

AS

Signal Processing and Speech Communication Lab.
Graz University of Technology

Adaptive Systems—Homework Assignment 1

v1.0

Name(s)

*Reindl Hannes 01532129
Samer Philip 01634718*

Matr.No(s).

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 **2019/12/6**. 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 **2019/12/6**.

All filenames consist of the assignment number and your matriculation number(s) such as Assignment1_MatrNo1_MatrNo2.*, for example,

Problem solutions:

Assignment1_01312345_01312346.pdf

OCTAVE/MATLAB files:

Assignment1_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.

Analytical Problem 1.1 (8 Points)—Theory

Linear Algebra

With the step from the continuous to the discrete domain, signal processing in general profits from many results that can be taken from linear algebra and matrix theory.

(a) (2 points) Perform the following tasks dealing with vectors and matrices in general.

- (i) Assume that you have a matrix \mathbf{A} whose columns \mathbf{a}_i contain some signals. Compute a linear combination of the columns, i.e., the signals, using a single vector \mathbf{x} . Assume that \mathbf{A} is a $(N \times 4)$ matrix. Carefully state the size of \mathbf{x} .
- (ii) Find a single matrix \mathbf{B} that allows performing the following tasks without changing the size of the vector \mathbf{a} and the matrix \mathbf{D} . Be careful regarding the dimensions of \mathbf{B} .
 - multiplication of the elements of a column vector $\mathbf{a} = [a_1, a_2, a_3]^\top$ by different constants b_1, b_2 and b_3 .
 - multiplication of the columns of an arbitrary $(N \times 3)$ matrix $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3]$ by different constants b_1, b_2 and b_3 .

Hint: You can multiply matrices from the left or from the right depending on the dimensions.

- (iii) Show that $\det(a\mathbf{A}) = a^N \det(\mathbf{A})$ assuming that \mathbf{A} is an $N \times N$ matrix and a is some real-valued scalar constant. It is enough to show this for an exemplary matrix \mathbf{A} of size (2×2) .

Mean-Square Error Filters

The Wiener filter yields the optimum solution in the *mean-square error* sense, while requiring stationarity of the involved signals as well as knowledge of their correlation functions. The derivation is based on the gradient of the *mean-square error* (MSE) cost function $J_{\text{MSE}}(\mathbf{c}) = \mathbb{E}[|e[n]|^2]$.

(b) (1.5 points) An alternative approach to derive the Wiener filter is by the so-called *principle of orthogonality*, which states that the error $e[n]$ is orthogonal to each filter input sample $x[n-k]$:

$$\mathbb{E}[e[n]\mathbf{x}[n]] = \mathbf{0}$$

- (iv) Derive the orthogonality principle from the MSE cost function $J_{\text{MSE}} = \mathbb{E}[|e[n]|^2]$
- (v) Derive the Wiener–Hopf solution from (iv)
- (vi) What property does the autocorrelation matrix need to fulfill to allow using the Wiener–Hopf solution?

(c) (4.5 points) Assume that the input signal $x[n]$ of your adaptive filter is of the form

$$x[n] = \alpha e^{j(\theta_0 n + \varphi)} + w[n]$$

where $\alpha \in \mathbb{R}$ is a constant amplitude, $\theta_0 \in \mathbb{R}$ a constant frequency, $\varphi \sim \mathcal{U}(-\pi, \pi]$ a uniformly distributed random phase and $w[n]$ zero-mean Gaussian noise with non-zero variance σ_w^2 . Assume that $w[n]$ and φ are independent. Compute the autocorrelation sequence $r_{xx}[n, k] = \mathbb{E}[x[n+k]x^*[n]]$ for the noisy signal and perform the following tasks.

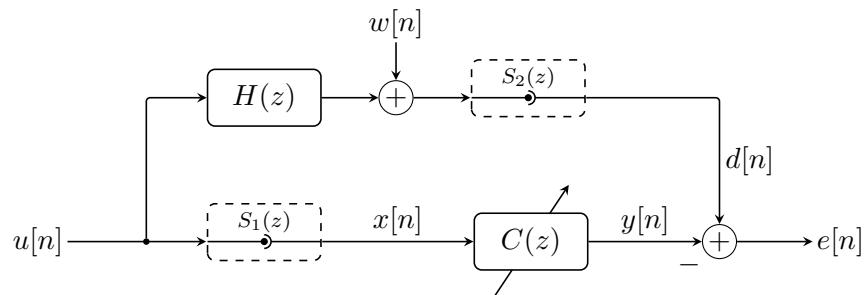
- (vii) Show that the signal $x[n]$ is wide sense stationary (WSS).
- (viii) Compute the autocorrelation matrix \mathbf{R}_{xx} . Is it always invertible? What happens for $\sigma_w^2 \rightarrow 0$?
- (ix) When dealing with stochastic signals, we can only analyze the frequency content of the signal by computing the Fourier transform (the DTFT) of the corresponding autocorrelation function, then called the power spectral density (PSD)¹ $S_{xx}(\theta)$. Compute and sketch the PSD, defined as

$$S_{xx}(\theta) = \sum_{k=-\infty}^{\infty} r_{xx}[k]e^{-j\theta k}.$$

Again show what happens for $\sigma_w^2 \rightarrow 0$. Can you use the PSD to find out whether the autocorrelation matrix is invertible?

Analytical Problem 1.2 (15 Points)—System Identification with Realistic Sensors

Consider the following system identification problem:



Assume that $u[n]$ and $w[n]$ are independent, jointly stationary, white noise processes with variances $\sigma_u^2 = 1$ and $\sigma_w^2 = \frac{1}{4}$, respectively. We want to identify the unknown LTI system $H(z) = 1 + \frac{1}{4}z^{-1}$ using a second-order adaptive filter $C(z)$ (i.e., $N = 3$ and $\mathbf{c} = [c_0, c_1, c_2]^T$). Unfortunately, as often the case in reality, we cannot assume that our adaptive filter has direct access to $x[n]$ or $d[n]$, but only to filtered versions of these. This can be due to non-ideal sensor responses $S_1(z)$ and $S_2(z)$ as indicated in the flowgraph, which are unknown to the user. This problem should help understanding the problems sensor imperfections can cause.

For all the following cases, determine

1. the values of the auto-correlation sequence $r_{xx}[k]$ and the autocorrelation matrix \mathbf{R}_{xx} ,
2. the values of the cross-correlation vector $\mathbf{p} = \mathbb{E}[d[n]\mathbf{x}[n]]$,
3. the optimal coefficient vector \mathbf{c}_{opt} in the sense of a minimum mean-squared error, i.e. $\mathbf{c}_{\text{opt}} = \underset{\mathbf{c}}{\operatorname{argmin}} J_{\text{MSE}}(\mathbf{c})$ with $J_{\text{MSE}}(\mathbf{c}) = \mathbb{E}[|e[n]|^2]$, and
4. the minimum mean-squared error $J_{\text{min}} = J_{\text{MSE}}(\mathbf{c}_{\text{opt}})$.

¹You can find additional information in Ch. 1.3 in [SM⁺05]

Always answer if you can identify the system correctly, and if so, why this is the case. If not, explain why system identification fails. Until stated otherwise assume that you do not know $S_1(z)$ and $S_2(z)$.

Hint: In some cases it may be helpful to redraw the signal model by taking the linearity of the systems into account.

(a) (3.5 points) Let $S_1(z) = S_2(z) = 1$.

(b) (3.5 points) Let $S_1(z) = S_2(z) = 1 + 0.5z^{-1}$.

(c) (3.5 points) Let $S_1(z) = 1$ and $S_2(z) = z^{-1}$.

(d) (3.5 points) Let $S_1(z) = 1$ and $S_2(z) = 1 + z^{-2}$.

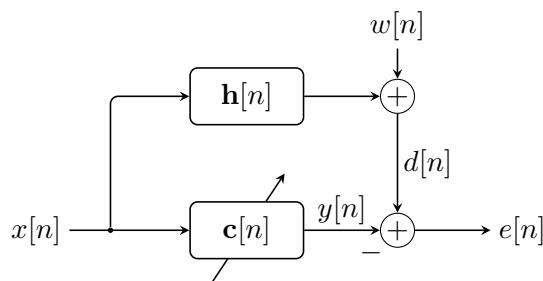
(e) (1 point) What happens when the sensor responses are known? Can the system be identified correctly? Can the cascade in (d) be identified correctly? You do not need to compute anything, just give a short explanation.

OCTAVE/MATLAB Problem 1.3 (12 Points)—Least–Squares Tracking of a Time–Varying System

Consider the system identification problem for a time–varying impulse response. The desired signal follows

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

where $\mathbf{h}[n]$ indicates the impulse response, now dependent on the current time index n .



(a) (5 points) Write an Octave/Matlab function that computes the *least-squares* optimum filter coefficients $\mathbf{c}_{\text{LS}}[n] = \text{argmin}_{\mathbf{c}} J_{\text{LS}}(\mathbf{c}, n)$ according to the cost function

$$J_{\text{LS}}(\mathbf{c}, n) = \sum_{k=0}^n |e[k]|^2.$$

You should hand in (at least) two Octave/Matlab file: One file that creates the signals, calls your function and performs all required plots and one file containing your function itself. Implement your function according to the following specifications:

```

function c = ls_filter( x, d, N )
% x ... input signal vector
% d ... desired output signal vector (of same length as x)
% N ... number of filter coefficients
  
```

(b) (1 point) For the ‘unknown’ system, implement a filter with the following time-varying 3-sample impulse response:

$$\mathbf{h}[n] = \begin{bmatrix} -1 \\ 2 - 0.97^n \\ 0.3 \cdot \cos(\theta n) \end{bmatrix}$$

where $\theta = \frac{3\pi}{1000}$. Visualize the time-varying impulse response $\mathbf{h}[n]$ for $n = 0 \dots 999$.

(c) (4 points) Assume now that the system is noise-free, i.e. $w[n] = 0$. Generate 1000 samples (for $n = 0 \dots 999$) of an input signal $x[n]$ drawn from a stationary white noise process with zero mean and variance $\sigma_x^2 = 1$. Compute the output $d[n]$ of the system, under the condition that initially all delay elements contain zeros, i.e., $x[n] = 0$ for $n < 0$.

The adaptive filter should have 3 coefficients ($N = 3$). By calling your function `ls_filter` with length- M segments of both $x[n]$ and $d[n]$, the coefficients of the adaptive filter $\mathbf{c}[n]$ for $n = 0 \dots 999$ can be computed. Note that we thus obtain $\mathbf{c}[n] = \operatorname{argmin}_{\mathbf{c}} J(\mathbf{c}, n)$ where the cost function becomes $J(\mathbf{c}, n) = \sum_{k=n-M+1}^n |e[k]|^2$. For segment lengths of $M = \{20, 50\}$, create plots comparing the time evolution elements of the coefficient vector $\mathbf{c}[n]$ to the actual impulse response $\mathbf{h}[n]$. Compare and discuss your results and explain the effect of different M .

Hint: You can display the results in any (understandable) way you like, for example using `plot()`, `subplot()` or `waterfall()`. Make sure to label the axis and indicate the displayed data with appropriate legends.

(d) (2 points) Repeat task **(c)** for $w[n]$ being a zero-mean white noise process with variance $\sigma_w^2 = 0.02$ and compare the obtained results.

OCTAVE/MATLAB Problem 1.4 (5 Points)—Bonus: Weighted Least-Squares

This task again deals with least-squares linear filtering, now trying to achieve better performance for time varying systems. Consider that observations of $x[n]$ and $d[n]$ for $n = 0 \dots M$ are given.

(a) (2 points) Derive the optimum filter coefficients \mathbf{c} in the sense of a weighted least-squares, i.e., find $\mathbf{c}_{wLS}[n] = \operatorname{argmin}_{\mathbf{c}} J_{wLS}(\mathbf{c}, n)$ for the weighted least-squares cost function

$$J_{wLS}(\mathbf{c}, n) = \sum_{k=n-M+1}^n g[n-k] \cdot |e[k]|^2.$$

(b) (1 point) Use your Octave/Matlab function `ls_filter(x, d, N)` from Problem 1.3 and extend it to implement the cost function $J_{wLS}(\mathbf{c}, n)$ for the weighted LS algorithm given above, naming the new function `ls_filter_weighted(x, d, N)`. If you want you can also add the weighting as an additional input argument.

(c) (2 points) Evaluate and compare the performance using the weighting function

$$g[m] = \lambda^{-m} \quad \text{for } m \geq 0$$

and using the time-varying system $\mathbf{h}[n]$ from Problem 1.3, but with $\theta = \frac{3\pi}{100}$ and $M = 50$. Try to find a suitable value for λ that accomplishes the tracking.

References

- [SM⁺05] Petre Stoica, Randolph L. Moses, et al. Spectral Analysis of Signals. 2005.

Adaptive Systems

EXERCISE 1

PROBLEM 1.1

some signals

$$i) \underline{A} = [\underline{a}_1, \underline{a}_2, \dots, \underline{a}_n]$$

$$\underline{a}_i = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad n$$

note: $\text{size}(\underline{A}) = (N \times 1)$

$$\underline{A} = [\underline{a}_1, \underline{a}_2, \underline{a}_3, \underline{a}_4]$$

$$\underline{A} \cdot \underline{x} \stackrel{!}{=} x_1 \cdot \underline{a}_1 + x_2 \cdot \underline{a}_2 + x_3 \cdot \underline{a}_3 + x_4 \cdot \underline{a}_4 \Rightarrow \underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad \underline{\text{size}(\underline{x}) = (4 \times 1)}$$

$$(N \times 4) \cdot (4 \times 1) = (N \times 1)$$

ii)

$$\underline{c} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\text{we want: } \begin{bmatrix} b_1 \cdot c_1 \\ b_2 \cdot c_2 \\ b_3 \cdot c_3 \end{bmatrix} = \underline{B} \cdot \underline{c} = \begin{bmatrix} b_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & b_3 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \Rightarrow \underline{B} \neq \text{eye}(3) \quad (3 \times 3)$$

$$(3 \times 3) \cdot (3 \times 1) = (3 \times 1)$$

$$\underline{B} = \begin{bmatrix} b_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & b_3 \end{bmatrix}$$

$$\underline{D} = [\underline{d}_1 \ \underline{d}_2 \ \underline{d}_3]$$

$$(N \times 3)$$

$$\text{we want: } \begin{bmatrix} b_1 \cdot \underline{d}_1 & b_2 \cdot \underline{d}_2 & b_3 \cdot \underline{d}_3 \end{bmatrix} \stackrel{!}{=} \underline{D} \cdot \underline{B} = [\underline{d}_1 \ \underline{d}_2 \ \underline{d}_3] \cdot \begin{bmatrix} b_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & b_3 \end{bmatrix}$$

$$= \underline{[b_1 \cdot \underline{d}_1 \quad b_2 \cdot \underline{d}_2 \quad b_3 \cdot \underline{d}_3]}$$

$$iii) \text{ point: } \det(\alpha \cdot \underline{A}) = \alpha^N \cdot \det(\underline{A}) ; \quad \# \text{ size}(\underline{A}) = (N \times N)$$

$$\text{e.g. } N=2: \det(\alpha \cdot \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix}) = \det \left(\begin{bmatrix} \alpha \cdot b_1 & \alpha \cdot b_2 \\ \alpha \cdot b_3 & \alpha \cdot b_4 \end{bmatrix} \right) = \alpha^2 \cdot b_1 \cdot b_4 - \alpha^2 \cdot b_2 \cdot b_3 =$$

$$= \alpha^2 \cdot (b_1 \cdot b_4 - b_2 \cdot b_3) = \underline{\alpha^2 \cdot \det(\underline{A})} \quad \checkmark$$

$$\begin{aligned}
 (IV) \quad J_{\text{MSE}} &= E\{|e[n]|^2\} = E\{e[n] \cdot e[n]^*\} = \\
 &= E\{(d[n] - y[n]) \cdot (d[n] - y[n])^*\} // (a-b)^* = a^* - b^* \\
 &= E\{|d[n]|^2 - d[n] \cdot y[n]^* - y[n] \cdot d[n]^* + |y[n]|^2\} \\
 &= E\{|d[n]|^2 - d[n] \cdot \left(\sum_{k=0}^{n-1} x[n-k] \cdot c_k[n]\right)^* - d[n]^* \cdot \left(\sum_{k=0}^{n-1} x[n-k] \cdot c_k[n]\right) + \left|\sum_{k=0}^{n-1} x[n-k] \cdot c_k[n]\right|^2\} // |d| = |d|^2 \\
 &\xrightarrow{\text{absolute}} \text{assuming: } \frac{d}{d} (f_n)^* = (\frac{d}{d} f_n)^* // \text{probably wrong, but I don't know what else I could do} \quad |(1-1j)| = \sqrt{2}
 \end{aligned}$$

∇_E For Minimum: $\nabla_E J_{\text{MSE}} = 0 ; \nabla_E = \frac{\partial}{\partial c_k}$

$$\begin{aligned}
 \nabla_E J_{\text{MSE}} &\stackrel{!}{=} 0 = \nabla_E E\{ \dots \} = \\
 &= E\{ \nabla_E (|d[n]|^2) - \nabla_E (d[n] \cdot \left(\sum_{k=0}^{n-1} x[n-k] \cdot c_k[n]\right)^*) - \nabla_E (d[n]^* \cdot \left(\sum_{k=0}^{n-1} x[n-k] \cdot c_k[n]\right)) + \nabla_E \left| \sum_{k=0}^{n-1} x[n-k] \cdot c_k[n] \right|^2 \} \\
 &= E\{ 0 - x[n-1] \cdot d[n] - d[n]^* \cdot x[n-1] + 2 \cdot \left(\sum_{k=0}^{n-1} x[n-k] \cdot c_k[n] \cdot x[n-k] \right) \}
 \end{aligned}$$

Assuming: $e[n] \in \mathbb{R} \Rightarrow e[n] = e[n]^*; x[n] = x[n]^*; d[n] = d[n]^* // \text{no clear how to solve with the complex conj.(-) Operator}$

$$= E\{-x[n-1] \cdot [c[n] + d[n]] + 2 \cdot y[n] \cdot x[n-1]\}$$

$$= E\{-2 \cdot x[n-1] \cdot [d[n] - y[n]]\}$$

$$0 = -2 \cdot E\{e[n] \cdot x[n-1]\}$$

$$\underline{0} = E\{e[n] \cdot x[n-1]\} // \text{Principle of orthogonality}$$

$x[n]$

$$\begin{aligned}
 (V) \quad 0 &= E\{e[n] \cdot x[n-1]\} = E\{e[n] \cdot \underline{x[n]}\} \\
 &= E\{(d[n] - \underline{c}^T \cdot \underline{x[n]}) \cdot \underline{x[n]}\} \\
 &= E\{\underline{x[n]} \cdot (d[n] - \underline{c}^T \cdot \underline{x[n]})\} \\
 &= E\{\underline{x[n]} \cdot d[n] - \underline{x[n]} \cdot \underline{c}^T \cdot \underline{x[n]}\} \\
 &= E\{\underline{x[n]} \cdot d[n]\} - E\{\underline{x[n]} \cdot \underline{c}^T\}
 \end{aligned}$$

$$0 = p - R_{xx} \cdot \underline{c}$$

$$R_{xx} \cdot \underline{c}^* = p$$

$$\underline{c} = R_{xx}^{-1} \cdot p // \text{Wiener Hopf}$$

(VI) R_{xx} needs to be invertible $\Rightarrow R_{xx}$ semi positive definite or $\det(R_{xx}) > 0$

c)

$$x[n] = a \cdot e^{j(\theta_0 \cdot n + \varphi)} + w[n]$$

 $a, \theta_0 \in \mathbb{R}$

(vii)

$$\varphi \sim U(-\pi, \pi)$$

variance

w[n] ... zero-mean Gaussian noise with σ_w^2

$$\rightarrow E[w[n]] = 0$$

$$r_{xx}[n+k] = E\{x[n+k] \cdot x^*[n]\}$$

$$a^2 = a$$

w[n]s and φ are independent $\rightarrow E[w[n] \cdot \varphi] = E[w[n]]$

$$E[\varphi] = 0$$

$$= E\{(a \cdot e^{j(\theta_0 \cdot (n+k) + \varphi)}) + w[n+k]\} \cdot (a \cdot e^{-j(\theta_0 \cdot (n) + \varphi)} + w[n]) \quad / (a+b)^2 = a^2 + b^2$$

$$= E\left\{ a^2 \cdot e^{j\theta_0 \cdot k} + a \cdot e^{j(\theta_0 \cdot (n+k) + \varphi)} \cdot w[n] + a \cdot e^{-j(\theta_0 \cdot n + \varphi)} \cdot w[n+k] + w[n+k] \cdot w[n] \right\} \quad / a^2 \cdot \delta[k]$$

$$= a^2 \cdot e^{j\theta_0 \cdot k} + a \cdot e^{j\theta_0 \cdot (n+k)} \cdot E\{e^{j\varphi} \cdot w[n]\} + a \cdot e^{-j\theta_0 \cdot n} \cdot E\{e^{-j\varphi} \cdot w[n+k]\} + \sigma_w^2 \cdot \delta[k]$$

$$= a^2 \cdot e^{j\theta_0 \cdot k} + a \cdot e^{j\theta_0 \cdot (n+k)} \cdot E\{e^{j\varphi}\} \cdot E\{w[n]\} + a \cdot e^{-j\theta_0 \cdot n} \cdot E\{e^{-j\varphi}\} \cdot E\{w[n+k]\} + \sigma_w^2 \cdot \delta[k]$$

$$r_{xx}[k] = a^2 \cdot e^{j\theta_0 \cdot k} + \sigma_w^2 \cdot \delta[k] \quad / \text{depends only on time shift } k$$

↳ Periodic delta

vii) for WSS: first and second order moment are not dependent on time n

$$1) m_x[n] = E\{x[n]\} = m_x$$

$$2) r_{xx}[n+k] = E\{x[n+k] \cdot x^*[n]\} = r_{xx}[k] \quad \checkmark$$

$$1) m_x[n] = E\{a \cdot e^{j(\theta_0 \cdot n + \varphi)} + w[n]\}$$

$$= a \cdot e^{j\theta_0 \cdot n} \cdot E\{e^{j\varphi}\} + E\{w[n]\}$$

$$= a \cdot e^{j\theta_0 \cdot n} \cdot E\{e^{j\varphi}\} \quad / \text{depends on } n \rightarrow \text{NOT WSS}$$

Q?

NRs

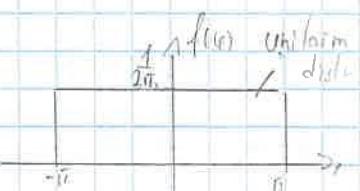
$$E[g(x)] = \int_{-\infty}^{\infty} g(x) \cdot f(x) dx$$

↳ probability dist.: uniform func.

$$g(x) \sim U(-\pi, \pi)$$

$$E[e^{j\varphi}] = \int_{-\pi}^{\pi} e^{j\varphi} \cdot f(\varphi) d\varphi$$

$$f(\varphi) =$$



$$= \int_{-\pi}^{\pi} e^{j\varphi} \cdot \frac{1}{2\pi} d\varphi = \left[\frac{1}{j} \cdot e^{j\varphi} \cdot \frac{1}{2\pi} \right]_{-\pi}^{\pi}$$

$$= \frac{1}{2\pi j} \cdot [e^{\pi} - e^{-\pi}] = \frac{1}{2\pi} \cdot [-1 - (-1)] = \underline{0}$$

$$\Rightarrow m_x[n] = a \cdot e^{j\theta_0 \cdot n} \cdot 0 = \underline{0} \Rightarrow \text{WSS}$$

$$f(\varphi) = \begin{cases} \frac{1}{2\pi} & -\pi \leq \varphi \leq \pi \\ 0 & \text{else.} \end{cases}$$

2

viii) $R_{xx} = ?$

hermitian \Leftrightarrow complex conjugate transpose: $X^H = (X)^*$

$$R_{xx} = E \left\{ X[n] \cdot X[n]^H \right\} = \begin{bmatrix} R_{xx}[0] & R_{xx}[1] & \cdots & R_{xx}[N-1] \\ R_{xx}[1] & R_{xx}[0] & & \\ \vdots & & \ddots & \\ R_{xx}[N-1] & \cdots & \cdots & R_{xx}[0] \end{bmatrix}$$

$$R_{xx}[n] = \alpha^2 \cdot e^{j\theta_n K} + b_w^2 \cdot \delta[n]$$

$N=1$:

$$R_{xx} = R_{xx}[0] = \alpha^2 \cdot e^0 + b_w^2 = \underline{\alpha^2 + b_w^2} \quad // \text{invertible for } |\alpha| \text{ OR } |b_w| > 0$$

$$\underline{N=2}: \quad R_{xx} = \begin{bmatrix} R_{xx}[0] & R_{xx}[1] \\ R_{xx}[1] & R_{xx}[0] \end{bmatrix} = \begin{bmatrix} \alpha^2 + b_w^2 & \alpha^2 \cdot e^{j\theta_1} \\ \alpha^2 \cdot e^{j\theta_0} & \alpha^2 + b_w^2 \end{bmatrix}$$

$$\begin{aligned} \det(R_{xx}) &= (\alpha^2 + b_w^2)^2 - 4 \cdot \alpha^2 \cdot e^{j\theta_0} \cdot \alpha^2 \cdot e^{j\theta_1} = \cancel{\alpha^4} + 2\alpha^2 \cdot b_w^2 + b_w^4 - \cancel{\alpha^4} = \\ &= 2\alpha^2 \cdot b_w^2 + b_w^4 = \underline{b_w^2 \cdot [2\alpha^2 + b_w^2]} \end{aligned}$$

$$\underline{N=3}: \quad R_{xx} = \begin{bmatrix} R_{xx}[0] & R_{xx}[1] & R_{xx}[2] \\ R_{xx}[1] & R_{xx}[0] & R_{xx}[1] \\ R_{xx}[2] & R_{xx}[1] & R_{xx}[0] \end{bmatrix} = \begin{bmatrix} \alpha^2 + b_w^2 & \alpha^2 \cdot e^{j\theta_1} & \alpha^2 \cdot e^{j\theta_2} \\ \alpha^2 \cdot e^{j\theta_0} & \alpha^2 + b_w^2 & \alpha^2 \cdot e^{j\theta_1} \\ \alpha^2 \cdot e^{j\theta_0} & \alpha^2 \cdot e^{j\theta_1} & \alpha^2 + b_w^2 \end{bmatrix} \quad // \text{invertible for } |b_w| > 0$$

$$\begin{aligned} \text{Sarrus: } \det(R_{xx}) &= (\alpha^2 + b_w^2)^3 + \alpha^2 \cdot e^{j\theta_0} \cdot \alpha^2 \cdot e^{j\theta_1} \cdot \alpha^2 \cdot e^{j\theta_2} + \alpha^2 \cdot e^{j\theta_1} \cdot \alpha^2 \cdot e^{j\theta_2} \cdot \alpha^2 \cdot e^{j\theta_0} \\ &\quad - \alpha^2 \cdot e^{j\theta_0} \cdot (\alpha^2 + b_w^2) \cdot \alpha^2 \cdot e^{j\theta_1} - \alpha^2 \cdot e^{j\theta_1} \cdot \alpha^2 \cdot e^{j\theta_2} \cdot (\alpha^2 + b_w^2) - (\alpha^2 + b_w^2) \cdot \alpha^2 \cdot e^{j\theta_0} \end{aligned}$$

$$\begin{aligned} &= (\alpha^2 + b_w^2)^3 + \alpha^6 \cdot 2 - [3\alpha^4] \cdot (\alpha^2 + b_w^2)^2 \\ &= \cancel{\alpha^6} + \cancel{3\alpha^4} \cdot \cancel{b_w^2} + 3\alpha^2 \cdot b_w^4 + b_w^6 + 2 \cdot \cancel{\alpha^6} - [\cancel{3\alpha^6} + \cancel{3\alpha^4} \cdot \cancel{b_w^2}] \\ &= \underline{3\alpha^2 \cdot b_w^4 + b_w^6} \end{aligned}$$

Aufgabe
1. Modul 2020

The $\det(R_{xx})$ for $N=4:10$ was calculated in MATLAB (example_1.m)

$$\underline{N=4}: \quad \det(R_{xx}) = 4\alpha^2 \cdot b_w^4 + b_w^8$$

$$\underline{N=5}: \quad \det(R_{xx}) = 5\alpha^2 \cdot b_w^8 + b_w^{10}$$

$$\text{generalize: } \det(R_{xx})_N = N \cdot \alpha^2 \cdot b_w^{2(N-1)} + b_w^{2N} \quad // \text{controlled for } N=6:10$$

$\rightarrow \det(R_{xx}) > 0 \text{ for } b_w \neq 0 \rightarrow \text{invertible for } b_w \neq 0$

ix)

$$S_{xx}(\theta) = \sum_{k=-\infty}^{\infty} r_{xx}[k] \cdot e^{j\theta \cdot k}$$

 $\rightarrow \theta \in [-\pi, \pi]$

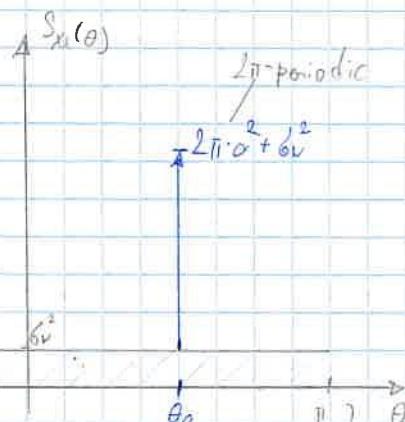
$$= \sum_{k=-\infty}^{\infty} (\alpha^2 e^{j\theta_0 \cdot k} + G_w^2 \cdot S_{xx}[k]) \cdot e^{-j\theta \cdot k}$$

$$= G_w^2 \cdot 1 \cdot e^{j\theta} + \sum_{k=-\infty}^{\infty} \alpha^2 \cdot e^{j\theta_0 \cdot k} \cdot e^{-j\theta \cdot k}$$

$$\left(= G_w^2 + \sum_{k=-\infty}^{\infty} \alpha^2 \cdot e^{-j(\theta - \theta_0) \cdot k} \right) \text{ // geometrische Reihe von rechts}$$

$$= G_w^2 + DTFT \left(\alpha^2 \cdot e^{j\theta_0 \cdot k} \right) \text{ // Formelsammlung}$$

$$= G_w^2 + 2\pi \cdot f_{1n} \cdot \delta(\theta - \theta_0) \cdot \alpha^2$$

 $\hookrightarrow 2\pi$ -periodic dirac delta

if x_{INJ} has a line spectrum with L lines, then B_{xx} is only invertible for $N \leq L$

\rightarrow due to the noise floor it is not a simple line spectrum (or has infinite many lines)

Therefore B_{xx} is invertible for $N \in \mathbb{N}$ as long as $G_w \neq 0$

if $G_w = 0$, then there would be only 1 line $\rightarrow N \leq 1$ ✓ it is true for calculated formulas

L

(4)

9) 1) Problem 12

$$r_{xx}[n, k] = E\{x[n+k] \cdot x^*[n]\} \quad \underline{h}^T = \begin{bmatrix} 1 & \frac{1}{4} & 0 \end{bmatrix}$$

$$x[n] = s_u * u[n] = u[n]$$

$$\begin{aligned} r_{xx}[n, k] &= E\{u[n+k] \cdot u^*[n]\} \\ &= s_u^2 \cdot s[k] = 1 \cdot s[k] \end{aligned}$$

$$\underline{R}_{xx} = \begin{pmatrix} r_{xx}[0] & r_{xx}[1] & r_{xx}[2] \\ r_{xx}[-1] & r_{xx}[0] & r_{xx}[1] \\ r_{xx}[-2] & r_{xx}[-1] & r_{xx}[0] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2) $\underline{p} = E\{d[n] \underline{x}[n]\}$

$$d[n] = s_u * (\underline{h}^T \underline{U}[n] + w[n]) = \underline{h}^T \underline{U}[n] + w[n]$$

$$\underline{p} = E\{(\underline{h}^T \underline{U}[n] + w[n]) \cdot \underline{U}[n]\}$$

$$\underline{p} = E\{\underline{h}^T \underline{U}[n] \cdot \underline{U}[n]\} + \underbrace{E\{w[n] \cdot \underline{U}[n]\}}$$

$$\underline{p} = E\left\{ \begin{bmatrix} 1 & \frac{1}{4} & 0 \end{bmatrix} \begin{bmatrix} \underline{U}[n] \\ \underline{U}[n-1] \\ \underline{U}[n-2] \end{bmatrix} \cdot \underline{U}[n] \right\}$$

$$= E\{(\underline{U}[n] + \frac{1}{4} \underline{U}[n-1]) \cdot \underline{U}[n]\}$$

$$= E\left\{ \begin{bmatrix} \underline{U}[n] \cdot \underline{U}[n] + \frac{1}{4} \underline{U}[n-1] \cdot \underline{U}[n] \\ \underline{U}[n] \cdot \underline{U}[n-1] + \frac{1}{4} \underline{U}[n-1] \cdot \underline{U}[n-1] \\ \underline{U}[n] \cdot \underline{U}[n-2] + \frac{1}{4} \underline{U}[n-1] \cdot \underline{U}[n-2] \end{bmatrix} \right\}$$

$$\underline{p} = \begin{bmatrix} \delta_u^2 \\ \frac{1}{4} \delta_u^2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{4} \\ 0 \end{bmatrix} = \underline{h}$$

3) $\underline{c} = \underline{R}_{xx}^{-1} \cdot \underline{p} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \frac{1}{4} \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{4} \\ 0 \end{bmatrix} = \underline{h} = \underline{c}_{opt}$

4) $S_{min} = S_{MSE}(\underline{c}_{opt}) \quad y[n] = \underline{c}^T \cdot \underline{U}[n]$

$$e[n] = d[n] - y[n] = (\underline{h}^T - \underline{c}^T) \underline{U}[n] - w[n]$$

$$S_{MSE} = E\{|e[n]|^2\} = E\{e[n] \cdot e^*[n]\}$$

$$\begin{aligned} S_{MSE}(\underline{c}_{opt}) &= E\{((\underline{h}^T - \underline{c}^T) \cdot \underline{U}[n] - w[n]) \cdot ((\underline{h}^T - \underline{c}^T) \cdot \underline{U}[n] - w[n])^*\} \\ &= E\{(-w[n]) \cdot (-w[n])^*\} = \frac{1}{4} = \delta_w^2 \end{aligned}$$

$$S_{min} = \frac{1}{4}$$

$$b) r_{xx}[n, k] = E\{x[n+k] x^*[n]\}$$

$$\begin{aligned} 1) x[n] &= \sum_{k=0}^{N-1} (s[n] + s[n-1] \cdot 0,5) \cdot u[n] \\ &= \sum_{k=0}^{N-1} (s[n] + s[n-1] \cdot 0,5) \cdot u[n+k] \\ &= u[n] + 0,5 u[n-1] \end{aligned}$$

$$\begin{aligned} r_{xx}[n, k] &= E\{ (u[n+k] + 0,5 u[n+k-1]) (u[n+k] + 0,5 u[n+k-1])^* \} \\ &= E\{ u[n+k] \cdot u^*[n+k] + u[n+k] \cdot u^*[n+k-1] \cdot 0,5 + 0,5 \cdot u[n+k-1] \cdot u^*[n+k-1] \\ &\quad + 0,5 u[n+k-1] \cdot 0,5 u^*[n+k-1] \} \\ &= s[k] + 0,5 s[k+1] + 0,5 s[k-1] + 0,25 s[k] \end{aligned}$$

$$r_{xx}[k] = 1,25 s[k] + 0,5 s[k+1] + 0,5 s[k-1]$$

$$R_{xx} = \begin{pmatrix} 1,25 & 0,5 & 0 \\ 0,5 & 1,25 & 0,5 \\ 0 & 0,5 & 1,25 \end{pmatrix}$$

$$2) d[n] = s_2 * (h^T \cdot u[n] + w[n])$$

$$\begin{aligned} &= \sum_{k=0}^{N-1} (s[n] + s[n-1]) \cdot (h^T \cdot u[n+k] + w[n+k]) \\ &= h^T u[n] + w[n] + h^T u[n-1] \cdot 0,5 + w[n-1] \cdot 0,5 \end{aligned}$$

$$\begin{aligned} p &= E\{d[n] \cdot x[n]\} = E\{(h^T u[n] + w[n] + h^T u[n-1] \cdot 0,5 + w[n-1] \cdot 0,5) \\ &\quad \cdot (u[n] + 0,5 \cdot u[n-1])\} \end{aligned}$$

$$\begin{aligned} p &= E\{h^T u[n] \cdot u[n] + h^T u[n] u[n-1] \cdot 0,5 + w[n] \cdot u[n] + w[n] u[n-1] \cdot 0,5 \\ &\quad + h^T u[n-1] \cdot 0,5 \cdot u[n] + h^T u[n-1] \cdot 0,5 \cdot 0,5 u[n-1] + w[n-1] \cdot u[n] + w[n-1] u[n-1]\} \end{aligned}$$

$$\begin{aligned} p &= E\{h^T u[n] u[n]\} + E\{h^T u[n] u[n-1]\} + 0 + 0 + E\{h^T u[n-1] \cdot 0,5 \cdot u[n]\} \\ &\quad + E\{h^T u[n-1] \cdot 0,25 \cdot u[n-1]\} + 0 + 0 \end{aligned}$$

$$\begin{aligned} &= E\{(u[n] + \frac{1}{4} u[n-1]) \cdot u[n]\} + 0,5 \cdot E\{(u[n] + \frac{1}{4} u[n-1]) \cdot u[n-1]\} \\ &\quad + 0,5 \cdot E\{(u[n-1] + \frac{1}{4} u[n-2]) \cdot u[n]\} + 0,25 \cdot E\{(u[n-1] + \frac{1}{4} u[n-2]) \cdot u[n-1]\} \end{aligned}$$

$$= \begin{bmatrix} 1 \\ \frac{1}{4} \\ 0 \end{bmatrix} + 0,5 \begin{bmatrix} \frac{1}{4} \\ 0 \\ 0 \end{bmatrix} + 0,5 \begin{bmatrix} 0 \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix} + 0,25 \begin{bmatrix} \frac{1}{4} \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1,375 \\ 0,1875 \\ 0,125 \end{bmatrix} = P = \begin{bmatrix} 11/18 \\ 13/16 \\ 11/18 \end{bmatrix}$$

$$3) \underline{c} = \underline{R}_{xx}^{-1} \cdot \underline{p} = \begin{bmatrix} \frac{84}{85} & -\frac{8}{77} & \frac{16}{85} \\ -\frac{8}{77} & \frac{20}{77} & -\frac{8}{77} \\ \frac{16}{85} & -\frac{8}{77} & \frac{84}{85} \end{bmatrix} \cdot \begin{bmatrix} 11/8 \\ 13/16 \\ 1/8 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/4 \\ 0 \end{bmatrix} = \underline{h}$$

$$4) S_{\min} = E\{\underline{e}[n] \cdot \underline{e}[n]^*\}$$

$$\underline{e}[n] = (\underline{h}^T - \underline{c}^T) \underline{u}[n] + (\underline{h}^T - \underline{c}^T) \underline{u}[n-1] \cdot 0,5 + \underline{w}[n] + \underline{w}[n-1] \cdot 0,5$$

0 0 if $\underline{c} = \underline{h}$

$$\begin{aligned} S_{\min} &= E\{(w[n] + 0,5 w[n-1]) \cdot (w[n] + 0,5 w[n-1])^*\} \\ &= E\{w[n]^2 + w[n] \cdot w[n-1] + 0,5 w[n-1] \cdot w[n] \\ &\quad + 0,25 w[n-1]^2\} \\ &= 0,25 s_w^2 + s_w^2 = \frac{1}{4} + \frac{1}{8/16} = \underline{\underline{\frac{5}{16}}} \end{aligned}$$

9) 1 is the same as 9)

$$r_{xx} = \{[k]\} \quad x[n] = \{[n]\}$$

$$\underline{N}_{XX} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2

$$d[n] = h^+ u[n-1] + w[n-1]$$

$$p = E \{ (\underline{h}^T \underline{u}[n-1] + w[n-1]) \cdot (\underline{u}[n]) \}$$

$$= E\{ \underline{h}^T \cdot \underline{u}[n-1] \cdot \underline{u}[n] + w[n-1] \cdot \underline{u}[n] \}$$

$$= E\{ (U[n-1] + \frac{1}{4}U[n-2]) \underline{U[n]} \} + E\{ U[n-1] \cdot \underline{U[n]} \}$$

$$= \begin{bmatrix} 0 \\ 1 \\ \frac{1}{4} \end{bmatrix}$$

$$3) \quad \underline{C} = \underline{R}_{xx}^{-1} \cdot \underline{p}_x = \begin{bmatrix} 0 \\ 1 \\ \frac{1}{2} \end{bmatrix}$$

$$4) y_{min} = E \{ [e[n] \cdot e^*[n]] \} \quad e[n] = h^T u[n-1] + w[n-1] + c^T u[n]$$

$$\begin{aligned} f_{\min} &= E\{(h^T \underline{U}_{[n-1]} + w_{[n-1]} - \underline{c}^T \underline{U}_{[n]}) \cdot (h^T \underline{U}_{[n-1]} + w_{[n-1]} - \underline{c}^T \underline{U}_{[n]})^*\} \\ &= E\{h^T \underline{U}_{[n-1]} h^T \underline{U}_{[n-1]} + \underbrace{h^T \underline{U}_{[n-1]} \cdot w_{[n-1]}}_{+ w_{[n-1]} \cdot h^T \underline{U}_{[n-1]} + w_{[n-1]} \cdot w_{[n-1]} + w_{[n-1]} \cdot \underline{c}^T \underline{U}_{[n]}} + \underline{c}^T \underline{U}_{[n]} h^T \underline{U}_{[n-1]} + \underline{c}^T \underline{U}_{[n]} \cdot w_{[n-1]} + \underline{c}^T \underline{U}_{[n]} \cdot \underline{c}^T \underline{U}_{[n]}\} \end{aligned}$$

$$= h^T \cdot E\{ u_{[n-1]} \cdot u_{[n-1]}^T \} h + h^T E\{ u_{[n-1]} u_{[n]}^T \} \leq$$

$$+ \delta_w^2 = \underline{c}^T E\{u_{Ln3} \cdot u_{Ln-13}\} \underline{h} + \underline{c}^T \underline{\alpha} E\{u_{Ln3} \cdot u_{Ln3}\} \underline{c}$$

$$= \underline{h}^T \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \underline{h} = \underline{h}^T \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \underline{c} + \frac{1}{4} + \underline{c}^T \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \underline{h} + \underline{c}^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \underline{c}$$

$$= 7 + \frac{1}{16} \neq 7 - \frac{1}{16} + \frac{1}{4} = 7 - \frac{1}{16} + 1 + \frac{1}{16} = \underline{\underline{\frac{1}{4}}} = 7_{\min}$$

d) 1) is the same as d) 1

$$r_{xx} \in \mathcal{F}[E] \quad x[E] = U[E]$$

$$\underline{R}_{XX} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2)

$$d[n] = \underline{h}^T \underline{u}[n] + w[n] + \underline{h}^T \cdot \underline{u}[n-2] + w[n-2]$$

$$P = E \{ (\underline{h}^T \underline{u}[n] + w[n] + \underline{h}^T \cdot \underline{u}[n-2] + w[n-2]) \cdot u[n] \}$$

$$= E\left\{ \left(U[n] + \frac{1}{4} U[n-1] \right) \cdot U[n] \right\} + E\left\{ \left(U[n-2] + \frac{1}{4} U[n-3] \right) \cdot U[n] \right\}$$

$$= \begin{bmatrix} 1 \\ \frac{1}{4} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{4} \\ 1 \end{bmatrix} = P$$

$$3) \underline{C} = R_{xx}^{-1} \cdot P = \begin{bmatrix} 1 \\ 1/4 \\ 1 \end{bmatrix}$$

$$4) \quad \beta_{\min} = \beta_{\text{MSE}} (\leq \text{opt}) \quad c[n] = h^T U[n] + w[n] + h^T U[n-2] + w[n-2] - c^T U[n]$$

$$\underline{Q} = \underline{h} - \underline{c} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$s_{min} = E\{e[n] \cdot e[n]\} = E[(a^T u[n] + w[n] + h^T u[n-2] + w[n-2])^2]$$

$$= E \{ \underline{\alpha}^T \underline{U}[n] \underline{\alpha}^T \underline{U}[n] + \underline{\alpha}^T \underline{U}[n] \underline{u}[n] + \underline{\alpha}^T \underline{u}[n] \underline{h}^T \underline{U}[n-1] + \underline{\alpha}^T \underline{u}[n] \}$$

+ 00000

$$= \underline{a}^T E\{\underline{u}[n]\underline{u}^T[n]\}\underline{a} + E\{\underline{a}^T\underline{u}[n]\underline{u}^T[n-2]\underline{h}\} + \underline{s_w}^2$$

$$+ \underline{h}^T E\{\underline{u}[n-2]\underline{u}^T[n]\}\underline{a} + \underline{h}^T E\{\underline{u}[n-1]\underline{u}^T[n-1]\}\underline{h} + \underline{s_w}^2$$

$$= \underline{q}^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \underline{q} + \underline{q}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \underline{h} + \underline{h}^T \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \underline{q} + \underline{h}^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \underline{h} + \frac{1}{2}$$

$$= \cancel{X} + \cancel{Y} - \cancel{X} + \cancel{Y} + \frac{1}{16} + \frac{1}{2} = \underline{\underline{-\frac{9}{16}}}$$

e)

- if the two sensor responses are the same, the system gets identified correctly
 - if the two sensor responses are the same
are the same
 - if the two sensor responses, but one has a delay (e.g. $S_2(z) = S_1(z) \cdot z^{-1}$)
The system won't be identified correctly, unless we introduce same delay
on our own (e.g. delay path through S_1 by z^{-1})
 - The system in d) cannot be identified correctly.

3 OCTAVE/MATLAB Problem 1.3

Least-Squares Tracking of a Time-Varying System

3.1 Task a)

For the first task a function called `ls_filter()` shall be implemented which computes the *least-squares* optimum filter coefficients $\mathbf{c}_{LS}[n] = \text{argmin}_{\mathbf{c}} J_{LS}(\mathbf{c}, n)$ according to the cost function

$$J_{LS}(\mathbf{c}, n) = \sum_{k=0}^n |e[k]|^2 \quad (1)$$

We know from the problem class:

$$\mathbf{c}_{LS} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{d} \quad (2)$$

The Matlab files can be found in the `.zip` file. The code is also appended to this PDF sheet (Section 3.5)

3.2 Task b)

For the second task the 'unknown' system filter coefficients shall be plotted. These are:

$$\mathbf{h}[n] = \begin{bmatrix} -1 \\ 2 - 0.97^n \\ 0.3 \cdot \cos(\theta n) \end{bmatrix} \quad (3)$$

where $\theta = \frac{3\pi}{1000}$ for $n \in [0, 999]$

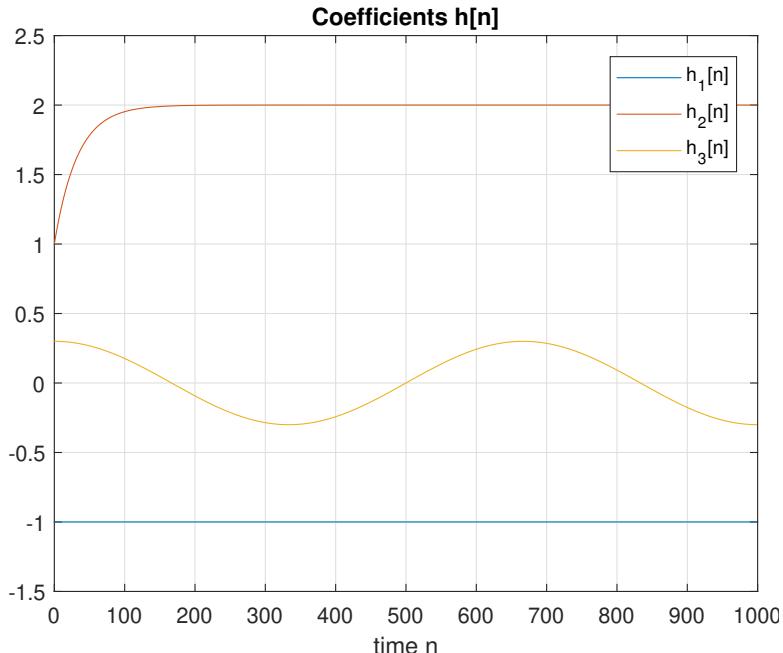


Figure 1: Coefficients for $\mathbf{h}[n]$

3.3 Task c)

In this task the coefficients were calculated using the *least-square* filter programmed in task a). The input signal $x[n]$ is a stationary white noise process with variance $\sigma_x^2 = 1$ (White noise processes always have a zero mean value). To compute the filter coefficients $\mathbf{c}[n]$ the error $e[n]$ has to be computed first. Therefore the values of $d[n]$ are needed and can be computed from the 'unknown' system $\mathbf{h}[n]$ (that is why we know the coefficients of $\mathbf{h}[n]$). In real life we would simply measure the data of $d[n]$). The noise $w[n] = 0$ this time and will be considered in task d).

The adaptive filter should have $N = 3$ coefficients (the filter order i.e. the amount of delay elements is $N - 1 = 2$).

$$\mathbf{c}[n] = \begin{bmatrix} c_1[n] \\ c_2[n] \\ c_3[n] \end{bmatrix} \quad (4)$$

The filter coefficients were computed by segmenting the input signal $x[n]$ and calling the function `ls_filter()`. This should be done for segment lengths $M = \{20, 50\}$.

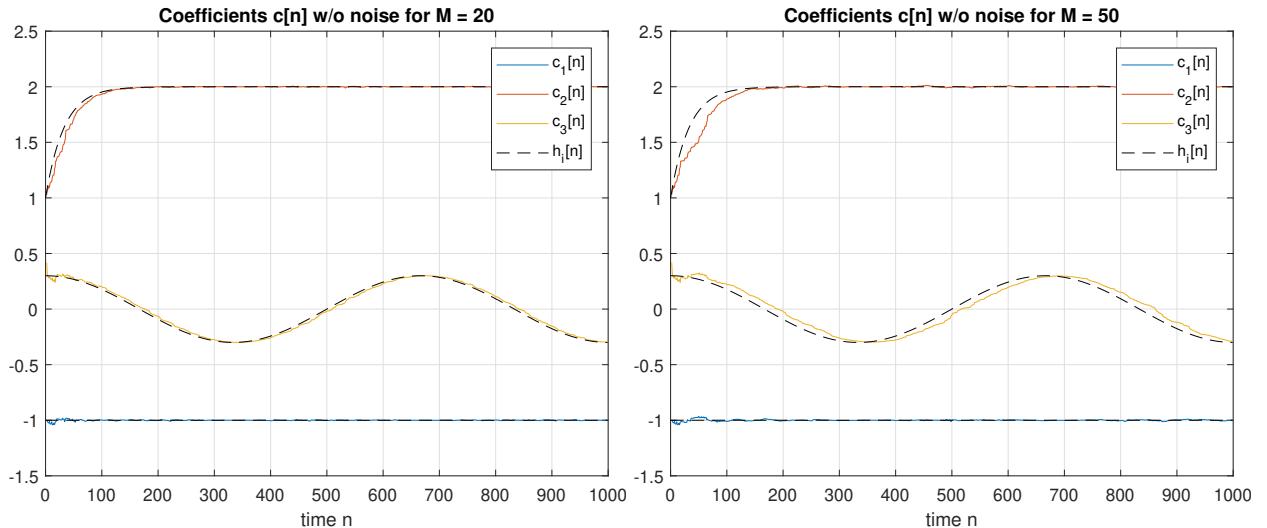


Figure 2: Coefficients for $\mathbf{c}[n]$ without noise and segment length $M = 20$ and $M = 50$

As we can see, the values for $\mathbf{h}[n]$ (black striped lines) correspond quite well with the coefficients which were calculated through the *least-square* sense. The first values of $\mathbf{c}[n]$ vary a lot from the correct solution due to assumption that $x[n] = 0$ for $n < 0$ and therefore the first $\mathbf{c}[n]$ get computed with only a few entries which are not equal to 0.

Between the two plots with different segment length M is only little difference. The initial deviation is longer due to the longer segment length and the plot gets a little bit more shifted (increases with segment length).

3.4 Task d)

Last but not least some noise $w[n]$ with variance $\sigma_w = 0.02$ disturbs the 'measured' data $d[n]$. Again the coefficients were calculated and plotted.

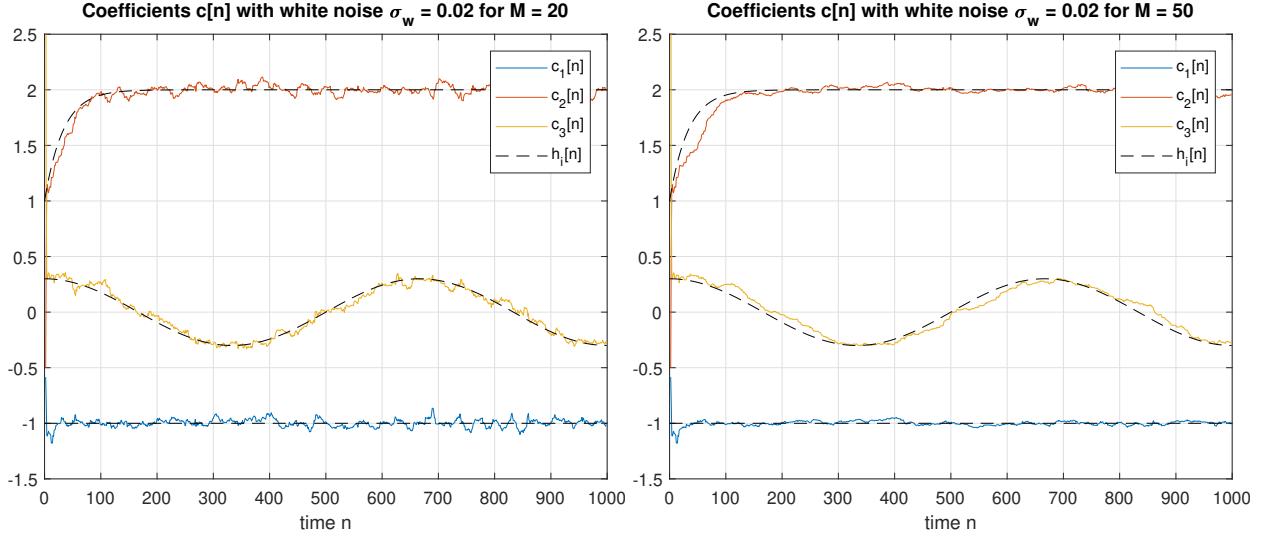


Figure 3: Coefficients for $c[n]$ with noise and segment length $M = 20$ and $M = 50$

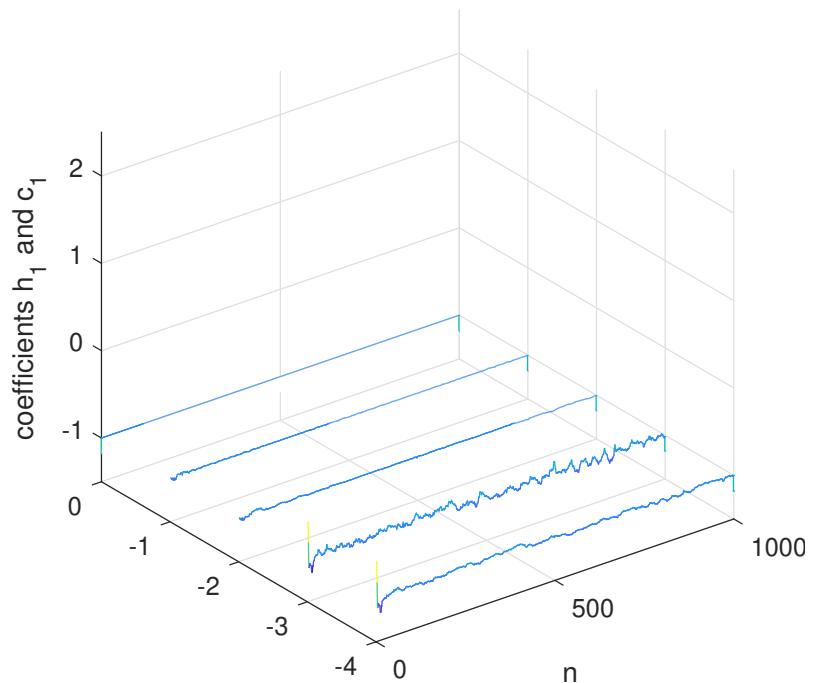
Between $M = 20$ and $M = 50$ is a lot of difference now. The additional noise $w[n]$ causes the coefficients to fluctuate. The plot with the lower segment length $M = 20$ fluctuates way more than the plot with segment length $M = 50$. Due to the property of white noise the effect it has should cancel out for infinite long observation. Since we only look at finite length of data we, some effects of the white noise still can be seen. To further decreases the noise the segment length can be increased.

WATERFALL PLOTS

Order of the lines, starting from top to bottom:

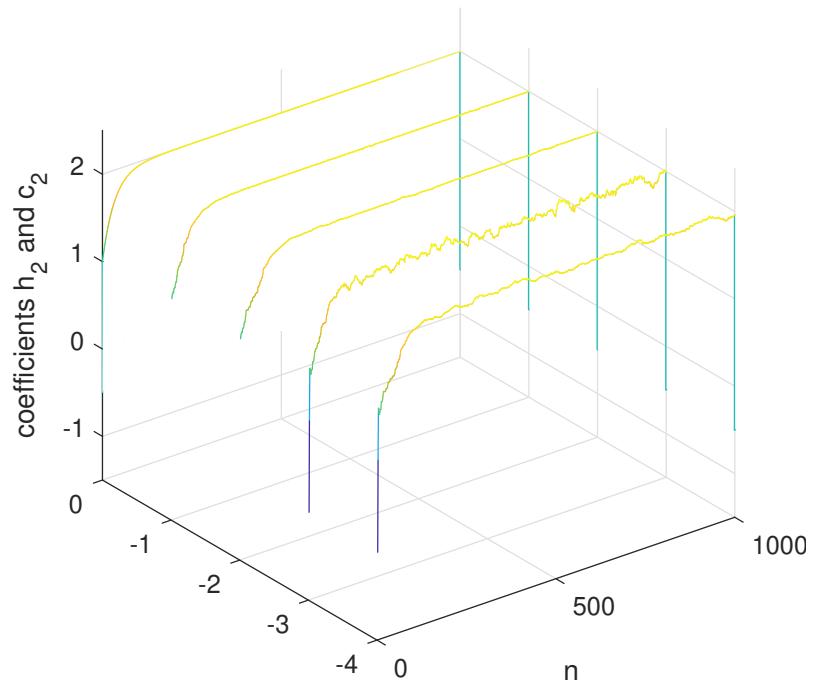
1. $h_i[n]$
2. $c_i[n]$ without measurement noise and for segment length $M = 20$
3. $c_i[n]$ without measurement noise and for segment length $M = 50$
4. $c_i[n]$ with measurement noise and for segment length $M = 20$
5. $c_i[n]$ with measurement noise and for segment length $M = 50$

where i denotes the i -th coefficient



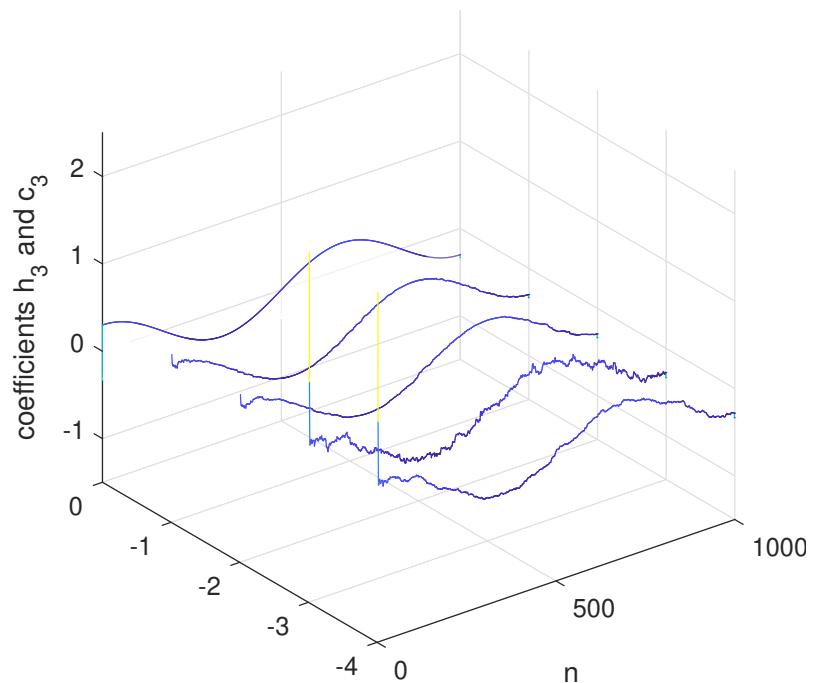
$h | M = 20 \& w = 0 | M = 50 \& w = 0 | M = 20 \& w \sim 0 | M = 50 \& w \sim 0;$

Figure 4: waterfall coefficients $i = 1$



$h \mid M = 20 \ \& \ w = 0 \mid M = 50 \ \& \ w = 0 \mid M = 20 \ \& \ w \sim= 0 \mid M = 50 \ \& \ w \sim= 0;$

Figure 5: waterfall coefficients $i = 2$



$h \mid M = 20 \ \& \ w = 0 \mid M = 50 \ \& \ w = 0 \mid M = 20 \ \& \ w \sim= 0 \mid M = 50 \ \& \ w \sim= 0;$

Figure 6: waterfall coefficients $i = 3$

3.5 Matlab Code

Main File

```
1 close all
2 clear all
3 clc
4
5 %suppres: "Warning: Matrix is singular to working precision."
6 id = 'MATLAB:singularMatrix';
7 warning('off',id)
8
9 %suppres: "Warning: Directory already exists."
10 id = 'MATLAB:MKDIR:DirectoryExists';
11 warning('off',id)
12
13 mkdir 'Figures' %create Figures folder
14
15
16 %
17 % a)
18 N = 2;
19 d = [-3; -5; 0]; %values from example 1.1
20 x = [-1; -1; 1];
21
22
23
24
25
26 k = N-1; %Pad with N - 1 zeroes for the values of x[-1], x[-2], ... x[-N+1]
27 x_pad = [zeros(k,1); x];
28 d_pad = [zeros(k,1); d];
29
30
31 c = ls_filter(x_pad, d_pad, N)
32
33 %"Probe"
34 d_hat = conv(x,c);
35 d_hat = d_hat(1:length(x)) %delete the values which assume that
36 % x[n] = 0 for n > length(x)
37
38
39
40 %
41 % b)
42 theta = 3*pi/1000;
43 n = 0:999;
44
45 h = [-1*ones(1,length(n)); 2-0.97.^n; 0.3*cos(theta*n)];
46 %h = h1[0], h1[1], ..., h1[n];
47 % h2[0], h2[1], ..., h2[n];
48 % h3[0], h3[1], ..., h3[n];
49
50
51 figure
52 plot(n,h)
53 legend('h_1[n]', 'h_2[n]', 'h_3[n]')
54 grid on
```

```

55     ylim([-1.5 2.5])
56     title('Coefficients h[n]')
57     xlabel('time n')
58
59     saveas(gcf, 'Figures/Coefficients_h', 'epsc')
60
61 %_____
62 % c) and d)
63 N = 3; %3 filter coefficients in h and c
64
65 x = randn(1,length(n)).'; %x[n] = 0 for n < 0 (or 1 in matlab)
66
67 d = vector_conv(x, h);
68 % d_test = vector_conv2(x, h)
69 % e_test = d - d_test
70 counter = 1;
71 for jj = 1:2
72
73     if jj == 1
74         w = 0;
75     else
76         %create white gaussian noise and change variance
77         w = transpose(randn(1,length(n)))./(1/sqrt(0.02));
78     end
79
80     d = d + w;% add noise after filter h
81
82
83 for M = [20, 50]
84     x_pad = [zeros(M-1,1); x]; %pad with M-1 zeros; x[n] = 0 for n < 0;
85     d_pad = [zeros(M-1,1); d]; %and pad d too for the newly created values
86     of x[n]
87
88     c = zeros(N,length(n));
89     for ii = n %ii is counts through the time n
90         c(:,ii+1) = ls_filter(x_pad(ii+1:M+ii), d_pad(ii+1:M+ii), N);
91     end
92
93
94     if w == 0
95         text = ['Coefficients c[n] w/o noise for M = ' num2str(M)];
96         text_saveas = ['Coefficients_c_without_noise_M=' num2str(M)];
97     else
98         text = ['Coefficients c[n] with white noise \sigma_w = ' num2str(
99             round(var(w),2)) ' for M = ' num2str(M)];
100        text_saveas = ['Coefficients_c_with_noise_M=' num2str(M)];
101    end
102
103    figure
104        plot(n,c)
105        hold on
106        plot(n,h,'--k')
107        legend('c_1[n]', 'c_2[n]', 'c_3[n]', 'h_i[n]')
108        grid on
109        title(text)
110        xlabel('time n')
111        ylim([-1.5, 2.5]) %due to some singularities, the first values
112                                of c can get quite big -> ruins the plot ->

```

```

111 % limit it
112 saveas(gcf,['Figures/' text_saveas] , 'epsc') %epsc to save the eps
113 in colour
114
115 c_plot(:, :, counter) = c;
116 counter = counter + 1;
117 end %for M
118
119 end %for jj
120
121 kk = 1;
122 X = [1; 1; 1; 1; 1]* n;
123 Y = [0; -1; -2; -3; -4]*ones(1, length(n));
124
125 for kk = 1:3
126 Z = [h(kk,:); c_plot(kk,:,1); c_plot(kk,:,2); c_plot(kk,:,3); c_plot(kk,:,4)];
127 %c_plot(kk,:,1) has the coefficients for M = 20, w = 0;
128 %c_plot(kk,:,2) has the coefficients for M = 50, w = 0;
129 %c_plot(kk,:,3) has the coefficients for M = 20, w ~ 0;
130 %c_plot(kk,:,4) has the coefficients for M = 50, w ~ 0;
131
132 text_saveas = ['waterfall_coefficients_' num2str(kk)];
133
134 figure
135 waterfall(X,Y,Z)
136 zlim([-1.5, 2.5])
137 xlabel('n')
138 ylabel(['coefficients h_ ' num2str(kk) ' and c_ ' num2str(kk)])
139 saveas(gcf,['Figures/' text_saveas] , 'epsc') %epsc to save the eps in colour
140 end
141
142
143 %create a placeholder function to overwrite the saveas function
144 function saveas(~, ~, ~)
145 disp('Figure not saved')
146 end
147
148
149 %seen from plots:
150 %w[n] = 0:
151 %theres only little difference between M = 20 and M = 50 and
152 %the values of c correspond quite well to the values of h,
153 %althrough the plot is a bit shifted(increses with segment
154 %length M). The first values of c are also not quite the same
155 %as h, due to the assumption that x[n] = 0, for n < 0 and the
156 %first c[n] gets computed with only one entry which is not 0;
157 %
158 %w[n] ~ 0:
159 %between M = 20 and M = 50 is a lot of difference now. Probably
160 %due to the higher amount of samples, the noise cancels out
161 %(and would fully cancel for M -> infinity(because white noise),
162 %but then the adaptiveness of system would get lost -> c[n] would
163 %get constant if every x[n] is taken into account)

```

Function ls_filter()

```
1 function c = ls_filter(x, d, N)
2 %computes the filter coefficients c for one time instance n, where
3 %corresponds to the last entry of x
4
5 % x is the input signal saved as col vector
6 % d is the reference signal saved as col vector
7 % N is the order of the filter i.e. the amount of coefficients in c
8
9 % Matrix X to compute the coefficients at time instance n
10 % X = [x[n-M+N], x[n-M+N-1], ..., x[n-M];
11 %       x[n-M+N+1], x[n-M+N], ..., x[n-M+1];
12 %       ...
13 %       x[n],           x[n-1]           ..., x[n-N+1] ];
14
15 % Make sure x and d are col vectors
16 % x = x(:);
17 % d = d(:);
18
19 if isrow(x) || isrow(d)
20     error('vector x or vector d is not a column vector')
21 end
22
23 M = length(x); %segment length
24
25 X = zeros(M-N+1,N); %create placeholder for entries of X
26
27 for ii = 0:N-1 %for order N coefficients of c we need N cols
28     X(:, ii+1) = x( (end-M+N)-ii:end-ii ); %end corresponds to current time n
29 end %to include M-N+1 we have to subtract
      -M+N
30
31 c = (X.' * X)^-1 * X.' * d(end-M+N:end); %pseudo inverse; multiply with
      segmented d
32
33 end
```

Function vector_conv()

```
1 function y = vector_conv(x, h)
2 %calculates the convolution sum defined in Adaptive System UE
3 %
4 %x has to be a column vector in form of
5 % x = x[0];
6 %     x[1];
7 %     ...
8 %     x[n-1]
9 %
10 %where n is the time variable
11
12
13 %h has to be matrix in the form of
14 % h = h1[0], h1[1], ..., h1[n-1];
15 %     h2[0], h2[1], ..., h2[n-1];
16 %     h3[0], h3[1], ..., h3[n-1]
17
18
19 x = x(:); %make sure that x is a col vector
20
21
22 N = size(h,1);
23
24
25 t = 1;
26 n = 0:length(x)-1;
27 x_zero_pad = [zeros(N-1,1); x]; %puts zeros for time x[-1], x[-2], ... x[-N+1]
28
29 y = zeros(length(n),1);
30 for n_shift = n + N
31     x_tap_input = x_zero_pad(n_shift:-1:n_shift-N+1);
32     y(t) = h(:,t)' * x_tap_input; % ' is hermitian transposed
33     t = t+1;
34 end
35
36
37 end
```

Function vector_conv2()

```
1 function y = vector_conv2(x, h)
2 % computes the convolution of x and the coefficients of h at time
3 % instance n for every n
4
5 y = zeros(length(x),1);
6
7 for n = 1:length(x)
8     temp = conv(x,h(:,n));
9     y(n) = temp(n);
10 end
11
12 end
```

4 OCTAVE/MATLAB Problem 1.4

Weighted-Least-Squares Tracking of a Time-Varying System

4.1 Task a)

For the first task the optimal filter coefficients shall be derived so that $\mathbf{c}_{wLS}[n] = \operatorname{argmin}_{\mathbf{c}} J_{wLS}(\mathbf{c}, n)$ according to the weighted least squares cost function

$$J_{wLS}(\mathbf{c}, n) = \sum_{k=n-M+1}^n g[n-k] \cdot |e[k]|^2 \quad (1)$$

The derivation was done by hand and can be found on the next page.

For reference the 'unknown' system filter coefficients were plotted. These are:

$$\mathbf{h}[n] = \begin{bmatrix} -1 \\ 2 - 0.97^n \\ 0.3 \cdot \cos(\theta n) \end{bmatrix} \quad (2)$$

Note: θ was changed from Problem 1.3 to Problem 1.4: $\theta = \frac{3\pi}{100}$

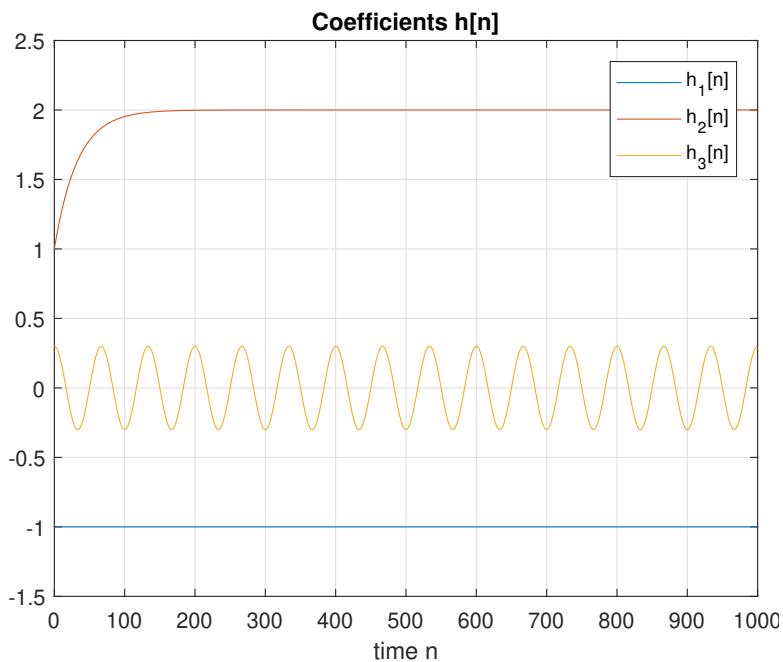


Figure 1: Coefficients for $\mathbf{h}[n]$

Problem 1.4

$$J_{WLS}(c, n) = \sum_{k=n-M+1}^n g[n-k] \cdot |e[k]|^2 \stackrel{e \in \mathbb{R}}{=} \sum_{k=n-M+1}^n e[k] \cdot g[n-k] \cdot e[k]$$

bc $g[n-k]$ "Lautvariable"

$$= \underline{e}^T[n] \cdot (\underline{G} \cdot \underline{e}[n]) = [\underline{e}[n], \underline{e}[n-1], \dots, \underline{e}[n-M+1]] \cdot \begin{bmatrix} g[n] & 0 & \dots & 0 \\ 0 & g[n-1] & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & g[0] \end{bmatrix} \cdot \begin{bmatrix} \underline{e}[n] \\ \underline{e}[n-1] \\ \vdots \\ \vdots \\ \underline{e}[n-M+1] \end{bmatrix}$$

P.1.1

from problem class: $\underline{e}[n] = \text{dar } \underline{X} \underline{c}[n]$

$$\begin{aligned} J_{WLS}(c, n) &= \underline{e}^T[n] \cdot \underline{G} \cdot \underline{e}[n] = (\underline{d} - \underline{X} \cdot \underline{c})^T \cdot \underline{G} \cdot (\underline{d} - \underline{X} \cdot \underline{c}) \\ &= (\underline{d}^T - \underline{c}^T \cdot \underline{X}^T) \cdot \underline{G} \cdot (\underline{d} - \underline{X} \cdot \underline{c}) = \\ &= (\underline{d}^T \underline{G} - \underline{c}^T \underline{X}^T \cdot \underline{G}) \cdot (\underline{d} - \underline{X} \cdot \underline{c}) \\ &\quad \text{scalar} \\ &= (\underline{d}^T \underline{G} \cdot \underline{d} - \underline{c}^T \underline{X}^T \underline{G} \cdot \underline{d} - (\underline{d}^T \underline{G} \cdot \underline{X} \cdot \underline{c}) + \underline{c}^T \underline{X}^T \underline{G} \cdot \underline{X} \cdot \underline{c}) \\ &= \underline{d}^T \underline{G} \cdot \underline{d} - \underline{c}^T \underline{X}^T \underline{G} \cdot \underline{d} - \underbrace{\underline{c}^T \underline{X}^T \underline{G} \cdot \underline{d}}_{\text{vector is in front}} + \underline{c}^T \underline{X}^T \underline{G} \cdot \underline{X} \cdot \underline{c} \quad // \underline{G}^T \underline{G} = \underline{G} \\ &= \underline{d}^T \underline{G} \cdot \underline{d} - \underline{c}^T \underline{X}^T \cdot \underline{G} \cdot \underline{d} + \underline{c}^T \underline{X}^T \cdot \underline{G} \cdot \underline{X} \cdot \underline{c} \\ &\quad \text{constant} \end{aligned}$$

$$\nabla_c(J_{WLS}) = 0 - \underline{X}^T \underline{G} \cdot \underline{d} + 2 \cdot \underline{X}^T \underline{G} \cdot \underline{X} \cdot \underline{c} = 0$$

$$\begin{aligned} \underline{X}^T \underline{G} \cdot \underline{X} \cdot \underline{c} &= \underline{X}^T \underline{G} \cdot \underline{c} \\ \underline{c} &= (\underline{X}^T \underline{G} \cdot \underline{X})^{-1} \cdot \underline{X}^T \underline{G} \cdot \underline{d} \end{aligned}$$

$$\boxed{c_{WLS} = (\underline{X}^T \underline{G} \cdot \underline{X})^{-1} \cdot \underline{X}^T \underline{G} \cdot \underline{d}}$$

4.2 Task b)

In the second task the already programmed `ls_filter()` was changed to include the derived formula from Task a). The new function `ls_filter_weighted(x, d, N, λ)` expects 4 input parameters. The new parameter λ is used for the calculation of the weights:

$$g[m] = \lambda^{-m} \quad (3)$$

The Matlab files can be found in the `.zip` file. The code is also appended to this PDF (Section 4.4)

4.3 Task c)

Different values for λ were tested and plotted for $\theta = \frac{3\pi}{100}$ and $M = 50$. For reference the coefficients of h were drawn as striped black lines.

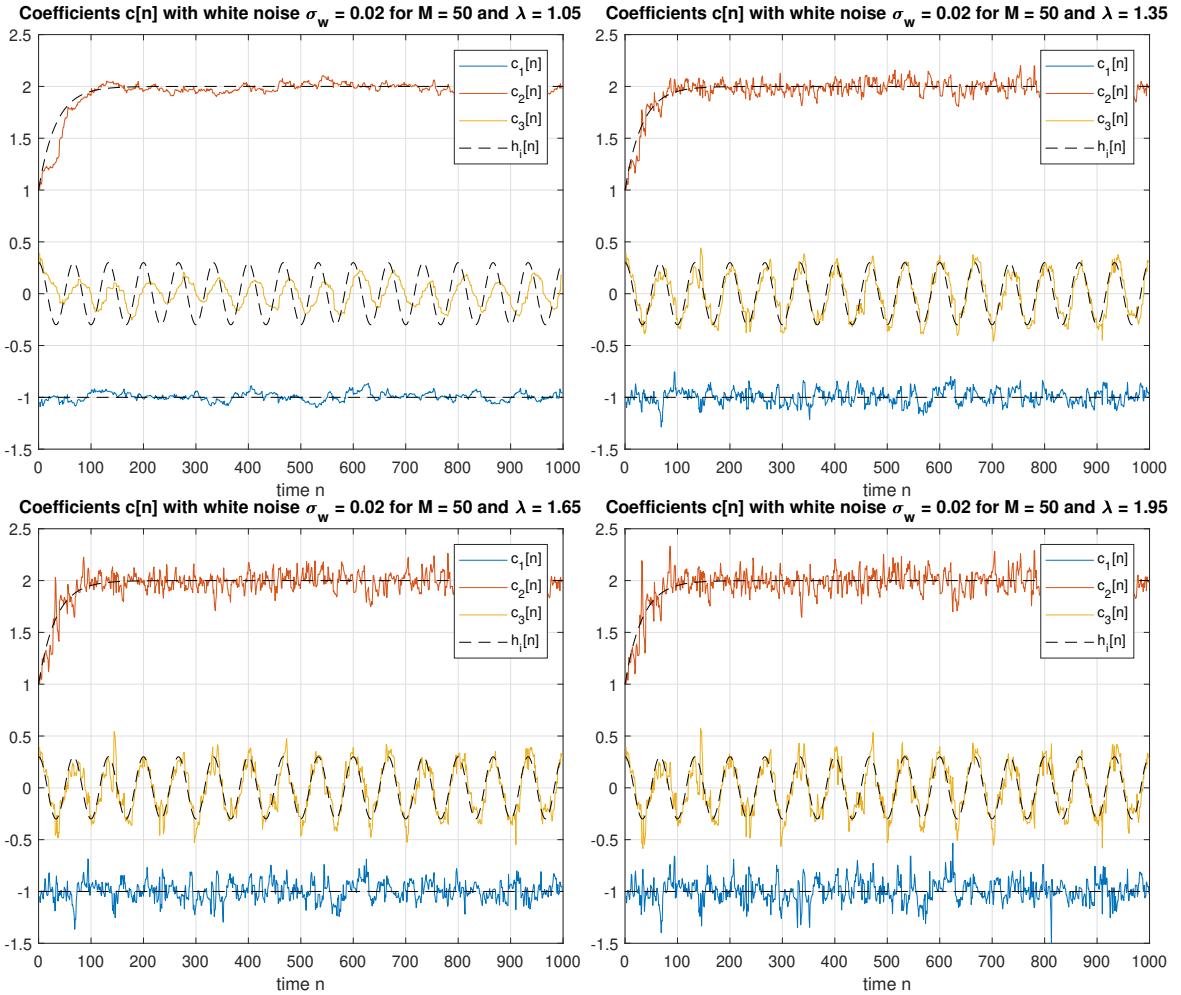


Figure 2: Coefficients for $c[n]$ with noise, segment length $M = 50$ and varying λ

The bigger the λ the greater the added noise. This is because the *weighted-least-squares* emphasises more on the newly gathered information for $e[n]$ which is corrupted by noise $w[n]$. But on the other hand the system is faster now, there is almost no delay between the real time varying coefficients of $h[n]$ and the calculated $c[n]$, which was caused by the segment length M .

A reasonable choice for λ seems to be $\lambda = 1.35$ because the delay between $h[n]$ and $c[n]$ is marginal and the noise doesn't seem too bad.

4.4 Matlab Code

Main File

```
1 close all
2 clear all
3 clc
4
5 %suppres: "Warning: Matrix is singular to working precision."
6 id = 'MATLAB:singularMatrix';
7 warning('off',id)
8
9 %suppres: "Warning: Directory already exists."
10 id = 'MATLAB:MKDIR:DirectoryExists';
11 warning('off',id)
12
13 mkdir 'Figures'
14
15 %
16 % calculate h
17 theta = 3*pi/100; %change theta from problem 1.3
18 n = 0:999;
19
20 h = [-1*ones(1,length(n)); 2-0.97.^n; 0.3*cos(theta*n)];
21 %h = h1[0], h1[1], ..., h1[n];
22 % h2[0], h2[1], ..., h2[n];
23 % h3[0], h3[1], ..., h3[n];
24
25
26 figure
27 plot(n,h)
28 legend('h_1[n]', 'h_2[n]', 'h_3[n]')
29 grid on
30 ylim([-1.5 2.5])
31 title('Coefficients h[n]')
32 xlabel('time n')
33
34 saveas(gcf, 'Figures/Coefficients_h', 'epsc')
35
36 %
37 % weighted system identification
38 N = 3; %3 filter coefficients in h and c
39
40 x = randn(1,length(n)).'; %x[n] = 0 for n < 0 (or 1 in matlab)
41
42 d = vector_conv(x, h);
43
44 %create white gaussian noise and change variance
45 w = transpose(randn(1,length(n)))./(1/sqrt(0.02));
46
47 d = d + w;% add noise after filter h
48
49
50 for lambda = 1.05:0.3:2
51
52 for M = [50]
53 x_pad = [zeros(M-1,1); x]; %pad with M-1 zeros; x[n] = 0 for n < 0;
```

```

54 d_pad = [zeros(M-1,1); d]; %and pad d too for the newly created values
55 % of x[n]
56
57 c = zeros(N,length(n));
58 for ii = n %ii is counts through the time n
59     c(:,ii+1) = ls_filter_weighted(x_pad(ii+1:M+ii), d_pad(ii+1:M+ii), N
60         , lambda);
61 end
62
63 text = ['Coefficients c[n] with white noise \sigma_w = ' num2str(round(
64     var(w),2)) ' for M = ' num2str(M) ' and \lambda = ' num2str(lambda)];
65 text_saveas = ['Coefficients_c_with_noise_M=' num2str(M) '_and_lambda='
66     strrep(num2str(lambda), '.', '_')];
67
68 figure
69     plot(n,c)
70     hold on
71     plot(n,h, '--k')
72     legend('c_1[n]', 'c_2[n]', 'c_3[n]', 'h_i[n]')
73     grid on
74     title(text)
75     xlabel('time n')
76     ylim([-1.5, 2.5]) %due to some singularities, the first values
77 % of c can get quite big -> ruins the plot ->
78 % limit it
79     saveas(gcf,['Figures/' text_saveas], 'epsc') %epsc to save the eps
80     in colour
81
82 end %for M
83
84 end %for lambda
85
86 %create a placeholder function to overwrite the saveas function
87 function saveas(~, ~, ~)
88     disp('Figure not saved')
89 end

```

Function ls_filter_weighted()

```
1 function c = ls_filter_weighted(x, d, N, lambda)
2 %computes the filter coefficients c for one time instance n, where
3 %corresponds to the last entry of x
4
5 % x is the input signal saved as col vector
6 % d is the reference signal saved as col vector
7 % N is the order of the filter i.e. the amount of coefficients in c
8
9 % Matrix X to compute the coefficients at time instance n
10 % X = [x[n-M+N], x[n-M+N-1], ..., x[n-M];
11 %       x[n-M+N+1], x[n-M+N], ..., x[n-M+1];
12 %       ...
13 %       x[n],           x[n-1]           ..., x[n-N+1] ];
14
15 % Make sure x and d are col vectors
16 % x = x(:);
17 % d = d(:);
18
19 if isrow(x) || isrow(d)
20     error('vector x or vector d is not a column vector')
21 end
22
23 M = length(x); %segment length
24
25 X = zeros(M-N+1,N); %create placeholder for entries of X
26
27 for ii = 0:N-1 %for order N coefficients of c we need N cols
28     X(:, ii+1) = x( (end-M+N)-ii:end-ii ); %end corresponds to current time n
29 end %to include M-N+1 we have to subtract
      -M+N
30
31
32 %compute the weighting matrix
33 % lambda = 1.5;
34 m = 0:-1:-(M-N);
35 g = lambda.^m;
36 G = diag(flip(g)); %flip because of form of matrix G (from g[n - k])
37
38 %compute coefficients with weighted input
39
40 c = (X.' * G * X)^-1 * X.' * G * d(end-M+N:end);
41
42 end
```

Function vector_conv()

```
1 function y = vector_conv(x, h)
2 %calculates the convolution sum defined in Adaptive System UE
3 %
4 %x has to be a column vector in form of
5 % x = x[0];
6 %     x[1];
7 %     ...
8 %     x[n-1]
9 %
10 %where n is the time variable
11
12
13 %h has to be matrix in the form of
14 % h = h1[0], h1[1], ..., h1[n-1];
15 %     h2[0], h2[1], ..., h2[n-1];
16 %     h3[0], h3[1], ..., h3[n-1]
17
18
19 x = x(:); %make sure that x is a col vector
20
21
22 N = size(h,1);
23
24
25 t = 1;
26 n = 0:length(x)-1;
27 x_zero_pad = [zeros(N-1,1); x]; %puts zeros for time x[-1], x[-2], ... x[-N+1]
28
29 y = zeros(length(n),1);
30 for n_shift = n + N
31     x_tap_input = x_zero_pad(n_shift:-1:n_shift-N+1);
32     y(t) = h(:,t)' * x_tap_input; % ' is hermitian transposed
33     t = t+1;
34 end
35
36
37 end
```