NACHRICHTENTECHNISCHES PRAKTIKUM

Codierung / Fehlererkennung

Technische Universität Braunschweig

Inhaltsverzeichnis

1 Einleitung

In diesem Praktikumsversuch soll eine vollständige Übertragungsstrecke mit Hilfe einer dafür entwickelten Software simuliert werden. Für eine bestimmte Quelle soll ein Fehlerschutz ermittelt werden, damit die Übertragung über einen Kanal mit einer bestimmten Bitfehlerrate möglichst fehlerfrei ist. Hierfür müssen die Parameter geeigneter Kanalcodes bestimmt werden und diese durch die Praktikumssoftware getestet und korrigiert werden. Das Praktikumsskript beschreibt die Komponenten des Übertragungssystems, wobei insbesondere auf die Kanalcodierung eingegangen wird. Die Kanalcodierung ist der Schwerpunkt in diesem Praktikumsversuch und soll ausführlich vermittelt werden. Im Anschluss an diese Beschreibung befindet sich eine kurze Einführung in die Praktikumssoftware.

2 Nachrichtenübertragung

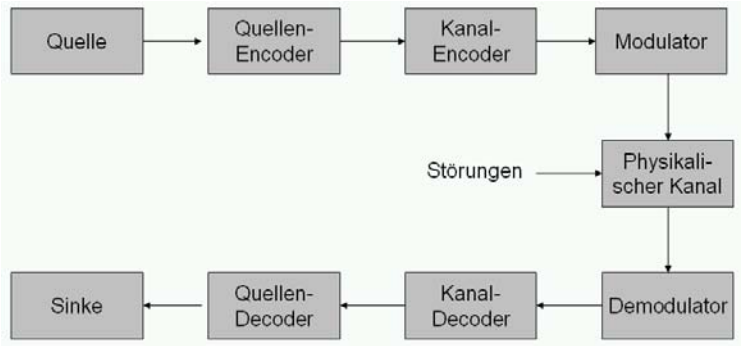


Abbildung 1 Übertragungsstrecke

Das Grundprinzip der digitalen Nachrichtenübertragung ist in Abbildung 1 dargestellt. Der erste Block des Systems ist die digitale oder analoge Informationsquelle. Die digitale Quelle erzeugt eine Folge diskreter Zeichen, d.h. ein wert- und zeitdiskretes Signal. Dieses muss, sofern es sich nicht um ein binäres Signal handelt, noch in ein solches codiert werden. Da hier ein digitales Kommunikationssystem betrachtet wird, muss im Falle der analogen Quelle eine Pulscodemodulation stattfinden. Das wert- und zeitkontinuierliche Signal, das zum Beispiel ein Sprach- oder Videosignal sein kann, wird damit in eine Binärfolge umgewandelt und entspricht nun einer digitalen Quelle. Das analoge Quellensignal wird hierfür unter Einhaltung des Abtasttheorems abgetastet und anschließend quantisiert. In einem dritten Schritt werden die quantisierten Abtastwerte in eine endliche Binärzahl umcodiert. Die Verbindung aus Quantisierer und Codierer wird auch als Analog-Digitalumsetzer bezeichnet (vgl. [1], S.198).

Dem Quellenblock schließt sich der Quellenencoder an, der zur Reduzierung der Redundanz dient.

Da bei der Übertragung von Daten auf Leitungen oder Speichermedien immer mit Störungen in Form von stochastisch verteilten Einzelfehlern oder gebündelten Fehlern zu rechnen ist, müssen die Daten geschützt werden. Die Kanalcodierung stellt Verfahren zur Verfügung, mit denen Nachrichten gegen auftretende Übertragungsfehler geschützt werden können. Es wird sendeseitig gezielt Redundanz hinzugefügt, so dass bei der Nachrichtenübertragung entstandene Fehler auf der Empfangsseite erkannt und korrigiert werden können. Es muss zwischen zwei grundsätzlichen Methoden unterschieden werden. Bei dem FEC-Verfahren (Forward Error Correction) dient die hinzugefügte Redundanz empfangsseitig zur Korrektur der Übertragungsfehler. Hier finden Blockcodes sowie Faltungscodes ihre Verwendung. Der Kanalencodierungsblock kann auch mehrere verkettete Codes oder einen Interleaver enthalten. Das ARQ-Verfahren (Automatic Repeat Request) beschränkt sich auf die Erkennung von Übertragungsfehlern und fordert über einen Rückkanal bei erkanntem Fehler eine Wiederholung der Daten an. Hierfür wird für die reine Fehlererkennung viel weniger Redundanz benötigt als für die Fehlerkorrektur, jedoch wird ein Rückkanal benötigt und es kann durch die erneute Übertragung zu erheblichen Verzögerungen kommen (vgl. [2], S.4).

Der Modulator dient als Schnittstelle zwischen Kanalencoder und physikalischem Kanal. Dieser Kanal kann keine diskreten Symbole übertragen, sondern nur zeitkontinuierliche Signale, so dass der Modulator den diskreten Symbolen solche Signale zuordnet, die über den physikalischen Kanal übertragbar sind. Dies schließt die Anpassung des Signals an den Übertragungsbereich des Kanals mit ein, so dass zum Beispiel das Basisbandsignal in die Bandpasslage verschoben werden muss.

Der physikalische Kanal verändert in der Regel die Signale bei der Übertragung, da er prinzipiell nicht ideal ist und keinen idealen Phasen- bzw. Frequenzgang besitzt. Zusätzlich machen sich Rauschen, Fading oder Übersprecheffekte als Störungen bemerkbar. Der Kanal kann drahtgebunden, ein Funkkanal oder auch ein Speichermedium, zum Beispiel eine CD oder ein Magnetband sein. Ein Speichermedium kann auch als Kanal angesehen werden, da hier ebenfalls Fehler in Form von Kratzern auftreten können.

Der Demodulator rekonstruiert aus den verfälschten zeitkontinuierlichen Sendesignalen die diskreten Infosymbole möglichst genau und der Kanaldecoder versucht nun unter der Zuhilfenahme der zugefügten Redundanz die eventuell vorhandenen Fehler zu korrigieren.

3 Quellencodierung

Der Quellencoder komprimiert die Nachricht ohne Informationsverlust auf eine minimale Anzahl von Symbolen, was auch Reduktion von Redundanz genannt wird. Dadurch kann mehr Information in der gleichen Zeit übertragen werden. Besonders bei der Bild- und Tonquellencodierung wird häufig eine Irrelevanzreduktion durchgeführt. Hierbei handelt es sich um weitergehende Kompression, wobei toleriert wird, dass Informationen verloren gehen, die für den Nutzer nicht von Bedeutung sind. Die Irrelevanzreduktion führt zu unwiederbringlichen Verlusten. In dem Versuch beschränken wir uns jedoch nur auf die Redundanzreduktion und diese wird hier beispielhaft anhand einer Huffman-Codierung demonstriert.

3.1 Grundlagen der Informationstheorie

Der Aufbau einer Nachricht aus Sicht der Informationstheorie ist in Abbildung 2 dargestellt. Es wird deutlich, dass die interessierende Information nur der Teil der Nachricht ist, der relevante und nicht redundante Elemente enthält.

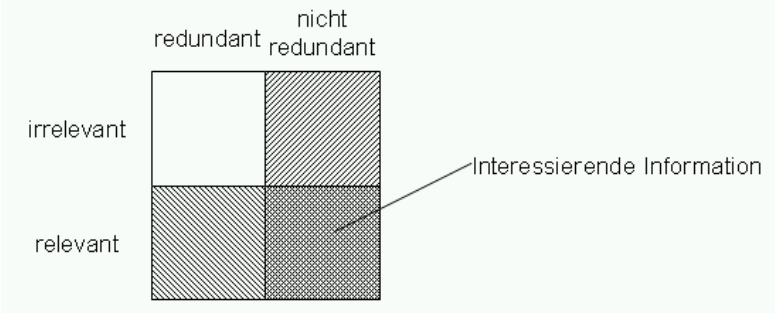


Abbildung 2 Aufbau einer Nachricht

Häufig vorkommende Ereignisse haben einen geringen und selten auftretende Ereignisse einen größeren Informationsgehalt $I(x_i)$, welcher als

$$I(x_i) = ld(\frac{1}{p(x_i)})$$

definiert ist. Für die Redundanzberechnung einer Informationsquelle mit N Symbolen wird die Gesamtentropie, also der Erwartungswert des Informationsgehalts

$$H(x) = \sum_{i=1}^{N} I(x_i) \cdot p(x_i)$$

und der Entscheidungsgehalt

$$H_0 = ld(N)$$

benötigt. Die Redundanz der Quelle ist nun die Differenz zwischen dem Entscheidungsgehalt und der Gesamtentropie

$$R_Q = H_o - H(x).$$

Die aus der Quellencodierung resultierende Redundanz des Codes ergibt sich durch

$$R_C = L - H(x), \text{ wobei } L = \sum_{i=1}^{N} p(x_i) \cdot L(x_i)$$

als die mittlere Codewortlänge bezeichnet wird (vgl. [3], S.4-21).

3.2 Huffman-Codierung

Bei der Huffman-Codierung werden häufig vorkommenden Symbolen kurze Codeworte und selten vorkommenden Symbolen längere Codeworte zugeordnet. Der Algorithmus dazu ist sehr einfach und ist hier beschrieben (vgl. [4], S. 33-38):

1. Symbole nach absteigender Wahrscheinlichkeit sortieren
2. Den 2 Symbolen mit den mit den kleinsten $p(x_i)$, $p(x_j)$ die Bits 0 und 1 zuordnen.
3. Diese 2 Symbole zusammenfassen und $p(x_i)+ p(x_j)$ addieren
4. Schritte 1-3 wiederholen bis $p(x_i)+ p(x_j)=1$.

Dieser Algorithmus ist anschaulich anhand eines Beispiels mit Symbolen der Auftrittswahrscheinlichkeiten *p(A)=0.04, p(B)=0.12, p(C)=0.07, p(D)=0.1, p(E)=0.11, p(F)=0.36, p(G)=0.07, p(H)=0.13* in Abbildung 3 demonstriert.

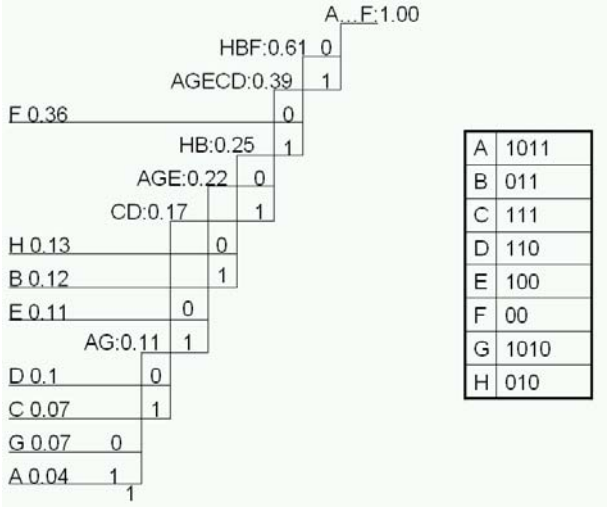


Abbildung 3 Beispiel einer Huffman-Codierung

4 Kanalcodierung

4.1 Einteilung der FEC-Codes

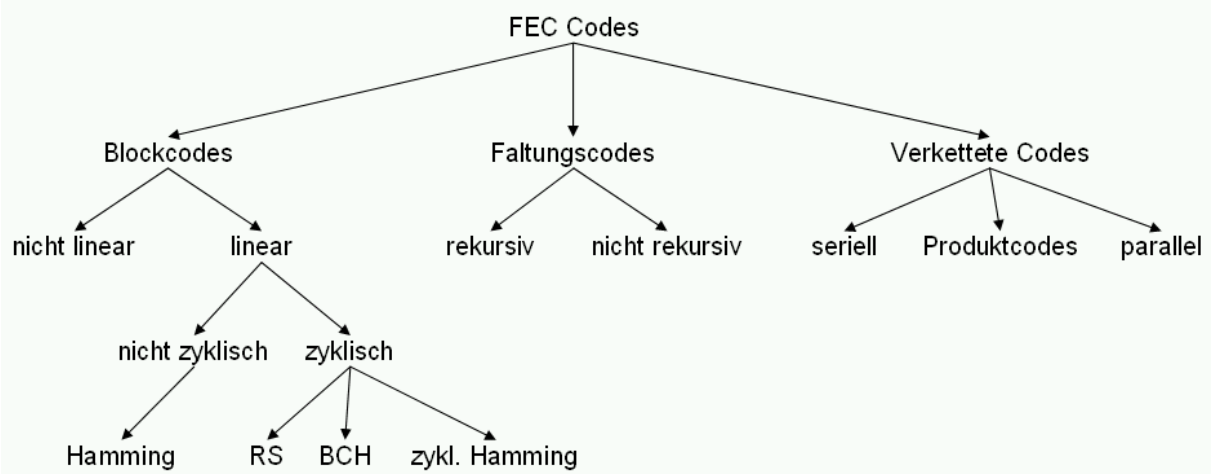


Abbildung 3 Einteilung der FEC-Codes

Dieses Kapitel beschäftigt sich nur mit den in dem Versuch eingesetzten FEC-Codes. Abbildung 3 zeigt eine Übersicht über die Vielzahl der fehlerkorrigierenden Kanalcodes und die Gliederung in drei große Codefamilien: Blockcodes, Faltungscodes und verkettete Codes. Die Blockcodes lassen sich aufteilen in lineare, nichtlineare und zyklische bzw. nichtzyklische Codes. Die Familie der Faltungscodes wird in rekursive und nichtrekursive Codes unterteilt. Die rekursiven Codes sollen hier jedoch nicht behandelt werden. Die dritte Gruppe ist nun eine Verkettung von Codes der beiden ersten Familien. Diese Codearten unterscheiden sich stark in den Eigenschaften und sind für unterschiedliche Fehlerarten prädestiniert.

4.2 Auftretende Fehlerarten

Da in Übertragungskanälen verschiedene Fehlerarten auftreten, erscheint es sinnvoll diese anhand der Abbildung 4 ([5], S. 58) zu charakterisieren.

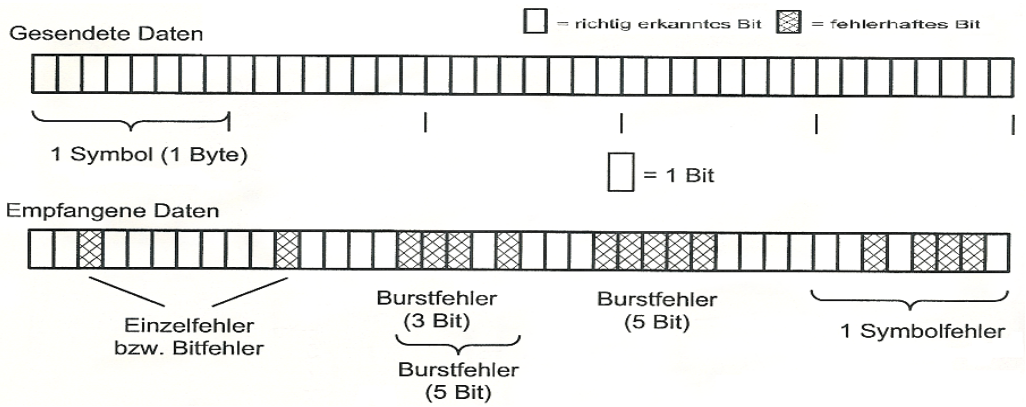


Abbildung 4 Fehlerübersicht

Man unterscheidet zwischen Einzelfehlern (Bitfehlern), Symbolfehlern und Burstfehlern, die auch Büschelfehler genannt werden. Als einen Büschelfehler der Länge L Bit bezeichnet man ein Fehler, bei dem das erste und letzte Bit des L Bit breiten Musters fehlerhaft sind und die dazwischen liegenden Bits nicht zwingend fehlerhaft seien müssen. Bei einem Büschelfehler sind die Fehlerstellen voneinander statistisch abhängig, während sie bei den Einzelfehlern stochastisch verteilt sind. Unter einem Symbolfehler bezeichnet man ein fehlerhaftes Symbol mit einer beliebigen Zahl an Bitfehlern.

4.3 Lineare Blockcodes

Diese Codes werden allgemein als (n,k,d_{min})-Blockcodes bezeichnet, wobei n die Zahl der Codestellen, k die Anzahl der Informationsstellen und d_{min} die minimale Hammingdistanz zweier Codeworte ist. Die Coderate der linearen Blockcodes beträgt

$$R = \frac{k}{n}.$$

Die Zahl der Codewörter beträgt demnach 2^k und die Anzahl der möglichen Empfangswörter ist 2^n. Die Hammingdistanz zweier Codewörter ist die Anzahl der Abweichungen der Stellen von den beiden Codewörtern. Die minimale Hammingdistanz zweier Codewörter d_{min} entspricht dem minimalem Gewicht der Codewörter bei den linearen Blockcodes. Das Gewicht eines Codewortes ist definiert durch die Anzahl der von Null verschiedenen Stellen. Die Hammingdistanz gibt Auskunft über die Fähigkeit der Korrektur von stochastisch verteilten Einzelfehlern. Der Code mit der minimalen Hammingdistanz d_{min} kann

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

Fehler korrigieren und

$$e = d_{\min} - 1$$

Fehler erkennen.

Es wird zwischen systematischen Codes, bei den die $m=n-k$ Korrekturstellen von den Informationsstellen getrennt sind, und den nicht-systematischen Codes unterschieden, wo diese Stellen nicht trennbar sind. Anhand des Hamming-Codes werden nun noch einige weitere Eigenschaften der linearen Blockcodes beschrieben (vgl. [4], S. 80-109).

4.3.1 Hamming-Codes

Der Hamming-Code besitzt die minimale Hammingdistanz $d_{min}=3$ und kann daher einen einzelnen Fehler in einem Codewort korrigieren und zwei Fehler erkennen. Die Codierung geschieht bei den linearen Blockcodes durch Matrixmultiplikation des Informationswortes *u*, welches als Binärvektor verstanden wird, mit der Generatormatrix *G*

$$c = u \cdot G .$$

Die Generatormatrix ist eine *(k,n)*-Matrix und ist bei den systematischen Codes darstellbar als

$$G = \left[I_k \; P \right]$$

mit der $k \times k$-Einheitsmatrix I_k und einer $k \times (n-k)$-Prüfstellenmatrix P, die für die eigentliche Berechnung der Prüfstellen zuständig ist. Bei einem systematischen (7,4)-Hamming-Code sieht die Prüfmatrix wie folgt aus:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Diese Generatormatrix ist bei den Hamming Codes eine $(2^m - 1, 2^m - 1 - m)$-Matrix, da hier

$$n = 2^m - 1$$

gilt. Man benötigt für die Definition eines Hamming-Codes daher nur die Anzahl der Prüfstellen *m*. Die Decodierung von Blockcodes geschieht nach der Übertragung mit Hilfe des Syndroms, welches sich durch Multiplikation des Empfangswortes *y* mit der transponierten Prüfmatrix *H* berechnen lässt

$$s = y \cdot H^T .$$

Die Prüfmatrix *H* ist orthogonal zu der Generatormatrix und berechnet sich bei systematischen Generatormatrizen über

$$H = \left[P^T I_{n-k} \right].$$

Wenn das Syndrom der Nullvektor ist, dann ist *y* ein gültiges Codewort. Entspricht das Syndrom der i-ten Spalte der der Prüfmatrix *H*, dann ist ein Einzelfehler an der Stelle *i* aufgetreten und diese Stelle kann korrigiert werden. Ist das Syndrom nicht in der Prüfmatrix vorhanden, dann kann der Fehler nur erkannt, nicht aber korrigiert werden. Treten mehrere Fehler auf, gibt es natürlich auch immer die Möglichkeit,

dass ein Codewort wieder in ein anderes Codewort gewandelt wird. In diesem Fall ist der Fehler natürlich nicht erkennbar (vgl. [4], S.89-109).

4.4 Zyklische Codes

4.4.1 Encodierung

Zyklische Codes bilden eine Teilmenge der linearen Blockcodes und können systematisch oder nicht systematisch sein. Ein linearer *(n,k)*-Blockcode mit *n* Codewortstellen und *k* Nachrichtenstellen heißt zyklisch, wenn jede zyklische Verschiebung eines Codewortes wieder auf ein Codewort führt. Da bereits eine Zeile der Generatormatrix eines zyklischen Codes

$$G = \begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ & g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ \vdots & & & & & & & \vdots \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & g_{n-k} \end{pmatrix}$$

die Generatormatrix vollständig charakterisiert, kann diese auch als Polynom

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_{n-k} x^{n-k}$$

beschrieben werden. Das zu codierende Nachrichtenwort hat durch diese vereinfachende Beschreibung die Gestalt

$$u(x) = u_0 + u_1 x + u_2 x^2 + \ldots + u_{k-1} x^{k-1}$$

und die Codewörter lassen sich durch

$$c(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1}$$

beschreiben.

Zu beachten ist, dass diese Rechenoperationen immer im *GF(2)* ausgeführt werden und damit bei allen Operationen *modulo-2* gerechnet wird. Für alle zugelassenen Codeworte gilt, dass das Generatorpolynom *g(x)* die Codewortpolynome *c(x)* ohne Rest teilt:

$$c(x) \bmod g(x) = 0 .$$

Daraus ergibt sich für die unsystematischen Codes die Codierungsvorschrift

$$c(x) = u(x) \cdot g(x) ,$$

und für die systematische Codierung folgende Vorschrift:

$$c(x) = x^{n-k} \cdot u(x) + r(x) \quad mit \quad r(x) = x^{n-k} \cdot u(x) \bmod g(x) .$$

Die Multiplikation $x^{n-k} \cdot u(x)$ bedeutet anschaulich für die systematischen Codes das Einfügen von *m=n-k* Nullstellen als Kontrollstellenplatzhalter. Anschließend wird das

so erhaltene Codewortpolynom durch das Generatorpolynom dividiert und der Rest *r(x)* in die zuvor zu Null gesetzten Kontrollstellen geschrieben (vgl. [4], S.167-183)

4.4.2 Bestimmung eines Generatorpolynoms

Die Bestimmung des Generatorpolynoms geschieht über die Periode eines Polynoms. Die Periode r eines Generatorpolynoms ist der kleinste Exponent mit dem

$$(x^r + 1) \bmod g(x) = 0$$

erfüllt ist. Es entsteht ein zyklischer Code, wenn die Codewortlänge *n* der Periode *r* entspricht:

$n = r \leq 2^m - 1$ mit *m=n-k.*

Für die Koeffizienten des Generatorpolynoms *g(x)* gilt: $g_0 = g_{n-k} = 1$. Polynome mit einer Periode $n=r=2^m-1$ werden auch als primitive Polynome bezeichnet und besitzen die maximale Periode. Ein Generatorpolynom wird erzeugt, indem $1 + x^n$ in Faktoren zerlegt wird. Jeder Faktor kann nun als Generatorpolynom verwendet werden, indem die Periode und damit die Codewortlänge berechnet wird (vgl. [4], S.176-177).
Im folgendem Beispiel wird die Bestimmung eines Generatorpolynoms noch näher erläutert:

Das Polynom $1 + x^7$ *wird in seine Faktoren zerlegt:*

$$x^7 + 1 = (x+1) \cdot (x^3 + x + 1) \cdot (x^3 + x^2 + 1)$$

Das Generatorpolynom soll vom Grad m=3 sein, daher stehen die letzten beiden Polynome zur Auswahl. Die Periodenberechnung geschieht über die Umkehrung des Gauß'schen Divisionsalgorithmus:

$$(x^3 + x^2 + 1) \cdot p(x) = x^n + 1$$

$$\begin{array}{cccccccc}
 & & & & x^3 & x^2 & 0 & 1 \\
 & & x^5 & x^4 & 0 & x^2 & & \\
 & x^6 & x^5 & 0 & x^3 & & & \\
x^7 & x^6 & 0 & x^4 & & & & \\
\hline
x^7 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array} \qquad \Rightarrow r = n = 7$$

Der betrachtete Faktor kann als Generatorpolynom für einen zyklischen (7,4)-Code genutzt werden. Es handelt sich sogar um ein primitives Polynom.

4.4.3 Decodierung

Im Nachrichtenkanal überlagert sich dem Codewort *c(x)* das Fehlermuster *f(x)* und am Decoder kommt das Empfangswort *y(x)=c(x)+f(x)* an. Dieses wird im Decoder durch das Generatorpolynom dividiert, um das Syndrom *s(x)* zu berechnen, welches gleich dem Divisionsrest ist. Ist das Syndrom gleich Null, handelt es sich bei dem Empfangswort um ein gültiges Codewort, andernfalls ist bei der Übertragung ein Fehler aufgetreten, und der Fehler kann mit Hilfe der Syndromtabelle korrigiert werden.

4.4.4 Zyklische Hamming-Codes

Zyklische Hammingcodes haben als Generatorpolynom ein primitives Polynom *g(x)* vom Grad m. Die Codewortlänge beträgt demnach $n=2^m-1$. Diese können einen einzelnen Bitfehler korrigieren.

4.4.5 Fire-Codes

Der Fire-Code hat ein Generatorpolynom der Form

$$g(x) = g_1(x) \cdot (x^{m_2} + 1),$$

wobei $g_1(x)$ primitiv und vom Grad m_1 ist und $g(x)$ insgesamt vom Grad $m= m_1+m_2$. Die Codewortlänge ist $n = (2^{m_1} - 1) \cdot m_2$ wobei $2^{m_1} - 1$ und m_2 teilerfremd sein müssen. Dieser Code korrigiert $b \le m_1$ bzw. $b \le m_2 / 2 + 1$ Bit breite Büschelfehler. Ein Spezialfall dieser Codes ist durch

$$g(x) = g_1(x) \cdot (x + 1)$$

definiert und wird als Abrahamson-Code bezeichnet (vgl. [6], S.250).

4.5 Reed-Solomon-Codes

Reed-Solomon-Codes gehören ebenfalls zur Klasse der zyklischen Codes, werden hier aber in einem weiteren Unterkapitel betrachtet, da diese sich von den Fire Codes und den zyklischen Hamming-Codes stark unterscheiden. Reed-Solomon Codes sind sehr leistungsfähige symbolorientierte Blockcodes, die sich hervorragend zur Korrektur von Büschelfehlern sowie Symbolfehlern eignen. Diese Codes haben eine große praktische Bedeutung und werden zum Beispiel im Mobilfunk und im digitalen Broadcastbereich eingesetzt. Da ein Symbol aus einer bestimmten Anzahl an Bits besteht, muss bei der Decodierung erkannt werden, welches Symbol fehlerhaft ist und anschließend muss der Wert des unverfälschten Symbols

berechnet werden. Es muss also nicht mehr nur die Fehlerstelle, sondern es muss auch der Fehlerwert berechnet werden. Dies führt zu einem erheblich größeren Rechenaufwand bei der Encodierung und Decodierung. Die Mathematik des Galois-Feldes, welches zur Darstellung der Symbole genutzt wird, ist für das Verständnis des Aufbaus der RS-Codes unabdingbar. Daher soll im folgenden darauf eingegangen werden.

4.5.1 Galois-Felder

Die Zahlenmenge *{0,1,…,p-1}* mit der Primzahl *p* bildet eine kommutative Gruppe[1] der Ordnung *p* bezüglich der *modulo-p* Addition $\oplus$ mit dem neutralen Element 0. Mit der *modulo-p* Multiplikation $\otimes$ mit dem neutralen Element 1 bildet diese Zahlenmenge den Körper *GF(p)*, welcher auch als Galois-Feld bezeichnet wird. Bis auf das Nullelement der Multiplikation existiert zu jedem Element *a* des *GF(p)* genau ein inverses Element $a' \in GF(p)$, welches für die die *modulo-p* Addition und die *modulo-p* Multiplikation wie folgt gebildet werden kann:

$$a \oplus a' = a \oplus (-a) = 0 \text{ und } \quad a \otimes a' = a \otimes (a^{-1}) = 1$$

Durch die inversen Elemente können nun auch die Subtraktion und die Division durchgeführt werden. Jedes Galois-Feld besitzt mindestens ein primitives Element *z*. Alle Elemente *a* des *GF(p)* außer dem Nullelement sind durch

$$a=z^i \text{ modulo } p \text{ mit } i=0,1,\ldots p-1$$

darstellbar als

$$GF(p)=\{z^0, z^1, z^2, \ldots, z^{p-1}\}.$$

In Abbildung 5 ist die *modulo-p* Addition und Multiplikation sowie die inversen Elemente des *GF(3)* dargestellt (vgl. [2], S.70-72)

$\oplus$	0	1	2	$\otimes$	0	1	2	a	0	1	2
0	0	1	2	0	0	0	0	-a	0	2	1
1	1	2	0	1	0	1	2	a	0	1	2
2	2	0	1	2	0	2	1	a^{-1}	X	1	2

Abbildung 5 Beispiel der Rechenregeln für das GF(3)

Das GF(p) mit der Primzahl *p=2* wird für die Darstellung von Bits genutzt. Da nun Reed-Solomon Codes symbolorientiert sind und jedes Symbol aus *m* Bits besteht, werden diese Zahlenkörper auf p^m Elemente erweitert, welches ähnlich der

[1] Die genaue mathematische Definition kann in ([6],S.33) nachgeschlagen werden. Dies ist aber für das weitere Verständnis nicht von Bedeutung.

Erweiterung reeller Zahlen auf komplexe Zahlen ist. Im häufig angewandten Erweiterungskörper $GF(p^m)=GF(2^8)$ können somit $p^m=256$ 8-Bit-Zeichen dargestellt werden. Jedes Element wird nun durch

$$c=c_0+c_1\alpha+\ldots+c_{m-1}\alpha^{m-1} \text{ mit } c_i \in GF(p)$$

beschrieben. Das Element $\alpha \in GF(p^m)$ welches durch

$$p(\alpha)=0$$

definiert ist, wird als Nullstelle oder primitives Element des primitiven Polynoms

$$p(x)=p_0+p_1x+\ldots+p_mx^m \text{ mit } p_i \in GF(p) \text{ und grad } p(x)=m,$$

bezeichnet, das den Erweiterungskörper aufspannt. Durch

$$\alpha^i \text{ modulo } p(\alpha), \text{ für } i=0,1, \ldots, p^m-2$$

ergeben sich bis auf das Nullelement alle möglichen Elemente c des $GF(p^m)$ (vgl. [6], S.53-59).

Es sei nun noch auf folgende Formel hingewiesen, die aufgrund der zyklischen Eigenschaft des Galois-Feldes zustande kommt und für die Umwandlung der Elemente in der Exponenten-Darstellung nützlich ist:

$$\alpha^k = \alpha^{k \bmod (p^m-1)}.$$

Die Addition zweier Elemente ist die *modulo-p* Addition der einander entsprechenden Polynomkoeffizienten. Die Multiplikation der Elemente ist eine Polynommultiplikation mit anschließender *modulo-p(α)*-Operation. Für jedes Galois-Feld $GF(p^m)$ existiert mindestes ein primitives Polynom und eine Auswahl an Polynomen ist in Abbildung 6 dargestellt.

m	primitives Polynom p(x)	m	primitives Polynom p(x)
1	$x+1$	6	x^6+x+1
2	x^2+x+1	7	x^7+x+1
3	x^3+x+1	8	$x^8+x^6+x^5+x^4+1$
4	x^4+x+1	9	x^9+x^4+1
5	x^5+x^2+1	10	$x^{10}+x^3+1$

Abbildung 6 Primitive Polynome

Ein Beispiel der Rechenregeln für das $GF(2^2)=\{0,1,\alpha,\alpha^2\}$ mit dem primitiven Polynom $p(x)=x^2+x+1$ sind tabellarisch in der Abbildung 7 dargestellt.

$\oplus$	0	1	α	α^2	$\otimes$	0	1	α	α^2
0	0	1	α	α^2	0	0	0	0	0
1	1	0	α^2	α	1	0	1	α	α^2
α	α	α^2	0	1	α	0	α	α^2	1
α^2	α^2	α	1	0	α^2	0	α^2	1	α

Abbildung 7 Rechenregeln für das GF(2^2)

Das Element α^2 entspricht *1+*α, da wie oben erwähnt *modulo-p(*α*)* gerechnet wird.

4.5.2 Encodierung

Für die Codierung des RS-Codes wird noch die diskrete Fourier-Transformation DFT und Rücktransformation IDFT im Galois-Feld sowie eine Transformationsvorschrift für Nullstellen im Galois-Feld benötigt. Die Begriffe Frequenz- und Zeitbereich werden nur für die Unterscheidung der Hin- bzw. Rücktransformation genutzt und haben mit physikalischen Gesichtspunkten nichts zu tun. Ist ein Codewortpolynom

$$c = (c_0, c_1, ..., c_{n-1}) = \sum_{i=0}^{n-1} c_i x^i$$

vom Grad kleiner gleich *n-1=*p^m*-2* mit dem primitiven Element α und den Koeffizienten $c_i \in GF(p^m)$ gegeben, dann kann dieses über die DFT

$$C_j = -c(\alpha^{-j}) = -\sum_{i=0}^{n-1} c_i \alpha^{-ij} \quad \textit{mit j=0,1,…,n-1}$$

in den Frequenzbereich überführt werden. Die Rücktransformation IDFT

$$c_i = C(\alpha^i) = \sum_{j=0}^{n-1} C_j \alpha^{ij} \quad \textit{mit i=0,1,…,n-1}$$

überführt dieses Polynom wieder in den Zeitbereich. Hat das Polynom *c* im Zeitbereich die Nullstelle α^{-j}, verschwindet der entsprechende Koeffizient C_j des Polynoms

$$C=(C_0,C_1,\ldots, C_{n-1}) = \sum_{j=0}^{n-1} C_j x^j$$

im Frequenzbereich (vgl. [7], S.108).

Bei den RS-Codes handelt es sich um einen Maximum-Distance-Separable-Code der Blocklänge *n=*2^m*-1*, denn er erfüllt die Singleton-Schranke

$$d_{\min} \le n - k + 1$$

mit Gleichheit. Über den Zusammenhang von $d_{\min}$ und der Anzahl der korrigierbaren Fehler *t* ergibt sich

$$n - k = 2 \cdot t \,,$$

womit die Anzahl der korrigierbaren Symbolfehler sofort einstellbar ist. Büschelfehler sind somit korrigierbar bis zu einer Länge von

$$t_{Büschel} = m \cdot (t-1) + 1 \,.$$

Die Codierung geschieht, wie in Abbildung 8 dargestellt, im Frequenzbereich, indem zunächst die *n-k* Stellen $C_o \ldots C_{2t-1}$ des Blockes der Länge $n=2^m-1$ zu Null gesetzt und in die *k* Stellen $C_{2t} \ldots C_{n-1}$ die Informationssymbole geschrieben werden (vgl. [7], S.110).

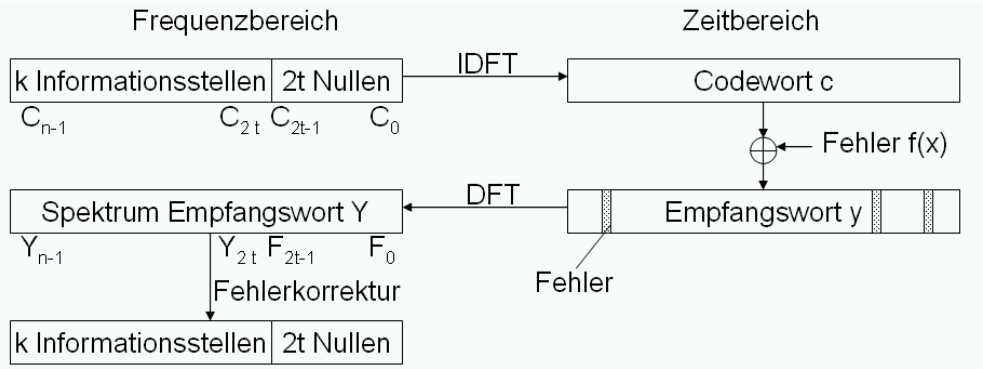


Abbildung 8 Übertragung mit RS-Code

An dieser Stelle soll noch darauf hingewiesen werden, dass auch kürzere Codewortlängen als $n=2^m-1$ gewählt werden können. Es handelt sich dann um einen verkürzten RS-Code. Anschließend wird eine IDFT durchgeführt und man erhält das gültige zu übertragende Codewort *c(x).*

4.5.3 Decodierung

Bei der Übertragung überlagert sich auf das Codewort ein Fehlerpolynom *f(x)* und es wird im Empfänger nun eine DFT des Empfangspolynoms *y(x)* durchgeführt. Durch dieses im Frequenzbereich dargestellte Empfangswort *Y(X)* ist schon ersichtlich, ob ein Fehler aufgetreten ist, denn dies ist der Fall, wenn eine oder mehr Stellen $F_o \ldots F_{2t-1}$ ungleich Null sind. Man beachte, dass bereits diese Stellen ein Teilspektrum des Fehlers sind, da diese Stellen vorher zu Null gesetzt wurden. Diese Eigenschaft macht man sich bei der anschließenden Fehlerkorrektur zu nutze. Mit Hilfe eines rückgekoppelten Schieberegisters kann nun das Fehlerspektrum Symbol für Symbol vollständig bestimmt werden und es findet die Korrektur des Empfangswortes statt (vgl. [7], S. 111-115).

4.6 BCH-Codes

Die Bose-Chaudhuri-Hocquenghem-Codes, welche abkürzend BCH-Codes genannt werden, sind ein binärer Spezialfall der RS-Codes. Sie entstehen aus RS-Codes dadurch, dass nun die Codesymbole nicht mehr aus mehreren Bits bestehen, sondern nun Elemente des *GF(2),* also Bits im Zeitbereich, sind. Die Blocklänge beträgt weiterhin $n=2^m-1$ und die DFT wird weiterhin auf dem $GF(2^m)$ ausgeführt. Die Definition durch die zu Null gesetzten Prüfstellen im Frequenzbereich zur Codierung ist auch bei den BCH-Codes der Fall. Dieser Code ist besonders geeignet zur Korrektur mehrerer statistisch unabhängiger Einzelfehler. Bei einem Kanal mit längeren Bündelfehlern sind die RS-Codes vorzuziehen. Im Einzelfall kann jedoch ein BCH-Codes auch bei Büschelfehlern mittlerer Länge gute Ergebnisse erzielen. Die Parameter für diese Codes inklusive der Anzahl der Korrekturmöglichkeit sind in der Praktikumssoftware als Tabelle hinterlegt.

4.7 Faltungscodes

4.7.1 Encodierung

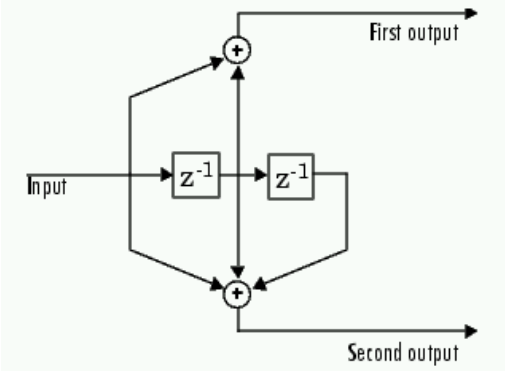


Abbildung 9 Beispiel eines Faltungsencoders

Faltungscodes sind Binärcodes, d.h. die Eingangsinformation muss immer als Bitfolge vorliegen, und sind prädestiniert zur Korrektur einzelner Bitfehler. Diese Codes teilen den Eingangsdatenstrom nicht in fest definierte Blöcke ein, sondern „verschmieren“ die Eingangsinformationen über mehrere Ausgangsdaten. Die Encodierung geschieht durch die Speicherung der Eingangsbits in einem Schieberegister, bei dem die Ausgangsbits an bestimmten Stellen kombinatorisch abgegriffen werden. Mathematisch entspricht dies einer Faltung der Informationsbits mit einem Satz von Generatorkoeffizienten. Abbildung 9 zeigt den Beispielaufbau eines Faltungsencoders zur Verdeutlichung der Begriffe. In diesem Beispiel ist die Eingangsrahmenbreite $k = 1$ Bit und die Ausgangsrahmenbreite $n=2$ Bit. Es wird also

1 Infobit in das Schieberegister geschoben und 2 Codebits werden erzeugt. Die resultierende Coderate beträgt in dem Beispiel *R = k/n = 1/2*. Als Gedächtnislänge wird die Speichertiefe des Encoders verstanden und ergibt sich aus der Schieberegisterlänge *S* multipliziert mit der Eingangsrahmenbreite *k*, hier also $S \cdot k = 6$. Die Beeinflussungslänge

$$L = (S + 1) \cdot k$$

hingegen gibt die Anzahl der Bits an, die bei der Faltung beteiligt sind. Die Lage der Abgriffe der Ausgangsbits lässt sich durch Generatorpolynome angeben, deren binäre Koeffizienten davon abhängen ob ein Abgriff an der entsprechenden Stelle besteht oder nicht. Für jeden Ausgangszweig eines Schieberegisters gibt es ein solches Generatorpolynom, welches häufig auch als Oktalzahl angegeben wird. Der Faltungscode wird insgesamt durch $n \cdot k$ Generatorpolynome beschrieben. Der Beispielcode wird durch die beiden Generatorpolynome $g_1 = 1+x+0x^2$ (6_{OKT}) und $g_2 = 1+x+x^2$ (7_{OKT}) definiert. Der Koeffizient mit dem höchsten Grad des Generatorpolynoms entspricht dem niederwertigsten Bit (LSB) des Schieberegisters (vgl. [7], S.118).

Die Parameter Beeinflussungslänge und Coderate sind entscheidend für den Decodieraufwand und die Leistungsfähigkeit des Codes. Die Generatorpolynome, die rechnergestützt ermittelt werden, fließen ebenfalls in die Leistungsfähigkeit mit ein. Im Gegensatz zu Blockcodes sind bei den Faltungscodes die Parameter *n* und *k* sehr klein und bewegen sich in der Größenordnung *n=2,3,4* und *k=1,2*. Schon mit Gedächtnislängen unter 8 Bits lassen sich sehr starke Codierungsgewinne erzielen, die mit komplizierten Blockcodes schritt halten können (vgl. [2], S.246-248).

4.7.2 Darstellung als Automat

Faltungsencoder können auch als Automaten gedeutet werden und werden durch $2^{S \cdot k}$ interne Zustände charakterisiert, welche sich aus den möglichen Binärkombinationen des Gedächtnis herleiten lassen. Abbildung 10 zeigt das als Mealy-Automat dargestellte Zustandsdiagramm des Beispiel-Encoders. Der Automat wechselt abhängig vom aktuellen Zustand und den *k* aktuellen Eingangsbits in einen neuen Zustand und liefert *n* Ausgangsbits.

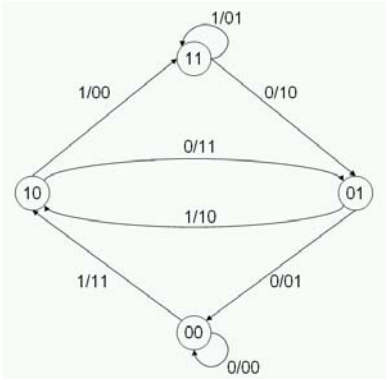


Abbildung 10 Zustandsdiagramm des Beispiel-Encoders

4.7.3 Darstellung als Trellisdiagramm

Eine weitere Darstellung, die ebenso mächtig ist wie die Schieberegisterdarstellung und das Zustandsdiagramm, ist das Trellissegment in Abbildung 11 bzw. das Trellisdiagramm in Abbildung 12.

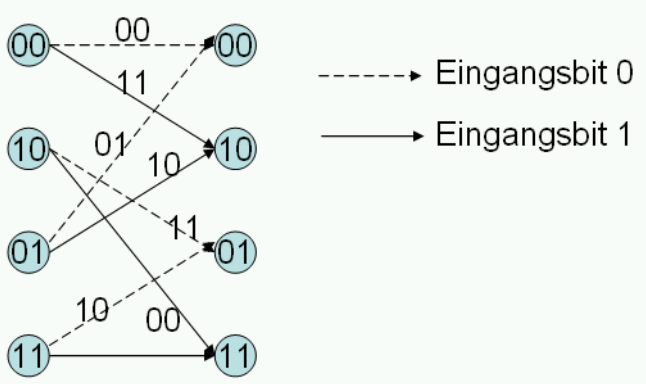


Abbildung 11 Trellissegment des Beispiel-Encoders

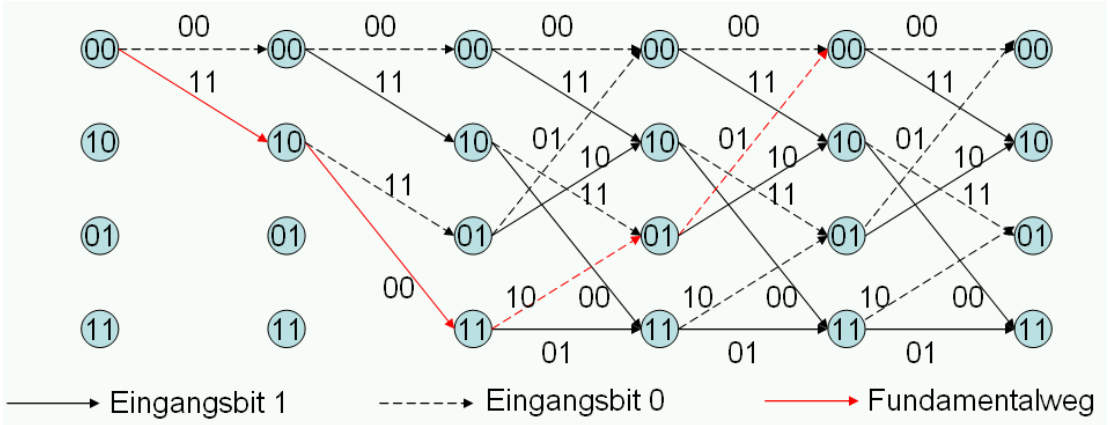


Abbildung 12 Trellisdiagramm für den Beispiel-Encoder

Von jedem *der* $2^{S \cdot k}$ Zustände gehen 2^k Kanten ab und es kommen 2^k Kanten an. Dies bedeutet, dass es insgesamt pro Trellissegment $2^{S \cdot k + k}$ Kanten gibt. Das Trellisdiagramm ist ein gerichteter Graph mit Anfangs- und Endzustand und setzt

sich aus hintereinander gereihten Trellissegmenten zusammen. Die fehlenden Kanten am Anfang lassen sich durch die Vorbelegung des Schieberegisters mit Nullen erklären (vgl. [2], S.259-261).

4.7.4 Punktierung und Terminierung

Faltungscodes können in verschiedene Klassen aufgeteilt werden, von denen zwei im folgendem Abschnitt definiert werden.

Bei den terminierten Faltungscodes werden nach einer bestimmten Anzahl *T* an Infobits $S \cdot k$ Nullen (tail bits) eingefügt, so dass der Endzustand immer der Nullzustand ist. Es entsteht so eine Blockcode, der *T* Infobits auf $(T + S \cdot k) \cdot n$ Codebits abbildet und so die Coderate auf

$$R_{ter\,\min iert} = \frac{T}{T + k \cdot S} \cdot R$$

reduziert. Bei Informationen mit einer festen Rahmenstruktur wird die Terminierung fast immer angewandt, da durch diese Maßnahme die Fehlerfortplanzung verhindert wird.

Die sehr niedrige Coderate von Faltungscodes sorgt zwar für gute Korrekurfähigkeit, jedoch wird durch sie auch viel Redundanz hinzufügt. Daher werden bei den punktierten Faltungscodes *P* Ausgangsrahmen von *n* Bits zu einem Block zusammengefasst und darin l Codebits gestrichen (punktiert), die nun nicht mit übertragen werden. Daraus ergibt sich die erhöhte Coderate

$$R_{punktiert} = \frac{P \cdot k}{n \cdot P - l}.$$

Einen Überblick über die verschiedenen Möglichkeiten einen Faltungscode mit *R=1/2* zu punktieren zeigt Abbildung 13. Obwohl ein nicht punktierter *R=k/n*-Code mit $k \neq 1$ etwas besser als ein punktierter Code mit gleicher Coderate *R=1/n* ist, wird aufgrund der einfacheren Decodierung in der Praxis häufig die Punktierung benutzt (vgl. [2], S.251).

Code Rates r	Puncturing pattern	Transmitted sequence (after parallel-to-serial conversion)
1/2	X: 1 Y: 1	$X_1 Y_1$
2/3	X: 1 0 Y: 1 1	$X_1 Y_1 Y_2$
3/4	X: 1 0 1 Y: 1 1 0	$X_1 Y_1 Y_2 X_3$
5/6	X: 1 0 1 0 1 Y: 1 1 0 1 0	$X_1 Y_1 Y_2 X_3 Y_4 X_5$
7/8	X: 1 0 0 0 1 0 1 Y: 1 1 1 1 0 1 0	$X_1 Y_1 Y_2 Y_3 Y_4 X_5 Y_6 X_7$

Abbildung 13 Punktierung von Codes der Coderate R=1/2

4.7.5 Freie Distanz und katastrophale Codes

Die Güte eines Faltungscodes wird durch das minimale Hamminggewicht aller Codefolgen, welches auch als freie Distanz d_f bezeichnet wird, bestimmt. Zwei Codefolgen unterscheiden sich dadurch an mindestens d_f Bitstellen. Einen optimalen Faltungscode erhält man, wenn d_f bei gleicher Coderate und gleicher Beeinflussungslänge K gegenüber allen anderen Faltungscodes maximal ist. Abbildung 14 ([2], S250) zeigt eine Übersicht über optimale Codes für zwei Coderaten. Weiterhin muss darauf geachtet werden, dass der Encoder nicht-katastrophal ist und dadurch lawinenartige Fehlerfortpflanzung verhindert wird. Unter einem nicht-katastrophalen Encoder versteht man einen Encoder, der jede Informationsfolge unendlichen Gewichts in eine Codefolge unendlichen Gewichts codiert (vgl. [2], S.254).

$g_1(x)$	$g_2(x)$	$g_3(x)$	d_f
$1+x+x^2$	$1+x^2$		5
$1+x+x^3$	$1+x+x^2+x^3$		6
$1+x^3+x^4$	$1+x+x^2+x^4$		7
$1+x^2+x^4+x^5$	$1+x+x^2+x^3+x^5$		8
$1+x^2+x^3+x^5+x^6$	$1+x+x^2+x^3+x^6$		10
$1+x+x^2$	$1+x^2$	$1+x+x^2$	8
$1+x+x^3$	$1+x+x^2+x^3$	$1+x^2+x^3$	10
$1+x^2+x^4$	$1+x+x^3+x^4$	$1+x+x^2+x^3+x^4$	12
$1+x^2+x^4+x^5$	$1+x+x^2+x^3+x^5$	$1+x^3+x^4+x^5$	13
$1+x^2+x^3+x^5+x^6$	$1+x+x^4+x^6$	$1+x+x^2+x^3+x^4+x^6$	15

Abbildung 14 Optimale Codes der Raten1/2 und 1/3 für m=2,3,4,5,6

Durch das Trellisdiagramm und das Zustandsdiagramm kann man diese beiden Eigenschaften sehr gut erkennen. Wenn es Zustände gibt, die bei gleich bleibendem Eingang nicht verlassen werden und der Ausgang identisch ist, handelt es sich um einen katastrophalen Code. In dem Standardbeispiel (siehe Abbildung 9) gibt es zwar zwei Zusände 00 und 11, die bei gleich bleibendem Eingang nicht verlassen werden, jedoch sind die Ausgangsbits 00 und 01 unterschiedlich. Es handelt sich um einen nicht-katastrophalen Encoder.

Handelt es sich bei dem betrachteten Encoder um einen nicht-katastrophalen Code, kann die freie Distanz aus den Fundamentalwegen bestimmt werden. Unter dem Fundamentalweg versteht man einen endlichen Weg durch das Trellisdiagramm, der beim Nullzustand beginnt und wieder dort endet, jedoch dazwischen den Nullzustand nicht berührt. In Abbildung 12 ist der Fundamentalweg mit minimalem Hamminggewicht, also mit der minimalen Anzahl an Einsen im Ausgang, eingezeichnet, woraus sich eine freie Distanz $d_f=4$ ergibt.

4.7.6 Decodierung

Die Decodierung der Faltungscodes geschieht in der Regel per Maximum-Likelihood-Decodierung mit Hilfe des Viterbi-Algorithmus. Die ML-Decodierung bedeutet für den angenommenen binären symmetrischen Kanal BSC, dass die Wortfehlerwahrscheinlichkeit P_w minimal wird, wenn zu der Empfangsfolge y als Schätzung die Codefolge $\hat{a}$ gewählt wird, welche von der Empfangsfolge den minimalen Hammingabstand hat:

$$d_H(y,\hat{a}) \leq d_H(y,b) \quad \textit{für alle Codeworte } b.$$

Dies ist äquivalent zu der Maximierung der Viterbi-Metrik Δ, welche zum Beispiel gleich der Summe der richtig empfangenen Bits ist, wenn man einem ausgewählten Pfad durch das Trellisdiagramm folgt. Je größer also die Metrik, desto höher ist die Wahrscheinlichkeit das der ausgewählte Weg durch das Trellis zu der richtigen gesendeten Codefolge gehört. Die Viterbi-Decodierung wird anhand der Trellisdiagramme durchgeführt. Während bei nicht-terminierten Codes der Anfangszustand als unbekannt angenommen wird und eventuell bestimmte Algorithmen zur Zustandssuche eingesetzt werden müssen, ist bei den terminierten Faltungscodes der Anfangs- und Endzustand immer der Nullzustand. Jedem der 2^{Sk} Zustände wird wie folgt eine Metrik zugeordnet: der Anfangswert ist Null. Zuerst werden die 2^k möglichen Übergänge (k Bits) mit den ersten k Bits der Empfangsfolge verglichen und die Zahl der übereinstimmenden Bits zu der Zahl des Ausgangszustands addiert, woraus sich die Metrik des Folgezustands ergibt. Von allen an einem Zustand ankommenden Pfaden wird nur der Pfad mit der größten Metrik beibehalten. Bei Pfaden mit der gleichen Metrik wird eine Zufallsauswahl getroffen. Das Decodierergebnis wird durch den Pfad mit der größten Metrik repräsentiert.

Im folgendem Abschnitt soll die Viterbi-Decodierung anhand des Standarbeispiels und Abbildung 15 noch näher erläutert werden. In Abbildung 11 sieht man die dazugehörigen Ein- und Ausgangsbits für die Zustandsübergänge. Die übertragene Informationsfolge sei „10100“, woraus sich durch Codierung mit dem Beispiel-Encoder die codierte Bitfolge „11 11 10 11 01“ ergibt. Auf der Übertragunsstrecke überlagern sich zwei Bitfehler (00 10 01 00 00), so dass die Folge „11 01 11 11 01“ empfangen wird. Der Decoder ist im Zustand „00“ und empfängt die Bitfolge „11“. Das Zustandsdiagramm in Abbildung 10 gibt Auskunft darüber, dass es aus dem Zustand „00“ die beiden Übergangspfade mit den Ausgangsbits „11“ in den Zustand

„10“ und den Ausgangsbits „00“ unter Beibehaltung des Zustands gibt. Nun wird die Metrik des Übergangs, also die Anzahl der Unterschiedlichen Bits, bestimmt und an den Folgezustand geschrieben, wie es in Abbildung 15a zu sehen ist. Ausgehend von den Folgezuständen „00“ und „10“ wird nun wieder die Metrik der Übergänge bestimmt und zu den schon vorhandenen Metriken addiert. In Abbildung 15b ist gut zu erkennen, dass, sobald in einen Zustand mehrere Pfade führen, die Pfade mit der geringeren Metrik verworfen werden können. Durch Auswertung der weiteren empfangenen Bitfolgen und sukzessives Löschen der Übergänge mit niedriger Metrik bildet sich in Abbildung 15c ein Weg heraus, dessen Metrik größer ist als die der anderen Pfade. Der Pfad mit der Pfad-Metrik $\Delta=8$ durch das Trellisdiagramm ist der wahrscheinlichste und durch Zurückverfolgung ergibt sich die wahrscheinlichste Zustandsfolge, woraus mit Hilfe des Zustandsdiagrammes die ursprüngliche Informationsfolge ermittelt werden kann (vgl. [7], S122-123).

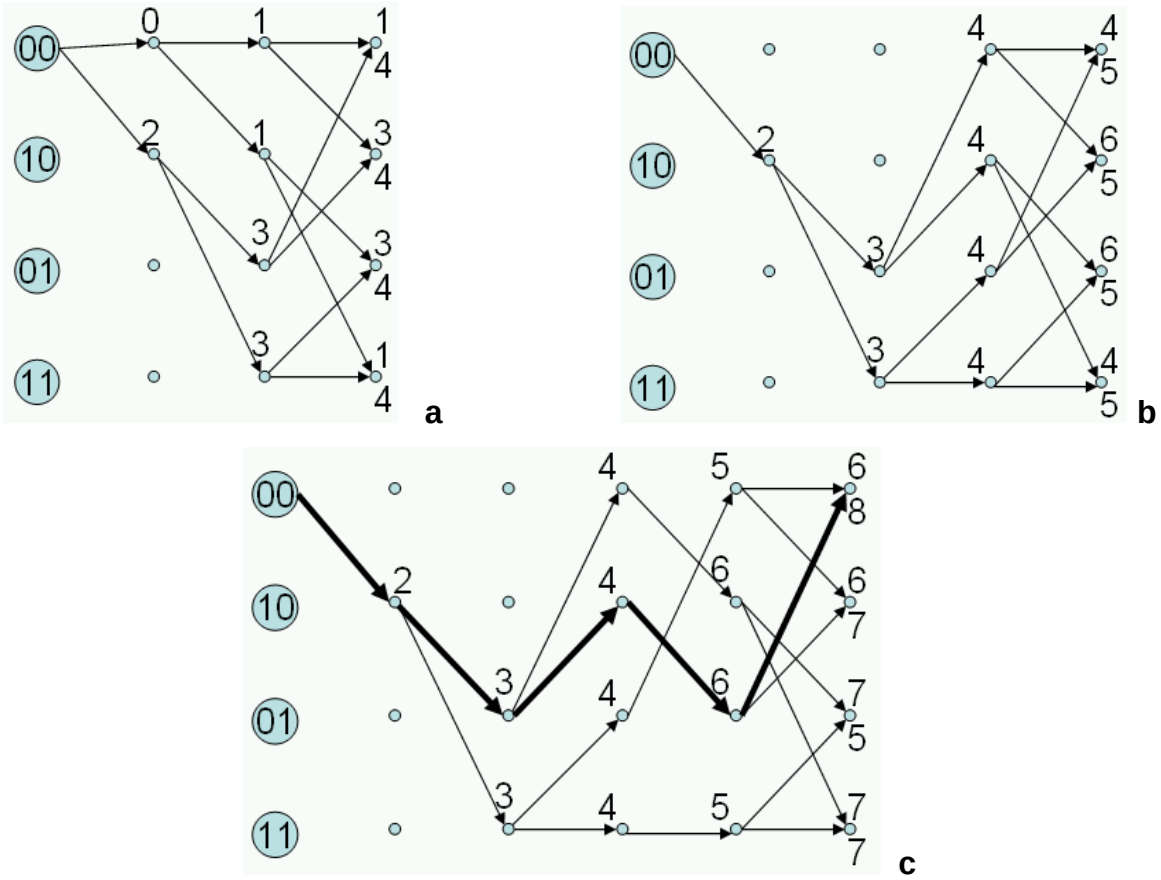


Abbildung 15 Trellisdiagramme zur Beispiel-Viterbi-Decodierung

4.7.7 Hard- und Softdecision

Die Abbildungen 16a und 17a ([7], S.125) zeigen die Wahrscheinlichkeitsdichteverteilungen eines binären Signals x_i, welches über einen Gaußkanal übertragen wurde. Durch die Addition des weißen Rauschen ergibt sich ein Signal y_i , dass nun nicht mehr diskrete Werte annimmt. Bei Harddecision wird der kontinuierliche Wertebereich von y_i durch nur eine Entscheiderschwelle (z. Bsp. 0.5) aufgeteilt. Alle Werte oberhalb der Schwelle werden y_i =1 zugeordnet, die Werte darunter y_i =0. Dagegen gibt es bei Softdecision mehrere Entscheiderschwellen. Diese in den einzelnen Bereichen liegenden Werte werden quantisiert und es entstehen mehrere Zwischenzustände wie in Abbildung 17a bei einer 3 bit Softdecision zu sehen ist. Dies führt zu einer genaueren Abschätzung der Wahrscheinlichkeit, ob ein ausgewählter Pfad durch das Trellis richtig ist, da nun auch nicht ganzzahlige Metriken möglich sind (vgl. [7], S.125).

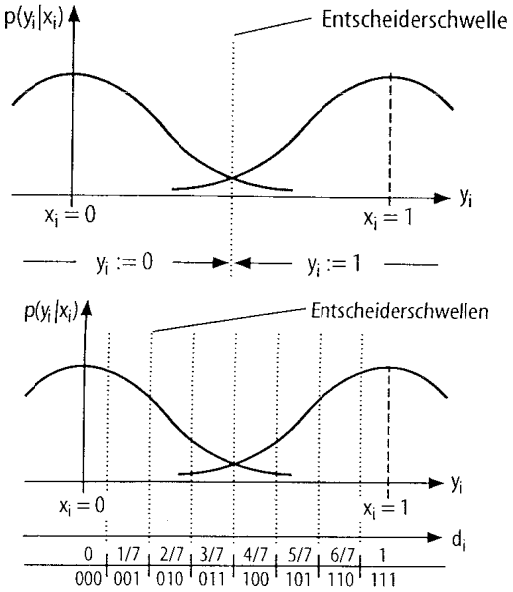


Abbildung 16a Harddecision

Abbildung 17a Softdecision

In den Abbildungen 16b und 17b sieht man Beispiele für die Hard- und Softdecision. Die durchgezogene Kurve entspricht dem Empfangspegel eines Signals und die Kreuze stehen für die Werte nach der Decision. Man erkennt hier deutlich den Unterschied der verschiedenen Quantisierungen.

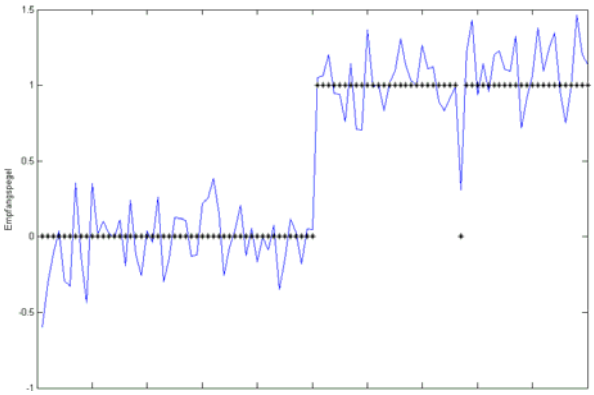


Abbildung 16b Beispiel Harddecision

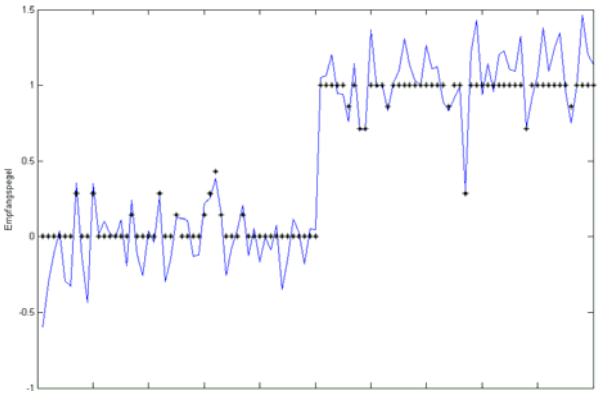


Abbildung 17b Beispiel Softdecision

4.8 Verkettung von Codes

4.8.1 Serielle Verkettung

Die Leistungsfähigkeit der Fehlerkorrektur kann durch eine Verkettung von Codes noch weiter erhöht werden. Damit verbunden ist natürlich auch eine weitere Verminderung der Coderate. In der Abbildung 18 ist die im Praktikumsversuch eingesetzte serielle Codeverkettung inklusive der resultierenden Datenraten dargestellt.

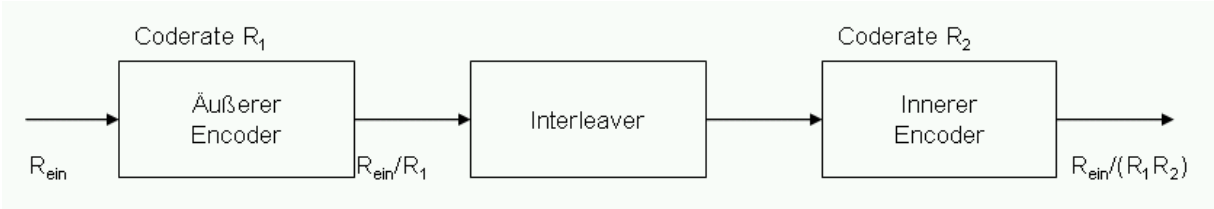


Abbildung 18 Verkettung von Codes

Der erste Code mit der Coderate R_1 wird äußerer Code genannt und wird auf die uncodierte Information mit der Informationsdatenrate R_{ein} angewandt. Durch den inneren Code mit der Coderate R_2 erhöht sich die Datenrate auf insgesamt $R_{ein}/(R_1R_2)$. Es können auf diesem Wege Faltungscodes und Blockcodes untereinander beliebig verkettet werden.

Oftmals geschieht die Codeverkettung auf dem Wege, dass der innere Code den äußeren Code vor zu vielen einzelnen Bitfehlern schützt und daher ein Faltungscode oder ein Blockcode für stochastisch verteilte Einzelfehler ist. Wenn der äußere Code ein symbolorientierter Code ist, so sollte darauf geachtet werden, dass der innere Code jeweils ein oder mehrere Symbole des äußeren Codes schützt wie in

Abbildung 19 ([7], S.129) dargestellt. Hier wird jeweils genau ein Symbol durch den inneren Code geschützt. Wenn bei dem hier betrachteten äußeren Symbolcode ein Interleaver eingesetzt werden soll, ist es sinnvoll einen Interleaver zu benutzen, der die einzelnen Bits der Symbole nicht trennt. Nur die Symbole sollen untereinander verteilt werden, damit die ursprüngliche Symbolstruktur vorhanden bleibt (vgl. [7], S.129).

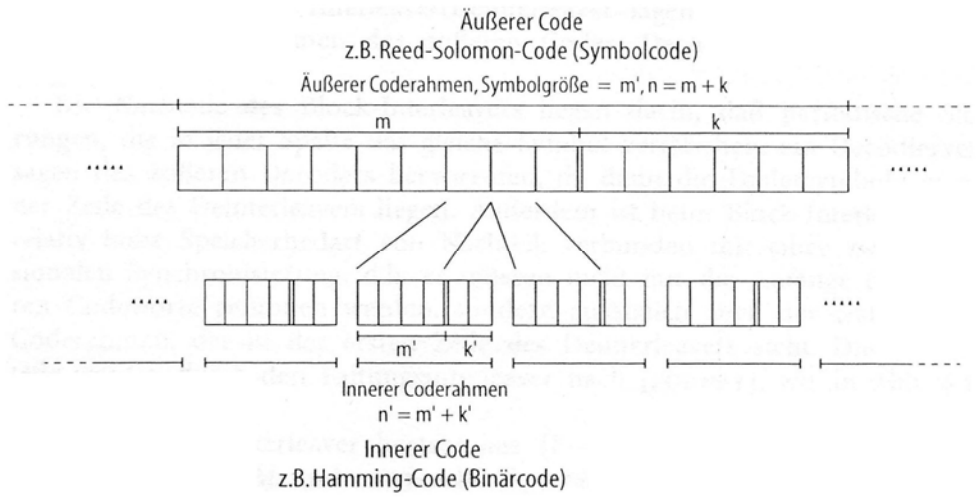


Abbildung 19 Symbolcode – Binärcode

4.8.2 Blockinterleaver

Mit den bisher betrachteten Kanalcodes ist es möglich, kürzere Burstfehler, Symbolfehler und Bitfehler zu korrigieren. Um auch lange Burstfehler korrigieren zu können, muss zwischen den äußeren und inneren Fehlerschutz ein Interleaver geschaltet werden. Dieser bewirkt eine Umsortierung der Symbole, und lange Burstfehler, die sich im Kanal auf die Codefolge lagern, werden auf mehrere Codeworte aufgeteilt, so dass diese eventuell korrigiert werden können. Die Symbole bzw. Bits, die vom äußeren Code ankommen, werden bei dem Interleaving zeilenweise eingelesen und spaltenweise wieder aus der Matrix ausgelesen, bevor sie zum inneren Code kommen. Kennzeichnend für den Interleaver ist die Interleavingtiefe *i*, die die Spalten und Zeilenzahl angibt und in der Abbildung 20 gleich 6 ist.

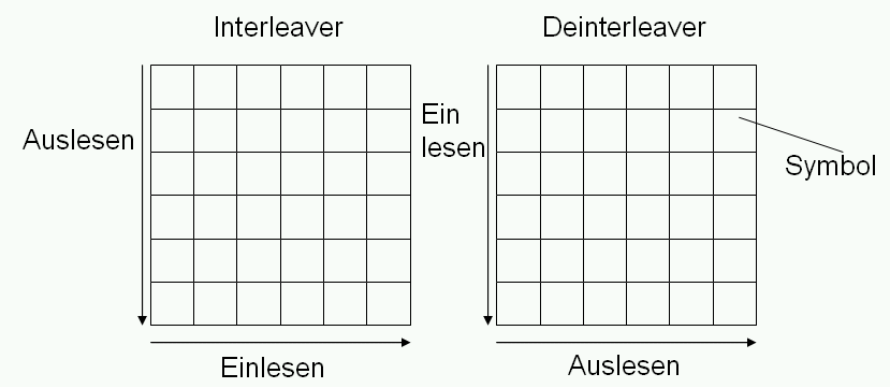


Abbildung 20 Blockinterleaving

Auf der Empfängerseite wird nun dieses Interveaving durch den Deinterleaver rückgängig gemacht.

4.8.3 Produktcodes

Durch den Interleaver wird auch ein Verfahren realisiert, dass als Produktcode bezeichnet wird. Dieser Code besteht aus einer Zeilen und Spaltenstruktur, die zwei aufeinanderfolgenden FEC's unterworfen wird. In Abbildung 21 sieht man den Aufbau eines solchen Codes.

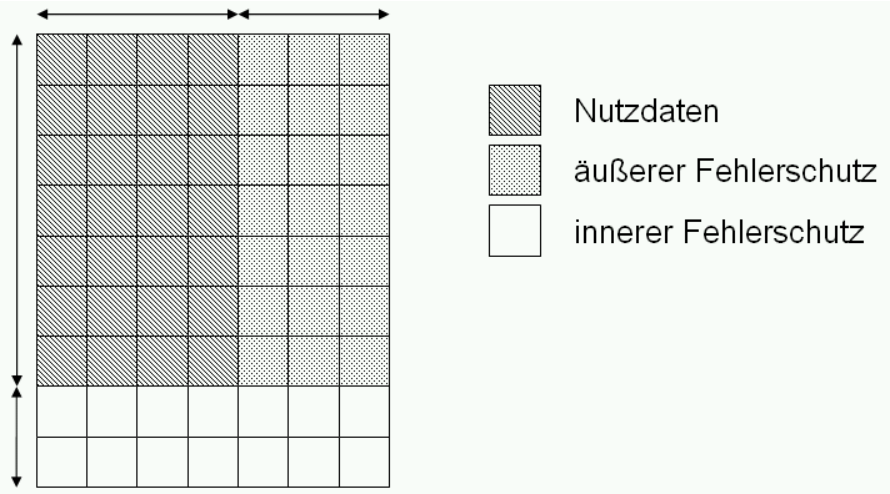


Abbildung 21 Produktcode

Diese Struktur wird mit dem Interleaver erreicht, indem zuerst auf die Nutzdaten der äußere Fehlerschutz angewandt wird, wobei darauf geachtet werden muss, dass diese entstehende Struktur quadratisch ist. Nun schließt sich ein Interleaving mit der Interleavingtiefe gleich der Codewortlänge des äußeren Fehlerschutzes an. Anschließend kann die innere Codierung erfolgen. Diese bietet neben der wiederholten Codierung der Daten auch einen Fehlerschutz des Fehlerschutzes.

5 Diskrete Kanäle

In diesem Versuch werden nur diskrete gedächtnislose Kanäle behandelt. Der Modulator, der physikalische Kanal und der Demodulator werden zusammengefasst und als diskreter Kanal bezeichnet. Dieser fungiert als Blackbox, in den Symbole eines endlichen Eingangsalphabets A_{in} hineingeschickt werden können und Symbole eines endlichen Ausgangsalphabets A_{out} herauskommen. Bei der hier verwendeten Hard-Decision gilt, dass diese beiden Alphabete identisch sind. Gedächtnislos bedeutet, dass das Empfangswort nur von dem aktuell gesendeten Wort abhängig ist.

5.1 Binärer symmetrischer Kanal

Unter dem binären symmetrischen Kanal (BSC) versteht man einen Spezialfall des diskreten gedächtnislosen Kanals mit Hard-Decision und mit dem Alphabet A_{in}= A_{out}={0,1}. Es gilt die Übergangswahrscheinlichkeit

$$P(y|x) = \begin{cases} 1-p_e & y = x \\ p_e & y \neq x \end{cases},$$

wobei *x* ein Sendesymbol und *y* ein Empfangssymbol darstellt. Dieser Kanal, welcher als Modell in Abbildung 22 gezeigt wird, ist durch die Bitfehlerwahrscheinlichkeit p_e eindeutig beschrieben. Für p_e=0.5 ist eine Übertragung nicht mehr möglich (worst case), für p_e=0 ist der Kanal fehlerfrei und für p_e=1 handelt es sich um einen invertierenden Kanal.

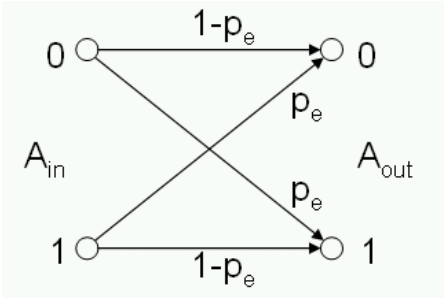


Abbildung 22 Binärer symmetrischer Kanal

5.2 AWGN-Kanal

AWGN-Kanal steht für “Additive White Gaussian Noise“-Kanal, und unter ihm versteht man einen Kanal mit binärem Input, dem weißes normalverteiltes Gaußsches Rauschen hinzuaddiert wird, so dass der Output nun kontinuierlich wird.

Da in diesem Praktikumsversuch binär im Basisband mit der BPSK moduliert wird und dadurch im Demodulator eine binäre Quantisierung stattfindet, ergibt sich hier wieder ein BSC mit endlichem Ausgangsalphabet. Unter der BPSK versteht man das Umwandeln des Bits 1 in das Sendesymbol -1 und des Bits 0 in das Sendesymbol 1. Einer der Vorteile der BPSK ist, dass durch einen größeren Abstand der Symbole zueinander der Signal-Störabstand deutlich erhöht wird. Der Parameter, der den AWGN-Kanal bestimmt, ist der Signal-Rausch-Abstand (SNR)

$$SNR[dB] = 10 \cdot \lg\left(\frac{E_{Signal}}{E_{Rauschen}}\right)$$

und ist durch das Verhältnis von der Energie des Signals zur Energie des Rauschen bestimmt.

6 Praktikumssoftware

Die Oberfläche der Praktikumssoftware ist so gegliedert, dass sie einer Übertragungsstrecke bis zum Kanal entspricht. Alle Hauptkomponenten sind sofort sichtbar und können übersichtlich über Popup-Menüs eingestellt werden. Im folgendem Abschnitt sind die einzelnen Einstellungsmöglichkeiten erklärt.

Quelle:

Die Bitquelle erzeugt eine gleichverteilte Bitfolge mit einer einstellbaren Anzahl an Bits.

Die Nullbitfolge ist eine Folge von Null-Bits, bei der die Anzahl einstellbar ist.

Bei der Symbolquelle kann die Anzahl der Bits von 2 bis 16 und weiterhin die Anzahl der Symbole eingestellt werden. Auch diese Folge ist wieder gleichverteilt.

Wenn die Bildquelle ausgesucht worden ist, kann ein graustufiges GIF-Bild ausgesucht werden. Dieses wird dann in eine Symbolfolge mit 8 Bit pro Symbol gewandelt.

Bei allen Quellenarten ist darauf zu achten, dass die Anzahl nicht zu groß gewählt wird. Es sollte in Abhängigkeit mit den weiteren verwendeten Komponenten zuerst eine sehr niedrige Anzahl (ca. 1000 Bits) eingestellt werden, um die Zeitdauer zu überprüfen. Die Anzahl kann dann bei Bedarf erhöht werden.

Quellencodierung:

Sollte eine Bildquelle oder eine Symbolquelle ausgewählt worden sein, kann hier die arithmetische Quellencodierung dazugeschaltet werden. Diese dient dem selben

Zweck wie die Huffman-Codierung. Eine Beschreibung der arithmetischen Codierung und der Huffman-Codierung findet man in [8] auf den Seiten 26 bis 36.

Äußerer und innerer Kanalcode:

Bei dem Hamming-Code muss die Anzahl der Prüfstellen m (von 2 bis 10) eingestellt werden. Dieser Parameter kann unter dem Menü Codeeigenschaften genauer spezifiziert werden.

Der zyklische Code verlangt die Anzahl der Codewortstellen n und die Anzahl der Nachrichtenwortstellen k. Weiterhin muss ein geeignetes Generatorpolynom gewählt werden. Dies ist in der Form $[X_1\ X_2 \ldots X_n]$ einzugeben, wobei X_i einem binären Koeffizienten des Polynoms entspricht. Diese sind in aufsteigender Reihenfolge einzugeben.

Bei dem BCH-Code muss die Codewortlänge n und die Nachrichtenwortlänge k angegeben werden. Diese Parameter können sich über den Menüpunkt BCH des Menü Codeeigenschaften besorgt werden.

Der RS-Code benötigt wieder die Codewortlänge n und die Nachrichtenwortlänge k sowie die Anzahl der Bits pro Symbol m. Die Parameter n und k sind hier jedoch auf die Symbolanzahl bezogen.

Der Faltungscode besitzt insgesamt vier Parameter. Das Feld Beeinflussungslänge verlangt die Eingabe in der Form $[X_1 \ldots X_n]$, wobei jedes X_i die Beeinflussungslänge für einen Eingang ist und für gewöhnlich Werte von 3 bis 7 annimmt. Die Verbindungsmatrix wird in der Form $[X_{11} \ldots X_{1n}; \ldots; X_{m1} \ldots X_{mn}]$ eingegeben, wobei hier jedes X_{ij} für ein Generatorpolynom in Oktaldarstellung steht. Diese Eingabe kann als Matrix angesehen werden und hat so viele Spalten wie Ausgänge vorhanden sind. Die Anzahl der Zeilen entspricht der Anzahl an Registern bzw. Eingängen. Die Blocklänge gibt die Anzahl der Bits an, nachdem der Faltungscode terminiert werden soll. In das Eingabefeld für die Punktierung muss im Falle einer Punktierung des Codes ein Muster $[X_1 \ldots X_n]$ eingegeben werden, wobei jedes X_i binär ist und für ein Bit der Codeausgangsbitfolge steht. Eine 1 bedeutet nicht punktiertes Bit und eine 0 bedeutet eine Punktierung, also eine Verwerfung des Bits.

Interleaver:

Der Blockinterleaver benötigt als Eingabe nur die Interleavingtiefe, da es sich um einen quadratischen Interleaver handelt. Wenn ein Symbolcode vorangestellt wird, ist

diese Tiefe auf die Symbole bezogen. Andernfalls zeigt diese Größe an, wieviele Bits in eine Zeile bzw. Spalte der Interleavermatrix passen.

Kanaleigenschaften:

Der Kanal kann störungsfrei sein, das bedeutet, dass keine Bitfehler auftreten.
Bei der Auswahl „sym. binär“ handelt es sich um den gewöhnlichen BSC-Kanal mit der einstellbaren Fehlerrate, welche zwischen 0 und 1 liegen muss.
Bei der Auswahl „Burstfehler“ kann der gesendeten Bitfolge ein Büschelfehler einer bestimmten Länge überlagert werden. Die Länge kann über das Startbit und Endbit eingestellt werden. Die Anzahl der Bitfehler innerhalb dieses Fehlers kann über die Fehlerrate eingestellt werden.
Der Symbolfehler mit der Symbolfehlerrate zwischen 0 und 1 überlagert sich auf Symbole. Es kann weiterhin entschieden werden, wie viele Bits einem Symbol entsprechen sollen. Die Anzahl der verfälschten Bits pro Symbol unterscheidet sich.
Der mit der BPSK modulierte Gaußkanal entspricht dem weiter oben beschriebenem AWGN. Hier kann als Parameter nur der Signal-Rausch-Abstand SNR eingestellt werden.

Übertragung starten:

Durch drücken des Button „Start“ wird die Simulation der Übertragungsstrecke gestartet und die Fehlerraten und Anzahl der Fehler werden in dem grau hinterlegten Feld angezeigt. Durch klicken der Schaltfläche „Auswertung“ öffnet sich ein weiteres Menu. In diesem kann man auswählen, welche Bitfolge graphisch angezeigt werden soll.

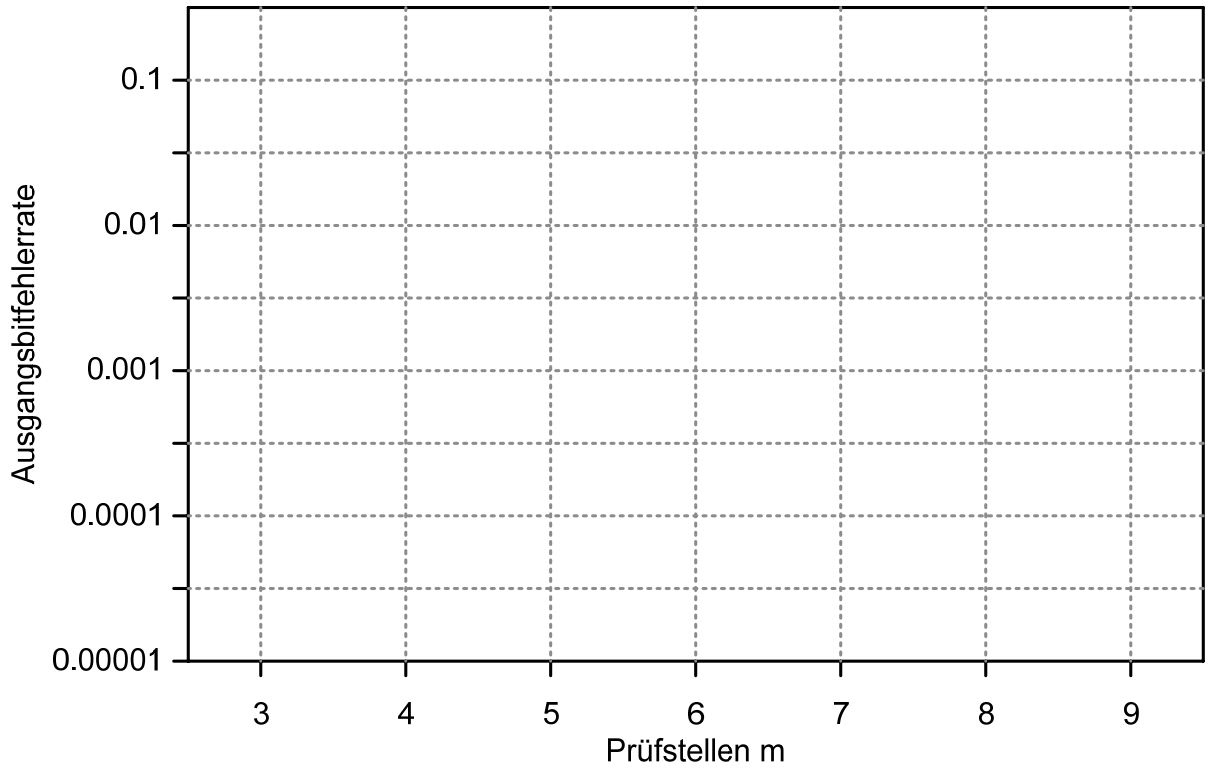
7 Praktikumsversuch

Beachten Sie, dass Sie bei allen gegebenen Aufgaben mit einer niedrigen Zahl an Bits (ca. 1000) beginnen, da bei einer höheren Anzahl an Bits die Rechenzeit dramatisch ansteigen kann. Sind Sie mit dem Ergebnis zufrieden und die Rechenzeit ist akzeptabel, können Sie die Zahl der Eingangsbits erhöhen.

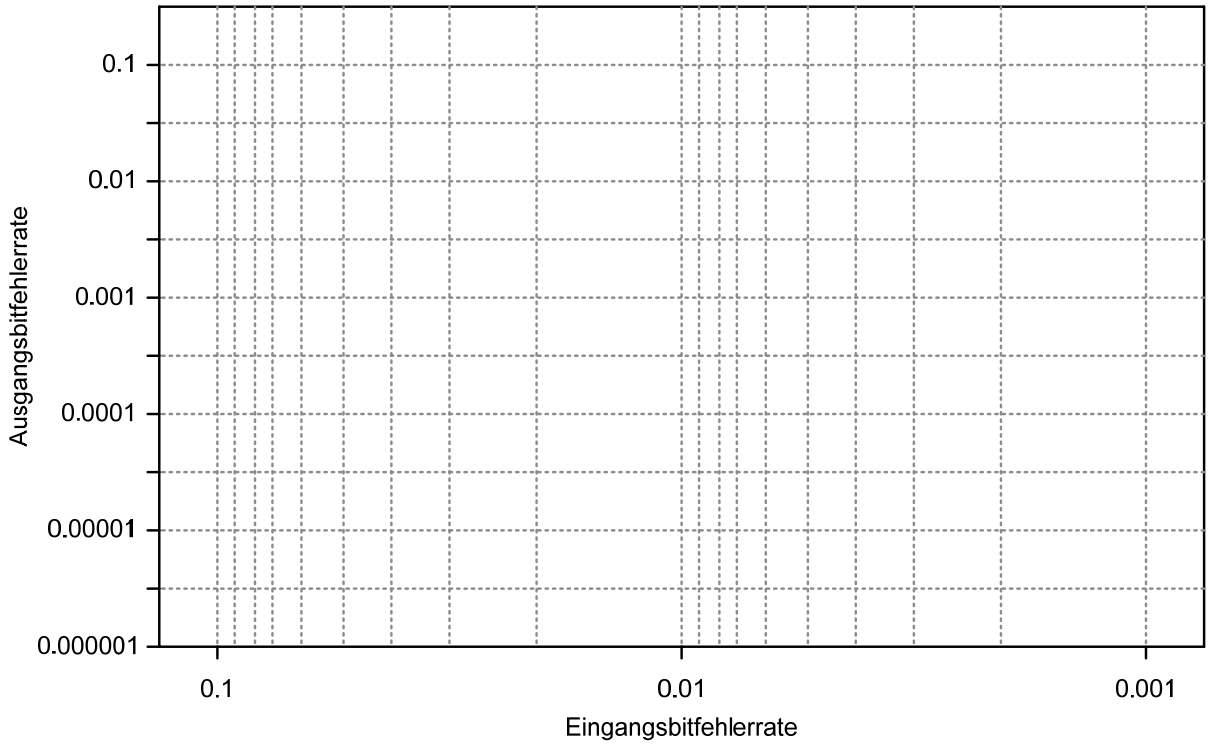
Aufgabe 1

a) Mit welcher Fehlerrate auf dem Kanal BER erreicht man eine Quality Error Rate QER=0.0001 am Ausgang des Decoders bei einer Hamming-Codierung mit m=3 ? Wählen Sie zur Bearbeitung den symmetrischen Binärkanal aus.

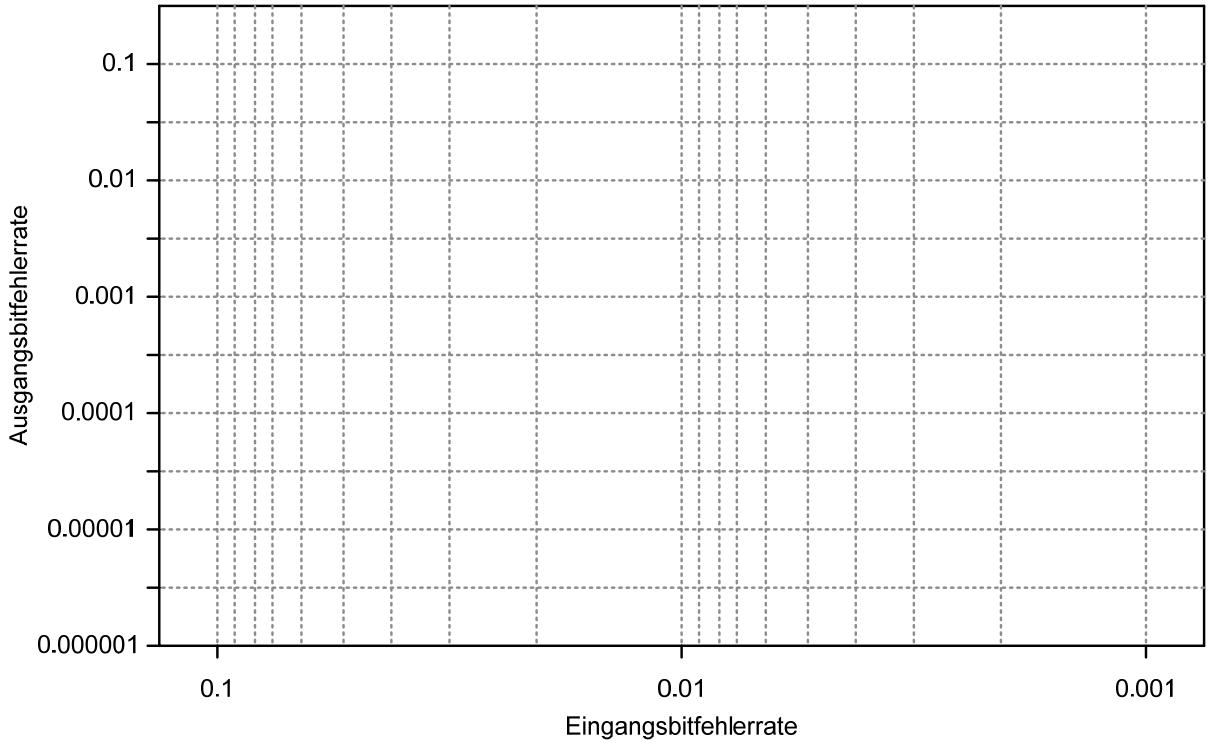
b) Setzen Sie für den Hamming-Code den Parameter m von m=3 nun auf 4,5,…,9 bei gleicher errechneter Bitfehlerrate und skizzieren Sie diesen Verlauf in dem Diagramm. Erklären Sie das beobachtete Verhalten.



c) Zeichnen Sie nun für die Hamming-Codes der Raten R=4/7 und R=26/31 ein Code-Diagramm. Tragen Sie dafür die Eingangsbitfehlerraten von 0.1 – 0.001 gegen die entstehenden Ausgangsbitfehlerraten in das Diagramm ein. Wie groß ist der Codegewinn für die BER am Eingang von 0.001? Wo liegt die Korrekturgrenze (Bitfehlerwahrscheinlichkeit am Ausgang größer oder gleich der Eingangsbitfehler-wahrscheinlichkeit)?



d) Versuchen Sie nun mit Hilfe der Software zwei Faltungscodes mit sehr guten Fehlerkorrektureigenschaften und mit der Beeinflussungslänge von 3 zu finden. Der erste Code soll eine Coderate von R=1/2 und der zweite eine Coderate von R=1/3 aufweisen. Benutzen Sie dafür keine Terminierung und keine Punktierung. Finden Sie auch einen katastrophalen Code? Tragen Sie die Werte wieder in das nachfolgende Diagramm. Suchen Sie sich einige Werte aus, terminieren Sie den Code und stellen Sie fest, ob es eine Verbesserung ergibt.



Aufgabe 2

a) Benutzen Sie für diese Aufgabe die Bildquelle mit dem Bild GTI.gif. Bis zu welcher Bitfehlerrate auf dem Kanal ist die Qualität subjektiv noch sehr gut (Fehlerfrei)? Bis zu welcher Bitfehlerrate hat das Bild noch hinreichend gute Qualität (nahezu fehlerfrei)? Ab wann ist die Bildqualität ohne FER zu schlecht (Fehler sofort wahrnehmbar)?

b) Aufgrund stark begrenzter Leitungskapazitäten dürfen nicht mehr als 150 kByte über den Kanal übertragen werden. Trotzdem soll das übertragene Bild eine hinreichend gute Qualität haben. Wie groß darf die Bitfehlerrate auf dem BSC Kanal höchstens sein, wenn hierfür ein Hamming-Code benutzt werden soll?

e) Eine 8-bit Symbolfolge soll mit einer Datenrate von R=20 Symbolen/Sekunde inklusive der Kanalcodierung über einen Gaußkanal geschickt werden. Die Bits sind BPSK moduliert (Kanaleigenschaft) und diese Folge soll mit einem optimalen Faltungscode der Rate R=1/2 und der Beeinflussungslänge 3 codiert werden. In einer Minute darf höchstens ein Bitfehler in den Informationsstellen auftreten. Wie niedrig darf der Signal-Rausch-Abstand noch sein? Um wie viel dB kann sich der SNR verschlechtern, wenn ein Faltungscode mit der Coderate R=1/3 gewählt wird?

Aufgabe 3

In dieser Aufgabe sollen verschiedene Kanäle untersucht und ein passender Kanalcode für die Übertragung gefunden werden. Untersuchen Sie den Kanal und schauen Sie sich die auftretenden Fehler an. Sie können die von Ihnen ausgesuchten Codes auch erst mal auf einem vorher definierten Kanal mit Burstfehlern, Symbolfehlern oder Bitfehlern testen.

a) Wie viel Redundanz wird benötigt, um mit einem Blockcode 1000 Bits fehlerfrei über den Kanal A zu übertragen? Die Codewortlänge soll dabei nicht größer als 255 Bits sein.

b) Es sollen über den Kanal B 10.000 Bits übertragen werden. Dabei dürfen bei fünf Übertragungen dieser Bits nicht mehr als 10 Fehler auftreten und die Coderate eines beliebigen Codes muss mindestens R=1/3 betragen.

c) 10.000 Bits sollen fehlerfrei über den Kanal C übertragen werden. Die Coderate soll mindestens R=0.6 betragen, die Blockstruktur des Codewortes soll aufgrund Speichermangel unter 100 Bits liegen und es darf kein Interleaver benutzt werden.

d) Für die binäre Übertragung über den gestörten Kanal D soll ein Code festgelegt werden. Zur Auswahl stehen BCH Codes oder ein RS-Code der Länge 31. Legen Sie jeweils die günstigsten Parameter fest. Es sollen alle auftretenden Fehler korrigiert werden. Welcher Code eignet sich hierfür besonders?

e) Gibt es eine Möglichkeit, 1000 Bits mit einem Code über den Kanal E zu senden, ohne dass ein Fehler auftritt. Beachten Sie, dass ein Codewort nicht über 256 Bits lang sein darf und die Coderate mindestens R=1/3 betragen soll. Wenn dies nicht möglich ist, wie hoch ist die Ausgangsbitfehlerrate mit dem geeignetsten Code?

f) Versuchen Sie eine fehlerfreie Übertragung von 10.000 Bits über den Kanal F zu realisieren. Benutzen Sie dafür 2 Codes in einer Serienschaltung ohne Interleaver. Die beiden Codes dürfen jeweils die Coderare R=1/3 und zusammen die Coderate von R=1/5 nicht unterschreiten.

8 Literaturverzeichnis

[1] Lüke, H. D.: Signalübertragung. 3. erweiterte Auflage. Springer Verlag, Berlin Heidelberg, 1985.

[2] Friedrichs, B.: Kanalcodierung: Grundlagen und Anwendungen in modernen Kommunikationssystemen. 1. Auflage. Springer Verlag, Berlin Heidelberg, 1996.

[3] Rohling, H.: Einführung in die Informations- und Codierungstheorie. 1. Auflage. Teubner Verlag, Stuttgart, 1995.

[4] Czylwik, A.: Vorlesungsskript Codierungstheorie. Technische Universität Braunschweig, 2001.

[5] Reimers, U.: Vorlesungsskript Aktuelle Systeme für die Elektronischen Medien. Technische Universität Braunschweig, 2003.

[6] Schneider-Obermann, Herbert: Kanalcodierung: Theorie und Praxis fehlerkorrigierender Codes. 1. Auflage. Vieweg Verlag, Braunschweig/Wiesbaden, 1998.

[7] Reimers, U.: Digitale Fernsehtechnik: Datenkompression und Übertragung für DVB. 1. Auflage. Springer Verlag, Berlin Heidelberg, 1995.

[8] Musmann, H. G.: Vorlesungsskript Quellencodierung. Universität Hannover, 2000.