Negazyklische Codes und Lee-Metrik

Studienarbeit

von

Günther Haas

Abteilung Informationstechnik
Universität Ulm
April 1996

S/1995/JM/2

Abteilung Informationstechnik   Universität Ulm

STUDIENARBEIT

Negazyklische Codes und Lee-Metrik

Erläuterungen:

Algebraische Decoder berechnen aus dem Hard Decision Empfangswort zunächst ein Syndrom, das ausschließlich vom Fehlerwort, nicht aber vom Codewort abhängt. Anschließend wird das zu einem gegebenen Syndrom wahrscheinlichste Fehlerwort berechnet. Als Maß für die Wahrscheinlichkeit dient dem Decoder die verwendete Metrik. Herkömmliche algebraische Decoder benutzen die Hamming - Metrik. Die Verwendung der Hamming - Metrik ist zwar für BPSK- oder orthogonale Modulationsverfahren sinnvoll, nicht aber für Modulationsverfahren wie höherwertige PSK oder QAM - Modulation. Die unterschiedliche Intersymbolfehlerwahrscheinlichkeit dieser Modulationsschemata wird durch die Verwendung der Lee - Metrik im Fall der PSK - Übertragung berücksichtigt. Für die Codierung von Symbolen einer PSK - Modulation unter Verwendung der Lee - Metrik eignen sich vor allem die von Berlekamp eingeführten negazyklischen Codes.

In der Studienarbeit sollen zunächst die Eigenschaften der negazyklischen Codes und ihre Decodierfähigkeit, untersucht werden. Zur Konstruktion beliebiger negazyklischer Codes sind entsprechende Funktionen in der Programmiersprache C zu implementieren. Danach soll ein algebraischer Decodieralgorithmus für negazyklische Codes in COSSAP implementiert werden. Mit Hilfe dieses Moduls wird dann die Decodierung sämtlicher negazyklischer Codes simuliert. Schließlich sollen die Simulationsergebnisse mit der Korrekturkapazität von Reed Solomon Codes unter Verwendung der Hamming - Metrik verglichen werden.

Abgabetermin: 10. April 1996

Bearbeiter: Günther Haas

Betreuer: Dipl.-Ing. Johannes Maucher, Zimmer 43.1.225

Prüfer: Prof. Dr.-Ing M. Bossert

Katalognr.: S/1995/JM/2

In Galois fields, full of flowers
primitive elements dance for hours
climbing sequentially throuh the trees
and shouting occasional parities.

The syndromes like ghosts in the misty damp
feed the smoldering fires of the Berlekamp
and high-flying exponents sometimes are downed
on the jagged peaks of the Gilbert bound.

S. B. Weinstein
Bell Telephone Labs., Inc.
Holmdel, N.J. 07733

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im Zusammenhang mit Datenübertragung über Kanäle aller Art wird heutzutage Kanalcodierung eingesetzt, um aufgetretene Fehler zu erkennen und zu korrigieren. Die zu übertragende Information wird im Sender codiert und muß auf der Empfängerseite wieder decodiert werden.
Eine Möglichkeit hierzu stellt die sogenannte algebraische Decodierung dar. Algebraische Decoder berechnen aus einem Hard Decision Empfangswort ein Syndrom, welches ausschließlich vom aufgetretenen Fehler, nicht jedoch vom gesendeten Codewort abhängt. Die nötigen Berechnungen werden dabei in einem endlichen Zahlenkörper durchgeführt. Aus dem Syndrom wird anschließend das wahrscheinlichste Fehlerwort berechnet und vom Empfangswort abgezogen. Sind weniger Fehler aufgetreten als korrigiert werden können, so erhält man das fehlerfreie, gesendete Codewort, dem eindeutig eine bestimmte Information zugeordnet werden kann.
Bei der Berechnung des Syndroms spielt die verwendete Metrik als Maß für die Wahrscheinlichkeit eines Fehlers eine entscheidende Rolle. Bei vielen Codes wie z.B. RS- oder BCH-Codes wird die sogenannte Hamming-Metrik verwendet. Diese ist gut geeignet für BPSK oder orthogonale Modulationsverfahren. In Verbindung mit mehrwertigen Modulationsverfahren wie MPSK oder QAM ist ihr Einsatz jedoch weniger sinnvoll, da hier ein gesendetes Symbol in manche Symbole mit größerer Wahrscheinlichkeit verfälscht wird als in andere. Für solche Modulationsverfahren existieren geeignetere Metriken und Codes, die die unterschiedlichen Intersymbolfehlerwahrscheinlichkeiten berücksichtigen. So bieten sich für MPSK-Modulation die sogenannte Lee-Metrik und die zugehörigen negazyklischen Codes an. Für QAM-Modulation existieren analog dazu die Mannheim-Metrik und die i-zyklischen Codes.

Im Rahmen der vorliegenden Arbeit wurden die negazyklischen Codes untersucht. Die Theorie hierzu ist in [Ber84] beschrieben, ebenso eine erste Methode für ihre algebraische Decodierung.

Im Vordergrund stand die Implementierung der beschriebenen Verfahren, um einen anschließenden Vergleich mit anderen nichtbinären Codes wie den RS-Codes zu ermöglichen, die auf der Hamming-Metrik basieren.
Dazu war zunächst die Erstellung einer C-Funktionenbibliothek für Berechnungen im Erweiterungskörper GF(p^m) und Rechnungen mit Polynomen erforderlich. Ins-

besondere die Polynom-Funktionen wurden dabei speziell für den Einsatz bei negazyklischen Codes ausgelegt. Die in [Gai93] und [Kre94] beschriebenen Funktionenbibliotheken dienten hierbei als Vorlage, konnten jedoch aufgrund ihrer ungünstig gewählten Datenstrukturen bzw. ihrer Beschränkung auf GF(2^m) nicht unverändert zum Einsatz kommen.
Mit Hilfe der C-Funktionen wurden dann ein Encoder und ein algebraischer Decoder für negazyklische Codes für das Simulationswerkzeug COSSAP implementiert.

Die vorliegende Arbeit ist wie folgt gegliedert:
In Kapitel 2 sind zunächst die wichtigsten Grundlagen zur Kanalcodierung und zu den verwendeten Metriken aufgeführt. Insbesondere wird die Konstruktion und die algebraische Decodierung von RS- und BCH-Codes beschrieben. Analog dazu wird bei der Beschreibung der negazyklischen Codes in Kapitel 3 vorgegangen. In Kapitel 4 werden die durchgeführten Berechnungen und Simulationen beschrieben und in Kapitel 5 findet man die zugehörigen Ergebnisse sowie einen Vergleich zwischen negazyklischen Codes und RS-Codes. In Kapitel 6 werden werden abschließend die Mindestdistanzen negazyklischer Codes betrachtet.
Im Anhang finden sich schließlich einige Tabellen mit Parametern negazyklischer Codes, Flußdiagramme wichtiger Algorithmen sowie eine kurze Beschreibung der programmierten COSSAP-Module.
Einen vollständigen Abdruck aller C- und MATLAB-Programme, die im Rahmen dieser Arbeit erstellt wurden, findet man in [Haa96].

Kapitel 2

Theoretische Grundlagen

In diesem Abschnitt sollen zunächst die wichtigsten Voraussetzungen zum Verständnis der folgenden Kapitel geschaffen werden. Zu diesem Zweck wird in Abschnitt 2.1 die Theorie der Erweiterungskörper sowie der Kreisteilungsklassen behandelt, die die Basis für die Konstruktion der Codes und deren algebraische Decodierung darstellen. Anschließend werden kurz Mattson-Solomon Polynome und die Newtonschen Gleichungen dargestellt, die zur Decodierung nötig sind. Da eine ausführlichere Einführung in die Zahlentheorie den Rahmen dieser Arbeit sprengen würde, sei hierfür auf Standardwerke der Kanalcodierung wie [Bla90], [Bos92] oder [MS83] sowie auf spezielle Literatur zur Zahlentheorie [HW58], [Has69] verwiesen. In Abschnitt 2.2 folgt die Beschreibung der verwendeten Metriken, insbesondere wird die Lee-Metrik für MPSK-Modulation eingeführt, auf der negazyklische Codes aufbauen. In Abschnitt 2.3 werden schließlich Blockcodes eingeführt. Insbesondere werden die zyklischen Reed-Solomon- und BCH-Codes behandelt, da die Beschreibung negazyklischer Codes in Kapitel 3 analog erfolgt. Die algebraische Decodierung der beschriebenen zyklischen Blockcodes basiert hauptsächlich auf dem Berlekamp-Algorithmus zur Lösung der sogenannten Schlüsselgleichung. Dieser ist in Anhang B.1 in Form eines Ablaufdiagramms dargestellt und spielt auch bei der algebraischen Decodierung von negazyklischen Codes (NC-Codes) in Kapitel 3 eine wesentliche Rolle. Auch hierzu finden sich weitergehende Beschreibungen in oben genannter Literatur und in [Ber84].

2.1 Mathematische Grundlagen

2.1.1 Galois-Felder

Algebraische Decodierverfahren sind dadurch gekennzeichnet, daß sie mit Rechenoperationen in endlichen Zahlenkörpern arbeiten. Diese endlichen Zahlenkörper werden Galois-Felder genannt und mit $GF(q)$ abgekürzt. Im folgenden werden Primkörper $GF(p)$ und Erweiterungskörper $GF(p^m)$ als wichtigste Vertreter von Galoisfeldern dargestellt.

Primkörper, kleinste Reste und absolut kleinste Reste

Als *Primkörper* bezeichnet man die Menge $GF(p) = \{0, 1, 2, , \ldots , p - 1\}$, wobei p eine Primzahl ist. Dann, und nur dann, erfüllen die Elemente der Menge bezüglich der Addition und der Multiplikation mod p die Axiome eines Körpers (siehe [Bos92], Seite 36f). Die Modulo-Rechnung ist dabei folgendermaßen definiert:

$$b = a \bmod p = a - \left\lfloor \frac{a}{p} \right\rfloor p \qquad p, a \in I\!N, \quad b \in GF(p), \tag{2.1}$$

wobei $\lfloor x \rfloor$ die größte ganze Zahl darstellt, die kleiner oder gleich x ist. Die Operation $a \bmod p$ ist eine eindeutige Abbildung und teilt die Menge der natürlichen Zahlen $I\!N$ in p Restklassen auf, d.h. in p Klassen von natürlichen Zahlen, die bezüglich der Division durch p den gleichen Rest besitzen. Man nennt $GF(p)$ daher auch Menge der *kleinsten Reste* mod p. Diese Menge stellt ein volles Restsystem mod p dar.
Ein weiteres volles Restsystem besteht aus den p aufeinanderfolgenden ganzen Zahlen x, für die gilt $|x| \leq \lfloor p/2 \rfloor$. Man nennt die Elemente der Menge $\{0, \pm 1, \pm 2, \ldots , \pm \lfloor p/2 \rfloor\}$ *absolut kleinste Reste* mod p (siehe auch [Has69], Seite 25). Ist p eine Primzahl, so besitzt auch die Menge der absolut kleinsten Reste die Eigenschaften eines Körpers. Für diesen Fall wird folgende Schreibweise vereinbart:

$$GF^A(p) = \left\{ 0, \pm 1, \pm 2, \ldots , \pm \left\lfloor \frac{a}{p} \right\rfloor \right\} \tag{2.2}$$

Die natürlichen Zahlen werden durch die Abbildung

$$b = a \bmod {}^A p = a - \left[\frac{a}{p} \right] p \qquad p, a \in I\!N, \quad b \in GF^A(p) \tag{2.3}$$

eindeutig auf $GF^A(p)$ abgebildet. $[x]$ stellt dabei die Rundung von x auf die nächste ganze Zahl dar. Zwischen $GF(p)$ und $GF^A(p)$ besteht ein Isomorphismus (siehe [Has69], Seite 23ff), d.h. eine eindeutig umkehrbare Zuordnung. Jeweils eine Restklasse $\bmod p$ und eine Restklasse $\bmod {}^A p$ enthalten die gleichen Elemente, jeweils ein kleinster Rest und ein absolut kleinster Rest sind einander eindeutig zugeordnet. Folgende Tabelle stellt die eindeutig umkehrbare Abbildung dar:

$GF(p)$	$GF^A(p)$
0	0
1	1
$\vdots$	$\vdots$
$\lfloor p/2 \rfloor$	$\lfloor p/2 \rfloor$
$\lfloor p/2 \rfloor + 1$	$- \lfloor p/2 \rfloor$
$\lfloor p/2 \rfloor + 2$	$- \lfloor p/2 \rfloor - 1$
$\vdots$	$\vdots$
p	$- 1$

Beispiel:

$GF(7)$	$GF^A(7)$
0	0
1	1
2	2
3	3
4	-3
5	-2
6	-1

Prinzipiell macht es also keinen Unterschied, ob Rechnungen in $GF(p)$ oder $GF^A(p)$ durchgeführt werden. Diese Tatsache wird in Abschnitt 2.2.3 bei der Definition der Lee-Metrik von Bedeutung sein.

Ordnung, primitives Element

Die *Ordnung* r eines Elements $a \in GF(p), \quad a \neq 0$ ist definiert als der kleinste positive ganzzahlige Exponent, für den gilt: $a^r \equiv 1 \bmod p$, das heißt $a^{i<r} \neq 1 \bmod p$. Allgemein gilt $r \leq p-1$ und für den Fall $r = p-1$ nennt man a *primitives Element*. Zu jedem Galois-Feld existiert mindestens ein primitives Element.
Für $r < p-1$ gilt immer $r \mid p-1$, d.h. die Ordnung eines Elements ist immer Teiler der maximal möglichen Ordnung (siehe [Bos92], Seite 38).
Für Erweiterungskörper $GF(p^m)$, die später eingeführt werden, kann die Definition der Ordnung sinngemäß übernommen werden, doch muß $p-1$ durch p^m-1 ersetzt werden. D.h. die Ordnung eines primitiven Elements aus einem Erweiterungskörper ist $p^m - 1$.

Irreduzible und primitive Polynome

Ein Polynom $p(x) = p_m x^m + \cdots + p_1 x + p_0$ mit Koeffizienten $p_i \in GF(p)$, welches nicht als Produkt von Polynomen kleineren Grades darstellbar ist, die auch nur Koeffizienten aus $GF(p)$ besitzen, nennt man *irreduzibles Polynom*. Notwendig für ein irreduzibles Polynom ist, daß es keine Nullstellen aus $GF(p)$ besitzt. Hinreichend ist diese Bedingung nicht, da auch ein Produkt von irreduziblen Polynomen diese Bedingung erfüllt, jedoch selbst kein irreduzibles Polynom darstellt.
Ein irreduzibles Polynom $p(x)$ besitzt laut Definition keine Nullstellen aus $GF(p)$. Man kann jedoch eine Nullstelle (oder Wurzel) α außerhalb von $GF(p)$ definieren, für die gilt: $p(\alpha) = 0$. Besitzt solch eine Wurzel α die Eigenschaft, daß

$$\alpha^i \bmod p(\alpha), \qquad i = 0, 1, \ldots, p^m - 2, \qquad m = grad\,(p(x)) \tag{2.4}$$

alle $p^m - 1$ verschiedenen Polynome mit Koeffizienten aus $GF(p)$ erzeugt, so nennt man $p(x)$ *primitives Polynom* und α *primitive Einheitswurzel*, da sie die maximale Ordnung $n = p^m - 1$ besitzt.

Die verwendete Modulo-Polynom-Rechnung $q(x) \bmod p(\alpha)$ bedeutet, daß man $p(\alpha)$ zu Null setzt und diese Gleichung auf $q(x)$ anwendet. Ein Beispiel soll dies verdeutlichen:

Beispiel 2.1: Modulo-Polynom-Rechnung

Seien die Polynom-Koeffizienten aus $GF(5)$ und $q(x) = 2x^2 + 3x + 1$, $p(x) = x^2 + x + 2$. Aus $p(x) = 0$ ergibt sich dann $x^2 = -x - 2 = 4x + 3$ und man erhält $q(x) = 2(4x+3) + 3x + 1 = x + 2 \bmod p(x)$ □

Erweiterungskörper

Sei $p(x)$ ein primitives Polynom vom Grad m. Nach Definition ist dann seine primitive Wurzel α nicht Element des Primkörpers $GF(p)$. Sie stellt ein Element des sogenannten *Erweiterungskörpers* $GF(p^m)$ dar. Dessen Elemente sind die Potenzen $\alpha^{-\infty} = 0, \alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p^m-2}$ (vergleiche (2.4)).
Der Erweiterungskörper besteht demnach aus p^m Elementen. Für diese bieten sich

aufgrund der Konstruktion des Erweiterungskörpers zwei verschiedene Darstellungsformen an. Zum einen ist jedes Element eindeutig durch seinen Exponenten $exp \in \{-\infty, 0, 1, \ldots, p^m-2\}$ gekennzeichnet. Andererseits stellt jedes Element ein Polynom in α dar, wobei es genügt, die Koeffizienten aus $GF(p)$ dieses Polynoms anzugeben. Man unterscheidet deshalb die sogenannte *Exponentendarstellung* und die *Komponentendarstellung*.

Da die Addition zweier Elemente aus $GF(p^m)$ in Komponentendarstellung durch Addition der Polynomkoeffizienten mod p erfolgt, während sich die Multiplikation am einfachsten durch Addition der Exponenten mod p^m-1 realisieren läßt, ist für nahezu alle Berechnungen im Erweiterungskörper eine Zuordnungstabelle zwischen beiden Darstellungformen (eine sogenannte *Logarithmentafel*) erforderlich. Solch eine Tabelle veranschaulicht auch sehr gut die Konstruktion eines Erweiterungskörpers.

Beispiel 2.2: Logarithmentafel des Erweiterungskörpers $GF(5^2)$ mit dem primitiven Polynom $p(x) = x^2 + x + 2,\ p_i \in GF(5)$ und dem primitiven Element α mit $p(\alpha) = 0$.

Exponent	Komponenten	Polynom	Berechnung (Koeffizienten mod 5)							
$-\infty$	0 0	0								
0	0 1	1								
1	1 0	α								
2	4 3	$4\alpha+3$	$=$	α^2	$=$	$-\alpha-2$				
3	4 2	$4\alpha+2$	$=$	α^3	$=$	$\alpha\cdot\alpha^2$	$=$	$4\alpha^2+3\alpha$	$=$	$19\alpha+12$
4	3 2	$3\alpha+2$	$=$	α^4	$=$	$\alpha\cdot\alpha^3$	$=$	$4\alpha^2+4\alpha$	$=$	$18\alpha+12$
5	4 4	$4\alpha+4$	$=$	α^5	$=$	$\alpha\cdot\alpha^4$	$=$	$3\alpha^2+2\alpha$	$=$	$14\alpha+9$
6	0 2	2	$=$	α^6	$=$	$\alpha\cdot\alpha^5$	$=$	$4\alpha^2+4\alpha$	$=$	$20\alpha+12$
7	2 0	2α	$=$	α^7	$=$	$\alpha\cdot\alpha^6$	$=$	2α		
8	3 1	$3\alpha+1$	$=$	α^8	$=$	$\alpha\cdot\alpha^7$	$=$	$2\alpha^2$	$=$	$8\alpha+6$
9	3 4	$3\alpha+4$	$=$	α^9	$=$	$\alpha\cdot\alpha^8$	$=$	$3\alpha^2+1\alpha$	$=$	$13\alpha+9$
10	1 4	$\alpha+4$	$=$	α^{10}	$=$	$\alpha\cdot\alpha^9$	$=$	$3\alpha^2+4\alpha$	$=$	$16\alpha+9$
11	3 3	$3\alpha+3$	$=$	α^{11}	$=$	$\alpha\cdot\alpha^{10}$	$=$	$\alpha^2+4\alpha$	$=$	$8\alpha+3$
12	0 4	4	$=$	α^{12}	$=$	$\alpha\cdot\alpha^{11}$	$=$	$3\alpha^2+3\alpha$	$=$	$15\alpha+9$
13	4 0	4α	$=$	α^{13}	$=$	$\alpha\cdot\alpha^{12}$	$=$	4α		
14	1 2	$\alpha+2$	$=$	α^{14}	$=$	$\alpha\cdot\alpha^{13}$	$=$	$4\alpha^2$	$=$	$16\alpha+12$
15	1 3	$\alpha+3$	$=$	α^{15}	$=$	$\alpha\cdot\alpha^{14}$	$=$	$\alpha^2+2\alpha$	$=$	$6\alpha+3$
16	2 3	$2\alpha+3$	$=$	α^{16}	$=$	$\alpha\cdot\alpha^{15}$	$=$	$\alpha^2+3\alpha$	$=$	$7\alpha+3$
17	1 1	$\alpha+1$	$=$	α^{17}	$=$	$\alpha\cdot\alpha^{16}$	$=$	$2\alpha^2+3\alpha$	$=$	$11\alpha+6$
18	0 3	3	$=$	α^{18}	$=$	$\alpha\cdot\alpha^{17}$	$=$	$\alpha^2+\alpha$	$=$	$5\alpha+3$
19	3 0	3α	$=$	α^{19}	$=$	$\alpha\cdot\alpha^{18}$	$=$	3α		
20	2 4	$2\alpha+4$	$=$	α^{20}	$=$	$\alpha\cdot\alpha^{19}$	$=$	$3\alpha^2$	$=$	$12\alpha+9$
21	2 1	$2\alpha+1$	$=$	α^{21}	$=$	$\alpha\cdot\alpha^{20}$	$=$	$2\alpha^2+4\alpha$	$=$	$12\alpha+6$
22	4 1	$4\alpha+1$	$=$	α^{22}	$=$	$\alpha\cdot\alpha^{21}$	$=$	$2\alpha^2+1\alpha$	$=$	$9\alpha+6$
23	2 2	$2\alpha+2$	$=$	α^{23}	$=$	$\alpha\cdot\alpha^{22}$	$=$	$4\alpha^2+\alpha$	$=$	$17\alpha+12$
24	0 1	1	$=$	α^{24}	$=$	$\alpha\cdot\alpha^{23}$	$=$	$2\alpha^2+2\alpha$	$=$	$10\alpha+6$

$$
\begin{array}{lll}
\alpha^{i \bmod 24}; \quad \text{Multiplikation} & : & \alpha^{20}\cdot\alpha^{16} = \alpha^{36 \bmod 24} = \alpha^{12} \\
\qquad\qquad \text{Addition} & : & \alpha^{20}+\alpha^{16} = (2\ 4)+(2\ 3) \bmod (5\ 5) = (4\ 2) = \alpha^3
\end{array}
$$

□

2.1.2 Kreisteilungsklassen

Sei p eine Primzahl, dann definiert man als *Kreisteilungsklasse* K_i *bezüglich einer Zahl* $n = p^m - 1$ die Menge

$$K_i := \left\{ i \cdot p^j \bmod n \mid j = 0, 1, \ldots, m-1 \right\}. \tag{2.5}$$

Kreisteilungsklassen besitzen folgende Eigenschaften:

$$\begin{aligned} |K_i| &\leq m \\ K_i \cap K_j &= \emptyset, \qquad i \neq j \\ K_0 &= \{0\} \\ \bigcup K_i &= \{0, 1, \ldots, n-1\} \end{aligned} \tag{2.6}$$

Ist $n = p^m - 1$ eine Primzahl so gilt:

$$|K_i| = m, \qquad \forall i \neq 0 \tag{2.7}$$

und die Anzahl der Kreisteilungsklassen beträgt dann:

$$\#(K_i) = \frac{n-1}{m} + 1 \tag{2.8}$$

Beispiel 2.3: Kreisteilungsklassen bezüglich der Zahl $n = 15 = 2^4 - 1$ $(p = 2, m = 4)$

$$\begin{aligned} K_0 &= \{0\} \\ K_1 &= \{1, 2, 4, 8\} \\ K_3 &= \{3, 6, 9, 12\} \\ K_5 &= \{5, 10\} \\ K_7 &= \{7, 11, 13, 14\} \end{aligned}$$

□

Konjugiert komplexe Elemente

Sei $p(x)$ ein irreduzibles Polynom vom Grad m mit Koeffizienten $p_i \in GF(p)$. Ist dann α eine Wurzel von $p(x)$, so sind auch

$$\alpha^p, \alpha^{p^2}, \ldots \alpha^{p^{m-1}} \tag{2.9}$$

Wurzeln von $p(x)$. Man nennt diese insgesamt m Wurzeln *konjugiert komplex.* Mit Hilfe der oben definierten Kreisteilungsklassen läßt sich die Menge M_1 der konjugiert komplexen Wurzeln um α in einfacher Form angeben:

$$M_1 = \bigcup_{u \in K_1} \alpha^u = \left\{ \alpha, \alpha^p, \alpha^{p^2}, \ldots, \alpha^{p^{m-1}} \right\} \tag{2.10}$$

Allgemeiner: Ist α primitives Element von $GF(p^m)$ und K_i eine Kreisteilungsklasse bezüglich der Zahl $n = p^m - 1$, so enthält die Menge

$$M_i = \bigcup_{u \in K_i} \alpha^u = \left\{ \alpha^i, \ldots \right\} \tag{2.11}$$

alle zu α^i *konjugiert komplexen Elemente.*

Minimale Polynome

Sei K_i eine Kreisteilungsklasse bezüglich $n = p^m - 1$, α primitives Element von $GF(p^m)$ und M_i wie oben definiert. Dann nennt man Polynome der Form

$$m_i(x) = \prod_{x_0 \in M_i} (x - x_0) = \prod_{u \in K_i} (x - \alpha^u), \tag{2.12}$$

die als Nullstellen genau alle konjugiert komplexen Elemente aus M_i besitzen, *minimale Polynome*. Minimale Polynome sind irreduzibel über $GF(p)$ und besitzen nur Koeffizienten aus $GF(p)$. Für ihren Grad gilt wegen Gleichung 2.6:

$$grad\big(m_i(x)\big) \leq m \tag{2.13}$$

Das Produkt aller minimalen Polynome eines Erweiterungskörpers, d.h. das Polynom welches jedes Element α^i $(i > 0)$ als Nullstelle besitzt, ergibt sich zu

$$\prod_{u=0}^{n-1} (x - \alpha^u) \;=\; (x^n - 1), \qquad \text{mit } n = p^m - 1 \tag{2.14}$$

und hat damit auch nur Koeffizienten aus $GF(p)$. Für ungerade Primzahlen p (also $p \neq 2$) ist obiges Polynom das Produkt der beiden folgenden Polynome :

$$\prod_{u \text{ gerade}} (x - \alpha^u) \;=\; \big(x^{\frac{n}{2}} - 1\big) \tag{2.15}$$

$$\prod_{u \text{ ungerade}} (x - \alpha^u) \;=\; \big(x^{\frac{n}{2}} + 1\big) \tag{2.16}$$

(Ein Beweis hierfür findet sich in Kapitel 3.1, Gleichung 3.12.) Da für $p \neq 2$ alle Kreisteilungsklassen entweder nur gerade oder ungerade Werte enthalten, lassen sich beide Produkte auch als Produkt minimaler Polynome schreiben. Dies spielt eine große Rolle bei negazyklischen Codes.

2.1.3 Mattson-Solomon Polynome, Diskrete Fourier-Transformation

Sei α ein Element aus $GF(p^m)$ mit der Ordnung n und seien $b(x), \beta(x)$ Polynome vom Grad $r \leq n - 1$ mit Koeffizienten aus $GF(p^m)$ so bezeichnet man folgenden Zusammenhang als *Diskrete Fourier-Transformation* (DFT):

$$b(x) \;\circ\!\!-\!\!\bullet\; \beta(x)$$

$$\beta_i \;=\; b(\alpha^i) \tag{2.17}$$

$$b_i \;=\; \frac{1}{n}\beta(\alpha^{-i}) \tag{2.18}$$

$$i \;=\; 0, 1, \ldots, n-1$$

Dabei stellt 2.18 die *Rücktransformation*[1] dar. $\beta(x)$ bezeichnet man auch als *Mattson-Solomon Polynom von* $b(x)$.
Der Zusammenhang zwischen den Koeffizienten der beiden Polynome läßt sich auch sehr schön durch folgende Matrixgleichung beschreiben:

$$\underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{\mathbf{n-1}} \end{pmatrix}}_{\boldsymbol{\beta}} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2)^1 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ \multicolumn{5}{c}{\dotfill} \\ 1 & (\alpha^{n-1})^1 & (\alpha^{n-1})^2 & \cdots & (\alpha^{n-1})^{n-1} \end{pmatrix}}_{\mathbf{M}_{(n)}} \underbrace{\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{pmatrix}}_{\mathbf{b}} \tag{2.19}$$

Im folgenden soll für die $(n \times n)$ Mattson-Solomon Transformations-Matrix die Bezeichnung $M_{(n)}$ verwendet werden. Mattson-Solomon Polynome, d.h. transformierte Polynome werden in der vorliegenden Arbeit wie in obigen Gleichungen mit griechischen Buchstaben bezeichnet, zum Beispiel $\beta(x)$. Die zugehörigen Vektoren werden wie alle Vektoren fettgedruckt, zum Beispiel $\boldsymbol{\beta}$.
Koeffizienten $\beta_i = 0$ entsprechen Nullstellen α^i in $b(x)$.

Rechenregeln

Für den Umgang mit transformierten Polynomen sind einige Rechenregeln erforderlich, die hier ohne Beweis oder Herleitung aufgeführt werden sollen.
Seien $b(x)$ und $d(x)$ Polynome vom Grad $r \leq n-1$ mit den Koeffizienten b_i bzw. d_i und

$$\begin{aligned} b(x) &\circ\!\!-\!\!\bullet\ \beta(x) \\ d(x) &\circ\!\!-\!\!\bullet\ \delta(x). \end{aligned}$$

Dann gilt der sogenannte *Faltungssatz*:

$$b_i \cdot d_i \ \circ\!\!-\!\!\bullet\ \beta(x) \cdot \delta(x) \bmod (x^n - 1) \tag{2.20}$$

$$b(x) \cdot d(x) \bmod (x^n - 1) \ \circ\!\!-\!\!\bullet\ \beta_i \cdot \delta_i \tag{2.21}$$

und folgende Additionsregeln:

$$b(x) + d(x) \ \circ\!\!-\!\!\bullet\ \beta(x) + \delta(x) \tag{2.22}$$

$$\text{bzw.} \qquad b_i + d_i \ \circ\!\!-\!\!\bullet\ \beta_i + \delta_i$$

2.1.4 Newtonsche Gleichungen

In diesem Abschnitt sollen die sogenannten *Newtonschen Gleichungen* vorgestellt werden, die in Kapitel 3.2 bei der Decodierung negazyklischer Codes verwendet werden. Dazu werden zunächst einige grundlegende Begriffe erklärt. Eine weitergehende Beschreibung der Problematik findet man in [Koc74].

[1]In der Literatur werden unterschiedliche Definitionen verwendet. Die angegebene eignet sich im Bezug auf [Ber84] am besten.

Symmetrische Polynome, Potenzsummen

Ein Polynom $p(X_1, X_2, \dots, X_t)$ in den Unbekannten $X_1, X_2, \dots, X_t$ heißt *symmetrisch*, wenn es bei jeder Permutation der Unbekannten unverändert bleibt.
Ein Beispiel für symmetrische Polynome sind die *Potenzsummen*

$$s_i = X_1^i + X_2^i + \cdots + X_t^i \qquad (i = 0, 1, 2, \dots) \tag{2.23}$$

Elementarsymmetrische Polynome, Vietascher Wurzelsatz

Die wichtigsten symmetrischen Polynome sind die sogenannten *elementarsymmetrischen Polynome*

$$\begin{aligned} a_1 &= X_1 + X_2 + \cdots + X_t \\ a_2 &= X_1X_2 + X_1X_3 + \cdots + X_{t-1}X_t \\ &\dots\dots\dots\dots\dots\dots \\ a_t &= X_1X_2\cdots X_t \end{aligned} \tag{2.24}$$

da sich mit ihrer Hilfe alle symmetrischen Polynome darstellen lassen. Insbesondere gilt, daß das Polynom

$$a(x) = x^t - a_1x^{t-1} + a_2x^{t-2} - \cdots + (-1)^{t-1}a_{t-1}x + (-1)^t a_t \tag{2.25}$$

die Nullstellen $X_1, X_2, \dots, X_t$ besitzt, da gilt:

$$a(x) = (x - X_1)(x - X_2)\cdots(x - X_t) \tag{2.26}$$

Dieser Zusammenhang wird auch als *Vietascher Wurzelsatz* bezeichnet (siehe [BSG91], Seite 135).

Newtonsche Gleichungen

Für den Spezialfall der Potenzsummen s_k wird der Zusammenhang zwischen den elementarsymmetrischen Polynomen a_i und den symmetrischen Polynomen durch die sogenannten *Newtonschen Gleichungen* folgendermaßen beschrieben:

$$\begin{aligned} s_l - a_1s_{l-1} + a_2s_{l-2} - \cdots + (-1)^t a_t s_{l-t} &= 0 \quad \text{für} \quad l > t \\ s_k - a_1s_{k-1} + a_2s_{k-2} - \cdots + (-1)^{k-1}a_{k-1}s_1 + (-1)^k k a_k &= 0 \quad \text{für} \quad k \le t \end{aligned}$$

Definiert man

$$\sigma_i = (-1)^i a_i \qquad (i = 1, 2, \dots, t) \tag{2.27}$$

so erhält man eine einfachere Schreibweise für die Newtonschen Gleichungen

$$s_l + \sigma_1 s_{l-1} + \cdots + \sigma_t s_{l-t} = 0 \qquad \text{für} \quad l > t \tag{2.28}$$

$$s_k + \sigma_1 s_{k-1} + \cdots + \sigma_{t-1}s_1 + \sigma_k k = 0 \qquad \text{für} \quad k \le t \tag{2.29}$$

Setzt man für k die Werte $1, 2, 3, \ldots$ ein, so erhält man das Gleichungssystem

$$\begin{array}{cl} l > t & s_l + s_{l-1}\sigma_1 + \ldots\ldots\ldots\ldots + s_{l-t}\sigma_t = 0 \\ \vdots & \ldots\ldots\ldots\ldots\ldots\ldots \\ l = t+1 & s_{t+1} + s_t\sigma_1 + \ldots\ldots\ldots\ldots + s_1\sigma_t = 0 \\ k = t & s_t + s_{t-1}\sigma_1 + \ldots\ldots + s_1\sigma_{t-1} + t\sigma_t = 0 \\ k = t-1 & s_{t-1} + s_{t-2}\sigma_1 + \cdots + s_1\sigma_{t-2} + (t-1)\sigma_{t-1} = 0 \\ \vdots & \ldots\ldots\ldots\ldots\ldots\ldots \\ k = 2 & s_2 + s_1\sigma_1 + 2\sigma_2 = 0 \\ k = 1 & s_1 + 1\sigma_1 = 0 \end{array} \tag{2.30}$$

Betrachtet man die σ_i als Koeffizienten eines Polynoms in der in dieser Arbeit üblichen Schreibweise

$$\sigma(x) = 1 + \sigma_1 x + \cdots + \sigma_{t-1}x^{t-1} + \sigma_t x^t, \tag{2.31}$$

so entspricht dies

$$\begin{aligned} \sigma(x) &= x^t \cdot a\left(x^{-1}\right) \\ &= x^t \cdot \prod_{i=1}^{t}(x^{-1} - X_i) \\ &= \prod_{i=1}^{t}(1 - X_i x) \end{aligned}$$

und $\sigma(x)$ besitzt die Nullstellen

$$X_1^{-1}, X_2^{-1}, \ldots, X_t^{-1}. \tag{2.32}$$

Die Newtonschen Gleichungen geben also den Zusammenhang zwischen den Koeffizienten σ_i eines Polynoms $\sigma(x)$ und den Potenzsummen s_k der Kehrwerte X_i seiner Wurzeln an. Dies wird in Kapitel 3.2 bei der algebraischen Decodierung negazyklischer Codes von Bedeutung sein.

2.2 Metriken

Bei der algebraischen Decodierung geht man prinzipiell immer gleich vor. Zuerst wird aus einem empfangenen fehlerbehafteten Codewort ein Syndrom berechnet, das nur vom aufgetretenen Fehler abhängt. Das eigentliche Problem liegt dann in der Aufgabe, aus diesem Syndrom einen Fehler bzw. ein Fehlerpolynom mit möglichst wenig Koeffizienten $\neq 0$ bzw. mit möglichst geringem Gewicht zu ermitteln, das dieses Syndrom erzeugt. Dabei wird berücksichtigt, daß Fehler mit geringem Gewicht wahrscheinlicher sind als solche mit großem Gewicht. Um das Gewicht eines Fehlervektors und damit die Distanz zweier Vektoren angeben zu können, ist die Definition einer Metrik notwendig.

Eine Metrik bzw. ein Abstand $d(x, y)$ zwischen zwei Punkten (Vektoren) wird durch folgende Eigenschaften definiert:

$$\begin{array}{lll} 1. & d(x,y) \geq 0, \quad d(x,y) = 0 \Leftrightarrow x = y & \text{positive Definitheit} \\ 2. & d(x,y) = d(y,x) & \text{Symmetrie} \\ 3. & d(x,y) \leq d(x,z) + d(z,y) & \text{Dreiecksungleichung} \end{array} \tag{2.33}$$

Im folgenden werden verschiedene bekannte Metriken kurz aufgeführt und die Lee-Metrik wird vorgestellt. Für Vektoren der Länge n wird folgende Schreibweise definiert: $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$. In Polynomschreibweise steht hierfür $v(x) = v_0 + v_1 x + \ldots v_{n-1} x^{n-1}$.

2.2.1 Euklidische Metrik

Die Euklidische Metrik ist für den eindimensionalen Raum folgendermaßen definiert:

$$d_E(x, y) = \sqrt{(x-y)^2} \tag{2.34}$$

Für den zwei- und n-dimensionalen Fall gilt :

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2} \tag{2.35}$$

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{(x_0 - y_0)^2 + \cdots + (x_{n-1} - y_{n-1})^2} \tag{2.36}$$

Prinzipiell ist die Euklidische Metrik die genaueste Metrik um den Abstand zweier Punkte im Vektorraum zu bestimmen. Da sie jedoch für den Körper der reellen Zahlen definiert ist, ist sie für algebraische Decodierverfahren, die mit endlichen Körpern arbeiten, ungeeignet. Trotzdem sei hier der Vollständigkeit halber auch ihre Norm $||\mathbf{x}||$ aufgeführt:

$$w_E(x) = ||\mathbf{x}|| = d(\mathbf{x}, 0) \tag{2.37}$$

2.2.2 Hamming-Metrik

Die Hamming-Metrik ist besonders geeignet für den binären Fall, d.h. bei Verwendung von Elementen aus $GF(2)$ bzw. Vektoren aus $\mathbb{F}_2^n$ mit Komponenten aus $GF(2)$. Aus diesem Grund bietet sich die Verwendung der Hamming-Metrik besonders in Kombination mit BPSK- oder orthogonalen Modulationsverfahren an. Wendet man die Hamming-Metrik im nichtbinären Fall an, also z.B. in Verbindung mit PSK- oder QAM-Modulation, so können Codes aufgrund dieser Metrik nur zwischen „Fehler“ und „Nichtfehler“ unterscheiden. Es kann nicht berücksichtigt werden, daß weniger starke Verfälschungen der Sendesymbole wahrscheinlicher sind als starke. Definiert ist die Hamming-Metrik wie folgt:

$$d_H(x, y) = \begin{cases} 0 & , \quad x = y \\ 1 & , \quad x \neq y \end{cases} \tag{2.38}$$

Die Hamming-Distanz zweier Vektoren $\mathbf{x}, \mathbf{y}$ ist die Anzahl der unterschiedlichen Stellen von $\mathbf{x}$ und $\mathbf{y}$. Im binären Fall $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ ergibt die Addition der Vektoren $\bmod\ 2$ in den unterschiedlichen Stellen eins. Diese Einsen müssen nur noch aufsummiert werden, um die Hamming-Distanz zweier binärer Vektoren zu bestimmen. Dies entspricht der Berechnung des Hamming-Gewichts der Summe $(\mathbf{x} + \mathbf{y})$.

$$d_H(\mathbf{x}, \mathbf{y}) = \sum_{j=0}^{n-1} d_H(x_j, y_j) \qquad \text{allgemein} \tag{2.39}$$

$$\begin{aligned} d_H(\mathbf{x}, \mathbf{y}) &= \sum_{j=0}^{n-1} (x_j + y_j) \bmod 2 \qquad \text{binär} \\ &= w_H(\mathbf{x} + \mathbf{y}) \end{aligned} \tag{2.40}$$

Das Hamming Gewicht eines Vektors $\mathbf{x}$ gibt also die Anzahl der von Null verschiedenen Stellen von $\mathbf{x}$ an. Im binären Fall ist die Schreibweise wieder vereinfacht:

$$\begin{aligned} w_H(\mathbf{x}) &= \sum_{j=0}^{n-1} d_H(x_j, 0) \qquad \text{allgemein} \\ &= d_H(\mathbf{x}, \mathbf{0}) \end{aligned} \tag{2.41}$$

$$w_H(\mathbf{x}) = \sum_{j=0}^{n-1} x_j \qquad \text{binär,} \quad x_j \in GF(2) \tag{2.42}$$

2.2.3 Lee-Metrik

Die Lee-Metrik ist weniger gebräuchlich als die beiden zuvor behandelten Metriken und soll daher etwas ausführlicher betrachtet werden. Sie eignet sich aufgrund ihrer Eigenschaften besonders gut für nichtbinäre Codes in Verbindung mit MPSK-Modulation. Insbesondere bauen die negazyklischen Codes darauf auf, die in Kapitel 3 beschrieben werden.

Sei $x, y \in GF(p)$ oder $x, y \in GF^A(p)$, so ist die Lee-Metrik wie folgt definiert:

$$d_L(x, y) = \left|(x - y) \bmod {}^A p\right| \tag{2.43}$$

Anschaulich bedeutet dies die Berechnung des Betrags des absolut kleinsten Restes von $(x - y)$. Dabei spielt es keine Rolle, ob x, y Elemente aus $GF(p)$ oder aus einem dazu isomorphen Körper wie z.B. $GF^A(p)$ darstellen, da einander zugeordnete Elemente als Argument der $\bmod^A$-Operation gleiche Ergebnisse liefern (siehe Abschnitt 2.1.1). Beispiel 2.4 auf Seite 15 soll dies verdeutlichen.
Auch die absolute Zuordnung der Elemente des verwendeten Körpers zu den MPSK-Sendesymbolen ist beliebig. Es muß allerdings gelten, daß benachbarte Elemente des Körpers auch benachbarte Sendesymbole kennzeichnen. Sonst geht die wichtige Eigenschaft verloren, daß benachbarte Symbole die geringste Distanz besitzen.

Es soll nun gezeigt werden, daß die obige Definition alle Anforderungen aus Gleichung 2.33 an eine Metrik erfüllt.

1. Positive Definitheit:

$$
\begin{array}{lll}
d_L(x,y) = \left|(x-y) \bmod {}^A p\right| \geq 0 & \text{weil} & |z| \geq 0 \quad \forall\, z \in GF^A(p) \\
d_L(x,y) = 0 \Leftrightarrow x = y & \text{weil} & 0 \bmod {}^A p = 0 \\
 & \text{und} & (x-y) \bmod {}^A p \neq 0 \text{ für } x \neq y \\
 & \text{wenn} & x, y \in GF(p).
\end{array}
$$

2. Symmetrie:

$$
\begin{array}{lll}
d_L(x,y) = d_L(y,x) & \text{weil} & (y-x) = -(x-y) \\
 & \text{und} & (-z) \bmod {}^A p = -(z \bmod {}^A p) \\
 & \text{und} & |-z| = |z|
\end{array}
$$

3. Dreiecksungleichung: $d_L(x,y) \leq d_L(x,z) + d(z,y)$

   Zu folgender Überlegung betrachte man Abbildung 2.1.
   Die Lee-Distanz $d_L(x,y)$ zwischen den Sendesymbolen x und y entspricht der Anzahl a von Kreisbogen mit Winkel $\beta = 2\pi/M$, die man zurücklegen muß, um auf kürzestem Wege auf dem Einheitskreis von einem Symbol zum anderen zu gelangen. Daher kann bei MPSK-Modulation, bei der den Sendesymbolen die Elemente des $GF(M)$ zugeordnet werden, $d(x,y)$ maximal den Wert $\lfloor M/2 \rfloor$ annehmen.
   Die rechte Seite der Dreiecksungleichung kann als zusammengesetzter Weg von x nach y betrachtet werden. Die Anzahl der auf diesem Weg nötigen Kreisbogen kann entweder gleich der Anzahl a von Kreisbogen auf dem kürzesten Weg sein, oder sie ist größer (z.B. gleich $b+c$) falls man in die „falsche Richtung" läuft. Der zusammengesetzte Weg kann also nicht kürzer sein als der direkte Weg. Es gilt demnach $a \leq b + c$. Die Dreiecksungleichung wird also von der Lee-Metrik erfüllt.

Für n-dimensionale Vektoren mit Komponenten aus $GF(p)$ (oder $GF^A(p)$) läßt sich nun analog zur Hamming-Distanz die *Lee-Distanz* definieren:

$$
d_L(\mathbf{x}, \mathbf{y}) = \sum_{j=0}^{n-1} d_L(x_j, y_j) \tag{2.44}
$$

Desweiteren kann man wie bei der Hamming-Metrik auch das Gewicht eine Vektors berechnen, indem man seine Distanz zum Nullvektor gleicher Länge berechnet. Das *Lee-Gewicht* berechnet sich wie folgt:

$$
w_L(\mathbf{x}) = \sum_{j=0}^{n-1} d_L(x_j, 0) \qquad \text{allgemein} \tag{2.45}
$$

$$
= \sum_{j=0}^{n-1} |x_j| \qquad \text{für } x_j \in GF^A(p) \tag{2.46}
$$

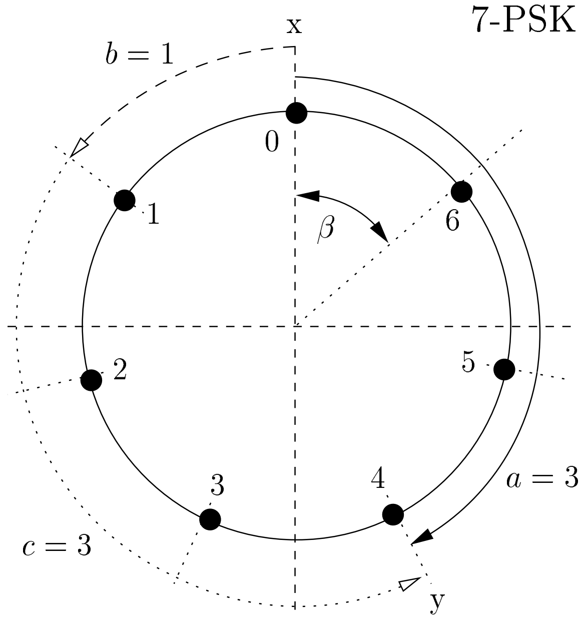

Abbildung 2.1: 7-PSK-Modulation

Beispiel 2.4: Berechnung von Lee-Distanzen zwischen Elementen aus $GF(7)$ und $GF^A(7)$

Anhand von Bild 2.2 sollen die oben definierten Begriffe Lee-Metrik, -Distanz und -Gewicht noch einmal veranschaulicht werden.

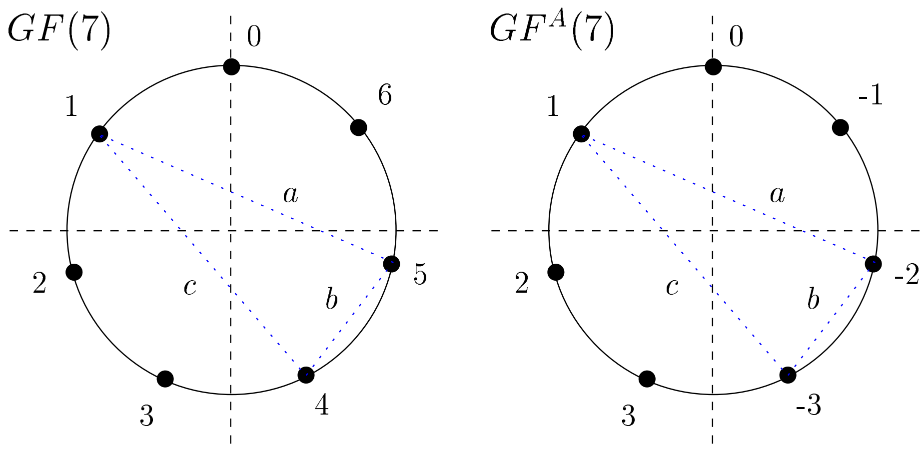

Abbildung 2.2: zu Beispiel 2.4:
Codierung der 7-PSK-Sendesymbole durch Elemente aus $GF(7)$ oder aus $GF^A(7)$

Zunächst werden die eingezeichneten Distanzen a, b, c bzw. a_A, b_A, c_A berechnet:

$$\begin{array}{lllllllllll}
a & = & d(1,5) & = & |(1-5) \bmod {}^A 7| & = & |(-4) \bmod {}^A 7| & = & |3| & = & 3 \\
a_A & = & d(1,-2) & = & |(1-(-2)) \bmod {}^A 7| & = & |(3) \bmod {}^A 7| & = & |3| & = & 3 \\
b & = & d(5,4) & = & |(5-4) \bmod {}^A 7| & = & |(1) \bmod {}^A 7| & = & |1| & = & 1 \\
b_A & = & d(-2,-3) & = & |(-2-(-3)) \bmod {}^A 7| & = & |(1) \bmod {}^A 7| & = & |1| & = & 1 \\
c & = & d(4,1) & = & |(4-1) \bmod {}^A 7| & = & |(3) \bmod {}^A 7| & = & |3| & = & 3 \\
c_A & = & d(-3,1) & = & |(-3-1) \bmod {}^A 7| & = & |(-4) \bmod {}^A 7| & = & |3| & = & 3
\end{array}$$

Wie in Abschnitt 2.1.1 beschrieben, ergeben sich die gleichen Distanzen, unabhängig davon, ob zur Codierung der Sendesymbole die Menge der kleinsten Reste $GF(7)$ oder die Menge der absolut kleinsten Reste $GF^A(7)$ verwendet wird. Es gilt $a_A = a, b_A = b$ und $c_A = c$.

Im Beispiel kann man auch gut die Gültigkeit der Dreiecksungleichung erkennen, es gilt:

$$\begin{aligned} a &< b + c \\ 3 &< 1 + 3 \end{aligned}$$

Im folgenden seien die Sendesymbole wie links in Abbildung 2.2 mit Elementen aus $GF(7)$ codiert. Die Lee-Distanz und die Lee-Gewichte der Vektoren $\mathbf{x} = (1, 3, 4, 0, 3)$ und $\mathbf{y} = (5, 3, 2, 6, 0)$ errechnen dann sich zu:

$$\begin{aligned} d_L(\mathbf{x}, \mathbf{y}) &= \sum_{j=0}^{4} d_L(x_j, y_j) = 3 + 0 + 2 + 1 + 3 = 9 \\ w_L(\mathbf{x}) &= \sum_{j=0}^{4} d_L(x_j, 0) = 1 + 3 + 3 + 0 + 3 = 10 \\ w_L(\mathbf{y}) &= \sum_{j=0}^{4} d_L(y_j, 0) = 2 + 3 + 2 + 1 + 0 = 8 \end{aligned}$$

□

2.3 Blockcodes

In Abbildung 2.3 ist die prinzipielle Struktur eines Systems dargestellt, welches unter Einsatz von Kanalcodierung Information über einen Kanal überträgt. Geht man von

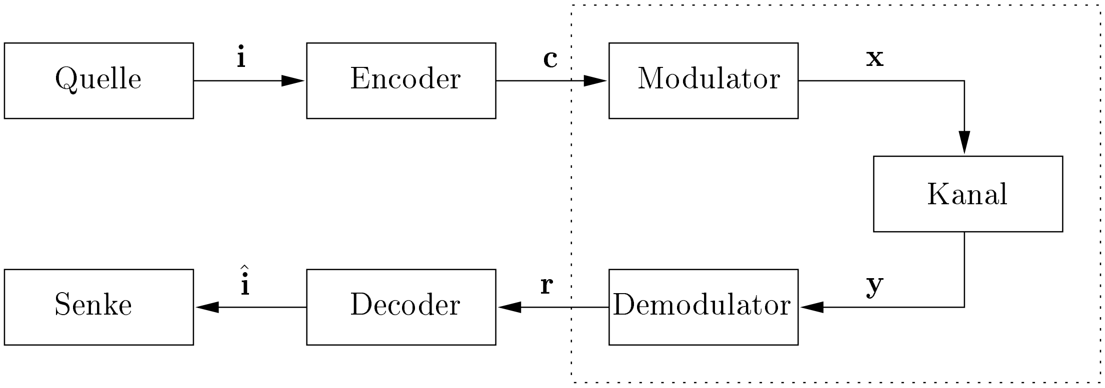

Abbildung 2.3: Übertragungsmodell

dieser Betrachtung auf der Signalebene zu einer Betrachtung auf der Symbolebene über, so kann man den gleichen Zusammenhang auch durch das Übertragungsmodell nach Bild 2.4 beschreiben. Die gesamte Information über den Kanal ist hier in der Fehlerquelle enthalten.
Für das Problem der Kanalcodierung ist eine Betrachtung auf dieser Ebene zunächst ausreichend, deshalb soll im folgenden von dem Modell nach Abbildung 2.4 ausgegangen werden.

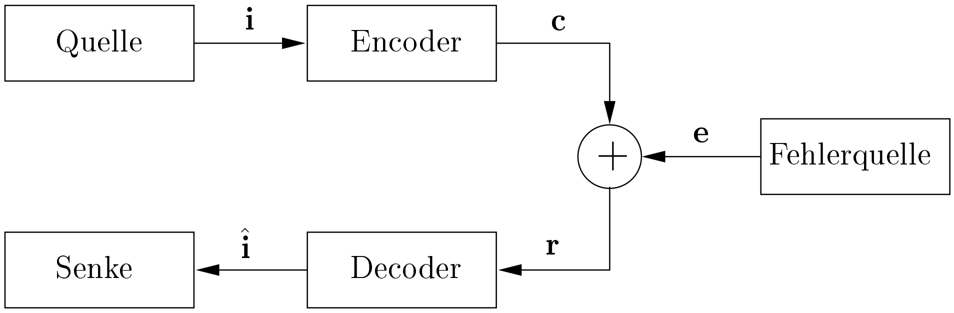

Abbildung 2.4: Übertragungsmodell auf Symbolebene

2.3.1 Grundbegriffe

Ein *Blockcode* $\mathcal{C}$ ordnet jedem Informationsvektor $\mathbf{i} = (i_0, i_1, \ldots, i_{k-1})$ der *Dimension* k umkehrbar eindeutig ein Codewort $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ der Länge $n \geq k$ zu. Bei den Komponenten der beiden Vektoren handelt es sich dabei um Elemente des Symbolalphabets $GF(q)$. Neben der Zuordnung bezeichnet man auch die Menge der q^k Codewörter als Blockcode $\mathcal{C}$.

Für den Fall, daß der Code $\mathcal{C}$ bezüglich der Vektoraddition zweier Codewörter und der Skalarmultiplikation eines Codeworts mit einem Element aus $GF(q)$ eine abgeschlossene Menge ist, so handelt es sich um einen Unterraum des $GF(q)^n$ und man spricht von einem *linearen Blockcode*. Anschaulich bedeutet dies, daß die Summe zweier Codewörter und somit auch das Vielfache eines Codedeworts wieder ein gültiges Codewort darstellt. Notwendige Bedingung ist daher, daß jeder lineare Blockcode den Nullvektor $\mathbf{0}$ der Länge n enthält.

Die *Coderate* $R = \frac{k}{n}$ gibt das Verhältnis von Informationssymbolen zu Codesymbolen wieder und $n - k$ gibt die Anzahl der *Redundanz*-Zeichen an.

Matrixschreibweise: Generatorpolynom, Prüfpolynom

Jeder Code benötigt eine Zuordnungsvorschrift, die zu einem Informationvektor $\mathbf{i}$ das zugehörige Codewort $\mathbf{c}$ angibt. Für lineare Blockcodes läßt sich diese Vorschrift einfach über die Multiplikation der Informationsvektors mit der sogenannten *Generatormatrix* $\mathbf{G}$ darstellen:

$$\mathbf{c} = \mathbf{i} \cdot \mathbf{G} \tag{2.47}$$

$\mathbf{G}$ ist eine $(k \times n)$-Matrix mit Elementen aus $GF(q)$ und besitzt den maximalen Rang k.
Ein Codewort stellt eine Linearkombination der k Zeilen der Generatormatrix dar. Jede Zeile ist also ein gültiges Codewort und die Matrixzeilen sind eine Basis des Unterraumes $\mathcal{C}$ von $GF(q)^n$.

Außer durch seine Generatormatrix $\mathbf{G}$ kann man einen linearen Blockcode auch durch seine *Prüfmatrix* $\mathbf{H}$ eindeutig beschreiben. Für ein gültiges Codewort des Codes $\mathcal{C}$, muß gelten:

$$\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0} \quad \text{bzw.} \quad \mathbf{H} \cdot \mathbf{c}^T = \mathbf{0} \tag{2.48}$$

D.h. das Produkt jeder beliebigen Zeile aus $\mathbf{H}$ und dem transponierten Codewort $\mathbf{c}^T$ muß den Wert Null ergeben. Die Zeilenvektoren von $\mathbf{H}$ sind also orthogonal zu allen Codewörtern aus $\mathcal{C}$.

Die Spaltenzahl der Prüfmatrix muß gleich n sein, damit die Operation $\mathbf{H}\cdot\mathbf{c}^T$ definiert ist.
Die Zeilenzahl ergibt sich aus einer anderen Überlegung: Um zu prüfen, ob $(n-k)$ Redundanzzeichen den korrekten Wert besitzen, der durch k Informationszeichen bestimmt wird, sind genau $(n-k)$ linear unabhängige Gleichungen nötig. Deutlicher wird dies später im Zusammenhang mit der Definition systematischer Codes. An dieser Stelle soll nur festgehalten sein, daß daraus hervorgeht, daß $\mathbf{H}$ eine $((n-k)\times n)$-Matrix vom Rang $(n-k)$ sein muß.

Die Zeilen von $\mathbf{H}$ bilden auf diese Weise die Basis für einen Unterraum von $GF(q)^n$ mit der Dimension $(n-k)$. Man nennt ihn den *dualen Code* $\mathcal{C}^\perp$ von $\mathcal{C}$. Die Generatormatrix des dualen Codes $\mathcal{C}^\perp$ ist gleich der Prüfmatrix von $\mathcal{C}$ und umgekehrt.

Aus Gleichung 2.47 und 2.48 ergibt sich der Zusammenhang zwischen Generator- und Prüfmatrix wie folgt:

$$\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0} \tag{2.49}$$

Systematische Codierung

Stehen bei einem Code an k Stellen der Codewörter unverändert die zugehörigen k Informationssymbole, so spricht man von einem *systematischen Code*. Dies ist äquivalent zu der Aussage, daß die Generatormatrix eines systematischen Codes alle k verschiedenen Einheitsvektoren der Länge k als Spalten enthält. An der Position, an der in $\mathbf{G}$ der j. Einheitsvektor steht, steht in den Codewörtern das j. Informationssymbol. Für den Fall, daß der Informationsvektor $\mathbf{i}$ wie in Abbildung 2.3.1 als ganzes unverändert (an Anfang) im Codewort $\mathbf{c}$ erscheint,

$$\mathbf{c} = (i_0, i_1, \ldots, i_{k-1}, c_k, c_{k+1}, \ldots, c_{n-1}), \tag{2.50}$$

liegt die Generatormatrix in der folgenden *systematischen Form* vor:

$$\mathbf{G} = (\mathbf{I}_k | \mathbf{A}) \tag{2.51}$$

Dabei ist $\mathbf{I}_k$ die $(k\times k)$-Einheitsmatrix und $\mathbf{A}$ eine $(k\times(n-k))$-Matrix.
Bei dieser Form der Generatormatrix läßt sich leicht die zugehörige Prüfmatrix berechnen. Man erhält

$$\mathbf{H} = \left(-\mathbf{A}^T | \mathbf{I}_{n-k}\right). \tag{2.52}$$

Für den Fall, daß die Generator- oder Prüfmatrix von $\mathcal{C}$ nicht in systematischer Form vorliegen, so läßt sich durch elementare Zeilenoperationen und Vertauschung von

Spalten ein zu $\mathcal{C}$ *äquivalenter Code* mit gleichen Parametern und systematischer Darstellung erzeugen.
Wie bereits oben angedeutet, wird in dieser systematischen Form der Prüfmatrix gut deutlich, weshalb sie $(n-k)$ Zeilen besitzen muß. Jedes der $(n-k)$ Redundanzsymbole hat aufgrund der Einheitsmatrix in $\mathbf{H}$ nur auf je eine Zeile von $\mathbf{H} \cdot \mathbf{c}_T$ Einfluß, in dieser einen jedoch als einziges Redundanzsymbol. Jede Zeile stellt dadurch eine Prüfgleichung für ein einzelnes Redundanzsymbol dar. Aus diesem Grund sind in $\mathbf{H}$ $(n-k)$ linear unabhängige Zeilen notwendig, um zu prüfen, ob ein korrektes Codewort vorliegt.

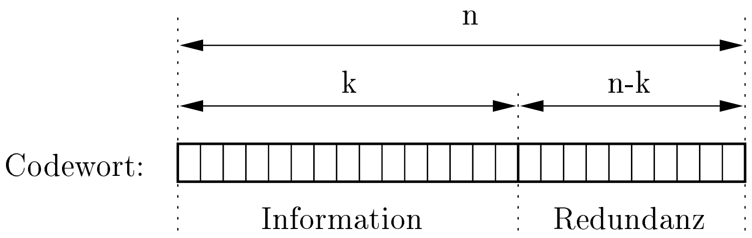

Abbildung 2.5: Codewort eines Codes mit systematischer Generatormatrix

Syndrom

Mit Hilfe der Prüfmatrix $\mathbf{H}$ kann man bei fehlerbehafteter Übertragung nach Abbildung 2.4 auf der Empfängerseite aus dem Empfangsvektor $\mathbf{r} = \mathbf{c} + \mathbf{f}$ einen Ausdruck $\mathbf{s}$ berechnen, der ausschließlich vom aufgetretenen Fehler $\mathbf{e}$ und nicht vom gesendeten Codewort abhängt.

$$\begin{aligned} \mathbf{s}^T &= \mathbf{H} \cdot \mathbf{r}^T = \mathbf{H} \cdot (\mathbf{c}^T + \mathbf{e}^T) = \mathbf{0} + \mathbf{H} \cdot \mathbf{e}^T = \mathbf{H} \cdot \mathbf{e}^T \\ \text{bzw.} \quad \mathbf{s} &= \mathbf{r} \cdot \mathbf{H}^T = \mathbf{f} \cdot \mathbf{H}^T \end{aligned} \tag{2.53}$$

Man nennt $\mathbf{s}$ das *Syndrom*. Für $\mathbf{e} = \mathbf{0}$ oder $\mathbf{e} \in \mathcal{C}$ erhält man $\mathbf{s} = \mathbf{0}$.
Liegt die Prüfmatrix in systematischer Form nach Gleichung 2.52 vor, so läßt sich die Bedeutung eines Syndroms $s \neq 0$ anschaulich erklären. In Abbildung 2.6 sind die Stellen eines Codeworts markiert, welche Einfluß auf die j-te von $(n-k)$ Prüfgleichungen haben. Die Syndromkomponente $s_j \neq 0, \quad 0 \leq j < n-k$ gibt an, daß an einer oder mehreren dieser Stellen Fehler aufgetreten sind.

Zyklische Codes, Polynomschreibweise

Einen linearen Code $\mathcal{C}$ mit der Eigenschaft, daß die *zyklische Verschiebung*

$$\circlearrowright_1 (\mathbf{c}) = (c_{n-1}, c_0, c_1, \dots, c_{n-2}) \tag{2.54}$$

eines Codeworts $\mathbf{c} = (c_0, c_1, \dots, c_{n-2}, c_{n-1})$ wiederum ein gültiges Codewort aus $\mathcal{C}$ darstellt, nennt man *zyklischen Code*. Der Index des Verschiebungszeichens $\circlearrowright_1$ soll

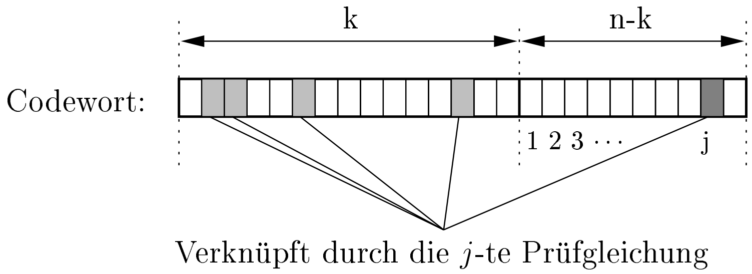

Abbildung 2.6: Stellen eines Codewortes, die Einfluß auf eine Prüfgleichung haben

dabei andeuten, daß es sich um eine Verschiebung um eine Stelle handelt.
Zyklische Codes können außer in der Matrixschreibweise auch in *Polynomschreibweise* dargestellt werden. Statt als Vektoren können Codewörter und Informationswörter als Polynome dargestellt werden. Die beiden Darstellungsformen sind äquivalent:

$$\begin{aligned} \mathbf{i} = (i_0, i_1, \ldots, i_{k-1}) &\iff i(x) = i_0 + i_1 x + \cdots + i_{k-1} x^{k-1} \\ \mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) &\iff c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} \\ i_j, c_j &\in GF(q) \end{aligned} \tag{2.55}$$

Da die Polynomkoeffizienten aus $GF(q)$ stammen, können sie auch den Wert Null annehmen, d.h. das Informationspolynom und das Codepolynom können auch einen kleineren Grad als $k-1$ bzw. $n-1$ besitzen.
In Polynomschreibweise läßt sich die zyklische Verschiebung eines Codeworts um eine Stelle als Multiplikation mit x darstellen. Damit dabei $c_{n-1}x^{n-1}$ nicht in $c_{n-1}x^n$ sondern in $c_{n-1}x^0 = c_{n-1}$ übergeht, ist es nötig $x^n \equiv 1$ zu setzen. Dies enspricht einer mod $(x^n - 1)$ - Rechnung. Man kann schreiben:

$$\begin{aligned} \circlearrowleft_1 \big(c(x)\big) &= x \cdot c(x) \mod (x^n - 1) \\ &= c_{n-1} + c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1} \end{aligned} \tag{2.56}$$

In dieser Darstellung wird die Generatormatrix $\mathbf{G}$ durch das *Generatorpolynom* $g(x)$ vom Grad $n-k$ ersetzt, und an die Stelle von Gleichung 2.47 tritt die Polynommultiplikation

$$c(x) = i(x) \cdot g(x). \tag{2.57}$$

Auch die Prüfmatrix wird durch ein Polynom, das sogenannte *Prüfpolynom* ersetzt. Analog zu Gleichung 2.49 gilt

$$\begin{aligned} g(x) \cdot h(x) &= x^n - 1 \\ &= 0 \mod (x^n - 1). \end{aligned} \tag{2.58}$$

Daraus ergibt sich bei gegebenem Generatorpolynom $g(x)$ sofort das Prüfpolynom

$$h(x) = \frac{x^n - 1}{g(x)} \tag{2.59}$$

Für ein gültiges Codewort $c(x)$ gilt dann analog zu Gleichung 2.48:

$$c(x) \cdot h(x) = 0 \quad \bmod (x^n - 1) \tag{2.60}$$

Durch eine einfache Überlegung läßt sich abschließend zeigen, daß der so konstruierte Code zyklisch ist. Sei $c(x)$ ein Codewort aus $\mathcal{C}$ und somit die Gleichungen 2.57 und 2.58 erfüllt, dann folgt

$$\begin{aligned}
\Longrightarrow \quad \circlearrowleft_1(\mathbf{c})h(x) \quad \bmod (x^n-1) &= \big(xc(x) \bmod (x^n-1)\big)h(x) \quad \bmod (x^n-1) \\
&= xc(x)h(x) \quad \bmod (x^n-1) \\
&= xi(x)g(x)h(x) \quad \bmod (x^n-1) \\
&= xi(x)(x^n-1) \quad \bmod (x^n-1) \\
&= 0
\end{aligned}$$

$\circlearrowleft_1(\mathbf{c})$ ist also auch Codewort aus $\mathcal{C}$, der Code ist zyklisch.

Auch für zyklische Blockcodes ist eine Darstellung in Matrixform möglich. Bei gegebenem $g(x)$ und $h(x)$ ergeben sich $\mathbf{G}$ und $\mathbf{H}$ zu

$$\mathbf{G} = \begin{pmatrix}
g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k} & & & & 0 \\
 & g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k} & & & \\
 & & g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k} & & \\
 & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
0 & & & & g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & g_{n-k}
\end{pmatrix} \tag{2.61}$$

$$\mathbf{H} = \begin{pmatrix}
h_k & h_{k-1} & h_{k-2} & \cdots & h_1 & h_0 & & & & 0 \\
 & h_k & h_{k-1} & h_{k-2} & \cdots & h_1 & h_0 & & & \\
 & & h_k & h_{k-1} & h_{k-2} & \cdots & h_1 & h_0 & & \\
 & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
0 & & & & h_k & h_{k-1} & h_{k-2} & \cdots & h_1 & h_0
\end{pmatrix}. \tag{2.62}$$

Nichtzyklische Blockcodes können dagegen nicht in Polynomschreibweise angegeben werden.

Codierungsmethoden

Für zyklische Codes bieten sich aufgrund der oben eingeführten Polynom- und Matrixschreibweise direkt zwei verschiedene Zuordnungen zwischen Informations- und Codewörtern an:

- Codierung durch Multiplikation mit dem Generatorpolynom nach (2.57):
  Diese Codierung $c(x) = i(x) \cdot g(x)$ ist nicht systematisch, doch ist die Rückgewinnung der Information aus einem Codewort relativ einfach:

  $$i(x) = \frac{c(x)}{g(x)} \tag{2.63}$$

  Auch wird sofort erkannt, wenn es sich bei $c(x)$ nicht um ein gültiges Codewort handelt. Die Polynomdivision liefert dann einen Rest ungleich Null.

- Systematische Codierung mit systematischer Generatormatrix nach (2.47):
Die Codierung $\mathbf{c} = \mathbf{i} \cdot \mathbf{G}$ hat den Vorteil, daß die Informationssymbole $i_0, \ldots, i_{k-1}$ direkt aus dem Codewort $c(x)$ ablesbar sind. Die Rückgewinnung der Information funktioniert daher noch einfacher als beim ersten Verfahren:

$$\mathbf{i} = (c_0, c_1, \ldots, c_{k-1}) \tag{2.64}$$

Da im folgenden, insbesondere bei der Behandlung der negazyklischen Codes in Kapitel 3, die Polynomschreibweise verwendet werden soll, sei hier noch eine Methode für systematische Codierung in dieser Schreibweise angegeben. Im Gegensatz zu (2.64) sollen die Informationssymbole jetzt dem „hinteren Teil" eines Codeworts entsprechen. Man legt folgende Zuordnung fest:

$$\begin{aligned} c_{n-1}x^{n-1} + \cdots + c_{n-k}x^{n-k} &= i_{k-1}x^{n-1} + \cdots + i_0 x^{n-k} \\ &= i(x) \cdot x^{n-k} \end{aligned} \tag{2.65}$$

Dann kann man die fehlenden $n-k$ Prüfstellen von $c(x)$ folgendermaßen berechnen:

$$\left(i(x) \cdot x^{n-k}\right)/g(x) = b(x) + rest(x)/g(x) \tag{2.66}$$

$$\begin{aligned} \Longrightarrow \qquad rest(x) &= i(x) \cdot x^{n-k} - b(x) \cdot g(x) \\ &= i(x) \cdot x^{n-k} \bmod g(x) \end{aligned} \tag{2.67}$$

$$\begin{aligned} \Longrightarrow \qquad c(x) &= i(x) \cdot x^{n-k} - rest(x) \\ &= i(x) \cdot x^{n-k} - \left(i(x) \cdot x^{n-k} \bmod g(x)\right) \end{aligned} \tag{2.68}$$

Die Informations-Rückgewinnung geschieht folgendermaßen:

$$i(x) = c_{n-1}x^{k-1} + c_{n-2}x^{k-2} + \cdots + c_{n-k+1}x + c_{n-k} \tag{2.69}$$

Um festzustellen, ob es sich bei einem Polynom um ein gültiges Codewort handelt, muß geprüft werden, ob eine Division durch $g(x)$ ohne Rest möglich ist. Eine andere Prüfmethode ist die Multiplikation mit dem Prüfpolynom $h(x)$ nach Gleichung 2.60.

Die aufgeführten Codierungsmethoden stellen nur zwei von vielen möglichen dar. Da im folgenden nur diese beiden eine Rolle spielen, sei bei weitergehendem Interesse auf entsprechende Literatur wie z.B. [Bos92], Seite 57ff verwiesen.

Mindestdistanz und Minimalgewicht

Unter der *Mindestdistanz* d_{min} eines Codes $\mathcal{C}$ versteht man die kleinste vorkommende Distanz zwischen zwei verschiedenen Codewörtern $\mathbf{a}$ und $\mathbf{b}$ des Codes:

$$d = \min_{\mathbf{a},\mathbf{b} \in \mathcal{C}} \{d(\mathbf{a}, \mathbf{b})\} \tag{2.70}$$

Der Wert hängt ab von der verwendeten Metrik $d(\mathbf{a}, \mathbf{b})$. Für die RS-Codes und BCH-Codes in Kapitel 2.3.2 ist dies die Hamming-Metrik d_H, für negazyklische Codes dagegen die Lee-Metrik d_L.
Für alle linearen Blockcodes gilt, daß die Mindestdistanz d eines Codes gleich seinem *Minimalgewicht*

$$w_{min} = \min_{\mathbf{a} \in \mathcal{C}} \{w(\mathbf{a})\} \tag{2.71}$$

ist. Dies gilt, da jede Summe und somit auch die Differenz zweier Codewörter wiederum ein Codewort darstellt.
Ein Code mit Mindestdistanz d kann Fehlerwörter vom Gewicht

$$t \leq t_{max} = \left\lfloor \frac{d-1}{2} \right\rfloor \tag{2.72}$$

eindeutig korrigieren. Mann kann auch sagen: Es sind $t \leq t_{max}$ Fehler korrigierbar. Dies wird durch folgende Überlegung deutlich:
In Abbildung 2.7 ist der Zusammenhang für gerade und ungerade Mindestdistanz zweidimensional dargestellt. Um ein Codewort kann man sich eine sogenannte *Korrekturkugel* mit Radius t_{max} denken. Da die Kugeln nicht überlappen, sind die innerhalb liegenden Vektoren eindeutig einem Codewort zugeordnet.

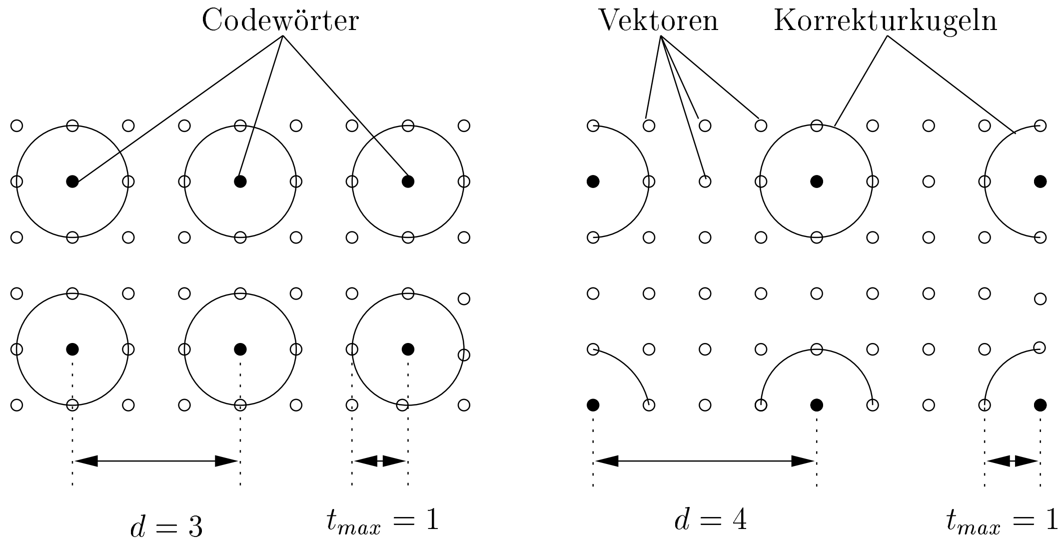

Abbildung 2.7: Korrekturkugeln und Mindestdistanz d

Zusammenhang zwischen Mindestdistanz und Prüfmatrix

Unter Verwendung der Hamming-Metrik ergibt sich folgender Zusammenhang zwischen der Prüfmatrix und der Mindestdistanz eines linearen Blockcodes:
Ein Code mit der Mindestdistanz d besitzt außer dem Nullwort $\mathbf{0}$ kein Codewort $\mathbf{c}$, bei dem weniger als d Komponenten c_i ungleich Null sind. Das heißt, alle Vekoren

mit $d-1$ oder weniger Komponenten $c_i \neq 0$ sind sicher kein Codewort, da für sie gilt $w_H(\mathbf{c}) < d$.
Bei der Prüfmatrix handelt es sich nach den vorangegangenen Ausführungen zur Matrixschreibweise um eine Matrix mit n Spalten und $n-k$ Zeilen mit Rang $n-k$. Das Syndrom stellt eine Linearkombination der Spalten $\mathbf{h}_i$ aus $\mathbf{H}$ dar:

$$\begin{aligned}\mathbf{s}^T &= \mathbf{H} \cdot \mathbf{r}^T = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{n-1}) \cdot (r_0, r_1, \ldots, r_{n-1})^T \\ &= r_0\mathbf{h}_0 + r_1\mathbf{h}_1 + \cdots + r_{n-1}\mathbf{h}_{n-1}\end{aligned}$$

Für ein gültiges Codewort muß Gleichung 2.48 erfüllt sein, d.h. es gilt $\mathbf{s}^T = \mathbf{0}$. Für Vektoren $\mathbf{r}$ mit Hamming-Gewicht $w_H(\mathbf{r}) < d$ muß sich $\mathbf{s}^T \neq \mathbf{0}$ ergeben, da diese Vektoren, keine Codewörter darstellen. Anschaulich bedeutet dies, daß eine Linearkombination von beliebigen $d-1$ (oder weniger) Spalten aus $\mathbf{H}$ nicht Null ergeben darf, wenn ein Code die Mindestdistanz d besitzt. D.h. beliebige $d-1$ Spalten müssen linear unabhängig sein. Es existiert jedoch mindestens eine Kombination von d Spalten, die Null ergibt, da die Mindestdistanz sonst größer als d wäre.
Da $\mathbf{H}$ immer genau den Rang $n-k$ besitzt, damit $n-k$ unabhängige Prüfgleichungen existieren, ergibt nach Definition des Ranges einer Matrix mindestens eine Linearkombination von $n-k+1$ Spalten Null. Es gilt also:

$$d - 1 \leq n - k \qquad \text{(Singleton-Schranke)} \tag{2.73}$$

Ein Code, bei dem Gleichheit in dieser Schranke gilt, wird MDS (Maximum Distance Separable) genannt.

Decodierprinzipien

Betrachtet man in Abbildung 2.3 die Empfangsseite nach dem Kanal, so kann man bezüglich des Demodulators und des Decoders verschiedene Prinzipien unterscheiden. Diese Prinzipien sollen hier jedoch nur kurz angesprochen werden, um zu verdeutlichen, wo die Schwerpunkte der folgenden Kapitel liegen. Für eine ausführliche Beschreibung sei z.B. auf [Bos92] verwiesen.
Der Demodulator oder Empfänger vergleicht ein über den Kanal übertragenes und unter Umständen verfälschtes Symbol mit den möglichen Sendesymbolen des Modulators und gibt den Wert des Symbols aus, das dem empfangenen Symbol am ähnlichsten ist. Er gibt also das am wahrscheinlichsten gesendete Symbol aus. Man unterscheidet zwei Fälle:

- *Hard-Decision-Decodierung (HD)*:
  Der Decoder erhält vom Demodulator außer dem mit größter Wahrscheinlichkeit empfangenen Symbol keine zusätzliche Information. Das Übertragungsmodell in Abbildung 2.4 stellt diese Situation dar, die im folgenden betrachtet werden soll.

- *Soft-Decision-Decodierung (HD)*:
  Der Demodulator liefert dem Decoder zusätzlich einen Wert, der die Zuverlässigkeit seiner Entscheidung bezüglich des empfangenen Symbols angibt. Dieser Fall soll im folgenden nicht weiter untersucht werden.

Für den Decoder sind prinzipiell folgende Ergebnisse möglich:

- *Korrekte Decodierung*:
  Das decodierte Codewort entspricht dem gesendeten Codewort. Ein Fehler ist nicht aufgetreten oder konnte korrigiert werden.

- *Falsche Decodierung*:
  Das decodierte Codewort ist nicht gleich dem gesendeten. Dieser Fall tritt z.B. auf, wenn der Fehlervektor einem gültigen Codewort entspricht.

- *Decodierversagen*:
  Der Decoder kann dem empfangenen Vektor $\mathbf{r}$ kein gültiges Codewort zuordnen. Dieser Fall wird vom Decoder im Gegensatz zu falscher Decodierung erkannt und kann nur bei manchen Decodierverfahren auftreten.

Mit Hilfe der so definierten Decodierergebnisse lassen sich verschiedene Decodierprinzipien beschreiben:

- *Fehlererkennung*:
  Ein Empfangsvektor **w**ird daraufhin geprüft, ob er ein gültiges Codewort darstellt oder nicht. Im ersten Fall ist kein Fehler aufgetreten oder der Fehler selbst ist ein Codewort und kann nicht erkannt werden. Falschdecodierung im Sinne eines nicht erkannten Fehlers ist also möglich, Decodierversagen dagegen nicht.

- *Begrenzte-Mindestdistanz-Decodierung (BMD)*:
  Eine Fehlerkorrektur findet nur statt, wenn der Empfangsvektor $\mathbf{r}$ innerhalb einer Korrekturkugel um ein gültiges Codewort liegt, d.h. wenn die Distanz zum nächstgelegenen Codewort die halbe Mindestdistanz nicht überschreitet. Ist dies doch der Fall, so liegt Decodierversagen vor. Auch die beiden anderen oben beschriebenen Decodierergebnisse sind möglich.
  In der vorliegenden Arbeit wird ausschließlich algebraische BMD-Decodierung behandelt.

- *Decodierung über die halbe Mindestdistanz*:
  Im Gegensatz zur BMD-Decodierung wird hier auch noch für Empfangsvektoren außerhalb von Korrekturkugeln korrigiert. Dennoch können alle Decodierergebnisse auftreten, also auch Decodierversagen.

- *Maximum-Likelihood-Decodierung (ML)*:
  Bei diesem Verfahren liegt Decodierung über die halbe Mindestdistanz vor, jedoch kann kein Decodierversagen auftreten. Für jeden Empfangsvektor $\mathbf{r}$ wird auf das am wahrscheinlichsten gesendete Codewort decodiert, d.h. auf das Codewort, welches die geringste Distanz zum Empfangsvektor besitzt. Sind zwei Codewörter gleich wahrscheinlich, so wird zufällig entschieden. Decodierversagen wird so verhindert.

ML-Decodierung stellt das bestmögliche Decodierverfahren dar. Aus Komplexitätsgründen werden in der Praxis jedoch oft die anderen Verfahren realisiert.

2.3.2 Wichtige zyklische Codes

In diesem Abschnitt sollen zwei bedeutende Vertreter zyklischer Blockcodes vorgestellt werden, Reed-Solomon-Codes und (primitive) BCH-Codes. Beide benutzen die Hamming-Metrik.
Da die grundlegenden Prinzipien der Konstruktion und Decodierung, insbesondere der BCH-Codes, denen der negazyklischen Codes (NC-Codes) für die Lee-Metrik sehr ähnlich sind, sollen sie hier ausführlich behandelt werden, um das Verständnis der NC-Codes zu erleichtern.
Die im folgenden verwendete Nomenklatur orientiert sich dabei weitgehend an der von Berlekamp in [Ber84], um den Übergang zu den negazyklischen Codes so einfach wie möglich zu halten. Diese Schreibweise unterscheidet sich von denen in anderen Quellen wie z.B. [Bla90],[MS83] oder [Bos92]. Auf die grundlegenden Gedanken und Vorgehensweisen hat die gewählte Form jedoch keinen entscheidenden Einfluß.

Reed-Solomon-Codes

Sei $\alpha \in GF(p^m)$ ein Element der Ordnung $n \leq p^m - 1$, so wird ein (n,k,d) RS-Code eindeutig definiert durch das Generatorpolynom

$$g(x) = \sum_{i=1}^{n-k} (x - \alpha^i) \tag{2.74}$$

Der Code $c(x)$ ist damit eindeutig charakterisiert durch die $n-k$ aufeinanderfolgenden Nullstellen $\alpha^1, \ldots, \alpha^{n-k}$ des Generatorpolynoms und jedes Codepolynoms. Man erhält die $n-k$ Prüfgleichungen

$$c(\alpha^i) = 0, \qquad i = 1, \ldots, n-k \tag{2.75}$$

und das Syndrom

$$\begin{aligned} s_i &= r(\alpha^i), \qquad\qquad i = 1, \ldots, n-k \\ &= c(\alpha^i) + e(\alpha^i) \\ &= 0 + e(\alpha^i) \end{aligned} \tag{2.76}$$

Die Prüfmatrix ergibt sich daraus zu

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2)^1 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{n-k})^1 & (\alpha^{n-k})^2 & \cdots & (\alpha^{n-k})^{n-1} \end{pmatrix} \tag{2.77}$$

Bei dieser Prüfmatrix sind beliebige $n-k$ Spalten linear unabhängig, das heißt, ein RS-Code besitzt die Mindestdistanz

$$d = n - k + 1. \tag{2.78}$$

Dies läßt sich mit Hilfe von Vandermonde-Matrizen über einen Widerspruchsbeweis zeigen (siehe [MS83], S.201 f). RS-Codes erfüllen also die Singleton-Schranke mit Gleichheit, sie sind MDS-Codes.

Die Koeffizienten des Informationspolynoms stammen bei RS-Codes ebenso wie die Koeffizienten des Generatorpolynoms und damit auch der Codewörter aus dem Erweiterungskörper $GF(p^m)$.

Wendet man auf das Generatorpolynom die Diskrete Fourier-Transformation nach 2.17 an, so erhält man im transformierten Bereich das Polynom

$$\gamma(x), \qquad mit \quad \gamma_i = g(\alpha^i), \quad i = 0, 1, \dots, \tag{2.79}$$

mit den $n-k$ aufeinanderfolgenden Koeffizienten

$$\gamma_1 = \dots = \gamma_{n-k} = 0. \tag{2.80}$$

Besonders gut sichtbar wird dieser „Block" von Nullen, wenn man für die Transformation die Darstellung mit der Mattson-Solomon Transformationsmatrix wählt. Die Koeffizienten des transformierten Generatorpolynoms

$$\boldsymbol{\gamma} = \mathbf{M}_{(n)} \cdot \mathbf{g}^T \tag{2.81}$$

ergeben sich zu:

$$\begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n-k} \\ \gamma_{n-k+1} \\ \vdots \\ \gamma_{n-1} \end{pmatrix} = \begin{pmatrix} \gamma_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \gamma_{n-k+1} \\ \vdots \\ \gamma_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha^1 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2)^1 & \cdots & (\alpha^2)^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{n-k})^1 & \cdots & (\alpha^{n-k})^{n-1} \\ 1 & (\alpha^{n-k+1})^1 & \cdots & (\alpha^{n-k+1})^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{n-1})^1 & \cdots & (\alpha^{n-1})^{n-1} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{n-k} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2.82}$$

Prinzipiell spielt es für die Decodierung von RS-Codes keine Rolle, welche aufeinanderfolgenden $n-k$ Nullstellen das Generatorpolynom besitzt bzw. wo der „Block" von Nullen im transformierten Bereich liegt. Entscheidend für die Decodierung ist nur die Tatsache, daß sie aufeinanderfolgen.
Aus diesem Grund soll in der vorliegenden Arbeit immer obige Wahl beibehalten werden, d.h. $g(x)$ soll die Nullstellen $\alpha^1, \alpha^2, \dots, \alpha^{n-k}$ enthalten. Dieser Fall wird auch in [Ber84] verwendet und läßt sich leicht auf die ähnliche Beschreibung der NC-Codes übertragen.
Der allgemeinere Fall, in dem $g(x)$ beliebige $n-k$ aufeinanderfolgende Nullstellen besitzt, wird in [Ber84] auf Seite 223 beschrieben. In leicht abweichender Schreibweise findet man diesen Fall z.B. auch in [Bos92], er soll hier nicht weiter verfolgt werden.

Betrachtet man nun ein Codewort $c(x)$ bzw. $\mathbf{c}$ des durch $g(x)$ erzeugten Codes, so muß dieses aufgrund von Gleichung 2.57 die gleichen Nullstellen besitzen wie $g(x)$. Analog sind deshalb im transformierten Codewort

$$\boldsymbol{\zeta} = \mathbf{M}_{(n)} \cdot \mathbf{c}^T, \qquad \mathbf{c} \in \mathcal{C} \tag{2.83}$$

die gleichen Koeffizienten Null wie in $\gamma(x)$, man erhält den gleichen „Block“ von Nullen:

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{n-k} \\ \zeta_{n-k+1} \\ \vdots \\ \zeta_{n-1} \end{pmatrix} = \begin{pmatrix} \zeta_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \zeta_{n-k+1} \\ \vdots \\ \zeta_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha^1 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2)^1 & \cdots & (\alpha^2)^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{n-k})^1 & \cdots & (\alpha^{n-k})^{n-1} \\ 1 & (\alpha^{n-k+1})^1 & \cdots & (\alpha^{n-k+1})^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{n-1})^1 & \cdots & (\alpha^{n-1})^{n-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix} \tag{2.84}$$

Vergleicht man die Gleichungen 2.77 und 2.84, so erkennt man, daß die Prüfmatrix identisch ist mit der 2. bis $(n-k+1)$. Zeile der Transformationsmatrix $\mathbf{M}_{(n)}$. Dies bedeutet, daß die Koeffizienten $\varrho_1, \ldots, \varrho_{n-k}$ des transformierten Empfangswortes $\boldsymbol{\varrho}$ die Syndromkoeffizienten darstellen und nur vom (transformierten) Fehlerwort abhängig sind, nicht jedoch vom gesendeten Codewort. Es gilt nämlich:

$$\boldsymbol{\varrho} = \mathbf{M}_{(n)} \cdot \mathbf{r}^T = \mathbf{M}_{(n)} \cdot (\mathbf{c}^T + \mathbf{e}^T) = \boldsymbol{\zeta} + \boldsymbol{\epsilon} \tag{2.85}$$

$$\begin{pmatrix} \varrho_0 \\ \varrho_1 \\ \varrho_2 \\ \vdots \\ \varrho_{n-k} \\ \varrho_{n-k+1} \\ \vdots \\ \varrho_{n-1} \end{pmatrix} = \begin{pmatrix} \varrho_0 \\ s_1 \\ s_2 \\ \vdots \\ s_{n-k} \\ \varrho_{n-k+1} \\ \vdots \\ \varrho_{n-1} \end{pmatrix} = \begin{pmatrix} \zeta_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \zeta_{n-k+1} \\ \vdots \\ \zeta_{n-1} \end{pmatrix} + \begin{pmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{n-k} \\ \epsilon_{n-k+1} \\ \vdots \\ \epsilon_{n-1} \end{pmatrix} \tag{2.86}$$

Die Auswertung der Syndromkoeffizienten im Rahmen der algebraischen Decodierung wird in Abschnitt 2.3.3 beschrieben. Zunächst soll nun noch auf BCH-Codes eingegangen werden.

BCH-Codes

BCH-Codes (benannt nach R.C.Bose, D.K.Ray Chaudhuri und A.Hocquenhem) können als eine Unterklasse von RS-Codes betrachtet werden.

Im Unterschied zu allgemeinen RS-Codes, bei denen die Informationssymbole Elemente aus $GF(p^m)$ sind, sind bei BCH-Codes die Koeffizienten eines Informationsvektors auch für $m > 1$ aus dem Primkörper $GF(p)$ zu wählen und auch die Koeffizienten von Generatorpolynom und Codewörtern stammen aus $GF(p)$. Da binäre Anwendungen heute überwiegen, wird meist $p = 2$ gewählt.

Wie bei RS-Codes muß das Generatorpolynom mehrere aufeinanderfolgende Nullstellen besitzen. Auch bei BCH-Codes sei hier nur der Fall betrachtet, daß es sich dabei um die Nullstellen $\alpha^1, \alpha^2, \ldots$ handelt und nicht der allgemeine Fall.

Wie im Kapitel Kreisteilungsklassen (2.1.2) beschrieben, besitzt ein Polynom genau dann nur Koeffizienten aus $GF(p)$, wenn es neben einer Wurzel α^i auch alle dazu konjugiert komplexen Elemente als Wurzeln besitzt. Es handelt sich dann um ein minimales Polynom oder das Produkt mehrerer minimaler Polynome.
Diese Tatsache wird bei der Konstruktion des Generatorpolynoms von BCH-Codes berücksichtigt. Ansonsten geht man genauso vor wie bei RS-Codes. Man setzt $g(x)$ praktisch aus aufeinanderfolgenden minimalen Polynomen zusammen.

Sei also α ein primitives Element von $GF(p^m)$ und

$$\mathcal{M} = \bigcup_i K_i$$

die Vereinigungsmenge beliebig vieler Kreisteilungsklassen K_i der Zahl $n = p^m - 1$ und seien m_i die zugehörigen minimalen Polynome, so ist

$$g(x) = \prod_i m_i = \prod_{j \in \mathcal{M}} (x - \alpha^j) \tag{2.87}$$

das Generatorpolynom eines BCH-Codes der Länge n und besitzt nur Koeffizienten aus $GF(p)$.

Analog zu den RS-Codes ist es für die Mindestdistanz eines BCH-Codes entscheidend, wieviele aufeinanderfolgende Nullstellen $g(x)$ besitzt. Besitzt $g(x)$ $d-1$ aufeinanderfolgende Nullstellen, so nennt man d die *geplante Mindestdistanz* eines BCH-Codes. Im Gegensatz zu RS-Codes gilt hier nicht $d-1 = n-k$, BCH-Codes sind *asymptotisch schlecht* (siehe [Bos92], Seite 101).
Die *wirkliche Mindestdistanz* δ ist größer oder gleich der geplanten: $\delta \geq d$.
Die Bestimmung der geplanten Mindestdistanz erfolgt beispielsweise durch Abzählen der aufeinanderfolgenden Nullstellen (siehe Beispiel 2.5). In [Bos92], Seite 93 wird eine einfachere Methode angegeben.

Die Dimension eines BCH-Codes ergibt sich zu

$$k = n - |\mathcal{M}| = n - grad\big(g(x)\big) \tag{2.88}$$

Der entscheidende Unterschied von BCH-Codes zu RS-Codes liegt in den zusätzlichen Nullstellen von $g(x)$, die nicht im Block der aufeinanderfolgenden enthalten sind.

Im transformierten Codewort

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{d-1} \\ \zeta_d \\ \vdots \\ \zeta_{n-1} \end{pmatrix} = \begin{pmatrix} \zeta_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \zeta_d \\ \vdots \\ \zeta_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha^1 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2)^1 & \cdots & (\alpha^2)^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{d-1})^1 & \cdots & (\alpha^{d-1})^{n-1} \\ 1 & (\alpha^d)^1 & \cdots & (\alpha^d)^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{n-1})^1 & \cdots & (\alpha^{n-1})^{n-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix} \tag{2.89}$$

treten nun neben dem „Block“ von Nullen

$$\zeta_1 = \cdots = \zeta_{d-1} = 0 \tag{2.90}$$

noch weitere einzelne Koeffizienten $\zeta_i = 0$ auf.

$$s_i = r(\alpha^i), \qquad i = 1, \ldots, d-1, \tag{2.91}$$

so kann kann auch die Decodierung völlig analog zu RS-Codes erfolgen (siehe Abschnitt 2.3.3). Wie bei RS-Codes spielen dann nur die aufeinanderfolgenden Nullstellen in $g(x)$ eine Rolle und die entsprechende Prüfmatrix lautet:

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2)^1 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{d-1})^1 & (\alpha^{d-1})^2 & \cdots & (\alpha^{d-1})^{n-1} \end{pmatrix} \tag{2.92}$$

In dieser Matrix sind mindestens beliebige $d-1$ Spalten linear unabhängig (siehe [MS83], Seite 201).

Beachtet man, daß mit einer Nullstelle auch alle dazu konjugiert komplexen Elemente Nullstellen des Generatorpolynoms $g(x)$ sein müssen, so genügt es, eine Prüfmatrix aufzustellen, in der je nur eine Wurzel eines minimalen Polynoms überprüft wird. Ist das Generatorpolynom das Produkt aus den l minimalen Polynomen $m_{i_1}, m_{i_2}, \ldots, m_{i_l}$, so erhält man eine gültige Prüfmatrix mit l Zeilen:

$$\mathbf{H}_{min} = \begin{pmatrix} 1 & (\alpha^{i_1})^1 & (\alpha^{i^1})^2 & \cdots & (\alpha^{i_1})^{n-1} \\ 1 & (\alpha^{i_2})^1 & (\alpha^{i_2})^2 & \cdots & (\alpha^{i_2})^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{i_l})^1 & (\alpha^{i_l})^2 & \cdots & (\alpha^{i_l})^{n-1} \end{pmatrix} \tag{2.93}$$

Ersetzt man jedes Element der Matrix durch eine Spalte mit seinen m Komponenten aus $GF(p)$, so erhält man eine Matrix $\mathbf{H}_{min*}$ mit $m \cdot l$ Zeilen. Ein Codewort enthält $n-k = grad\big(g(x)\big)$ Redundanzzeichen und wegen Gleichung 2.13 gilt:

$$grad\big(g(x)\big) \leq m \cdot l. \tag{2.94}$$

Ist der Grad des Generatorpolynoms kleiner als die Anzahl der Matrix-Zeilen, so sind nicht alle Zeilen der Matrix $\mathbf{H}_{min*}$ linear unabhängig und bereits $n-k$ linear unabhängige Zeilen aus $\mathbf{H}_{min*}$ stellen auch eine gültige Prüfmatrix dar (vergleiche [MS83], S.203).

Beispiel 2.5: Konstruktion eines BCH-Codes

Das folgende Beispiel für einen binären BCH-Code stammt aus [Bos92], Seite 93. Es soll hier analog zu der vorangehenden Beschreibung von BCH-Codes angegeben werden.

Es wird $p=2$ und $m=4$ gewählt. Als primitives Polynom wird $p(x)=x^4+x+1$ verwendet, α sei das primitive Element in $GF(2^4)$. Die dadurch erzeugte Logarithmentafel soll an dieser Stelle nicht aufgeführt werden, ist aber in [Bos92] auf Seite 44 abgedruckt.
Der BCH-Codes besitzt dann die Länge $n=p^m-1=15$ und die Kreisteilungsklassen bezüglich n lauten:

$$\begin{aligned} K_1 &= \{1,2,4,8\} & K_3 &= \{5,10\} \\ K_2 &= \{3,6,9,12\} & K_7 &= \{7,11,13,14\} \end{aligned}$$

Man erhält die minimalen Polynome

$$\begin{aligned} m_1 &= (x-\alpha^1)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8) = x^4+x+1 \\ m_3 &= (x-\alpha^3)(x-\alpha^6)(x-\alpha^9)(x-\alpha^{12}) = x^4+x^3+x^2+x+1 \\ m_5 &= (x-\alpha^5)(x-\alpha^{10}) \\ m_7 &= (x-\alpha^7)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{14}). \end{aligned}$$

Wählt man nun

$$\mathcal{M} = K_1 \cup K_3 = \{\underbrace{1,2,3,4}_{4},6,8,9,12\},$$

so lautet das Generatorpolynom $g(x)$ mit Koeffizienten aus $GF(2)$

$$\begin{aligned} g(x) &= m_1(x)\cdot m_3(x) \\ &= x^8+x^7+x^6+x^4+1 \\ &= \underbrace{(x-\alpha^1)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)}_{\text{4 aufeinanderfolgende Wurzeln}}(x-\alpha^6)(x-\alpha^8)(x-\alpha^9)(x-\alpha^{12}). \end{aligned}$$

Da es 4 aufeinanderfolgende Wurzeln enthält, gilt $d-1=4$ und der so konstruierte BCH-Code besitzt die Mindestdistanz

$$d = 5.$$

Die Anzahl der korrigierbaren Fehler beträgt

$$t_{max} = \left\lfloor \frac{d-1}{2} \right\rfloor = \frac{4}{2} = 2$$

und die Dimension des Codes ist

$$k = n - |\mathcal{M}| = 15-8 = 7.$$

Betrachtet man den BCH-Code als RS-Code, so lautet die Prüfmatrix nach Gleichung 2.92:

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2)^1 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ 1 & (\alpha^3)^1 & (\alpha^3)^2 & \cdots & (\alpha^3)^{n-1} \\ 1 & (\alpha^4)^1 & (\alpha^4)^2 & \cdots & (\alpha^4)^{n-1} \end{pmatrix}$$

Da das Generatorpolynom sich aus nur zwei minimalen Polynomen zusammensetzt, stellt nach Gleichung 2.93 auch die folgende Matrix eine gültige Prüfmatrix dar:

$$\begin{aligned}\mathbf{H}_{min} &= \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{14} \\ 1 & (\alpha^3)^1 & (\alpha^3)^2 & \cdots & (\alpha^3)^{14} \end{pmatrix} \\ &= \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{14} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{12} \end{pmatrix}\end{aligned}$$

Wählt man für die Elemente des Erweiterungskörpers die Komponentendarstellung und stellt die 4 Komponenten untereinander in 4 Zeilen dar, so erhält man die binäre Form der Prüfmatrix:

$$\mathbf{H}_{min*} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ & & & & \\ 1 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 1 \end{pmatrix}$$

Da beide in $g(x)$ enthaltenen minimalen Polynome den maximalen Grad $m = 4$ besitzen, ist $n-k = 8$ und die 8 Zeilen der Matrix $\mathbf{H}_{min*}$ sind offenbar linear unabhängig. Sie entsprechen 8 Prüfgleichungen.

□

2.3.3 Algebraische Decodierung von RS- und BCH-Codes mit Hilfe des Berlekamp-Algorithmus

Die Decodierung von RS- und BCH-Codes ist prinzipiell identisch, es muß lediglich beachtet werden, daß die Koeffizienten von Codewörtern, Generator- und Informationspolynomen bei BCH-Codes Elemente aus $GF(p)$ sind, bei RS-Codes dagegen aus $GF(p^m)$. Im folgenden wird vom allgemeineren Fall der RS-Codes ausgegangen. Die notwendigen Berechnungen finden sowohl bei RS- als auch bei BCH-Codes im Erweiterungskörper $GF(p^m)$ mit dem primitivem Element α statt.
Zur Decodierung eines empfangenen und fehlerbehafteten Codeworts $r(x)$ muß der Fehler $e(x)$ berechnet werden. Dazu sei folgende Schreibweise vereinbart: Sind an insgesamt t Stellen Fehler aufgetreten, so lautet das Fehlerpolynom

$$e(x) = e_{j_1}x^{j_1} + e_{j_2}x^{j_2} + \cdots + e_{j_t}x^{j_t}. \tag{2.95}$$

Die *Fehlerwerte* bezeichnet man dann mit

$$Y_i = e_{j_i}, \qquad i = 1, \ldots, t \tag{2.96}$$

und die *Fehlerstellenwerte*

$$X_i = \alpha^{j_i} \tag{2.97}$$

kennzeichnen die Fehlerstellen, an denen $e_{j_i} \neq 0$ gilt. Die Syndromkoeffizienten lassen sich dann wie folgt schreiben:

$$s_k = e(\alpha^k) = \sum_{j=0}^{n-1} e_j(\alpha^k)^j = \sum_{i=1}^{t} Y_i X_i^k \tag{2.98}$$

Nach Gleichung 2.76 bzw. 2.91 sind bei einem RS- oder BCH-Code mit Mindestdistanz d nur die Syndromkoeffizienten $s_1, \ldots, s_{d-1}$ bekannt. Die Anzahl korrigierbarer Fehler beträgt für den Fall ungerader Mindestdistanz d, der im folgenden vorausgesetzt werden soll, $t_m = (d-1)/2$. Im folgenden gilt daher

$$(d-1) = 2t_m. \tag{2.99}$$

Zusätzlich zu den bereits oben verwendeten Polynomen werden nun noch zwei weitere eingeführt:

- Das *Fehlerstellenpolynom* $\sigma(x) \bullet\!\!-\!\!\circ\, f(x)$ wird definiert über die Gleichung:

  $$f_i \cdot e_i \quad = \quad 0 \tag{2.100}$$

  Durch Anwendung des Faltungssatzes erhält man die entsprechende Gleichung im transformierten Bereich:

  $$\sigma(x) \cdot \epsilon(x) \quad = \quad 0 \quad \bmod x^n - 1 \tag{2.101}$$

  Ergänzend soll $\sigma_0 = f\left(\alpha^0\right) = 1$ gelten.

  Diese Definition dient dazu, die Anzahl der Hamming-Fehler, also das Hamming-Gewicht $d_L(\mathbf{e})$ des Fehlervektors auf den Grad des Fehlerstellenpolynoms $\sigma(x)$ abzubilden. Dies ist nötig, da das Ziel der algebraischen Decodierung darin liegt, einen Fehlervektor $\mathbf{e}$ mit möglichst kleinem Gewicht zu finden, der das vorliegende Syndrom erzeugt, und da die Eigenschaft *kleines Gewicht* nicht algebraisch verwertbar ist. Die Eigenschaft *kleiner Grad* eines Polynoms ist dagegen algebraisch zu verwerten.

  Die Abbildung der Fehlerzahl auf den Grad von $\sigma(x)$ funktioniert wie folgt:
  Damit Gleichung 2.100 erfüllt wird, muß gelten:

  $$\begin{aligned} e_{i_j} &\neq 0 \\ \Longrightarrow \qquad f_{i_j} = \frac{1}{n}\sigma(\alpha^{-i_j}) &\stackrel{!}{=} 0. \end{aligned}$$

  D.h. $\sigma(x)$ besitzt die Nullstelle $X_i = \alpha^{-j_i}$, wenn die Komponente r_{j_i} des Empfangswortes fehlerhaft ist.
  Gelingt es, ein $\sigma(x)$ mit möglichst kleinem Grad zu berechnen, dann gilt außerdem für $e_i = 0$ immer $f_i \neq 0$ und der Grad von $\sigma(x)$ ist damit gleich der Anzahl t der aufgetretenen Fehler.

  $$\sigma(x) = \sum_{i=1}^{t}(1 - X_i x) \quad = \quad 1 + \sigma_1 x + \cdots + \sigma_t x^t \tag{2.102}$$
  $$\text{mit} \quad X_a \neq X_b \quad \text{für} \quad a \neq b$$

Das Fehlerstellenpolynom $\sigma(x)$ dient der Bestimmung der Fehlerstellen, nicht aber zur Bestimmung der Fehlerwerte e_i. Die Kehrwerte X_i seiner Wurzeln sind die Fehlerstellenwerte und kennzeichnen die Fehlerstellen des Empfangswortes.

Aufgrund von Gleichung 2.100 kann $\sigma(x)$ nur einfache Wurzeln besitzen. Bei der Decodierung negazyklischer Codes in Kapitel 3.2 ist dies nicht der Fall.

- Das *Fehlerberechnungspolynom* $\omega(x)$ sei folgendermaßen definert:

$$\omega(x) = \sigma(x) + \sum_{l} \left(x X_i Y_i \prod_{j \neq l} (1 - X_j x) \right) \tag{2.103}$$

Durch die Umformungen

$$\begin{aligned}
\omega(x) &= \sigma(x) + \sum_{l=1}^{t} \left(x X_l Y_l \frac{\sigma(x)}{1 - X_l x} \right) \\
&= \sigma(x) \left(1 + \sum_{l=1}^{t} \left(x X_l Y_l \sum_{k=0}^{\infty} (X_l x)^k \right) \right) \\
&= \sigma(x) \left(1 + \sum_{l=1}^{t} \left(Y_l \sum_{k=1}^{\infty} (X_l x)^k \right) \right) \\
&= \sigma(x) \left(1 + \sum_{l=1}^{t} \left(\sum_{k=1}^{\infty} Y_l X_l^k x^k \right) \right) \\
&= \sigma(x) \left(1 + \sum_{k=1}^{\infty} \left(\sum_{l=1}^{t} Y_l X_l^k \right) x^k \right)
\end{aligned}$$

und mit Hilfe des (unendlichen) Syndrom-Polynoms

$$s(x) = \sum_{k=1}^{\infty} s_k x^k = \sum_{k=1}^{\infty} \left(\sum_{l=1}^{t} Y_l X_l^k \right) x^k \tag{2.104}$$

mit Koeffizienten s_k nach Gleichung 2.98 gelangt man zur Gleichung

$$\omega(x) = \sigma(x) \big[1 + s(x) \big] \tag{2.105}$$

Da nur die Koeffizienten $s_1, \dots, s_{2t_m}$ aus dem Empfangsvektor berechnet werden können, muß versucht werden eine Lösung zur Gleichung 2.106 zu finden.

$$\boxed{\big[1 + s(x)\big] \sigma(x) = \omega(x) \mod x^{2t+1}} \tag{2.106}$$

Man bezeichnet diese Gleichung als *Schlüsselgleichung*. Sie entspricht den t_m Gleichungen des Gleichungssystems 2.101, die nur von bekannten Koeffizienten $s_i = e_i$, $i = 1, \dots, 2t_m$ abhängen. Eine darauf basierende Herleitung findet man in [Bos92], Seite 59 ff.

Das eigentliche Problem der Decodierung besteht nun in der Bestimmung eines Fehlerstellenpolynoms $\sigma(x)$ mit möglichst geringem Grad, welches die Schlüsselgleichung erfüllt. Dies ist äquivalent zu der Suche nach einem Fehlervektor $\mathbf{e}$ mit kleinstmöglichem Hamming-Gewicht $d_H(\mathbf{e})$, welcher das Syndrom $s(x)$ erzeugt.

Der *Berlekamp-Algorithmus* übernimmt diese Aufgabe. Er ist in Anhang B.1 in Form eines Ablaufdiagramms dargestellt. Darin wird im Hinblick auf Kapitel 3.2 eine etwas abweichende Schreibweise verwendet. Das Fehlerstellenpolynom wird statt mit $\sigma(x)$ mit $\Phi(x)$ bezeichnet und statt $s(x)$ wird $T(x)$ verwendet. Im Gegensatz zum Berlekamp-Massey-Algorithmus, wie er z.B. in [Bos92] angegeben ist, wird beim Berlekamp-Algorithmus neben dem Fehlerstellenpolynom gleichzeitig das Fehlerberechnungspolynom $\omega(x)$ berechnet.

Liegen die Polynome $\sigma(x)$ und $\omega(x)$ vor, so müssen zunächst die Nullstellen X_i^{-1} von $\sigma(x)$ bestimmt werden, die die Fehlerpositionen angeben. Dies kann beispielsweise durch probieren d.h. einsetzen aller Elemente $\alpha^i \in GF(p^m)$ in $\sigma(x)$ erfolgen. Man nennt diese Art der Nullstellenbestimmung *Chien-Search*.

Der letzte Schritt der Decodierung ist die Fehlerwertberechnung. Sie ist für binäre BCH-Codes mit $e_i \in GF(2)$ nicht erforderlich, da hier ein Fehler nur den Wert 1 besitzen kann. Für nichtbinäre Codes berechnet man Y_i, indem man in Gleichung 2.103 die Kehrwerte X_i^{-1} der Fehlerstellenwerte für x einsetzt und dann nach Y_i auflöst. Man erhält:

$$\begin{aligned}
\omega(X_i^{-1}) &= \sigma(X_i^{-1}) + \sum_{l=1}^{t}\left(X_i^{-1}X_lY_l\prod_{j\neq l}(1-X_jX_i^{-1})\right)\\
&= 0 + Y_i\prod_{j\neq i}(1-X_jX_i^{-1})\\
Y_i &= \frac{\omega(X_i^{-1})}{\prod\limits_{j\neq i}(1-X_jX_i^{-1})} \qquad (2.107)
\end{aligned}$$

In Tabelle 2.1 sind abschließend noch einmal alle verwendeten Polynome und die zugehörigen Koeffizienten-Vektoren aufgelistet.

Informationspolynom	$i(x)$		
Generatorpolynom	$g(x)$	$\circ\!\!-\!\!\bullet$	$\gamma(x)$
Codewort	$c(x) = i(x) \cdot g(x)$	$\circ\!\!-\!\!\bullet$	$\zeta(x)$
	$\mathbf{c}$	$\circ\!\!-\!\!\bullet$	$\boldsymbol{\zeta}$
Fehlerpolynom	$e(x) = \sum_{i=1}^{t} Y_i X_i$	$\circ\!\!-\!\!\bullet$	$\epsilon(x)$
	$\mathbf{e}$	$\circ\!\!-\!\!\bullet$	$\boldsymbol{\epsilon}$
Empfangswort	$r(x) = c(x) + e(x)$	$\circ\!\!-\!\!\bullet$	$\varrho(x) = \varrho_0 + \sum_{i=1}^{d-1} \epsilon_i x^i + \sum_{i=d}^{n-1} \varrho_i x^i$
	$\mathbf{r}$	$\circ\!\!-\!\!\bullet$	$\boldsymbol{\varrho}$
Syndrom			$s(x) = \sum_{i=1}^{d-1} e_i x^i$
			$\mathbf{s} = \mathbf{H} \cdot \mathbf{r}^T$
Fehlerstellenpolynom	$f(x)$	$\circ\!\!-\!\!\bullet$	$\sigma(x) = \prod_{i=1}^{t}(1 - X_i x)$
Fehlerberechnungspolynom			$\omega(x)$

Tabelle 2.1: Variablen bei der algebraischen Decodierung von RS- und BCH-Codes

Kapitel 3

Negazyklische Codes

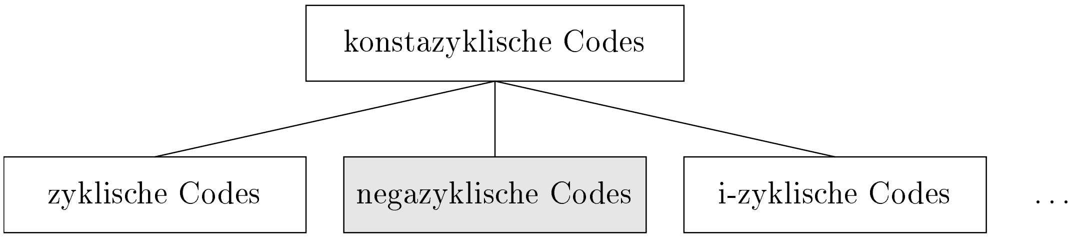

Abbildung 3.1: Einordnung negazyklischer Codes

Die negazyklischen Codes (NC-Codes), welche von Berlekamp in [Ber84] beschrieben werden, gehören ebenso wie die zyklischen Codes zur Oberklasse der konstazyklischen Codes. Ihre Konstruktion verläuft ähnlich wie die der zuvor beschriebenen RS- und BCH-Codes und wird in Kapitel 3.1 beschrieben. Die Decodierung unterscheidet sich stärker von der zyklischer Codes, da statt der Hamming-Metrik die Lee-Metrik verwendet wird. Wichtige Elemente wie der Berlekamp-Algorithmus kommen jedoch auch bei der algebraischen Decodierung negazyklischer Codes in Kapitel 3.2 zur Anwendung.

Zunächst soll noch der Begriff *negazyklisch* näher erläutert werden. In Anlehnung an Gleichung 2.54 bezeichnet man die Operation

$$\circlearrowright_{\kappa,1}(\mathbf{c}) = (\kappa c_{n-1}, c_0, c_1, \dots, c_{n-2}) \tag{3.1}$$

als *konstazyklische Verschiebung* des Vektors $\mathbf{c} = (c_0, c_1, \dots, c_{n-2}, c_{n-1})$, da der einzige Unterschied zu einer zyklischen Verschiebung in der Multiplikation des Elements $\mathbf{c}_{n-1}$ mit der Konstanten κ liegt. Die zyklische Verschiebung stellt also eine konstazyklische Verschiebung mit $\kappa = 1$ dar.

$$\circlearrowright_1(\mathbf{c}) = \circlearrowright_{1,1}(\mathbf{c}) \tag{3.2}$$

Für $\kappa = -1$ spricht man von einer *negazyklischen Verschiebung*:

$$\circlearrowleft_{-1,1}(\mathbf{c}) = (-c_{n-1}, c_0, c_1, \dots, c_{n-2}) \tag{3.3}$$

Das Element c_{n-1} wird bei der Verschiebung negiert. Einen Code, bei dem $\circlearrowleft_{-1,1}(\mathbf{c})$ für jedes beliebige Codewort $\mathbf{c}$ wieder ein gültiges Codewort ergibt, bezeichnet man als *negazyklisch*.
An dieser Stelle sei darauf hingewiesen, daß noch andere konstazyklische Codes existieren. Eine erste Beschreibung *i-zyklischer Codes* für $\kappa = i = \sqrt{-1}$ findet man z.B. in [Hub94b].

Für alle konstazyklischen Codes ist eine Darstellung in Polynomschreibweise möglich. In Gleichung 2.56 wurde bereits die zyklische Verschiebung in Polynomschreibweise angegeben. Allgemein kann man eine konstazyklische Verschiebung beschreiben als

$$\begin{aligned} \circlearrowleft_{\kappa,1}\big(c(x)\big) &= x \cdot c(x) \mod (x^n - \kappa) \\ &= \kappa c_{n-1} + c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1} \end{aligned} \tag{3.4}$$

und für negazyklische Codes ist

$$\begin{aligned} \circlearrowleft_{-1,1}\big(c(x)\big) &= x \cdot c(x) \mod (x^n + 1) \\ &= -c_{n-1} + c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1} \end{aligned} \tag{3.5}$$

wiederum die Darstellung eines Codeworts.

3.1 Konstruktion

NC-Codes kann man besonders gut mit binären BCH-Codes vergleichen.

Bei binären BCH-Codes ist keine Berechnung der Fehlerwerte notwendig, da mit der Information „Fehler in $\mathbf{r}$ an Stelle i“ die Information „Fehlerwert $e_i = 1$“ untrennbar verbunden ist. Dies hängt zusammen mit der verwendeten Hamming-Metrik, die nur unterscheiden kann, ob ein Fehler an der Stelle i aufgetreten ist oder nicht. Das Hamming-Gewicht eines Fehlervektors gibt an, wieviele Fehler innerhalb eines Codeworts aufgetreten sind. Dabei entspricht ein Fehler der Veränderung eines Codesymbols um den Wert 1. Man benötigt bei binären BCH-Codes der Länge n genau n verschiedene Fehlerstellenwerte X_i um eindeutig jede mögliche Fehlerstelle kennzeichnen zu können.

Negazyklische Codes sind im allgemeinen nicht binär, dennoch ist keine getrennte Fehlerwertberechnung notwendig. Bei NC-Codes gibt das Lee-Gewicht des Fehlervektors nach Gleichung 2.45 an, wieviele Fehler aufgetreten sind. Bei einer Verfälschung eines Codesymbols $c_i \in GF(p)$ um den Wert e_i handelt es sich also nicht um einen Fehler vom Wert e_i, sondern um ($e_i \mod {}^{A}p$) Fehler vom Wert 1 oder -1. Man unterscheidet demnach positive und negative Fehler, der Wert eines Fehlers besitzt jedoch immer den Betrag 1.
Bei einem Code der Länge n existieren daher $2n$ verschiedene Fehlermuster mit einem

Fehler vom Betrag 1, also doppelt so viele, wie bei einem binären BCH-Code, da dort die Fehler +1 und −1 nicht unterscheidbar sind. Für NC-Codes benötigt man deshalb $2n$ verschiedene Fehlerstellenwerte X_i. Diese erhält man folgendermaßen:

Sei α ein Element aus $GF(p^m)$ von der Ordnung $2n$, so gilt:

$$\begin{aligned} \alpha^{2n} &= 1 && (3.6) \\ \iff \quad \alpha^n &= -1 && (3.7) \end{aligned}$$

Der Fall $\alpha^n = 1$ ist ausgeschlossen, da die Definition der Ordnung r eines Elements besagt, daß für alle $x < r$ gilt: $\alpha^x \neq 1$. Die nötigen $2n$ Fehlerstellenwerte werden nun folgendermaßen den möglichen Einzelfehler zugeordnet (siehe auch Tabelle 3.1):

- $X_i = \alpha^j, \quad 0 \leq j \leq n-1$
  kennzeichnet einen positiven Fehler im Empfangssymbol r_j
- $X_i = -\alpha^j = \alpha^{n+j}, \quad 0 \leq j \leq n-1$
  kennzeichnet einen negativen Fehler im Empfangssymbol r_j

Fehlerstellenwert für		
positiven Fehler	negativen Fehler	im Empfangs-symbol
α^0	$\alpha^n = -\alpha^0$	r_0
α^1	$\alpha^{n+1} = -\alpha^1$	r_1
α^2	$\alpha^{n+2} = -\alpha^2$	r_2
$\vdots$		$\vdots$
α^j	$\alpha^{n+j} = -\alpha^j$	r_j
$\vdots$	$\vdots$	$\vdots$
α^{n-1}	$\alpha^{2n-1} = -\alpha^{n-1}$	r_{n-1}

Tabelle 3.1: Zuordnung der Fehlerstellenwerte $X_i = \alpha^j$ zu den möglichen Einzelfehlern

Die Länge eines negazyklischen Codes ist beschränkt, da die Ordnung eines Elements aus $GF(p^m)$ maximal den Wert $p^m - 1$ besitzen kann. Besitzt α diese größtmögliche Ordnung, so ist es ein primitives Element und der Code besitzt die maximale mögliche Länge

$$n = \frac{p^m - 1}{2} \tag{3.8}$$

In Anlehnung an die abschließende Überlegung zu zyklischen Codes auf Seite 21 und an Gleichung 2.58 ist ein Code genau dann negazyklisch, d.h. die Verschiebung eines Codeworts $\mathrm{mod}(x^n + 1)$ ist wieder Codewort, wenn für das Produkt aus Generator- und Prüfpolynom gilt:

$$\begin{aligned} g(x) \cdot h(x) &= x^n + 1 \\ &= 0 \quad \mathrm{mod}\ (x^n + 1). && (3.9) \end{aligned}$$

Daraus ergibt sich bei bekanntem Generatorpolynom sofort das Prüfpolynom

$$h(x) = \frac{x^n + 1}{g(x)} \tag{3.10}$$

und für jedes Codewort muß gelten:

$$c(x) \cdot h(x) = 0 \quad \bmod (x^n + 1) \tag{3.11}$$

Da nun wegen Gleichung 3.7 gilt:

$$(\alpha^j)^n = \begin{cases} 1 & , \quad \text{für j gerade} \\ -1 & , \quad \text{für j ungerade,} \end{cases} \tag{3.12}$$

kann α^j nur für ungerades j Nullstelle von $x^n + 1$ sein. Im Gegensatz dazu besitzt $x^n - 1$ nur Nullstellen α^j mit geradem Exponenten j. Dies liefert den Beweis für die Gleichungen 2.15 und 2.16, wenn man berücksichtigt, daß dort n als $p^m - 1$ und nicht wie hier als $(p^m - 1)/2$ definiert ist.

Neben der Einschränkung, daß $g(x)$ nur Nullstellen α^j mit ungeraden Exponenten besitzen darf, kommt bei negazyklischen Codes prinzipiell keine zu denen von BCH-Codes hinzu. Man wählt völlig analog eine Menge beliebiger Kreisteilungsklassen K_i ($i\ ungerade$):

$$\mathcal{M} = \bigcup_{i\ ungerade} K_i$$

$$g(x) = \prod_{i\ ungerade} m_i = \prod_{j \in \mathcal{M}} (x - \alpha^j) \tag{3.13}$$

Dabei sind K_i die Kreisteilungsklassen bezüglich $p^m - 1$ und enthalten für $p \neq 2$ nur ungerade Elemente, wenn i ungerade ist. Bei $m_i(x)$ handelt es sich um die zugehörigen minimalen Polynome. Die Koeffizienten des Generatorpolynoms sind aufgrund seiner Konstruktion aus $GF(p)$ und gleiches gilt für die Codepolynome und muß für die Wahl der Informationspolynome gelten.

Die Dimension eines NC-Codes ergibt sich wie bei BCH-Codes zu

$$k = n - |\mathcal{M}| = n - grad\big(g(x)\big) \tag{3.14}$$

Die Korrekturfähigkeit von NC-Codes wird wie bei RS- und BCH-Codes durch die Anzahl aufeinanderfolgender Wurzeln im Generatorpolynom bestimmt. Sind die Wurzeln $\alpha^1, \alpha^2, \ldots, \alpha^{2t_{max}}$ enthalten, können bei RS- und BCH-Codes mit algebraischer Decodierung t_{max} Fehler korrigiert werden.
Bei NC-Codes ist die Anzahl der aufeinanderfolgenden Wurzeln α^j mit ungeraden Exponenten j entscheidend. Es gilt:

Enthält ein Generatorpolynom $g(x)$ eines NC-Codes über $GF(p)$ die aufeinanderfolgenden Wurzeln $\alpha^1, \alpha^3, \alpha^5, \dots, \alpha^{2t_m-3}, \alpha^{2t_m-1}$, wobei

$$t_m \leq \frac{p-1}{2} \tag{3.15}$$

gelte und $\alpha \in GF(p^m)$ eine $2n$-te Einheitswurzel sei, so kann dieser Code der Länge $n = (p^m - 1)/2$ bis zu t_m Fehler korrigieren. D.h. er kann alle Fehlervektoren $\mathbf{e}$ der Länge n vom Lee-Gewicht $t \leq t_m$ korrigieren.

Einen konstruktiven Beweis hierzu liefert Berlekamp in [Ber84], Seite 213ff in Form eines Decodieralgorithmus, der in Kapitel 3.2 beschrieben wird.
Zuvor soll jedoch noch die Prüfmatrix und das Syndrom angegeben werden. Dazu werden zunächst die Koeffizienten des Mattson-Solomon Polynoms $\zeta(x)$ eines Codeworts $c(x)$ betrachtet:

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{2t_m-1} \\ \zeta_{2t_m} \\ \vdots \\ \zeta_{n-1} \end{pmatrix} = \begin{pmatrix} \zeta_0 \\ 0 \\ \zeta_2 \\ \vdots \\ 0 \\ \zeta_{2t} \\ \vdots \\ \zeta_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha^1 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2)^1 & \cdots & (\alpha^2)^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{2t_m-1})^1 & \cdots & (\alpha^{2t_m-1})^{n-1} \\ 1 & (\alpha^{2t_m})^1 & \cdots & (\alpha^{2t_m})^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{n-1})^1 & \cdots & (\alpha^{n-1})^{n-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix} \tag{3.16}$$

Aufgrund der aufeinanderfolgenden Nullstellen in $g(x)$ gilt für jedes Codewort:

$$\zeta_1 = \zeta_3 = \zeta_5 = \cdots = \zeta_{2t_m-1} = 0.$$

Außerdem sind noch weitere einzelne Koeffizienten ζ_i Null, die aber wie bei BCH-Codes nicht zur algebraischen Decodierung genutzt werden. Die Prüfmatrix eines NC-Codes ergibt sich demnach zu:

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^3)^1 & (\alpha^3)^2 & \cdots & (\alpha^3)^{n-1} \\ 1 & (\alpha^5)^1 & (\alpha^5)^2 & \cdots & (\alpha^5)^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & (\alpha^{2t_m-1})^1 & (\alpha^{2t_m-1})^2 & \cdots & (\alpha^{2t_m-1})^{n-1} \end{pmatrix} \tag{3.17}$$

Wie $\mathbf{H}_{min}$ bei BCH-Codes besitzt $\mathbf{H}$ so viele Zeilen wie minimale Polynome in $g(x)$ verwendet werden (vergleiche Gleichung 2.93).

Die Koeffizienten des Syndroms $\mathbf{s} = \mathbf{H} \cdot \mathbf{r}^T$ berechnen sich aus dem Empfangsvektor $\mathbf{r}$ analog zu Gleichung 2.76 bzw. 2.91 wie folgt

$$\begin{aligned} s_l &= r(\alpha^l), \qquad l = 1, 3, 5, \dots, 2t_m - 1 \\ &= e(\alpha^l) = \epsilon_l, \end{aligned} \tag{3.18}$$

und sind nur vom aufgetretenen Fehler $e(x)$ abhängig. Alle Syndromkoeffizienten mit geradem Index sind hier im Unterschied zu BCH-Codes unbekannt.

Beispiel 3.1: Konstruktion eines NC-Codes

Dieses Beispiel ist strukturiert wie Beispiel 2.5 zur Konstruktion von BCH-Codes, damit ein direkter Vergleich möglich ist.

Es wird $p = 5$ und $m = 2$ gewählt. Als primitives Polynom wird $p(x) = x^2 + x + 2$ verwendet, α sei das primitive Element in $GF(5^2)$. Die dadurch erzeugte Logarithmentafel ist in Beispiel 2.2 auf Seite 6 angegeben.
Der NC-Codes besitzt dann die Länge $n = (p^m - 1)/2 = 12$ und die Kreisteilungsklassen bezüglich $p^m - 1 = 24$ lauten:

$$
\begin{aligned}
K_1 &= \{1,5\} \\
K_3 &= \{3,15\} \\
K_7 &= \{7,11\} \\
K_9 &= \{9,21\} \\
K_{13} &= \{13,17\} \\
K_{19} &= \{19,23\}
\end{aligned}
\qquad\qquad
\begin{aligned}
K_2 &= \{2,10\} \\
K_4 &= \{4,20\} \\
K_6 &= \{6\} \\
K_8 &= \{8,16\} \\
K_{12} &= \{12\} \\
K_{14} &= \{14,22\} \\
K_{18} &= \{18\}
\end{aligned}
$$

Man sieht, daß eine Kreisteilungsklasse entweder nur gerade oder nur ungerade Zahlen enthält.

Zur Konstruktion eines NC-Codes werden nach Gleichung 3.13 nur die folgenden minimalen Polynome m_i mit ungeradem i verwendet:

$$
\begin{aligned}
m_1 &= (x-\alpha^1)(x-\alpha^5) = x^2 + x + 2 \\
m_3 &= (x-\alpha^3)(x-\alpha^{15}) = x^2 + 3 \\
m_7 &= (x-\alpha^7)(x-\alpha^{11}) \\
m_9 &= (x-\alpha^9)(x-\alpha^{21}) \\
m_{13} &= (x-\alpha^{13})(x-\alpha^{17}) \\
m_{19} &= (x-\alpha^{19})(x-\alpha^{23})
\end{aligned}
$$

Wählt man

$$\mathcal{M} = K_1 \cup K_3 = \{\underbrace{1,3,5,}_{3}15\},$$

so lautet das Generatorpolynom des NC-Codes

$$
\begin{aligned}
g(x) &= m_1(x) \cdot m_3(x) \\
&= x^4 + x^3 + 3x + 2 \\
&= \underbrace{(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)}_{3}(x-\alpha^{15})
\end{aligned}
$$

Es besitzt nur Koeffizienten aus $GF(5)$ und hat $t_{Block} = 3$ aufeinanderfolgende Wurzeln α^i mit ungeradem Exponenten.
Bedingung 3.15 ist für $t_m = t_{Block}$ nicht erfüllt:

$$t_{Block} = 3 \quad > \quad 2 = \frac{p-1}{2}$$

Man wählt daher für t_m den größten Wert, der Gleichung 3.15 noch erfüllt:

$$t_m = \frac{p-1}{2} = 2$$

Der so konstruierte Code kann also (bei Decodierung nach Berlekamp) 2 Fehler korrigieren. Er besitzt die Dimension

$$k = n - |\mathcal{M}| = 12 - 4 = 8.$$

Nach Gleichung 3.17 erhält man die folgende Prüfmatrix:

$$\begin{aligned} \mathbf{H} &= \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{11} \\ 1 & (\alpha^3)^1 & (\alpha^3)^2 & \cdots & (\alpha^3)^{11} \end{pmatrix} \\ &= \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{11} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^9 \end{pmatrix} \end{aligned}$$

Geht man nun wie in Beispiel 2.5 vor und verwendet für die Elemente aus $GF(5^2)$ die Komponentendarstellung, so erhält man eine Prüfmatrix mit Elementen aus $GF(5)$:

$$\mathbf{H}_* = \begin{pmatrix} 1 & 0 & 3 & \cdots & 3 \\ 0 & 1 & 4 & \cdots & 3 \\ & & & & \\ 1 & 2 & 2 & \cdots & 4 \\ 0 & 4 & 0 & \cdots & 3 \end{pmatrix}$$

□

3.2 Algebraische Decodierung negazyklischer Codes

Sei folgender Fehler

$$e(x) = e_o + e_1 x + e_2 x^2 + \cdots + e_{n-1} x^{n-1}.$$

vom Lee-Gewicht

$$t = w_L\big(e(x)\big) \le t_m, \tag{3.19}$$

d.h. mit maximal t_m Einzelfehlern aufgetreten, dann läßt sich das Syndrom mit Hilfe der in Tabelle 3.1 definierten Fehlerstellenwerte

$$X_i = \alpha^{j_i}$$

auch wie folgt schreiben:

$$s_l = e(\alpha^l) = \sum_{j=0}^{n-1} e_j (\alpha^l)^j = \sum_{i=1}^{t} X_i^l \qquad l = 1, 3, 5, \ldots, 2t_m - 1 \tag{3.20}$$

Die Syndromkoeffizienten sind also Potenzsummen gemäß Gleichung 2.23. Vergleicht man diese Darstellung mit Gleichung 2.98, so fällt auf, daß in (3.20) die Fehlerwerte

Y_i nicht vorkommen. Dies liegt daran, daß Einzelfehler immer nur die Werte $+1$ oder -1 besitzen können und daß das Vorzeichen mit Hilfe zweier verschiedener X_i berücksichtigt wird. Treten an einer Stelle j mehrere Fehler auf, d.h. es gilt $|e_j| > 1$, so besitzen zwei oder mehr verschiedene Fehlerstellenwerte X_i den gleichen Wert $\pm\alpha^j$, was bei RS- und BCH-Codes nicht erlaubt ist. Gerade aus diesem Grund kann aber das *Fehlerstellenpolynom* $\sigma(x)$ für NC-Codes genauso definiert werden wie für RS- und BCH-Codes in Gleichung 2.102:

$$\sigma(x) = \prod_{i=1}^{t}(1 - X_i x) = 1 + \sigma_1 x + \cdots + \sigma_t x^t \tag{3.21}$$

Wie erwähnt sind hier jedoch mehrfache Nullstellen X_i^{-1} möglich, da in einem Codesymbol mehrere Fehler auftreten können. Bei bekanntem $\sigma(x)$ kann man dessen Nullstellen X_i^{-1} wie bei zyklischen Codes durch Chien-Search bestimmen, kennt damit die Fehlerstellen (und Werte ± 1) und somit auch das Fehlerpolynom $e(x)$.

Ziel der algebraischen Decodierung ist die Bestimmung eines Fehlerpolynoms $e(x)$ mit möglichst geringem Lee-Gewicht t, welches das Syndrom $s(x)$ erzeugt. Dies entspricht der Suche nach einem Fehlerstellenpolynom $\sigma(x)$ mit möglichst kleinem Grad t. Der Zusammenhang zwischen den bekannten Syndromkoeffizienten (Potenzsummen) s_l und den Koeffizienten des Polynoms $\sigma(x)$ wird durch die *Newtonschen Gleichungen* 2.30 beschrieben. Es wird nun zunächst die Polynom-Darstellung der Newtonschen Gleichungen nach [Ber84] hergeleitet.
Mit Hilfe der Produktregel errechnet sich die Ableitung von $\sigma(x)$ zu

$$\sigma'(x) = -\sum_{i=1}^{t} X_i \prod_{j\neq i}(1 - X_j x) = -\sum_{i=1}^{t} X_i \frac{\prod_{j=1}^{t}(1 - X_j x)}{(1 - X_i x)} \tag{3.22}$$

und man erhält

$$\begin{aligned} -\frac{x\sigma'(x)}{\sigma(x)} &= \frac{x\sum_{i=1}^{t} X_i \frac{\prod_{j=1}^{t}(1-X_j x)}{(1-X_i x)}}{\prod_{i=1}^{t}(1 - X_i x)} = \frac{\prod_{j=1}^{t}(1 - X_j x)\cdot\sum_{i=1}^{t}\frac{X_i x}{(1-X_i x)}}{\prod_{i=1}^{t}(1 - X_i x)} \\ &= \sum_{i=1}^{t}\frac{X_i x}{(1 - X_i x)} \end{aligned}$$

Die Summanden können als geometrische Reihe dargestellt werden und es ergibt sich

$$-\frac{x\sigma'(x)}{\sigma(x)} = \sum_{i=1}^{t}\sum_{k=1}^{\infty}(X_i x)^k = \sum_{k=1}^{\infty} x^k \sum_{i=1}^{t} X_i^k = \sum_{k=1}^{\infty} x^k s_k = s(x)$$

Hierbei stellt $s(x)$ das (unendliche) Syndrompolynom aus Gleichung 2.104 dar. Man erhält die Polynomdarstellung der Newtonschen Gleichungen:

$$s(x)\sigma(x) + x\sigma'(x) = 0 \tag{3.23}$$

Jeder Koeffizient des Polynoms muß Null sein, dies entspricht jeweils einer Gleichung im Gleichungssystem 2.30, was man durch Ausmultiplizieren überprüfen kann.

Da die Koeffizienten s_l mit geraden Index l bei negazyklischen Codes unbekannt sind, trennt man $s(x)$ und auch $\sigma(x)$ jeweils in zwei Polynome auf: Die Polynome s_e und σ_e mit geraden Potenzen von x und die Polynome s_o und σ_o mit ungeraden Potenzen.

$$\begin{aligned}
\sigma_e(x) &= 1 + \sigma_2 x^2 + \sigma_4 x^4 + \cdots \\
\sigma_o(x) &= \sigma_1 x + \sigma_3 x^3 + \cdots
\end{aligned}$$

$$\begin{aligned}
\text{unbekannt:} \quad s_e(x) &= 1 + s_2 x^2 + s_4 x^4 + \cdots \\
\text{bekannt:} \quad s_o(x) &= s_1 x + s_3 x^3 + \cdots
\end{aligned}$$

Der Übersichtlichkeit halber wird in den folgenden Umformungen das Argument x bei den Polynomen weggelassen.

Durch einsetzen in (3.23) ergibt sich

$$\begin{aligned}
0 &= (s_o + s_e)(\sigma_o + \sigma_e) + x(\sigma_o + \sigma_e)' \\
&= \underbrace{s_o\sigma_o + s_e\sigma_e + x\sigma_e'}_{gerade\ Potenzen\ von\ x} + \underbrace{s_o\sigma_e + s_e\sigma_o + x\sigma_o'}_{ungerade\ Potenzen\ von\ x}
\end{aligned}$$

Man erhält so eine Aufteilung von Gleichung 3.23 in eine Gleichung mit geraden und eine mit ungeraden Potenzen von x:

$$\begin{aligned}
s_o\sigma_o + s_e\sigma_e + x\sigma_e' &= 0 \\
s_o\sigma_e + s_e\sigma_o + x\sigma_o' &= 0
\end{aligned}$$

Durch Multiplikation dieser beiden Gleichungen mit σ_o bzw. σ_e und anschließende Subtraktion erhält man die Gleichung

$$s_o(\sigma_o^2 + \sigma_e^2) + x(\sigma_o\sigma_e' - \sigma_e\sigma_o') = 0 \tag{3.24}$$

welche nur noch $s_o(x)$, also die Koeffizienten s_l des Syndroms mit ungeraden Indizes, enthält und unter bestimmten Bedingungen gelöst werden kann.

An dieser Stelle beginnt der eigentliche konstruktive Beweis von Berlekamp zur Korrekturfähigkeit negazyklischer Codes, der die Grundlage für eine algebraische Decodierung darstellt. Prinzipiell ist der Beweis die Umformung von Gleichung 3.24 in eine Form, die der Schlüsselgleichung 2.106 entspricht und die mit Hilfe des Berlekamp-Algorithmus gelöst werden kann.

Zunächst wird (3.24) durch σ_e^2 geteilt.

$$s_o\left(\left(\frac{\sigma_o}{\sigma_e}\right)^2) - 1\right) = x\left(\frac{\sigma_o\sigma_e' - \sigma_e\sigma_o'}{\sigma_e^2}\right) = x\left(\frac{\sigma_o}{\sigma_e}\right)' \tag{3.25}$$

Abkürzend kann man dann schreiben:

$$U(x) = \frac{\sigma_o(x)}{\sigma_e(x)} \tag{3.26}$$

$$\Longrightarrow \qquad s_o\big(U^2(x) - 1\big) = xU'(x)$$

Durch Auflösen nach $U'(x)$ und anschließende Integration nach x erhält man

$$U(x) = \int \frac{1}{x} s_o(x) \left(-1 + U^2(x)\right) \, dx \tag{3.27}$$

Wertet man das Integral aus, stellt man fest, daß $U(x)$ nur Potenzen von x mit ungeraden Exponenten besitzt. Es gilt:

$$\begin{aligned}
U_i &= 0 \qquad \text{für } i \text{ ungerade} && (3.28)\\
U_1 &= \frac{-s_1}{1}\\
U_3 &= \frac{-s_3 + s_3 U_1^2}{3}\\
U_5 &= \frac{-s_5 + s_3 U_1^2 + s_1 2(U_1 U_3)}{5}\\
U_7 &= \frac{-s_7 + s_5 U_1^2 + s_3(2U_1U_3) + s_1(2U_1U_5 + U_3^2)}{7}\\
&\vdots\\
U_{(2j-1)} &= \frac{-s_{(2j-1)} + \sum\limits_{i=1}^{j-1} s_{(2i-1)} \sum\limits_{k=1}^{j-i-1} U_k U_{(2(j-i-1)-k)}}{2j-1} && (3.29)
\end{aligned}$$

Bei einem NC-Code über dem Primkörper $GF(p)$ tritt bei der Berechnung des Koeffizienten U_p eine Division durch $p \equiv 0$ auf. Um dieses Problem zu vermeiden, sollen im folgenden immer nur die Koeffizienten $U_1, U_3, \ldots, U_{(2t_m-1)}$ berechnet werden. Wegen der Bedingung $2t_m - 1 < p$ tritt das Problem dann nicht mehr auf. Es gilt also ab hier:

$$\begin{aligned}
U(x) &= U_1 x + U_3 x^3 + \cdots + U_{2t_m-1} x^{2t_m-1}\\
&= \frac{\sigma_o(x)}{\sigma_e(x)} \quad \bmod x^{2t_m} && (3.30)
\end{aligned}$$

Da $U(x)$ eine ungerade Funktion in x ist, kann man eine gerade Funktion $T(x^2)$ wie folgt definieren:

$$\begin{aligned}
T(x^2) &= \big[1 + xU(x)\big]^{-1} - 1 && (3.31)\\
&= T_1 x^2 + T_2 x^4 + \cdots + T_{t_m} x^{2t_m}
\end{aligned}$$

Die Koeffizienten T_i lassen sich durch die folgende Umformung und einen anschließenden Koeffizientenvergleich bestimmen.

$$\begin{aligned}
1 + T(x^2) &= \big[1 + xU(x)\big]^{-1} && (3.32)\\
1 &= \big[1 + T(x^2)\big]\big[1 + xU(x)\big]
\end{aligned}$$

Man erhält:

$$\begin{aligned}
T_1 &= -U_1 \\
T_2 &= -U_3 - T_1U_1 \\
T_3 &= -U_5 - T_1U_3 - T_2U_1 \\
T_i &= -U_{2i-1} - T_1U_{2i-3} - T_2U_{2i-5} - \cdots - T_{i-1}U_1 \\
&= -U_{2i-1} - \sum_{j=1}^{i-1} T_jU_{2i-2j-1} \qquad (i = 1, 2, \ldots, t_m)
\end{aligned} \tag{3.33}$$

Da demnach zur Berechnung von T_i nur die Koeffizienten $U_1, \ldots, U_{2i-1}$ und $T_1, \ldots, T_{i-1}$ benötigt werden, können U_{2i-1} und T_i abwechselnd rekursiv berechnet werden.

Führt man nun noch folgende Hilfspolynome ein

$$\omega(x^2) = \sigma_e(x) \tag{3.34}$$

$$\Phi(x^2) = \sigma_e(x) + x\sigma_o(x); \tag{3.35}$$

so ergibt sich nach Gleichung 3.30

$$\begin{aligned}
1 + xU(x) &= 1 + x\frac{\sigma_o(x)}{\sigma_e(x)} \\
&= \frac{\sigma_o(x) + x\sigma_e(x)}{\sigma_e(x)} = \frac{\Phi(x^2)}{\omega(x^2)} \\
\overset{(3.32)}{\Longrightarrow} \qquad 1 + T(x^2) &= \frac{\omega(x^2)}{\Phi(x^2)} \\
\Longrightarrow \qquad 1 + T(x) &= \frac{\omega(x)}{\Phi(x)}
\end{aligned}$$

und man erhält die Gleichung

$$\boxed{\big[1 + T(x)\big]\Phi(x) = \omega(x)} \tag{3.36}$$

die der Schlüsselgleichung 2.106 entspricht. Unter bestimmten Bedingungen läßt sich diese Gleichung mit Hilfe des Berlekamp-Algorithmus lösen. Geht man davon aus, daß nicht mehr als t_m Fehler aufgetreten sind, so muß gelten:

1. $\Phi(0) = \omega(0) = 0$
2. Grad $\Phi \leq \dfrac{1 + t_m}{2}$
3. Grad $\omega \leq \dfrac{t_m}{2}$
4. $\Phi(x)$ und $\omega(x)$ sind relativ prim
   $\Longrightarrow \quad \sigma_e(x)$ und $\sigma_o(x)$ sind relativ prim und besitzen somit keine gemeinsamen Nullstellen.

Unter diesen Voraussetzungen wird mit Hilfe des Berlekamp-Algorithmus (Abbildung B.1) ein Polynom $\Phi(x)$ mit möglichst kleinem Grad sowie das zugehörige $\omega(x)$ bestimmt, die Gleichung 3.36 lösen.

Im Unterschied zur Decodierung zyklischer Codes muß anschließend aus $\Phi(x)$ und $\omega(x)$ noch das Fehlerstellenpolynom $\sigma(x)$ berechnet werden. Dies geschieht mit Hilfe der Gleichungen 3.34 und 3.35 wie folgt:

$$\sigma_e(x) = \omega(x^2) \tag{3.37}$$

$$\sigma_o(x) = \frac{\Phi(x^2) - \omega(x^2)}{x} \tag{3.38}$$

$$\sigma(x) = \sigma_e(x) + \sigma_o(x) \tag{3.39}$$

Im letzten Schritt der Decodierung müssen noch die Nullstellen $X_1^{-1}, X_2^{-1}, \ldots, X_t^{-1}$ des Fehlerwertpolynoms $\sigma(x)$ berechnet werden, um auf das Fehlerpolynom $e(x)$ schließen zu können. Dabei wird folgendermaßen vorgegangen:
Man geht von einem Fehlerpolynom $e(x) = e_0 + e_1 x + \cdots + e_{n-1}x^{n-1} = 0$ bzw. Fehlervektor $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1}) = \mathbf{0}$ aus. Das Polynom $r(x) = r_0 + r_1 x + \cdots + r_{n-1}x^{n-1}$ stellt den empfangenen Vektor dar.
Man berechnet nun für $j = 0, 1, \ldots, n-1$ jeweils die beiden Werte a_j und b_j:

$$a_j = \sigma_e(x = \alpha^{-j}) \tag{3.40}$$

$$b_j = \sigma_o(x = \alpha^{-j}) \tag{3.41}$$

Anschließend wird für jedes j geprüft, welche der folgenden Aussagen zutrifft:

- $a_j = -b_j \implies$ positiver Fehler in $r_j \implies e_j := e_j + 1$
  Begründung: $a_j + b_j = \sigma(\alpha^{-j}) = 0$, d.h. α^{-j} ist Nullstelle von $\sigma(x)$. Dieser Nullstelle entspricht der Fehlerstellenwert $X_i = \alpha^j$, der wegen $j < n$ einen positiven Fehler markiert (siehe Tabelle 3.1).

- $a_j = b_j \implies$ negativer Fehler in $r_j \implies e_j := e_j - 1$
  Begründung: $a_j - b_j = \sigma_e(\alpha^{-j}) - \sigma_o(\alpha^{-j}) = \sigma_e(-\alpha^{-j}) + \sigma_o(-\alpha^{-j}) = \sigma(-\alpha^{-j}) = 0$, d.h. $-\alpha^{-j} = \alpha^{n-j}$ ist Nullstelle von $\sigma(x)$. Dieser Nullstelle entspricht der Fehlerstellenwert $X_i = \alpha^{j-n} = \alpha^{n+j}$, der nach Tabelle 3.1 einen negativen Fehler kennzeichnet.

- $a_j = b_j = 0 \implies$ Decodierversagen
  Begründung: α^j und α^{-j} sind Nullstellen von $\sigma(x)$, d.h. in r_j ist ein positiver und ein negativer Fehler gleichzeitig aufgetreten - also kein Fehler (vgl. Bedingung 4 zu Gleichung 3.36).

- sonst: kein (weiterer) Fehler in r_i

Ist danach das Lee-Gewicht des Fehlers $w_L(\mathbf{e})$ noch kleiner als der Grad t des Fehlerstellenpolynoms $\sigma(x)$, so besitzt $\sigma(x)$ mehrfache Nullstellen, d.h. man muß noch weitere Fehlerstellenwerte bestimmen. Dazu ersetzt man σ_e und σ_o durch ihre Ableitungen

$$\sigma_e(x) := \frac{d}{dx}\sigma_e(x) \tag{3.42}$$
$$\sigma_o(x) := \frac{d}{dx}\sigma_o(x) \tag{3.43}$$

und beginnt wieder mit der Berechnung der Werte a_j und b_j (Gleichungen 3.40 und 3.41). Dabei kann man sich auf die Werte j beschränken, für die $e_j \neq 0$ gilt.

Insgesamt durchläuft man diesen Kreislauf von Auswerten und Differenzieren solange, bis man alle Nullstellen von $\sigma(x)$ gefunden und damit das Fehlerpolynom $e(x)$ mit dem Lee-Gewicht t vollständig bestimmt hat.

Wie sich im Laufe der Implementierung des Decodierverfahrens herausgestellt hat, genügt es zum Erkennen von Decodierversagen nicht, nur die Bedingungen 1 - 4 der Gleichung 3.36 zu überprüfen. Daher wird zur Erkennung eines möglichen Decodierversagens abschließend geprüft, ob der korrigierte Empfangsvektor $\mathbf{r} - \mathbf{e}$ ein gültiges Codewort darstellt. Ist dies der Fall, so gibt der Decodieralgorithmus als Ergebnis $\mathbf{r} - \mathbf{e}$ aus. Im anderen Fall liegt Decodierversagen vor, es kann keine Korrektur vorgenommen werden und $\mathbf{r}$ wird unverändert ausgegeben.

Aus dem Ausgabevektor der Decodierung muß schließlich noch die Information zurückgewonnen werden. Hierbei muß muß das verwendete Codierverfahren berücksichtigt werden (siehe Kapitel 2.3.1, Seite 21).

Ein Flußdiagramm zu dem oben beschriebenen Decodieralgorithmus ist in Anhang B.2 abgebildet.

Beispiel 3.2: Algebraische Decodierung eines NC-Codes

Als Beispiel für die Decodierung wird ein anderer NC-Code gewählt als in Beispiel 3.1, da die Berechnungen im Erweiterungskörper $GF(5^2)$ relativ aufwendig sind.
Man findet dieses Beispiel in etwas weniger ausführlicher Form auch in [Ber84] auf Seite 217.

Es wird $p = 31$ und $m = 1$ gewählt, sodaß jedes Element durch nur eine Komponente dargestellt wird. Als primitives Polynom wird $p(x) = x - 3$ verwendet, $\alpha = 3$ ist das zugehörige primitive Element in $GF(31)$. Es ergibt sich die folgende Logarithmentafel:

Exp.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Komp.	1	3	9	27	19	26	16	17	20	29	25	13	8	24	10

Exp.	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Komp.	30	28	22	4	12	5	15	14	11	2	6	18	23	7	21
≡	−1	−3	−9	−27	−19	−26	−16	−17	−20	−29	−25	−13	−8	−24	−10

Der NC-Codes besitzt die Länge $n = (31 - 1)/2 = 15$ und jede Kreisteilungsklasse besitzt nur ein Element:

$$K_i = \{i\}$$

Die zugehörigen minimalen Polynome lauten :

$$m_i = (x - \alpha^i) = (x - 3^i)$$

Das Generatorpolynom eines 4 Fehler korrigierenden Codes lautet dann

$$\begin{aligned} g(x) &= m_1(x) \cdot m_3(x) \cdot m_5(x) \cdot m_7(x) = x^4 + 20x^3 + 15x^2 + 28x + 28 \\ &= \underbrace{(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)}_{t_m=4}. \end{aligned}$$

Bedingung 3.15 ist erfüllt:

$$2t_m - 1 \quad = \quad 7 < p$$

Der Code besitzt die Dimension

$$k \quad = \quad n - |\mathcal{M}| = 12 - 4 = 8.$$

Ist $\mathbf{i} = (14, 0, 25, 5, 16, 28, 27, 7, 14, 16, 15)$ der zu übertragende Informationsvektor, so ergibt sich bei Codierung durch Multiplikation mit dem Generatorpolynom (siehe Seite 21) das Codewort $\mathbf{c}$.
Bei der Übertragung über einen gestörten Kanal sei der Fehlervektor $\mathbf{e}$ vom Lee-Gewicht 4 aufgetreten und es wurde der Vektor $\mathbf{r}$ empfangen:

$$\begin{array}{rl} & \mathbf{c}= (20,20,11,4,16,9,14,23,19,25,5,17,1,6,15) \\ + & \mathbf{e}= (0,0,0,1,0,0,0,-1,0,-2,0,0,0,0,0) \\ \hline = & \mathbf{r}= (20,20,11,5,16,9,14,22,19,23,5,17,1,6,15) \end{array}$$

Da die Zahl der korrigierbaren Fehler nicht überschritten wurde, muß die Decodierung den aufgetretenen Fehlervektor finden und das richtige Codewort ausgeben.
Die Decodierung beginnt mit der Berechnung der Syndromkoeffizienten s_i sowie der Koeffizienten U_i für $i = 1, 3, \ldots, 2t_m - 1$:

$$\begin{aligned} s_1 &= r(\alpha^1) = r(3^1) \\ &= 20 + 20 \cdot 3 + 11 \cdot 9 + 5 \cdot 27 + 16 \cdot 19 + 9 \cdot 26 + 14 \cdot 16 + 22 \cdot 17 \\ &\quad +19 \cdot 20 + 23 \cdot 29 + 5 \cdot 25 + 17 \cdot 13 + 1 \cdot 8 + 6 \cdot 24 + 15 \cdot 10 = 3145 \\ &= 14 \mod 31 \\ s_3 &= r(3^3) = r(27) = 30 \\ s_5 &= r(3^5) = r(26) = 6 \\ s_7 &= r(3^7) = r(17) = 11 \end{aligned}$$

$$\begin{aligned} U_1 &= \frac{-s_1}{1} = -14 = 17 \mod 31 \\ U_3 &= \frac{-s_3 + s_3 U_1^2}{3} = \frac{-30 + (-14)^2 \cdot 14}{-3} = (-30 + 16) \cdot 21 \\ &= -15 = 16 \mod 31 \\ U_5 &= \ldots = 1 \\ U_7 &= \ldots = 0 \end{aligned}$$

Daraus können nach Gleichung 3.33 die Koeffizienten des Polynoms $T(x)$ berechnet werden:

$$\begin{aligned} T_1 &= -U_1 = -17 = 14 \mod 31 \\ T_2 &= -U_3 - T_1 U_1 = -16 - 17 \cdot 14 = 25 \mod 31 \\ T_3 &= -U_5 - T_1 U_3 - T_2 U_1 = \ldots = 1 \\ T_7 &= -U_7 - T_1 U_5 - T_2 U_3 - T_3 U_1 = \cdots = 3 \end{aligned}$$

$$\Longrightarrow \qquad 1 + T(x) \quad = \quad 1 + 14x + 25x^2 + x^3 + 3x^4$$

Der Berlekamp-Algorithmus liefert als Lösung der Schlüsselgleichung 3.36 die Polynome $\Phi(x)$ und $\omega(x)$. An dieser Stelle wird nur das Ergebnis angegeben, die Zwischenergebnisse findet man in [Ber84] in Aufgabe 9.2.

$$\begin{aligned}\Phi(x) &= 1 + 5x + 22x^2 \\ \omega(x) &= 1 + 19x + 24x^2\end{aligned}$$

$$\Longrightarrow \qquad \begin{aligned}\sigma_e(x) &= \omega(x^2) = 1 + 19x^2 + 24x^4 \\ \sigma_o(x) &= \frac{\Phi(x^2) - \omega(x^2)}{x} = 17x + 29x^3\end{aligned}$$

Der Grad des Fehlerstellenpolynoms $\sigma(x) = \sigma_e(x) + \sigma_o(x)$ ist 4, d.h. es sind 4 Fehler vom Betrag 1 aufgetreten und $\sigma(x)$ besitzt 4 Nullstellen. Mit Hilfe von $\sigma_e(x)$ und $\sigma_o(x)$ lassen sich wie auf Seite 48 beschrieben die Fehlerstellenwerte bzw. der Fehlervektor ermitteln. Im Beispiel erhält man:

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
α^{-j}	1	21	7	23	18	6	2	11	14	15	5	12	4	22	28
$a_j = \sigma_e(\alpha^{-j})$	13	8	28	**11**	11	14	27	**5**	18	**15**	7	30	1	5	8
$b_j = \sigma_o(\alpha^{-j})$	15	1	22	**20**	19	11	18	**5**	20	**15**	21	3	2	3	3
1. Durchgang: $e_{1,j}$				$+1$				-1		-1					
$a_j = \sigma'_e(\alpha^{-j})$				20				9		**0**					
$b_j = \sigma'_o(\alpha^{-j})$				5				4		**0**					
2. Durchgang: $e_{2,j}$										-1					
$e_j = e_{1,j} + e_{2,j}$	0	0	0	$\mathbf{+1}$	0	0	0	$\mathbf{-1}$	0	$\mathbf{-2}$	0	0	0	0	0

Da man im 1. Durchgang nur 3 Fehler findet, müssen die Ableitungen $\sigma'_e(x) = 7x + 3x^3$ und $\sigma'_o(x) = 17 + 25x^2$ berechnet und im 2. Durchgang ausgewertet werden. Danach kann abgebrochen werden, da alle 4 Fehler gefunden sind.
Der ermittelte Fehlervektor stimmt wie erwartet mit dem wirklich aufgetretenen Fehler überein, durch Subtraktion erhält man das gesendete Codewort:

$$\mathbf{c} = \mathbf{r} - \mathbf{e}$$

□

3.3 Parameter von NC-Codes

Zur Bezeichnung negazyklischer Codes über dem $GF(p)$ soll folgende Schreibweise verwendet werden:

$$\mathrm{NC}(n, k, d, t_m)$$

Dabei haben die ersten beiden Parameter die gleiche Bedeutung wie bei RS- oder BCH-Codes: n stellt die Länge und k die Dimension des NC-Codes dar. Das heißt n ist die Anzahl der Codesymbole pro Codewort und k die Anzahl der Informationssymbole pro Codewort.

d ist die Lee-Mindestdistanz des NC-Codes und da NC-Codes linear sind, ist d gleichzeitig das kleinste Lee-Gewicht eines Codeworts $\mathbf{c} \neq \mathbf{0}$. d darf nicht verwechselt werden mit der Hamming-Mindestdistanz bei RS- oder BCH-Codes.

Da die Gewichtsverteilungen von NC-Codes in dieser Arbeit nur an Rande untersucht worden sind, wird d in den folgenden Kapiteln oftmals nicht angegeben. Eine Tabelle von Mindestdistanzen findet sich in Anhang A.2. Weitere Informationen dazu findet man in Kapitel 6.

Der vierte Parameter t_m gibt an, wieviele Fehler man mit dem vorliegenden Code bei algebraischer Decodierung nach dem Verfahren von Berlekamp korrigieren kann. Für t_m muß 3.15 erfüllt sein.

Stellt α ein primitives Element aus $GF(p^m)$ dar, und wird die Konstruktion des NC-Codes über den Primkörper $GF(p)$ nach Kapitel 3.1 und die algebraische Decodierung nach Kapitel 3.2 durchgeführt, erhält man folgende Werte für die Parameter des Codes:

$$p \qquad \text{ist Primzahl}$$

$$n = \frac{p^m - 1}{2} \tag{3.44}$$

$$k = n - m \cdot t_m \tag{3.45}$$

$$t_m \leq \frac{p-1}{2} \tag{3.46}$$

Die Parameter einiger NC-Codes sind in Anhang A.1 aufgelistet.

Um einen direkten Vergleich von NC-Codes mit RS- und BCH-Codes zu vereinfachen, werden im folgenden auch für diese Codes vier Parameter angegeben:

$$\mathrm{RS}(n, k, d, t_{max})$$
$$\mathrm{BCH}(n, k, d, t_{max})$$

t_{max} stellt hier die Anzahl der korrigierbaren Hamming-Fehler bei Decodierung bis zur halben Mindestdistanz dar und ist direkt aus d zu berechnen (siehe Gleichung 2.72):

$$t_{max} = \left\lfloor \frac{d-1}{2} \right\rfloor$$

Üblicherweise werden nur n, k und d angegeben.

Kapitel 4

Zu den Simulationen

Ein großer Teil dieser Studienarbeit bestand aus der Implementierung eines Encoders und Decoders für negazyklische Codes als primitive Module für das Simulationsprogramm COSSAP. Eine kurze Dokumentation der Module in Form ihrer Model Definition Files findet man in Anhang C. Eine Beschreibung der gesamten C-Programmbibliothek sowie der COSSAP-Module stellt [Haa96] dar.
Mit den neuen Modulen wurden Simulationen in COSSAP durchgeführt. In diesem Kapitel soll zunächst der Simulationaufbau vorgestellt und auf aufgetretene Probleme hingewiesen werden. Die aufgezeichneten Fehlerraten werden definiert und es werden Gleichungen zur theoretischen Berechnung der Block-Fehlerwahrscheinlichkeiten angegeben. In Kapitel 5 werden dann die Simulationsergebnisse vorgestellt und mit den theoretischen Fehlerkurven sowie mit den Fehlerkurven vergleichbarer RS-Codes verglichen.

4.1 Der Simulationsaufbau für NC-Codes

Für sämtliche Simulationen und Tests mit NC-Codes in COSSAP wurde der Aufbau nach Abbildung 4.1 verwendet. Prinzipiell entspricht er dem Übertragungsmodell aus Bild 2.3. Als Kanal wird ein komplexwertiger AWGN-Kanal verwendet, der zum Sendesignal additiv weißes Gaußsches Rauschen hinzufügt. Beim Modulator handelt es sich um einen MPSK-Modulator, der die Codesymbole $c_j \in GF(p)$ gemäß Abbildung 2.2 (linkes Teilbild) auf $M = p$ Sendesymbole der komplexen Ebene abbildet. Dabei ist p immer eine Primzahl.

Die nichtbinäre Zufallsquelle

Da es sich bei NC-Codes um nichtbinäre Codes handelt, bei denen sowohl Informations- als auch Codesymbole Elemente des $GF(p)$ sind, benötigt man für Simulationen eine Informations-Quelle, die solche Symbole liefert. Dabei sollen aufeinanderfolgende Symbole statistisch unabhängig sein und bei der Wahrscheinlichkeitsverteilung soll es

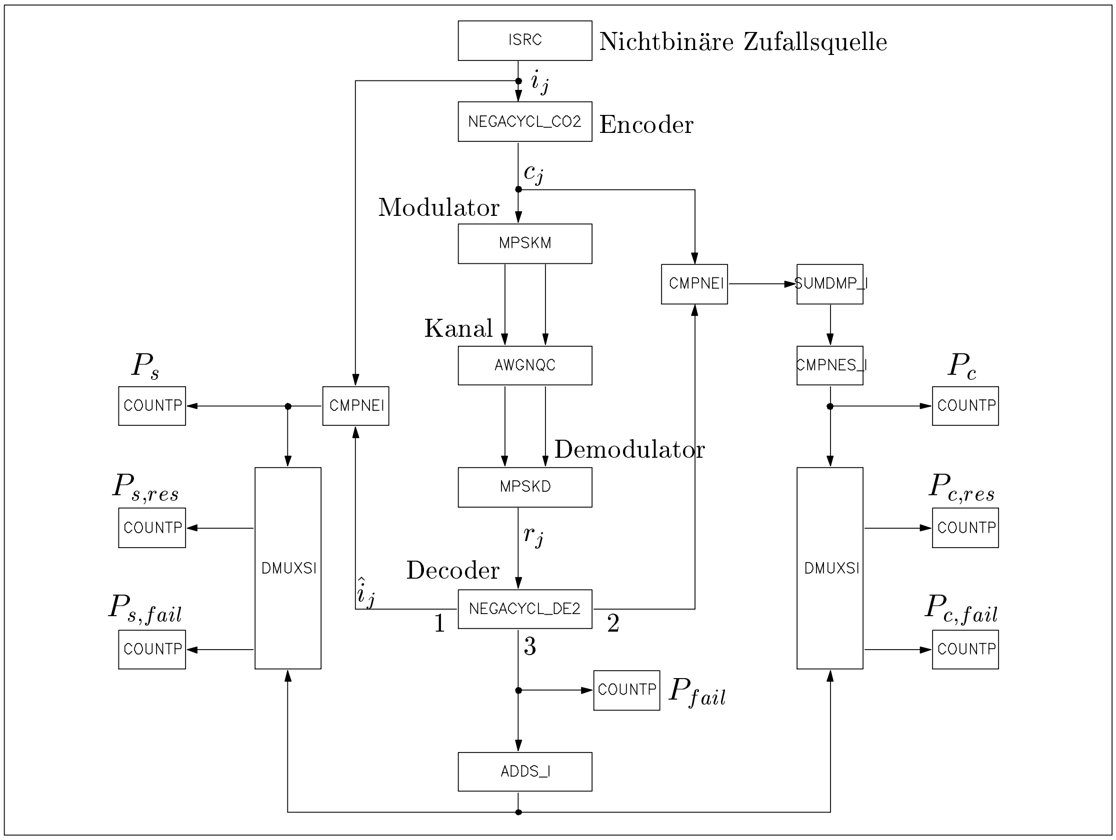

Abbildung 4.1: Simulationsaufbau

sich um eine Normalverteilung handeln. Folgende Möglichkeiten wurden zunächst in Betracht gezogen:

1. Binäre Zufallsquelle:
   Immer $\lfloor log_2(p) \rfloor$ Bit werden zu einer Dualzahl zusammengefaßt, die Dezimaldarstellung dieser Dualzahl wird ausgegeben. Man erhält mit jeweils gleicher Wahrscheinlichkeit $1/(2^q)$ eine Zahl aus der Menge $\{0, 1, \ldots, 2^q - 1\}$ mit $2^q < p$ Elementen; die Zahlen $\{2^q -, \ldots, p-1\}$ kommen nie als Informationssymbole vor.

2. Binäre Quelle mit additiven Sägezahnsignal:
   Um wirklich alle Elemente aus $GF(p)$ zu erhalten, wird zum Zufallssignal, welches die 1. Quelle liefert, ein diskretes Sägezahnsignal mod p aufaddiert. Das heißt es werden nacheinander die Elemente $\ldots, 0, 1, \ldots, p-1, 0, 1, 2, \ldots$ zu aufeinanderfolgenden Zufallszahlen addiert. Dies bewirkt, daß die Menge der Elemente, die nicht auftreten können, von einem zum anderen Zufallssymbol wechselt. Im Mittel kommt so jedes Element aus $GF(p)$ gleich häufig vor.

3. Konstante Quelle:
   Es wird immer die Zahl 0 ausgegeben.

4. Nichtbinäre Zufallsquelle:
   Diese Quelle liefert in zufälliger Reihenfolge Elemente $i_j \in GF(p)$. Die Wahrscheinlichkeiten sind gleichverteilt, d.h. jedes Element kommt im Mittel gleich häufig vor.

Die Quellen wurden in obiger Reihenfolge eingesetzt. Dabei stand zunächst die Idee einer binären Informationsquelle im Vordergrund, wie sie heute im Rahmen digitaler Datenverarbeitung überall zu finden ist. Für den Einsatz nichtbinärer Codes ist es von großer Bedeutung, eine „verlustfreie“ Möglichkeit der Abbildung einer binären Folge auf eine Folge nichtbinärer Symbole zu finden, welche dann codiert werden kann. Die Methoden 1 und 2 genügen diesem Anspruch nicht, da manche Informationssymbole nie vorkommen können, die prinzipiell zulässig wären.

Da das Problem der Abbildung von binären auf nichtbinäre Symbole offenbar nicht trivial und nicht wesentlicher Bestandteil dieser Studienarbeit ist, wurde es hier nicht weiter untersucht. Statt dessen wurde nur noch die Symbolebene betrachtet.

Da es prinzipiell zur Bestimmung von Fehlerhäufigkeiten bei Übertragung mit Kanalcodierung mittels Simulation keine Rolle spielen darf, *welche* Information übertagen wird, wurde mit Methode 3 auch die einfachste mögliche Quelle betrachtet.

Methode 4 erfüllt schließlich als einzige die oben gestellten Anforderungen. Sie wurde wurde mit Hilfe des Zufallsgenerators in C im COSSAP-Modul `ISCR` implementiert.

Nachdem bei den ersten Simulationen mit 1000 Symbolfehler als Abbruchkriterium bei unterschiedliche Quellen etwas unterschiedliche Werte für die gleiche Fehlerrate ermittelt wurden, wurde die Zahl der absolut übertragenen Symbole auf 1 Million erhöht. Diese längeren Simulationen lieferten nahezu unabhängig von der verwendeten Quelle gleiche Ergebnisse. Dies ist die Bestätigung dafür, daß die übertragene Information keine Auswirkung auf die Fehlerrate hat, wenn man die Simulationszeit genügend groß wählt.

Alle Simulationen, deren Ergebnisse hier aufgeführt sind, wurden schließlich mit dem Modul `ISRC` (4. Methode) als Quelle durchgeführt.
Mit dem Abbruchkriterium 1000 Symbolfehler wurde ein Kompromiß zwischen Genauigkeit und Simulationsdauer gewählt.
Für die Codierung wurde die systematische Methode nach Gleichung 2.68 gewählt, da sie wegen dem möglichen Auftreten von Decodierversagen bessere Ergebnisse als eine nicht systematische Codierung liefert.

4.2 Aufgezeichnete Fehlerhäufigkeiten

Im Simulationsaufbau nach Bild 4.1 werden insgesamt 7 verschiedene Fehlerraten aufgezeichnet. Sie sollen nun definiert werden. In Abschnitt 4.3 werden dann die entsprechenden theoretischen Fehlerwahrscheinlichkeiten angegeben.

- **Symbolrestfehlerrate** P_s (symbol error rate)
  Häufigkeit, mit der ein gesendetes Informationssymbol i_j nicht mit dem decodierten Informationssymbol $\hat{i}_j$ übereinstimmt.

- **Symbolfehlerrate bei Decodierversagen** $P_{s,fail}$
  Häufigkeit, mit der ein gesendetes Informationssymbol i_j im Falle von Decodierversagen nicht mit dem decodierten Symbol $\hat{i}_j$ übereinstimmt.

- **Symbolgrundfehlerrate** $P_{s,res}$ (residual symbol error rate)
  Häufigkeit, mit der ein gesendetes Informationssymbol i_j nicht mit dem decodierten Symbol $\hat{i}_j$ übereinstimmt, wenn bekannt ist, daß bei der Decodierung kein Decodierversagen aufgetreten ist.

- **Blockrestfehlerrate** P_c
  Häufigkeit, mit der ein gesendetes Codewort $\mathbf{c}$ nicht mit dem decodierten Codewort $\hat{\mathbf{c}}$ übereinstimmt

- **Blockgrundfehlerrate** $P_{c,res}$
  Häufigkeit, mit der ein gesendetes Codewort $\mathbf{c}$ nicht mit dem decodierten Codewort $\hat{\mathbf{c}}$ übereinstimmt, wenn kein Decodierversagen aufgetreten ist.

- **Blockfehlerrate bei Decodierversagen** $P_{c,fail}$
  Häufigkeit, mit der ein gesendetes Codewort $\mathbf{c}$ im Falle von Decodierversagen nicht dem decodierten Codewort $\hat{\mathbf{c}}$ entspricht. Da der Decoder bei Decodierversagen nie ein gültiges Codewort ausgeben kann, muß man im Falle korrekter Implementierung immer $P_{c,fail} = 1$ erhalten. Dieser Wert dient daher nur zu Prüfzwecken.

- **Decodierversagenshäufigkeit** P_{fail}
  Häufigkeit, mit der Decodierversagen auftritt.

Neben diesen aufgezeichneten Fehlerraten spielen für den Vergleich von codierter zu uncodierter Übertragung auch noch die folgenden Häufigkeiten eine Rolle:

- **Symbolfehlerrate** $P_{s,uncoded}$:
  Häufigkeit, mit der der Demodulator bei uncodierter Übertragung auf ein Symbol entscheidet, das nicht mit dem gesendeten Symbol übereinstimmt

- **Blockfehlerrate** $P_{c,uncoded}$
  Häufigkeit, mit der der Demodulator bei uncodierter Übertragung bei mindestens einem von n aufeinanderfolgenden Empfangssymbolen nicht auf das gesendete Symbol entscheidet.

4.3 Theoretische Fehlerwahrscheinlichkeiten

Als Fehlerwahrscheinlichkeiten bezeichnet man die theoretischen Werte für die Fehlerhäufigkeiten. An dieser Stelle werden sie aus zwei verschiedenen Gründen aufgeführt. Erstens kann man durch Vergleich der theoretischen Fehlerkurve mit den durch Simulation erhaltenen Ergebnissen die korrekte Implementierung der verwendeten Algorithmen prüfen. Zweitens können bestimmte Fehlerkurven wesentlich schneller berechnet als durch Simulation erzeugt werden. Im folgenden wird immer von MPSK-Übertragung über einen komplexen AWGN-Kanal mit der einseitigen Rauschleistungsdichte N_0 ausgegangen (vergleiche Abbildung 4.1) .

4.3.1 Fehlerwahrscheinlichkeiten bei MPSK-Übertragung

Zunächst werden die Gleichungen für Symbol- und Blockfehlerwahrscheinlichkeit bei uncodierter MPSK-Übertragung angegeben, die die Grundlage zur Berechnung bei codierter Übertragung bilden.

Symbolfehlerwahrscheinlichkeit $P_{s,uncoded}$

$P_s, uncoded$ gibt an, mit welcher Wahrscheinlichkeit ein MPSK-Symbol durch das additive Rauschen in ein anderes MPSK-Sendesymbol verfälscht wird. Betrachtet man

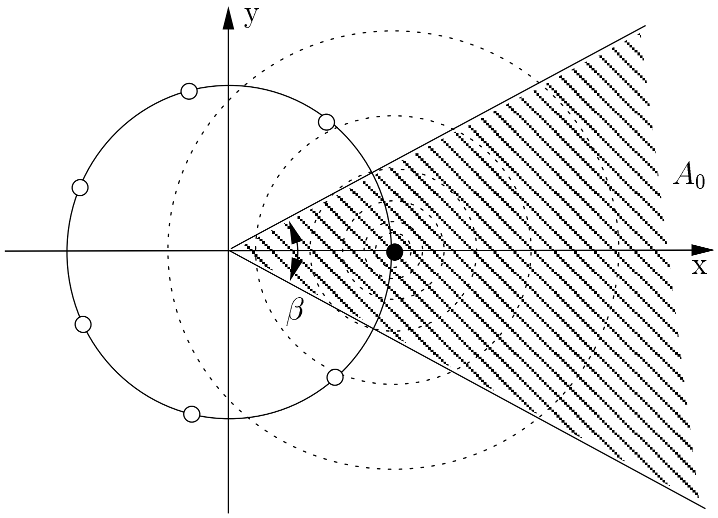

Abbildung 4.2: MPSK, Bereich für korrekte Entscheidung

Abbildung 4.2 und geht davon aus, daß der schwarze Punkt auf der x-Achse das Sendesymbol darstellt, so wird der Demodulator bei allen Empfangsvektoren, die innerhalb des schraffierten Winkelstücks A_0 liegen, auf das richtige Symbol entscheiden. Für alle Empfangsvektoren, die außerhalb liegen, tritt ein Symbolfehler auf. Die punktierten Kreise um das Sendesymbol deuten dabei die Höhenlinien der Wahrscheinlichkeitsdichteverteilung (in Form einer Gaussglocke) an.

In [Pro89] Seite 258ff ist die Demodulation mit Hilfe eines Matched-Filters beschrieben. Das Ergebnis, welches man nach dem Matched-Filter erhält, ist unabhängig von der Form des verwendeten Elementarsignals folgender Vektor im komplexen Signalraum:

$$V = X + jY = 2\alpha E_s + N \tag{4.1}$$

Für den betrachteten AWGN-Kanal mit der (einseitigen) Rauschleistungsdichte N_0 ist die Signal-Verstärkung $\alpha = 1$. E_s stellt die Signalenergie des Elementarsignals dar und N ist eine komplexwertige gaußsche Zufallsvariable mit Mittelwert Null und der Varianz $\sigma = 2\alpha E_s N_0$. Normiert man die Sendeenergie E_s auf den Wert 1, so entspricht der Vektor V bei rauschfreier Übertragung ($N = 0$) dem Sendesymbol auf dem Einheitskreis in Bild 4.2.

Für $N \neq 0$ sind die beiden Komponenten

$$X = 2E_s + \mathrm{Re}(N) \tag{4.2}$$
$$Y = \mathrm{Im}(N) \tag{4.3}$$

statistisch unabhängige gaußverteilte Zufallsvallsvariablen. Ihre gemeinsame Wahrscheinlichkeitsdichte lautet:

$$p(x,y) = p(x)p(y) = \frac{1}{2\pi\sigma^2} e^{-\left((x-2\alpha E_s)^2+y^2\right)/2\sigma^2} \tag{4.4}$$

Aufgrund der Symmetrie der MPSK-Modulation ist eine Darstellung in Polarkoordinaten günstiger:

$$r = \sqrt{x^2+y^2} \tag{4.5}$$
$$\varphi = \tan^{-1}\left(\frac{y}{x}\right) \tag{4.6}$$

$$p(r,\varphi) = r \cdot \frac{1}{2\pi\sigma^2} e^{-\left(r^2+4\alpha^2 E_s^2-4\alpha E_s r\cos\varphi\right)/2\sigma^2} \tag{4.7}$$

Und da die Entscheidungsgrenzen zwischen zwei Sendesymbolen radial verlaufen (siehe Bild 4.3, Seite 60), kann man die Abhängigkeit in r durch Integration beseitigen. Nach längerer Rechnung erhält man die winkelabhängige Wahrscheinlichkeitsdichte

$$\begin{aligned} p(\varphi) &= \int_{r=0}^{\infty} p(r,\varphi)\,dr \\ &= \frac{1}{2\pi} e^{-\gamma} \left(1 + \sqrt{4\pi\gamma}\cos(\varphi) e^{\gamma\cos^2(\varphi)} \left(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\sqrt{2\gamma}\cos(\varphi)} e^{-z^2/2}\,dz\right)\right) \end{aligned} \tag{4.8}$$

bei dem Signal- zu Rauschleistungsverhältnis

$$\gamma = \frac{E_s}{N_0}. \tag{4.9}$$

Mit Hilfe der komplementären Fehlerfunktion (complementary error function)

$$\begin{aligned} \mathrm{erfc}(x) &= \frac{2}{\sqrt{\pi}} \int_t^{\infty} e^{-t^2}\,dt \\ &= \sqrt{\frac{2}{\pi}} \int_{\sqrt{2}z}^{\infty} e^{-\frac{z^2}{2}}\,dz \end{aligned} \tag{4.10}$$

kann man die winkelabhängige Wahrscheinlichkeitsdichte wie folgt angeben:

$$p(\varphi) \;=\; \frac{1}{2\pi}e^{-\gamma}\left(1+\sqrt{4\pi\gamma}\cos\varphi e^{\gamma\cos^2(\varphi)}\left(\frac{1}{2}\operatorname{erfc}\left(\sqrt{\gamma}\cos\varphi\right)\right)\right) \tag{4.11}$$

Die Symbolfehlerwahrscheinlichkeit läßt sich dann berechnen als die Wahrscheinlichkeit, mit der das Empfangssymbol bzw. der Vektor V nicht innerhalb des Winkelstücks A_0 liegt:

$$\begin{aligned}
P_{s,uncoded} &= 1-\iint_{(A_0)} p(x,y)\,dx\,dy \\
&= 1-\int_{-\frac{\pi}{M}}^{\frac{\pi}{M}} p(\varphi)\,d\varphi \\
&= 1-2\int_0^{\frac{\pi}{M}} p(\varphi)\,d\varphi
\end{aligned} \tag{4.12}$$

Eine Auswertung dieses Integrals ist aufgrund des uneigentlichen Integrals in $p(\varphi)$ nur numerisch möglich.

Blockfehlerwahrscheinlichkeit $P_{c,uncoded}$

Bei bekannter Symbolfehlerwahrscheinlichkeit erhält man die Blockfehlerwahrscheinlichkeit bei uncodierter Übertragung nach folgender Überlegung: Ein Block von n Symbolen ist fehlerhaft, wenn nicht alle n Symbole auf der Empfangsseite richtig erkannt werden. Die Wahrscheinlichkeit für die korrekte Übertragung eines Symbols ist $(1-P_{s,uncoded})$ und man erhält:

$$\begin{aligned}
P_{c,uncoded} &= \sum_{i=1}^{n}\binom{n}{i}P_{s,uncoded}^{i}\left(1-P_{s,uncoded}\right)^{n-i} \\
&= 1-\left(1-P_{s,uncoded}\right)^{n}
\end{aligned} \tag{4.13}$$

4.3.2 Die Blockrestfehlerwahrscheinlichkeit P_c bei RS-und BCH-Codes

Bei codierter Übertragung mit RS- oder BCH-Codes tritt ein Blockfehler auf, wenn die Zahl der aufgetretenen Hamming-Fehler die Zahl t_m der korrigierbaren Fehler überschreitet. Die Blockrestfehlerwahrscheinlichkeit ergibt sich mit Hilfe des Binomischen Satzes zu:

$$\begin{aligned}
P_c &= \sum_{i=t_m+1}^{n}\binom{n}{i}P_{s,uncoded}^{i}\left(1-P_{s,uncoded}\right)^{n-i} \\
&= 1-\sum_{i=0}^{t_m}\binom{n}{i}P_{s,uncoded}^{i}\left(1-P_{s,uncoded}\right)^{n-i}
\end{aligned} \tag{4.14}$$

4.3.3 Die Blockrestfehlerwahrscheinlichkeit P_c bei NC-Codes

Für NC-Codes ist die Berechnung der Blockrestfehlerwahrscheinlichkeit komplizierter als für Codes, bei denen die Hamming-Metrik verwendet wird, da aufgrund der Lee-Metrik bereits bei der Übertragung eines einzelnen Symbols mehrere Fehler auftreten können. Betrachtet man Abbildung 4.3 und geht davon aus, daß der schwarze Punkt das Sendesymbol darstellt, dann genügt es nicht wie in Abbildung 4.2 zwei Winkelbereiche zu unterscheiden, sondern man muß alle Wahrscheinlichkeiten P_j für das Auftreten von j Fehlern in einem Symbol berechnen. P_j gibt an, mit welcher Wahrscheinlichkeit das empfangene Symbol bzw. der Vektor im Signalraum im Winkelbereich A_{+j} oder A_{-j} liegt.

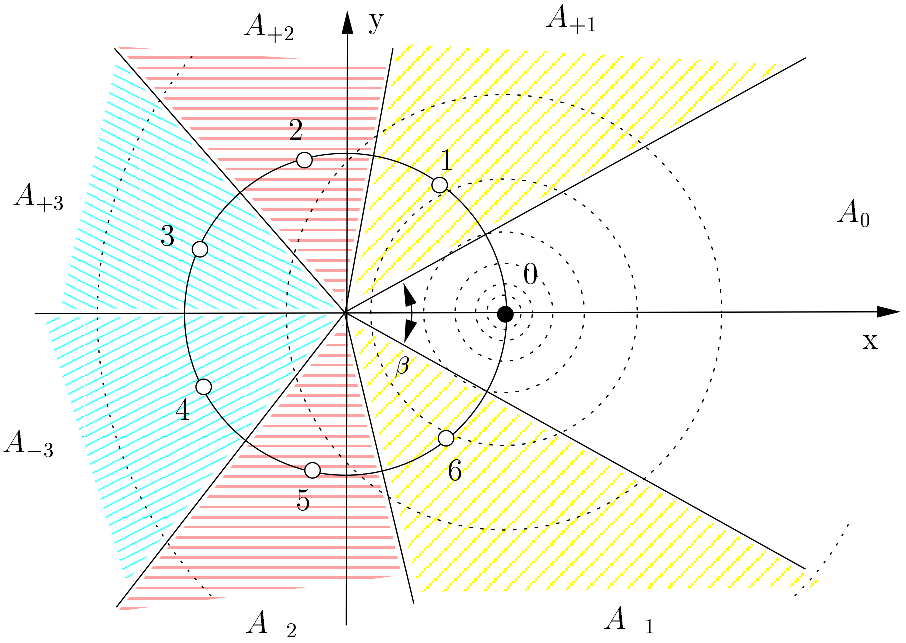

Abbildung 4.3: MPSK, Winkelbereiche für die Berechnung von P_c bei NC-Codes

Für MPSK-Übertragung mit einem Code mit Symbolen aus $GF(p)$ gilt:

$$M = p \tag{4.15}$$

$$\delta = \left\lfloor \frac{M-1}{2} \right\rfloor \tag{4.16}$$

Dabei gibt δ an, wieviele Fehler maximal in einem Symbol auftreten können.

Die Wahrscheinlichkeiten P_j ergeben sich analog zu Gleichung 4.12:

$$\begin{aligned}
P_0 &= 1 - P_{s,uncoded} = \iint_{(A_0)} p(x,y)\,dx\,dy \\
&= \int_{-\frac{\pi}{M}}^{\frac{\pi}{M}} p(\varphi)\,d\varphi \\
P_1 &= \iint_{(A_{-1})} p(x,y)\,dx\,dy + \iint_{(A_{+1})} p(x,y)\,dx\;dy \\
&= 2\int_{\frac{\pi}{M}}^{3\frac{\pi}{M}} p(\varphi)\,d\varphi \\
&\vdots \\
P_j &= \iint_{(A_{-j})} p(x,y)\,dx\,dy + \iint_{(A_{+j})} p(x,y)\,dx\,dy \\
&= 2\int_{(2j-1)\frac{\pi}{M}}^{(2j+1)\frac{\pi}{M}} p(\varphi)\,d\varphi \qquad (j \leq \delta)
\end{aligned} \tag{4.17}$$

Die Koeffizienten k_i der sogenannten *Lee Composition* $\mathbf{k} = (k_0, k_1, k_2, \ldots, k_\delta)$ geben an, wieviele Symbole von Lee-Gewicht $0, 1, 2, \ldots, \delta$ ein Fehlervektor $\mathbf{e}$ enthält, d.h. bei wievielen Symbolen eines Codeworts bei der Übertragung $0, 1, 2, \ldots, \delta$ Fehler aufgetreten sind (siehe auch Kapitel 6). Der folgende *Polynomialkoeffizient* (siehe [BSG91], Seite 105f) gibt dann die Anzahl möglicher Fehlervektoren $\mathbf{e}$ an, die die gleiche Lee Composition $\mathbf{k}$ besitzen:

$$\binom{n}{k_0, k_1, \ldots, k_\delta} = \frac{n!}{k_0!k_1!\cdots k_\delta!}, \qquad \sum_{i=0}^{\delta} k_i = n \tag{4.18}$$

k_0 gibt die Zahl der unverfälscht übertragenen Symbole an und es gilt:

$$k_0 = n - \sum_{j=1}^{\delta} k_j \tag{4.19}$$

Um die Wahrscheinlichkeit $P_{(w_L=t)}$ zu berechnen, mit der ein Fehlervektor vom Lee-Gewicht t auftritt muß man alle Fehlervektoren berücksichtigen, für deren Lee Composition $(\sum_{i=1}^{\delta} ik_i) = t$ gilt. Mit dem Polynomischen Satz ([BSG91], Seite 106) erhält man dann:

$$P_{(w_L=t)} = \sum_{(\sum\limits_{i=1}^{\delta} ik_i)=t} \binom{n}{k_0, k_1, \ldots, k_\delta} \prod_{j=0}^{\delta} P_j^{k_j} \tag{4.20}$$

Kann ein NC-Code nun bis zu t_m Lee-Fehler korrigieren, so tritt ein Blockfehler auf, wenn das Lee-Gewicht des Fehlers größer als t_m ist. Die Blockrestfehlerwahrscheinlichkeit ergibt sich demnach zu:

$$\begin{aligned} P_c &= 1 - \sum_{t=0}^{t_m} P_{(w_L=t)} \\ &= 1 - \sum_{t=0}^{t_m} \sum_{(\sum\limits_{i=1}^{\delta} ik_i)=t} \binom{n}{k_0, k_1, \ldots, k_\delta} \prod_{j=0}^{\delta} P_j^{k_j} \qquad (4.21) \\ P_c &= 1 - \sum_{t=0}^{t_m} \sum_{(\sum\limits_{i=1}^{\delta} ik_i)=t} \frac{n!}{(n-k_1-\cdots-k_\delta)!k_1!\cdots k_\delta!} P_0^{(n-k_1-\cdots-k_\delta)} P_1^{k_1} \cdots P_\delta^{k_\delta} \end{aligned}$$

Kapitel 5

Simulationsergebnisse

In diesem Kapitel werden die Simulationsergebnisse vorgestellt, die mit dem Simulationsaufbau aus Kapitel 4.1 erzielt wurden.

Aus der Vielfalt möglicher NC-Codes wurden für die Simulationen Codes für 17-PSK- und 31-PSK Modulation ausgewählt. Dies hat folgende Gründe:

- 17 und 31 sind Primzahlen die als Nachbarn Zweierpotenzen (16, 32) besitzen, so daß ein Vergleich mit RS-Codes über dem $GF(2^m)$ vereinfacht wird, da ähnliche Symbolalphabete vorliegen.

- Beide verwendeten Primzahlen sind genügend groß um relativ lange Codes mit nicht allzugroßen Werten für m in $GF(p^m)$ zu erhalten.

Für die gewählten Symbolalphabete wurde jeweils eine ganze Schar von Codes bzw. Fehlerkurven ermittelt, um insbesondere bei kurzen Codes herauszufinden, ab wann eine Erhöhung der Korrekturfähigkeit aufgrund der damit verbundenen Verringerung der Coderate R keinen weiteren Gewinn mehr liefert.
Die Kurvenscharen für die Blockrestfehlerrate P_c und die Symbolrestfehlerrate P_s sind in Abschnitt 5.1 abgebildet. Anhand von P_c wird in den gleichen Abbildungen ein Vergleich mit den theoretisch berechneten Kurven durchgeführt.
In Abschnitt 5.2 wird ein Vergleich der simulierten NC-Codes mit RS-Codes über $GF(2^m)$ durchgeführt. Noch aussagekräftiger ist der Vergleich zwischen NC-Codes und verkürzten RS-Codes über $GF(p)$ in Abschnitt 5.3.

Obwohl in erster Linie die Ergebnisse für P_c und P_s ausgewertet wurden, soll in Abbildung 5.1 kurz ein Überblick über die Gesamtheit der aufgezeichneten und in Kapitel 4.2 beschriebenen Fehlerraten gegeben werden.

Wie erwartet gilt $P_{c,fail} = 1$, d.h. bei erkannten Decodierversagen liegt immer ein Blockfehler vor. Die Erkennung von Decodierversagen ist offenbar korrekt implementiert. Es wird zumindest nicht fälschlicherweise auf Decodierversagen entschieden.

Die Kurven der Grundfehlerraten $P_{c,res}$ und $P_{s,res}$ sind durchgehend besser als die der Restfehlerraten P_c und P_s. Auch dies entspricht der Theorie.

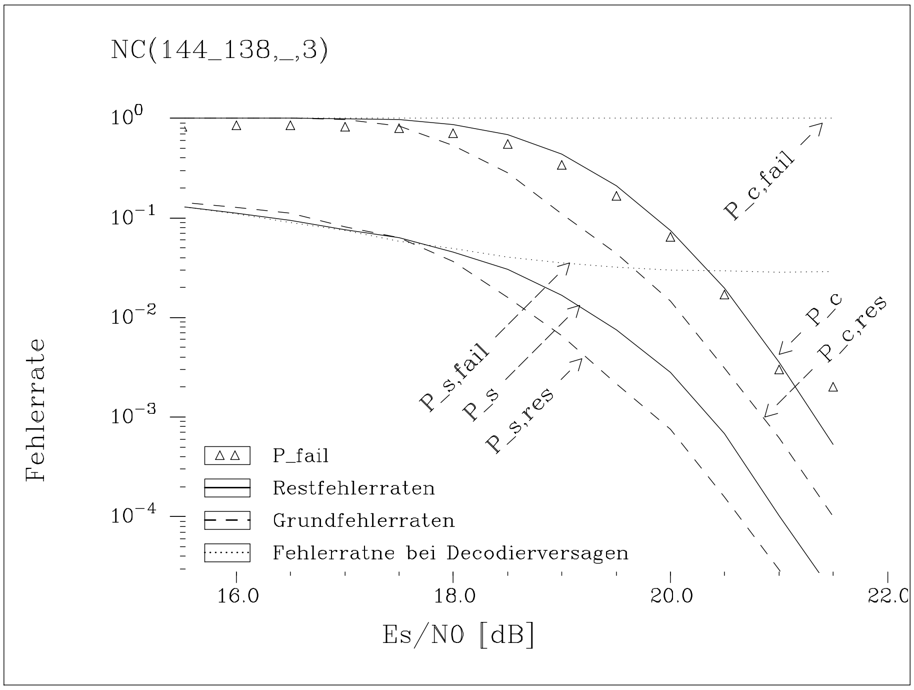

Abbildung 5.1: Aufgezeichnete Fehlerraten am Beispiel NC(144,138,d,3) mit 17PSK

5.1 Restfehlerraten P_c und P_s einiger NC-Codes

Für die folgenden Abbildungen gilt:

Im linken Bild werden Blockrestfehlerraten angegeben. Dabei wurden die durchgezogenen Kurven nach Gleichung 4.21 mit Hilfe von MATLAB numerisch berechnet. Die Punkte stellen die Simulationsergebnisse dar. Die Übereinstimmung von Simulation und Theorie ist dabei sehr gut, die implementierten Module funktionieren fehlerfrei.
Im rechten Bild sind jeweils die simulierten Symbolfehlerraten der entsprechenden NC-Codes abgebildet.
In beiden Bildern ist gestrichelt auch die Fehlerrate bei uncodierter Übertragung angegeben. Bei der Blockrestfehlerrate muß dabei immer die jeweilige Codelänge berücksichtigt werden.

Für die beiden kurzen Codes $NC(8, 8-t, d, t)$ und $NC(15, 15-t, d, t)$ sind die Kurvenscharen jeweils auf zwei Bilder aufgeteilt, um folgenden Effekt besser darstellen zu können: Mit zunehmender Fehlerkorrekturfähigkeit, d.h. wachsendem Wert t bzw. t_m erhält man zunächst bessere Fehlerraten. Erhöht man die Zahl der korrigierbaren Fehler t immer weiter, so überwiegt schließlich der Effekt der stark abnehmenden Rate $R = k/n$ und die Fehlerraten werden bei gleichem E_s/N_0 wieder schlechter.

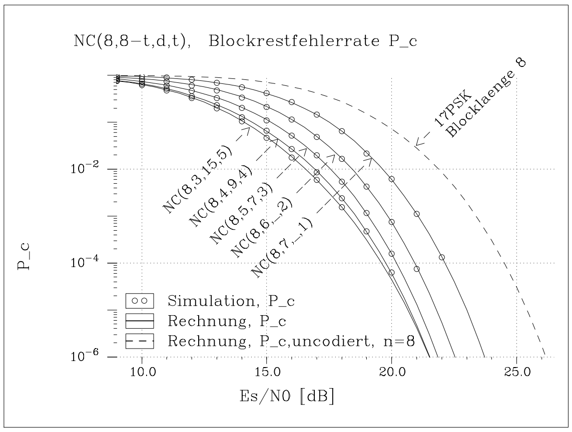

Abbildung 5.2: Blockrestfehlerrate P_c für NC$(8, 8-t, d, t)$ und 17-PSK (Teil 1)

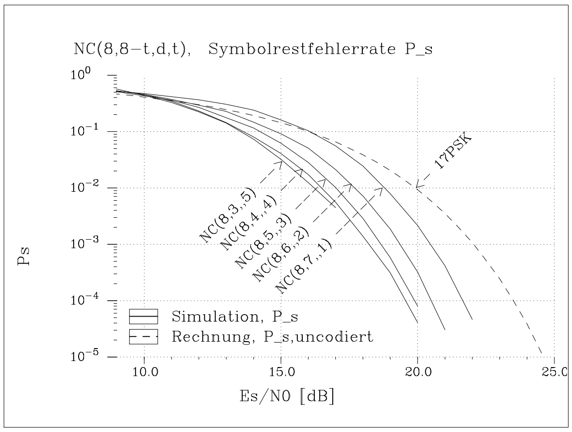

Abbildung 5.3: Symbolrestfehlerrate P_s für NC$(8, 8-t, d, t)$ und 17-PSK (Teil 1)

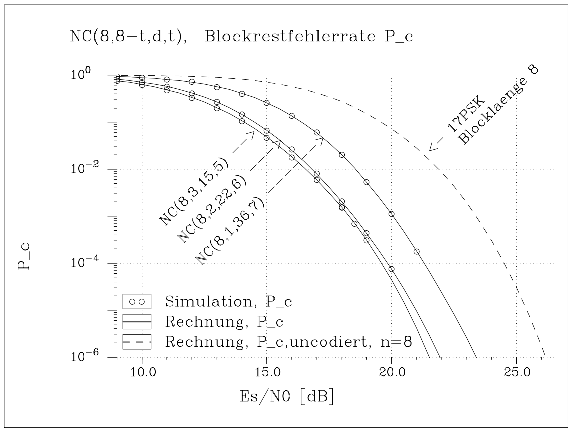

Abbildung 5.4: Blockrestfehlerrate P_c für NC$(8, 8-t, d, t)$ und 17-PSK (Teil 2)

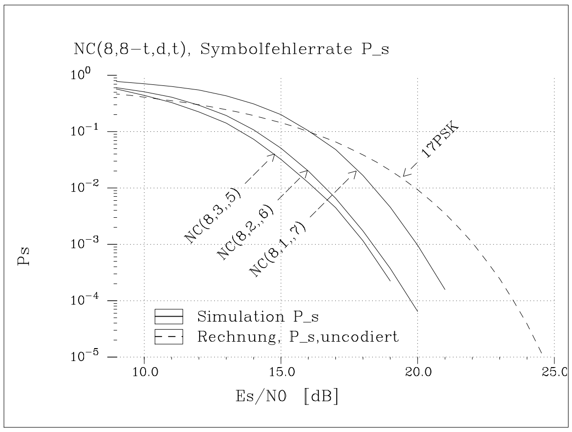

Abbildung 5.5: Symbolrestfehlerrate P_s für NC$(8, 8-t, d, t)$ und 17-PSK (Teil 2)

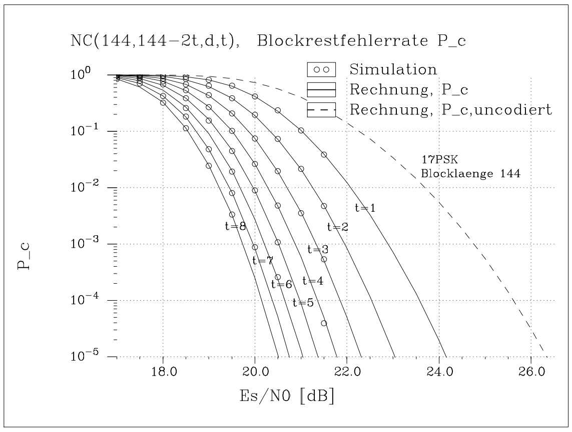

Abbildung 5.6: Blockrestfehlerrate P_c für NC$(144, 144-2t, d, t)$ und 17-PSK

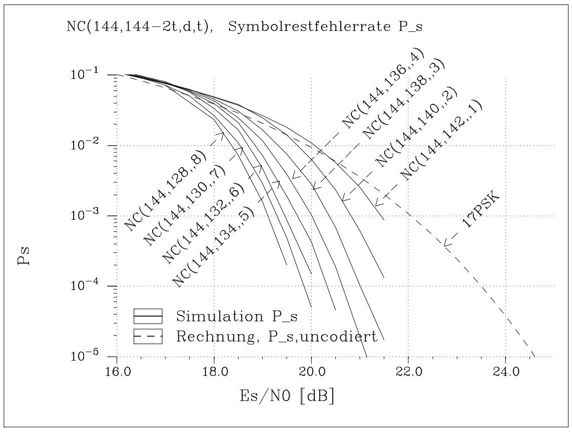

Abbildung 5.7: Symbolrestfehlerrate P_s für NC$(144, 144-2t, d, t)$ und 17-PSK

Wie man in den Abbildungen 5.6 und 5.7 sieht tritt dieser Effekt bei langen Codes nicht auf. Hier nimmt die Coderate R auch für maximale Korrekturfähigkeit nicht so stark ab, daß dieser Effekt stärker als der Codierungsgewinn wäre. Bei langen Codes ergeben sich wirklich für denjenigen Code die besten Fehlerraten, der die meisten Fehler korrigieren kann.

Im folgenden sind die simulierten Restfehlerraten der Codes NC$(15, 15-t, d, t)$ aufgetragen. Da es sich um relativ kurze Codes handelt, tritt auch bei ihnen der oben beschriebene Effekt auf.
Die theoretischen Blockrestfehlerwahrscheinlichkeiten wurden in diesem Fall nicht berechnet, da die Berechnung aufgrund des großen Symbolalphabets $GF(31)$ wesentlich länger dauert als bei den zuvor betrachteten NC-Codes.

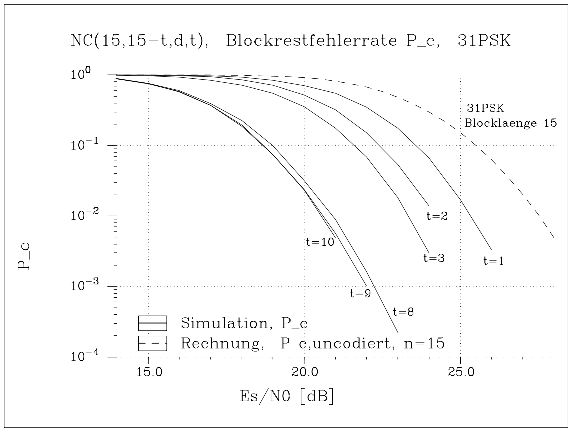

Abbildung 5.8: Blockrestfehlerrate P_c für NC$(15, 15-t, d, t)$, 31-PSK (Teil 1)

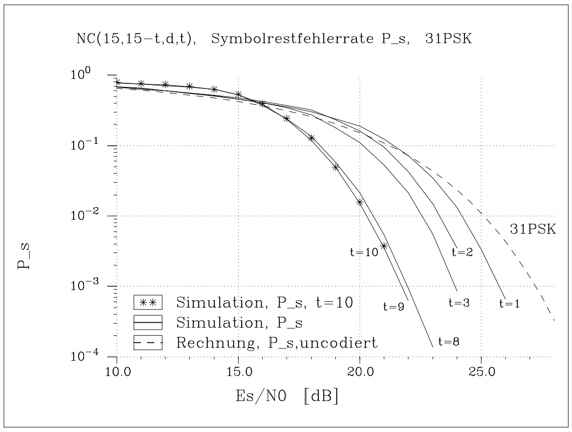

Abbildung 5.9: Symbolrestfehlerrate P_s für NC$(15, 15-t, d, t)$, 31-PSK (Teil 1)

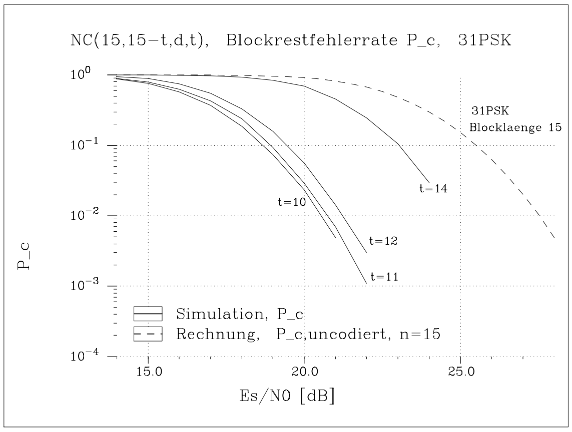

Abbildung 5.10: Blockrestfehlerrate P_c für NC$(15, 15-t, d, t)$, 31-PSK (Teil 2)

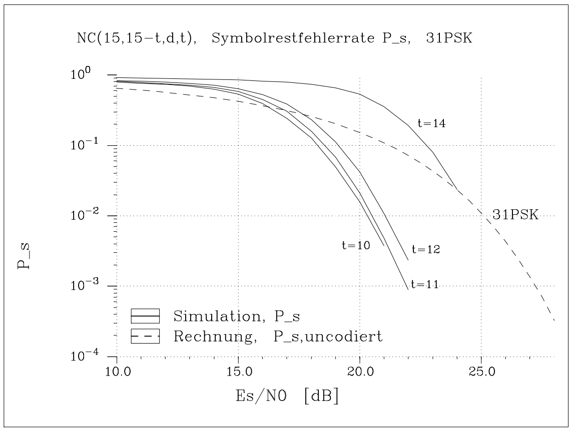

Abbildung 5.11: Symbolrestfehlerrate P_s für NC$(15, 15-t, d, t)$, 31-PSK (Teil 2)

5.2 Vergleich mit RS-Codes über $GF(2^m)$

Zur Beurteilung des Codierungsgewinns von NC-Codes wurden zunächst RS-Codes über dem Erweiterungskörper $GF(2^m)$ zum Vergleich herangezogen, da für diese bereits eine Implementierung in COSSAP vorlag (siehe [Kre94]).

Bezüglich des Symbolalphabets ist eine relativ gute Übereinstimmung möglich, wenn man als Primzahl p des NC-Codes den Nachbarn einer Zweierpotenz wählt. Der Unterschied der Fehlerraten bei uncodierter MPSK Übertragung ist dann relativ gering. An dieser Stelle werden die simulierten NC-Codes über $GF(17)$ mit RS-Codes über $GF(2^4)$ verglichen. Die Modulationsverfahren sind also 17-PSK bzw. 16-PSK.

Die Codeparameter sind weniger gut vergleichbar als die Symbolalphabete. Die Länge eines RS-Codes ist $n_{RS} = p^m - 1$, im vorliegenden Fall also $n_{RS} = 15$. Die Länge eines NC-Codes beträgt dagegen $n_{NC} = (p^m - 1)/2$, im Beispiel also $n_{NC} = 8, 144, \ldots$.

Für den Vergleich wurden schließlich Codes mit möglichst ähnlicher Coderate $R = k/n$ herangezogen. Die unterschiedlichen Codelängen dürfen jedoch nicht ganz vernachlässigt werden, da lange Codes aus statistischen Gründen immer steiler abfallende Fehlerkurven besitzen als kürzere Codes mit gleicher Rate.

Die folgenden Codes werden jeweils anhand der berechneten Blockrestfehlerwahrscheinlichkeiten und der simulierten Symbolfehlerwahrscheinlichkeiten verglichen:

NC-Code, 17-PSK	R_{NC}	R_{RS}	RS-Code, 16-PSK	siehe Abbildung
$NC(8,4,9,4)$	0.5	0.47	$RS(15,7,9,4)$	5.12, 5.13
$NC(8,6,,2)$	0.75	0.73	$RS(15,11,5,2)$	5.12, 5.13
$NC(8,7,,1)$	0.88	0.87	$RS(15,13,3,1)$	5.12, 5.13
$NC(144,128,\_,8)$	0.89			5.14, 5.15
$NC(144,130,\_,7)$	0.90			5.14, 5.15
$NC(144,140,\ ,2)$	0.97			5.14, 5.15

Wie man in den Abbildungen 5.12 und 5.13 sieht, erzielt man für $R = 0.9$ mit den Codes $NC(8, 8-t, d, t)$ trotz ihrer geringeren Länge vergleichbare Ergebnisse wie mit den RS(15,15-t,d,t)-Codes. Für geringere Coderaten erzielt man sogar einen zusätzlichen Gewinn von etwa 1dB bei einer Blockrestfehlerrate von 10^{-6}.
Dieser Gewinn ist in erster Linie darauf zurückzuführen, daß die Lee-Metrik, die bei NC-Codes verwendet wird, besser für die verwendete MPSK-Modulation geeignet ist als die Hamming-Metrik der RS-Codes, da sie die unterschiedlichen Intersymbolfehlerwahrscheinlichkeiten berücksichtigt.

Der Vergleich der Codes NC(144,144-2t,d,t) mit den RS-Codes ist schwieriger, da sie wesentlich länger sind und der Abfall ihrer Restfehlerkurven wesentlich steiler ist. So erzielen die betrachteten NC- und RS-Code mit Rate $R = 0.88$ bei etwa 18.5dB gleiche Werte für P_c. Für $P_c = 10^{-6}$ erhält man hier mit den NC-Codes jedoch einen Gewinn von etwa 2.5dB. In diesem Bereich ist der Code $NC(144, 140, \_, 2)$ mit Rate 0.97 vergleichbar mit dem Code $RS(15, 13, 3, 1)$ der Rate 0.87.
Hier ist es schwierig zu entscheiden, ob der Gewinn durch die Lee-Metrik oder die größere Länge des NC-Codes erzielt wird.

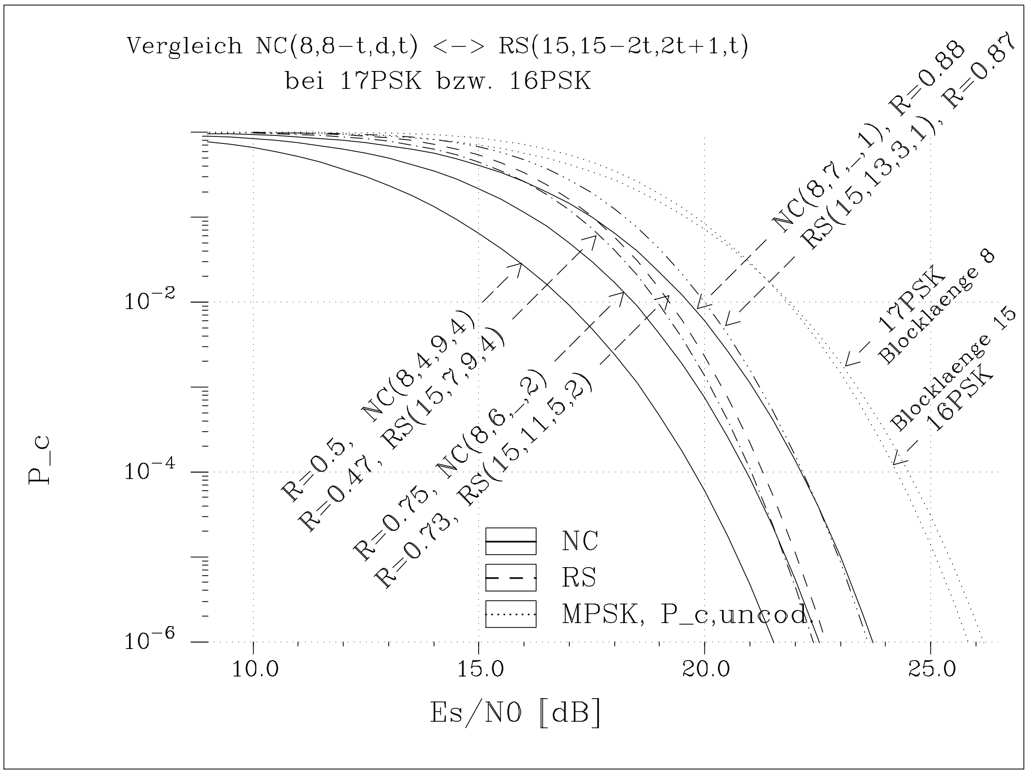

Abbildung 5.12: Vergleich zwischen NC(8,8-t,d,t) und RS-Codes, P_c

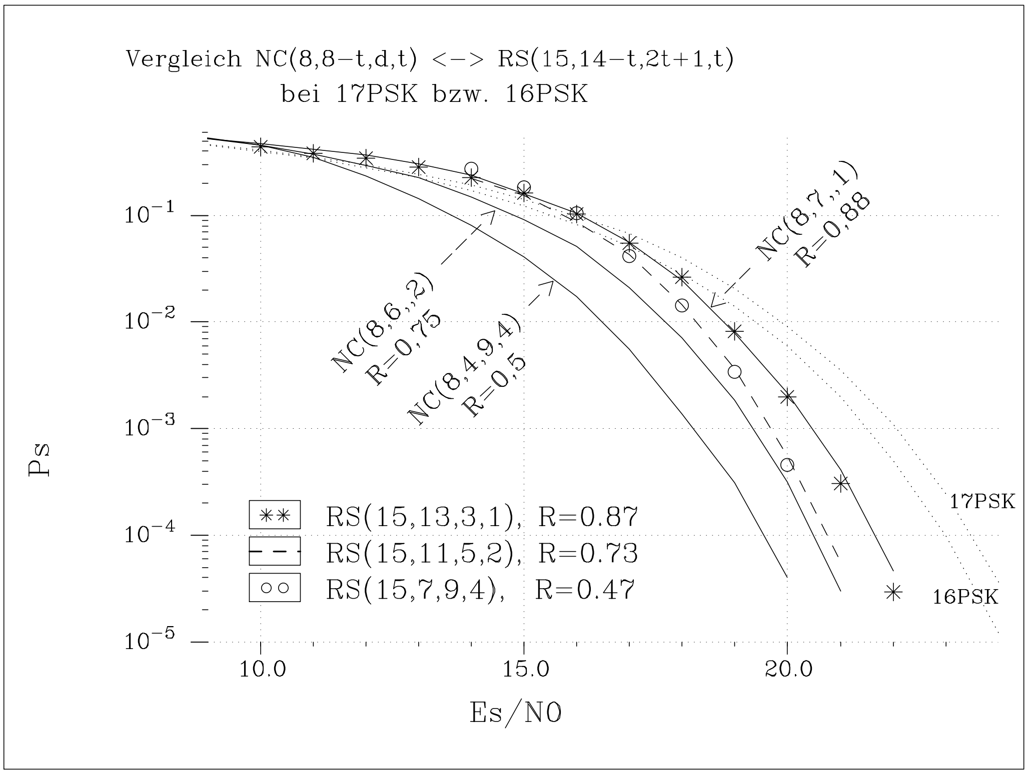

Abbildung 5.13: Vergleich zwischen NC(8,8-t,d,t) und RS-Codes, P_s

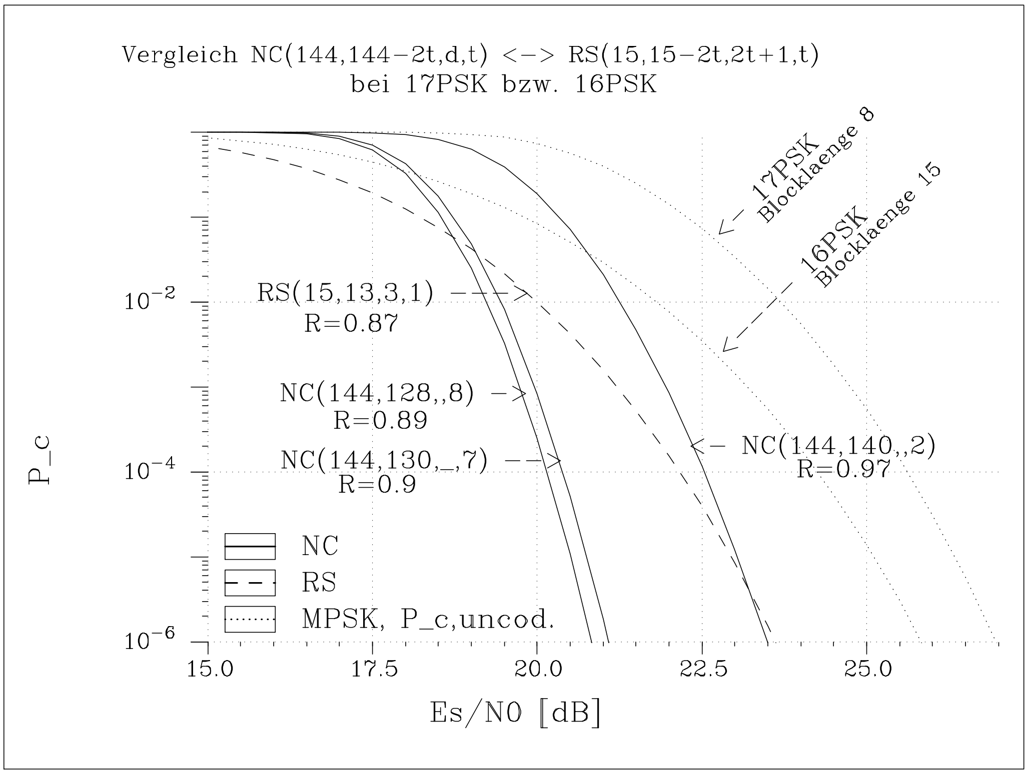

Abbildung 5.14: Vergleich zwischen NC(144,144-2t,d,t) und RS-Codes, P_c

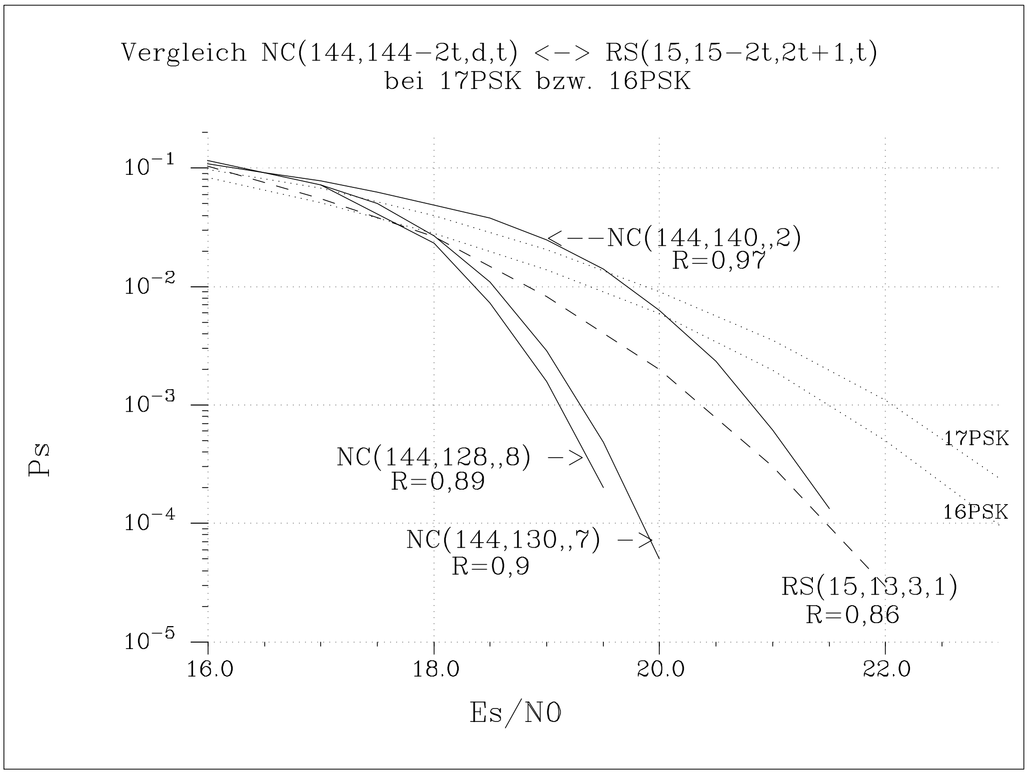

Abbildung 5.15: Vergleich zwischen NC(144,144-2t,d,t) und RS-Codes, P_s

5.3 Vergleich mit verkürzten RS-Codes über $GF(p)$

RS-Codes besitzen im allgemeinen Fall das Symbolalphabet $GF(p^m)$, NC-Codes sind dagegen Codes über $GF(p)$. Ein Vergleich bei identischem Symbolalphabet ist also nur für $m = 1$ möglich. Für diesen Fall sind RS-Codes identisch zu nichtbinären BCH-Codes und besitzen das Symbolalphabet $GF(p)$.
Um gleiche Codelängen zu erhalten, ist es nötig, die RS-Codes zu verkürzen. Man verringert hierzu die Länge n und die Dimension k um den gleichen Wert. Die Mindestdistanz eines RS-Codes bleibt dabei unverändert, die Coderate R wird jedoch kleiner.

Folgende Codes werden in Verbindung mit 17-PSK anhand der berechneten Blockrestfehlerwahrscheinlichkeiten in Abbildung 5.16 verglichen:

NC-Code	R	verkürzter RS-Code	RS-Code
NC(8, 7,, 1)	0.875		
NC(8, 6,, 2)	0.75	RS(8, 6, 3, 1)	RS(16, 14, 3, 1)
NC(8, 5, 7, 3)	0.63		
NC(8, 4, 9, 4)	0.5	RS(8, 4, 5, 2)	RS(16, 12, 5, 2)
NC(8, 3, 15, 5)	0.375		
	0.25	RS(8, 2, 7, 3)	RS(16, 10, 7, 3)

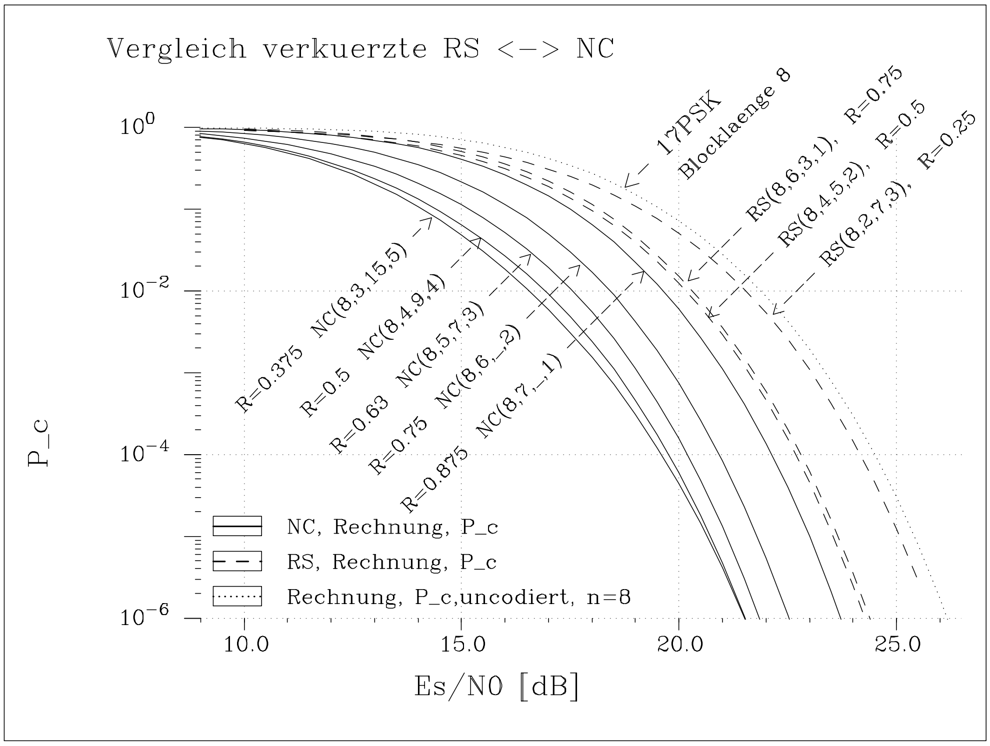

Abbildung 5.16: Vergleich von NC-Codes mit verkürzten RS-Codes über $GF(17)$

Selbst der beste der verwendeten RS-Codes RS$(8, 4, 5, 2)$ ist noch schlechter als der schlechteste, wesentlich höherratige NC-Code NC$(8, 7, _, 1)$.

Der vergleichbare Code NC(8,4,9,4) mit der Rate $R = 0.5$ erzielt gegenüber dem verkürzten RS-Code sogar einen zusätzlichen Gewinn von fast 3dB bei $P_c = 10^{-6}$.

Der Vorteil der Lee-Metrik gegenüber der Hamming-Metrik bei MPSK-Modulation wird hier besonders gut deutlich.

5.4 Vergleich mit nichtbinären BCH-Codes

Für NC-Codes der Länge $(p^m - 1)/2$ mit $m > 1$ ist ein Vergleich mit RS-Codes nicht gut möglich, da RS-Codes für $m > 1$ das Symbolalphabet $GF(p^m)$ und nicht $GF(p)$ besitzen. Für solche längeren NC-Codes wäre ein Vergleich mit verkürzten nichtbinären BCH-Codes am sinnvollsten, da diese ebenso wie NC-Codes immer nur Codesymbole aus $GF(p)$ besitzen.
Die Berechnung der Blockrestfehlerrate für solche BCH-Codes kann bei bekannten Parametern wie für RS-Codes nach Gleichung 4.14 erfolgen. Da nichtbinäre BCH-Codes jedoch ähnlich wie NC-Codes wenig verbreitet sind und ihre Konstruktion für $m > 1$ nicht trivial ist, soll an dieser Stelle nur auf diese Vergleichsmöglichkeit hingewiesen werden. Berechnungen hierzu wurden nicht durchgeführt.

5.5 Beispiel für Fehlerkorrektur und Decodierversagen

Zum Abschluß dieses Kapitels über Simulationsergebnisse sollen zwei Abbildungen den Unterschied der Lee-Metrik zur Hamming-Metrik noch einmal anschaulich verdeutlichen. Desweiteren sollen sie veranschaulichen, wann bei einem NC-Code Decodierversagen auftritt.

In Abbildung 5.17 sind für einen kurzen Ausschnitt Sendesymbole, Empfangssymbole sowie die Symbole nach der Decodierung aufgetragen.
Beim 49. Symbol erkennt man deutlich, wie in einem einzelnen Symbol 2 Lee-Fehler auftreten können. Bezüglich der Hamming-Metrik ist diese Verfälschung des Symbols um den Betrag 2 nur ein Hamming-Fehler.
Die Symbole 56 bis 63 stellen ein Codewort der Länge 8 dar. Da insgesamt 9 Lee-Fehler in diesem Codewort aufgetreten sind und der Code NC(8,3,15,5) maximal 5 Fehler korrigieren kann, tritt Decodierversagen auf. Dies ist dadurch gekennzeichnet, daß die durchgezogene Linie auf den Wert 1 springt und daß nach der Decodierung noch genausoviele Fehler (gekennzeichnet durch die Kreise) vorhanden sind wie direkt nach dem Empfang des Codeworts (Sterne).

Abbildung 5.18 zeigt für den gleichen Code, daß alle Codeworte, bei deren Übertragung maximal 5 Lee-Fehler aufgetreten sind , korrekt decodiert wurden. Nach der Decodierung ist die Zahl der Fehler in diesen Codeworten gleich Null. Für alle anderen Codeworte wird Decodierversagen erkannt und keine Korrektur vorgenommen.

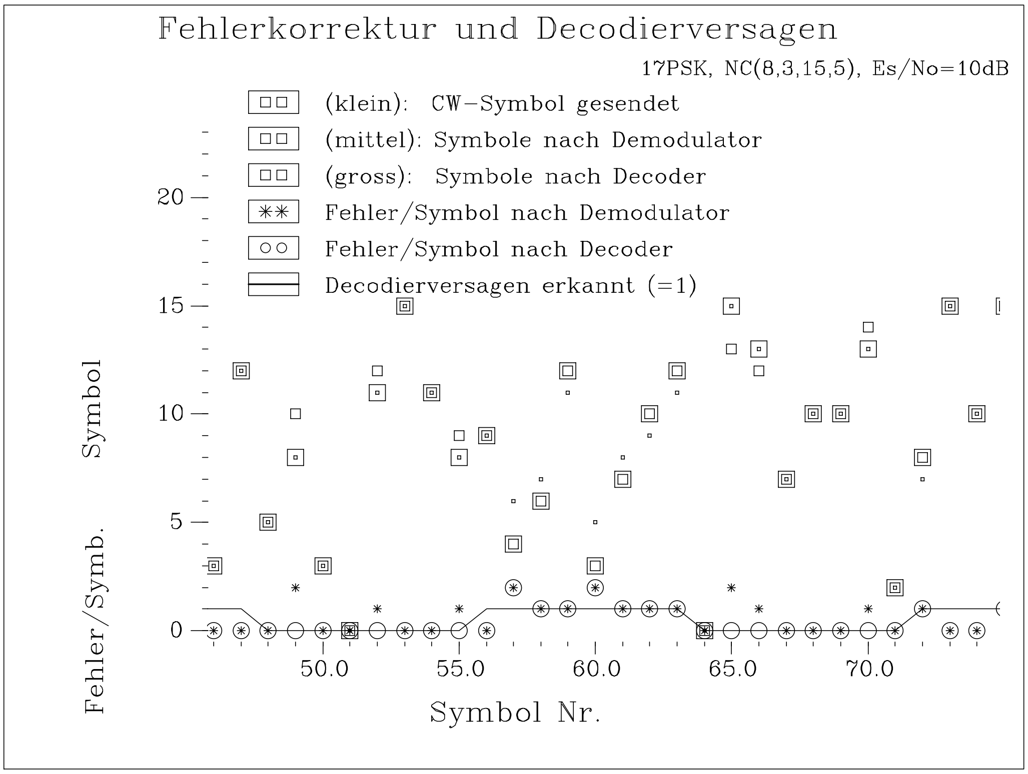

Abbildung 5.17: Fehlerkorrektur und Decodierversagen

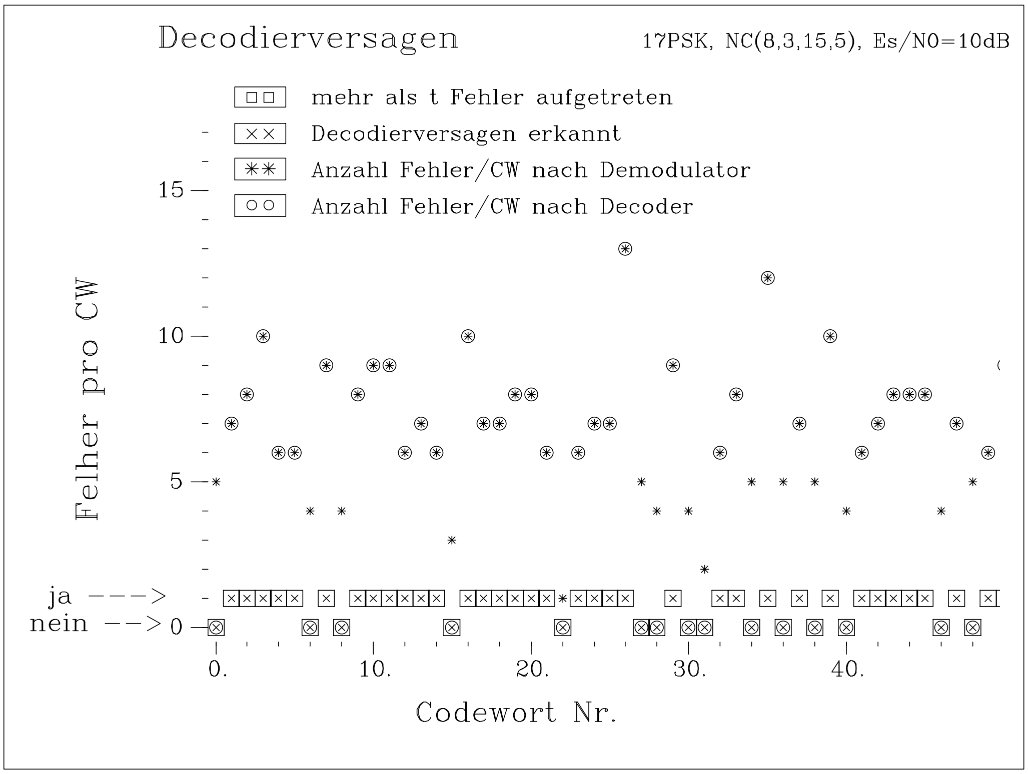

Abbildung 5.18: Decodierversagen

Kapitel 6

Anmerkungen zur Gewichtsverteilung und Mindestdistanz von NC-Codes

Während der Durchführung der vorliegenden Studienarbeit wurde außer der Codierung und Decodierung von NC-Codes auch ein Verfahren zur Bestimmung der Gewichtsverteilungen von NC-Codes in der Programmiersprache C implementiert. Dieses Programm ist wie alle anderen in [Haa96] abgedruckt.

An dieser Stelle soll nur beschrieben werden, wie die Tabellen mit den bisher unbekannten Lee-Mindestdistanzen in Anhang A.2 berechnet wurden, welche Informationen sie enthalten und welche weiteren Untersuchungsmöglichkeiten sich bieten.

Zum Verständnis der Vorgehensweise ist die Definition zweier Begriffe notwendig (siehe auch [MS83], Seite 145f):

Die Koeffizient k_i der sogenannten *Lee Composition*

$$\mathbf{k} = (k_0, k_1, k_2, \ldots, k_\delta) \tag{6.1}$$

eines Vektors der Länge n mit Elementen aus $GF(p)$ gibt an, wieviele Elemente vom Lee-Gewicht i der entsprechende Vektor enthält. Für die Summe der Koeffizienten muß gelten: $\sum_{i=0}^{\delta} k_i = n$.
Das Lee-Gewicht eines Vektors läßt sich auch aus seiner Lee Composition $\mathbf{k}$ berechnen:

$$w_L(\mathbf{c}) = \sum_{i=0}^{\delta} i \cdot k_i \tag{6.2}$$

Als *Lee Enumerator* eines NC-Codes $\mathcal{C}$ bezeichnet man folgenden Ausdruck:

$$\mathcal{L}_{\mathcal{C}}(x_0, x_1, \ldots, x_\delta) = \sum_{\mathbf{c}\in\mathcal{C}} x_o^{k_0} x_1^{k_1} \cdots x_\delta^{k_\delta} \tag{6.3}$$

Er gibt an, wieviele Codewörter des Codes $\mathcal{C}$ welche Lee Composition besitzen und wird zur Berechnung der Gewichtsverteilung des dualen Codes $\mathcal{C}$ mit Hilfe der Mac

Williams Identität benötigt (siehe [MS83], Seite 145).

Die Mindestdistanzen in Anhang A.2 wurden mit dem implementierten Verfahren nach folgendem Prinzip berechnet:

Zunächst werden durch Multiplikation aller zulässigen Informationspolynome mit einem vorgegebenen Generatorpolynom alle Codeworte des jeweiligen Codes erzeugt. Zu jedem dieser Codeworte wird die Lee Composition bestimmt. Es wird festgehalten, welche Lee Composition wie oft vorkommt. Dies entspricht dem Aufstellen des Lee Enumerator.
Anhand der Lee Compositions und ihrer Anzahl kann mit Hilfe von Gleichung 6.2 leicht die Gewichtsverteilung des untersuchten Codes angegeben werden.
In den Tabellen in Anhang A.2 sind aus der Gewichtsverteilung schließlich nur die Minimalgewichte bzw. Mindestdistanzen der untersuchten Codes angegeben.

Der zunächst umständlich erscheinende Weg über Lee Enumerator und Lee Compositions für die Berechnungen wurde gewählt, um später die Berechnung der Gewichtsverteilungen dualer Codes zu ermöglichen. Die dazu nötige Mac Williams Identität wurde im Rahmen dieser Arbeit jedoch nicht implementiert.
Im Zusammenhang mit der vorhandenen Implementierung sei insbesondere auf die speicherplatzsparende Indizierungsmöglichkeit für die Lee Compositions hingewiesen (siehe [Haa96]).

Neben den Mindestdistanzen der untersuchten Codes sind in den Tabellen im Anhang der Wert für die halbe Mindestdistanz $t_{max} = \left\lfloor \frac{d-1}{2} \right\rfloor$, die Anzahl der aufeinanderfolgenden Wurzeln im Generatorpolynom t_{Block} sowie die Zahl t_m der nach Berlekamp korrigierbaren Fehler angegeben.

Vergleicht man diese Werte, so fällt folgendes auf:

- Viele NC-Codes können mit dem Verfahren von Berlekamp nicht bis zur halben Mindestdistanz decodiert werden, es gilt: $t_{max} > t_m$.
  Ein Beispiel dafür ist der Code NC(5,1,15,4) in Tabelle A.2, der nach Berlekamp nur $t_m = 4$ Fehler korrigieren kann obwohl $t_{max} = 7$ gilt.

- Es existieren Codes, mit wenigen aufeinanderfolgenden Nullstellen im Generatorpolynom, die eine größere Mindestdistanz besitzen als die besten nach Berlekamp konstruierten NC-Codes.
  Das erste Beispiel hierfür findet man in Tabelle A.4: Der Code NC(8,2,22,6) besitzt die Mindestdistanz $d = 22$, es existiert jedoch ein Code gleicher Länge $n = 8$ und Dimension $k = 2$ mit der Mindestdistanz $d = 23$.

Es bleibt zu untersuchen, inwiefern diese Beobachtungen auch für längere NC-Codes (mit $m > 1$) gelten.
Für lange hochratige Codes ist die Berechnung der Gewichtsverteilungen aufgrund der riesigen Anzahl von Codewörtern praktisch nur noch mit Hilfe der dualen Codes und der Mac Williams Identität möglich.

Kapitel 7

Zusammenfassung und Ausblick

Im Rahmen dieser Studienarbeit wurden die Eigenschaften und die Decodierfähigkeit negazyklischer Codes untersucht, welche die Lee-Metrik verwenden. Insbesondere wurden die Codierung von NC-Codes und ein algebraisches Decodierverfahren als Modul für das Simulationsprogramm COSSAP implementiert. Als Grundlage hierzu diente die Beschreibung negazyklischer Codes durch Berlekamp sowie sein konstruktiver Beweis zur algebraischen Decodierung in [Ber84]. Sowohl die Konstruktion als auch die Decodierung wurden in Kapitel 3 beschrieben.

Im Anschluß wurden Simulationen zur Informationsübertragung mit NC-Codes in Verbindung mit MPSK-Modulation in COSSAP durchgeführt. Dabei wurden Symbol- und Blockrestfehlerraten aufgezeichnet. Desweiteren wurden Gleichungen zur Berechnung der theoretischen Blockrestfehlerwahrscheinlichkeiten hergeleitet (siehe Kapitel 4) und mit MATLAB numerisch ausgewertet. Die simulierten Fehlerraten wurden dadurch verifiziert und es wurde eine Alternative zur relativ zeitaufwendigen Simulation geschaffen.
Die ermittelten Fehlerkurven sind in Kapitel 5 dargestellt. Sie wurden mit den Fehlerkurven von RS-Codes verglichen. Dabei konnten für die untersuchten NC-Codes zusätzliche Codierungsgewinne von bis zu 3dB gegenüber vergleichbaren verkürzten RS-Codes registriert werden. Diese sind in erster Linie auf die speziell auf die MPSK-Modulation abgestimmte Lee-Metrik zurückzuführen.

Schließlich wurden für einige der untersuchten Codes die Gewichtsverteilung bzw. die Mindestdistanz ermittelt, um eine Beurteilung des realisierten Decodierverfahrens zu ermöglichen. Die Ergebnisse wurden in Kapitel 6 dargestellt und zeigen, daß das Decodierverfahren trotz der guten erzielten Ergebnisse noch verbesserungsfähig ist.

Abschließend soll nun noch ein kurzer Ausblick gegeben werden, welche Möglichkeiten der Weiterführung der vorliegenden Arbeit denkbar sind, welche Probleme noch zu lösen sind und welche vergleichbaren Ansätze bereits existieren.

- Die Abbildung von *binären Informationen* (Bits) auf nicht binäre Codesymbole wurde im Rahmen dieser Arbeit nicht näher untersucht. Sie stellt aber für den

Einsatz der nichtbinären NC-Codes in der Praxis ein nicht zu vernachlässigendes Problem dar, da heutzutage in allen Bereichen binäre Anwendungen dominieren.

- Im Zusammenhang mit dem Berlekamp-Algorithmus wäre es interessant, eine Art *Auslöschungskorrektur* zu realisieren. Insbesondere wäre es dadurch möglich auch MPSK-Modulationsschemata einzusetzen, bei denen man für M keine Primzahl sondern z.B eine Zweierpotenz wählt. D.h. das Codesymbolalphabet müßte nicht mehr gezwungenermaßen mit dem Sendesymbolalphabet übereinstimmen. Eine Abbildung zwischen Bits und verwendeten Codesymbolen wäre dann unter Umständen einfacher.

- Eine algebraische Decodierung *bis zur halben Mindestdistanz* wäre wünschenswert, ist aber mit dem Verfahren nach Berlekamp aufgrund bestimmter Einschränkungen nicht möglich. Dies wirft die Frage nach einem modifizierten oder gänzlich neuen Decodierverfahren auf.
  Einen interessanten Ausgangspunkt für ein neues Verfahren bieten unter Umständen die sogenannten *Gröbner Basen* (siehe [BW93]). Untersuchungen wurden hierzu noch nicht durchgeführt.

- Analog zu den negazyklischen Codes und der Lee-Metrik für MPSK-Modulation existieren auch für andere Modulationsverfahren spezielle Metriken und Codes:

  In [Hub94b] findet man eine erste Beschreibung *i-zyklischer Codes* über der sogenannten *Mannheim-Metrik*, die für QAM Modulation eingesetzt werden können. Eine erste Untersuchung zu den Gewichtsverteilungen dieser Codes stellt [Mau96] dar.

  Codes über Eisenstein-Jacobi Integers die auch zur Klasse der konstazyklischen Codes zählen, werden schließlich in [Hub94a] beschrieben. Die verwendeten Sendesymbolalphabete können als Teilmenge von QAM-Alphabeten betrachtet werden und besitzen von allen erwähnten Verfahren die größte „Packungsdichte“. Daher sind für diesen Ansatz die größten Codierungsgewinne zu erwarten.

  Für beide Codes existiert nach Kenntnis des Autors zur Zeit noch kein algebraisches Decodierverfahren.

Anhang A

Parameter negazyklischer Codes

A.1 Die Parameter n, k und t_m

A.2 Größte Mindestdistanzen d (für $m = 1$)

	p	5				7				11			13			17			19		23		29		31	
	m	1	2	3	4	1	2	3	4	1	2	3	1	2	3	1	2	3	1	2	1	2	1	2	1	2
t_m	n	2	12	62	312	3	24	171	1200	5	60	665	6	84	1098	8	144	2456	9	180	11	264	14	420	15	480
1	k	1	10	59	308	2	22	168	1196	4	58	662	5	82	1095	7	142	2453	8	178	10	262	13	418	14	478
2			8	56	304	1	20	165	1192	3	56	659	4	80	1092	6	140	2450	7	176	9	260	12	416	13	476
3							18	162	1188	2	54	656	3	78	1089	5	138	2447	6	174	8	258	11	414	12	474
4										1	52	653	2	76	1086	4	136	2444	5	172	7	256	10	412	11	472
5											50	650	1	74	1083	3	134	2441	4	170	6	254	9	410	10	470
6														72	1080	2	132	2438	3	168	5	252	8	408	9	468
7																1	130	2435	2	166	4	250	7	406	8	466
8																	128	2432	1	164	3	248	6	404	7	464
9																				162	2	246	5	402	6	462
10																					1	244	4	400	5	460
11																						242	3	398	4	458
12																							2	396	3	456
13																							1	394	2	454
14																								392	1	452

Tabelle A.1: Parameter n, k und t_m von NC-Codes

p	m	n	k	d	t_{max}	t_{Block}	t_m	Generatorpolynom	
5	1	2	1	3	1	1	**1**	$(x-\alpha^1)$	$= x+\alpha^3$
7	1	3	1	6	2	2	**2**	$(x-\alpha^1)(x-\alpha^3)$	$= x^2+\alpha^5x+\alpha^4$
7	1	3	1	6	2	2	0	$(x-\alpha^3)(x-\alpha^5)$	$= x^2+\alpha x+\alpha^2$
7	1	3	2	3	1	1	**1**	$(x-\alpha^1)$	$= x+\alpha^4$
7	1	3	2	3	1	1	0	$(x-\alpha^5)$	$= x+\alpha^2$
11	1	5	1	15	7	4	**4**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)$	$= x^4+\alpha^9x^3+\alpha^8x^2+\alpha^7x^x+\alpha^6$
11	1	5	1	15	7	4	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)$	$= x^4+\alpha x^3+\alpha^2x^2+\alpha^3x^x+\alpha^4$
11	1	5	1	15	7	3	3	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^9)$	$= x^4+\alpha^7x^3+\alpha^4x^2+\alpha x^x+\alpha^8$
11	1	5	1	15	7	3	1	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)$	$= x^4+\alpha^3x^3+\alpha^6x^2+\alpha^9x^x+\alpha^2$
11	1	5	2	8	3	3	**3**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)$	$= x^3+\alpha^6x^2+x+\alpha^4$
11	1	5	2	8	3	3	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)$	$= x^3+\alpha^4x^2+\alpha^4x+1$
11	1	5	2	8	3	3	0	$(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)$	$= x^3+\alpha^6x^2+\alpha^8x^x+\alpha^6$
11	1	5	2	8	3	2	1	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^7)$	$= x^3+\alpha^7x^2+\alpha^3x+\alpha^8$
11	1	5	2	8	3	2	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^9)$	$= x^3+\alpha^5x^2+\alpha^9x+\alpha^2$
11	1	5	2	8	3	1	1	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^9)$	$= x^3+\alpha x^2+\alpha x+1$
11	1	5	3	5	2	2	**2**	$(x-\alpha^1)(x-\alpha^3)$	$= x^2+\alpha x+\alpha^4$
11	1	5	3	5	2	2	0	$(x-\alpha^7)(x-\alpha^9)$	$= x^2+\alpha^7x+\alpha^6$
11	1	5	3	5	2	1	1	$(x-\alpha^1)(x-\alpha^7)$	$= x^2+x+\alpha^8$
11	1	5	3	5	2	1	1	$(x-\alpha^1)(x-\alpha^9)$	$= x^2+\alpha^9x+1$
11	1	5	3	5	2	1	0	$(x-\alpha^3)(x-\alpha^7)$	$= x^2+\alpha^6x+1$
11	1	5	3	5	2	1	0	$(x-\alpha^3)(x-\alpha^9)$	$= x^2+\alpha^2x+\alpha^2$
11	1	5	4	3	1	1	**1**	$(x-\alpha^1)$	$= x+\alpha^6$
11	1	5	4	3	1	1	0	$(x-\alpha^3)$	$= x+\alpha^8$
11	1	5	4	3	1	1	0	$(x-\alpha^7)$	$= x+\alpha^2$
11	1	5	4	3	1	1	0	$(x-\alpha^9)$	$= x+\alpha^4$

Tabelle A.2: Größte vorkommende Mindestdistanzen und zugehörige Generatorpolynome (Teil 1)

p	m	n	k	d	t_{max}	t_{Block}	t_m	Generatorpolynom
13	1	6	1	21	10	5	**5**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9) = x^5+\alpha^{11}x^4+\alpha^{10}x^3+\alpha^9x^2+\alpha^8x+\alpha^7$
13	1	6	1	21	10	5	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11}) = x^5+\alpha^1x^4+\alpha^2x^3+\alpha^3x^2+\alpha^4x+\alpha^5$
13	1	6	1	21	10	3	3	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^9)(x-\alpha^{11}) = x^5+\alpha^7x^4+\alpha^2x^3+\alpha^9x^2+\alpha^4x+\alpha^{11}$
13	1	6	1	21	10	3	2	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11}) = x^5+\alpha^5x^4+\alpha^{10}x^3+\alpha^3x^2+\alpha^8x+\alpha^1$
13	1	6	2	13	6	4	**4**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7) = x^4+x^3+\alpha^3x^2+\alpha^8x+\alpha^4$
13	1	6	2	13	6	3	3	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^{11}) = x^4+\alpha^{10}x^3+\alpha^{11}x^2+\alpha^2x+\alpha^8$
13	1	6	2	13	6	3	1	$(x-\alpha^1)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11}) = x^4+\alpha^6x^3+\alpha^3x^2+\alpha^2x+\alpha^4$
13	1	6	2	13	6	4	0	$(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11}) = x^4+\alpha^4x^3+\alpha^{11}x^2+\alpha^8x+\alpha^8$
13	1	6	3	8	3	3	0	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^7) = x^3+\alpha^{11}x^2+\alpha^8x+\alpha^7$
13	1	6	3	8	3	3	0	$(x-\alpha^5)(x-\alpha^7)(x-\alpha^9) = x^3+\alpha^8x^2+\alpha^9x+\alpha^3$
13	1	6	3	8	3	2	2	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^{11}) = x^3+\alpha^2x^2+\alpha^9x+\alpha^9$
13	1	6	3	8	3	2	1	$(x-\alpha^1)(x-\alpha^9)(x-\alpha^{11}) = x^3+x^2+\alpha^5x+\alpha^3$
13	1	6	3	7	3	3	**3**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5) = x^3+\alpha^4x^2+\alpha^1x+\alpha^3$
13	1	6	3	7	3	2	2	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^7) = x^3+\alpha^9x^2+\alpha^8x+\alpha^5$
13	1	6	3	7	3	2	1	$(x-\alpha^1)(x-\alpha^7)(x-\alpha^9) = x^3+\alpha^3x^2+\alpha^8x+\alpha^{11}$
13	1	6	3	7	3	2	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^{11}) = x^3+\alpha^9x^2+\alpha^4x+\alpha$
13	1	6	3	7	3	2	0	$(x-\alpha^5)(x-\alpha^9)(x-\alpha^{11}) = x^3+\alpha^3x^2+\alpha^4x+\alpha^7$
13	1	6	3	7	3	3	0	$(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11}) = x^3+\alpha^{10}x^2+\alpha^1x+\alpha^9$
13	1	6	4	5	2	2	**2**	$(x-\alpha^1)(x-\alpha^3) = x^2+\alpha^{10}x+\alpha^4$
13	1	6	4	5	2	2	0	$(x-\alpha^3)(x-\alpha^5) = x^2+\alpha x+\alpha^8$
13	1	6	4	5	2	1	0	$(x-\alpha^3)(x-\alpha^7) = x^2+\alpha^{11}x+\alpha^{10}$
13	1	6	4	5	2	1	1	$(x-\alpha^1)(x-\alpha^9) = x^2+\alpha^5x+\alpha^{10}$
13	1	6	4	5	2	1	0	$(x-\alpha^5)(x-\alpha^9) = x^2+\alpha x+\alpha^2$
13	1	6	4	5	2	2	0	$(x-\alpha^7)(x-\alpha^9) = x^2+\alpha^4x+\alpha^4$
13	1	6	4	5	2	1	0	$(x-\alpha^3)(x-\alpha^{11}) = x^2+\alpha^7x+\alpha^2$
13	1	6	4	5	2	2	0	$(x-\alpha^9)(x-\alpha^{11}) = x^2+\alpha^6x+\alpha^8$
13	1	6	5	3	1	1	**1**	$(x-\alpha^1) = x+\alpha^7$
13	1	6	5	3	1	1	0	$(x-\alpha^5) = x+\alpha^{11}$
13	1	6	5	3	1	1	0	$(x-\alpha^7) = x+\alpha^1$
13	1	6	5	3	1	1	0	$(x-\alpha^{11}) = x+\alpha^5$

Tabelle A.3: Größte vorkommende Mindestdistanzen und zugehörige Generatorpolynome (Teil 2)

p	m	n	k	d	t_{max}	t_{Block}	t_m	Generatorpolynom
17	1	8	1	36	17	7	**7**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})$
17	1	8	1	36	17	6	6	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{15})$
17	1	8	1	36	17	5	5	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	1	36	17	4	4	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	1	36	17	4	3	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	1	36	17	5	2	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	1	36	17	6	1	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	1	36	17	7	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	2	23	11	6	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})$
17	1	8	2	23	11	4	1	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{15})$
17	1	8	2	23	11	2	2	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	2	23	11	3	3	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	2	22	10	6	**6**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})$
17	1	8	2	22	10			...
17	1	8	3	16	7	4	4	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^{11})$
17	1	8	3	16	7	4	4	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^{13})$
17	1	8	3	16	7	4	1	$(x-\alpha^1)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})$
17	1	8	3	16	7	3	3	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})$
17	1	8	3	16	7	4	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)(x-\alpha^{15})$
17	1	8	3	16	7	3	1	$(x-\alpha^1)(x-\alpha^7)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	3	16	7	4	0	$(x-\alpha^3)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	3	16	7	4	0	$(x-\alpha^5)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	3	15	7	5	**5**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)$
17	1	8	3	15	7			...

Tabelle A.4: Größte vorkommende Mindestdistanzen und zugehörige Generatorpolynome (Teil 3)

p	m	n	k	d	t_{max}	t_{Block}	t_m	Generatorpolynom
17	1	8	4	12	5	4	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)$
17	1	8	4	12	5	2	1	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^7)(x-\alpha^{11})$
17	1	8	4	12	5	2	2	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^7)(x-\alpha^{13})$
17	1	8	4	12	5	4	0	$(x-\alpha^7)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{13})$
17	1	8	4	12	5	3	3	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^{15})$
17	1	8	4	12	5	2	0	$(x-\alpha^5)(x-\alpha^9)(x-\alpha^{11})(x-\alpha^{15})$
17	1	8	4	12	5	2	0	$(x-\alpha^3)(x-\alpha^9)(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	4	12	5	3	1	$(x-\alpha^1)(x-\alpha^{11})(x-\alpha^{13})(x-\alpha^{15})$
17	1	8	4	11	5	3	1	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^7)(x-\alpha^9)$
17	1	8	4	11	5			...
17	1	8	4	9	4	4	**4**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)$
17	1	8	4	9	4			...
17	1	8	5	8	3	2	2	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^7)$
17	1	8	5	8	3	2	1	$(x-\alpha^1)(x-\alpha^5)(x-\alpha^7)$
17	1	8	5	8	3	2	0	$(x-\alpha^3)(x-\alpha^5)(x-\alpha^7)$
17	1	8	5	8	3			...
17	1	8	5	7	3	3	**3**	$(x-\alpha^1)(x-\alpha^3)(x-\alpha^5)$
17	1	8	5	7	3			...

Tabelle A.5: Größte vorkommende Mindestdistanzen und zugehörige Generatorpolynome (Teil 4)

Anhang B

Flußdiagramme wichtiger Algorithmen

B.1 Berlekamp-Algorithmus

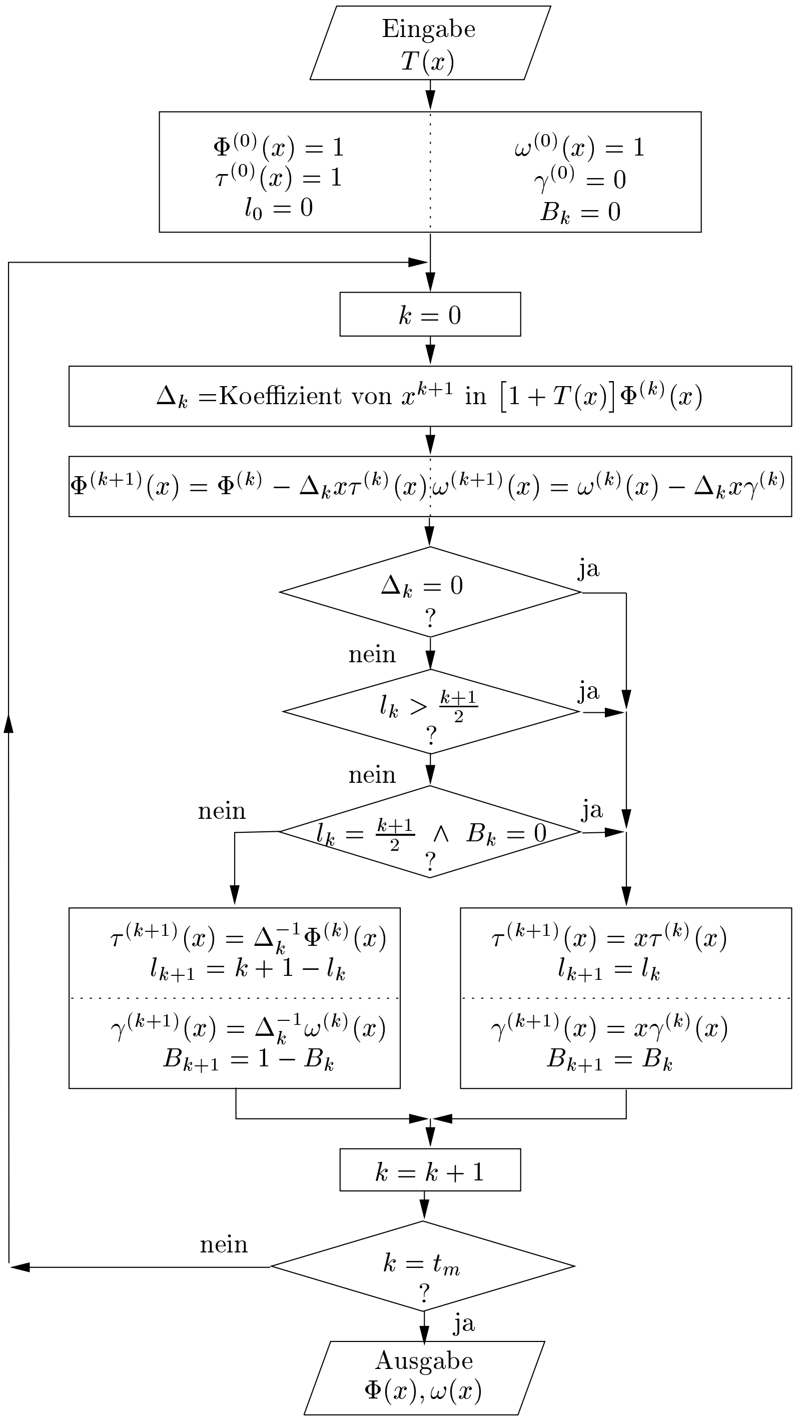

Abbildung B.1:
Berlekamp-Algorithmus zur Lösung der Gleichung $\big[1 + T(x)\big]\Phi(x) = \omega(x)$

B.2 Decodieralgorithmus für NC-Codes

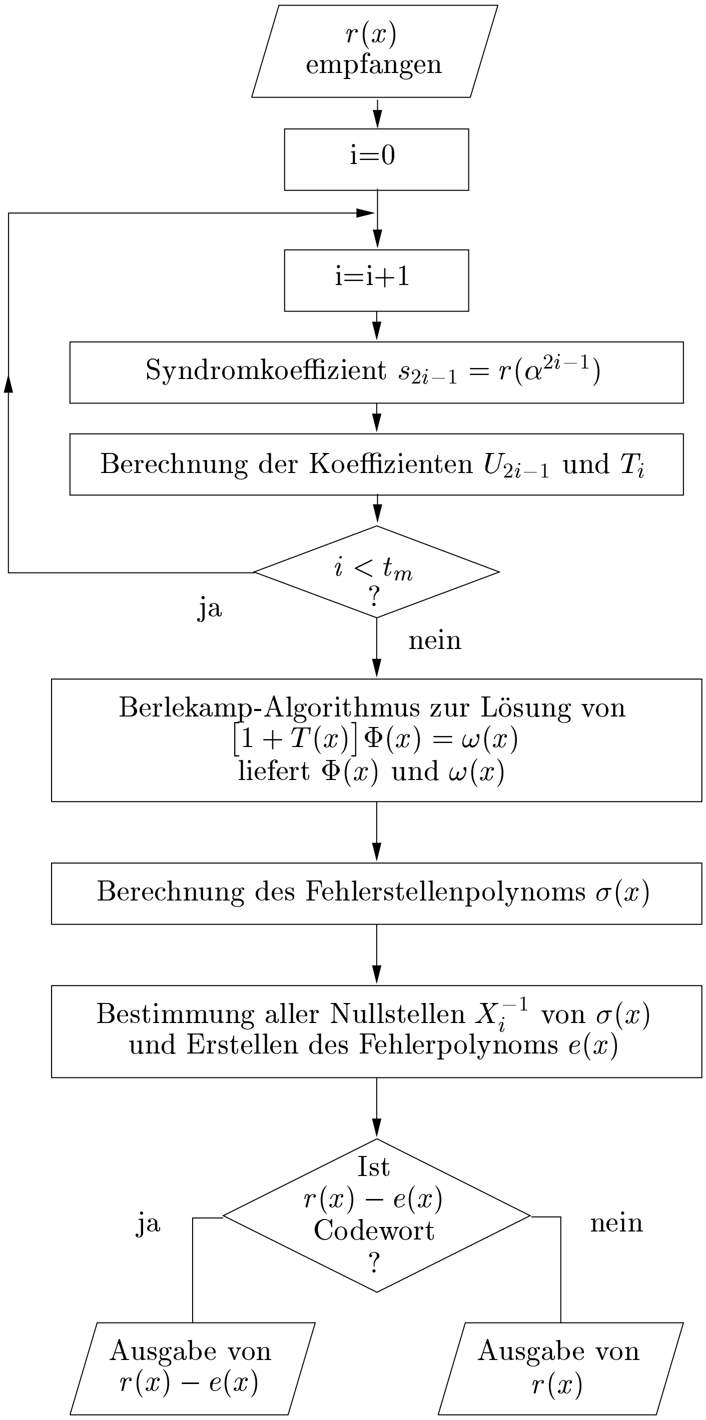

Abbildung B.2: Decodieralgorithmus für negazyklische Codes

Anhang C

Implementierung in COSSAP

Im folgenden sind die Model Definition Files der Primitiven Module aufgeführt, die im Rahmen dieser Studienarbeit implementiert wurden. Sie beschreiben die Parameter und die Funktionen dieser Module. Die Namen der Module lauten:

`NEGACYCL_CO2`	(Encoder für negazyklische Codes)
`NEGACYCL_DE2`	(Decoder für negazyklische Codes)
`ISRC`	(nichtbinäre Zufallsquelle)

Die vollständige C-Funktionsbibliothek, die speziell für diese Module angefertigt wurde, sowie die Programmlistings der Module sind in [Haa96] abgedruckt.

C.1 Der Encoder `NEGACYCL_CO2`

```
.NAME
 NEGACYCL_CO2

.SHORT_DESCRIPTION
 Implementation of an Encoder for negacyclic codes.

.DESCRIPTION
 This model codes each k nonbinary input symbols to a nonbinary codeword of
 length n using a Negacyclic Code.
 The coding method is chosen by the parameter 'mode'.

 The parameters of a (nonbinary) negacyclic code are:
 p, m, t
 n = (p^m - 1) / 2       length
 1 <= t <= (p-1)/2        number of correctable errors
 k = n - t*m             dimension

 Information and code symbols are of GF(p)
```

```
 The name of the corresponding Encoder is NEGACYCL_DE2.


.IMPLEMENTATIONS
 1 SOURCE_CODE C negacycl_co2

.CLASSES
 1 CODEC
 2 CHANNEL_CODING

.INPUT_PORTS
 1 INFOWORD (I)
   input signal with symbols of GF(p)
   k symbols are an information vektor

.OUTPUT_PORTS
 1 CODEWORD (I)
   output signal resulting of coding k input symbols of GF(p)
   to a codeword of n symbols of GF(p)

.PARAMETERS
 1 p (I)
   has to be a prime, code symbols are of GF(p),
   ( suited modulation method: p-PSK )
 2 m (I)
   used galois-field: GF(p^m)
   code-length: n = (p^m-1)/2
 3 t (I)
   number of correctable errors
   1 <= t <= (p-1)/2
 4 mode (I)
   Codingmode: 0   cw(x) = i(x) * g(x)
               1   systematical coding with g(x)

.ERROR_CODES
 1 WRONG PARAMETER
   The parameters of this model are wrong

.HISTORY
 6.1  15.3.1996 : Guenther Haas

.END
```

C.2 Der Decoder NEGACYCL_DE2

```
.NAME
 NEGACYCL_DE2

.SHORT_DESCRIPTION
 Implementation of a decoder for negacyclic codes.

.DESCRIPTION
 This model corrects an incoming incorrect codeword of a Negacyclic Code using
 the Berlekamp Algorithm (Hard Decision), if less than t errors have occured.
 Otherwise the received vector isn't corrected.
 After this, the codeword is decoded systematical or not, depending on
 parameter 'mode'.
 The information vector and the (corrected) codeword are written to the
 output ports INFOWORD and RECEIVED VECTOR, besides STATUS is set to 0 if
 decoding was successfull - in case of failure, it's set to 1.

 The parameters of a (nonbinary) negacyclic code are:
 p, m, t
 n = (p^m - 1)/2      length
 1 <= t <= (p-1)/2     number of correctable errors
 k = n - t*m          dimension

 Information and code symbols are of GF(p)


 The name of the corresponding Encoder is NEGACYCL_CO2.

.IMPLEMENTATIONS
 1 SOURCE_CODE C negacycl_de2

.CLASSES
 1 CODEC
 2 CHANNEL_CODING

.INPUT_PORTS
 1 RECEIVED_VECTOR (I)
   input signal with symbols of GF(p): codeword + errors
   n symbols are one vektor
   (n is the length of a codeword)

.OUTPUT_PORTS
 1 INFOWORD (I)
   output signal: information vector of k symbols of GF(p)
   resulting from decoding the corrected codeword (status=0)
                           the received vector (status=1)
   using the decoding method given by parameter 'mode'
```

```
 2 CORRECTED_VECTOR (I)
   corrected codeword or received vector depending on the value of 'status'
 3 STATUS (I)
   status of the output vectors:
   1 = failure in decoding (more than t errors have occured)
   0 = OK, successfull decoding
   (only one value per codeword)

.PARAMETERS
 1 p (I)
   has to be a prime, code symbols are of GF(p),
   ( suited modulation method: p-PSK )
 2 m (I)
   used galois-field: GF(p^m)
   code-length: n = (p^m-1)/2
 3 t (I)
   number of correctable errors
   1 <= t <= (p-1)/2
 4 mode (I)
   (De-) Codingmode: 0   cw(x) = i(x) * g(x)
                     1   systematical coding with g(x)

.ERROR_CODES
 1 WRONG PARAMETER
   Some parameters of this model are wrong.
 2 IRGENDEINFEHLER
   An error has occured during decoding.
   ( Look at the COSSAP Command Window. )

.HISTORY
 6.1  18.3.1996 : Guenther Haas

.END
```

C.3 Die nichtbinäre Zufallsquelle ISRC

```
.NAME
 ISRC

.SHORT_DESCRIPTION
 Implementation of a nonbinary random source.

.DESCRIPTION
 This model is a nonbinary randam source, which supplies integer values of
 {0,1,...,p-1}. It's based on the c-functions 'rand' and 'srand(init-value)'.
```

```
 If p is a prime the random values are of the Galois Field GF(p).

.IMPLEMENTATIONS
 1 SOURCE_CODE C isrc

.CLASSES
 1 SOURCE

.OUTPUT_PORTS
 1 OUT_INT (I)
   Integer random output signal.
   Symbols of {0,1,...,p-1}.

.PARAMETERS
 1 p (I)
   establishes the maximum integer value p-1
 2 init_value (I)
   random seed, start value for the random generator

.HISTORY
 6.1  15.3.1995 : Guenther Haas

.END
```

Anhang D

Abkürzungen und Formelzeichen

D.1 Abkürzungen

AWGN	**A**dditive **W**hite **G**aussian **N**oise
BCH	**B**ose Ray–**C**haudhuri **H**ocquenghem
BMD	**B**egrenzte–**M**indestdistanz–**D**ecodierung bzw. **B**ounded **M**inimum **D**istance
BPSK	**B**inary **P**hase **S**hift **K**eying
`C`	Programmiersprache `C`
LGS	**L**ineares **G**leichungs**s**ystem
C	Programmiersprache C
COSSAP	**C**ommunication **S**ystem **S**imulation and **A**nalysis **P**ackage
$GF(q)$	**G**alois**f**eld
$GF^A(q)$	Körper der absolut kleinsten Reste
HD	**H**ard–**D**ecision
MATLAB	**M**atrix **L**aboratory
ML	**M**aximum–**L**ikelihood
MPSK	**M**–**P**hase **S**hift **K**eying
NC	**N**egazyklischer **C**ode bzw. **N**ega**c**yclic Code
QAM	**Q**uadratur **A**mplituden **M**odulation
RS	**R**eed **S**olomon (Code)
SD	**S**oft–**D**ecision
SDML	**S**oft–**D**ecision **M**aximum–**L**ikelihood

D.2 Formelzeichen

$\mathcal{C}$	Code
$\mathcal{C}^\perp$	dualer Code von $\mathcal{C}$
$\mathbf{c}$	Codevektor
$c(x)$	Codepolynom
$\hat{\mathbf{c}}$	decodiertes Codewort
d	Mindestdistanz
$d(x, y)$	Distanz zweier Symbole
$d(\mathbf{x}, \mathbf{y})$	Distanz zweier Vektoren
d_E	Euklidische Metrik
d_H	Hamming-Metrik
d_L	Lee-Metrik
E_s	Symbolenergie
$\mathbf{e}$	Fehlervektor
$e(x)$	Fehlerpolynom
$f(x)$	Fehlerstellenpolynom
$\mathbf{G}$	Generatormatrix eines Blockcodes
$g(x)$	Generatorpolynom eines zyklischen oder negazyklischen Codes
$\mathbf{H}$	Prüfmatrix eines Block– oder Faltungscodes
$\mathbf{H}_{min}$	Prüfmatrix mit möglichst geringer Zeilenzahl
$\mathbf{H}_*$	Prüfmatrix mit Elementen aus $GF(p)$
$h(x)$	Prüfpolynom eines zyklischen oder negazyklischen Codes
$\mathbf{I}_k$	$(k \times k)$–Einheitsmatrix
$\mathbf{i}$	Informationsvektor
$i(x)$	Informationspolynom
$\hat{\mathbf{i}}$	geschätzte Informationsfolge
k	Dimension eines Block- oder Faltungscodes, Anzahl der Komponenten von $\mathbf{i}$
$\mathbf{k}$	Lee Composition
k_i	Komponente der Lee Composition
K_i	Kreisteilungsklasse
$\mathcal{L}_\mathcal{C}$	Lee Enumerator
$m_i(x)$	minimales Polynom
$M_{(n)}$	$(n \times n)$ - Mattson-Solomon Transformationsmatrix
M_i	Menge der zu α^i konjugiert komplexen Wurzeln
$\mathcal{M}$	Menge von Kreisteilungsklassen
n	Länge des Block– oder Faltungscodes, Anzahl der Komponenten von $\mathbf{c}$
N	komplexwertige, gaußverteilte Zufallsvariable
N_0	einseitige Rauschleistungsdichte
p	Primzahl
$p(x)$	Polynom in x
p_i	Polynomkoeffizienten
$p(x, y)$	Wahrscheinlichkeitsdichte in kartesischen Koordinaten
$p(r, \varphi)$	Wahrscheinlichkeitsdichte in Polarkoordinaten
$p(\varphi)$	winkelabhängige Wahrscheinlichkeitsdichte

P_c	Blockrestfehlerwahrscheinlichkeit
$P_{c,fail}$	Blockfehlerwahrscheinlichkeit bei Decodierversagen
$P_{c,res}$	Blockgrundfehlerwahrscheinlichkeit
$P_{c,uncoded}$	Blockfehlerwahrscheinlichkeit (bei uncodierter Übertragung)
P_{fail}	Decodierversagenshäufigkeit
P_s	Symbolrestfehlerwahrscheinlichkeit
$P_{s,fail}$	Symbolfehlerwahrscheinlichkeit bei Decodierversagen
$P_{s,res}$	Symbolgrundfehlerwahrscheinlichkeit
$P_{s,uncoded}$	Symbolfehlerwahrscheinlichkeit (bei uncodierter Übertragung)
q	Größe des Symbolalphabets $GF(q)$
r	Ordnung eines Elements aus $GF(q) oder Radius bei Polarkoordinaten$
R	Coderate ($R = \frac{k}{n}$)
$\mathbf{r}$	Empfangsvektor
$r(x)$	Empfangspolynom
$\mathbf{s}$	Syndrom
s_i	Syndromkoeffizient
$s(x)$	Syndrom in Polynomschreibweise
$s_e(x)$	Anteil von $s(x)$ mit geraden Exponenten
$s_o(x)$	Anteil von $s(x)$ mit ungeraden Exponenten
t	Fehlerzahl
t_m	Anzahl korrigierbarer Fehler bei NC-Codes nach Berlekamp
t_{max}	Anzahl korrigierbarer Fehler bei Decodierung von bis zur halben Mindestdistanz
$T(x)$	Hilfspolynom bei algebraischer Decodierung von NC-Codes
$U(x)$	Hilfspolynom bei algebraischer Decodierung von NC-Codes
V	Vektor im komplexen Signalraum
$w(\mathbf{x})$	Gewicht eines Vektors
w_{min}	Minimalgewicht eines Codes
$w_E(\mathbf{x})$	Euklidische Norm $\|\|\mathbf{x}\|\|$ eines Vektors
$w_H(\mathbf{x})$	Hamming-Gewicht eines Vektors
$w_L(\mathbf{x})$	Lee-Gewicht eines Vektors
$\mathbf{x}$	komplexer Sendevektor
X	Realteil von V
X_i	Fehlerstellenwert
X_i^{-1}	Nullstelle des Fehlerstellenpolynoms
$\mathbf{y}$	komplexer Empfangsvektor
Y	Imaginärteil von V
Y_i	Fehlerwert bei Decodierung von RS- und BCH-Codes
α	primitives Element eines Galloisfeldes
$\boldsymbol{\zeta}$	
$\zeta(x)$	$\bullet\!\!-\!\!\circ\ c(x)$
δ	wirkliche Mindestdistanz bei BCH-Codes bzw. maximale Fehlerzahl pro Symbol bei NC-Codes
$\boldsymbol{\epsilon}$	
$\epsilon(x)$	$\bullet\!\!-\!\!\circ\ e(x)$

$\boldsymbol{\gamma}$	
$\gamma(x)$	$\bullet\!\!-\!\!\circ\ g(x)$
κ	Konstante bei konstazyklischer Verschiebung
π	Kreiszahl
$\boldsymbol{\varrho}$	$\bullet\!\!-\!\!\circ\ r(x)$
σ	Standardabweichung
$\sigma(x)$	$\bullet\!\!-\!\!\circ\ f(x)$, Fehlerstellenpolynom bei algeb. Decodierung
$\sigma_e(x)$	Anteil von $\sigma(x)$ mit geraden Exponenten
$\sigma_o(x)$	Anteil von $\sigma(x)$ mit ungeraden Exponenten
φ	Winkel bei Polarkoordinaten
$\Phi(x)$	Fehlerstellenpolynom im Berlekamp-Algorithmus
$\omega(x)$	Fehlerberechnungspolynom

Abbildungsverzeichnis

Literaturverzeichnis

[Ber84] Elwyn R. Berlekamp. *Algebraic Coding Theory.* Aegean Park Press, Laguna Hills, CA (1984).

[Bla90] Richard E. Blahut. *Digital Transmission of Information.* Addison-Wesley Publishing Company, (1990).

[Bos92] M. Bossert. *Kanalcodierung.* B.G. Teubner, Stuttgart (1992).

[BSG91] I.N. Bronstein, K.A. Semendjajew und G. Grosche. *Taschenbuch der Mathematik*, Bd. 25. Verlag Harry Deutsch, Thun und Frankfurt/Main (1991).

[BW93] Thomas Becker und Volker Weispfenning. *Gröbner Bases.* Springer, New York, Heidelberg (1993).

[Gai93] Jürgen Gaiser. *Implementierung von Funktionen zur Reed-Solomon-Codierung/-Decodierung.* Studienarbeit, Universität Ulm, Abt. Informationstechnik, 1993. (S/1993/Br02).

[Haa96] Günther Haas. *Programmdokumentation zur Studienarbeit „Negazyklische Codes und Lee-Metrik“.* , Universität Ulm, Abt. Informationstechnik, 1996.

[Has69] Helmut Hasse. *Zahlentheorie*, Bd. 3. Akademie-Verlag, Berlin (1969).

[Hub94a] Klaus Huber. *Codes Over Eisenstein-Jacobi Integers.* Contemporary Mathematics **168** (1994), 165–179.

[Hub94b] Klaus Huber. *Codes Over Gaussian Integers.* IEEE Transactions on Information Theory **40** (January 1994) 1, 207–216.

[HW58] G.H. Hardy und E.M. Wright. *Einführung in die Zahlentheorie.* R. Oldenbourg, München (1958).

[Koc74] R. Kochendörffer. *Einführung in die Algebra*, 4. Aufl. VEB Deutscher Verlag der Wissenschaften, Berlin (1974).

[Kre94] Michael Kreuzer. *Optimierung eines REED-Solomon Encoders/Decoders und Implementierung in COSSAP.* Studienarbeit, Universität Ulm, Abt. Informationstechnik, 1994. (S/1994/RL1).

[Lin93] J. Lindner. *Nachrichtentechnik I,II.* Vorlesungsmitschrift, 1993.

[Mau96] Johannes Maucher. *On i-cyclic codes and their Mannheim weight.* Vorabdruck (1996).

[MS83] F.J MacWilliams und N.J. Sloane. *Theory of Error Correcting Codes.* North–Holland, Amsterdam (1983).

[Pro89] John G. Proakis. *Digital Communications*, 2. Aufl. Computer Science. McGRAW-HILL International Editions, (1989).