Bakkalaureatsarbeit

Wiederholungscode

Görtschacher Lukas

Institut für Signalverarbeitung und Sprachkommunikation
Technische Universität Graz
Vorstand: Univ.-Prof. Dipl.-Ing. Dr. techn. Gernot Kubin

Betreuer: Dipl.-Ing. Bernhard Geiger

Graz, im Oktober 2012

Inhaltsverzeichnis

1 Warum Kanalcodierung?

Claude E. Shannon beschrieb in einem Artikel [1] ein Kommunikationsystem mit folgender schematischer Abbildung 1. Dabei ist der Weg zwischen einer Quelle und einer Senke beschrieben, welcher über einen Sender, über einen verrauschten Kanal und einem Empfänger geht. Es

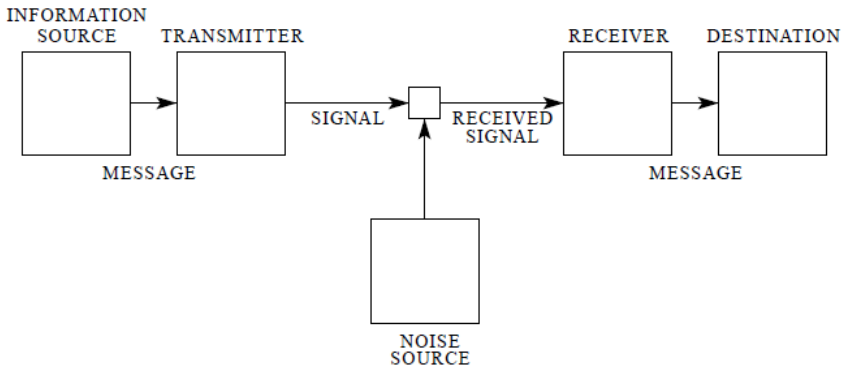

Abbildung 1: Kommunikationssystem nach Shannon

stellt sich nun die Frage, ob eine Nachricht, die von der Quelle generiert wird, am geforderten Ziel noch erkennbar bzw. nachvollziehbar ist. Die Anforderung an die Codierung ist es nun, die Nachvollziehbarkeit der Information (Nachricht) auf der Empfangsseite sicherzustellen.

Es werden im Folgenden einige wichtige Begriffe der Nachrichtentechnik erläutert, bevor auf die eigentliche Codierung eingegangen wird.

2 Wichtige Begriffe

2.1 Entropie

Die Entropie oder der mittlerer Informationsgehalt einer Zufallsvariable ist abhängig von den Auftrittswahrscheinlichkeiten der einzelnen Realisierung (Zeichen) und ist folgendermaßen definiert:

$$H = \sum_{i=1}^{n} p_i \cdot \log_a \left(\frac{1}{p_i} \right) \tag{1}$$

Wird für die Basis a des Logarithmus 2 gewählt, so kann die Entropie in $bit/Zeichen$ angegeben werden. Die Entropie wird am größten, wenn alle Zeichen mit der selben Wahrscheinlichkeit auftreten(vgl. [2]).

2.2 Redundanz

Die Redundanz beschreibt die überflüssigen bits pro Zeichen, d.h.: bits in denen keine Information steckt.

$$R = \overline{m} - H \tag{2}$$

$$r = \frac{\overline{m} - H}{\overline{m}} \tag{3}$$

R steht hier für die absolute Redundanz und r für die relative Redundanz. $\overline{m}$ ist der mittlere Binärstellenaufwand, der angibt, wieviele bits im Durchschnitt für die Darstellung eines Zeichens verwendet werden. $\overline{m}$ wird folgendermaßen berechnet.

$$\overline{m} = \sum_{i=1}^{n} m_i \cdot p_i \tag{4}$$

Dabei ist m_i die Anzahl der Binärstellen und p_i die Auftrittswahrscheinlichkeit des i-ten Zeichens.

2.3 Transinformation

In Abbildung 2 ist ein gestörter, gedächtnisloser und zeitdiskreter Kanal dargestellt. Bevor auf die Transinformation eingegangen wird, werden die einzelnen Entropien, welche im Bild dargestellt sind besprochen. Da die Senke den Empfänger als Quelle sieht, kann man von zwei

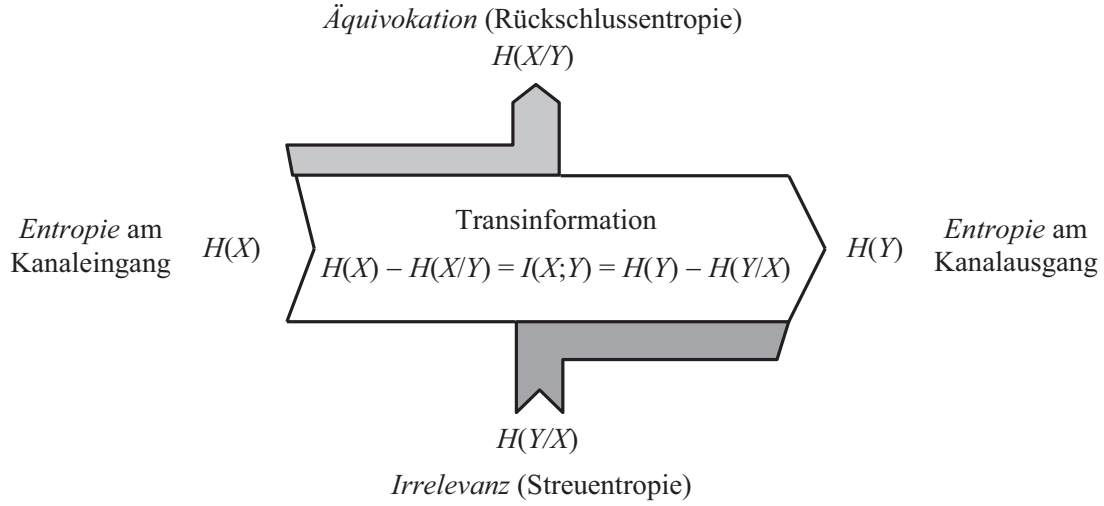

Abbildung 2: Transinformation [4, S.93]

Quellen mit den möglichen Zeichen X und Y ausgehen, die statistisch voneinander abhängig sind. $H(X)$ und $H(Y)$ beschreiben nun jeweils die Entropie am Eingang bzw. am Ausgang des Kanals.

$$H(X) = \sum_{x \in \mathcal{X}} p(x) \cdot \mathrm{ld}\left(\frac{1}{p(x)}\right) \tag{5}$$

$$H(Y) = \sum_{y \in \mathcal{Y}} p(y) \cdot \mathrm{ld}\left(\frac{1}{p(y)}\right) \tag{6}$$

Die Verbundentropie beschreibt die Entropie von Zeichenpaaren aus X und Y und wird folgendermaßen berechnet:

$$H(X,Y) = \sum_{x} \sum_{y} p(x,y) \cdot \mathrm{ld}\left(\frac{1}{p(x,y)}\right) \tag{7}$$

Die bedingte Entropie $H(X|Y)$ wird als Äquivokation bezeichnet. Sie beschreibt den Informationsverlust am Kanal, also die Unsicherheit bezüglich X, wenn der Ausgang $Y = y$ bekannt ist. Bei einem ungestörten Kanal ist $H(X|Y) = 0$ und bei einem völlig gestörten Kanal ist $H(X|Y) = H(X)$.

$$H(X|Y) = \sum_x \sum_y p(x,y) \cdot \mathrm{ld}\left(\frac{1}{p(x|y)}\right) = \sum_y p(y) \cdot H(X|Y=y) \tag{8}$$

Die bedingte Entropie $H(Y|X)$ wird als Irrelevanz bezeichnet. Sie beschreibt die Fehlinformationen am Empfänger, wenn das gesendete Zeichen x bekannt ist.

$$H(Y|X) = \sum_x \sum_y p(x,y) \cdot \mathrm{ld}\left(\frac{1}{p(y|x)}\right) \tag{9}$$

Es gibt nun mehrere Möglichkeiten zur Darstellung der Transinformation (vgl.[4]):

$$T = H(X) - H(X|Y) \tag{10}$$

$$T = H(Y) - H(Y|X) \tag{11}$$

2.4 Kanalcodierung

Abbildung 3 zeigt schematisch ein Übertragungssystem mit Kanalcodierung. Der Signalrauschabstand ist ein wichtiger Begriff für die Übertragungsqualität von Kanälen. Naheliegend wäre eine Erhöhung der Sendeleistung um die Qualität zu steigern. Das ist allerdings meist mit einigen Nachteilen verknüpft, wie z.B.: Stören von anderen Systemen, Überschreitung von Maximalwerten oder den Betrieb von Bauelementen in einem kritischen Bereich. Auch die Veringerung der Rauschleistung ist meist nicht möglich, bzw. mit zu hohen Kosten verbunden. Bei den Kanalcodierungsverfahren werden nun redundante bits hinzugefügt, wodurch Fehler erkannt, oder sogar korrigiert werden können. Dies geht allerdings bei gleichem Signalrauschabstand auf Kosten der Coderate[1] (vgl.[2]).

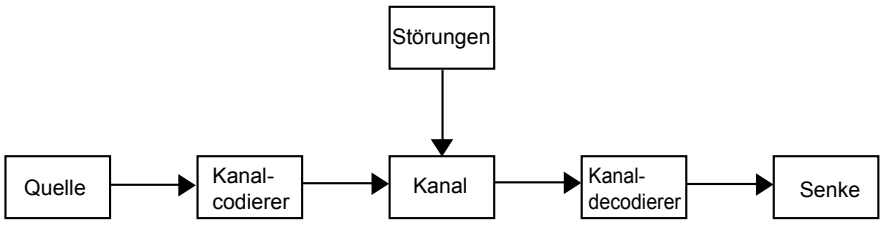

Abbildung 3: Kanalcodierung

[1] Entspricht dem Verhältnis von Informationsbits zu Kanalbits

2.5 AWGN-Kanal

Der AWGN-Kanal ist ein Kanalmodell, bei dem das Nutzsignal durch additives weißes gaußsches Rauschen beeinflusst wird. Es zeichnet sich durch eine konstante spektrale Rauschleistungsdichte und einer gaußverteilten Amplitude aus (vgl. [5],[6]).

2.6 BER Kurven

Bei diesen Diagrammen werden die Bitwehlerwahrscheinlichkeit und das Verhältnis der Energie pro Informationsbit E_b zur spektralen Rauschleistungsdichte N_0 gegenübergestellt. $\frac{E_b}{N_0}$ ist dabei proportional zum Signalrauschabstand $\frac{S}{N}$, wobei die Bandbreite vernachlässigt wird. S stellt die Signalleistung des codierten Signals und N die Rauschleistung dar.

2.7 Hamming Abstand

Er beschreibt die Anzahl der Stellen in welchen sich zwei beliebige, gültige Codewörter eines Codes unterscheiden. Angenommen, die beiden Bitfolgen 100010 und 100011 sind solche Codewörter, dann unterscheiden sie sich in einer Stelle und der Hamming Abstand beträgt 1. Er wird in weiterer Folge mit D bezeichnet.

2.8 Abstand

Unter dem Abstand d versteht man den kleinsten Hamming Abstand, den zwei beliebige Codewörter eines Codes haben.

2.9 Codiergewinn

Den Codiergewinn kann man aus den BER-Kurven erkennen. Er beschreibt jene Signalenergie, die bei gegebener Bitfehlerwahrscheinlichkeit durch Kanalcodierung eingespart werden kann.

3 Wiederholungscode

In diesem Kapitel werden nun die Eigenschaften des Wiederholungscodes beschrieben. Auf der Grundlage der Simulationen mit Matlab ergaben sich interessante Fragen bezüglich der mathematischen Hintergründe. So wurde der Großteil der Simulationen auch analytisch nachgewiesen. Außerdem wurden theoretische Grenzen des Codes berechnet und in den Simulationen dargestellt. Die mit Matlab entworfenen Modelle basieren auf binären, antipodalen Signalschemen wie z.B.: BPSK und auf einem AWGN-Kanal.

3.1 Definition

Der Wiederholungscode gehört zu einer einfachen Klasse von fehlerkorrigierenden Codes. Dabei wird jedes Informationsbit k mal wiederholt. Es entsteht ein Code mit zwei zulässigen Codewörtern, welche einen Abstand von $d = k$ besitzen.

3.2 Coderate

Bei einem Informationsbit werden k bits übertragen, was zur folgenden Coderate führt

$$CR = \frac{1}{k} \tag{12}$$

3.3 Redundanz

Die Redundanz bei diesem Verfahren ist natürlich sehr hoch.

$$R = k - 1 \tag{13}$$

und

$$r = \frac{k-1}{k}. \tag{14}$$

3.4 Codierung und Decodierung

Bei der Codierung werden wie schon erwähnt, die Informationsbits k mal wiederholt und über den Kanal geschickt. Diese Kanalbits können dann am Kanalausgang auf verschiedene Arten decodiert werden. Bei der Decodierung mit harter Entscheidung werden die modulierten Kanalbits nur auf größer oder kleiner als 0 überprüft. Die Mehrheit der Zustände der k Kanalbits entscheidet über den Zustand des Informationsbits. Beim Decodieren mit weicher Entscheidung wird ein Decodierer mit unendlich hoher Quantisierung verwendet und die Entscheidung aufgrund der Mittlung der k modulierten Kanalbits getroffen. Bei den Decodierern mit endlicher Quantisierung wird der Bereich, indem sich die Amplituden der empfangenen Kanalbits befinden unterschiedlich gewichtet. So werden die Kanalbits auf Zustände wie z.B. „wahrscheinlich eine 0“, „wahrscheinlich eine 1“, „ziemlich sicher eine 0“, „ziemlich sicher eine 1“, usw. überprüft.

3.5 Codiergewinn

Wie aus den Simulationen mit Matlab hervorgeht, kann mit diesem Verfahren bei einem AWGN-Kanal kein Codiergewinn erziehlt werden, er wird in der Regel sogar negativ. Das kommt daher, das sich bei konstanter Signalleistung die Signalenergie pro Informationsbit mit steigender Redundanz verkleinert und somit auch $\frac{E_b}{N_0}$ geringer wird. Letzteres ergibt sich natürlich auch bei anderen Codierungsverfahren, wobei aber trotzdem ein Codiergewinn erzielt wird.

Die mit Matlab simulierten BER-Kurven werden nun dargestellt und beschrieben. Bei diesem Code wurden Realisierungen mit drei, fünf und sieben Wiederholungen gewählt. Dabei wird bei der Decodierung eine eindeutige Entscheidung getroffen und somit sind diese Codes perfekt. Bei einer geraden Anzahl von Wiederholungen kommt noch eine Zufallsentscheidung hinzu, wenn genau die Hälfte der wiederholten bits falsch empfangen werden.

4 Decodierung mit harter Entscheidung

Von harter Entscheidung spricht man dann, wenn man nur die Information, ob das Signal größer oder kleiner als Null ist, zur Decodierung verwendet. Somit gehen Informationen über die Amplitude verloren, ein solcher Decodierer ist aber in der Praxis einfach zu realisieren.

4.1 Berechnung der BER- Kurve

Zum Verständnis der Berechnung kann Abbildung 4 betrachtet werden. Dabei handelt es sich um die Wahrscheinlichkeitsdichtefunktion (engl.: PDF) wie es sich bei diesem Modulationsschema durch das additive weiße Rauschen ergibt. Voraussetzung für diese Darstellung ist, dass die zwei Codewörter mit gleicher Wahrscheinlichkeit gesendet werden. Die beiden farbigen Flächen ergeben dann die Wahrscheinlichkeit für ein falsch empfangenes Kanalbit an. Es handelt sich also um die Wahrscheinlichkeit, dass eine 0 gesendet und eine 1 empfangen wurde (und umgekehrt). Wegen der Gleichheit genügt es, die Berechnung nur für eine bedingte Wahrscheinlichkeit durchzuführen. Diese Berechnung gilt für perfekte Codes, d.h. die Anzahl der Wiederholungen ist ungerade. Das Rauschen wird mit der Zufallsvariable N

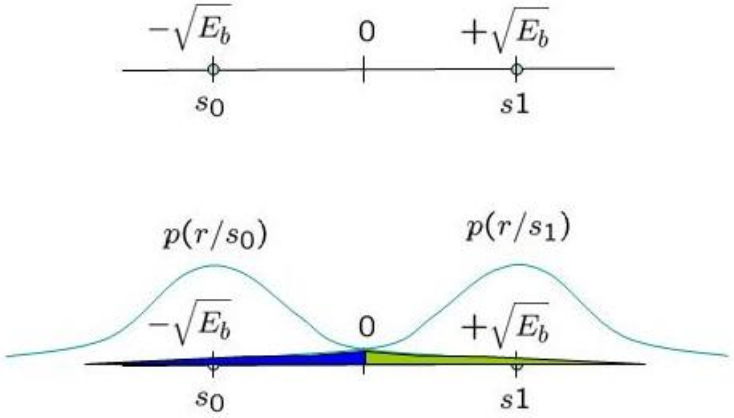

Abbildung 4: bedingte PDF [11]

bezeichnet und ist -normiert auf die Bandbreite- $(0, \frac{N_0}{2})$ verteilt.

$$N \sim \left(0, \frac{N_0}{2}\right) \tag{15}$$

Die gesuchte bedingte Wahrscheinlichkeit berechnet sich folgendermaßen: Wie schon erwähnt, teilt sich die Energie des Informationsbits auf die k übertragenen Kanalbits auf und E_b muss durch k dividiert werden. Gesucht ist nun die Wahrscheinlichkeit

$$P\left(N > \sqrt{\frac{E_b}{k}}\right) \tag{16}$$

Wird die Verteilung auf die Standardnormalverteilung Z zurückgeführt, ergibt sich durch die Eigenschaften von N folgendes. σ steht für die Standardabweichung.

$$N = \sigma \cdot Z = \sqrt{\frac{E_b}{2}} \cdot Z \tag{17}$$

Eingesetzt in Gleichung (16) ergibt sich nun eine Standardnormalverteilung, die $(0, 1)$ verteilt ist. Dieser Schritt wird für eine leichtere Berechnung benötigt.

$$P\left(N > \sqrt{\frac{E_b}{k}}\right) = P\left(Z > \sqrt{\frac{E_b}{N_0} \cdot \frac{2}{k}}\right) \tag{18}$$

Dieses Ergebnis kann man jetzt mit Hilfe der Errorfunktion berechnen. Diese ist Teil von Matlab und Octave.

$$p_e = P\left(N > \sqrt{\frac{E_b}{k}}\right) = \frac{1}{2} \cdot \operatorname{erfc}\left(\frac{\sqrt{\frac{E_b}{N_0} \cdot \frac{2}{k}}}{\sqrt{2}}\right) \tag{19}$$

Ein Codewort wird jetzt falsch decodiert, wenn mehr als die Hälfte der Kanalbits falsch empfangen wird. Die Wahrscheinlichkeit dafür kann man mit der Binomialverteilung berechnen. Die Wahrscheinlichkeit für ein falsch decodiertes Codewort wird mit p_c bezeichnet und berechnet dich zu

$$p_c = \sum_{i=\lceil \frac{k}{2} \rceil}^{k} \binom{k}{i} \cdot p_e^i \cdot (1 - p_e)^{k-i} \tag{20}$$

4.2 Simulation der BER-Kurven

In den folgenden Abbildungen werden die analytischen und simulierten Ergebnisse gezeigt und diskutiert. Die Anzahl der Wiederholungen wurden so gewählt, dass perfekte Codes entstehen. Die durchgehenden Kurven in den Graphen stellen die Berechnungen dar, die simulierten Fehler werden mit einem Kreuz markiert. Zum Vergleich wurde auch immer die uncodierte BPSK Kurve dargestellt.

Abbildung 6 zeigt die berechneten und simulierten BER- Kurven. Dabei ist interessant, dass mit steigendem Wiederholungsgrad der Codiergewinn schlechter wird, also immer größerer Codierverlust auftritt. Später wird gezeigt, dass auch mit optimaler Decodierung mit weicher Entscheidung kein Codiergewinn erzielt wird und genau 0 ist. Die Kurven in Abbildung 5 zeigen die BER bei konstanter SNR und steigendem Wiederholungsgrad k und man sieht, dass die BER bei steigendem k schlechter wird. Aus diesem Grund ergeben sich die erwähnten Kurvenverläufe in Abbildung 6. Übertragen wurden bei dieser Simulation $k \cdot 2.000.000$ bits. Abbildung 7 zeigt die BER-Kurven für den Fall, dass die Energie des Informationsbits nicht auf alle k Kanalbits aufgeteilt wird, sondern dass jedes Kanalbit mit Energie $\mathrm{E_b}$ übertragen wird. Man sieht, dass die Kurven deutlich besser werden, wobei jedoch die benötigte Energie viel größer wird.

5 Decodierung mit weicher Entscheidung

Bei dieser Art von Decodierung gehen auch die Amplituden der empfangenen Signale in die Entscheidung mit ein. Wie man später bei den Simulationen sehen kann, wird dadurch ein Codiergewinn gegenüber Decodierung mit harter Entscheidung erzielt. Eine bessere Quantisierung der Amplitude hängt mit einem höheren Hardwareaufwand des Decodierers zusammen. Für die Berechnung und die Simulation wurde eine unendlich hohe Quantisierung zugelassen. D.h.: Die A/D-Wandlung wurde vernachlässigt.

5.1 Berechnung der BER- Kurve

Bei der Berechnung dieser Kurven kann man jetzt von Summen von unabhängigen Normalverteilungen ausgehen, welche wieder normalverteilt sind. Für diese neue Verteilung werden einfach die Mittelwerte und die Varianzen addiert (vgl. [7]). Nachdem das Rauschen immer

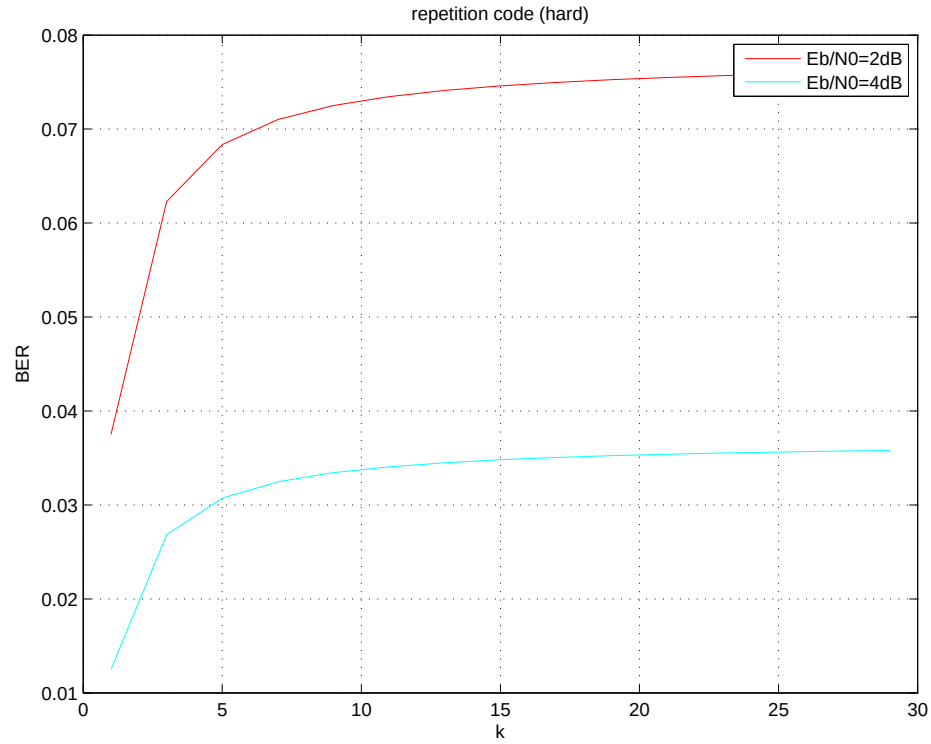

Abbildung 5: Wiederholungscode bei konstanter SNR

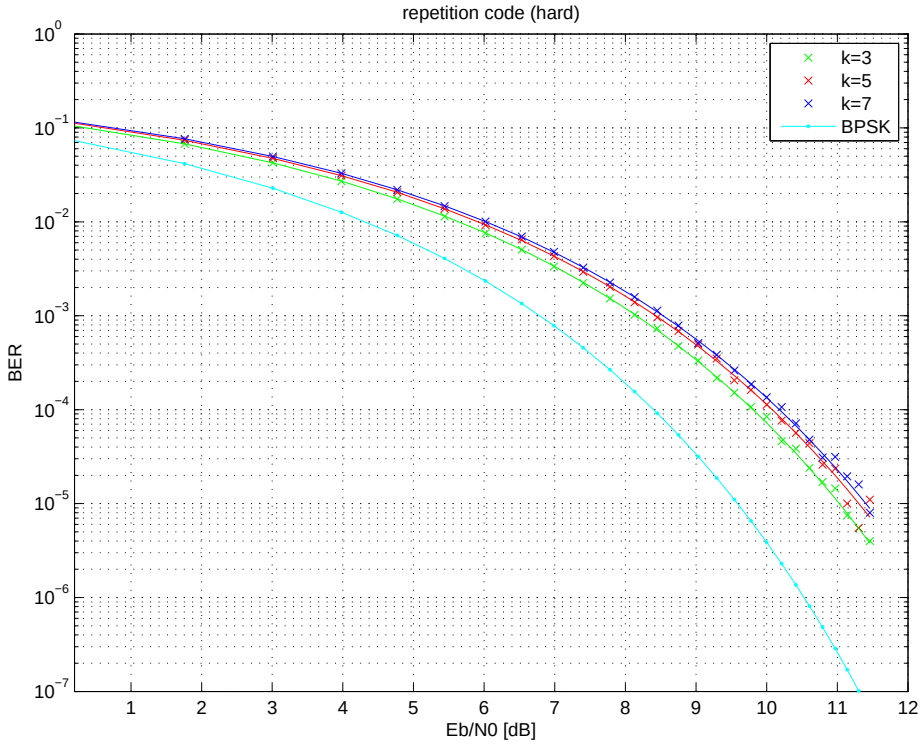

Abbildung 6: Wiederholungscode mit harter Entscheidung

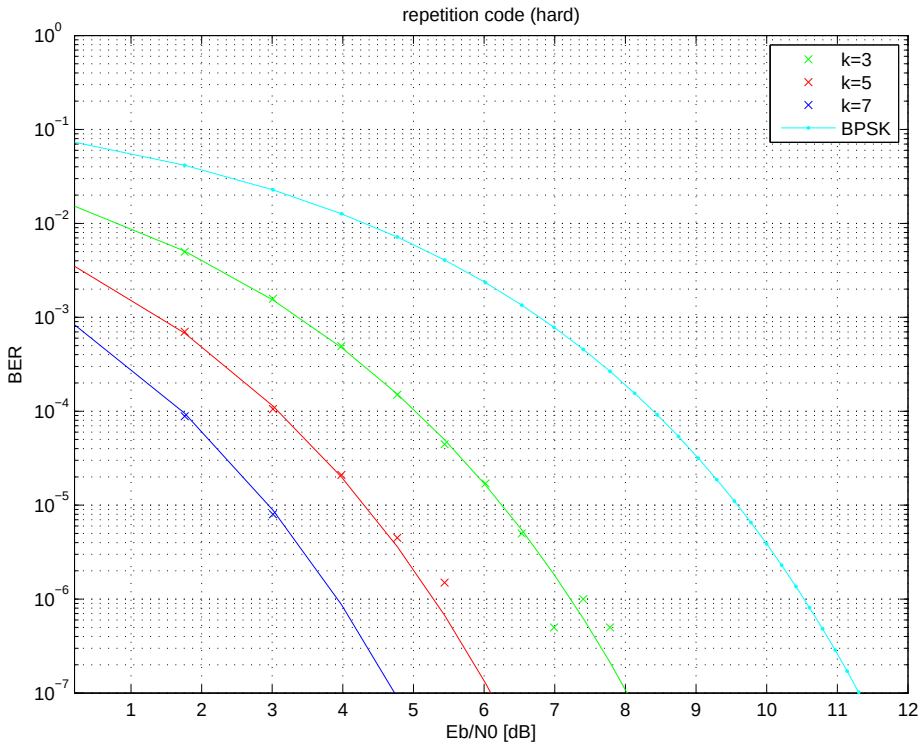

Abbildung 7: Repetitioncode mit vernachlässigter Coderate

den gleichen Mittelwert und die gleiche Varianz besitzt, ergibt sich eine $\left(0, k \cdot \frac{N_0}{2}\right)$) Verteilung, wobei k die Anzahl der Wiederholungen ist. Dieses gesamte Rauschen wird nun der Addition der unverrauschten Signale überlagert. Somit erhält man die gleiche Berechnung, wie beim harten Decodieren mit neuen Parametern. Das Rauschen wird hier mit der Zufallsvariable N_k bezeichnet.

$$N_k \sim \left(0, k \cdot \frac{N_0}{2}\right) \tag{21}$$

Die gesuchte bedingte Wahrscheinlichkeit berechnet sich folgendermaßen.

$$P\left(N_k > k \cdot \sqrt{\frac{E_b}{k}}\right) \tag{22}$$

Zurückgeführt auf die Standardnormalverteilung ergibt sich:

$$P\left(N_k > k \cdot \sqrt{\frac{E_b}{k}}\right) = P\left(Z > k \cdot \sqrt{\frac{E_b}{N_0} \cdot \frac{2}{k^2}}\right) \tag{23}$$

Dabei kürzen sich die Wiederholungen heraus und die Fehlerwahrscheinlichkeit entspricht der eines BPSK modulierten Signals ohne Codierung.

$$p_c = P\left(N_k > k \cdot \sqrt{\frac{E_b}{k}}\right) = \frac{1}{2} \cdot \operatorname{erfc}\left(\frac{\sqrt{2 \cdot \frac{E_b}{N_0}}}{\sqrt{2}}\right) = \frac{1}{2} \cdot \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{24}$$

5.2 Simulation der BER-Kurven

Wie schon erwähnt und jetzt auch berechnet, fallen die BER- Kurven genau zusammen, was auch der Kurve des uncodierten BPSK entspricht (siehe Abbildung 8). Die beiden Decodierer können nun also in Abbildung 6 verglichen werden. Es handelt sich hierbei jedoch nur

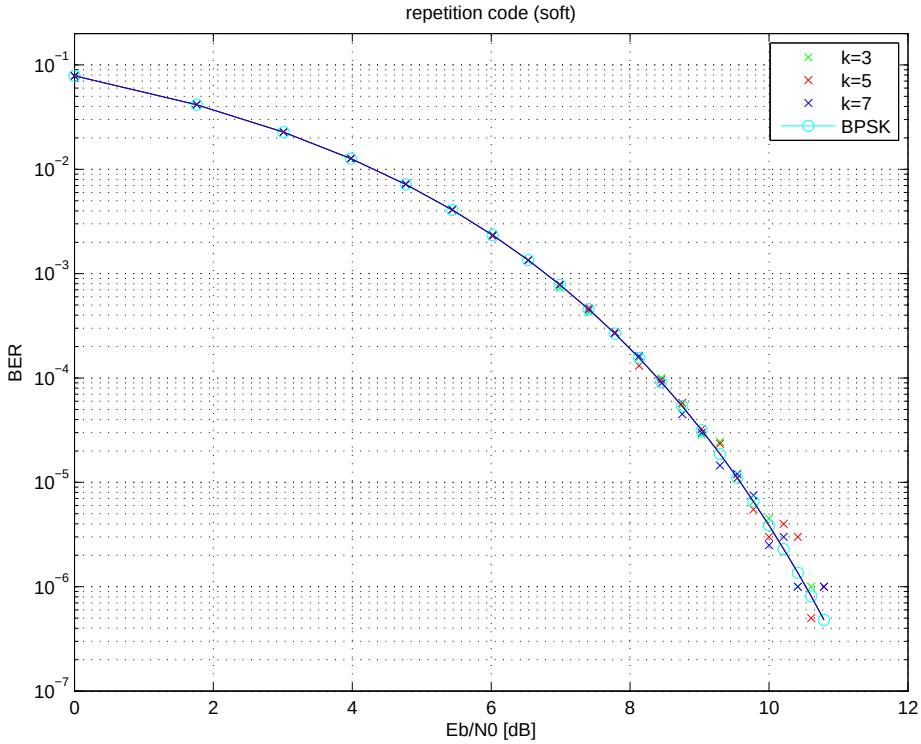

Abbildung 8: Repetitioncode mit weicher Decodierung

um eine theoretische Darstellung, da für solche Kurven unendlich viele Quantisierungsstufen notwendig wären. Um der Realität näher zu kommen, wurden Kurven mit zwei, bzw. drei bit simuliert. D.h. es wird zwischen vier, bzw. acht Werten am Empfänger unterschieden. Abbildungen 9 und 10 zeigen diese Kurven im Vergleich zu einer harten Decodierung und zur BPSK- Kurve. Es ist zu erkennen, dass die 3 bit- Quantisierung schon sehr nahe an die optimale BPSK- Kurve herankommt.

6 Obere und untere Schranken für die BER

Im Folgenden wird die Fehlerwahrscheinlichkeit mit der bedingten Entropie in Verbindung gebracht und daraus die Grenzen berechnet.

6.1 Fano Ungleichung

Dabei wird von einer Zufallsvariable Y ausgegangen, wobei der Wert einer korrelierten Zufallsvariable X geschätzt wird. Die Fano Ungleichung setzt nun die Fehlerwahrscheinlichkeit durch Schätzen der Zufallsvariable X in Beziehung mit der bedingten Entropy $H(X|Y)$. Es kann gezeigt werden, dass die Fehlerwahrscheinlichkeit bei der Abschätzung von X durch Y

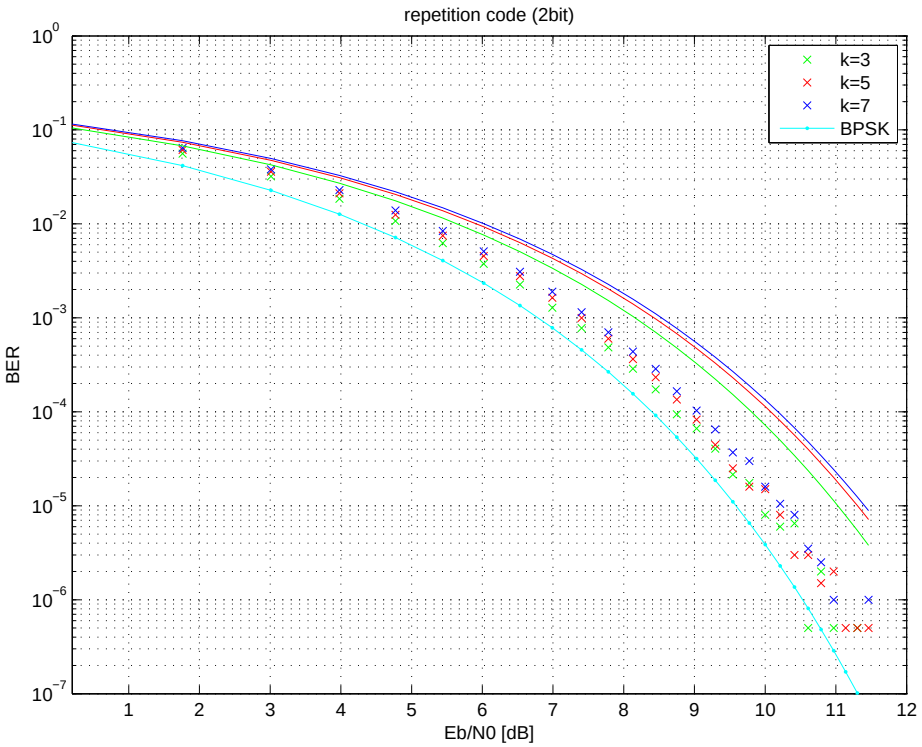

Abbildung 9: Repetitioncode mit 2 bit Quantisierung

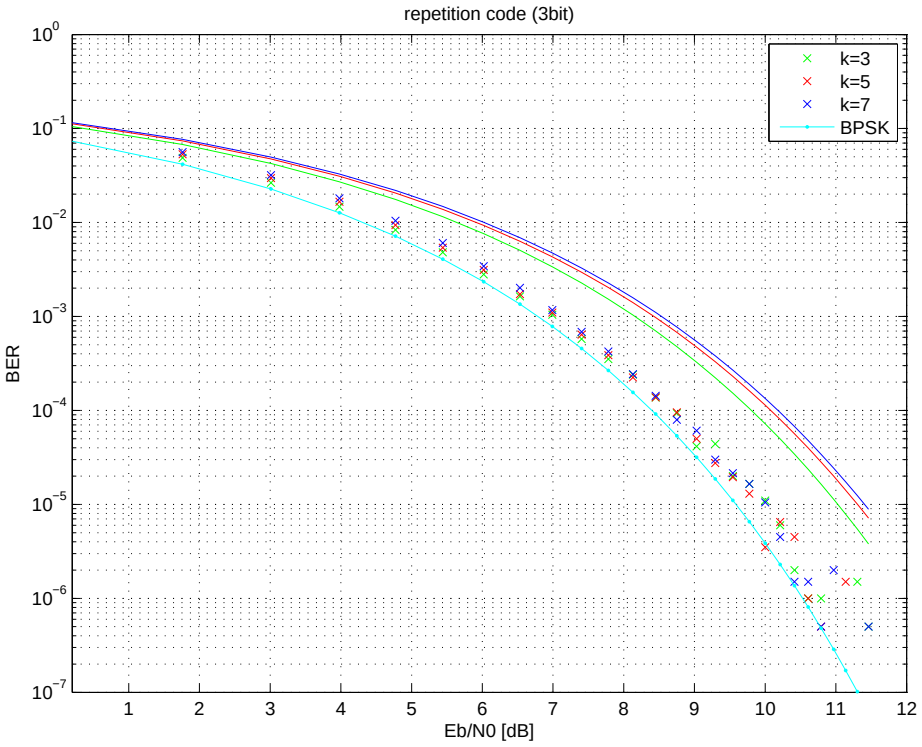

Abbildung 10: Repetitioncode mit 3 bit Quantisierung

nur dann null ist, wenn $H(X|Y) = 0$. Man kann nun annehmen, dass die Fehlerwahrscheinlichkeit der Abschätzung nur dann klein ist, wenn die bedingte Entropy klein ist. Die Fano Ungleichung beruht auf dieser Idee.

Die Fano Ungleichung besagt nun folgendes: Für jeden Schätzer $\hat{X}$, sodass $X \to Y \to \hat{X}$, mit $P_e = Pr(X \neq \hat{X})$ gilt [9]:

$$H(P_e) + P_e \log |\mathcal{X}| \geq H(X|\hat{X}) \geq H(X|Y) \tag{25}$$

wobei $\mathcal{X}$ für das Alphabet der Eingangssymbole steht.

Die Ungleichung kann bestärkt werden, wenn der Schätzer $\hat{X}$ Werte in $\mathcal{X}$ annimmt, sodass gilt [9]:

$$H(P_e) + P_e \log(|\mathcal{X}| - 1) \geq H(X|Y) \tag{26}$$

Für die Simulationen wurden binäre Eingangssymbole verwendet, d.h.: das Alphabet besteht aus 0 und 1, also ist $|\mathcal{X}| = 2$. Berücksichtigt man dies in der Ungleichung, so fällt der Term mit dem Logarithmus heraus und man erhält:

$$H(P_e) \geq H(X|Y) \tag{27}$$

6.2 Berechnung der unteren Grenze

Zur Berechnung der Entropien ist es notwendig, die Wahrscheinlichkeiten zu kennen. Die Wahrscheinlichkeit $p(y|x)$ ist eine Funktion vom SNR bzw. von $\frac{E_b}{N_0}$ und den Schwellwerten. Zur Veranschaulichung sei wieder auf Abbildung 4 verwiesen. Im Gegensatz dazu wird aber nicht mehr notwendigerweise 0 als Schwellwert benutzt, sondern es können beliebige und beliebig viele betrachtet werden. Aus den verschiedenen Werten des SNR ergeben sich verschiedene PDFs $f_{Y|X}$. Zwischen den verschiedenen Schwellwerten ergeben sich Bereiche mit bestimmten Wahrscheinlichkeiten $p(y|x)$. Wird mit t die Anzahl der Schwellwerte bezeichnet, so ergeben sich $t + 1$ Bereiche. Auf eine detaillierte Berechnung wird hier verzichtet, da sie der Berechnung beim Schwellwert 0 (mit harter Entscheidung) ähnlich ist.

Setzt man die Schwellwerte symmetrisch zum Nullpunkt, was durchaus sinnvoll ist, so erhält man als Beispiel mit zwei Schwellwerten folgende Form, wobei y_1, y_2 und y_3 die Quantisierungswerte darstellen.

$p(y\|x)$	y_1	y_2	y_3
$x = -1$	p_1	p_2	p_3
$x = +1$	p_3	p_2	p_1

Bis jetzt wurde noch nicht berücksichtigt, dass zur Entscheidungsfindung ja k Wiederhohlungen betrachtet werden. Dazu wird nun $p(y^k|x)$ eingeführt, welche alle möglichen Kombinationen von k Werten beinhaltet. Als Beispiel wird hier Wiederholungscode mit $k = 3$ und eine weiche Entscheidung mit zwei Schwellwerten betrachtet.

$p(y^1, y^2, y^3\|x)$	y_1,y_1,y_1	y_1,y_1,y_2	...	y_3,y_3,y_3
x_1	$p_1 \cdot p_1 \cdot p_1$	$p_1 \cdot p_1 \cdot p_2$	...	$p_3 \cdot p_3 \cdot p_3$
x_2	$p_3 \cdot p_3 \cdot p_3$	$p_2 \cdot p_3 \cdot p_3$	...	$p_1 \cdot p_1 \cdot p_1$

Für die Berechnung der Verbundwahrscheinlichkeit wurde der Multiplikationssatz werwendet, welcher besagt, dass $p(x, y) = p(x) \cdot p(y|x)$ ist [7]. Die beiden Eingangswerte treten mit

$p(y^1, y^2, y^3, x)$	y_1,y_1,y_1	y_1,y_1,y_2	...	y_3,y_3,y_3
x_1	$\frac{1}{2} \cdot p_1 \cdot p_1 \cdot p_1$	$\frac{1}{2} \cdot p_1 \cdot p_1 \cdot p_2$	...	$\frac{1}{2} \cdot p_3 \cdot p_3 \cdot p_3$
x_2	$\frac{1}{2} \cdot p_3 \cdot p_3 \cdot p_3$	$\frac{1}{2} \cdot p_2 \cdot p_3 \cdot p_3$	...	$\frac{1}{2} \cdot p_1 \cdot p_1 \cdot p_1$

Tabelle 1: $p(y^1, y^2, y^3, x)$ bei drei Wiederhohlungen

gleicher Wahrscheinlichkeit auf, wodurch $p(x) = \frac{1}{2}$ ist (Tabelle 1). Die Wahrscheinlichkeit $p(y)$ kann mit Hilfe der totalen Wahrscheinlichkeit berechnet werden [7]. Dies entspricht dem Aufsummieren der einzelnen Spalten der Tabelle 1.

$$p(y) = \sum_x p(x) \cdot p(y|x) \tag{28}$$

Jetzt kann man die Entropien $H(Y)$, $H(X)$ und $H(Y|X)$ mit den schon bekannten Formeln berechnen. Aus den Formeln für die Transinformation kann dann die gewünschte Entropie $H(X|Y)$ berechnet werden.

$$T = H(X) - H(X|Y) \tag{29}$$

$$T = H(Y) - H(Y|X) \tag{30}$$

$$H(X|Y) = H(X) + H(Y|X) - H(Y) \tag{31}$$

Schlussendlich kann P_e aus der Ungleichung $H(P_e) \geq H(X|Y)$ bestimmt werden.

6.3 Berechnung der oberen Grenze

Es existiert ein weiterer Zusammenhang zwischen der bedingten Entropy $H(X|Y)$ und der Fehlerwahrscheinlichkeit P_e, welcher als Abschätzung nach oben verwendet werden kann. Daraus erhält man nun neben der bereits berechneten unteren Grenze eine obere Grenze für die Fehlerwahrscheinlichkeit [10].

$$H(X|Y) \geq 2 \cdot \log(2) \cdot P_e = 2 \cdot P_e \tag{32}$$

6.4 Darstellung der Grenzen

Wie schon gezeigt, sind die BER-Kurven von der Lage und der Anzahl der Schwellwerte abhängig. Bei der Berechnung der unteren Grenze wurde die Fehlerwahrscheinlichkeit sukzessive verkleinert, bis die Grenze gefunden wurde. Abbildungen 11 und 12 zeigen die simulierten Werte sowie die durch durchgezogenen Linien gekennzeichneten Grenzen mit drei bzw. fünf Schwellwerten. Orientiert man sich an der BPSK Kurve, so erkennt man den Unterschied zwischen den beiden Decodierern. Wie zu erwarten, wird das Verhalten mit Erhöhung der Schwellwerte verbessert. Als Beispiel werden die Schwellwerte und die Gewichtung der Stufen bei der Decodierung mit drei Schwellwerten gezeigt. Ist das empfangene Signal zwischen 0 und $0,44$, so wird es mit $0,\dot{3}$ gewichtet. Ist es größer als $0,44$, so wird es mit 1 gewichtet. Bei dem vorliebenden BPSK-Signal werden diese Schwellwerte einfach gespiegelt und mit negativem Vorzeichen gewichtet und man erhält den vollständigen Decodierer. Beim Empfang aller k Kanalbits wird dann eine Entscheidung getroffen.

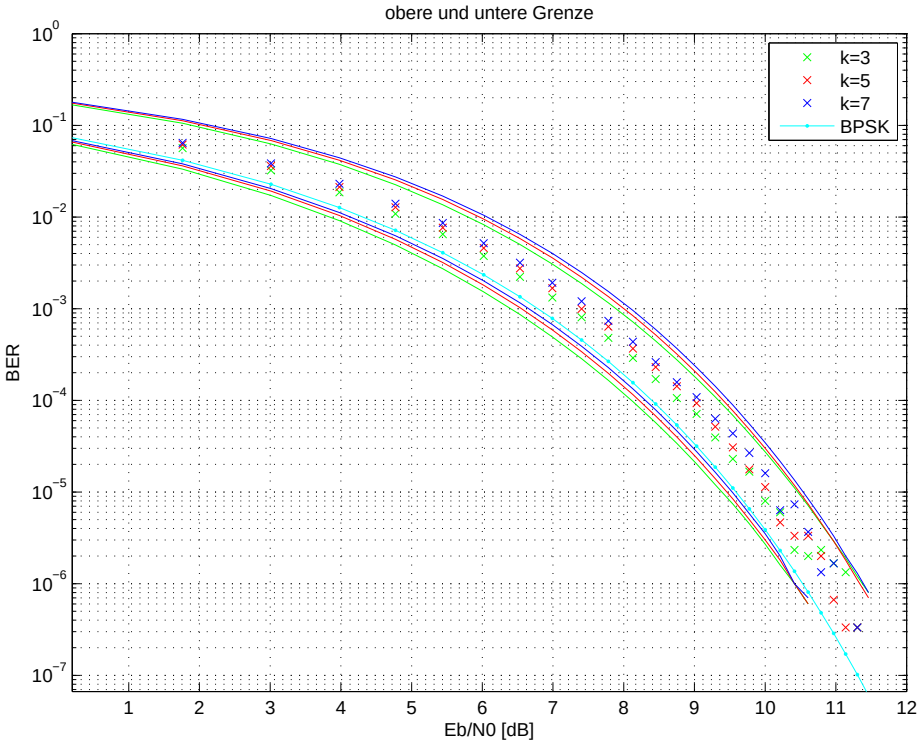

Abbildung 11: Decodierung mit drei Schwellwerten

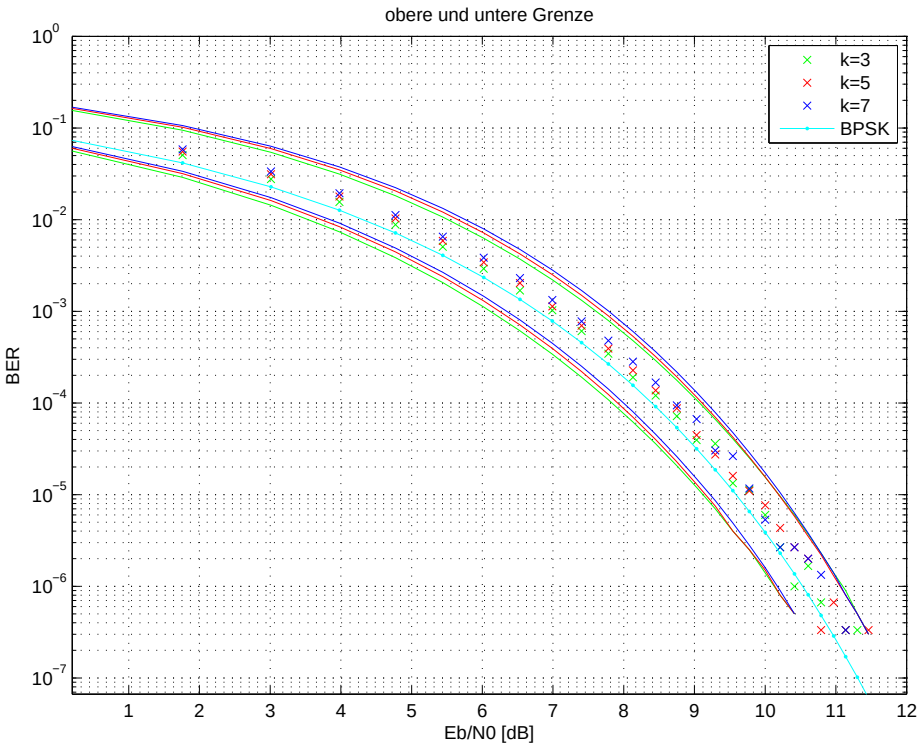

Abbildung 12: Decodierung mit fünf Schwellwerten

7 Matlab (Octave) Code

7.1 Decodierung mit harter Entscheidung

```
% Goertschacher Lukas
% repetition code hard decoding
% 2012

clear all
close all
% generating a binary randomvector with length n and equal probability of 0 and 1
n=2000000;
inputv = round(rand(1,n));
%will consist all ber's
berall=[];
adecvbiterrorall=[];
% specify the number of repetitions
for m=3:2:7

%%% Encoding

% copying all elements m- times to get the wanted repetition
        encv = inputv(ones(1,m),:);
        encv = encv(:).';

% replace the elemts that are zero with -1, to get the antipodal representation of the signal.
% now we have the signal with bit energy of 1.
        encv(encv==0)=-1;
        %encv=encv./sqrt(m);
        for i=2:1:28

%  Eb/N0, energy per bit per noise PSD.
% note that the energy per bit Eb is 1!
                EbtoN0(1,i)=i/2;
                N0=1/EbtoN0(1,i);

% generating a noise vector with the length of the inputvector
% AWGN is N~(0,N0/2) distributet
% randn gives a N~(0,1) distribution
% multiply with m to observe the redundancy
                noise =sqrt(N0/2*m)* randn(1,length(encv));

% adding the noise to the coded vector enc
                recv= encv+noise;

%  Decoding
% make a trashhold decision if 0 or 1
                recv(recv>0)=1;
                recv(recv<0)=0;

% decoding the vektor recv
                decv =reshape(recv, [m,length(inputv)]);

% summing up the zeros of each column and decide for 0 or 1
                outputv=sum(decv==0)<m/2;
                biterror(1,i)=sum(xor(inputv,outputv)==1);
        end

% compute the ber analytically
        abiterror=1/2*erfc(sqrt(2*EbtoN0/m)/sqrt(2));
        adecvbiterror=zeros(1,length(abiterror));

        for j=ceil(m/2):m
                adecvbiterror=adecvbiterror+nchoosek(m,j) .*abiterror.^(j).*(1.-abiterror).^(m-j);
        end
        adecvbiterrorall=[adecvbiterrorall;adecvbiterror];
% calculate the ber of BPSK
    bpsk=1/2*erfc(sqrt(2*EbtoN0)/sqrt(2));
% calculate the logarithm of EbtoN0
        logEbtoN0=(EbtoN0~=0).*(10*log10(EbtoN0));
% calculate the biterrorrate
        ber=biterror/n;
        ber(ber==0)=NaN;
% berall consists of the ber's of each loop
        berall=[berall;ber];

end

color=['r','c','b','b','g','c','c'];
% plotting the curves with logerithm y- scale
% plot the simulation and the analytic computation
figure;
semilogy(logEbtoN0(~isnan(berall(1,:))),berall(1,(~isnan(berall(1,:)))),['x',color(5)],...
    logEbtoN0(~isnan(berall(2,:))),berall(2,(~isnan(berall(2,:)))),['x',color(1)],...
    logEbtoN0(~isnan(berall(3,:))),berall(3,(~isnan(berall(3,:)))),['x',color(3)],...
    logEbtoN0,bpsk,['.-',color(2)],...
    logEbtoN0,adecvbiterrorall(1,:),['-',color(5)],...
    logEbtoN0,adecvbiterrorall(2,:),['-',color(1)],...
```

```
85      logEbtoN0,adecvbiterrorall(3,:),['-',color(3)])
86  axis([0.2 12 2/(10*n) 1])
87  grid on;
88  title('repetition code (hard)');
89  legend('k=3','k=5','k=7','BPSK')
90  xlabel('Eb/N0 [dB]');
91  ylabel('BER');
92
93  print('-dpdf','rep_hard_cr.pdf')
94
95  % plot the simulation only
96  figure;
97  semilogy(logEbtoN0(~isnan(berall(1,:))),berall(1,(~isnan(berall(1,:)))),['x',color(5)],...
98      logEbtoN0(~isnan(berall(2,:))),berall(2,(~isnan(berall(2,:)))),['x',color(1)],...
99      logEbtoN0(~isnan(berall(3,:))),berall(3,(~isnan(berall(3,:)))),['x',color(3)],...
100     logEbtoN0,bpsk,['.-',color(2)])
101 axis([0.2 12 2/(10*n) 1])
102 grid on;
103 title('repetition code (hard)');
104 legend('k=3','k=5','k=7','BPSK')
105 xlabel('Eb/N0 [dB]');
106 ylabel('BER');
107
108 print('-dpdf','rep_hard_real_cr.pdf')
109
110 % plot the analytic computation only
111 figure;
112 semilogy(logEbtoN0,adecvbiterrorall(1,:),['-',color(5)],...
113     logEbtoN0,adecvbiterrorall(2,:),['-',color(1)],...
114     logEbtoN0,adecvbiterrorall(3,:),['-',color(3)],...
115     logEbtoN0,bpsk,['.-',color(2)])
116 axis([0.2 12 2/(10*n) 1])
117 grid on;
118 title('repetition code (hard)');
119 legend('k=3','k=5','k=7','BPSK')
120 xlabel('Eb/N0 [dB]');
121 ylabel('BER');
122 print('-dpdf','rep_hard_anal_cr.pdf')
```

7.2 Decodierung mit weicher Entscheidung

```
1  % Goertschacher Lukas
2  % repetition code with infinite resolution
3  % 2012
4  clear all
5  close all
6
7  % generating a binary randomvector with length n and equal probability of 0 and 1
8  n=2000000;
9  inputv = round(rand(1,n));
10 %will consist all ber's
11 berall=[];
12 abiterrorall=[];
13 % specify the number of repetitions
14 for m=3:2:7
15
16 % Encoding
17 % copying all elements m- times to get the wanted repetition
18         encv = inputv(ones(1,m),:);
19         encv = encv(:).';
20
21 % replace the elemts that are zero with -1, to get the antipodal representation of the signal.
22 % now we have the signal with bit energy of 1.
23         encv(encv==0)=-1;
24
25 % define the parameters for the loop. steps 1/a and range b/8
26         a=2;
27         b=24;
28
29         for i=a:1:b
30
31 %  Eb/N0, energy per bit per noise PSD.
32 % note that the energy per bit Eb is 1!
33
34                 EbtoN0(1,i-a+1)=i/2;
35                 N0=1/EbtoN0(1,i-a+1);
36
37 % generating a noise vector with the length of the inputvector
38 % AWGN is N~(0,N0/2) distributet
39 % randn gives a N~(0,1) distribution
40 %multiply N0 with m to reduse the bit energy
41                 noise =sqrt(N0*m/2)* randn(1,length(encv));
42
43 % adding the noise to the coded vector enc
44                 recv= encv+noise;
45
46 %  Decoding
47 % softdecoding the vektor recv
```

```
48                  decv =reshape(recv, [m,length(inputv)]);
49                  outputv=(sum(decv)/m)>0;
50
51 % summing up the zeros of each column and decide for 0 or 1
52                  biterror(1,i-a+1)=sum(xor(inputv,outputv)==1);
53
54          end
55
56 % softdecoding analytically
57 % one m is to reduse EbtoN0 because the higher redundancy
58          abiterror=1/2*erfc(m*sqrt(2*EbtoN0/m/m)/sqrt(2));
59          abiterrorall=[abiterrorall;abiterror];
60 % calculate the ber of BPSK
61     bpsk=1/2*erfc(sqrt(2*EbtoN0)/sqrt(2));
62
63 % calculate the logarithm of EbtoN0
64          logEbtoN0=(EbtoN0~=0).*(10*log10(EbtoN0));
65 % calculate the biterrorrate
66          ber=biterror/n;
67          ber(ber==0)=NaN;
68 % berall consists of the ber's of each loop
69          berall=[berall;ber];
70
71 end
72
73 color=['r','c','b','b','g','c','c'];
74 % plotting the curves with logerithm y- scale
75 figure;
76 semilogy(logEbtoN0(~isnan(berall(1,:))),berall(1,(~isnan(berall(1,:)))),['x',color(5)],...
77          logEbtoN0(~isnan(berall(2,:))),berall(2,(~isnan(berall(2,:)))),['x',color(1)],...
78          logEbtoN0(~isnan(berall(3,:))),berall(3,(~isnan(berall(3,:)))),['x',color(3)],...
79          logEbtoN0,bpsk,['-o',color(2)],...
80              logEbtoN0,abiterrorall(1,:),['-',color(5)],...
81              logEbtoN0,abiterrorall(2,:),['-',color(1)],...
82              logEbtoN0,abiterrorall(3,:),['-',color(3)])
83 axis([0 12 2/(10*n) 0.2])
84 grid on;
85 legend('k=3','k=5','k=7','BPSK')
86 title('repetition code (soft)');
87 xlabel('Eb/N0 [dB]');
88 ylabel('BER');
89
90 print('-dpdf','rep_soft.pdf')
```

7.3 Decodierung mit verschiedenen Auflösungen

```
1 % Goertschacher Lukas
2 % repetition code soft decoding with different tresholds with call up fano
3 % 2012
4 clear all
5 close all
6 format long
7
8 % generating a binary randomvector with length n and equal probability of 0 and 1
9 n=3000000;
10 % quantization of decoding
11 quant=4;
12 inputv = round(rand(1,n));
13 %will consist all ber's
14 berall=[];
15 adecvbiterrorall=[];
16 % specify the number of repetitions
17 for m=3:2:7
18
19 %%% Encoding
20
21 % copying all elements m- times to get the wanted repetition
22          encv = inputv(ones(1,m),:);
23          encv = encv(:).';
24
25 % replace the elemts that are zero with -1, to get the antipodal representation of the signal.
26 % now we have the signal with bit energy of 1.
27          encv(encv==0)=-1;
28          %encv=encv./sqrt(m);
29          for i=1:1:28
30
31 %  Eb/N0, energy per bit per noise PSD.
32 % note that the energy per bit Eb is 1!
33                  EbtoN0(1,i)=i/2;
34                  N0=1/EbtoN0(1,i);
35
36 % generating a noise vector with the length of the inputvector
37 % AWGN is N~(0,N0/2) distributet
38 % randn gives a N~(0,1) distribution
39 % multiply with m to observe the redundancy
40                  noise =sqrt(N0/2*m)* randn(1,length(encv));
41
42 % adding the noise to the coded vector enc
```

```
43                  recv= encv+noise;
44
45 %  Decoding
46 % 2 or 3 bit softdecoding or 5 tresholds
47          if quant==3;
48              recv(recv>5*0.33333/2)=1;
49              recv(recv>3*0.33333/2 & recv<5*0.33333/2)=0.6;
50              recv(recv>0.33333/2 & recv<3*0.33333/2)=0.2;
51              recv(recv>0 & recv<0.33333/2)=0.1;
52              recv(recv>-0.33333/2 & recv<0)=-0.1;
53              recv(recv<-3*0.33333/2 & recv>-0.33333/2)=-0.2;
54              recv(recv<-5*0.33333/2 & recv>-3*0.33333/2)=-0.6;
55              recv(recv<-5*0.33333/2)=-1;
56          elseif quant==2;
57              recv(recv>0.44)=1;
58              recv(recv>0 & recv<0.44)=0.3333;
59              recv(recv>-0.44 & recv<0)=-0.3333;
60              recv(recv<-0.44)=-1;
61          else
62              recv(recv>0.66666)=1;
63              recv(recv>0.24 & recv<0.66666)=0.4;
64              recv(recv>0 & recv<0.24)=0.1;
65              recv(recv>-0.24 & recv<0)=-0.1;
66              recv(recv<-0.66666 & recv>-0.24)=-0.4;
67              recv(recv<-0.66666)=-1;
68          end
69
70 % decoding the vektor recv
71                  decv =reshape(recv, [m,length(inputv)]);
72
73 % summing up the zeros of each column and decide for 0 or 1
74          outputv=sum(decv)>0;
75                  %outputv=sum(decv==0)<m/2;
76                  biterror(1,i)=sum(xor(inputv,outputv)==1);
77          end
78
79 % compute the ber analytically
80          abiterror=1/2*erfc(sqrt(2*EbtoN0/m)/sqrt(2));
81          adecvbiterror=zeros(1,length(abiterror));
82
83          for j=ceil(m/2):m
84                  adecvbiterror=adecvbiterror+nchoosek(m,j) .*abiterror.^(j).*(1.-abiterror).^(m-j);
85          end
86          adecvbiterrorall=[adecvbiterrorall;adecvbiterror];
87 % calculate the ber of BPSK
88      bpsk=1/2*erfc(sqrt(2*EbtoN0)/sqrt(2));
89 % calculate the logarithm of EbtoN0
90          logEbtoN0=(EbtoN0~=0).*(10*log10(EbtoN0));
91      size(logEbtoN0)
92
93 % calculate the biterrorrate
94          ber=biterror/n;
95          ber(ber==0)=NaN;
96 % berall consists of the ber's of each loop
97          berall=[berall;ber];
98
99 end
100 % execute the fanoinequaltity
101 [Perror3 Perror5 Perror7 ub3 ub5 ub7]=fano(EbtoN0);
102
103 color=['r','c','b','b','g','c','c'];
104 % plotting the curves with logerithm y- scale
105 % plot the simulation and the analytic computation
106 figure;
107 semilogy(logEbtoN0(~isnan(berall(1,:))),berall(1,(~isnan(berall(1,:)))),['x',color(5)],...
108      logEbtoN0(~isnan(berall(2,:))),berall(2,(~isnan(berall(2,:)))),['x',color(1)],...
109      logEbtoN0(~isnan(berall(3,:))),berall(3,(~isnan(berall(3,:)))),['x',color(3)],...
110      logEbtoN0,bpsk,['.-',color(2)],...
111      logEbtoN0,adecvbiterrorall(1,:),['-',color(5)],...
112      logEbtoN0,adecvbiterrorall(2,:),['-',color(1)],...
113      logEbtoN0,adecvbiterrorall(3,:),['-',color(3)])
114 axis([0.2 12 2/(10*n) 1])
115 grid on;
116 title('repetition code (hard)');
117 legend('k=3','k=5','k=7','BPSK')
118 xlabel('Eb/N0 [dB]');
119 ylabel('BER');
120
121 %print('-dpdf','rep_hard_3bit.pdf')
122
123 % plot the upper and lower bound
124 figure;
125 semilogy(logEbtoN0(~isnan(berall(1,:))),berall(1,(~isnan(berall(1,:)))),['x',color(5)],...
126      logEbtoN0(~isnan(berall(2,:))),berall(2,(~isnan(berall(2,:)))),['x',color(1)],...
127      logEbtoN0(~isnan(berall(3,:))),berall(3,(~isnan(berall(3,:)))),['x',color(3)],...
128      logEbtoN0,bpsk,['.-',color(2)],logEbtoN0,Perror3,['-','g'],...
129      logEbtoN0,Perror5,['-','r'],logEbtoN0,Perror7,['-','b'],...
130      logEbtoN0,ub3,['-','g'],logEbtoN0,ub5,['-','r'],logEbtoN0,ub7,['-','b'])
131 axis([0.2 12 2/(10*n) 1])
132 grid on;
```

```
title('obere_und_untere_Grenze');
legend('k=3','k=5','k=7','BPSK')
xlabel('Eb/N0_[dB]');
ylabel('BER');

print('-dpdf','rep_soft_5t_ublb2.pdf')

% plot the analytic computation only
figure;
semilogy(logEbtoN0,adecvbiterrorall(1,:),['-',color(5)],...
    logEbtoN0,adecvbiterrorall(2,:),['-',color(1)],...
    logEbtoN0,adecvbiterrorall(3,:),['-',color(3)],...
    logEbtoN0,bpsk,['.-',color(2)],logEbtoN0,Perror3,['-x','g'],...
    logEbtoN0,Perror5,['-x','r'],logEbtoN0,Perror7,['x','b'])
axis([0.2 12 2/(10*n) 1])
grid on;
title('repetition_code_(hard)');
legend('k=3','k=5','k=7','BPSK')
xlabel('Eb/N0_[dB]');
ylabel('BER');
%print('-dpdf','fano4.pdf')
```

7.4 Fano Ungleichung

```
% Goertschacher Lukas
% repetition code fano inequality
% 2012

function[Pe3lb Pe5lb Pe7lb Pe3ub Pe5ub Pe7ub]=fano(EbtoN0)

% give the trashholds
t=[-0.44 0 0.44];
t=[-2*0.33333 -0.24 0 0.24 2*0.33333];
% Pt consits of the errorprob from the trashholds to infinity
Pt=[];
% Ptm consists of the errorprob for all m
Ptm=[];
for m=3:2:7
    for i=1:length(t)
        Pt(i,:)=1/2*erfc(sqrt(EbtoN0/m)*abs((t(i)-1)));
    end
    Ptm=[Ptm;Pt];
end
Ptm=Ptm';
Pcondall=[];
% compute the errorprob in the regions devidet from the trashholds Pcond=P(Y|X).
for i=1:length(t):3*length(t)
    Ptm1=Ptm(1:length(EbtoN0),i:i+length(t)-1);
    Pcond(:,1)=Ptm1(:,1);
    Pcond(:,length(t)+1)=1.-Ptm1(:,length(t));
    for k=1:length(t)-1
        Pcond(:,k+1)=Ptm1(:,k+1)-Ptm1(:,k);
    end
        % consists of the conditional probs for all EbtoN0 and all k
        Pcondall=[Pcondall reshape([Pcond';fliplr(Pcond)'],length(t)+1,2*length(EbtoN0))'];
end

% calculate the conditional probability of the words, P(Y1,Y2,Y3|X)
Pcondyyy3=Pcondall(1:end,1:length(t)+1);
B=[];
A=Pcondyyy3;
for j=1:2
    for i=1:2*length(EbtoN0)
        B(i,:)=kron(Pcondyyy3(i,:),A(i,:));
    end
    Pcondyyy3=B;
    B=[];
end
% calculate the conditional probability of the words, P(Y1,Y2,Y3,Y4,Y5|X)
Pcondyyy5=Pcondall(1:end,length(t)+2:2*(length(t)+1));
B=[];
A=Pcondyyy5;
for j=1:4
    for i=1:2*length(EbtoN0)
        B(i,:)=kron(Pcondyyy5(i,:),A(i,:));
    end
    Pcondyyy5=B;
    B=[];
end
% calculate the conditional probability of the words, P(Y1,Y2,Y3,Y4,Y5,Y6,Y7|X)
Pcondyyy7=Pcondall(1:end,2*(length(t)+1)+1:end);
B=[];
A=Pcondyyy7;
for j=1:6
    for i=1:2*length(EbtoN0)
        B(i,:)=kron(Pcondyyy7(i,:),A(i,:));
    end
    Pcondyyy7=B;
```

```
65      B=[];
66  end
67
68  % compute the entropy for k=3
69  HX=0.5*log2(1/0.5)+0.5*log2(1/0.5);
70  % compute P(Y) with Bayes
71  PY3=reshape(Pcondyyy3.*0.5,2,(length(t)+1)^3*length(EbtoN0));
72  PY3=sum(PY3)';
73  PY3=reshape(PY3,length(EbtoN0),(length(t)+1)^3);
74  % compute the entrpy H(Y)
75  HY3=-PY3.*log2(PY3);
76  HY3=sum(HY3')';
77  % compute the of Y, given X.
78  HYX3=sum((Pcondyyy3.*0.5.*log2(1./Pcondyyy3))');
79  HYX3=reshape(HYX3,2,length(EbtoN0));
80  HYX3=sum(HYX3)';
81  % compute the conditional entropy
82  HXY3=-(HY3-HYX3-repmat(HX,length(EbtoN0),1));
83
84
85  % compute the entropy for k=5
86
87  HX=0.5*log2(1/0.5)+0.5*log2(1/0.5);
88  PY5=reshape(Pcondyyy5.*0.5,2,(length(t)+1)^5*length(EbtoN0));
89  PY5=sum(PY5)';
90  PY5=reshape(PY5,length(EbtoN0),(length(t)+1)^5);
91  HY5=-PY5.*log2(PY5);
92  HY5=sum(HY5')';
93
94  HYX5=sum((Pcondyyy5.*0.5.*log2(1./Pcondyyy5))');
95  HYX5=reshape(HYX5,2,length(EbtoN0));
96  HYX5=sum(HYX5)';
97  HXY5=-(HY5-HYX5-repmat(HX,length(EbtoN0),1));
98
99
100 % compute the entropy k=7
101
102 HX=0.5*log2(1/0.5)+0.5*log2(1/0.5);
103 PY7=reshape(Pcondyyy7.*0.5,2,(length(t)+1)^7*length(EbtoN0));
104 PY7=sum(PY7)';
105 PY7=reshape(PY7,length(EbtoN0),(length(t)+1)^7);
106 HY7=-PY7.*log2(PY7);
107 HY7=sum(HY7')';
108
109 HYX7=sum((Pcondyyy7.*0.5.*log2(1./Pcondyyy7))');
110 HYX7=reshape(HYX7,2,length(EbtoN0));
111 HYX7=sum(HYX7)';
112 HXY7=-(HY7-HYX7-repmat(HX,length(EbtoN0),1));
113
114 % calculate the lower bound
115 % find the errorrate by approximate the fano inequality
116 % H(X|Y) <= H(Pe)+Pe*ld(|X|-1)
117 % H(Pe)= -Pe*ld(Pe)-(1-Pe)*ld(1-Pe)
118 for i=1:length(EbtoN0)
119     Pe3lb(i)=0.2;
120     HP=1;
121     while HP>0.00000001 %&& Pe3lb(i)>=0.0001
122           HP=-Pe3lb(i)*log2(Pe3lb(i))-(1-Pe3lb(i))*log2(1-Pe3lb(i))-HXY3(i);
123           if HP>0.001
124             Pe3lb(i)=Pe3lb(i)-0.00001;
125           elseif HP<0.001 && HP>0.00001
126             Pe3lb(i)=Pe3lb(i)-0.000001;
127           else
128             Pe3lb(i)=Pe3lb(i)-0.0000001;
129           end
130     end
131     if HP<0.00001
132         Pe3lb(i)=Pe3lb(i)+0.0000001;
133     else
134         Pe3lb(i)=Pe3lb(i)+0.000001;
135     end;
136
137     Pe5lb(i)=0.2;
138     HP=1;
139     while HP>0.00000001 %&& Pe5lb(i)>=0.0001
140           HP=-Pe5lb(i)*log2(Pe5lb(i))-(1-Pe5lb(i))*log2(1-Pe5lb(i))-HXY5(i);
141           if HP>0.001
142             Pe5lb(i)=Pe5lb(i)-0.00001;
143           elseif HP<0.001 && HP>0.00001
144             Pe5lb(i)=Pe5lb(i)-0.000001;
145           else
146             Pe5lb(i)=Pe5lb(i)-0.0000001;
147           end
148           Pe5lb(i);
149     end
150     if HP<0.00001
151         Pe5lb(i)=Pe5lb(i)+0.0000001;
152     else
153         Pe5lb(i)=Pe5lb(i)+0.000001;
154     end
```

```
155
156     Pe7lb(i)=0.2;
157     HP=1;
158     while HP>0.00000001 %&& Pe7lb(i)>=0.0001
159           HP=-Pe7lb(i)*log2(Pe7lb(i))-(1-Pe7lb(i))*log2(1-Pe7lb(i))-HXY7(i);
160          if HP>0.001
161             Pe7lb(i)=Pe7lb(i)-0.00001;
162           elseif HP<0.001 && HP>0.00001
163             Pe7lb(i)=Pe7lb(i)-0.000001;
164           else
165             Pe7lb(i)=Pe7lb(i)-0.0000001;
166           end
167     end
168     if HP<0.00001
169         Pe7lb(i)=Pe7lb(i)+0.0000001;
170     else
171         Pe7lb(i)=Pe7lb(i)+0.000001;
172     end
173 end
174 % calculate the upper bound
175 % H(X|Y) >= 2*ld(2)*Pe
176 for i=1:length(EbtoN0)
177     Pe3ub(i)=HXY3(i)./(2*log2(2));
178     Pe5ub(i)=HXY5(i)./(2*log2(2));
179     Pe7ub(i)=HXY7(i)./(2*log2(2));
180
181 end
```

Abbildungsverzeichnis

Literatur

[1] Shannon, Claude E., A Mathematical Theory of Communication, Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, 1948

[2] Göbel, Jürgen, Informationstheorie und Codierungsverfahren, VDE Verlag Gmbh, Berlin und Offenbach, 2007

[3] Glavieux, Alain, Channel Coding in Communication Networks: From Theory to Turbocodes, Iste Publishing Company, ISBN: 978-1-905209-24-8, 2007

[4] Werner, Martin, Information und Codierung : Grundlagen und Anwendungen, 2., vollst. überarb. und erw. Aufl., Wiesbaden : Vieweg + Teubner, ISBN: 978-3-8348-0232-3, 2008

[5] Görne, Thomas, Tontechnik. 1. Aufl., Carl Hanser Verlag, Leipzig, 2006, ISBN 3-446-40198-9

[6] Henle, Hubert Das Tonstudio Handbuch. 5.Auflage, GC Carstensen Verlag, München, 2001, ISBN 3-910098-19-3

[7] Papula, Lothar, Mathematik für Ingenieure und Naturwissenschaftler Band 3, Vieweg+Teubner Verlag, Springer Fachmedien Wiesbaden GmbH, 2011

[8] Veeravalli, V.V., 2007, http://www.ifp.illinois.edu/ vvv/e-ce559/handouts/notes5.pdf am 20.04.2012

[9] Cover, Thomas M., Thomas, Joy A., Elements of Imformation Theory, Second edition, John Wiley and sons, Inc., Hoboken, New Jersey, 2006

[10] Chu, J.T., Chueh, J.C., Inequalities between information measures and error probability, Journal of the Franklin Institude, 1966

[11] Sankar, Krishna, Bit Error Rate (BER) for BPSK modulation, 2007