Vorlesungsskript Kanalcodierung II

von
DR.-ING. VOLKER KÜHN

aktualisiert von
DR.-ING. DIRK WÜBBEN

Fachbereich Physik/Elektrotechnik (FB 1)
Arbeitsbereich Nachrichtentechnik
Postfach 33 04 40
D-28334 Bremen

Version 2.4

(04.04.2011)

Inhaltsverzeichnis

Kapitel 1

Verkettete Codes

1.1 Einführung

Motivation
Ziel der Codierungstheorie ist es, die Kanalkapazität von Shannon zu erreichen. Praktische Codes wie die im letzten Semester behandelten Faltungs- und Blockcodes sind weit von dieser theoretischen Grenze entfernt. Ein entscheidender Nachteil von ihnen besteht darin, dass mit zunehmender Leistungsfähigkeit auch der Decodieraufwand ansteigt, meistens wächst er exponentiell an. Daher setzt die praktische Realisierbarkeit der Leistungsfähigkeit dieser Codes schnell Grenzen. Zwar ist es oft nur eine Frage der Zeit, bis die Technologie einen höheren Aufwand bei der Decodierung erlaubt. Trotzdem stellt sich die Frage, ob nur durch Vergrößerung der Einflusslänge bei Faltungscodes bzw. der Blocklänge bei Blockcodes die Kapazität nach Shannon prinzipiell erreicht werden kann.

Einen anderen Weg zur Konstruktion leistungsfähiger FEC-Codes zeigte Forney bereits im Jahr 1966, als er erstmals die Verkettung einfacher Teilcodes vorstellte [For66]. Seit dieser Zeit sind gerade in den vergangenen Jahren enorme Fortschritte zu verzeichnen gewesen. Hervorzuheben ist hier das Jahr 1993, als zum ersten Mal die sogenannten *Turbo-Codes* präsentiert wurden, einer geschickten parallelen Anordnung zweier Faltungscodes, mit denen die Kapazitätsgrenze von Shannon so dicht wie noch nie erreicht wurde (Abstand nur noch 0,5 dB bei einer Fehlerrate von $P_b = 10^{-5}$).

Idee
Die grundlegende Idee besteht also darin, einfache Codes geschickt miteinander zu verknüpfen, so dass ein Gesamtcode entsteht, der leistungsfähiger als die einzelnen Komponentencodes ist. Gleichzeitig muss er aber auch einfacher zu decodieren sein. Dies soll an einem Beispiel näher erläutert werden. Die heute zur Verfügung stehende Technologie erlaubt beispielsweise im Consumerbereich die Decodierung eines Faltungscodes mit der Einflusslänge $L_c = 9$, d.h. es ist ein Trellisdiagramm bestehend aus $2^8 = 256$ Zuständen abzuarbeiten. Würde man nun 2 Faltungscodes mit $L_c = 3$ verknüpfen, ergäbe sich ein Aufwand, der einem Trellisdiagramm von $2 \cdot 2^2 = 8$ Zuständen, also nur der 64-ste Teil des Aufwandes. Berücksichtigt man nun eine wiederholte Decodierung des verketteten Codes (Turbo-Decodierung, z.B. 6 Iterationen, mehr dazu später), ergibt sich ein Aufwand von $6 \cdot 8 = 48$ Zuständen, also immer noch weniger als ein Fünftel des ursprünglichen Aufwandes. Ob der resultierende Gesamtcode genauso gut oder gar besser ist, wird im Folgenden ausführlich diskutiert.

Prinzipiell unterscheiden wir zwei Arten der Codeverkettung.

Serielle Verkettung

- Codes sind seriell hintereinander angeordnet
- Jeder nachfolgende Codierer codiert den gesamten Datenstrom inklusive der bereits erzeugten Redun-

danzbit

- Bei zwei Teilcodes spricht man auch von einem inneren Code (C_2) und einem äußeren Code (C_1)

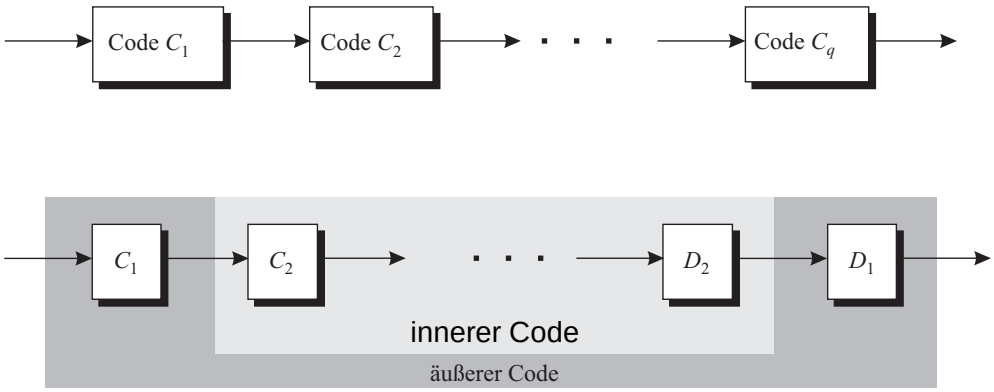


Bild 1.1: Serielle Codeverkettung

Parallele Verkettung

- Teilcodes sind nun parallel angeordnet
- Jeder Teilcodierer erhält nur die Informationsbit, nicht die Redundanzbit der übrigen Codierer
- Die Ausgangssignale der einzelnen Teilcodierer sind durch parallel-seriell-Umsetzung zu einem Datenstrom zusammenzufügen.

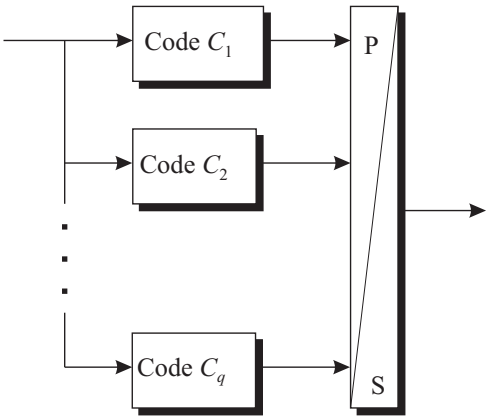


Bild 1.2: Parallele Codeverkettung

In vielen praktischen Systemen sind ebenfalls verkettete Codiersysteme zu finden. So kommen z.B. im GSM-Netz seriell verknüpfte Faltungscodes und CRC-Codes zum Einsatz. Während erstere zur Fehlerkorrektur dienen, bewerkstelligen letztere die Fehlererkennung. Sehr wichtige Kontrollkanäle verwenden anstelle der CRC-Codes auch Fire-Codes zur Fehlerkorrektur.

Bei Weltraummissionen wie beispielsweise der Galileo-Mission findet die Übertragung bei sehr niedrigen Signal-Rausch-Abständen statt, da die Sendeleistung streng begrenzt ist und sehr große Entfernungen zu überbrücken sind. Aus diesem Grund sind hier Codes zu bevorzugen, die unter diesen extremen Randbedingungen gute Ergebnisse erzielen. Das System der NASA verwendet als inneren Code einen Faltungscode, für den mit dem Viterbi-Algorithmus ein effizientes Verfahren zur Soft-Decision-Decodierung zur Verfügung steht. Als äußerer Code wird ein Reed-Solomon-Code eingesetzt, der sehr gut zur Korrektur von Bündelfehlern geeignet ist und somit die Fehlerbündel am Ausgang des Viterbi-Algorithmus korrigieren kann.

Im Rahmen dieser Vorlesung sollen nicht alle Einzelheiten, die bei der Verkettung von Codes eine Rolle spielen behandelt werden. Vielmehr ist es die Aufgabe, ein grundlegendes Verständnis für die grobe Funktionsweise verschiedenen Prinzipien zu entwickeln. Insbesondere die iterative Decodierung in Kapitel 1.7 mit ihren speziellen Decodieralgorithmen ist ein sehr komplexes Feld, das nicht in seiner Vollständigkeit abgehandelt werden kann und noch Gegenstand der aktuellen Forschung ist.

Abschnitt 1.3.2 enthält einen einfachen Einstieg in das Gebiet der verketteten Codes (*concatenated codes*) und erläutert am Beispiel einfacher Produktcodes deren Aufbau und Struktur sowie den Unterschied zwischen serieller und paralleler Verkettung. Danach werden in aller Kürze die erwähnten Turbo-Codes vorgestellt und im Anschluß ein leistungsfähiges Verfahren zur Decodierung verketteter Codes erklärt. Bevor jedoch auf eine geschickte Verkettung einfacher Teilcodes eingegangen wird, erfolgt in nächsten Abschnitt zunächst ein kurzer Einschub zur Erklärung des *Interleaving*.

1.2 *Interleaving* (Codespreizung)

Diese auch als **Codespreizung** oder **Verschachtelung** bekannte Technik beeinflußt in hohem Maße die Leistungsfähigkeit verketteter Codes und stellt somit eine wichtige zu optimierende Komponente dar. Interleaver werden nicht nur in Zusammenhang mit verketteten Codes verwendet, sondern kommen auch bei der Spreizung von Bündelfehlern, die z.B. durch Fading entstanden sind, zum Einsatz.

1.2.1 Blockinterleaver

Der Begriff *Interleaving* beschreibt die Permutation einer Symbolfolge $\mathbf{x}$, d.h. die Veränderung der Reihenfolge der in $\mathbf{x}$ enthaltenen Symbole. Der einfachste Fall ist der sogenannte **Blockinterleaver**, welcher in Bild 1.3 abgebildet ist. In diesem Beispiel besteht er aus 3 Zeilen und 5 Spalten. Er wird Spalte für Spalte mit dem Eingangsvektor

$$\mathbf{x} = (x_0\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ x_{10}\ x_{11}\ x_{12}\ x_{13}\ x_{14})$$

beschrieben, allerdings zeilenweise ausgelesen. Somit erhalten wir

$$\tilde{\mathbf{x}} = (x_0\ x_3\ x_6\ x_9\ x_{12}\ x_1\ x_4\ x_7\ x_{10}\ x_{13}\ x_2\ x_5\ x_8\ x_{11}\ x_{14})$$

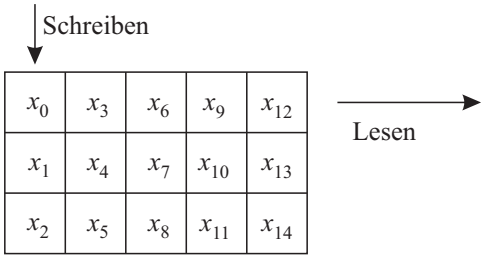


Bild 1.3: Allgemeiner Aufbau eines Interleavers

Es ist zu erkennen, dass zwischen ursprünglich benachbarten Symbolen nun ein Abstand von $L_I = 5$ existiert. Dieser Abstand wird als **Interleavingtiefe** bezeichnet. Die optimale Dimensionierung des Interleavers hängt von mehreren Faktoren ab und wird auch durch den jeweiligen Verwendungszweck beeinflußt.

Anzahl der Spalten
Die Anzahl der Spalten bestimmt direkt die Interleavingtiefe L_I. Sollen beispielsweise Bündelfehler bis zu einer Länge b am Eingang eines Viterbi-Decodierers aufgebrochen werden, muss $L_I \geq b$ gelten, damit sichergestellt ist, dass sich der Bündelfehler in L_I Einzelfehler aufteilt. Dabei ist leicht einzusehen, dass die Interleavingtiefe vergrößert werden kann, wenn die Anzahl der Spalten erhöht wird.

Anzahl der Zeilen
Die zweite Dimension, also die Anzahl der Zeilen, hängt von folgender Betrachtung ab. Bei einem Faltungscode der Einflusslänge $L_c = 5$ sind beispielsweise aufgrund des Gedächtnisses 5 aufeinander folgende Codeworte eng miteinander korreliert. Bei einer Coderate von $R_c = 1/2$ heißt das, dass 10 aufeinander folgende Bit korreliert sind. Sollen diese durch den Interleaver möglichst weit auseinander gespreizt werden (nämlich mit L_I, um sie gut vor Bündelfehlern zu schützen), so muss die Zeilenzahl $L_c/R_c = 10$ betragen. Dann ist sichergestellt, dass jedes dieser 10 Bit zu seinem Nachbarn einen Abstand von L_I besitzt.

Verzögerungszeiten
Dabei gilt zu beachten, dass der Inhalt in der Regel erst ausgelesen werden kann, wenn der Speicherbereich komplett beschrieben worden ist. Hierdurch entsteht eine systembedingte Verzögerungszeit

$$\Delta t = \text{Zeilen} \cdot \text{Spalten} \cdot T_b \tag{1.1}$$

(Regelungstechniker würden von Totzeit sprechen), die kritische Werte nicht überschreiten darf. So gibt es für die Duplex-Sprachübertragung zwar keinen festen Grenzwert einer maximalen Verzögerungszeit, mehr als einige 10 ms gelten aber als nicht tolerierbar. Hieraus folgt, dass z.B. bei einer Datenrate von 9,6 kbit/s eine Interleavergröße von nur etwa 400 bit schon an der oberen Grenze liegt, denn es gilt für Interleaving (Sender) und De-Interleaving (Empfänger)

$$2 \cdot \Delta t = 2 \cdot \frac{400}{9600/s} = 83,3 \text{ ms}.$$

Im GSM-Netz werden für den vollratigen Sprachkanal etwa 22,8 kbit/s übertragen. Für diese Bruttodatenrate ergibt sich bei 400 Bit Interleavergröße eine Verzögerungszeit von

$$2 \cdot \Delta t = 2 \cdot \frac{400}{22.800/s} = 35 \text{ ms}.$$

In der Literatur findet man auch Angaben über maximal zulässige Verzögerungszeiten die wesentlich restriktiver sind (z.B. 20 ms).

1.2.2 Faltungsinterleaving

Auf das sogenannte Faltungsinterleaving soll an dieser Stelle nicht weiter eingegangen werden. Es sei nur soviel gesagt, dass die Funktionsweise der des Blockinterleavings stark ähnelt. Allerdings erreicht man die gleiche Interleavingtiefe schon für geringere Verzögerungszeiten. Für weiterführende Informationen wird auf die Literatur [Fri96] verwiesen.

1.2.3 Zufallsinterleaving

Insbesondere das Blockinterleaving sorgt mit seiner sehr regelmäßigen Struktur dafür, dass Symbole mit einem gewissen Abstand auch nach der Verschachtelung noch den gleichen Abstand zueinander oder aber ein ganzzahliges Vielfaches davon besitzen. Dies führt insbesondere bei den später noch zu behandelnden Turbo-Codes dazu, dass sich schlechte Distanzeigenschaften des Codes ergeben. Daher ist es in diesen Fällen von Vorteil, die Codespreizung quasi-zufällig zu gestalten, d.h. die Art der Permutation ist dem Empfänger natürlich bekannt, sie sollte aber keine systematische Struktur aufweisen. Derartige Interleaver können nicht systematisch konstruiert werden, sondern müssen per Rechnersuche iterativ optimiert werden.

1.3 Serielle Codeverkettung

1.3.1 Vorbetrachtungen

Beispiel 1: Serielle Verkettung von (3,2,2)-SPC-Code und (4,3,2)-SPC-Code
Wir wollen uns anhand eines sehr einfachen Beispiels der geschickten Verkettung zweier Codes nähern. Dazu betrachten wir zunächst einen einfachen (3,2,2)-SPC-Code (*Single-Parity-Check*), der seriell mit einem (4,3,2)-SPC-Code verknüpft werden soll. Die Coderate des verketteten Codes beträgt $R_c = 2/4 = 1/2$, über die Mindestdistanz muss im Folgenden diskutiert werden.

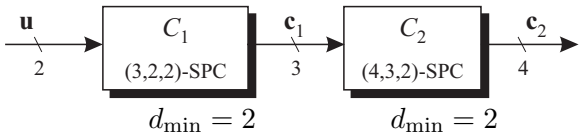


Bild 1.4: Serielle Codeverkettung zweier SPC-Codes

SPC-Codes hängen dem Informationswort $\mathbf{u}$ ein einfaches Paritätsbit an, sie besitzen also die Minimaldistanz $d_{\min} = 2$. Es stellt sich nun die Frage, welche Mindestdistanz die Verkettung hat. Rein anschaulich würde man eine Distanz größer als 2 vermuten. Für dieses einfache Beispiel lässt sich das Distanzspektrum noch anhand einer Tabelle erläutern.

$\mathbf{u}$	$\mathbf{c}_1$	$\mathbf{c}_2$	$w_H(\mathbf{c}_2)$
0 0	0 0 0	0 0 0 0	0
0 1	0 1 1	0 1 1 0	2
1 0	1 0 1	1 0 1 0	2
1 1	1 1 0	1 1 0 0	2

Tabelle 1.1: Codeworte der Verkettung von äußerem (3,2,2)-SPC und innerem (4,3,2)-SPC

Da es sich um einen linearen Code handelt, reicht es aus, die Gewichte der Codewörter zu betrachten und nicht die Distanzen untereinander. Es ist sofort ersichtlich, dass die Mindestdistanz nach wie vor $d_{\min} = 2$ beträgt. Offensichtlich führt eine Verkettung von Codes nicht automatisch zu einer Verbesserung der Distanzeigenschaften. Das folgende Beispiel veranschaulicht, dass bei ungeschickter Kombination zweier Codes der Gesamtcode die freie Distanz des inneren Codes übernimmt.

Beispiel 2: Serielle Verkettung von (4,3,2)-SPC (C_1) und (7,4,3)-Hamming-Code (C_2)
Die Coderate dieser Verkettung lautet $R_c = 3/7$.

$\mathbf{u}$	$\mathbf{c}_1$	$\mathbf{c}_2$	$w_H(\mathbf{c}_2)$	$\tilde{\mathbf{c}}_2$	$w_H(\tilde{\mathbf{c}}_2)$
0 0 0	0 0 0 0	0 0 0 0 0 0 0	0	0 0 0 0 0 0 0	0
0 0 1	0 0 1 1	0 0 1 1 0 0 1	3	0 0 0 1 1 1 1	4
0 1 0	0 1 0 1	0 1 0 1 0 1 0	3	0 1 1 0 0 1 1	4
0 1 1	0 1 1 0	0 1 1 0 0 1 1	4	0 1 1 1 1 0 0	4
1 0 0	1 0 0 1	1 0 0 1 1 0 0	3	1 0 1 0 1 0 1	4
1 0 1	1 0 1 0	1 0 1 0 1 0 1	4	1 0 1 1 0 1 0	4
1 1 0	1 1 0 0	1 1 0 0 1 1 0	4	1 1 0 0 1 1 0	4
1 1 1	1 1 1 1	1 1 1 1 1 1 1	7	1 1 0 1 0 0 1	4

Tabelle 1.2: Codeworte der Verkettung von äußerem (4,3,2)-SPC und innerem (7,4,3)-Hamming-Code ($\tilde{\mathbf{c}}_2$ sind geschickt gewählte Codeworte zur Erhöhung der Mindestdistanz auf $d_{\min} = 4$)

Der äußere (4,3,2)-SPC-Code besitzt die Mindestdistanz $d_{\min}^{(1)} = 2$, während der innere (7,4,3)-Hammingcode

bekannterweise die Mindestdistanz $d_{\min}^{(2)} = 3$ hat. Sie stellt gemäß obiger Tabelle auch die kleinste Distanz des verketteten Codes dar. Hieraus folgt, dass die Verkettung zweier Codes nicht zwingend zu einem besseren Gesamtcode führt. Vielmehr wirkt sich der äußere Code gar nicht aus und die Mindestdistanz des inneren Codes bestimmt auch die Mindestdistanz des Gesamtcodes.

Verantwortlich für das Scheitern einer Codeverkettung ist die Zuordnung der Codeworte $\mathbf{c}_1$ des äußeren Codes auf die Codeworte $\mathbf{c}_2$ des inneren. Der äußere Codierer sorgt durch das Hinzufügen von Redundanz für eine Teilmengenbildung, d.h. den $2^4 = 16$ prinzipiell möglichen Eingangsworten am inneren Codierer treten tatsächlich nur $2^3 = 8$ auf. Dies hat zur Folge, dass von 16 möglichen Hamming-Codeworten nur 8 verwendet werden. Erfolgt die Auswahl der 8 verwendeten Codeworte so ungeschickt, dass sie zufällig die kleinst mögliche Hamming-Distanz untereinander aufweisen, wird hierdurch auch die Gesamtdistanz des verketteten Codes dominiert; der äußere Code wirkt sich nicht auf die Distanzeigenschaften aus. Hieraus kann das Ziel formuliert werden, dass die tatsächlich benutzten 8 Codeworte so zu wählen sind, dass sie möglichst große Distanzen untereinander aufweisen. Dann würde sich – wie in Tabelle 1.2 mit $\tilde{\mathbf{c}}_2$ gezeigt – für den Gesamtcode eine höhere minimale Hamming-Distanz (im Beispiel $d_{\min} = 4$) ergeben, als sie der innere Code besitzt.

Wie muss die Verkettung von Codes im allgemeinen erfolgen, damit sich für den verketteten Code optimale Distanzeigenschaften ergeben?

1.3.2 Produktcodes

Eine geschickte Verkettung von Codes stellen die Produktcodes dar. Auch wenn es auf den ersten Blick nicht sofort ersichtlich ist, stellt der folgende Aufbau die serielle Verkettung zweier Codes dar.

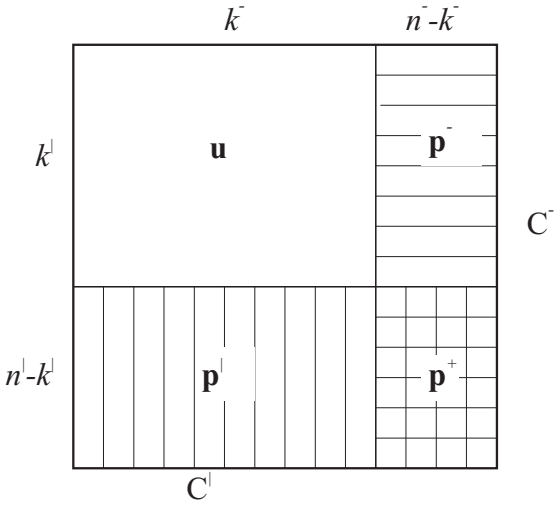


Bild 1.5: Allgemeiner Aufbau eines Produktcodes

Ein Matrix $\mathbf{u}$ bestehend aus $k^| \cdot k^-$ Informationsbit wird in $k^|$ Zeilen und k^- Spalten angeordnet. Die Informationsbit dieser Matrix werden dann zeilenweise einer *horizontalen Codierung* durch einen systematischen (n^-, k^-, d^-)-Blockcode C^- unterzogen. Jede Zeile stellt ein eigenes Codewort dar, so dass insgesamt $k^|$ Codewörter generiert wurden. Die erzeugten Prüfbit sind in der Prüfmatrix $\mathbf{p}^-$ zusammengefaßt. Die Coderate des horizontalen Codes beträgt $R_c^- = \frac{k^-}{n^-}$.

Anschließend erfolgt eine *vertikale Codierung* aller bisher erzeugten Codeworte mit einem ebenfalls systematischen $(n^|, k^|, d^|)$-Blockcode $C^|$, wobei jede Spalte ein eigenes Codewort darstellt. Die Prüfbit der Informationsmatrix $\mathbf{u}$ sind in $\mathbf{p}^|$ zusammengefaßt worden, die Prüfbit der Prüfmatrix $\mathbf{p}^-$ in $\mathbf{p}^+$. Die Coderate beträgt hier $R_c^| = \frac{k^|}{n^|}$.

Anmerkung:
Werden lineare Teilcodes eingesetzt (wir gehen in der Vorlesung immer davon aus), so stellen alle Zeilen als auch alle Spalten gültige Codeworte von C^- bzw. $C^|$ dar. Dies ist nicht unbedingt selbstverständlich, da die Prüfmatrix $\mathbf{p}^+$ lediglich durch die vertikale Codierung erzeugt wurde. Diese stellt aber eine Linearkombination der horizontalen Codeworte dar, wodurch sich aufgrund der Linearität wiederum ein gültiges Codewort aus C^- ergibt.

Die in Bild 1.5 dargestellte Anordnung entspricht einer seriellen Verkettung von C^- und $C^|$, da alle von C^- erzeugten Codeworte anschließend komplett durch $C^|$ abermals codiert werden.

Die Coderate lautet somit

$$R_c = \frac{k^- \cdot k^|}{n^- \cdot n^|} = R_c^- \cdot R_c^| \tag{1.2}$$

und setzt sich somit aus dem Produkt der Coderaten von C^- und $C^|$ zusammen. Für die Minimaldistanz des Produktcodes gilt

$$d_{\min} = d_{\min}^- \cdot d_{\min}^| \, . \tag{1.3}$$

Während Gl. (1.2) selbsterklärend ist, lässt sich Gl. (1.3) folgendermaßen veranschaulichen. Wir stellen uns vor, dass $\mathbf{u}$ nur eine Zeile enthält, die Binärstellen ungleich Null enthält. Das sich ergebende Codewort dieser Zeile habe das minimale Gewicht $d_{\min}^-$ des horizontalen Codes. Für jede der $d_{\min}^-$ Binärstellen ungleich Null (und nur für diese) erzeugt der vertikale Code $C^|$ genau $d_{\min}^-$ Codeworte, welche jeweils mindestens das Gewicht $d_{\min}^|$ besitzen. Die kleinste vorkommende Distanz ergibt sich also aus dem Produkt der minimalen Distanzen der beiden Teilcodes und ist somit größer als diese.

Frage: Worin besteht der Unterschied zwischen den Produktcodes und der einfachen seriellen Verkettung?

Der Produktcode aus Bild 1.5 besitzt inhärent einen Blockinterleaver. Dies ist der Tatsache zu entnehmen, dass C^- mit der zeilenweisen Codierung andere Informationsbit zu einem Codewort zusammenfaßt als $C^|$ mit der vertikalen Codierung. Produktcodes enthalten also einen Blockinterleaver mit n^- Spalten und $k^|$ Zeilen. Hierdurch erhöht sich die Mindestdistanz des verketteten Codes gegenüber dem Beispiel aus Abschnitt 1.3.1.

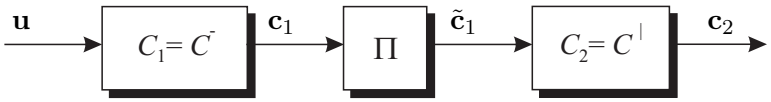


Bild 1.6: Serielle Verkettung zweier Codes mit Interleaving

Die Auswirkungen einer größeren Mindestdistanz sollen an folgendem Beispiel verdeutlicht werden.

Beispiel 3:
Wir konstruieren einen (12,6,4)-Produktcode aus den schon von Beispiel 1 in Abschnitt 1.3.1 bekannten (3,2,2)- und (4,3,2)-SPC-Codes. (siehe Bild 1.7). Die Coderate des Produktcodes $R_c = 6/12 = 1/2$ verändert sich nicht gegenüber der des verketteten Codes ohne Codespreizung. Der Unterschied besteht vielmehr darin, dass durch den Einsatz des Interleavers 3 Informationsworte $\mathbf{u}$ gemeinsam ein verkettetes Codewort ergeben. Hierdurch wurde die Blocklänge der einfachen Verknüpfung ohne Codespreizung von $n = 4$ auf jetzt $n = 12$ vergrößert, was – wie wir wissen – für Blockcodes ein großer Vorteil ist. Dies führt zu folgender Verbesserung.

Die als Teilcode verwendeten SPC-Codes besitzen beide die Mindestdistanz 2 und können somit 1 Fehler erkennen, allerdings keine Fehler korrigieren. Die Verknüpfung zu einem Produktcode erhöht nun nach Gl. (1.3) die Mindestdistanz auf 4, so dass 1 Fehler korrigiert und 3 Fehler erkannt werden können.

Die Fehlerkorrektur ist in Bild 1.7 noch einmal veranschaulicht. Ist das Symbol x_1 fehlerhaft, so sprechen sowohl die Zeilenprüfung der 2. Zeile als auch die Spaltenprüfung der 1. Spalte an. Beide können nur einen Fehler erkennen, ihr Schnittpunkt gibt jedoch exakt die fehlerhafte Stelle an.

x_0	x_4	x_8
x_1	x_5	x_9
x_2	x_6	x_{10}
x_3	x_7	x_{11}

Bild 1.7: Beispiel 3: (12,6,4)-Produktcode bestehend aus (3,2,2)- und (4,3,2)-SPC-Codes

Die Singleton-Schranke aus dem letzten Semester gab an, welche Mindestdistanz $d_{\min}$ ein (n,k)-Code maximal annehmen kann. Sie beträgt für das obige Beispiel

$$d_{\min} \leq n-k+1 = 12-6+1 = 7$$

und zeigt, dass dieser Produktcode auch kein MDS-Code (*Maximum Distance Separable*) ist (vgl. Hamming-Schranke: $2^{n-k} = 2^6 = 64 \geq \sum_{r=0}^{t} \binom{n}{r} = \sum_{r=0}^{1} \binom{12}{r} = 13$).

Beispiel 4:
Die beiden aus Beispiel 2 bekannten Teilcodes werden nun zu einem (28,12,4)-Produktcode kombiniert (siehe Bild 1.8). Die Coderate $R_c = 12/28 = 3/7$ verändert sich nicht gegenüber der des verketteten Codes ohne Codespreizung. Die neue minimale Gesamtdistanz beträgt jetzt $d_{\min} = 2 \cdot 3 = 6$. Damit lassen sich nun 2 Fehler korrigieren und 5 erkennen.

x_0	x_7	x_{14}	x_{21}
x_1	x_8	x_{15}	x_{22}
x_2	x_9	x_{16}	x_{23}
x_3	x_{10}	x_{17}	x_{24}
x_4	x_{11}	x_{18}	x_{25}
x_5	x_{12}	x_{19}	x_{26}
x_6	x_{13}	x_{20}	x_{27}

x_0	x_7	x_{14}	x_{21}
x_1	x_8	x_{15}	x_{22}
x_2	x_9	x_{16}	x_{23}
x_3	x_{10}	x_{17}	x_{24}
x_4	x_{11}	x_{18}	x_{25}
x_5	x_{12}	x_{19}	x_{26}
x_6	x_{13}	x_{20}	x_{27}

Bild 1.8: Beispiel 4: (28,12,6)-Produktcode bestehend aus (4,3,2)-SPC-Code und (7,4,3)-Hamming-Code

Die Singleton-Schranke lautet hier

$$d_{\min} \leq n-k+1 = 28-12+1 = 17$$

und zeigt, dass auch dieser Produktcode kein MDS-Code ist (Hamming-Schranke: $2^{14} \geq \sum_{r=0}^{2} \binom{29}{r} = 407$).

Bei diesen sehr einfachen und übersichtlichen Betrachtungen ist zu beachten, dass als Komponentencodes nicht sehr leistungsfähige Codes eingesetzt wurden. Zwar ist es das Ziel, mit einfachen Teilcodes mächtige verkettete Codes zu generieren, ein (3,2,2)-SPC-Code ist für seine Coderate von $R_c = 2/3$ allerdings nicht sehr geeignet. Wir werden später anhand einiger Beispiele sehen, dass sich Blockcodes insbesondere für hohe Coderaten eignen und gerade nicht für Raten von 1/2 oder geringer.

1.3.3 Wahl der Teilcodes

Selbstverständlich können nicht nur Blockcodes in der oben beschriebenen Art und Weise seriell miteinander verknüpft werden. Die gleiche Methode ist auch auf zwei oder mehrere Faltungscodes und auch auf Kombi-

nationen von Block- und Faltungscodes anwendbar. Prinzipiell stellt sich hier die Frage, welchen Code man als inneren und welchen man als äußeren Code wählt. Diese Frage ist nicht grundsätzlich zu beantworten, die Antwort hängt von vielen verschiedenen Faktoren ab. Beispielsweise ist es hinsichtlich des Distanzspektrums eines verketteten Codes von Vorteil, wenn der äußere Code eine möglichst große freie Distanz besitzt. Demnach müßte der stärkere von zwei Codes als äußerer Code eingesetzt werden.

Wie sich später allerdings zeigen wird, können verkettete Codes aus Gründen der Realisierbarkeit im allgemeinen nicht nach dem *Maximum Likelihood*-Kriterium decodiert werden. Vielmehr wird die in Abschnitt 1.7 beschriebene iterative Decodierung eingesetzt. Für sie ist es günstiger, den stärkeren Code innen einzusetzen, da dieser zuerst decodiert wird und somit eine bessere Ausgangsbasis für die Decodierung des äußeren Codes liefert.

Im Folgenden werden zwei serielle Codeverkettungen behandelt, die beide Faltungscodes einsetzen und die in Bild 1.6 dargestellte Struktur aufweisen. Die erste Verkettung besteht aus zwei über einen Interleaver verbundenen Faltungscodes. Der äußere Codierer kann problemlos terminiert werden, wohingegen der innere Codierer die codierte Sequenz des äußeren erhält und sein Trellisdiagramm somit offen bleibt.

Die zweite Verkettung setzt sich aus einem äußeren (terminierten) Faltungscode und einem inneren Walsh-Hadamard-Code zusammen. Der Walsh-Hadamard-Code ist ein linearer, systematischer Blockcode der Rate $\mathrm{ld}(M)/M$, d.h. er ordnet beispielsweise einem 6-Bit-Eingangswort ein 64-Bit-Ausgangswort zu. Damit ist der WH-Code ein niedrigratiger Code. Eine Besonderheit besteht darin, dass seine Ausgangsworte orthogonal zueinander sind. Aus diesem Grund werden WH-Codes auch als orthogonale Modulationsverfahren interpretiert. Die Kombination von äußerem Faltungscode und innerem WH-Code wird übrigens in einem Mobilfunksystem der zweiten Generation, dem Qualcomm-System IS-95 in den USA verwendet.

1.4 Parallele Codeverkettung (Turbo-Codes)

1.4.1 Modifikation der Produktcodes

Am Anfang dieses Kapitels wurde schon die parallele Verkettung von Codes angesprochen. Der Unterschied zur seriellen Verknüpfung besteht darin, dass hier **ausschließlich** die Informationsbit von mehreren Teilcodes codiert werden, die jeweiligen Prüfbit jedoch nicht. Dies lässt sich leicht auf die weiter vorne betrachteten Produktcodes übertragen. Wir können den in Bild 1.5 dargestellten Produktcode in eine parallele Verkettung zweier Codes überführen, indem wir die Prüfbit der Prüfbit $\mathbf{p}^+$ entfernen. Es ergibt sich dann folgende Struktur.

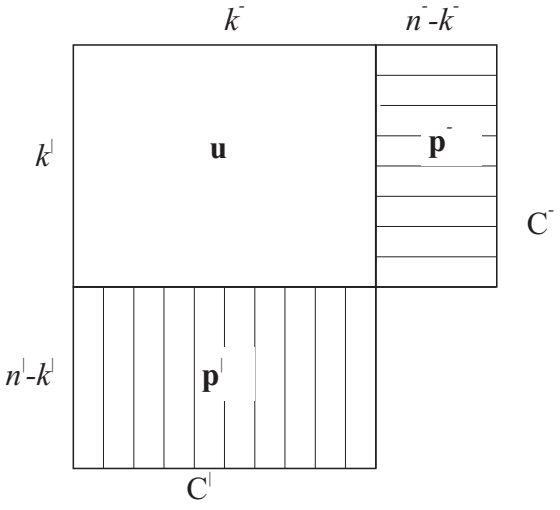


Bild 1.9: Allgemeine Struktur eines unvollständigen Produktcodes

Man spricht in diesem Zusammenhang auch von einem unvollständigen Produktcode. Er hat die Coderate

$$\begin{aligned} R_c &= \frac{k^- \cdot k^|}{k^- \cdot k^| + (n^- - k^-) \cdot k^| + (n^| - k^|) \cdot k^-} = \frac{1}{1 + (n^- - k^-)\frac{1}{k^-} + (n^| - k^|)\frac{1}{k^|}} \\ &= \frac{1}{1 + \frac{1}{R_c^-} - 1 + \frac{1}{R_c^|} - 1} = \frac{1}{\frac{1}{R_c^|} + \frac{1}{R_c^-} - 1} = \frac{R_c^| \cdot R_c^-}{R_c^- + R_c^| - R_c^- \cdot R_c^|} \end{aligned} \tag{1.4}$$

Hinsichtlich der Mindestdistanz gilt die Beziehung

$$d_{\min} = d_{\min}^- + d_{\min}^| - 1 \; . \tag{1.5}$$

Gl. (1.5) ist folgendermaßen zu verstehen. Wir nehmen wieder an, dass der Informationsteil $\mathbf{u}$ nur eine Zeile enthält, die exakt ein Element ungleich Null besitzt, alle übrigen Zeilen sollen nur Nullen enthalten. Ferner habe das zugehörige Codewort dieser Zeile das minimale Gewicht $d_{\min}^-$. Die vertikale Decodierung mit $C^|$ erzeugt somit nur ein einziges Codewort ungleich Null, da der Prüfteil nicht mehr codiert wird. Dieses Codewort hat aber ein minimales Gewicht von $d_{\min}^|$, so dass sich beide Gewichte addieren. Jetzt muss nur noch berücksichtigt werden, dass das Informationsbit, welches ungleich Null war, in beiden Codeworten vorkommt und nur einmal übertragen wird, woraus sich Gl. (1.5) ergibt.

Beispiel 5: Unvollständiger Produktcode mit (3,2,2)- und (4,3,2)-SPC-Codes
Wir wollen nun die beiden zuvor betrachteten Beispiele 3 und 4 auf unvollständige Produktcodes anwenden. Aus den beiden SPC-Codes konstruieren wir einen (11,6,3)-Produktcode (siehe Bild 1.10). Die Coderate $R_c = 6/11$ verändert sich in diesem Fall wenig gegenüber dem vollständigen Produktcode. Allerdings verringert sich die minimale Distanz gemäß Gl. (1.5) von 4 auf 3. Damit lässt sich immer noch 1 Fehler korrigieren, aber nur noch 2 Fehler erkennen.

x_0	x_4	x_8
x_1	x_5	x_9
x_2	x_6	x_{10}
x_3	x_7	

Bild 1.10: Beispiel 5: (11,6,4)-Produktcode bestehend aus (3,2,2)- und (4,3,2)-SPC-Codes

Die eingekreisten Elemente kennzeichnen die Binärstellen gleich 1 für ein mögliches Codewort mit minimalem Gewicht. Die Singleton-Schranke lautet

$$d_{\min} \leq n - k + 1 = 11 - 6 + 1 = 6.$$

Beispiel 6:
Die beiden aus Beispiel 4 bekannten Teilcodes werden nun zu einem unvollständigen (25,12,4)-Produktcode kombiniert (siehe Bild 1.11). Die Coderate $R_c = 12/25$ vergrößert sich etwas gegenüber dem vollständigen Produktcode. Die neue minimale Gesamtdistanz beträgt jetzt $d_{\min} = 2 + 3 - 1 = 4$. Damit lassen sich nun 2 Fehler erkennen und 1 Fehler korrigieren.

Die Singleton-Schranke lautet hier

$$d_{\min} \leq n - k + 1 = 25 - 12 + 1 = 14.$$

x_0	x_7	x_{14}	x_{21}
x_1	x_8	x_{15}	x_{22}
x_2	x_9	x_{16}	x_{23}
x_3	x_{10}	x_{17}	x_{24}
x_4	x_{11}	x_{18}	
x_5	x_{12}	x_{19}	
x_6	x_{13}	x_{20}	

Bild 1.11: Beispiel 6: (25,12,4)-Produktcode bestehend aus (4,3,2)-SPC-Code und (7,4,3)-Hamming-Code

1.4.2 Turbo-Codes

Die so genannten *Turbo-Codes* wurden erstmals im Jahr 1993 von Berrou, Glavieux und Thitimajshima vorgestellt. Ihre Leistungsfähigkeit versetzte die gesamte Fachwelt in helle Aufregung, war es doch erstmals gelungen, sich der Kapazitätsgrenze nach Shannon auf bis zu 0,5 dB zu nähern (gängige im letzten Semester vorgestellte Faltungscodes liegen etwa 3-5 dB von ihr entfernt). Dieser gewaltige Unterschied wird noch einmal in Bild 1.12 veranschaulicht. Es ist zu erkennen, dass mit zunehmender Einflusslänge L_c die Faltungscodes an Leistungsfähigkeit gewinnen, diese Gewinne scheinen aber immer kleiner zu werden. Damit ist abgesehen vom exponentiell steigenden Decodieraufwand mit dieser Maßnahme die Kapazitätsgrenze nach Shannon voraussichtlich nicht zu erreichen.

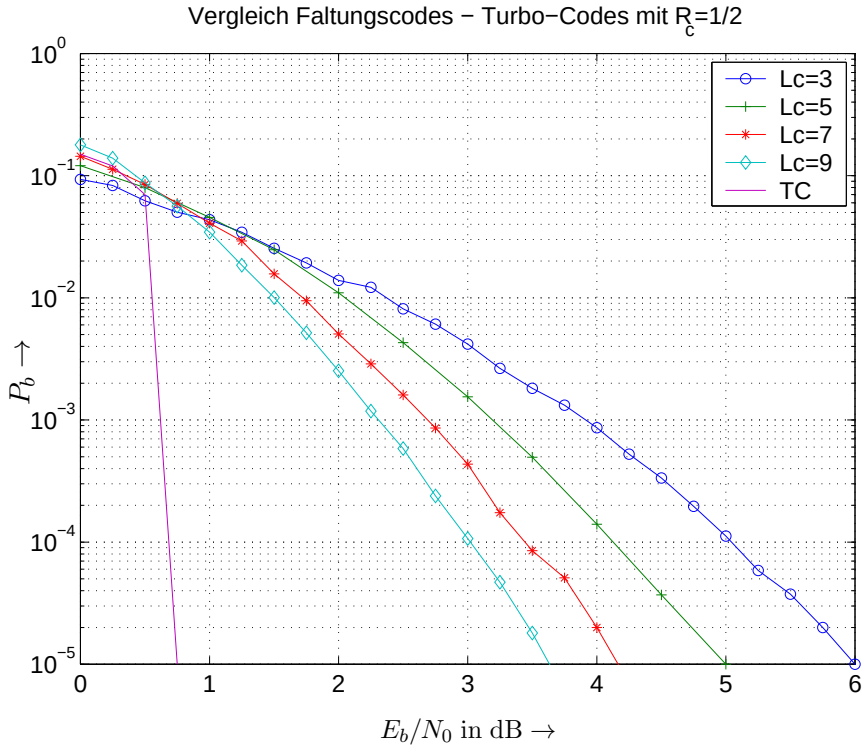


Bild 1.12: Vergleich der Bitfehlerraten für klassische Faltungscodes und Turbo-Code

Demgegenüber erreicht der hier dargestellte Turbo-Code eine Fehlerrate von $P_b = 10^{-5}$ bei 0,5 dB, was einem Gewinn gegenüber dem Faltungscode mit $L_c = 9$ von knapp 3 dB entspricht. Dies ist mit konventionellen Faltungscodes nicht zu erreichen.

Dieses Ergebnis ist ein wichtiger Meilenstein in der Codierungstheorie, da mit Hilfe der Turbo-Codes viele wichtige Eigenschaften verketteter Codes zu analysieren sind. Neben der geschickten Kombination mehrerer Codes, die Gegenstand dieses Abschnitts ist, spielt ferner der iterative Decodierprozeß eine wichtige Rolle. Seiner speziellen Struktur verdanken die Turbo-Codes auch ihren Namen. Da diese spezielle Art der Decodierung auch für die im vorigen Abschnitt behandelten Produktcodes angewendet wird, erfolgt die Beschreibung separat im nächsten Abschnitt.

Seit ihrer ersten Vorstellung beschäftigen sich weltweit unzählige Forscher mit den Turbo-Codes. Sie sorgten gewissermaßen für eine Initialzündung, den die iterative Decodierung wurde nun auf viele Bereiche auch außerhalb der Codierung angewandt (mehr dazu im nächsten Abschnitt). Einige sehr interessante und wichtige Eigenschaften der Turbo-Codes wurden erst nach ihrer Entwicklung analysiert und verstanden, so z.B. der eigentliche Grund für ihre bisher noch nie erreichte Leistungsfähigkeit. Wir wollen in dieser Vorlesung lediglich die Grundlagen der Turbo-Codes erläutern. Auch werden nicht alle Aussagen durch mathematische Beweise oder Herleitungen belegt. Vielmehr geht es darum, das prinzipielle Verständnis für diese spezielle Art der Codeverkettung zu verstehen.

Allgemeiner Aufbau von Turbo-Codes

Den generellen Aufbau eines Turbo-Codierers zeigt Bild 1.13. Sie bestehen aus einer parallelen Verkettung von i.a. q Faltungscodes, wobei die zum Einsatz kommenden Codes nicht zwingend unterschiedlich sein müssen. In den meisten Fällen werden sogar identische Teilcodes verwendet. Jeder Teilcodierer C_i erhält die gleiche Eingangsfolge $\mathbf{u}$, allerdings in jeweils unterschiedlicher Reihenfolge ($\mathbf{u}_i$). Die verschiedenen Permutationen von $\mathbf{u}$ in $\mathbf{u}_i$ werden über Interleaver Π_i realisiert, wobei der Interleaver Π_1 vor dem ersten Teilcode C_1 auch weggelassen werden kann.

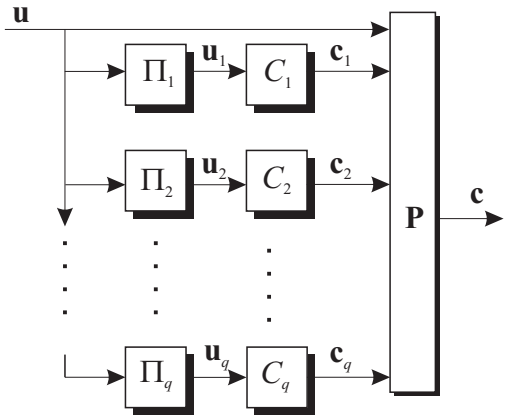


Bild 1.13: Allgemeiner Aufbau eines Turbo-Codes bestehend aus q Teilcodes

Als Teilcodes werden in der Regel systematische Codes eingesetzt, die Ausgänge der Codierer in Bild 1.13 liefern demnach nur die redundanten Prüfbit. Die Informationsbit $\mathbf{u}$ werden über den oberen Zweig direkt an den Ausgang geführt. **Turbo-Codes sind also systematische Codes!** Zur Anpassung der Gesamtcoderate können sowohl die Prüfbit als auch die Informationsbit punktiert werden. Hierzu dient die Punktierungsmatrix $\mathbf{P}$.

Die Coderate des verketteten Codes lässt sich wie folgt berechnen. Alle Codierer C_i erhalten Eingangssequenzen $\mathbf{u}_i$ gleicher Länge $k = L_\Pi$. Sie besitzen die Coderaten $R_{c,i} = k/n_i$. Da $\mathbf{u}$ nur einmal übertragen wird, fügt jeder Teilcode $n_i - k$ Prüfbit hinzu. Werden nur die Prüfbit der Teilcodierer C_i und nicht $\mathbf{u}$ punktiert, so ergibt sich insgesamt die Coderate

$$\begin{aligned} R_c &= \frac{k}{k + (n_1 - k) + (n_2 - k) + \cdots + (n_q - k)} = \frac{k}{\sum_{i=1}^{q} n_i - (q-1) \cdot k} \\ &= \frac{1}{\sum_{i=1}^{q} \frac{n_i}{k} - (q-1)} = \frac{1}{\sum_{i=1}^{q} \frac{1}{R_{c,i}} - (q-1)} \,. \end{aligned} \tag{1.6}$$

Aus Gründen der Übersichtlichkeit beschränken wir uns im Folgenden auf lediglich 2 Teilcodes. Mit dieser Einschränkung können weiterhin alle wesentlichen Aspekte erläutert werden. Eine Erweiterung auf mehr als 2 Komponentencodes ist allerdings jederzeit möglich (s. Bild 1.14). Die Gesamtcoderate für $q = 2$ lautet

$$R_c = \frac{1}{\frac{1}{R_{c,1}} + \frac{1}{R_{c,2}} - 1} = \frac{R_{c,1} \cdot R_{c,2}}{R_{c,1} + R_{c,2} - R_{c,1} \cdot R_{c,2}} \tag{1.7}$$

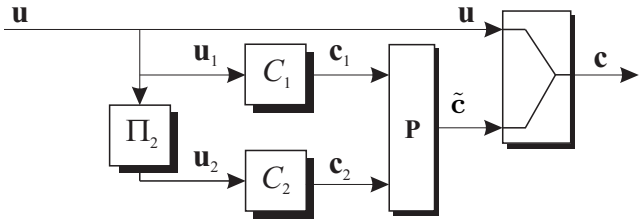


Bild 1.14: Aufbau eines Turbo-Codes bestehend aus 2 Teilcodes

1.4.3 Wahl der Teilcodes

Selbstverständlich hat die konkrete Wahl der Teilcodes auch bei der parallelen Verkettung direkte Auswirkungen auf die Leistungsfähigkeit des Gesamtcodes. Aus diesem Grund sollen hier nun einige wichtige Eigenschaften der Teilcodes beschrieben werden. Dabei beziehen sich die hier gemachten Aussagen auf die im letzten Abschnitt behandelten Turbo-Codes.

Als Teilcodes werden rekursive, systematische Faltungscodes (RSC-Code) eingesetzt. Daher erzeugen die Teilcodes C_i wie in Bild 1.15 dargestellt lediglich die Prüfbit, die Informationsbit werden am Ende gesondert hinzugefügt. Hinsichtlich der Einflusslänge L_c und der Coderate R_c der Teilcodes ist zu sagen, dass der Sinn der Codeverkettung die Synthese 'einfacher' Codes zu einem 'großen' Gesamtcode war. Aus diesem Grund wird L_c zwecks einfacher Decodierung nicht allzu groß gewählt und liegt in der Praxis im Bereich $3 \leq L_c \leq 5$. Selbstverständlich lassen sich Unterschiede in der Leistungsfähigkeit in Abhängigkeit von der Einflusslänge der Teilcodes feststellen. Die Coderate der C_i beträgt in der Praxis $R_c = 1/n$, da größere Raten immer durch Punktierung erreicht werden können. Es gibt aber auch Ansätze, Teilcodes höherer Rate einsetzen, die dann geringfügig bessere Ergebnisse erzielen.

Ein wesentlicher Grund für die herausragende Leistungsfähigkeit der Turbo-Codes ist die Rekursivität der Teilcodes. Man kann zeigen, dass Bitfehlerrate P_b und Interleavergröße L_Π über die Beziehung

$$P_b \sim L_\Pi^{1-w_{\min}} \tag{1.8}$$

zusammenhängen, wobei $w_{\min}$ das minimale Eingangsgewicht eines Teilcodes beschreibt, für das ein endliches Ausgangsgewicht erzielt wird. Bekannterweise besitzen RSC-Codes eine IIR-Struktur, so dass für ein endliches Hamminggewicht der Ausgangssequenz mindestens ein Eingangsgewicht von $w_{\min} = 2$ erforderlich ist. Daher gilt für RSC-Codes

$$P_b^{\mathrm{RSC}} \sim L_\Pi^{-1} \,, \tag{1.9}$$

d.h. die Bitfehlerrate ist proportional zum Kehrwert der Interleavergröße. Anders ausgedrückt, mit wachsendem L_Π wird der Turbo-Code immer besser!

Für NSC-Codes gilt hingegen $w_{\min} = 1$, da sie eine endliche Impulsantwort besitzen. Damit ist die Bitfehlerrate eines aus NSC-Codes zusammengesetzten Turbo-Codes entsprechend Gl. (1.8) nicht von der Interleavergröße abhängig. Somit ist klar, warum in der Praxis stets RSC-Codes zum Einsatz kommen.

Ferner kann gezeigt werden, dass die freie Distanz d_f kein geeigneter Parameter zur Optimierung der Teilcodes mehr ist. Vielmehr ist hier die sogenannte *effektive Distanz*

$$d_{\mathrm{eff}} = w_{\min} + 2 \cdot c_{\min} \tag{1.10}$$

von Vorteil. Dies ist folgendermaßen zu interpretieren.

Turbo-Codes sind systematische Codes, das Gewicht $w_{\min}$ der Infobit geht also auch in das Gesamtgewicht ein. Für $w_{\min}$ Einsen am Eingang sei das minimale Gewicht der Prüfbit eines Teilcodes $c_{\min}$. Bei 2 identischen Teilcodes beträgt dann das Mindestgewicht des Turbo-Codes für $w = 2$ gerade d_{eff}.

Hieraus folgt direkt, dass geeignete Teilcodes für $w = 2$ das Gewicht der Prüfbit maximieren müssen. Dieses Ziel kann erreicht werden, wenn das Polynom zur Rückkopplung teilerfremd (prim) ist. Dann nämlich bildet das Schieberegister eine Sequenz maximaler Länge (m-Sequenz), d.h. der minimale Abstand der beiden Einsen am Eingang wird unter der Nebenbedingung einer endlichen Ausgangssequenz maximal. Hierdurch werden mit $w = 2$ die längst möglichen Sequenzen erzeugt, die natürlich ein höheres Gewicht als kurze Sequenzen haben können. Die übrigen Generatoren der Teilcodes sind dann derart zu wählen, dass das Gewicht der Prüfbit für das spezielle teilerfremde Rückkopplungspolynom maximal wird!

Beispiel 7: Turbo-Code mit $q = 2$ identischen Teilcodes mit $L_c = 3$ und $R_c = 1/2$

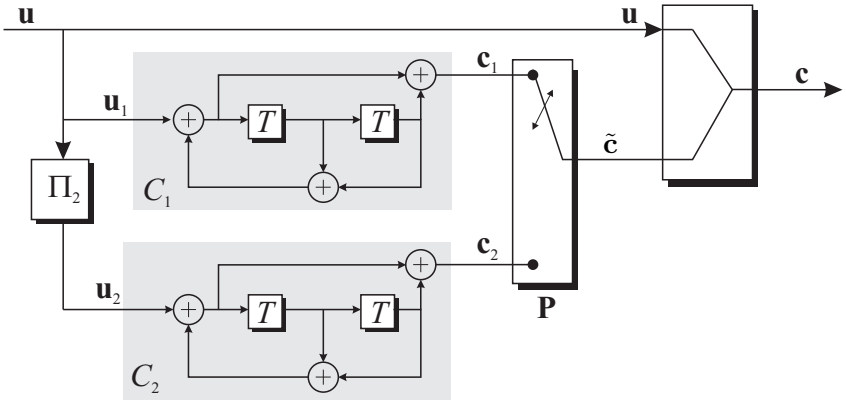


Bild 1.15: Beispiel 7: Struktur eines Turbo-Codes mit $R_c = 1/2$ bestehend aus 2 Teilcodes mit $\mathbf{g}_1 = 5_8$ und $\mathbf{g}_2 = 7_8$

- Prüfbit der Teilcodes werden alternierend übertragen $\longrightarrow \mathbf{P} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

  $\longrightarrow$ Coderate der Teilcodes ist $R_{c,1} = R_{c,2} = 2/3$, die des Gesamtcodes $R_c = 1/2$

- Einsatz rekursiver Teilcodes mit $\mathbf{g}_1 = 5$ und $\mathbf{g}_2 = 7$, wobei $\mathbf{g}_2$ zur Rückkopplung eingesetzt wird

- Polynom $g_2(D) = 1 + D + D^2$ ist teilerfremd

  $\longrightarrow$ Schieberegister bildet eine Sequenz maximaler Länge (m-Sequenz) mit $L = 2^2 - 1 = 3$

  $\longrightarrow\ d_{\text{eff}} = 2 + 2 \cdot 4 = 10$

- Polynom $g_1(D) = 1 + D^2$ ist **nicht** teilerfremd, denn es gilt $1 + D^2 = (1 + D)^2$

  $\longrightarrow$ Schieberegister bildet eine Sequenz mit der Länge $L = 2$

  $\longrightarrow\ d_{\text{eff}} = 2 + 2 \cdot 3 = 8$

$\longrightarrow$ Teilcode mit $\mathbf{g}_1$ als Rückkopplungspolynom würde schlechteren Turbo-Code bilden!

Beispiel 8: Turbo-Code mit $q = 2$ Teilcodes mit $L_c = 5$ und $R_c = 2/3$

- Teilcodes der Originalrate $R_c = 1/2$ werden zur Rate $R_c = 4/5$ punktiert

  $\longrightarrow \mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

  $\longrightarrow R_c = 2/3$

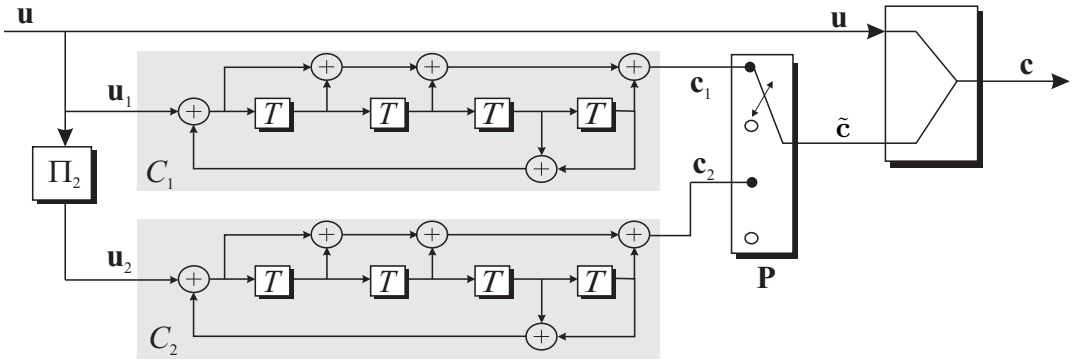


Bild 1.16: Beispiel 8: Struktur eines Turbo-Codes mit $R_c = 2/3$ bestehend aus 2 Teilcodes mit $\mathbf{g}_1 = 23_8$ und $\mathbf{g}_2 = 35_8$

- Einsatz rekursiver Teilcodes mit $\mathbf{g}_1 = 23_8$ und $\mathbf{g}_2 = 35_8$, wobei $\mathbf{g}_1$ zur Rückkopplung eingesetzt wird
- Polynom $g_1(D) = 1 + D^3 + D^4$ ist teilerfremd
  $\longrightarrow$ Schieberegister bildet eine Sequenz maximaler Länge (m-Sequenz) mit $L = 2^4 - 1 = 15$
- Polynom $g_2(D) = 1 + D + D^2 + D^4$ ist **nicht** teilerfremd, denn es gilt $D^4 + D^2 + D + 1 = (D + 1) \cdot (D^3 + D^2 + 1)$
  $\longrightarrow$ Schieberegister erzeugt periodische Sequenzen der Längen $L = 1$ und $L = 7$

$\longrightarrow$ Teilcode mit $\mathbf{g}_2$ als Rückkopplungspolynom würde schlechteren Turbo-Code bilden

Hinsichtlich der Punktierung der Teilcodes ist darauf zu achten, dass keiner der beteiligten Komponentencodes katastrophal wird (s. 1. Semester). Mit zunehmender Punktierung sind in der Regel Teilcodes größerer Einflusslänge zu wählen, damit die kleinste Distanz größer als eins bleibt und somit ein Codiergewinn erhalten bleibt.

1.5 Einfluss der Interleavers

Eine ebenso wichtige Rolle wie die Teilcodes spielt sowohl für die serielle wie auch für die parallele Verkettung der Interleaver. Er soll vermeiden, dass der Gesamtcode Ausgangsfolgen mit geringem Gewicht enthält. Für parallele Verkettungen wie den Turbo-Codes soll der Interleaver verhindern, dass an beiden Teilcodierern gleichzeitig Sequenzen $\mathbf{c}_i$ mit geringem Gewicht der Prüfbit auftreten. Dann hätte nämlich die Ausgangssequenz $\mathbf{c}$ des gesamten Turbo-Codes ein geringes Gewicht und der Code aufgrund der Linearität eine geringe kleinste Distanz. Dies muss vermieden werden, d.h. wenn die Ausgangsbit von C_1 ein geringes Gewicht haben, sollte der Interleaver Π_2 die Eingangsfolge $\mathbf{u}$ derart permutieren, dass die zu $\mathbf{u}_2$ gehörende Ausgangsfolge $\mathbf{c}_2$ ein hohes Gewicht besitzt. Dann hat der Turbo-Code insgesamt eine höhere Mindestdistanz. Das Verwürfeln der Eingangssequenz $\mathbf{u}$ führt also nicht nur zu einer Permutation der codierten Sequenz von $\mathbf{c}_1$ nach $\mathbf{c}_2$, sondern sorgt dafür, dass C_2 eine völlig andere Ausgangssequenz generiert. **Der Interleaver beeinflußt also direkt die Mindestdistanz des Turbo-Codes.**

Ähnliches gilt auch für die serielle Verkettung. Bekanntlich sorgt der äußere Code für eine Auswahl bestimmter Codeworte bzw. Codesequenzen des inneren Codes. Der Interleaver hat hier die Aufgabe, die Teilmenge des inneren Codes so auszuwählen, dass die Distanzeigenschaften des Teilcodes optimiert werden. Werden beispielsweise ein Faltungscode als äußerer Code und ein linearer Blockcode als innerer Code eingesetzt, so

sind die 'Einsen' der faltungscodierten Sequenz derart zu verschachteln, dass sie auf möglichst viele verschiedene Codeworte des Blockcodes aufgeteilt werden. Dann nämlich besitzt der Gesamtcode eine optimale freie Distanz.

Ein weiterer wichtiger Aspekt, der im nächsten Abschnitt auch noch diskutiert wird, betrifft nicht das Mindestgewicht selbst, sondern die Anzahl der Pfade mit einem bestimmten Gewicht. Durch das Interleaving wird nämlich die Anzahl der Sequenzen mit geringem Gewicht drastisch reduziert. Wie wir wissen, geht diese in die Koeffizienten c_d des Distanzspektrums und somit auch direkt in die Bitfehlerrate ein. **Der Interleaver beeinflußt also auch die Häufigkeit von Sequenzen mit bestimmten Gewicht und somit das gesamte Distanzspektrum.**

Ein letzter wichtiger Gesichtspunkt betrifft die statistische Unabhängigkeit der $\mathbf{c}_i$ an den Ausgängen der einzelnen Teilcodierer und greift somit den schon oben diskutierten Punkt wieder auf. Durch die Permutation der Informationssequenz erzeugen beide Teilcodierer unterschiedliche Ausgangssequenzen. Für den Decodierprozeß ist es sehr wichtig, dass $\mathbf{c}_1$ und $\mathbf{c}_2$ möglichst statistisch unabhängig sind (s. Abschnitt 1.7). Dies muss durch den Interleaver gewährleistet werden. Es ist leicht einzusehen, dass diese Forderung um so besser erfüllt wird, je größer der Interleaver ist, da dann die einzelnen Binärstellen weiter auseinander gespreizt werden können.

Während bei den Produktcodes vorwiegend einfache Blockinterleaver eingesetzt werden, finden bei den Turbo-Codes ab einer bestimmten Blocklänge quasi-zufällige Interleaver Verwendung. Der Grund hierfür kann wie folgt erklärt werden. Wir nehmen an, dass Teilcodes eingesetzt werden, deren rekursives Polynom zu einer m-Sequenz mit Periodendauer L führt. Zwei Einsen im Abstand L zueinander würden dann zu einer Ausgangssequenz mit endlichem Gewicht führen. Entsprechend der obigen Diskussion wäre es jetzt wünschenswert, wenn durch die Permutation Π_2 eine Ausgangssequenz generiert wird, die ein wesentlich größeres Gewicht besitzt. Dies wird bei Blockinterleavern für folgende Konstellation aber gerade nicht erreicht.

	$\vdots$		$\vdots$	
$\cdots$	1	$\cdots$	1	$\cdots$
	$\vdots$		$\vdots$	
$\cdots$	1	$\cdots$	1	$\cdots$
	$\vdots$		$\vdots$	

Sind vier Einsen quadratisch mit dem Abstand L zu den direkten Nachbarn angeordnet, so bleibt nach der Permutation mit einem Blockinterleaver diese Anordnung erhalten. Dies führt dazu, dass beide Teilcodes bei bestimmten Eingangssequenzen Codefolgen mit geringem Gewicht erzeugen. Einen Ausweg bieten *Random Interleaver*, die eine quasi zufällige Permutation durchführen und somit diese regelmäßigen Strukturen aufbrechen. Der Fall, dass beide Teilcodes niedergewichtige Sequenzen erzeugen, tritt dann wesentlich seltener auf.

Beispiel: Teilcodes aus Beispiel 7 mit $L_c = 3$, Blockinterleaver mit Länge 4x4=16
Originalinformationsfolge: $\mathbf{u} = (1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0)$
Informationsfolge nach Interleaving: $\mathbf{u}_2 = (1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0)$

1	0	1	0
0	0	0	0
1	0	1	0
0	0	0	0

Hieraus folgt, dass die obige Eingangsfolge $\mathbf{u}$ aus dem Interleaving unverändert hervorgeht und somit beide Komponentencodes C_1 und C_2 gleiche Ausgangssequenzen $\mathbf{c}_1$ bzw. $\mathbf{c}_2$ erzeugen. Zur Vermeidung solcher

Effekte müssen regelmäßige Strukturen im Interleaver vermieden werden. Da die Codespreizung nicht rein zufällig erfolgen kann, werden häufig durch systematische Rechnersuche optimierte Interleaver *gezüchtet*. Dazu geht man von einer im Prinzip willkürlich gewählten Startpermutation aus und betrachtet zunächst alle Eingangsfolgen mit einem Gewicht von $w_H(\mathbf{u}) = 2$. Die Permutationsvorschrift wird für diese Sequenzen nun so geformt, dass der obige Effekt nicht mehr auftritt. Anschließend wird die Prozedur von Eingangssequenzen mit $w_H(\mathbf{u}) = 3$ wiederholt usw. Natürlich ist der Rechenaufwand extrem hoch, weshalb diese Optimierungsstrategie aus Gründen der Realisierbarkeit nur für geringe Gewichte durchgeführt werden kann und somit auch nicht zu einem optimalen Interleaver führt. Die Verbesserungen gegenüber der Ausgangspermutation sind allerdings beachtlich.

1.6 Distanzeigenschaften und Abschätzung der Leistungsfähigkeit

Wie in den vorangegangenen Abschnitten deutlich wurde, haben Teilcodes als auch Interleaver entscheidenden Einfluss auf das Distanzspektrum und somit auch die Leistungsfähigkeit verketteter Codes. Daher soll in diesem Abschnitt das Distanzspektrum von verketteten Codes in sehr kurzer Form erläutert und damit die Frage beantwortet werden, warum verkettete Codes und im speziellen die Turbo-Codes eigentlich so gut sind. Anhand der uns schon bekannten Abschätzung

$$P_b \leq \frac{1}{2} \cdot \sum_{d=d_f}^{\infty} c_d \cdot \operatorname{erfc}\left(\sqrt{\frac{dR_c E_b}{N_0}}\right) \tag{1.11}$$

können vorab schon zwei prinzipielle Aussagen getroffen werden.

1. Die freie Distanz d_f eines Codes, d.h. die kleinste Hammingdistanz, die zwei Codesequenzen zueinander haben können, sollte so groß wie möglich sein.
2. Die Anzahl der Pfade insbesondere mit kleinen Distanzen sollte so klein wie möglich sein.

Um den immens hohen Aufwand der Berücksichtigung eines konkreten Interleavers zu umgehen, bedienen wir uns eines theoretischen Hilfsmittels, dem sogenannten *Uniform Interleaver*. Er berücksichtigt alle möglichen Permutationsvorschriften eines Interleavers der Länge L_π und repräsentiert somit einen 'mittleren' Interleaver.

Aus dem letzten Semester ist uns noch die IOWEF (*Input Output Weight Enumerating Function*)

$$A(W, D) = \sum_{w=0}^{k} \sum_{d=0}^{n} A_{w,d} \cdot W^w D^d$$

bekannt, welche auch öfters als Distanzspektrum bezeichnet wurde. Sie ist für verkettete Codes leicht zu modifizieren. So lautet beispielsweise die bedingte IOWEF für ein bestimmtes Eingangsgewicht w

$$A(w, D) = \sum_{d=0}^{n} A_{w,d} \cdot D^d \tag{1.12}$$

und für ein bestimmtes Ausgangsgewicht d

$$A(W, d) = \sum_{w=0}^{k} A_{w,d} \cdot W^w \ . \tag{1.13}$$

Wir müssen nun zwischen der seriellen und der parallelen Verkettung unterscheiden.

Parallele Verkettung

Eine wichtige Eigenschaft der parallelen Verkettung ist die Tatsache, dass alle Teilcodierer C_i jeweils eine permutierte Version $\mathbf{u}_i$ der gleichen Eingangssequenz $\mathbf{u}$ erhalten. Aus diesem Grund besitzen alle $\mathbf{u}_i$ das gleiche Eingangsgewicht w. Weiterhin erzeugen die Teilcodes nur die Prüfbit, während die Informationsbit direkt an den Ausgang gelangen. Aus diesem Grund wird statt des Gewichts d der gesamten codierten Sequenz nur das Gewicht c der Prüfbit in der IOWEF der Teilcodes berücksichtigt (es gilt $d = w + c$). Unter Verwendung des *Uniform Interleaver* lautet dann die bedingte IOWEF des parallel verketteten Codes für 2 Teilcodes

$$A^{\mathrm{par}}(w, C) = \frac{A_1(w, C) \cdot A_2(w, C)}{\binom{L_\pi}{w}} = \sum_c A^{\mathrm{par}}_{w,c} \cdot C^c \,. \tag{1.14}$$

Durch die Multiplikation der bzgl. w bedingten IOWEF's $A_1(w, D)$ und $A_2(w, D)$ der Teilcodes C_1 und C_2 wird erreicht, dass stets zwei Ausgangssequenzen mit gleichem Eingangsgewicht kombiniert werden. Außerdem bewirkt diese Multiplikation die Erfassung aller Kombinationsmöglichkeiten der Ausgangssequenzen beider Teilcodes (*Uniform Interleaver*). Die Division sorgt für die notwendige Mittelung, da $\binom{L_\pi}{w}$ die Anzahl der Permutationsmöglichkeiten von w Einsen in einem Block der Länge L_π angibt. Die Koeffizienten c_d aus Gl. (1.11) ergeben sich zu

$$c_d = \sum_{w+c=d} \frac{w}{L_\pi} \cdot A^{\mathrm{par}}_{w,c} \,. \tag{1.15}$$

Serielle Verkettung

Der wichtigste Unterschied zur parallelen Verkettung besteht bei der seriellen Verkettung darin, dass das Ausgangsgewicht des äußeren Codes gleich dem Eingangsgewicht des inneren Codes ist. Wir erhalten somit die IOWEF des Gesamtcodes

$$A^{\mathrm{ser}}(W, D) = \sum_l \frac{A_1(W, l) \cdot A_2(l, D)}{\binom{L_\pi}{l}} = \sum_w \sum_d A^{\mathrm{ser}}_{w,d} \cdot W^w D^d \,. \tag{1.16}$$

Außerdem ist zu beachten, dass der Interleaver nicht mit den Informationsbit $\mathbf{u}$ gefüllt wird, sondern mit der codierten Sequenz des äußeren Codes $\mathbf{c}_1$. Deshalb erfolgt die Mittelung über den Faktor $\binom{L_\pi}{l}$. Für die c_d gilt

$$c_d = \sum_w \frac{w}{L_\pi \cdot R_c^1} \cdot A^{ser}_{w,d} \,. \tag{1.17}$$

Bild 1.17 stellt die Koeffizienten c_d über der Distanz d für den Turbo-Code aus Beispiel 7 mit zwei identischen Komponentencodes $\mathbf{g}_1 = 7_8$ und $\mathbf{g}_2 = 5_8$ (Gesamtrate $R_c = 1/3$) und den Interleaverlängen $L_\Pi = 100$ und $L_\Pi = 400$ dar. Zum Vergleich sind die c_d auch für einen drittelratigen Faltungscode der Einflusslänge $L_c = 9$ illustriert. Wegen der Verwendung des *Uniform Interleavers* nehmen die c_d auch Werte kleiner Eins an, sie stellen Mittelwerte über alle möglichen Permutationsvorschriften dar.

- Der Turbo-Code besitzt mit $d_f = 5$ eine deutlich kleinere freie Distanz als der Faltungscode ($d_f = 18$)
- Die Koeffizienten c_d der Turbo-Codes sind aufgrund des Interleavings allerdings deutlich kleiner als die des Faltungscodes
- Der Unterschied nimmt mit wachsender Interleavergröße zu (Beachte logarithmische Darstellung!)

Bild 1.18 zeigt die mit den obigen Koeffizienten und Gl. (1.11) berechneten Bitfehlerkurven (*Union Bound*) für den AWGN- und auch den 1-Pfad Rayleigh-Fading-Kanal. Für beide Kanäle ergeben sich in etwa die gleichen Verhältnisse, weshalb im Folgenden nur auf den AWGN-Kanal näher eingegangen wird. Folgende Effekte sind zu beobachten:

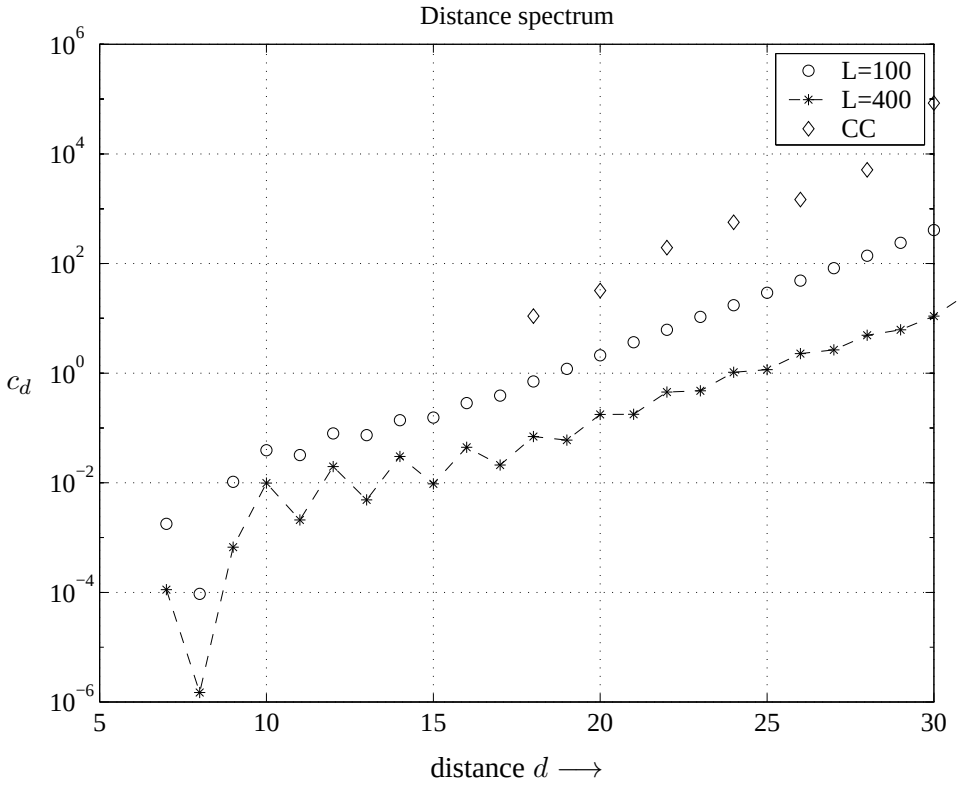


Bild 1.17: Distanzspektrum von Turbo-Code aus Beispiel 7 für verschiedene Interleavergrößen

- Für kleine Signal-Rausch-Abstände sind die Turbo-Codes wesentlich besser als die Faltungscodes (abgesehen vom Divergenzbereich der *Union Bound*)
- Die Vorteile werden mit steigender Interleavergröße größer
- Im weiteren Verlauf flachen die Bitfehlerkurven der Turbo-Codes ab
- Die des Faltungscodes werden dagegen immer steiler, so dass sich die Kurven zwischen 4 dB und 6 dB je nach Interleavergröße schneiden und die Faltungscodes besser sind

Erklärungen:

- Die freie Distanz bestimmt den asymptotischen Verlauf der Bitfehlerkurve (Verlauf für sehr große Signal-Rausch-Abstände). Daher ist es nicht verwunderlich, wenn in diesen Bereichen der Faltungscodes den Turbo-Codes überlegen ist. Es ist allerdings darauf zu achten, dass die Fehlerrate für diese Signal-Rausch-Abstände schon sehr niedrig ist und viele praktische Systeme für höhere Fehlerraten ausgelegt sind (z.B. $P_b = 10^{-3}$).
- Im Bereich kleiner Signal-Rausch-Abstände scheint die freie Distanz nur eine untergeordnete Rolle zu spielen, sonst wären hier die Turbo-Codes nicht so leistungsfähig. Vielmehr kommt es hier auf die Anzahl von Sequenzen mit bestimmtem Gewicht an.
  - Pfade mit geringem Gewicht werden hier fast ständig verwechselt; sie sollten - wenn überhaupt - nur sehr selten auftreten (kleines c_d). Eigentlich ein Vorteil für Faltungscode, der erst ab $d = 18$ Sequenzen besitzt.
  - Mit steigender Distanz wachsen die c_d bei Faltungscodes stark an; man kann zeigen, dass sie die Bitfehlerrate für kleine Signal-Rausch-Abstände dominieren (c_d wachsen schneller als die $\mathrm{erfc}()$ abfällt).
  - Für Turbo-Codes wachsen die c_d mit steigender Distanz wesentlich langsamer, wodurch Sequenzen mit kleinen Distanzen die Bitfehlerrate dominieren.

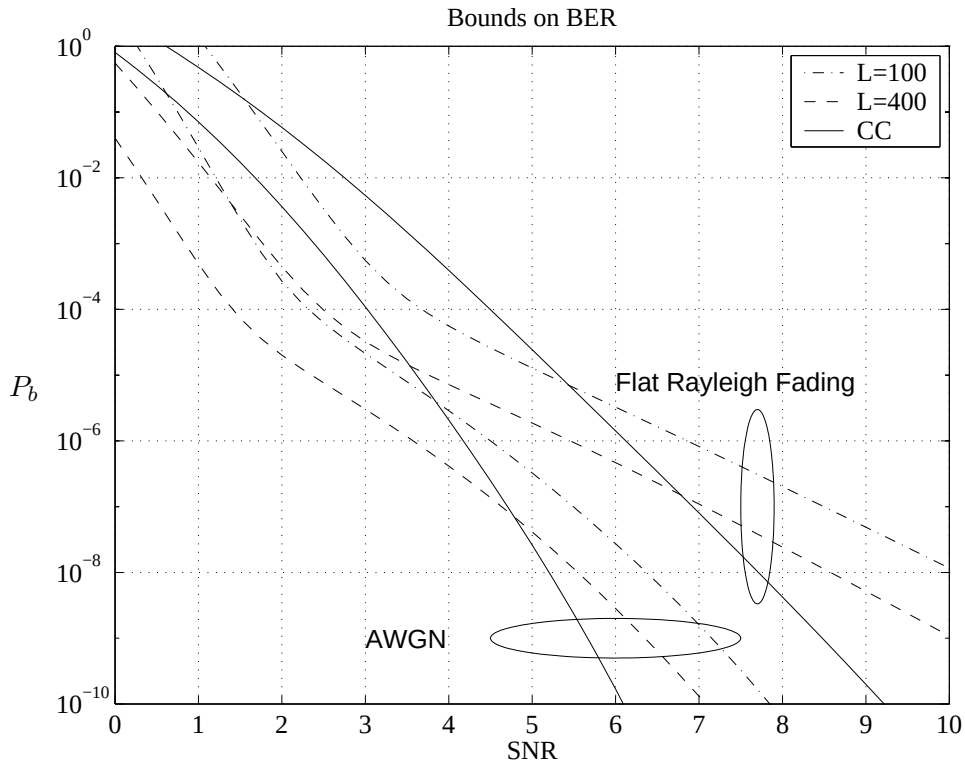


Bild 1.18: Analytische Abschätzung der Bitfehlerrate für Turbo-Codes aus Beispiel 7

Aus den obigen Erläuterungen folgt, dass zum Erreichen der Kapazitätsgrenze nach Shannon nicht nur die freie Distanz d_f, sondern auch die Anzahl der Pfade mit bestimmtem Gewicht ausschlaggebend ist. Hieraus ergeben sich andere Optimierungskonzepte als von den Faltungs- und Blockcodes bisher bekannt war.

1.7 Decodierung verketteter Codes

Eine sehr große Bedeutung kommt der Decodierung von verketteten Codes zu. Die analytische Abschätzung der Bitfehlerwahrscheinlichkeit mit Hilfe der *Union Bound* basiert stets auf der Annahme einer optimalen *Maximum Likelihood*-Decodierung. Diese ist aber gerade für verkettete Systeme i.a. nicht durchführbar. Zweck der Codeverkettung war es ja auch, relativ einfache Teilcodes zu verwenden, die sich mit geringem Aufwand decodieren lassen.

Wir erhalten somit auch im Empfänger eine Verkettung mehrerer Decodierer. Diese Konstellation führt unweigerlich zu einer suboptimalen Decodierung, die nicht mehr das *Maximum-Likelihood*-Kriterium erfüllt. Außerdem ist zu beachten, das bisher sowohl für Faltungscodes als auch für Blockcodes nur *Hard-Decision*-Algorithmen betrachtet wurden, d.h. Decodierverfahren, die als Ergebnis hart entschiedene Bit ausgeben. Während der für Faltungscodes verwendete Viterbi-Algorithmus zumindest in der Lage ist, *Soft*-Werte des Kanals zu verarbeiten, setzen die gängigen Verfahren für Blockcodes normalerweise eine *Hard-Decision* vor der Decodierung voraus. Jede harte Entscheidung vor der letzten Empfängerstufe resultiert aber automatisch in einem Informationsverlust, der nicht mehr kompensiert werden kann.

Um mit der verketteten Decodierung so dicht wie möglich an die optimale *Maximum-Likelihood*-Decodierung zu gelangen, sind also Verfahren erforderlich, die sowohl *Soft*-Werte verarbeiten als auch ausgeben können. Diese werden *Soft-In/Soft-Out*-Algorithmen genannt und spielen in der aktuellen Forschung eine wichtige Rolle. Wir wollen hier zuerst ein geeignetes Maß für die *Soft*-Information einführen und dann einige wichtige Algorithmen kennenlernen, die diese Information bereitstellen können.

1.7.1 Definition der *Soft*-Information

Im letzten Semester haben wir zwei verschiedene Decodierprinzipien kennengelernt, das MAP-Kriterium (Maximum a posteriori) und das *Maximum-Likelihood*-Kriterium. Der Unterschied besteht darin, dass bei letzterem alle Eingangssignale als gleichwahrscheinlich angenommen werden, während beim MAP-Kriterium eine unterschiedliche Verteilung des Eingangsalphabets berücksichtigt wird. Da dies für die weitere Betrachtung sehr wichtig ist, wollen wir im Folgenden stets das MAP-Kriterium verwenden, wobei die Kanalcodierung bei der Vorstellung der *Soft*-Information zunächst vernachlässigt wird. Die bisherige Notation wird hier weitgehend übernommen, so dass die logischen Bit $u\in\{0,1\}$ per BPSK-Modulation den Kanalsymbolen entsprechend

$$u = 0 \rightarrow x = +1 \tag{1.18}$$

bzw.

$$u = 1 \rightarrow x = -1 \tag{1.19}$$

zugeordnet werden. Das MAP-Kriterium lautet damit

$$P(u=0|y) = P(x=+1|y) \underset{>}{\overset{<}{}} P(u=1|y) = P(x=-1|y) \tag{1.20}$$

Mit Hilfe der Bayes-Regel können wir Gl. (1.20) umformen, wobei $P(\cdot)$ die Auftrittswahrscheinlichkeit eines Symbols beschreibt, während $p(\cdot)$ die Wahrscheinlichkeitsdichte angibt. Wir erhalten

$$\begin{aligned} & \frac{p(x=+1,y)}{p(y)} \underset{>}{\overset{<}{}} \frac{p(x=-1,y)}{p(y)} \\ \Longleftrightarrow \quad & \frac{p(x=+1,y)}{p(x=-1,y)} \underset{>}{\overset{<}{}} 1 \\ L(\hat{x}) \quad := \quad & \ln\frac{p(x=+1,y)}{p(x=-1,y)} = \underbrace{\ln\frac{p(y|x=+1)}{p(y|x=-1)}}_{L(y|x)} + \underbrace{\ln\frac{P(x=+1)}{P(x=-1)}}_{L_a(x)} \underset{>}{\overset{<}{}} 0\ . \end{aligned} \tag{1.21}$$

Die L-Werte in Gl. (1.21) werden *log-likelihood-ratios* genannt, da sie aus dem Logarithmus eines Wahrscheinlichkeitsverhältnisses hervorgehen. Sie stellen die sogenannte *Soft*-Information dar, da ihr Vorzeichen eine harte Entscheidung über das betrachtete Bit x angibt, während ihr Betrag ein Maß für die Zuverlässigkeit dieser Entscheidung darstellt.

Das logarithmische Verhältnis der Wahrscheinlichkeiten $\ln\frac{P(u=0)}{P(u=1)}$ (*log-likelihood ratio*) ist ein geeignetes Maß für die Zuverlässigkeit einer Entscheidung.

Dies heißt jedoch nicht, dass das in Gl. (1.21) dargestellte *Log-Likelihood*-Verhältnis die optimale Zuverlässigkeitsinformation ist. Sie erfüllt lediglich die rein anschaulich motivierte Vorstellung einer Zuverlässigkeitsinformation, was allerdings auch auf die Vorschrift

$$L(x) = P(x=+1) - P(x=-1)$$

zutrifft. Auch hier nimmt L positive Werte für $P(x=+1) > P(x=-1)$ und ansonsten negative Werte an. Trotzdem hat sich in der Praxis das LLR als Zuverlässigkeitsinformation durchgesetzt, weshalb wir es auch im Rahmen dieser Vorlesung verwenden wollen.

Der geschätzte *Soft*-Wert $L(\hat{x})$ in Gl. (1.21) setzt sich bei einer uncodierten Übertragung aus zwei Anteilen zusammen, dem Term $L(y|x)$, der die Übergangswahrscheinlichkeiten des Kanals enthält, und dem Term $L_a(x)$, welcher unabhängig von den Kanalausgangswerten y ist und **a-priori-Wissen** über das gesendete Symbol x repräsentiert. Ist dem Empfänger beispielsweise bekannt, mit welcher Wahrscheinlichkeit $x=+1$ bzw. $x=-1$

auftreten, kann dies gewinnbringend genutzt werden. Aufgrund der statistischen Unabhängigkeit der a-priori-Information von den empfangenen Symbolen können die LLR's einfach addiert werden (dies gilt für statistisch abhängige Signale nicht). Da x ein rein binäres Signal darstellt, gilt selbstverständlich

$$P(x=+1)+P(x=-1)=1$$

und wir erhalten den in Bild 1.19 dargestellten Verlauf von $L_a(x)$ in Abhängigkeit von $P(x=1)$.

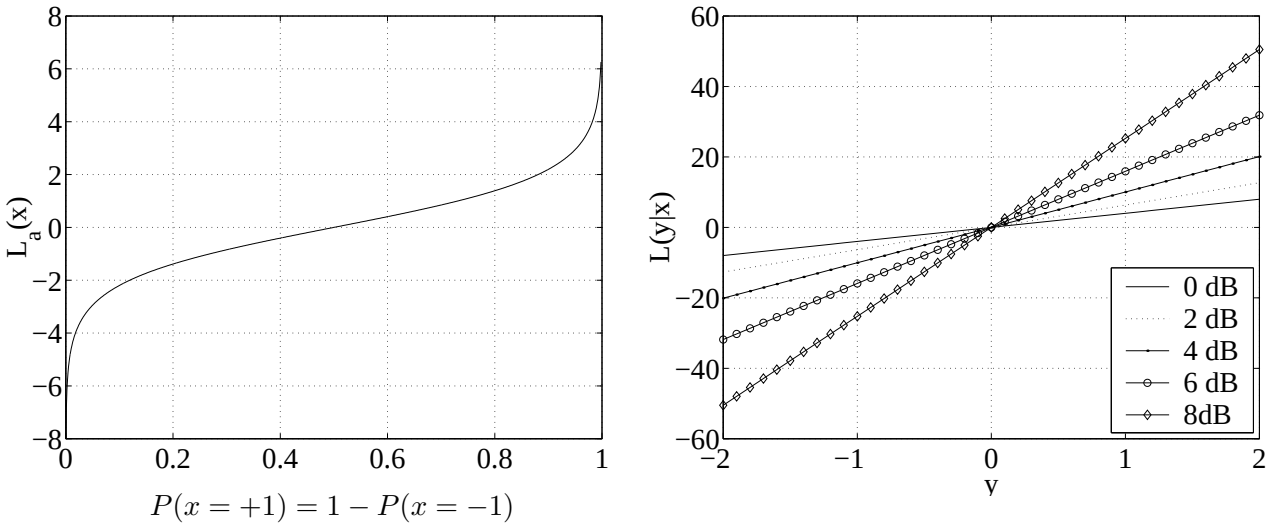


Bild 1.19: Verlauf des *Log-Likelihood*-Verhältnisses in Abhängigkeit von $P(x=+1)$ bzw. y

Im linken Bild fällt zunächst auf, dass der Verlauf symmetrisch zum Punkt (0,5;0) ist. Hier findet ein Vorzeichenwechsel statt. Für $P(x=+1)>0,5$ ist die Wahrscheinlichkeit für eine '+1' größer als für eine '-1', so dass $L_a(x)$ ab hier positive Werte annimmt. Je größer die Differenz zwischen $P(x=+1)$ und $P(x=-1)$ ist, desto größer werden die L-Werte, was belegt, dass sie ein geeignetes Maß für die Zuverlässigkeit eines Symbols sind. Sind '+1' und '-1' gleich wahrscheinlich ($P(x=+1)=0,5$), hat $L_a(x)$ den Wert Null, eine Entscheidung wäre rein zufällig. Trägt man $L_a(x)$ über $P(x=-1)$ auf, kehren sich lediglich die Vorzeichen um und man erhält einen zur Abszisse gespiegelten Verlauf.

Für den einfachen AWGN-Kanal und den 1-Pfad Rayleigh-Kanal nimmt der Term $L(y|x)$ aus Gl. (1.21) folgende konkrete Form an.

$$\begin{aligned} L(y|x) &= \ln\frac{\exp\left(-\frac{(y-\alpha\sqrt{E_s/T_s})^2}{2\sigma^2}\right)}{\exp\left(-\frac{(y+\alpha\sqrt{E_s/T_s})^2}{2\sigma^2}\right)}\,; \quad \sigma^2=\frac{N_0}{2T_s} \\ &= \frac{4\alpha\sqrt{E_s/T_s}y}{N_0/T_s} \\ &= \underbrace{4\alpha\frac{E_s}{N_0}}_{L_{ch}}y' \quad \text{mit} \quad y'=\frac{y}{\sqrt{E_s/T_s}} \end{aligned} \tag{1.22}$$

In Gl. (1.22) repräsentiert y' den auf $\sqrt{E_s/T_s}$ normierten Empfangswert y. Der Faktor α gibt den Fading-Koeffizienten an, der im Fall des AWGN-Kanals den Wert $\alpha=1$ besitzt. Der Koeffizient L_{ch} beschreibt die Zuverlässigkeit des Kanals, welche natürlich vom Signal-Rausch-Verhältnis E_s/N_0, aber auch von der Fading-Amplitude α abhängt. Für große L_{ch} ist der Kanal sehr zuverlässig, für kleine Werte besteht hingegen eine große Unsicherheit bzgl. des empfangenen Symbols. Diese Zusammenhänge illustriert der rechte Teil von Bild 1.19.

Aus Gl. (1.22) folgt, dass am Ausgang eines *matched*-Filters direkt die *Log-Likelihood*-Verhältnisse anliegen. Wenn wir in der Lage sind, eine geeignete Arithmetik für die LLR's zu finden, müssen wir diese nicht mehr

in Wahrscheinlichkeiten umrechnen, sondern können die empfangenen Werte direkt weiterverarbeiten. Eine solche Arithmetik wird von Hagenauer als L-**Algebra** bezeichnet [Hag96] und im nächsten Abschnitt noch eingehend behandelt.

Man kann natürlich aus den LLR's auch auf die Wahrscheinlichkeiten $P(x=+1)$ bzw. $P(x=-1)$ zurückrechnen. Es ergeben sich die folgenden Ausdrücke

$$P(x=+1|y) = \frac{e^{L(\hat{x})}}{1+e^{L(\hat{x})}} \tag{1.23}$$

$$P(x=-1|y) = \frac{1}{1+e^{L(\hat{x})}} \, . \tag{1.24}$$

Bezogen auf das Symbol x gilt

$$P(x=i|y) = \frac{e^{L(\hat{x})/2}}{1+e^{L(\hat{x})}} \cdot e^{iL(\hat{x})/2} \quad \text{mit} \quad i \in \{-1,+1\} \, . \tag{1.25}$$

Die Wahrscheinlichkeit für die Richtigkeit eines Empfangswertes $P(\hat{x}$ korrekt$)$ ist ebenfalls einfach zu bestimmen. Für $x=+1$ liegt eine korrekte Entscheidung vor, wenn $L(\hat{x})$ positiv ist, d.h.

$$P(\hat{x} \text{ korrekt}|x=+1) = \frac{e^{L(\hat{x})}}{1+e^{L(\hat{x})}} = \frac{e^{|L(\hat{x})|}}{1+e^{|L(\hat{x})|}} \, .$$

Für $x=-1$ muss $L(\hat{x})$ hingegen negativ sein und es folgt

$$P(\hat{x} \text{ korrekt}|x=-1) = \frac{1}{1+e^{L(\hat{x})}} = \frac{1}{1+e^{-|L(\hat{x})|}} = \frac{e^{|L(\hat{x})|}}{1+e^{|L(\hat{x})|}} \, .$$

Wir erhalten also in beiden Fällen das gleiche Ergebnis

$$P(x \text{ korrekt}) = \frac{e^{|L(\hat{x})|}}{1+e^{|L(\hat{x})|}} \, . \tag{1.26}$$

Ferner gilt für den Erwartungswert einer Datenentscheidung

$$\mathrm{E}\{\hat{x}\} = \sum_{i=\pm 1} i \cdot P(x=i) = \frac{e^{L(\hat{x})}}{1+e^{L(\hat{x})}} - \frac{1}{1+e^{L(\hat{x})}} = \tanh(L(\hat{x})/2) \, . \tag{1.27}$$

1.7.2 Rechnen mit *Log-Likelihood*-Werten (L-Algebra)

Wie schon aus dem letzten Semester bekannt ist, werden die Prüfbit eines Codes durch modulo-2-Addition bestimmter Informationsbit u_i berechnet. Damit gewinnt auch die Berechnung der L-Werte von verknüpften Zufallsvariablen an Bedeutung. Wir betrachten zunächst zwei **statistisch unabhängige** Symbole $x_1 = 1-2u_1$ und $x_2 = 1-2u_2$. Das LLR ihrer modulo-2-Summe berechnet sich nach

$$\begin{aligned}
L(u_1 \oplus u_2) &= \ln \frac{P(u_1 \oplus u_2 = 0)}{P(u_1 \oplus u_2 = 1)} \\
&= \ln \frac{P(x_1 \cdot x_2 = +1)}{P(x_1 \cdot x_2 = -1)} \\
&= \ln \frac{P(x_1=+1) \cdot P(x_2=+1) + P(x_1=-1) \cdot P(x_2=-1)}{P(x_1=+1) \cdot P(x_2=-1) + P(x_1=-1) \cdot P(x_2=+1)} \\
&= \ln \frac{P(x_1=+1)/P(x_1=-1) \cdot P(x_2=+1)/P(x_2=-1) + 1}{P(x_1=+1)/P(x_1=-1) + P(x_2=+1)/P(x_2=-1)} \\
L(x_1 \cdot x_2) &= \ln \frac{\exp\left(L(x_1)+L(x_2)\right)+1}{\exp\left(L(x_1)\right)+\exp\left(L(x_2)\right)} \, .
\end{aligned} \tag{1.28}$$

Mit Gl. (1.28) steht nun ein Funktional zur Verfügung, um das *log-likelihood*-Verhältnis der Verknüpfung zweier statistisch unabhängiger Größen zu berechnen. Die mathematischen Umrechnungen werden in der Literatur auch als L-**Algebra** bezeichnet. Gl. (1.28) lässt sich mit Hilfe der Beziehungen $\tanh(x/2) = (e^x - 1)/(e^x + 1)$ und $\ln \frac{1+x}{1-x} = 2\,\mathrm{arthanh}(x)$ in folgende Form umschreiben[1]

$$L(u_1 \oplus u_2) = \ln \frac{1 + \tanh(L(x_1)/2) \cdot \tanh(L(x_2)/2)}{1 - \tanh(L(x_1)/2) \cdot \tanh(L(x_2)/2)} \tag{1.29}$$

$$= 2\,\mathrm{arthanh}(\tanh(L(x_1)/2) \cdot \tanh(L(x_2)/2))$$

$$= 2\,\mathrm{arthanh}(\lambda_1 \cdot \lambda_2) \text{ mit } \lambda_i = \tanh(L(x_i)/2)\ . \tag{1.30}$$

Durch Gl. (1.30) ist eine einfache schaltungstechnische Realisierung möglich, wie sie in Bild 1.20 dargestellt ist. Die Eingangswerte $L(x_i)$ stellen die Ausgangssignale des *matched*-Filters dar, die dann über die tanh-Funktion nichtlinear abgebildet werden. Das Ergebnis der artanh-Funktion des Produktes stellt dann das gesuchte LLR dar.

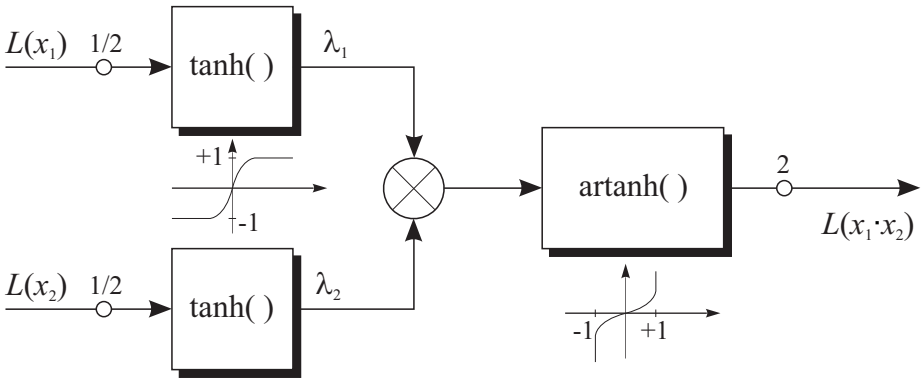


Bild 1.20: Berechnung des LLR für das Produkt zweier statistisch unabhängiger Signale

Betrachtet man den Verlauf der tanh-Funktion, so lässt sich einfach eine Approximation ableiten. Für betragsmäßig große LLR's gerät der tanh in die Sättigung, er strebt asymptotisch gegen ± 1. Dazwischen besitzt er einen annähernd linearen Verlauf mit einem Winkel von $\pi/4$ im Nullpunkt. Da das Produkt der λ_i gebildet wird, spielen Werte in der Nähe von ± 1 für den Betrag des Ergebnisses keine Rolle, dieser wird aufgrund der 'Fast-Linearität' durch das Minimum der Eingangsbeträge bestimmt. Das Vorzeichen ergibt sich hingegen aus dem Produkt der einzelnen Vorzeichen. Wir erhalten also folgende vereinfachende Approximation

$$L(u_1 \oplus u_2) \approx \mathrm{sgn}\{L(x_1)\} \cdot \mathrm{sgn}\{L(x_2)\} \cdot \min\{|L(x_1)|, |L(x_2)|\} \tag{1.31}$$

Allgemein können die Ausdrücke in den letzten Gleichungen auch für mehr als 2 Variablen angegeben werden. Ein Beweis für die Gültigkeit kann per vollständiger Induktion erfolgen (s. Übung Kanalcodierung 2). Im Folgenden sind für n statistisch unabhängige Symbole kurz die Ergebnisse aufgeführt.

$$L(u_1 \oplus \ldots \oplus u_n) = \ln \frac{\prod_{i=1}^{n}(e^{L(u_i)} + 1) + \prod_{i=1}^{n}(e^{L(u_i)} - 1)}{\prod_{i=1}^{n}(e^{L(u_i)} + 1) - \prod_{i=1}^{n}(e^{L(u_i)} - 1)} \tag{1.32}$$

$$= \ln \frac{1 + \prod_{i=1}^{n} \tanh(L(x_i)/2)}{1 - \prod_{i=1}^{n} \tanh(L(x_i)/2)} = 2\mathrm{arthanh}\left(\prod_{i=1}^{n} \tanh(L(x_i)/2)\right) \tag{1.33}$$

$$\approx \prod_{i=1}^{n} \mathrm{sgn}\{L(x_i)\} \cdot \min_i\{|L(x_i)|\} \tag{1.34}$$

[1] Die Umformung lässt sich leichter zeigen, wenn Gl. (1.28) ausgehend von Gl. (1.30) abgeleitet wird.

1.7.3 Allgemeiner Ansatz zur *Soft-Output*-Decodierung

Betrachten wir nun den Fall einer Kanalcodierung. Das Ziel besteht darin, das bekannte MAP-Kriterium zu erfüllen. Anhand einer empfangenen Sequenz $\mathbf{y}$ soll eine Aussage über die Informationsbit $\mathbf{u}$ getroffen werden. Genauer gesagt: wir wollen anhand von $\mathbf{y}$ für jedes Bit u_i eine Entscheidung und die dazugehörige Zuverlässigkeit bestimmen. Gemäß dem sogenannten *Symbol-by-Symbol*-MAP-Kriterium muss das *Log-Likelihood*-Verhältnis

$$L(\hat{u}_i) = \ln \frac{p(u_i = 0, \mathbf{y})}{p(u_i = 1, \mathbf{y})} \tag{1.35}$$

berechnet werden. Die Verbundwahrscheinlichkeiten in Gl. (1.35) sind nicht direkt zugänglich. Sie müssen vielmehr mit Hilfe einiger elementarer Umformungen abgeleitet werden. Dabei hilft die Beziehung $P(a) = \sum_i P(a, b_i)$, denn wir teilen nun den gesamten Coderaum in 2 Teilmengen $\Gamma_i^{(1)}$ und $\Gamma_i^{(0)}$ auf. $\Gamma_i^{(1)}$ enthält nur die Codeworte $\mathbf{c}$, deren i-tes Informationssymbol eine '1' ist ($u_i = 1$). Für $\Gamma_i^{(0)}$ gilt entsprechend $u_i = 0$. Wir erhalten

$$L(\hat{u}_i) = \ln \frac{\sum\limits_{\mathbf{c} \in \Gamma_i^{(0)}} p(\mathbf{c}, \mathbf{y})}{\sum\limits_{\mathbf{c} \in \Gamma_i^{(1)}} p(\mathbf{c}, \mathbf{y})} \tag{1.36}$$

$$= \ln \frac{\sum\limits_{\mathbf{c} \in \Gamma_i^{(0)}} p(\mathbf{y}|\mathbf{c}) \cdot P(\mathbf{c})}{\sum\limits_{\mathbf{c} \in \Gamma_i^{(1)}} p(\mathbf{y}|\mathbf{c}) \cdot P(\mathbf{c})} \tag{1.37}$$

Für den AWGN-Kanal sind alle aufeinander folgenden Rauschwerte statistisch unabhängig voneinander, was für die Binärstellen eines Codewortes natürlich nicht gilt (die Codierung fügt statistische Abhängigkeiten ein). Die bedingte Wahrscheinlichkeitsdichte $p(\mathbf{y}|\mathbf{c})$ stellt jedoch die Wahrscheinlichkeitsdichte unter der Hypothese $\mathbf{c}$ dar ($\mathbf{c}$ ist keine Zufallsgröße mehr, sondern eine feste Annahme). Deswegen können die Elemente y_i von $\mathbf{y}$ als statistisch unabhängig angesehen werden (die statistischen Abhängigkeiten sind in der Hypothese $\mathbf{c}$ enthalten). Die bedingte Wahrscheinlichkeitsdichte der Vektoren kann dann in das Produkt der Dichten der Vektorelemente überführt werden.

$$L(\hat{u}_i) = \ln \frac{\sum\limits_{\mathbf{c} \in \Gamma_i^{(0)}} \prod\limits_{j=0}^{n-1} p(y_j|c_j) \cdot P(\mathbf{c})}{\sum\limits_{\mathbf{c} \in \Gamma_i^{(1)}} \prod\limits_{j=0}^{n-1} p(y_j|c_j) \cdot P(\mathbf{c})} \tag{1.38}$$

Außerdem ist ein Codewort $\mathbf{c}$ eindeutig durch seine Informationsbit $\mathbf{u}$ bestimmt, wodurch für die Auftrittswahrscheinlichkeit $P(\mathbf{c}) = P(\mathbf{u})$ gilt. Die Informationsbit u_i sind allerdings statistisch unabhängig voneinander (**nicht die Codebit** c_i), so dass $P(\mathbf{c}) = \prod_{i=0}^{k-1} P(u_i)$ gilt.

$$L(\hat{u}_i) = \ln \frac{\sum\limits_{\mathbf{c} \in \Gamma_i^{(0)}} \prod\limits_{j=0}^{n-1} p(y_j|c_j) \cdot \prod\limits_{j=0}^{k-1} P(u_j)}{\sum\limits_{\mathbf{c} \in \Gamma_i^{(1)}} \prod\limits_{j=0}^{n-1} p(y_j|c_j) \cdot \prod\limits_{j=0}^{k-1} P(u_j)} \tag{1.39}$$

Für systematische Codierer gilt $u_i = c_i$, weshalb der mit der i-ten Stelle korrespondierende Term $p(y_i|c_i)$ in Zähler und Nenner jeweils konstant ist. Daher kann er zusammen mit $P(u_i)$ aus dem Produkt und der Summe

herausgezogen werden

$$L(\hat{u}_i) = \underbrace{\ln \frac{p(y_i|u_i=0)}{p(y_i|u_i=1)}}_{L_{ch}\cdot y_i} + \underbrace{\ln \frac{P(u_i=0)}{P(u_i=1)}}_{L_a(u_i)} + \underbrace{\ln \frac{\sum\limits_{\mathbf{c}\in\Gamma_i^{(0)}} \prod\limits_{\substack{j=0\\ j\neq i}}^{n-1} p(y_j|c_j) \cdot \prod\limits_{\substack{j=0\\ j\neq i}}^{k-1} P(u_j)}{\sum\limits_{\mathbf{c}\in\Gamma_i^{(1)}} \prod\limits_{\substack{j=0\\ j\neq i}}^{n-1} p(y_j|c_j) \cdot \prod\limits_{\substack{j=0\\ j\neq i}}^{k-1} P(u_j)}}_{L_e(\hat{u}_i)} . \tag{1.40}$$

Aus Gl. (1.40) geht hervor, dass sich $L(\hat{u}_i)$ bei einer systematischen Codierung aus drei Anteilen zusammensetzt:

$$\boxed{L(\hat{u}_i) = L_{ch} y_i + L_a(u_i) + L_e(\hat{u}_i) \ ,} \tag{1.41}$$

dem *log-likelihood*-Verhältnis des direkt empfangenen Symbols y_i - also der systematischen Komponente -, der a-priori-Information $L_a(u_i)$ (schon vom uncodierten Fall bekannt) und einem Anteil $L_e(\hat{u}_i)$, der nicht von u_i bzw. y_i selbst abhängt. Dieser wird vielmehr 'von außen' aus allen durch die Codierung mit u_i verknüpften Bit berechnet und daher **extrinsische Information** genannt. Erfahren die einzelnen Codesymbole während der Übertragung statistisch unabhängige Störungen (wie z.B. beim AWGN-Kanal), so ist $L_e(u_i)$ statistisch unabhängig von $L_a(u_i)$ und $L_{ch}y_i$ und liefert daher einen Beitrag zum Decodierergebnis, der die Zuverlässigkeit der Entscheidung erhöhen kann. **Diese Zerlegung kann nicht für nicht-systematische Codes vorgenommen werden!**

Wir können die extrinsische Information auch noch kompakter schreiben und erhalten

$$L_e(\hat{u}_i) = \ln \frac{\sum\limits_{\mathbf{c}\in\Gamma_i^{(0)}} \prod\limits_{\substack{j=0\\ j\neq i}}^{n-1} p(y_j;c_j)}{\sum\limits_{\mathbf{c}\in\Gamma_i^{(1)}} \prod\limits_{\substack{j=0\\ j\neq i}}^{n-1} p(y_j;c_j)} \quad \text{mit} \quad p(y_j;c_j) = \begin{cases} p(y_j|c_j)\cdot P(u_j) & \text{für } 0 \le j < k \\ p(y_j|c_j) & \text{für } k \le j < n \ . \end{cases} \tag{1.42}$$

Unter Zuhilfenahme der LLR's lautet die extrinsische Information

$$L_e(\hat{u}_i) = \ln \frac{\sum\limits_{\mathbf{c}\in\Gamma_i^{(0)}} \prod\limits_{\substack{j=0\\ j\neq i}}^{n-1} \exp\left[-L(c_j;y_j)\cdot c_j\right]}{\sum\limits_{\mathbf{c}\in\Gamma_i^{(1)}} \prod\limits_{\substack{j=0\\ j\neq i}}^{n-1} \exp\left[-L(c_j;y_j))\cdot c_j\right]} \quad \text{mit} \quad L(c_l;y_l) = \begin{cases} L_{ch}y_l + L_a(u_l) & \text{für} \quad 0 \le l < k \\ L_{ch}y_l & \text{für} \quad k \le l < n \ . \end{cases} \tag{1.43}$$

Mit den Gleichungen (1.41) und (1.42) kann also eine Decodierung durchgeführt werden, die neben der reinen Hard-Decision auch ein Maß für die Zuverlässigkeit der Decodierentscheidung liefert. Ein Problem besteht allerdings darin, dass zur Berechnung der extrinsischen Information über alle Codeworte $\mathbf{c}$ des Coderaums Γ summiert werden muss. Man kann sich leicht vorstellen, dass diese direkte Realisierung der *Soft-Output*-Decodierung sehr aufwendig ist, insbesondere dann, wenn Codes mit sehr großen Alphabeten zum Einsatz kommen. Für einen (7,4,3)-Hamming-Code mit seinen $2^4 = 16$ Codeworten dürfte eine Berechnung von Gl. (1.42) kein Problem sein, für einen (255,247,3)-Hamming-Code gibt es allerdings $2^{247} = 2,3 \cdot 10^{74}$ Codeworte, so dass Gl. (1.42) selbst auf Hochleistungsrechnern nicht mehr zu berechnen ist.

Decodierung über den *dualen Code*

Ist die Anzahl der Prüfbit relativ klein, gibt es die Möglichkeit, die Decodierung über den *dualen Code* durchzuführen. Dieser ist aus dem letzten Semester noch bekannt und stellt den zum Originalcode C orthogonalen Code $C^{\perp}$ dar. Es soll an dieser Stelle nicht auf Einzelheiten dieser Decodierungsmöglichkeit eingegangen, die Decodiervorschrift jedoch kurz erläutert werden. Der Vorteil besteht darin, dass jetzt über alle Codewörter aus

$C^{\perp}$ summiert werden muss. Für den oben erwähnten (255,247,3)-Hamming-Code wären das nur $2^8 = 256$, was zu einer deutlichen Vereinfachung führt. Das Decodierergebnis lautet [Off96]

$$L(\hat{u}_i) = L_{ch} y_i + L_a(u_i) + \ln \frac{\sum\limits_{\mathbf{c}' \in \Gamma^{\perp}} \prod\limits_{\substack{l=0 \\ l \neq i}}^{n-1} [\tanh(L(y_l; c_l)/2)]^{c'_l}}{\sum\limits_{\mathbf{c}' \in \Gamma^{\perp}} (-1)^{c'_i} \prod\limits_{\substack{l=0 \\ l \neq i}}^{n-1} [\tanh(L(y_l; c_l)/2)]^{c'_l}} . \tag{1.44}$$

In Gl. (1.44) stellt c'_l stellt das l-te Codebit des Codewortes $\mathbf{c}'$ aus $C^{\perp}$ dar. Es wird also in Zähler und Nenner über alle 2^{n-k} Codeworte des dualen Codes $C^{\perp}$ summiert, wobei die einzelnen Summanden sich aus dem Produkt der tanh-Terme der einzelnen Codebit zusammensetzen. Eine weitere Möglichkeit zur Reduktion des Rechenaufwandes besteht in der Ausnutzung der Markov-Eigenschaft von Codes, die in der Darstellung eines Trellisdiagramms resultiert.

Der Ansatz über den dualen Code führt allerdings auch nur dann zum Ziel, wenn die Anzahl der Redundanzbit gering ist. Sonst ergibt sich das gleiche Problem eines nicht zu bewältigenden Rechenaufwandes wie schon bei der direkten Decodierung des Originalcodes. Für diese Fälle gibt es eine Reihe von suboptimalen Algorithmen, die allerdings noch Gegenstand der aktuellen Forschung sind. Unter anderem wird versucht, leistungsfähige Codes zu finden, die einen geringen Decodieraufwand erfordern (*Low Density Parity Check Codes*). Im Rahmen dieser Vorlesung soll auf diese Verfahren nicht weiter eingegangen werden.

***Soft-Output*-Decodierung am Beispiel eines (4,3,2)-SPC-Codes**
Wir wollen nun am Beispiel eines einfachen (4,3,2)-SPC-Codes die L-Algebra genauer betrachten (s. Bild 1.21). Gegeben ist ein Informationswort

$$\mathbf{u} = (1\ 0\ 1) ,$$

das zunächst codiert ($\mathbf{c}$) und BPSK-moduliert wird

$$\mathbf{c} = (1\ 0\ 1\ 0) \quad \longrightarrow \quad \mathbf{x} = (-1\ +1\ -1\ +1) .$$

Während der Übertragung z.B. über einen AWGN-Kanal erfährt $\mathbf{x}$ Störungen, so dass der Vektor

$$\mathbf{y} = (-0,8\ +1,1\ \underline{+0,3}\ +0,4)$$

empfangen wird. Das Vorzeichen der Stelle x_2 wurde also verfälscht, zusätzlich sind die Beträge verrauscht. Wir gehen zunächst davon aus, dass keine a-priori-Information zur Verfügung steht, d.h. $L_a(u_i) = 0,\ i = 0, 1, 2$.

Bei einer einfachen Hard-Decision-Decodierung können wir den obigen Einzelfehler lediglich erkennen, allerdings nicht korrigieren.

Für eine Soft-Decision-Decodierung berechnen wir zunächst die LLR's für jedes y_i entsprechend Gl. (1.22) und erhalten für einen angenommenen Signal-Rausch-Abstand von 2 dB ($L_{ch} = 6,34$)

$$L_{ch}\mathbf{y} = (-5,1\ +7,0\ +1,9\ +2,5) .$$

Anschließend werden die statistischen Bindungen der einzelnen Symbole, die durch die Codierung eingebracht wurden, ausgenutzt und die extrinsische Information aus Gl. (1.41) bestimmt. Dazu stellen wir folgende Betrachtung an.

Bei einer Decodierung über den originalen Code nach Gl. (1.43) wären für den einfachen Parity-Check-Code $2^3 = 8$ Codeworte zu betrachten. Demgegenüber besteht der orthogonale Coderaum $C^{\perp}$ nur aus $2^{n-k} = 2$ Elementen, die Berechnung von $L_e(\hat{u}_i)$ über den dualen Code erfordert also nur die Berücksichtigung einer Prüfgleichung (das Nullwort wird nicht benötigt).

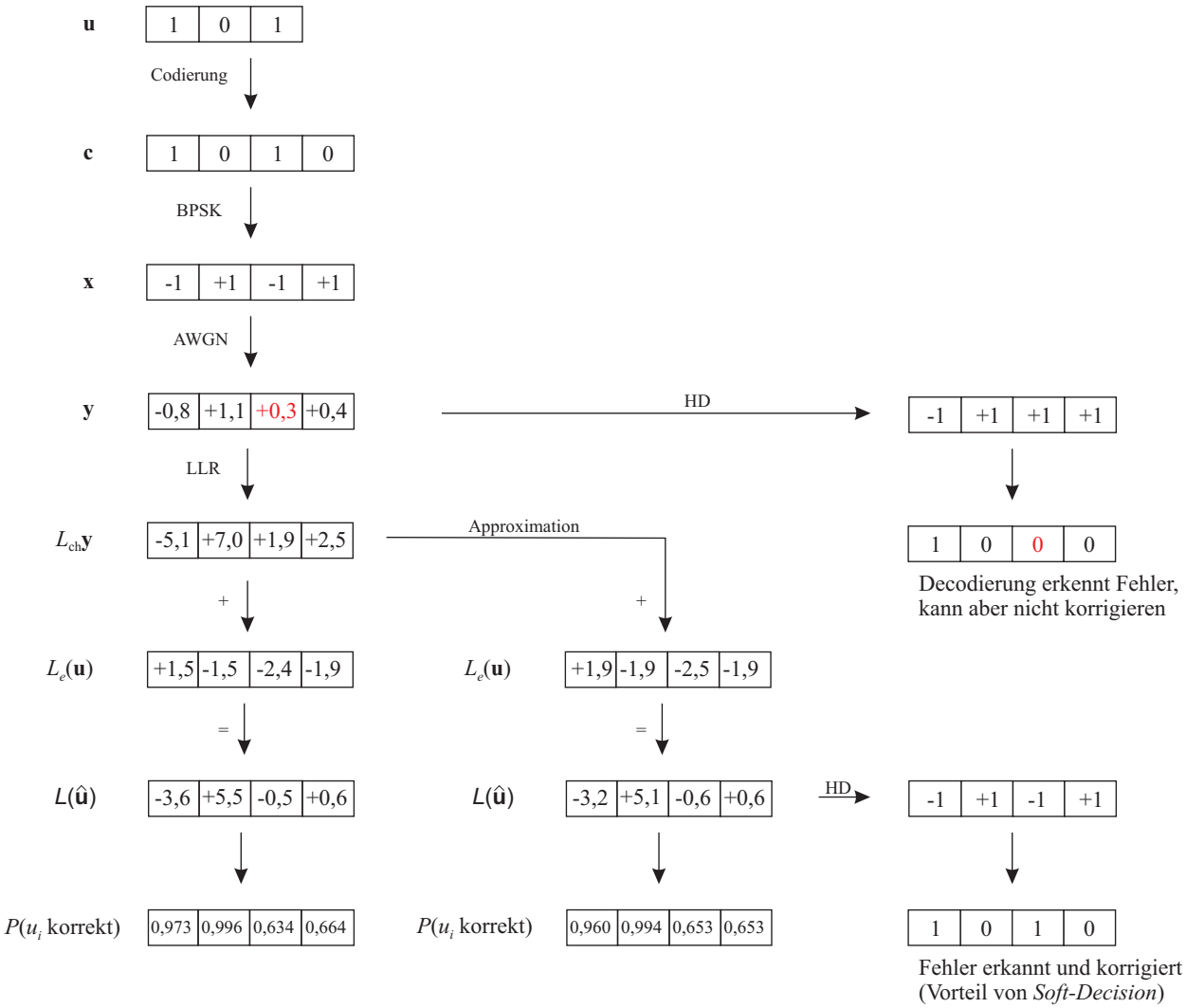


Bild 1.21: Darstellung der Soft-In/Soft-Out-Decodierung für einfachen SPC-Code

Da Zähler und Nenner in Gl. (1.44) für das Nullwort Eins ergeben, ist für SPC-Codes nur noch das Codewort $\mathbf{c}' = (1\,1\,\ldots\,1) \in \Gamma^{\perp}$ zu betrachten. Gl. (1.44) reduziert sich damit auf

$$L(\hat{u}_i) = L_{ch} y_i + \ln \frac{1 + \prod\limits_{\substack{l=0 \\ l \neq i}}^{n-1} [\tanh(L(y_l; c_l)/2)]}{1 - \prod\limits_{\substack{l=0 \\ l \neq i}}^{n-1} [\tanh(L(y_l; c_l)/2)]} \; . \tag{1.45}$$

Mit Hilfe der Beziehung $\ln \frac{1+x}{1-x} = 2\,\mathrm{arthanh}(x)$ erhalten wir schließlich (vgl. auch Gl. (1.33))

$$L(\hat{u}_i) = L_{ch} y_i + 2\,\mathrm{arthanh}\left(\prod_{\substack{l=0 \\ l \neq i}}^{n-1} [\tanh(L(y_l; c_l)/2)] \right) \tag{1.46}$$

$$\approx L_{ch} y_i + \prod_{\substack{l=0 \\ l \neq i}}^{n-1} \mathrm{sgn}\{L(y_l; c_l)\} \cdot \min_{l \neq i} \{|L(y_l; c_l)|\} \; . \tag{1.47}$$

Gl. (1.46) ist folgendermaßen zu interpretieren. Für alle $\mathbf{c} \in \Gamma$ gilt bekanntlich $\mathbf{c} \cdot \mathbf{c}' = 0$, d.h. die Quersumme der Binärstellen von $\mathbf{c}$ modulo 2 ergibt immer Null. Damit lässt sich aber jedes Bit c_i aus der modulo-2-Summe

aller übrigen Binärstellen berechnen $c_i = (\sum_{j \neq i} c_j) \mod 2$. Soll diese Berechnung nun mit Soft-Werten durchgeführt werden, ist Gl. (1.33) anzuwenden, wir erhalten mit dem zweiten Term in Gl. (1.46) also die extrinsische Information über das Bit u_i.

Für u_0 lautet die extrinsische Information

$$L_e(\hat{u}_0) = L_e(\hat{c}_0) = L(c_1 \oplus c_2 \oplus c_3)$$

und kann nach den Gleichungen (1.32), (1.33) oder (1.34) berechnet werden. Die gleichen Berechnungen werden für die übrigen Informationsbit u_2 und u_3 ausgeführt, so dass wir mit der Näherung aus Gl. (1.34)

$$L_e(\hat{\mathbf{u}}) = (+1,9 \ -1,9 \ -2,5 \ -1,9)$$

erhalten.

Es ist zu erkennen, dass durch die Verfälschung des Vorzeichens von x_2 für alle Stellen bis auf $i = 2$ die extrinsische Information eine anderes Ergebnis (Vorzeichen) liefert als die direkt empfangenen Werte. Damit kann sie das Decodierergebnis auch für diese Symbole verfälschen, vorausgesetzt, ihr Betrag ist größer als der der direkt empfangenen Werte. Auf der anderen Seite besteht die Möglichkeit, das verfälsche Vorzeichen von c_2 mit Hilfe von $L_e(\hat{u}_2)$ zu korrigieren.

Da $L_e(\hat{u}_i)$ statistisch unabhängig von den direkten Komponenten $L_{ch}y_i$ ist, können beide LLR's addiert werden. Ob die extrinsische Information dann das Vorzeichen des in unserem Beispiel verfälschten Bits c_2 korrigieren kann und die übrigen (korrekten) nicht mehr verändert, hängt von den Beträgen der Summanden ab. Das Ergebnis lautet

$$L(\hat{\mathbf{u}}) = L_{ch} \cdot \mathbf{y} + L_e(\hat{\mathbf{u}}) = (-3,2 \ +5,1 \ -0,6 \ +0,6) \quad \longrightarrow \quad \mathbf{c} = (1\,0\,1\,0) \ .$$

Wir erkennen, dass die Vorzeichen zwar korrekt sind, der Betrag für $\hat{u}_2$ ist allerdings deutlich kleiner als bei den übrigen Stellen und die Entscheidung somit unsicherer. Dies wird klarer, wenn man sich die zu den LLR's zugehörigen Wahrscheinlichkeiten anschaut (s. Bild 1.21). Die Wahrscheinlichkeit für die Richtigkeit der Vorzeichen beträgt dort nämlich nur $63,5\%$, während sie für die restlichen Stellen nahe Eins liegt.

Das bessere Ergebnis gegenüber der Hard-Decision-Decodierung besteht nun nicht in der Verwendung von *Soft-Output*-Algorithmen, sondern in der Ausnutzung der weichen Ausgangswerte des Kanals (*Soft-Decision*-Decodierung). Die weichen Ausgangswerte des Decodierers kommen erst zu Geltung, wenn weitere nachfolgende Stufen der Signalverarbeitung gewinnbringend Gebrauch von ihnen machen können. Dies wird der nächste Abschnitt verdeutlichen.

1.7.4 BCJR-Algorithmus am Beispiel von Faltungscodes

Der BCJR-Algorithmus wurde erstmals im Jahr 1972 von Bahl, Cocke, Jelinek und Raviv vorgestellt. Er stellt eine allgemeine Vorschrift zur *Symbol-by-Symbol*-MAP-Decodierung dar, die nicht nur zur Decodierung, sondern auch zur Entzerrung gedächtnisbehafteter Kanäle eingesetzt werden kann. Der Vorteil des BCJR-Algorithmus gegenüber der direkten Realisierung von Gl. (1.43) besteht in der effizienten Ausnutzung der Markov-Eigenschaft des Kanals bzw. des Codierers. Er setzt wie schon der Viterbi-Algorithmus die Darstellung des Systems (in unserem Fall des Kanalcodes) durch ein Trellisdiagramm voraus. Es gibt zwar auch eine Trellisrepräsentation für Blockcodes, diese müßte hier jedoch erst neu eingeführt werden. Aus diesem Grund greifen wir auf binäre $1/n$-ratige Faltungscodes zurück, für die das Trellisdiagramm schon bekannt ist (s. Kanalcodierung I).

Ausgangspunkt zur *Symbol-by-Symbol*-MAP-Decodierung ist wiederum Gl. (1.35)

$$L(\hat{u}_i) = \ln \frac{p(u_i = 0, \mathbf{y})}{p(u_i = 1, \mathbf{y})} \ .$$

Wir betrachten nun den Ausschnitt eines Trellisdiagramms für einen rekursiven, systematischen Faltungscodes der Einflusslänge $L_c = 3$. Aus Gründen der Anschaulichkeit wird in den Bildern stets dieser einfache RSC-Code mit nur 4 Zuständen verwendet, die mathematische Herleitung jedoch allgemein gehalten.

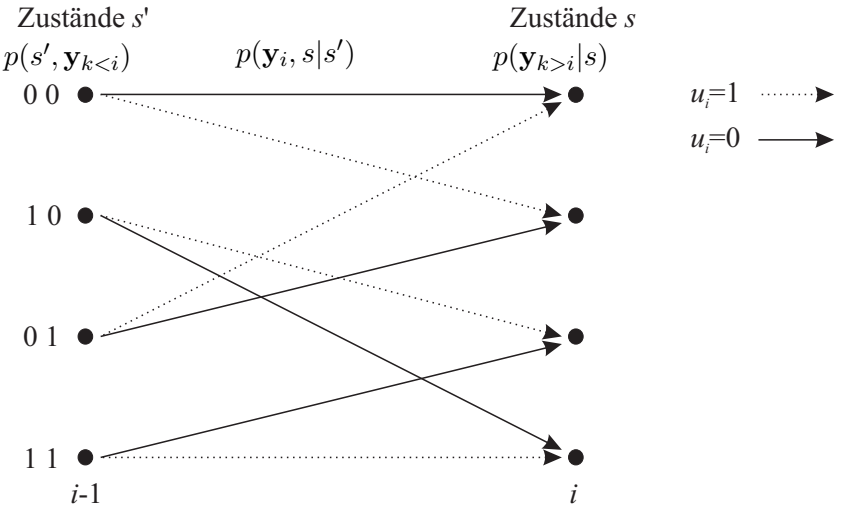


Bild 1.22: Ausschnitt eines Trellisdiagramms zur Erläuterung des BCJR-Algorithmus

Vom Zeitpunkt $i-1$ zum Zeitpunkt i findet durch das Informationsbit u_i ein Übergang von Zustand s' zu Zustand s statt. Alle möglichen Zustandsübergänge $s' \rightarrow s$ lassen sich in zwei Klassen aufteilen, in solche, die mit $u_i = 0$ korrespondieren und die übrigen, die mit $u_i = 1$ verknüpft sind. Bezeichnen wir ein Zustandspaar mit (s', s), so können wir obige Gleichung in

$$L(\hat{u}_i) = \ln \frac{\sum\limits_{(s',s),u_i=0} p(s', s, \mathbf{y})}{\sum\limits_{(s',s),u_i=1} p(s', s, \mathbf{y})} \tag{1.48}$$

umschreiben. Nun wird der Empfangsvektor $\mathbf{y}$ in drei Anteile zerlegt, einen Anteil $\mathbf{y}_{k<i}$, der alle empfangenen Symbole vor dem betrachteten Zeitpunkt i enthält, den aktuellen Empfangsvektor $\mathbf{y}_i = (y_{i,0}, \cdots, y_{i,n-1})$ bestehend aus n Empfangswerten und einen Anteil $\mathbf{y}_{k>i}$, der alle nach $\mathbf{y}_i$ empfangenen Symbole beinhaltet.

$$\begin{aligned} L(\hat{u}_i) &= \ln \frac{\sum\limits_{(s',s),u_i=0} p(s', s, \mathbf{y}_{k<i}, \mathbf{y}_i, \mathbf{y}_{k>i})}{\sum\limits_{(s',s),u_i=1} p(s', s, \mathbf{y}_{k<i}, \mathbf{y}_i, \mathbf{y}_{k>i})} \\ &= \ln \frac{\sum\limits_{(s',s),u_i=0} p(\mathbf{y}_{k>i}|s', s, \mathbf{y}_{k<i}, \mathbf{y}_i) \cdot p(s', s, \mathbf{y}_{k<i}, \mathbf{y}_i)}{\sum\limits_{(s',s),u_i=1} p(\mathbf{y}_{k>i}|s', s, \mathbf{y}_{k<i}, \mathbf{y}_i) \cdot p(s', s, \mathbf{y}_{k<i}, \mathbf{y}_i)} \end{aligned} \tag{1.49}$$

Für den ersten Faktor in Zähler und Nenner gilt: Ist der Zustand s zum Zeitpunkt i bekannt, so sind alle übrigen Größen (s', $\mathbf{y}_i$, $\mathbf{y}_{k<i}$) für den Vektor $\mathbf{y}_{k>i}$ ohne Belang, so dass

$$\beta_i(s) := p(\mathbf{y}_{k>i}|s', s, \mathbf{y}_{k<i}, \mathbf{y}_i) = p(\mathbf{y}_{k>i}|s) \tag{1.50}$$

gilt. $\beta_i(s)$ gibt die Wahrscheinlichkeitsdichte für die Teilfolge $\mathbf{y}_{k>i}$ an, wenn zum Zeitpunkt i der Zustand s im

Trellisdiagramm angenommen wird. Damit lautet Gl. (1.49) jetzt

$$\begin{aligned} L(\hat{u}_i) &= \ln \frac{\sum\limits_{(s',s),u_i=0} \beta_i(s) \cdot p(s, \mathbf{y}_i | s', \mathbf{y}_{k<i}) \cdot p(s', \mathbf{y}_{k<i})}{\sum\limits_{(s',s),u_i=1} \beta_i(s) \cdot p(s, \mathbf{y}_i | s', \mathbf{y}_{k<i}) \cdot p(s', \mathbf{y}_{k<i})} \\ &= \ln \frac{\sum\limits_{(s',s),u_i=0} \beta_i(s) \cdot p(s, \mathbf{y}_i | s') \cdot p(s', \mathbf{y}_{k<i})}{\sum\limits_{(s',s),u_i=1} \beta_i(s) \cdot p(s, \mathbf{y}_i | s') \cdot p(s', \mathbf{y}_{k<i})} \end{aligned} \tag{1.51}$$

Der mittlere Faktor in Zähler und Nenner von Gl. (1.51) wurde in der gleichen Art und Weise abgeleitet wie schon β. Er beinhaltet die Übergangswahrscheinlichkeit des Kanals, nämlich die Wahrscheinlichkeit des Auftretens von y_i bei bekanntem Zustandswechsel von s' nach s. Es gilt

$$\begin{aligned} \gamma_i(s', s) &:= p(s, \mathbf{y}_i | s') = p(s', s, \mathbf{y}_i)/P(s') = p(\mathbf{y}_i | s', s) \cdot \frac{P(s', s)}{P(s')} \\ &= p(\mathbf{y}_i | s', s) \cdot P(s | s') \; . \end{aligned} \tag{1.52}$$

Der Faktor $p(\mathbf{y}_i|s', s)$ beschreibt die Übergangswahrscheinlichkeit des Kanals, während $P(s|s')$ ein a-priori-Wissen verkörpert. Da $u_i = 0$ und $u_i = 1$ in der Regel gleichwahrscheinlich sind, treten auch die Übergänge $s' \rightarrow s$ (sie korrespondieren mit $u = 0$ bzw. $u = 1$) mit der gleichen Häufigkeit auf, es gilt $P(s|s') = 1/2$. Besitzt der Decodierer allerdings a-priori-Wissen über ein Informationsbit u_i, so ist dies gleichbedeutend mit einem Wissen über die Wahrscheinlichkeit der Übergänge $s' \rightarrow s$ zum Zeitpunkt i. Dieses Wissen kann über die Größe γ entsprechend Gl. (1.52) in den Decodierprozeß eingebracht werden.

Abschließend bleibt noch die Verbundwahrscheinlichkeitsdichte $p(s', \mathbf{y}_{k<i})$ zu erwähnen, die mit

$$\alpha_{i-1}(s') := p(s', \mathbf{y}_{k<i}) \tag{1.53}$$

abgekürzt wird. Es ergibt sich also der kompakte Ausdruck

$$L(\hat{u}_i) = \ln \frac{\sum\limits_{(s',s),u_i=0} \alpha_{i-1}(s') \cdot \gamma_i(s', s) \cdot \beta_i(s)}{\sum\limits_{(s',s),u_i=1} \alpha_{i-1}(s') \cdot \gamma_i(s', s) \cdot \beta_i(s)} \; . \tag{1.54}$$

Die Summanden aus Gl. (1.48) lassen sich also in drei Anteile gliedern, wobei $\alpha_{i-1}(s')$ die Zeitpunkte $k < i$, $\gamma_i(s', s)$ den aktuellen Zeitpunkt und $\beta_i(s)$ alle nachfolgenden Zeitpunkte $k > i$ abdeckt. α und β lassen sich rekursiv berechnen, wie folgende Rechnung zeigt.

$$\begin{aligned} \alpha_i(s) &= p(s, \mathbf{y}_{k<i+1}) = \sum_{s'} p(s', s, \mathbf{y}_{k<i+1}) = \sum_{s'} p(s', s, \mathbf{y}_{k<i}, \mathbf{y}_i) \\ &= \sum_{s'} p(s, \mathbf{y}_i | s', \mathbf{y}_{k<i}) \cdot p(s', \mathbf{y}_{k<i}) = \sum_{s'} p(s, \mathbf{y}_i | s') \cdot p(s', \mathbf{y}_{k<i}) \\ &= \sum_{s'} \gamma_i(s', s) \cdot \alpha_{i-1}(s') \end{aligned} \tag{1.55}$$

Gl. (1.55) illustriert, dass sich die α durch eine Vorwärtsrekursion sukzessive berechnen lassen, da $\alpha_i(s)$ zum aktuellen Zeitpunkt i von den vorangegangenen $\alpha_{i-1}(s')$ des Zeitpunktes $i-1$ abhängt. Für β gilt entsprechend

$$\begin{aligned}\beta_{i-1}(s') &= p(\mathbf{y}_{k>i-1}|s') = \sum_s p(s', s, \mathbf{y}_i, \mathbf{y}_{k>i})/P(s') \\ &= \sum_s p(\mathbf{y}_{k>i}|s', s, \mathbf{y}_i) \cdot \frac{p(s', s, \mathbf{y}_i)}{P(s')} = \sum_s p(\mathbf{y}_{k>i}|s) \cdot p(\mathbf{y}_i, s|s') \\ &= \sum_s \gamma_i(s', s) \cdot \beta_i(s) \,, \end{aligned} \tag{1.56}$$

d.h. β wird mit einer Rückwärtsrekursion bestimmt (s. Bild 1.23).

Initialisierung
Zur Realisierung des *Symbol-by-Symbol*-MAP-Algorithmus ist das Trellisdiagramm demnach zweimal abzuarbeiten. Zunächst findet eine Vorwärtsrekursion statt, die zeitgleich mit dem Empfang der Kanalsymbole ablaufen kann. Da wir als Startzustand zum Zeitpunkt $i = 0$ stets den Nullzustand annehmen, sind die Werte α wie folgt zu initialisieren:

$$\alpha_0(s') = \begin{cases} 1 & s' = 0 \\ 0 & s' \neq 0 \,. \end{cases} \tag{1.57}$$

Anschließend ist die Rückwärtsrekursion zur Bestimmung der β durchzuführen. Hier müssen 2 Fälle unterschieden werden. Ist der Endzustand zum Zeitpunkt N dem Decodierer bekannt (z.B. der Nullzustand durch Anfügen von *Tailbit*), lautet die Initialisierung

$$\beta_N(s) = \begin{cases} 1 & s = 0 \\ 0 & s \neq 0 \,. \end{cases} \tag{1.58}$$

Ist der Endzustand unbekannt, kann nach

$$\beta_N(s) = \alpha_N(s) \tag{1.59}$$

oder

$$\beta_N(s) = 2^{-m} \tag{1.60}$$

initialisiert werden, wobei m das Gedächtnis des Faltungscodes repräsentiert. Die prinzipielle Abarbeitung des Trellisdiagramms illustriert noch einmal Bild 1.23.

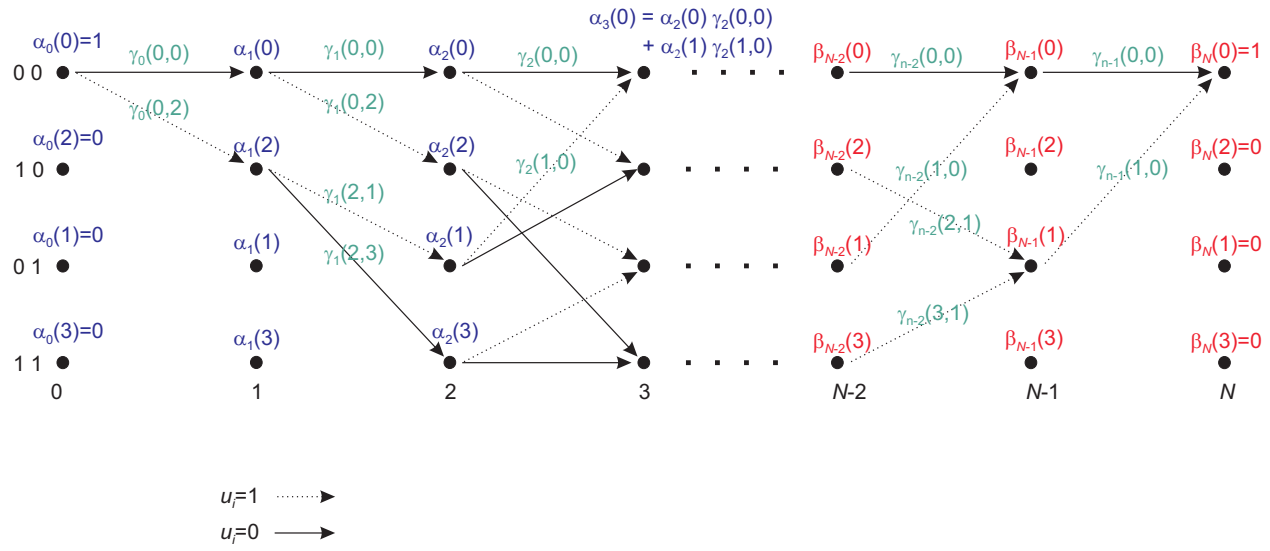


Bild 1.23: Abarbeitung des Trellisdiagramms durch BCJR-Algorithmus

1.7.5 Iterative ('Turbo')-Decodierung am Beispiel zweier parallel verketteter (5,4,2)-SPC-Codes

Die in den vorangegangenen Abschnitten vorgestellten *Soft-Output*-Algorithmen bieten nur dann Vorteile, wenn nachgeschaltete Module von der Zuverlässigkeitsinformation Gebrauch machen können. In den hier behandelten verketteten Codiersystemen erfolgt die Decodierung in der Regel durch serielle Aneinanderreihung der Teildecodierer, und zwar unabhängig davon, ob eine serielle oder eine parallele Codeverkettung vorliegt. Die prinzipielle Vorgehensweise soll zunächst anhand eines konkreten Produktcodes erläutert und dann in einer allgemeineren Form präsentiert werden.

Gegeben sei die folgende (4x4)-Matrix $\mathbf{u}$ bestehend aus 16 Informationsbit

$$\mathbf{u} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} .$$

Sie wird sowohl horizontal als auch vertikal mit einem (5,4,2)-SPC-Code codiert, was zusammen mit der BPSK-Modulation zur Codematrix

$$\mathbf{x} = \begin{pmatrix} -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & \end{pmatrix}$$

führt. Es ist zu beachten, dass die rechte untere Ecke der Matrix unbesetzt bleibt, es handelt sich also um eine parallele Verkettung der beiden SPC-Codes (unvollständiger Produktcode). Durch die auf dem Kanal einwirkenden Störungen erhalten wir die Empfangsmatrix $\mathbf{y}$, deren Elemente entsprechend Gl. (1.22) in *log-likelihood*-Verhältnisse umgerechnet werden. ($E_b/N_0 = 2dB$).

$$\mathbf{y} = \left(\begin{array}{rrrr|r} 0,1 & 1,2 & 0,2 & -0,5 & 1,0 \\ 0,8 & -0,7 & 0,6 & -0,1 & -1,5 \\ -1,2 & 0,5 & -0,9 & 1,2 & 0,2 \\ 0,2 & -0,2 & 1,3 & -1,5 & -2,0 \\ \hline 0,3 & -0,9 & 1,2 & -1,1 & \end{array}\right) \Longrightarrow L_{ch}\mathbf{y} = \left(\begin{array}{rrrr|r} 0,6 & 7,6 & 1,3 & -3,2 & 6,3 \\ 5,1 & -4,4 & 3,8 & -0,6 & -9,5 \\ -7,6 & 3,2 & -5,7 & 7,6 & 1,3 \\ 1,3 & -1,3 & 8,2 & -9,5 & -12,7 \\ \hline 1,9 & -5,7 & 7,6 & -7,0 & \end{array}\right)$$

Diese Matrix bildet nun den Ausgangspunkt für einen **iterativen Decodierprozeß**, der im Folgenden beschrieben wird und mit der vertikalen Decodierung beginnt. Wir wissen aus Gl. (1.41)

$$L(\hat{u}_i) = L_{ch}y_i + L_a(u_i) + L_e(\hat{u}_i) ,$$

dass sich das LLR bei systematischen Codes aus drei Anteilen zusammensetzt, einer a-priori-Information $L_a(u_i)$, einem systematischen direkten Anteil $L_{ch}y_i$ und der extrinsischen Information $L_e(\hat{u}_i)$. Während $L_a(u_i)$ uns jetzt noch nicht zur Verfügung steht, stellt die Matrix $L_{ch}\mathbf{y}$ den direkten, schon berechneten Anteil dar und $L_e(\hat{u}_i)$ soll mit Hilfe von Gl. (1.34) berechnet werden. Wir erhalten nach der vertikalen Decodierung $D^{|}$ zu jedem Informationsbit eine extrinsische Information $L^{|}_{e,1}(\hat{u}_i)$, z.B.

$$\begin{aligned} L^{|}_{e,1}(\hat{u}_0) &= = \min(5,1;\ 7,6;\ 1,3;\ 1,9) \cdot (+1)(-1)(+1)(+1) = -1,3 \\ L^{|}_{e,1}(\hat{u}_1) &= = \min(0,6;\ 7,6;\ 1,3;\ 1,9) \cdot (+1)(-1)(+1)(+1) = -0,6 \\ &\vdots \end{aligned}$$

die zur direkten Komponente hinzu addiert wird und dann $L^{|}_1(\hat{u}_i)$ ergibt ($\mathbf{y}_s$ stellt den Anteil der Informationsbit von $\mathbf{y}$ dar).

1. Iteration, vertikale Decodierung

$L_{ch}\mathbf{y}$

0,6	7,6	1,3	-3,2	6,3
5,1	-4,4	3,8	-0,6	-9,5
-7,6	3,2	-5,7	7,6	1,3
1,3	-1,3	8,2	-9,5	-12,7
1,9	-5,7	7,6	-7,0	

$\Downarrow$

$L_{e,1}^{|}(\hat{\mathbf{u}})$

-1,3	-1,3	-3,8	-0,6	
-0,6	1,3	-1,3	-3,2	
0,6	-1,3	1,3	0,6	
-0,6	3,2	-1,3	-0,6	

$\Longrightarrow$

$L_1^{|}(\hat{\mathbf{u}}) = L_{ch}\mathbf{y}_s + L_{e,1}^{|}(\hat{\mathbf{u}})$

-0,7	6,3	-2,5	-3,8	
4,5	-3,1	2,5	-3,8	
-7,0	1,9	-4,4	8,2	
0,7	1,9	6,9	-10,1	

Damit ist die erste vertikale Decodierung abgeschlossen. Die berechnete extrinsische Information $L_{e,1}^{|}(\hat{u}_i)$ stellt das Wissen über ein bestimmtes Informationsbit u_i eines Codewortes aus der Sicht aller übrigen Bit $u_{j\neq i}$ dieses Codewortes dar, nicht aber aus der Sicht von u_i selbst. Da bei der horizontalen Decodierung D^- nun andere Binärstellen zu einem Codewort zusammengefaßt werden als bei der vertikalen, ist die extrinsische Information statistisch unabhängig von den Symbolen c_i eines horizontalen Codewortes und ein a-priori-Wissen für D^-.

$$L_{a,1}^{-}(\hat{\mathbf{u}}) = L_{e,1}^{|}(\hat{\mathbf{u}}) \tag{1.61}$$

Für die horizontale Decodierung wird daher $L_{a,1}^{-}(\hat{\mathbf{u}})$ zu jedem empfangenen LLR $L_{ch}\mathbf{y}$ hinzu addiert (beide Informationen beschreiben das gleiche Bit und sind statistisch unabhängig voneinander)

$$\begin{aligned} L_{ch}y_i + L_{e,1}^{|}(\hat{u}_i) \qquad &, \quad i = 0 \ldots 15 \\ L_{ch}y_0 + L_{e,1}^{|}(\hat{u}_0) &= 0,6 + (-1,3) = -0,7 \\ L_{ch}y_1 + L_{e,1}^{|}(\hat{u}_1) &= 5,1 + (-0,6) = 4,5 \end{aligned}$$

und wir erhalten folgende Eingangsmatrix für die erste horizontale Decodierung. Mit diesen 'neuen' Eingangswerten startet jetzt die horizontale Decodierung D^-.

1. Iteration, horizontale Decodierung

$L_{ch}\mathbf{y} + L_{a,1}^{-}(\mathbf{u})$

-0,7	6,3	-2,5	-3,8	6,3
4,5	-3,1	2,5	-3,8	-9,5
-7,0	1,9	-4,4	8,2	1,3
0,7	1,9	6,9	-10,1	-12,7
1,9	-5,7	7,6	-7,0	

$\Longrightarrow$

$\mathbf{L}_{e,1}^{-}(\hat{\mathbf{u}})$

2,5	-0,7	0,7	0,7	
-2,5	2,5	-3,1	2,5	
-1,3	1,3	-1,3	1,3	
1,9	0,7	0,7	-0,7	

$\Downarrow$

$\mathbf{L}_1^{-}(\hat{\mathbf{u}}) = L_{ch}\mathbf{y}_s + L_{a,1}^{-}(\mathbf{u}) + L_{e,1}^{-}(\hat{\mathbf{u}})$

1,8	5,6	-1,8	-3,1	
2,0	-0,6	-0,6	-1,3	
-8,3	3,2	-5,7	9,5	
2,6	2,6	7,6	-10,8	

1. Iteration, geschätzte Daten

Nach der ersten Iteration erhalten wir folgende geschätzten Daten

$$\hat{\mathbf{u}}_1 = \begin{array}{|c|c|c|c|}\hline 0&0&1&1\\\hline 0&1&1&1\\\hline 1&0&1&0\\\hline 0&0&0&1\\\hline\end{array}$$

Ein Vergleich mit der gesendeten Informationsmatrix $\mathbf{u}$ zeigt, dass zwei Fehler aufgetreten sind, in der ersten Zeile das erste und das dritte Bit. Allerdings ist das Decodierergebnis noch verbesserungsfähig, da der $D^{|}$ bisher noch nicht die extrinsische Information $L_{e,1}^{-}(\hat{\mathbf{u}})$ des horizontalen Decodierers als a-priori-Information ausnutzen konnte. Folglich führen wir nun eine zweite Iteration durch, in der $D^{|}$ wieder beginnt, diesmal allerdings mit Unterstützung des a-priori Wissens $L_{a,2}^{|}(\mathbf{u}) = L_{e,1}^{-}(\hat{\mathbf{u}})$. Die Eingangsmatrix lautet

$$L_{ch}\mathbf{y} + L_{a,2}^{|}(\mathbf{u}) = \begin{array}{|r|r|r|r|r|}\hline 3{,}1&6{,}9&2{,}1&-2{,}5&6{,}3\\\hline 2{,}6&-1{,}9&0{,}7&1{,}9&-9{,}5\\\hline -8{,}9&4{,}5&-7{,}0&8{,}9&1{,}3\\\hline 3{,}2&-0{,}6&8{,}9&-10{,}2&-12{,}7\\\hline 1{,}9&-5{,}7&7{,}6&-7{,}0&\\\hline\end{array}.$$

Es ist zu beachten, dass die extrinsische Information $L_{e,1}^{|}(\hat{\mathbf{u}})$ von $D^{|}$ der ersten Iteration nicht wieder als a-priori-Wissen an den Eingang von $D^{|}$ zurückgeführt wird. Dies ist sehr wichtig, da sonst eine starke Korrelation mit den übrigen Eingangswerten vorhanden wäre. Es wird also stets **nur der für einen bestimmten Decodierer neue Informationsanteil (die extrinsische Information des anderen Decodierers) weitergeleitet**. Die vertikale Decodierung liefert in der zweiten Iteration jetzt folgendes Ergebnis.

2. Iteration, vertikaler Decoder

$L_{ch}\mathbf{y} + L_{a,2}^{|}(\mathbf{u})$

3,1	6,9	2,1	-2,5	6,3
2,6	-1,9	0,7	1,9	-9,5
-8,9	4,5	-7,0	8,9	1,3
3,2	-0,6	8,9	-10,2	-17,2
1,9	-5,7	7,6	-7,0	

$\Downarrow$

$\mathbf{L}_{e,2}^{|}(\hat{\mathbf{u}})$

-1,9	-0,6	-0,7	1,9	
-1,9	0,6	-2,1	-2,5	
1,9	-0,6	0,7	-1,9	
-1,9	1,9	-0,7	1,9	

$\Longrightarrow$

$\mathbf{L}_1^{|}(\hat{\mathbf{u}}) = L_{ch}\mathbf{y} + L_{a,2}^{|}(\mathbf{y}) + L_{e,2}^{|}(\hat{\mathbf{u}})$

1,2	6,3	1,4	0,6	
0,7	-1,3	-1,4	-0,6	
-7,0	3,9	-6,3	7,0	
1,3	1,3	8,2	-8,3	

2. Iteration, horizontaler Decodierer

Die Eingangswerte des horizontalen Decodierers und das Decodierergebnis lauten in der zweiten Iteration:

$L_{ch}\mathbf{y} + L_{a,2}^{-}(\mathbf{u})$

-1,3	7,0	0,6	-1,3	6,3
3,2	-3,8	1,7	-3,1	-9,5
-5,7	2,6	-5,0	5,7	1,3
-0,6	0,6	7,5	-7,6	-12,7
1,9	-5,7	7,6	-7,0	

$\Longrightarrow$

$\mathbf{L}_{e,2}^{-}(\hat{\mathbf{u}})$

-0,6	0,6	1,3	-0,6	
-1,7	1,7	-3,1	1,7	
-1,3	1,3	-1,3	1,3	
0,6	-0,6	-0,6	0,6	

$\Downarrow$

$\mathbf{L}_{2}^{-}(\hat{\mathbf{u}}) = L_{ch}\mathbf{y} + L_{a,2}^{-}(\mathbf{y}) + L_{e,2}^{-}(\hat{\mathbf{u}})$

-1,9	7,6	2,1	-1,9	
1,5	-2,1	-1,4	-1,4	
-7,0	3,9	-6,3	7,0	
-0,0	0,0	6,9	-7,0	

2. Iteration, geschätzte Daten

$$\hat{\mathbf{u}}_2 = \begin{array}{|c|c|c|c|}\hline 1 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline x & x & 0 & 1 \\ \hline \end{array}$$

Wie zu sehen ist, sind die zuvor falsch decodierten Binärstellen der ersten Zeile korrigiert worden, dafür treten nun zwei Unsicherheiten in der letzten Zeile auf (1. und 2. Spalte). Die LLR's gleich Null lassen nur eine rein zufällige Entscheidung zu, so dass die Fehlerwahrscheinlichkeit bei diesen beiden Binärstellen bei 50% liegt. In der Hoffnung auf eine Verbesserung dieser beiden Fehler führen wir die Decodierung noch ein drittes Mal durch.

3. Iteration, vertikaler Decodierer

$L_{ch}\mathbf{y} + L_{a,3}^{|}(\mathbf{u})$

0,0	8,2	2,6	-3,8	6,3
3,4	-2,7	0,7	1,1	-9,5
-8,9	4,5	-7,0	8,9	1,3
1,9	-1,9	8,2	-8,9	-12,7
1,9	-5,7	7,6	-7,0	

$\Downarrow$

$\mathbf{L}_{e,3}^{|}(\hat{\mathbf{u}})$

-1,9	-1,9	-0,7	1,1	
0,0	1,9	-2,6	-3,8	
0,0	-1,9	0,7	-1,1	
0,0	2,7	-0,7	1,1	

$\Longrightarrow$

$\mathbf{L}_{3}^{|}(\hat{\mathbf{u}}) = L_{ch}\mathbf{y} + L_{a,3}^{|}(\mathbf{y}) + L_{e,3}^{|}(\hat{\mathbf{u}})$

-1,9	6,3	1,9	-2,7	
3,4	-0,8	-1,9	-2,7	
-8,9	2,6	-6,3	7,8	
1,9	0,8	7,5	-7,8	

3. Iteration, horizontaler Decodierer

$L_{ch}\mathbf{y} + L_{a,3}^{-}(\mathbf{u})$

-1,3	5,7	0,6	-2,1	6,3
5,1	-2,5	1,2	-4,4	-9,5
-7,6	1,3	-5,0	7,5	1,3
1,3	1,4	7,5	-8,4	-12,7
1,9	-5,7	7,6	-7,0	

$\Longrightarrow$

$\mathbf{L}_{e,3}^{-}(\hat{\mathbf{u}})$

-0,6	0,6	1,3	-0,6	
-1,2	1,2	-2,5	1,2	
-1,3	1,3	-1,3	1,3	
1,4	1,3	1,3	-1,3	

$\Downarrow$

$\mathbf{L}_{3}^{-}(\hat{\mathbf{u}}) = L_{ch}\mathbf{y} + L_{a,3}^{-}(\mathbf{y}) + L_{e,3}^{-}(\hat{\mathbf{u}})$

-1,9	6,3	1,9	-2,7	
3,9	-1,3	-1,3	-3,2	
-8,9	2,6	-6,3	8,8	
2,7	2,7	8,8	-9,7	

3. Iteration, geschätzte Daten

$$\hat{\mathbf{u}}_3 = \begin{array}{|c|c|c|c|}\hline 1 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Nach der dritten Iteration sind nun alle Informationsbit richtig geschätzt worden. Dieses Beispiel soll verdeutlichen, dass bei verketteten Codes die extrinsische Information einer Binärstelle als a-priori-Information für die übrigen Codes eingesetzt werden kann. So konnte in dem obigen Beispiel mit jeder Iteration eine Verbesserung und am Ende sogar eine fehlerfreie Decodierung erzielt werden. Selbstverständlich führt die iterative Decodierung nicht immer zu einer fehlerfreien Lösung.

1.7.6 Generelles Konzept der iterativen Decodierung

Wie im vorangegangenen Abschnitt deutlich wurde, besteht die Grundidee der iterativen Decodierung darin, bei jedem Teildecodierprozeß eine extrinsische Information zu extrahieren, die den nachfolgenden Decodierern als a-priori-Information dienen kann. Es darf jeweils nur diese extrinsische Information weitergereicht werden, da sonst statistische Abhängigkeiten zu den übrigen Eingangswerten eines Decodierers auftreten, die das Decodierergebnis verschlechtern. Bild 1.24 veranschaulicht die prinzipielle Struktur dieses Decodierprozesses für das Beispiel einer parallelen Codeverkettung[2].

Sowohl der horizontale als auch der vertikale Decodierer besitzen zwei Ausgänge: Einer liefert die extrinsische Information $L_e(\hat{\mathbf{u}})$, die zusammen mit den Kanalausgangswerten das Eingangssignal eines Teildecodierers bildet. Der zweite Ausgang enthält das komplette Decodierergebnis $L(\hat{\mathbf{u}})$, aus dem dann durch Hard-Decision die Decodierentscheidung gewonnen werden kann.

Der Name 'Turbo-Codes'
Für die in Abschnitt 1.4.2 vorgestellten Turbo-Codes ist Bild 1.24 leicht zu modifizieren, um das Interleaving explizit sichtbar zu machen. Ferner ist Gl. (1.41) zu entnehmen ist, dass die extrinsische Information durch

[2]Bei einer seriellen Codeverkettung sind die Codesymbole $\mathbf{c}_1$ des äußeren Codierers C_1 die Eingangssymbole des inneren Codierers C_2. Daher muss der äußere Decodierer D_2 Soft-Werte für die codierten Bit $\mathbf{c}_1$ liefern, nicht nur für die Informationsbit $\mathbf{u}$.

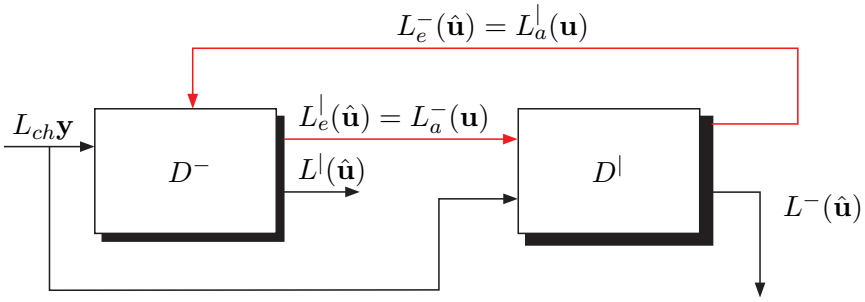


Bild 1.24: Iterativer Decodierprozeß

Subtraktion

$$L_e(\hat{u}_i) = L(\hat{u}_i) - L_{ch}y_i - L_a(u_i) \tag{1.62}$$

extrahiert werden kann. Wir erhalten somit die in Bild 1.25 skizzierte Struktur.

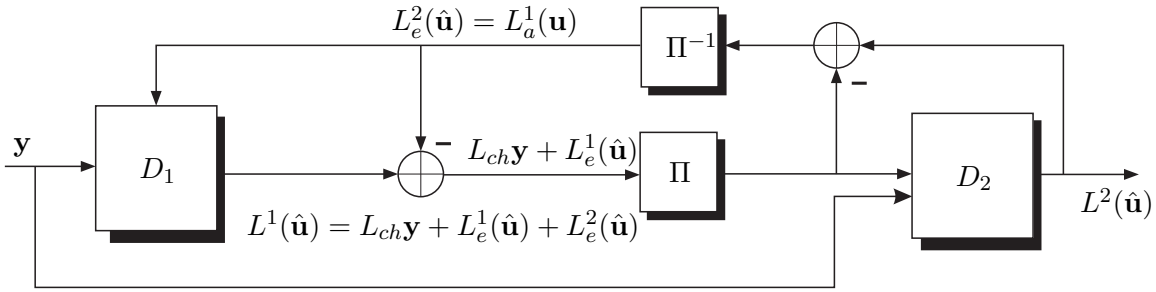


Bild 1.25: Iterativer Decodierprozeß für Turbo-Codes

Die von den Decodierern erzeugten *Soft*-Werte $L^1(\hat{\mathbf{u}})$ bzw. $L^2(\hat{\mathbf{u}})$ setzten sich bekanntermaßen aus drei Anteilen zusammen, von denen der Anteil der a-priori-Information $L_a(\hat{\mathbf{u}})$ subtrahiert wird. Übrig bleiben der direkte Informationsanteil der Kanalausgangswerte $L_{ch}\mathbf{y}$ und die extrinsische Information $L_e(\hat{\mathbf{u}})$, wobei letztere für den nachfolgenden Decodierer eine a-priori-Information bildet.

1.7.7 Ergebnisse zur iterativen Decodierung

Im letzten Abschnitt sollen noch einmal ein paar exemplarische Ergebnisse zur iterativen Decodierung vorgestellt werden, um die recht theoretischen Herleitungen der letzten Abschnitte zu veranschaulichen. Außerdem kann die Abhängigkeit der Leistungsfähigkeit verketteter Codes von wichtigen Parametern verdeutlicht werden.

Die Bilder 1.26 und 1.27 zeigen die Ergebnisse der *Union-Bound*-Abschätzung für drei verschiedene Produktcodes. Sie wurden aus drei Hamming-Codes konstruiert, dem (7,4)-Hamming-Code, dem (15,11)-Hamming-Code und dem (31,26)-Hamming-Code. Dabei wurde nur die Mindestdistanz $d_{\min} = 5$ berücksichtigt. Sie ist für alle drei Codes identisch, da Hamming-Codes stets die Mindestdistanz 3 besitzen und somit der unvollständige Produktcode die Mindestdistanz $3 + 3 - 1 = 5$. Lediglich der Koeffizient c_5 aus Gl. (1.11) ist unterschiedlich, weshalb die Verläufe sich leicht unterscheiden ((7,4): $c_5 = 0,5625$, (15,11): $c_5 = 0,2975$, (31,26): $c_5 = 0,1479$). Asymptotisch spielt dieser Faktor keine Rolle mehr.

Die Kurven sind also nur für große Signal-Rausch-Abstände genau. Trotzdem sind sie geeignet, um einige tendenzielle Aussagen zu treffen. Trägt man die Kurven über E_s/N_0 auf, ergeben sich wegen

$$P_b \leq c_5 \cdot \operatorname{erfc}\left(\sqrt{5\frac{E_s}{N_0}}\right)$$

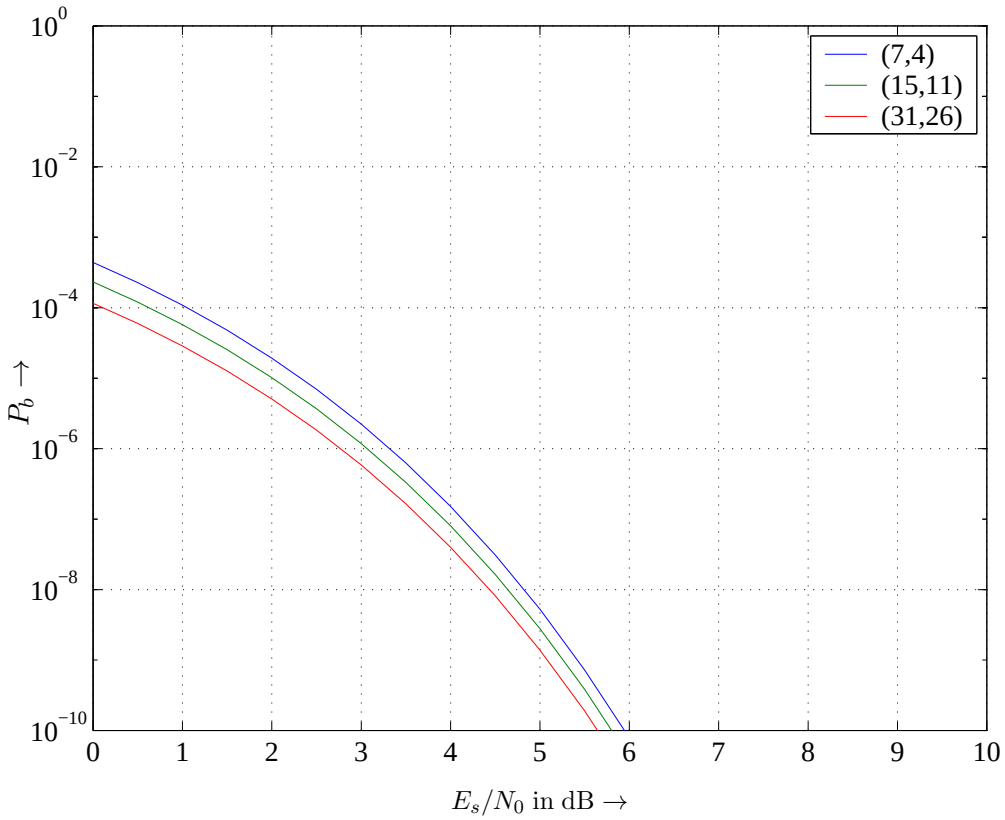


Bild 1.26: *Union-Bound*-Abschätzung (asymptotisch) für Produktcodes a) über E_s/N_0 und b) über E_b/N_0

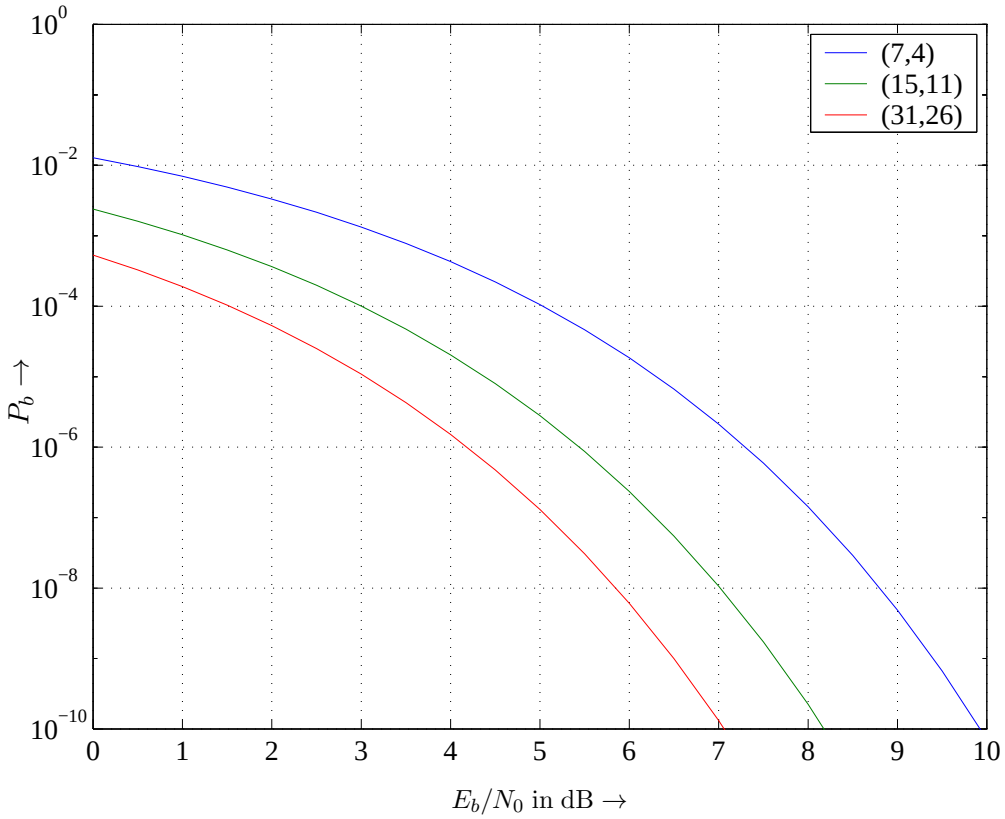


Bild 1.27: *Union-Bound*-Abschätzung (asymptotisch) für Produktcodes über E_b/N_0

nahezu keine Unterschiede, was aufgrund der gleichen Mindestdistanz nicht verwunderlich ist. Allerdings ist zu beachten, dass die drei Codes jeweils unterschiedliche Coderaten besitzen und somit auch die erforderliche

Bandbreite variiert. Berücksichtigt man diese Tatsache und trägt über E_b/N_0 auf,

$$P_b \leq c_5 \cdot \mathrm{erfc}\left(\sqrt{5\frac{R_c E_b}{N_0}}\right)$$

so erhält man die in Bild (1.27) dargestellten Verläufe. Es wird der deutliche Vorteil des (31,26)-Hamming-Codes deutlich, da er einerseits den geringsten Koeffizienten c_5 besitzt, andererseits die Distanz $d_{\min} = 5$ mit wesentlich geringerer Redundanz erreicht und somit eine größere spektrale Effizienz besitzt. Bezogen auf die je Informationsbit übertragene Energie ist dieser Produktcode also der beste der drei Kandidaten.

In den folgenden drei Bildern (1.28 - 1.30) sind Simulationsergebnisse dargestellt. Es wurden insgesamt stets 3 Decodierdurchläufe durchgeführt. Zum Vergleich ist jeweils das Ergebnis der analytischen Abschätzung ebenfalls mit aufgeführt.

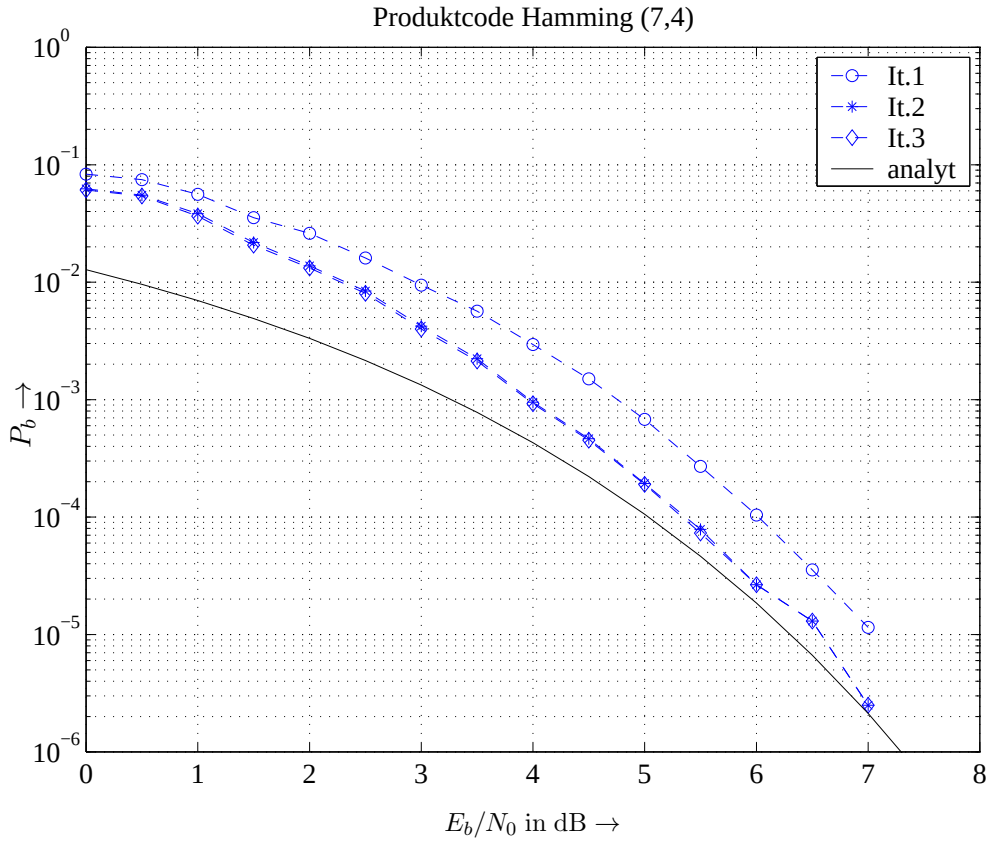


Bild 1.28: Bitfehlerraten für Produktcode aus zwei (7,4)-Hamming-Codes

Es ist erkennbar, dass mit jeder Iteration der zusätzliche Gewinn kleiner ausfällt. Dies liegt an der Tatsache, dass stets die gleichen Informationsbit geschätzt werden und die als a-priori-Information genutzte extrinsische Information immer stärker mit den übrigen Eingangsdaten korreliert. (Dies verdeutlicht noch einmal die Wichtigkeit der statistischen Unabhängigkeit der a-priori-Information von den übrigen Eingangssignalen). Die Gewinne zusätzlicher Iterationen fallen um so höher aus, je größer der Interleaver ist. Je weiter benachbarte Bit durch ihn auseinander gewürfelt werden, desto besser ist ihre statistische Unabhängigkeit erfüllt.

In Bild 1.30 ist außerdem deutlich zu erkennen, dass die Bitfehlerkurven ab einem gewissen Signal-Rausch-Abstand abflachen. Dies ist nicht auf Simulationsungenauigkeiten zurückzuführen, wie der Vergleich mit dem asymptotischen Verlauf der analytischen Abschätzung zeigt. Für große Signal-Rausch-Abstände dominiert einzig und allein die Minimaldistanz, und diese bestimmt die Steigung der Fehlerkurven. In Bild 1.29 is dieser Effekt ebenfalls zu beobachten, allerdings nicht so ausgeprägt.

Die Bilder 1.32 und 1.31 zeigen noch einmal den Vergleich der drei Produktcodes, nun aber anhand von Simulationsergebnissen. Über E_s/N_0 aufgetragen ist zu erkennen, dass die Mindestdistanz für große Signal-Rausch-

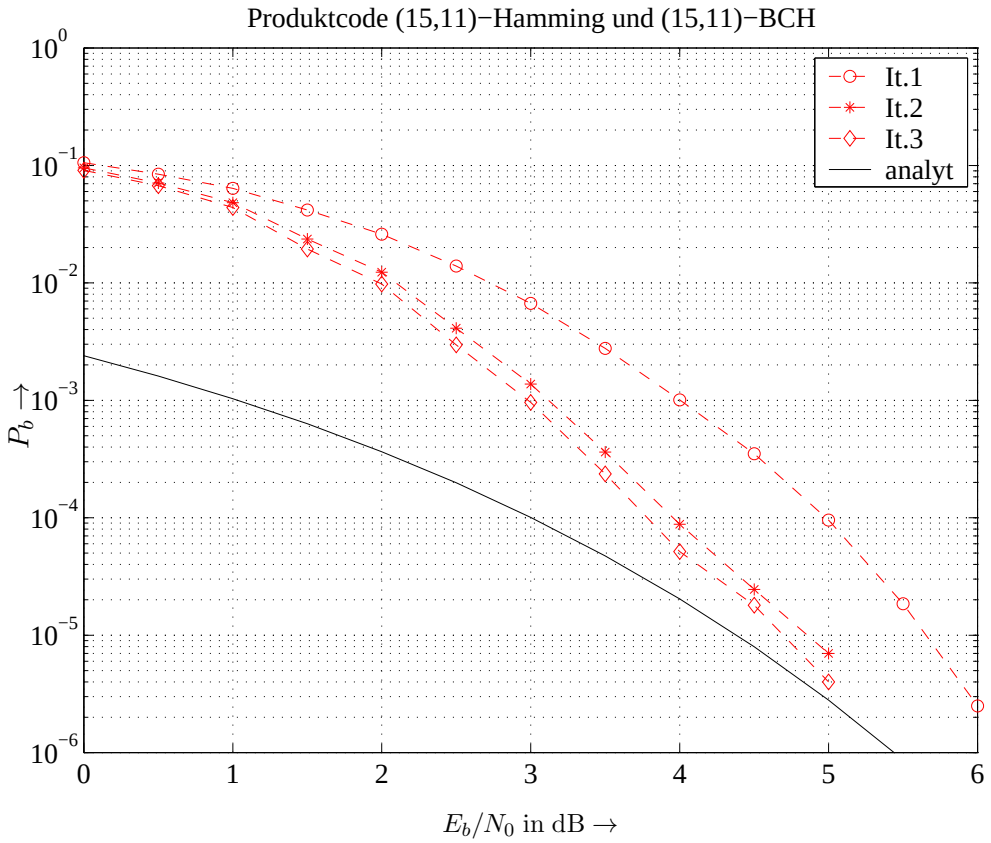


Bild 1.29: Bitfehlerraten für Produktcode aus zwei (15,11)-Hamming-Codes

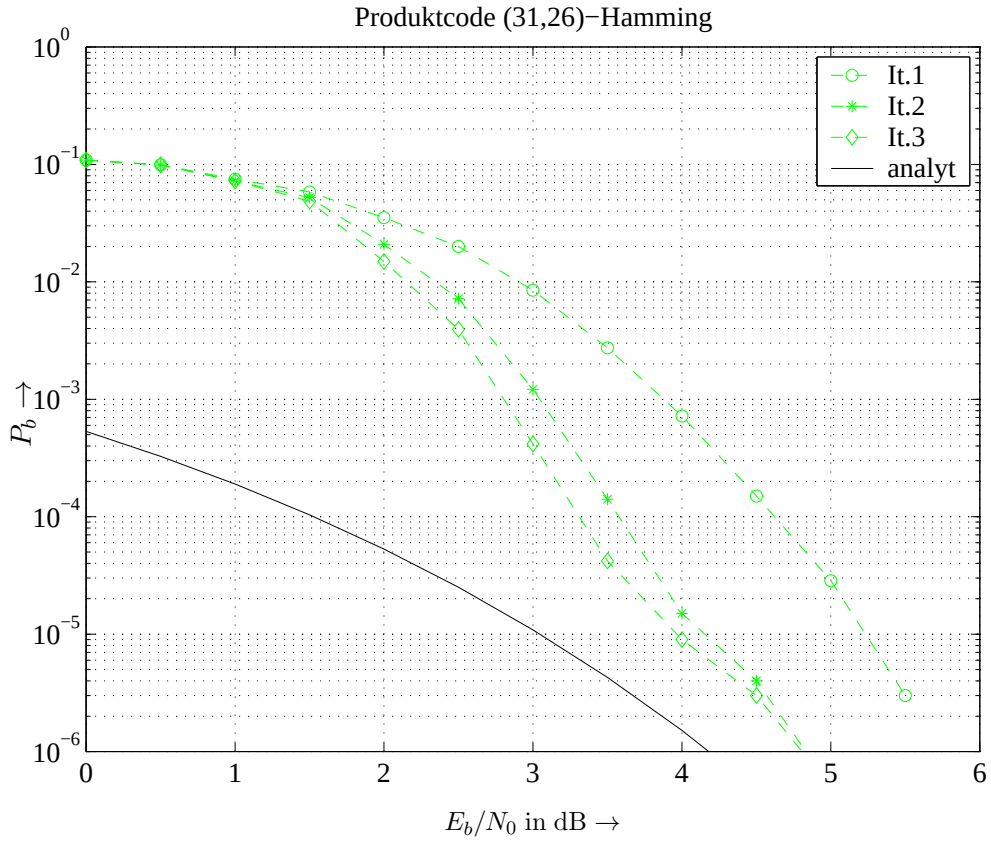


Bild 1.30: Bitfehlerraten für Produktcode aus zwei (31,26)-Hamming-Codes

Abstände dominiert, hier verlaufen alle drei Kurven sehr dicht beieinander. Für kleinere SNR erweist sich der (7,4)-Hamming-Code als der beste Komponentencode.

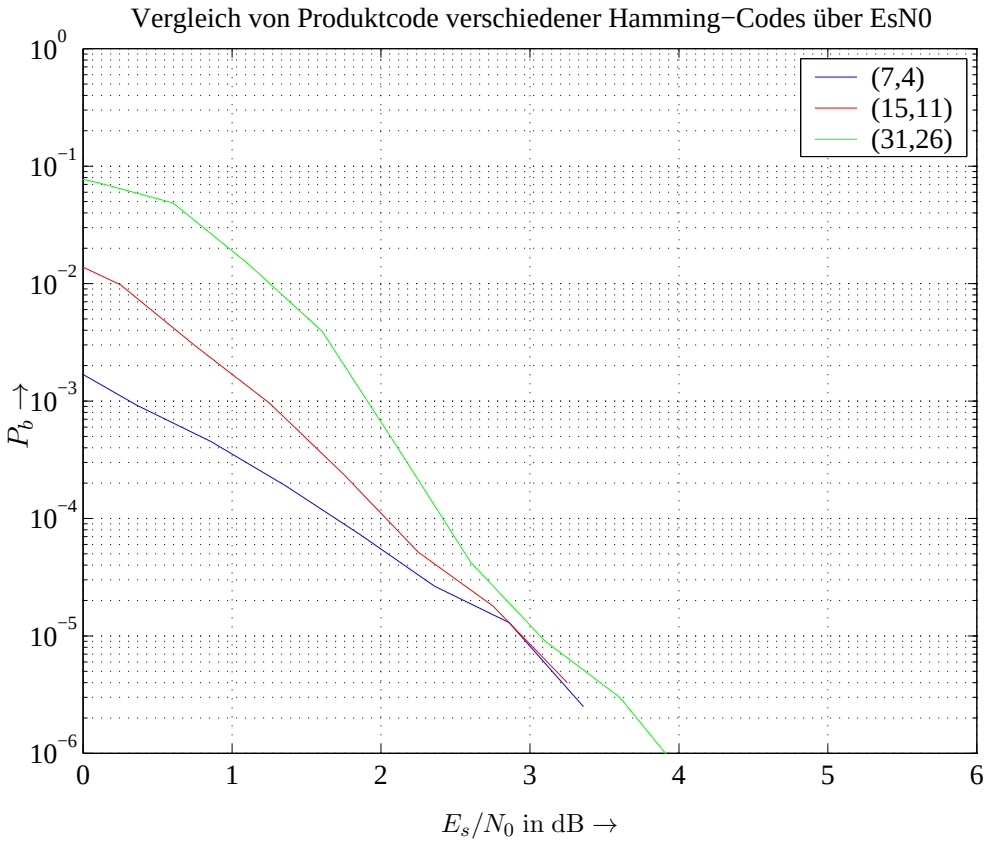


Bild 1.31: Vergleich der Bitfehlerraten für Produktcodes über E_s/N_0

Berücksichtigt man wiederum die unterschiedlichen Coderaten, so ergeben sich die in Bild 1.32 dargestellten Verhältnisse. Aufgrund der besten spektralen Effizienz besitzt der (31,26)-Hamming-Code die größte Leistungsfähigkeit.

Zum Ende dieses Kapitels zeigen die Bilder 1.33 bis 1.37 die Ergebnisse für die in Abschnitt 1.4.2 vorgestellten Turbo-Codes. Es wird deutlich, dass auch hier mit zunehmender Iterationszahl die Gewinne immer kleiner werden. Weiterhin führt eine Vergrößerung des Interleavers zu einer Verringerung der Bitfehlerrate. Allerdings sind hier lediglich einfache Blockinterleaver eingesetzt worden. Wie schon diskutiert wurde, sind aber pseudozufällige Interleaver besser geeignet. Dies ist daran zu erkennen, dass eine Vergrößerung von 400 auf 900 Bit keinen großen Gewinn mehr liefert.

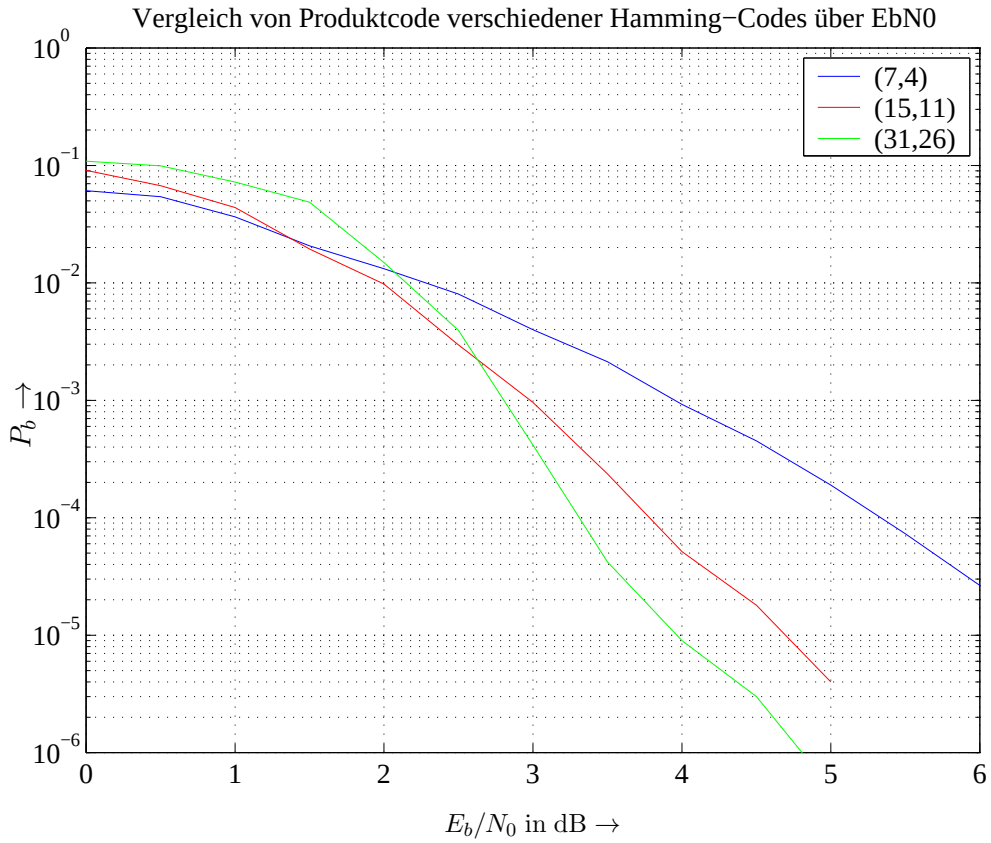


Bild 1.32: Vergleich der Bitfehlerraten für Produktcodes über E_b/N_0

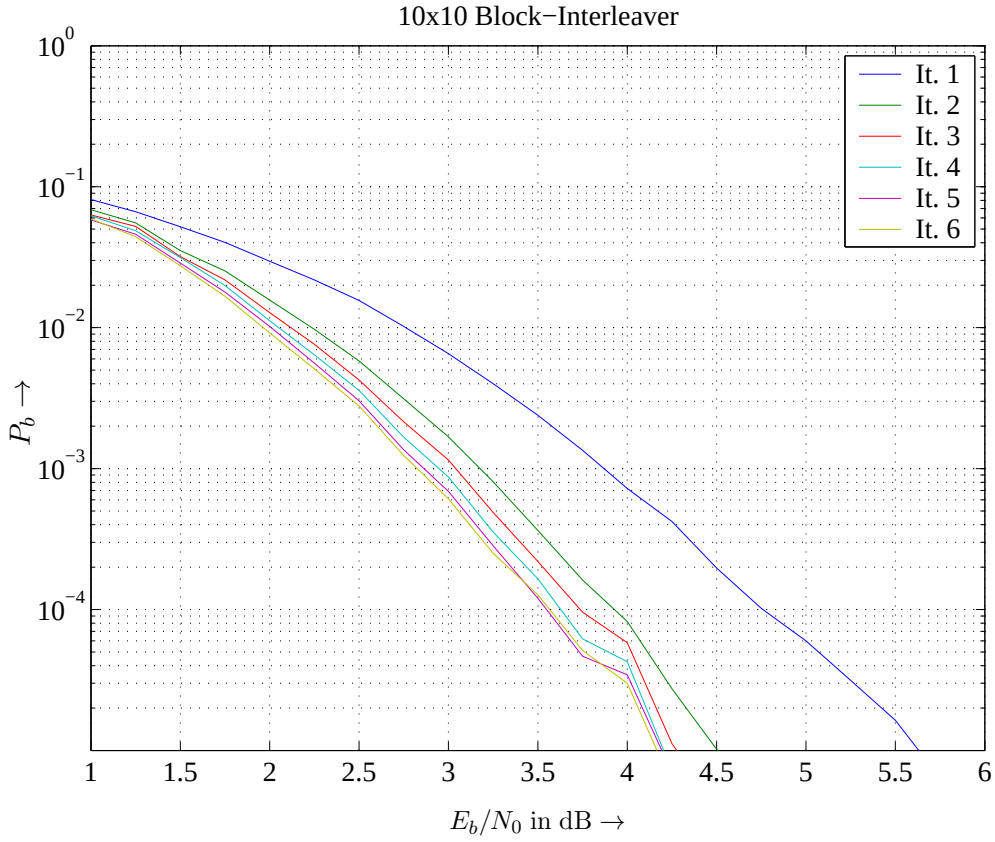


Bild 1.33: Bitfehlerraten für Turbo-Code mit 10x10-Blockinterleaver

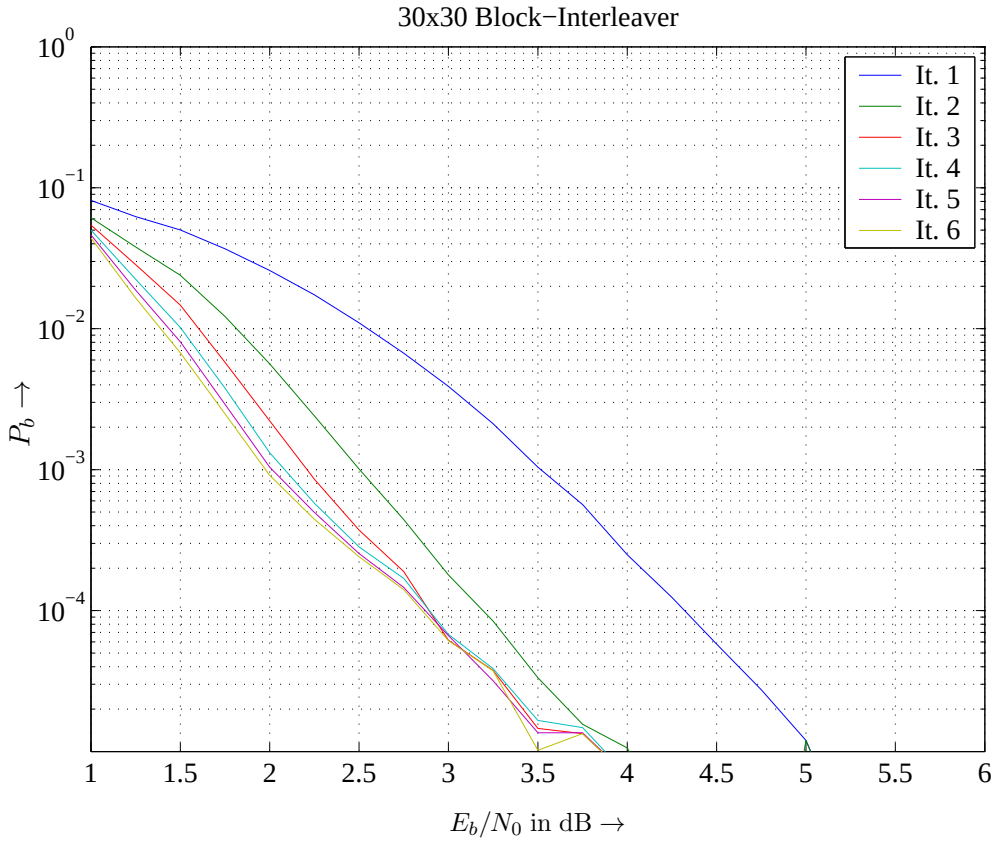


Bild 1.34: Bitfehlerraten für Turbo-Code mit 30x30-Blockinterleaver

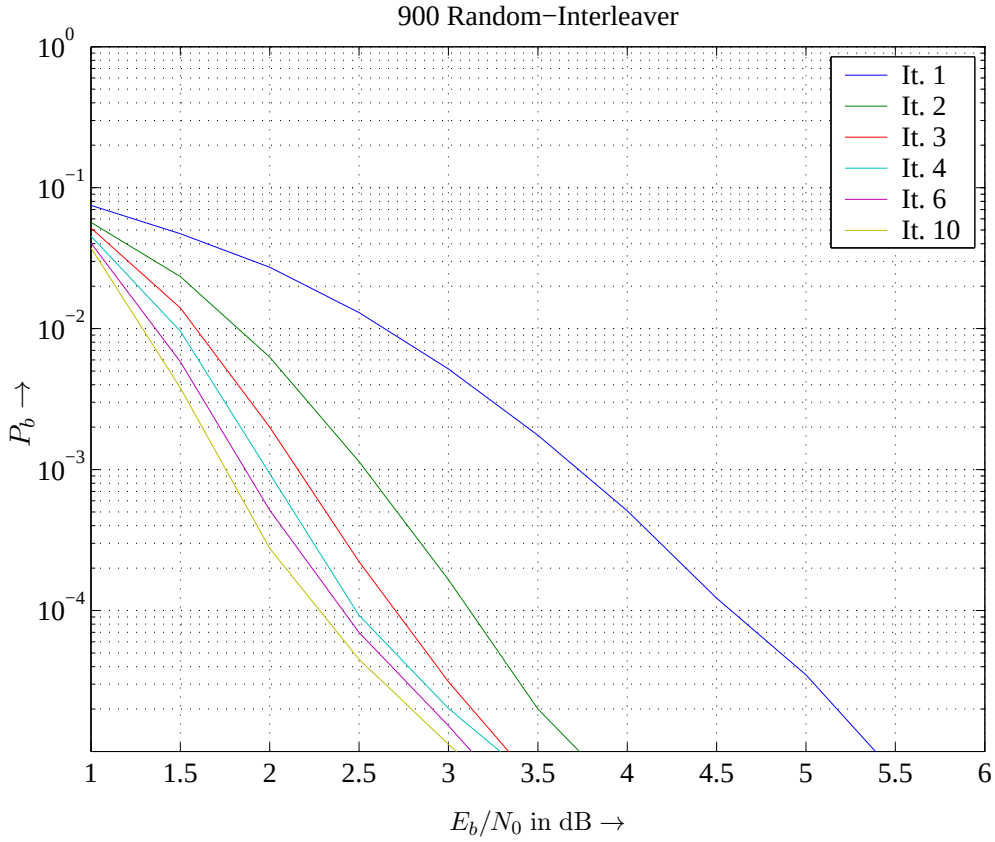


Bild 1.35: Bitfehlerraten für Turbo-Code mit 30x30-Zufallsinterleaver

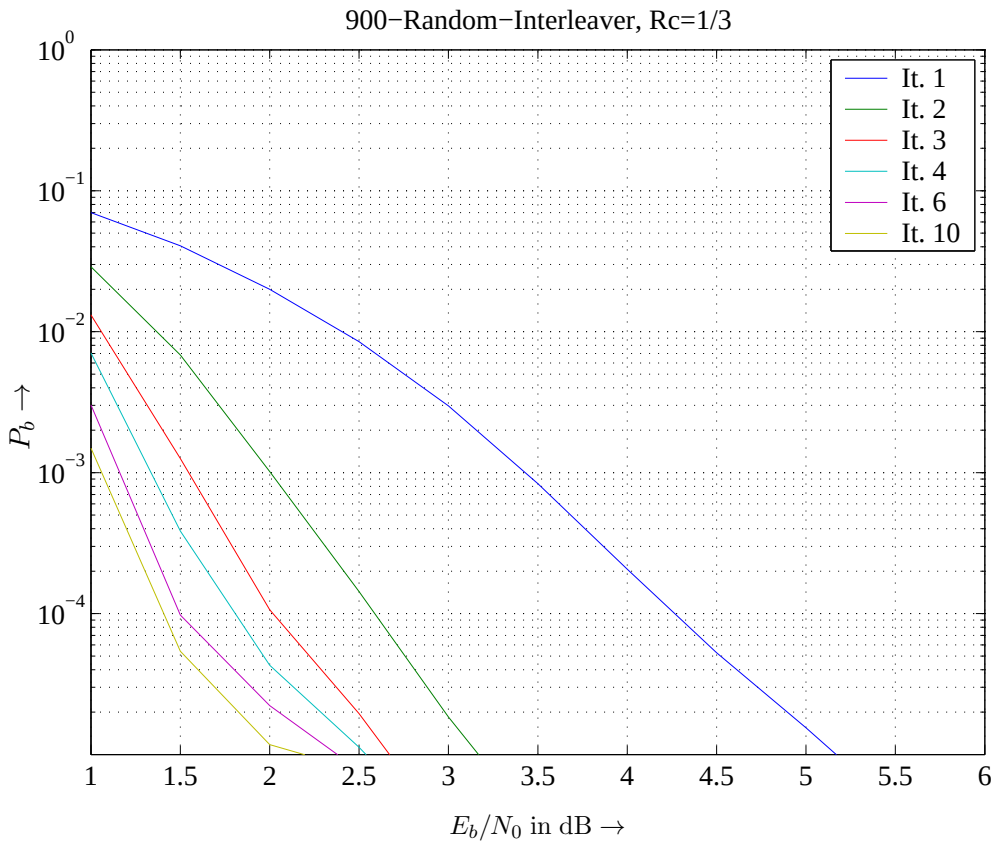


Bild 1.36: Vergleich der Bitfehlerraten für Turbo-Code mit 30x30-Zufallsinterleaver und $R_c = 1/3$

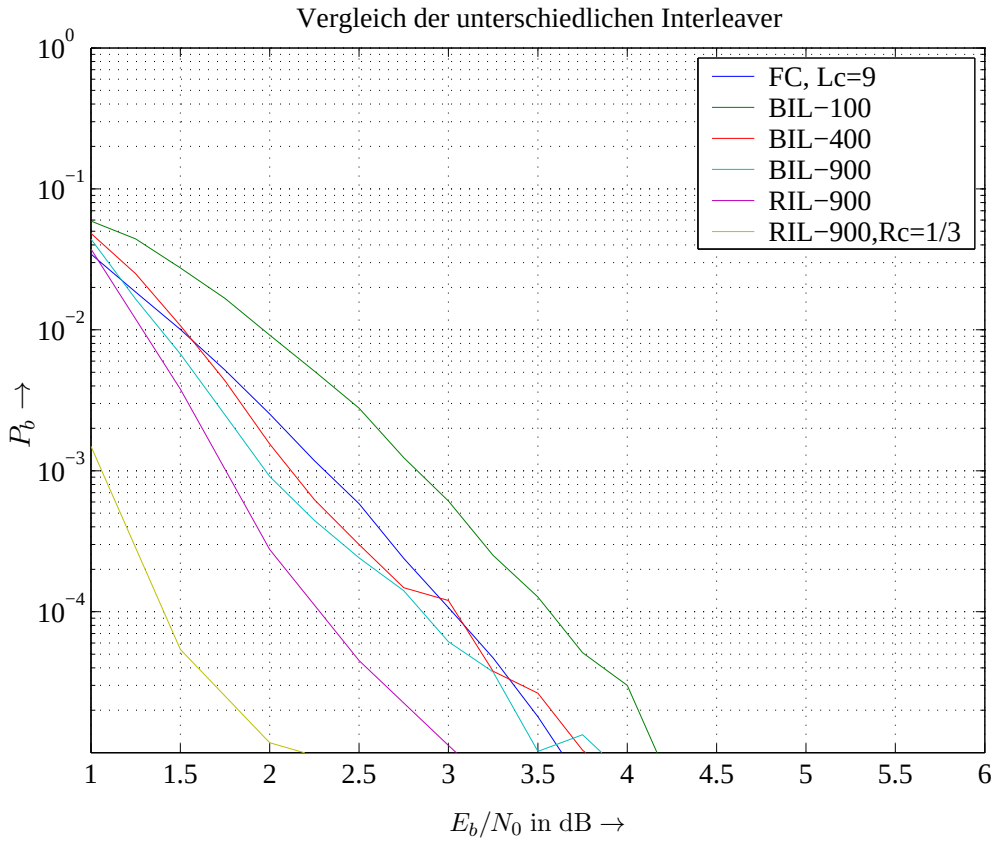


Bild 1.37: Vergleich der Bitfehlerraten für Turbo-Code mit verschiedenen Interleavergrößen ($R_c = 1/2$)

Kapitel 2

Trelliscodierte Modulation (TCM)

2.1 Einführung

Im ersten Teil der Vorlesung 'Kanalcodierung I' wurden verschiedene Verfahren der Kanalcodierung zur Fehlererkennung und Fehlerkorrektur vorgestellt. Alle Verfahren basieren auf der Technik, dem Datenstrom Redundanz hinzuzufügen. Hierdurch wird nur ein Teil der möglichen Sequenzen tatsächlich zur Übertragung verwendet, so dass sich die Distanz zwischen gültigen Codefolgen erhöht. Anhand der zugefügten Redundanz ist es dann möglich, Fehler zu erkennen und sogar zu korrigieren.

Für viele Kanäle ist die jedem Nutzer zur Verfügung stehende Bandbreite jedoch streng begrenzt (Beispiel: Telefonkanäle, ca. 3 kHz). Bei einer binären Modulation können schon im uncodierten Fall nur geringe Datenraten übertragen werden. Eine zusätzliche Kanalcodierung würde die Datenrate weiter reduzieren, so dass eine Datenübertragung über derartige Kanäle unattraktiv wird. Eine Lösung dieses Problems bietet die **Codierte Modulation**, die Kanalcodierung und hochstufige Modulation gewinnbringend miteinander kombiniert. Aus diesem Grund beschäftigt sich der nächste Abschnitt mit mehrstufigen linearen Modulationsverfahren, die schon aus der Vorlesung 'Nachrichtenübertragung' bekannt sein dürften.

Historie
Die Entwicklung der codierten Modulation geht zurück auf die 70er und 80er Jahre.

1977: Mehrstufencodes (*Multilevel Codes*) mit Kaskadierter Decodierung (*Multistage Decoding*)
Erstes praktisches Verfahren von Imai / Hirakawa (fand zunächst keine große Beachtung)

1982: Trelliscodierte Modulation von Ungerböck
Bahnbrechende Arbeit von Gottfried Ungerböck; Durchbruch in vielen Anwendungsbereichen wie der Modemtechnik und der Satellitenkommunikation

2.2 Lineare digitale Modulationsverfahren

2.2.1 Grundlagen

Im letzten Semester wurden in Zusammenhang mit der Codierung ausschließlich binäre Modulationsverfahren, wie z.B. BPSK bzw. 2-ASK betrachtet. Die Sendesymbole waren somit antipodale Signale mit den Werten $x(l) = \pm\sqrt{E_s/T_s}$, d.h. es kann $m = 1$ Binärstelle je Symbol übertragen werden. Bei der Codierten Modulation betrachten wir in dieser Vorlesung dagegen mehrstufige Modulationsverfahren, und zwar die M-PSK sowie die M-QAM. Sie gelten als bandbreiteneffiziente Verfahren, da sie mit zunehmender Stufigkeit immer weniger

Bandbreite bei konstanter Informationsrate benötigen. Dazu ändern wir die Notation gegenüber dem letzten Semester geringfügig.

Der Kanalcodierer erhalte nun Vektoren $\mathbf{u}(l) = (u_1(l) \cdots u_k(l))$ der Länge k mit binären Elementen $u_i(l) \in \{0,1\}$ im Symboltakt T_s. Er liefert dann im gleichen Takt Vektoren $\mathbf{c}(l) = (c_0(l) \cdots c_{m-1}(l))$ mit $c_i(l) \in \{0,1\}$ und $m > k$, die dann im Signalraumcodierer (*Mapper*) auf eines von $M = 2^m$ skalaren Kanalsymbolen $x(l)$ abgebildet werden. Die modifizierte Struktur des Datenübertragungssystems ist in Bild 2.1 dargestellt. Dabei ist zu beachten, dass der Signalraumdecodierer nicht unbedingt hart entschiedene Werte liefern muss, sondern er kann auch *Soft-Werte* berechnen (vgl. Kapitel 1). Dies wäre dann für den nachfolgenden Decodierer von Vorteil.

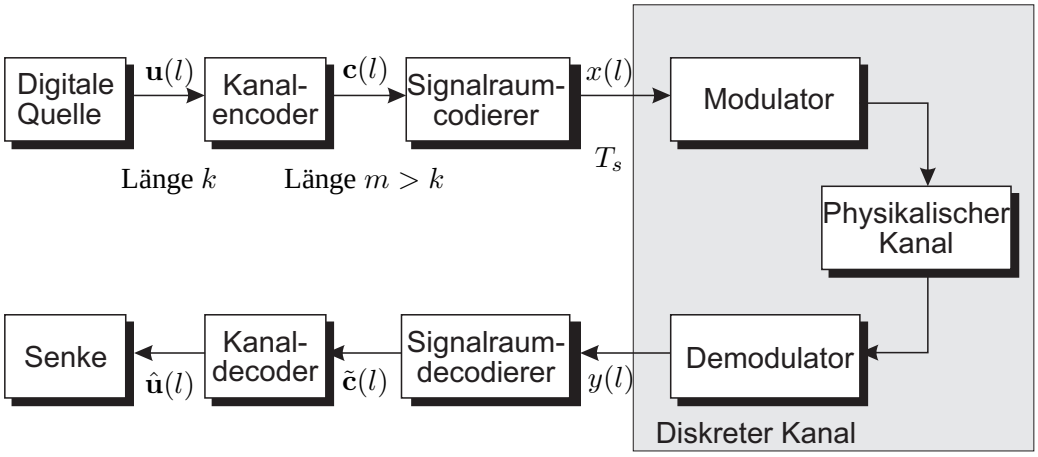


Bild 2.1: Blockschaltbild des digitalen Übertragungssystems

Im Rahmen der Vorlesung beschränken wir uns auf lineare digitale Modulationsverfahren, also beispielsweise Phasen- und Amplitudenmodulation. Nichtlineare Verfahren wie z.B. CPM (*Continuous Phase Modulation*), zu der auch die im GSM-Standard verwendete MSK (*Minimum Shift Keying*) gehört, sollen hier nicht behandelt werden. Bild 2.2 zeigt typische Signalraumkonstellationen linearer Modulationsverfahren. Es ist zu beachten, dass die Signale jetzt in der Regel komplexwertig sind. Im letzten Semester konnten wir ohne Einschränkung der Allgemeingültigkeit von reellen Symbolen ausgehen. Dies gilt nun nicht mehr, was direkte Auswirkungen auf die Berechnung von Fehlerwahrscheinlichkeiten und Metriken mit sich bringt. Die Größe Δ_0 beschreibt in Bild 2.2 den euklidischen Abstand zwischen benachbarten Symbolen in Abhängigkeit der Symbolenergie E_s/T_s und stellt somit die kleinste euklidische Distanz im Signalraum dar.

Selbstverständlich enthält Bild 2.2 nur eine kleine Auswahl von Modulationsverfahren. *State-of-the-Art* TCM-Verfahren verwenden heute die 960-QAM, um über Telefonkanäle Datenraten von bis zu 56 kbit/s zu übertragen. Dabei ist zu beachten, dass die Symbolrate und damit die Signalbandbreite weiterhin auf 3,2 kHz begrenzt sind. Im Rahmen dieser Vorlesung sollen anhand einiger überschaubarer Beispiele die Prinzipien der Codierten Modulation veranschaulicht werden.

Es stellt sich weiterhin die Frage, wie der Signalraumcodierer die binären Daten der codierten Sequenz $\mathbf{c}(l)$ auf die Symbolfolge $x(l)$ abbildet. Normalerweise wird ohne Berücksichtigung der Kanalcodierung ein **Gray-Code** gewählt. Dieser zeichnet sich dadurch aus, dass sich benachbarte Symbole bezüglich ihrer binären Kennung nur durch ein Bit unterscheiden. Eine derartige Zuordnung garantiert nun, dass beim Verwechseln zweier benachbarter Symbole nur ein einzelnes Bit gestört wird. Da diese Art der Verwechslung die häufigste ist, wird somit eine minimale Bitfehlerrate erzielt. Bild 2.3 zeigt die entsprechende Gray-Codierung für die obigen Modulationsverfahren.

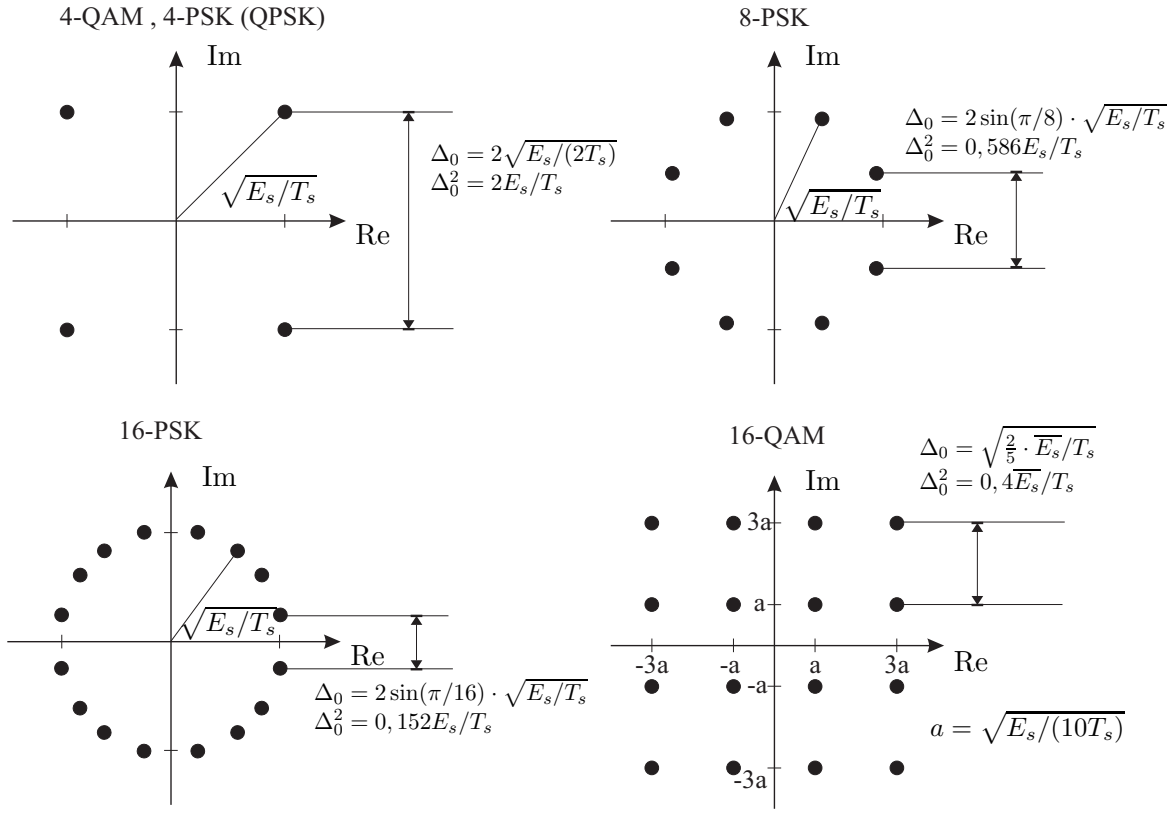


Bild 2.2: Signalraumdiagramme für einige hochstufige lineare Modulationsverfahren

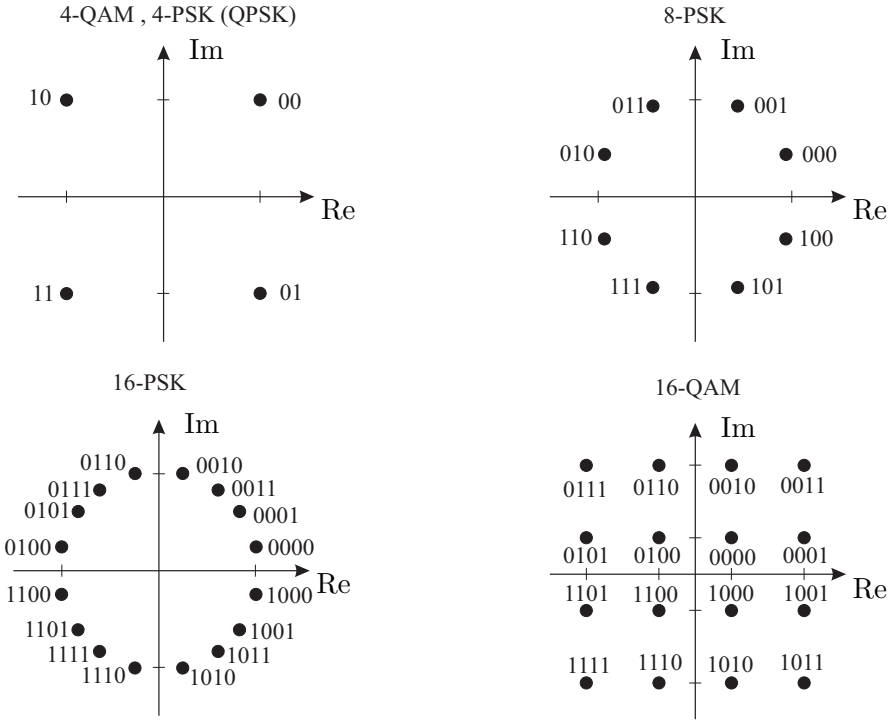


Bild 2.3: Gray-Codierung für einige lineare Modulationsverfahren

2.2.2 Bandbreiteneffizienz linearer Modulationsverfahren

Ein wichtiges Maß zur Beurteilung eines Modulationsverfahrens ist die Bandbreiteneffizienz. Sie spielt insbesondere bei der codierten Modulation eine wesentliche Rolle und hängt nicht nur von der Stufigkeit des

Modulationsverfahrens, sondern auch von der Impulsformung mit dem Tiefpaß $g_r(t)$ ab. Bei einer Symbolrate r_s beträgt die Symboldauer $T_s = 1/r_s$, wodurch sich mit $g_r(t)$ die Bandbreite

$$B = \frac{\alpha}{T_s} \tag{2.1}$$

ergibt. Der Wert α im Zähler von Gl. (2.1) hängt dabei direkt von der Impulsformung ab. Dies soll anhand eines Beispiels verdeutlicht werden.

Beispiel: Kosinus-roll-off-Impulsformung
Die Impulsantwort eines Kosinus-roll-off-Filters hat bekanntermaßen die Form

$$g_r(t) = \frac{\sin\left(\pi\frac{t}{T_s}\right)}{\pi\frac{t}{T_s}} \cdot \frac{\cos\left(\pi r\frac{t}{T_s}\right)}{1-\left(2r\frac{t}{T_s}\right)^2} = \frac{\sin(\omega_N t)}{\omega_N t} \cdot \frac{\cos(\omega_N r t)}{1-(4f_N r t)^2},$$

wobei $f_N = \omega_N/(2\pi) = 1/(2T_S)$ die Nyquist-Frequenz bezeichnet. Das zugehörige Spektrum lautet

$$G_r(j\omega) = \begin{cases} 1 & \text{für } \frac{|\omega|}{\omega_N} \le 1-r \\ \frac{1}{2}\cdot\left[1+\cos\left(\frac{\pi}{2r}(\frac{|\omega|}{\omega_N}-(1-r))\right)\right] & \text{für } 1-r \le \frac{|\omega|}{\omega_N} \le 1+r \\ 0 & \text{für } \frac{|\omega|}{\omega_N} \ge 1+r \end{cases}.$$

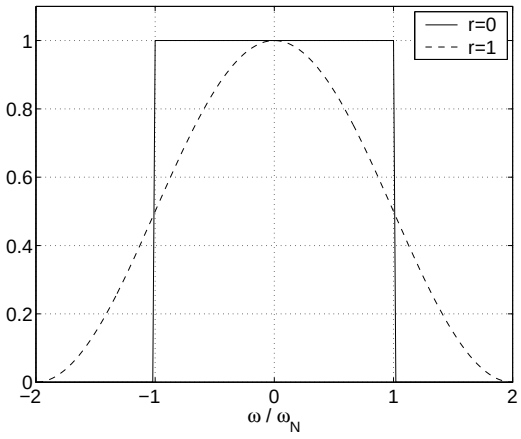
Damit ergibt sich z.B. für die beiden Extremfälle $r = 0$ (idealer Tiefpaß) und $r = 1$:

$$G_{r=0}(j\omega) = \begin{cases} 1 & \text{für } \frac{|\omega|}{\omega_N} \le 1 \\ 0 & \text{sonst} \end{cases}$$

$$\Longrightarrow \quad B = 2f_N = \frac{1}{T_s} = r_s \longrightarrow \alpha = 1$$

$$G_{r=1}(j\omega) = \begin{cases} \frac{1}{2}\cdot\left[1+\cos\left(\frac{\pi}{2}\frac{\omega}{\omega_N}\right)\right] & \text{für } \frac{|\omega|}{\omega_N} \le 2 \\ 0 & \text{sonst} \end{cases}$$

$$\Longrightarrow \quad B = 4f_N = \frac{2}{T_s} = 2r_s \longrightarrow \alpha = 2$$



Für den idealen Tiefpaß entspricht die Bandbreite B im Übertragungsband (bei angenommener Verschiebung ins Übertragungsband durch eine Amplitudenmodulation) der Symbolrate r_s bzw. dem Kehrwert der Symboldauer T_s. Es ist zu beachten, dass im äquivalenten Basisband nur die Hälfte der Bandbreite benötigt wird (AM verdoppelt die Bandbreite). Ein Kosinus-roll-off-Filter mit $r = 1$ benötigt dagegen die doppelte Bandbreite $B = 2/T_s$. **Im weiteren Verlauf soll idealisierend stets der ideale Tiefpaß ($\alpha = 1$) angenommen werden!**

Definition der spektralen Effizienz:

$$\boxed{\eta = \frac{\textbf{Info-Datenrate}}{\textbf{Bandbreite}} = \frac{r_b}{B} = \frac{1/T_b}{\alpha/T_s} = \frac{T_s}{T_b} = m} \tag{2.2}$$

Für den idealen Tiefpaß gilt also, dass die spektrale Effizienz η im Übertragungsband gleich der Anzahl Bit je Kanalsymbol ist.

2.2.3 Fehlerwahrscheinlichkeit linearer Modulationsverfahren

Zur Beurteilung der Leistungsfähigkeit linearer Modulationsverfahren soll nun die Fehlerwahrscheinlichkeit betrachtet werden. Wir nehmen dazu einen AWGN-Kanal nach Bild 2.4 mit spektraler Rauschleistungsdichte $N_0/2$ an, wobei zu beachten ist, dass die Größen $x(l)$, $n(l)$ und $y(l)$ nun komplexwertig sein können.

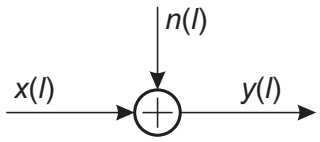


Bild 2.4: Darstellung AWGN-Kanal

- Symbolfehlerwahrscheinlichkeit nach *Maximum Likelihood*-Kriterium

$$P_s = P\left(\|y(l) - x(l)\|^2 > \|y(l) - x'(l)\|^2\right) \qquad \forall \quad x(l),\, x'(l) \in \mathcal{A}_{in},\; x(l) \neq x'(l) \tag{2.3}$$

- Exakte Berechnung der Fehlerrate i.a. sehr aufwendig

$\longrightarrow$ Näherung: In den meisten Fällen werden direkt benachbarte Symbole verwechselt, da sie die geringste euklidische Distanz zueinander besitzen!

$\longrightarrow$ Wahrscheinlichkeit, dass zwei benachbarte Symbole verwechselt werden, dominiert die Gesamtfehlerrate (s. [Kam04])

$\longrightarrow$ Euklidische Distanz zwischen benachbarten Symbolen Δ_0 ist ausschlaggebend (je größer Δ_0, desto geringer die Fehlerrate)

- Laut Bild 2.2 nimmt Δ_0 mit wachsender Stufigkeit des Modulationsverfahrens ab
- Für PSK-Modulation gilt allgemein:

$$\Delta_0 = 2\sin\left(\frac{\pi}{M}\right) \cdot \sqrt{E_s/T_s} \tag{2.4}$$

$\Longrightarrow$ **Mit steigender spektraler Effizienz wächst gleichzeitig auch die Fehlerhäufigkeit!**

M-PSK

Im folgenden betrachten wir zunächst die digitale Phasenmodulation. Zur Berechnung der Auftrittswahrscheinlichkeit eines Fehlers nehmen wir Bild 2.3 zur Hilfe. Ohne Beschränkung der Allgemeingültigkeit betrachten wir die beiden Symbole (000) und (100) einer 8-PSK mit der euklidische Distanz Δ_0. Wird das Symbol (100) gesendet, tritt genau dann ein Fehler auf, wenn der Imaginärteil n_i'' des Rauschens größer als $\Delta_0/2$ ist. Die Wahrscheinlichkeit hierfür ist vom Signal-Rausch-Abstand E_s/N_0 bzw. E_b/N_0 abhängig und lautet

$$\begin{aligned} P\left(n_i'' > \frac{\Delta_0}{2}\right) &= P\left(n_i'' > \sqrt{E_s/T_s} \cdot \sin\left(\frac{\pi}{M}\right)\right) \\ &= \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_s}{N_0}} \cdot \sin(\pi/M)\right) = \frac{1}{2}\mathrm{erfc}\left(\sqrt{m\frac{E_b}{N_0}} \cdot \sin(\pi/M)\right) . \end{aligned} \tag{2.5}$$

Es ist zu beachten, dass pro Symbol m Informationsbit übertragen werden, weshalb der Zusammenhang $E_b = E_s/m$ gilt. Da jedes Symbol zwei Nachbarn hat, mit denen es verwechselt werden kann, führt auch $n_i'' < -\Delta_0/2$ zu einer fehlerhaften Entscheidung. Aufgrund der geltenden Symmetrie lässt sich daher die Symbolfehlerwahrscheinlichkeit folgendermaßen abschätzen:

$$P_s \approx 2 \cdot P\left(n_i'' > \frac{\Delta_0}{2}\right) = \mathrm{erfc}\left(\sqrt{\frac{E_s}{N_0}} \cdot \sin(\pi/M)\right) = \mathrm{erfc}\left(\sqrt{m\frac{E_b}{N_0}} \cdot \sin(\pi/M)\right) . \tag{2.6}$$

M-QAM

Für eine M-QAM-Modulation kann eine ähnliche Vorgehensweise wie bei der M-PSK verwendet werden. Da Real- und Imaginärteil des Rauschens als auch des QAM-Signals voneinander unabhängig sind, können die beiden Dimensionen getrennt voneinander betrachtet werden. Aufgrund der hier angenommenen symmetrischen Signalräume besitzen die sich ergebenden $\sqrt{M}$-ASK-Dimensionen die gleiche Fehlerwahrscheinlichkeit. Ein Unterschied zur PSK ist allerdings, dass die Symbole nun unterschiedliche Energien besitzen. Daher wird zunächst die mittlere Symbolenergie einer $\sqrt{M}$-ASK berechnet. Es gilt (Potenzreihenentwicklung)

$$\overline{E_s}/T_s = \frac{2}{\sqrt{M}} \cdot \sum_{i=1}^{\sqrt{M}/2} [(2i-1)a]^2 = \frac{2a^2}{\sqrt{M}} \cdot \frac{\frac{\sqrt{M}}{2}(4\frac{M}{4}-1)}{3} = a^2 \frac{M-1}{3} \ . \tag{2.7}$$

Zu einer Fehlentscheidung kommt es wiederum genau dann, wenn der Rauschwert die halbe Mindestdistanz überschreitet. Dieser Fall tritt mit der Wahrscheinlichkeit

$$P\left(n_i' > \Delta_0/2 = a\right) = \frac{1}{2} \cdot \mathrm{erfc}\left(\frac{a}{\sqrt{N_0/T_s}}\right) \ . \tag{2.8}$$

Bei der Berechnung der Symbolfehlerwahrscheinlichkeit ist jetzt noch die Tatsache zu berücksichtigen, das jedes Symbol 2 direkte Nachbarn hat, die beiden äußeren Symbole ausgenommen. Trägt man dieser Besonderheit durch Einführen eines Mittelungsfaktor $\frac{2\sqrt{M}-2}{\sqrt{M}}$ Rechnung, so ergibt sich mit Hilfe von Gl. (2.7)

$$P_s^{\sqrt{M}-\mathrm{ASK}} \approx \frac{\sqrt{M}-1}{\sqrt{M}} \cdot \mathrm{erfc}\left(\sqrt{\frac{\overline{E_s}}{N_0} \cdot \frac{3}{M-1}}\right) = \frac{\sqrt{M}-1}{\sqrt{M}} \cdot \mathrm{erfc}\left(\sqrt{m\frac{\overline{E_b}}{N_0} \cdot \frac{3}{M-1}}\right) \ . \tag{2.9}$$

Die Symbolfehlerrate für die M-QAM berechnet sich letztendlich aus der Überlegung, dass ein Symbol nur dann korrekt empfangen wurde, wenn sowohl Real- als auch Imaginärteil gleichzeitig korrekt sind. Die Auftrittswahrscheinlichkeit für diesen Fall ergänzt sich mit der Symbolfehlerwahrscheinlichkeit zu Eins und wir erhalten

$$P_s^{M-\mathrm{QAM}} = 1 - \left(1 - P_s^{\sqrt{M}-\mathrm{ASK}}\right)^2 = 2P_s^{\sqrt{M}-\mathrm{ASK}} - \left(P_s^{\sqrt{M}-\mathrm{ASK}}\right)^2 \ . \tag{2.10}$$

Die Bitfehlerrate, d.h. die relative Häufigkeit falscher Informationsbit kann einfach abgeschätzt werden, wenn eine Gray-Codierung vorausgesetzt wird. Dann führt nämlich die Detektion eines benachbarten (falschen) Symbols zu genau einem falschen Informationsbit und es gilt

$$P_b \approx \frac{1}{m} \cdot P_s \tag{2.11}$$

Aus den Bildern 2.5 und 2.6 ist ersichtlich, dass mit zunehmender Stufigkeit der Modulationsverfahren auch die Fehleranfälligkeit ansteigt. Dies bedeutet, dass spektrale Effizienz und Leistungsfähigkeit miteinander konkurrieren und ein entsprechender Kompromiß gefunden werden muss! Bemerkenswert ist die Tatsache, dass die 16-QAM nur unwesentlich schlechter ist als die 8-PSK, obwohl ihre spektrale Effizienz größer ist als die der 8-PSK. Dies liegt an der besseren Anordnung der Symbole im Signalraum, wodurch die zur Verfügung stehende Fläche effizienter genutzt wird und somit im Mittel die Distanz zwischen benachbarten Symbolen größer ist.

Es ist zu beachten, dass der Signal-Rausch-Abstand in Bild 2.6 auf die Energie pro Informationsbit E_b/N_0 bezogen ist. Die verschiedenen Modulationsverfahren besitzen unterschiedliche spektrale Effizienzen, so z.B. die QPSK $\eta = 2$ Bit/s/Hz, die 8-PSK $\eta = 3$ Bit/s/Hz und 16-PSK sowie 16-QAM $\eta = 4$ Bit/s/Hz. Für eine konstante Symbolenergie E_s würde dies bedeuten, dass pro Informationsbit mit zunehmender Stufigkeit des Modulationsverfahrens immer weniger Energie übertragen wird. So besitzt bei der QPSK jedes Informationsbit die Energie $E_b = E_s/2$, bei der 16-PSK dagegen nur noch $E_b = E_s/4$. Dafür benötigt die 16-PSK gegenüber der QPSK aber auch nur ein Viertel der Bandbreite. Um den Vergleich fairer zu gestalten, wird daher häufig nicht über die Symbolenergie E_s aufgetragen, sondern über die Energie je Informationsbit aufgetragen.

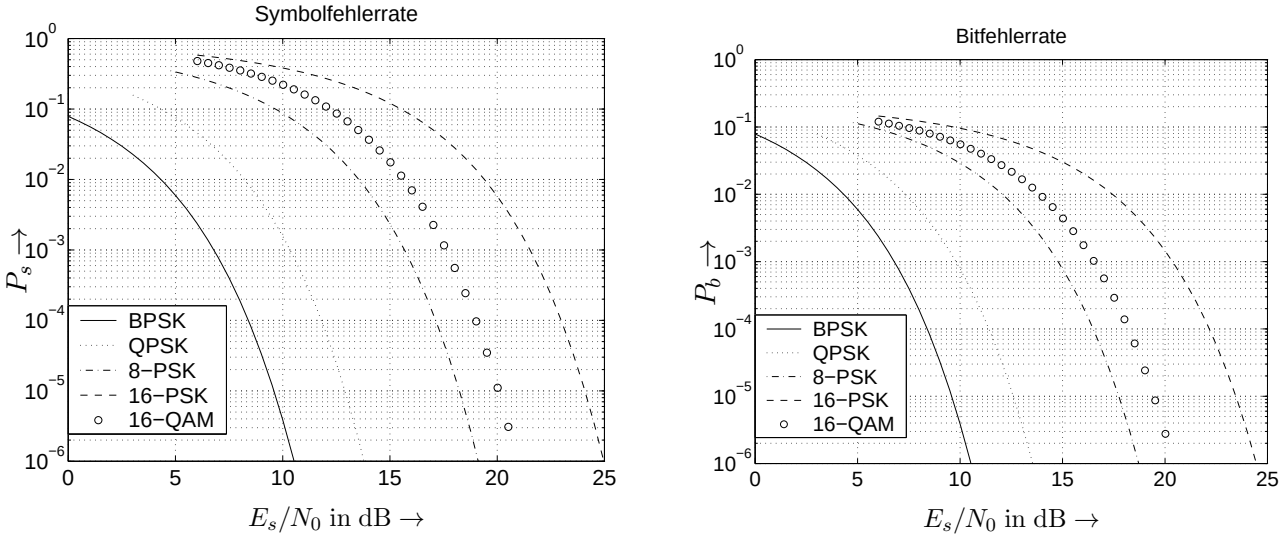


Bild 2.5: Symbol- und Bitfehlerraten für verschiedene Modulationsverfahren über E_s/N_0

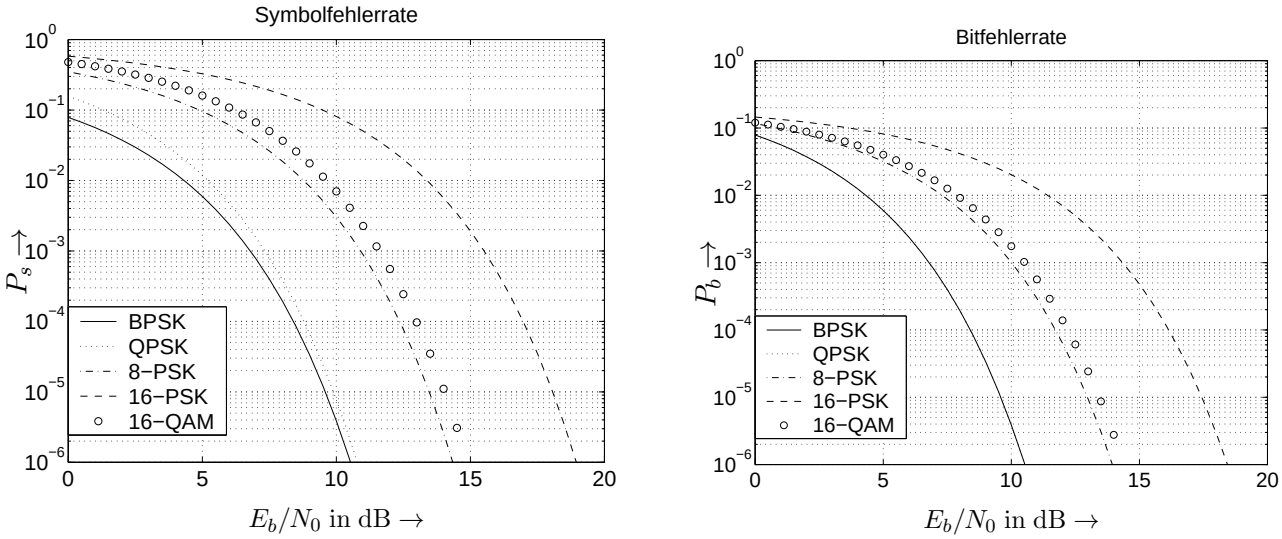


Bild 2.6: Symbol- und Bitfehlerraten für verschiedene Modulationsverfahren über E_b/N_0

2.3 Prinzip der codierten Modulation

2.3.1 Grundsätzliche Vorgehensweise

Das generelle Prinzip der codierten Modulation lässt sich nun einfach beschreiben. Man vervielfacht die Anzahl möglicher Sendesymbole beispielsweise von M auf $\tilde{M} > M$. Dies führt zu einer Erhöhung der Anzahl Binärstellen pro Kanalsymbol, d.h. statt m können nun $\tilde{m}$ Bit je Symbol übertragen werden, ohne dass sich die Signalbandbreite erhöht. Die zusätzlichen $\tilde{m} - m$ Bit werden durch eine Kanalcodierung erzeugt! Diese prinzipielle Strategie soll im Folgenden an einem Beispiel veranschaulicht werden.

Beispiel:
Wir legen zunächst folgende Vereinbarung zugrunde. Die Informationsdatenrate sei r_b, die Datenrate der codierten Bit $\mathbf{c}$ sei r_c und die Symbolrate am Ausgang des Signalraumcodierers ist r_s.

1. Uncodierte QPSK-Übertragung

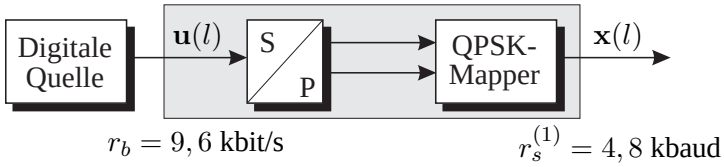


Bild 2.7: Darstellung der Raten bei QPSK-Modulation

- Die Rate des uncodierten binären Datenstroms beträgt $r_b = 9.600$ bit/s.
- Bei QPSK-Modulation werden pro Kanalsymbol 2 Bit übertragen

  $\Longrightarrow$ die Symbolrate beträgt im uncodierten Fall (s. Bild 2.7)

$$r_s^{(1)} = \frac{r_b}{m} = \frac{9.600}{2} = 4.800 \text{ Symbole/s} = 4.800 \text{ Baud}$$

2. Zusätzliche Kanalcodierung mit Coderate $R_c = 2/3$ erhöht die Datenrate um den Faktor 3/2 (s. Bild 2.8)

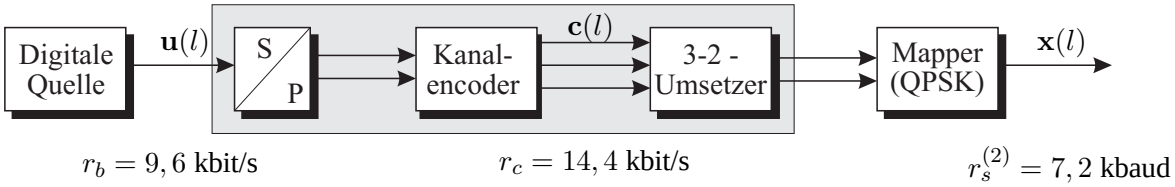


Bild 2.8: Darstellung der Raten bei Kanalcodierung und QPSK-Modulation

$$\begin{aligned} r_c &= \frac{r_b}{R_c} = 14.400 \text{ bit/s} \\ r_s^{(2)} &= \frac{r_c}{m} = 7.200 \text{ Baud} > r_s^{(1)} \end{aligned}$$

3. Bei Verwendung einer 8-PSK-Modulation ($\tilde{m} = 3$) ergibt sich folgender Zusammenhang (s. Bild 2.9):

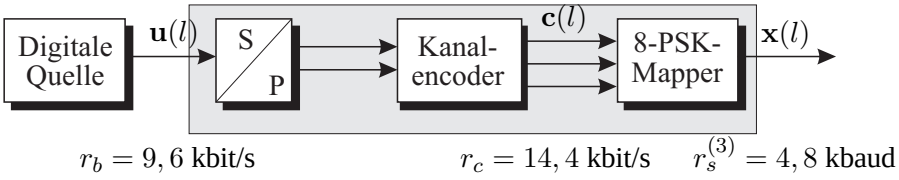


Bild 2.9: Darstellung der Raten bei Kanalcodierung und 8-PSK-Modulation

$$r_s^{(3)} = \frac{r_c}{\tilde{m}} = \frac{r_b}{R_c \cdot \tilde{m}} = \frac{9.600}{2/3 \cdot 3} = 4,8 \text{ kbaud} \stackrel{!}{=} r_s^{(1)}$$

d.h. es wird die gleiche Bandbreite wie im uncodierten Fall benötigt!

$\longrightarrow$ **Durch Verwendung der 8-PSK wird Bandbreitenerhöhung der Kanalcodierung kompensiert!**
Pro 8-PSK-Symbol werden 2 Informationsbit und 1 Prüfbit übertragen.

Aber: 8-PSK ist schlechter als QPSK!

Frage: Gibt es noch einen Codiergewinn und wie groß ist er?

2.3.2 Weg zur einheitlichen Betrachtung von Codierung und Modulation

Zur Beantwortung der obigen Frage betrachten wir zunächst Codierung und Modulation als getrennte Einheiten. Die Kanalcodierung erfolgt mit der Rate $R_c = m/(m+1)$, bevor die Modulation $m+1$ codierte Bit auf ein Kanalsymbol abbildet. Die spektrale Effizient beträgt damit $\eta = m$ bit/s/Hz. Im Empfänger findet dann die Demodulation und getrennt von ihr die Decodierung statt.

Die Kanalcodierung muss also die größere Fehleranfälligkeit des mehrstufigen Modulationsverfahrens mehr als kompensieren, um insgesamt einen Gewinn zu erzielen!

Wir wollen uns das Ergebnis anhand eines einfachen Beispiels erläutern und betrachten einen Faltungscode mit Rate $R_c = 2/3$ und Einflußlänge $L_c = 7$, kombiniert mit der 8-PSK-Modulation.

- Übergang von QPSK auf 8-PSK führt nach Bild 2.5 im uncodierten Fall zu einem Verlust von $E_s/N_0 = 5,3$ dB (für E_b/N_0 reduziert sich der Verlust auf 3,6 dB)
- Codiergewinn bei obigen Faltungscode beträgt etwa 6 dB gegenüber dem uncodierten Fall (vgl. Ergebnisse aus Vorlesung Kanalcodierung I)

$\longrightarrow$ Die Bilanz ergibt nur einen kleinen Gewinn von etwa 0,7 dB

Es scheint also, dass diese 'triviale' Zuordnung von codierten Bit auf Kanalsymbole nicht die gewünschte Leistungsfähigkeit bieten kann! Dies steht im Gegensatz zur binären Übertragung, wo die Art der Zuordnung von codierten Bit auf die Kanalsymbole keinen Einfluß auf die Leistungsfähigkeit hatte.

Lösung: Die Struktur des Codes muss bei der Zuordnung auf die Kanalsymbole berücksichtigt werden (s. Bild 2.10)

$\Longrightarrow$ Kanalcodierung und Modulation verschmelzen zu einer Einheit

$\Longrightarrow$ **Codierte Modulation**

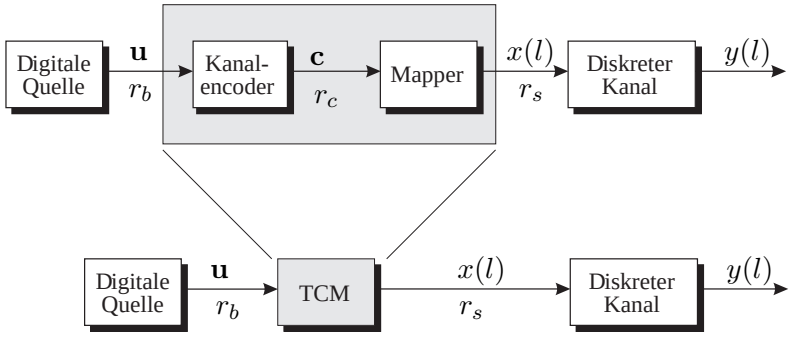


Bild 2.10: Zusammenfassen von Codierung und Modulation

Im Empfänger können dann Demodulation und Decodierung nicht mehr länger getrennt betrachtet werden. Der optimale Empfänger führt demnach eine *Maximum-Likelihood*-Sequenzschätzung durch, indem er diejenige Symbolfolge $\hat{\mathbf{x}} = (x_0 \cdots x_{N-1})$ der Länge N bestimmt, die zur empfangenen Sequenz $\mathbf{y} = (y_0 \cdots y_{N-1})$ die geringste quadratische euklidische Distanz besitzt.

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(\|\mathbf{y} - \mathbf{x}\|^2 \right) \tag{2.12}$$

Ziel der Codierten Modulation muss es also sein, die Zuordnung von Codeworten auf die Kanalsymbole derart zu gestalten, dass die möglichen Sequenzen eine maximale euklidische Distanz untereinander aufweisen. Ferner

wird durch Gl. (2.12) deutlich, dass der Faltungscodierer ein Gedächtnis einbringt, so dass am Empfänger ganze Symbolfolgen betrachtet werden müssen.

Mit anderen Worten:
Die minimale quadratische euklidische Distanz

$$\Delta_f^2 = \min_{\mathbf{x}^{(1)},\mathbf{x}^{(2)}} d_e(\mathbf{x}^{(1)},\mathbf{x}^{(2)}) = \min_{\mathbf{x}^{(1)},\mathbf{x}^{(2)}} \|\mathbf{x}^{(1)} - \mathbf{x}^{(2)}\|^2 \tag{2.13}$$

zwischen zwei beliebigen Symbolfolgen $\mathbf{x}^{(1)}$ und $\mathbf{x}^{(2)}$ muss maximiert werden.

Damit liegt ein anderes Optimierungsproblem vor als im Fall der reinen Kanalcodierung. Dort kam es darauf an, die minimale Hammingdistanz bzw. die freie Distanz, d.h. die kleinste Anzahl unterschiedlicher Symbole einer Sequenz zu maximieren. Lediglich für den binären Fall führen beide Ansätze zur identischen Ergebnissen, da hier die Anzahl unterschiedlicher Symbole direkt die euklidische Distanz bestimmt.

Anmerkung zu Schwundkanälen
Es sei an dieser Stelle darauf hingewiesen, dass das obige Kriterium, die minimale euklidische Distanz zwischen den Sequenzen zu maximieren, nur für den AWGN-Kanal zu optimalen Ergebnissen führt. Bei Schwundkanälen, die im Bereich der Mobilfunkübertragung fast immer vorliegen, müssen andere Strategien gewählt werden.

Im folgenden Beispiel werden noch einmal Rice-, Rayleigh- und AWGN-Kanal gegenübergestellt. Ohne die einzelnen Optimierungsansätze herzuleiten, sollen sie an dieser Stelle kurz erläutert werden. Der Rice-Kanal

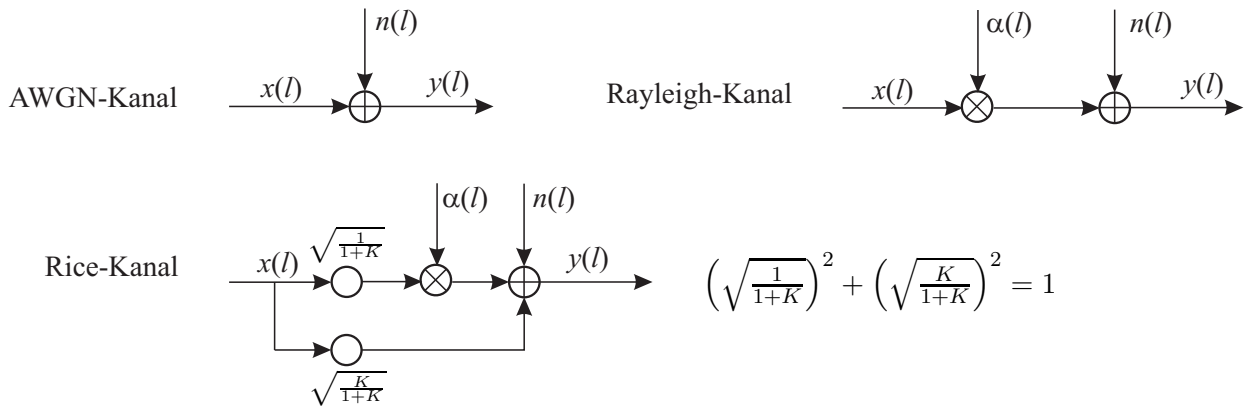


Bild 2.11: Darstellung einiger nicht-frequenzselektiver Kanalmodelle

stellt im Prinzip den allgemeinsten der drei aufgeführten Kanäle dar. Er berücksichtigt neben den schon vom Rayleigh-Kanal bekannten gestreuten Komponenten auch einen direkt ankommenden Signalanteil, wobei der Parameter K das Leistungsverhältnis zwischen direkter und gestreuten Komponenten angibt. Für den Extremfall $K = 0$ existiert keine direkte Komponente, wir erhalten den reinen Rayleigh-Kanal. Für $K \to \infty$ existiert dagegen nur eine direkte Komponente und wir erhalten den AWGN-Kanal.

Optimierungsstrategien:

$K = 0$ Reiner Rayleigh-Kanal:
1) Die Länge (Anzahl Symbole) der kürzesten Fehlersequenz muss maximiert werden.
2) Das Produkt der Pfaddistanzen entlang dieses Pfades ist zu maximieren.

$K > 0$ Rice-Kanal:
Die Reihenfolge der Kriterien 1) und 2) kehrt sich mit wachsendem K allmählich um.

$K \to \infty$ AWGN-Kanal:
Kleinste euklidische Distanz ist zu maximieren.

Aufgrund der Komplexität der Thematik und der Vielzahl von verschiedenen Spezialfällen wird im weiteren Verlauf lediglich der AWGN-Kanal und die mit ihm verbundenen Optimierungen betrachtet.

2.3.3 Informationstheoretische Betrachtung

Um grundsätzlich zu klären, wie groß der theoretische Gewinn durch Vergrößerung des Signalraumalphabets werden kann, kann die Kanalkapazität herangezogen werden. In Kapitel 2 des letzten Semesters wurde die Gleichung zur Berechnung der Kanalkapazität C bei diskretem Eingangs-, aber kontinuierlichem Ausgangsalphabet (AWGN), berechnet:

$$C = 2^{-k} \cdot \int_{\mathcal{A}_{out}} \sum_{\nu} p_{y|x}(\vartheta|x = X_\nu) \cdot \log_2 \frac{p_{y|x}(\vartheta|x = X_\nu)}{2^{-k} \cdot \sum_l p_{y|x}(\vartheta|x = X_l)} \, d\vartheta. \tag{2.14}$$

Basierend auf den Signalraumalphabeten der bisher betrachteten Modulationsverfahren liefert Gl. (2.14) für den AWGN-Kanal die in Bild 2.12 dargestellten Ergebnisse.

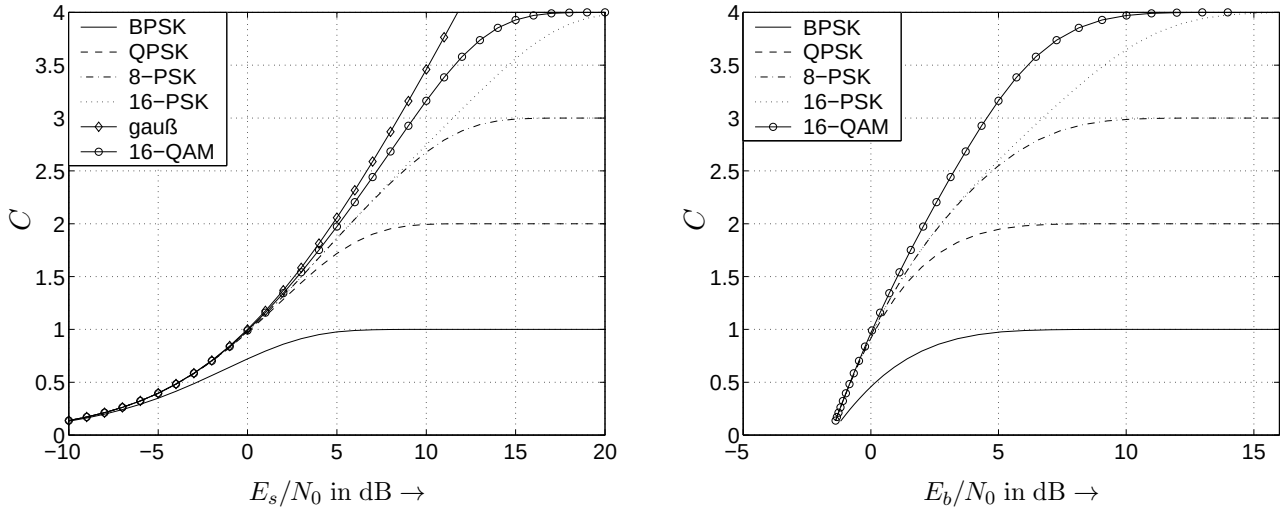


Bild 2.12: Kapazität für den AWGN-Kanal und verschiedene Modulationsverfahren

Interpretation:

- Darstellung über E_s/N_0:
  - Mit wachsendem M nimmt auch Kanalkapazität zu
  - Für sinkenden Signal-Rausch-Abstand ($E_s/N_0 \to -\infty$) strebt die Kapazität gegen Null
  - Asymptotisch ($E_s/N_0 \to \infty$) strebt die Kanalkapazität gegen m, also die spektrale Effizienz
  - Kontinuierlich gaußverteilte Eingangssignale weisen die maximal mögliche Kapazität auf
  - 16-QAM besitzt größere Kapazität als 16-PSK (sie nutzt den Signalraum effizienter aus); asymptotisch erreichen natürlich beide die gleiche spektrale Effizienz $\eta = 4$ bit/s/Hz
- Darstellung über E_b/N_0:
  - Bedingte Wahrscheinlichkeit $p(\mathbf{y}|\mathbf{x})$ hängt von E_s/N_0 ab
  - Es gilt aber $C = f(E_s/N_0) = f(R_c \cdot E_b/N_0)$

  $\longrightarrow$ Soll die Kanalkapazität voll ausgeschöpft werden, erhält man für $R_c = C$ eine implizite Gleichung, die Punkt für Punkt nach E_b/N_0 aufgelöst werden muss.
  - Absolute Grenze für fehlerfreie Übertragung aus letztem Semester noch bekannt: $E_b/N_0 \approx -1.59$ dB (unabhängig vom Modulationsverfahren)

$\Longrightarrow$ Für $E_b/N_0 < -1,59$ dB ist auch mit noch so großem Aufwand keine fehlerfreie Übertragung mehr möglich!

– Qualitative Verhältnisse ändern sich im Vergleich zu (E_s/N_0) nicht.

Zusammenfassend kann festgestellt werden, dass alle Modulationsverfahren für große Signal-Rausch-Abstände eine Kapazität erreichen, die ihrer spektralen Effizienz entspricht. Dies ist einleuchtend, da mit einer 8-PSK nie mehr als 3 Bit je Symbol übertragen werden können.

Im Folgenden soll nun der Übergang von einer uncodierten QPSK auf eine 2/3-ratig codierte 8-PSK untersucht werden. Mit Hilfe von Gl. (2.6) bzw. Bild 2.6 kann gezeigt werden, dass für die uncodierte QPSK ein Signal-Rausch-Abstand von ca. 9,5 dB erforderlich ist, um eine Bitfehlerrate von $P_b = 10^{-5}$ zu erreichen. Die spektrale Effizienz beträgt dabei $\eta = 2$ bit/s/Hz. Aus Bild 2.12 ist hingegen ersichtlich, dass eine fehlerfreie Übertragung mit $\eta = 2$ bit/s/Hz bei einer 8-PSK schon bei $E_b/N_0 = 2,5$ dB möglich ist. Demnach ist theoretisch ein Gewinn von etwa 7 dB möglich, wenn man von der uncodierten QPSK auf eine 2/3-ratig codierte 8-PSK übergeht. Der direkte Ansatz zur codierten Modulation erreichte nur einen Gewinn von 0,7 dB und kann demnach als sehr schlecht angesehen werden. Codierer und Modulator müssen vielmehr zu einer Einheit verschmelzen, um die euklidischen Distanzen der möglichen Symbolfolgen zu optimieren.

Außerdem ersichtlich:

Eine stärkere Vergrößerung des Signalraumalphabets als eine Verdopplung bringt keinen zusätzlichen Gewinn mehr

$\Longrightarrow$ **Verdopplung von** M **auf** $2M$ **bzw. von** m **auf** $m+1$ **reicht aus!**

$\Longrightarrow$ **Codes der Raten** $R_c = m/m+1$ **kommen zum Einsatz!**

2.4 TCM nach Ungerböck

2.4.1 Trellisrepräsentation

Wie aus dem Namen der Trelliscodierten Modulation schon hervorgeht, lässt sich dieser Ansatz zur codierten Modulation mit Hilfe eines Trellisdiagramms graphisch darstellen. Dies entspricht auch der Beschreibung von Faltungscodes, die in den meisten Fällen auch als Kanalcodierungsverfahren eingesetzt werden. Kommen Blockcodes zum Einsatz, spricht man von *Blockcodierter Modulation*.

Im Folgenden soll der Vorteil der Verknüpfung von Codierung und Modulation anhand einiger Beispiele veranschaulicht werden. Dazu betrachten wir das schon öfters verwendete Beispiel der uncodierten QPSK und der 2/3-ratig codierten 8-PSK. Als Maß des Gewinns dient das Verhältnis der minimalen quadratischen euklidischen Distanzen des Sequenzen

$$\gamma = \frac{\Delta_f^2(\text{8-PSK-TCM})}{\Delta_f^2(\text{QPSK})}\,, \tag{2.15}$$

da diese im wesentlichen die Leistungsfähigkeit beeinflussen (s. Abschnitt 2.2).

Beispiel: Uncodierte QPSK im Vergleich zu 2/3-ratig codierter 8-PSK

- Bei der uncodierten QPSK werden 2 Informationsbit je Symbol übertragen ($\eta = 2$ bit/s/Hz).
- Gleiche spektrale Effizienz bei 8-PSK bedingt 2 Infobit und 1 Prüfbit je 8-PSK-Symbol (Codierung mit $R_c = 2/3$)
- **Die Symbolenergie ist in allen Fällen auf 1 normiert (**$E_s = T_s$**).**

Uncodierte QPSK

Bei einer uncodierten Übertragung sind aufeinanderfolgende Symbole statistisch unabhängig voneinander, d.h. alle Symbolabfolgen sind innerhalb einer Sequenz möglich. Diese Struktur kann entsprechend Bild 2.13 durch ein Trellisdiagramm mit nur einem Zustand (Gedächtnis Null) dargestellt werden. Die minimale quadratische euklidische Distanz zwischen zwei Sequenzen ist somit durch die minimale quadratische euklidische Distanz zwischen zwei Symbolen bestimmt.

$$\Delta_f^2 = \min_{\mathbf{x}^{(1)},\mathbf{x}^{(2)}} d_e^2(\mathbf{x}^{(1)},\mathbf{x}^{(2)}) = \Delta_0^2 = 2$$

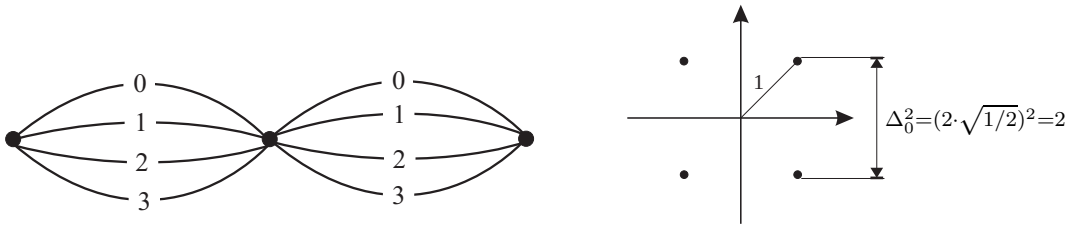


Bild 2.13: Trellisdiagramm mit 1 Zustand

Codierte 8-PSK mit 2 Zuständen

Durch die Vergrößerung des Gedächtnisses auf Eins ergibt sich ein Trellisdiagramm mit zwei Zuständen (s. Bild 2.14), wobei mögliche Symbolfolgen durch Pfade im Trellis dargestellt werden. Die Symbole der 8-PSK seien entgegen dem Uhrzeigersinn fortlaufend numeriert! Durch die Codierung sind nun nicht mehr alle Symbolfolgen möglich, so dass sich die euklidische Distanz zwischen den Sequenzen erhöht. Da 2 Informationsbit je Symbol übertragen werden, gibt es vier Übergänge von und zu jedem Zustand. Bei 2 Zuständen treten damit auch parallele Übergänge auf, weshalb die kürzeste Abweichung zwischen 2 Sequenzen aus 2 parallelen Zweigen besteht.

Intuition: Euklidische Distanz zwischen parallelen Zweigen muss maximiert werden!

Aus diesem Grund werden parallelen Übergängen gegenüberliegende Symbole ($\Delta_3^2 = 4$) zugeordnet. Zur Bestimmung der minimalen euklidischen Distanz zweier Symbolfolgen sind in Bild 2.14 für parallele und nicht-parallele Pfade die Sequenzen mit der geringsten euklidischen Distanz zueinander fett hervorgehoben.

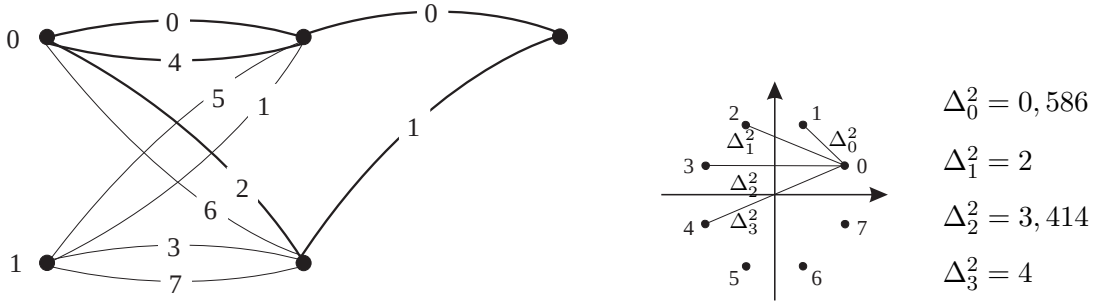


Bild 2.14: Trellisdiagramm mit 2 Zuständen

- Parallele Zweige: $d_{ep}^2 = \Delta_3^2 = 4$
- Zustandsfolge 0-1-0: $d_{ef}^2 = d_e^2(0,2) + d_e^2(0,1) = \Delta_1^2 + \Delta_0^2 = 2 + 0,586 = 2,586$

Ausschlaggebend ist das Minimum der beiden Distanzen. Der Gewinn gegenüber der uncodierten QPSK beträgt

$$\boxed{\gamma = \frac{\Delta_f^2}{\Delta_0^2} = \frac{\min(d_{ep}^2, d_{ef}^2)}{\Delta_0^2} = \frac{2,586}{2} = 1,293 \mathrel{\hat{=}} 1,12 \text{ dB}}\,.$$

Der Gewinn von 1,12 dB ist noch weit vom optimalen informationstheoretischen Wert (7 dB) entfernt. Eine weitere Verbesserung kann durch die Erhöhung der Anzahl an Zuständen im Trellisdiagramm erreicht werden

Codierte 8-PSK mit 4 Zuständen

Bild 2.15 zeigt die Darstellung durch das Trellisdiagramm. Zur Berechnung der quadratischen euklidischen Distanzen sind wiederum parallele und nicht-parallele Sequenzen zu unterscheiden.

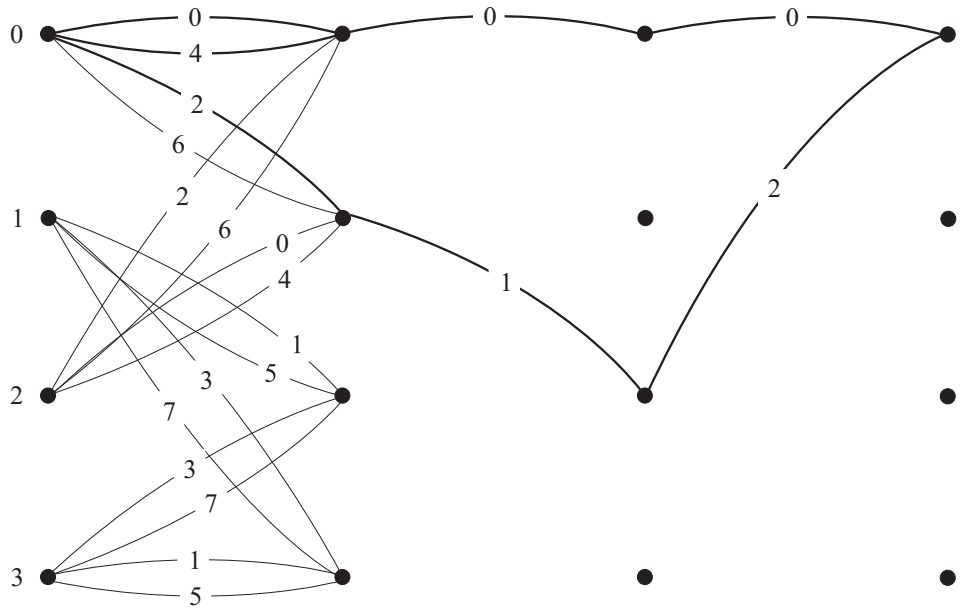


Bild 2.15: Trellisdiagramm mit 4 Zuständen

- Parallele Übergänge: $d_{ep}^2 = \Delta_3^2 = 4$
- Zustandsfolge 0 - 1 - 2 - 0:

$$d_{ef}^2 = d_e^2(0,2) + d_e^2(0,1) + d_e^2(0,2) = \Delta_1^2 + \Delta_0^2 + \Delta_1^2 = 2 + 0,586 + 2 = 4,586$$

Der Gewinn gegenüber der uncodierten QPSK beträgt somit

$$\boxed{\gamma = \frac{\min(d_{ep}^2, d_{ef}^2)}{\Delta_0^2} = \frac{4}{2} = 2 \mathrel{\hat{=}} 3 \text{ dB}}\,.$$

Der Gewinn ist jetzt durch die euklidische Distanz paralleler Übergänge bestimmt, da sie die minimal mögliche Distanz zwischen zwei Symbolfolgen festlegen. Daher müssen für weitere Verbesserungen parallele Zweige vermieden werden, was z.B. durch eine Erhöhung der Anzahl von Zuständen oder ein anderes Modulationsverfahren geschehen kann.

Codierte 8-PSK mit 8 Zuständen

Bild 2.16 zeigt das Trellisdiagramm für ein Gedächtnis von drei. Das erste Symbol an jedem Knoten gehört zum obersten Pfad, das zweite zum zweit obersten usw. Es werden folgende Distanzen für parallele und nicht-parallele Sequenzen erzielt.

- Parallele Übergänge existieren nicht mehr
- Zustandsfolge 0 - 1 - 2 - 0:

$$d_{ef}^2 = d_e^2(0,6) + d_e^2(0,7) + d_e^2(0,6) = \Delta_1^2 + \Delta_0^2 + \Delta_1^2 = 2 + 0,586 + 2 = 4,586$$

Damit beträgt der Gewinn gegenüber der uncodierten QPSK

$$\boxed{\gamma = \frac{d_{ef}^2}{\Delta_0^2} = \frac{4,586}{2} = 2,293 \mathrel{\hat{=}} 3,6 \text{ dB}} .$$

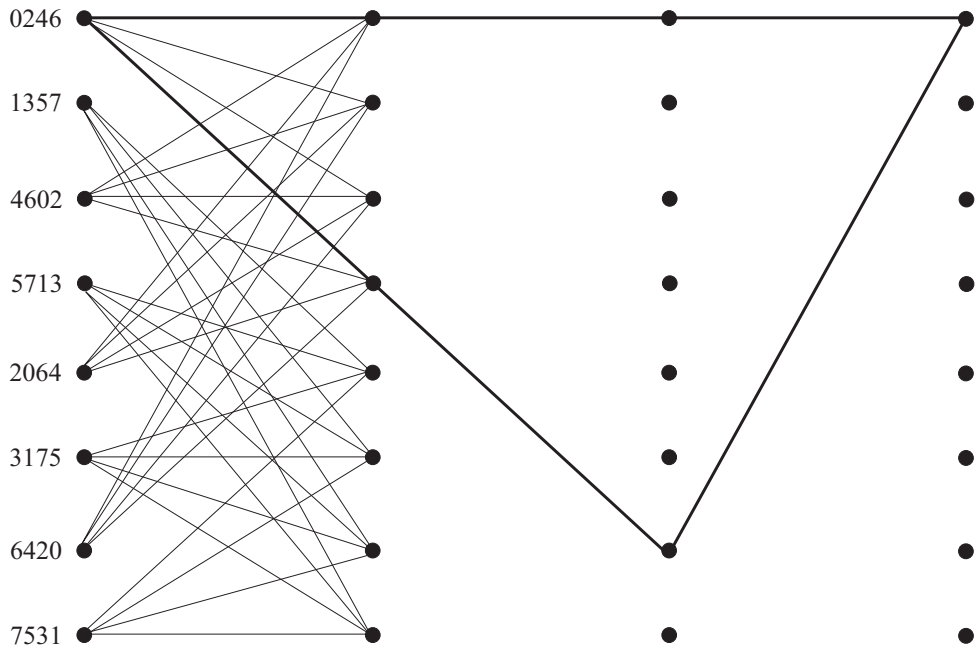


Bild 2.16: Trellisdiagramm mit 8 Zuständen

Aus den vorangegangenen Beispielen ist ersichtlich, dass sich das Einbringen eines Gedächtnisses gewinnbringend auswirkt. Es können nicht mehr alle Symbolkombinationen in einer Sequenz auftreten, wodurch sich die Distanzen zwischen den gültigen Symbolfolgen erhöhen, was schließlich in einem Gewinn gegenüber dem uncodierten Fall resultiert. Wie auch bei den Faltungscodes führt eine Erhöhung der Anzahl der Zustände zu einer Vergrößerung der Leistungsfähigkeit, natürlich auch verbunden mit einem höheren Decodieraufwand. Der betrachtete Gewinn stellt sich in der Praxis erst asymptotisch (für sehr große Signal-Rausch-Abstände) ein. Für genauere Abschätzungen ist wie auch bei Block- und Faltungscodes das gesamte Distanzspektrum zu betrachten.

Frage: Kann eine Systematik gefunden werden, mit der die Konstruktion von optimalen TCM-Codierern möglich ist?

Antwort: Nein!
Es sind (für den AWGN-Kanal) optimale Strukturen durch systematische Rechnersuche gefunden worden, nicht aber durch geeignete Kostruktionsvorschriften. Allerdings gibt es ein paar heuristisch motivierte Regeln, die bei der Suche helfen, aber keine Garantie für ein optimales Ergebnis gewähren.

2.4.2 *Set-Partitioning* nach Ungerböck

Ziel des *Mapping by Set-Partitioning* von Ungerböck ist es, die Distanzeigenschaften des TCM-Codes zu optimieren. Dabei müssen parallele Zweige (soweit vorhanden) mit weit auseinander liegenden Symbolen korrespondieren, wohingegen 'normale' Zweige, die zu unterschiedlichen Zuständen führen, durch die Trellisstruktur besser geschützt sind und dichter zusammen liegenden Symbolen zugeordnet werden können. Dies führt zur Strategie der sukzessiven Zerlegung des Signalraums in Teilmengen, wie es in Bild 2.17 dargestellt ist.

1. Beginne mit dem Gesamtsignalraum $\mathbf{B} = \mathcal{A}_{in}$
2. Teile $\mathbf{B}$ in 2 Teilmengen $\mathbf{B}_0^{(1)}$ und $\mathbf{B}_1^{(1)}$, so dass sich die euklidischen Distanzen zwischen Symbolen innerhalb der Teilmengen vergrößern
3. Wiederhole die obige Zerlegung mit $\mathbf{B}_0^{(1)}$ und $\mathbf{B}_1^{(1)}$ und den aus ihnen gebildeten Teilmengen $\mathbf{B}_0^{(2)}$, $\mathbf{B}_1^{(2)}$, $\mathbf{B}_2^{(2)}$ und $\mathbf{B}_3^{(2)}$ solange, bis nur noch 1 Symbol je Mengen enthalten ist.

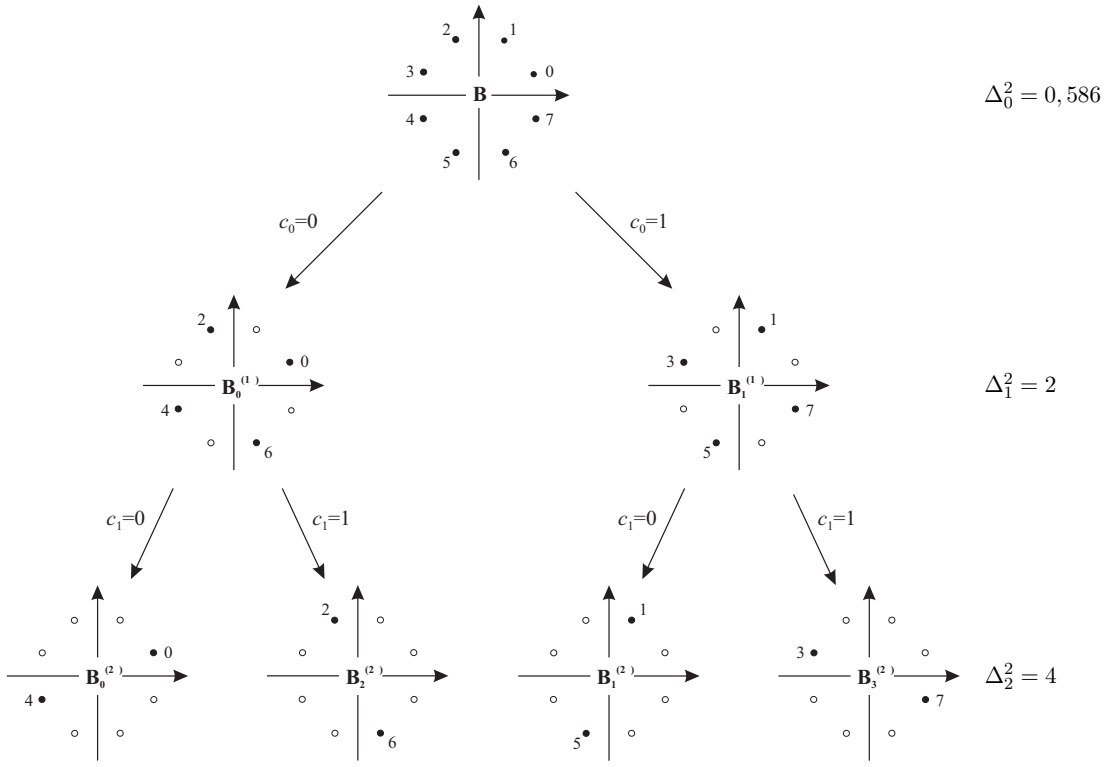


Bild 2.17: *Set-Partitioning* nach Ungerböck für 8-PSK

Mit jedem Partitionierungsschritt nimmt die kleinste quadratische euklidische Distanz innerhalb der Teilmengen zu, so dass sie von $\Delta_0^2 = 0,586$ über $\Delta_1^2 = 2$ auf $\Delta_2^2 = 4$ anwächst. Im allgemeinen werden $m' = m + 1$ Partitionierungsschritte ausgeführt, so dass die Mengen der untersten Partitionierungsebene nur noch 1 Element enthalten. In Bild 2.18 wird das Prinzip des *Set-Partitioning* auch für eine 16-QAM demonstriert.

Die Entscheidung, wie sich die Partitionierung auf die Symbolzuordnung für einen konkreten Vektor $\mathbf{c}$ auswirkt, ist die nächste zu lösende Frage. Dabei ist festzuhalten, dass das Teilsystem bis zum Signalraumcodierer linear ist, während die Signalraumzuordnung nichtlinear ist. Es gilt

$$x(l) = f\left(\mathbf{c}(l)\right) = f\left(c_0(l)\; c_1(l)\; \cdots\; c_m(l)\right), \tag{2.16}$$

wobei $f()$ eine nichtlineare Funktion ist. Ein allgemeiner gefaßter Ansatz von Calderbank und Mazo beschäftigt sich intensiv mit Eigenschaften der Zuordnungsfunktion $f()$ [BDMS91].

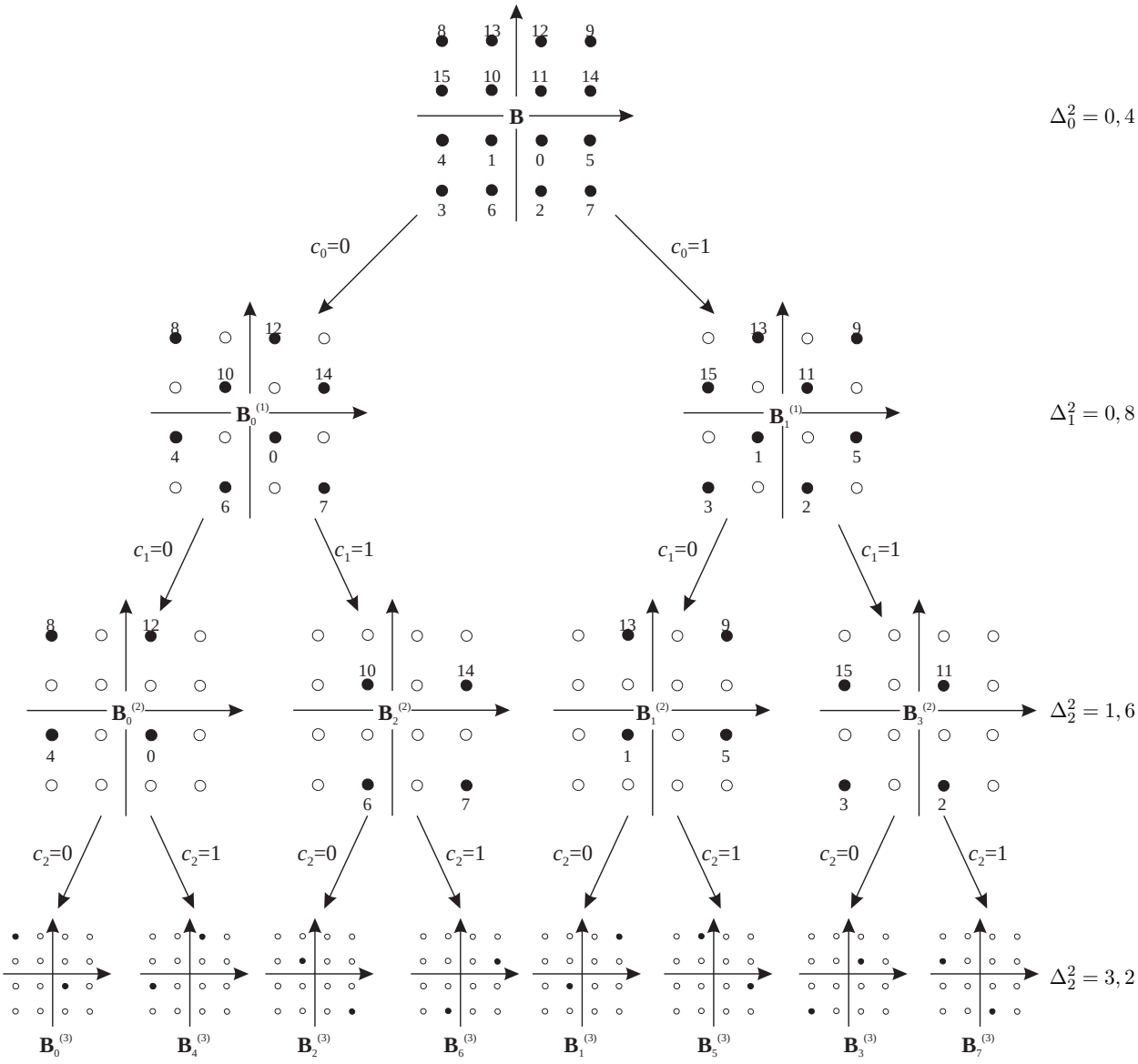


Bild 2.18: *Set-Partitioning* nach Ungerböck für 16-QAM

Natürliche Zuordnung

Unter der *natürlichen* Zuordnung versteht man die Interpretation des Vektors $\mathbf{c}$ als umgekehrte Dualzahl $(c_m \ \ldots \ c_0) \ \rightarrow \ c_m \cdot 2^m + \ \cdots \ + c_0$, die Symbole werden dann einfach durchnumeriert (s. Bild 2.17). So entspricht das Symbol '1' der (0 0 1), das Symbol '3' der (0 1 1) sowie die '6' der (1 1 0). Die Durchnumerierung der Symbole in den einzelnen Teilmengen beeinflußt dabei nicht die Leistungsfähigkeit der TCM, da die Distanzeigenschaften der Sequenzen erhalten bleiben. Es wird hierdurch lediglich eine andere Zuordnung der Informationssequenz auf die Kanalsymbolfolge realisiert.

2.4.3 Struktur des TCM-Codierers

Aus der informationstheoretischen Betrachtung ist noch bekannt, dass eine Verdopplung des Signalraumalphabets ($m \rightarrow m+1$) ausreicht und eine weitere Vergrößerung keinen nennenswerten Gewinn mehr ergibt. Somit sind lediglich Codierer der Rate $R_c = k/k+1$ zu betrachten. Allgemein ergibt sich nach Ungerböck folgenden Struktur des TCM-Codierers (Bild 2.19).

- Beim TCM-Codierer liegen m Informationsbit am Eingang an, von denen k Bit $(u_1 \ \ldots \ u_k)$ durch

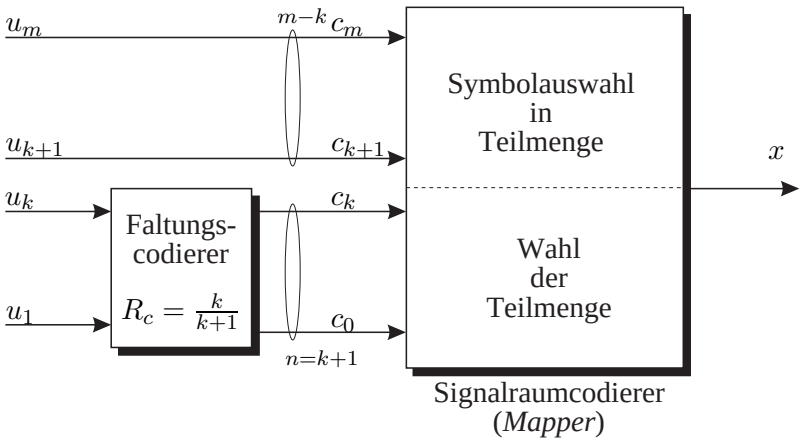


Bild 2.19: Allgemeiner Aufbau eines TCM-Codierers

Faltungscodierer mit $R_c = k/k+1$ codiert werden

- Die restlichen $m-k$ Bit bleiben uncodiert ($u_{k+1} = c_{k+1}, \cdots, u_m = c_m$)
- Zusammen mit den $k+1$ codierten Werte $(c_0 \ \cdots \ c_k)$ liegen insgesamt $m+1$ Bit am Eingang des Signalraumcodierers an
- Die Gesamtrate beträgt $m/m+1$
- Spezialfälle:
  $k = m$ Alle Informationsbit werden codiert $\longrightarrow$ Faltungscode mit $R_c = m/(m+1)$
  $k = 1$ Faltungscode der Rate $R_c = 1/2$ codiert nur 1 Informationsbit, die restlichen $m-1$ Bit bleiben uncodiert

Für den AWGN-Kanal muss das Ziel aller Optimierungen sein, die minimale quadratische euklidische Distanz zu maximieren. Die Frage nach einer optimalen Struktur betrifft 2 Bereiche, die nur gemeinsam optimiert werden können. Zum Einen sind die zum Einsatz kommenden Faltungscodes an das neue Kriterium anzupassen. Die im letzten Semester vorgestellten Codes waren hinsichtlich ihrer freien Distanz d_f optimiert worden, hier gelten andere Randbedingungen, so dass die dort aufgeführten Codes hier nicht unbedingt die beste Wahl darstellen.

Der zweite Aspekt betrifft den Signalraumcodierer. Hier stellt sich die Frage, wie die 2^{m+1} verschiedenen Vektoren $\mathbf{c} = (c_0 \ \cdots \ c_m)$ auf die Symbole x_l abgebildet werden sollen. Soll die euklidische Distanz zwischen Sequenzen im Trellis maximiert werden, ist bei der Wahl einer konkreten Zuordnungsvorschrift auch die Struktur des Faltungscodes zu berücksichtigen.

An dieser Stelle soll jedoch nicht auf effiziente Suchalgorithmen eingegangen werden. Sie können in der Literatur [Ung82, Ung87] nachgelesen werden. Bei der Suche nach einer optimalen Zuordnungsvorschrift sind die folgenden heuristisch motivierten Richtlinien hilfreich:

1. Uncodierte Binärstellen $(u_{k+1} \ \ldots \ u_m)$ bestimmen Symbole aus den $m-k$ untersten Partitionierungsmengen $\longrightarrow$ Symbole sind hier am weitesten voneinander entfernt
2. Zweige, die im gleichen Zustand entspringen oder in den gleichen Zustand münden, werden Symbolen der gleichen Teilmenge zugeordnet
3. Alle Symbole des Signalraums werden gleich häufig benutzt.

Die Berücksichtigung dieser Regeln führt zwar nicht zu einer eindeutigen Lösung, sie liefern jedoch wichtige Anhaltspunkte auf dem Weg zu einer optimalen Struktur. Hierdurch schränkt sich die Anzahl möglicher Konfigurationen schon merklich ein, was die Suche vereinfacht. Der erste Punkt gewährleistet, dass uncodierte Bit den größtmöglichen Schutz erhalten, da sie Symbolen mit größtmöglichem euklidischen Abstand zugeordnet werden.

2.4.4 Optimale Codes nach Ungerböck

Aus dem letzten Semester ist bereits bekannt:

- RSC-Codes: Systematische und rekursive Faltungscodes
- NSC-Codes: Nicht-systematische und nicht-rekursive Faltungscodes

Beide Codearten besitzen identische Distanzeigenschaften und lassen sich ineinander überführen. Ein prinzipieller Unterschied besteht zum einen darin, dass RSC-Codes eine unendlich lange Impulsantwort besitzen, während sie bei NSC-Codes endlich ist. Außerdem können die Informationsbit bei RSC-Codes im fehlerfreien Fall direkt - ohne Decodierung - aus der empfangenen Sequenz abgelesen werden, dies ist bei NSC-Codes nicht möglich.

Aufgrund der Äquivalenz beider Codes können sie natürlich auch gleichwertig in ein TCM-Schema nach Bild 2.19 eingesetzt werden. In der Regel erhalten jedoch die rekursiven systematischen Codes den Vorzug. Eine allgemeine Form zeigt dann Bild 2.20.

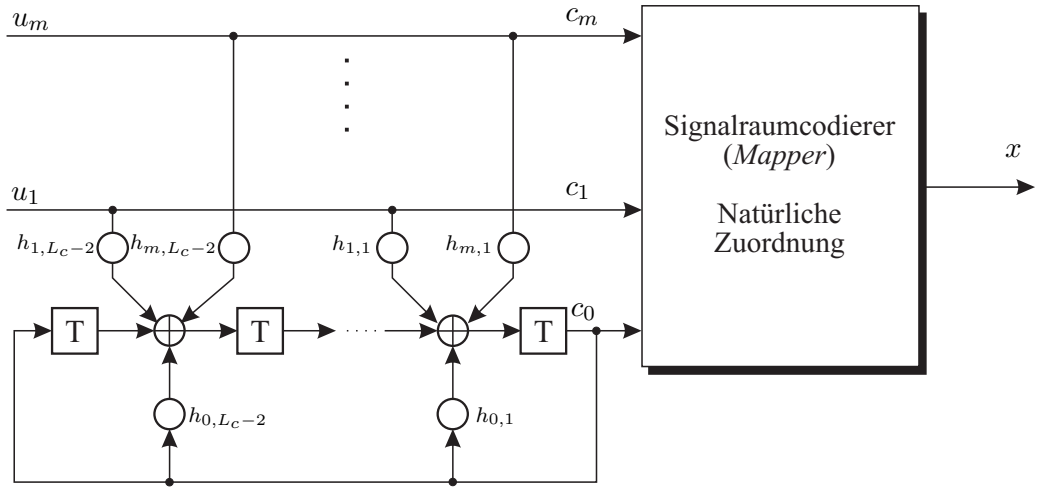


Bild 2.20: Allgemeiner Aufbau eines systematischen TCM-Codierers mit RSC-Code

Wie schon erläutert liegen m Informationsbit $(u_1 \ \ldots \ u_m)$ am Eingang an, die dann über Generatorkoeffizienten $h_{i,j}$ mit den modulo-2-Addierern des Faltungscodes verbunden werden. Die Struktur des Faltungscodes ist zunächst etwas ungewöhnlich, da die Addierer zwischen den Speicherelementen angeordnet sind. Sie erinnert aber an die Realisierung der Schieberegisterstruktur von Blockcodes mit Hilfe des Generatorpolynoms (im Gegensatz zur Realisierung über das Prüfpolynom).

Sollen $m-k$ Bit $(u_{k+1} \ \ldots \ u_m)$ uncodiert bleiben, d.h. nicht den Ausgang des Faltungscodes beeinflussen, sind die entsprechenden Koeffizienten $h_{k+1,i}$ bis $h_{m,i}$ für alle $i = 1 \ \ldots \ L_c - 2$ zu Null zu setzen. Durch die Wahl eines RSC-Codes liegen alle Informationsbit direkt am Signalraumcodierer an, mit Hilfe der rekursiven Struktur wird das zusätzliche Prüfbit erzeugt. Die konkrete Wahl der Koeffizienten $h_{i,j}$ ist kein triviales Problem und lässt sich nur mit Hilfe aufwendiger Suchalgorithmen lösen, die als Optimierungskriterium die schon erwähnten Distanzbetrachtungen besitzen. Eine Auflistung der wichtigsten von Ungerböck gefundenen

Konfigurationen, die optimale TCM-Strukturen für den AWGN-Kanal darstellen, enthalten die folgenden der Literatur entnommen Tabellen.

Tabellen mit den wichtigsten optimalen Ungerböck-TCM-Codes für verschiedene Modulationsverfahren

Anhand der Tabellen 2.1 und 2.2 ist zu erkennen, dass mit zunehmender Anzahl von Zuständen (Registerlänge) die euklidische Distanz zwischen möglichen Symbolfolgen zunimmt. Generell lassen sich Unterschiede zwischen 8-PSK und 16-PSK feststellen. Bei der 8-PSK ist die minimale euklidische Distanz für 4 Zustände noch durch die parallelen Übergänge bestimmt (*). Danach treten keine parallelen Übergänge mehr auf, die kleinste vorkommende euklidische Distanz ist jetzt größer als die gegenüberliegender Symbole ($\Delta_f^2 > 4$). Die 16-PSK verhält sich anders. Für kleine Registerlängen dominieren zunächst Sequenzen mit einer euklidischen Distanz, die kleiner ist als die paralleler Übergänge ist. Dies liegt an der dichten Anordnung der Symbole im Signalraum. Für 64 und 128 Zustände sind unterschiedliche Sequenzabschnitte dann so lang, dass ihre euklidische Distanz größer als die der parallelen Übergänge wird (*). Dann dominieren letztere mit $\Delta_f^2 = 2$ die Fehlerrate. Erst ab 256 Zuständen treten keine parallelen Übergänge mehr auf, die minimale Distanz wird größer als 2.

Anzahl Zustände	k	h_0	h_1	h_2	Δ_f^2	$G_{8-PSK/QPSK}$ [dB]	Gewinn bei $P_b = 10^{-5}$ [dB]
4	1	5	2		4,000*	3,01	2,4
8	2	11	02	04	4,586	3,60	2,8
16	2	23	04	16	5,172	4,13	3,0
32	2	45	16	34	5,758	4,59	3,3
64	2	103	030	066	6,343	5,01	3,6
128	2	277	054	122	6,586	5,17	
256	2	435	072	130	7,515	5,75	

Tabelle 2.1: TCM-Konfiguration für 8-PSK-TCM nach Ungerböck für verschiedene Anzahl von Zuständen [Fri96]

Anzahl Zustände	k	h_0	h_1	h_2	Δ_f^2	$G_{16-PSK/8-PSK}$ [dB]	Gewinn bei $P_b = 10^{-5}$ [dB]
4	1	5	2		1,324	3,54	2,3
8	1	13	04		1,476	4,01	2,7
16	1	23	04		1,628	4,44	2,9
32	1	45	10		1,910	5,13	3,2
64	1	103	024		2,000*	5,33	3,5
128	1	203	024		2,000*	5,33	
256	2	427	176	374	2,085	5,51	

Tabelle 2.2: TCM-Konfiguration für 16-PSK-TCM nach Ungerböck für verschiedene Anzahl von Zuständen [Fri96]

Im Folgenden sollen einige Beispiele die Struktur des TCM-Codierers veranschaulichen. Dabei verwenden wir die schon bekannten Beispiele aus den Bildern 2.14 bis 2.16 und zeigen die zugehörigen Codierer. Das Trellisdiagramm aus Bild 2.14 gehört beispielsweise zu der in Bild 2.21 gezeigten Codiererstruktur. Es wird deutlich, dass parallele Zweige immer gegenüberliegenden Symbolen (0,4) oder (2,6) zugeordnet werden. Welches der beiden Symbole gesendet wird, steuert das uncodierte Bit c_2. Die Bit c_0 und c_1 bestimmen hingegen, welcher Teilmenge die beiden Symbole entnommen werden. So weisen alle von Zustand 0 ausgehenden Zweige gerade Symbolnummern auf, während den Pfaden von Zustand 1 die ungeraden Symbolnummern zugeordnet sind. Ausschlaggebend hierfür ist c_0, dass in diesem einfachen Fall direkt den Zustand des Codierers angibt.

Die Bilder 2.22 und 2.23 zeigen die Strukturen für 4 bzw. 8 Zustände. In dieser systematischen Form kommen jetzt rekursive Codes zum Einsatz, es existieren allerdings äquivalente, rückkopplungsfreie Strukturen, bei de-

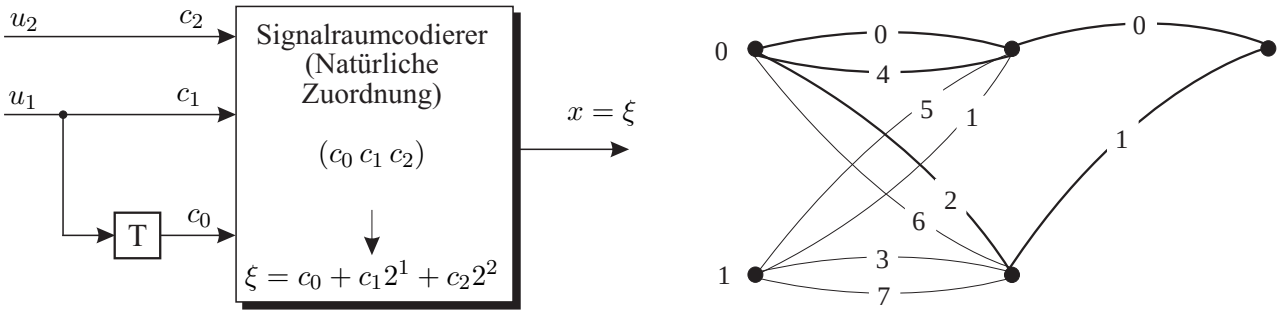


Bild 2.21: Optimaler TCM-Codierers für 8-PSK mit 2 Zuständen und Trellisdiagramm

nen der Vektor $\mathbf{c}$ die Informationsbit $\mathbf{u}$ nicht mehr explizit enthält. Man erkennt weiterhin, dass bei 4 Zuständen ein Informationsbit uncodiert bleibt (parallele Übergänge im Trellisdiagramm).

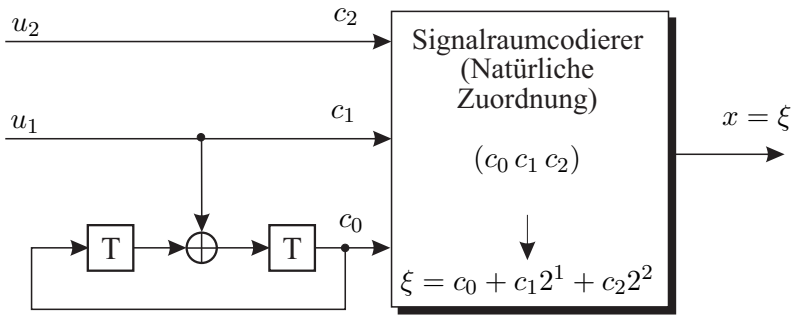


Bild 2.22: Optimaler TCM-Codierers für 8-PSK mit 4 Zuständen

Bei 8 Zuständen gibt es dann genügend mögliche Übergänge, dass keine parallelen Zweige mehr vorkommen. Hier sind entsprechend Bild 2.23 beide Informationsbit an der Codierung beteiligt, es gibt hier keine uncodierten Bit mehr.

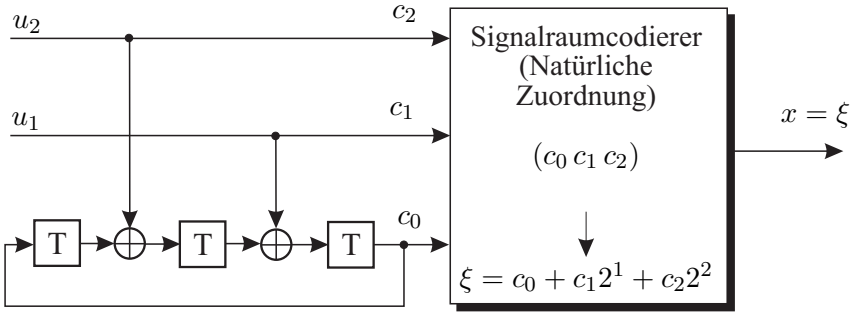


Bild 2.23: Optimaler TCM-Codierers für 8-PSK mit 8 Zuständen

2.5 ML-Decodierung mit dem Viterbi-Algorithmus

Das Prinzip der *Maximum Likelihood*-Decodierung ist schon von Block- und Faltungscodes bekannt. Es wird die Sequenz $\hat{\mathbf{x}}$ mit der minimalen euklidischen Distanz zur empfangenen Sequenz $\mathbf{y}$ gesucht. Daher ist das Bestreben bei der Optimierung der TCM auch, die minimale euklidische Distanz zwischen den Symbolfolgen zu maximieren und somit die Entscheidungssicherheit zu erhöhen.

Bei der TCM wird nun nicht zunächst die Demodulation und dann die Decodierung durchgeführt (s. direkter Ansatz zur Kombination von Codierung und Modulation). Aufgrund der Verschmelzung von Codierer und Modulator im Sender erfolgt auch im Empfänger eine gemeinsame Demodulation und Decodierung. Wir sprechen deshalb von *TCM-Decodierung*. Sie umfaßt auch die uncodierten Bit eines Symbols.

Wir gehen im Folgenden davon aus, dass unser diskretes Kanalmodell den analogen Teil des Demodulators beinhaltet und somit zeitdiskrete, aber wertekontinuierliche Signale liefert. Die quadratische euklidische Distanz ergibt sich dann wie folgt:

$$\begin{aligned} d_e^2(\mathbf{x},\mathbf{y}) &= \|\mathbf{y}-\mathbf{x}\|^2 \\ &= \sum_l (y(l)-x(l))\cdot(y(l)-x(l))^* \\ &= \sum_l \left(|y(l)|^2 - x(l)\cdot y(l)^* - x(l)^*\cdot y(l) + |x(l)|^2\right) \\ &= \sum_l \left(|x(l)|^2 + |y(l)|^2 - 2\cdot \mathrm{Re}\left\{x(l)^*\cdot y(l)\right\}\right) \ . \end{aligned}$$

Die ML-Decodierung erfolgt nun nach

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} d_e^2(\mathbf{x},\mathbf{y}) \ . \tag{2.17}$$

Werden PSK-Modulationsverfahren eingesetzt, ist die Symbolenergie $E_s = |x(l)|^2$ konstant. Dann trägt der Summand $|x(l)|^2$ wie schon bei der binären Modulation im letzten Semester nicht zur Unterscheidung verschiedener Symbolfolgen bei. Gleiches gilt für den Term $|y(l)|^2$. Die Vernachlässigung beider Terme führt dann zur Korrelationsmetrik

$$\mu(\mathbf{x},\mathbf{y}) = \sum_l \mathrm{Re}\left\{x(l)^*\cdot y(l)\right\} = \sum_l x'(l)y'(l) + x''(l)y''(l) \ .$$

Die ML-Decodierung liefert für die Korrelationsmetrik ein äquivalentes Ergebnis.

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \mu(\mathbf{x},\mathbf{y}) \ . \tag{2.18}$$

Eine effiziente Realisierung der ML-Decodierung stellt wie schon bei den Faltungscodes der Viterbi-Algorithmus dar. Da sich TCM-Codes auch durch ein Trellisdiagramm beschreiben lassen, führt der TCM-Decodierer exakt die gleichen Operationen wie der Viterbi-Faltungsdecodierer aus. Der einzige Unterschied besteht in der Berechnung der Metriken, die für die TCM wie oben angegeben verwendet werden müssen.

Treten parallele Pfade zwischen Zuständen auf, so ist vorab jeweils der beste unter ihnen zu bestimmen. Nur dieser geht in die Metrikberechnung der Zustände ein, alle übrigen werden nicht weiter berücksichtigt. Diese Vorgehensweise wird auch dem *Set-Partitioning* nach Ungerböck gerecht, wonach parallelen Zweigen die uncodierten Bit zugeordnet werden, welche auch nicht mit der Trellisstruktur des Faltungscodes verknüpft sind.

2.6 Distanzeigenschaften und Abschätzung der Leistungsfähigkeit

Das Distanzspektrum von Faltungscodes ist schon aus dem letzten Semester bekannt. Zur Erinnerung sind noch einmal die drei wichtigsten Ausdrücke zur analytischen Bestimmung der Bitfehlerrate angegeben.

- Distanzspektrum
$$T(W,D,L) = \sum_w \sum_d \sum_l T_{w,d,l}\cdot W^w \cdot D^d \cdot L^l$$

- Schranke für die Bitfehlerrate (*Union Bound*)

$$P_b \leq \sum_d c_d \cdot P_d = \frac{1}{2} \cdot \sum_d c_d \cdot \operatorname{erfc}\left(\sqrt{dR_c \frac{E_b}{N_0}}\right)$$

- Koeffizient c_d aus Distanzspektrum

$$c_d = \sum_w \sum_l w \cdot T_{w,d,l}$$

Ein wesentlicher Unterschied zur TCM besteht allerdings darin, dass **Faltungscodes linear** waren! Aufgrund der Abbildung des Vektors $\mathbf{c}$ auf ein Symbol x ist die **TCM nichtlinear**! Hieraus folgt:

$\rightarrow$ Vergleich aller Sequenzen mit der Nullfolge ist nicht mehr ausreichend.

$\rightarrow$ Es müssen alle Sequenzen untereinander verglichen werden.

$\rightarrow$ Deutlich höherer Aufwand

Abschätzung der Fehlerwahrscheinlichkeit bei ML-Decodierung mit *Union Bound*:

Die Herleitung der *Union Bound*-Abschätzung verläuft bei der TCM analog zu der bei den Faltungscodes. Wir treffen zunächst die Annahme, dass $\mathbf{x}$ die gesendete Symbolfolge und $\mathbf{x}'$ eine beliebige andere Sequenz repräsentieren. Ferner sei $\mathcal{M}(\mathbf{x})$ die Menge aller Empfangssequenzen $\mathbf{y}$, die zu $\mathbf{x}$ die geringste euklidische Distanz haben. Dann ist $\bar{\mathcal{M}}(\mathbf{x})$ die zu $\mathcal{M}(\mathbf{x})$ komplementäre Menge, d.h. $\bar{\mathcal{M}}(\mathbf{x})$ enthält alle $\mathbf{y}$, die dichter an irgendeiner Sequenz $\mathbf{x}'$ als an $\mathbf{x}$ liegen.

Die Fehlerwahrscheinlichkeit für $\mathbf{x}$ lautet

$$\begin{aligned}
P_e(\mathbf{x}) &= P\left(\mathbf{y} \notin \mathcal{M}(\mathbf{x})\right) \qquad \text{mit} \qquad \mathcal{M}(\mathbf{x}) = \{\mathbf{y} | P(\mathbf{y}|\mathbf{x}) > P(\mathbf{y}|\mathbf{x}'), \forall\, \mathbf{x}' \in \Gamma\} \\
&= P\left(\mathbf{y} \in \bar{\mathcal{M}}(\mathbf{x})\right) \qquad \text{mit} \qquad \bar{\mathcal{M}}(\mathbf{x}) = \{\mathbf{y} | P(\mathbf{y}|\mathbf{x}) < P(\mathbf{y}|\mathbf{x}'), \forall\, \mathbf{x}' \in \Gamma\} \\
&= P\left(\mathbf{y} \in \bigcup_{\mathbf{x}'} \bar{\mathcal{M}}(\mathbf{x}, \mathbf{x}')\right) \qquad \text{mit} \qquad \bar{\mathcal{M}}(\mathbf{x}, \mathbf{x}') = \{\mathbf{y} | P(\mathbf{y}|\mathbf{x}) < P(\mathbf{y}|\mathbf{x}')\} \\
&\leq \sum_{\mathbf{x}'} P\left(\mathbf{y} \in \mathcal{M}(\mathbf{x}, \mathbf{x}')\right) = \sum_{\mathbf{x}'} P\left(\mathbf{x} \longrightarrow \mathbf{x}'\right) \ . \qquad (2.19)
\end{aligned}$$

Das Gleichheitszeichen in Gl. (2.19) ist gültig, wenn die Mengen $\bar{\mathcal{M}}(\mathbf{x}, \mathbf{x}')$ disjunkt sind, d.h. keine Sequenz $\mathbf{y}$ in mehr als einer Menge $\bar{\mathcal{M}}(\mathbf{x}, \mathbf{x}')$ enthalten ist. Während bei den Faltungscodes die Wahrscheinlichkeit $P(\mathbf{x} \longrightarrow \mathbf{x}')$ immer auf die Nullfolge bezogen werden konnte, gilt dies wegen der Nichtlinearität der TCM nun nicht mehr. Folglich erhalten wir eine Fehlerwahrscheinlichkeit, die spezifisch für die betrachtete Sendefolge $\mathbf{x}$ ist. Entsprechend der Funktionsweise des Viterbi-Algorithmus gilt mit $\mathbf{y} = \mathbf{x} + \mathbf{n}$:

$$\begin{aligned}
P\left(\mathbf{x} \longrightarrow \mathbf{x}'\right) &= P\left(\|\mathbf{y} - \mathbf{x}\|^2 > \|\mathbf{y} - \mathbf{x}'\|^2\right) = P\left(\|\mathbf{n}\|^2 > \|\mathbf{x} + \mathbf{n} - \mathbf{x}'\|^2\right) \\
&= P\left(\mathbf{n}^H\mathbf{n} > (\mathbf{x} + \mathbf{n} - \mathbf{x}')^H(\mathbf{x} + \mathbf{n} - \mathbf{x}')\right) \\
&= P\left(\mathbf{n}^H\mathbf{n} > \|\mathbf{x} - \mathbf{x}'\|^2 + (\mathbf{x} - \mathbf{x}')^H\mathbf{n} + \mathbf{n}^H(\mathbf{x} - \mathbf{x}') + \mathbf{n}^H\mathbf{n}\right) \\
&= P\left((\mathbf{x} - \mathbf{x}')^H\mathbf{n} + [(\mathbf{x} - \mathbf{x}')^H\mathbf{n}]^* < -\|\mathbf{x} - \mathbf{x}'\|^2\right) \\
&= P\left(2\mathrm{Re}\left\{(\mathbf{x} - \mathbf{x}')^H\mathbf{n}\right\} < -\|\mathbf{x} - \mathbf{x}'\|^2\right) \\
&= P\left(\underbrace{\mathrm{Re}\left\{(\mathbf{x} - \mathbf{x}')^H \cdot \mathbf{n}\right\}}_{Y} < \underbrace{-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2}_{X}\right) \qquad (2.20)
\end{aligned}$$

Mit der quadratischen Euclidischen Distanz

$$d_e^2(\mathbf{x},\mathbf{x}') = \frac{\|\mathbf{x}-\mathbf{x}'\|^2}{E_s/T_s}\ , \tag{2.21}$$

ergibt sich für die mittelwertfreie Zufallsvariable Y in Gl. (2.20) ist die Varianz

$$\sigma_Y^2 = \frac{N_0}{T_s}\cdot\|\mathbf{x}-\mathbf{x}'\|^2 = d_e^2(\mathbf{x},\mathbf{x}')\cdot\frac{E_sN_0}{T_s^2}\ . \tag{2.22}$$

Der Term X ist konstant und gibt die quadratische euklidische Distanz zwischen $\mathbf{x}$ und $\mathbf{x}'$ an. Für Gl. (2.20) ergibt sich somit

$$P\left(\mathbf{x}\longrightarrow\mathbf{x}'\right) = \frac{1}{2}\cdot\operatorname{erfc}\left(\sqrt{\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{4N_0/T_s}}\right) = \frac{1}{2}\cdot\operatorname{erfc}\left(\sqrt{d_e^2(\mathbf{x},\mathbf{x}')\cdot\frac{E_s/T_s}{4N_0/T_s}}\right)\ . \tag{2.23}$$

Die Fehlerwahrscheinlichkeit für eine konkrete Symbolfolge $\mathbf{x}$ lautet damit

$$P_e(\mathbf{x}) \le \sum_{\mathbf{x}'} P\left(\mathbf{x}\longrightarrow\mathbf{x}'\right) = \frac{1}{2}\cdot\sum_{\mathbf{x}'}\operatorname{erfc}\left(\sqrt{d_e^2(\mathbf{x},\mathbf{x}')\cdot\frac{E_s}{4N_0}}\right)\ . \tag{2.24}$$

Es ist zu erkennen, dass nicht nur die Hamming-Distanz, also die Anzahl unterschiedlicher Symbole in $\mathbf{x}$ und $\mathbf{x}'$ eine Rolle spielt, sondern auch die tatsächliche euklidische Distanz zwischen den Symbolen. Gl. (2.24) kann durch eine Näherung weiter vereinfacht werden

$$\operatorname{erfc}\left(\sqrt{\frac{x+y}{2}}\right) \le \operatorname{erfc}\left(\sqrt{\frac{x}{2}}\right)\cdot e^{-\frac{y}{2}} \tag{2.25}$$

Mit dieser Näherung folgt aus Gl. (2.24)

$$\begin{aligned} P_e(\mathbf{x}) &\le \frac{1}{2}\cdot\sum_{\mathbf{x}'}\operatorname{erfc}\left(\sqrt{\underbrace{\Delta_f^2\cdot\frac{E_s}{4N_0}}_{x/2}+\underbrace{d_e^2(\mathbf{x}-\mathbf{x}'))\cdot\frac{E_s}{4N_0}-\Delta_f^2\cdot\frac{E_s}{4N_0}}_{y/2}}\right)\\ &\le \frac{1}{2}\cdot\operatorname{erfc}\left(\sqrt{\Delta_f^2\cdot\frac{E_s}{4N_0}}\right)\cdot e^{\Delta_f^2\cdot\frac{E_s}{4N_0}}\cdot\sum_{\mathbf{x}'}\exp\left(-d_e^2(\mathbf{x},\mathbf{x}')\cdot\frac{E_s}{4N_0}\right) \end{aligned} \tag{2.26}$$

Aus Gl. (2.26) ist ersichtlich, dass nur noch der letzte Term von der euklidischen Distanz zwischen $\mathbf{x}$ und $\mathbf{x}'$ abhängt. Dies erweist sich gleich als nützlich, da nun eine einfache Beschreibung mit Hilfe des Distanzspektrums möglich ist (wie auch bei Faltungscodes). Um nun auf die Gesamtfehlerwahrscheinlichkeit P_e schließen zu können, sind alle Sequenzen $\mathbf{x}$ zu betrachten. Allgemein gilt der Zusammenhang für die Gesamtfehlerwahrscheinlichkeit:

$$\begin{aligned} P_e &= \sum_{\mathbf{x}} P\left(\text{Decodierfehler},\mathbf{x}\right)\\ &= \sum_{\mathbf{x}} P\left(\mathbf{x}\right)\cdot P\left(\text{Decodierfehler}|\mathbf{x}\right)\\ &= \sum_{\mathbf{x}} P\left(\mathbf{x}\right)\cdot P_e\left(\mathbf{x}\right)\\ &\le \frac{1}{2}\cdot\operatorname{erfc}\left(\sqrt{\Delta_f^2\cdot\frac{E_s}{4N_0}}\right)\cdot e^{\Delta_f^2\cdot\frac{E_s}{4N_0}}\cdot\sum_{\mathbf{x}} P\left(\mathbf{x}\right)\cdot\sum_{\mathbf{x}'}\exp\left(-d_e^2(\mathbf{x},\mathbf{x}')\cdot\frac{E_s}{4N_0}\right) \end{aligned} \tag{2.27}$$

Der letzte Exponentialausdruck in Gl. (2.27) kann nun mit Hilfe des Distanzspektrums beschrieben werden. Da bei der TCM wie bereits erwähnt nicht nur der Nullpfad, sondern **alle** Sequenzen betrachtet werden müssen, ist

auch deren Auftrittswahrscheinlichkeit $P(\mathbf{x})$ zu beachten. Wir erhalten folgenden Ausdruck für das Distanzspektrum einer TCM

$$T(W,D) = \sum_{\mathbf{x}} P(\mathbf{x}) \cdot \sum_{\mathbf{x}'} D^{d_e^2(\mathbf{x}-\mathbf{x}')} \cdot W^{w(\mathbf{x},\mathbf{x}')} \tag{2.28}$$

Die Funktion $w(\mathbf{x},\mathbf{x}')$ in Gl. (2.28) gibt die Anzahl der Bitfehler bei einer Verwechslung von $\mathbf{x}$ und $\mathbf{x}'$ an. Sind beispielsweise $\mathbf{x}$ und $\mathbf{x}'$ aus den Informationsfolgen $\mathbf{u}$ und $\mathbf{u}'$ hervorgegangen, so gilt

$$w(\mathbf{x},\mathbf{x}') = d_H(\mathbf{u},\mathbf{u}') \; .$$

Ein Vergleich von Gl. (2.27) und Gl. (2.28) zeigt, dass bei geeigneter Wahl von D und W das Distanzspektrum in der Abschätzung der Fehlerwahrscheinlichkeit P_e schon enthalten ist. Mit $D = e^{-\frac{E_s}{4N_0}}$ und $W = 1$ erhalten wir die Symbolfehlerrate

$$P_e \leq \frac{1}{2} \cdot \mathrm{erfc}\left(\sqrt{\Delta_f^2 \cdot \frac{E_s}{4N_0}}\right) \cdot e^{\Delta_f^2 \cdot \frac{E_s}{4N_0}} \cdot T\left(D = e^{-\frac{E_s}{4N_0}}, W = 1\right) \tag{2.29}$$

Um eine Abschätzung der Bitfehlerrate P_b zu erhalten, ist nun die Anzahl der unterschiedlichen Infobit $w(\mathbf{x},\mathbf{x}')$ zwischen $\mathbf{x}$ und $\mathbf{x}'$ zu berücksichtigen.

$$P_b \leq \frac{1}{2} \cdot \mathrm{erfc}\left(\sqrt{\Delta_f^2 \cdot \frac{E_s}{4N_0}}\right) \cdot e^{\Delta_f^2 \cdot \frac{E_s}{4N_0}} \cdot \sum_{\mathbf{x}} P(\mathbf{x}) \cdot \sum_{\mathbf{x}'} \frac{w(\mathbf{x},\mathbf{x}')}{m} \cdot \exp\left(-d_e^2(\mathbf{x},\mathbf{x}') \cdot \frac{E_s}{4N_0}\right) \tag{2.30}$$

$$\leq \frac{1}{2} \cdot \mathrm{erfc}\left(\sqrt{\Delta_f^2 \cdot \frac{E_s}{4N_0}}\right) \cdot e^{\Delta_f^2 \cdot \frac{E_s}{4N_0}} \cdot \frac{1}{m} \cdot \frac{\partial T(D,W)}{\partial W} \Bigg|_{D=e^{-\frac{E_s}{4N_0}}, W=1} \tag{2.31}$$

Die Bilder 2.24 und 2.25 zeigen die Ergebnisse der analytischen Abschätzung der Bitfehlerrate in Abhängigkeit des Signal-Rausch-Abstandes E_b/N_0 in dB. Sie sind für die besten Codes nach Ungerböck erzielt worden [VWZP89]. Es ist zu erkennen, dass mit zunehmender Anzahl von Zuständen die Leistungsfähigkeit ständig steigt. Für die codierte 8-PSK wird bei einer Bitfehlerrate von $P_b = 10^{-5}$ ein maximaler Gewinn gegenüber der uncodierten QPSK von 3,6 dB erzielt (64 Zustände). Dies liegt noch etwa 3,4 dB von dem nach Shannon theoretisch möglichen Gewinn entfernt. Zwar kann durch weitere Erhöhung der Anzahl an Zuständen die Leistungsfähigkeit weiter verbessert werden, allerdings steigt damit aber auch der Decodieraufwand exponentiell an (siehe Faltungscodes). Außerdem ist es unwahrscheinlich, dass dieser Weg überhaupt zum theoretischen Optimum führt. Allerdings bleibt festzuhalten, dass mit der trelliscodierten Modulation die Leistungsfähigkeit von Übertragungssystemen wesentlich verbessert werden kann, ohne die spektrale Effizienz des Systems zu verändern. Es ist lediglich ein größerer Rechenaufwand im Empfänger erforderlich.

2.7 Pragmatischer Ansatz nach Viterbi

Ein praktischer Nachteil der im letzten Abschnitt vorgestellten Strukturen wird deutlich, wenn man moderne Konzepte der Nachrichtenübertragung betrachtet. So spielt die Anpassung an zeitvariante Kanaleigenschaften, aber auch die Adaptivität bzgl. der vom Nutzer geforderten Datenraten eine immer wichtigere Rolle. Die Resource *Bandbreite* soll nicht länger einem Nutzer für die Dauer einer Verbindung fest zugewiesen werden, sondern je nach Anforderung flexibel vergeben werden. So kann sie in Zeiten geringer Nutzung anderen Netzteilnehmern zur Verfügung gestellt werden und die Effizienz des Systems gesteigert werden.

Im Hinblick auf die optimale TCM nach Ungerböck bedeutet dies, dass je nach Anforderung zwischen Konstellationen mit unterschiedlicher spektraler Effizienz η umgeschaltet werden muss. Da die TCM aber für jedes η eigens optimiert wurde und somit unterschiedliche Schieberegisterstrukturen zum Einsatz kommen, erfordert dieser Ansatz einen hohen Aufwand an Hardware (mehrfache Realisierung für verschiedene η). Dies wirkt sich insbesondere im Empfänger aus.

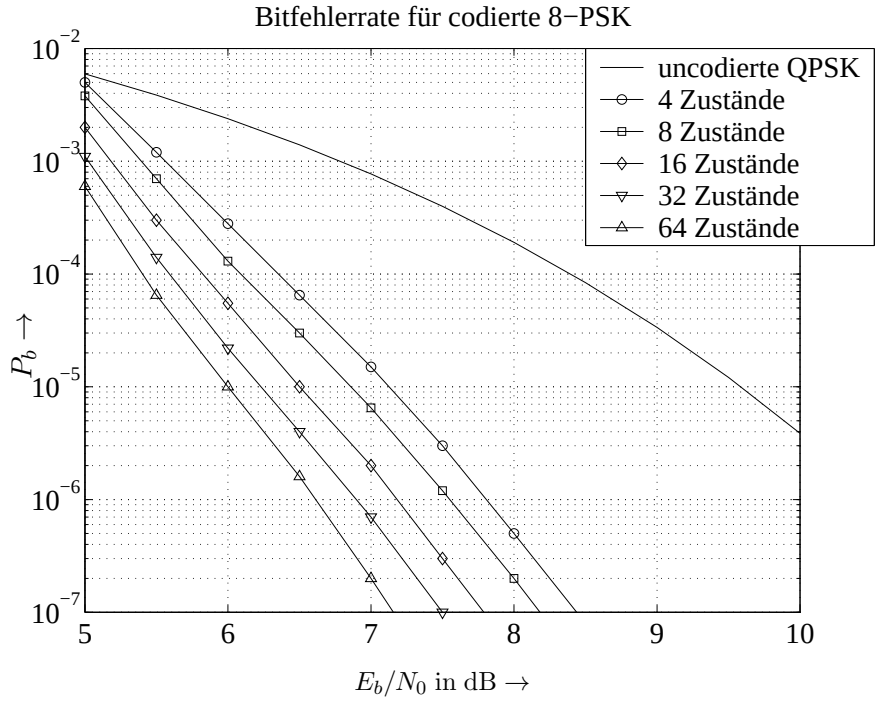


Bild 2.24: Analytischen Abschätzung der Bitfehlerrate für codierte 8-PSK und unterschiedliche Anzahl von Zuständen [VWZP89]

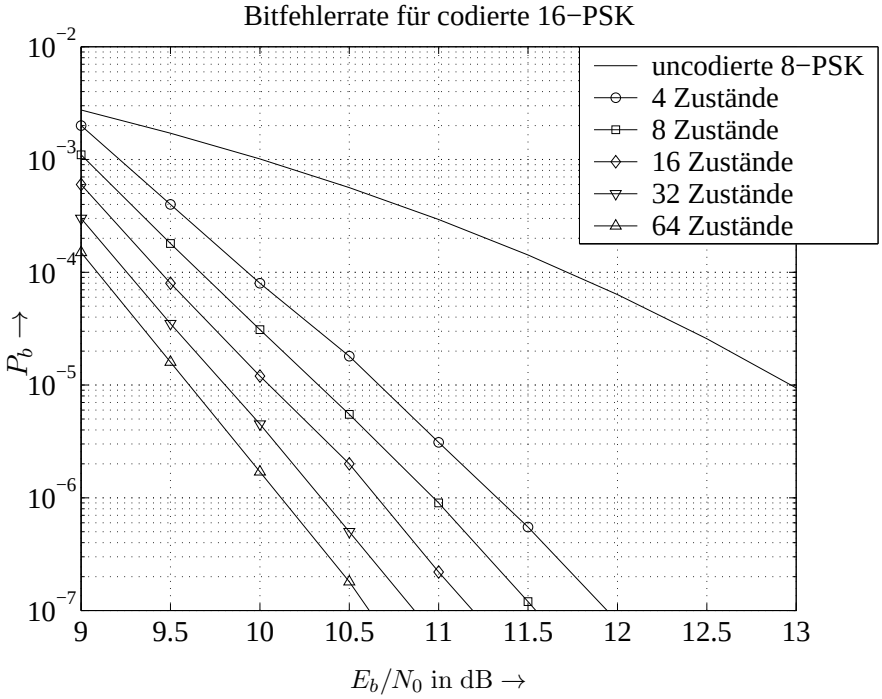


Bild 2.25: Analytischen Abschätzung der Bitfehlerrate für codierte 16-PSK und unterschiedliche Anzahl von Zuständen [VWZP89]

Einen anderen Ansatz, der die geforderte Flexibilität besser berücksichtigt, stellt die *Pragmatische TCM* von Viterbi dar [VWZP89]. Ihr Konzept ist in Bild 2.26 illustriert.

Der Ansatz der pragmatischen TCM besteht aus einem konventionellen halbratigen Faltungscodes (NSC) mit der Einflußlänge $L_c = 7$, wie er im letzten Semester vorgestellt wurde. Reicht eine spektrale Effizienz von $\eta = 1$ bit/s/Hz aus, so wird zur Übertragung eine QPSK zusammen mit dem NSC-Code ($R_c = 1/2$) verwendet.

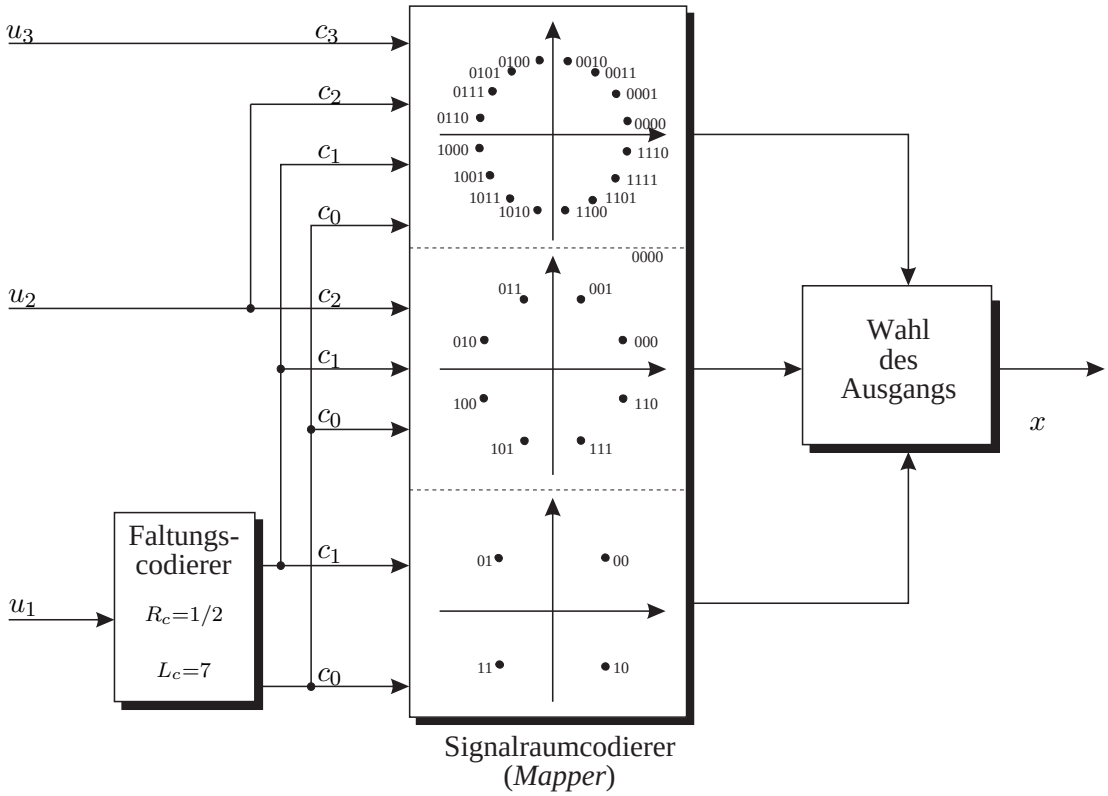


Bild 2.26: Struktur des pragmatischen TCM-Codierers nach Viterbi

Für $\eta = 2$ bit/s/Hz ist dagegen die 8-PSK zu wählen. Dabei wird das erste Informationsbit u_1 wie bisher durch den Faltungscode codiert, an der Codierer- bzw. Decodiererstruktur im Empfänger (Viterbi) ändert sich also nichts. Das dritte nun zu nutzende Bit eines 8-PSK-Symbols wird mit dem zweiten Informationsbit u_2 (uncodiert) belegt, welches bestimmt, ob die obere (000, 001, 011, 010) oder die untere Halbebene (100, 101, 111, 110) des 8-PSK-Signalraums verwendet wird. Die Ausgangsbit c_0 und c_1 des NSC-Codes legen dann das konkrete Symbol aus der gewählten Halbebene fest. Im Trellisdiagramm treten jetzt also von und zu jedem Zustand 2 parallele Zweige auf.

Für $\eta = 3$ bit/s/Hz muss nun noch ein zweites uncodiertes Bit u_3 hinzugenommen werden. Die beiden uncodierten Bit u_2 und u_3 bestimmen nun den Quadranten, aus dem die codierten Bit c_0 und c_1 das letztendlich zu sendende Symbol bestimmen.

Durch diese Struktur ist ein hohes Maß an Flexibilität gegeben, da sich bei ändernder spektraler Effizienz nur die Anzahl der je Symbol uncodiert übertragenen Informationsbit ändert, nicht aber die Struktur der Codierung. Natürlich führt dieser Ansatz nicht zu einer optimalen Struktur, die Verluste gegenüber der optimalen TCM nach Ungerböck halten sich jedoch in Grenzen. Bei einer Bitfehlerrate von $P_b = 10^{-5}$ verzeichnet die pragmatische TCM für die 8-PSK nur einen Verlust von etwa 0,4 dB gegenüber Ungerböcks TCM, bei der 16-PSK sind beide Verfahren sogar identisch, weil hier auch die Struktur nach Ungerböck einen halbratigen Faltungscode einsetzt.

2.8 Mehrstufencodes nach Imai

2.8.1 Struktur des Codierers

Es wurde bereits mehrfach erwähnt, dass die hier vorgestellte TCM nach Ungerböck beim AWGN-Kanal zu optimalen Ergebnissen führt. Die verschiedenen Kombinationen von Codierungs- und Modulationsverfahren

führen stets zur maximalen kleinsten euklidischen Distanzen, die für die Leistungsfähigkeit beim AWGN-Kanal entscheidend sind.

Für Schwundkanäle, wie sie häufig im Bereich des Mobilfunks auftreten, gelten allerdings andere Optimierungskriterien. Hier ist beispielsweise die kleinste Anzahl unterschiedlicher Symbole zweier Sequenzen entscheidend. Während im binären Fall beide Kriterien identisch sind, gilt dies für höherstufige Modulationsverfahren nicht mehr. Ist die kleinste Anzahl unterschiedlicher Symbole bei zwei Ansätzen identisch, entscheidet das größere Produkt der euklidischen Distanzen entlang zweier Pfade.

Der in diesem Abschnitt behandelte Ansatz von Imai wird in der Literatur als Mehrstufencode (*Multilevel Code*, MLC) bezeichnet. Er zeigt insbesondere bei Schwundkanälen ein wesentlich besseres Verhalten. Die prinzipielle Struktur ist in Bild 2.27 dargestellt und ähnelt in seinem grundsätzlichen Aufbau dem von Ungerböck.

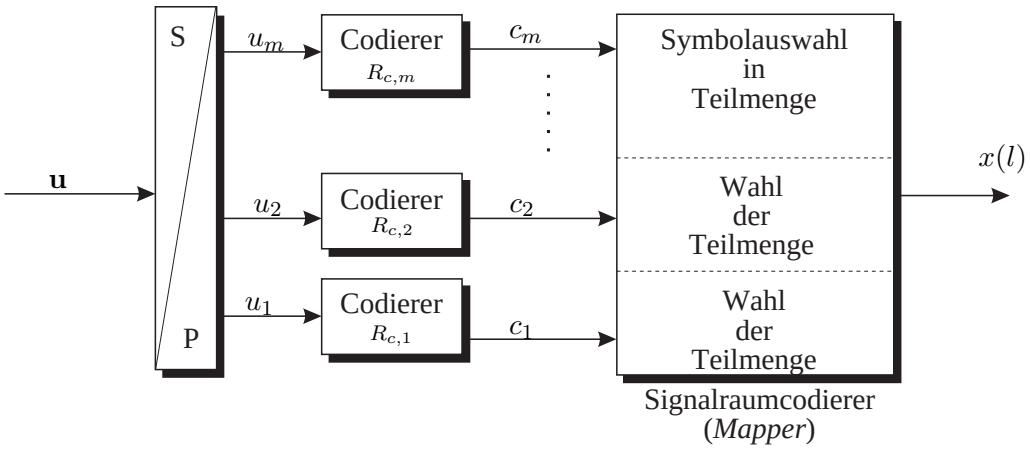


Bild 2.27: Struktur der Mehrstufencodes nach Imai

Der Unterschied besteht darin, dass bei MLC's nicht mehr rein zwischen codierten und uncodierten Bit unterschieden wird. Vielmehr wird nach der seriell-parallel-Umsetzung jedem Zweig ein eigener Codierer mit entsprechender Komplexität und Coderate zugeordnet. Im Extremfall besitzen einige der Codierer die Coderate $R_c = 1$, d.h. an den Signalraumcodierer können auch uncodierte Bit gelangen. Eine mögliche Strategie ist die folgende:

- Zerteile den Signalraum nach dem *Set-Partitioning* nach Ungerböck.
- Aufgrund der wachsenden euklidischen Distanz in den Teilmengen besitzen diese nach jedem Partitionierungsschritt eine größere *äquivalente Kanalkapazität*.
- Jeder Zweig des Mehrstufencodes entscheidet über einen Partitionierungsschritt.
- Der Code in einem Zweig ist dann derart zu dimensionieren, dass er der Kanalkapazität der entsprechenden Partitionierungsmenge gerecht wird.
- Da die euklidische Distanz von Partitionsstufe zu Partitionsstufe stetig zunimmt, wächst auch die Kanalkapazität an und die Codes werden immer schwächer.
- Im Extremfall besitzen die Zweige, die mit den letzten Partitionierungsschritten korrespondieren, gar keinen Codierer mehr (wie auch bei der TCM nach Ungerböck und der pragmatischen TCM nach Viterbi).

2.8.2 Prinzip der Decodierung

Die Decodierung erfolgt nach dem in Bild 2.28 gezeigten Verfahren, welches als kaskadierte Decodierung bezeichnet wird. Zunächst wird der Code, welcher mit der ersten Partitionierungsstufe korrespondiert, decodiert.

Dies bedeutet, dass bei der Decodierung nicht zwischen zwei Symbolen, sondern zwischen den beiden Partitionierungsmengen entschieden wird. Der Decodierer 1 liefert zwei Ausgänge, die geschätzten Informationsbit $\hat{u}_1$ und die geschätzte recodierte Folge $\hat{c}_1$. Letztere enthält die (geschätzte) Wahl der Partitionierungsmengen der ersten Stufe zu den jeweiligen Zeitpunkten und ist somit eine wichtige Entscheidungsgrundlage für den zweiten Decodierer. Dieser beschränkt sich nun nämlich bei seiner Suche nach dem ML-Pfad auf die durch $\hat{c}_1$ festgelegten Partitionierungsmengen. Das Verfahren wird so weiter geführt, bis die unterste Partitionierungsebene erreicht wird und der Decodierer m zwischen den beiden verbliebenen Symbolen entscheiden muss.

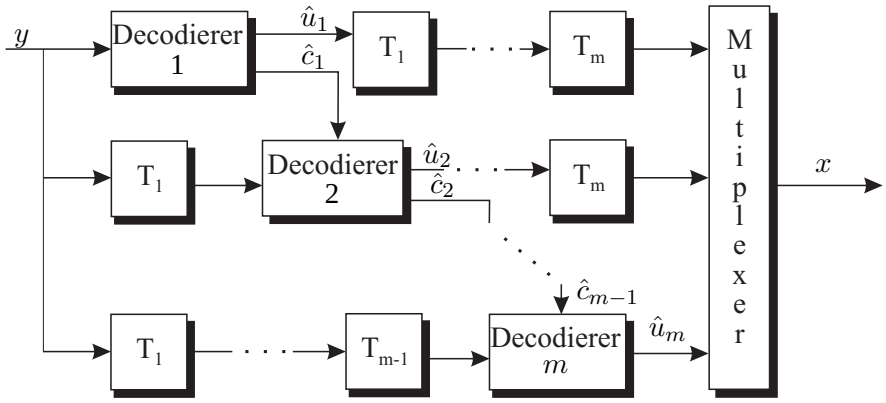


Bild 2.28: Struktur der kaskadierten Decodierung von Mehrstufencodes

Die Struktur hat neben ihrem hohen Aufwand auch noch den Nachteil, dass Fehlentscheidungen eines Decodierers unweigerlich zu Folgefehlern in den nachfolgenden Decodierern führen, es kommt also zu einer Fehlerfortpflanzung. Um diesen Effekt zu minimieren, werden z.B. zwischen den einzelnen Decodierern (und natürlich auch den Codierern) Interleaver eingesetzt, die die Bündelfehler am Ausgang eines Decodierers in Einzelfehler spreizen. Des weiteren kann der gesamte Decodiervorgang auch wiederholt werden, d.h. der erste Decodierer decodiert nun erneut den empfangenen Datenstrom, jetzt aber mit der Kenntnis aller übrigen Entscheidungen – außer seiner eigenen –, usw. (siehe Bild 2.29). Wir erhalten somit einen iterativen Decodierprozeß. Wie später noch gezeigt wird, ist es dann von Vorteil, Decodieralgorithmen einzusetzen, die neben der harten Entscheidung auch ein Maß für die Zuverlässigkeit des Ergebnisses liefern (*Soft-In/Soft-Out-Decoder*).

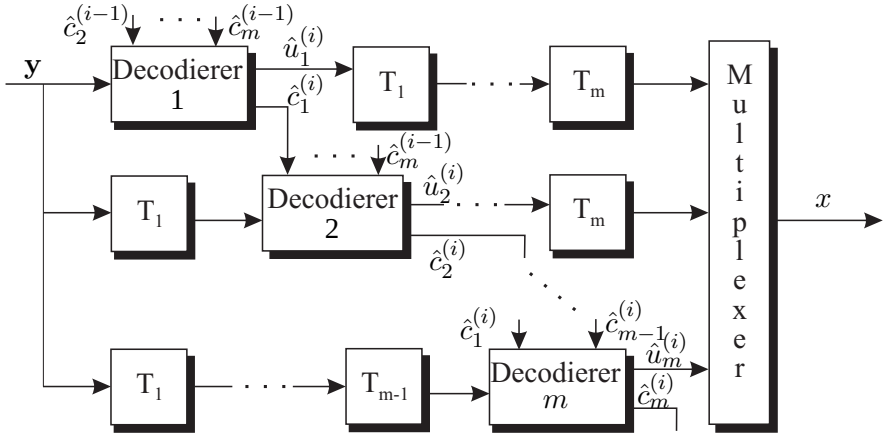


Bild 2.29: Struktur der kaskadierten Decodierung mit Iterationen von Mehrstufencodes

2.8.3 Optimierung

Bezüglich einer Optimierung der gesamten Anordnung stellt sich nun die Frage, wie die Coderaten in den einzelnen Stufen zu wählen sind, damit sich insgesamt ein möglichst leistungsfähiger Code ergibt. Eine einfache Möglichkeit besteht beispielsweise darin, allen Stufen den gleichen Code zuzuweisen. Dieser Ansatz trägt aber

nicht der Tatsache Rechnung, dass jeder Partitionierungsstufe ein anderes Signalraumalphabet zur Verfügung steht. Damit sieht jeder Codierer einen unterschiedlichen äquivalenten Übertragungskanal mit entsprechender Kanalkapazität. Außerdem wäre es sinnvoll, möglichst starke Codes in den ersten Stufen einzusetzen, um bei der Decodierung Folgefehler zu vermeiden.

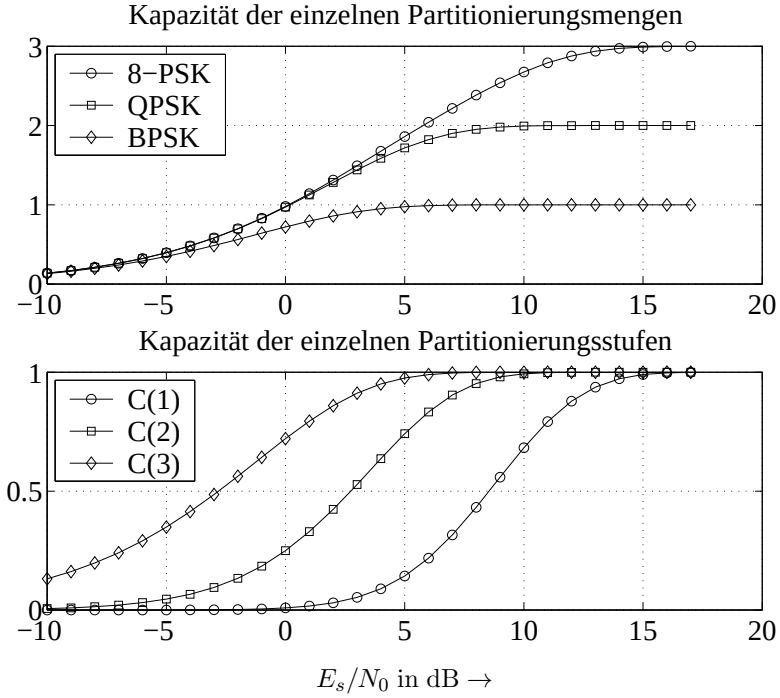


Bild 2.30: Kanalkapazität für Mehrstufencodes (8-PSK) beim AWGN-Kanal

Theoretisch kann dieses Problem mit Hilfe der Informationstheorie gelöst werden. Für jede Teilmenge des Signalraumalphabets, die durch das *Set-Partitioning* gebildet wird, kann die Kanalkapazität gemäß Gl. (2.14) berechnet werden. Beispielsweise besitzt das Gesamtsystem für eine 8-PSK die im oberen Diagramm von Bild 2.30 gezeigte Kapazität. Im ersten Partitionierungsschritt entstehen zwei QPSK-Signalräume mit entsprechend verringerter Kapazität, im zweiten Schritt dann vier BPSK-Mengen.

Dabei ist zu beachten, dass die codierten Bit der einzelnen Stufen nicht ein Symbol innerhalb der jeweiligen partitionierten Mengen auswählen, sondern sie bestimmen die Menge selbst, die dann in der folgenden Partitionierungsstufe verwendet werden soll. Somit wird die Kapazität einer bestimmten Stufe nicht anhand der Symbole einer Teilmenge bestimmt, sondern durch Partitionierung der Teilmengen. Es kann gezeigt werden, dass die Kapazität $C(1)$ der ersten Partitionierungsstufe aus der Differenz der Kapazitäten von 8-PSK und QPSK berechnet wird

$$C(1) = C(8-PSK) - C(QPSK)\ . \tag{2.32}$$

Entsprechend gilt für die nächste Partitionierungsstufe

$$C(2) = C(QPSK) - C(BPSK) \tag{2.33}$$

und trivialerweise $C(3) = C(BPSK)$. Die Ergebnisse zeigt das untere Diagramm in Bild 2.30 für eine 8-PSK. So weist die erste Partitionierungsstufe die geringste Kapazität auf, da hier zwischen zwei QPSK-Mengen unterschieden werden muss, deren gegenseitige euklidische Distanzen sehr gering sind. Außerdem kommt noch der Nachteil der Mehrfachrepräsentation hinzu, da jede Menge aus mehreren Symbolen besteht. In der zweiten Stufe nimmt die Kapazität zu, hier wird zwischen BPSK-Mengen unterschieden. Die größte Kapazität besitzt die letzte Stufe, da hier nur eine normale BPSK-Modulation stattfindet, die Kurve ist schon aus dem letzten Semester bekannt. Die Summe aller Kurven ergibt dann die Kapazitätskurve der 8-PSK aus dem oberen Diagramm.

Die Codierung ist nun derart vorzunehmen, dass in den ersten Partitionierungsstufen starke Codes (Gedächtnis, Coderate) eingesetzt werden, da hier die Kapazität am geringsten ist. In den letzten Ebenen der Partitionierung reichen dann relativ schwache Codes aus, unter Umständen kann sogar ganz auf eine Codierung verzichtet werden. Die Bilder 2.31 und 2.32 zeigen die Ergebnisse für eine 16-PSK und eine 16-QAM.

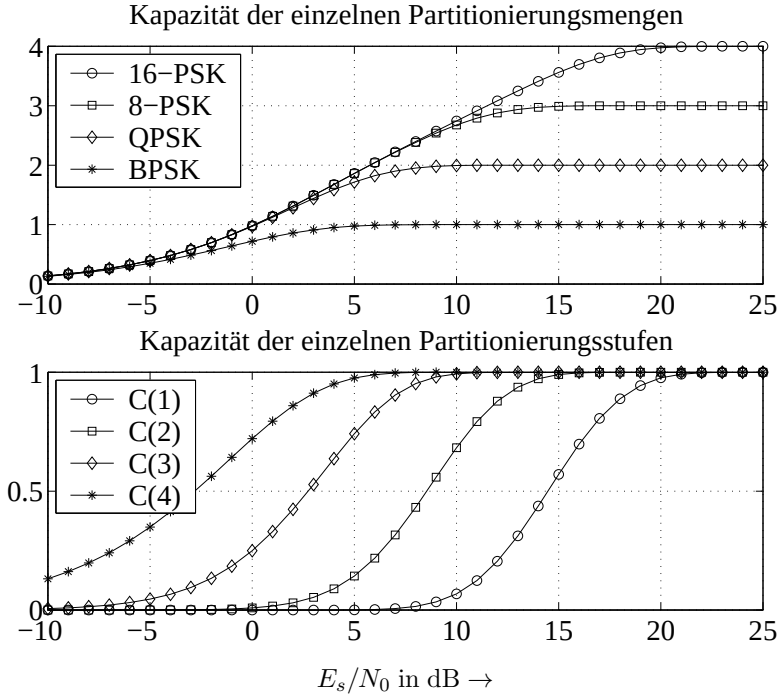


Bild 2.31: Kanalkapazität für Mehrstufencodes (16-PSK) beim AWGN-Kanal

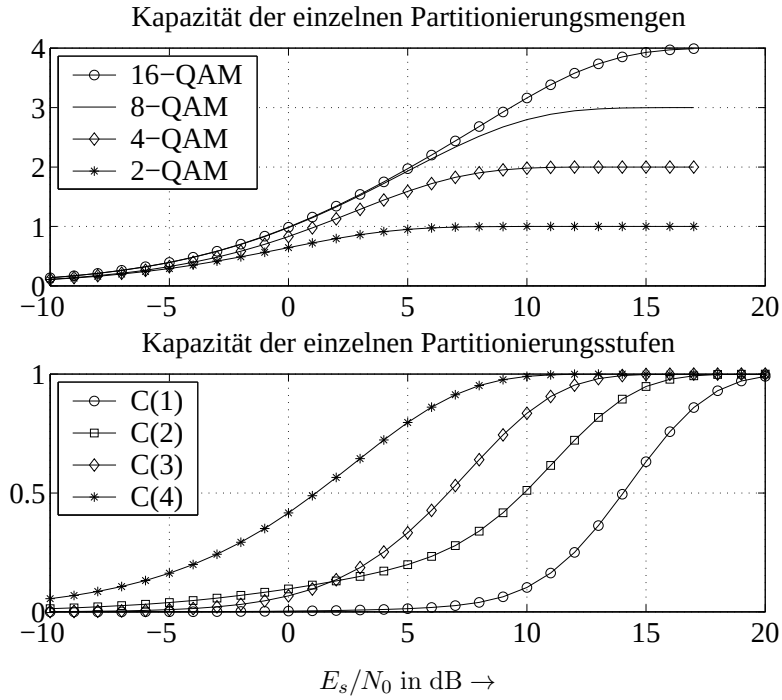


Bild 2.32: Kanalkapazität für Mehrstufencodes (16-QAM) beim AWGN-Kanal

2.9 TCM in der Modemtechnik

Modem: **Mo**dulator - **Dem**odulator

Der Telefonkanal ist ein prädestiniertes Medium für den Einsatz der TCM. Hier gilt es nämlich, über einen sehr schmalbandigen Kanal möglichst hohe Datenraten mit einer möglichst hohen Qualität (geringe Fehlerrate) zu übertragen. Die derzeit aktuellen Modems nach V.90 können bis zu 55,6 kbit/s über einen Kanal mit nur 3,2 kHz Bandbreite übertragen; das ist mit einer einfachen binären Modulationsform nicht möglich. Hier müssen bandbreiteneffiziente Verfahren wie die TCM zum Einsatz kommen.

Telefonkanal:

Bandbreite:	AWGN-Kanal mit $B \leq 3.200$ Hz
Signal-Rausch-Abstand:	$E_s/N_0 = 28$ dB $\ldots$ 36 dB
Kanalkapazität:	$C^{1D} = B \cdot \mathrm{ld}\left(1 + \frac{E_s}{N_0}\right) = 30 \ \ldots \ 38$ kbit/s
	(bei reiner AWGN-Störung)
	$C^{2D} = 2B \cdot \mathrm{ld}\left(1 + \frac{E_s}{2N_0}\right) = 56 \ \ldots \ 70$ kbit/s

Entgegen der Annahme reiner AWGN-Störungen treten auf Telefonkanälen auch Impulsstörungen, Übersprechen, Mehrwegeausbreitung und nichtlineare Störungen auf. Die tatsächliche Kapazität liegt also unter der oben angegebenen.

Standards durch CCITT / ITU

CCITT: *Comité Consultatif International de Télégraphique et Téléphonique)*
jetzt ITU: *International Telecommunication Union*

V.26:
- 1962, 1968 als Standard
- uncodierte 4-PSK
- 2,4 kbit/s bzw. 1,2 kbaud
- analoger, fester Entzerrer (Kompromiß für mittleres Kanalprofil)

V.27:
- 1967
- uncodierte 8-PSK
- 4,8 kbit/s bzw. 1,6 kbaud
- durch größere Bandbreite dispersiver Kanal $\longrightarrow$ System empfindlicher
- einstellbarer, analoger Entzerrer

V.29:
- uncodierte 16-QAM
- 9,6 kbit/s bzw. 2,4 kbaud
- System noch empfindlicher
- Entzerrung durch digitalen T-Entzerrer

Erst jetzt hält die Kanalcodierung Einzug in die Modemtechnik, da die technologische Entwicklung mittlerweile soweit voran geschritten ist, dass relativ aufwendige Verfahren wie die Decodierung der TCM realisierbar sind.

V.32:
- 1981

  - 32-QAM TCM (rotationsinvariant, von Wei)
  - 14,4 kbit/s bzw. 3,6 kbaud
  - *Fractional Spacing Equalizer*, digitaler T-Entzerrer, Überabtastung

V.33:
  - 128-QAM TCM
  - 14,4 kbit/s bzw. 2,4 kbaud
  - 64-QAM TCM
  - 12 kbit/s bzw. 2,4 kbaud

V.34:
  - 960-QAM TCM
  - adaptive Anpassung an den Kanal (Ausmessen des Kanals erforderlich)
  - Coderaten $R_c = 2/3$ (16 Zustände), $R_c = 3/4$ (32 Zustände), $R_c = 4/5$ (64 Zustände)
  - B = 3,2 kHz
  - 2,4 kbit/s . . . 28,8 kbit/s

V.34:
  - 960-QAM TCM
  - adaptive Anpassung an den Kanal (Ausmessen des Kanals erforderlich)
  - Coderaten $R_c = 2/3$ (16 Zustände), $R_c = 3/4$ (32 Zustände), $R_c = 4/5$ (64 Zustände)
  - B = 3,2 kHz
  - 2,4 kbit/s . . . 28,8 kbit/s

Ein wichtiger Aspekt der Modem-Technik ist die Entzerrung des Telefonkanals. Schon in den 80er Jahren, als noch uncodiert übertragen wurde, konnte die Datenrate aufgrund besserer Entzerrungsverfahren deutlich verbessert werden. Nach den analogen Entzerrern wurden vorwiegend digitale T-Entzerrer eingesetzt, da die optimale MLSE-Lösung viel zu aufwendig war. Entscheidungsrückgekoppelte Entzerrer eignen sich ebenfalls nicht, weil einerseits die Rückkopplung der detektierten Werte ohne Einbeziehung der Kanaldecodierung zu viele Folgefehler verursachen würde, und andererseits eine Berücksichtigung der Decodierung zu große Verzögerungszeiten mit sich bringt. Eine Alternative stellt die adaptive Vorcodierung nach Tomlinson-Harashima dar, die nach entsprechender Kanalschätzung die Signale sendeseitig so vorverzerrt, dass nach der Übertragung nur noch geringe Kanaleinflüsse auftreten.

Kapitel 3

Verfahren zur adaptiven Fehlerkontrolle

3.1 Einführung

Bisher: FEC-Codierung (*Forward Error Correction*)

- Konstruktion möglichst leistungsfähiger Codes zur Fehlerkorrektur
- Durch feste Coderate R_c ist auch die Übertragungsrate konstant

$\longrightarrow$ **Durchsatz ist unabhängig vom Übertragungskanal!**

- Es ist kein Rückkanal erforderlich
- Nachteile:
  - Bei 'guten' Übertragungsbedingungen wird zuviel Redundanz hinzugefügt
    $\longrightarrow$ geringe Bandbreiteneffizienz
  - Bei 'schlechten' Übertragungsbedingungen reicht Korrekturfähigkeit des FEC-Codes nicht aus
    $\longrightarrow$ es treten nicht-korrigierbare Übertragungsfehler auf

$\longrightarrow$ **Qualität der Übertragung ist abhängig vom Übertragungskanal!**

Jetzt: ARQ-Verfahren (*Automatic Repeat Request*)
Unter ARQ-Verfahren versteht man Übertragungsprotokolle, die im Fall einer fehlerhaften Übertragung die falsch empfangenen Bereiche einer Nachricht wiederholen, also sozusagen Redundanz zufügen (Wiederholungscode). Da diese Redundanz aber ausschließlich im Fehlerfall eingefügt wird, spricht man von adaptiven Verfahren. Ist der Kanal sehr schlecht, treten häufig Fehler auf und es sind viele Wiederholungen erforderlich (hohe Redundanz). Bei guten Kanälen reichen dagegen sehr wenige Wiederholungen und damit eine geringe Redundanz aus. Die Redundanz paßt sich also den aktuellen Kanalbedingungen an (adaptiv)!

ARQ-Verfahren kommen in der Praxis sehr häufig zum Einsatz, und zwar immer dann, wenn sehr hohe Anforderungen an die Übertragungssicherheit, d.h. an die Fehlerrate gestellt werden. Im Zweifelsfall kann so lange wiederholt werden, bis endlich eine fehlerfreie Übertragung zustande gekommen ist. Hierdurch wird deutlich, dass die Nettodatenrate während schlechter Kanalbedingungen drastisch reduziert werden kann (s. auch Abschnitt 3.5). Folgende Bedingungen müssen für ein ARQ-System erfüllt sein:

- Es wird eine paketorientierte Übertragung (Burst-Betrieb) vorausgesetzt
- Es ist ein Rückkanal erforderlich, über den der Empfänger dem Sender mitteilen kann, dass ein Paket fehlerhaft war

- Es sind fehlererkennende Codes einzusetzen.

Bild 3.1 zeigt die allgemeine Struktur eines ARQ-Systems. Die im Empfänger ankommenden Datenpakete werden zunächst decodiert und durch die Überprüfung des Syndroms auf Fehlerfreiheit getestet. Liegt ein Übertragungsfehler vor, so fordert die ARQ-Steuerung des Empfängers die Wiederholung des korrupten Blocks an, indem ein NAK (*Negative Acknowledgement*) zum Sender übertragen wird. Dessen ARQ-Steuereinheit wertet das ARQ-Signal aus und initiiert die erneute Übertragung des entsprechenden Blocks. Im fehlerfreien Fall sendet der Empfänger ein ACK-Signal (*Acknowledgement*), wodurch der Sender erfährt, dass ein Paket fehlerfrei übertragen wurde und nicht mehr länger im Speicher gepuffert werden muß.

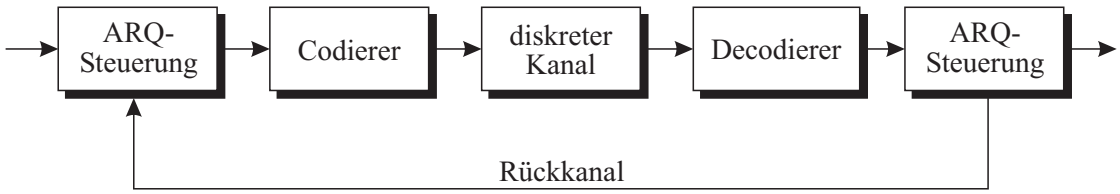


Bild 3.1: Prinzipielle Struktur eines ARQ-Systems

Fehlererkennende Codes
Wie aus dem letzten Semester bekannt ist, eignen sich CRC-Codes (*Cyclic Redundancy Check*) hervorragend zur Detektion von Bündelfehlern. Es handelt sich bei ihnen um lineare, zyklische und systematische Blockcodes, deren Generatorpolynom die Form

$$g(D) = (1 + D) \cdot p(D)$$

hat, wobei $p(D)$ ein primitives Polynom vom Grad r ist. Die Decodierung erfolgt über die Berechnung des Syndrompolynoms $s(D)$. Für $s(D) \neq 0$ wurde ein Fehler erkannt, für $s(D) = 0$ liegt entweder ein nicht erkennbarer Fehler oder aber kein Fehler vor. CRC-Codes besitzen folgende Eigenschaften:

- Alle Fehlermuster mit $w_H(\mathbf{e}) \leq 3$ werden erkannt.
- Alle Fehler mit ungeradem Gewicht werden erkannt.
- Alle Bündelfehler bis zur Länge $r + 1$ werden erkannt.
- Von den Bündelfehlern mit einer Länge von $r + 2$ wird nur eine Quote von 2^{-r} nicht erkannt
- Von den Bündelfehlern mit einer Länge von $\geq r + 3$ wird nur eine Quote von $2^{-(r+1)}$ nicht erkannt.

Es ist leicht einzusehen, dass die Leistungsfähigkeit der CRC-Codes entscheidenden Einfluß auf die Leistungsfähigkeit des gesamten ARQ-Systems hat. Werden Fehler durch ihn nicht erkannt, können die fehlerhaften Pakete auch nicht neu angefordert werden und das ARQ-System versagt.

3.2 Zuverlässigkeit der ARQ-Verfahren bei idealem Rückkanal

Es existieren zwei wichtige Qualitätsmerkmale, die ein ARQ-Prinzip charakterisieren. Zum einen ist die Zuverlässigkeit der Verfahren entscheidend, die durch die Auftrittswahrscheinlichkeit P_{ue} unerkannter Fehler bestimmt wird.

Des weiteren bestimmt der Datendurchsatz

$$\eta = \frac{\text{Anzahl fehlerfrei empfangener Infobit}}{\text{Gesamtzahl übertragener Bit}} = \frac{\text{Anzahl fehlerfrei empfangener Blöcke}}{\text{Gesamtzahl übertragener Blöcke}} \cdot R_c \qquad (3.1)$$

die Effizienz des Systems, wobei R_c die Coderate des eingesetzten fehlererkennenden Codes beschreibt. Der Durchsatz η entspricht der bekannten Coderate bei FEC-Systemen, ist hier allerdings variabel und paßt sich dem Übertragungskanal an. Er hängt von den jeweiligen ARQ-Strategien ab und wird daher in den nächsten Abschnitten für jedes Verfahren getrennt ermittelt.

In diesem Abschnitt beschäftigen wir uns mit der Zuverlässigkeit der ARQ-Systeme und gehen zunächst von einem idealen Rückkanal aus, die ACK- bzw. NAK-Signale des Empfängers erreichen also unverfälscht den Sender. Weiterhin gelten die Vereinbarungen:

P_{ue} Auftrittswahrscheinlichkeit eines unerkannten Fehlers (*undetected error*)

P_{ed} Auftrittswahrscheinlichkeit eines erkennbaren Fehlers (*error detected*)

P_w Wahrscheinlichkeit für das Versagen des ARQ-Systems (Fehler wird nicht erkannt)

Dann gilt:

$$\begin{aligned} P_w &= \underbrace{P_{ue}}_{\substack{\text{Fehler nicht}\\ \text{erkannt}}} + \underbrace{P_{ed} \cdot P_{ue}}_{\substack{\text{1 Wiederholung,}\\ \text{Fehler nicht erkannt}}} + \underbrace{P_{ed}^2 \cdot P_{ue}}_{\substack{\text{2 Wiederholungen,}\\ \text{Fehler nicht erkannt}}} + \cdots \\ &= P_{ue} \cdot \sum_{i=0}^{\infty} P_{ed}^i \\ &= \frac{P_{ue}}{1 - P_{ed}} \end{aligned} \tag{3.2}$$

Gl. (3.2) erlaubt folgende Interpretation: Die Zuverlässigkeit eines ARQ-Systems ist unabhängig von dem verwendeten Verfahren. Lediglich der zum Einsatz kommende fehlererkennende Code, z.B. CRC-Codes, bestimmt die Wahrscheinlichkeit für das Auftreten nicht erkennbarer Fehlermuster. Für einen Genie-Code, der alle Fehlermuster erkennen würde, wäre $P_{ue} = 0$ und somit auch $P_w = 0$. Der zur Fehlererkennung eingesetzte Code sollte also möglichst gut sein, damit sehr viele Fehlermuster erkannt werden. Das ARQ-Verfahren selbst hat allerdings keinen Einfluß auf die Zuverlässigkeit.

Bei FEC-Verfahren ist der Datendurchsatz konstant, die Zuverlässigkeit hängt hingegen vom Kanalzustand ab.
ARQ-Verfahren garantieren unabhängig vom Kanal eine gleich bleibende Übertragungssicherheit, ihr Durchsatz wird allerdings stark vom Kanal beeinflußt.

3.3 Klassische ARQ-Verfahren

Wir unterscheiden drei klassische ARQ-Verfahren, die im folgenden kurz erläutert werden. Sie können auch kombiniert werden, und zwar sowohl untereinander als auch mit FEC-Verfahren wie z.B. den Faltungscodes. Die letztgenannte Kombination wird im folgenden noch im Abschnitt 3.5 behandelt.

Neben der Funktionsbeschreibung der einzelnen ARQ-Strategien wird auch das zweite Leistungsmerkmal, der Datendurchsatz analysiert. Unter der Annahme eines idealen (störungsfreien) Rückkanals können wir die Effizienz η des SW-Verfahrens leicht berechnen. Wir setzen ferner voraus, dass ein Genie-Code, der alle Fehlermuster erkennt ($P_{ue} = 0 \;\rightarrow\; P_w = 0$), zum Einsatz kommt. Diese Annahme ist erlaubt, denn zur Berechnung des Datendurchsatzes spielen nicht erkannte Fehler keine Rolle, da sie nicht zu Wiederholungen führen.

3.3.1 *Stop & Wait*-Verfahren (SW)

Die einfachste, aber auch schlechteste Methode zur adaptiven Fehlerkontrolle ist das *Stop & Wait*-Verfahren. Die Funktionsweise ist in Bild 3.2 dargestellt. Rot eingefärbte Pakete sind fehlerhaft empfangen worden, blaue Pakete zeigen Wiederholungen an. Diese Darstellung wird auch in den folgenden Bildern beibehalten.

Es wird ein Block der Dauer T_B gesendet und solange mit dem Senden des zweiten Blocks gewartet, bis vom Empfänger die Bestätigung für den korrekten Empfang kommt (*ACK - Acknowledgement*). Tritt während der Übertragung ein Fehler auf, welcher im Empfänger erkannt wird, so sendet dieser ein NAK (*Negative Acknowledgement*), das den Sender zum Wiederholen des fehlerhaften Blocks veranlaßt.

Die Zeit, die zwischen zwei gesendeten Blöcken verstreicht, wird mit T_G bezeichnet und hängt von der Verzögerung der gesamten Datenübertragungsstrecke (Hin- und Rückkanal) ab (*round trip time*). Die Gesamtdauer zum Übertragen eines Blocks beträgt beim SW-Verfahren somit

$$T_t = T_B + T_G \tag{3.3}$$

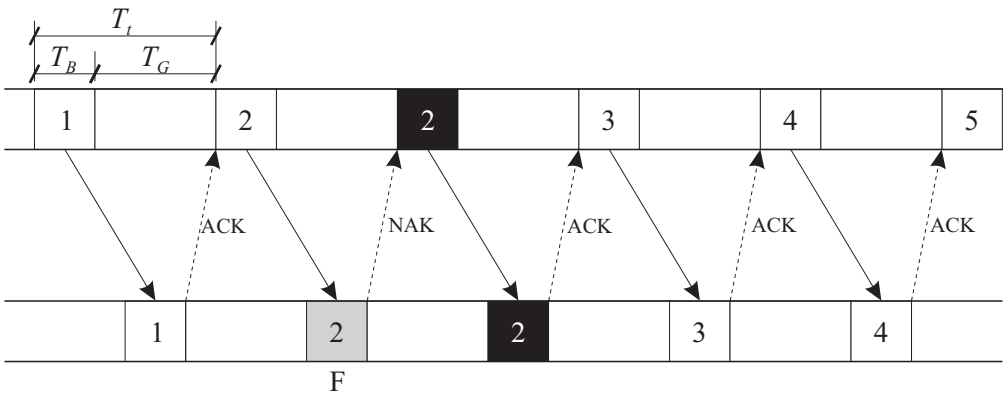


Bild 3.2: Funktionsweise des *Stop & Wait*-Verfahrens

Vorteile:

- Leicht zu implementieren
- Nur kleiner Puffer im Sender (1 Block), gar kein Puffer im Empfänger erforderlich!

Nachteile:

- Geringer Datendurchsatz durch hohe Leerlaufzeiten

Datendurchsatz bei idealem Rückkanal
Entsprechend Bild 3.2 benötigt ein fehlerfrei empfangenes Paket die Zeit $T_t = T_B + T_G$. Die Wahrscheinlich beträgt hierfür $P_c = 1 - P_{ed}$, da $P_{ue} = 0$ gilt. Werden fehlerhafte Blöcke empfangen, gilt folgender Zusammenhang:

Übertragung	$P_c = (1 - P_{ed})$	T_t
1 Wiederholung	$P_c = P_{ed}(1 - P_{ed})$	$2T_t$
2 Wiederholungen	$P_c = P_{ed}^2(1 - P_{ed})$	$3T_t$
3 Wiederholungen	$P_c = P_{ed}^3(1 - P_{ed})$	$4T_t$

Hieraus folgt für die mittlere Übertragungszeit eines Blocks:

$$\begin{aligned} T_{AV} &= (1-P_{ed})\cdot T_t + P_{ed}(1-P_{ed})\cdot 2T_t + P_{ed}^2(1-P_{ed})\cdot 3T_t + \cdots \\ &= T_t(1-P_{ed})\cdot\sum_{i=0}^{\infty}(i+1)\cdot P_{ed}^i \\ &= \frac{T_t(1-P_{ed})}{(1-P_{ed})^2} = \frac{T_t}{1-P_{ed}}\,. \end{aligned} \tag{3.4}$$

Die Effizienz η berechnet sich nun aus dem Verhältnis der Dauer eines Blocks T_B zur mittleren Übertragungszeit T_{AV} multipliziert mit der Coderate R_c des CRC-Codes. Wir erhalten

$$\begin{aligned} \eta_{SW} &= \frac{T_B}{T_{AV}}\cdot R_c = \frac{T_B}{T_t}\cdot(1-P_{ed})\cdot R_c = \frac{T_B}{T_B+T_G}\cdot(1-P_{ed})\cdot R_c \\ &= \frac{1-P_{ed}}{1+T_G/T_B}\cdot R_c\,. \end{aligned} \tag{3.5}$$

Die Durchsatzrate η wird also einerseits über P_{ed} durch den Kanal beeinflußt, andererseits aber auch durch die Systemrandbedingungen, die sich im Verhältnis T_G/T_B widerspiegeln. Geht die Fehlerwahrscheinlichkeit des Kanals und damit auch P_{ed} gegen Null, so gilt $\eta_{SW} = R_C/(1+T_G/T_B)$. Ist zusätzlich die Leerlaufzeit $T_G << T_B$, nimmt Gl. (3.5) die Form $\eta_{SW} = R_c$ an.

3.3.2 *Go-Back*-N-Verfahren (GB-N)

Das *Go-Back*-N-Verfahren stellt eine wesentliche Verbesserung gegenüber dem SW-Prinzip dar. Seine Funktionsweise zeigt Bild 3.3. Es wird nach dem Senden eines Blocks zunächst nicht mehr auf eine Antwort des Empfängers gewartet, sondern die nachfolgenden Blöcke werden direkt nacheinander übertragen. Wird im Empfänger ein Fehler detektiert und das NAK an den Sender geleitet, geht dieser bis zum fehlerhaften Block zurück (*go-back*-N) und wiederholt den falschen und auch alle nach ihm gesendeten Pakete, unabhängig davon, ob diese falsch waren oder nicht. Dies erklärt auch den Namen des Verfahrens.

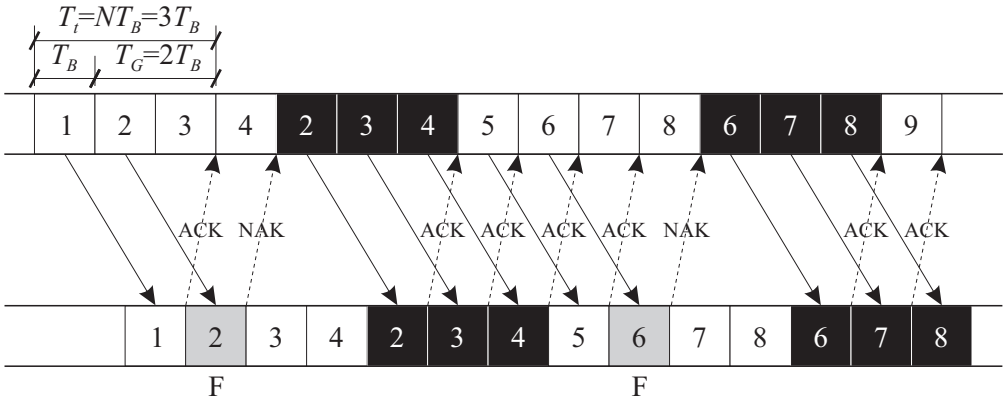


Bild 3.3: Funktionsweise des *Go-Back*-N-Verfahrens, $N = 3$

Der entscheidende Parameter $N = \lceil T_t/T_B \rceil = \lceil T_G/T_B \rceil + 1$ gibt an, wieviele Pakete im Fehlerfall zu wiederholen sind und hängt wie auch T_G vom *round trip delay* ab. Der Vorteil besteht in einem höheren Datendurchsatz gegenüber der SW-Methode. Allerdings wird im Sender mehr Speicher zum Puffern der gesendeten Blöcke benötigt, da nun die letzten N Blöcke, für die noch kein ACK empfangen wurde, gesichert werden müssen. Dies erfordert auch einen höheren Protokollaufwand.

Datendurchsatz bei idealem Rückkanal
Für das GB-N-Verfahren gilt:

Fehlerfrei	$P_c = 1 - P_{ed}$	T_B
1 Wiederholung	$P_c = P_{ed}(1 - P_{ed})$	$(N+1)T_B$
2 Wiederholungen	$P_c = P_{ed}^2(1 - P_{ed})$	$(2N+1)T_B$
3 Wiederholungen	$P_c = P_{ed}^3(1 - P_{ed})$	$(3N+1)T_B$

Wir erhalten

$$\begin{aligned} T_{AV} &= (1-P_{ed}) \cdot T_B + P_{ed}(1-P_{ed}) \cdot (N+1)T_B + P_{ed}^2(1-P_{ed}) \cdot (2N+1)T_B + \cdots \\ &= T_B(1-P_{ed}) \cdot \sum_{i=0}^{\infty}(iN+1) \cdot P_{ed}^i = T_B(1-P_{ed}) \cdot \frac{1+(N-1)P_{ed}}{(1-P_{ed})^2} \\ &= T_B \cdot \frac{1+(N-1)P_{ed}}{(1-P_{ed})} \, . \end{aligned} \tag{3.6}$$

Die Effizienz η lautet nun

$$\eta_{GB-N} = \frac{T_B}{T_{AV}} \cdot R_c = \frac{1-P_{ed}}{1+(N-1)P_{ed}} \cdot R_c \, . \tag{3.7}$$

Für $P_{ed} \to 0$ strebt η_{GB-N} gegen die Coderate R_c des CRC-Codes.

3.3.3 *Selective Repeat*-Verfahren (SR)

Das SR-Verfahren besitzt die größte Effizienz, da es kontinuierlich die Blöcke sendet und im Fehlerfall nur die wirklich fehlerhaften Pakete wiederholt. Diese Prozedur ist in Bild 3.4 illustriert.

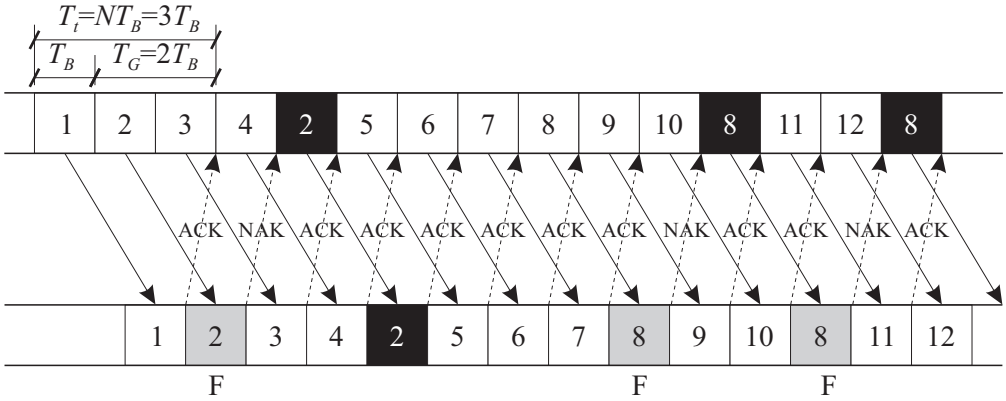


Bild 3.4: Funktionsweise des *Selective-Repeat*-Verfahrens

Es ist sofort vorstellbar, dass dieses Verfahren den größten Datendurchsatz verspricht. Allerdings ist der erhöhte Aufwand nicht zu unterschätzen. Zunächst bringt das SR-Prinzip einen Mehraufwand bzgl. der Protokollebene mit sich. Alle Pakete müssen durchnumeriert werden, damit sie im Empfänger wieder in die richtige Reihenfolge gebracht werden können und der Sender im Fehlerfall überhaupt weiß, welcher Block zu wiederholen ist.

Außerdem ist theoretisch ein unendlich großer Puffer im Empfänger erforderlich, da nach dem Empfang eines inkorrekten Blocks stets neue Pakete gesendet werden, die solange zwischengespeichert werden müssen, bis das entsprechende Paket fehlerfrei empfangen wurde. Kommt es mehrfach falsch am Empfänger an, vervielfacht sich der erforderliche Speicher. In der Realität steht im Empfänger nur ein endlich großer Speicher zur Verfügung, so dass es während schlechter Übertragungsbedingungen (viele Wiederholungen) zum Überlaufen

des Puffers und damit zu einem Datenverlust, da fehlerfrei empfangene Pakete im Sender nicht mehr gespeichert werden.

Datendurchsatz bei idealem Rückkanal
Für das SR-Verfahren gilt:

Fehlerfrei	$P_c = 1 - P_{ed}$	T_B
1 Wiederholung	$P_c = P_{ed}(1 - P_{ed})$	$2T_B$
2 Wiederholungen	$P_c = P_{ed}^2(1 - P_{ed})$	$3T_B$
3 Wiederholungen	$P_c = P_{ed}^3(1 - P_{ed})$	$4T_B$

Wir erhalten

$$\begin{aligned} T_{AV} &= (1 - P_{ed}) \cdot T_B + P_{ed}(1 - P_{ed}) \cdot 2T_B + P_{ed}^2(1 - P_{ed}) \cdot 3T_B + \cdots \\ &= T_B(1 - P_{ed}) \cdot \sum_{i=0}^{\infty}(i + 1) \cdot P_{ed}^i \\ &= T_B \cdot \frac{1 - P_{ed}}{(1 - P_{ed})^2} \\ &= \frac{T_B}{(1 - P_{ed})} \end{aligned} \tag{3.8}$$

Die Effizienz η lautet nun

$$\eta_{SR} = (1 - P_{ed}) \cdot R_c \tag{3.9}$$

Für $P_{ed} \to 0$ strebt η_{SR} gegen die Coderate R_c des CRC-Codes.

3.3.4 Kombination von *Selective Repeat*-Verfahren und *Go-Back*-N

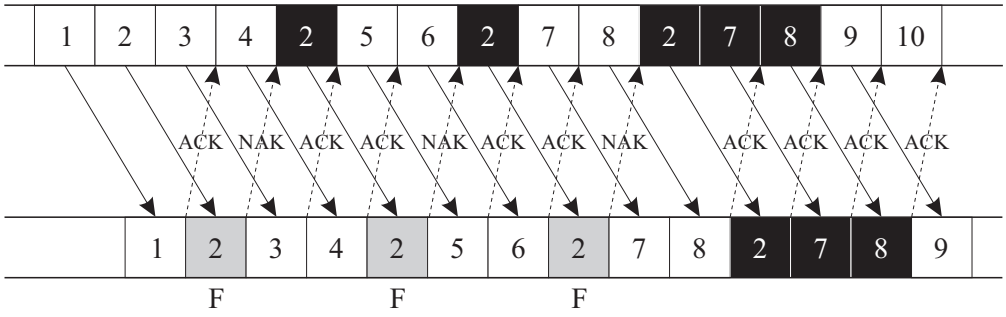


Bild 3.5: Funktionsweise des *Selective-Repeat/Go-Back-*N-Verfahrens, $N = 3$

Um diesen in der Praxis stark ins Gewicht fallenden Nachteil zu umgehen, kann eine Kombination von SR und GB-N verwendet werden. Dem liegt folgende Vorgehensweise zugrunde (s. Bild 3.5):

- Die Übertragung startet zunächst nach dem SR-Verfahren
- Wird bei schlechten Kanalbedingungen ein Block mehrmals fehlerhaft empfangen, erfolgt eine Umschaltung auf das *Go-Back-*N-Prinzip

$\longrightarrow$ Von nun an werden die N letzten Blöcke im Fehlerfall wiederholt. Dies reduziert zwar den Datendurchsatz, allerdings wird ein Pufferüberlauf im Empfänger verhindert, da zwischen den Wiederholungen keine neuen, sondern stets die gleichen Pakete übertragen werden.

3.3.5 *Selective Repeat*-Verfahren mit *Stutter*-Modus

Eine andere Möglichkeit, das Überlaufen des Puffers zu verhindern, besteht im Umschalten vom SR-Verfahren in einen *Stotter*-Modus (engl. *stutter*), der den bereits mehrfach fehlerhaft empfangenen Block solange wiederholt, bis er schließlich fehlerfrei übertragen wurde. Diese Methode ist einfacher als die letzte zu implementieren, ihr Durchsatz ist aber auch geringer, da der Stotter-Modus prinzipiell dem SW-Verfahren entspricht. Die schematische Darstellung zeigt Bild 3.6.

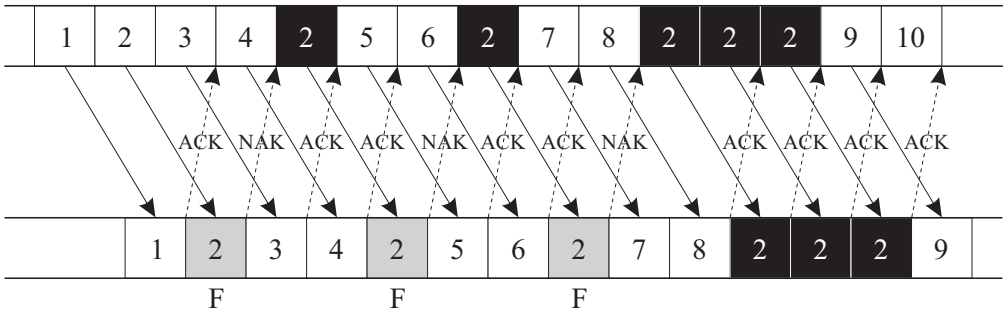


Bild 3.6: Funktionsweise des *Selective-Repeat/Stutter*-Verfahrens

3.3.6 Vergleich der ARQ-Strategien

In diesem Abschnitt sollen die Durchsatzraten η der verschiedenen ARQ-Strategien für unterschiedliche Anwendungsbereiche verglichen werden. Zum einen wurde eine typische Satellitenverbindung über geostationäre Satelliten betrachtet, zum anderen eine Richtfunkstrecke. Beide Übertragungskanäle können in guter Näherung als AWGN-Kanäle aufgefaßt werden, d.h. lediglich additives weißes gaußverteiltes Rauschen stört die Übertragung. Der wesentliche Unterschied besteht in der Entfernung zwischen Sender und Empfänger und damit im *round trip delay*.

Geostationäre Satelliten befinden sich in einer Umlaufbahn ca. 36.000 km über dem Äquator. Gehen wir von dem Szenarium aus, dass die Verbindung Erde-Satellit-Erde für Hin- und Rückkanal verwendet wird, ergibt sich eine Gesamtstrecke von

$$s = 4 \cdot 36.000 \text{ km} = 144.000 \text{ km} ,$$

die einer Laufzeit des Signals von

$$T_G = \frac{s}{c} = \frac{144 \cdot 10^6 \text{ m}}{3 \cdot 10^8 \text{m/s}} = 0.48 \text{ s} .$$

Die Effizienz der ARQ-Verfahren SW und GB-N hängt vom Verhältnis von T_G und der Paketdauer T_B ab. Nehmen wir eine Blockdauer von $T_B = 20$ ms an, wie es in der Sprachübertragung üblich ist, so gilt $N = 25$. Bei kürzeren Paketzeiten wie z.B. $T_B = 6$ ms gilt dagegen $N = 81$. Demgegenüber treten bei einer Richtfunkstrecke nahezu keine Verzögerungszeiten auf, wir nehmen daher in diesem Fall $N = 2$ an.

Bild 3.7 zeigt die Durchsatzraten η der drei ARQ-Strategien für die oben diskutierten Anwendungsbereiche. Dabei wurde ein einfacher (127,71)-BCH-Code angenommen. Der konkrete Code beeinflußt die Kurven lediglich bzgl. der Coderate und damit der asymptotischen Effizienz und hinsichtlich der Berechnung der Fehlerwahrscheinlichkeit P_{ed}.

Es ist zu erkennen, dass das *Selective Repeat*-Verfahren unabhängig von der systembedingten Verzögerungszeit ist. Für die beiden anderen Strategien gilt, dass die Effizienz η mit sinkender Verzögerungszeit T_G zunimmt. Dabei kommt das *Go Back-N*-Verfahren für $N = 2$ dem SR-Verfahren schon recht nahe. Der SW-Algorithmus besitzt aufgrund der ständig zugeführten Redundanz (T_G/T_B) mit Abstand die geringste Effizienz.

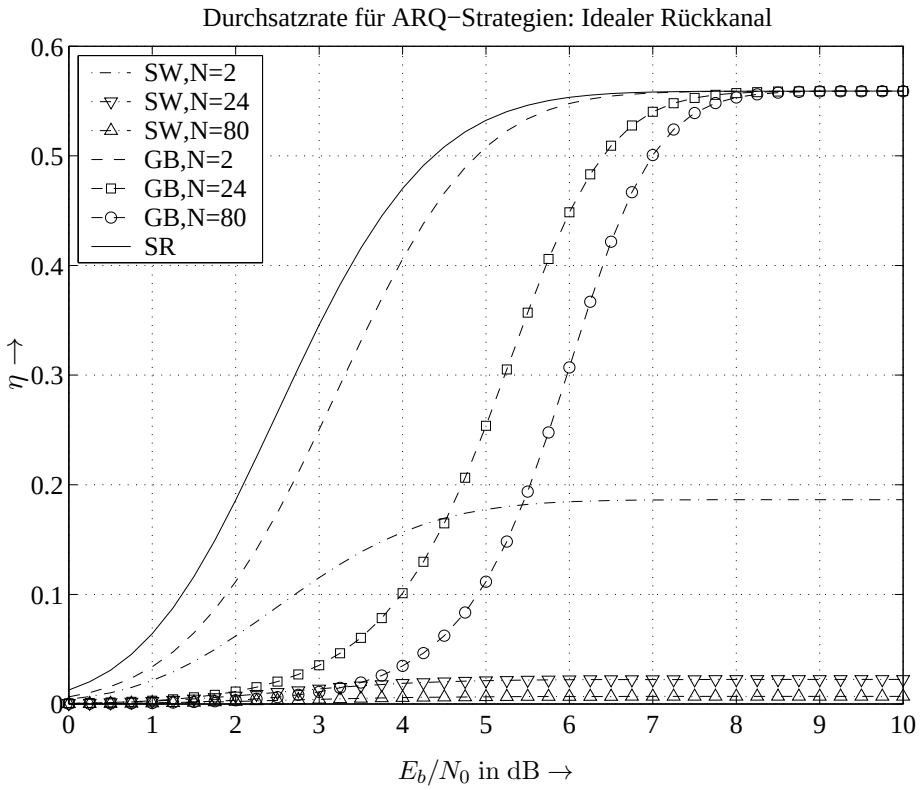


Bild 3.7: Vergleich der ARQ-Strategien

3.4 Leistungsfähigkeit bei realem Rückkanal

3.4.1 Modellbildung

Die bisherigen Ergebnisse galten alle unter der idealisierenden Annahme eines idealen (störungsfreien) Rückkanals. Dies ist in der Realität aber nie gegeben, so dass man für praktische Systeme auch Fehler im Rückkanal berücksichtigen muss. Diese Fehler führen dazu, dass vom Empfänger gesendete NAK- und ACK-Signale entweder miteinander vertauscht werden (NAK $\rightarrow$ ACK bzw. ACK $\rightarrow$ NAK) oder aber komplett verloren gehen und damit gar nicht den Sender erreichen. Daher werden i.a. folgende Gegenmaßnahmen ergriffen:

- **Zeitreferenz im Sender:**
  trifft die Antwort des Empfängers über einen gesendeten Block nicht in einem definierten Zeitintervall ein, so wird sicherheitshalber von einer erneuten Anforderung ausgegangen und der entsprechende Block erneut übertragen.
- **Zeitreferenz im Empfänger:**
  Falls nach einem gesendeten NAK der angeforderte Block nicht in einer definierten Zeit erneut am Empfänger ankommt, wird erneut ein NAK gesendet.
- Falls ein Codewort im Empfänger ankommt, dass bereits als korrekt deklariert wurde, wird abermals ein ACK gesendet und das Duplikat verworfen.

In der jetzt folgenden Analyse wollen wir annehmen, dass keine NAK-/ACK-Signale verloren gehen, sondern dass sie nur miteinander vertauscht werden können. Wir stellen uns dann das Übertragungssystem durch ein Zustandsdiagramm vor (Mealy-Automat), dessen Zustandsübergängen die Parameter a bis g zugeordnet sind (s. Bild 3.8). Welche Werte a bis g konkret annehmen, hängt von der jeweils gesuchten Zielfunktion ab.

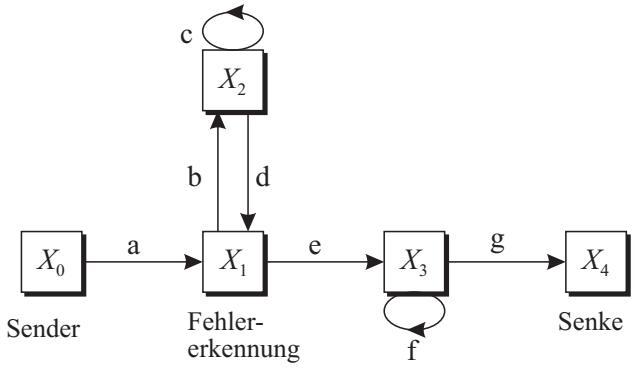


Bild 3.8: Zustandsdiagramm zur Berücksichtigung des realen Rückkanals

Der Sender (X_0) überträgt einen Datenblock (a), der im Empfänger zunächst durch Syndromdecodierung auf Fehlerfreiheit überprüft wird (X_1). Wird ein Fehler erkannt (b), muss ein NAK-Signal an den den Sender zurückgegeben werden (X_2). Wird das NAK-Signal auf dem Rückkanal verfälscht, muss es erneut übertragen werden (c), andernfalls kommt der wiederholte Block am Empfänger an (d) und die Fehlererkennung beginnt von neuem. Wird kein Fehler detektiert (e), gelangen wir in den Zustand X_3, wo nun das ACK-Signal an den Sender geschickt wird. Wird dieses verfälscht (f), erfolgt eine Wiederholung, ansonsten (g) gelangen wir zur Senke X_4.

Je nachdem, ob nun Zuverlässigkeit oder der Datendurchsatz der verschiedenen ARQ-Strategien bestimmt werden sollen, sind unterschiedliche Werte für die Parameter a bis g anzusetzen. Das Ziel besteht in allen Fällen darin, die Übertragungsfunktion vom Sender X_0 zur Senke X_4 zu bestimmen. Dazu existieren verschiedene Lösungsansätze. Eine Möglichkeit besteht in der Mason'schen Pfad-Schleifen-Regel, die ganz allgemein für beliebige Netzwerke angesetzt werden kann. Für dieses einfache Beispiel ist es aber ausreichend, das lineare Gleichungssystem aufzustellen und zu lösen.

$$\begin{aligned}
X_1 &= aX_0 + \quad + dX_2 + \quad \Rightarrow \quad X_1 = aX_0 + \tfrac{bd}{1-c}X_1 \\
X_2 &= \quad + bX_1 + cX_2 + \quad \Rightarrow \quad X_2 = \tfrac{b}{1-c}X_1 \\
X_3 &= \quad + eX_1 + \quad + fX_3 \quad \Rightarrow \quad X_3 = \tfrac{e}{1-f}X_1 \\
X_4 &= \quad + \quad + \quad + gX_3 \quad \Rightarrow \quad X_4 = \tfrac{eg}{1-f}X_1
\end{aligned}$$

Wir erhalten schließlich folgende Übertragungsfunktion

$$H = \frac{X_4}{X_0} = \frac{aeg(1-c)}{(1-f)(1-c-bd)} \; . \tag{3.10}$$

3.4.2 Zuverlässigkeit bei realem Rückkanal

In Abschnitt 3.2 wurde gezeigt, dass die Zuverlässigkeit eines ARQ-Systems unabhängig von den Kanaleigenschaften ist und lediglich von der Leistungsfähigkeit des eingesetzten fehlererkennenden Codes beeinflußt wird. Es gilt nun zu überprüfen, ob dies auch für reale, d.h. gestörte Rückkanäle gilt. Dazu setzten wir folgende Ausdrücke für die Parameter ein:

$$a = 1 \tag{3.11}$$

$$b = P_{ed} \tag{3.12}$$

$$c = P_{NA} \qquad \text{Wahrscheinlichkeit für Fehler NAK } \rightarrow \text{ ACK} \tag{3.13}$$

$$d = 1 - P_{NA} \quad \text{Wahrscheinlichkeit für korrekten Empfang von NAK} \tag{3.14}$$

$$e = P_{ue} \tag{3.15}$$

$$f = P_{AN} \quad \text{Wahrscheinlichkeit für Fehler ACK } \rightarrow \text{ NAK} \tag{3.16}$$

$$g = 1 - P_{AN} \quad \text{Wahrscheinlichkeit für korrekten Empfang von ACK} \tag{3.17}$$

Ein erkennbarer Fehler tritt mit der Wahrscheinlichkeit P_{ed} auf. In diesem Fall wird ein NAK-Signal gesendet, welches mit der Wahrscheinlichkeit P_{NA} auf dem Rückkanal verfälscht wird. Dann bleibt die Wiederholung des fehlerhaften Datenpakets aus und das NAK muss erneut übertragen werden. Die Prozedur wiederholt sich solange (Selbstschleife c in X_2), bis das NAK korrekt am Sender angekommen ($d = 1 - P_{NA}$) ist und dieser das fehlerhafte Paket erneut überträgt. Dann beginnt in X_1 die Fehlererkennung von neuem.

Die Wahl von $e = P_{ue}$ in Gl. (3.15) bedeutet, dass nur die Fälle betrachtet werden, in denen der fehlererkennende Code versagt. Gl. (3.10) liefert dann die Restfehlerwahrscheinlichkeit. Je kleiner sie ist, desto größer ist die Zuverlässigkeit. Wird kein Fehler detektiert, muss das ACK-Signal gesendet werden, das aber auch verfälscht werden kann und dann ebenfalls wiederholt werden muss (Selbstschleife f in X_3). Dies hat für die Zuverlässigkeit des Systems zunächst keine Bedeutung, sehr wohl aber für den Datendurchsatz, der in den folgenden Abschnitten betrachtet wird.

Setzen wir die oben aufgeführten Parameter in Gl. (3.10) ein, ergibt sich

$$H = \frac{P_{ue}(1-P_{AN})(1-P_{NA})}{(1-P_{AN})(1-P_{NA}-P_{ed}(1-P_{NA}))} = \frac{P_{ue}(1-P_{NA})}{(1-P_{ed})(1-P_{NA})} = \frac{P_{ue}}{1-P_{ed}} \, . \tag{3.18}$$

Dies ist das schon aus Gl. (3.2) bekannte Resultat. Damit wird deutlich, dass auch der reale Rückkanal keinen Einfluß auf die Zuverlässigkeit, also die Restfehlerwahrscheinlichkeit des ARQ-Systems, hat.

3.4.3 Datendurchsatz beim SW-Verfahren

Soll der Datendurchsatz, also die Effizienz der ARQ-Systeme berechnet werden, kommt es darauf an, wie lange die Übertragung eines Datenpakets **im Mittel** dauert. Wir definieren dazu das Zeitmaß

$$\kappa = \frac{T_B + T_G}{T_B} \, , \tag{3.19}$$

das die auf die Paketdauer T_B normierte Übertragungsdauer eines Pakets $T_B + T_G$ im fehlerfreien Fall beschreibt. Damit sich durch die Multiplikation der Übergangsparameter die Zeiten der einzelnen Vorgänge addieren, wird ein Platzhalter D eingeführt, dessen Exponent den zeitlichen Einfluß eines Zustandsübergangs darstellt. Wir erhalten folgende Zuordnungen:

$$a = D^{\kappa} \tag{3.20}$$

$$b = P_{ed} \tag{3.21}$$

$$c = P_{NA} \cdot D^{\kappa} \tag{3.22}$$

$$d = (1 - P_{NA}) \cdot D^{\kappa} \tag{3.23}$$

$$e = 1 - P_{ed} \tag{3.24}$$

$$f = P_{AN} \cdot D^{\kappa} \tag{3.25}$$

$$g = 1 - P_{AN} \tag{3.26}$$

Die Übertragung eines Blocks dauert bezogen auf T_B beim SW-Verfahren κ Zeiteinheiten. Gleiches gilt für die Übertragung der NAK- und ACK-Signale. Werden sie auf dem Rückkanal verfälscht, wir ein fehlerhafter

Block nicht wiederholt (der tatsächlich gesendete wird verworfen) oder aber ein korrekter Block unnötigerweise mehrfach übertragen, was den Datendurchsatz reduziert, d.h. die effektive Dauer für die Übertragung eines Blocks erhöht.

Die Wahl von $e = 1 - P_{ed}$ bedeutet, dass wir als fehlererkennenden Code einen Genie-Code annehmen, der alle möglichen Fehler erkennt. Sie resultiert aus der Überlegung, dass bei einem nicht erkannten Fehler das System sowieso versagt und somit die aufgetretenen Verzögerungen uninteressant sind. Bzgl. der Effizienz konzentrieren wir uns somit auf den Fall, dass das System perfekt funktioniert. Wir erhalten

$$H_{SW}(D) = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA}D^{\kappa})D^{\kappa}}{(1-P_{NA}D^{\kappa}-P_{ed}(1-P_{NA})D^{\kappa})(1-P_{AN}D^{\kappa})} \tag{3.27}$$

Die mittlere Übertragungszeit je Block steckt implizit in den Exponenten von D. Wie schon bei der Transferfunktion von Faltungscodes müssen wir Gl. (3.27) nach D differenzieren und die Ableitung an der Stelle $D = 1$ auswerten. Für die mittlere Paketübertragungsdauer ergibt sich

$$\frac{T_{AV}}{T_B} = \left.\frac{\partial H_{SW}(D)}{\partial D}\right|_{D=1} = \frac{\kappa(1-P_{ed}P_{AN}-P_{NA}+P_{ed}P_{NA})}{(1-P_{ed})(1-P_{AN})(1-P_{NA})} \tag{3.28}$$

Die Durchsatzrate beinhaltet auch noch die Coderate des Kanalcodierungsverfahrens, so dass die Effizienz der *Stop&Wait*-Strategie insgesamt

$$\eta = \frac{T_B}{T_{AV}} \cdot R_c = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA})}{(1+\frac{T_G}{T_B})(1-P_{ed}P_{AN}-P_{NA}+P_{ed}P_{NA})} \cdot R_c \tag{3.29}$$

lautet. Betrachten wir nun zwei Spezialfälle, zum einen den eines idealen Rückkanals ($P_{AN} = P_{NA} = 0$), zum anderen den eines symmetrischen Rückkanals, d.h. ($P_{AN} = P_{NA}$):

$$P_{AN} = P_{NA} = 0 \quad \Longrightarrow \quad \eta = \frac{1-P_{ed}}{1+\frac{T_G}{T_B}} R_c = \eta_I$$

$$P_{AN} = P_{NA} \quad \Longrightarrow \quad \eta = \frac{(1-P_{ed})(1-P_{AN})}{1+\frac{T_G}{T_B}} R_c = (1-P_{AN})\eta_I$$

3.4.4 Datendurchsatz beim GB-N-Verfahren

Für das *Go-Back-N*-Verfahren sind die Parameter a bis g etwas anders zu wählen. Zunächst entfällt die Leerlaufzeit T_G, da die Datenpakete ohne Pause hintereinander gesendet werden. Im fehlerfreien Fall dauert die Übertragung eines Blocks also nur noch T_B (anstatt $T_B + T_G$). Tritt ein erkennbarer Fehler auf, werden dagegen N Blöcke erneut übertragen, nicht mehr nur der fehlerhafte Block (siehe d). Wird das dazu erforderliche NAK-Signal durch den Rückkanal verfälscht, muss es erneut gesendet werden. Dies führt dazu, dass ein Block mehr wiederholt werden muss, weshalb dieser Fall zu einer Verzögerung von einer Paketdauer führt (siehe c). Die Annahme eines idealen Genie-Codes wird weiterhin aufrecht gehalten.

$$a = D \tag{3.30}$$

$$b = P_{ed} \tag{3.31}$$

$$c = P_{NA} \cdot D \tag{3.32}$$

$$d = (1-P_{NA}) \cdot D^N \tag{3.33}$$

$$e = 1 - P_{ed} \tag{3.34}$$

$$f = P_{AN} \cdot D^N \tag{3.35}$$

$$g = 1 - P_{AN} \tag{3.36}$$

Wir erhalten folgende Übertragungsfunktion:

$$H_{GB-N}(D) = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA}D)D}{(1-P_{NA}D-P_{ed}(1-P_{NA})D^N)(1-P_{AN}D^N)} \tag{3.37}$$

Die mittlere Übertragungszeit pro erfolgreich übertragenem Paket lautet

$$\frac{T_{AV}}{T_B} = \frac{1-P_{NA}-P_{ed}P_{AN}+P_{ed}P_{NA}+(N-1)(P_{AN}+P_{ed}-P_{AN}P_{NA}-P_{ed}P_{NA}-2P_{ed}P_{AN}+2P_{ed}P_{AN}P_{NA})}{(1-P_{ed})(1-P_{AN})(1-P_{NA})} \tag{3.38}$$

und die spektrale Effizienz beträgt

$$\eta = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA})}{1-P_{NA}-P_{ed}P_{AN}+P_{ed}P_{NA}+(N-1)(P_{AN}+P_{ed}-P_{AN}P_{NA}-P_{ed}P_{NA}-2P_{ed}P_{AN}+2P_{ed}P_{AN}P_{NA})} R_c \; . \tag{3.39}$$

Für die schon im letzten Abschnitt betrachteten Spezialfälle $P_{AN} = P_{NA}$ und $P_{AN} = P_{NA} = 0$ erhalten wir

$$P_{AN} = P_{NA} \quad \Longrightarrow \quad \eta = \frac{(1-P_{ed})(1-P_{AN})}{1+(N-1)(P_{AN}+P_{ed}-2P_{AN}P_{ed})} R_c$$

$$P_{AN} = P_{NA} = 0 \quad \Longrightarrow \quad \eta = \frac{1-P_{ed}}{1+(N-1)P_{ed}} R_c$$

3.4.5 Datendurchsatz beim SR-Verfahren

Für das *Selective-Repeat*-Verfahren werden die Parameter wie folgt festgesetzt:

$$a = D \tag{3.40}$$

$$b = P_{ed} \tag{3.41}$$

$$c = P_{NA} \tag{3.42}$$

$$d = (1-P_{NA}) \cdot D \tag{3.43}$$

$$e = 1-P_{ed} \tag{3.44}$$

$$f = P_{AN} \cdot D \tag{3.45}$$

$$g = 1-P_{AN} \tag{3.46}$$

Die leichten Unterschiede zum GB-N-Verfahren sind so zu erklären: Wird im Fehlerfall das NAK in ein ACK verfälscht, läuft die Übertragung irrtümlicherweise ohne Wiederholung weiter. Dies führt aber nicht zu einer zusätzlichen Verzögerung, da keine bereits korrekt empfangenen Pakete verworfen werden (siehe c, lediglich Überlauf des Puffers möglich), sondern nur das betroffene Paket wiederholt wird (siehe d). Die versehentliche Wiederholung eines schon korrekt übertragenen Blocks verursacht dagegen für beide ARQ-Strategien eine Verzögerung von einer Paketdauer.

Wir erhalten folgende Übertragungsfunktion:

$$H_{SR}(D) = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA})D}{(1-P_{NA}-P_{ed}(1-P_{NA})D)(1-P_{AN}D)} \tag{3.47}$$

Die mittlere Übertragungszeit pro erfolgreich übertragenem Paket lautet

$$\frac{T_{AV}}{T_B} = \frac{1-P_{ed}P_{AN}}{(1-P_{ed})(1-P_{AN})} \tag{3.48}$$

und die spektrale Effizienz beträgt

$$\eta = \frac{(1-P_{ed})(1-P_{AN})}{1-P_{ed}P_{AN}} R_c \; . \tag{3.49}$$

3.4.6 Vergleich der ARQ-Strategien

Beim Vergleich der ARQ-Strategien soll zunächst erst einmal der Einfluß des störungsbehafteten Rückkanals auf die jeweiligen Verfahren untersucht werden. Dazu zeigen die Bilder 3.9 bis 3.11 die Datendurchsätze η für verschiedene *round trip delays* N verschiedene Fehlerwahrscheinlichkeiten P_{AN}. Es ist ersichtlich, dass das *Stop & Wait*-Verfahren selbst bei zuverlässigen Rückkanälen ($P_{AN} = 10^{-4}$) den geringsten Datendurchsatz besitzt (vgl. idealer Rückkanal). Dementsprechend wirkt sich auch eine Degradation der Qualität des Rückkanals kaum aus.

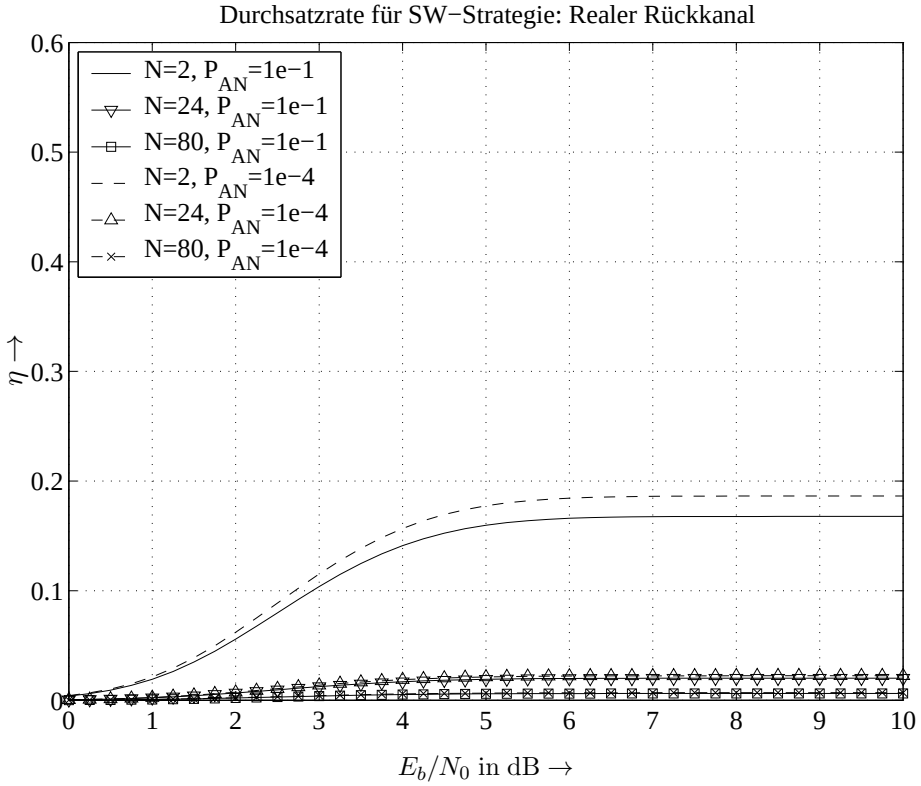


Bild 3.9: Vergleich verschiedener Fehlerwahrscheinlichkeiten P_{AN} für die SW-Strategie

Beim GB-N-Verfahren nähern sich die Kurven verschiedener N für einen zuverlässigen Rückkanal ($P_{AN} = 10^{-4}$) asymptotisch an. Weist der Rückkanal allerdings ein hohes Fehlerpotential auf ($P_{AN} = 10^{-2}$), so wird die Effizienz nachhaltig reduziert, d.h. selbst für gegen unendlich strebende Signal-Rausch-Abstände wird nicht die gleiche Effizienz erreicht wie für gute Rückkanäle. Dieser Effekt wirkt sich um so stärker aus, je größer der Parameter N, also die Anzahl der zu wiederholenden Blöcke, ist.

Das SR-Verfahren besitzt die geringste Empfindlichkeit gegenüber gestörten Rückkanälen. Dies war auch zu erwarten, da bei unnötig wiederholten Blöcken (ACK $\rightarrow$ NAK) nicht gleich N Pakete wiederholt werden, sondern nur ein einziger, wodurch sich der Durchsatz nicht dramatisch reduziert. Für den Fall (NAK $\rightarrow$ ACK) wird der empfangene Block im Empfänger nicht verworfen, sondern zwischengespeichert, so dass dies die Effizienz gar nicht beeinflußt (siehe Unabhängigkeit von η von P_{NA} in Gl. (3.49)). Allerdings bleibt festzustellen, dass die SR-Strategie nur in der Realität so nicht umzusetzen ist, da ein unendlich großer Puffer benötigt würde.

Für Übertragungssysteme mit kurzen Reichweiten, z.B. die erwähnte Richtfunkstrecke mit $N = 2$, zeigt Bild 3.12 einen direkten Vergleich der diskutierten ARQ-Strategien. Es ist erkennbar, dass der SR-Verfahren die größte Effizienz besitzt. Für geringe Qualitäten des Rückkanals nimmt der Unterschied zu. Ein Vergleich zwischen SR und GB-N zeigt, dass für $P_{AN} = 10^{-4}$ die Datendurchsätze asymptotisch gleich sind, während für $P_{AN} = 10^{-2}$ ein konstanter Unterschied von etwa 0.05 verbleibt. Für höhere Verzögerungszeiten ($N = 25$ bzw. $N = 81$) wäre der Unterschied zwischen SR auf der einen und SW bzw. GB-N auf der anderen Seite noch größer ausgefallen.

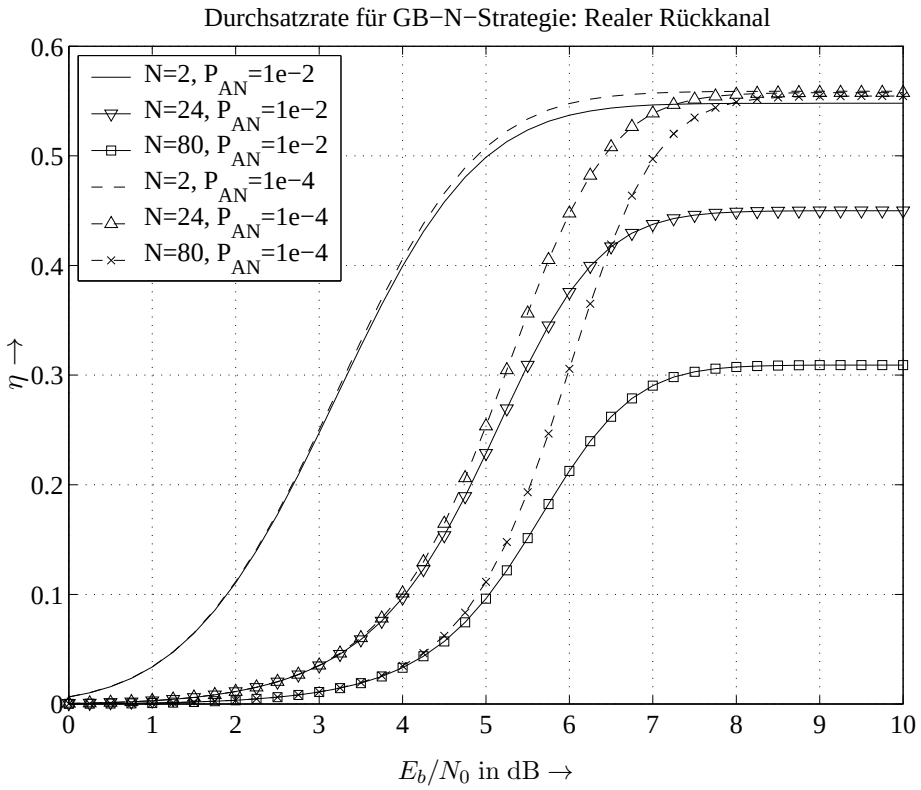


Bild 3.10: Vergleich verschiedener Fehlerwahrscheinlichkeiten P_{AN} für die GB-Strategie

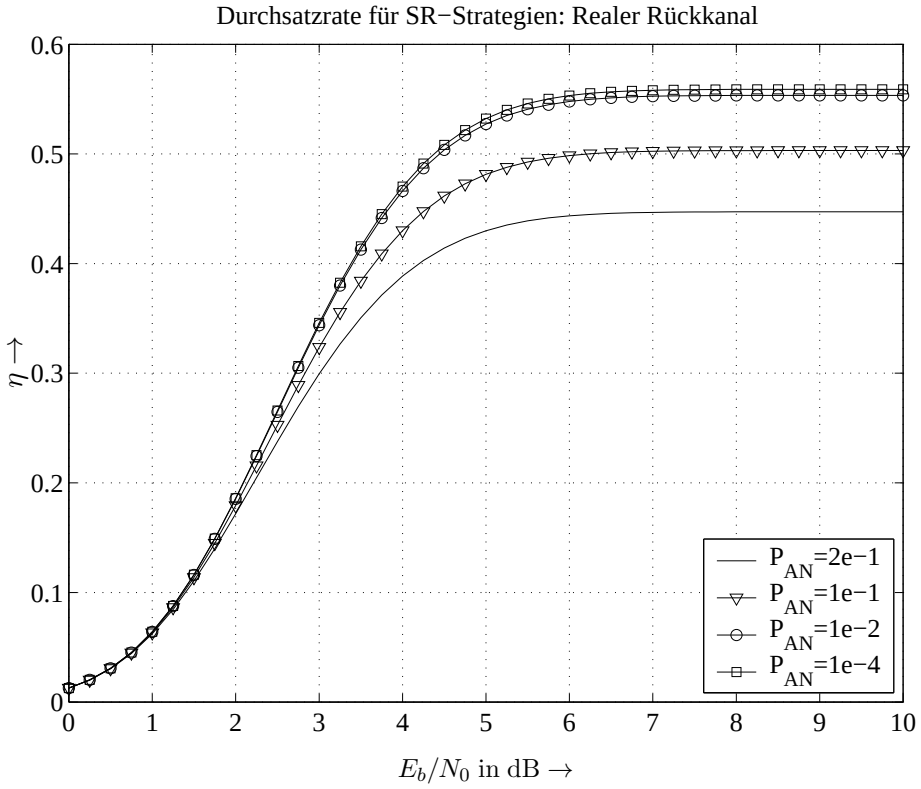


Bild 3.11: Vergleich verschiedener Fehlerwahrscheinlichkeiten P_{AN} für die SR-Strategie

3.5 Hybride FEC/ARQ-Systeme

Die beiden bisher betrachteten prinzipiellen Ansätze zur Kanalcodierung, die FEC- und die ARQ-Verfahren, besitzen, jedes für sich genommen, eine Reihe von Nachteilen. So kann man zwar mit reinen FEC-Verfahren

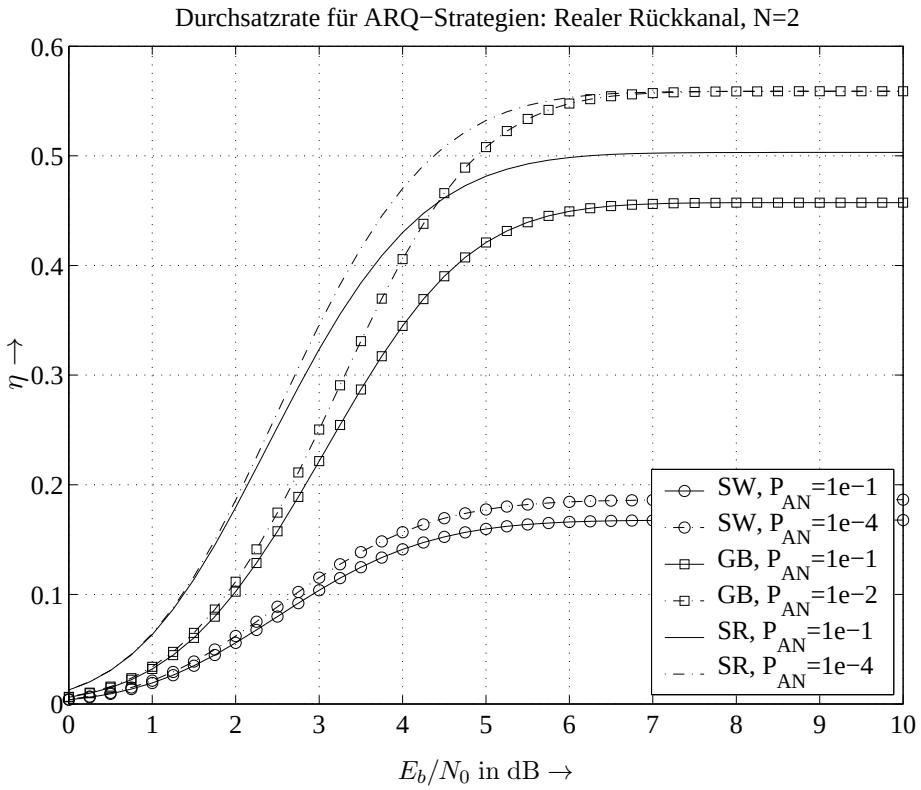


Bild 3.12: Vergleich der ARQ-Strategien für reale Rückkanäle mit $N = 2$

sehr niedrige Fehlerraten ab bestimmten Signal-Rausch-Abständen erzielen, für schlechtere Kanalbedingungen können sie jedoch keine fehlerfreie Übertragung garantieren. Andererseits fügen sie bei guter Kanalqualität viel mehr Redundanz zu, als eigentlich nötig wäre und reduzieren damit den möglichen Datendurchsatz.

Reine ARQ-Verfahren verhalten sich grundlegend anders. Mit einem entsprechend leistungsfähigen CRC-Code kann eine nahezu fehlerfreie Übertragung gewährleistet werden, allerdings auf Kosten einer u.U. gegen Null strebenden Durchsatzrate. Für gute Übertragungsbedingungen erreicht man hohe Durchsatzraten, da der fehlererkennende Code im Vergleich zu korrigierenden Codes weniger Redundanz benötigt. Wird der Kanal allerdings schlechter, verringern die ständigen Wiederholungen von fehlerhaften Blöcken die Effizienz dramatisch.

Es liegt also sehr nahe, die Vorteile beider Strategien gewinnbringend miteinander zu kombinieren. Bei guten bis mittleren Kanalbedingungen sorgt der FEC-Code für eine fast fehlerfreie Übertragung, der Nachteil des häufigen Wiederholens bei reinen ARQ-Techniken ist überwunden. Treten bei schlechteren Übertragungsbedingungen doch noch Fehler auf, sorgt die ARQ-Steuerung für entsprechende Wiederholungen. Beide Verfahren ergänzen sich also hervorragend zu einem sogenannten **hybriden FEC/ARQ-System**, dessen Struktur Bild 3.13 zeigt. Der zum ARQ-System gehörende fehlererkennende Code bildet den äußeren, der FEC-Code den inneren Code einer seriellen Verkettung.

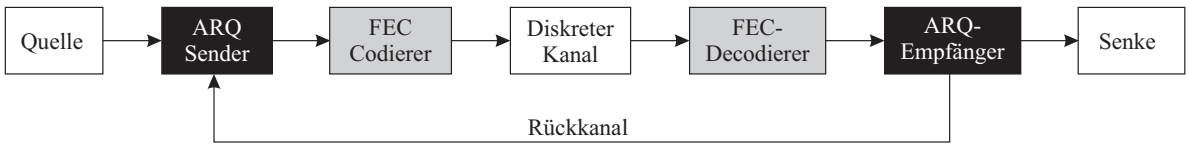


Bild 3.13: Struktur eines hybriden FEC-ARQ-Systems

3.5.1 Typ-I hybrides ARQ-System

Das sogenannte **Typ-I hybride ARQ-System** stellt die einfachste Art der Kombination von FEC und ARQ dar. Es besitzt eine gute Leistungsfähigkeit bei annähernd konstanten Übertragungsbedingungen, also quasi zeitinvarianten Kanälen. Entsprechend Bild 3.14 werden den Informationsbit zunächst die Prüfbit eines fehlererkennenden CRC-Codes angehängt (äußerer Code). Danach erfolgt die Codierung mit einem fehlerkorrigierenden FEC-Code, z.B. einem Faltungscode als inneren Code.

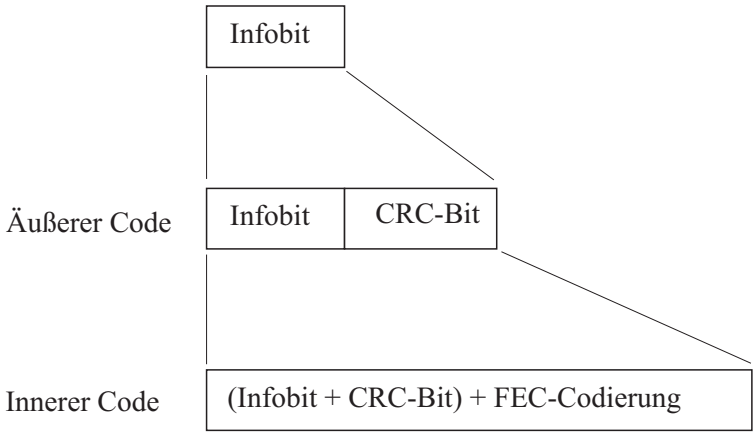


Bild 3.14: Codierung beim Typ-I hybriden ARQ-System

Die Funktionsweise ist mit der reiner ARQ-Verfahren identisch und in Bild 3.15 skizziert. Im fehlerfreien Fall erfolgt das Senden eines ACK-Signals, woraufhin der Sender den nächsten Block übermittelt. Wird ein Fehler detektiert, muss der betroffene Block wiederholt werden, inklusive der Redundanz des FEC-Codes. Der für bestimmte Kanäle erzielbare bessere Durchsatz im Vergleich zu reinen ARQ-Strategien wird durch die Möglichkeit der Fehlerkorrektur mit Hilfe des FEC-Codes erzielt. Ist dieser an den quasi konstanten Kanal angepaßt, kann eine nahezu optimale Durchsatzrate erreicht werden.

Allerdings besitzt das Typ-I hybride Verfahren weiterhin den gravierenden Nachteil reiner FEC-Systeme, dass keine ausreichende Adaption für zeitvariante Kanäle gelingt. Während guter Kanaleigenschaften wird durch den fehlerkorrigierenden Code mehr Redundanz als nötig eingebracht, die Effizienz verringert sich hier. Daher ist das Typ-I hybride Verfahren nicht für zeitvariante Kanäle geeignet. Für sie ist vielmehr eine Adaptivität auch des FEC-Codes erforderlich, auf die im folgenden Abschnitt näher eingegangen wird.

3.5.2 Hybrides ARQ-System mit ratenkompatiblen Faltungscodes

gesucht:	bessere Adaptivität der FEC-Komponente
	guter Kanal $\rightarrow$ wenig Redundanz, z.B. reines ARQ-System
	schlechter Kanal $\rightarrow$ viel Redundanz durch fehlerkorrigierenden Code
Lösung:	Redundanz des FEC-Codes wird nicht auf einmal übertragen, sondern sukzessive, je nachdem, wie oft Fehler aufgetreten sind und Wiederholungen angefordert werden. Wurde sofort fehlerfrei übertragen, reduziert sich hierdurch die übertragene Redundanz.

Für dieses ARQ-Prinzip eignen sich hervorragend punktierte Faltungscodes. Sie wurden für diese Anwendung erstmals von Hagenauer [Hag88] als **ratenkompatible punktierte Faltungscodes** (*Rate-Compatible Punctured Convolutional Codes*, RCPC-Codes) vorgestellt. Den Basiscode bildet ein gewöhnlicher Faltungscode mit der Coderate $R_c = 1/4$. Für ihn werden dann mehrere Punktierungsmatrizen $\mathbf{P}_l$ mit den Elementen $p_{i,j}(l)$

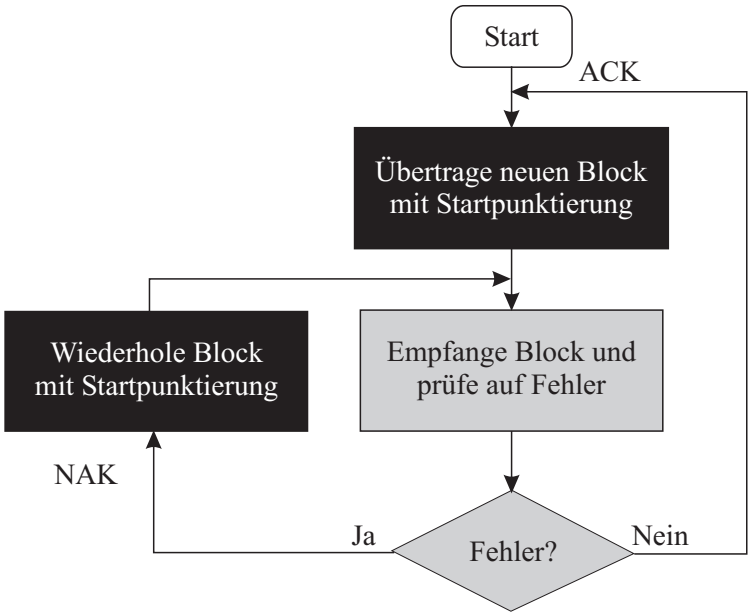


Bild 3.15: Protokollablauf beim Typ-I hybriden ARQ-System

entworfen, von denen jede eine bestimmte Coderate

$$R_c^{(l)} = \frac{L_P}{L_P + l} \tag{3.50}$$

repräsentiert, wobei L_P die Punktierungsperiode darstellt und der Parameter l im Intervall $1 \cdots (n-1)L_P$ liegt. Mit sinkendem Index l lassen sich dann Coderaten von $R_c = 1/4$ (keine Punktierung, $l = (n-1)L_P$) bis $R_c = 8/9$ ($l = 1$) einstellen.

Mit Hilfe der verschiedenen Punktierungsmatrizen kann zum einen Redundanz (Prüfbit) nachgesendet werden, wenn im Empfänger ein nicht korrigierbarer Fehler detektiert wurde. Weiterhin kann die Coderate schon direkt im Sender durch die Wahl einer geeigneten Matrix auf die momentanen Übertragungsbedingungen adaptiert werden. Hierzu kann z.B. eine Kanalzustandsschätzung im Empfänger dienen, deren Ergebnis über den Rückkanal dem Sender mitgeteilt wird.

Soll die Coderate gewechselt werden, ist auf die Bedingung der **Ratenkompatibilität** zu achten. Sie kann für einen Bezugsindex l_0 gemäß

$$p_{i,j}(l_0) = 1 \quad \Longrightarrow \quad p_{i,j}(l) = 1 \quad \forall\, l \geq l_0 \geq 1 \tag{3.51}$$

$$p_{i,j}(l_0) = 0 \quad \Longrightarrow \quad p_{i,j}(l) = 0 \quad \forall\, l \leq l_0 \leq (n-1)L_P - 1 \tag{3.52}$$

formuliert werden und besagt, dass bei einem Wechsel der Punktierungsmatrix von l zu $l+i$ (mehr Redundanz) alle bisher übertragenen Stellen auch weiterhin übertragen werden müssen. Es dürfen nur diejenigen Redundanzbit hinzukommen, die bisher punktiert wurden. Weiterhin dürfen bei einer Verringerung der Redundanz entsprechend Gl. (3.52) nur bisher übertragene Codestellen zusätzlich punktiert werden, die bisher punktierten Stellen werden auch weiterhin ausgeblendet. Die Ratenkompatibilität garantiert, dass beim Umschalten der Punktierungsmatrix keine Information verloren geht.

Die Realisierung des obigen Ansatzes ist in Bild 3.16 dargestellt. Im Sender werden die Informationsbit zunächst mit einem CRC-Code und dann mit einem FEC-Code niedriger Coderate (viel Redundanz) codiert. Die erste Übertragung beinhaltet aber nur einen Teil der FEC-Redundanz, der Rest wird in einem Puffer gespeichert. Stellt der Empfänger einen Fehler fest, speichert er den empfangenen Block ebenfalls und sendet ein NAK-Signal. Dies veranlaßt den Sender, eine erneute Übertragung zu initiieren. Es wird aber nicht

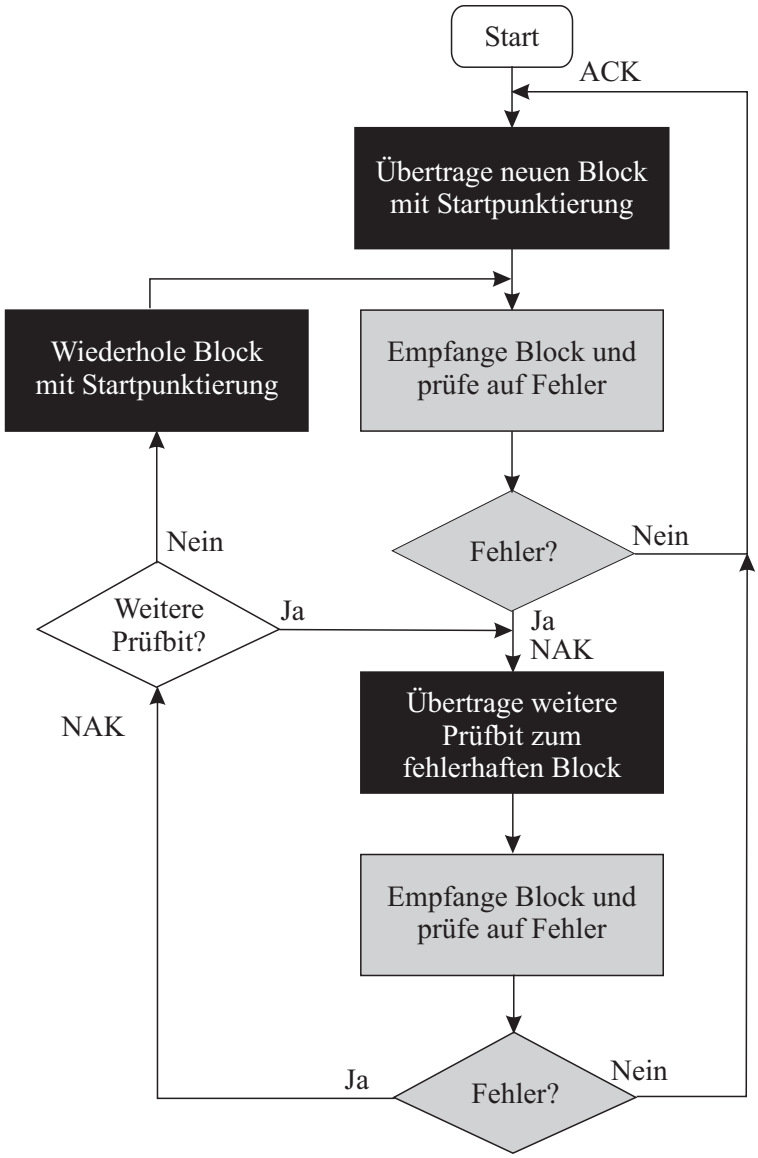


Bild 3.16: Protokollablauf beim hybriden ARQ-System mit RCPC-Codes

das ursprünglich gesendete Paket wiederholt, sondern vielmehr die bisher zurückgehaltenen Prüfbit der FEC-Codierung gesendet. Mit ihnen kann der Empfänger dann zusammen mit dem gepufferten, fehlerbehafteten Block einen erneuten Decodierversuch unternehmen. Die nun zur Verfügung stehende zusätzliche Redundanz soll jetzt zu einer fehlerfreien Decodierung führen.

Dabei werden in der Praxis nicht alle möglichen Coderaten tatsächlich genutzt. Es würde auch keinen Sinn machen, jedes Prüfbit einzelnen nachzusenden, da ein einzelnes zusätzliches Bit mit hoher Wahrscheinlichkeit nicht sofort zu einer erfolgreichen Fehlerkorrektur führt und somit mehrere Durchläufe erforderlich wären, von denen jeder die Verzögerungszeit erhöht. Vielmehr ist es sinnvoll, bestimmte Bündel zu definieren und diese im Fehlerfall zu übertragen. Dabei können beim Nachsenden der Prüfbit auch mehrere Bündel zusammengefaßt werden, um unnötige erfolglose Decodierversuche zu vermeiden. Wir müssen also einen Kompromiß zwischen möglichst geringer Redundanz und möglichst kleiner Verzögerungszeit suchen. Welche Strategie den erhofften Erfolg bringt, hängt entscheidend vom Übertragungskanal ab und kann nicht pauschal beantwortet werden.

Beispiel 1:
Basiscode mit $L_c = 5$ und $R_c = 1/4$

Generatorpolynome:

$$\begin{aligned}\mathbf{g}_0 &= 1 + D + D^4\\ \mathbf{g}_1 &= 1 + D^2 + D^3 + D^4\\ \mathbf{g}_2 &= 1 + D + D^2 + D^4\\ \mathbf{g}_3 &= 1 + D + D^3 + D^4\end{aligned}$$

Punktierungsmatrizen (Punktierungsperiode $L_P = 8 \longrightarrow$ 8 Spalten, Coderate $R_c = 1/4 \longrightarrow$ 4 Zeilen):

$$R_c = \frac{1}{4} \longrightarrow \mathbf{P}_0 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1 \end{pmatrix} \qquad R_c = \frac{4}{15} \longrightarrow \mathbf{P}_1 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&0 & 1&1&1&0 \end{pmatrix}$$

$$R_c = \frac{2}{7} \longrightarrow \mathbf{P}_2 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&0&1&0 & 1&0&1&0 \end{pmatrix} \qquad R_c = \frac{4}{13} \longrightarrow \mathbf{P}_3 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&0&0&0 & 1&0&0&0 \end{pmatrix}$$

$$R_c = \frac{1}{3} \longrightarrow \mathbf{P}_4 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix} \qquad R_c = \frac{4}{11} \longrightarrow \mathbf{P}_5 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&0 & 1&1&1&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

$$R_c = \frac{4}{10} \longrightarrow \mathbf{P}_6 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&0&0 & 1&1&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix} \qquad R_c = \frac{4}{9} \longrightarrow \mathbf{P}_7 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&0&0&0 & 1&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

$$R_c = \frac{1}{2} \longrightarrow \mathbf{P}_8 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix} \qquad R_c = \frac{4}{7} \longrightarrow \mathbf{P}_9 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&0 & 1&1&1&0\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

$$R_c = \frac{2}{3} \longrightarrow \mathbf{P}_{10} = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&0&1&0 & 1&0&1&0\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix} \qquad R_c = \frac{4}{5} \longrightarrow \mathbf{P}_{11} = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&0&0&0 & 1&0&0&0\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

$$R_c = \frac{8}{9} \longrightarrow \mathbf{P}_{12} = \begin{pmatrix} 1&1&1&1 & 0&1&1&1\\ 1&0&0&0 & 1&0&0&0\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

Beispiel 2:
Basiscode mit $L_c = 7$ und $R_c = 1/3$

Generatorpolynome:

$$\begin{aligned}\mathbf{g}_0 &= 1 + D + D^3D^4 + D^6\\ \mathbf{g}_1 &= 1 + D^3 + D^4 + D^5 + D^6\\ \mathbf{g}_2 &= 1 + D^2 + D^5 + D^6\end{aligned}$$

Punktierungsmatrizen (Punktierungsperiode $L_P = 8 \longrightarrow$ 8 Spalten, Coderate $R_c = 1/3 \longrightarrow$ 3 Zeilen):

$$R_c = \frac{1}{3} \longrightarrow \mathbf{P}_0 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \end{pmatrix} \qquad R_c = \frac{4}{11} \longrightarrow \mathbf{P}_1 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,0 & 1\,1\,1\,0 \end{pmatrix}$$

$$R_c = \frac{4}{10} \longrightarrow \mathbf{P}_2 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,0\,0 & 1\,1\,0\,0 \end{pmatrix} \qquad R_c = \frac{4}{9} \longrightarrow \mathbf{P}_3 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,0\,0\,0 & 1\,0\,0\,0 \end{pmatrix}$$

$$R_c = \frac{1}{2} \longrightarrow \mathbf{P}_4 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix} \qquad R_c = \frac{4}{7} \longrightarrow \mathbf{P}_5 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,0 & 1\,1\,1\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix}$$

$$R_c = \frac{2}{3} \longrightarrow \mathbf{P}_6 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,0\,1\,0 & 1\,0\,1\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix} \qquad R_c = \frac{4}{5} \longrightarrow \mathbf{P}_7 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,0\,0\,0 & 1\,0\,0\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix}$$

$$R_c = \frac{8}{9} \longrightarrow \mathbf{P}_8 = \begin{pmatrix} 1\,1\,1\,1 & 0\,1\,1\,1 \\ 1\,0\,0\,0 & 1\,0\,0\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix}$$

3.5.3 Typ-II hybrides System

Ein Nachteil des oben beschriebenen Systems besteht darin, dass der erste gesendete Block im Fehlerfall immer wieder für die Decodierung herangezogen wird. Zwar können die nachträglich übertragenen Prüfbit häufig zur fehlerfreien Decodierung führen, wenn das erste Paket aber sehr stark gestört ist, bleiben unter Umständen aber alle Decodierversuche erfolglos. Diesen Nachteil behebt das **Typ-II hybride ARQ-System**. Die prinzipielle Vorgehensweise ähnelt der des vorigen Verfahrens und ist in Bild 3.17 illustriert.

Auch beim Typ-II hybriden ARQ-Verfahren wird im Sender zunächst Redundanz zurückgehalten, die dann bei Eintreten eines Übertragungsfehlers nachgeliefert wird. Voraussetzung ist hier allerdings der Einsatz invertierbarer Codes, d.h. sind die Prüfbit eines Codewortes korrekt empfangen worden, kann eindeutig auf die Informationsbit zurückgeschlossen werden. Dies hat den Vorteil, dass bei zeitvarianten Kanälen die nachgesendete Redundanz unter Umständen viel besser übertragen wurde als der Originalblock und dann allein, also ohne die Nutzung der gepufferten ersten Version, fehlerfrei decodiert werden kann. In diesem Fall ist eine gemeinsame Decodierung beider Blöcke überflüssig, sie wäre sogar nachteilig. Konventionelle Blockcodes erfüllen die Bedingung der Invertierbarkeit nicht immer, da in der Regel weniger Prüfbit als Informationsbit vorhanden sind und somit eine bijektive Abbildung zwischen Informations- und Prüfbit ausgeschlossen ist.

Verfahren nach Lin/Yu
Bei der Realisierung nach Lin und Yu wird der Informationsvektor $\mathbf{u}$ zunächst mit einem fehlererkennenden Code C_1 codiert, es werden also Prüfbit angehängt. Das Codewort lautet ($\mathbf{c}_1 = \mathbf{u}, \mathbf{q}_1$). Zusätzlich erfolgt die Codierung mit einem zweiten, systematischen, fehlerkorrigierenden und invertierbaren Code C_2 (s. Bild 3.18), dessen Prüfteil $\mathbf{p}_2$ allerdings nicht gesendet, sondern im Sender gepuffert wird. Zuerst wird $\mathbf{c}_1$ gesendet. Tritt während der Übertragung ein Fehler auf, so wird der empfangene Block $(\mathbf{u}, \mathbf{q}_1)$ im Empfänger gespeichert und ein NAK-Signal gesendet. Dies veranlaßt den Sender, die Prüfbit $\mathbf{p}_2$ des Codes C_2 ebenfalls mit dem fehlererkennenden Code C_1 zu codieren $\mathbf{c}_2 = (\mathbf{p}_2, \mathbf{q}_2)$ und zu übertragen.

Es ist zu beachten, dass bei der ersten Übertragung nur die Informationsbit und einige wenige Prüfbit des fehlererkennenden Codes C_1 gesendet wurden, die Redundanz ist also sehr gering. Im Fehlerfall werden die

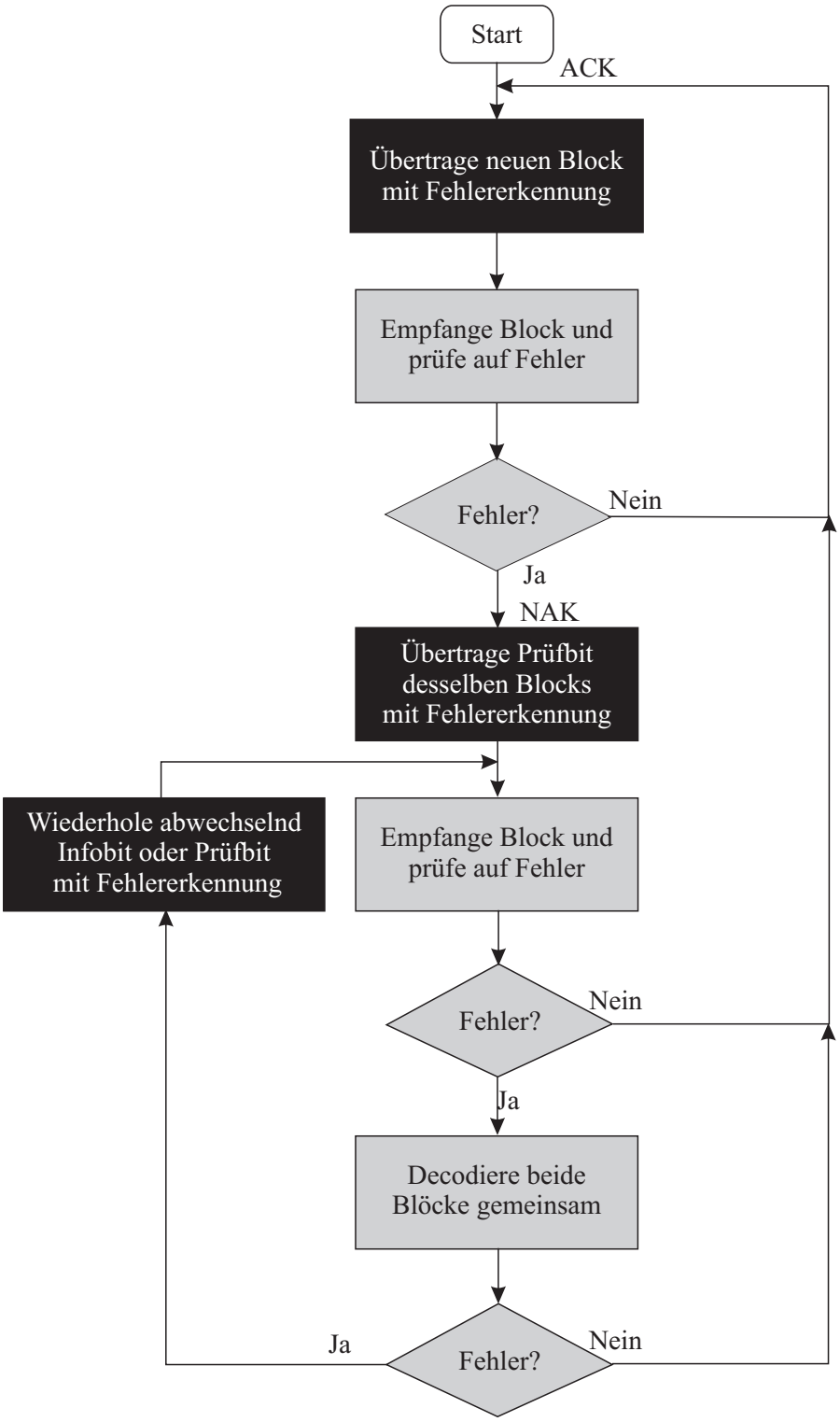


Bild 3.17: Protokollablauf beim Typ-II hybriden ARQ-Systems

Informationsbit dann nicht mehr übertragen, sondern die Prüfbit des fehlerkorrigierenden und invertierbaren Codes C_2 (zusammen mit Prüfbit $\mathbf{q}_2$).

Im Empfänger werden nun zunächst die empfangenen Prüfbit $\mathbf{p}_2$ anhand der Checksumme $\mathbf{q}_2$ auf Fehlerfreiheit überprüft. Ist dies der Fall, so können aufgrund der Invertierbarkeit von Code C_2 die Informationsbit $\mathbf{u}$ direkt aus $\mathbf{p}_2$ berechnet werden. Wird dagegen ein Fehler detektiert, so erfolgt die gemeinsame Decodierung von $(\mathbf{u}, \mathbf{p}_2)$ des Codes C_2. Ist diese auch erfolglos, werden die Informationsbit zusammen mit $\mathbf{q}_1$ erneut übertragen,

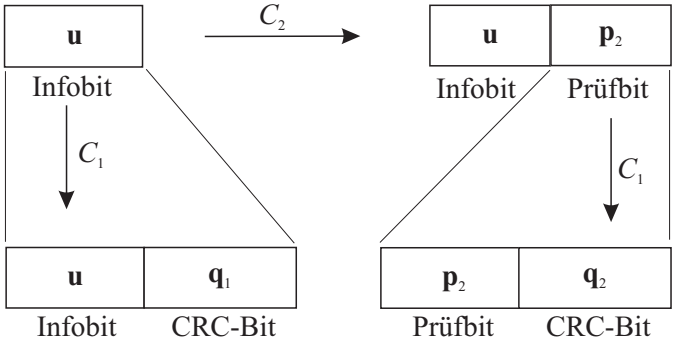


Bild 3.18: Codierung beim Typ-II hybriden ARQ-System

zunächst allein decodiert und im Fehlerfall erneut zusammen mit $\mathbf{p}_2$ decodiert usw.

Verfahren nach Lugand/Costello
Eine zweite, sehr ähnliche Variante des Typ-II hybriden ARQ-Verfahrens wurde von Lugand und Costello vorgestellt. Hier wird als fehlerkorrigierender Code C_2 ein halbratiger Faltungscode mit den Generatorpolynomen $g_0(D)$ und $g_1(D)$ eingesetzt. Da die Schieberegisterstruktur eines nicht-rekursiven Faltungscodes als FIR-Filter interpretiert werden kann, ist der Code durch ein IIR-Filter einfach invertierbar. Der CRC-Code besitze das Generatorpolynom $g(D)$.

Die Vorgehensweise ist nun die gleiche wie beim Lin/Yu-Verfahren. Zuerst wird das Codewort $c_0(D) = u(D) \cdot g_0(D) \cdot g(D)$ berechnet und übertragen. Dies entspricht einer Hälfte des halbratigen Faltungscodes, es ist also noch keine Redundanz durch C_2 zugefügt worden. Die Informationsbit sind durch die Faltung mit $g_0(D)$ lediglich miteinander kombiniert worden. Tritt kein Fehler auf, so kann aus der Sequenz $u(D) \cdot g_0(D)$ mit Hilfe des IIR-Filters die gesuchte Informationsfolge $u(D)$ geschätzt werden. Im Fehlerfall wird dagegen im Sender auch das zweite Generatorpolynom des Faltungscodes genutzt. Wir bilden $c_1(D) = u(D) \cdot g_1(D) \cdot g(D)$ und übertragen es. Auch $c_1(D)$ kann einzeln decodiert werden.

Ist auch dieser Versuch gescheitert, werden $(c_0(D), c_1(D))$ gemeinsam mit dem Viterbi Algorithmus decodiert. Schlägt auch diese Decodierung fehl, ist die Sequenz $c_0(D)$ zu wiederholen, zunächst einzeln zu decodieren, dann gegebenenfalls gemeinsam mit $c_1(D)$, usw.

3.6 Typ-III hybrides System

Das Typ-II hybride ARQ-System hat gegenüber dem Typ-I hybriden Ansatz den Vorteil, dass jede Teilübertragung für sich einzeln decodierbar ist und somit nicht unter den *Altlasten* vorangegangener Versuche leidet. Zudem besitzt es weiterhin den Vorteil, mehrere, durch unterschiedliche Übertragungsbedingungen gestörte Empfangssignale miteinander zu kombinieren und somit Diversität auszunutzen.

Diese Idee greift nun das letzte in diesem Kapitel vorgestellte Verfahren noch einmal auf, nämlich das **Typ-III hybride Verfahren** [Kal95]. Es verwendet die schon erwähnten ratenkompatiblen punktierten Faltungscodes und nutzt gleichzeitig die Vorteile der Typ-II hybriden Strategien. Dies wird durch die sogenannte komplementäre Punktierung erreicht.

Komplementäre Punktierung bedeutet, dass ein niederratiger Faltungscode genommen wird, der mit verschiedenen Punktierungsmatrizen $\mathbf{P}_i$ punktiert wird, wobei sich die $\mathbf{P}_i$ derart ergänzen, dass jedes Bit mindestens einmal übertragen wird. Jede Punktierungsmatrix einzeln gesehen sorgt dafür, dass nur wenig Redundanz je Übertragung gesendet wird, die zugehörige Sequenz ist einzeln decodierbar.

Nachgesendete Versionen anderer $\mathbf{P}_i$ können im Fehlerfall ohne Probleme mit ihren Vorgängern kombiniert

und mit einem einzigen Viterbi Algorithmus decodiert werden. Dabei kann es durchaus vorkommen, dass einzelne Symbole mehrfach übertragen wurden. Lediglich das komplette Ausblenden bestimmter Bit durch alle Punktierungsmatrizen ist zu vermeiden.

Komplementäre Punktierungsmatrizen können durch zyklisches Verschieben der Spalten von $\mathbf{P}_i$ gebildet werden.

Beispiel

$$\mathbf{P}_1 = \begin{pmatrix} 1\,1\,0\,1 & 1\,1\,0\,1 \\ 1\,0\,1\,1 & 1\,0\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 0\,1\,1\,0 & 0\,1\,1\,0 \end{pmatrix}$$

$$\mathbf{P}_2 = \begin{pmatrix} 0\,1\,1\,1 & 0\,1\,1\,1 \\ 1\,1\,1\,0 & 1\,1\,1\,0 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,0\,0\,1 & 1\,0\,0\,1 \end{pmatrix}$$

$$\mathbf{P}_1 + \mathbf{P}_2 = \begin{pmatrix} 1\,2\,1\,2 & 1\,2\,1\,2 \\ 2\,1\,2\,1 & 2\,1\,2\,1 \\ 2\,2\,2\,2 & 2\,2\,2\,2 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \end{pmatrix}$$

Es ist zu erkennen, dass die Punktierungsmatrizen $\mathbf{P}_i$ die primitive Periode $L_P = 4$ besitzen. Sie haben jedoch 8 Spalten, damit sie durch zyklisches Verschieben der Spalten um $L_P/2 = 2$ Positionen ineinander überführt werden können.

Für dieses einfache Beispiel würden wir ein System erhalten, dass bei jeder Übertragung einen drittelratigen Faltungscode verwendet, insgesamt wird bei 2 Übertragungen eine Gesamtcoderate von $R_c = 1/6$ erzielt, was höher ist als zweimal den viertelratigen Faltungscode zu verwenden.

Literaturverzeichnis

[BDMS91] E. Biglieri, D. Divsalar, P. McLane, and M. Simon. *Introduction to Trellis-Coded Modulation with Applications*. Macmillan Publishing Company, New York, 1991.

[Bos98] M. Bossert. *Kanalcodierung*. Teubner, Stuttgart, 1998.

[For66] G. D. Forney. *Concatenated Codes*. The M.I.T. Press, Cambridge, Massachusetts, 1966.

[Fri96] Bernd Friedrichs. *Kanalcodierung*. Springer Verlag, Berlin, 1996.

[Hag88] J. Hagenauer. Rate-Coompatible Punctured Convolutional Codes (RCPC-Codes and their Applications. *IEEE Transactions on Communications*, 36(4):389–400, April 1988.

[Hag96] J. Hagenauer. Source-controlled channel decoding. *IEEE Tran. on Communications*, 43(9):2449–2457, September 1996.

[Kal95] S. Kallel. Complementary-Punctured Convolutional (CPC) Codes and their Applications. *IEEE Transactions on Communications*, 43(6):2005–2009, June 1995.

[Kam04] K. D. Kammeyer. *Nachrichtenübertragung*. Teubner, Stuttgart, 2004.

[Off96] E. Offer. *Decodierung mit Qualitätsinformation bei verketteten Codiersystemen*. PhD thesis, Deutsche Forschungsanstalt für Luft- und Raumfahrt (DLR), VDI Fortschrittsberichte, Juni 1996.

[Ung82] G. Ungerböck. Channel Coding with Multilevel Phase Signals. *IEEE Transactions on Information Theory*, IT-28:56–67, January 1982.

[Ung87] G. Ungerböck. Trellis-Coded Modulation with Redundant Signal Sets. Part I: Introduction; Part II: State of the Art. *IEEE Communications Magazine*, 25(2):5–21, February 1987.

[VWZP89] A.J. Viterbi, J.K. Wolf, E. Zehavi, and R. Padovani. A Pragmatic Approach to Trellis-Coded Modulation. *IEEE Communications Magazine*, 27(6):11–19, July 1989.