Signal Processing and Speech Communication Lab.
Graz University of Technology

AS

Adaptive Systems—Homework Assignment 1 v1.0

Name(s) 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 $\mathbf{a} = [a_1, a_2, a_3]^\mathsf{T}$ 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 form 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 $J_{\mathrm{MSE}}(\mathbf{c}) = \mathsf{E}\big[|e[n]|^2\big]$.

(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 $x[n-k]$:

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

(*iv*) Derive the orthogonality principle from the MSE cost function $J_{\mathrm{MSE}} = \mathsf{E}\big[|e[n]|^2\big]$

(*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 $r_{xx}[n,k] = \mathsf{E}\big[x[n+k]x^*[n]\big]$ 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 $\sigma_w^2 \to 0$?

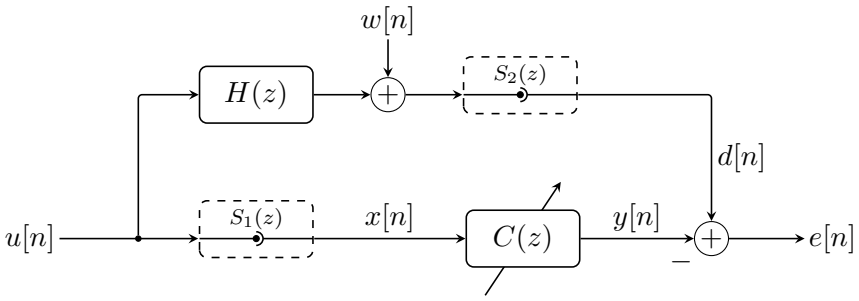
(*ix*) When dealing with stochastic signals, we can only analyze the frequency content of the signal by computing the Fourier transform (the DTFT) of the corresponding autocorrelation function, then called the power spectral density (PSD)[1] $S_{xx}(\theta)$. Compute and sketch the PSD, defined as

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

Again show what happens for $\sigma_w^2 \to 0$. 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 $\mathbf{p} = \mathsf{E}\big[d[n]\mathbf{x}[n]\big]$,
3. the optimal coefficient vector $\mathbf{c}_{\text{opt}}$ in the sense of a minimum mean-squared error, i.e. $\mathbf{c}_{\text{opt}} = \text{argmin}_{\mathbf{c}}\, J_{\text{MSE}}(\mathbf{c})$ with $J_{\text{MSE}}(\mathbf{c}) = \mathsf{E}\big[|e[n]|^2\big]$, and
4. the minimum mean-squared error $J_{\text{min}} = J_{\text{MSE}}(\mathbf{c}_{\text{opt}})$.

[1] You 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 $S_2(z) = 1 + z^{-2}$.

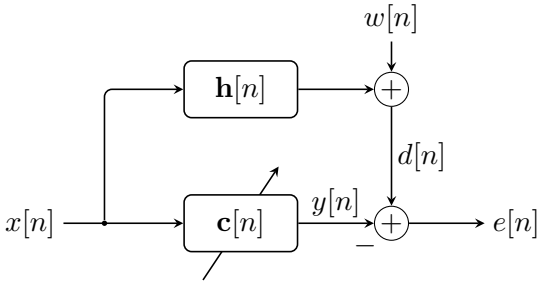
(e) (1 point) What happens when the sensor responses are known? Can the system be identified correctly? Can the cascade in **(d)** be identified correctly? You do not need to compute anything, just give a short explanation.

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

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

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 $\mathbf{c}_{\mathrm{LS}}[n] = \mathrm{argmin}_{\mathbf{c}}\, J_{\mathrm{LS}}(\mathbf{c}, n)$ according to the cost function

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

You should hand in (at least) two Octave/Matlab file: One file that creates the signals, calls your function and preforms 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 $n = 0 \ldots 999$.

(c) (4 points) Assume now that the system is noise–free, i.e. $w[n] = 0$. Generate 1000 samples (for $n = 0 \ldots 999$) 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 $n = 0 \ldots 999$ can be computed. Note that we thus obtain $\mathbf{c}[n] = \text{argmin}_{\mathbf{c}}\, J(\mathbf{c}, n)$ 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 $n = 0 \ldots M$ are given.

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

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

(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 $J_{\text{wLS}}(\mathbf{c}, n)$ 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

a

i) $\underline{A} = [\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_n]$ (some signal) $\qquad \underline{a}_i = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \in \mathbb{R}$

now: size$(\underline{A}) = (N \times 4)$ (row x col)

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

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

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

ii) $\underset{(3\times 1)}{\underline{a}} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$

we want: $\underset{(3\times 1)}{\begin{bmatrix} b_1 \cdot a_1 \\ b_2 \cdot a_2 \\ b_3 \cdot a_3 \end{bmatrix}} = \underline{B} \cdot \underline{a} = \begin{bmatrix} b_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & b_3 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \Rightarrow \underline{B = \text{eye}(3) \neq X}$ (3×3)

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

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

$\underset{(N\times 3)}{\underline{D}} = [\underline{d}_1 \ \underline{d}_2 \ \underline{d}_3]$

we want: $\underset{(N\times 3)}{[b_1 \underline{d}_1 \quad b_2 \underline{d}_2 \quad b_3 \underline{d}_3]} \overset{!}{=} \underline{D} \cdot \underline{B} = [\underline{d}_1 \ \underline{d}_2 \ \underline{d}_3] \cdot \begin{bmatrix} b_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & b_3 \end{bmatrix}$

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

iii) pic of: $\det(a \cdot \underline{A}) = a^N \cdot \det(\underline{A})$; # size$(\underline{A}) = (N \times N)$

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

$= a^2 \cdot (b_1 \cdot b_4 - b_2 \cdot b_3) = \underline{a^2 \cdot \det(A)}$ ✓ ✓

(iv) $J_{MSE} = E\{|e[n]|^2\} = E\{e[n] \cdot e^*[n]\}$

$= E\{(d[n] - y[n]) \cdot (d^*[n] - y^*[n])\}$ // $(a-b)^* = a^* - b^*$

$= E\{|d[n]|^2 - d[n] \cdot y^*[n] - y[n] \cdot d^*[n] + |y[n]|^2\}$

$= E\{|d[n]|^2 - d[n] \cdot \left(\sum_{k=0}^{N-1} x[n-k] \cdot c_k[n]\right)^* - d^*[n] \cdot \left(\sum_{k=0}^{N-1} x[n-k] \cdot c_k[n]\right) + \left|\sum_{k=0}^{N-1} x[n-k] \cdot c_k[n]\right|^2\}$ // $|a|^2 = |a^2|$

↳ assuming: $\frac{d}{dx}(f(x))^* = \left(\frac{d}{dx} f(x)\right)^*$ // probably wrong, but I don't know what else I could do

$|(1-1j)|^2 = \sqrt{2}^2$

$|(1-1j)^2| = |1-2j+j^2| = |-2j| = 2$

→ For Minimum: $\nabla_k J_{MSE} \stackrel{!}{=} 0$; $\nabla_k = \frac{\partial}{\partial c_k}$

$\nabla_k J_{MSE} \stackrel{!}{=} 0 = \nabla_k E\{$ —— // —— $\}$

$= E\{\nabla_k(|d[n]|^2) - \nabla_k\left(d[n] \cdot \left(\sum x[n-k] \cdot c_k[n]\right)^*\right) - \nabla_k d^*[n] \cdot \left(\sum x[n-k] \cdot c_k[n]\right) + \nabla_k \left|\sum x[n-k] \cdot c_k[n]\right|^2\}$

$= E\{0 - x[n-k]^* \cdot d[n] - d^*[n] \cdot x[n-k] + 2 \cdot \underbrace{\left(\sum x[n-k] \cdot c_k[n]\right)}_{y[n]} \cdot x[n-k]\}$ 1

Assuming: $e[n] \in \mathbb{R} \rightarrow e[n] = e^*[n]$; $x[n] = x^*[n]$; $d[n] = d^*[n]$ // no clue how to solve with the complex conj. C) - Opembis

$= E\{-x[n-k] \cdot [\underbrace{d[n] + d[n]}_{2d[n]}] + 2 \cdot y[n] \cdot x[n-k]\}$

$= E\{-2 \cdot x[n-k] \cdot [\underbrace{d[n] - y[n]}_{e[n]}]\}$

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

$\underline{0 = E\{e[n] \cdot \underbrace{x[n-k]}_{\underline{x}[n]}\}}$ // Principle of orthogonality

(v) $0 = E\{e[n] \cdot x[n-k]\} = E\{e[n] \cdot \underline{x}[n]\}$

$= E\{(d[n] - \underline{c}^T \cdot \underline{x}[n]) \cdot \underline{x}[n]\}$

$= E\{\underline{x}[n] \cdot (d[n] - \underline{c}^T \cdot \underline{x}[n])\}$

$= E\{\underline{x}[n] \cdot d[n] - \underline{x}[n] \cdot \underline{c}^T \cdot \underline{x}[n]\}$ // $(\underline{c}^T)^T = \underline{c}$

$= E\{\underline{x}[n] \cdot d[n]\} - E\{\underline{x}[n] \cdot \underline{x}^T[n] \cdot \underline{c}\}$

$0 = \underline{p} - \underline{R}_{xx} \cdot \underline{c}$ // $\underline{p}$, $\underline{R}_{xx}$

$\underline{R}_{xx} \cdot \underline{c} = \underline{p}$

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

(vi) $\underline{R}_{xx}$ needs to be invertible $\Rightarrow$ $\underline{R}_{xx}$ semi positive definit or $\det(\underline{R}_{xx}) > 0$

c)

$X[n] = a \cdot e^{j(\theta_0 \cdot n + \varphi)} + w[n]$ $\quad a, \theta_0 \in \mathbb{R}$

(vii) $\varphi \sim \mathcal{U}[-\pi, \pi]$ ✓

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

↳ $E\{w[n]\} = 0$

$r_{xx}[n,k] = E\{X[n+k] \cdot X^*[n]\}$ $\quad a^* = a$ $\quad w[n]$ and φ are independet → $E\{w[n] \cdot \varphi\} = E\{w[n]\} \cdot E\{\varphi\}$

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

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

↳ $E\{w[n+k] \cdot w^*[n]\} = r_{ww}[k]$

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

↓ independ!

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

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

↳ Kronecker delta

✓

vii) <u>for WSS:</u> first and second order moment are not dependet on time n

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

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

✓

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

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

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

Q1

<u>NR:</u> $E\{g(x)\} = \int_{-\infty}^{\infty} g(x) \cdot f(x)\, dx$ ↳ probability distribution func.

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

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

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

$\varphi \sim \mathcal{U}(-\pi, \pi)$

$f(\varphi) =$

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

✓

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

viii) $\underline{R_{xx}} = ?$ hermitian = complex conj and transposed $X^H = (X^T)^*$

$$\underline{R_{xx}} = E\{\underline{x}[n] \cdot \underline{x}^H[n]\} = \begin{bmatrix} r_{xx}[0] & r_{xx}[1] & \cdots & r_{xx}[N-1] \\ r_{xx}[-1] & r_{xx}[0] & & \vdots \\ \vdots & & \ddots & \vdots \\ r_{xx}[-N+1] & \cdots & \cdots & r_{xx}[0] \end{bmatrix}$$

$$r_{xx}[n] = a^2 \cdot e^{j\vartheta_0 \cdot k} + \sigma_w^2 \cdot \delta[n]$$

N = 1:

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

N = 2:

$$\underline{R_{xx}} = \begin{bmatrix} r_{xx}[0] & r_{xx}[1] \\ r_{xx}[-1] & r_{xx}[0] \end{bmatrix} = \begin{bmatrix} a^2 + \sigma_w^2 & a^2 \cdot e^{j\vartheta_0} \\ a^2 \cdot e^{-j\vartheta_0} & a^2 + \sigma_w^2 \end{bmatrix}$$

$$\det(\underline{R_{xx}}) = (a^2 + \sigma_w^2)^2 - a^4 \cdot e^{j\vartheta_0} \cdot e^{-j\vartheta_0} = a^4 + 2a^2\sigma_w^2 + \sigma_w^4 - a^4 =$$

$$= 2a^2 \cdot \sigma_w^2 + \sigma_w^4 = \underline{\sigma_w^2 \cdot [2a^2 + \sigma_w^2]} \quad \text{invertible for } |\sigma_w| > 0$$

N = 3:

$$\underline{R_{xx}} = \begin{bmatrix} r_{xx}[0] & r_{xx}[1] & r_{xx}[2] \\ r_{xx}[-1] & r_{xx}[0] & r_{xx}[1] \\ r_{xx}[-2] & r_{xx}[-1] & r_{xx}[0] \end{bmatrix} = \begin{bmatrix} a^2+\sigma_w^2 & a^2 e^{j\vartheta_0} & a^2 e^{2j\vartheta_0} \\ a^2 e^{-j\vartheta_0} & a^2+\sigma_w^2 & a^2 e^{j\vartheta_0} \\ a^2 e^{-2j\vartheta_0} & a^2 e^{-j\vartheta_0} & a^2+\sigma_w^2 \end{bmatrix}$$

Sarrus:

$$\det(\underline{R_{xx}}) = (a^2+\sigma_w^2)^3 + a^2 e^{j\vartheta_0} a^2 e^{j\vartheta_0} a^2 e^{-2j\vartheta_0} + a^2 e^{2j\vartheta_0} a^2 e^{-j\vartheta_0} a^2 e^{-j\vartheta_0}$$
$$- a^2 e^{-2j\vartheta_0} \cdot (a^2+\sigma_w^2) \cdot a^2 e^{2j\vartheta_0} - a^2 e^{-j\vartheta_0} \cdot a^2 e^{j\vartheta_0} \cdot (a^2+\sigma_w^2) - (a^2+\sigma_w^2) \cdot a^2 e^{-j\vartheta_0} e^{j\vartheta_0}$$

$$= (a^2+\sigma_w^2)^3 + a^6 \cdot 2 - [3a^4] \cdot (a^2+\sigma_w^2)$$

$$= a^6 + 3a^4\sigma_w^2 + 3a^2\sigma_w^4 + \sigma_w^6 + 2a^6 - [3a^6 + 3a^4\sigma_w^2]$$

$$= \underline{3a^2 \cdot \sigma_w^4 + \sigma_w^6}$$

Addition ↑ Multiplication

The $\det(\underline{R_{xx}})$ for $N = 4:10$ was calculated in MATLAB (example_1_a.m

N = 4: $\det(\underline{R_{xx}}) = 4a^2 \cdot \sigma_w^6 + \sigma_w^8$

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

Order Coefficient

generalize: $\det(\underline{R_{xx}})_N = N \cdot a^2 \cdot \sigma_w^{2(N-1)} + \sigma_w^{2N}$ // controlled for $N = 6:10$

$\rightarrow \det(\underline{R_{xx}}) > 0$ for $\sigma_w^2 \neq 0 \rightarrow$ invertible for $\sigma_w \neq 0$

ix)

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

$$= \sum_{k=-\infty}^{\infty} \left(a^2 e^{j\theta_0 k} + \sigma_w^2 \cdot \underbrace{\delta[k]}_{=0 \text{ for } k \neq 0} \right) \cdot e^{-j\theta k}$$

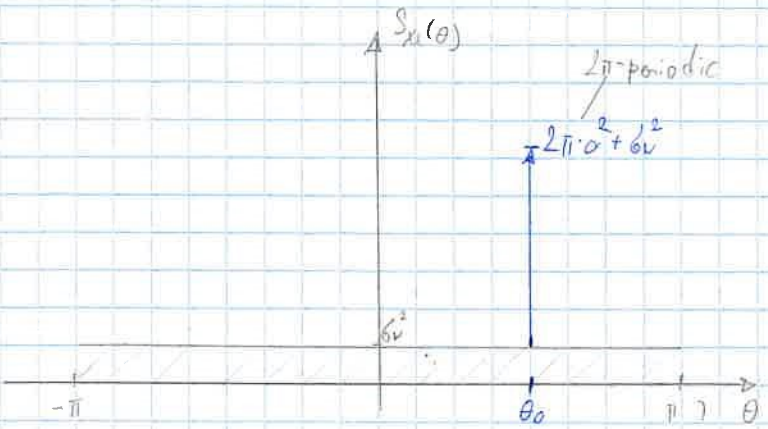
$$= \sigma_w^2 \cdot 1 \cdot e^{-j0} + \sum_{k=-\infty}^{\infty} a^2 \cdot e^{j\theta_0 \cdot k} \cdot e^{-j\theta \cdot k}$$

$$\left(= \sigma_w^2 + \sum_{k=-\infty}^{\infty} a^2 \cdot e^{-j(\theta - \theta_0) \cdot k} \right)$$ // „geometrische Reihe" doesn't converge

$$= \sigma_w^2 + DTFT_k\left(a^2 \cdot e^{j\theta_0 \cdot k} \right)$$ // „Formelsammlung"

$$= \sigma_w^2 + 2\pi \cdot \delta_{2\pi}(\theta - \theta_0) \cdot a^2$$

↳ 2π-periodic dirac delta



|| if $x[n]$ has a line spectrum with L lines, then R_{xx} is only invertible for $N \leq L$

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

therefore R_{xx} is invertible for $N \in \mathbb{N}$ as long as $\sigma_w^2 \neq 0$

if $\sigma_w = 0$, then there would be only 1 line → $N \leq 1$ ✓ // is true for calculated formulas

2

a)

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

$X[n] = S_1 * U[n] = U[n]$

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

$= \delta_u^2 \cdot \delta[k] = 1 \cdot \delta[k]$

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

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

$d[n] = S_2 * (\underline{h}^T \underline{U}[n] + w[n]) = \underline{h}^T \underline{u}[n] + w[n]$

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

$\underline{p} = E\{\underline{h}^T \cdot \underline{u}[n] \cdot \underline{u}[n]\} + \underbrace{E\{w[n] \cdot \underline{u}[n]\}}_{0}$

$\underline{p} = E\{[1 \;\; \frac{1}{4} \;\; 0] \begin{bmatrix} u[n] \\ u[n-1] \\ u[n-1] \end{bmatrix} \cdot \underline{u}[n]\}$

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

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

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

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) $J_{min} = J_{MSE}(\underline{c}_{opt}) \qquad y[n] = \underline{c}^T \cdot \underline{u}[n]$

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

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

$J_{MSE}(\underline{c}_{opt}) = E\{((\underline{h}^T - \underline{h}^T) \cdot u[n] - w[n]) \cdot ((\underline{h}^T - \underline{h}^T) \cdot u[n] - w[n])^*\}$

$= E\{(-w[n]) \cdot (-w[n])^*\} = \frac{1}{4} = \delta_w^2$

$J_{min} = \underline{\underline{\frac{1}{4}}}$ ✓

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

1) $X[n] = s_1 * u[n] = (\delta[n] + \delta[n-1]\cdot 0{,}5) * u[n]$

$$= \sum_{k=0}^{N-1} (\delta[n] + \delta[n-1]\cdot 0{,}5)\cdot u[n+k]$$

$$= u[n] + 0{,}5\, u[n-1]$$

$r_{xx}[n,k] = E\{(u[n+k] + 0{,}5\,u[n+k-1])(u[n] + u[n-1]\cdot 0{,}5)^*\}$

$= E\{u[n+k]\cdot u^*[n] + u[n+k]\cdot u^*[n-1]\cdot 0{,}5 + 0{,}5\cdot u[n+k-1]\cdot u^*[n]$

$+ 0{,}5\, u[n+k-1]\cdot 0{,}5\, u^*[n+k-1]\}$

$= \delta[k] + 0{,}5\,\delta[k+1] + 0{,}5\,\delta[k-1] + 0{,}25\,\delta[k]$

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

$$\underline{R}_{xx} = \begin{pmatrix} 1{,}25 & 0{,}5 & 0 \\ 0{,}5 & 1{,}25 & 0{,}5 \\ 0 & 0{,}5 & 1{,}25 \end{pmatrix}$$

✓

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

$$= \sum_{k=0}^{N-1} (\delta[n] + 0{,}5\,\delta[n-1])\cdot (\underline{h}^T\cdot \underline{u}[n+k] + w[n+k])$$

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

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

$\cdot (\underline{u}[n] + 0{,}5\cdot \underline{u}[n-1])\}$

$\underline{p} = E\{\underline{h}^T\underline{u}[n]\cdot \underline{u}[n] + \underline{h}^T\underline{u}[n]\,\underline{u}[n-1]\cdot 0{,}5 + w[n]\cdot \underline{u}[n] + w[n]\,\underline{u}[n-1]\cdot 0{,}5$

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

$\underline{p} = E\{\underline{h}^T\underline{u}[n]\underline{u}[n]\} + E\{\underline{h}^T\underline{u}[n]\cdot \underline{u}[n-1]\}^{0{,}5} + 0 + 0 + E\{\underline{h}^T\underline{u}[n-1]\cdot 0{,}5\cdot \underline{u}[n]\}$

$+ E\{\underline{h}^T\underline{u}[n-1]\cdot 0{,}25\cdot \underline{u}[n-1]\} + 0 + 0$

$= E\{(u[n] + \tfrac{1}{4}u[n-1])\cdot \underline{u}[n]\} + 0{,}5\cdot E\{(u[n] + \tfrac{1}{4}u[n-1])\cdot \underline{u}[n-1]\}$

$+ 0{,}5\, E\{(u[n-1] + \tfrac{1}{4}u[n-2])\cdot \underline{u}[n]\} + 0{,}25\, E\{(u[n-1] + \tfrac{1}{4}u[n-2])\cdot \underline{u}[n-1]\}$

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

$$= \begin{bmatrix} 1{,}375 \\ 0{,}8125 \\ 0{,}125 \end{bmatrix} = \underline{p} = \begin{bmatrix} 11/8 \\ 13/16 \\ 1/8 \end{bmatrix}$$

✓

3) $\underline{c} = \underline{R}_{xx}^{-1} \cdot \underline{p} = \begin{bmatrix} \frac{84}{85} & -\frac{8}{17} & \frac{16}{85} \\ -\frac{8}{17} & \frac{20}{17} & -\frac{8}{17} \\ \frac{16}{85} & -\frac{8}{17} & \frac{84}{85} \end{bmatrix} \cdot \begin{bmatrix} 11/8 \\ 13/16 \\ 1/8 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/4 \\ 0 \end{bmatrix} = \underline{h}$ ✓

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

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

$\xi_{min} = E\{ (w[n] + 0{,}5\, w[n-1]) \cdot (w[n] + 0{,}5\, w[n-1])^* \}$

$= E\{ w[n] \cdot w[n] + 0{,}5\, w[n] \cdot w[n-1] + 0{,}5\, w[n-1] \cdot w[n] + 0{,}25\, w[n-1] \cdot w[n-1] \}$

$= 0{,}25\, \sigma_w^2 + \sigma_w^2 = \frac{1}{4} + \frac{1}{16} = \underline{\underline{\frac{5}{16}}}$ ✓

c)

1) is the same as a) 1

$r_{xx} = \delta[k] \qquad x[n] = u[n]$

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

2)

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

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

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

$= E\{(u[n-1] + \frac{1}{4}u[n-2])\,\underline{u}[n]\} + E\{\underbrace{w[n-1] \cdot u[n]}_{0}\}$

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

3) $\underline{c} = \underline{R}_{xx}^{-1} \cdot \underline{p} = \begin{bmatrix} 0 \\ 1 \\ 1/4 \end{bmatrix}$ ✓

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

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

$= E\{\underline{h}^T \underline{u}[n-1]\, \underline{h}^T \underline{u}[n-1] + \underline{h}^T \underline{u}[n-1] \cdot \underbrace{w[n-1]}_{0} - \underline{h}^T \underline{u}[n-1]\, \underline{c}^T \underline{u}[n]$

$+ \underbrace{w[n-1] \cdot \underline{h}^T \underline{u}[n-1]}_{0} + w[n-1] \cdot w[n-1] - \underbrace{w[n-1] \cdot \underline{c}^T \underline{u}[n]}_{0}$

$- \underline{c}^T \underline{u}[n]\, \underline{h}^T \underline{u}[n-1] - \underbrace{\underline{c}^T \underline{u}[n] \cdot w[n-1]}_{0} + \underline{c}^T \underline{u}[n] \cdot \underline{c}^T \underline{u}[n]\}$

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

$+ \delta_w^2 - \underline{c}^T E\{\underline{u}[n] \cdot \underline{u}^T[n-1]\}\, \underline{h} + \underline{c}^T E\{\underline{u}[n] \cdot \underline{u}^T[n]\}\, \underline{c}$

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

$= 1 + \frac{1}{16} - 1 - \frac{1}{16} + \frac{1}{4} - 1 - \frac{1}{16} + 1 + \frac{1}{16} = \underline{\underline{\frac{1}{4}}} = J_{min}$

✓

d) 1) is the same as a) 1

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

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

✓

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

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

$$= E\{(u[n] + \tfrac{1}{4} u[n-1]) \cdot \underline{u}[n]\} + E\{(u[n-2] + \tfrac{1}{4} u[n-3]) \cdot \underline{u}[n]\}$$

$$= \begin{bmatrix} 1 \\ \frac{1}{4} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/4 \\ 1 \end{bmatrix} = \underline{p}$$

✓

3) $\underline{c} = \underline{R}_{xx}^{-1} \cdot \underline{p} = \begin{bmatrix} 1 \\ 1/4 \\ 1 \end{bmatrix}$

✓

4) $J_{Min} = J_{MSE}(\underline{c}_{opt})$ $\quad e[n] = \underline{h}^T \underline{u}[n] + w[n] + \underline{h}^T \underline{u}[n-2] + w[n-2] - \underline{c}^T \underline{u}[n]$

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

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

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

$+ \dots\dots$

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

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

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

$= 1 + 1 + 1 + 1 + \frac{1}{16} + \frac{1}{2} = \frac{9}{16}$

✓

e)

- ✓ if the two sensor respones are the same, the system gets identified correctly
- ✓ if the two sensor respones, 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
- ✓ The system in d) cannot be identified correctly.

16.1

3 OCTAVE/MATLAB Problem 1.3

Least-Squares Tracking of a Time-Varying System

17.1

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

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

We know from the problem class:

$$c_{LS} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{d} \tag{2}$$

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

17.2

3.2 Task b)

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

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

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

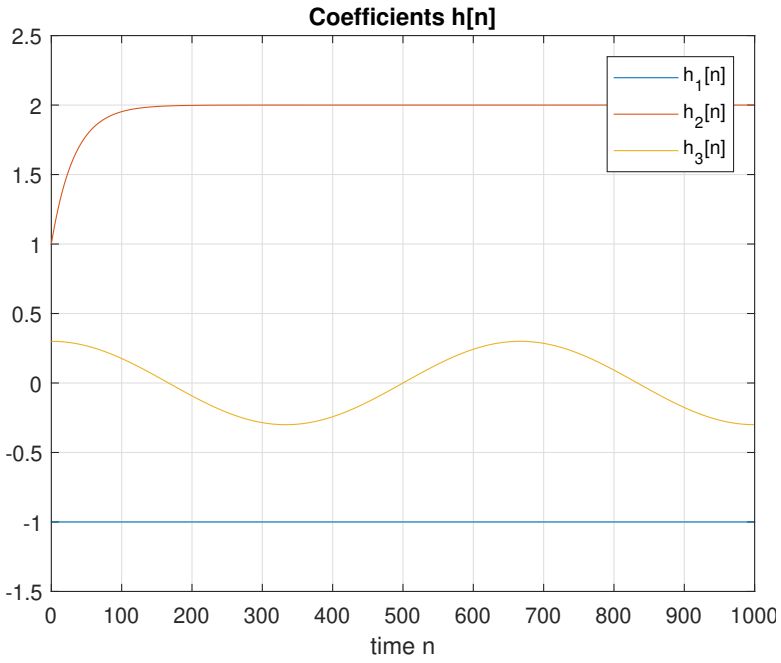


Figure 1: Coefficients for **h**[n]

18.1

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 **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 **h**[n] (that is why we know the coefficients of **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).

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

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}.

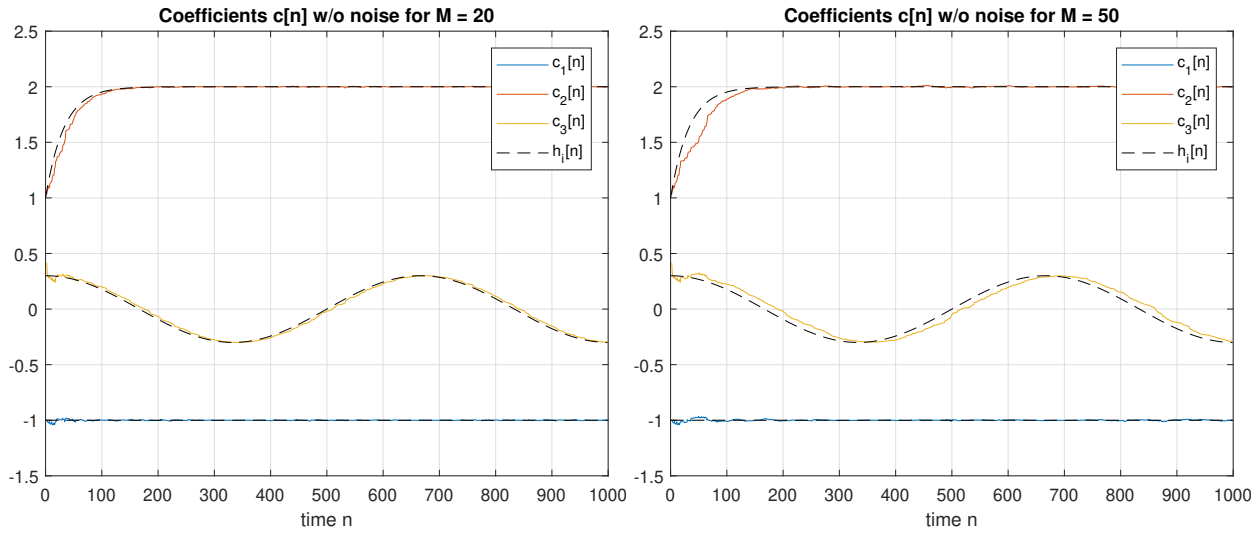


Figure 2: Coefficients for **c**[n] *without* noise and segment length M = 20 and M = 50

As we can see, the values for **h**[n] (black striped lines) correspond quite well with the coefficients which were calculated through the *least-square* sense. The first values of **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).

19.1

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.

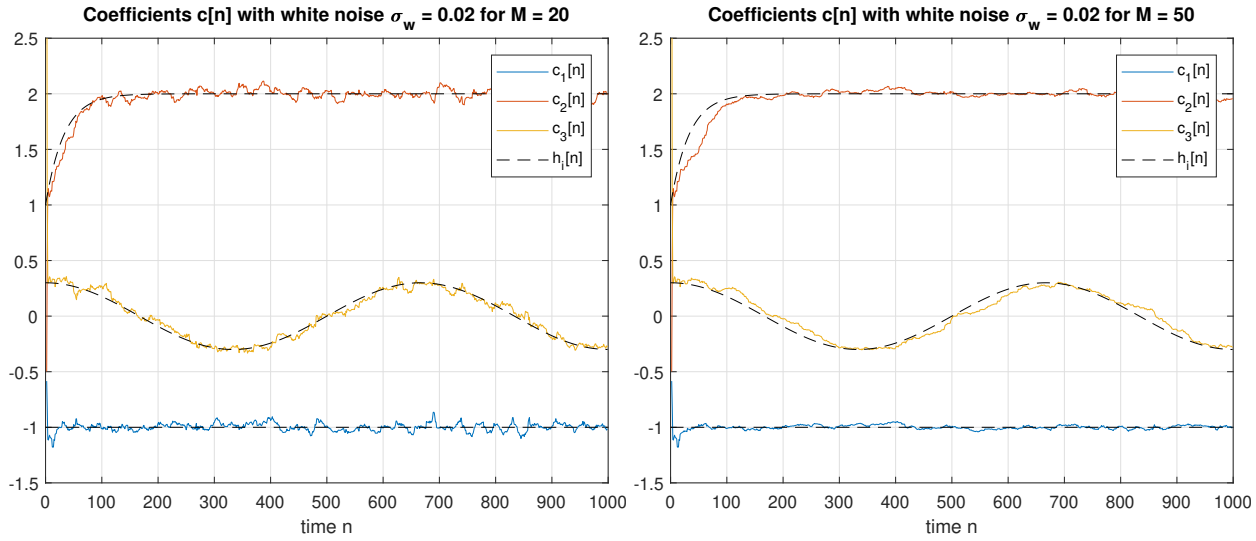


Figure 3: Coefficients for **c**[n] *with* noise and segment length M = 20 and M = 50

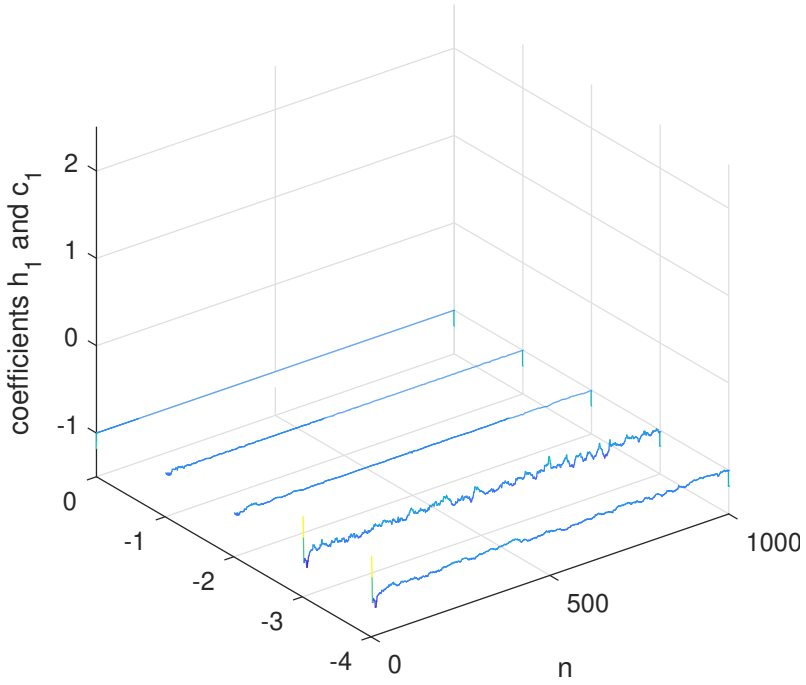
Between $M = 20$ and $M = 50$ is a lot of difference now. The additional noise $w[n]$ causes the coefficients to fluctuate. The plot with the lower segment length $M = 20$ fluctuates way more than the plot with segment length $M = 50$. Due to the property of white noise the effect it has should cancel 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:

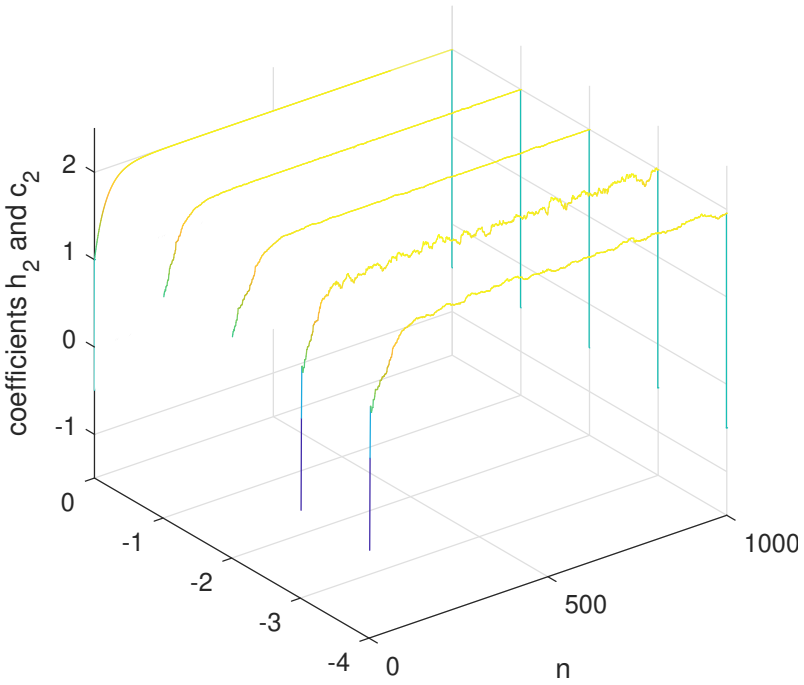
1. $h_i[n]$
2. $c_i[n]$ *without* measurement noise and for segment length $M = 20$
3. $c_i[n]$ *without* measurement noise and for segment length $M = 50$
4. $c_i[n]$ *with* measurement noise and for segment length $M = 20$
5. $c_i[n]$ *with* measurement noise and for segment length $M = 50$

where i denotes the i-th coefficient



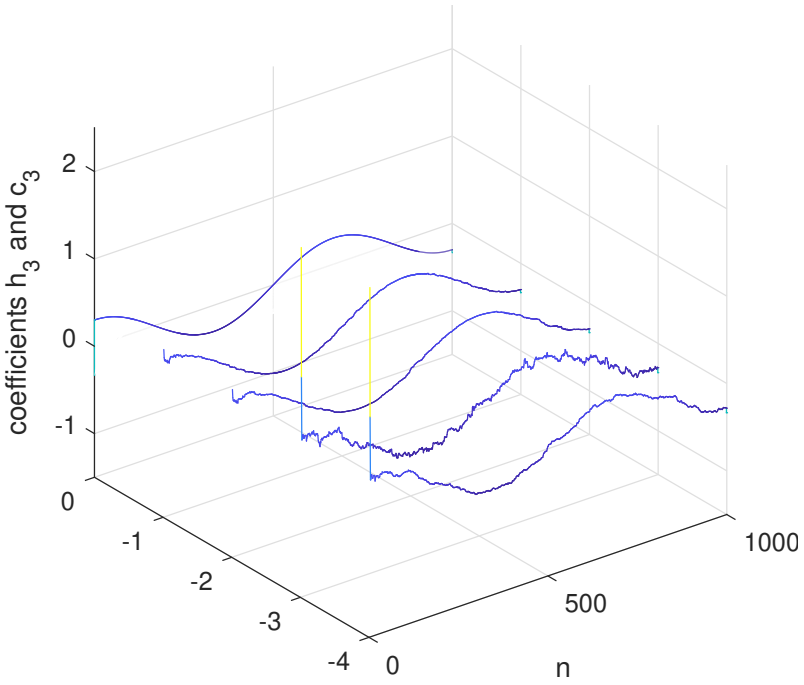
h | M = 20 & w = 0 | M = 50 & w = 0 | M = 20 & w ~= 0 | M = 50 & w ~= 0;

Figure 4: waterfall coefficients i = 1



h | M = 20 & w = 0 | M = 50 & w = 0 | M = 20 & w ~= 0 | M = 50 & w ~= 0;

Figure 5: waterfall coefficients i = 2



h | M = 20 & w = 0 | M = 50 & w = 0 | M = 20 & w ~= 0 | M = 50 & w ~= 0;

Figure 6: waterfall coefficients i = 3

3.5 Matlab Code

Main File

```
close all
clear all
clc

%suppres: "Warning: Matrix is singular to working precision."
id = 'MATLAB:singularMatrix';
warning('off',id)

%suppres: "Warning: Directory already exists."
id = 'MATLAB:MKDIR:DirectoryExists';
warning('off',id)

mkdir 'Figures' %create Figures folder



%-------------------------------------------------------------------------
% a)
N = 2;
d = [-3; -5; 0]; %values from example 1.1
x = [-1; -1; 1];




k = N-1; %Pad with N - 1 zeroes for the values of x[-1], x[-2], ... x[-N+1]
x_pad = [zeros(k,1); x];
d_pad = [zeros(k,1); d];


c = ls_filter(x_pad, d_pad, N)

%"Probe"
d_hat = conv(x,c);
d_hat = d_hat(1:length(x)) %delete the values which assume that
%                           x[n] = 0 for n > length(x)



%-------------------------------------------------------------------------
% b)
theta = 3*pi/1000;
n = 0:999;

h = [-1*ones(1,length(n)); 2-0.97.^n; 0.3*cos(theta*n)];
%h = h1[0], h1[1], ..., h1[n];
%    h2[0], h2[1], ..., h2[n];
%    h3[0], h3[1], ..., h3[n];


figure
    plot(n,h)
    legend('h_1[n]','h_2[n]','h_3[n]')
    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
                of x[n]
86
87          c = zeros(N,length(n));
88          for ii = n %ii is counts through the time n
89              c(:,ii+1) = ls_filter(x_pad(ii+1:M+ii), d_pad(ii+1:M+ii), N);
90          end
91
92
93          if w == 0
94              text = ['Coefficients c[n] w/o noise for M = ' num2str(M)];
95              text_saveas = ['Coefficients_c_without_noise_M=' num2str(M)];
96          else
97              text = ['Coefficients c[n] with white noise \sigma_w = ' num2str(
                    round(var(w),2)) ' for M = ' num2str(M)];
98              text_saveas = ['Coefficients_c_with_noise_M=' num2str(M)];
99          end
100
101         figure
102             plot(n,c)
103             hold on
104             plot(n,h,'--k')
105             legend('c_1[n]','c_2[n]','c_3[n]','h_i[n]')
106             grid on
107             title(text)
108             xlabel('time n')
109             ylim([-1.5, 2.5]) %due to some singularities, the first values
110 %                               of c can get quite big -> ruins the plot ->
```

```
%                                limit it
            saveas(gcf,['Figures/' text_saveas] ,'epsc') %epsc to save the eps
                in colour

        c_plot(:,:,counter) = c;
        counter = counter + 1;
    end  %for M

end %for jj

kk = 1;
X = [1; 1; 1; 1; 1]* n;
Y = [0; -1; -2; -3; -4]*ones(1,length(n));

for kk = 1:3
Z = [h(kk,:); c_plot(kk,:,1); c_plot(kk,:,2); c_plot(kk,:,3); c_plot(kk,:,4)];
%c_plot(kk,:,1) has the coefficients for M = 20, w = 0;
%c_plot(kk,:,2) has the coefficients for M = 50, w = 0;
%c_plot(kk,:,3) has the coefficients for M = 20, w ~= 0;
%c_plot(kk,:,4) has the coefficients for M = 50, w ~= 0;

 text_saveas = ['waterfall_coefficients_' num2str(kk)];

figure
    waterfall(X,Y,Z)
    zlim([-1.5, 2.5])
    xlabel('n')
    zlabel(['coefficients h_' num2str(kk) ' and c_' num2str(kk)])
    ylabel('h | M = 20 & w = 0 |  M = 50 & w = 0 |  M = 20 & w ~= 0 |  M = 50 &
       w  ~= 0;')
    saveas(gcf,['Figures/' text_saveas] ,'epsc') %epsc to save the eps in colour
end


%create a placeholder function to overwrite the saveas function
function saveas(~, ~, ~)
    disp('Figure not saved')
end


%seen from plots:
%w[n] = 0:
%theres only little difference between M = 20 and M = 50 and
%the values of c correspond quite well to the values of h,
%altough the plot is a bit shifted(increses with segment
%length M). The first values of c are also not quite the same
%as h, due to the assumption that x[n] = 0, for n < 0 and the
%first c[n] gets computed with only one entry which is not 0;
%
%w[n] ~= 0:
%between M = 20 and M = 50 is a lot of difference now. Probably
%due to the higher amount of samples, the noise cancels out
%(and would fully cancel for M -> infty(because white noise),
%but then the adaptiveness of system would get lost -> c[n] would
%get constant if every x[n] is taken into account)
```

Function ls_filter()

```
function c = ls_filter(x, d, N)
%computes the filter coefficients c for one time instance n, where
%corresponds to the last entry of x

% x is the input signal saved as col vector
% d is the reference signal saved as col vector
% N is the order of the filter i.e. the amount of coefficients in c

% Matrix X to compute the coefficients at time instance n
% X = [x[n-M+N],   x[n-M+N-1],  ..., x[n-M];
%      x[n-M+N+1], x[n-M+N],    ..., x[n-M+1];
%      ...                      ...,
%      x[n],       x[n-1]       ..., x[n-N+1] ];

% Make sure x and d are col vectors
% x = x(:);
% d = d(:);

if isrow(x) || isrow(d)
    error('vector x or vector d is not a column vector')
end

M = length(x); %segment length

X = zeros(M-N+1,N); %create placeholder for entries of X

for ii = 0:N-1 %for order N coefficients of c we need N cols
    X(:,ii+1) = x( (end-M+N)-ii:end-ii ); %end corresponds to current time n
end                                       %to include M-N+1 we have to substract
    -M+N

c = (X.' * X)^-1 * X.' * d(end-M+N:end); %pseudo inverse; multiply with
   segmented d

end
```

Function vector_conv()

```
function y = vector_conv(x, h)
%calcultes the concolution sum defined in Adaptive System UE
%
%x has to be a column vector in form of
% x = x[0];
%     x[1];
%       ...;
%     x[n-1]
%
%where n is the time variable


%h has to be matrix in the form of
% h = h1[0], h1[1], ..., h1[n-1];
%     h2[0], h2[1], ..., h2[n-1];
%     h3[0], h3[1], ..., h3[n-1]


x = x(:); %make sure that x is a col vector


N = size(h,1);


t = 1;
n = 0:length(x)-1;
x_zero_pad = [zeros(N-1,1); x]; %puts zeros for time x[-1], x[-2], ... x[-N+1]

y = zeros(length(n),1);
for n_shift = n + N
    x_tap_input = x_zero_pad(n_shift:-1:n_shift-N+1);
    y(t) = h(:,t)' * x_tap_input; % ' is hermitian transposed
    t = t+1;
end


end
```

Function vector_conv2()

```
 function y = vector_conv2(x, h)
% computes the convolution of x and the coefficients of h at time
% instance n for every n

 y = zeros(length(x),1);

 for n = 1:length(x)
        temp = conv(x,h(:,n));
    y(n) = temp(n);
 end

 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 coeffiecents shall be derived so that $\mathbf{c}_{wLS}[n] = \text{argmin}_{\mathbf{c}}\ J_{wLS}(\mathbf{c},n)$ according to the wegihted least squares cost function

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

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:

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

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

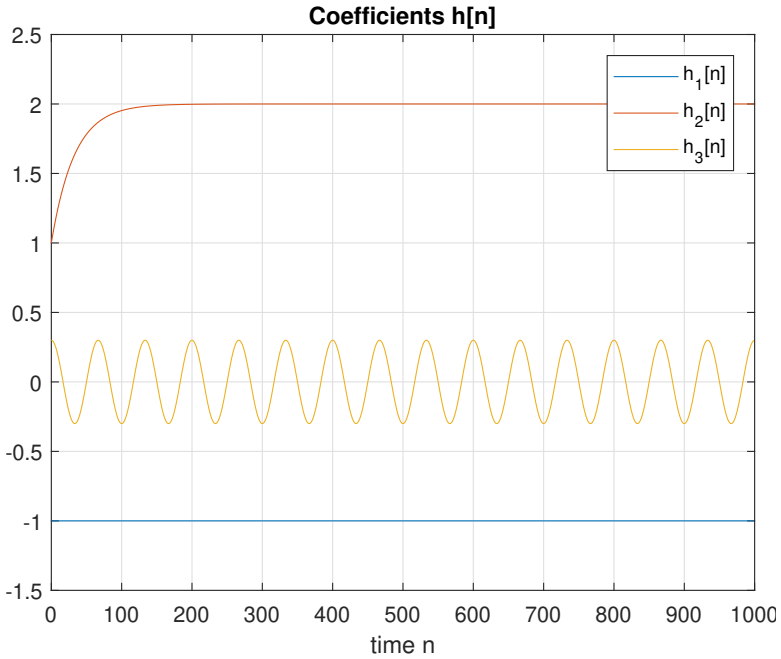


Figure 1: Coefficients for $\mathbf{h}$[n]

Problem 1.4

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

"Laufvariable"

bc $g[n-k]$

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

from problem class: $\underline{e}[n] = \underline{d}[n] - \underline{X}\cdot \underline{c}[n]$ (p. 1.01)

$$\underline{d}[n] = \begin{bmatrix} d[n] \\ 0 \\ \vdots \\ d[n-M+1] \end{bmatrix}$$

$$\underline{X} = \begin{bmatrix} x[n-M+1] & \cdots & x[n-M+1-M] \\ \vdots & & \vdots \\ x[n] & \cdots & x[n-M] \end{bmatrix}$$

$$J_{WLS}(c,n) = \underline{e}^T[n]\cdot \underline{G}\cdot \underline{e}[n] = (\underline{d}-\underline{X}\cdot \underline{c})^T\cdot \underline{G}\cdot(\underline{d}-\underline{X}\cdot \underline{c})$$

$$= (\underline{d}^T - \underline{c}^T\cdot \underline{X}^T)\cdot \underline{G}\cdot(\underline{d}-\underline{X}\cdot \underline{c}) =$$

$$= (\underline{d}^T\cdot \underline{G} - \underline{c}^T\cdot \underline{X}^T\cdot \underline{G})\cdot(\underline{d}-\underline{X}\cdot \underline{c}) =$$

$$= (\underline{d}^T\underline{G}\underline{d} - \underline{c}^T\cdot \underline{X}^T\cdot \underline{G}\cdot \underline{d} - (\underline{d}^T\cdot \underline{G}\cdot \underline{X}\cdot \underline{c})^T + \underline{c}^T\cdot \underline{X}^T\cdot \underline{G}\cdot \underline{X}\cdot \underline{c})$$

(scalar)

$$= \underline{d}^T\cdot \underline{G}\cdot \underline{d} - \underline{c}^T\cdot \underline{X}^T\cdot \underline{G}\cdot \underline{d} - \underline{c}^T\cdot \underline{X}^T\cdot \underline{G}^T\cdot \underline{d} + \underline{c}^T\cdot \underline{X}^T\cdot \underline{G}\cdot \underline{X}\cdot \underline{c} \quad // \ \underline{G}^T = \underline{G}$$

vector is in front

$$= \underline{d}^T\cdot \underline{G}\cdot \underline{d} - \underline{c}^T\cdot \underline{X}^T\cdot \underline{G}\cdot \underline{d} - \underline{c}^T\cdot \underline{X}^T\cdot \underline{G}\cdot \underline{X}\cdot \underline{c}$$

(constant)

$$\nabla_{\underline{c}}(J_{WLS}) = 0 - \underline{X}^T\cdot \underline{G}\cdot \underline{d} - \underline{X}^T\cdot \underline{G}\cdot \underline{X}\cdot \underline{c} \overset{!}{=} 0$$

$$\underline{X}^T\cdot \underline{G}\cdot \underline{X}\cdot \underline{c} = \underline{X}^T\cdot \underline{G}\cdot \underline{d}$$

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

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

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:

$$g[m] = \lambda^{-m} \tag{3}$$

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.

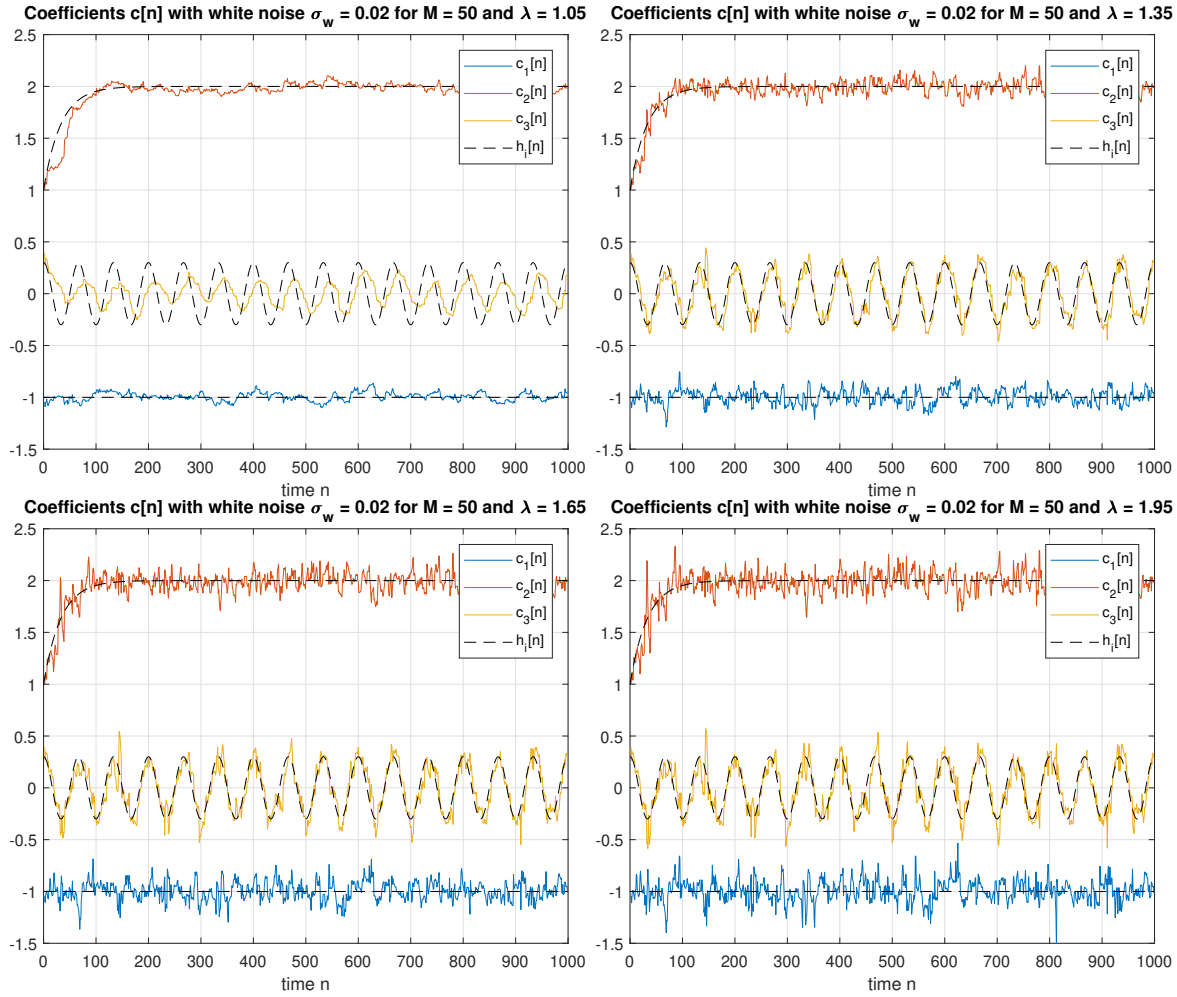


Figure 2: Coefficients for **c**[n] with noise, segment length M = 50 and varying λ

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 $\mathbf{h}[n]$ and the calculated $\mathbf{c}[n]$, which was caused by the segment length M.
A reasonable choice for λ seems to be $\lambda = 1.35$ because the delay between $\mathbf{h}[n]$ and $\mathbf{c}[n]$ is marginal and the noise doesnt seem too bad.

4.4 Matlab Code

Main File

```
close all
clear all
clc

%suppres: "Warning: Matrix is singular to working precision."
id = 'MATLAB:singularMatrix';
warning('off',id)

%suppres: "Warning: Directory already exists."
id = 'MATLAB:MKDIR:DirectoryExists';
warning('off',id)

mkdir 'Figures'

%-------------------------------------------------------------------------
% calculate h
theta = 3*pi/100; %change theta from problem 1.3
n = 0:999;

h = [-1*ones(1,length(n)); 2-0.97.^n; 0.3*cos(theta*n)];
%h = h1[0], h1[1], ..., h1[n];
%    h2[0], h2[1], ..., h2[n];
%    h3[0], h3[1], ..., h3[n];


figure
    plot(n,h)
    legend('h_1[n]','h_2[n]','h_3[n]')
    grid on
    ylim([-1.5 2.5])
    title('Coefficients h[n]')
    xlabel('time n')

    saveas(gcf,'Figures/Coefficients_h', 'epsc')

%-------------------------------------------------------------------------
% weighted system identification
N = 3; %3 filter coefficients in h and c

x = randn(1,length(n)).'; %x[n] = 0 for n < 0 (or 1 in matlab)

d = vector_conv(x, h);

%create white gaussian noise and change variance
w = transpose(randn(1,length(n)))./(1/sqrt(0.02));

d = d + w;% add noise after filter h


for lambda = 1.05:0.3:2

    for M = [50]
        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
                of x[n]
55
56          c = zeros(N, length(n));
57          for ii = n %ii is counts through the time n
58              c(:,ii+1) = ls_filter_weighted(x_pad(ii+1:M+ii), d_pad(ii+1:M+ii), N
                    , lambda);
59          end
60
61
62          text = ['Coefficients c[n] with white noise \sigma_w = ' num2str(round(
                var(w),2)) ' for M = ' num2str(M) ' and \lambda = ' num2str(lambda)];
63          text_saveas = ['Coefficients_c_with_noise_M=' num2str(M) '_and_lambda='
                strrep(num2str(lambda),'.','_')];
64
65
66          figure
67              plot(n,c)
68              hold on
69              plot(n,h, '--k')
70              legend('c_1[n]','c_2[n]','c_3[n]', 'h_i[n]')
71              grid on
72              title(text)
73              xlabel('time n')
74              ylim([-1.5, 2.5])  %due to some singularities, the first values
75      %                           of c can get quite big -> ruins the plot ->
76      %                           limit it
77              saveas(gcf,['Figures/' text_saveas] ,'epsc') %epsc to save the eps
                    in colour
78
79
80      end  %for M
81
82  end %for lambda
83
84  %create a placeholder function to overwrite the saveas function
85  function saveas(~, ~, ~)
86      disp('Figure not saved')
87  end
```

Function ls_filter_weighted()

```
function c = ls_filter_weighted(x, d, N, lambda)
%computes the filter coefficients c for one time instance n, where
%corresponds to the last entry of x

% x is the input signal saved as col vector
% d is the reference signal saved as col vector
% N is the order of the filter i.e. the amount of coefficients in c

% Matrix X to compute the coefficients at time instance n
% X = [x[n-M+N],   x[n-M+N-1],  ..., x[n-M];
%      x[n-M+N+1], x[n-M+N],    ..., x[n-M+1];
%      ...                      ...,
%      x[n],       x[n-1]       ..., x[n-N+1] ];

% Make sure x and d are col vectors
% x = x(:);
% d = d(:);

if isrow(x) || isrow(d)
    error('vector x or vector d is not a column vector')
end

M = length(x); %segment length

X = zeros(M-N+1,N); %create placeholder for entries of X

for ii = 0:N-1 %for order N coefficients of c we need N cols
    X(:,ii+1) = x( (end-M+N)-ii:end-ii ); %end corresponds to current time n
end                                       %to include M-N+1 we have to substract
    -M+N


%compute the weighting matrix
% lambda = 1.5;
m = 0:-1:-(M-N);
g = lambda.^m;
G = diag(flip(g)); %flip because of form of matrix G (from g[n - k])

%compute coefficients with weighted input

c = (X.' * G * X)^-1 * X.' * G * d(end-M+N:end);

end
```

Function vector_conv()

```
function y = vector_conv(x, h)
%calcultes the concolution sum defined in Adaptive System UE
%
%x has to be a column vector in form of
% x = x[0];
%     x[1];
%       ...;
%     x[n-1]
%
%where n is the time variable


%h has to be matrix in the form of
% h = h1[0], h1[1], ..., h1[n-1];
%     h2[0], h2[1], ..., h2[n-1];
%     h3[0], h3[1], ..., h3[n-1]


x = x(:); %make sure that x is a col vector


N = size(h,1);


t = 1;
n = 0:length(x)-1;
x_zero_pad = [zeros(N-1,1); x]; %puts zeros for time x[-1], x[-2], ... x[-N+1]

y = zeros(length(n),1);
for n_shift = n + N
    x_tap_input = x_zero_pad(n_shift:-1:n_shift-N+1);
    y(t) = h(:,t)' * x_tap_input; % ' is hermitian transposed
    t = t+1;
end


end
```

Index der Kommentare

16.1 You are completely correct here, but... for (d), if you knew the sensors' imperfections, can't you compensate for their effects? I'm not talking about taking an inverse (IIR), but rather something simpler... below is a simple explanation:

In cases (a) and (b), even if the sensors' imperfections where to be unknown, you were still able to identify your filter H(z). You calculated c_opt yourself without changing anything about the given system. So, what you learned from here is that, if the imperfections in the sensors are the same in both branches, then you will be able to identify your filter H(z) even if you didn't know the imperfections! The important part here is that the order of the cascaded filter in the adaptive branch must equal or be larger than the order of the cascaded filer in the system branch (this was the case in (a) and (b)).

Keep in mind, we are using here white noise to excite the filters, and these imperfections will filter some frequencies from it, thus you will not be able to identify all types of filters, especially if the imperfections are cutting out relevant frequencies.

In cases (c) and (d), if we knew these imperfections (i.e., s1 and s2) then we can always compensate for each another by modifying the cascaded filters as:
H(z)S2(z)S1(z) and S1(z)C(z)S2(z)
Therefore, you get identical effects on both branches (similar to (b)), and the order of the cascaded filters is not a problem anymore! (adaptive branch > system branch). But as mentioned before with the frequencies... not all types of filters can be identified!

If you are interested, try solving (d), but now use s1 = s2 = [1 0 1]', or even more complex, s1 = s2 = [1 2 3 4 5]'. If you do everything correct, you should get c_opt = h. But have a look now on J_min, you will notice an increase in the cost function (compared to (a) and (b)). This is because your input signal is not 'white' anymore due to filtering, i.e., not i.i.d. anymore.

17.1 5/5P

17.2 1/1P

18.1 4/4P

19.1 2/2P