\{\hat{\boldsymbol{x}}_N, \hat{\boldsymbol{x}}_{N,av}, \hat{\boldsymbol{x}}_{dec}, \hat{\boldsymbol{x}}_{sign}\}$. Typical values are

$$\{0.0031, 0.1027, 0.0040, 0.0040\}$$

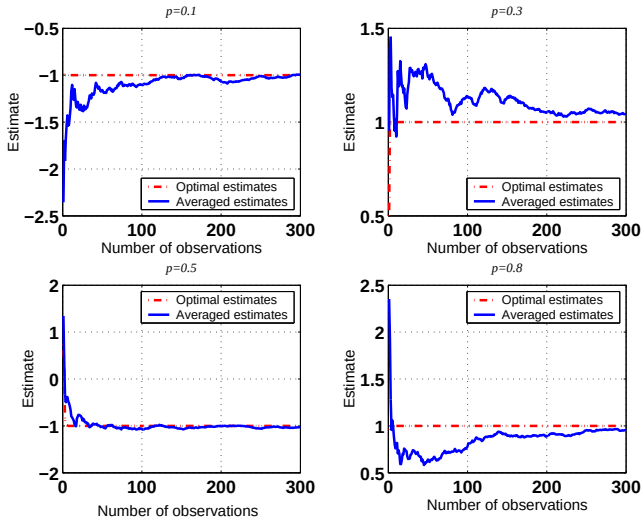

Figure I.2. The plots show the values of the optimal estimates $\hat{x}_N$ (*dotted lines*) and the averaged estimates $\hat{x}_{N,av}$ (*solid lines*) for different choices of N (the number of observations) and for different values of p (which determines the probability distribution of $\boldsymbol{x}$). Observe how the averaged estimates are significantly less reliable for a smaller number of observations.

PART II: LINEAR ESTIMATION

COMPUTER PROJECTS

Project II.1 (Linear equalization and decision devices) The programs that solve this project are the following.

1. partA.m This program solves parts (a) and (c), which deal with BPSK data. The program generates 5 plots that show the time and scattering diagrams of the transmitted sequence $\{\boldsymbol{s}(i)\}$, the received sequence $\{\boldsymbol{y}(i)\}$ (which is the input to the equalizer), and the equalizer output $\{\hat{\boldsymbol{s}}(i-\Delta)\}$ for several values of Δ. A final plot shows the values of the estimated m.m.s.e. and the number of erroneous decisions as a function of Δ. The program allows the user to change σ_v^2. Typical outputs of this program are shown in the sequel.

   Figure 1 shows four graphs displayed in a 2×2 matrix grid. The graphs on the left column pertain to the transmitted sequence $\boldsymbol{s}(i)$, while the graphs on the right column pertain to the received sequence $\boldsymbol{y}(i)$. The graphs on the bottom row are scatter diagrams; they show where the symbols $\{\boldsymbol{s}(i), \boldsymbol{y}(i)\}$ are located in the complex plane for all 2000 transmissions. We thus see that the $\boldsymbol{s}(i)$ are concentrated at ± 1, as expected, while the $\boldsymbol{y}(i)$ appear to be concentrated around the four values $\{-1.5, -0.5, 0.5, 1.5\}$. The graphs on the top row are time diagrams; they show what values the symbols $\{\boldsymbol{s}(i), \boldsymbol{y}(i)\}$ assume over the 2000 transmissions. Clearly, the $\boldsymbol{s}(i)$ are either ± 1, while the $\boldsymbol{y}(i)$ assume values around $\{\pm 0.5, \pm 1.5\}$.

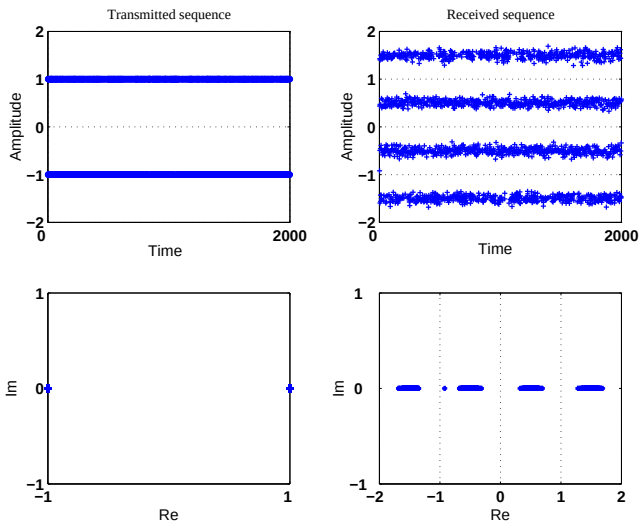

Figure II.1. The plots show the time (*top row*) and scatter (*bottom row*) diagrams of the transmitted and received sequences $\{\boldsymbol{s}(i), \boldsymbol{y}(i)\}$ for BPSK transmissions. Observe how the $\boldsymbol{s}(i)$ are concentrated at ± 1 while the $\boldsymbol{y}(i)$ are concentrated around $\{\pm 0.5, \pm 1.5\}$.

Figure 2 again shows four graphs displayed in a 2×2 matrix grid. The graphs on the left column pertain to the received sequence $\boldsymbol{y}(i)$, while the graphs on the right column pertain to the output of the equalizer, $\hat{\boldsymbol{s}}(i-\Delta)$, for $\Delta = 0$, i.e., $\hat{\boldsymbol{s}}(i)$. The graphs on the bottom row are the scatter diagrams while the graphs on the top row are the time diagrams. Observe how the symbols at the output of the equalizer are concentrated around ± 1.

Figure 3 is similar to Fig. 2, except that it relates to $\Delta = 1$. Comparing Figs. 2 and 3, observe how the scatter diagram of the output of the equalizer is more spread in the latter case. Figure 4 is also similar to Fig. 2, except that it now relates to $\Delta = 3$. Observe how the performance of the equalizer is poor in this case.

Figure 5 shows two plots: one pertains to the m.m.s.e as a function of Δ, while the other pertains to the number of erroneous decisions as a function of Δ. It is seen that although the choice $\Delta = 0$ leads to the lowest m.m.s.e. at the assumed SNR level of 25 dB, all three choices $\Delta = 0, 1, 2$ lead to zero errors.

Figure 6 again shows the time and scatter diagrams of the equalizer input (*left*) and output (*right*) using $\Delta = 0$ and $\sigma_v^2 = 0.05$, i.e., SNR= 14 dB. Observe how the equalizer performance is more challenging at this SNR level. Still, the number of erroneous decisions came out as zero in this run.

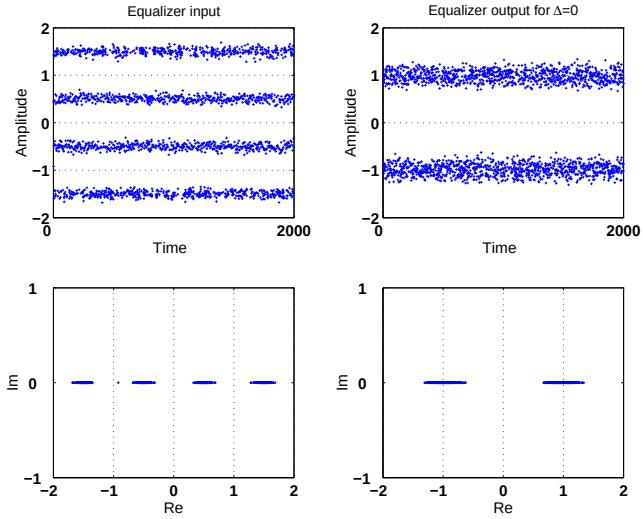

Figure II.2. The plots show the time (*top row*) and scatter (*bottom row*) diagrams of the input and output of the equalizer, i.e., $\{\boldsymbol{y}(i), \hat{\boldsymbol{s}}(i)\}$, for $\Delta = 0$ and BPSK transmissions. Observe how the equalizer output is concentrated around ± 1.

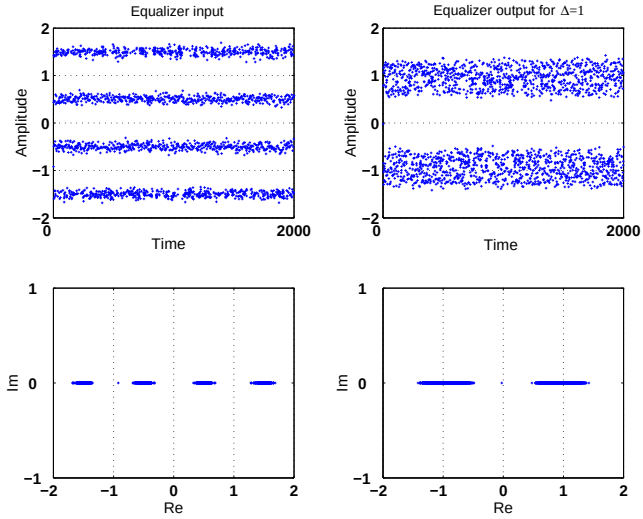

Figure II.3. The plots show the time (*top row*) and scatter (*bottom row*) diagrams of the input and output of the equalizer, i.e., $\{\boldsymbol{y}(i), \hat{\boldsymbol{s}}(i)\}$, for $\Delta = 1$ and BPSK transmissions. Observe how the equalizer output is again concentrated around ± 1.

2. partB.m This program solves parts (b) and (d), which deal with QPSK data. The program generates 3 plots that show the scatter diagrams of the transmitted sequence $\{\boldsymbol{s}(i)\}$, the received sequence $\{\boldsymbol{y}(i)\}$, and the equalizer output $\{\hat{\boldsymbol{s}}(i - \Delta)\}$ for several values of Δ. A final plot shows the values of the estimated m.m.s.e. and the number of erroneous decisions as a function of Δ. The program allows the designer to change the value of the noise variance σ_v^2. Typical outputs of this program are shown in the sequel.

   Figure 7 shows the scatter diagrams of the transmitted (*left*) and received (*right*) sequences, $\{\boldsymbol{s}(i), \boldsymbol{y}(i)\}$.

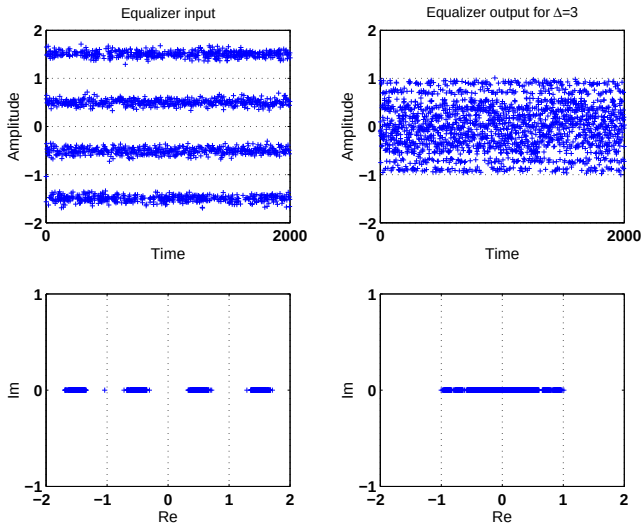

Figure II.4. The plots show the time (*top row*) and scatter (*bottom row*) diagrams of the input and output of the equalizer i.e., $\{\boldsymbol{y}(i), \hat{\boldsymbol{s}}(i)\}$, for $\Delta = 3$ and BPSK transmissions. Observe how the performance of the equalizer is poor in this case.

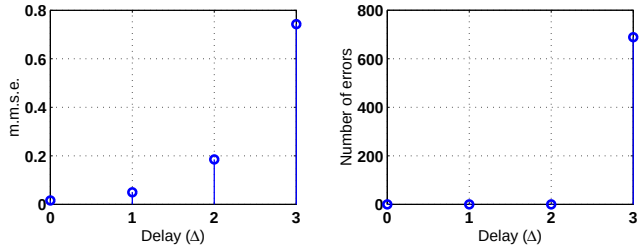

Figure II.5. The plots show the m.m.s.e (*left*) and the number of erroneous decisions (*right*) as a function of Δ for BPSK transmissions.

Observe how the $\boldsymbol{s}(\cdot)$ are concentrated at $\sqrt{2}(\pm 1 \pm j)/2$, as expected, while the $\boldsymbol{y}(i)$ appear to be concentrated around 16 values.

Figure 8 shows the scatter diagrams of the equalizer input (*left*) and output (*right*) sequences for the cases $\Delta = 0$ and $\Delta = 1$. Observe how the equalizer output is now concentrated around the QPSK constellation symbols.

Figure 9 repeats the same scatter diagrams for the cases $\Delta = 2$ and $\Delta = 3$. Observe how the equalizer fails completely for $\Delta = 3$.

The plot showing the m.m.s.e. and the number of erroneous decisions as a function of Δ is similar to Fig. 5. Figure 10, on the other hand, shows the time and scatter diagrams of the equalizer input (*left*) and output (*right*) using $\Delta = 0$ and $\Delta = 1$ for $\sigma_v^2 = 0.05$, i.e., SNR= 14 dB. The equalizer performance is more challenging at this SNR level. Still, the number of erroneous decisions came out as zero in this run.

3. partE.m This program solves parts (e) and (g) for BPSK data and for various noise levels. It generates a plot of the symbol error rate, measured as the ratio of the number of erroneous decisions to the total number of transmissions (adjusted to 20000) as a function of the SNR at the input of the equalizer. The plot shows the SER curves with and without equalization. A typical output is shown in the left plot of Fig. 11.

4. partF.m This program solves parts (f) and (g) for QPSK data and for various noise levels. It also generates a plot of the symbol error rate. The plot shows the SER curves with and without equalization.

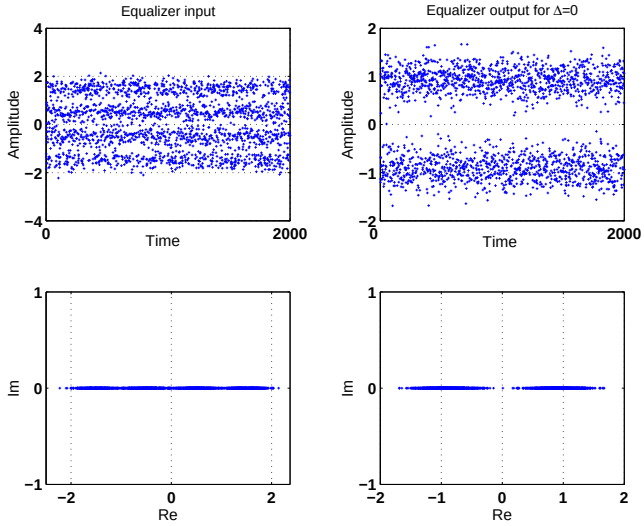

Figure II.6. The plots show the time (*top row*) and scatter (*bottom row*) diagrams of the input and output of the equalizer, i.e., $\{\boldsymbol{y}(i), \hat{\boldsymbol{s}}(i)\}$, for $\Delta = 0$, BPSK transmissions, and $\sigma_v^2 = 0.05$ (i.e., SNR= 14 dB).

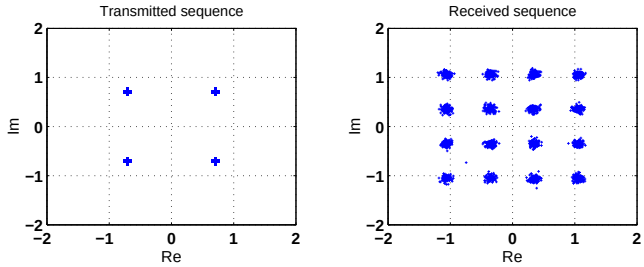

Figure II.7. The plots show the scatter diagrams of the transmitted (*left*) and received (*right*) sequences $\{\boldsymbol{s}(i), \boldsymbol{y}(i)\}$ for QPSK transmissions.

A typical output is shown in the right plot of Fig. 11.

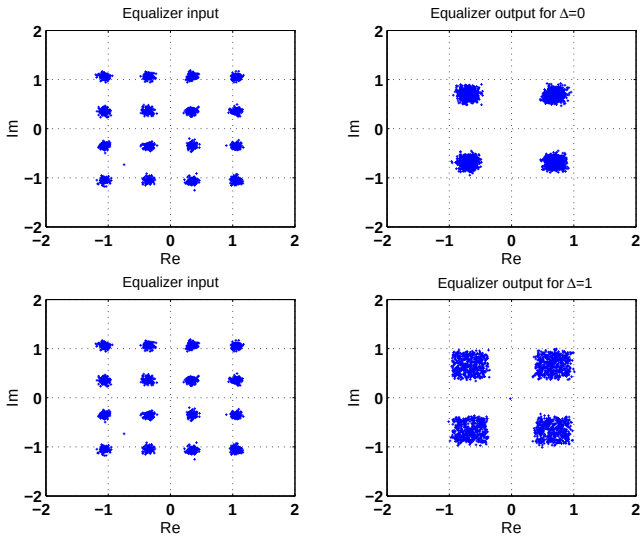

Figure II.8. The plots show the scatter diagrams of the equalizer input (*left*) and output (*right*) for the cases $\Delta = 0$ and $\Delta = 1$ for QPSK transmissions.

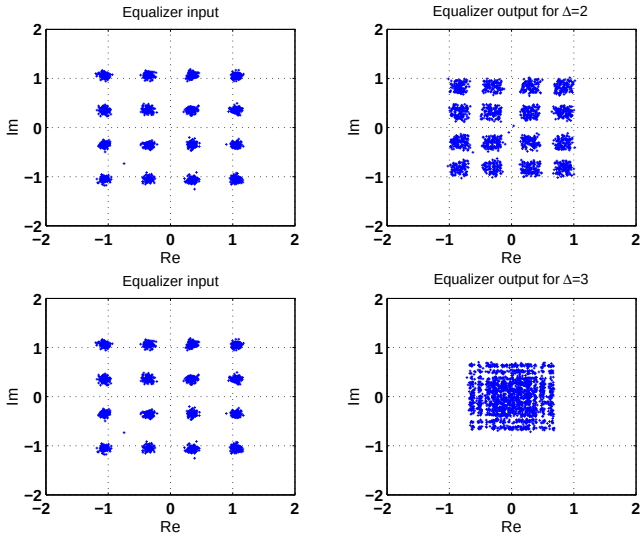

Figure II.9. The plots show the scatter diagrams of the equalizer input (*left*) and output (*right*) for the cases $\Delta = 2$ and $\Delta = 3$ for QPSK transmissions.

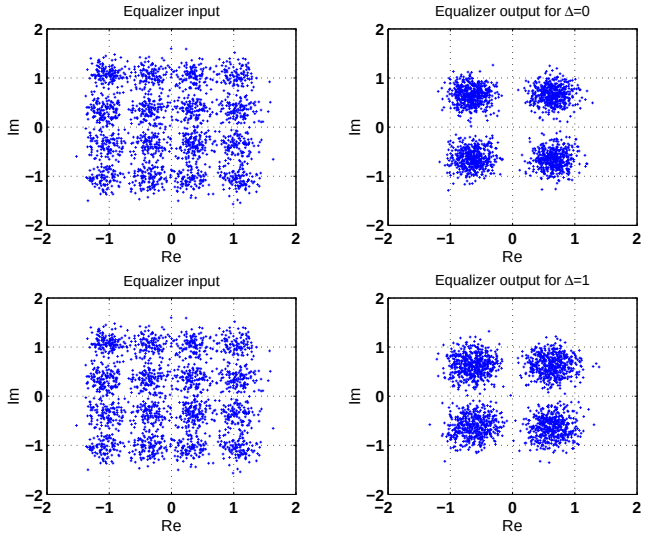

Figure II.10. The plots show the scatter diagrams of the equalizer input (*left*) and output (*right*) for QPSK data with $\Delta = 0$ (*top row*) and $\Delta = 1$ (*bottom row*) and at SNR= 14 dB.

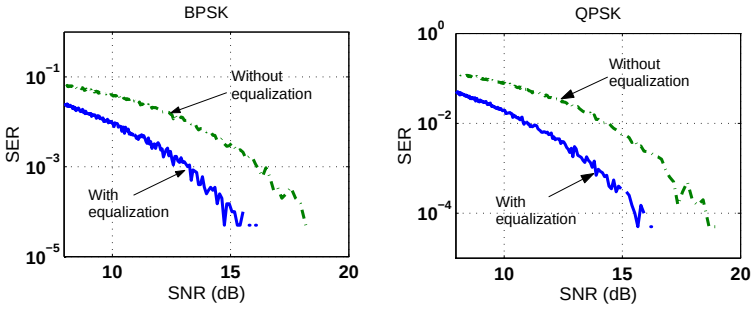

Figure II.11. Plots of the SER over 20000 transmitted BPSK symbols (*left*) and QPSK symbols (*right*), with and without equalization, for SNR values between 8 and 20 dB and using $\Delta = 1$.

Project II.2 (Beamforming) The programs that solve this project are the following.

1. partA.m This program solves part (a). It generates two plots. One plot shows a portion of the baseband sinusoidal wave $s(t)$ along with the corresponding signals received at the four antennas. A second plot shows $s(t)$ and the output of the beamformer — see Figs. 12 and 13. The estimated value of the m.m.s.e. is 0.0494, whereas the theoretical m.m.s.e. is 0.05.

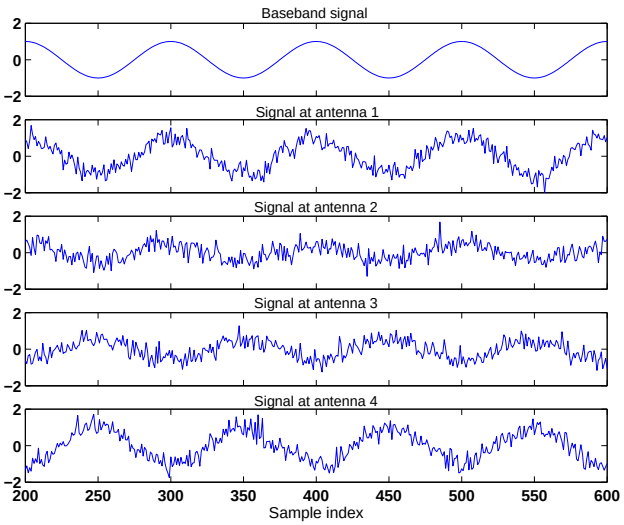

Figure II.12. The plots show the baseband signal $s(t)$ (*top*) along with the signals received at the four antennas. In this simulation, the noise at each antenna is assumed white, and the noises across the antennas are assumed uncorrelated.

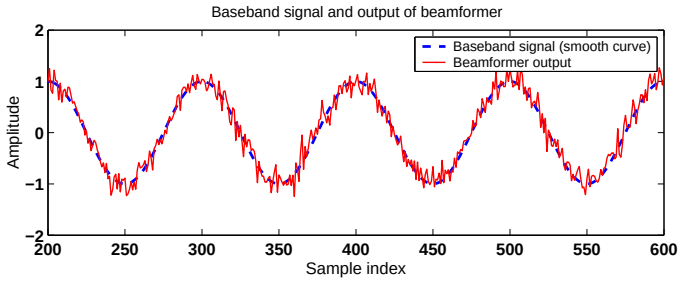

Figure II.13. The plot shows the baseband signal (*solid line*) and the output of the beamformer (*dotted line*).

2. partB.m This program solves part (b). It generates two plots. One plot shows the baseband sinusoidal waves at 30^o and 45^o, along with the signals received at the four antennas. A second plot shows the baseband waves again, along with the combined signal $s(t)$, and the beamformer output superimposed on the signal arriving at 30^o — see Figs. 14 and 15. The estimated value of the m.m.s.e. is 0.067, while the theoretical m.m.s.e. is 0.05.

3. partC.m This program solves part (c). It generates four plots. Two of the plots deal with the case of a single baseband signal at 30^o; they plot the baseband signal and the signals at the antennas, as well as the baseband signal and the beamformer output. Two other plots deal with the case of two incident signals at 30^o and 45^o; they plot the incident baseband signals and the signals at the antennas, as well as the combined signal and the beamformer output —

see Figs. 16–19. The estimated value of the m.m.s.e. is 0.0907 for a single incident wave and 0.1013 for the case of two incident waves, while the theoretical m.m.s.e. is 0.0901.

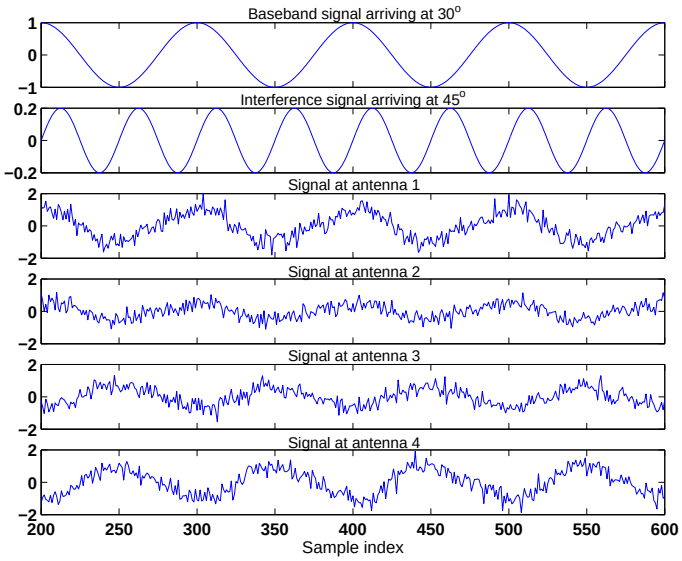

Figure II.14. The plots show the baseband signal at 30^o (*first from top*) and the interfering signal at 45^o (*second from top*), along with the signals received at the four antennas.

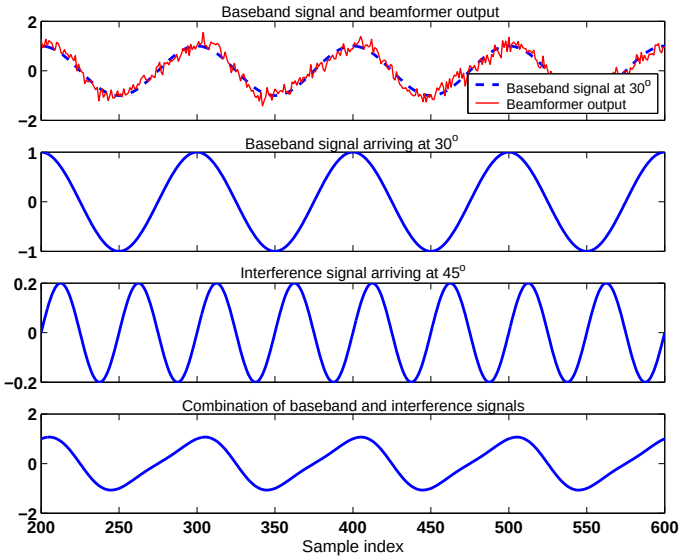

Figure II.15. The figure shows the baseband signal at 30^o and the interfering signal at 45^o (*middle plots*), along with the combined signal (*bottom plot*), and the beamformer output (*dotted line*) superimposed on the signal arriving at 30^o (*solid line in top plot*).

4. partD.m This program solves part (d) and generates two polar plots that show the power gain of the beamformer as a function of θ — see Fig. 20.

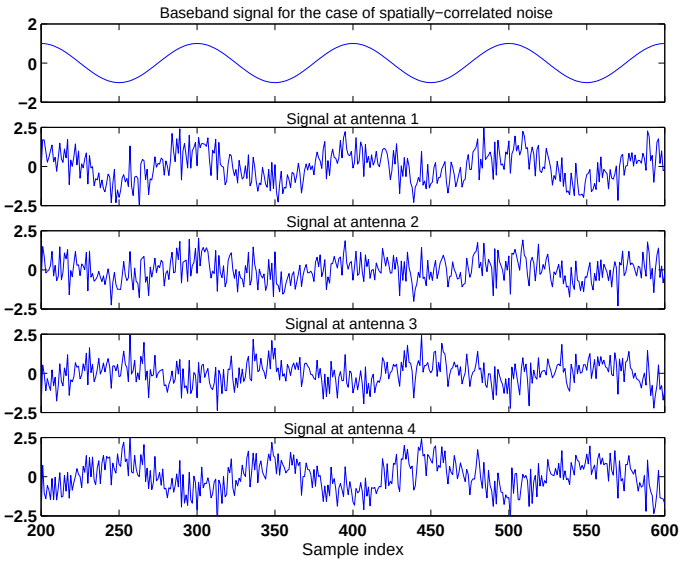

Figure II.16. The plots show the baseband signal (*top*) along with the signals received at the four antennas. In this simulation, the noise at each antenna is assumed white, and the noises across the antennas are spatially correlated.

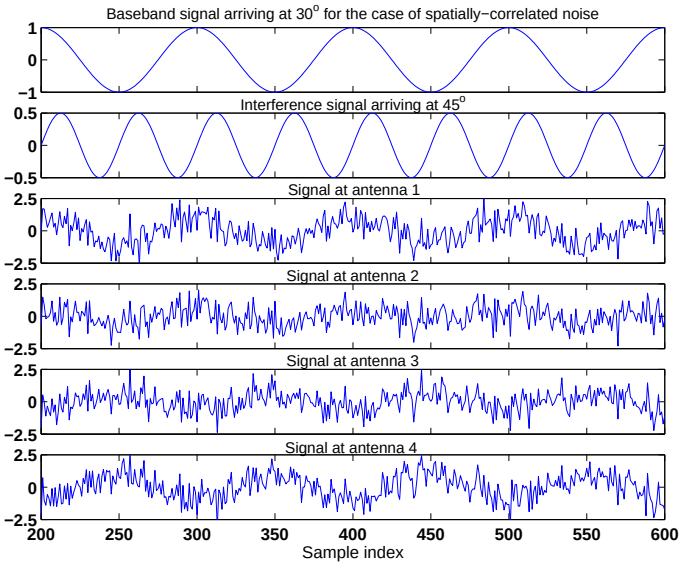

Figure II.17. The plots show the baseband signal at 30^o (*first from top*), the interfering signal at 45^o (*second from top*), along with the signals received at the four antennas for the case of spatially-correlated noise.

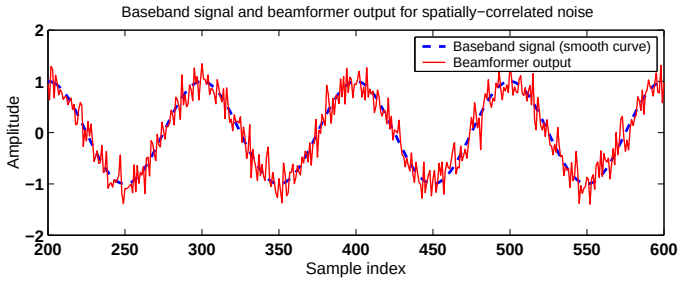

Figure II.18. The plot shows the baseband signal (*solid line*) and the output of the beamformer (*dotted line*) in the case of spatially-correlated noise.

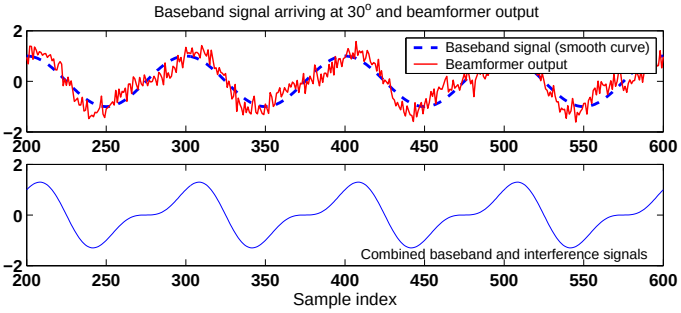

Figure II.19. The figure shows the combined incident signal (*bottom*), along with the beamformer output (*dotted line*) superimposed on the signal arriving at 30^o (*solid line in top plot*).

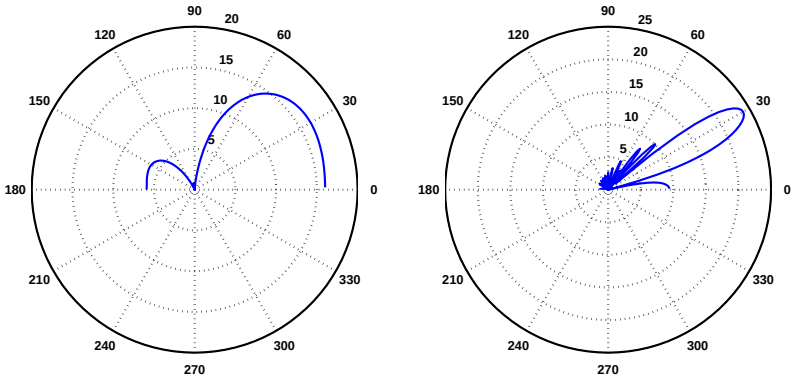

Figure II.20. A polar plot of the power gain of the beamformer in dB as a function of the direction of arrival θ for the case of 4 antennas (*left*) and 25 antennas (*right*).

Project II.3 (Decision feedback equalization) The programs that solve this project are the following.

1. partA.m This program solves part (a) and generates a plot showing the impulse response sequence and the magnitude of the frequency response of the channel, $|C(e^{j\omega})|$ over $[0, \pi]$. Its output is shown in Fig. 21.

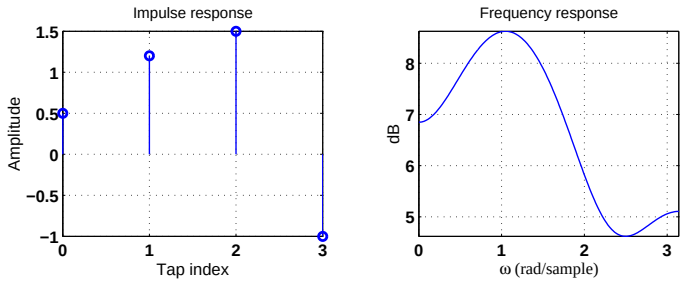

Figure II.21. Impulse and frequency response of the channel $C(z) = 0.5 + 1.2z^{-1} + 1.5z^{-2} - z^{-3}$.

2. partB.m This program solves part (b) and generates a plot showing a scatter diagram of the transmitted and received sequences $\{\boldsymbol{s}(i), \boldsymbol{y}(i)\}$. A typical output is shown in Fig. 22.

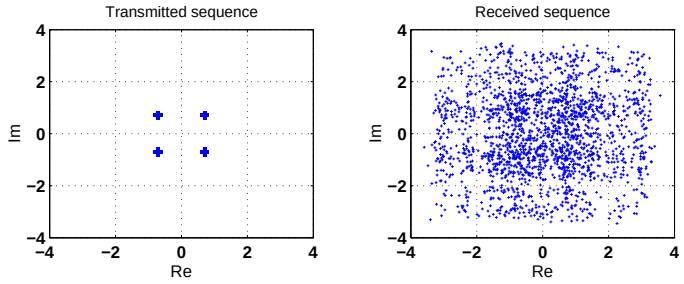

Figure II.22. Scatter diagrams of the transmitted (*left*) and received sequences (*right*), $\{\boldsymbol{s}(i), \boldsymbol{y}(i)\}$, for QPSK transmissions.

3. partC.m This program solves parts (c) and (d) and generates three plots. One plot shows scatter diagrams for the received sequence $\{\boldsymbol{y}(i)\}$ and the sequence at the input of the decision device (for the case $\Delta = 5$) — see Fig. 23. A second plot shows the number of erroneous decisions as a function of Δ, as well as the m.m.s.e. (both theoretical and measured) as a function of Δ — see Fig. 24. A third plot shows the impulse response sequences of the channel, the combination channel-feedforward filter, and the feedback filter delayed by Δ — see Fig. 25. The impulse response of the feedback filter has also been extended by some zeros in the plot in order to match its length with the convolution of the channel and the feedforward filter for ease of comparison. Observe how the taps of the feedback filter cancel the post ISI. The number of errors for this simulation was zero.
4. partE.m This program generates a plot showing the number of erroneous decisions as a function of the length of the feedforward filter, L — see the plot on the left in Fig. 26.
5. partF.m This program generates a plot showing the number of erroneous decisions as a function of the length of the feedback filter, Q — see the plot on the right in Fig. 26.
6. partG.m This program solves part (g) and generates a plot showing the symbol error rate as a function of the SNR level at the input of the equalizer. Each point in the plot is measured using 2×10^4 transmissions. A typical output is shown in Fig. 27.

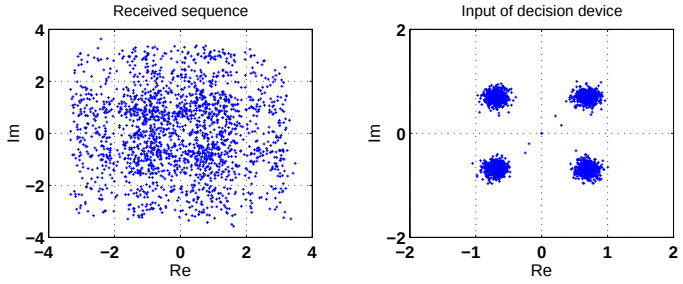

Figure II.23. Scatter diagrams of the received sequence (input of the equalizer) and the sequence at the input of the decision device for $\Delta = 5$ and QPSK transmissions.

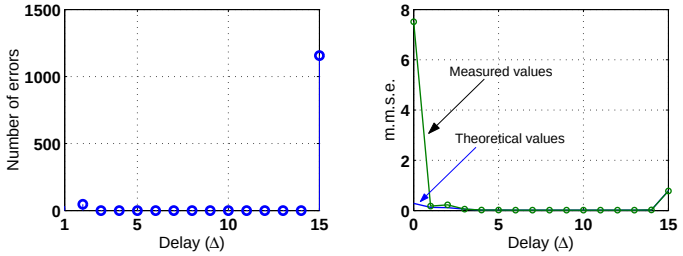

Figure II.24. The plot on the left shows the number of erroneous decisions as a function of Δ, while the plot on the right shows the m.m.s.e. as a function of Δ as deduced from both theory and measurements.

7. partH.m This program solves part (h) and generates a plot that shows the scatter diagrams of the received sequence and of the sequence at the input to the decision device (for $L = 4$ and $\Delta = 4$). The plot also shows the number of erroneous decisions as a function of L, as well as the theoretical and estimated values of the m.m.s.e. for various L. A typical output is shown in Fig. 28. The number of errors in this simulation was zero.
8. partI.m This program solves part (i) and generates a plot that shows the impulse and frequency responses of both the actual channel and its estimate, as well as scatter diagrams of the received sequence $\{\boldsymbol{y}(i)\}$ and the sequence at the input of the decision device — see Fig. 29.
9. partJ.m This program solves part (j) and generates a plot that shows the impulse and frequency responses of the actual channel and its estimate, as well as scatter diagrams of the received sequence $\{\boldsymbol{y}(i)\}$ and the sequence at the input of the decision device — see Fig. 30.

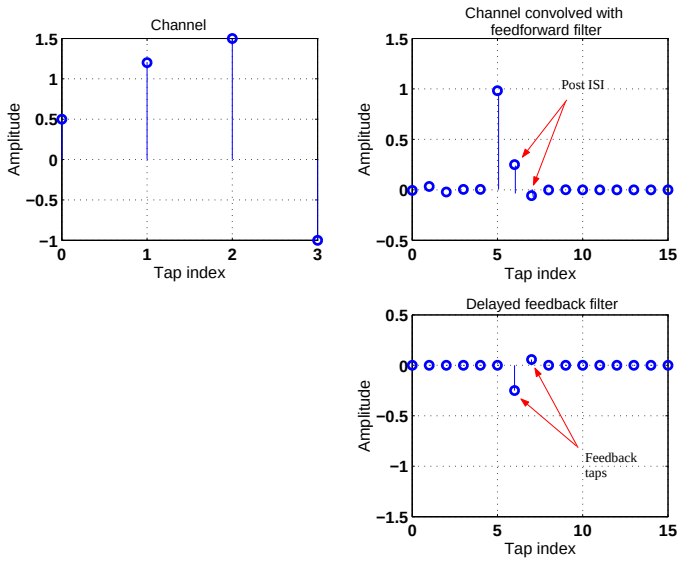

Figure II.25. The top left plot shows the impulse response sequence of the channel. The two plots on the right show the convolution of the channel and feedforward filter impulse responses (*top*) and the feedback filter response delayed by Δ (*bottom*).

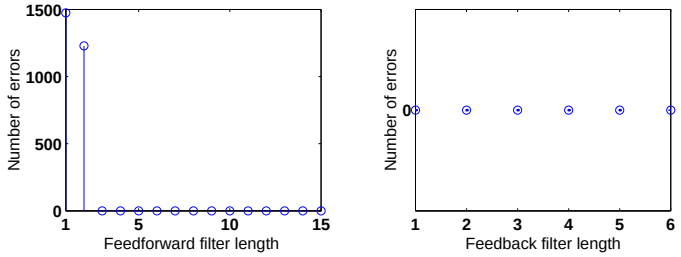

Figure II.26. Number of erroneous decisions as a function of the feedforward filter length using $\Delta = 5$ and $Q = 2$ (*left*), and as a function of the feedback filter length using $\Delta = 5$ and $L = 6$ (*right*).

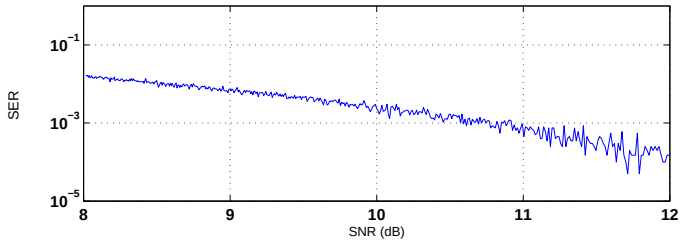

Figure II.27. The plot shows the symbol error rate as a function of the SNR level at the input of the equalizer. The simulation assumes $\Delta = 5$, $L = 6$ and $Q = 1$.

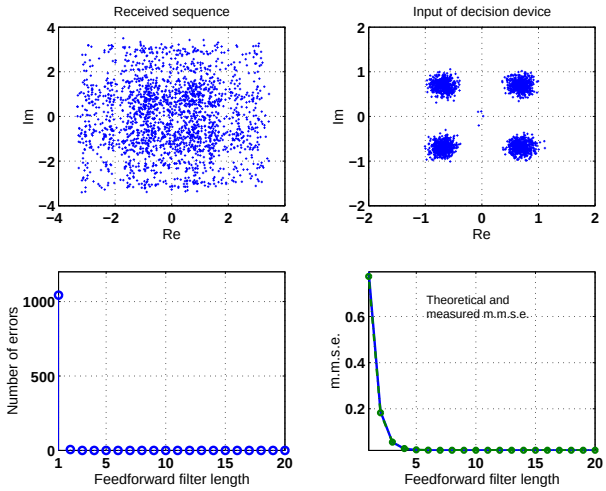

Figure II.28. The plots in the first row show the scatter diagrams of the received sequence and of the sequence at the input of the decision device for $L = 4$ and $\Delta = 4$, using a linear equalizer structure. The plots in the second row show the number of erroneous decisions as a function of the feedforward filter length (L), as well as the theoretical and estimated values of m.m.s.e. for various values of L.

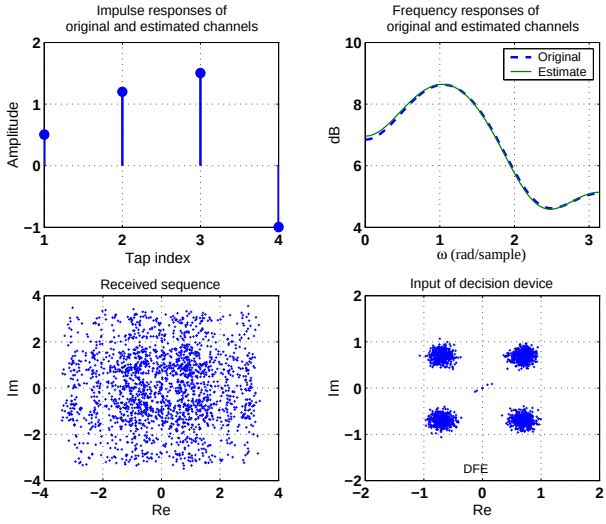

Figure II.29. The plots in the first row show the impulse and frequency responses of the channel and its estimate. The plots in the second row show the scatter diagrams of the received sequence and the sequence at the input of the decision device. This simulation pertains to a DFE implementation with $L = 6$, $Q = 1$, and $\Delta = 5$.

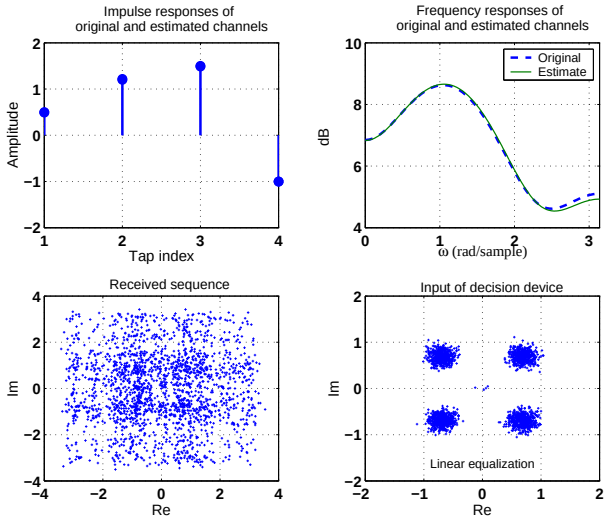

Figure II.30. The plots in the first row show the impulse and frequency responses of the channel and its estimate. The plots in the second row show the scatter diagrams of the received sequence and the sequence at the input of the decision device. This simulation pertains to a linear equalizer with $L = 4$ and $\Delta = 4$.

PART III: STOCHASTIC-GRADIENT METHODS

COMPUTER PROJECTS

Project III.1 (Constant-modulus criterion) The programs that solve this project are the following.

1. partA.m This program prints the locations of the global minima as well as the minimum value of the cost function at these minima. It also plots $J(w)$.
2. partC.m This program generates three plots — see Figs. 1–3. Each plot shows the evolution of $w(i)$ as a function of time, as well as the evolution of $J[w(i)]$.

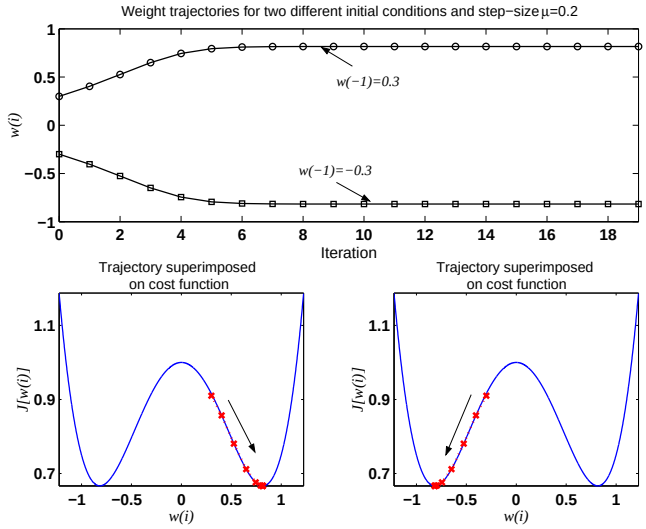

Figure III.1. The top plot shows the evolution of the scalar weight $w(i)$ from two different initial conditions and for $\mu = 0.2$; the filter is seen to converge to the global minimum that is closest to the initial condition. The bottom plot shows the evolution of $J[w(i)]$ superimposed on the cost function $J(w)$ for both initial conditions.

3. partD.m This program generates a plot showing the contour curves of $J(w)$ superimposed on the trajectories of the weight estimates — see Fig. 4.

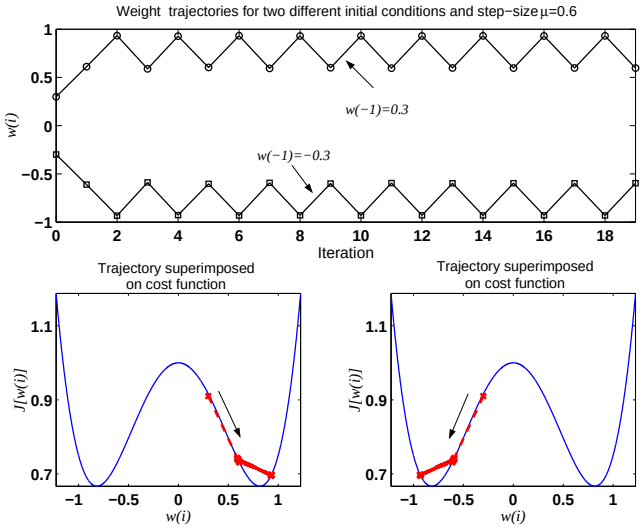

Figure III.2. The top plot shows the evolution of $w(i)$ starting from two different initial conditions and for $\mu = 0.6$. Due to the larger value of the step-size, the filter is seen to oscillate around the global minimum that is closest to the initial condition. The bottom plot shows the evolution of $J[w(i)]$ superimposed on the cost function $J(w)$ for both initial conditions.

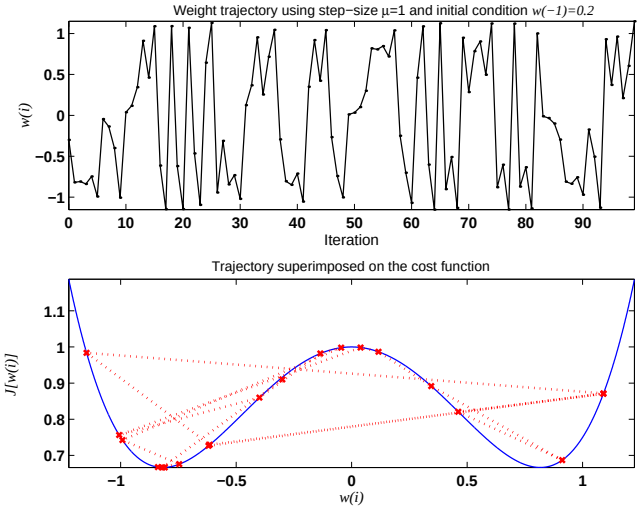

Figure III.3. The top plot shows the evolution of $w(i)$ starting from two different initial conditions and for a large step-size, $\mu = 1$. In this case, the filter does not converge.

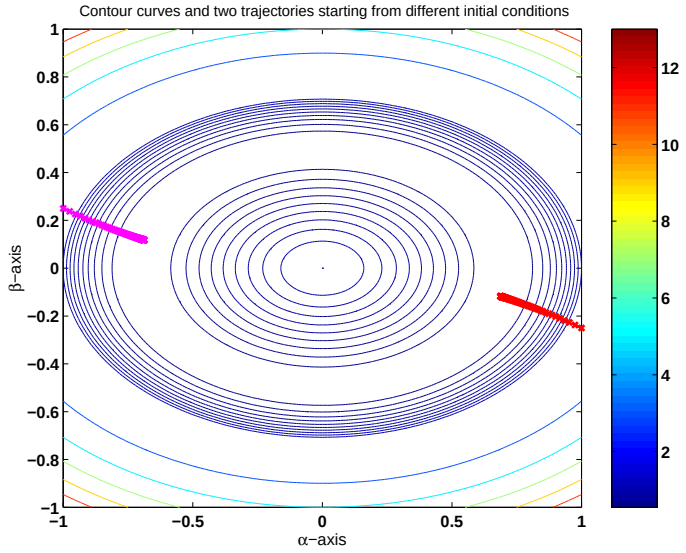

Figure III.4. A plot of the contour curves of $J(w)$, along with the trajectories of the resulting weight estimates starting from two different initial conditions (one is on the left and the other is on the right). Here $w = \text{col}\{\alpha, \beta\}$.

Project III.2 (Constant-modulus algorithm) The programs that solve this project are the following.

1. partA.m This program generates a plot showing the learning curve of the steepest-descent method and an ensemble-average learning curve for CMA2-2. The program allows the user to modify the values of γ, w_{-1}, L, and N. A typical output is shown in Fig. 5.

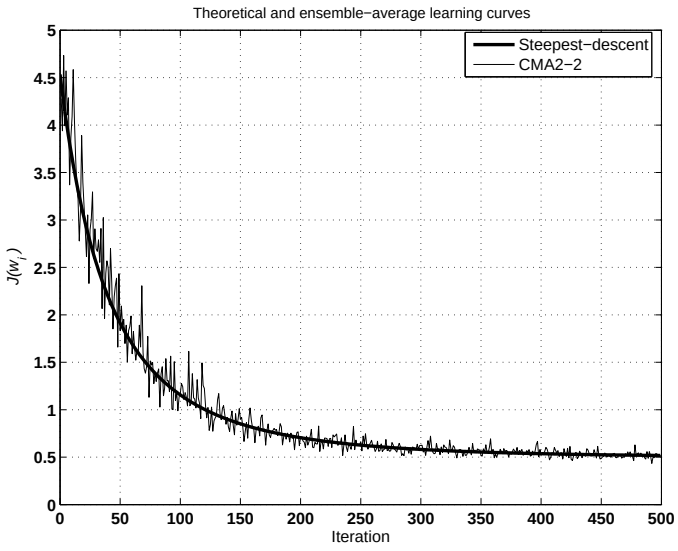

Figure III.5. Learning curve of the steepest-descent method corresponding to the cost function $J(w) = \mathsf{E}\left(\gamma - |\boldsymbol{u}w|^2\right)^2$, along with an ensemble-average curve generated by running CMA2-2 and averaging over 200 experiments.

2. partB.m This program generates two plots. One plot shows the contour curves of the cost function along with the four weight-vector trajectories generated by CMA2-2 (see Fig. 6). The second plot shows the same contour curves with the weight-vector trajectories generated by steepest-descent (see Fig. 7). The program allows the user to modify the values of γ, w_{-1}, and N.

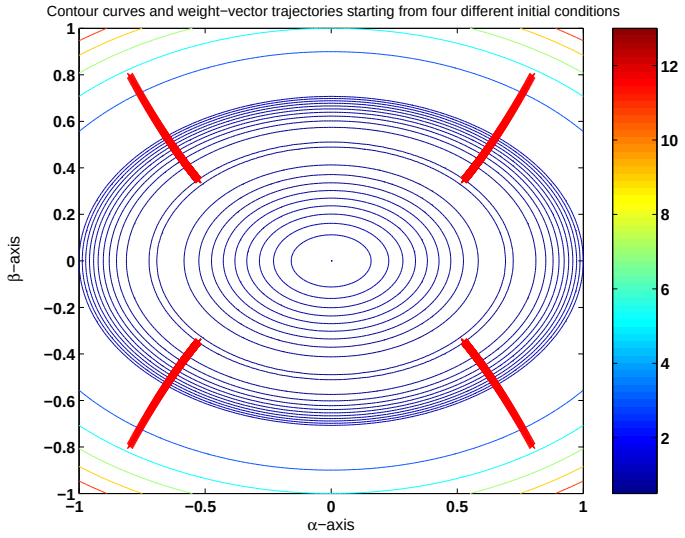

Figure III.6. Contour curves of the cost function $J(w) = \mathsf{E}\left(\gamma - |\boldsymbol{u}w|^2\right)^2$, along with the trajectories generated by the steepest-descent method from four distinct initial conditions.

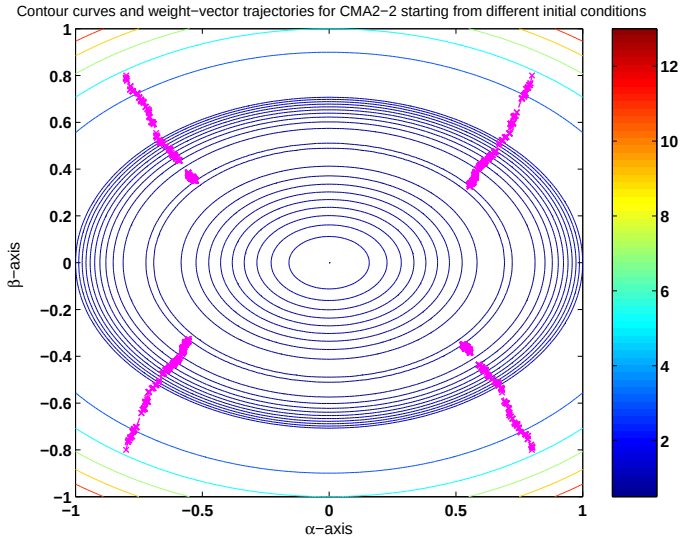

Figure III.7. Contour curves of the cost function $J(w) = \mathsf{E}\left(\gamma - |\boldsymbol{u}w|^2\right)^2$, along with the trajectories generated by CMA2-2 from four distinct initial conditions.

Project III.3 (Adaptive channel equalization) The programs that solve this project are the following.

1. partA.m This program solves part (a) and generates two plots showing the scatter diagrams of the transmitted, received, and estimated sequences $\{s(i), u(i), \hat{s}(i-\Delta)\}$ during training and decision-directed modes of operation. A typical output is shown in Figs. 8–9. No erroneous decisions were observed in this simulation.

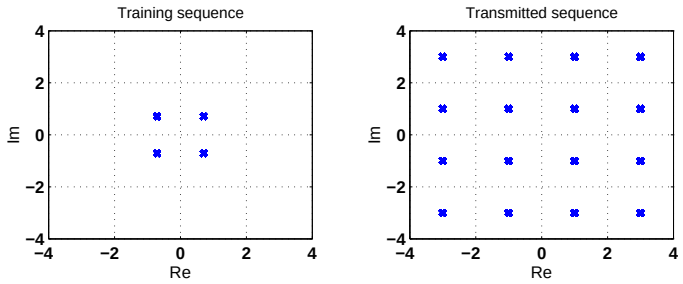

Figure III.8. Scatter diagrams of the QPSK training sequence (*left*) and the 16-QAM transmitted data (*right*).

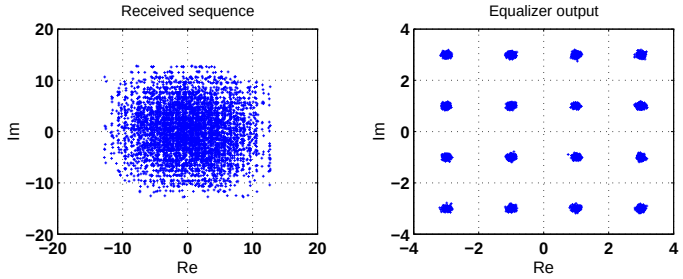

Figure III.9. Scatter diagrams of the received sequence (*left*) and the output of the equalizer (*right*). Observe how the equalizer outputs are centered around the symbols of the 16-QAM constellation.

2. partB.m This program solves part (b) and generates a plot showing the scatter diagrams of the output of the equalizer for three lengths of training period (150, 300, and 500), and for two adaptation schemes (LMS and ϵ-NLMS), as shown in Fig. 10. No erroneous decisions were observed in this simulation in the last two cases of ϵ-NLMS, while 1 error was observed in the last case for LMS. Observe how the performance of LMS is inferior. If the training period is increased further, the performance of LMS can be improved.
3. partC.m This program solves part (c) and generates a plot showing the scatter diagram of the output of the equalizer when the transmitted data is selected from a 256-QAM constellation and the equalizer is trained using ϵ-NLMS with 500 QPSK symbols. A typical output is shown in Fig. 11. No erroneous decisions were observed in this simulation.
4. partD.m This program solves part (d) and generates a plot of the SER curve versus SNR for several QAM constellations. A typical output is shown in Fig. 12. Observe how, for a fixed SNR, the probability of error increases as the order of the constellation increases.
5. partE.m This program solves part (e) and generates two plots of the scatter diagrams of the input and output of the equalizer for two different configurations. A typical output is shown in Fig. 13. No erroneous decisions were observed in both simulations.
6. partF.m This program solves part (f) and generates a plot of the SER curve versus SNR for several QAM constellations. A typical output is shown in Fig. 14.
7. partG.m This program solves part (g). Figure 15 shows the impulse and frequency responses of the channel, and it is seen that the channel has a pronounced spectral null; a fact that makes the equalization task more challenging. Figure 16 shows the frequency and impulse responses of the channel, the RLS

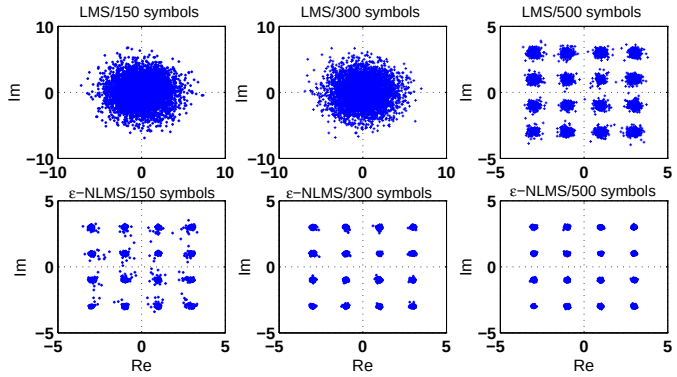

Figure III.10. Scatter diagrams of the signal at the output of the equalizer for three different number of training symbols, 150, 300, and 500, and for both algorithms, LMS (*top row*) and ϵ-NLMS (*bottom row*).

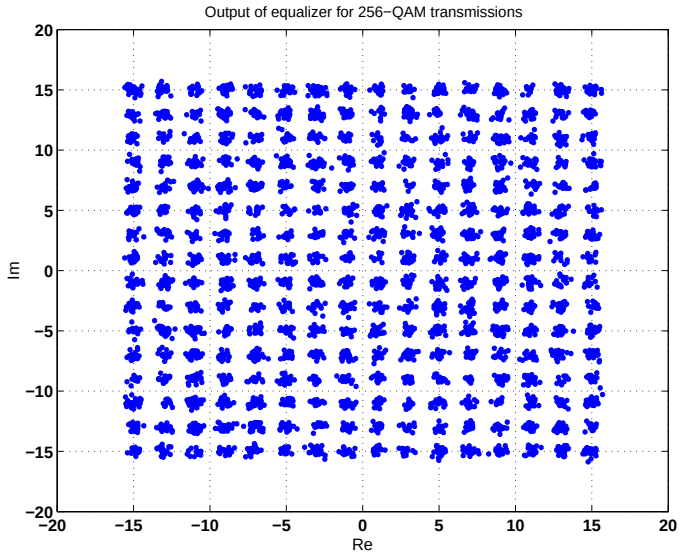

Figure III.11. Scatter diagram of the signal at the output of the equalizer. The equalizer is trained with 500 QPSK symbols using $\epsilon-$NLMS and the transmitted data during the decision-directed mode is from a 256-QAM constellation.

equalizer at the end of the adaptation process, and of the convolution of the channel and equalizer impulse response sequences (we are only plotting the real parts of the impulse response sequences). Observe how the combined frequency response still exhibits a spectral null; the equalizer has not been successful in fully removing the null from the channel spectrum (i.e., it has not been successful in flattening the channel). The program also generates a plot of the scatter diagrams of the transmitted 4-QAM data and the outputs of the linear equalizer for two cases. In one case, the equalizer is trained with RLS for 100 iterations, and in the second case the equalizer is trained with ϵ-NLMS for 2000 iterations. A typical output is shown in Fig. 17. Observe how ϵ-NLMS fails for this channel while no erroneous decisions were observed for RLS in this simulation. The performance of RLS can be improved by increasing the length of the equalizer.

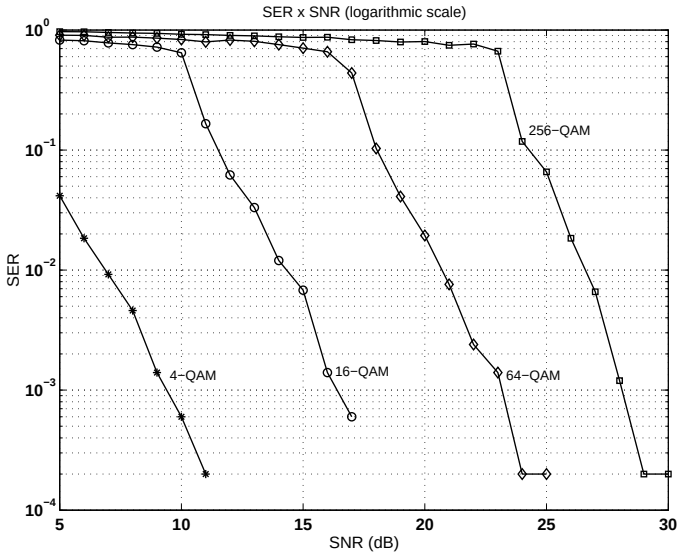

Figure III.12. Plots of the SER as a function of the SNR at the input of the equalizer and the order of the QAM constellation. The adaptive filter is trained with ϵ-NLMS using 500 QPSK training symbols.

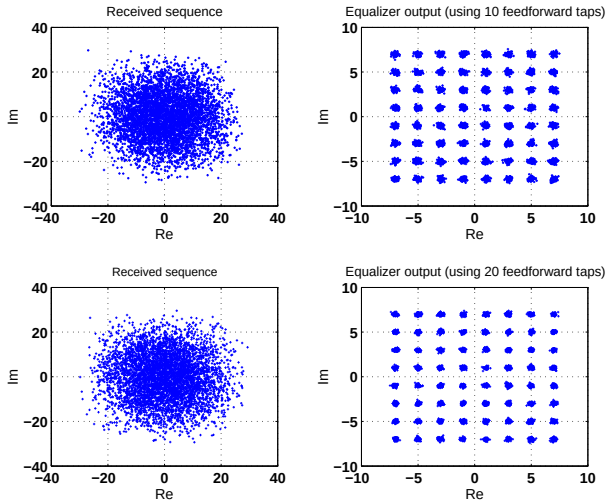

Figure III.13. Scatter diagrams of the input and output of the decision-feedback equalizer for 64-QAM transmissions. The top row corresponds to a DFE with 10 feedforward taps, 2 feedback taps, and delay $\Delta = 7$. The lower plot corresponds to a DFE with 20 feedforward taps, 2 feedback taps, and delay $\Delta = 10$.

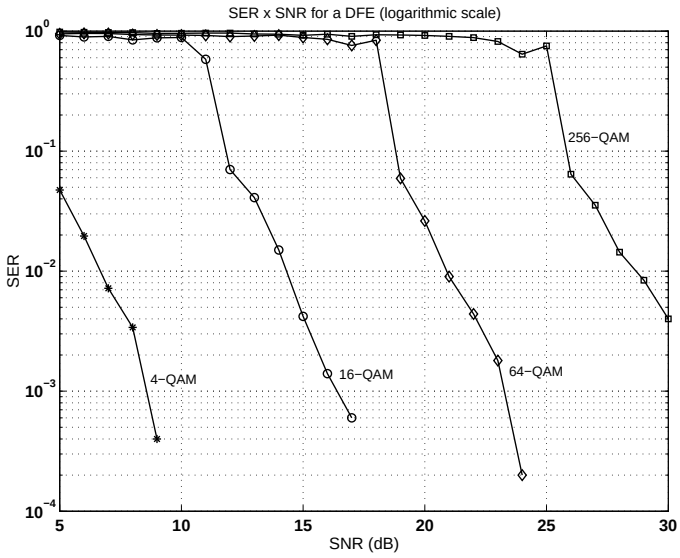

Figure III.14. A plot of the SER as a function of the SNR at the input of the equalizer and the order of the QAM constellation. The DFE is trained with ϵ-NLMS using 500 QPSK training symbols.

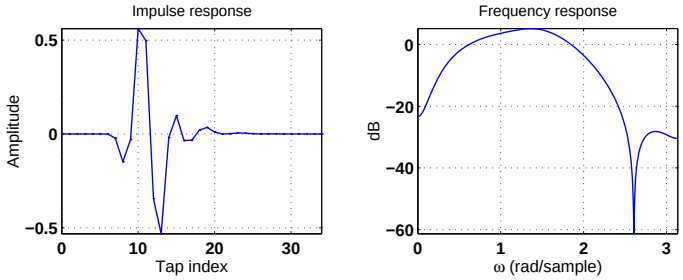

Figure III.15. Impulse (*left*) and frequency (*right*) responses of channel.

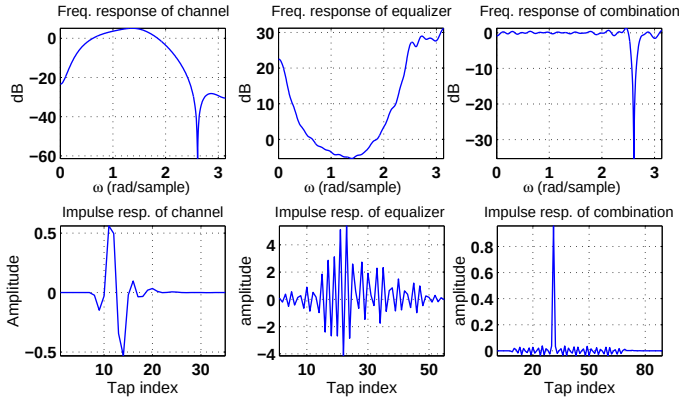

Figure III.16. Frequency and impulse responses of the channel (*left*), the RLS equalizer at the end of the adaptation process (*center*), and of the convolution of the channel and equalizer (*right*).

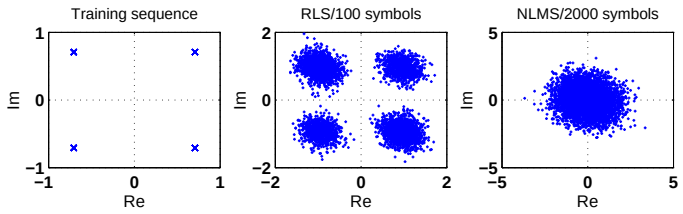

Figure III.17. Scatter diagrams of the transmitted sequence and the outputs of the equalizer for two training algorithms: RLS (*center*) and ϵ-NLMS (*far right*). The former is trained with 100 QPSK symbols while the latter is trained with 2000 QPSK symbols.

Project III.4 (Blind adaptive equalization) The programs that solve this project are the following.

1. partA.m This program solves part (a) and generates two plots. Figure 18 shows the impulse responses of the channel, the equalizer, and the combination channel-equalizer. Observe that the latter has a peak at sample 19 so that the delay introduced by the channel and equalizer is 19. Figure 19 shows a typical plot of the scatter diagrams at transmission, reception, and output of the equalizer.

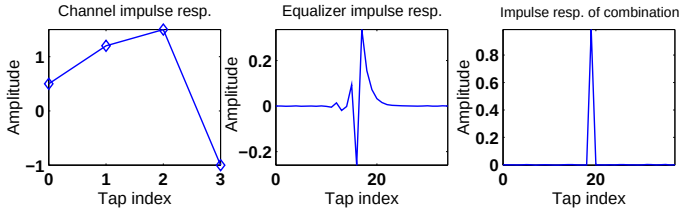

Figure III.18. Impulse responses of the channel, the blind equalizer, and the combination channel-equalizer for QPSK transmissions. Observe that the latter has a peak at sample 19.

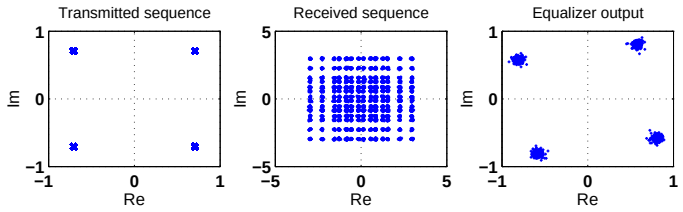

Figure III.19. Scatter diagrams of the transmitted sequence, received sequence, and the output of the equalizer for QPSK transmissions and using CMA2-2. The first 2000 samples are ignored in order to give the equalizer sufficient time for convergence.

2. partB.m This program solves part (b) and generates three plots. Figure 20 shows the impulse responses of the channel, the equalizer, and the combination channel-equalizer. Figure 21 shows a typical plot of the scatter diagrams at transmission, reception, and output of the equalizer. Observe that although symbols from 16−QAM do not have constant modulus, CMA2-2 still functions properly; albeit more slowly. Figure 22 shows the real and imaginary parts of 50 transmitted symbols and the corresponding 50 decisions by the slicer (with the delay of 19 samples accounted for in order to synchronize the signals). These 50 symbols are chosen from the later part of the data after the equalizer has converged.

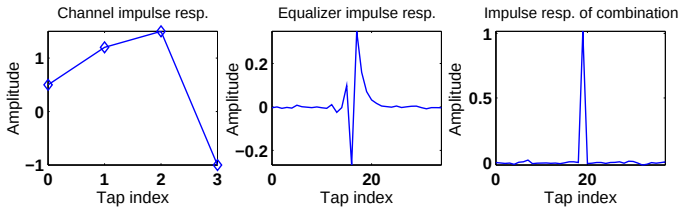

Figure III.20. Impulse responses of the channel, the blind equalizer, and the combination channel-equalizer for 16-QAM transmissions.

3. partC.m This program solves part (c) and generates three plots as in part (b). Here we only show in Fig. 23 the resulting scatter diagrams for MMA. Observe how the scatter diagram at the output of the equalizer is more focused when compared with CMA2-2.
4. partD.m This program solves part (d) and generates scatter diagrams for three additional blind adaptive algorithms. Typical outputs are shown in Fig. 24.

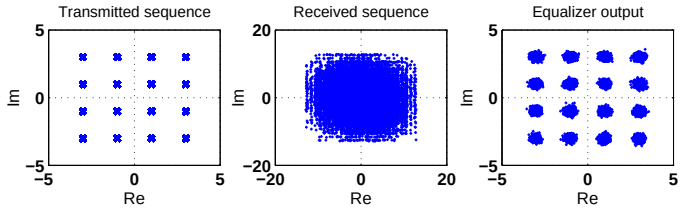

Figure III.21. Scatter diagrams of the transmitted sequence, received sequence, and the output of the equalizer for 16-QAM transmissions and using CMA2-2. The first 15000 samples are ignored in order to give the equalizer sufficient time for convergence.

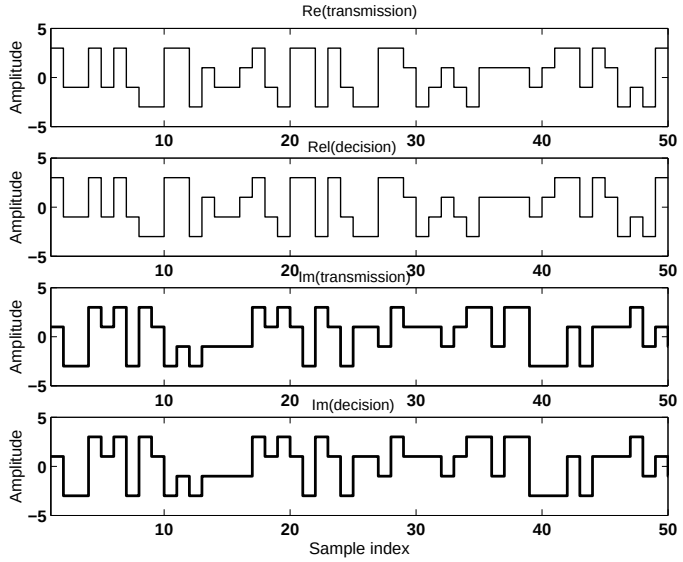

Figure III.22. Time diagrams of 50 transmitted symbols and the corresponding decisions after equalizer convergence. The top two plots correspond to the real parts, while the bottom two plots correspond to the imaginary parts.

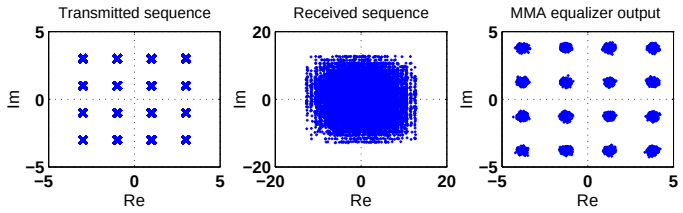

Figure III.23. Scatter diagrams of the transmitted sequence, received sequence, and the output of the equalizer for 16-QAM transmissions and using MMA. The first 15000 samples are ignored in order to give the equalizer sufficient time for convergence.

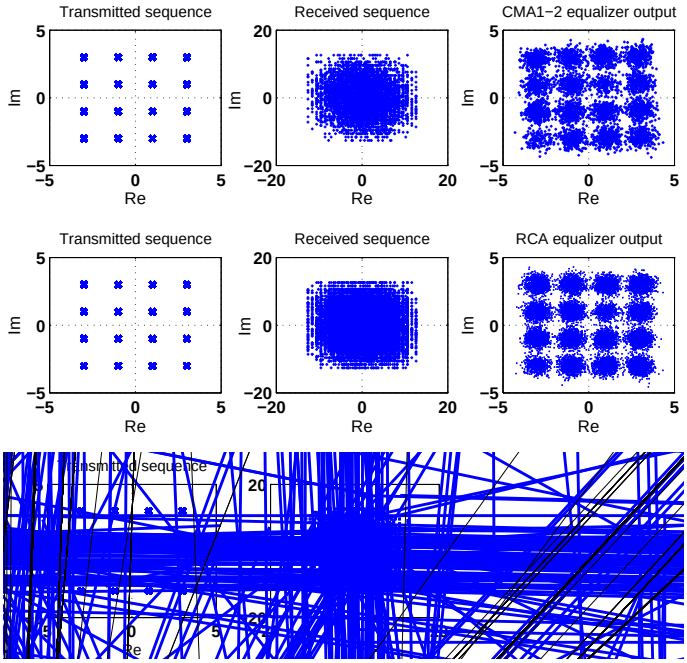

Figure III.24. Scatter diagrams of the transmitted (*left*) and received sequences (*center*), and of the output of the equalizer (*right*) for three blind algorithms: CMA1-2 (*top row*), RCA (*middle row*), and stop-and-go algorithm (*bottom row*).

PART IV: MEAN-SQUARE PERFORMANCE

COMPUTER PROJECTS

Project IV.1 (Line echo cancellation) The programs that solve this project are the following.

1. partA.m This program generates two plots showing the impulse response sequence of the channel as well as its frequency response, as shown in Fig. 1. It is seen that the bandwidth of the echo path model is approximately 3.4 kHz.

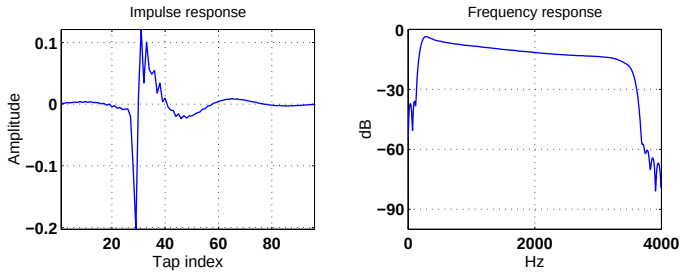

Figure IV.1. Impulse and frequency responses of the echo path model.

2. partB.m This program solves parts B and C — see Fig.2. The powers of the input and output signals are 0 and -6.3 dB, respectively, so that ERL ≈ 6.3 dB and the amplitude of a signal going through the channel is reduced by $1/2$.

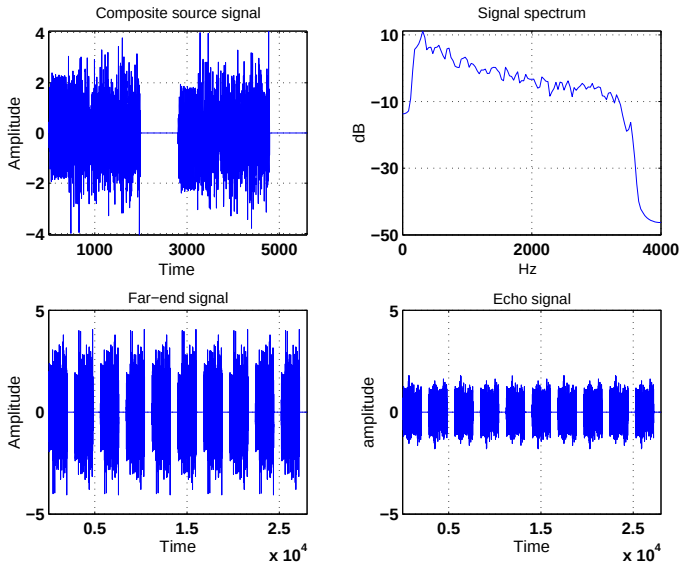

Figure IV.2. The top row shows the CSS data and their spectrum, while the bottom row shows 5 blocks of CSS data and the resulting echo.

3. partD.m This program solves parts D and E and generates two plots. Figure 3 shows the echo path impulse response and its estimate, while Fig. 4 shows the far-end signal, the echo, and the error signal. The resulting ERLE was found to be around 56 dB. Consequently, the total attenuation from the far-end input to the output of the adaptive filter is seen to be ERL + ERLE ≈ 62 dB.
4. partF.m The power of the echo signal is the MSE of the adaptive filter, so that

$$P_{\text{error}}(\text{dB}) \;\approx\; 10\log_{10}\left(\sigma_v^2 \;+\; \frac{\mu\sigma_v^2}{2-\mu}\right) \;=\; -39.42\,\text{dB}$$

Since $P_{\text{echo}} \approx -6.3$ dB, we find that the theoretical ERLE is ≈ 33.1 dB. The simulated value in this experiment is ≈ 29.5 dB. The simulated value will get closer to theory if the simulation is run for a longer period of time.

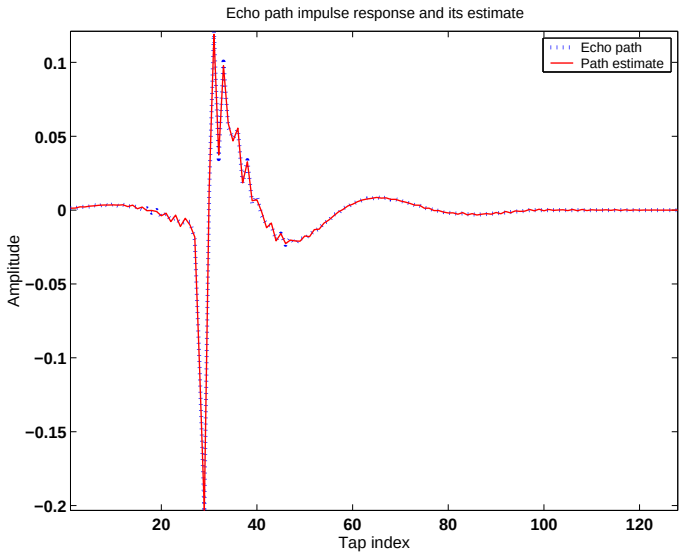

Figure IV.3. The figure shows the impulse response sequences of the echo path and its estimate by the adaptive filter.

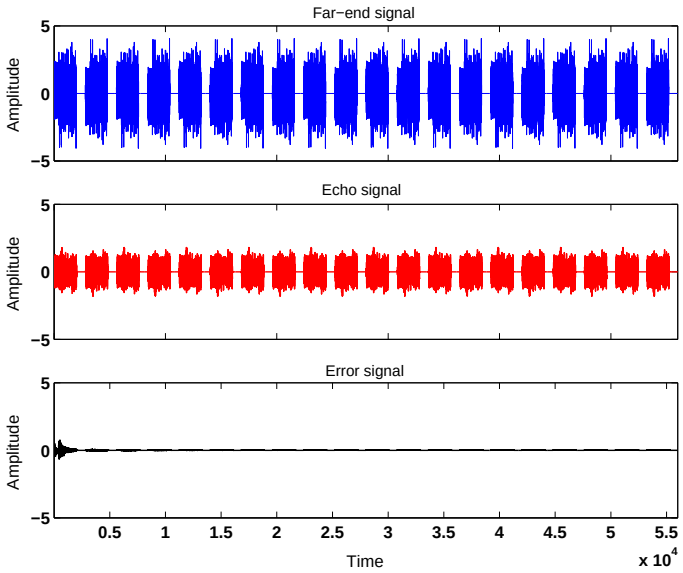

Figure IV.4. The top row shows the far-end signal, while the middle row shows the corresponding echo and the last row shows the resulting error signal.

Project IV.2 (Tracking a Rayleigh fading channel) The programs that solve this project are the following.

1. partB.m This program generates three plots. One plot shows the amplitude and phase components of a Rayleigh fading sequence, while a second plot shows the cumulative distribution function of the amplitude sequence. Typical outputs are shown in Figs. 5 (for the first 500 samples) and 6. The latter figure can be used, for example, to infer that the amplitude of $|\boldsymbol{x}(n)|$ is attenuated by more than 10 dB for 10% of the time, and so forth.

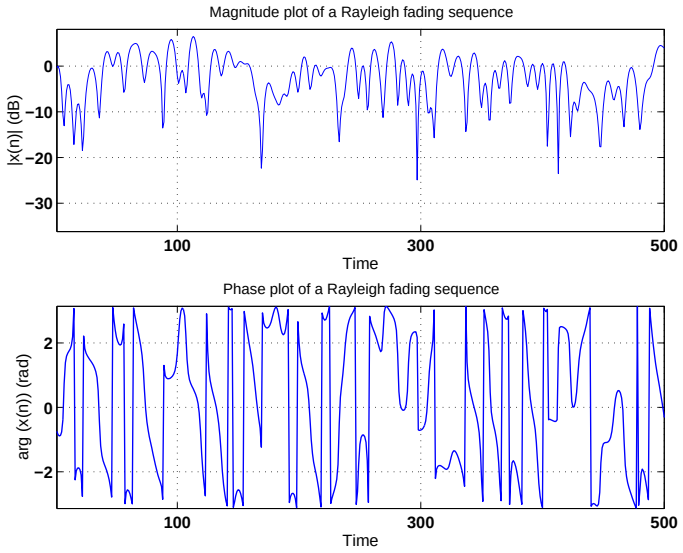

Figure IV.5. Amplitude and phase plots of a Rayleigh fading sequence with Doppler frequency $f_D = 66.67$Hz and sampling frequency $f_s = 1$ kHz.

Observe that the sequence $\boldsymbol{x}(n)$ in Fig. 5 undergoes rapid variations. This is because in this case, the Doppler frequency and the sampling frequency are essentially of the same order of magnitude ($f_D = 66.67$Hz and $f_s = 1000$Hz). In loose terms, the inverse of f_D gives an indication of the time interval needed for the sequence $\boldsymbol{x}(n)$ to undergo a significant change in its value. In our example, this time amounts to ≈ 15ms, which translates into 15 samples at the assumed sampling rate. For the sake of comparison, Fig. 7 plots the amplitude component of another Rayleigh fading sequence with the same Doppler frequency $f_D = 66.67$Hz but with sampling period $T_s = 1\mu$s. The ratio of f_s to f_D is now ≈ 15000 samples so that the variations in $\boldsymbol{x}(n)$ occur now at a significantly slower pace.

2. partC.m This program solves parts C, D, and E. It generates a plot of the learning curve of LMS by running it for 30000 iterations and averaging over 100 experiments. The channel rays are assumed to fade at 10Hz and the sampling period is set to $T_s = 0.8\mu$s. Figure 8 shows a typical trajectory of the amplitude of the first ray and its estimate by the LMS filter. The leftmost plot shows that the trajectories are in very good agreement. The rightmost plot zooms onto the interval $[1000, 1100]$.

   Figure 9 shows only the first 1000 samples of a typical learning curve. In addition, the simulated and theoretical MSE values in this case are 0.001056 and 0.001026, respectively. The theoretical values obtained from the expressions shown in parts (d) and (e) are almost identical. This is because at $f_D = 10$Hz, the value of α is unity for all practical purposes. Also, the difference that is observed between simulation and theory for the MSE is in part due to the approximate first-order auto-regressive model that is being used to model the Rayleigh fading behavior.

3. partF.m This program solves part F and generates a plot of the MSE as a function of the Doppler frequency over the range 10Hz to 25Hz. A typical output is shown in Fig. 10. It is seen that as the Doppler frequency increases, the tracking performance of LMS deteriorates. This behavior is expected since higher Doppler frequencies correspond to faster variations in the channel; remember that a higher Doppler frequency translates into a larger covariance matrix Q and, therefore, into more appreciable channel nonstationarities.

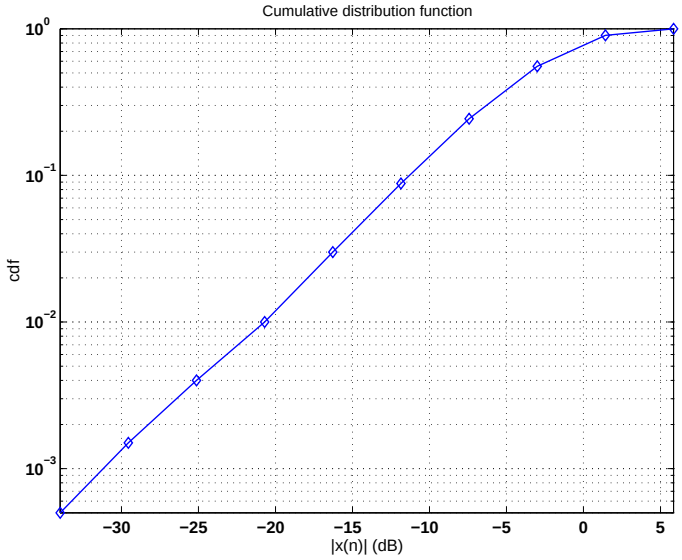

Figure IV.6. The cumulative distribution function of a Rayleigh fading sequence corresponding to a vehicle moving at 80Km/h.

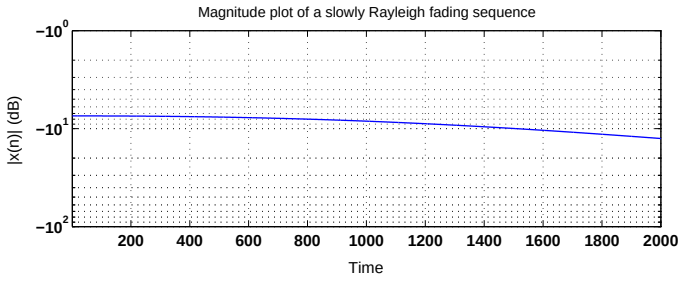

Figure IV.7. Amplitude plot of a Rayleigh fading sequence with Doppler frequency $f_D = 66.67$Hz and sampling frequency $f_s = 1$MHz.

4. partG.m This program solves part G. It generates a plot of the filter MSE as a function of the step-size. The Doppler frequency is fixed at 10Hz for both rays, and the MSE values are obtained by averaging over 50 experiments — see Fig. 11.

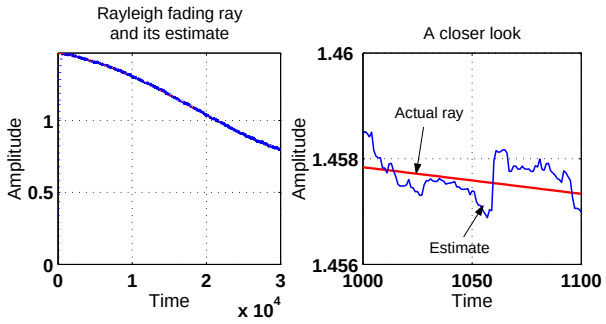

Figure IV.8. The leftmost plot shows a typical trajectory of the amplitude of the first Rayleigh fading ray and its estimate (both trajectories are in almost perfect agreement). The rightmost plots zooms on the interval $1000 \leq i \leq 1100$.

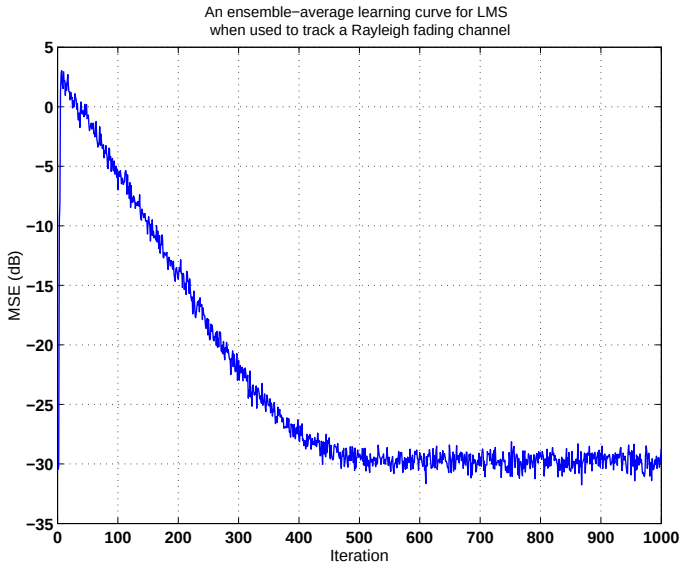

Figure IV.9. The first 1000 samples of a learning curve that results from tracking a two-ray Rayleigh fading channel using LMS.

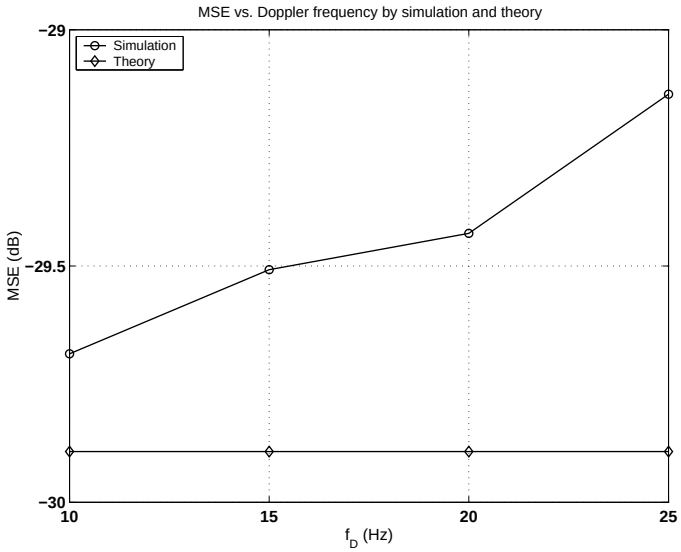

Figure IV.10. A plot of the tracking MSE of the LMS filter as a function of the Doppler frequency of the Rayleigh fading rays.

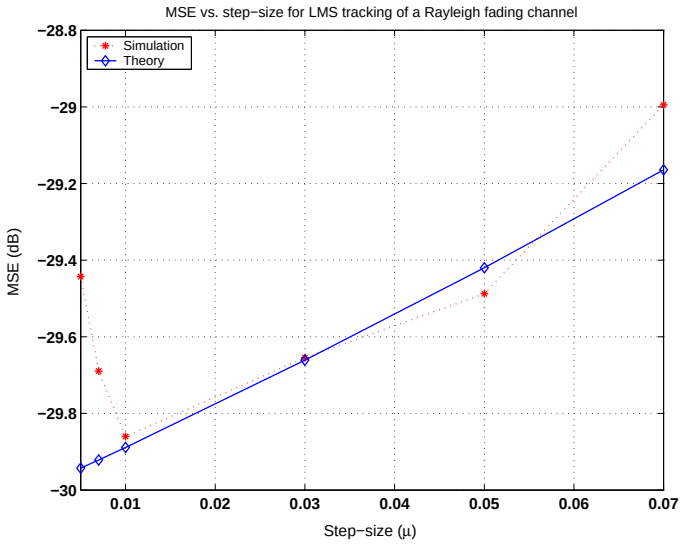

Figure IV.11. A plot of the tracking MSE as a function of the step-size for Rayleigh fading rays with Doppler frequency at 10Hz.

PART V: TRANSIENT PERFORMANCE

COMPUTER PROJECT

Project V.1 (Transient behavior of LMS) The programs that solve this project are the following.

1. partA.m This program solves parts A, B, and C. It generates data $\{\boldsymbol{u}_i, \boldsymbol{d}(i)\}$ with the desired specifications. Figure 1 shows a typical ensemble-average learning curve for the weight-error vector, when the step-size is fixed at $\mu = 0.01$. The ensemble-average curve is superimposed onto the theoretical curve. It is seen that there is a good match between theory and practice. Figure 2 plots the MSD as a function of the step-size. Observe how the filter starts to diverge for step-sizes close to 0.1. The rightmost plot in Fig. 2 shows the behavior of the function $f(\mu)$ over the same range of step-sizes. The plot shows that $f(\mu)$ exceeds one around $\mu \approx 0.092$. We thus find that the simulation results are consistent with theory.

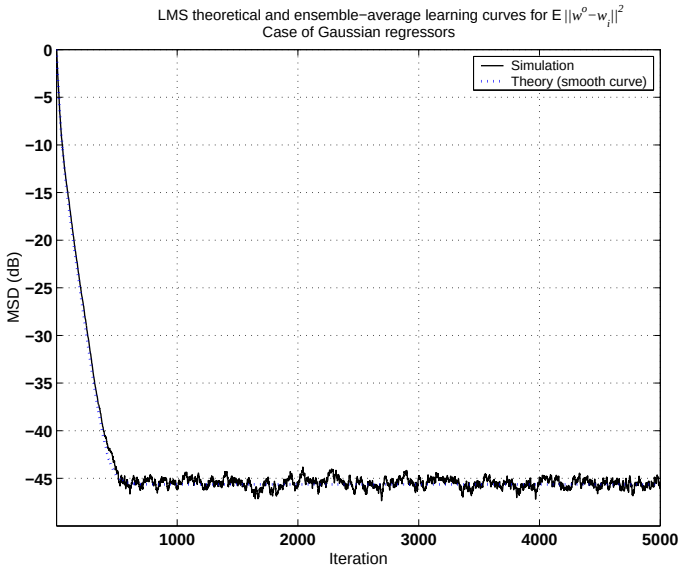

Figure V.1. Theoretical and simulated learning curves for the weight-error vector of LMS operating with a step-size $\mu = 0.01$ and using Gaussian regressors.

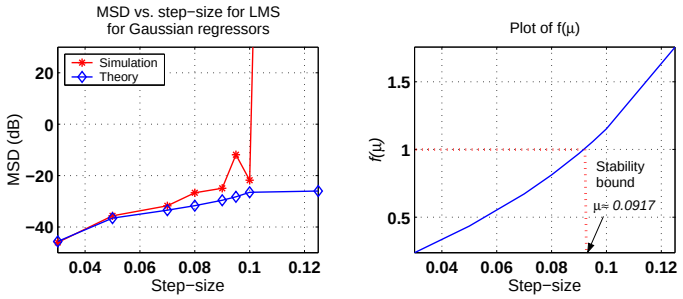

Figure V.2. Theoretical and simulated MSD of LMS as a function of the step-size for Gaussian regressors (*leftmost plot*). The rightmost plot shows the behavior of the function $f(\mu)$ over the same range of step-sizes.

2. partD.m This program generates two plots. Figure 3 shows a typical ensemble-average learning curve for the weight-error vector, when the step-size is fixed at $\mu = 0.05$. The ensemble-average curve is superimposed onto the theoretical curve that follows from Prob. V.7. It is seen that there is also a good match between theory and practice for non-Gaussian regressors.

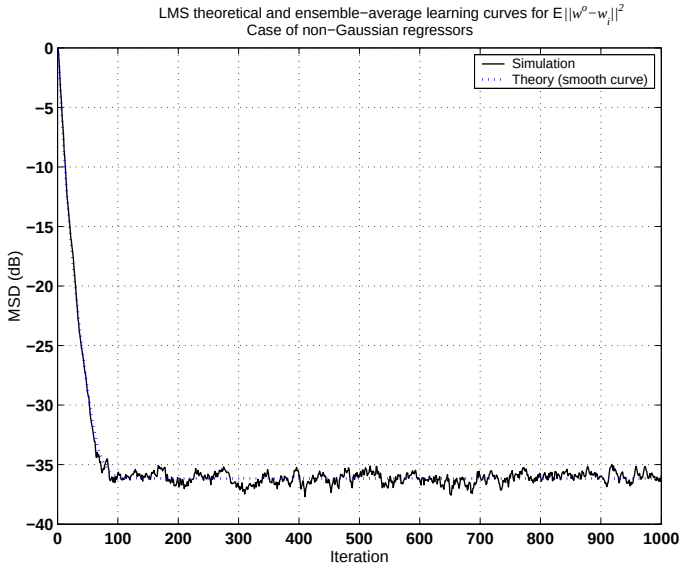

Figure V.3. Theoretical and simulated learning curves for the weight-error vector of LMS operating with a step-size $\mu = 0.05$ and using non-Gaussian regressors.

Figure 4 plots the MSD as a function of the step-size. The theoretical values are obtained by using the following expression

$$\text{MSD} \;=\; \mu^2\sigma_v^2 r'^{\mathsf{T}}(\mathrm{I}-F)^{-1}q, \qquad \text{EMSE} \;=\; \mu^2\sigma_v^2 r'^{\mathsf{T}}(\mathrm{I}-F)^{-1}r$$

where $\{r, r', q, F\}$ are defined in the statement of the theorem. Observe how the filter starts to diverge for step-sizes close to 0.1. Actually, using the estimated values for A and B that are generated by the program, it is found that

$$\frac{1}{\lambda_{\max}(A^{-1}B)} \;\approx\; 0.103, \qquad \frac{1}{\max\left\{\lambda(H)\in\mathbb{R}^+\right\}} \;\approx\; 0.204$$

so that we again find that the simulation results are consistent with theory.

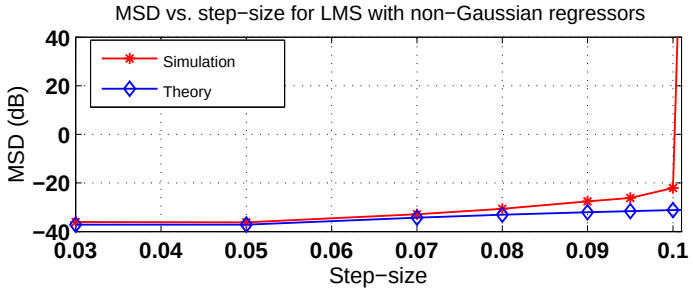

Figure V.4. Theoretical and simulated MSD of LMS as a function of the step-size for non-Gaussian regressors.

PART VI: BLOCK ADAPTIVE FILTERS

COMPUTER PROJECT

Project VI.1 (Acoustic echo cancellation) The programs that solve this project are the following.

1. partA.m This program generates two plots showing the impulse response sequence of the room as well as its frequency response up to 4 kHz, as shown in Fig. 1.

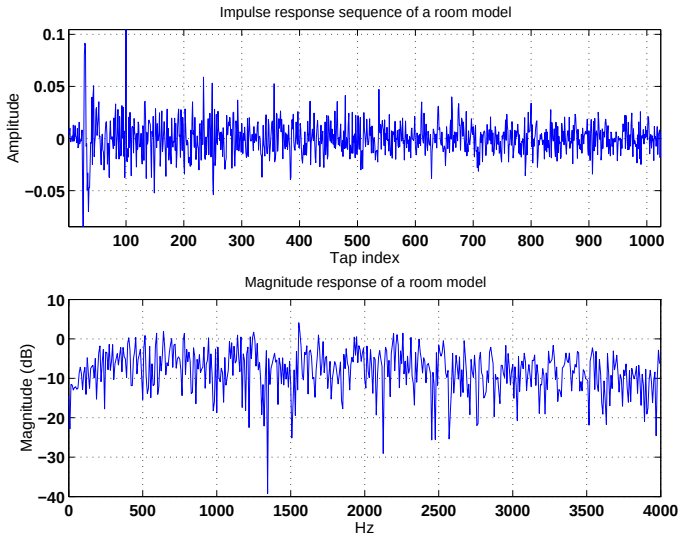

Figure VI.1. Impulse and frequency responses of a room model.

2. partB.m This program generates two plots. One plot shows the loudspeaker signal, the echo signal, and the error signals that are left after adaptation by $\epsilon-$NLMS using $\mu = 0.5$ and $\mu = 0.1$. The second plot shows the variation of the relative filter mismatch as a function of the step-size. Typical outputs are shown in Figs. 2 and 3.

   It is seen from Fig. 2 that the error signal is reduced faster for $\mu = 0.5$ than for $\mu = 0.1$.

3. partC.m This program generates a plot showing the loudspeaker signal, the echo signal, and the error signals that are left after adaptation by $\epsilon-$NLMS using $\mu = 0.1$ and $\epsilon-$NLMS using power normalization with $\mu = 0.1/32$ (see Fig. 4). The mismatch by the former algorithm is found to be -10.49 dB while the mismatch by the latter algorithm is found to be -9.72 dB. The performance of both algorithms is therefore similar although, of course, $\epsilon-$NLMS with power normalization is less complex computationally. Comparing the convergence of the error signal when $\mu = 0.1$ in the DFT-block adaptive implementation with that in Fig. 2, we see that the block implementation results in faster convergence.

4. partD.m This program generates a plot showing the loudspeaker signal, the echo signal, and the error signal that is left after adaptation using $\epsilon-$NLMS with power normalization and $\mu = 0.03/32$ — see Fig. 5.

5. partE.m This program generates learning curves for ϵ-NLMS and the constrained DFT-block adaptive implementation of Alg. 28.7 by averaging over 50 experiments and smoothing the resulting curves using a sliding window of width 10 — see Fig. 6, which shows the superior convergence performance of the block adaptive filter.

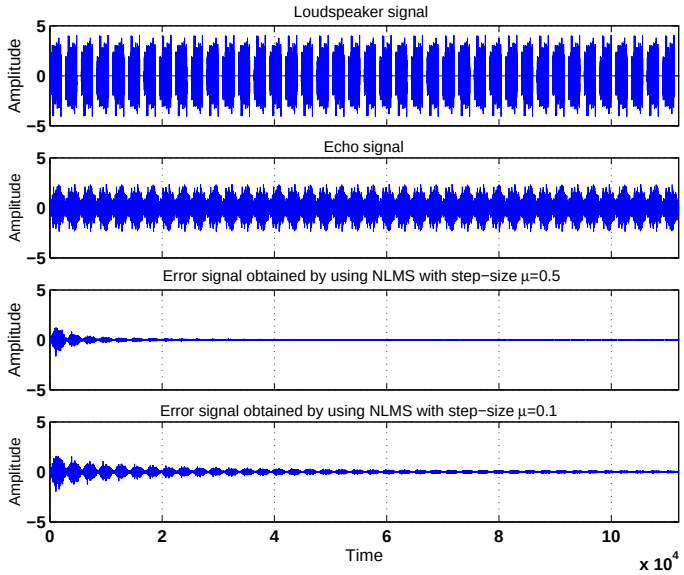

Figure VI.2. The top row shows the loudspeaker signal, while the second row shows the corresponding echo and the last two rows show the resulting error signal for different step-sizes in the $\epsilon-$NLMS implementation.

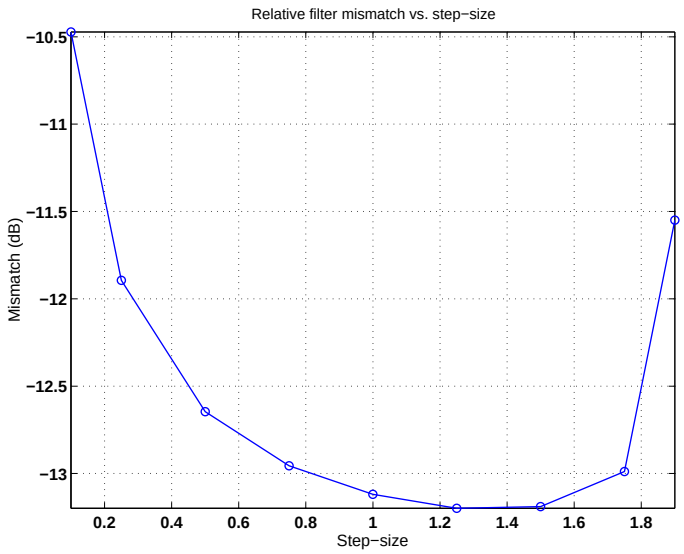

Figure VI.3. Relative filter mismatch for $\epsilon-$NLMS as a function of the step-size; the values are computed after training with 20 CSS blocks.

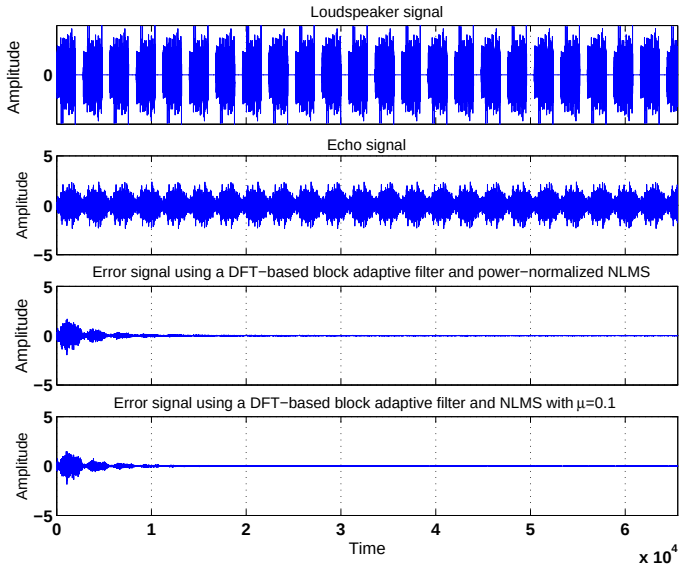

Figure VI.4. Performance of NLMS-based DFT-block adaptive implementations. The top row shows the loudspeaker signal, while the second row shows the corresponding echo and the last two rows show the error signals.

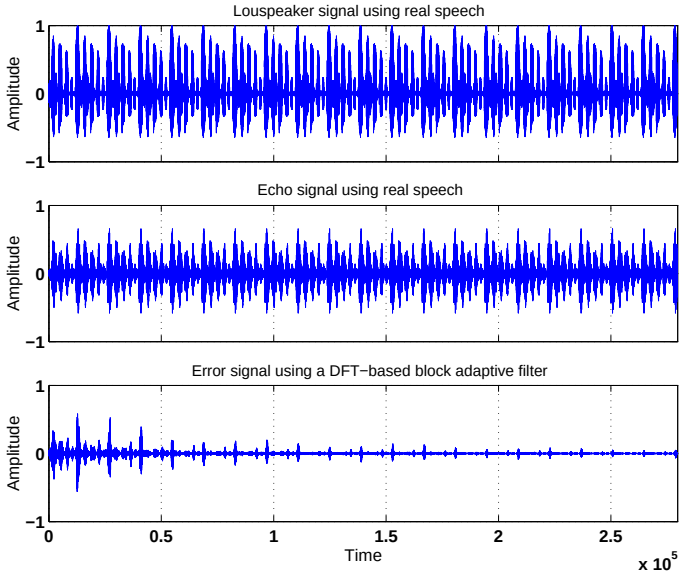

Figure VI.5. Performance of a DFT-block adaptive implementation applied to a real speech signal. The top row shows the loudspeaker signal, while the middle row shows the corresponding echo and the last row shows the error signal left after cancellation.

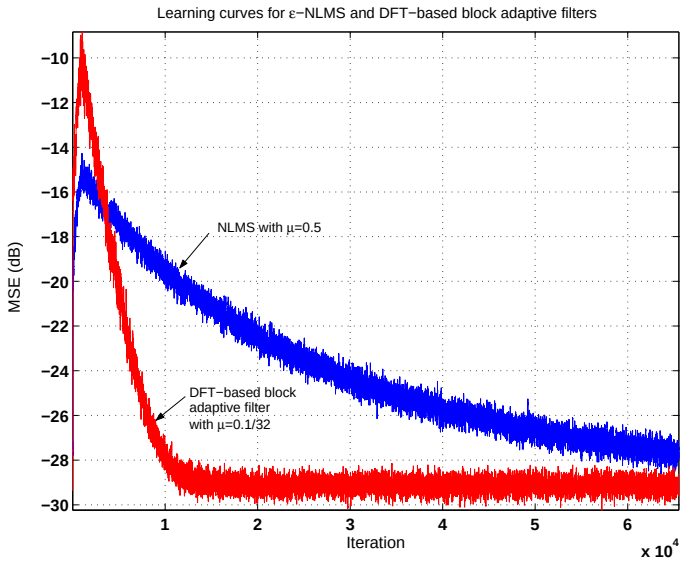

Figure VI.6. Learning curves for $\epsilon-$NLMS and the constrained DFT block adaptive filters using an AR(1) loudspeaker signal.

PART VII: LEAST-SQUARES METHODS

COMPUTER PROJECTS

Project VII.1 (OFDM receiver) The programs that solve this project are the following.

1. partA.m This program generates a plot showing the estimated channel taps both in the time domain and in the frequency domain for the two training schemes of Fig. 1. A typical output is shown in Figs. 1 and 2.

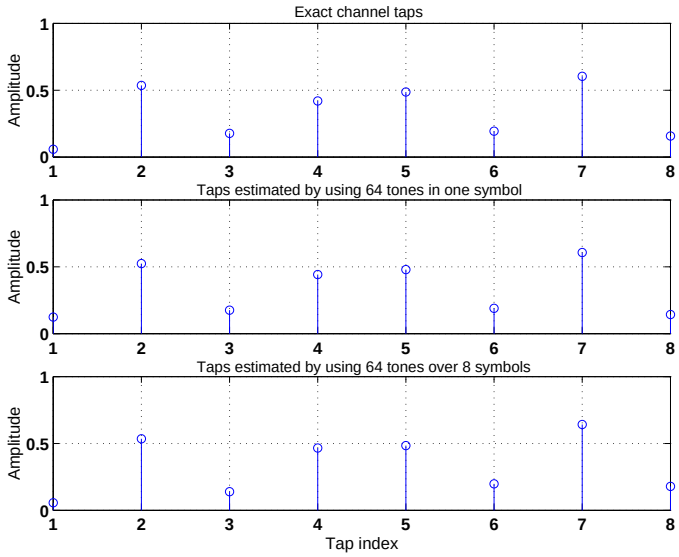

Figure VII.1. Magnitude of the channel taps in the time domain: exact taps (*top plot*), estimated taps using 64 tones in one symbol (*middle plot*) and estimated taps using 64 tones over 8 symbols (*bottom plot*).

2. partB.m This program generates a plot of the BER as a function of the SNR for both training schemes of Fig. 1 by averaging over 100 random channels. A typical plot is shown in Fig. 3. The program also plots the scatter diagrams of the transmitted, received, and equalized data at SNR = 25 dB. A typical plot is shown in Fig. 4.

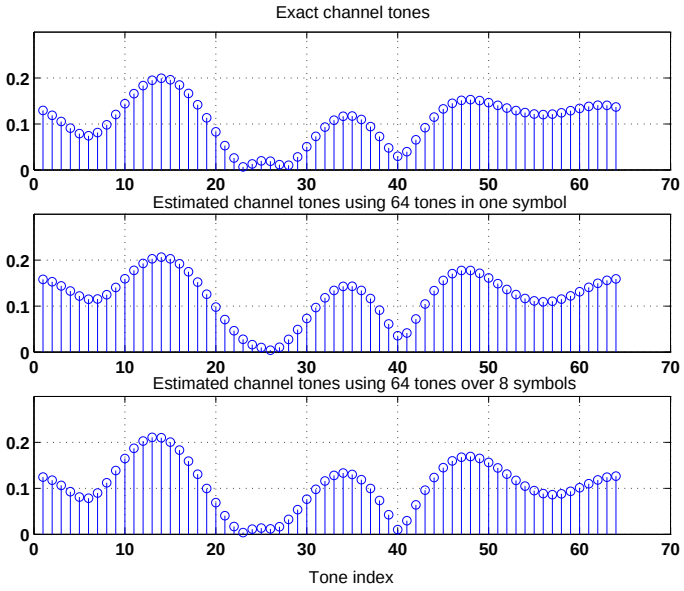

Figure VII.2. Magnitude of the channel tones (Λ) in the frequency domain: exact channel tones (*top plot*), estimated tones using 64 tones in one symbol (*middle plot*), and estimated tones using 64 tones over 8 symbols (*bottom plot*).

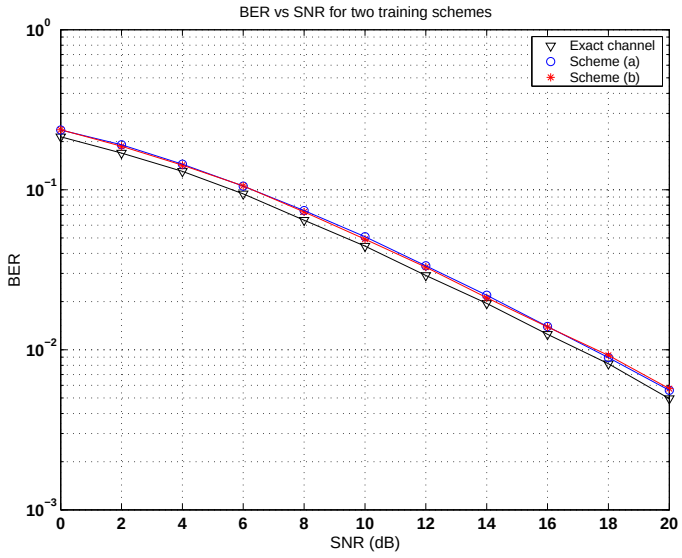

Figure VII.3. BER versus SNR for the two training schemes of Fig. 10.7 using $N = 64$ (*symbol size*), $P = 16$ (*cyclic prefix length*), and $M = 8$ (*channel length*).

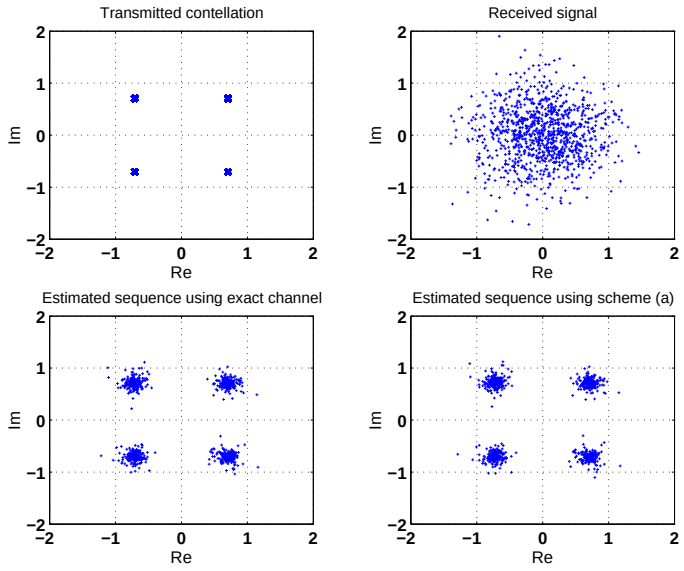

Figure VII.4. Scatter diagrams of the transmitted and received sequences (*top row*) and the recovered data using the exact channel and training scheme (a) from Fig. 10.7 (*bottom row*) at $\mathsf{SNR} = 25$ dB.

Project VII.2 (Tracking Rayleigh fading channels) The programs that solve this project are the following.

1. partA.m This program generates two plots. One plot shows the ensemble-average learning curves for ϵ-NLMS, RLS, and extended RLS — see Fig. 5. With the values chosen for f_D and T_s, the values of α and q turn out to be $\alpha = 0.99999999936835$ and $q = 1.263309457044670 \times 10^{-9}$. That is, α is essentially equal to one and q is negligible so that the time-variation in $\boldsymbol{w}_i^o$ is slow. For this reason, there is practically no difference in performance between RLS and its extended version. In addition, in this situation, although RLS and extended RLS converge faster than ϵ-NLMS, the steady-state tracking performance for all algorithms is similar. Moreover, recall from Table 21.2 that the steady-state MSE performance of ϵ-NLMS and RLS can be approximated by

$$\begin{aligned}\mathsf{MSE}^{\mathsf{LMS}} &= \frac{\mu\sigma_v^2}{2-\mu}\mathsf{Tr}(R_u)\mathsf{E}\left(\frac{1}{\|\boldsymbol{u}_i\|^2}\right) + \frac{\mu^{-1}}{2-\mu}\mathsf{Tr}(R_u)\mathsf{Tr}(Q)\\ \mathsf{MSE}^{\mathsf{RLS}} &= \frac{\sigma_v^2(1-\lambda)M \;+\; \frac{1}{(1-\lambda)}\mathsf{Tr}(QR_u)}{2-(1-\lambda)M}\end{aligned}$$

where, in our case, $Q = q\mathrm{I}$, $R_u = \mathrm{I}$, $M = 5$, $\sigma_v^2 = 0.001$, $\lambda = 0.995$. These expressions result in the theoretical values $\mathsf{MSE}^{\mathsf{LMS}} \approx -28.95\mathsf{dB}$ and $\mathsf{MSE}^{\mathsf{RLS}} \approx -29.77\mathsf{dB}$, where the expectation $\mathsf{E}\,(1/\|\boldsymbol{u}_i\|^2)$ is evaluated numerically via ensemble-averaging. The values obtained from simulation (by averaging the last 100 values of the ensemble-average error curves) are in good match with theory, namely, $\mathsf{MSE}^{\mathsf{LMS}} \approx -29.10\mathsf{dB}$ and $\mathsf{MSE}^{\mathsf{RLS}} \approx -28.57\mathsf{dB}$ (from simulation). In Fig. 6, we also plot ensemble-average curves for the mean-square deviation, $\mathsf{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2$. It is again seen that the steady-state performances are similar and they tend to approximately -70 dB.

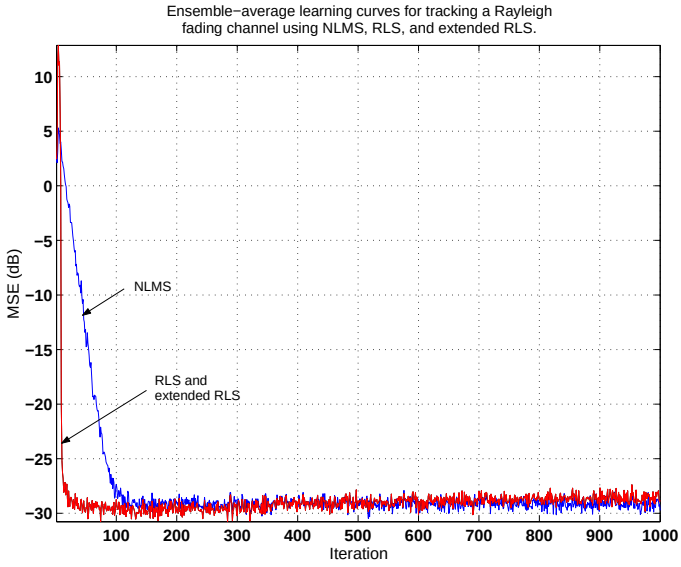

Figure VII.5. Ensemble-average learning curves for $\epsilon-$NLMS, RLS, and extended RLS when used in tracking a Rayleigh fading multipath channel with $f_D = 10\mathsf{Hz}$ and $T_s = 0.8\mu\mathsf{s}$.

2. partB.m This program generates two plots for the case $f_D = 80\mathsf{Hz}$. One plot shows the ensemble-average learning curves for ϵ-NLMS, RLS, and extended RLS — see Fig. 7. With the values chosen for f_D and T_s, the value of α is still close to one and the value of q is still small, namely, $\alpha = 0.99999995957410$ and $q = 8.085179670214160 \times 10^{-8}$. For this reason, there is still practically no difference in performance between RLS and its extended version. Now, however, it is observed that the steady-state performance of ϵ-NLMS is superior:

$$\mathsf{MSE}^{\mathsf{LMS}} \approx -28.34\ \mathsf{dB}, \qquad \mathsf{MSE}^{\mathsf{RLS}} \approx -16.73\ \mathsf{dB} \quad \text{(simulation)}$$

In Fig. 8, we also plot ensemble-average curves for the mean-square deviation, $\mathsf{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2$.

3. partC.m This program generates two plots. One plot shows the ensemble-average learning and mean-square error curves for ϵ-NLMS, RLS, and extended RLS for the case of $\alpha = 0.95$ and $q = 0.1$ — see

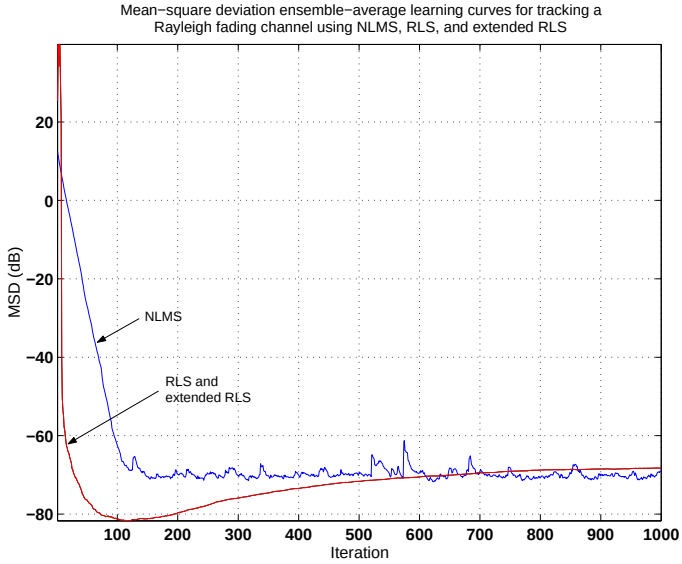

Figure VII.6. Ensemble-average mean-square deviation curves for $\epsilon-$NLMS, RLS, and extended RLS when used in tracking a Rayleigh fading multipath channel with $f_D = 10$Hz and $T_s = 0.8\mu$s.

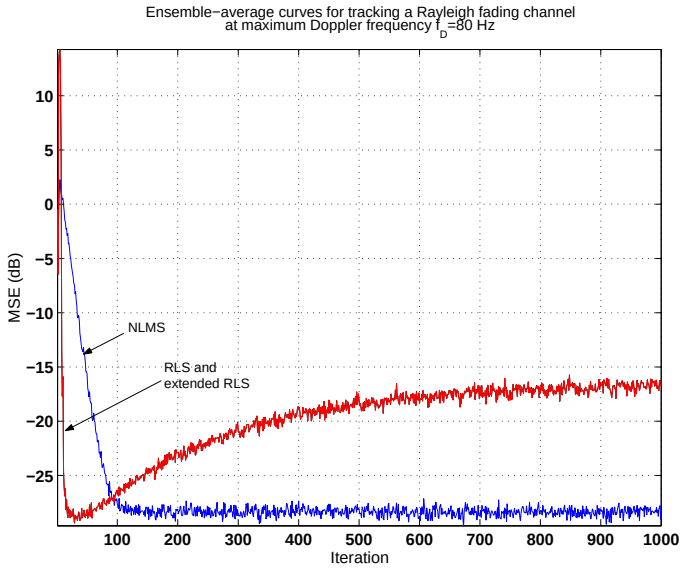

Figure VII.7. Ensemble-average learning curves for $\epsilon-$NLMS, RLS, and extended RLS when used in tracking a Rayleigh fading multipath channel with $f_D = 80$Hz and $T_s = 0.8\mu$s.

Fig. 9, while the other shows the same curves for the case $\alpha = 1$ and $q = 0.1$ — see Fig. 10. It is seen from these figures that the performance of extended RLS is now superior.

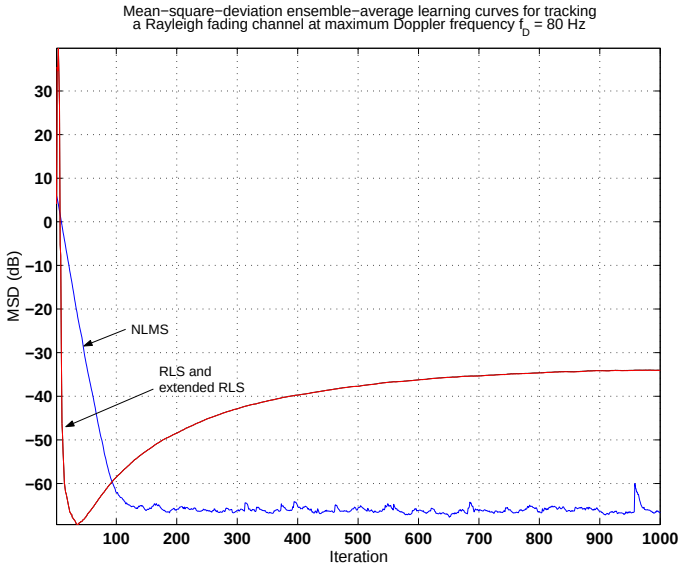

Figure VII.8. Ensemble-average mean-square deviation curves for $\epsilon-$NLMS, RLS, and extended RLS when used in tracking a Rayleigh fading multipath channel with $f_D = 80$Hz and $T_s = 0.8\mu$s.

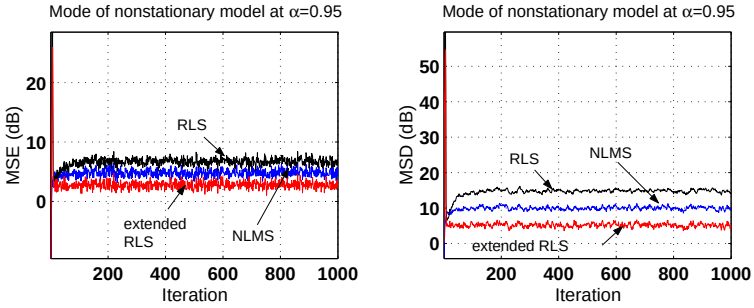

Figure VII.9. Ensemble-average learning (*left*) and mean-square deviation (*right*) curves for $\epsilon-$NLMS, RLS, and extended RLS when used in tracking a multipath channel with $\alpha = 0.95$ and $q = 0.1$.

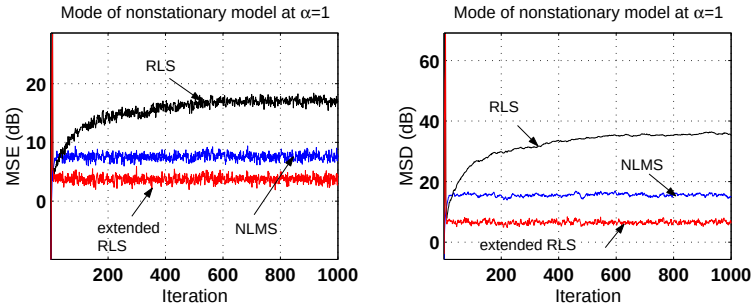

Figure VII.10. Ensemble-average learning (*left*) and mean-square deviation (*right*) curves for $\epsilon-$NLMS, RLS, and extended RLS when used in tracking a multipath channel with $\alpha = 1$ and $q = 0.1$.

PART VIII: ARRAY ALGORITHMS

COMPUTER PROJECT

Project VIII.1 (Performance of array implementations in finite precision) The program that solves this project is the following.

1. partA.m The program generates ensemble-average learning curves for RLS and its QR array variant using three choices for the number of bits, $\{30, 25, 16\}$ and averaging over 100 experiments. A typical output is shown in Fig. 1. It is seen that the QR array method is superior in finite-precision implementations. For instance, at 16 bits, RLS fails while the QR method still functions properly.

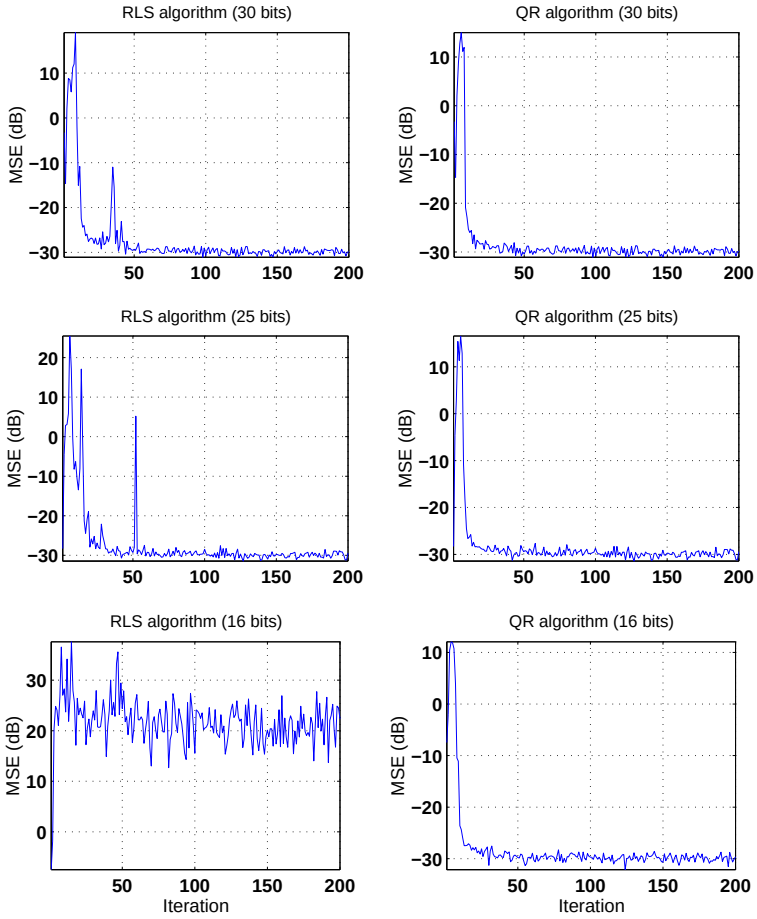

Figure VIII.1. Ensemble-average learning curves for RLS and its QR array variant for three choices of the number of quantization bits.

PART IX: FAST RLS ALGORITHMS

COMPUTER PROJECT

Project IX.1 (Stability issues in fast least-squares) The programs that solve this project are the following.

1. partA.m This program generates ensemble-average learning curves for four fast fixed-order implementations. A typical output is shown in Fig. 1. It is seen that in floating-point precision, the algorithms behave rather similarly.

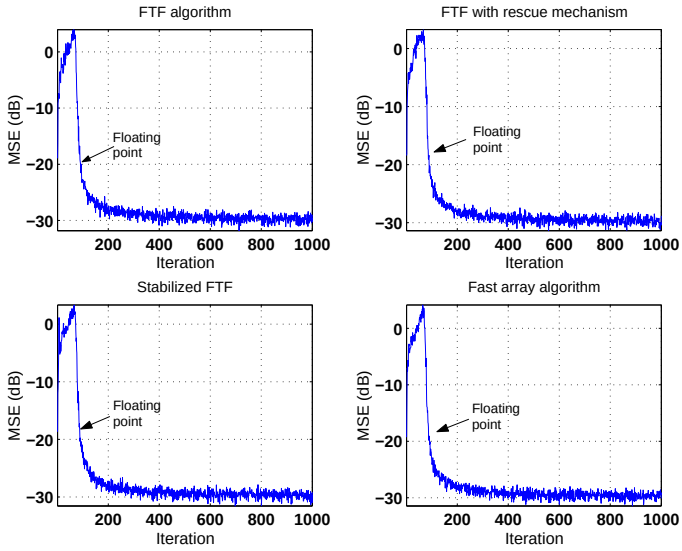

Figure IX.1. Ensemble-average learning curves for four fast fixed-order filter implementations in floating-point precision.

2. partB.m This program generates ensemble-average learning curves for four fast fixed-order filters using quantized implementations with 36 bits for both signals and coefficients. A typical output is shown in Fig. 2. It is seen that, even at this number of bits, the standard FTF implementation becomes unstable. When implemented with the rescue mechanism (39.10), the rescue procedure is called upon on several occasions but it still fails to recover filter performance. The stabilized FTF implementation also fails in this experiment. Only the fast array implementation is able to maintain the original filter performance in this case.

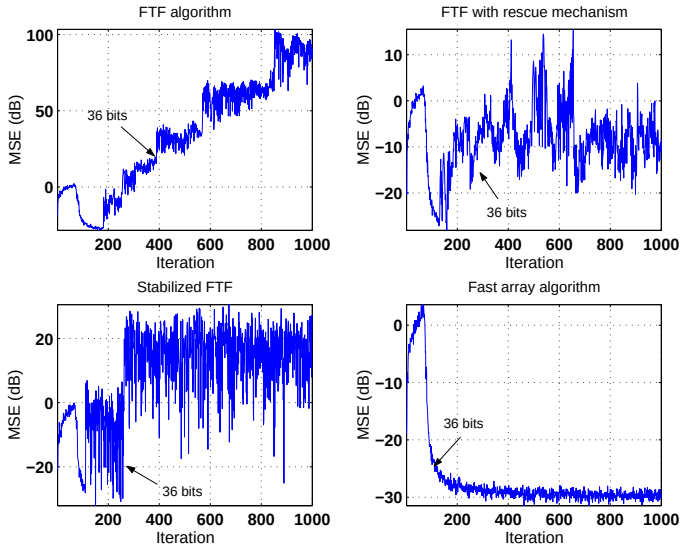

Figure IX.2. Ensemble-average learning curves for four fast fixed-order filter implementations averaged over 100 experiments in a quantized environment with 36 bits.

PART X: LATTICE FILTERS

COMPUTER PROJECT

Project X.1 (Performance of lattice filters in finite precision) The programs that solve this project are the following.

1. partA.m This program generates ensemble-average learning curves for the various lattice filters for different choices of the number of bits. The results are shown in Figs. 1 through 5. All filters work well at 35 bits, but some filters start facing difficulties at lower number of bits. It seems from the figures that the array lattice form is the most reliable in finite precision, while the *a posteriori* lattice form with error feedback is the least reliable.

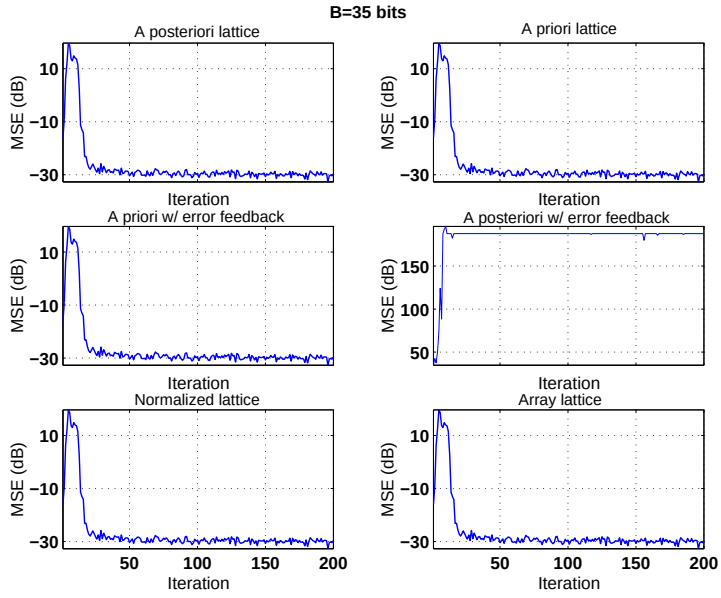

Figure X.1. Ensemble-average learning curves for various lattice implementations using 35 bits.

2. partB.m This program generates ensemble-average learning curves for the various lattice filters in the presence of an impulsive interference at iteration $i = 200$ and assuming floating-point arithmetic. The result is shown in Fig. 6. It is seen that all algorithms recover from the effect of the impulsive disturbance.

3. partC.m This program generates ensemble-average learning curves for the various lattice filters in the presence of an impulsive interference at iteration $i = 200$ and assuming finite-precision arithmetic with $B = 20$ and $B = 16$ bits. The results are shown in Figs. 7 and 8.

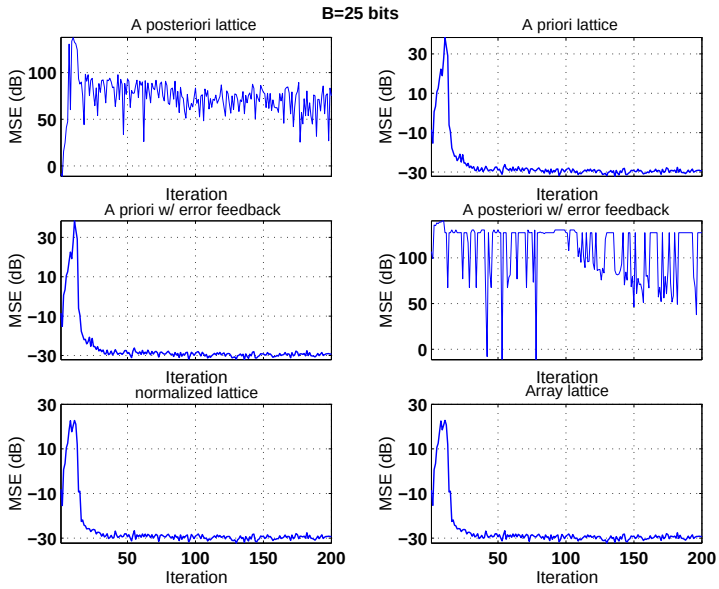

Figure X.2. Ensemble-average learning curves for various lattice implementations using 25 bits.

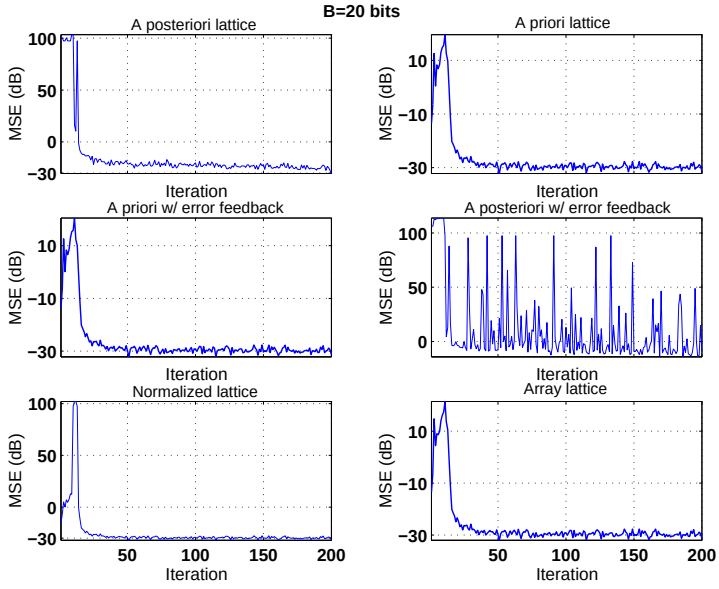

Figure X.3. Ensemble-average learning curves for various lattice implementations using 20 bits.

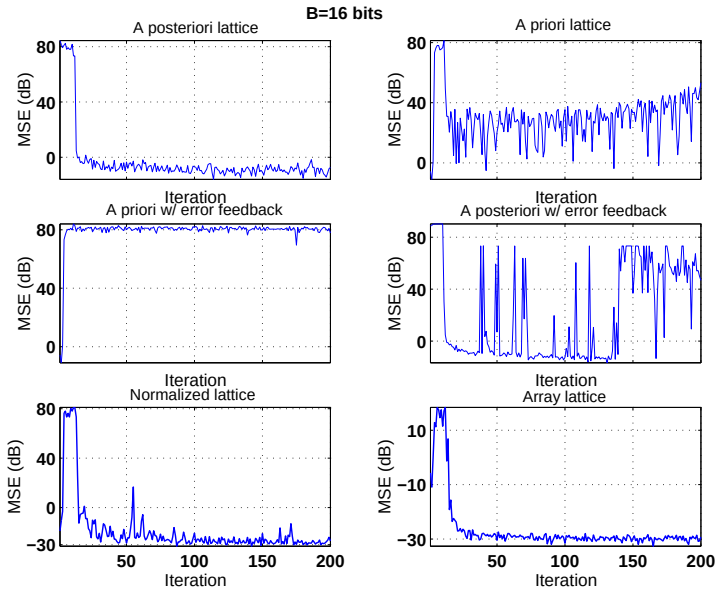

Figure X.4. Ensemble-average learning curves for various lattice implementations using 16 bits.

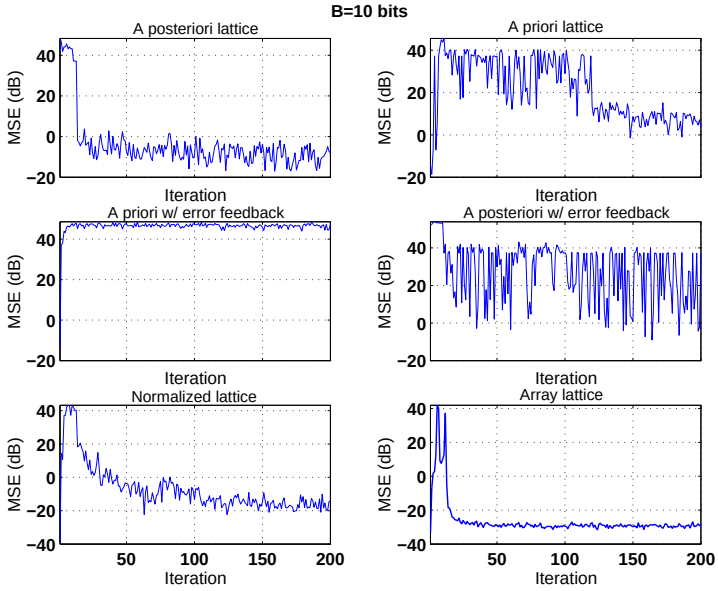

Figure X.5. Ensemble-average learning curves for various lattice implementations using 10 bits.

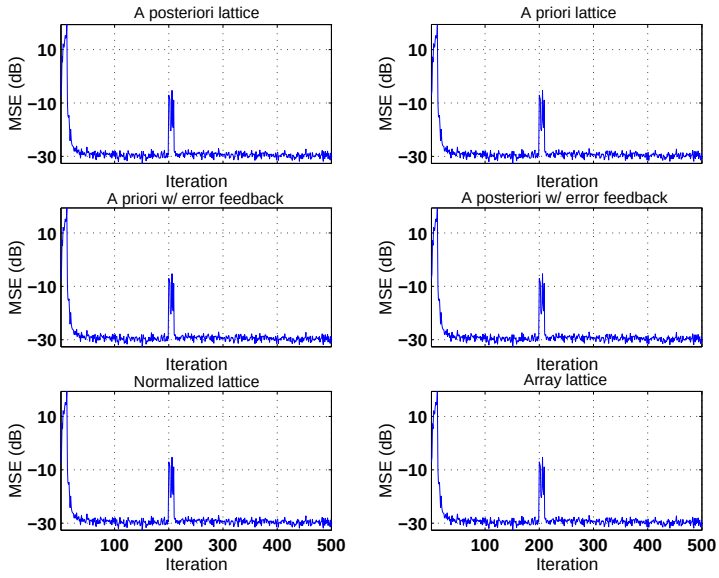

Figure X.6. Ensemble-average learning curves for various lattice implementations in floating-point precision with an impulsive disturbance occurring at iteration $i = 200$.

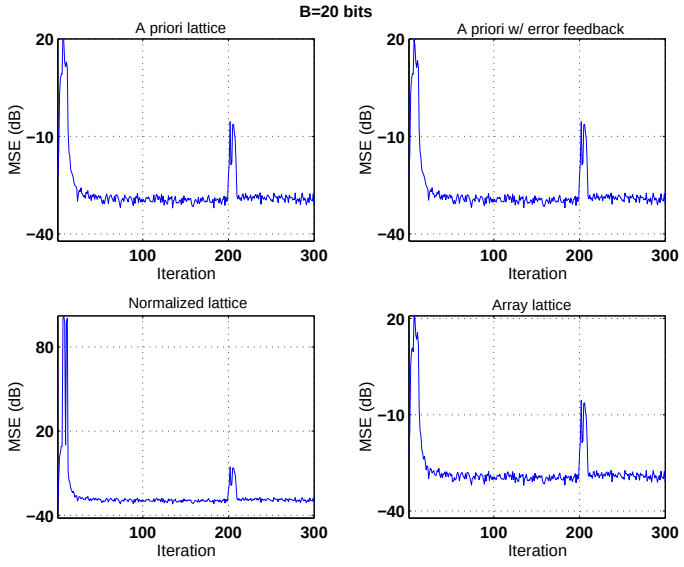

Figure X.7. Ensemble-average learning curves for various lattice implementations in 20-bits precision with an impulsive disturbance occurring at iteration $i = 200$.

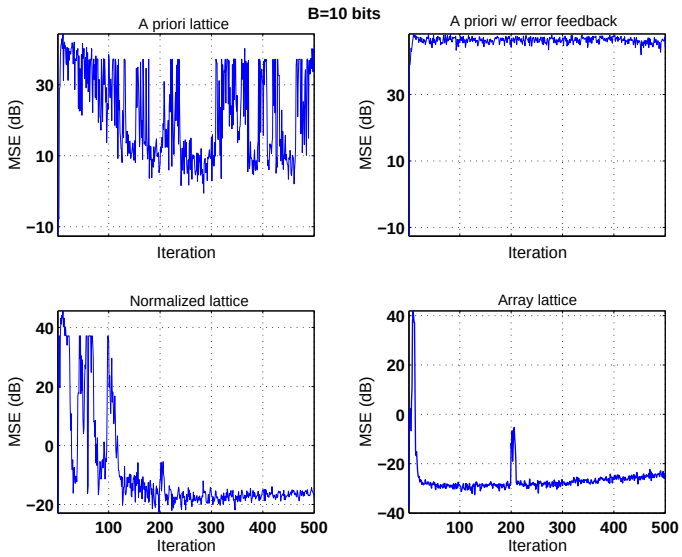

Figure X.8. Ensemble-average learning curves for various lattice implementations in 10-bits precision with an impulsive disturbance occurring at iteration $i = 200$.

PART XI: ROBUST FILTERS

COMPUTER PROJECT

Project XI.1 (Active noise control) The programs that solve this project are the following.

1. partA.m This program generates a plot of the function $g(\alpha)$, which is shown in Fig. 1. The plot on the right zooms on the interval $\alpha \in [0, 0.1]$. It is seen that $g(\alpha) < 1$ for $\alpha < 0.062$. Also, $g(\alpha)$ is minimized at $\alpha = 0.031$. Note however that for all values of α in the range $\alpha \in (0, 0.062)$, the function $g(\alpha)$ is essentially invariant and assumes values very close to one.

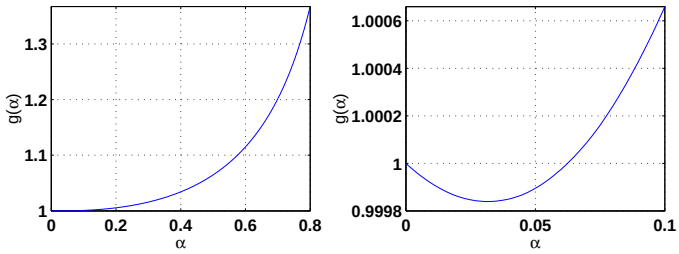

Figure XI.1. A plot of the function $g(\alpha)$ (left) and a zoom on the interval $\alpha \in (0, 0.062)$.

2. partB.m This program generates ensemble-average mean-square deviation curves (i.e., $\mathsf{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2$) for several values of α using binary input data. A typical output is shown in Fig. 2.

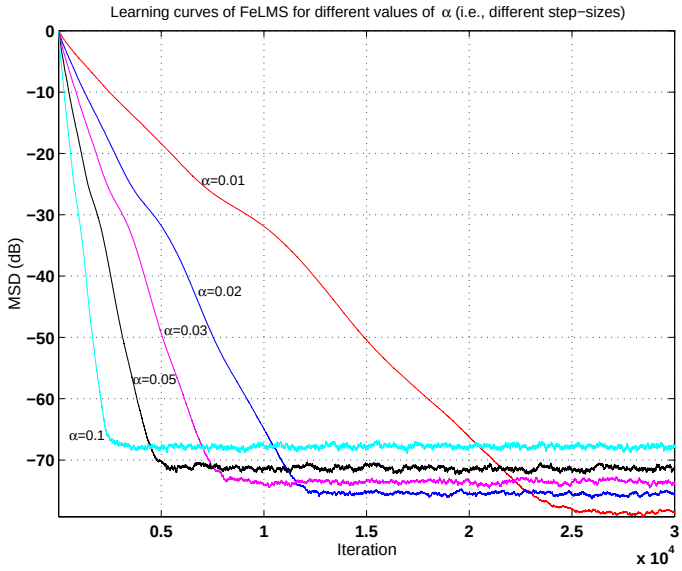

Figure XI.2. Ensemble-average mean-square deviation curves of the filtered-error LMS algorithm for various choices of α and for binary input data.

3. partC.m This program generates ensemble-average learning curves for the filtered-error algorithm for several values of α and using both binary input and white Gaussian input. Typical outputs are shown in Figs. 3–4. It is observed that the algorithm shows unstable behavior for values of α beyond 0.5.
4. partD.m This program generates ensemble-average learning curves for four algorithms with $\alpha = 0.15$. A typical output is shown in Fig. 5. It is seen that NLMS shows best convergence performance, followed by filtered-error LMS, modified filtered-x LMS and filtered-x LMS. It is also seen that the modification to FxLMS improves convergence.

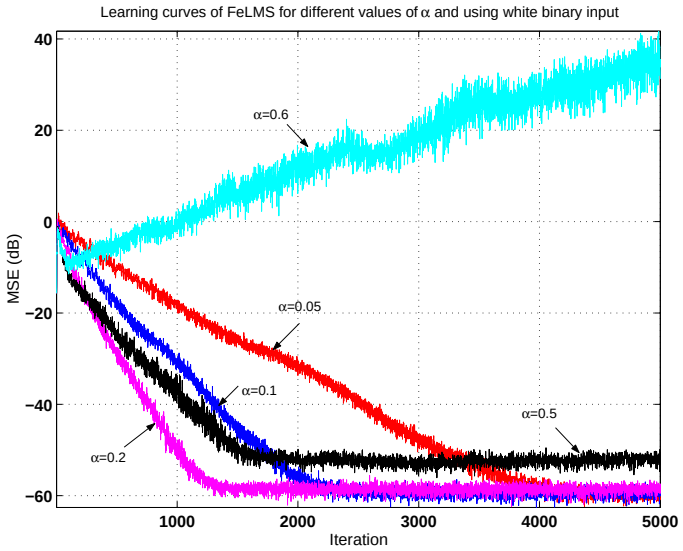

Figure XI.3. Ensemble-average learning curves of filtered-error LMS for various choices of α and for both binary input.

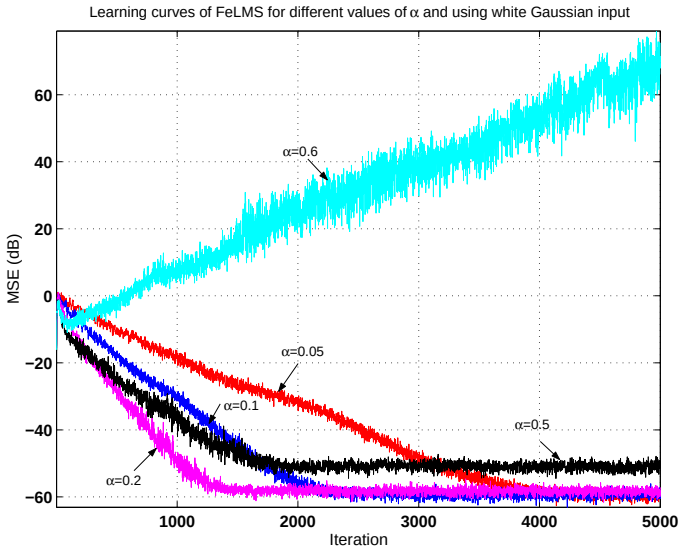

Figure XI.4. Ensemble-average learning curves of filtered-error LMS for various choices of α and for Gaussian input.

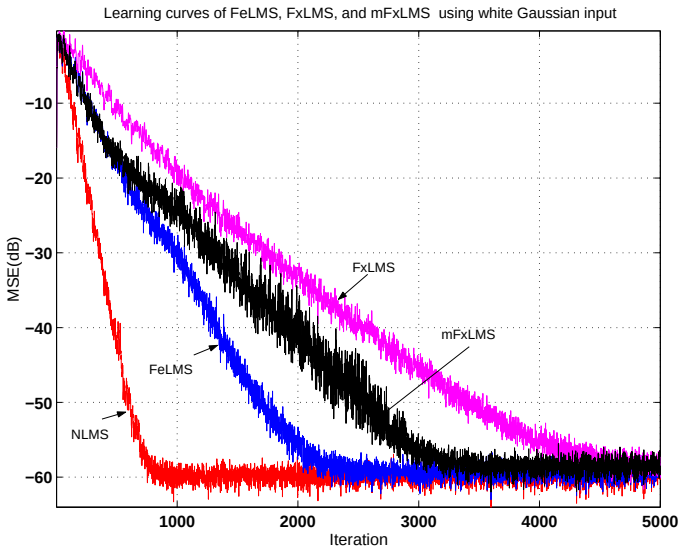

Figure XI.5. Ensemble-average learning curves of NLMS, FeLMS, FxLMS and mFxLMS for $\alpha = 0.15$ and white Gaussian input.