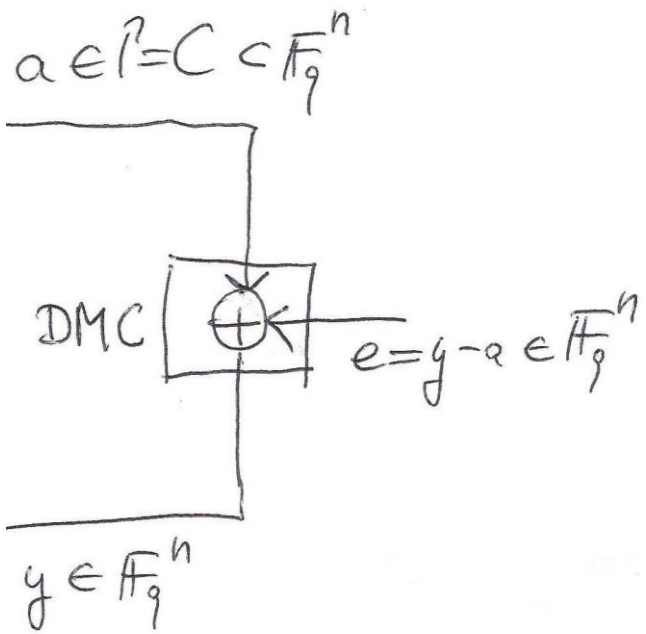
Unterstützende Materialien zur Vorlesung

Verfahren zur Kanalcodierung – Teil 4

Prof. Dr. Bernd Friedrichs
KIT CEL

Inhalt

- Nebenklassen-Zerlegung, Syndrom-Decodierung
- Definition zyklischer Codes und Polynombeschreibung
- Generatorpolynom
- Prüfpolynom
- Systematische Encodierung
- Syndrom
- Erkennung von Einzelfehlern und Bündelfehlern sowie CRC-Codes
- Korrektur von Einzelfehlern und Bündelfehlern



Definition 4.7. *Vorausgesetzt wird ein $(n,k)_q$-Code $\mathcal{C}$ mit der Prüfmatrix $\boldsymbol{H} \in \mathbb{F}_q^{n-k,n}$. Zum Empfangswort $\boldsymbol{y}$ wird das Syndrom wie folgt definiert:*

$$\boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^T. \tag{4.6.1}$$

Das Syndrom hat also die Länge $n-k$. Für die Darstellung $\boldsymbol{y} = \boldsymbol{a} + \boldsymbol{e}$ mit einem Codewort $\boldsymbol{a}$ und einem Fehlerwort $\boldsymbol{e}$ gilt:

$$\boldsymbol{s} = (\boldsymbol{a} + \boldsymbol{e})\boldsymbol{H}^T = \boldsymbol{a}\boldsymbol{H}^T + \boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{e}\boldsymbol{H}^T. \tag{4.6.2}$$

Das Syndrom eines Wortes ist genau dann das Nullwort, wenn das Wort ein Codewort ist. Das Syndrom des Empfangswortes ist also unabhängig vom gesendeten Codewort und nur abhängig vom überlagerten Fehlermuster. Es gibt $q^n - q^k$ verschiedene Fehlermuster, die keine Codewörter sind, und es gibt q^{n-k} verschiedene Syndrome. Folglich ist ein Fehlermuster durch sein Syndrom nicht eindeutig gekennzeichnet.

Medizinische Fachbegriffe
Symptom = Anzeichen einer Krankheit; für eine Krankheit charakteristische Erscheinungen
Syndrom = durch das gemeinsame Auftreten bestimmter charakteristischer Symptome gekennzeichnetes Krankheitsbild

Die Syndrome werden als $\boldsymbol{s}_\mu$ durchnumeriert mit $0 \leq \mu \leq q^{n-k} - 1$ und der Festlegung $\boldsymbol{s}_0 = \boldsymbol{0}$. Für jedes Syndrom wird die Menge der Fehlermuster definiert, die zu diesem Syndrom führen:

$$\mathcal{M}_\mu = \{\boldsymbol{e} \in \mathbb{F}_q^n \mid \boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{s}_\mu\}. \tag{4.6.3}$$

Da alle Codewörter das Syndrom Null haben, gilt $\mathcal{M}_0 = \mathcal{C}$ und alle anderen $\mathcal{M}_\mu$ enthalten kein Codewort. Ferner sind die Mengen alle disjunkt, denn ein Fehlermuster kann nicht verschiedene Syndrome haben. Es sei $\boldsymbol{e}, \boldsymbol{e}' \in \mathcal{M}_\mu$. Aus $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{e}'\boldsymbol{H}^T$ folgt $(\boldsymbol{e}' - \boldsymbol{e})\boldsymbol{H}^T = \boldsymbol{e}'\boldsymbol{H}^T - \boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{0}$. Also ist die Differenz von zwei Wörtern aus $\mathcal{M}_\mu$ stets ein Codewort. Für ein beliebiges $\boldsymbol{e} \in \mathcal{M}_\mu$ gilt also:

$$\boldsymbol{e} + \mathcal{C} = \{\boldsymbol{e} + \boldsymbol{a} \mid \boldsymbol{a} \in \mathcal{C}\} = \mathcal{M}_\mu. \tag{4.6.4}$$

Die Menge $\mathcal{M}_\mu$ kann also dargestellt werden als Summe eines beliebigen Elementes aus $\mathcal{M}_\mu$ und der Codemenge. Folglich hat jede Menge $\mathcal{M}_\mu$ die gleiche Mächtigkeit:

$$\left|\mathcal{M}_\mu\right| = \left|\mathcal{C}\right| = q^k = \frac{q^n}{q^{n-k}} = \frac{\text{Anzahl aller Wörter}}{\text{Anzahl der Syndrome}}. \tag{4.6.5}$$

Die q^{n-k} Mengen $\mathcal{M}_\mu$ bilden eine eindeutige disjunkte Zerlegung von $\mathbb{F}_q^n$:

$$\mathbb{F}_q^n = \biguplus_{\mu=0}^{q^{n-k}-1} \mathcal{M}_\mu. \tag{4.6.6}$$

Definition 4.8. *Die Mengen $\mathcal{M}_\mu$ heißen* Nebenklassen *(cosets) und die Zerlegung (4.6.6) heißt* Nebenklassen-Zerlegung *(standard array). Jedes $\boldsymbol{e} \in \mathcal{M}_\mu$ kann als* Anführer *(coset leader) in der Darstellung $\mathcal{M}_\mu = \boldsymbol{e} + \mathcal{C}$ dienen.*

In jeder Menge $\mathcal{M}_\mu$ wird nun ein Anführer $\boldsymbol{e}_\mu$ minimalen Hamminggewichts festgelegt:

$$\mathcal{M}_\mu = \boldsymbol{e}_\mu + \mathcal{C} \qquad \text{mit} \qquad w_H(\boldsymbol{e}_\mu) \leq w_H(\boldsymbol{e}) \ \text{ für alle } \boldsymbol{e} \in \mathcal{M}_\mu. \tag{4.6.7}$$

Diese Anführer minimalen Gewichts sind nicht notwendigerweise eindeutig bestimmt. Jedoch ist in $\mathcal{M}_0 = \mathcal{C}$ natürlich $\boldsymbol{e}_0 = \boldsymbol{0}$ eindeutig.

Beispiel 4.10. Betrachte den $(5,2)_2$-Code $\mathcal{C} = \{00000, 10110, 01011, 11101\}$ mit

$$\boldsymbol{G} = \left(\begin{array}{cc|ccc} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{array}\right) \quad , \quad \boldsymbol{H} = \left(\begin{array}{cc|ccc} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}\right).$$

Daraus ergibt sich die Nebenklassen-Zerlegung in eindeutiger Weise wie folgt:

μ	e_μ	$\mathcal{M}_\mu$				s_μ
0	00000	00000	10110	01011	11101	000
1	00001	00001	10111	01010	11100	001
2	00010	00010	10100	01001	11111	010
3	00100	00100	10010	01111	11001	100
4	01000	01000	11110	00011	10101	011
5	10000	10000	00110	11011	01101	110
6	11000	11000	01110	10011	00101	101
7	01100	01100	11010	00111	10001	111

$\mathcal{M}_0$ ist der Code. In $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5$ sind die Anführer e_μ minimalen Gewichts jeweils eindeutig bestimmt. In $\mathcal{M}_6$ und $\mathcal{M}_7$ gibt es jeweils zwei Wörter mit minimalem Gewicht, so daß hier die Anführer willkürlich festgelegt werden müssen. Beispielsweise gilt

$$\begin{aligned} 01100 + \mathcal{C} &= \{01100, 11010, 00111, 10001\} \\ &= \{10001, 00111, 11010, 01100\} \;=\; 10001 + \mathcal{C}. \end{aligned}$$

$= M_7$

Es ist leicht nachvollziehbar, daß zu allen Fehlermustern aus $\mathcal{M}_\mu$ das Syndrom s_μ gehört und daß $\mathcal{M}_\mu = e_\mu + \mathcal{C}$ gilt. Für dieses Beispiel wird noch $d_{\min} = 3$, $t = 1$ und $L_t = 1 + n = 6$ (zur Erklärung siehe Satz 4.13) vermerkt. ■

Diese Zerlegung sieht aufwendig aus, und enthält doch nicht mehr Informationen als die kompakte Generatormatrix

$$G = \left(\begin{array}{cc|ccc} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{array}\right)$$

Der Vorteil der Zerlegung wird sich bei der Decodierung erweisen.

Satz 4.12. *Vorausgesetzt wird ein $(n,k)_q$-Code mit den Anführern $\boldsymbol{e}_\mu$ minimalen Gewichts in der Nebenklassen-Zerlegung. Mit dem folgenden Verfahren wird der Maximum-Likelihood-Decoder realisiert:*

Wenn das Empfangswort $\boldsymbol{y}$ in einer Nebenklasse $\mathcal{M}_\mu$ mit dem Anführer $\boldsymbol{e}_\mu$ liegt, dann wird als Schätzung für das gesendete Codewort $\hat{\boldsymbol{a}} = \boldsymbol{y} - \boldsymbol{e}_\mu$ gewählt.

Beweis: Wenn $\boldsymbol{y}$ in $\mathcal{M}_\mu$ liegt, so existiert eine Darstellung $\boldsymbol{y} = \boldsymbol{e}_\mu + \boldsymbol{a}'$ mit $\boldsymbol{a}' \in \mathcal{C}$. Somit folgt $\hat{\boldsymbol{a}} = \boldsymbol{y} - \boldsymbol{e}_\mu = \boldsymbol{a}'$ und damit ist $\hat{\boldsymbol{a}} = \boldsymbol{a}'$ als Codewort nachgewiesen.

Sei nun $\boldsymbol{b}$ ein beliebiges Codewort. Zu zeigen ist $d_H(\boldsymbol{y}, \hat{\boldsymbol{a}}) \leq d_H(\boldsymbol{y}, \boldsymbol{b})$:
Wegen $\hat{\boldsymbol{a}}, \boldsymbol{b} \in \mathcal{C}$ folgt $\hat{\boldsymbol{a}} - \boldsymbol{b} \in \mathcal{C}$ und somit $\boldsymbol{e}_\mu + (\hat{\boldsymbol{a}} - \boldsymbol{b}) \in \mathcal{M}_\mu$. Da $\boldsymbol{e}_\mu$ minimales Gewicht in $\mathcal{M}_\mu$ hat, gilt folglich $w_H(\boldsymbol{e}_\mu) \leq w_H(\boldsymbol{e}_\mu + \hat{\boldsymbol{a}} - \boldsymbol{b})$. Mit $\boldsymbol{e}_\mu = \boldsymbol{y} - \hat{\boldsymbol{a}}$ folgt:

$$d_H(\boldsymbol{y}, \hat{\boldsymbol{a}}) = w_H(\boldsymbol{y} - \hat{\boldsymbol{a}}) \leq w_H(\boldsymbol{y} - \boldsymbol{b}) = d_H(\boldsymbol{y}, \boldsymbol{b}).$$

Also hat $\hat{\boldsymbol{a}}$ von $\boldsymbol{y}$ einen Abstand kleiner oder gleich als jedes andere Codewort und somit wird die ML-Regel realisiert. ■

Die sogenannte *Nebenklassen-Decodierung* kann wie folgt ablaufen: In der Nebenklassen-Zerlegung wird $\boldsymbol{y}$ gesucht und damit ist μ und weiter $\boldsymbol{e}_\mu$ bekannt. Dann ist $\hat{\boldsymbol{a}} = \boldsymbol{y} - \boldsymbol{e}_\mu$ die ML-Schätzung.

Als sogenannte *Syndrom-Decodierung* wird das Verfahren rechnerisch vereinfacht: Es wird eine Tabelle mit q^{n-k} Eintragungen $(\boldsymbol{s}_\mu, \boldsymbol{e}_\mu)$ angelegt. Zum Empfangswort $\boldsymbol{y}$ wird das Syndrom $\boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^T$ berechnet und in der Tabelle wird $\boldsymbol{s}_\mu$ mit $\boldsymbol{s} = \boldsymbol{s}_\mu$ gesucht. Eine weitere Vereinfachung wird mit folgendem Beispiel erklärt:

Beispiel 4.11. (Fortsetzung von Beispiel 4.10) Nachfolgend sind zwei Tabellen angegeben, aus denen $\boldsymbol{e}_\mu$ zu $\boldsymbol{s}_\mu$ abgelesen werden kann. Die linke Tabelle ist lediglich ein Ausschnitt aus der Tabelle von Beispiel 4.10, während die rechte Tabelle eine Umordnung der linken Tabelle ist:

μ	$\boldsymbol{s}_\mu$	$\boldsymbol{e}_\mu$
0	000	00000
1	001	00001
2	010	00010
3	100	00100
4	011	01000
5	110	10000
6	101	11000
7	111	01100

ν	$\boldsymbol{s}_\nu$	$\boldsymbol{e}_\nu$
0	000	00000
1	001	00001
2	010	00010
3	011	01000
4	100	00100
5	101	11000
6	110	10000
7	111	01100

In der rechten Tabelle sind die Syndrome als Dualzahlen durchnumeriert, so daß die Syndrome direkt als Adresse für einen Speicher verwendbar sind, der lediglich die Anführer $\boldsymbol{e}_\nu$ enthält. Damit entfällt die Suche nach $\boldsymbol{s} = \boldsymbol{s}_\mu$. ■

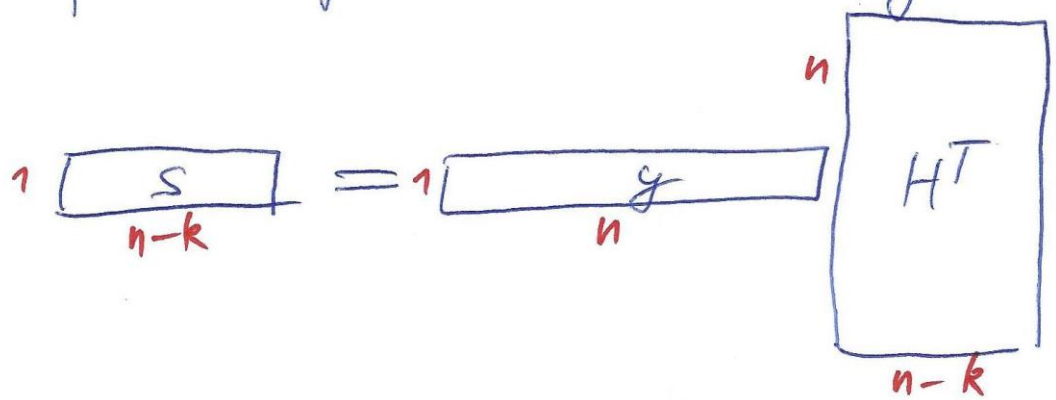
Beispiel: y = 10101

- Nebenklassen-Decod.: y liegt in M_4, also $\mu=4$, $e_4=01000$, also $\hat{a}=11101$
- Syndrom-Decod.: y liefert $s=yH^T=011$, also $\mu=4$ bzw. $v=3$, also $e_\mu=e_v=01000$, also $\hat{a}=11101$

S_v entspricht *v* und braucht deshalb nicht gespeichert zu werden

Die möglichen Mehrdeutigkeiten bei der Wahl der Anführer in den Nebenklassen korrespondieren mit der möglichen Mehrdeutigkeit bei der ML-Decodierung. Bei der BM-Decodierung entfallen diese Mehrdeutigkeiten, da der Decoder nur bei maximal t Fehlern richtig arbeiten muß:

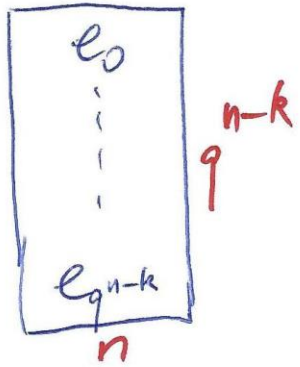
Aufwand Syndrom-Decodierung



erfordert $n(n-k) = n^2(1-R)$ Add/Mult-Ops

d.h. Aufwand $\sim n^2$ für die Syndrom-Berechnung

$\Longrightarrow$ vertretbarer Aufwand



Die Tabelle der Fehlermuster erfordert einen $q^{n-k} \cdot n$ Bit großen Speicher

$\Longrightarrow$ völlig utopischer Aufwand

Definition 5.1. *Ein linearer $(n,k)_q$-Blockcode $\mathcal{C}$ heißt zyklisch, wenn jede zyklische Verschiebung eines Codewortes wieder ein Codewort ist, d.h.*

$$(a_0,\ldots,a_{n-2},a_{n-1})\in\mathcal{C} \quad\Longrightarrow\quad (a_{n-1},a_0,\ldots,a_{n-2})\in\mathcal{C}. \tag{5.1.1}$$

Durch mehrfache zyklische Verschiebung ergibt sich:

$$\begin{aligned}(a_{n-2},a_{n-1},a_0,\ldots,a_{n-4},a_{n-3})&\in\mathcal{C}\\ (a_{n-3},a_{n-2},a_{n-1},a_0,\ldots,a_{n-4})&\in\mathcal{C}\\ &\cdots\cdots\cdots\\ (a_1,\ldots,a_{n-3},a_{n-2},a_{n-1},a_0)&\in\mathcal{C}.\end{aligned}$$

Es ist also unbedeutend, ob in der Definition die Verschiebung nach rechts oder links gefordert wird.

Beispiel 5.1. Betrachte den $(7,4)_2$-Hamming-Code aus den Beispielen 1.2 und 4.1 mit der systematischen Generatormatrix

$$\boldsymbol{G} = \left(\begin{array}{cccc|ccc} 1&0&0&0&0&1&1 \\ 0&1&0&0&1&0&1 \\ 0&0&1&0&1&1&0 \\ 0&0&0&1&1&1&1 \end{array}\right).$$

Offensichtlich ist dieser Code nicht zyklisch, denn 0001111 ist ein Codewort, aber die zyklische Verschiebung 1111000 ist kein Codewort. Mit einer Vertauschung von Spalten ergibt sich ein äquivalenter Code mit $\boldsymbol{G}_2$:

$$\boldsymbol{G}_2 = \begin{pmatrix} 1&1&0&1&0&0&0 \\ 0&1&1&0&1&0&0 \\ 0&0&1&1&0&1&0 \\ 0&1&1&1&0&0&1 \end{pmatrix}, \quad \boldsymbol{G}_3 = \begin{pmatrix} 1&1&0&1&0&0&0 \\ 0&1&1&0&1&0&0 \\ 0&0&1&1&0&1&0 \\ 0&0&0&1&1&0&1 \end{pmatrix}.$$

Durch die elementaren Zeilenoperationen (addiere Zeile 2 zu Zeile 4) ergibt sich ein identischer Code mit $\boldsymbol{G}_3$. Dieser Code ist zyklisch, wie schnell einsehbar ist: Zunächst gehen die Zeilen der Generatormatrix durch zyklische Verschiebungen auseinander hervor. Die Verschiebung der letzten Zeile ergibt 1000110 und dies ist ein Codewort zum Infowort 1110. Mit $\boldsymbol{G}_3$ ergibt sich nebenstehender Encoder

Das Nullwort und das Einswort verändern sich nicht bei zyklischer Verschiebung. Die folgenden 7 Wörter gehen durch zyklische Verschiebung ineinander über, wobei natürlich das Hamminggewicht 3 immer konstant bleibt. Das gleiche gilt für die letzten 7 Wörter vom Hamminggewicht 4. ■

$\boldsymbol{u}$	$\boldsymbol{a}$
0000	0000000
1011	1111111
1000	1101000
0100	0110100
0010	0011010
0001	0001101
1110	1000110
0111	0100011
1101	1010001
1010	1110010
0101	0111001
1100	1011100
0110	0101110
0011	0010111
1111	1001011
1001	1100101

Im nächsten Abschnitt wird gezeigt, daß alle zyklischen Codes eine Generatormatrix mit Bandstruktur besitzen. Die Umkehrung gilt jedoch nicht, wie das Beispiel

$$\boldsymbol{G} = \begin{pmatrix} 1\,1\,0\,1\,0 \\ 0\,1\,1\,0\,1 \end{pmatrix}$$

mit $\mathcal{C} = \{00000, 11010, 01101, 10111\}$ zeigt. Aus der Generatormatrix ist nicht direkt ablesbar, ob der Code zyklisch ist.

Zur Beschreibung zyklischer Codes sind Generatormatrizen wenig geeignet, weil Bandmatrizen nur umständlich zu handhaben sind und weil das vorangehende Beispiel schon andeutet, daß allein schon eine Zeile der Generatormatrix charakterisierend für den Code ist. Generell werden zyklische Codes nicht mit Vektoren und Matrizen, sondern mit Polynomen beschrieben:

Definition 5.2. *Ein Vektor wird mit einem Polynom wie folgt identifiziert:*

$$\begin{array}{rcll} \boldsymbol{a} & = & (a_0, \ldots, a_{n-1}) & \in \mathbb{F}_q^n \\ & \updownarrow & & \\ a(x) & = & \sum_{i=0}^{n-1} a_i x^i & \in \mathbb{F}_q[x]_{n-1}. \end{array}$$

Abkürzend wird auch $w_H(a(x))$ für $w_H(\boldsymbol{a})$ geschrieben. Mit $\mathbb{F}_q[x]$ wird die Menge aller Polynome beliebigen Grades und mit $\mathbb{F}_q[x]_r$ wird die Menge aller Polynome vom Grad kleiner oder gleich r bezeichnet, wobei die Koeffizienten jeweils aus $\mathbb{F}_q$ sind.

Die Größe x (nicht zu verwechseln mit dem Input des diskreten Kanals) hat dabei lediglich die Bedeutung eines Platzhalters – stattdessen könnte auch y oder z^{-1} oder D geschrieben werden. Für Wörter der Länge n gilt für $\boldsymbol{a} \leftrightarrow a(x)$ beispielsweise:

$$\begin{aligned} 0000\ldots0 &\leftrightarrow 0 \\ 1000\ldots0 &\leftrightarrow 1 \\ 0100\ldots0 &\leftrightarrow x \\ 0010\ldots0 &\leftrightarrow x^2 \\ 000\ldots01 &\leftrightarrow x^{n-1} \\ 1111\ldots1 &\leftrightarrow 1 + x + x^2 + \cdots + x^{n-1} = \frac{x^n - 1}{x - 1}. \end{aligned}$$

Zwar existiert ein Polynom $1/(x-1)$ nicht, aber die letzte Gleichung ist so zu verstehen, daß gilt:

$$(x-1)(1 + x + x^2 + \cdots + x^{n-1}) = x^n - 1. \tag{5.1.2}$$

Ein Polynom, bei dem der höchste Koeffizient ungleich Null den Wert 1 hat, wird als *normiertes Polynom* bezeichnet. Allgemein gilt die *Gradformel*:

$$\text{Grad}\big(a(x)b(x)\big) = \text{Grad } a(x) + \text{Grad } b(x). \qquad \text{(A.6.1)}$$

Dabei wird festgelegt: Skalare ungleich Null haben den Grad 0 und die Null hat den Grad $-\infty$.

Die Beschreibung durch Polynome ist prinzipiell eindeutig, d.h. es gilt $\boldsymbol{a} = \boldsymbol{b}$ genau dann, wenn $a(x) = b(x)$ gilt. Der Addition von Vektoren bzw. Wörtern entspricht die Addition von Polynomen. Die Codemenge $\mathcal{C}$ bei einem $(n,k)_q$-Code ist eine Teilmenge bzw. Untervektorraum von $\mathbb{F}_q[x]_{n-1}$ und die Menge aller Infowörter $\mathbb{F}_q^k$ entspricht genau $\mathbb{F}_q[x]_{k-1}$. Natürlich ist $\mathcal{C}$ als Menge von Polynomen nicht multiplikativ abgeschlossen, da bei der Multiplikation von Polynomen vom Grad $\leq n-1$ höhere Grade entstehen.

Satz A.4 (Divisionstheorem). *Zu zwei vorgegebenen Polynomen $b(x)$ und $g(x) \neq 0$ aus $\mathbb{K}[x]$ existieren eindeutig bestimmte Polynome $\alpha(x)$ und $r(x)$ aus $\mathbb{K}[x]$ mit*

Setze K[x] = F_q[x]

$$b(x) = \alpha(x)g(x) + r(x) \quad mit \quad \text{Grad } r(x) < \text{Grad } g(x) \tag{A.6.4}$$

Dividend = Quotient · Divisor + Rest.

Für den Rest $r(x)$ *bei Division von $b(x)$ durch $g(x)$ wird folgende Schreibweise verwendet (*Restklassenbildung*):*

$$r(x) = b(x) \text{ modulo } g(x) \quad , \quad r(x) = R_{g(x)}[b(x)]. \tag{A.6.5}$$

Dabei ist Grad $R_{g(x)}[b(x)] <$ Grad $g(x)$. *Rechnen modulo $g(x)$ bedeutet, daß $g(x)$ durch Null ersetzt werden kann. Das Polynom $\alpha(x)$ ist normalerweise von untergeordneter Bedeutung. (Die eckigen Klammern sind nicht zu verwechseln mit der Bildung der Äquivalenzklassen.)*

Beispiel A.9. Divisionsverfahren in $\mathbb{F}_2[x]$ mit $b(x) = x^7$, $g(x) = x^3 + x + 1$:

$$
\begin{array}{rcl}
x^7 & : & (x^3 + x + 1) \quad = \quad x^4 + x^2 + x + 1 \quad = \quad \alpha(x) \\
\underline{x^7 + x^5 + x^4} & & \\
x^5 + x^4 & & \\
\underline{x^5 + x^3 + x^2} & & \\
x^4 + x^3 + x^2 & & \\
\underline{x^4 + x^2 + x} & & \\
x^3 + x & & \\
\underline{x^3 + x + 1} & & \\
1 & = & r(x).
\end{array}
$$

Also gilt $x^7 = \alpha(x)(x^3+x+1)+1$ bzw. $x^7+1 = \alpha(x)g(x)$ sowie $R_{x^3+x+1}[x^7] = 1$. Rechnen modulo x^3+x+1 bedeutet, daß $x^3+x+1 = 0$ bzw. $x^3 = x+1$ gesetzt werden kann:

$$
\begin{aligned}
x^7 = x^3x^3x \;&=\; (x+1)(x+1)x \;=\; (x^2+2x+1)x \\
= x^3 + x \;&=\; (x+1) + x \;=\; 1 \quad \text{modulo } g(x).
\end{aligned}
$$

Damit ergibt sich $r(x)$ schneller als beim Divisionsverfahren – allerdings ohne das Polynom $\alpha(x)$. ■

Satz A.5 (Restklassen-Arithmetik). *Für die Restklassen-Arithmetik gelten folgende Regeln in* $\mathbb{K}[x]$*:* Setze K[x] = F_q[x]

$$R_{g(x)}[a(x)+b(x)] = R_{g(x)}[a(x)] + R_{g(x)}[b(x)] \tag{A.6.6}$$

$$R_{g(x)}[a(x)\cdot b(x)] = R_{g(x)}\Big[R_{g(x)}[a(x)]\cdot R_{g(x)}[b(x)]\Big] \tag{A.6.7}$$

$$R_{g(x)}[a(x)g(x)] = 0 \tag{A.6.8}$$

$$R_{g(x)}[a(x)] = R_{g(x)}\Big[R_{g(x)h(x)}[a(x)]\Big] \tag{A.6.9}$$

$$\text{Grad } a(x) < \text{Grad } g(x) \quad \Longrightarrow \quad R_{g(x)}[a(x)] = a(x) \tag{A.6.10}$$

$$R_{x^n-1}[x^m] = x^{m \text{ modulo } n} = x^{R_n[m]}. \tag{A.6.11}$$

Beim Rechnen modulo (x^n-1) *wird* x^n *durch 1 ersetzt bzw. die Potenz* m *durch* m *modulo* n*. Die Potenzrechnung erfolgt dabei in* $\mathbb{Z}$ *unabhängig von* $\mathbb{K}$*.*

Als erster Vorteil der Polynombeschreibung ergibt sich eine sehr kompakte Beschreibung der zyklischen Verschiebung mit Hilfe von $x^n - 1$:

Satz 5.1. *Die m-fache zyklische Verschiebung von* $(a_0, a_1, \ldots, a_{n-1}) \leftrightarrow a(x)$ *ergibt* $(a_{n-m}, \ldots, a_{n-1}, a_0, \ldots, a_{n-m-1}) \leftrightarrow R_{x^n-1}[x^m a(x)]$.

Beweis:

$$
\begin{aligned}
& R_{x^n-1}[x^m a(x)] \\
&= R_{x^n-1}[a_0 x^m + \cdots + a_{n-m-1} x^{n-1} + a_{n-m} x^n + \cdots + a_{n-1} x^{n-1+m}] \\
&= a_0 x^m + \cdots + a_{n-m-1} x^{n-1} + a_{n-m} x^0 + \cdots + a_{n-1} x^{m-1}.
\end{aligned}
$$

Dies entspricht dem zyklisch verschobenen Wort. ■

Definition 5.3. *Mit dem Generatorpolynom $g(x) \in \mathbb{F}_q[x]_{n-k}$ vom Grad $n-k$ über $\mathbb{F}_q$ wird ein linearer $(n,k)_q$-Blockcode wie folgt erzeugt:*

$$\mathcal{C} = \left\{ u(x)g(x) \;\middle|\; u(x) \in \mathbb{F}_q[x]_{k-1} \right\} \tag{5.2.1}$$

$$= \left\{ a(x) \in \mathbb{F}_q[x]_{n-1} \;\middle|\; R_{g(x)}[a(x)] = 0 \right\}. \tag{5.2.2}$$

Das Generatorpolynom $g(x) = g_0 + g_1 x + \ldots + g_{n-k-1}x^{n-k-1} + x^{n-k}$ wird stets als normiert vorausgesetzt. Mit dem Generatorpolynom wird ein Encoder in folgender Form ermöglicht:

$$u(x) \quad \mapsto \quad a(x) = u(x)g(x). \tag{5.2.3}$$

Für $u(x) = 1$ folgt $g(x) \in \mathcal{C}$ und für $u(x) = 0$ folgt $0 = 0(x) \in \mathcal{C}$. Für $u(x) \neq 0$ gilt Grad $a(x) \geq n-k$. Also existiert außer dem Nullwort bzw. Nullpolynom kein Codewort vom Grad $< n-k$:

$$\mathbb{F}_q[x]_{n-k-1} \cap \mathcal{C} = \{0\}. \tag{5.2.4}$$

Offensichtlich gilt auch die Abschätzung $d_{\min} \leq w_H(g(x))$, was direkt der Singleton-Schranke aus Satz 3.7 entspricht.

Das Generatorpolynom nach Definition 5.3 ist nur von der Anzahl $n-k$ der Prüfstellen abhängig und damit wird ein linearer Code erzeugt. Um einen zyklischen Code zu erzeugen, muß noch eine Abhängigkeit von n hinzukommen. Der folgende Satz gibt dazu ein leicht nachprüfbares Kriterium an:

Satz 5.2. *Es sei $g(x)$ ein Generatorpolynom vom Grad $n-k$ für einen $(n,k)_q$-Code $\mathcal{C}$. Dann gilt:*

$$\mathcal{C} \text{ ist zyklisch} \iff g(x) \text{ ist ein Teiler von } x^n - 1. \tag{5.2.5}$$

Bei einem zyklischen Code ist durch Vorgabe von $\mathcal{C}$ das normierte Generatorpolynom eindeutig bestimmt.

Ein zyklischer Code wird einerseits durch ein normiertes Generatorpolynom vom Grad $n-k$ mit $n-k$ Koeffizienten und andererseits durch eine Generatormatrix mit $k \cdot n$ Koeffizienten beschrieben. Den Zusammenhang gibt folgender Satz:

Satz 5.3. *Der zyklische $(n,k)_q$-Code werde erzeugt durch das Generatorpolynom $g(x) = g_0 + g_1 x + \ldots + g_{n-k-1}x^{n-k-1} + x^{n-k}$. Dann wird der Code auch erzeugt durch die Generatormatrix $\boldsymbol{G} \in \mathbb{F}_q^{k,n}$:*

$$\boldsymbol{G} = \begin{pmatrix} g_0 \ldots\ldots g_{n-k} & & \\ & \ddots & \\ & & g_0 \ldots\ldots g_{n-k} \end{pmatrix} \leftrightarrow \begin{pmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{pmatrix}. \qquad (5.2.7)$$

Die Polynom-Multiplikation
a(x)=u(x)g(x)
bedeutet Faltung der Polynom-Koeffizienten

Die Matrix-Multiplikation
a=uG
mit dem speziellen Aufbau von G bedeutet ebenfalls Faltung der Koeffizienten

Beweis: Sei $u(x) \leftrightarrow (u_0, \ldots, u_{k-1})$ und $a(x) \leftrightarrow (a_0, \ldots, a_{n-1})$. Dann entspricht die Polynom-Multiplikation $a(x) = u(x)g(x)$ der Koeffizienten-Faltung

$$a_i = \sum_{\nu=\max\{0,i-n+k\}}^{\min\{k-1,i\}} u_\nu g_{i-\nu}$$

und dies entspricht genau der Matrix-Multiplikation $\boldsymbol{a} = \boldsymbol{u}\boldsymbol{G}$. Die Darstellung von $\boldsymbol{G}$ als Polynomvektor ist offensichtlich. ■

Zunächst wird an die Matrixbeschreibung eines linearen Codes mit der Generatormatrix $\boldsymbol{G} \in \mathbb{F}_q^{k,n}$ und der Prüfmatrix $\boldsymbol{H} \in \mathbb{F}_q^{n-k,n}$ mit $\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}$ erinnert:

$$\mathcal{C} = \left\{ \boldsymbol{u}\boldsymbol{G} \;\middle|\; \boldsymbol{u} \in \mathbb{F}_q^k \right\} = \left\{ \boldsymbol{a} \in \mathbb{F}_q^n \;\middle|\; \boldsymbol{a}\boldsymbol{H}^T = \boldsymbol{0} \right\}.$$

Entsprechend gilt für die Polynombeschreibung:

Satz 5.4. *Durch das Generatorpolynom $g(x)$ vom Grad $n-k$ werde der zyklische $(n,k)_q$-Code $\mathcal{C}$ erzeugt. Da $g(x)$ ein Teiler von $x^n - 1$ ist, existiert ein* Prüfpolynom *(parity check polynomial)* $h(x) \in \mathbb{F}_q[x]_k$ *vom Grad k, so daß*

$$g(x)h(x) = x^n - 1 \tag{5.3.1}$$

gilt. Mit $g(x)$ ist auch $h(x)$ normiert. Dann ist der Code eindeutig durch $h(x)$ wie folgt charakterisiert:

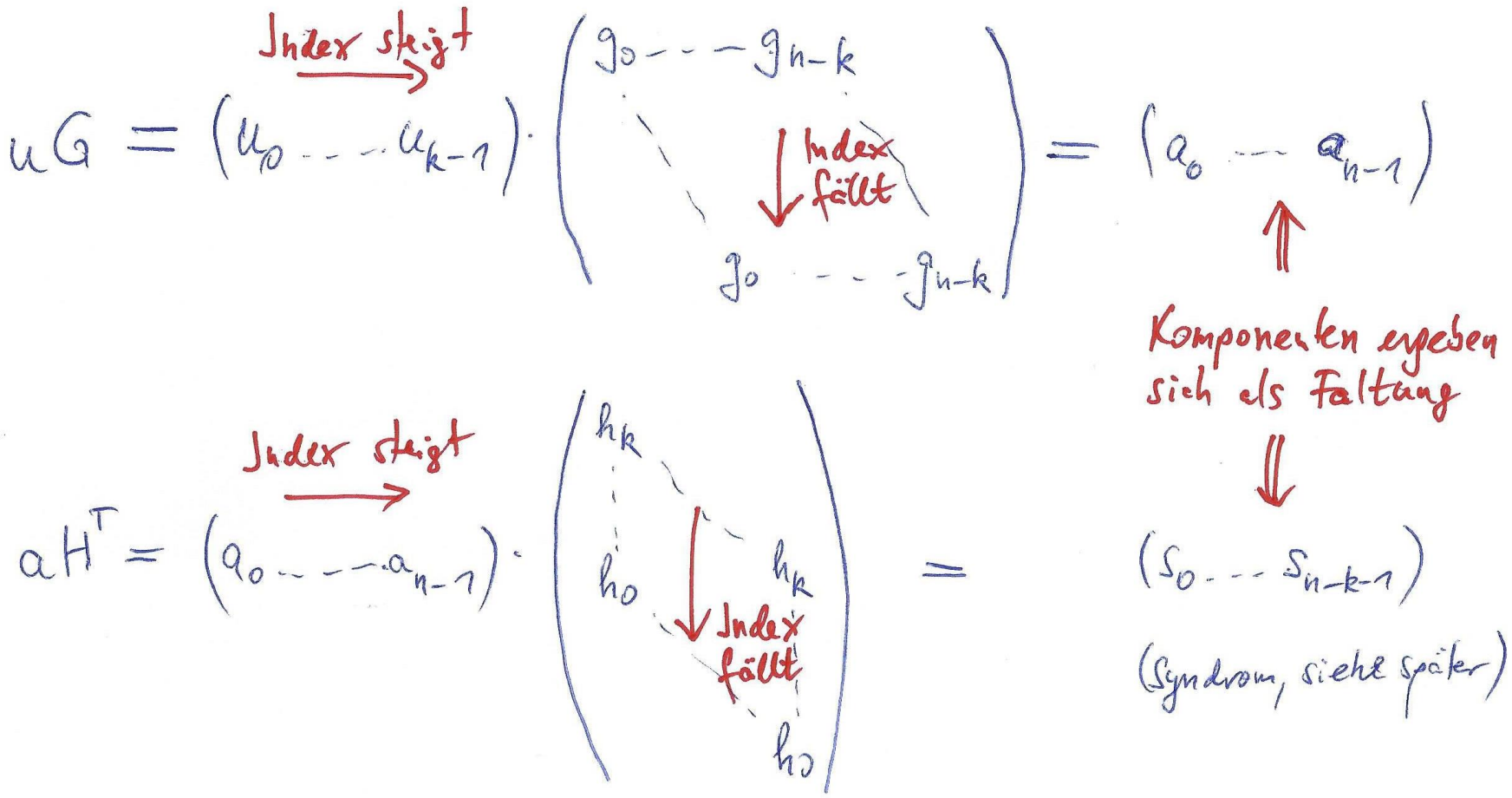
$$\mathcal{C} = \left\{ a(x) \in \mathbb{F}_q[x]_{n-1} \;\middle|\; R_{x^n-1}[a(x)h(x)] = 0 \right\}. \tag{5.3.2}$$

Satz 5.5. *Unter den Voraussetzungen des vorangehenden Satzes mit einem Prüfpolynom* $h(x) = h_0 + h_1 x + \ldots + h_{k-1}x^{k-1} + x^k$ *wird eine Prüfmatrix* $\boldsymbol{H} \in \mathbb{F}_q^{n-k,n}$ *gegeben durch:*

$$\boldsymbol{H} = \begin{pmatrix} h_k \ldots\ldots h_0 & & \\ & h_k \ldots\ldots h_0 & \\ & & h_k \ldots\ldots h_0 \end{pmatrix} \leftrightarrow \begin{pmatrix} \bar{h}(x) \\ x\bar{h}(x) \\ \\ \vdots \\ \\ x^{n-k-1}\bar{h}(x) \end{pmatrix}. \tag{5.3.3}$$

Dabei ist $\bar{h}(x) = x^k h(x^{-1}) = 1 + h_{k-1}x + \ldots + h_0 x^k$ *das reziproke Polynom zu* $h(x)$ *(siehe auch (A.6.3)). Natürlich ist die Reihenfolge der Zeilen in* $\boldsymbol{H}$ *wie in* $\boldsymbol{G}$ *beliebig.*

Polynom-Multiplikation = Faltung der Koeffizienten = Matrizen-Multiplikation mit Bandstruktur



Beispiel 5.2. **(1)** Für den $(n,1)_2$-Wiederholungscode gilt

$$g(x) = 1 + x + \cdots + x^{n-1} = \frac{x^n - 1}{x - 1} = \frac{x^n + 1}{x + 1},$$

denn aus $u(x) = u_0$ folgt $a(x) = u(x)g(x) \leftrightarrow (u_0, \ldots, u_0)$. Daraus ergibt sich $h(x) = x - 1 = x + 1$ und weiter ist

$$\begin{aligned} 0 = R_{x^n-1}[a(x)h(x)] &= R_{x^n-1}[a(x)x - a(x)] \\ &= R_{x^n-1}[a_0x + \cdots + a_{n-1}x^n - a_0 - a_1x - \cdots - a_{n-1}x^{n-1}] \\ &= R_{x^n-1}[-a_0 + (a_0 - a_1)x + \cdots + (a_{n-2} - a_{n-1})x^{n-1} + a_{n-1}x^n] \\ &= (a_{n-1} - a_0) + (a_0 - a_1)x + \cdots + (a_{n-2} - a_{n-1})x^{n-1} \end{aligned}$$

äquivalent mit $a_{n-1} = a_0, a_0 = a_1, a_1 = a_2, \ldots, a_{n-2} = a_{n-1}$ bzw. $a_0 = a_1 = \cdots = a_{n-1}$. Nach (5.2.7) und (5.3.3) gilt:

$$\boldsymbol{G} = \begin{pmatrix} 1 & 1 & \ldots & 1 & 1 \end{pmatrix} \quad , \quad \boldsymbol{H} = \begin{pmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & & & & \\ & & & 1 & 1 & \\ & & & & 1 & 1 \end{pmatrix}.$$

Durch die elementaren Zeilenoperationen ergibt sich die in Beispiel 4.3(2) angegebene Form von $\boldsymbol{H}$.

(2) Mit ähnlichen Methoden ergibt sich für den $(n, n-1)_2$-Parity Check Code $g(x) = x - 1$ und $h(x) = (x^n - 1)/(x - 1)$. ■

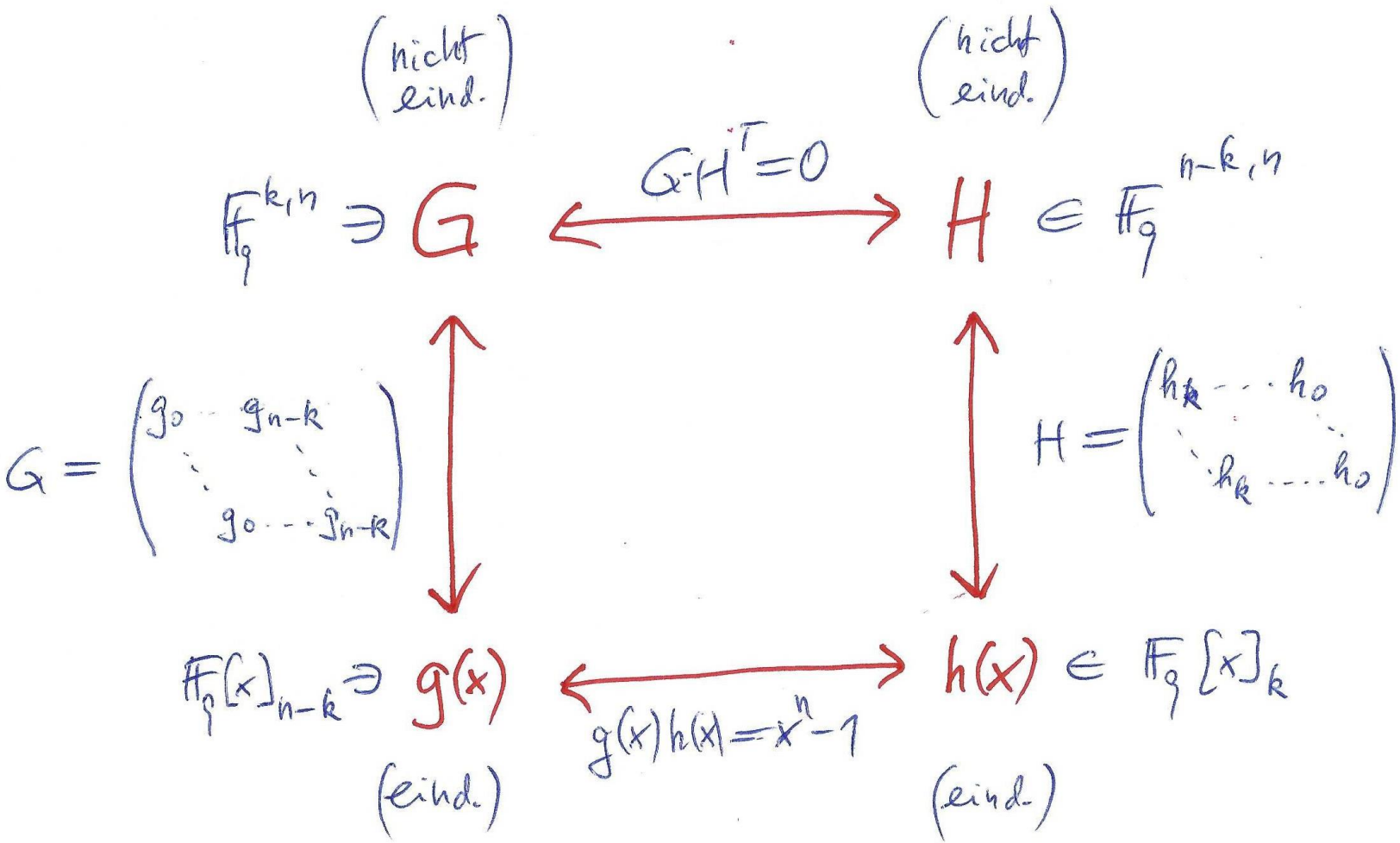
Beispiel 5.3. Sei $g(x) = 1+x+x^3$ und $h(x) = 1+x+x^2+x^4$. Wegen $g(x)h(x) = x^7 - 1$ in $\mathbb{F}_2$ sind dies Generator- und Prüfpolynom für einen zyklischen $(7,4)_2$-Code. Nach (5.2.7) und (5.3.3) gilt mit $\bar{h}(x) = 1 + x^2 + x^3 + x^4$:

$$\boldsymbol{G} = \boldsymbol{G}_3 = \begin{pmatrix} 1\,1\,0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,1\,0\,0 \\ 0\,0\,1\,1\,0\,1\,0 \\ 0\,0\,0\,1\,1\,0\,1 \end{pmatrix} \leftrightarrow \begin{pmatrix} 1+x+x^3 \\ x+x^2+x^4 \\ x^2+x^3+x^5 \\ x^3+x^4+x^6 \end{pmatrix} = \begin{pmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ x^3g(x) \end{pmatrix},$$

$$\boldsymbol{H} = \begin{pmatrix} 1\,0\,1\,1\,1\,0\,0 \\ 0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,1\,1\,1 \end{pmatrix} \leftrightarrow \begin{pmatrix} 1+x^2+x^3+x^4 \\ x+x^3+x^4+x^5 \\ x^2+x^4+x^5+x^6 \end{pmatrix} = \begin{pmatrix} \bar{h}(x) \\ x\bar{h}(x) \\ x^2\bar{h}(x) \end{pmatrix}.$$

$\boldsymbol{G}$ entspricht genau der Generatormatrix $\boldsymbol{G}_3$ der zyklischen Version des Hamming-Codes aus Beispiel 5.1. Man kann $\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}$ verifizieren. $\boldsymbol{H}$ entsteht natürlich durch Spaltenpermutation aus der Prüfmatrix von Beispiel 4.3(1). Durch die elementaren Zeilenoperationen ergibt sich aus $\boldsymbol{G}_3$ ein systematischer zyklischer Code mit:

$$\boldsymbol{G}_4 = \begin{pmatrix} 1\,0\,0\,0 & 1\,1\,0 \\ 0\,1\,0\,0 & 0\,1\,1 \\ 0\,0\,1\,0 & 1\,1\,1 \\ 0\,0\,0\,1 & 1\,0\,1 \end{pmatrix}.$$


(nicht eind.)
(nicht eind.)
$\mathbb{F}_q^{k,n} \ni G$
$G \cdot H^T = 0$
$H \in \mathbb{F}_q^{n-k,n}$
$G = \begin{pmatrix} g_0 & & g_{n-k} \\ & \ddots & \\ g_0 & & g_{n-k} \end{pmatrix}$
$H = \begin{pmatrix} h_k & \cdots & h_0 & & \\ & \ddots & & \ddots & \\ & & h_k & \cdots & h_0 \end{pmatrix}$
$\mathbb{F}_q[x]_{n-k} \ni g(x)$
$h(x) \in \mathbb{F}_q[x]_k$
$g(x)h(x) = x^n - 1$
(eind.)
(eind.)

Jeder zyklische Code kann systematisch encodiert werden (denn der zyklische Code hat eine Generatormatrix die in systematische Form überführt werden kann wobei der Code identisch bleibt).
Die direkte Encodierung gemäß a(x)=u(x)g(x) erzeugt einen nicht-systematischen Code und wird deshalb nicht verwendet.

Satz 5.7 (Systematische Encodierung mit dem Generatorpolynom).
Es sei $g(x) = g_0 + g_1 x + \cdots + g_{n-k-1}x^{n-k-1} + x^{n-k}$ *ein Generatorpolynom für einen zyklischen* $(n,k)_q$*-Code. Zum Infowort* $\boldsymbol{u} = (u_0, \ldots, u_{k-1}) \leftrightarrow u(x)$ *wird als Prüfwort*

$$\boldsymbol{p} = (p_0, \ldots, p_{n-k-1}) \quad \leftrightarrow \quad p(x) = R_{g(x)}[-x^{n-k}u(x)] \qquad (5.4.1)$$

bestimmt und als Codewort wird verwendet:

$$\boldsymbol{a} = (\underbrace{p_0, \ldots, p_{n-k-1}}_{n-k \text{ Prüfstellen}}, \underbrace{u_0, \ldots, u_{k-1}}_{k \text{ Infostellen}}) \quad \leftrightarrow \quad a(x) = p(x) + x^{n-k}u(x). \qquad (5.4.2)$$

Beweis: Zu zeigen ist $a(x) \in \mathcal{C}$. Aus (5.4.1) und (5.4.2) folgt:

$$\begin{aligned} R_{g(x)}[a(x)] &= R_{g(x)}[R_{g(x)}[-x^{n-k}u(x)] + x^{n-k}u(x)] \\ &= R_{g(x)}[-x^{n-k}u(x) + x^{n-k}u(x)] \;=\; 0. \end{aligned}$$

Damit ist $a(x)$ als Vielfaches von $g(x)$ nachgewiesen. ■

Die praktische Realisierung mit einem *linearen rückgekoppelten Schieberegister* (LFSR, linear feedback shift register) zeigt Bild 5.1. Zu Beginn ist das Register mit Nullen vorbelegt: $r_m^{(0)} = 0$. Der Reihe nach werden $u_{k-1}, \ldots, u_0$ eingeschoben. Danach enthält das Register die Prüfstellen: $r_m^{(k)} = p_m$.

Aufwand: k = #Shifts, n-k = Registerlänge
Insgesamt also $k(n-k)=n^2R(1-R)$ Add-Mult-Ops.

Bei q=2 entfallen die Multiplikationen
(Leitung vorhanden oder nicht)

Die Additionen sind parallelisierbar

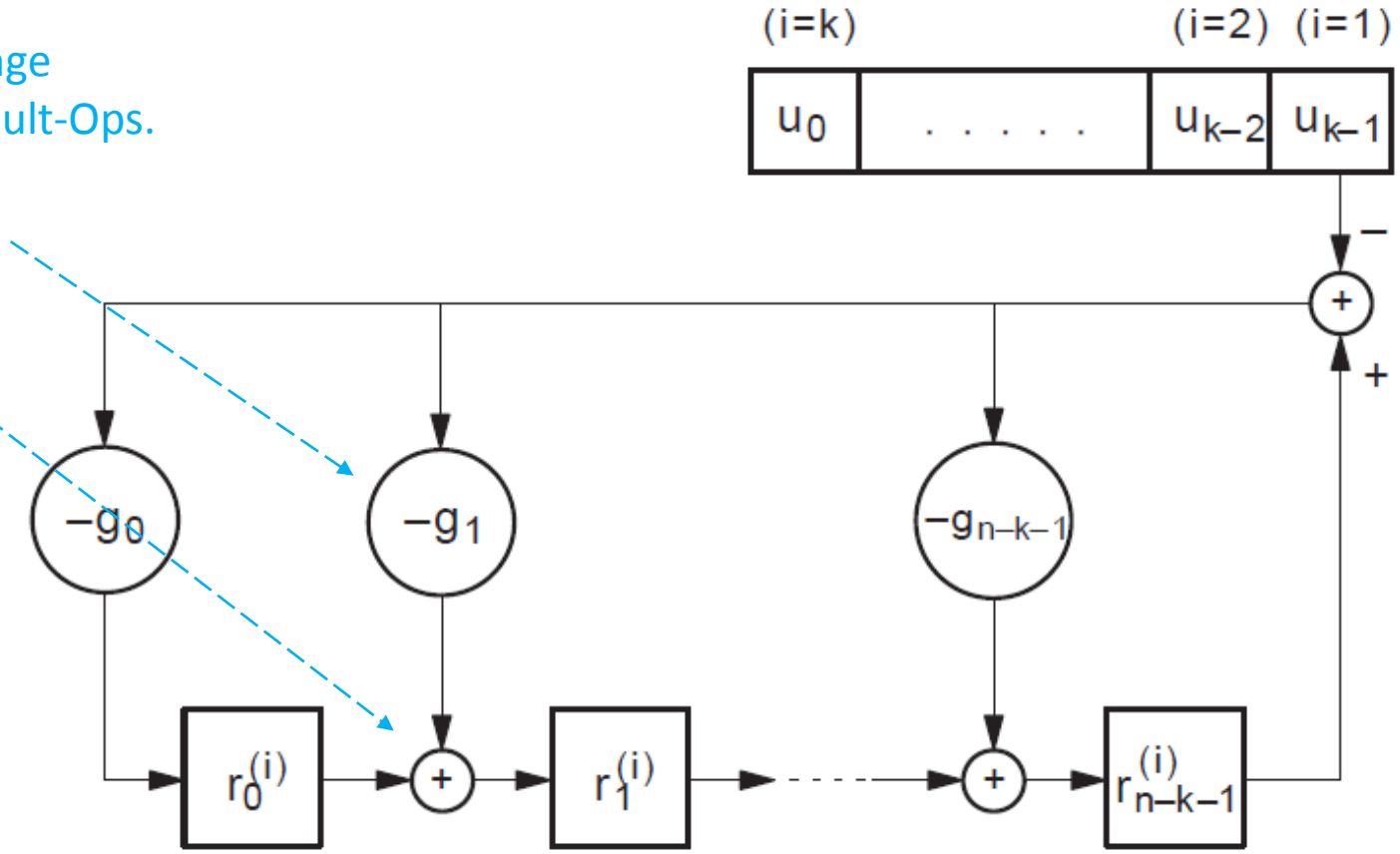


Bild 5.1. Systematische Encodierung mit $g(x)$

Beispiel 5.4. Betrachte den $(7,4)_2$-Hamming-Code mit dem Generatorpolynom $g(x) = 1 + x + x^3$. Sei $u(x) = 1 + x^2 + x^3 \leftrightarrow 1011$. Dann gilt

$$p(x) = R_{g(x)}[x^{n-k}u(x)] = R_{x^3+x+1}[x^6 + x^5 + x^3] = 1,$$

was auf verschiedene Arten berechnet werden kann. Erstens kann das Divisionsverfahren angewendet werden. Zweitens ist eine direkte Berechnung wie folgt möglich:

$$\begin{aligned}
R_{g(x)}[x^3] &= x + 1 \\
R_{g(x)}[x^4] &= R_{g(x)}[x^2 + x] = x^2 + x \\
R_{g(x)}[x^5] &= R_{g(x)}[x^3 + x^2] = x^2 + x + 1 \\
R_{g(x)}[x^6] &= R_{g(x)}[x^3 + x^2 + x] = (x + 1) + (x^2 + x) = x^2 + 1 \\
R_{g(x)}[x^6 + x^5 + x^3] &= (x^2 + 1) + (x^2 + x + 1) + (x + 1) = 1.
\end{aligned}$$

Drittens kann die Berechnung mit dem rückgekoppelten Schieberegister erfolgen, wie es in Bild 5.2 dargestellt ist. Mit

$$a(x) = p(x) + x^3u(x) = 1 + x^3 + x^5 + x^6 \leftrightarrow 1001011$$

ergibt sich dann das Codewort.

i	u_i	$r^{(i)}_{n-k-1}$	$\mathbf{r}^{(i)}$
0			0 0 0
1	1	0	1 1 0
2	1	0	1 0 1
3	0	1	1 0 0
4	1	0	1 0 0 = **p**

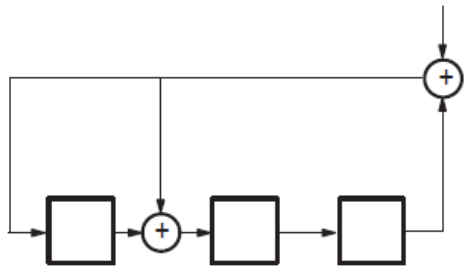


Bild 5.2. Beispiel zur systematischen Encodierung mit $g(x)$

Anstelle des Generatorpolynoms kann auch das Prüfpolynom zur systematischen Encodierung verwendet werden:

Satz 5.8 (Systematische Encodierung mit dem Prüfpolynom). *Es sei* $h(x) = h_0 + h_1 x + \cdots + h_{k-1}x^{k-1} + x^k$ *ein Prüfpolynom für einen zyklischen* $(n,k)_q$*-Code. Dann wird das Codewort wie folgt gewählt:*

$$\boldsymbol{a} = (\underbrace{a_0, \ldots, a_{n-k-1}}_{n-k \text{ Prüfstellen}}, \underbrace{a_{n-k}, \ldots, a_{n-1}}_{k \text{ Infostellen}}). \tag{5.4.3}$$

Die Prüfstellen werden dabei wie folgt bestimmt ($m = n-k-1, n-k-2, \ldots, 0$)*:*

$$a_m = -\sum_{i=0}^{k-1} h_i a_{m-i+k} = \mathit{Funktion}(a_{m+1}, \ldots, a_{m+k}). \tag{5.4.4}$$

Die praktische Realisierung mit einem rückgekoppelten Schieberegister zeigt Bild 5.3. Zu Beginn ($m = n - k - 1$) ist das Register mit dem Infowort $a_{n-k}, \ldots, a_{n-1}$ vorbelegt. Der Reihe nach entstehen am Eingang des Registers die Prüfstellen $a_{n-k-1}, \ldots, a_0$ mit $m = n - k - 1, n - k - 2, \ldots, 0$.

Aufwand: n-k = #Shifts, k = Registerlänge
Insgesamt also (n-k)k=$n^2R(1-R)$ Add-Mult-Ops.

Bei q=2 entfallen die Multiplikationen
(Leitung vorhanden oder nicht)

Die Additionen sind sequentiell auszuführen
(nicht parallelisierbar)

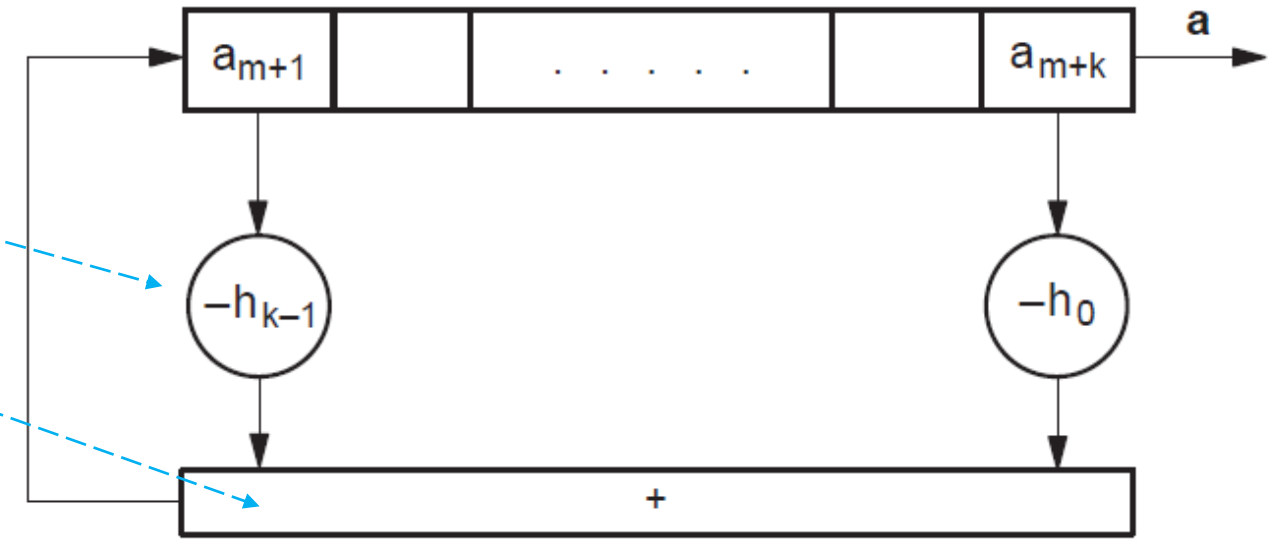


Bild 5.3. Systematische Encodierung mit $h(x)$

Beispiel 5.5. Betrachte wieder den $(7,4)_2$-Hamming-Code mit dem Prüfpolynom $h(x) = 1 + x + x^2 + x^4$. Wie bei Beispiel 5.4 sei $u(x) = 1 + x^2 + x^3$. Im Schieberegister aus Bild 5.4 wird $a_m = a_{m+2} + a_{m+3} + a_{m+4}$ realisiert und damit ergibt sich wieder das gleiche $a(x)$. ■

m	$a_{m+1} \dots a_{m+4}$	a_m
2	1 0 1 1	0
1	0 1 0 1	0
0	0 0 1 0	1

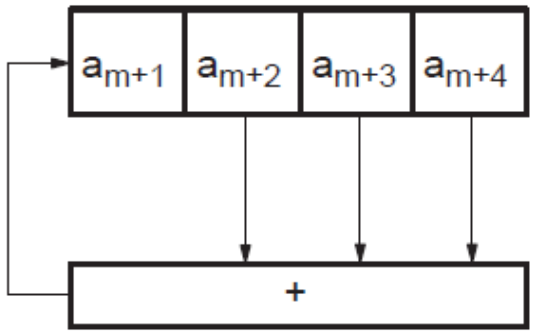


Bild 5.4. Beispiel zur systematischen Encodierung mit $h(x)$

Insgesamt sind jetzt vier verschiedenen Methoden zur Encodierung bekannt:

(1) Matrix-Operation: $\boldsymbol{a} = \boldsymbol{u}\boldsymbol{G}$.

(2) Polynom-Operation: $a(x) = u(x)g(x)$ (nicht-systematisch).

(3) Schieberegister-Implementierung mit $g(x)$.

(4) Schieberegister-Implementierung mit $h(x)$.

Bei (3) hat das Register die Länge $n-k$ mit k Shifts und bei (4) die Länge k mit $n-k$ Shifts. Die Anzahl der Add/Mult-Operationen beträgt also bei (3) und (4) wie auch bei (2) jeweils $(n-k)k = n^2(1-R)R$. Dagegen sind bei (1) $nk = n^2R$ Operationen notwendig. In allen vier Fällen führt eine Änderung in den Codeparametern mindestens zu neuen Koeffizienten in $\boldsymbol{G}$ bzw. $g(x)$ bzw. $h(x)$. Bei Galoisfeldern mit großem q sind Addition und Multiplikation etwa aufwandsgleich und dominieren gegenüber den Shifts. Die Methode (3) hat den Vorteil, daß die jeweils $n-k$ Operationen pro Shift parallelisierbar sind beim Einsatz von $n-k$ Prozessoren. Dagegen muß die Addiererkette bei (4) sequentiell abgearbeitet werden. Insbesondere bei großen Coderaten ist (3) deshalb günstiger als (4) einzustufen. (1) und (2) werden normalerweise nicht verwendet.

Tabelle 5.1. Zusammenfassung Polynome

Grad	Polynom	Name
$n-k$	$g(x)$	Generatorpolynom
k	$h(x) = (x^n - 1)/g(x)$	Prüfpolynom
$\leq k-1$	$u(x)$	Infopolynom
$\leq n-1$	$a(x) = u(x)g(x), R_{x^n-1}[a(x)h(x)] = 0$	Codepolynom
$\leq n-1$	$e(x)$	Fehlerpolynom
$\leq n-1$	$y(x) = a(x) + e(x)$	Empfangspolynom
$\leq n-k-1$	$s(x) = R_{g(x)}[y(x)] = R_{g(x)}[e(x)]$	Syndrompolynom
Normierungen: $g_{n-k} = h_k = 1, \quad g_0 h_0 = 1$		

Die allermeisten Decodierverfahren, sowohl für einfache wie für komplizierte Codes, erfordern die Berechnung des Syndroms. Eigentlich ist das Syndrom hier nicht neu zu definieren, denn über $g(x)$ ist $\boldsymbol{G}$ und $\boldsymbol{H}$ und damit $\boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^T$ bestimmt. Das nachfolgend definierte Syndrompolynom $s(x)$ stimmt mit dem Syndromvektor $\boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^T$ nicht notwendigerweise überein, da die Prüfmatrix $\boldsymbol{H}$ nicht eindeutig bestimmt ist.

Definition 5.4. *Sei $g(x)$ das Generatorpolynom vom Grad $n-k$ eines zyklischen $(n,k)_q$-Codes. Zum Empfangswort $y(x) = a(x) + e(x)$ wird das* Syndrom *des Empfangswortes bzw. Fehlermusters definiert als Polynom vom Grad $\leq n-k-1$:*

$$s(x) = R_{g(x)}[y(x)] = R_{g(x)}[e(x)]. \tag{5.5.1}$$

Das Syndrom ist vom Codewort $a(x) = u(x)g(x)$ unabhängig, denn $R_{g(x)}[a(x)] = 0$ ist stets erfüllt. Das Syndrom ist wie bei Definition 4.7 durch $n-k$ Koeffizienten gekennzeichnet. Natürlich gilt

$$s(x) = 0 \quad \Longleftrightarrow \quad y(x) \in \mathcal{C} \quad \Longleftrightarrow \quad e(x) \in \mathcal{C}. \tag{5.5.2}$$

Wie bei der Encodierung mit dem Generatorpolynom kann das Syndrom durch ein rückgekoppeltes Schieberegister berechnet werden. Das Prinzip zeigt Bild 5.5: Zu Beginn ist das Register mit Nullen vorbelegt: $r_m^{(0)} = 0$. Der Reihe nach werden $y_{n-1}, \ldots, y_0$ eingeschoben. Danach enthält das Register das Syndrom: $r_m^{(n)} = s_m$. Im Gegensatz zur Encodierung wird hier das Register n statt k mal benutzt, so daß $n(n-k) = n^2(1-R)$ Operationen erforderlich sind.

Aufwand: n = #Shifts, n-k = Registerlänge
Insgesamt also $n(n-k)=n^2(1-R)$ Add-Mult-Ops.

Bei q=2 entfallen die Multiplikationen
(Leitung vorhanden oder nicht)

Die Additionen sind parallelisierbar

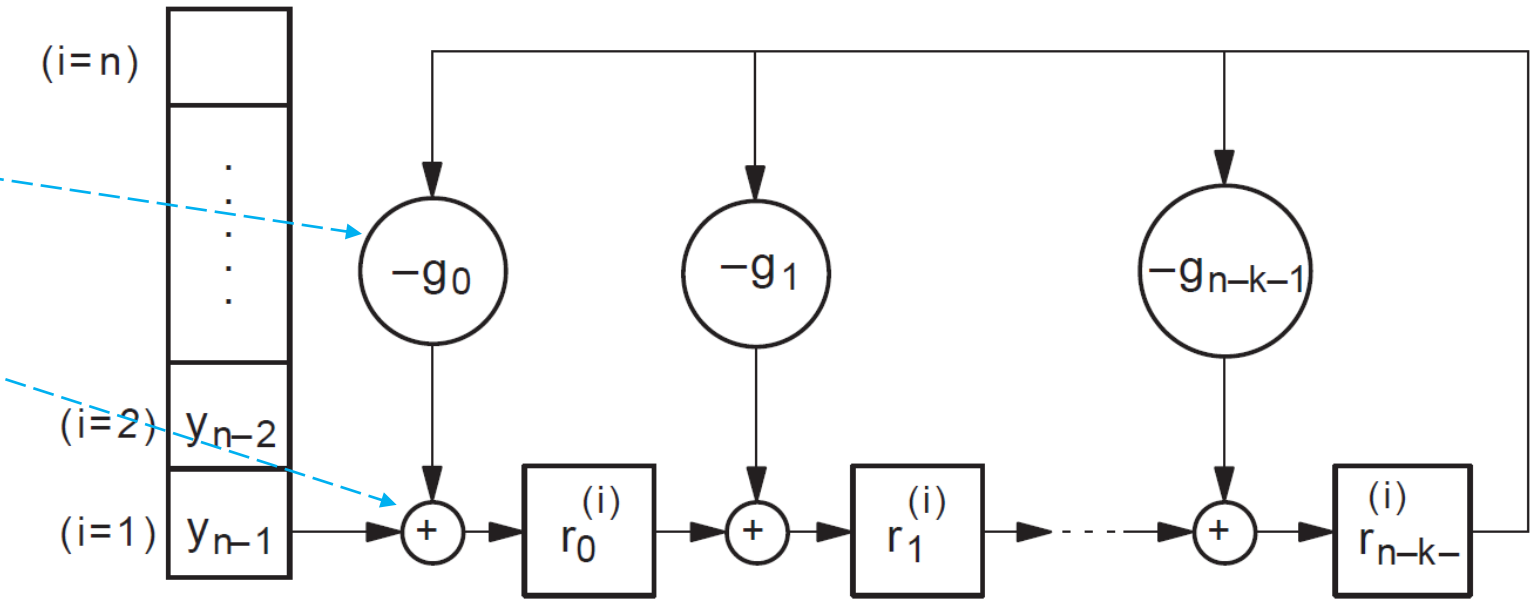


Bild 5.5. Syndrom-Berechnung mit einem rückgekoppelten Schieberegister

Beispiel 5.6. Betrachte wieder den $(7,4)_2$-Hamming-Code mit dem Generatorpolynom $g(x) = 1 + x + x^3$. Nach Beispiel 5.4 ist $a(x) = 1 + x^3 + x^5 + x^6$ ein Codewort. Sei $e(x) = x^6$ und somit $y(x) = 1 + x^3 + x^5 \leftrightarrow 1001010$. Das Syndrom kann durch das Divisionsverfahren oder direkt oder mit dem rückgekoppelten Schieberegister wie in Bild 5.6 berechnet werden: $s(x) = 1 + x^2 \leftrightarrow 101$. ■

i	y_{n-i}	$r^{(i)}_{n-k-1}$	$\mathbf{r}^{(i)}$
0			0 0 0
1	0	0	0 0 0
2	1	0	1 0 0
3	0	0	0 1 0
4	1	0	1 0 1
5	0	1	1 0 0 = 110 + 010 (Rück + Shift)
6	0	0	0 1 0
7	1	0	1 0 1 = **s**

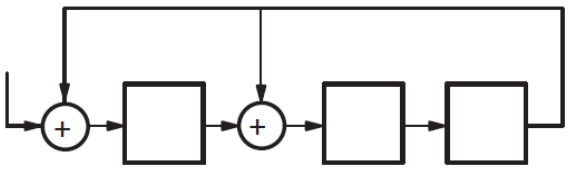


Bild 5.6. Beispiel zur Syndrom-Berechnung

Es wird weiterhin ein diskreter Kanal mit Hard-Decision vorausgesetzt. Bei der Fehlererkennung gibt es prinzipiell nur zwei mögliche Alternativen:

$s(x) = 0$: Das Empfangswort ist ein Codewort, d.h. es liegt entweder eine fehlerfreie Übertragung vor oder das Fehlermuster ist ein Codewort und damit prinzipiell nicht erkennbar.

$s(x) \neq 0$: Das Empfangswort ist kein Codewort und wird damit als fehlerhaft erkannt.

Bisher wurden ausschließlich gedächtnislose Kanäle behandelt, bei denen die Fehler statistisch unabhängig voneinander auftreten, so daß ein Fehlerwort (Fehlermuster) aus einer Reihe von *Einzelfehlern* besteht. Die Anzahl dieser Einzelfehler entspricht dem Hamminggewicht des Fehlerwortes und dieses Gewicht ist binomialverteilt gemäß (1.3.9). Viele reale Kanäle sind jedoch gedächtnisbehaftet und erzeugen *Bündelfehler* anstelle von Einzelfehlern. Sowohl zur Erkennung wie zur Korrektur von Bündelfehlern sind zyklische Codes sehr gut geeignet.

Kanaltyp	**Statistische Beschreibung**	**Leistungsfähigkeit des Codes wird erfasst durch**
Einzelfehler = gedächtnislos	statistisch unabhängig, binomialverteilt	Fehlerwahrscheinlichkeit, Codierungsgewinn
Bündelfehler = gedächtnisbehaftet	keine	Länge und Anzahl der erkennbaren (korrigierbaren) Bündelfehler, kein P_b oder P_w

Definition 5.5. *Ein Bündelfehler (burst error) $e(x)$ der Länge t ist dadurch gekennzeichnet, daß höchstens t aufeinanderfolgende Stellen ungleich Null sind, d.h.*

$$e(x) = x^i b(x) \quad mit \quad \text{Grad } b(x) < t,\ 0 \le i \le n - t. \tag{5.6.1}$$

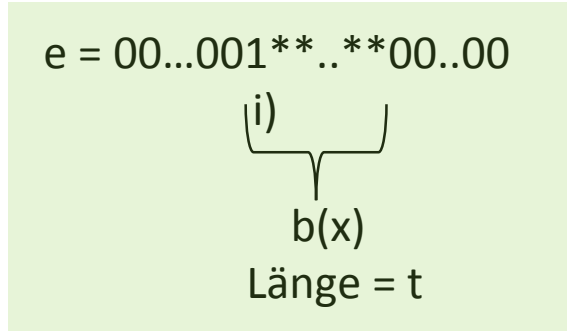
Durch Grad $b(x) < t$ wird $w_H(\boldsymbol{b}) \le t$ und $w_H(\boldsymbol{e}) \le t$ impliziert, aber die Umkehrung gilt natürlich nicht. Selbstverständlich kann ein Bündelfehler auch fehlerfreie Stellen enthalten. Es kann vereinbart werden, daß Beginn und Ende eines Bündelfehlers fehlerhaft sind, aber das ist unwichtig.

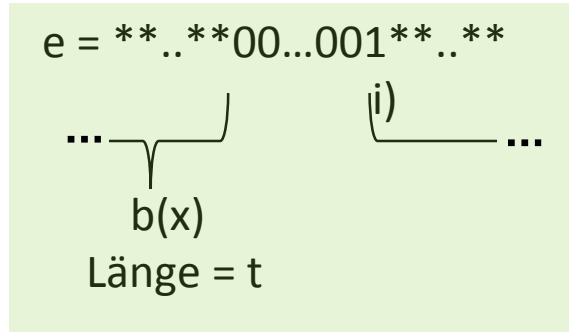
Bei zyklischen Codes wird der Begriff des Bündelfehlers sinnvollerweise dahingehend erweitert, daß der Bündelfehler auch am Ende eines Wortes beginnen darf und sich dann zyklisch am Anfang fortsetzt: Ein zyklischer Bündelfehler $e(x)$ der Länge t ist dadurch gekennzeichnet, daß höchstens t zyklisch aufeinanderfolgende Stellen ungleich Null sind, d.h.

$$e(x) = R_{x^n-1}[x^i b(x)] \quad mit \quad \text{Grad } b(x) < t. \tag{5.6.2}$$

Beispiele für Bündelfehler bei $n = 8, t = 4$:

10000000 : Bündelfehler
00100100 : Bündelfehler
00100010 : kein Bündelfehler, kein zyklischer Bündelfehler
00100001 : kein Bündelfehler, aber zyklischer Bündelfehler.





mathematisch sinnvoll,
physikalisch abwegig

Leider sind Bündelfehler-erzeugende Kanäle nicht so einfach zu beschreiben wie der Einzelfehler-erzeugende DMC. Solange aber nicht die exakte Berechnung von Fehlerwahrscheinlichkeiten gefordert wird, reicht oft die simple Vorstellung aus, daß die Fehler gebündelt auftreten. Ein Bündelfehler der Länge t wird als wahrscheinlicher angesehen als t verstreute Einzelfehler. Dazu werden einige Beispiele betrachtet:

$$\begin{aligned}\boldsymbol{e}_1 &= \ldots 0000111100000000000\ldots\\ \boldsymbol{e}_2 &= \ldots 0000101001010000000\ldots\\ \boldsymbol{e}_3 &= \ldots 0000100010000000000\ldots\end{aligned}$$

Wenn für den Kanal nur Bündelfehler der Länge ≤ 4 unterstellt werden, dann ist das Fehlermuster $\boldsymbol{e}_1$ wahrscheinlicher als $\boldsymbol{e}_2$ und selbst noch wahrscheinlicher als $\boldsymbol{e}_3$, obwohl $w_H(\boldsymbol{e}_3) < w_H(\boldsymbol{e}_1)$ gilt.

Bei binären Kanälen beträgt die Fehlerrate innerhalb eines Bündelfehlers normalerweise rund 50%.

Beispiel 5.7 Consider a non-cyclic linear $(6, 2, 3)_2$ code with

$$\mathcal{C} = \{000000, 111000, 000111, 111111\}.$$

nicht-zyklisch,
schlechte Bündelfehlererkennung

The burst error 111000 is equal to a codeword and is not detected. Permutations lead to an equivalent code also with $d_{\min} = 3$,

$$\mathcal{C} = \{000000, 101010, 010101, 111111\}.$$

zyklisch,
gute Bündelfehlererkennung

This code detects all $L_4 - 1 = 45 < 6 \cdot 2^3 = 48$ cyclic burst errors of a length smaller than or equal to 4 because these burst errors are not codewords:

100000	110000	101000	111000	100100	101100	110100	111100
010000	011000	010100	011100	010010	010110	011010	011110
001000	001100	001010	001110	001001	001011	001101	001111
000100	000110	000101	000111		100101	100110	100111
000010	000011	100010	100011		110010	010011	110011
000001	100001	010001	110001		011001	101001	111001.

Achtung:
Dieses Beispiel in der deutschen Version des Buches war fehlerhaft

Independent of the permutation $d_{\min} = 2$ arbitrary single errors are always detected in this example. ■

Für die Erkennung von Bündelfehlern sind äquivalente Codes nicht gleich gut – insbesondere wenn ein nicht-zyklischer mit einem zyklischen Code verglichen wird. Durch Permutationen kann ein guter Code zu einem schlechten Code werden. Das gleiche gilt für die Korrektur von Bündelfehlern.

Dagegen sind bei der Erkennung bzw. Korrektur von Einzelfehlern äquivalente Codes als gleichwertig zu betrachten.

Satz 5.9. *Ein (beliebig ungünstiger) zyklischer $(n,k)_q$-Code erkennt alle Bündelfehler bis zur Länge $t' \leq n-k$. Von den Bündelfehlern größerer Länge wird nur eine sehr kleine Quote nicht erkannt:*

$$\textit{Quote nicht erkannter Bündelfehler} = \left\{ \begin{array}{ll} \dfrac{q^{-(n-k-1)}}{q-1} & t' = n-k+1 \\ q^{-(n-k)} & t' \geq n-k+2 \end{array} \right\}. \qquad (5.6.4)$$

Beweis: Natürlich wird $e(x) = R_{x^n-1}[x^i b(x)]$ genau dann erkannt, wenn auch $b(x)$ erkannt wird. Sei $b(x) \neq 0$ und Grad $b(x) = t' - 1$:

"$t' \leq n-k$": Nach (5.2.4) ist $b(x) \notin \mathcal{C}$.

Die Quote der erkennbaren Bündelfehler bei t‘ > n-k ist weniger wichtig.

Zum Vergleich: Ein linearer Code erkennt immer $d_{\min} - 1$ Einzelfehler (Satz 3.4) und nach der Singleton-Schranke gilt $d_{\min} - 1 \leq n - k$ (Satz 3.7). Bestenfalls sind also $n - k$ Einzelfehler erkennbar bzw. bestenfalls ein Bündelfehler der Länge $n - k$. Bei einem zyklischen Code ist dagegen immer die Erkennung eines Bündelfehlers der Länge $n - k$ garantiert – auch wenn der Code sehr schlecht gewählt wurde.

Eine besondere und praktisch außerordentlich wichtige Klasse von Codes zur Fehlererkennung behandelt der folgende Satz:

Satz 5.10 (CRC-Codes). *Ein zyklischer $(2^r-1, 2^r-r-2, 4)_2$-Code wird als* CRC-Code *(Cyclic Redundancy Check) bezeichnet, wenn das Generatorpolynom von der Form*

$$g(x) = (1+x)p(x) \tag{5.6.5}$$

ist, wobei $p(x)$ ein primitives Polynom vom Grad r sein muß !

(1) Alle Fehlermuster bis zum Gewicht 3 werden erkannt.
(2) Alle Fehlermuster ungeraden Gewichts werden erkannt.
(3) Alle Bündelfehler bis zur Länge $r+1$ werden erkannt.
(4) Von den Bündelfehlern der Länge $r+2$ wird nur eine Quote von 2^{-r} nicht erkannt.
(5) Von den Bündelfehlern der Länge $\geq r+3$ wird nur eine Quote von $2^{-(r+1)}$ nicht erkannt.

Beim CRC-Code mit 16 Prüfbits werden also von den Bündelfehlern 100% bei Länge ≤ 16 und 99,9969% bei Länge $= 17$ und 99,9985% bei Länge ≥ 18 erkannt, sofern die Anzahl der Infobits ≤ 32751 ist.

CRC-Codes werden oft als verkürzte Codes (siehe Definition 4.6) betrieben, sind dann aber nicht mehr zyklisch.

In internationalen Standards sind CRC-Generatorpolynome genormt:

$$x^{16}+x^{12}+x^5+1 = (x+1)(x^{15}+x^{14}+x^{13}+x^{12}+x^4+x^3+x^2+x+1)$$
$$x^{12}+x^{11}+x^3+x^2+x+1 = (x+1)(x^{11}+x^2+1)$$
$$x^8+x^2+x+1 = (x+1)(x^7+x^6+x^5+x^4+x^3+x^2+1).$$

Wie vorangehend wird weiterhin ein diskreter Kanal mit Hard-Decision sowie ein zyklischer $(n,k)_q$-Code vorausgesetzt. Nach Abschnitt 3.2 werden t Einzelfehler korrigiert, wenn die Decodierkugeln vom Radius $t = \lfloor (d_{\min} - 1)/2 \rfloor$ disjunkt sind, d.h. für alle Codewörter $\boldsymbol{a}, \boldsymbol{a}' \in \mathcal{C}$ mit $\boldsymbol{a} \neq \boldsymbol{a}'$ und alle Fehlermuster $\boldsymbol{e}, \boldsymbol{e}' \in K_t(\boldsymbol{0})$ gilt stets $\boldsymbol{a} + \boldsymbol{e} \neq \boldsymbol{a}' + \boldsymbol{e}'$ (siehe dazu auch Aufgabe 3.26). In entsprechender Weise wird nun die Korrektur von Bündelfehlern definiert:

Definition 5.6. *Ein zyklischer $(n,k)_q$-Code $\mathcal{C}$ korrigiert einen (zyklischen) Bündelfehler der Länge t pro Codewort, wenn für alle Codewörter $\boldsymbol{a}, \boldsymbol{a}' \in \mathcal{C}$ mit $\boldsymbol{a} \neq \boldsymbol{a}'$ und alle (zyklischen) Bündelfehler $\boldsymbol{e}, \boldsymbol{e}'$ der Länge $\leq t$ stets*

$$\boldsymbol{a} + \boldsymbol{e} \neq \boldsymbol{a}' + \boldsymbol{e}'$$

gilt. Kein Empfangswort darf also in mehrfacher Weise als $\boldsymbol{y} = \boldsymbol{a} + \boldsymbol{e}$ darstellbar sein, sondern entweder eindeutig oder überhaupt nicht.

Eine äquivalente Kennzeichnung ist, daß die Syndrome aller (zyklischen) Bündelfehler verschieden sind (siehe Aufgabe 4.14).

In entsprechender Weise wird definiert, ob ein Code mehrere (zyklische) Bündelfehler pro Codewort korrigieren kann.

Bei der Bündelfehler-Korrektur wird also nicht verlangt, daß jeder (zyklische) Bündelfehler richtig ML-decodiert wird, sondern es werden nur diejenigen Codewörtern betrachtet, zu denen die Differenz $\boldsymbol{y} - \boldsymbol{a} = \boldsymbol{e}$ ebenfalls ein (zyklischer) Bündelfehler ist. Für jeden korrigierbaren (zyklischen) Bündelfehler muß genau ein solches Codewort existieren.

Ein Code zur Bündelfehler-Korrektur muß also einerseits die in Definition 5.6 geforderte Eigenschaft besitzen und andererseits muß der Decoder eine spezielle Decodierregel realisieren. Bündelfehler-korrigierende Codes werden nicht bezüglich der Minimaldistanz oder der Gewichtsverteilung entworfen, sondern so, daß die Korrektur bestimmter Fehlermuster garantiert wird. Natürlich kann der gleiche Code sowohl zur Korrektur von Einzelfehlern wie zur Korrektur von Bündelfehlern benutzt werden – aber nur im Prinzip, denn leistungsfähige Codes zur Korrektur von Bündelfehlern sind anders aufgebaut als Codes zur Korrektur von Einzelfehlern.

Bei der Fehlerkorrektur können entweder einige unabhängige Einzelfehler oder ein längerer Bündelfehler oder mehrere kürzere Bündelfehler korrigiert werden – aber nicht alles gleichzeitig. Diese Wahlmöglichkeit gibt es bei der Fehlererkennung nicht.

Obige Bemerkungen sind ziemlich formal(istisch) und eher unbedeutend – denn bei der Einführung von RS-Codes zur Korrektur von Bündelfehlern wird dies alles wieder in einem neuen Licht erscheinen!

Tabelle 5.2. Leistungsfähigkeit zyklischer $(n, k, d_{\min})_q$-Codes

	Einzelfehler (Anzahl)	Bündelfehler (Länge)
Erkennung	$t' = d_{\min} - 1$ $\leq n - k$ bestenfalls (MDS)	$t' = n - k$ garantiert
Korrektur	$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$ $\leq \left\lfloor \frac{n-k}{2} \right\rfloor$ bestenfalls (MDS)	$t \leq \left\lfloor \frac{n-k}{2} \right\rfloor$ bestenfalls

Nochmals: die üblichen RS-Codes zur Korrektur von Bündelfehlern basieren auf einer etwas anderen Sichtweise auf Bündelfehler.