Adaptive Systems

Assignment 3

Autor: Ebner Thomas (0831246), Nöhmer Stefan (0830668)

Datum: Graz, 27. Januar 2012
Version.: alpha 1.0

1 Analytical Problem 3.1: Predictive Encoding of an AR Process

a) Die Autokorrelationsfaktoren ergeben sich wie folgt:
Dabei verschwindet $E\{w[n]u[n-k]\}$ für $k > 0$, da das weisse Rauschen unkorreliert mit $u[n-k]$ für $k > 0$ ist!

$$r_{uu}[k] = E\{u[n]u[n-k]\} \tag{1.1}$$
$$= E\{(w[n] + u[n-1] - 1/8u[n-2])((w[n-k] + u[n-1-k] - 1/8u[n-2-k])\} \tag{1.2}$$
$$= E\{\underbrace{w[n]u[n-k]}_{0} + \underbrace{u[n-1]u[n-k]}_{r_{uu}[k-1]} - 1/8\underbrace{u[n-2]u[n-k]}_{r_{uu}[k-2]} \tag{1.3}$$

$$r_{uu}[0] = Eu^2[n] = \sigma_u^2 = 1 \tag{1.4}$$
$$r_{uu}[1] = r_{uu}[0] - 1/8\underbrace{r_{uu}[1-2]}_{=r_{uu}[1]} = 8/9 \tag{1.5}$$
$$r_{uu}[2] = r_{uu}[1] - 1/8r_{uu}[0] = 55/72 \tag{1.6}$$

$$r_{uu} = E\{(w[n] + u[n-1] - 1/8u[n-2])^2\} \tag{1.7}$$
$$= E\{\underbrace{w^2[n]}_{\sigma_w^2} + \underbrace{w[n]u[n-1]}_{0} - 1/8\underbrace{w[n]u[n-2]}_{0} \tag{1.8}$$
$$+ \underbrace{w[n]u[n-1]}_{0} + \underbrace{u^2[n-1]}_{\sigma_u^2} - 1/8\underbrace{u[n-1]u[n-2]}_{r_{uu}[1]} \tag{1.9}$$
$$- 1/8\underbrace{w[n]u[n-2]}_{0} - 1/8\underbrace{u[n-1]u[n-2]}_{r_{uu}[1]} + 1/64\underbrace{u^2[n-2]}_{\sigma_u^2}\} \tag{1.10}$$
$$r_{uu}[0] = \sigma_u^2 = \sigma_w^2 + \sigma_u^2 - 2/8r_{uu}[1] + 1/64\sigma_u^2 \tag{1.11}$$
$$\Rightarrow \sigma_r = 2/9 - 1/64 = 119/576 \tag{1.12}$$

Der Noisegain G_G berechnet sich wie folgt:

$$G_G = \frac{\sigma_u^2}{\sigma_w^2} = 576/119 \tag{1.13}$$

b) Die Jule Walker Gleichung ist wie folgt definiert:

$$\underline{c}_{MSE} = R_{uu}^{-1} \cdot \begin{bmatrix} r_{uu}[1] \\ r_{uu}[2] \\ \ldots \\ r_{uu}[N] \end{bmatrix} \tag{1.14}$$

für N=1 ergibt sich somit für c_{MSE}

$$c_{MSE} = c_0 = \frac{1}{r_{uu}[0]} \cdot r_{uu}[1] = 8/9 \tag{1.15}$$

Um nun die Varianz des Fehlers $e[n]$ zu berechnen wird wieder einfach in $E\{e^2[n]\}$ eingesetzt:

$$\begin{aligned} e[n] &= u[n] - u[n-1] \cdot 8/9 && (1.16)\\ &= w[n] + 1/9u[n-1] - 1/8u[n-2] && (1.17) \end{aligned}$$

$$\begin{aligned} E\{e^2[n]\} = E\{&\underbrace{w^2[n]}_{\sigma_w^2} + 1/8\underbrace{u[n-2]w[n]}_{0} - 1/9\underbrace{w[n]u[n-1]}_{0} + 1/81\underbrace{u^2[n-1]}_{\sigma_u^2} && (1.18)\\ &- 1/72\underbrace{u[n-1]u[n-2]}_{r_{uu}[1]} - \underbrace{w[n]1/8u[n-2]}_{0} - 1/72\underbrace{u[n-1]u[n-2]}_{r_{uu}[1]} && (1.19)\\ &+ 1/64\underbrace{u^2[n-2]}_{\sigma_u^2}\} && (1.20)\\ &= \sigma_w^2 + 1/81\sigma_u^2 - 1/36r_{uu}[1] + 1/64\sigma_u^2 && (1.21)\\ &= 17/81 && (1.22) \end{aligned}$$

Der Prediction Gain G_C berechnet sich wie folgt:

$$G_C = \frac{\sigma_u^2}{\sigma_e^2} = 81/17 \tag{1.23}$$

Der verwendete lineare Predictor ist nicht optimal, da das Signal $u[n]$ von einem AR prozess stammt der aus Weissem Rauschen mit einem Filter 1. Ordnung erzeugt wurde. Somit reicht ein linearer Predictor 1. Ordnung nicht aus. Am Signal $e[n]$ erkennt man auch, dass das Signal noch kein weisses rauschen ist.

c) Für N=2 wurde die gleiche Vorhergehensweise verwendet:

$$\underline{c}_{MSE} = R_{uu}^{-1} \cdot \begin{bmatrix} r_{uu}[1] \\ r_{uu}[2] \\ \dots \\ r_{uu}[N] \end{bmatrix} = \begin{bmatrix} 1 & 8/9 \\ 8/9 & 1 \end{bmatrix}^{-} 1 \cdot \begin{bmatrix} 8/9 \\ 55/72 \end{bmatrix} = \begin{bmatrix} 1 \\ -1/8 \end{bmatrix} \tag{1.24}$$

$$e[n] = u[n] - 1 \cdot u[n-1] + 1/8 \cdot u[n-2] = w[n] \tag{1.25}$$

$$G_C = \frac{\sigma_u^2}{\sigma_e^2} = \frac{1}{\sigma_w^2} = \frac{576}{119} \tag{1.26}$$

Der verwendete Predictor ist nun optimal, da er aus dem AR prozess wieder das weisse rauschen gewinnt. Eine bessere Vorhersage ist nicht mehr möglich, da weisses Rauschen ja nicht vorherhersehbar ist.

d) N=1: der Prediction Error filter besitzt folgende Differenzengleichung:

$$e[n] = u[n] - 8/9u[n-1] \tag{1.27}$$

Somit ergibt sich für die Übertragungsfunktion:

$$H_e(z) = 1 - 8/9z^{-1} \tag{1.28}$$

Für $S(z)$ wird einfach das Inverse von $H_e(z)$ ermittelt:

$$S(z) = \frac{1}{1 - 8/9z^{-1}} \tag{1.29}$$

Um den Noisegain zu ermitteln wird das Quadrat der Norm der Impulsantwort des Filters $S(z)$ ermittelt:

$$G_s = ||g_s[n]||^2 = \sum_{i=0}^{\inf} (8/9)^{2n} = \frac{1}{1 - 64/81} = \frac{81}{17} \tag{1.30}$$

Da es sich bei dem Input des Filters S(z) um kein weisses Rauschen handelt, ist keine Berechnung von $\sigma_{\hat{u}}^2$ möglich. ($e[n]$ ist kein weisses Rauschen!)
N=2: der Prediction Error filter besitzt folgende Differenzengleichung:

$$e[n] = u[n] - u[n-1] - u[n-1] + 1/8 \cdot u[n-2] \tag{1.31}$$

Somit ergibt sich für die Übertragungsfunktion:

$$H_e(z) = 1 - z^{-1} + 1/8z^{-2} \tag{1.32}$$

Für $S(z)$ wird einfach das Inverse von $H_e(z)$ ermittelt:

$$S(z) = \frac{1}{1 - z^{-1} + 1/8z^{-2}} \tag{1.33}$$

Für die Ermittelung des Noisegains wurde die Tatsache ausgenutzt, dass der Filter $S(z)$ genau das Inverse von $H_e(z)$ darstellt. Der Noisegain von $H_e(z)$ wurde bereits in Aufgabe 3.1.a) ermittelt.
Dadurch ergibt sich nun folgender Noisegain:

$$G_s = \frac{576}{119} \tag{1.34}$$

Da bei N=1 das Eingangssignal des Filters $S(z)$ weisses Rauschen ist, ist eine Berechnung von $\sigma_{\hat{u}}^2$ mittels G_s möglich.

e) Um das Signal $\hat{u}[n]$ verwenden wir zunächst eine Darstellung im Z-Bereich. Weiters verwenden wir auch die Tatsache, dass für die Ermittelung von $S(z)$ das Inverse des Prediction-error-filters verwendet wurde. Somit ist:

$$S(z) = \frac{U(z)}{E(z)} \tag{1.35}$$

$$\hat{U}[n] = \hat{E}(z) \cdot S(z) \tag{1.36}$$
$$= (E(z) + Q(z)) \cdot g \cdot S(z) \tag{1.37}$$
$$= (E(z) + Q(z)) \cdot g \cdot \frac{U(z)}{E(z)} \tag{1.38}$$
$$= (U(z) + Q(z) \cdot S(z)) \cdot g \tag{1.39}$$

Wieder in den Zeitbereich zurücktransformiert ergibt das:

$$\hat{u}[n] = g \cdot u[n] + g \cdot q[n] * s[n] \tag{1.40}$$

$$r[n] = \hat{u}[n] - u[n] = (1-g)u[n] + g \cdot q[n] * s[n] \tag{1.41}$$

$$E\{r^2[n]\} = E\{(1-g)^2u^2[n] + \underbrace{2(1-g)u[n]g \cdot q[n] * s[n]}_{=0, da unkorreliert} + \underbrace{(g \cdot q[n] * s[n])^2}_{g^2 v \sigma_q^2 v ||s[n]||^2}\} \tag{1.42}$$
$$= (1-g)^2 \cdot \sigma_u^2 + g^2\sigma_q^2||s[n]||^2 \tag{1.43}$$
$$= (1-g)^2 \cdot \sigma_u^2 + g^2\sigma_q^2 G_c \tag{1.44}$$

$$\sigma_q^2 = \frac{\sigma_e^2}{2^{2R}-1} = \sigma_u^2/G_c \cdot \frac{1}{2^{2R}-1} \tag{1.45}$$

$$g = \frac{\sigma_e^2}{\sigma_e^2 + \sigma_q^2} = \frac{\sigma_u^2/G_C}{\sigma_u^2/G_C + \sigma_u^2/G_c\frac{1}{2^{2R}-1}} = \frac{1}{1+\frac{1}{2^{2R}-1}} = \frac{2^{2R}-1}{2^{2R}} \tag{1.46}$$

$$\sigma_r^2 = [(\frac{2^{2R}-1}{2^{2R}} - 1)^2 + \frac{(2^{2R}-1)^2}{2^{4R}}\frac{1}{G_c}\frac{G_c}{2^{2R}-1}]\sigma_u^2 \tag{1.47}$$
$$= [(\frac{-1}{2^{2R}})^2 + \frac{2^{2R}-1}{2^{4R}}]\sigma_u^2 \tag{1.48}$$
$$= \frac{\sigma_u^2}{2^{2R}} \tag{1.49}$$

f) Bei einer Bitrate von 1 Bit / Sample ergibt sich nun folgende Varianz von $r[n]$ bei N=1 und N=2:

$$\sigma_r^2 = \frac{\sigma_u^2}{4} \tag{1.50}$$

Nun wird Die Varianz von $r[n]$ bei direkter Codierung berechnet. Um die Varianz von $r[n]$ zu berechnen wurde ein formel aus der Signalverarbeitungs-VO für den SNR bei einer quantisierung verwendet:

$$SNR = \frac{\sigma_u^2}{\sigma_q^2} = 2^2 = 4 \tag{1.51}$$

Somit ergit sich für σ_q^2:

$$\sigma_q^2 = \frac{\sigma_u^2}{4} \tag{1.52}$$

$$r[n] = \hat{u}[n] - u[n] = q[n] \tag{1.53}$$

$$\sigma_r^2 = \sigma_q^2 = \frac{\sigma_u^2}{4} \tag{1.54}$$

Somit ergibt sich für direkte Codierung und linear Predictive Coding der Die gleiche varianz des Rekonstruktionsfehlers. =¿ kein Gain bei der Verwendung von Prediction!

2 Matlab Problem 3.2: Information vs. Energy

Zur Berechnung der Ausgangswerte des Systems wurde das System mit dem vorgegebenen Quantizer implementiert (siehe Listings) und für 10^6 Samples mit 1 bis 5 Bit Simulationen durchgeführt. Der Quantisierer führt teilt dabei den Eingangsbereich in eine der Anzahl an Bit entsprechenden Anzahl Stufen und rundet das Eingangssignal auf die nächstliegende Stufe.
Mit der vorgegebenen Funktion `CondEntropy` wurden die bedingten Entropien berechnet, welche dem Informationsgehalt eines Wertes entspricht, wenn die vorherigen Werte bekannt sind.
Darstellungen 2.1, 2.2 und 2.3 stellen die bedingten Entropien bei unterschiedlichen N dar.
Bei $N = 0$ (Abb. 2.1) sinkt die Entropie nach dem ersten Zeichen schnell, und danach langsamer. Führt man *Linear Prediction* ein mit $N = 1$ (Abb. 2.2), so bleibt die Entropie weitgehend erhalten und sinkt nur langsam, der Informationsgehalt der übertragenen Werte bleibt also höher. Erhöht man N auf 2 (Abb. 2.3), so beginnt die Entropie wieder zu sinken, und zwar sogar schneller als im Fall $N = 0$.

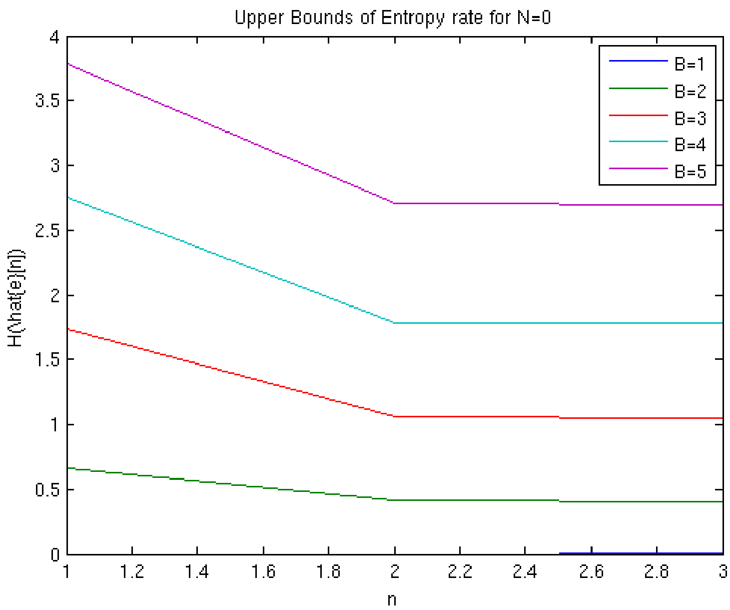

Abbildung 2.1: Entropie für $N = 0$

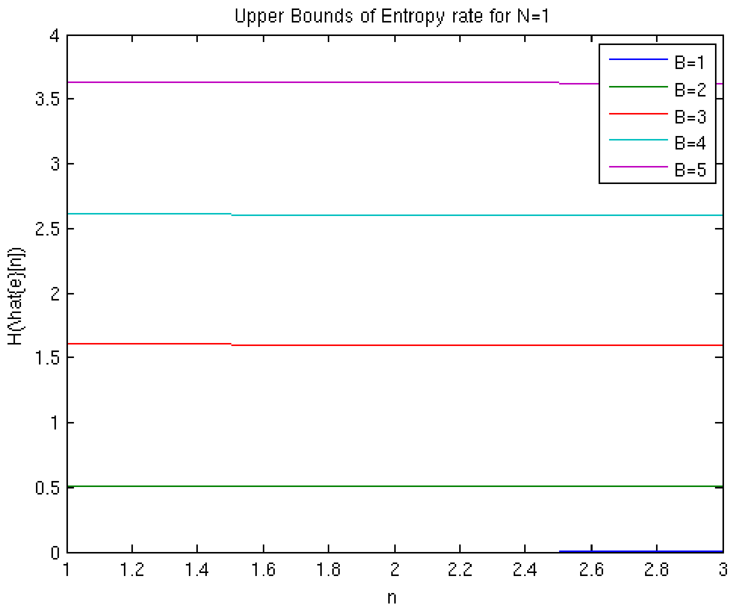

Abbildung 2.2: Entropie für $N = 1$

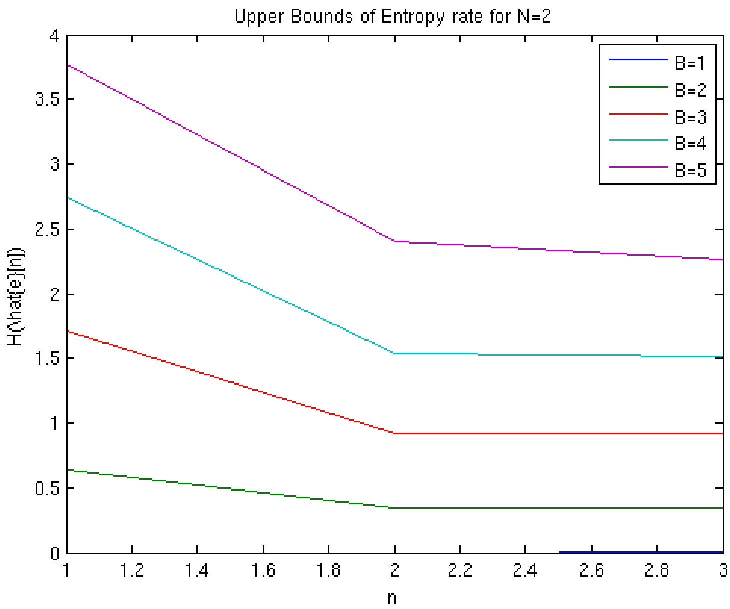

Abbildung 2.3: Entropie für $N = 2$

3 Analytical Problem 3.3: Least Squares IIR Identification

Wir kennen das IIR-System $u[n] = v[n] + u[n-1] - \frac{1}{8}u[n-2]$, von dem M Input-Output-Samples vorhanden sind. Für die folgende Differenzengleichung sollen die Parameter a_k und b_k bestimmt werden, wobei die Anzahl der Parameter $N = N_a + N_b + 1$ ist und $M \geq N$ ist:

$$u[n] = \sum_{k=0}^{N_b} b_k v[n-k] - \sum_{k=1}^{N_a} a_k u[n-k] \tag{3.1}$$

a) Es soll eine LS-Lösung (*least squares*) für die Parameter gefunden werden.
Dazu wird zunächst ein Parametervektor $\mathbf{c} = \begin{bmatrix} \mathbf{b} \\ -\mathbf{a} \end{bmatrix}$ eingeführt.

Weiters muss ein *tap input/output vector* $\mathbf{x}[n] = \begin{bmatrix} \mathbf{v}[n] \\ \mathbf{u}[n] \end{bmatrix}$ eingeführt werden, mit $\mathbf{v}[n] = \begin{bmatrix} v[n-0] \\ v[n-1] \\ \vdots \\ v[n-N_b] \end{bmatrix}$

und $\mathbf{u}[n] = \begin{bmatrix} u[n-1] \\ u[n-2] \\ \vdots \\ u[n-N_a] \end{bmatrix}$.

Für die LS-Solution gehen wir nun wie gewohnt vor. Der quadratische Fehler soll minimiert werden ($e[n]$ ist der Fehler, $d[n]$ das gewünschte Signal). Dazu wird eine einfache Gleichung aufgestellt und in Vektorschreibweise übergeführt:

$$e[n] = d[n] - y[n] = d[n] - \mathbf{c}^T\mathbf{x}[n] = d[n] - \mathbf{x}^T[n]\mathbf{c} \quad \Rightarrow \mathbf{e} = \mathbf{d} - \mathbf{X}\mathbf{c} \tag{3.2}$$

$\mathbf{X}$ ist die Designmatrix mit folgender Struktur:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T[0] \\ \mathbf{x}^T[1] \\ \vdots \\ \mathbf{x}^T[M] \end{bmatrix} \tag{3.3}$$

Jetzt bestimmen wir die LS-Solution wie gehabt:

$$\begin{aligned} J(\mathbf{c}) &= \sum_{n=0}^{M} |e[n]|^2 = ||\mathbf{e}||^2 = \langle \mathbf{e}, \mathbf{e} \rangle = \mathbf{e}^T\mathbf{e} = (\mathbf{d} - \mathbf{X}\mathbf{c})^T(\mathbf{d} - \mathbf{X}\mathbf{c}) = (\mathbf{d}^T - \mathbf{c}^T\mathbf{X}^T)(\mathbf{d} - \mathbf{X}\mathbf{c}) && (3.4) \\ &= \mathbf{d}^T\mathbf{d} - \mathbf{d}^T\mathbf{X}\mathbf{c} - \underbrace{\mathbf{c}^T\mathbf{X}^T\mathbf{d}}_{\mathbf{d}^T\mathbf{X}\mathbf{c}} + \mathbf{c}^T\mathbf{X}^T\mathbf{X}\mathbf{c} = \mathbf{d}^T\mathbf{d} - 2\mathbf{d}^T\mathbf{X}\mathbf{c} + \mathbf{c}^T\mathbf{X}^T\mathbf{X}\mathbf{c} && (3.5) \end{aligned}$$

Die Lösung ergibt sich durch Ableiten und Null setzen der Kostenfunktion:

$$\begin{aligned}
\nabla_{\mathbf{c}} J(\mathbf{c}) &= 0 - 2\mathbf{d}^T\mathbf{X} + 2\mathbf{c}^T\mathbf{X}^T\mathbf{X} \overset{!}{=} 0 && (3.6)\\
&\Rightarrow \mathbf{d}^T\mathbf{X} \overset{!}{=} \mathbf{c}^T\mathbf{X}^T\mathbf{X} && (3.7)\\
&\Rightarrow \mathbf{X}^T\mathbf{d} \overset{!}{=} \mathbf{X}^T\mathbf{X}^T\mathbf{c} && (3.8)\\
&\Rightarrow \mathbf{c}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{d} && (3.9)
\end{aligned}$$

Aus dieser Lösung kann man die Parameter $\{a_k\}$ und $\{b_k\}$ ablesen (siehe oben).
Diese Lösung entspricht der LS-Solution für FIR-Filter! **X** und **d** haben jedoch wie oben beschrieben eine spezielle Form *:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T[0] \\ \mathbf{x}^T[1] \\ \vdots \\ \mathbf{x}^T[M] \end{bmatrix} = \begin{bmatrix} v[0] & v[-1] & v[-2] & \cdots & v[-N_b] & u[-1] & u[-2] & \cdots & u[-N_a] \\ v[1] & v[0] & v[-1] & \cdots & v[-(N_b+1)] & u[0] & u[-1] & \cdots & u[-(N_a+1)] \\ v[2] & v[1] & v[0] & \cdots & v[-(N_b+2)] & u[1] & u[0] & \cdots & u[-(N_a+2)] \\ \vdots & \ddots & & & & & & & \end{bmatrix} \tag{3.10}$$

Das gewünschte Signal $d[n]$ entspricht dem Ausgangssignal des IIR-Filters:

$$d[n] = v[n] + d[n-1] - \frac{1}{8}d[n-2] \quad \Rightarrow \mathbf{d} = \begin{bmatrix} v[0] + 0 - \frac{1}{8}0 \\ v[1] + v[0] - \frac{1}{8}0 \\ v[2] + (v[1] + v[0]) - \frac{1}{8}v[0] \\ v[3] + (v[2] + v[1] + v[0] - \frac{1}{8}v[0]) - \frac{1}{8}(v[1] + v[0]) \\ \vdots \end{bmatrix} \tag{3.11}$$

b) Nun soll für die Eingangsfolge $\{v[n]\} = \{1, 0, 0\}$ die Ausgangsfolge $\{u[n]\}$ berechnet werden.
Für die Anzahl der Parameter gilt $N_a = 1, N_b = 0$, es gibt also nur 2 Parameter: a_1 und b_0.
Wir evaluieren nun die Differenzengleichung für ein paar Iterationen (die Werte werden später benötigt):

$$\begin{aligned}
u[n] &= v[n] + u[n-1] - \frac{1}{8}u[n-2] && (3.12)\\
\Rightarrow u[0] &= v[0] + u[-1] - \frac{1}{8}u[-2] = v[0] + 0 - \frac{1}{8}0 = v[0] = 1 && (3.13)\\
u[1] &= v[1] + u[0] - \frac{1}{8}u[-1] = 1 && (3.14)\\
u[2] &= v[2] + u[1] - \frac{1}{8}u[0] = 0 + 1 - \frac{1}{8}1 = \frac{7}{8} && (3.15)\\
u[3] &= v[3] + u[2] - \frac{1}{8}u[1] = 0 + \frac{7}{8} - \frac{1}{8}1 = \frac{3}{4} && (3.16)\\
u[4] &= v[4] + u[3] - \frac{1}{8}u[2] = 0 + \frac{3}{4} - \frac{1}{8}\frac{7}{8} = \frac{41}{64} && (3.17)\\
u[5] &= v[5] + u[4] - \frac{1}{8}u[3] = 0 + \frac{41}{64} - \frac{1}{8}\frac{3}{4} = \frac{35}{64} && (3.18)\\
&\vdots && (3.19)
\end{aligned}$$

* für die nicht vorhandenen Werte (z.B. $v[-1]$) wird 0 eingesetzt.

Mit diesen berechneten Werten kann man nun die Matrix $\mathbf{X}$ und den Vektor $\mathbf{d}$ aufstellen (für die vorher berechneten 5 Iterationen):

$$\mathbf{X} = \begin{bmatrix} v[0] & u[-1] \\ v[1] & u[0] \\ v[2] & u[1] \\ v[3] & u[2] \\ v[4] & u[3] \\ v[5] & u[4] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & \frac{7}{8} \\ 0 & \frac{3}{4} \\ 0 & \frac{41}{64} \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} u[0] \\ u[1] \\ u[2] \\ u[3] \\ u[4] \\ u[5] \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \frac{7}{8} \\ \frac{3}{4} \\ \frac{41}{64} \\ \frac{35}{64} \end{bmatrix} \tag{3.20}$$

Mit der oben aufgestellten Formel für $\mathbf{c}_{LS}$ berechnen wir nun die Parameter a_1 und b_0 (in Matlab):

$$\mathbf{c}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{d} \stackrel{Matlab}{=} \begin{bmatrix} 1 \\ 0.8993 \end{bmatrix} \Rightarrow b_0 = 1, a_1 = -0.8993 \tag{3.21}$$

c) Verglichen mit dem Modell aus Problem 3.1 mit N=1 stimmen die Parameter gut überein. Im Problem 3.3 kommt es jedoch stark darauf an, wie viele Iterationen für die Berechnung von $\mathbf{c}_{LS}$ verwendet werden. Je mehr Iterationen, desto genauer wird das Ergebnis (desto genauer kommt es an das Ergebnis von Problem 3.1 heran). Das liegt in der Natur des IIR-Filters, da es sehr lange dauert, bis die Ausgangswerte des IIR-Filters auf vernachlässigbar kleine Werte gesunken sind.
Zur Bestimmung der Frequence response bringen wir die Differenzengleichung zuerst in den z-Bereich:

$$u[n] = b_0 v[n] - a_1 u[n-1] \quad \Leftrightarrow \quad U(z) = b_0 V(z) - a_1 U(z) z^{-1} \tag{3.22}$$

Nun stellen wir die Übertragungsfunktion $H(z)$ auf:

$$H(z) = \frac{U(z)}{V(z)} = \frac{b_0}{1 + a_1 z^{-1}} = b_0 \frac{z}{z + a_1} \tag{3.23}$$

Mit dem Befehl `freqz` kann man in Matlab die Frequency response dieses Systems plotten. Das Ergebnis ist in Abbildung 3.1 dargestellt.
Die Frequency response stimmt sehr gut mit der von Problem 3.1 überein. Man erkennt die Ähnlichkeit der beiden Probleme.
Bei der Rücktransformation der Übertragungsfunktion aus dem z-Bereich ergibt sich für die Impulsantwort (mit Hilfe einer Transformationstabelle):

$$h[n] = b_0 \cdot (-a_1)^n \tag{3.24}$$

Daraus kann der Noise Gain berechnet werden:

$$NG = ||\mathbf{h}||^2 = \sum_{n=0}^{\infty} (b_0(-a_1)^n)^2 = b_0^2 \sum_{n=0}^{\infty} (-a_1)^{2n} = 1^2 \sum_{n=0}^{\infty} (0.8993)^{2n} = \sum_{n=0}^{\infty} (0.8993^2)^n = \frac{1}{1 - 0.8993^2} = 5.23 \tag{3.25}$$

Dieser ist auch wieder höher als bei Problem 3.1, nimmt aber mit steigender Anzahl von berücksichtigten Iterationen ab.

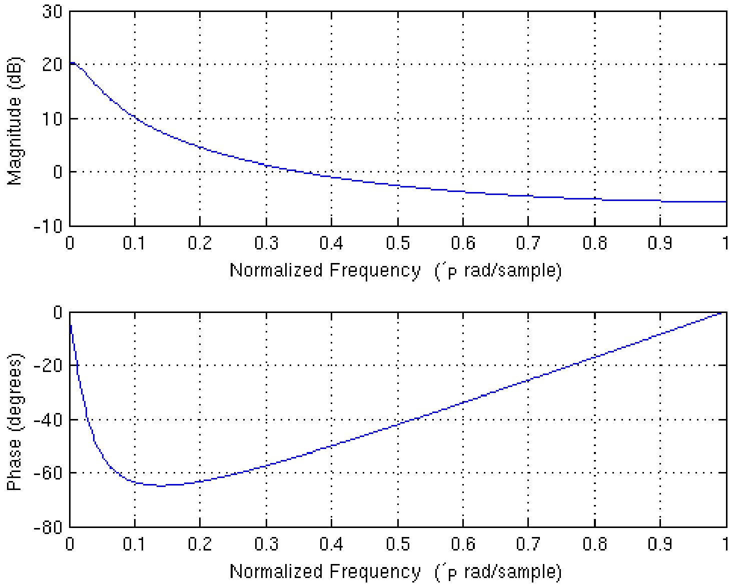

Abbildung 3.1: Frequency response des Systems aus Problem 3.3

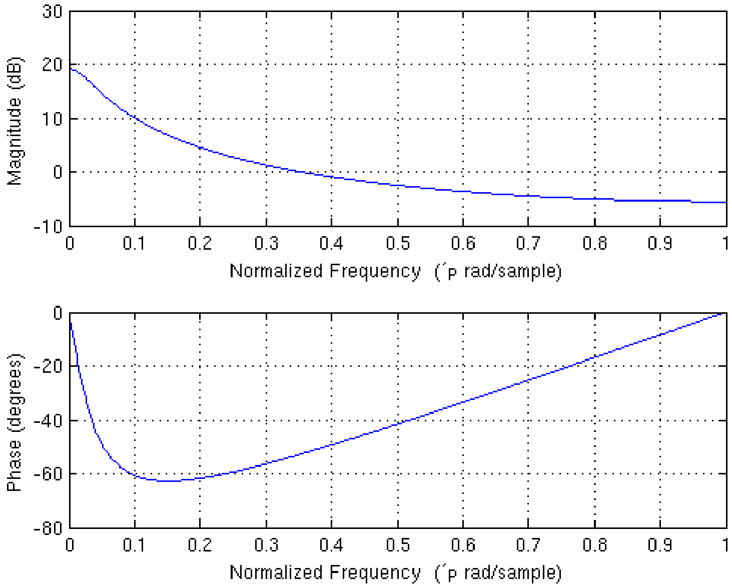

Abbildung 3.2: Frequency response des Systems aus Problem 3.1

4 Matlab Problem 3.4: Periodic Interference Cancelation

Zur Dämpfung der DTMF Störsignale wurde die Tatsache benutzt, dass es sich um periodische Störungen handelt.
Es wurde der in der Übung vorgestellte Filter "Cancelation of periodic interferences using a linear predictor" verwendet. Zunächst wurden die Parameter per Hand grob optimiert, sodass die Störungen möglichst wenig zu hören waren und das Nutzsignal möglichst wenig verzerrt wurde. Zu guter Letzt wurden mittels einer Schleife mehrere Parameter durchgetestet und jene Parameter verwendet, bei der der SNR maximal wurde.
Die optimalen Parameter waren: $N = 236$, $\mu = 0.00205$, $\Delta = 16$. Dabei ergab sich eine SNR von $3.81dB$.

5 Listings

5.1 Information vs. Energy

```
Ns = 10^6;  % number of samples

w = randn(1, 2 + Ns);

u = zeros(1, 2 + Ns);
u(1) = 0;
u(2) = 0;

e1 = zeros(1, 2 + Ns);
e2 = zeros(1, 2 + Ns);


%% calculate e[n] for different N

for i = 3:Ns

    u(i) = w(i) + u(i-1) - 1/8 * u(i-2);

    e1(i) = u(i) - 8/9 * u(i-1);  % N=1
    e2(i) = u(i) + 1 * u(i-1) - 1/8 * u(i-2);  % N=2

end


%% calculate ehat for different N and B

[eh01, M01] = BGsQuantizer(u, 1);
[eh02, M02] = BGsQuantizer(u, 2);
[eh03, M03] = BGsQuantizer(u, 3);
[eh04, M04] = BGsQuantizer(u, 4);
[eh05, M05] = BGsQuantizer(u, 5);

[eh11, M11] = BGsQuantizer(e1, 1);
[eh12, M12] = BGsQuantizer(e1, 2);
[eh13, M13] = BGsQuantizer(e1, 3);
[eh14, M14] = BGsQuantizer(e1, 4);
[eh15, M15] = BGsQuantizer(e1, 5);

[eh21, M21] = BGsQuantizer(e2, 1);
[eh22, M22] = BGsQuantizer(e2, 2);
[eh23, M23] = BGsQuantizer(e2, 3);
[eh24, M24] = BGsQuantizer(e2, 4);
[eh25, M25] = BGsQuantizer(e2, 5);


%% calculate entropy for different N and B

H01 = CondEntropy(eh01, M01);
H02 = CondEntropy(eh02, M02);
H03 = CondEntropy(eh03, M03);
H04 = CondEntropy(eh04, M04);
H05 = CondEntropy(eh05, M05);

H11 = CondEntropy(eh11, M11);
H12 = CondEntropy(eh12, M12);
H13 = CondEntropy(eh13, M13);
H14 = CondEntropy(eh14, M14);
H15 = CondEntropy(eh15, M15);

H21 = CondEntropy(eh21, M21);
H22 = CondEntropy(eh22, M22);
```

```
H23 = CondEntropy(eh23, M23);
H24 = CondEntropy(eh24, M24);
H25 = CondEntropy(eh25, M25);


%% plot results

x = 1:3;

figure(1);
plot(x, H01, x, H02, x, H03, x, H04, x, H05);
title('Upper Bounds of Entropy rate for N=0');
xlabel('n');
ylabel('H(\hat{e}[n])');
legend('B=1', 'B=2', 'B=3', 'B=4', 'B=5');

figure(2);
plot(x, H11, x, H12, x, H13, x, H14, x, H15);
title('Upper Bounds of Entropy rate for N=1');
xlabel('n');
ylabel('H(\hat{e}[n])');
legend('B=1', 'B=2', 'B=3', 'B=4', 'B=5');

figure(3);
plot(x, H21, x, H22, x, H23, x, H24, x, H25);
title('Upper Bounds of Entropy rate for N=2');
xlabel('n');
ylabel('H(\hat{e}[n])');
legend('B=1', 'B=2', 'B=3', 'B=4', 'B=5');
```

5.2 Noise Cancelation

```
%load('speech_signals.mat');

% optimal parameters
N = 236;
mu = 0.00205;
delta = 16;

% parameter search
%mmse_min = inf;
%N_min = 0;
%mu_min = 0;
%delta_min = 0;

d = dtmfs(:);

%for N = 235:1:237
%    for mu = 0.0019:0.00005:0.0021
%        for delta = 15:1:17

            x = [zeros(delta,1);d(1:end-delta)];


            [ y, e, c] = nlms2( x, d, N, mu);

            %e = d;

            MMSQE = sum((e - clean).^2);

%            if MMSQE < mmse_min
%                mmse_min = MMSQE;
%                N_min = N;
%                mu_min = mu;
%                delta_min = delta;
%            end

%        end
%    end
%end
```

```

%MMSQE = mmse_min;

SNR = 10*log(sum((clean-mean(clean)).^2)/MMSQE);

%disp(['optimal parameters: N=', num2str(N_min), ', mu=', num2str(mu_min), ',
                              delta=', num2str(delta_min)]);
disp(['SNR: ', num2str(SNR), 'dB']);
```