

Etwas Algebra – endliche Körper Eines haben wir aus unseren Beispielen gelernt: Wenn wir eine Chance haben wollen, gute Codes zu konstruieren, dann sollten wir mit den Codewörtern in geeigneter Weise *rechnen* können. Wenn wir uns also unsere Codewörter als n -Tupel über einem endlichen Alphabet A vorstellen, so kommt uns doch gleich der euklidische dreidimensionale Raum $\mathbb{R}^3$ in den Sinn oder besser allgemeiner $\mathbb{R}^n$. Mit diesen können wir *rechnen*, nämlich Vektoren komponentenweise addieren und Unterräume davon bilden (z. B. Geraden und Ebenen). Unterräume besitzen **Basisvektoren** und eine **Dimension**. Was hindert uns nun daran, statt des unendlichen **Körpers** $\mathbb{R}$ der reellen Zahlen endliche Körper K zu betrachten, wie z. B. $K = \{0, 1\}$ oder $K = \{0, 1, -1\}$? Eigentlich gar nichts, also tun wir's. Und wenn wir schon dabei sind: Das übliche **Skalarprodukt** von Vektoren in $\mathbb{R}^3$ oder $\mathbb{R}^n$ lässt sich natürlich genauso für Vektoren in K^n bilden.

Grundlagen linearer Codes Mit diesem algebraischen Rüstzeug im *Handgepäck* gehen wir wieder auf die Suche nach guten Codes. Jetzt haben wir aber einen strategischen Vorteil: Wir können uns das Alphabet A als endlichen Körper K vorstellen und gehen kurzerhand dazu über, nur noch **lineare Codes** zu betrachten, d. h. Unterräume des Raumes der n -Tupel K^n über K . Ein linearer Code hat dann eine **Dimension** k und auch **Basisvektoren**, die linear unabhängig sind und den Code aufspannen. Damit haben wir endlich auch ein einfaches Maß für die Redundanz, nämlich die **Rate** k/n : Je größer die Rate, desto kleiner die Redundanz. Außerdem müssen wir uns jetzt nicht mehr alle Codewörter merken, sondern es reicht, einen Satz von Basisvektoren zu kennen. Diese schreiben wir übersichtshalber als Zeilen einer Matrix (mit k Zeilen und n Spalten) und nennen sie **Generatormatrix** unseres Codes. Wir überprüfen unser Konzept an den Eingangsbeispielen sowie an zwei berühmten Codes – dem **kleinen binären Hamming-Code** und dem **ternären Golay-Code**.

Erweiterte und duale Codes Die Idee, mit linearem Code zu arbeiten, scheint ganz brauchbar zu sein. Dann verfolgen wir sie noch ein Stück weiter. Wir können z. B. einen linearen Code um eine Stelle verlängern, indem wir – analog zu unserem ersten Beispiel – dort eine Paritätsprüfung durchführen. Man nennt diesen den **erweiterten Code**. Außerdem bringen wir jetzt das Skalarprodukt *in Stellung* und betrachten die Vektoren, die auf allen unseren Codewörtern *senkrecht* stehen, d. h. Skalarprodukt 0 mit ihnen haben. Man nennt diesen Code den **dualen Code** und seine Generatormatrix die **Kontrollmatrix** unseres Ausgangscodes. Auch diese Konzepte testen wir an den Beispielen des letzten Abschnitts.

Maximum-Likelihood- und Syndromdecodierung Jetzt sollten wir unsere Euphorie aber etwas bremsen und unser Vorgehen nochmals hinterfragen: Was sind denn eigentlich unsere Kriterien an gute Codes? Wir sind bislang von großem Minimalabstand und großer Rate ausgegangen. Das ist auch nicht ganz falsch, aber es fehlt ein weiteres, möglicherweise noch wichtigeres Kriterium: Der Code muss nach Datenübertragung korrekt und möglichst effizient decodierbar sein. Decodieren will man offensichtlich ein empfangenes Wort zu *dem* Codewort, für das es am wahrscheinlichsten ist, dass das empfangene Wort aus ihm entstanden ist (sog. **Maximum-Likelihood-Decodierung**). Wir vergewissern uns zunächst, dass dieses Prinzip – zumindest für in der Praxis gängige Kanäle – mit unserer bisherigen Denkweise der Verwendung des kürzesten Hamming-Abstands übereinstimmt (sog. **Hamming-Decodierung**). Außerdem studieren wir das Standarddecodierverfahren für lineare Codes, die **Syndromdecodierung**.

Codierung linearer Codes und Gesamtszenario Nachdem wir uns mit dem immens wichtigen Thema Decodieren von linearen Codes intensiv auseinandergesetzt haben, wollen wir nun noch überlegen, wie man besonders effizient codieren kann. Am besten wäre es doch, wenn man bei jedem Codewort an den ersten k Stellen die eigentliche Information ablesen könnte und die restlichen $n - k$ Stellen für die redundante Information vorbehalten wären. Wir sehen, dass dies immer möglich ist, und nennen diese Form der Codierung **systematisch**. Das ist jetzt Anlass genug, uns nochmals das **Gesamtszenario der Kanalcodierung und -decodierung** klarzumachen – am besten mit dem kleinen binären Hamming-Code und seiner digitalen Schaltlogik.

2.1 Etwas Algebra – endliche Körper

In unserer Definition von Blockcodes haben wir bislang eine beliebige endliche Menge A als Alphabet zugrunde gelegt. In unseren Beispielen allerdings haben wir Alphabete benutzt, mit denen wir rechnen konnten. Konkret waren dies binäre Ziffern sowie die Reste ganzer Zahlen modulo 97, 11, 10 oder 9. Zu diesem Zweck mussten wir allerdings zusätzlich einige Buchstaben in Ziffern konvertieren. Es wird sich bei unserer Suche nach guten

Codes herausstellen, dass es sinnvoll ist, sich auf *rechenbare* Alphabete zu konzentrieren. Denn einerseits hilft dies bei der Konstruktion und Herleitung der Eigenschaften von Codes. Andererseits muss man ja auch bedenken, dass die Codierung und Decodierung in der Regel computergestützt erfolgt und vor allem für die Decodierung schnelle Verfahren erforderlich sind.

2.1.1 Der Körper der rationalen und reellen Zahlen

Mit den **rationalen Zahlen** $\mathbb{Q}$ und den **reellen Zahlen** $\mathbb{R}$ kann man z. B. gut rechnen. Man kann sie addieren und multiplizieren und es gelten dabei vernünftige Rechenregeln.

- $a + (b + c) = (a + b) + c$, $a(bc) = (ab)c$ **Assoziativgesetze**.
- $a + b = b + a$, $ab = ba$ **Kommutativgesetze**.
- $a(b + c) = ab + ac$ **Distributivgesetz**.
- Es gibt neutrale Elemente 0 und 1 bzgl. Addition und Multiplikation.
- Zu jedem Element a gibt es ein inverses Element bzgl. der Addition, nämlich $-a$, und
- zu jedem Element $b \neq 0$ gibt es ein inverses Element bzgl. der Multiplikation, nämlich b^{-1} .

Man nennt solche Mengen in der Mathematik einen **Körper** (engl. **Field**).

2.1.2 Endliche Körper

Sei dazu p eine Primzahl und $\mathbb{Z}_p$ die Reste modulo p . Wenn klar ist, dass wir uns in $\mathbb{Z}_p$ bewegen, lassen wir auch den Zusatz „mod p “ weg und schreiben $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$. Ist auch $\mathbb{Z}_p$ ein Körper? Alle Rechenregeln im Körper übertragen sich trivialerweise von $\mathbb{R}$ bzw. $\mathbb{Q}$ auf die Reste $\mathbb{Z}_p$, nur beim multiplikativen Inversen muss man etwas aufpassen. Sei dazu b eine natürliche Zahl mit $0 < b < p$. Da p eine Primzahl ist, sind b und p teilerfremd und es gibt ganze Zahlen r, s mit $rb + sp = 1$. Das folgt aus dem **euklidischen Algorithmus**, mithilfe dessen man den größten gemeinsamen Teiler zweier Zahlen als deren Vielfachsumme darstellen kann. Lesen wir diese Gleichung in $\mathbb{Z}_p$, so folgt, $rb = 1$ und $0 \neq b \in \mathbb{Z}_p$ hat ein multiplikatives Inverses, nämlich $b^{-1} = r$.

Also ist $\mathbb{Z}_p$ ein **Körper** mit p Elementen. Insbesondere ist $\mathbb{Z}_2 = \{0, 1\}$ der Körper der **binären Zahlen (Bits)** und $\mathbb{Z}_3 = \{0, 1, -1\}$ der Körper der **ternären Zahlen**.

Mit $\mathbb{Z}_p$ kann man sehr effektiv rechnen, da die Division durch p mit Rest und allgemeiner der euklidische Algorithmus effizient implementierbar sind.

Es gibt sogar zu jeder Primzahlpotenz p^m einen Körper K mit $q = p^m$ Elementen. Für $q = p$ haben wir den Körper eben explizit konstruiert, für echte p -Potenzen ist das nicht ganz so einfach. Wir werden erst wieder in Abschn. 5.1 darauf zurückkommen. In den folgenden Abschnitten werden wir häufig von Körpern K mit q Elementen sprechen. Zum Verständnis und auch für die Beispiele reicht es aber im Moment völlig aus, dass man sich $K = \mathbb{Z}_p$ vorstellt.

Es stellt sich die naheliegende Frage, ob auch $\mathbb{Z}_m$ ein Körper für zusammengesetztes m ist, z. B. für $m = 10$ oder $m = 9$. Das stimmt leider nicht. Nehmen wir speziell $m = 10$, dann gilt $2 \neq 0$ und $5 \neq 0$ in $\mathbb{Z}_{10}$ gelesen. Hätte also 2 ein multiplikatives Inverses r in $\mathbb{Z}_{10}$, so würde $2r = 1$ gelten. Wir multiplizieren die Gleichung mit $5 \neq 0$ und erhalten $5 = 5 \cdot 1 = 5(2r) = 10r = 0r = 0$, ein Widerspruch. Man nennt solche Mengen ohne multiplikatives Inverses einen kommutativen **Ring**. Wir werden Ringe jedoch nicht weiter betrachten.

Jetzt sind wir aber fast schon dort, wo wir hinwollten. Statt eines abstrakten endlichen Alphabets A für Blockcodes können endliche Körper K genommen werden. Da gibt es kleine, z. B. $\{0, 1\}$, beliebig große, aber eben immer nur solche von der Größe einer Primzahlpotenz. Aber Blockcodes sind ja n -Tupel über dem Alphabet $A = K$. Also sollten wir uns auch n -Tupel über K näher anschauen.

2.1.3 K^n als Vektorraum über dem endlichen Körper K

Über dem Körper $\mathbb{R}$ beispielsweise können wir Vektorräume bilden, den bekannten dreidimensionalen euklidischen Vektorraum $\mathbb{R}^3$, mit dem wir unsere Umgebung wahrnehmen und der eindimensionale Unterräume (Geraden) und zweidimensionale Unterräume (Ebenen) hat. Oder etwas abstrakter den Vektorraum der n -Tupel $\mathbb{R}^n$, mit dem manche Leser sicher auch schon etwas Erfahrung gesammelt haben. Warum also nicht einfach unsere Blockcodes als Teilmengen von K^n auffassen und für K^n alles nachmachen, was wir für $\mathbb{R}^n$ schon kennen?

Wir fassen also K^n als **Vektorraum** auf mit

- komponentenweiser Addition von Vektoren

$$(v_1, \dots, v_n) + (w_1, \dots, w_n) = (v_1 + w_1, \dots, v_n + w_n)$$

und

- komponentenweiser Multiplikation mit Elementen von K , nämlich

$$c(v_1, \dots, v_n) = (cv_1, \dots, cv_n).$$

Ein **Unterraum** U von K^n ist eine Teilmenge, die unter den eben genannten Vektorraumoperationen abgeschlossen ist.

K^n hat viele Unterräume, z. B. eindimensionale Geraden und zweidimensionale Ebenen, wobei man bei der geometrischen Interpretation etwas vorsichtig sein muss. Jedes Objekt ist nämlich endlich, so hat eine Gerade genau $|K| = q$ Punkte, die Gerade mit Richtungsvektor $(1, 0, \dots, 0)$ besteht z. B. genau aus der Punktmenge $\{(c, 0, \dots, 0) | c \in K\}$.

Besonders wichtig ist aber die Tatsache, dass man für jeden Unterraum U von K^n Basisvektoren finden kann.

Basisvektoren sind eine Menge von Vektoren $\{u_1, \dots, u_k\}$ im Unterraum U , die jeden Vektor im Unterraum U als **K -Linearkombination erzeugen**, selbst aber **linear unabhängig** sind. Man bezeichnet die Menge $\{u_1, \dots, u_k\}$ auch kurz als **Basis**. Es gilt also:

- Zu jedem u aus dem Unterraum U gibt es Elemente $c_1, \dots, c_k \in K$ mit $u = c_1u_1 + \dots + c_ku_k$.
- Ist $d_1u_1 + \dots + d_ku_k = 0$ für $d_i \in K$, so folgt, $d_1 = \dots = d_k = 0$.

Eine Basis ist jedoch durch U *nicht* eindeutig bestimmt. Geometrisch kennt man das ja etwa von Ebenen im $\mathbb{R}^3$, die man auf viele Möglichkeiten mit zwei Vektoren aufspannen kann.

Die Anzahl der Basisvektoren k ist durch U eindeutig bestimmt. Man bezeichnet sie als **Dimension** $k = \dim(U)$ des Unterraums.

Diese Eigenschaften von Vektorräumen über einem Körper K , die ja für $\mathbb{R}^3$ bzw. $\mathbb{R}^n$ weitgehend bekannt sein sollten, erfordern bei ihrer Herleitung die Existenz von multiplikativen inversen Elementen in K . Weil im Folgenden besonders wichtig, wollen wir uns hier kurz überlegen, warum es bei endlichen Körpern K in Unterräumen $U \subseteq K^n$ stets Basisvektoren gibt und deren Anzahl eindeutig bestimmt ist.

Seien also $\{v_1, \dots, v_m\}$ erzeugende Vektoren von U mit kleinstmöglicher Anzahl m . Solche gibt es jedenfalls, da $U \subseteq K^n$ endlich ist. Wären diese v_i nicht linear unabhängig, so könnte man $d_1 v_1 + \dots + d_m v_m = 0$ schreiben mit – sagen wir – $d_1 \neq 0$. Damit ist aber $v_1 = -d_1^{-1}(d_2 v_2 + \dots + d_m v_m)$ und folglich kann jedes Element aus U bereits als K -Linearkombination der Vektoren $\{v_2, \dots, v_m\}$ geschrieben werden, entgegen der minimalen Wahl von m . Also ist $\{v_1, \dots, v_m\}$ eine Basis von U .

Seien nun $\{u_1, \dots, u_k\}$ und $\{v_1, \dots, v_m\}$ zwei Basen von U . Dann lässt sich $v_1 \neq 0$ schreiben als $v_1 = c_1 u_1 + \dots + c_k u_k$ mit – sagen wir wieder – $c_1 \neq 0$. Hieraus folgt $u_1 = c_1^{-1}(v_1 - c_2 u_2 - \dots - c_k u_k)$ und somit erzeugen auch die Vektoren $\{v_1, u_2, \dots, u_k\}$ den Unterraum U . Sei nun $d_1 v_1 + d_2 u_2 + \dots + d_k u_k = 0$. Setzen wir hier v_1 ein, so ergibt sich $d_1 c_1 u_1 + (d_1 c_2 + d_2) u_2 + \dots + (d_1 c_k + d_k) u_k = 0$. Wegen der linearen Unabhängigkeit der u_i und $c_1 \neq 0$ folgt zunächst $d_1 = 0$ und damit auch für die restlichen $d_i = 0$. Dies zeigt, dass die $\{v_1, u_2, \dots, u_k\}$ auch linear unabhängig sind und daher eine Basis von U bilden. Man hat also u_1 durch v_1 ausgetauscht (sog. **Steinitz'scher Austauschatz**). Dieses Argument wiederholt angewandt – also formal ein Induktionsbeweis – zeigt, dass m der u_i durch die v_i ausgetauscht werden können, also insbesondere $m \leq k$. Dreht man diese Argumentation um, so kann man genauso gut k der v_j durch die u_j austauschen, also gilt auch $k \leq m$. Insgesamt ist damit $k = m$ gezeigt und beide Basen haben gleich viele Elemente, d. h. der Begriff der Dimension macht Sinn.

Für n -Tupel über Ringen (statt Körpern) gelten die Aussagen über Basis und Dimension nicht. Das Rechnen mit Basen und Dimensionen ist jedoch wichtig im Zusammenhang mit der Konstruktion von guten Codes. Das ist der Grund, warum man sich auf Vektorräume über Körpern konzentriert.

2.1.4 Das Skalarprodukt auf K^n für endliche Körper K

Jetzt gehen wir noch einen letzten Schritt weiter und machen auch das bekannte **Skalarprodukt** im $\mathbb{R}^3$ bzw. $\mathbb{R}^n$ nach.

Wir definieren für $v = (v_1, \dots, v_n)$ und $w = (w_1, \dots, w_n) \in K^n$ das **Skalarprodukt** als $\langle v, w \rangle = v_1 w_1 + \dots + v_n w_n$.

Wie man vom Skalarprodukt im $\mathbb{R}^n$ weiß oder aber sofort nachrechnet, gelten die folgenden Rechenregeln für $u, v, w \in K^n$ und $c \in K$.

- $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$
- $\langle cu, v \rangle = c \cdot \langle u, v \rangle$
- $\langle u, v \rangle = \langle v, u \rangle$

- $\langle 0, v \rangle = 0$
- Ist $\langle u, v \rangle = 0$ für alle v , so ist $u = 0$, sog. **Regularität**.

Im euklidischen Raum ist man gewöhnt, dass man mit dem Skalarprodukt überprüfen kann, ob zwei Vektoren senkrecht aufeinander stehen, nämlich genau dann, wenn ihr Skalarprodukt gleich 0 ist. Die geometrische Anschauung des *Senkrechtstehens* muss man aber leider bei endlichen Körpern über Bord werfen. Viel schlimmer noch: Vektoren können *auf sich selbst senkrecht stehen*. Nimmt man beispielsweise $(1, 1) \in (\mathbb{Z}_2)^2$, so gilt, $\langle (1, 1), (1, 1) \rangle = 1 + 1 = 0$.

Wir verzichten hier auf konkrete Beispiele von Unterräumen über endlichen Körpern und deren Skalarprodukt, da uns diese ohnehin in den folgenden Abschnitten als Codes dauernd begegnen werden.

2.2 Grundlagen linearer Codes

2.2.1 Was versteht man unter linearen Codes?

Sei K ein endlicher Körper und $|K| = q$. Ein Blockcode C mit Alphabet K und Länge n heißt **linearer Code**, wenn C ein Unterraum des Vektorraums K^n ist. Hat C die Dimension $\dim(C) = k$ und Minimalabstand $d(C) = d$, so heißt C ein **$[n, k, d]_q$ -Code**.

Ist q aus dem Zusammenhang heraus klar oder spielt d bei der aktuellen Überlegung keine Rolle, so spricht man auch kurz von $[n, k, d]$ - oder $[n, k]$ -Codes. Der Code C hat damit $|C| = q^k$ Elemente, sog. **Codewörter**. Das heißt nicht unbedingt, dass jedes Codewort mit einer Informationseinheit aus B belegt sein muss. Zur Erinnerung, die **Codierfunktion** ist eine eindeutige Zuordnung „Element von B “ $\rightarrow$ „Codewort in C “, die jedoch nicht unbedingt jedes Codewort erreichen muss. Man sagt dazu **injektiv**, aber nicht notwendig **surjektiv**. Zum Beispiel muss im ISBN-Code nicht jede mögliche Zahl im Code durch eine real existierende Buchnummer belegt sein, denn es müssen ja auch noch Nummern für Neuerscheinungen frei sein. In jedem Fall muss aber allgemein $|B| \leq |C| = q^k$ gelten. Wie man am besten eine solche Codierfunktion wählt, stellen wir vorläufig bis zu Abschn. 2.5 zurück.

2.2.2 Die Rate linearer Codes

Wir hatten den Begriff der Rate für Blockcodes im Zusammenhang mit dem Satz von Shannon bislang nur intuitiv definiert. Die Rate ist ein Maß für die Redundanz, je höher

nämlich die Rate ist, desto geringer ist die Redundanz. Für lineare Codes vereinfacht sich das.

Die **Rate** linearer Codes ist gegeben als Verhältnis von Dimension k und Länge n , also **Rate** $= k/n$.

Rate k/n und Minimalabstand d sind gegenläufig. Größerer Minimalabstand (d. h. bessere Fehlerkorrektureigenschaft) bedeutet gleichzeitig geringere Rate (d. h. größere Redundanz).

Das **Ziel der Codierungstheorie** kann man also wie folgt präzisieren. Man finde möglichst lineare Codes von

- für die jeweilige Anwendung praktikabler Länge n mit
- möglichst großem Minimalabstand d und
- möglichst großer Rate k/n , für die es aber auch
- schnelle und effiziente Codier- und Decodierverfahren gibt.

Die Konstruktionsmethode für *gute* Codes im Zusammenhang mit der **Gilbert-Schranke** liefert zwar Codes mit großer Rate, aber eben leider wenig handhabbare nichtlineare Codes. Wir werden uns im Folgenden stets auf lineare Codes beschränken. Für diese lassen sich z. B. der Hamming-Abstand und damit der Minimalabstand einfacher berechnen.

2.2.3 Gewicht und Minimalgewicht

Sei K ein endlicher Körper, $d(\cdot, \cdot)$ der Hamming-Abstand auf K^n und C ein linearer Code in K^n .

- Man nennt $wt(v) = d(v, 0) = |\{i \mid v_i \neq 0\}|$ das **Gewicht** von $v = (v_1, \dots, v_n) \in K^n$.
- Ist $C \neq \{0\}$, so heißt $wt(C) = \min\{wt(c) \mid 0 \neq c \in C\}$ das **Minimalgewicht** von C . Für $C = \{0\}$ setzt man sinngemäß $wt(C) = 0$.

2.2.4 Minimalabstand bei linearen Codes

Mit den obigen Bezeichnungen gilt:

- $d(v, v') = d(v + w, v' + w)$ für alle $v, v', w \in K^n$, sog. **Translationsinvarianz**,
- $d(C) = wt(C)$, d. h. das Minimalgewicht ist gleich dem Minimalabstand.

Die erste Aussage ist sehr einfach einzusehen, da sich v und v' an genau denselben Stellen unterscheiden wie $v + w$ und $v' + w$. Die zweite Aussage hingegen ergibt sich aus

$$\begin{aligned} d(C) &= \min\{d(v, w) \mid v, w \in C, v \neq w\} = \min\{d(v - w, 0) \mid v, w \in C, v \neq w\} \\ &= \min\{d(u, 0) \mid u \in C, u \neq 0\} = wt(C). \end{aligned}$$

Für lineare Codes reduziert sich daher die Bestimmung des Minimalabstands auf die des Minimalgewichts, also von $|C|(|C| - 1)/2$ Abstandsbestimmungen auf $|C| - 1$ Gewichtsbestimmungen.

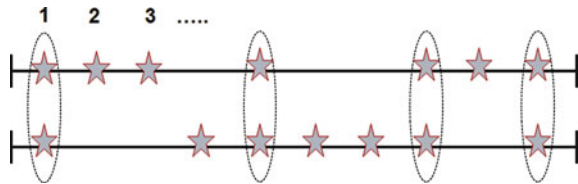
2.2.5 Der Träger

Im Zusammenhang mit dem Gewicht eines Vektors ist manchmal auch der Begriff des Trägers hilfreich. Sei also K ein endlicher Körper und $v = (v_1, \dots, v_n) \in K^n$. Unter dem **Träger** von v versteht man die Menge $Tr(v) = \{i \mid v_i \neq 0\}$, also genau die Menge der Indizes von v mit Koordinate $v_i \neq 0$. Es gilt dabei $wt(v) = |Tr(v)|$. Den Begriff des Trägers benutzt man häufig dann, wenn man zwei Vektoren v und w addieren will. Es gilt nämlich:

- $wt(v + w) \geq wt(v) + wt(w) - 2|Tr(v) \cap Tr(w)|$,
- $wt(v + w) = wt(v) + wt(w) - 2|Tr(v) \cap Tr(w)|$ für binäre Vektoren über $K = \mathbb{Z}_2$.

Am besten man macht sich das an der Visualisierung in Abb. 2.1 klar. Hier sind symbolisch die zwei Vektoren v und w aufgemalt und die Träger mit Sternen markiert.

Abb. 2.1 Die Träger zweier Vektoren



2.2.6 Generatormatrix

Sei C ein $[n, k, d]$ -Code über dem endlichen Körper K . Als Unterraum des K^n hat C eine Basis, sagen wir, $g_1 = (g_{11}, \dots, g_{1n}), \dots, g_k = (g_{k1}, \dots, g_{kn})$. Dann heißt die Matrix

$$G = \begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix} = \begin{pmatrix} g_{11} & \cdots & g_{1n} \\ \vdots & & \vdots \\ g_{k1} & \cdots & g_{kn} \end{pmatrix}$$

Generatormatrix (oder Erzeugermatrix) von C .

Die Generatormatrix hängt also von der Wahl der Basis ab. Wir werden im Folgenden keine Matrizenrechnung verwenden. Es hat sich aber nun mal eingebürgert, die Basis eines Codes in dieser kompakten Schreibweise als Generatormatrix aufzuschreiben.

2.2.7 Beispiel: Lineare Codes

Jetzt wird es sicher Zeit für ein paar Beispiele. Zunächst jedoch eine allgemeine Bemerkung: Wenn wir nun lineare Codes konstruieren, so müssen wir uns also Unterräume des K^n verschaffen. Dabei ist man häufig versucht, der Einfachheit halber einige der sog. **kanonischen Basisvektoren** $(1, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots$ mit unterzubringen. Wenn man das macht, hat man schon verloren, denn diese drücken das Minimalgewicht des Codes ja runter auf 1. Dies ist der Grund, warum man hier und auch später recht knifflige Konstruktionen wählt.

Paritätsprüfungs-, Wiederholungs- und ISBN-Code

Wir kommen zunächst einmal auf einige unserer Eingangsbeispiele zurück. Der Paritätsprüfungscode, der Wiederholungscode und der ISBN-Code sind allesamt lineare Codes über $\mathbb{Z}_2$ bzw. $\mathbb{Z}_{11}$, da die Erweiterungsstellen jeweils durch lineare Gleichungen der übrigen Stellen ohne additive Konstanten beschrieben werden und $\mathbb{Z}_2$ bzw. $\mathbb{Z}_{11}$ jeweils Körper sind. Die Codes, bei denen modulo 10 bzw. 9 gerechnet wird, sehen zwar zunächst auch linear aus, sind es aber nicht, da sie über dem Ring $\mathbb{Z}_{10}$ bzw. $\mathbb{Z}_9$ gebildet werden.

Der Paritätsprüfungscode hat Parameter $[3, 2, 2]_2$ und Generatormatrix

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

Natürlich kann man auch allgemeiner ein binäres n -Tupel um ein Paritätsbit erweitern und erhält so einen $[n + 1, n, 2]_2$ -Code.

Der Wiederholungscode hat Parameter $[6, 2, 3]_2$ und Generatormatrix

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

Der ISBN-Code hat Parameter $[10, 9, 2]_{11}$ und folgende Generatormatrix mit 10 Spalten und 9 Zeilen:

$$G = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & \dots & 0 & 2 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & 1 & 9 \end{pmatrix}.$$

IBAN-Code

Der IBAN-Code ist etwas diffiziler. Man berechnet dort die Prüfziffer bezogen auf „98“, sodass bei der Validierung 1 heraus kommt. Hätte man sie auf „97“ bezogen, so wäre dieser Code C linear gewesen und bei der Validierung wäre 0 herausgekommen.

Nehmen wir hierfür konkret das Beispiel DE für Deutschland. Hier ist die Kontoidentifikation 18-stellig, zusätzliche 4 Ziffern für „DE“ ergeben eine 22-stellige Zahl. Wie wir wissen, ist $\mathbb{Z}_{97}$ ein Körper, da 97 eine Primzahl ist. Wir betrachten $(\mathbb{Z}_{97})^{23}$ und lesen die einzelnen Stellen modulo 97, also in $\mathbb{Z}_{97}$. Der Code lässt sich dann beschreiben als

$$C = \{(c_1, \dots, c_{23}) \mid c_i \in \mathbb{Z}_{97}, 10^{23}c_1 + 10^{22}c_2 + \dots + 10^3c_{21} + 10^2c_{22} + c_{23} = 0\}.$$

Bis auf die letzte Stelle c_{23} kommen bei allen anderen nur Werte zwischen 0 und 9 vor. Der Code hat Länge 23, Dimension 22 und als Generatormatrix

$$G = \begin{pmatrix} 1 & 0 & \dots & 0 & r_{23} \\ 0 & 1 & 0 & \dots & 0 & r_{22} \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & 1 & r_2 \end{pmatrix}$$

mit 23 Spalten und 22 Zeilen, wobei sich r_i aus $10^i + r_i = 0$ berechnet.

Der kleine binäre Hamming-Code $Ham_2(3)$ mit Parameter $[7, 4, 3]_2$

Sei $C = \{(c_1, \dots, c_7) \mid c_i \in \mathbb{Z}_2, c_1 + c_2 + c_5 + c_7 = 0, c_3 + c_5 + c_6 + c_7 = 0, c_1 + c_4 + c_6 + c_7 = 0\}$. Dann hat C als Generatormatrix

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

Man sieht dies daran, dass die Zeilenvektoren von G einerseits die drei linearen Gleichungen in der Definition von C erfüllen und andererseits offenbar linear unabhängig sind. Wegen $\dim(C) = 4$ hat $|C|$ genau $2^4 - 1 = 15$ Codewörter $\neq 0$. Diese kann man alle noch im Kopf auf ihr Gewicht hin überprüfen und findet $wt(C) = 3$. Also ist C ein $[7, 4, 3]_2$ -Code und heißt **kleiner binärer Hamming-Code** $Ham_2(3)$. Er gehört zu einer ganzen Familie von Codes, die wir in Abschn. 3.1 kennenlernen werden. Dort sehen wir auch einen Weg, wie man das Minimalgewicht des Codes bestimmt, ohne alle Codewörter einzeln zu überprüfen. Übrigens: $Ham_2(3)$ ist äquivalent zu dem Code, den wir in Abschn. 1.3 bereits als kleinen kombinatorischen Code explizit mit seinen 16 Elementen angegeben haben.

Der ternäre Golay-Code G_{11} mit Parameter $[11, 6, 5]_3$

Der Schweizer Elektroingenieur **Marcel Golay** (1902–1989) hat diesen und die in den nächsten Abschnitten beschriebenen, nach ihm benannten berühmten Codes 1949 publiziert. Er hat dafür je eine Generatormatrix angegeben. Wir haben eben nämlich gesehen, dass die Beschreibung eines Codes mittels Generatormatrix meist überschaubarer ist als die Definition in Mengenschreibweise. Hier ist eine Generatormatrix für den ternären Golay-Code G_{11} über $K = \mathbb{Z}_3$.

$$G = \begin{pmatrix} 1 & & & & & & 1 & 1 & 1 & 1 & 1 \\ & 1 & & & & & 0 & 1 & -1 & -1 & 1 \\ & & 1 & & & & 1 & 0 & 1 & -1 & -1 \\ & & & 1 & & & -1 & 1 & 0 & 1 & -1 \\ & & & & 1 & & -1 & -1 & 1 & 0 & 1 \\ & & & & & 1 & 1 & -1 & -1 & 1 & 0 \end{pmatrix}$$

Klar ist, dass der Code Länge 11, Dimension 6 und damit $3^6 - 1 = 728$ Codewörter $\neq 0$ hat. Das Minimalgewicht zu bestimmen, wird hier im Kopf schon schier unmöglich und auf dem Papier sehr mühsam. Ein kleines Rechenprogramm hilft oder ein Trick, den wir im nächsten Abschnitt sehen werden.

Ein kleiner Reed-Solomon-Code mit Parameter $[4, 3, 2]_5$

Als Beispiel für $K = \mathbb{Z}_5$ betrachten wir den linearen Code C über K mit Generatormatrix

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 4 & 1 \end{pmatrix}.$$

Da die Zeilen z_1, z_2 und z_3 von G offenbar linear unabhängig sind, hat C Dimension 3.

Wäre der Minimalabstand $d(C) \geq 3$, so wären mit zwei verschiedenen Codewörtern (c_1, c_2, c_3, c_4) und (c'_1, c'_2, c'_3, c'_4) auch die 4-Tupel $(c_1, c_2, 0, 0)$ und $(c'_1, c'_2, 0, 0)$ verschieden. Das kann natürlich nicht sein wegen $|C| = 5^3$. Etwas Rechnung zeigt außerdem, dass

keiner der kanonischen Basisvektoren $(1, 0, 0, 0), \dots, (0, 0, 0, 1)$ sich als Linearkombination von z_1 bis z_3 darstellen lässt. Damit sind diese nicht in C und folglich ist $d(C) = 2$. Also ist C ein $[4, 3, 2]_5$ -Code. Er gehört zur Familie der **Reed-Solomon-Codes**, die wir allerdings erst in Abschn. 5.2 genauer betrachten werden.

2.3 Erweiterte und duale Codes

Im Zusammenhang mit linearen Codes ist es häufig hilfreich, die folgenden beiden aus einem linearen Code abgeleiteten Codes zu betrachten.

2.3.1 Der erweiterte Code

Sei C ein linearer $[n, k, d]$ -Code über dem endlichen Körper K . Man bezeichnet mit

$$C^\wedge = \{(c_1, \dots, c_n, c_{n+1}) \mid (c_1, \dots, c_n) \in C, c_1 + \dots + c_n + c_{n+1} = 0\}$$

den **erweiterten Code** zu C .

Der Code $C^\wedge$ ist offensichtlich ein linearer $[n+1, k, d^\wedge]$ -Code über K , wobei $d \leq d^\wedge \leq d+1$ gilt. Man erhält eine Basis von $C^\wedge$, indem man eine Basis von C um die sog. **Paritätsziffer** an Position $n+1$ erweitert. Ist C binär, so ist die Anzahl der Einsen in jedem Codewort aus $C^\wedge$ gerade und $C^\wedge$ hat folglich gerades Minimalgewicht. Ist also für einen binären Code d gerade, so ist $d^\wedge = d$, ist d ungerade, so ist $d^\wedge = d+1$. Jedenfalls ist $C^\wedge$ eine Konstruktion, die unserem Paritätsprüfungscode in den Eingangsbeispielen entspricht, nur dass hier ein beliebiger linearer Ausgangscode mit einer Paritätsziffer erweitert wird.

2.3.2 Der duale Code

Sei wieder C ein linearer $[n, k, d]$ -Code über dem endlichen Körper K und $\langle \cdot, \cdot \rangle$ das Skalarprodukt auf K^n , also $\langle (x_1, \dots, x_n), (y_1, \dots, y_n) \rangle = x_1 y_1 + \dots + x_n y_n$. Dann heißt $C^\perp = \{v \in K^n \mid \langle v, c \rangle = 0 \text{ für alle } c \in C\}$ der **duale Code** zu C .

$C^\perp$ hat natürlich ebenfalls Länge n und ist ein linearer Code wegen der Linearität des Skalarprodukts. Aber Vorsicht: Die anschauliche Vorstellung, dass Vektoren aus $C^\perp$ *senkrecht* auf Vektoren aus C stehen, wie wir das in $\mathbb{R}^n$ gewohnt sind, gilt hier nicht. Es gibt sogar Codes, für die $C = C^\perp$ gilt; diese nennt man **selbstdual**. Wir kommen gleich zu einer wichtigen Eigenschaften von $C^\perp$.

$$\text{Es gilt } \dim(C^\perp) = n - \dim(C) = n - k.$$

Leider liegt diese Aussage nicht ganz auf der Hand. Wir wollen daher zunächst ein eher heuristisches Argument anführen. Sei dazu $v = (x_1, \dots, x_n) \in C^\perp$ und $c_1 = (c_{11}, \dots, c_{1n}), \dots, c_k = (c_{k1}, \dots, c_{kn})$ eine Basis von C . Nach Definition von $C^\perp$ gilt dann $0 = \langle v, c_i \rangle = x_1 c_{i1} + \dots + x_n c_{in}$ für $i = 1, \dots, k$. Wir haben es daher mit einem linearen Gleichungssystem über dem Körper K mit k Gleichungen, n Unbestimmten x_j und $k \leq n$ zu tun. Hier scheint ein kurzer Einschub – mit Analogie zu linearen Gleichungen über $\mathbb{R}$ – sinnvoll. Für $k < n$, also bei weniger Gleichungen als Unbestimmten, ergeben sich mehr als eine Lösung. Ein einfaches Beispiel hierzu ist *eine* Gleichung mit *zwei* Unbestimmten x, y über dem Körper $\mathbb{R}$, also $ax + by = 0$. Die Lösungsmenge ist dann eine Gerade in der (x, y) -Ebene, also ein eindimensionaler Unterraum des $\mathbb{R}^2$. Aber halt: Stimmt das wirklich? Voraussetzung dafür ist natürlich, dass nicht beide Koeffizienten a und b gleich 0 sind oder, etwas kunstvoller ausgedrückt, dass der eine Vektor $(a, b) \neq 0$ linear unabhängig ist. Hier ist die Verallgemeinerung dieser Tatsache: Wir betrachten nämlich die aus den Koeffizienten unseres Gleichungssystems zeilenweise gebildeten Vektoren $c_1 = (c_{11}, \dots, c_{1n}), \dots, c_k = (c_{k1}, \dots, c_{kn})$. In unserem Fall wissen wir ja, dass die $c_1, \dots, c_k$ als Basis von C linear unabhängig sind. Daher bildet die Lösungsmenge $(x_1, \dots, x_n)$ des Gleichungssystems einen $(n - k)$ -dimensionalen Unterraum des K^n . Anschaulich gesprochen ist das nicht verwunderlich: Bei der Berechnung der x_j hat man nämlich genau $n - k$ *Freiheitsgrade*. Also hat die Lösungsmenge $(x_1, \dots, x_n)$ des Gleichungssystems, welche nach Konstruktion genau $C^\perp$ ist, die Dimension $n - k$.

Hier ist ein formaleres Argument, welches allerdings deutlich mehr lineare Algebra verwendet. Da nämlich die Basisvektoren $c_1, \dots, c_k$ von C linear unabhängig sind, hat die zugehörige Generatormatrix maximalen Rang k . Daher ist die Abbildung $w \rightarrow (\langle w, c_1 \rangle, \dots, \langle w, c_k \rangle)$ ein Epimorphismus von K^n auf K^k mit Kern $= C^\perp$. Dann folgt aus dem Homomorphiesatz $n - \dim(C^\perp) = \dim(K^n) - \dim(\text{Kern}) = \dim(K^k) = k$.

Als Folgerung erhalten wir $C = C^{\perp\perp}$.

Offensichtlich ist nämlich $C \subseteq C^{\perp\perp}$ und wegen $\dim(C^{\perp\perp}) = n - \dim(C^\perp) = n - (n - \dim(C)) = \dim(C)$ folgt $C = C^{\perp\perp}$.

2.3.3 Kontrollmatrix

Sei wieder C ein linearer $[n, k, d]$ -Code über dem endlichen Körper K und $h_1 = (h_{11}, \dots, h_{1n}), \dots, h_{n-k} = (h_{(n-k)1}, \dots, h_{(n-k)n})$ eine Basis von $C^\perp$. Dann ist die Matrix

$$H = \begin{pmatrix} h_1 \\ \vdots \\ h_{n-k} \end{pmatrix} = \begin{pmatrix} h_{11} & \dots & h_{1n} \\ \vdots & & \vdots \\ h_{(n-k)1} & \dots & h_{(n-k)n} \end{pmatrix}$$

eine Generatormatrix von $C^\perp$. Wegen $C = C^{\perp\perp}$ ist $v = (x_1, \dots, x_n) \in K^n$ genau dann in C , wenn $\langle v, h_i \rangle = 0$ für $i = 1, \dots, n-k$. Man nennt daher H auch **Kontrollmatrix** für C und die $\langle v, h_i \rangle = x_1 h_{i1} + \dots + x_n h_{in} = 0$ **Kontrollgleichungen** für v .

Die Kontrollgleichungen kann man auch in Vektorschreibweise für die Spaltenvektoren $s_1, \dots, s_n$ der Kontrollmatrix H schreiben als $x_1 s_1 + \dots + x_n s_n = 0$. Es gelten mithin die beiden folgenden wichtigen Aussagen.

- Genau dann ist eine Matrix M Kontrollmatrix für C , wenn sie eine Generatormatrix von $C^\perp$ ist (nach Definition).
- Genau dann ist eine Matrix M Generatormatrix für C , wenn sie eine Kontrollmatrix von $C^\perp$ ist (wegen $C = C^{\perp\perp}$).

2.3.4 Zusammenhang zwischen erweitertem Code und dualem Code

Ist H eine Kontrollmatrix für den linearen $[n, k, d]_q$ -Code C , d. h. eine Generatormatrix von $C^\perp$, so ist

$$H^\wedge = \begin{pmatrix} 1 & 1 & \dots & 1 \\ & & & 0 \\ & H & & \vdots \\ & & & 0 \end{pmatrix}$$

eine Kontrollmatrix des erweiterten Codes $C^\wedge$, d. h. Generatormatrix von $(C^\wedge)^\perp$.

Man muss sich dazu überlegen, dass die Zeilenvektoren von $H^\wedge$ eine Basis von $(C^\wedge)^\perp$ bilden. Zunächst stimmt mal die Länge, denn $(C^\wedge)^\perp$ hat Länge $n + 1$. Nach Voraussetzung sind die Zeilenvektoren von H linear unabhängig, da H Generatormatrix von $C^\perp$ ist. Wegen der einzigen 1 in der letzten Spalte von $H^\wedge$ ist aber auch die erste Zeile $(1, \dots, 1)$ von $H^\wedge$ linear unabhängig von den restlichen Zeilenvektoren. Also sind alle Zeilenvektoren linear unabhängig und $H^\wedge$ erzeugt folglich einen Code der Dimension $n - k + 1$. Es gilt aber auch andererseits $\dim((C^\wedge)^\perp) = (n + 1) - \dim(C^\wedge) = (n + 1) - k = n - k + 1$. Jetzt muss man sich nur noch überlegen, dass alle Zeilenvektoren von $H^\wedge$ Skalarprodukt 0 mit den Elementen des Codes $C^\wedge$ bilden. Dies ist aber klar für die erste Zeile $(1, \dots, 1)$ von $H^\wedge$, da dies ja genau die Paritätsbedingung in der Definition von $C^\wedge$ ist, und auch für die übrigen Zeilen nach Definition von H als Kontrollmatrix von C und wegen der 0 in der letzten Spalte.

2.3.5 Beispiel: Erweiterte und duale Codes

Kontrollmatrizen und -gleichungen spielen eine wichtige Rolle bei der Decodierung von linearen Codes, wie wir im nächsten Abschnitt sehen werden. Jetzt ist es aber sicherlich Zeit für einige ausführliche Beispiele.

Der duale ISBN-Code

Der duale Code des ISBN-Codes hat Länge 10, Dimension 1 und Generatormatrix

$$H = \begin{pmatrix} 10 & 9 & \dots & 2 & 1 \end{pmatrix}.$$

Diese ist Kontrollmatrix des ISBN-Codes und liefert genau die Kontrollgleichung, über die der ISBN-Code definiert ist. Der duale Code hat zwar beeindruckendes Minimalgewicht $d = 10$, aber eine katastrophale Rate von $k/n = 1/10$.

Der kleine binäre Simplexcode $\text{Sim}_2(3)$ mit Parameter $[7, 3, 4]_2$

Wir haben im letzten Abschnitt eine Generatormatrix G_H für den kleinen binären Hamming-Code $\text{Ham}_2(3)$ erstellt, nämlich

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

Wir wollen nun den dualen Code zu $\text{Ham}_2(3)$ bestimmen. Das geht bei diesem kleinen Beispiel noch im Kopf. Wir suchen nämlich einen Code der Länge 7 und von Dimension 3, dessen Codewörter $(c_1, \dots, c_7)$ alle mit den Zeilenvektoren von G_H Skalarprodukt

0 bilden. Wie man leicht nachrechnet, erfüllen die Zeilenvektoren der Matrix

$$G_S = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

diese Bedingung und sind linear unabhängig. Daher ist G_S eine Generatormatrix von $\text{Ham}_2(3)^\perp$. Auch hier ist die Anzahl $2^3 - 1 = 7$ der Codewörter $\neq 0$ noch übersichtlich genug, um zu sehen, dass der Code Minimalabstand $d = 4$ hat. Er heißt **kleiner binärer Simplexcode** $\text{Sim}_2(3)$ und gehört zu einer Serie von Codes, die wir ebenfalls in Abschn. 3.1 näher betrachten werden. Natürlich ist G_S wiederum Kontrollmatrix zu $\text{Ham}_2(3)$.

Der erweiterte ternäre Golay-Code G_{12} mit Parameter $[12, 6, 6]_3$

Wir betrachten den erweiterten Code G_{12} zum ternären Golay-Code G_{11} , den wir im letzten Abschnitt kurz angerissen haben. Dieser hat offenbar Generatormatrix

$$G^\wedge = \begin{pmatrix} 1 & & & & 1 & 1 & 1 & 1 & 1 & 0 \\ & 1 & & & 0 & 1 & -1 & -1 & 1 & -1 \\ & & 1 & & 1 & 0 & 1 & -1 & -1 & -1 \\ & & & 1 & -1 & 1 & 0 & 1 & -1 & -1 \\ & & & & 1 & -1 & -1 & 1 & 0 & -1 \\ & & & & & 1 & 1 & -1 & -1 & 1 & 0 & -1 \end{pmatrix}$$

und ist daher ein sechsdimensionaler Code der Länge 12. Bilden wir nun die Skalarprodukte $\langle g_i, g_j \rangle$ der Zeilenvektoren $g_i (i = 1, \dots, 6)$ von $G^\wedge$, d. h. einer Basis von G_{12} , so stellen wir fest, dass stets $\langle g_i, g_j \rangle = 0$ gilt; also ist G_{12} selbstdual. Wir wollen noch das Minimalgewicht von G_{12} bestimmen. Sei dazu $c = (c_1, \dots, c_{12}) \in G_{12}$. Dann gilt einerseits $\langle c, c \rangle = 0$, da G_{12} selbstdual ist. Da andererseits die $c_i \neq 0$ alle entweder 1 oder -1 sind, gilt $0 = \langle c, c \rangle = c_1^2 + \dots + c_{12}^2 = \text{wt}(c) \cdot 1$ als Gleichung in $\mathbb{Z}_3$. Dies zeigt, dass $\text{wt}(c)$ ein Vielfaches von 3 sein muss. Man beachte nun, dass eine Linearkombination von zwei Zeilen von $G^\wedge$ ein Gewicht von genau 2 (aus den ersten sechs Positionen) plus mindestens 2 (aus den letzten sechs Positionen) hat, also insgesamt genau 6. Dies wiederum zeigt, dass eine Linearkombination von zwei Zeilen genau zweimal die 0 an den letzten sechs Positionen hat. Daraus folgt letztlich, dass eine Linearkombination von drei Zeilen von $G^\wedge$ ein Gewicht von genau 3 (aus den ersten sechs Positionen) plus mindestens 1 (aus den letzten sechs Positionen) hat, also insgesamt mindestens 6. Daher ist G_{12} ein selbstdualer $[12, 6, 6]_3$ -Code.

Der ternäre Golay-Code G_{11} mit Parameter $[11, 6, 5]_3$

Es steht aus dem letzten Abschnitt noch die Bestimmung des Minimalgewichts von G_{11} aus. Da aber G_{11} aus G_{12} durch Weglassen der letzten Position und damit der Paritätsziffer entsteht, hat G_{11} ein Minimalgewicht von $d = 5$.

Der erweiterte kleine binäre Hamming-Code $\text{Ham}_2(3)^\wedge$ mit Parameter $[8, 4, 4]_2$

Hier ist zunächst wieder eine Generatormatrix des erweiterten Codes:

$$G_H^\wedge = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Es handelt sich also um einen Code der Länge 8, von Dimension 4 und als binärer Code mit einem Minimalgewicht von $d^\wedge = 4$. Außerdem stellt man durch Skalarproduktbildung der Zeilen von $G_H^\wedge$ fest, dass der Code selbstdual ist. Insgesamt ist also $\text{Ham}_2(3)^\wedge$ ein selbstdualer $[8, 4, 4]_2$ -Code.

Man sieht also, wie trickreich man manchmal vorgehen muss, um gute lineare Codes zu konstruieren. In den nächsten Abschnitten werden wir sehen, wie man ganze Serien solcher Codes definieren kann. Jetzt aber erst mal eine Pause beim Konstruieren. Wir wollen uns im nächsten Abschnitt den Verfahren und Hindernissen beim Decodieren widmen.

2.4 Maximum-Likelihood- und Syndromdecodierung

2.4.1 Hamming-Decodierung

Sei C vorübergehend nur ein Blockcode der Länge n und Minimalabstand d über dem Alphabet A , $|A| = q$. Wir haben uns bereits in Abschn. 1.3 mit den Kugeln vom Radius e um ein Codewort $c \in C$ beschäftigt, nämlich $B_e(c) = \{v \in A^n \mid d(v, c) \leq e\}$. Ist dabei $2e + 1 \leq d$, so sind jeweils zwei solche Kugeln disjunkt. Auf diese Weise haben wir den Begriff der e -fehlerkorrigierenden Codes herausgearbeitet. Treten nämlich bei jedem Codewort höchstens e Fehler auf, so kann man ein empfangenes Wort v korrekt zu dem einzigen Codewort im Abstand höchstens e korrigieren. Leider weiß man aber in der Praxis nicht, ob wirklich bei jedem ankommenden Wort v maximal e Fehler aufgetreten sind. Dennoch muss man – in der Regel ein Computer – automatisch reagieren.

Man korrigiert v in *das* (oder wenn nicht eindeutig *ein*) Codewort, das an v am nächsten dran liegt, d. h. man sucht $c \in C$ mit $d(v, c) = \min\{d(v, c') \mid c' \in C\}$. Man nennt dieses Verfahren auch **Hamming-Decodierung**.

Bei BD-Decodierung würde man zu Codewörtern im Abstand von mehr als e jedenfalls nicht korrigieren. Dabei wissen wir, dass das Verfahren das richtige Codewort liefert, falls höchstens e Fehler aufgetreten sind. Aber macht dieses Vorgehen auch wirklich Sinn? Lösen wir uns mal von unseren bisherigen Betrachtungen und fragen uns, auf welche Weise wir eigentlich gerne decodieren würden.

2.4.2 Maximum-Likelihood-Decodierung

Wir würden gerne ein empfangenes Wort v zu dem Codewort $c \in C$ korrigieren, für das die Wahrscheinlichkeit P am höchsten ist, dass v aus ihm stammt. Formaler heißt dies für die bedingte Wahrscheinlichkeit $P(\cdot | \cdot)$, dass wir $c \in C$ suchen mit $P(v|c) = \max\{P(v|c') | c' \in C\}$. Dieses Vorgehen wird als **Maximum-Likelihood-Decodierung** bezeichnet.

Obwohl intuitiv richtig, bringt uns das aber zunächst nicht viel weiter, da wir die bedingten Wahrscheinlichkeiten gar nicht kennen. Im Gegenteil, hier tritt noch eine andere Fragestellung auf. Wenn z. B. ein deutscher Text gesendet wurde, so haben die Buchstaben im Text eine statistisch bekannte Häufigkeit. Wird also etwa ein V empfangen, das mit irgendeiner ausgefeilten Decodiermethode in den Buchstaben Q decodiert wird, ist denn dann die **Wahrscheinlichkeit**, dass V aus dem viel häufigeren E stammt, nicht **a priori** viel größer? Diese Frage stellt sich aber in der Praxis meist nicht. Denn der zu kanalcodierende String besteht aus einer langen Kette von – in der Regel binären – Ziffern, der die Quellencodierung nicht nur von Texten (d. h. Buchstaben), sondern auch von anderen, in Texten eingelagerten Objekten (Fotos, Grafiken etc.) darstellt. Dieser String wird bei der Kanalcodierung unabhängig von Sinninhalten jeweils in Blöcke zerschnitten und entsprechend dem verwendeten Code codiert.

Jetzt mag man einwenden, dass manchmal doch nur eine reine Textnachricht übertragen und jeder Buchstabe genau als einer dieser Blöcke quellencodiert wird. Hier noch ein weiteres Argument: Wie in Abschn. 1.1 beschrieben, werden Nachrichten häufig vorher mit Methoden der Kryptografie verschlüsselt. Ein wichtiges Merkmal dabei ist, dass statistische Häufigkeiten von Buchstaben *verschmiert* werden. Denn sonst wäre genau die Analyse der Häufigkeiten ein Angriffspunkt gegen die Chiffrierung. Zur Erinnerung: Die in der Einleitung erwähnte **Cäsar-Chiffre**, bei der jeder Buchstabe im Alphabet um drei Buchstaben nach hinten geschoben wird, kann man nämlich mit statistischen Auswertungen sofort knacken.

2.4.3 Hamming-Decodierung versus Maximum-Likelihood-Decodierung

Unsere Rettung aus dem Dilemma ist nun ein Resultat, das wiederum aus der **Informationstheorie** stammt und das im Wesentlichen besagt:

Für in der Praxis gängige Kanäle entspricht die **Hamming-Decodierung** dem eigentlich gewünschten **Maximum-Likelihood**-Verfahren.

Die Voraussetzung dieses Resultats und damit die Bedingung an den Kanal lauten konkret für ein Alphabet A mit $|A| = q$:

- Jedes $a \in A$ wird mit der gleichen Wahrscheinlichkeit $p < (q - 1)/2$ verfälscht.
- Die $q - 1$ Möglichkeiten, in die ein $a \in A$ verfälscht werden kann, sind alle gleich wahrscheinlich.

Für binäre Codes läuft dies also darauf hinaus, dass die Fehlerwahrscheinlichkeit pro Bit gleich $p < 1/2$ sein muss; man spricht dann von einem **binären symmetrischen Kanal**.

Wir wollen uns das Resultat für diesen Fall kurz überlegen. Dazu werde $c' = (c'_1, \dots, c'_n) \in C$ gesendet und $v = (v_1, \dots, v_n)$ empfangen mit Hamming-Abstand $d = d(v, c')$. Dann berechnet sich die bedingte Wahrscheinlichkeit $P(v|c')$ als

$$P(v|c') = \prod_i P(v_i|c'_i) = p^d (1 - p)^{n-d} = (p/(1 - p))^d (1 - p)^n.$$

Wegen $p < \frac{1}{2}$ ist der Wert $a = p/(1 - p)$ kleiner als 1 und somit die Funktion $f(x) = a^x$ monoton fallend, wie man aus der Analysis weiß. Daraus schließen wir, dass die Wahrscheinlichkeit $P(v|c) = \max\{P(v|c') | c' \in C\}$ genau für das (oder die) $c \in C$ ihr Maximum annimmt, für das (oder die) der Hamming-Abstand $d(v, c)$ minimal ist.

2.4.4 Syndromdecodierung

Wir sind damit zumindest so weit, dass unsere bisherigen Überlegungen sinnvoll waren und wir mit der Hamming-Decodierung weiterarbeiten können. Bei nichtlinearen Blockcodes bleibt beim Codieren daher nichts anderes übrig, als zu jedem empfangenen Wort v anhand der Liste aller Codewörter $c' \in C$ sämtliche Abstände $d(v, c')$ zu berechnen und ein c zu bestimmen, für das der Abstand minimal wird. Das ist aufwendig. Für lineare Codes geht das viel besser. Hierzu verwenden wir die Begriffe **Syndrom** und **Nebenklasse**, die wir jetzt erläutern wollen.

Sei wieder C ein $[n, k, d]$ -Code über dem endlichen Körper K , $|K| = q$ mit der Kontrollmatrix

$$H = \begin{pmatrix} h_1 \\ \vdots \\ h_{n-k} \end{pmatrix} = \begin{pmatrix} h_{11} & \dots & h_{1n} \\ \vdots & & \vdots \\ h_{(n-k)1} & \dots & h_{(n-k)n} \end{pmatrix}.$$

Für einen Vektor $v \in K^n$ bezeichnet man den Vektor $s = (s_1, \dots, s_{n-k})$ mit $s_i = \langle v, h_i \rangle$ als das **Syndrom** von v .

Wir erinnern uns, dass nach Definition der Kontrollmatrix v genau dann ein Codewort in C ist, wenn sein Syndrom gleich $(0, \dots, 0)$ ist. Im Übrigen: Für ein beliebiges $h \in C^\perp$, also h eine Linearkombination der h_i , spricht man bei $\langle v, h \rangle$ von einem **Kontrollwert** für v .

Weiterhin nennt man $\{v + c \mid c \in C\}$ eine **Nebenklasse** von C in K^n .

Wir überlegen uns zunächst, dass K^n die disjunkte Vereinigung von q^{n-k} solcher Nebenklassen ist. Wählt man nämlich ein $u \in K^n$, welches nicht in der Nebenklasse zu v liegt, so ist die entsprechende Nebenklasse $\{u + c \mid c \in C\}$ disjunkt dazu. Sonst gäbe es nämlich $c_1 \in C$ mit $v + c_1 = u + c_2$ und damit doch $u \in \{v + c \mid c \in C\}$. Auf diese Weise konstruieren wir uns wegen $|\{v + c \mid c \in C\}| = |C| = q^k$ sukzessive q^{n-k} solcher Nebenklassen.

Wegen $\langle v + c, h_i \rangle = \langle v, h_i \rangle$ für alle $c \in C$ ist das Syndrom für alle Elemente einer Nebenklasse gleich.

Man beachte aber, dass die Syndrome verschieden sind für verschiedene Nebenklassen. Sind nämlich für zwei Nebenklassen $\{v + c \mid c \in C\}$ und $\{u + c \mid c \in C\}$ die Syndrome gleich, so folgt $\langle v - u, h_i \rangle = 0$ für alle i und die Kontrollgleichungen erzwingen $v - u \in C$, d. h. v und u liegen in derselben Nebenklasse.

Jetzt können wir die **Syndromdecodierung** beschreiben. In jeder der Nebenklassen suchen wir dazu einen Vektor f mit minimalem Gewicht $wt(f)$ innerhalb seiner Nebenklasse. Sofern f nicht eindeutig ist, wählen wir einen davon aus und nennen f einen **Nebenklassenführer**. Zu jedem Nebenklassenführer berechnen wir außerdem noch sein Syndrom $(\langle f, h_1 \rangle, \dots, \langle f, h_{n-k} \rangle)$. Damit haben wir also eine Liste von insgesamt q^{n-k} Nebenklassenführern erstellt zusammen mit ihren jeweiligen Syndromen, die alle verschieden sind. Diese Rechnung ist zwar aufwendig, muss aber nur einmal für jeden Code gemacht werden und kann dann für jede weitere Decodierung genutzt werden.

Jetzt zur operativen Anwendung: Ein empfangenes Wort $v \in K^n$ muss ja in genau einer Nebenklasse liegen. Wir bilden daher sein Syndrom $(s_1, \dots, s_{n-k}) = (\dots, \langle v, h_i \rangle, \dots)$ und suchen in unserer Liste den zugehörigen Nebenklassenführer f mit genau diesem Syndrom. Dann korrigieren wir v zu $c = v - f \in C$. Da f von minimalem Gewicht gewählt wurde, hat also das Codewort c kleinstmöglichen Abstand $d(v, c) = d(c + f, c) = d(f, 0) = wt(f)$ vom empfangenen Wort v .

Falls man mit der Syndrommethode BD-Decodierung (Bounded Distance) betreiben will, d. h. nur bis zur Fehlerkorrekturkapazität e eines Codes korrigieren möchte, so sepa-

riert man von vornherein alle Nebenklassenführer f , die ein Gewicht $wt(f) > e$ haben. Ermittelt man dann bei der Syndromdecodierung ein solches f , so korrigiert man nicht und markiert ggf. das empfangene Wort v entsprechend.

2.4.5 Beispiel: Syndromdecodierung

Ein kleiner binärer linearer Code mit Parameter [5,2,3]

Wir schauen uns nun ein konkretes Beispiel an, das übersichtlich genug ist, die einzelnen Schritte halbwegs im Kopf nachzuverfolgen. Sei dazu C ein binärer Code mit Kontrollmatrix

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

Wie man nachrechnet, ist $C = \{(0, 0, 0, 0, 0), (1, 0, 1, 1, 0), (0, 1, 0, 1, 1), (1, 1, 1, 0, 1)\}$ ein $[5, 2, 3]_2$ -Code. Er hat $2^{5-2} = 2^3 = 8$ Nebenklassen und ebenso viele Nebenklassenführer. Jede Nebenklasse hat $2^2 = 4$ Elemente, C selbst ist eine davon mit Nebenklassenführer $(0, 0, 0, 0, 0)$. Da $(1, 0, 0, 0, 0)$ nicht in C liegt, ist $\{(1, 0, 0, 0, 0), (0, 0, 1, 1, 0), (1, 1, 0, 1, 1), (0, 1, 1, 0, 1)\}$ eine weitere Nebenklasse mit Nebenklassenführer $(1, 0, 0, 0, 0)$ und Syndrom $(1, 1, 0)$. So bildet man sukzessive alle acht Nebenklassen. Ein bisschen Rechnung liefert letztlich folgende Liste von Nebenklassenführern mit zugehörigem Syndrom.

$$\begin{aligned} f_1 &= (0, 0, 0, 0, 0) && \text{mit Syndrom } (0, 0, 0). \\ f_2 &= (1, 0, 0, 0, 0) && \text{mit Syndrom } (1, 1, 0). \\ f_3 &= (0, 1, 0, 0, 0) && \text{mit Syndrom } (0, 1, 1). \\ f_4 &= (0, 0, 1, 0, 0) && \text{mit Syndrom } (1, 0, 0). \\ f_5 &= (0, 0, 0, 1, 0) && \text{mit Syndrom } (0, 1, 0). \\ f_6 &= (0, 0, 0, 0, 1) && \text{mit Syndrom } (0, 0, 1). \\ f_7 &= (1, 1, 0, 0, 0) && \text{mit Syndrom } (1, 0, 1). \\ f_8 &= (0, 1, 1, 0, 0) && \text{mit Syndrom } (1, 1, 1). \end{aligned}$$

Wird also etwa $v = (0, 1, 0, 1, 1)$ empfangen, so ist sein Syndrom $(0, 0, 0)$ und wir korrigieren v zu $v + f_1 = v$, d.h. v war bereits ein Codewort. Das Wort $v = (1, 1, 1, 1, 1)$ mit Syndrom $(0, 1, 0)$ korrigieren wir zum Codewort $v + f_5 = (1, 1, 1, 0, 1)$ und $v = (1, 0, 0, 1, 1)$ mit Syndrom $(1, 0, 1)$ zu $v + f_7 = (0, 1, 0, 1, 1)$.

Wegen Minimalabstand $d = 3$ ist C nur 1-fehlerkorrigierend. Bei BD-Decodierung würde man also im Falle der Nebenklassenführer f_7 und f_8 nicht korrigieren. Das empfangene Wort $v = (1, 0, 0, 1, 1)$ hat nämlich auch Abstand 2 zum Codewort $(1, 0, 1, 1, 0)$.

2.4.6 Komplexität: Hamming- und Syndromdecodierung

Ein wichtiges Kriterium für die Güte eines Decodierverfahrens ist seine **Komplexität**, d. h. die Größenordnung der benötigten elementaren arithmetischen Operationen und Vergleiche **asymptotisch** betrachtet für große Codelängen n . Komplexitäten benennen wir stets mit dem Symbol „ $\sim$ “. Man muss allerdings stets dabei beachten, dass die Komplexität nur einen Anhaltspunkt für die Schnelligkeit eines Verfahrens liefert. Mitentscheidend ist aber letztlich auch immer, wie die einzelnen Operationen in ihrem jeweiligen Kontext rechnerimplementierbar sind und wie viele Speicherzugriffe dafür benötigt werden.

Betrachten wir zunächst die Hamming-Decodierung eines $[n, k, d]_q$ -Codes C . Für ein empfangenes Wort v müssen wir hierbei für jedes der q^k Codewörter $c \in C$ die Abstände $d(v, c)$ bestimmen; dies erfordert jeweils n Vergleiche bzw. Differenzbildungen mit Überprüfung auf $\neq 0$. Insgesamt benötigt die Hamming-Decodierung also nq^k elementare Operationen, d. h. die Komplexität ist $\sim nq^k$.

Bei der Syndromdecodierung muss man zunächst alle $\langle v, h_i \rangle$ berechnen, d. h. $2(n-k)n$ elementare Operationen ausgeführt. Das Syndrom muss man anschließend mit den Syndromen der q^{n-k} Nebenklassenführer f vergleichen, was insgesamt $(n-k)q^{n-k}$ elementare Operationen erfordert. Die Korrektur von v durch $v - f$ benötigt nur n elementare Operationen. Dies und den quadratischen Term $2(n-k)n$ kann man asymptotisch vernachlässigen. Daher ist bei Syndromdecodierung die Komplexität insgesamt $\sim (n-k)q^{n-k}$. Für Codes mit guter Rate k/n (d. h. k deutlich größer als $n/2$) ist somit die Syndromdecodierung wesentlich schneller.

Wir haben allerdings auch oben gesehen, dass die Syndromdecodierung selbst bei vergleichsweise kleinen Codes immer noch recht aufwendig ist. Um einen Eindruck vom benötigten Speicherplatzbedarf zu bekommen, nehmen wir mal einen binären Code C der Länge 70 und von Dimension 50, also mit Parameter $[70, 50]$. C hat dann 2^{20} Nebenklassenführer, jeder Nebenklassenführer mit seinem Syndrom benötigt 90 Bits. Daher ist der Speicherplatzbedarf $90 \cdot 2^{20}$ Bits ~ 12 MBytes. Speichert man jedoch für die Hamming-Decodierung alle 2^{50} Codewörter, so hat man einen Speicherplatzbedarf von $70 \cdot 2^{50}$ Bits ~ 9000 TBytes. Dies zeigt die Notwendigkeit, gezielt nach Codes zu suchen, die effiziente Decodieralgorithmen besitzen.

2.5 Codierung linearer Codes und Gesamtszenario

2.5.1 Generatormatrix in systematischer Form

Sei also wieder C ein linearer $[n, k, d]$ -Code über dem endlichen Körper K mit Generatormatrix

$$G = \begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix} = \begin{pmatrix} g_{11} & \cdots & g_{1n} \\ \vdots & & \vdots \\ g_{k1} & \cdots & g_{kn} \end{pmatrix}.$$

Wir nutzen nun folgende beiden Tatsachen.

- Mit $g_1, \dots, g_k$ ist auch $x_1 g_1, g_2 + x_2 g_1, \dots, g_k + x_k g_1$ (für $x_i \in K, x_1 \neq 0$) wieder eine Basis von C .
- Die Vertauschung der Stellen von C führt zu einem äquivalenten Code und ändert dessen Parameter nicht.

Da g_1 mindestens eine Komponente $\neq 0$ hat, können wir diese durch Stellenvertauschung und damit Übergang zu einem äquivalenten Code an die Position g_{11} bringen. Nun addieren wir Vielfache von g_1 zu den g_i derart, dass $g_{11} = 1$ und $g_{21} = \dots = g_{k1} = 0$ gilt. Dieses Verfahren wenden wir nun auf die zweite Zeile der so entstandenen Matrix an und weiter sukzessive bis zur letzten Zeile. Auf diese Weise haben wir uns folgende Tatsache überlegt:

Jede Generatormatrix lässt sich in die sog. **systematische Form** (oder Standardform) $G_{SF} = (E_k | P)$ überführen. Hierbei bezeichnet E_k die Einheitsmatrix mit 1 auf der Diagonalen und 0 sonst und P ist eine Matrix mit k Zeilen und $n - k$ Spalten.

2.5.2 Systematische Codierung

Wie versprochen, kommen wir jetzt nochmals auf die Frage nach der Wahl einer bestmöglichen Codierfunktion zurück. Üblicherweise verwendet man nämlich für die zu codierenden Informationseinheiten B ebenfalls Tupel über dem gleichen Alphabet $A = K$, mit dem der lineare Code C gebildet wird. Hat C die Dimension k , so kann man damit offenbar Informationsumfänge $B \subseteq K^k$ codieren, wobei die Elemente aus B also k -Tupel über dem endlichen Körper K sind. Die Codierfunktion muss daher *einigen* der k -Tupel $(i_1, \dots, i_k)$ über K jeweils ein Codewort zuordnen. Da man die Codierfunktion aber **algorithmisch** definieren möchte, ordnet man dabei in der Praxis üblicherweise *jedem* k -Tupel $(i_1, \dots, i_k)$ ein Codewort $c \in C$ zu, unabhängig davon, ob dieses Tupel $(i_1, \dots, i_k)$ wirklich mit einer Information aus B belegt ist oder nicht.

Zum Codieren benötigen wir eine eindeutige Abbildungsvorschrift $K^k \rightarrow C \subseteq K^n$. Wir nutzen dazu unsere Generatormatrix G , bei der ja die Zeilenvektoren $g_1, \dots, g_k$ eine Basis von C bilden. Als Codierfunktion wählen wir dann die Zuordnung $(i_1, \dots, i_k) \rightarrow i_1 g_1 + \dots + i_k g_k$.

Was ist aber nun der Vorteil der systematischen Form einer Generatormatrix?

Wenn die Generatormatrix von C in systematischer Form $G_{SF} = (E_k | P)$ vorliegt, so sieht die Codierfunktion konkreter so aus: $(i_1, \dots, i_k) \rightarrow (i_1, \dots, i_k, p_1, \dots, p_{n-k})$. Dabei stehen also die **Informationsziffern** $i_1, \dots, i_k$ an den ersten k Positionen und die **Paritätsziffern** $p_1, \dots, p_{n-k}$ berechnen sich aus den Informationsziffern und den Einträgen der Matrix P . Eine solche Codierfunktion bezeichnet man als **systematisch**.

Eingangsbeispiele und ternärer Golay-Code aus Abschn. 2.2 sind systematisch.

2.5.3 Kanalcodierung und -decodierung – Gesamtszenario

Wir haben nun bereits einige Codes konkret konstruiert, haben uns überlegt, dass man mittels Generatormatrix in systematischer Form gut codieren kann, und haben auch schon Verfahren zur Decodierung gesehen. Deshalb wollen wir hier nochmals auf Abschn. 1.1 zurückkommen und das Gesamtszenario fehlerkorrigierender Codes besprechen. In unserem Modell besteht die Information aus k -Tupeln über einem Körper K (häufig binär), die dann in einen fehlerkorrigierenden Code C , nämlich einen Unterraum des Vektorraums der n -Tupel K^n ($n > k$), kanalcodiert werden. Bevor aber wieder nach Empfang kanaldcodiert werden kann, d. h. Codewörter in $C \subseteq K^n$ wieder zurück nach K^k konvertiert werden können, müssen ja zunächst die Übertragungsfehler in einem empfangenen n -Tupel $v \in K^n$ korrigiert werden, d. h. v muss zu einem gültigen Codewort abgeändert werden. Auch diesen Schritt bezeichnet man lax als *Decodieren* und meint damit das Korrigieren von Fehlern mittels eines fehlerkorrigierenden Codes, also z. B. Hamming- und Syndromdecodierung. In Abb. 2.2 ist der Ablauf nochmals zusammengefasst.

Algorithmen zum Decodieren im Sinne der Fehlerkorrektur liegen nicht ganz so auf der Hand, wie wir gesehen haben und noch sehen werden. Das Kanalcodieren und auch -decodieren ist in der Regel durch die Definition des Codes, z. B. die Generatormatrix in systematischer Form, unmittelbar gegeben. Dennoch sucht man auch hier nach sehr effizienter Implementierung, um die Gesamtlaufzeit zu verkürzen. Optimal hierzu sind Schieberegister, auf die wir in Abschn. 6.2 näher eingehen werden.



Abb. 2.2 Gesamtszenario der Kanalcodierung und -decodierung

2.5.4 Beispiel: Gesamtszenario der Kanalcodierung und -decodierung

Der kleine binäre Hamming-Code $Ham_2(3)$

Zur Illustration des Ablaufs wieder ein Beispiel, nämlich der kleine binäre Hamming-Code $Ham_2(3)$, aber in systematischer Form. Hierzu betrachten wir zunächst einen zu unserem Simplexcode $Sim_2(3)$ äquivalenten Code mit folgender Generatormatrix:

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

Der duale Code $Ham_2(3)$ ist dann in systematischer Form mit Generatormatrix:

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

Damit wird die vierstellige Information (i_1, i_2, i_3, i_4) codiert in $(i_1, i_2, i_3, i_4, p_1 = i_1 + i_2 + i_4, p_2 = i_1 + i_3 + i_4, p_3 = i_2 + i_3 + i_4)$ mit den drei Paritätsbits p_1, p_2 und p_3 . Abb. 2.3 zeigt die zugehörige digitale Schaltlogik.

Nach dem Senden werde das Wort $(i'_1, i'_2, i'_3, i'_4, p'_1, p'_2, p'_3)$ empfangen. Nun kommt unsere Fehlerkorrektur zum Tragen, z. B. das Syndromverfahren. Im Fall des kleinen binären Hamming-Codes berechnet sich das Syndrom (s_1, s_2, s_3) gemäß $s_1 = i'_1 + i'_2 + i'_4 + p'_1$, $s_2 = i'_1 + i'_3 + i'_4 + p'_2$ und $s_3 = i'_2 + i'_3 + i'_4 + p'_3$. Die Korrekturlogik des Syndromverfahrens entscheidet nun, ob bzw. welche der Informationsbits i'_1, i'_2, i'_3, i'_4 geändert werden müssen.

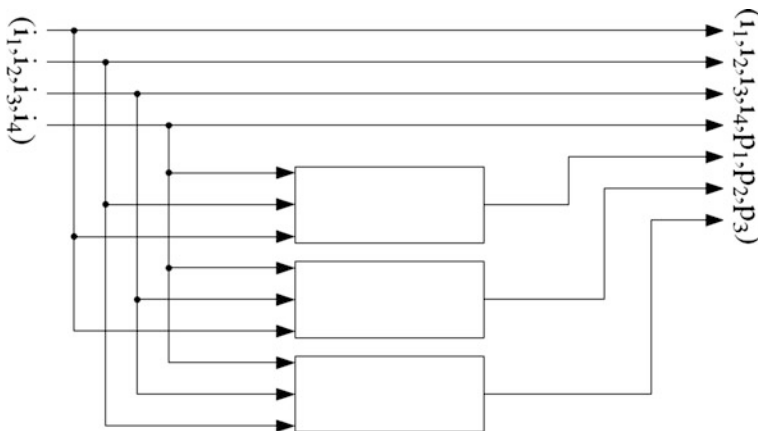


Abb. 2.3 Schaltlogik für die Codierung des kleinen binären Hamming-Codes

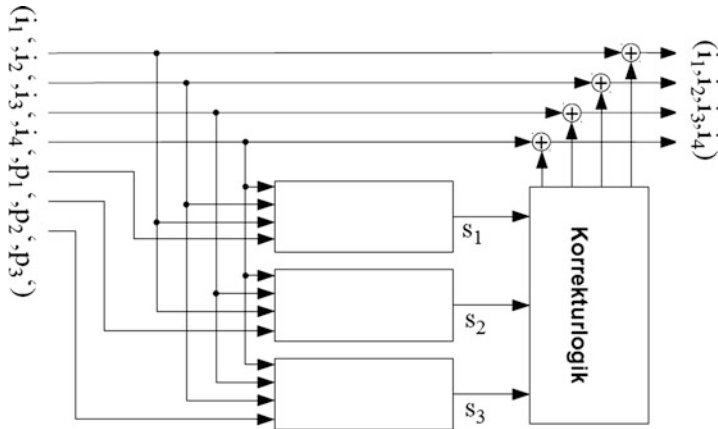


Abb. 2.4 Schaltlogik für die Decodierung des kleinen binären Hamming-Codes

Da der Hamming-Code Minimalabstand 3 hat, wird ja nur bei einem Fehler korrekt korrigiert. Das Kanaldecodieren ist dann wiederum sehr einfach, da durch die systematische Form der Generatormatrix die Informationsbits genau an den ersten vier Stellen ablesbar sind. Abb. 2.4 zeigt die zugehörige digitale Logik.

Fehlerkorrigierende Codes

Konstruieren, Anwenden, Decodieren

Manz, O.

2017, VIII, 279 S. 85 Abb., 25 Abb. in Farbe., Softcover

ISBN: 978-3-658-14651-1