{ } } } } } } } } } } { }

Özgü Dogan (326 048)
Boris Henckell (325 779)

6. Juli 2012

Gruppe: D03

}

} {}

{ }

}

{ }

}

In diesem Termin beschäftigen wir uns mit einem Signalangepasstem Empfangsfilter. Wir werden eine zufällige Bitfolge mittels eines Formfilters in unterschiedliche Signalformen umwandeln, an der empfängerseite wieder decodieren und abschließend die Bitfehlerrate überprüfen. Abschließend werden wir kontrollieren welchen Einfluss die Wahl der Sendeform auf die Bitfehlerrate hat.

Rauschen kann ein übertragenes Signal beeinflussen und dadurch verfälschen. Bei Binärer Übertragung können daraus Bitfehler resultieren, die natürlich unerwünscht sind und vermieden werden sollten. Um daher den Effekt des Rauschens und dadurch auch die Bitfehlerrate zu minimieren wird auf der Seite des Empfängers nach der Übertragung durch den Kanal ein Signalangepasstes Empfangsfilter verwendet.
Außerdem lässt sich die Bitfehlerrate durch eine geschickte Wahl der Sendeformen weiter minimieren.
Beides werden wir im folgenen umsetzten und durchmessen.

Zur Übertragung einer Bitfolge über einen Kanal werden die einzelnen Bits in zu definierenden Sendeformen umgewandelt und in dieser Form über den Kanal übertragen. Diese Sendeform kann selbstverständlich während der Übertragung durch Rauschen verfälscht werden, was zur Folge hat, dass der Empfänger Schwierigkeiten hat zu entscheiden, ob das Übertragene Bit ursprünglich eine binäre 1 oder eine binäre 0 dargestellt hat. Das verwendete Sende-Empfangssystem sollte darauf ausgelegt sein, den Empfänger diese Entscheidung so leicht wie möglich zu machen.
Als Aussage über die Qualität eines Sende-Empfangssystem dient der “Signal-To-Noise-Ratio”(SNR) sowie die Bitfehlerrate (P_{Bit}). Ist nämlich das SNR, sprich das Verhältnis zwischen Sendeleistung zu Rauschleistung, am Empfänger möglichst groß, fällt es dem Entscheider leichter eine Richtige Entscheidung zu treffen. Die Bitfehlerrate sinkt bei großem SNR, da sie die Fehlerrate pro Bit darstellt. Fällt der Entscheider zunehmend richtige Entscheidungen sinkt daher die Bitfehlerrate.
Das SNR am Entscheider lässt sich folgendermaßen berechnen:

$$SNR_E = \frac{|f(t_a)|^2}{\sigma_r^2}$$

$|f(t_a)|^2$ stellt in diesem Fall die Amplitude des Nutzsignals zum gegebenen Abtastzeitpunkt und σ_r^2 die Leistung des Rauschens dar.

Für die weitere Betrachtung ist es notwendig zwischen Unipolarer und Bipolarer Übertragung zu unterscheiden. Die Unipolate Übertragung hat den Vorteil, dass sie mit weniger Aufwand umgesetzt werden kann, da nur eine Sendeform für eine der beiden Zustände notwendig ist. Der andere Zustand wird dann durch eine 0 übertragen. Dem entgegen nutzt die Bipolare Übertragung für jeden Zustand eine eigene Sendeform. Logischerweise ist diese Implementierung aufwendiger. Jedoch zahlt sich dieser Aufwand aus, da der Entscheider SNR für Bipolare Übertragung verbessert.
Um diese Verbesserung mathematisch zu erfassen betrachten wir die Bitenergie(E_b), die

Leistungsdichte von Weißem Rauschen($\frac{N_0}{2}$) sowie den Kreuzkorrelationskoeffizenten (ρ) ein. Die Bitenergie E_b ist die Energie eines Sendeimpulses und errechnet sich mit: [1]

$$E_b = \int_{-\infty}^{\infty} \left|s(t)|^2\right| \mathrm{d}t$$

Der Kreuzkorrelationskoeffizient ρ stellt die normierte Kreuzkorrelation zwischen den zwei Sendeimpulsen dar.[2]

$$\rho = \frac{1}{E_b} \int_0^{T_{Bit}} s_0(t) \cdot s_1(t) \mathrm{d}t$$

Abhängig von den Formen der Sendeimpulse kann dieser Koeffizent Werte von 1, beiden Signalformen stimmen überein, bis -1, die beiden Signalformen sind genau invertiert zueinander, annehmen.

Nun lässt sich mit einigen Umformungen die oben aufgeführte Formel für das Entscheider SNR umformen.
Für Unipolare Übertragung ergibt sich dann:

$$max\{SNR_E\} = \frac{2E_b}{N_0}$$

Im Gegensatz dazu ergibt sich für die Bipolate Übertragung:

$$SNR = \frac{4E_b}{N_0}(1-\rho)$$

Werden die Sendeformen entgegengesetzt zueinander und mit der selben Bitenergie wie bei der Unipolaren Übertragung gewählt, steigt das SNR_{EBi} auf den vierfachen Wert von dem der Unipolaren Übertragung.

$$\} \quad \{\}$$

Zunächst sollten wir uns als Vorbereitung für den Praktikumstermin mit dem Inhalt des Kapitels zu optimalen Empfängerstrukturen aus dem Skript vertraut machen und nachvollziehen können.

Als nächstes sollte mit Hilfe einer Matlab Datei bipolare Sendeimpulse für eine bipolare Übertragung definiert werden. Diese Sendeimpulse sollten als Vektoren SF0 und SF1 mit -1 und 1 ein Kreuzkorrelationsergebnis von $\rho = 0, -\frac{1}{3}$ oder -1 liefern, wobei die Vektoren für $\rho = -1$ und $\rho = -\frac{1}{3}$ drei Werte und für $\rho = 0$ vier Werte besitzen müssen.
Als Kontrolle dafür, sollte in der Matlab-Datei tatsächlich die Kreuzkorrelation durchgeführt und das Ergebnis untersucht werden. Außerdem wird die Bitenergie E_B der Sendeimpulse berechnet, wobei die Vektoren als zeitkontinuierliche Spannungsverläufe der Dauer T_B angenommen wird, bei denen Die Spannungsamplituden nur $+1\ V$ oder $-1\ V$ betragen können. Es sollte beachtet werden, dass $T_{SF} = 20\mu s$ konstant bleiben und sich somit das T_B mit $T_B = 2 \cdot N \cdot_{SF}$ mit N Werten in SF0 bzw. SF1 berechnen lässt. Als Bedingung sollte noch

gelten, dass die Bitenergien für SF0 und SF1 als Paar für ein ρ gleich sein sollten.

Wir haben uns für die folgenden Signalformpaare entschieden:

Für $\rho = -1$:

$$SF_0 = [-1, \quad 1, -1\,]$$
$$SF_1 = [\quad 1, -1, \quad 1\,]$$

Für $\rho = -\frac{1}{3}$:

$$SF_0 = [-1, -1, \quad 1\,]$$
$$SF_1 = [\quad 1, -1, -1\,]$$

Für $\rho = 0$:

$$SF_0 = [-1, \quad 1, -1, \quad 1\,]$$
$$SF_1 = [\quad 1, \quad 1, -1, -1\,]$$

Als Vorbereitung für den praktischen Teil des Versuchs, sollte eine Funktion SAF implementiert werden, welche einen SAF-Empfänger in Matlab realiseren soll. Der Input besteht aus DataSamples, ClkSamples und SFSamples, welche gegeben sind. der Output dagegen soll dabei der Vektor Values sein, in der die Werte nach dem signalangepasstem Filter eingetragen sein sollen.

Um das umzusetzen haben wir zunächst das Clk-Signal auf $0V$ angehoben, auf $1V$ normiert, mit einem Entscheider in Binäre Informationene umgewandelt und so versetzt, dass wir immer den Anfang eines neuen Bits markiert haben. Anschließend haben wir die beiden verwendeten Sendeformen invertiert, jeweils das Signal mit der invertierten Sendeform gefaltet und die Faltung voneinander abgezogen. Zum Schluss haben wir dieses Resultierende Signal an den Zeitpunkten abgetastet an dem das überarbeitete Clocksignal eine eins besitzt.
Auf diese Weise haben wir das Signalangepasse Filter realisiert.

Nach der Implementierung der SAF-Funktion sollten wir uns mit dem Versuchsaufbau und der Durchführung vertraut machen und ein Blockschaltbild dazu entwerfen. Die genaue Vorgehensweise ist in dem Abschnitt Durchführung detaliert erläutert. Hier ist das entworfene Blockschaltbild zu dem Versuch:

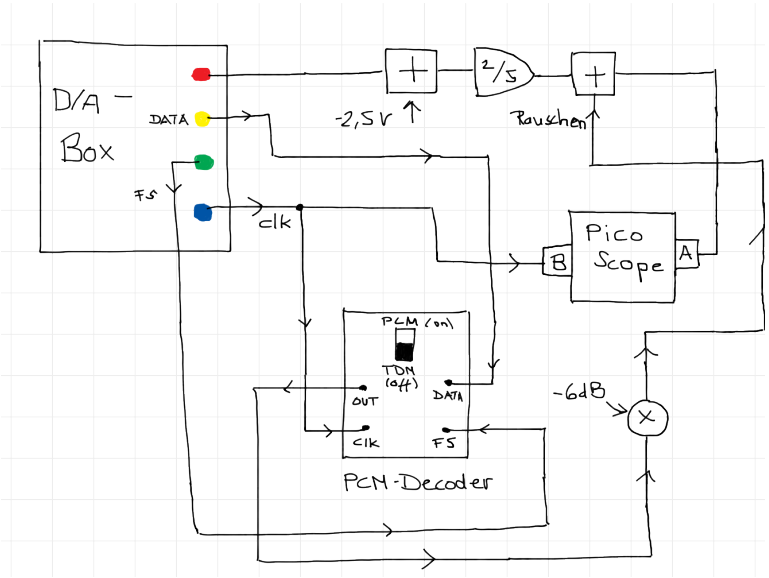

Abb. 1: Blockschaltbild des Versuchsaufbau

Um die implementierte Funktion SAF zu überprüfen, führten wir mithilfe der vorgegebenen ENue_SAF_Messumgebung.m durch, welche das Hauptprogramm der Messung darstellt. Mit dem Parameter Simulation (für die Vorbereitung bleibt dieser bei 1) kann man entscheiden, ob die Ausgabe und Einstellungen des Steckbretts simuliert werden, oder ob die Daten über die D/A-Box ausgegeben und per PicoScope eingelesen werden. Mit dem Parameter SAF kann man einstellen, ob mit dem DataSignal eine Sendeformung und dem Empfangssignal eine signalangepasste Filterung durchgeführt wird. Weiterhin sollten wir für die Vorbereitung die Kanal- und Filtereinstellungen nicht verändern, wohingegen der NoiseFaktor, welcher den Faktor des Rauschens einstellt, variiert werden durfte um die Leistung AWGN (additive white Gaussain noise) des Kanals zu beeinflussen.
Mit der Funktion Channel führt man die Übertragung entweder als Simulation oder auf dem ETT aus, wobei die Einstellungen der Simulation denen des wirklichen Aufbaus entsprechen. Es wird ein Rückgabevektor Y ausgegeben, welcher die Werte nach der Abtastung des Kanals enthält. Wenn SAF deaktiviert ist, wird eine einfache Nachabtastung mit einem Schwellwert von $0V$ durchgeführt.
Außerdem entspricht der Rückgabewert Noise einer Rauschmessung des Kanals, wozu eine 0-Bit-Folge gesendet und das Empfangssignal ohne SAF oder Nachabtastung als PuciScope-Sample-Signal zurückgegeben wird. Dieses Signal enthält ungefähr 10000 Samples.

Zuletzt sollte beantwortet werden, wie der Multiplikationsfaktor für das Rauschen variiert werden müsste, damit die Wasserfallkurve ausreichend ermittelt werden kann. Im Versuch beträgt das Rauschen dabei $-6dB$.

$$\{ \quad \}$$

$$\{\} \qquad \{$$

Die erste Aufgabe des Praktikums beschäftigt sich fast ausschließlich mit dem Aufbar des Versuches, um den Signalverlauf mit der Matlab Funktion ParallelOUT($[0101010], 100$) zu untersuchen. Dazu wird die D/A-Box, welcher an den Computer angeschlossen und somit über die Matlab Dateien steuerbar ist, als Quelle für jegliche verwendete Signale verwendet. Die D/A-Box besitzt vier

Ausgänge, die mit unterschiedlichen Farben gekennzeichnet sind. Der rote Ausgang gibt das DataSignal aus. Dieser kann eine Amplitude von $0V$ oder $5V$ besitzen. Um ihn auf den von uns erwünschte Spannungsbereich von $\pm 1V$ zu bringen, wird dieses Signal zunächst mit einer Gleichspannung von $-2.5V$ aus der variablen Spannungsquelle des Steckbretts addiert und danach mit einem Faktor von $\frac{2}{5}$ gedämpft. Damit erreichen wir den nötigen Spannungsbereich, welcher auf dem A Kanal des PicoScopes kontrolliert wird.
Der blaue Ausgang der D/A-Box gibt das Clock-Signal wieder. Dieser wird auf den B Kanal des PicoScope geführt und ebenfalls kontrolliert.
Als nächstes wird der PCM Decoder mit der D/A-Box verbunden, indem das Clock-Signal auch auf den Clock-Eingang des Decoder geführt wird. Weiterhin wird das Signal aus dem grünen Ausgang der Box, welcher das Frame-Signal (FS) wiedergibt, mit dem FS-Eingang des Decoders und die PCM-codierten Datenworte, die aus dem gelben Ausgang der Box entnommen werden können, mit dem PCM Data-Eingang des Decoders kontaktiert. Wichtig ist auch, dass der Schalter auf dem Decoder Modul auf PCM geschaltet ist und nicht auf TDM.
Somit ist der PCM Decoder mit allen Signalen beliefert, die er zum decodieren braucht. Daher kann man nun das Output Signal verwenden, welcher eine Spannung des Faktors wiedergibt, die für die Verstärkung oder Dämpfung des Rauschens dient. Diese Spannung wird an einem Multiplikator mit dem $-6dB$ Rauschen multipliziert und an dem Addierer mit Multiplikatoren mit dem DataSignal, welcher bereits auf den korrekten Spannungsbereich eingestellt wurde, addiert. Das Ergebniss dieser Addition wird weiterhin auf den A Kanal des PicosScopes geführt und ausgewertet.

Zuletzt wird das verrauschte Signal überprüft, indem mit der Funktion PCM_Decod(192) ein Verstärkungsfaktor für das Rauschen gesetzt wird. Das verrauschte Signal wird mithilfe der PicoScope-Software und der Matlab-Funktion ParallelOUT($[0101010], 100$) dargestellt und auch mit den Werten 0, 128 und 255 untersucht.

Der gesamte Versuchsaufbau sah folgendermaßen aus:

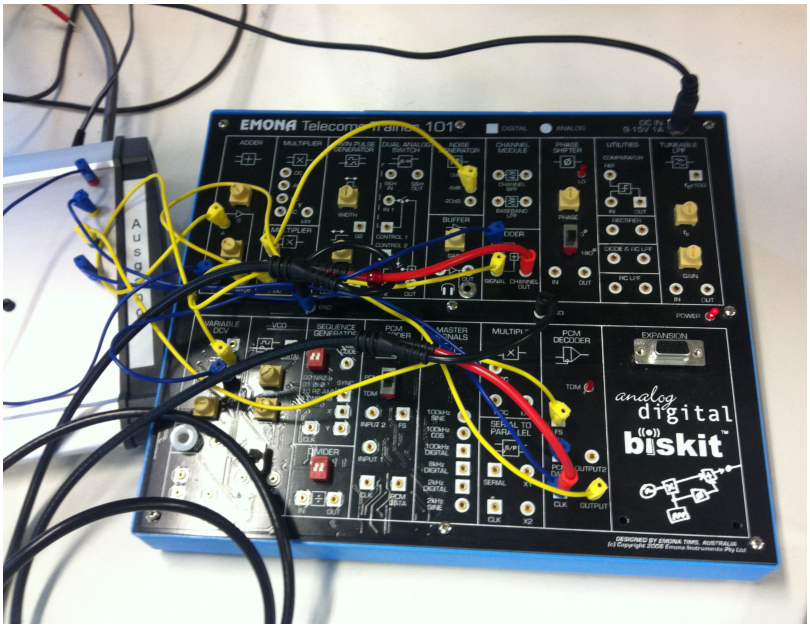

Abb. 2: Versuchsaufbau

$\{\}$ $\{$ $\}$

Das Ergebnis der Bitfehlermessung sollte die Darstellung von unterschiedlichen Wassefallkurven sein. Dafür wurden am Ende die Wasserfallkurven für die drei unterschiedlichen untersucht, welche wir bereits in der Vorbereitung untersucht hatten. Es wurden die BER-Messungen aller drei -Werte mit 15 verschiedenen, sinnvollen Verstärkungsfaktoren für das Rauschen durchgeführt. Dabei sollten wir beachten, dass für größere SNR-Werte die Steigung stärker und eine BER von 0 für ein vorhandenes Rauschen nicht aussagekräftig ist, wenn die Messung des BER eine minimale Auflösung von 10^-4 besitzt. Danach wurden die Bitfehlerraten der übertragenen Informationsbits für den optimalen Abtastzeitpunkt T_B gemessen und in ein Diagramm zusammengefasst. Dieses Diagramm enthält nun die Wasserfallkurven von $= -1$, von $= -\frac{1}{3}$ und von $= 0$ mit 15 Messwerten.
Die Ergebnisse wurden geplottet und in der Auswertung diskutiert.

$\}$

$\}$ $\{\}$

Nun werden die entstandenen Wasserfallkurven aus der simulierten Vorbereitungsaufgabe ausgewertet. Es sind drei Kurven, je einer für ein , in Bitfehlerrate über dem SNR dargestellt. Wie in der Theorie erläutert, stellt sich ein von 1 genau dann ein, wenn beide Sendeimpulse übereinander stimmen. Wenn sie genau invertiert zueinander sind, nimmt den Wert -1 an. Dies bedeutet, dass der Unterschied zwischen beiden Signalimpulsen zum Zeitpunkt des Abtastens am größten ist. Daher fällt es dem Entscheider leichter das richtige Bit zu identifizieren.
Bei einem von 0 ist dieser besagte Abstand im Abtastzeitpunkt um eine Bitenergie kleiner, wodurch es für den Entscheider schwieriger wird eine richtige Wahl zwischen einer 1 und einer 0 zu treffen, sobald das Signal mit Rauschen überlagert ist.

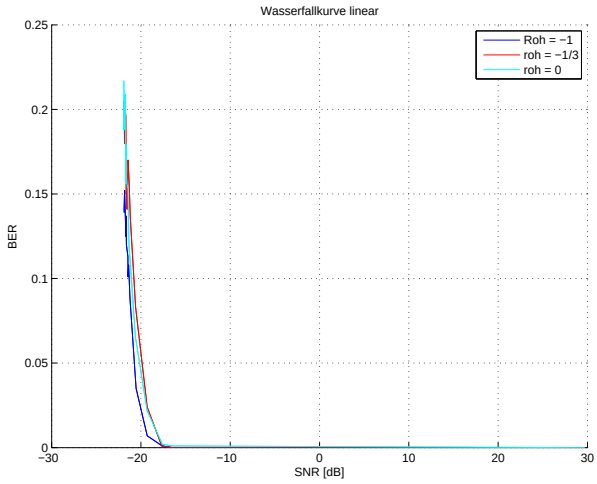

Abb. 3: Wasserfallkurve linear simuliert

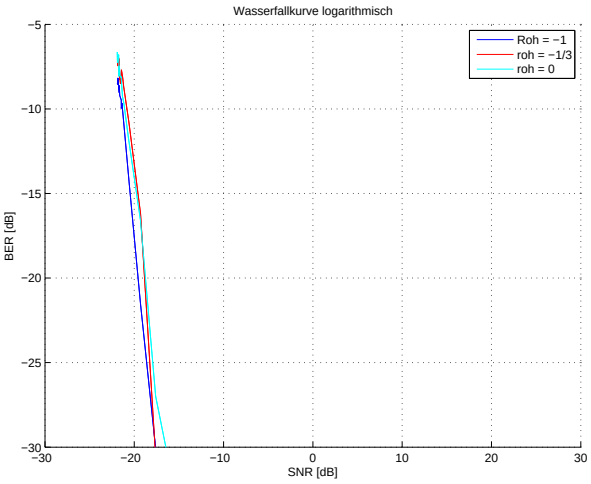

Abb. 4: Wasserfallkurve logarithmisch simuliert

Diese Beobachtungen können wir machen, wenn wir uns die simulierten Wasserfallkurven angucken. Es fällt auf, dass in der linearen darstellung die blaue

Wasserfallkurve, welcher zu $= -1$ gehört, die kleinste Bitfehlerrate beim kleinsten SNR aufweist, was darauf deutet, dass der Entscheider genauer arbeitet, auch wenn das Rauschen im Vergleich am größten ist. Unerwartet ist jedoch der Verlauf der anderen Wasserfallkurven, die ungefähr gleich aussehen. Wir hätten erwartet, dass die Kurve von $= 0$ eine größere Bitfehlerrate bei einem größeren SNR aufweist, was aber nicht der Fall ist.

TODO:

warum ist das so?

Dennoch kann man zusammenfassen, dass alle drei Wasserfallkurven ab einem SNR von ca. $-17dB$ eine Bitfehlerrate von null besitzen. Das liegt daran, dass das Rauschen so klein wird, dass der Entscheider auch bei zwei nicht so weit auseinander liegenden Signalimpulsen keine Probleme hat die korrekte Entscheidung zu treffen.

In der logarithmischen Darstellung können wir das gleiche wie in der linearen Darstellung feststellen. Auch wenn das BER logarithmisch ist, sind die Anordnungen der Wasserfallkurven ziemlich gleich geblieben.

$\{\}$ $\{$

Da diese Aufgabe nur zum korrekten Aufbau des versuchs gedient hat, gibt es hier keine Ergebnisse, worüber inhaltlich diskutiert werden müsste. Daher stellen wir nur die verrauschten Signale dar, die während der Messung mit der Funktion ParallelOut($[0101010], 100$) aufgenommen wurden.

Hier das Signal mit additivem Rauschen mit dem Verstärkungsfaktor 0:

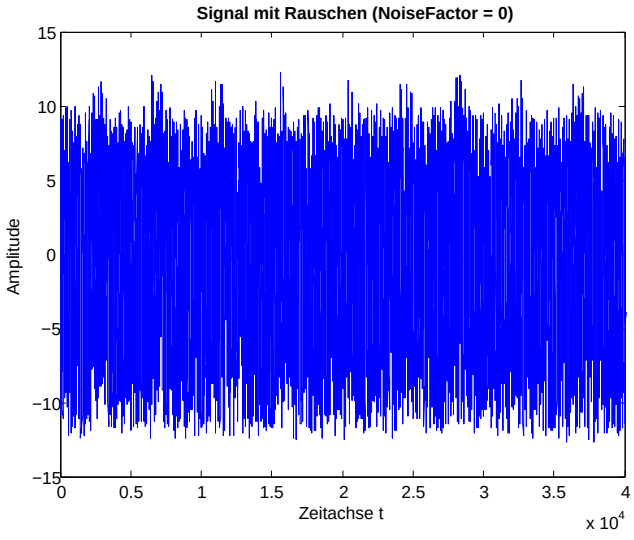

Abb. 5: Signal + Rauschen (NoiseFactor = 0)

Hier das entstandene Signal mit Rauschen bei einem Verstärkungsfaktor von 128:

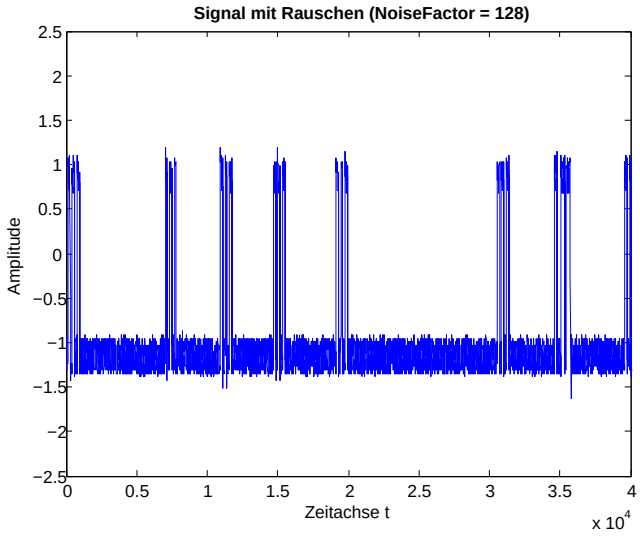

Abb. 6: Signal + Rauschen (NoiseFactor = 128)

Und zuletzt das Signal mit Rauschen bei einem Verstärkungsfaktor mit 255:

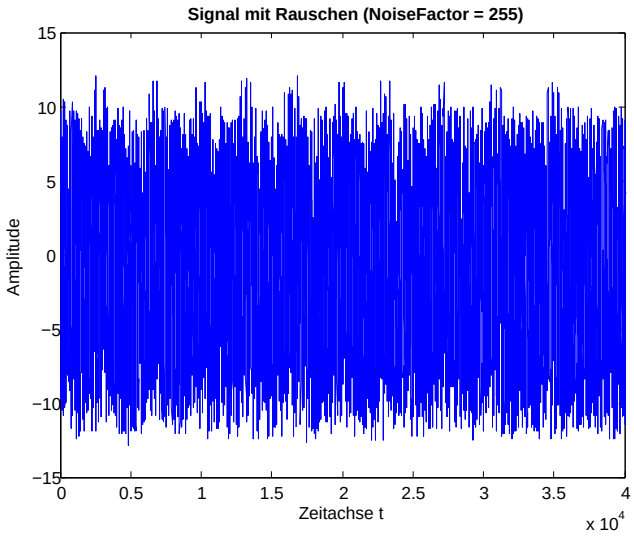

Abb. 7: Signal + Rauschen (NoiseFactor = 255)

Es fällt auf, dass die mit relativ großem Rauschen überlagerten Signale bei den Verstärkungsfaktoren 0 und 255 ungefähr gleich groß in der Amplitude sind. Diese Beobachtung bestätigt sich mit der Spannungsmessung, welche wir am Output des PCM-Decoders durchgeführt haben. Die Verstärkungsspannung, die ausgegeben wurde, betrug bei dem Faktor 0 genau $2V$ und bei dem Faktor 255 genau $-2V$. Damit erklärt sich die große Ähnlichkeit der entstehenden Rauschsignale.
Bei einem Verstärkungsfaktor von 128 dagegen tritt so gut wie gar kein Rauschen auf und wir erhalten das ursprünglich rauschfreie Signal im Spannungsbereich von -1 bis $1V$. Das liegt daran, dass bei 128 eine Output-Spannung von fast $0V$ aus dem PCM-Decoder ausgegeben werden, wodurch das additive Rauschen eben-

falls zu fast null wird.
Da die Verstärkungsfaktoren 0 und 255 jeweils die invertierten Verstärkungsspannungen produzieren, beschlossen wir, die Bitfehlermessung in der zweiten Aufgabe mit nur einem wachsendem Rauschfaktor von 0 bis 128 durch zu führen, da dieser Bereich für eine sinnvolle Auswertung der Wasserfallkurven ausreicht.

$\{\}$ $\{$ $\}$

Nun betrachten wir die Wasserfallkurven aus der Bitfehlermessung der zweiten Aufgabe. Auch hier erwarten wir ähnliche Ergebnisse wie in der Auswertung der Simulationsaufgabe aus der Vorbereitung:

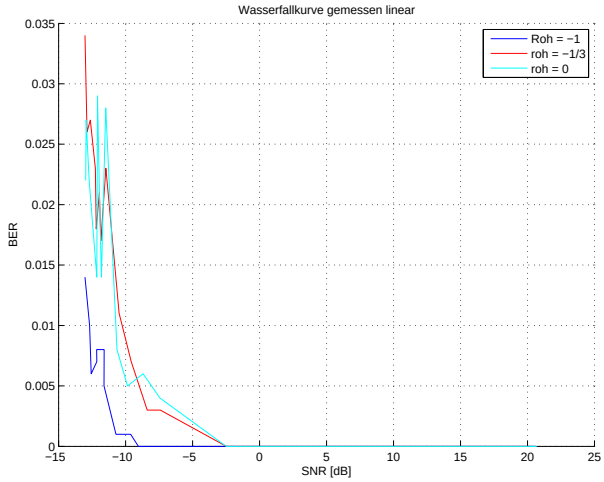

Abb. 8: Wasserfallkurve linear gemessen

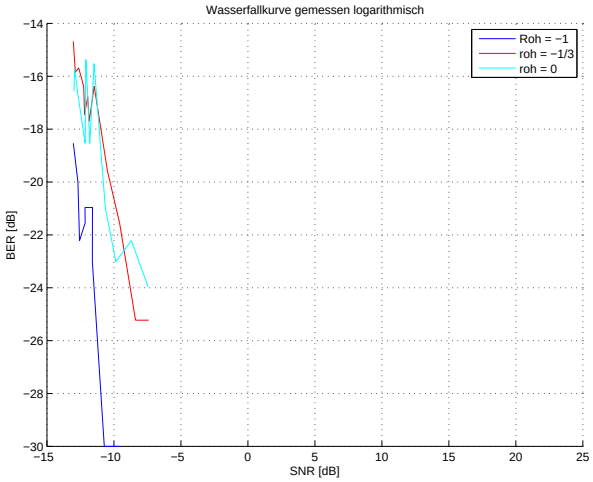

Abb. 9: Wasserfallkurve logarithmisch gemessen

In der realen Messung der Bitfehlerraten in Abhängigkeit zum SNR erfüllen die drei Wasserfallkurven noch deutlicher unsere Erwartungen. Angefangen mit der blauen Kurve, welche wieder zum $= -1$ gehört, kann man sehr deutlich sehen, dass die Bitfehlerrate bei einem SNR von $-13dB$ kleiner als die Hälfte der Bitfehlerrate der anderen Wasserfallkurven und somit am geringsten ist. Bei einem SNR von ca. $-8dB$ erreicht diese Wasserfallkurve schon eine Bitfehlerrate von null, womit wir sagen können, dass der Entscheider hier sehr genau gearbeitet hat.
Anders als in der Simulation kann man in der realen Messungen einen sehr kleinen Unterschied zwischen der Wasserfallkurve von $= -\frac{1}{3}$ und $= 0$ sehen. Im Durchschnitt weist die Wasserfallkurve bei null eine etwas höhere Bitfehlerrate bei den untersuchten SNR. Diese Beobachtung erfüllt unsere Erwartungen, die wir bei diesen zwei Korrelationsergebnissen bezüglich der Signalimpulse an den Entscheider haben.
Dennoch kann man auch in dieser Darstellung feststellen, dass alle drei Wasserfallkurven an einer SNR von ca. $-2dB$ keine Bitfehler mehr vorweisen können, da das vorhandene kleine Rauschen den Entscheider nicht mehr beeinträchtigt.

Auch die logarithmische Darstellung der Bitfehlerratenachse ändert nichts an der Auswertung des Entscheidungsvermögens bei unterschiedlichen SNR-Werten und den untersuchten s.

TODO:

ich weiß nicht, was man zur log. Darstellung noch sagen kann. . .

$$\{ \quad \}$$

TODO:

soll ich das machen?

[1] Prof. Dr.-Ing. Sikora, Thomas; Prof. Dr.-Ing. Noll, Peter: Einführung in die Nachrichtenübertragung, S.356

[2] Prof. Dr.-Ing. Sikora, Thomas; Prof. Dr.-Ing. Noll, Peter: Einführung in die Nachrichtenübertragung, S.363