Signal Processing and Speech Communication Lab.
Graz University of Technology

AS

Adaptive Systems—Homework Assignment 2 v1.0

Name(s)	Matr.No(s).
Hannes Reindl	01532129
Philip Samer	01634718

Your solutions to the problems (your calculations, the answers to each task as well as the OCTAVE/MATLAB plots) have to be uploaded to the TeachCenter as a single `*.pdf` file, no later than **2019/1/20**. Use **this page** as the title page and fill in your **name(s) and matriculation number(s)**. Submitting your homework as a LaTeX document can earn you **up to 3 bonus points!**
All scripts needed for your OCTAVE/MATLAB solutions (all `*.m` files) have to be uploaded to the TeachCenter as a single `*.zip` archive, no later than **2019/1/20**.
All filenames consist of the assignment number and your matriculation number(s) such as `Assignment2_MatrNo1_MatrNo2.*`, for example,

Problem solutions:	`Assignment2_01312345_01312346.pdf`
OCTAVE/MATLAB files:	`Assignment2_01312345_01312346.zip`

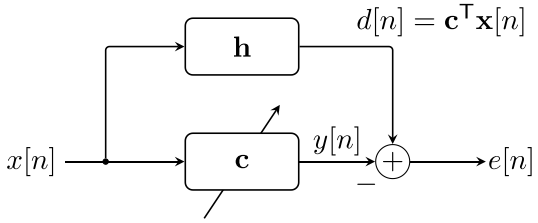
Please make sure that your approaches, procedures, and results are clearly presented. Justify your answers! A single upload of the files per group is sufficient.

Deterministic-Gradient Algorithms (Steepest-Descent Algorithms)
Algorithms belonging to the class of *deterministic* gradient overcome the problem of inverting the autocorrelation matrix $\mathbf{R}_{xx}$ needed to compute the optimum filter coefficients in the MSE-sense, i.e., the Wiener-Hopf solution. The gradient search algorithm in turn iteratively updates the coefficients according to the equation

$$\mathbf{c}[n] = \mathbf{c}[n-1] - \tilde{\mu}\nabla_{\mathbf{c}} J(\mathbf{c})\big|_{\mathbf{c}=\mathbf{c}[n-1]} \tag{1}$$

by tacking a step in the direction of the negative gradient. As the gradient of the MSE cost function $J(\mathbf{c}) = J_{\text{MSE}}(\mathbf{c})$ is used, the behaviour of these algorithm(s) is deterministic.

Analytical Problem 2.1 (13 Points)—Deterministic-Gradient Algorithm



A variation of the standard gradient search from (1) is the *coefficient–leakage* gradient search algorithm. This algorithm uses the coefficient update rule

$$\mathbf{c}[n] = (1 - \mu\alpha)\mathbf{c}[n-1] + \mu(\mathbf{p} - \mathbf{R}_{xx}\mathbf{c}[n-1]) \tag{2}$$

where $\alpha \in \mathbb{R}$ is the leakage coefficient ($0 \leq \alpha < 1$) and $\mu \in \mathbb{R}$ the step-size. Assume that $x[n] \in \mathbb{R}$ and $d[n] \in \mathbb{R}$. Use an arbitrary filter order N, i.e., $\mathbf{c}[n] \in \mathbb{R}^N$

(a) (2 points) Show how to derive the update equation for the coefficient–leakage gradient search algorithm from the modified cost function

$$J(\mathbf{c}[n]) = \mathsf{E}\big[e^2[n]\big] + \alpha||\mathbf{c}[n]||^2$$

(b) (2 points) Assume that the algorithm converged. Where does it converge to?

(c) (2 points) Transform the update equation for the filter coefficients $\mathbf{c}[n]$ to adapt the misalignment vector defined as $\mathbf{v}[n] = \mathbf{c}[n] - \mathbf{c}_\infty$.

(d) (2 points) Apply a unitary coordinate transform $\mathcal{T}(\cdot)$ such that the transformed components of the misalignment vector $\tilde{\mathbf{v}}[n] = \mathcal{T}(\mathbf{v}[n])$ are decoupled.

(e) (2 points) Use the decoupled update equation of the misalignment vector and express it as a function of time n and the initial transformed misalignment vector $\tilde{\mathbf{v}}[0]$.

(f) (3 points) Find an expression of the time constants for each of the decoupled coefficients. What is the influence of the leakage parameter α?

OCTAVE/MATLAB Problem 2.2 (6 Points)—Signal Statistics

The optimal solution to a system identification problem in the mean-square error sense can be found in closed form by using the Wiener-Hopf or iteratively by applying the *gradient descent* algorithm, both of which rely on the knowledge of the signal statistics. Assuming stationarity, one needs to know the autocorrelation of the filter input signal $x[n] \in \mathbb{R}$

$$r_{xx}[k] = \mathsf{E}\big[x[n+k]x[n]\big]$$

as well as the cross-correlation between $x[n]$ and the reference signal $d[n] \in \mathbb{R}$ defined as

$$p[k] = \mathsf{E}\big[d[n]x[n-k]\big].$$

Computing the statistical expectation $\mathsf{E}\big[\cdot\big]$ (or *ensemble average*) requires averaging over different realizations of a random variable or process, which needs to be approximated in an actual application scenario. The *mean ergodic* (m.e.) and *correlation ergodic* (c.e.) theorem state that the ensemble averages can be exchanged with time averages under certain conditions, which are usually fulfilled by wide sense stationary (WSS) processes. These time-averages for mean and correlation are defined in terms of the sample averages as

$$\mu_x = \mathsf{E}\big[x[n]\big] \quad \overset{\text{m.e.}}{\longleftrightarrow} \quad \hat{\mu}[P] = \frac{1}{P}\sum_{n=0}^{P-1} x[n]$$

$$r_{xy}[k] = \mathsf{E}\big[x[n+k]y[n]\big] \quad \overset{\text{c.e.}}{\longleftrightarrow} \quad \hat{r}_{xy}[k,P] = \frac{1}{P}\sum_{n=0}^{P-1} x[n+k]y[n], \qquad 0 \le k \le P-1$$

and are computed over a signal window of size P. In connection to system identification it will be sufficient to evaluate time lags $0 \le k \le N-1$ with $N \le P$ (implying the need for at least as many samples to average as filter coefficients N).

(a) (2 points) Show analytically that the expected values of both sample averages are equal to the quantities that are approximated, i.e., show that $\mathsf{E}\big[\hat{\mu}[P]\big] = \mu_x$ and $\mathsf{E}\big[\hat{r}_{xy}[k,P]\big] = r_{xy}[k]$ assuming $x[n]$ and $y[n]$ are jointly WSS.

(b) (2 points) Write an Octave/Matlab script that uses the sample correlation $\hat{r}_{xy}[k,P]$ to estimate the cross- and autocorrelation functions $r_{dx}[k] = p[k]$ and $r_{xx}[k]$ for the signals $x[n]$ and $d[n]$ provided in the file `data.mat`. Refer to Appendix 1 for a detailed description of the file content.
Hint: Be careful that you are not allowed to mix different realizations of $x[n]$ and $d[n]$ when computing the sample correlations.

(c) (2 points) Compute the sample estimates for different window sizes P for different signal realizations and compare them to the analytical values for $r_{xx}[k]$ and $p[k]$ given in Appendix 1. Plot and discuss the results. How does the accuracy of the estimates scale with the window size P?
Hint: Compute the statistics separately for some realizations, you do not need to use all.

Stochastic-Gradient Algorithms
In contrast to the gradient search algorithm where a *deterministic* gradient is used to iteratively compute the Wiener-Hopf solution, the LMS algorithm uses a *stochastic* approximation of the gradient and is thus part of the class of *stochastic* gradient algorithms. As shown in the problem class, the update equation for the LMS-algorithm assuming real signals

$$\mathbf{c}[n] = \mathbf{c}[n-1] + \mu e[n]\mathbf{x}[n] \tag{3}$$

can be derived from the deterministic gradient algorithm by using an instantaneous estimate of the gradient found to be the input signal $\mathbf{x}[n]$ scaled with the *a-priori* error $e[n]$. The result is a random update of the filter coefficients form one iteration cycle to the next, justifying the use of the term *stochastic* gradient.

Analytical Problem 2.3 (6 Points)—LMS Algorithm

Two often used variations of the standard LMS algorithm given in (3) either employ a leakage factor applied to the previous coefficients or a signal power dependent step size. The corresponding update equations assuming real signals are then

$$\text{coefficient-leakage LMS}: \quad \mathbf{c}[n] = (1-\mu\alpha)\mathbf{c}[n-1] + \mu e[n]\mathbf{x}[n] \qquad 0 \leq \alpha < 1 \tag{4}$$

$$\text{normalized LMS}: \quad \mathbf{c}[n] = \mathbf{c}[n-1] + \frac{\mu}{||\mathbf{x}[n]||^2 + \alpha} e[n]\mathbf{x}[n]. \qquad \alpha > 0 \tag{5}$$

(a) (4 points) The convergence properties of the LMS algorithm can only be examined *on average*. Using the assumption of convergence on average, i.e., assuming that

$$\mathsf{E}\big[\mathbf{c}[n]\big] = \mathsf{E}\big[\mathbf{c}[n-1]\big] = \mathsf{E}\big[\mathbf{c}_\infty\big]$$

compute the expected converged solution of the LMS (3) and the coefficient–leakage LMS (4) for a noisy system identification scenario assuming real coefficients. For your derivations assume a white Gaussian input signal $x[n] \in \mathbb{R}$ with non-zero variance σ_x^2 and uncorrelated white Gaussian noise $w[n]$ with non-zero variance σ_w^2 (e.g. see Fig. 1 below). Compare the results you obtained with a general system response $\mathbf{h} \in \mathbb{R}^N$. What do you observe?
Hint: Compute $\mathsf{E}\big[\mathbf{c}[n]\big]$ to find the converged coefficients in terms of a general, unknown system response $\mathbf{h}$.

(b) (2 points) Explain the effect of the normalization term in the normalized LMS algorithm (5). What is the effect of α?
Hint: Think about the step-size μ and its relation to convergence.

OCTAVE/MATLAB Problem 2.4 (7 Points)—Gradient Algorithms

Implement the standard *deterministic* and *stochastic* gradient algorithms examined in theory in the previous tasks, and use the provided script `gradient_test.m` to test the performance. The script and some needed data files are contained in `gradient_test.zip`.

(a) (4 points) Write Matlab a function `lms_algorithm()` that implements the standard LMS algorithm (3) and the normalized LMS (5) according to the following specification:

```
function [y,e,c] = lms_algorithm(x,d,N,mu,alpha,OPTS,c0)
% INPUTS:
% x ....... input signal vector (column vector)
% d ....... desired output signal (of same dimensions as x)
% N ....... number of filter coefficients
% mu ...... step-size parameter
% alpha ... algorithm dependent parameter
% OPTS .... 0 for standard LMS, 1 for normalized LMS
% c0 ...... initial coefficient vector (optional column vector; default all zeros)
% OUTPUTS:
% y ....... output signal vector (same length as x)
% e ....... error signal vector (same length as x)
% c ....... coefficient matrix (N rows, number of columns = length of x)
```

(b) (2 points) Write Matlab a function `gd_algorithm()` implementing the standard gradient descent algorithm (1) according to the specification below. Be careful, that in this form the signal statistics are estimated beforehand and not adapted/changed during execution.

```
function [y,e,c] = gd_algorithm(x,d,N,mu,Rxx,p,c0)
% INPUTS:
% x ....... input signal vector (column vector)
% d ....... desired output signal (of same dimensions as x)
% N ....... number of filter coefficients
% mu ...... step-size parameter
% Rxx ..... autocorrelation matrix
% p ....... cross-correlation vector (column vector)
% c0 ...... initial coefficient vector (optional column vector; default all zeros)
% OUTPUTS:
% y ....... output signal vector (same length as x)
% e ....... error signal vector (same length as x)
% c ....... coefficient matrix (N rows, number of columns = length of x)
```

(c) (1 points) The first 3 plots are only for validation of your implementation. Briefly explain and discuss what you observe in the fourth plot.

OCTAVE/MATLAB Problem 2.5 (10 Points)—Performance Analysis

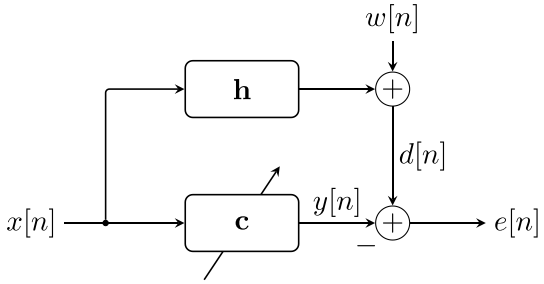


Figure 1: System identification in a noisy environment.

This problem will quantify and compare the performance of the *deterministic* and *stochastic* gradient algorithms implemented in the previous task. The input and reference data for the system is provided in the file `data.mat`, providing you with multiple realizations of the needed signals $x[n]$ and $d[n]$ and with the unknown filter coefficients $\mathbf{h}$. Refer to Appendix 1 for a detailed description of the content of `data.mat` and the general system setup.
For the noisy system identification problem shown in Fig. 1 we want to investigate the convergence behavior of both the deterministic and the stochastic gradient algorithms implemented in the previous task. For investigating the performance we will use the misalignment vector $\mathbf{v}[n]$ defined as

$$\mathbf{v}[n] = \mathbf{c}[n] - \mathbf{h}.$$

For each subtask, visualize the behavior of each component k of the misalignment vector over time n as

$$\ln \frac{|\mathsf{E}\big[v_k[n]\big]|}{|\mathsf{E}\big[v_k[0]\big]|}.$$

Also plot the logarithm of the MSE, i.e., plot

$$\ln \frac{\mathsf{E}\big[e^2[n]\big]}{\mathsf{E}\big[e^2[0]\big]}.$$

To compute the expectations $\mathsf{E}\big[\cdot\big]$ you can average over the results of different realizations using

$$\mathsf{E}\big[\mathbf{v}[n]\big] \approx \frac{1}{R}\sum_{r=1}^{R}\mathbf{v}_r[n] \qquad \text{and} \qquad \mathsf{E}\big[e^2[n]\big] \approx \frac{1}{R}\sum_{r=1}^{R}e_r^2[n]$$

where $\mathbf{v}_r[n]$ and $e_r[n]$ are the results obtained for a realization r of the input signal and reference signal.
Perform the following tasks using your implementations of the stochastic and deterministic gradient algorithms from the previous task. Until stated otherwise assume that you know the true filter order, i.e., use $N = \dim(\mathbf{c}[n]) = \dim(\mathbf{h})$.

(a) (3 points) Investigate the effect of the step-size parameter μ when using the standard LMS algorithm (3) and the standard gradient descent algorithm (1) with the analytical values for $\mathbf{R}_{xx}$ and $\mathbf{p}$. Use $\mu = \{0.0001, 0.001, 0.01, 1\}$. Are the algorithms stable in all these cases?

(b) (3 points) To investigate the effect of the number of samples used to compute the sample correlations, now compare the behavior of gradient search algorithm (1) using sample estimates

with values using signal windows of size $P = \{10, 100, 1000\}$ and compare it to the results using the analytical values for the statistics. Use a stepsize $\mu = 0.0005$. What do you observe?

(c) (2 points) Now try to drastically increase the adaptive filter order, using $N = \{10, 100\}$ coefficients and the standard LMS algorithm (3) and the standard gradient descent algorithm using the analytical solutions for $\mathbf{R}_{xx}$ and $\mathbf{p}$ with a step size $\mu = 0.001$. What do you observe in terms the resulting error? Use a random initialization for $\mathbf{c}[0]$.

(d) (2 points) How does the MSE of the different algorithms behave *after* convergence in all previous tasks? How do the different algorithms behave for different signal realizations?

(e) Bonus: (4 points) Modify your functions `gd_algorithm()` and `lms_algorithm()` such that they implement the coefficient-leakage variants of the corresponding algorithms, i.e., (2) and (4) and show where the algorithms converge to. Compare the results to your analytical solutions.

Analytical Problem 2.6 (5 Points)—Bonus: Wiener-Hopf Solution

In most scenarios it is a strong limitation when only assuming real coefficients. The mean-squared error cost function for general signals (real or complex) is given as

$$J_{\text{MSE}}(\mathbf{c}) = \mathsf{E}\big[|e[n]|^2\big] = \mathsf{E}\big[e[n]e^*[n]\big]$$

with error $e[n] = y[n] - d[n]$, filter output $y[n] = \mathbf{c}^{\mathsf{H}}\mathbf{x}[n]$ and noisy reference signal $d[n] = \mathbf{h}^{\mathsf{H}}\mathbf{x}[n] + w[n]$, corresponding to a noisy system identification scenario. Assume that the filter input signal $x[n]$ and noise $w[n]$ are statistically independent.

(a) (4 points) Derive the Wiener-Hopf solution for complex filter coefficients $\mathbf{c} \in \mathbb{C}^N$ and a complex filter input signal $x[n] \in \mathbb{C}$. Use the definition of the partial derivative w.r.t. a complex coefficient $c_k = a_k + jb_k$ as

$$\frac{\partial}{\partial c_k} = \frac{\partial}{\partial a_k} + j\frac{\partial}{\partial b_k}$$

and the corresponding gradient w.r.t. to the filter coefficient vector $\mathbf{c}$ defined as

$$\nabla_{\mathbf{c}} = \begin{bmatrix} \frac{\partial}{\partial c_1} \\ \vdots \\ \frac{\partial}{\partial c_k} \\ \vdots \\ \frac{\partial}{\partial c_N} \end{bmatrix}$$

and carefully specify the corresponding vector and matrices. Clearly indicate where you use transpose $(\cdot)^{\mathsf{T}}$ and where a conjugate transpose $(\cdot)^{\mathsf{H}}$.

(b) (1 points) Write down the resulting expression for the resulting minimum MSE $J_{\text{min}} = J_{\text{MSE}}(\mathbf{c}_{\text{MSE}})$. Simplify as much as possible.

Appendix 1—Data Description (`data.mat`)

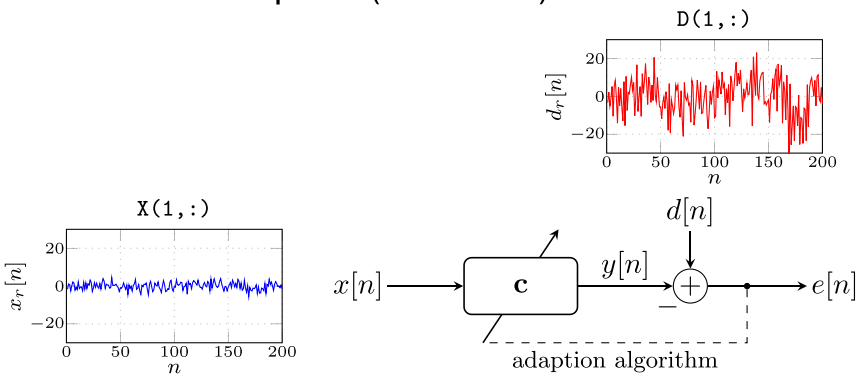


Figure 2: General system identification

The provided data used throughout this assignment is provided in the file `data.mat`. The file contains multiple realizations of a filter input signal $x[n]$, a reference signal $d[n]$, stored in the data matrices `X` and `D` respectively, and the true filter coefficients $\mathtt{h} \in \mathbb{R}^N$ with $N = 6$. It further contains the standard deviations $\mathtt{sigma\_u} = \sigma_u$ of $u[n]$ and $\mathtt{sigma\_w} = \sigma_w$ of $w[n]$.
To allow an investigation of the filter behavior in a somewhat statistical sense, $R = 100$ realizations of the signals $x[n]$ and $d[n]$, both with length $L = 10000$ are provided: each row of `X` and `D` contains a new realization, resulting in signal matrices with equal size $\mathtt{X} \in \mathbb{R}^{R\times L}$ and $\mathtt{D} \in \mathbb{R}^{R\times L}$. The underlying problem is a general noisy system identification, as shown in Fig. 1 or Fig. 2, where the latter also visualizes the signals available at the filter for performing the identification task. Depending on what type of adaption algorithm is used for finding the optimal coefficients, the signal statistics might be needed, the estimation of which is treated in Problem 2.2. For comparison, the true statistics used for generating the data are given here.

Filter Input Signal The input signal $x[n]$ is a superposition of a sinusoidal with a uniformly distributed random phase $\varphi \sim \mathcal{U}(-\pi, \pi]$, frequency $\theta = \pi/2$ and amplitude $A = 1$ and zero mean white Gaussian noise with variance $\sigma_u^2 = 4$, i.e., $u[n] \sim \mathcal{N}(0, \sigma_u^2)$. The (stationary) random signal $x[n]$ can thus be written as

$$x[n] = A\sin(\theta n + \varphi) + u[n].$$

with autocorrelation function given as

$$r_{xx}[k] = \frac{A^2}{2}\cos(\theta k) + \sigma_u^2\delta[k]$$

Reference Signal The reference signal $d[n]$ is corrupted by zero-mean additive white Gaussian noise $w[n] \sim \mathcal{N}(0, \sigma_w^2)$ with variance $\sigma_w^2 = 0.01$ after passing through the unknown system $\mathbf{h}$ and can thus be written as

$$d[n] = \mathbf{h}^\mathsf{T}\mathbf{x}[n] + w[n].$$

Unknown System The true coefficient vector $\mathbf{h}$ of the system which should be identified is given as

$$\mathbf{h} = [2,\ -0.5,\ 4,\ -2,\ -1,\ 2]^\mathsf{T}$$

which is only needed for the performance evaluation of the algorithms, i.e., to compute the misalignment vectors $\mathbf{v}[n] = \mathbf{c}[n] - \mathbf{h}$ at each iteration step.