Überblick zu Kapitel 1 von „Einführung in die Kanalcodierung"

Das erste Kapitel behandelt **Blockcodes** zur Fehlererkennung und Fehlerkorrektur und liefert die Grundlagen zur Beschreibung effektiverer Codes wie zum Beispiel die *Reed–Solomon–Codes* (siehe Kapitel 2), die *Faltungscodes* (Kapitel 3) sowie die iterativ decodierbaren *Produkt–* (*Turbo–Codes*) und *Low–density Parity–check Codes* (Kapitel 4). Wir beschränken uns hier auf binäre Codes.

Man bezeichnet dieses spezifische Fachgebiet als **Kanalcodierung** im Gegensatz zur **Quellencodierung** (Redundanzminderung aus Gründen der Datenkomprimierung) und zur **Leitungscodierung** (zusätzliche Redundanz zur Anpassung des Digitalsignals an die spektralen Eigenschaften des Übertragungsmediums).

Im Einzelnen werden behandelt:

- Definitionen und einführende Beispiele zur *Fehlererkennung* und *Fehlererkorrektur*,
- eine kurze Wiederholung geeigneter *Kanalmodelle* und *Entscheiderstrukturen*,
- bekannte *binäre Blockcodes* wie Single Parity-check–, Wiederholungs– und Hamming–Code,
- die allgemeine Beschreibung linearer Codes mittels *Generatormatrix* und *Prüfmatrix*,
- die Decodiermöglichkeiten für Blockcodes, unter anderem die *Syndromdecodierung*,
- einfache Näherungen und obere Schranken für die *Blockfehlerwahrscheinlichkeit*, sowie
- eine *informationstheoretische Grenze* der Kanalcodierung.

Geeignete Literatur: [Böch15] – [Bos98] – [Bos99] – [CT06] – [Fri96] – [Gal68] – [Hag04] – [Hin02] – [JZ99] – [KM+08] – [Liv10] – [RL09] – [Sha48] – [Trö74]

Die grundlegende Theorie wird auf 60 Seiten dargelegt. Dazu beinhaltet dieses Kapitel noch 80 Grafiken, 17 Aufgaben und zehn Zusatzaufgaben mit insgesamt 143 Teilaufgaben, sowie zwei Lernvideos (LV) und sieben Interaktionsmodule (IM):

- **Eigenschaften des Übertragungskanals** (LV zu Kap. 1.2 – Dauer 5:50)
- **Komplementäre Gaußsche Fehlerfunktionen** (IM zu Kap. 1.2)
- **Symbolfehlerwahrscheinlichkeit von Digitalsystemen** (IM zu Kap. 1.2)
- **WDF, VTF und Momente** (IM zu Kap. 1.2)
- **Ereigniswahrscheinlichkeiten der Binomialverteilung** (IM zu Kap. 1.3)
- **Galoisfeld: Eigenschaften und Anwendungen** (LV zu Kap. 1.4 – 3–teilig, Dauer: ca. 39:00)
- **Gram–Schmidt–Verfahren** (IM zu Kap. 1.4: Herleitung der Basisfunktionen)
- **Mehrstufiges PSK und Union Bound** (IM zu Kap. 1.6: Union Bound)

Fehlererkennung und Fehlerkorrektur

Bei einem jeden Nachrichtenübertragungssystem kommt es zu Übertragungsfehlern. Man kann zwar die Wahrscheinlichkeit p_S für einen solchen Symbolfehler sehr klein halten, zum Beispiel durch eine sehr große Signalenergie. Die Symbolfehlerwahrscheinlichkeit $p_S = 0$ ist aber wegen der Gaußschen WDF des stets vorhandenen thermischen Rauschens nie erreichbar.

Insbesondere bei stark gestörten Kanälen und auch für sicherheitskritische Anwendungen ist es deshalb unumgänglich, die zu übertragenden Daten angepasst an Anwendung und Kanal besonders zu schützen. Dazu fügt man beim Sender Redundanz hinzu und nutzt diese Redundanz beim Empfänger, um die Anzahl der Decodierfehler zu verringern. Dabei unterscheidet man:

- **Fehlererkennung** (englisch: *Error Detection*): Der Decoder prüft die Integrität der empfangenen Blöcke und markiert gefundene Fehler. Eventuell informiert der Empfänger den Sender über fehlerhafte Blöcke via Rückkanal, so dass dieser den entsprechenden Block noch einmal sendet.
- **Fehlerkorrektur** (englisch: *Error Correction*): Der Decoder erkennt einen (oder mehrere) Bitfehler und liefert für diese weitere Informationen, z.B. deren Positionen im übertragenen Block. Damit können unter Umständen die entstandenen Fehler vollständig korrigiert werden.

Die **Kanalcodierung** (englisch: *Channel Coding* oder auch *Error–Control Coding*) umfasst sowohl Verfahren zur Fehlererkennung als auch solche zur Fehlerkorrektur.

Alle ARQ–Verfahren (englisch: *Automatic Repeat Request*) nutzen ausschließlich Fehlererkennung. Für die Fehlererkennung ist weniger Redundanz erforderlich als für eine Fehlerkorrektur. Ein Nachteil der ARQ ist der geringe Durchsatz bei schlechter Kanalqualität, also dann, wenn häufig ganze Datenblöcke vom Empfänger neu angefordert werden müssen.

In diesem Buch behandeln wir größtenteils die **Vorwärtsfehlerkorrektur** (englisch: *Forward Error Correction*, FEC), die bei einem ausreichend guten Kanal (großes SNR) zu sehr kleinen Fehlerraten führt. Bei schlechteren Kanalbedingungen ändert sich am Durchsatz nichts, das heißt, es wird die gleiche Informationsmenge übertragen. Allerdings kann dann die Fehlerrate sehr große Werte annehmen.

Oft werden FEC– und ARQ–Verfahren kombiniert, und zwischen diesen die Redundanz so aufgeteilt,

- dass eine kleine Anzahl von Fehlern noch korrigierbar ist,
- bei vielen Fehlern aber eine Wiederholung des Blocks angefordert wird.

Einige einführende Beispiele (1)

Es folgen zunächst einige Beispiele zur **Fehlererkennung**. Auf der nächsten Seite werden dann ein paar Einsatzgebiete von Fehlerkorrektur genannt.

Single Parity Check Code (SPC)

Ergänzt man $k = 4$ Bit um ein sog. Prüfbit (englisch: *Parity Bit*) derart, dass die Summe aller Einsen geradzahlig ist, zum Beispiel (mit rotem Prüfbit)

0000**0**, 0001**1**, ... , 1111**0**, ...

so kann man einen Einzelfehler sehr einfach erkennen. Zwei Fehler innerhalb eines Codewortes bleiben dagegen unerkannt. Die deutsche Bezeichnung ist *Paritätsprüfcode*.

International Standard Book Number (ISBN)

Seit den 1960er Jahren werden alle Bücher mit 10–stelligen Kennzahlen (ISBN–10) versehen. Seit 2007 ist zusätzlich noch die Angabe entsprechend ISBN–13 verpflichtend. Beispielsweise lauten diese für das Fachbuch **[Söd93]:**

3–540–57215–5 (ISBN–10) bzw. **978–3–54057215–2** (ISBN–13).

Die letzte Ziffer z_{10} (rot markiert) ergibt sich bei ISBN–10 aus den vorherigen Ziffern $z_1 = 3$, $z_2 = 5$, ... , $z_9 = 5$ nach folgender Rechenregel:

$$z_{10} = \left(\sum_{i=1}^{9} i \cdot z_i\right) \bmod 11 = (1 \cdot 3 + 2 \cdot 5 + ... + 9 \cdot 5) \bmod 11 = 5.$$

Zu beachten ist, dass $z_{10} = 10$ als z_{10} = „X" geschrieben werden muss (römische Zahlendarstellung von „10"), da sich die Zahl 10 im Zehnersystem nicht als Ziffer darstellen lässt.

Entsprechend gilt für die Prüfziffer bei ISBN–13:

$$\begin{aligned} z_{13} &= 10 - \left(\sum_{i=1}^{12} z_i \cdot 3^{(i+1) \bmod 2}\right) \bmod 10 = \\ &= 10 - \left[(9 + 8 + 5 + 0 + 7 + 1) \cdot 1 + (7 + 3 + 4 + 5 + 2 + 5) \cdot 3\right] \bmod 10 = \\ &= 10 - (108 \bmod 10) = 10 - 8 = 2. \end{aligned}$$

Bei beiden Varianten werden im Gegensatz zum obigen Paritätsprüfcode (SPC) auch Zahlendreher wie 57 ⇒ 75 erkannt, da hier unterschiedliche Positionen verschieden gewichtet werden.

Strichcode (eindimensionaler Barcode)

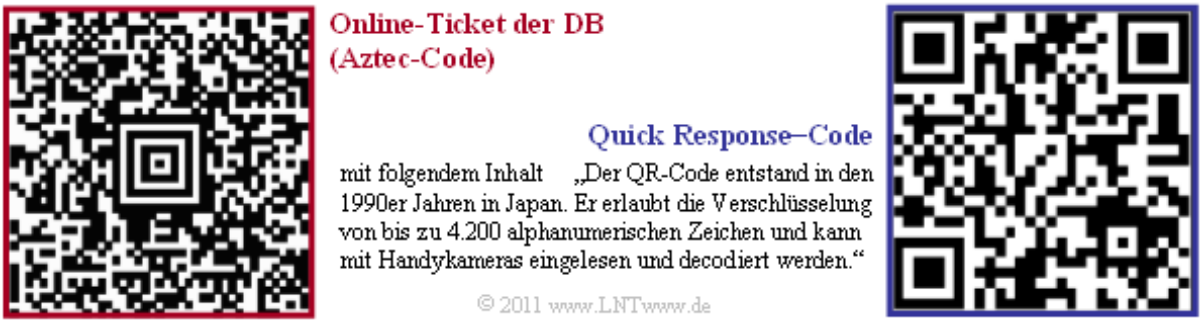
Der am weitesten verbreitete fehlererkennende Code weltweit ist der Strichcode oder Balkencode (englisch: *Bar Code*) zur Kennzeichnung von Produkten, zum Beispiel EAN–13 (*European Article Number*) mit 13 Ziffern. Diese werden durch verschieden breite Balken und Lücken dargestellt und können mit einem opto–elektronischen Lesegerät leicht entschlüsselt werden. Die ersten drei Ziffern kennzeichnen das Land (beispielsweise Deutschland: 400 – 440), die nächsten vier bzw. fünf Stellen den Hersteller und das Produkt. Die letzte Ziffer ist die Prüfziffer z_{13}, die sich genau so berechnet wie bei ISBN–13.

Einige einführende Beispiele (2)

Es folgen noch einige Beispiele zur **Fehlerkorrektur**. Genauere Details hierzu finden Sie in **[KM+09].**

2D–Barcodes für Online–Tickets

Wenn Sie eine Fahrkarte der Deutschen Bahn online buchen und ausdrucken, finden Sie ein Beispiel eines zweidimensionalen Barcodes, nämlich den 1995 von Andy Longacre bei der Firma Welch Allyn in den USA entwickelten Aztec–Code, mit dem Datenmengen bis zu 3000 Zeichen codiert werden können. Aufgrund der Reed–Solomon–Fehlerkorrektur ist die Rekonstruktion des Dateninhalts auch dann noch möglich, wenn bis zu 40% des Codes zerstört wurden, zum Beispiel durch Knicken der Fahrkarte oder durch Kaffeeflecken.



Rechts ist ein QR–Code (*Quick Response*) mit zugehörigem Inhalt dargestellt. Der QR–Code wurde 1994 für die Autoindustrie in Japan zur Kennzeichnung von Bauteilen entwickelt und erlaubt ebenfalls eine Fehlerkorrektur. Inzwischen ist der Einsatz des QR–Codes sehr vielfältig. In Japan findet man ihn auf nahezu jedem Werbeplakat und auf jeder Visitenkarte. Auch in Deutschland setzt er sich mehr und mehr durch.

Bei allen 2D–Barcodes gibt es quadratische Markierungen zur Kalibrierung des Lesegerätes.

Codes für die Satelliten– und Weltraumkommunikation

Eines der ersten Einsatzgebiete von Fehlerkorrekturverfahren war die Kommunikation von/zu Satelliten und Raumfähren, also Übertragungsstrecken, die durch niedrige Sendeleistungen und große Pfadverluste gekennzeichnet sind. Schon 1977 wurde bei der *Raum–Mission Voyager 1* zu Neptun und Uranus eine Kanalcodierung eingesetzt, und zwar in Form der seriellen Verkettung eines **Reed–Solomon–Codes** und eines **Faltungscodes.**

Damit genügte schon der Leistungskennwert $10 \cdot \lg E_B/N_0 \approx 2$ dB, um die geforderte Fehlerrate $5 \cdot 10^{-5}$ (bezogen auf die komprimierten Daten nach der Quellencodierung) zu erreichen. Ohne Kanalcodierung sind dagegen für die gleiche Fehlerrate fast 9 dB erforderlich, also eine um den Faktor $10^{0.7} \approx 5$ größere Sendeleistung.

Auch das geplante Marsprojekt (die Datenübertragung vom Mars zur Erde mit 5W–Lasern) wird nur mit einem ausgeklügelten Codierschema erfolgreich sein.

Die Aufzählung wird auf der nächsten Seite fortgesetzt.

Einige einführende Beispiele (3)

Kanalcodes für die Mobilkommunikation

Ein weiteres umsatzstarkes Anwendungsgebiet, das ohne Kanalcodierung nicht funktionieren würde, ist die **Mobilkommunikation.** Hier ergäben sich bei ungünstigen Bedingungen ohne Codierung Fehlerraten im Prozentbereich und aufgrund von Abschattungen und Mehrwegeausbreitung (Echos) treten die Fehler oft gebündelt auf. Die Fehlerbündellänge beträgt dabei manchmal einige hundert Bit.

- Bei der Sprachübertragung im **GSM–System** werden die 182 wichtigsten (Klasse 1a und 1b) der insgesamt 260 Bit eines Sprachrahmens (20 ms) zusammen mit einigen wenigen Paritäts– und Tailbits faltungscodiert (mit Memory $m = 4$ und Rate $R = 1/2$) und verwürfelt. Zusammen mit den 78 weniger wichtigen und deshalb uncodierten Bits der Klasse 2 führt dies dazu, dass die Bitrate von 13 kbit/s auf 22.4 kbit/s ansteigt.
- Man nutzt die (relative) Redundanz von $r = (22.4 – 13)/22.4 \approx 0.42$ zur Fehlerkorrektur. Anzumerken ist, dass „$r = 0.42$" aufgrund der hier verwendeten Definition aussagt, das 42% der codierten Bits redundant sind. Mit dem Bezugswert „Bitrate der uncodierten Folge" ergäbe sich $r = 9.4/13 = 0.72$ mit der Aussage: Zu den Informationsbits werden 72% Prüfbits hinzugefügt.
- Bei **UMTS** (*Universal Mobile Telecommunications System*) werden **Faltungscodes** mit den Raten $R = 1/2$ bzw. $R = 1/3$ eingesetzt. Bei den UMTS–Modi für höhere Datenraten und entsprechend geringeren Spreizfaktoren verwendet man dagegen **Turbo–Codes** der Rate $R = 1/3$ und iterative Decodierung. Abhängig von der Anzahl der Iterationen erzielt man hier gegenüber der Faltungscodierung Gewinne von bis zu 3 dB.

Fehlerschutz der Compact Disc

Bei einer CD (*Compact Disc*) verwendet man einen *cross–interleaved* **Reed–Solomon–Code** (RS) und anschließend eine sogenannte **Eight–to–Fourteen–Modulation.** Die Redundanz nutzt man zur Fehlererkennung und –korrektur. Dieses Codierschema zeigt folgende Charakteristika:

- Die gemeinsame Coderate der zwei RS–Komponentencodes beträgt $R_{\rm RS} = 24/28 \cdot 28/32 = 3/4$. Durch die 8–to–14–Modulation und einiger Kontrollbits kommt man zur Gesamtcoderate $R \approx 1/3$.
- Bei statistisch unabhängigen Fehlern gemäß dem BSC–Modell (*Binary Symmetric Channel*) ist eine vollständige Korrektur möglich, so lange die Bitfehlerrate den Wert 10^{-3} nicht überschreitet.
- Der CD–spezifische *Cross Interleaver* verwürfelt 108 Blöcke miteinander, so dass die 588 Bit eines Blockes (jedes Bit entspricht ca. 0.28 µm) auf etwa 1.75 cm verteilt werden.
- Mit der Coderate $R \approx 33\%$ kann man ca. 10% Erasures korrigieren und die verloren gegangenen Werte lassen sich durch Interpolation (näherungsweise) rekonstruieren $\Rightarrow$ *Fehlerverschleierung*.

Zusammenfassend lässt sich sagen: Weist eine CD einen Kratzer von 1.75 mm Länge in Abspielrichtung auf (also mehr als 6000 aufeinanderfolgende Erasures), so sind immer noch 90% aller Bit eines Blockes fehlerfrei, so dass sich auch die fehlenden 10% rekonstruieren lassen, oder dass die Auslöschungen zumindest so verschleiert werden können, dass sie nicht hörbar sind.

Auf der nächsten Seite folgt eine Demonstration zur Korrekturfähigkeit der CD.

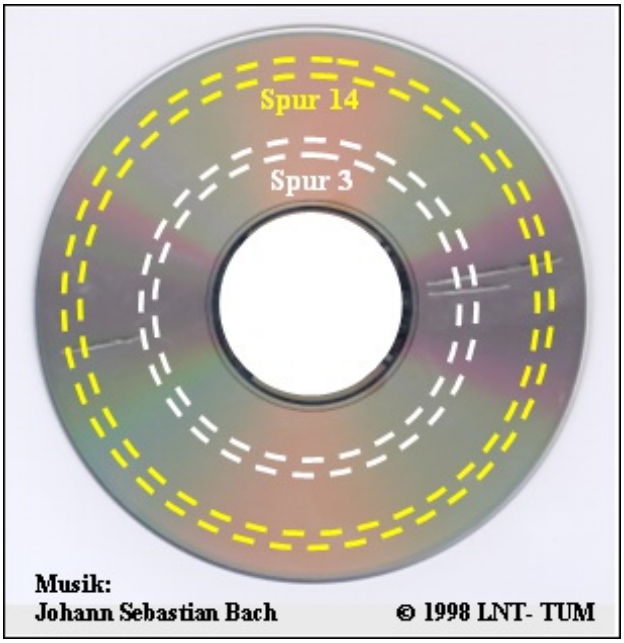
Die „Geschlitzte CD" – eine Demonstration des LNT der TUM

Ende der 1990er Jahre haben Mitarbeiter des Lehrstuhls für Nachrichtentechnik der TU München unter Leitung von **Professor Joachim Hagenauer** eine Musik–CD gezielt beschädigt, indem insgesamt drei Schlitze von jeweils mehr als einem Millimeter Breite eingefräst wurden. Damit fehlen bei jedem Defekt fast 4000 fortlaufende Bit der Audiocodierung.

Die Grafik zeigt die „geschlitzte CD". Sowohl in der Spur 3 als auch in der Spur 14 gibt es bei jeder Umdrehung zwei solcher fehlerhafter Bereiche. Sie können sich die Musikqualität mit Hilfe der beiden Audioplayer (Abspielzeit jeweils ca. 15 Sekunden) verdeutlichen. Die Theorie zu dieser Audio–Demo finden Sie auf der **vorherigen Seite.**


Spur 14


Spur 3



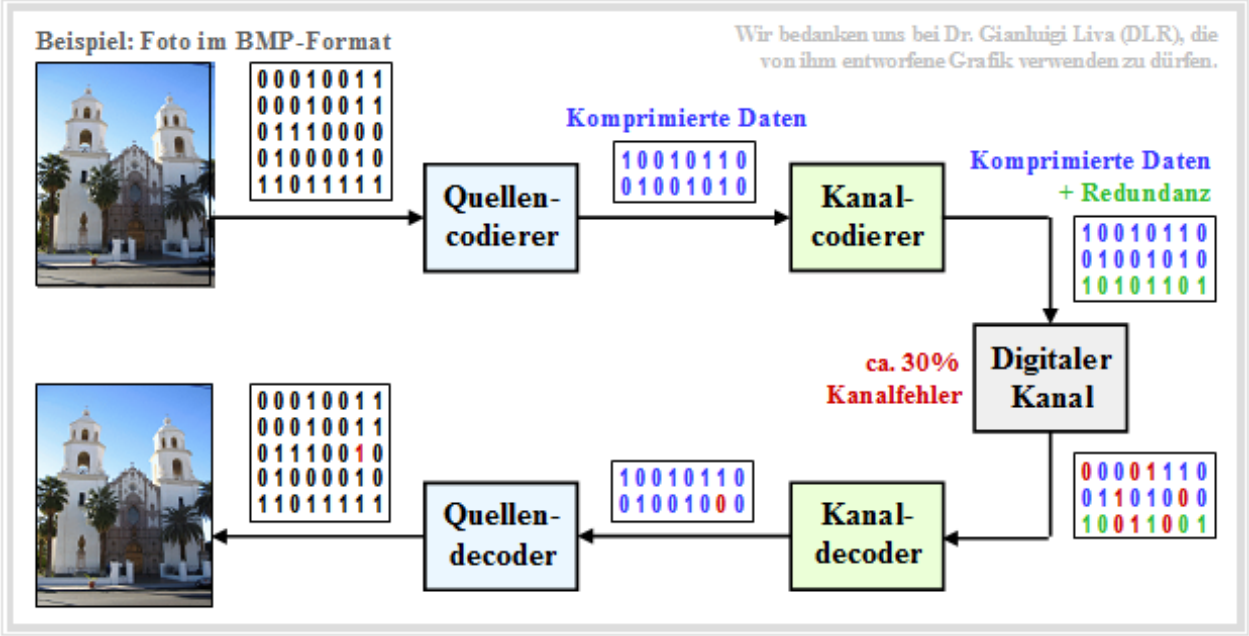
Resumee dieser Audiodemo:

- Die Fehlerkorrektur der CD basiert auf zwei seriell–verketteten Reed–Solomon–Codes sowie *Eight–to–Fourteen Modulation*. Die Gesamtcoderate zur RS–Fehlerkorrektur beträgt $R = 3/4$.
- Ebenso wichtig für die Funktionsfähigkeit der CD wie die Codes ist der dazwischen geschaltete Interleaver, der die ausgelöschten Bits („Erasures") über eine Länge von fast 2 cm verteilt.
- Bei der Spur 14 liegen die beiden defekten Bereiche genügend weit auseinander. Deshalb ist der Reed–Solomon–Decoder in der Lage, die fehlenden Daten zu rekonstruieren.
- Bei der Spur 3 folgen die beiden Fehlerblöcke in sehr kurzem Abstand aufeinander, so dass der Korrekturalgorithmus versagt. Das Resultat ist ein fast periodisches Klackgeräusch.

Wir bedanken uns bei Rainer Bauer, **Thomas Hindelang** und **Manfred Jürgens** herzlich dafür, diese Audio–Demo verwenden zu dürfen.

Zusammenspiel zwischen Quellen– und Kanalcodierung (1)

Die Nachrichtenübertragung natürlicher Quellen wie Sprache, Musik, Bilder, Videos, usw. geschieht meist entsprechend dem nachfolgend skizzierten zeitdiskreten Modell.



Zu dieser aus **[Liv10]** entnommenen Grafik ist Folgendes anzumerken:

- Quelle und Sinke sind digitalisiert und werden durch (gleich viele) Nullen und Einsen repräsentiert.
- Der Quellencodierer komprimiert die binären Daten – im betrachteten Beispiel ein Digitalfoto – und reduziert somit die Redundanz der Quelle.
- Der Kanalcodierer fügt wieder Redundanz hinzu und zwar gezielt, so dass die auf dem Kanal entstandenen Fehler im Kanaldecoder größtenteils korrigiert werden können.
- Für den Kanal wird hier ein zeitdiskretes Modell mit binärem Eingang und Ausgang verwendet, das auch die Komponenten der technischen Sende– und Empfangseinrichtungen (Modulator, Entscheider, Taktwiedergewinnung) geeignet berücksichtigen sollte.

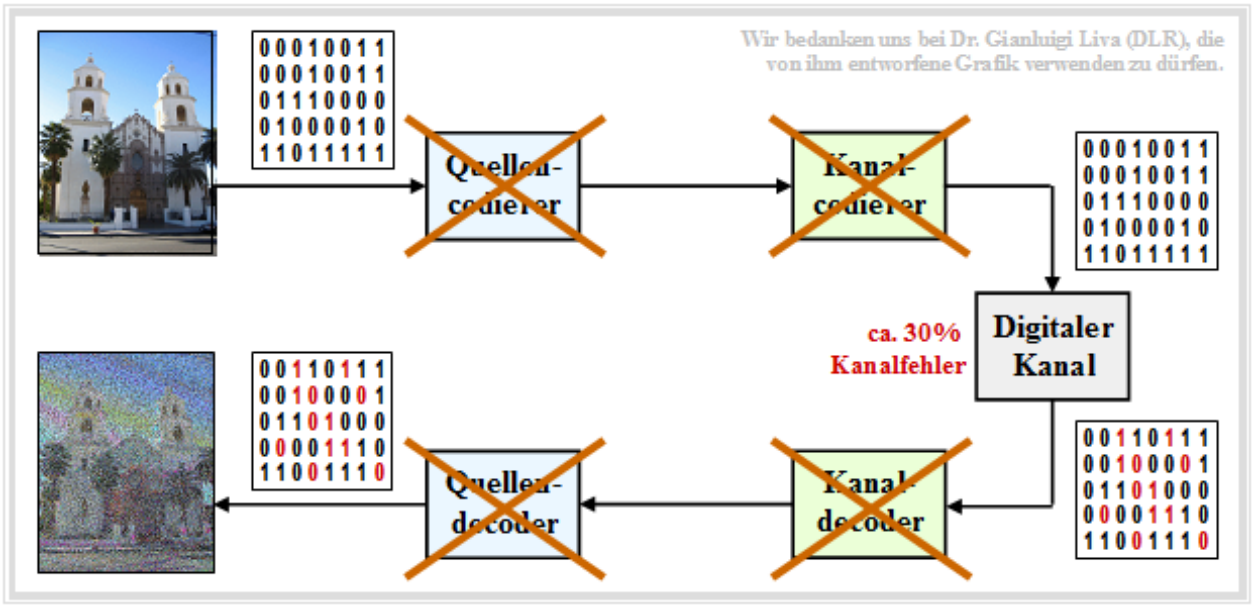
Bei richtiger Dimensionierung von Quellen– und Kanalcodierung ist die Qualität des empfangenen Fotos hinreichend gut, auch wenn die Sinkensymbolfolge aufgrund nichtkorrigierter Fehlermuster nicht exakt mit der Quellensymbolfolge übereinstimmen wird. Im Beispiel erkennt man einen Bitfehler.

Für obige Grafik wurde beispielhaft angenommen, dass der Quellencoder die Daten um den Faktor 40/16 = 2.5 komprimiert und der Kanalcoder 50% Redundanz hinzufügt. Übertragen werden müssen also nur 60% aller Quellensymbole, zum Beispiel 24 statt 40.

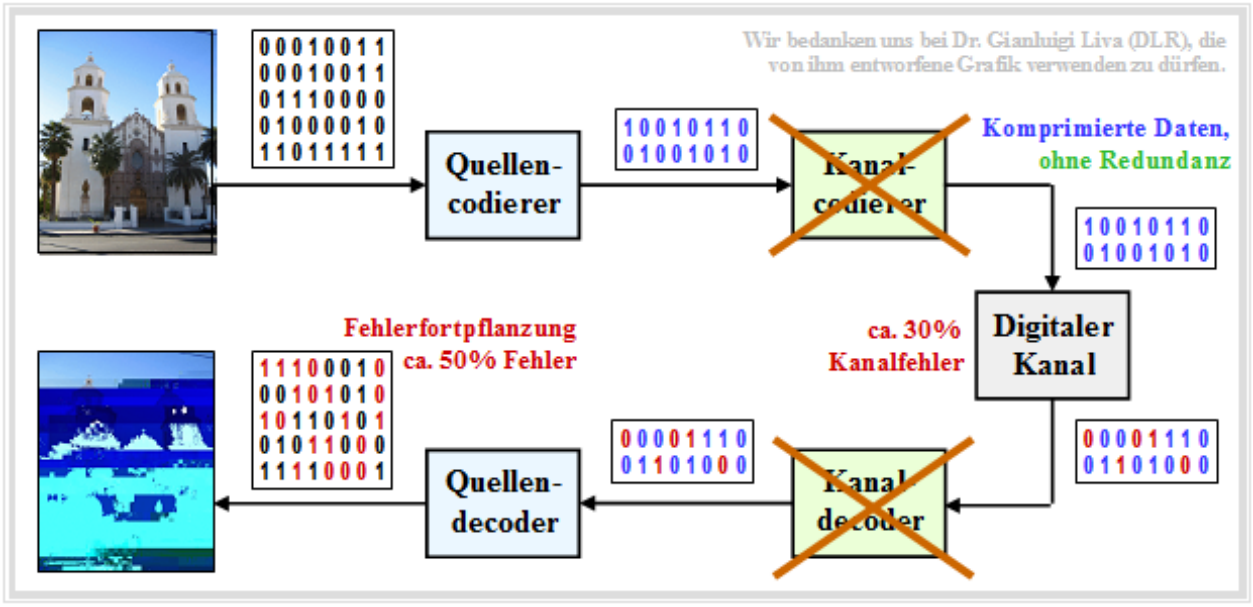
Würde man auf die **Quellencodierung** verzichten, in dem man das ursprüngliche Foto im BMP–Format übertragen würde und nicht das komprimierte JPG–Bild, so wäre die Qualität vergleichbar, aber eine um den Faktor 2.5 höhere Bitrate und damit sehr viel mehr Aufwand erforderlich.

Zusammenspiel zwischen Quellen– und Kanalcodierung (2)

Würde man sowohl auf die Quellen– als auch auf die Kanalcodierung verzichten, also direkt die BMP–Daten ohne Fehlerschutz übertragen, so wäre das Ergebnis trotz (um den Faktor 40/24) größerer Bitrate äußerst dürftig.



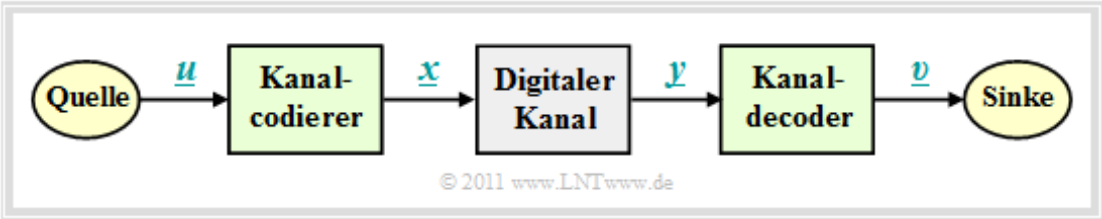
Die untere Grafik zeigt das Ergebnis für den Fall, dass man die komprimierten Daten (zum Beispiel JPG) ohne Fehlersicherungsmaßnahmen direkt überträgt. Dieses Codierschema (Quellencodierung, aber keine Kanalcodierung) sollte auf jeden Fall vermieden werden.



Da die komprimierte Quelle nur noch wenig Redundanz besitzt, führt jeder einzelne Übertragungsfehler dazu, dass ganze Bildblöcke falsch decodiert werden.

Blockschaltbild und Voraussetzungen

Im weiteren Verlauf gehen wir von folgendem Blockschaltbild aus:



Dabei gelten folgende Voraussetzungen:

- Der Vektor $\underline{u} = (u_1, u_2, \ldots, u_k)$ kennzeichnet einen **Informationsblock** mit k Symbolen. Meist beschränken wir uns auf Binärsymbole (Bits) $\Rightarrow u_i \in \{0, 1\}$ für $i = 1, 2, \ldots, k$ mit gleichen Auftrittswahrscheinlichkeiten für Nullen und Einsen.
- Jeder Informationsblock $\underline{u}$ wird durch ein **Codewort** (oder *Codeblock*) $\underline{x} = (x_1, x_2, \ldots, x_n)$ mit $n > k$, $x_i \in \{0, 1\}$ dargestellt. Man spricht dann von einem binären (n, k)–Blockcode C. Die Zuordnung bezeichnen wir mit $\underline{x} = \text{enc}(\underline{u})$, wobei „enc" für „Encoder–Funktion" steht.
- Das **Empfangswort** $\underline{y}$ ergibt sich aus dem Codewort $\underline{x}$ durch Modulo–2–Addition mit dem ebenfalls binären Fehlervektor $\underline{e} = (e_1, e_2, \ldots, e_n)$, wobei „$e_i = 1$" für einen Übertragungfehler steht und „$e_i = 0$" anzeigt, dass das i–te Bit des Codewortes richtig übertragen wurde. Es gilt also:

$$\underline{y} = \underline{x} \oplus \underline{e}, \quad y_i = x_i \oplus e_i, \quad i = 1, \ldots, n,$$
$$x_i \in \{0, 1\}, \quad e_i \in \{0, 1\} \quad \Rightarrow \quad y_i \in \{0, 1\}.$$

- Die Beschreibung durch das **digitale Kanalmodell** – also mit binärem Eingang und Ausgang – ist allerdings nur dann anwendbar, wenn das Übertragungssystem harte Entscheidungen trifft – siehe **Kapitel 1.2.** Systeme mit *Soft Decision* sind mit diesem einfachen Modell nicht modellierbar.
- Der Vektor $\underline{v}$ nach der **Kanaldecodierung** hat die gleiche Länge k wie der Informationsblock $\underline{u}$. Den Decodiervorgang beschreiben wir mit der „Decoder–Funktion" als $\underline{v} = \text{dec}(\underline{y})$. Im fehlerfreien Fall gilt analog zu $\underline{x} = \text{enc}(\underline{u})$ auch $\underline{v} = \text{enc}^{-1}(\underline{y})$.
- Ist der **Fehlervektor** $\underline{e} \neq 0$, so ist $\underline{y}$ meist kein gültiges Element des verwendeten Blockcodes, und der Decodiervorgang ist dann keine reine Zuordnung $\underline{y} \rightarrow \underline{v}$, sondern bezeichnet dann eine auf maximale Übereinstimmung (mimimale Fehlerwahrscheinlichkeit) basierende Schätzung von $\underline{v}$.

Einige wichtige Definitionen zur Blockcodierung

Wir betrachten nun den beispielhaften binären Blockcode

$$\mathcal{C} = \{(0,0,0,0,0)\,,(0,1,0,1,0)\,,(1,0,1,0,1)\,,(1,1,1,1,1)\}\,.$$

Dieser Code wäre zum Zwecke der Fehlererkennung oder –korrektur weniger gut geeignet. Aber er ist so konstruiert, dass er die Berechnung wichtiger Beschreibungsgrößen anschaulich verdeutlicht:

- Jedes einzelne Codewort $\underline{x}$ wird durch fünf Bit beschrieben. Im gesamten Buch drücken wir diesen Sachverhalt durch die **Codelänge** (englisch: *Code Length*) $n = 5$ aus.
- Der obige Code beinhaltet vier Elemente. Damit ist der **Codeumfang** (englisch: *Size*) $|C| = 4$. Entsprechend gibt es auch vier eindeutige Zuordnungen (englisch: *Mappings*) zwischen $\underline{u}$ und $\underline{x}$.
- Die **Informationsblocklänge** $\underline{u}$ wird mit k bezeichnet. Da bei allen binären Codes $|C| = 2^k$ gilt, folgt aus $|C| = 4$ der Wert $k = 2$. Die Zuordnungen zwischen $\underline{u}$ und $\underline{x}$ lauten bei obigem Code C:

$$\underline{u}_0 = (0,0) \leftrightarrow (0,0,0,0,0) = \underline{x}_0\,,\quad \underline{u}_1 = (0,1) \leftrightarrow (0,1,0,1,0) = \underline{x}_1\,,$$
$$\underline{u}_2 = (1,0) \leftrightarrow (1,0,1,0,1) = \underline{x}_2\,,\quad \underline{u}_3 = (1,1) \leftrightarrow (1,1,1,1,1) = \underline{x}_3\,.$$

- Der Code weist die **Coderate** $R = k/n = 2/5$ auf. Dementsprechend beträgt seine **Redundanz** $1 – R$, also 60%. Ohne Fehlerschutz – also für den Fall $n = k$ – wäre die Coderate $R = 1$.
- Eine kleine Coderate zeigt an, dass von den n Bits eines Codewortes nur sehr wenige tatsächlich Information tragen. Beispielsweise gilt für einen Wiederholungscode ($k = 1$) mit $n = 10$: $R = 0.1$.
- Das **Hamming–Gewicht** $w_{\rm H}(\underline{x})$ des Codewortes $\underline{x}$ gibt die Zahl der Codewortelemente $x_i \neq 0$ an. Bei einem binären Code $\Rightarrow x_i \in \{0, 1\}$ ist $w_{\rm H}(\underline{x})$ gleich der Summe $x_1 + x_2 + \ldots + x_n$:

$$w_{\rm H}(\underline{x}_0) = 0\,,\quad w_{\rm H}(\underline{x}_1) = 2\,,\quad w_{\rm H}(\underline{x}_2) = 3\,,\quad w_{\rm H}(\underline{x}_3) = 5\,.$$

- Die **Hamming–Distanz** $d_{\rm H}(\underline{x}, \underline{x}')$ zwischen den beiden Codeworten $\underline{x}$ und $\underline{x}'$ bezeichnet die Anzahl der Bitpositionen, in denen sich $\underline{x}$ und $\underline{x}'$ unterscheiden:

$$d_{\rm H}(\underline{x}_0, \underline{x}_1) = 2\,,\quad d_{\rm H}(\underline{x}_0, \underline{x}_2) = 3\,,\quad d_{\rm H}(\underline{x}_0, \underline{x}_3) = 5\,,$$
$$d_{\rm H}(\underline{x}_1, \underline{x}_2) = 5\,,\quad d_{\rm H}(\underline{x}_1, \underline{x}_3) = 3\,,\quad d_{\rm H}(\underline{x}_2, \underline{x}_3) = 2\,.$$

- Eine wichtige Eigenschaft eines Codes C, die seine Korrekturfähigkeit wesentlich beeinflusst, ist die **minimale Distanz** zwischen zwei beliebigen Codeworten:

$$d_{\rm min}(\mathcal{C}) = \min_{\substack{\underline{x},\,\underline{x}' \in \mathcal{C} \\ \underline{x} \neq \underline{x}'}} d_{\rm H}(\underline{x}, \underline{x}')\,.$$

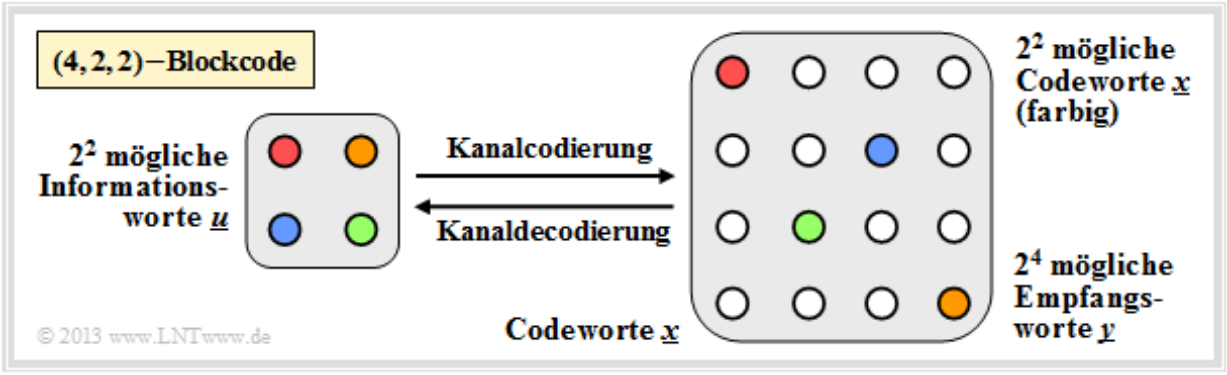
- Manchmal fügt man den Parameter $d_{\rm min}$ noch an die Kurzbezeichnung an und spricht dann von einem **(n, k, $d_{\rm min}$)–Blockcode**. Nach dieser Nomenklatur handelt es sich im hier betrachteten Beispiel um einen (5, 2, 2)–Blockcode.

Beispiele für Fehlererkennung und Fehlerkorrektur

Beispiel A: Wir betrachten zunächst einen **(4, 2, 2)–Blockcode** mit den Zuordnungen

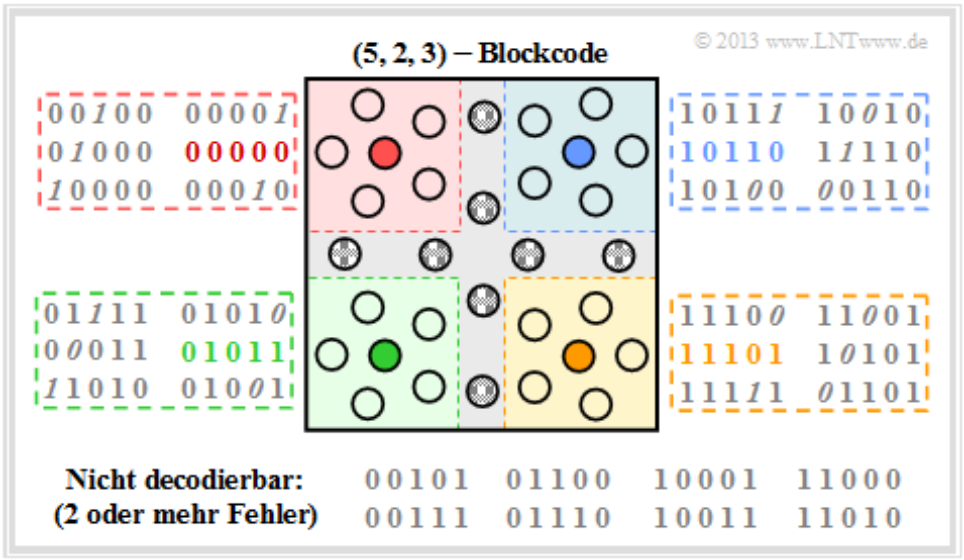
$$\underline{u}_0 = (0, 0) \leftrightarrow (0, 0, 0, 0) = \underline{x}_0, \quad \underline{u}_1 = (0, 1) \leftrightarrow (0, 1, 0, 1) = \underline{x}_1,$$
$$\underline{u}_2 = (1, 0) \leftrightarrow (1, 0, 1, 0) = \underline{x}_2, \quad \underline{u}_3 = (1, 1) \leftrightarrow (1, 1, 1, 1) = \underline{x}_3.$$

Die nach rechts bzw. links zeigenden Pfeile verdeutlichen den Codiervorgang bzw. die Decodierung.



Rechts sind alle $2^4 = 16$ möglichen Empfangsworte $\underline{y}$ dargestellt. Von diesen können $2^n - 2^k = 12$ nur durch Bitfehler entstanden sein. Empfängt der Decoder ein „weißes" Codewort, so erkennt er zwar einen Fehler, er kann diesen aber wegen $d_{\rm min} = 2$ nicht korrigieren. Empfängt er $\underline{y} = (0, 0, 0, 1)$, so kann mit gleicher Wahrscheinlichkeit das Codewort $\underline{x}_0 = (0, 0, 0, 0)$ oder $\underline{x}_1 = (0, 1, 0, 1)$ gesendet worden sein.

Im **Beispiel B** betrachten wir den **(5, 2, 3)–Blockcode** gemäß der zweiten Grafik mit den (gültigen) Codeworten (0, 0, 0, 0, 0), (0, 1, 0, 1, 1), (1, 0, 1, 1, 0) und (1, 1, 1, 0, 1). Dargestellt ist die Empfängerseite. Von den $2^n - 2^k = 28$ unzulässigen Codeworten lassen sich nun 20 einem gültigen Codewort (rot, grün, blau oder ocker) zuordnen, wenn man davon ausgeht, dass ein einziger Bitfehler wahrscheinlicher ist als deren zwei. In der Grafik erkennt man verfälschte Bit an der Kursivschrift.



Die Fehlerkorrektur ist aufgrund der minimalen Distanz $d_{\rm min} = 3$ zwischen den Codeworten möglich. Allerdings sind acht Empfangsworte nicht decodierbar. Beispielsweise könnte das Empfangswort $\underline{y} = (0, 0, 1, 0, 1)$ sowohl aus dem Codewort $\underline{x} = (0, 0, 0, 0, 0)$ durch zwei Bitfehler entstanden sein oder auch aus $\underline{x} = (1, 1, 1, 0, 1)$. Auch in diesem Fall wären zwei Bitfehler aufgetreten.

Zur Nomenklatur in diesem Buch

Eine Zielvorgabe unseres Lerntutorials *LNTwww* war, das gesamte Fachgebiet der Nachrichtentechnik und der zugehörigen Grundlagenfächer mit einheitlicher Nomenklatur zu beschreiben. In diesem zuletzt in Angriff genommenen LNTwww–Buch „Einführung in die Kanalcodierung" müssen nun doch einige Änderungen hinsichtlich der Nomenklatur vorgenommen werden. Die Gründe hierfür sind:

- Die Codierungstheorie ist ein weitgehend in sich abgeschlossenes Fachgebiet und nur wenige Autoren von einschlägigen Fachbüchern zu diesem Gebiet versuchen, einen Zusammenhang mit anderen Aspekten der Digitalsignalübertragung herzustellen.
- Die Autoren der wichtigsten Bücher zur Kanalcodierung – englischsprachige und auch deutsche – verwenden eine in weiten Bereichen einheitliche Nomenklatur. Deshalb erlauben wir uns nicht, die Bezeichnungen zur Kanalcodierung in unser Übertragungstechnik–Schema zu pressen.

Die wichtigsten Änderungen gegenüber den anderen LNTwww–Büchern sind:

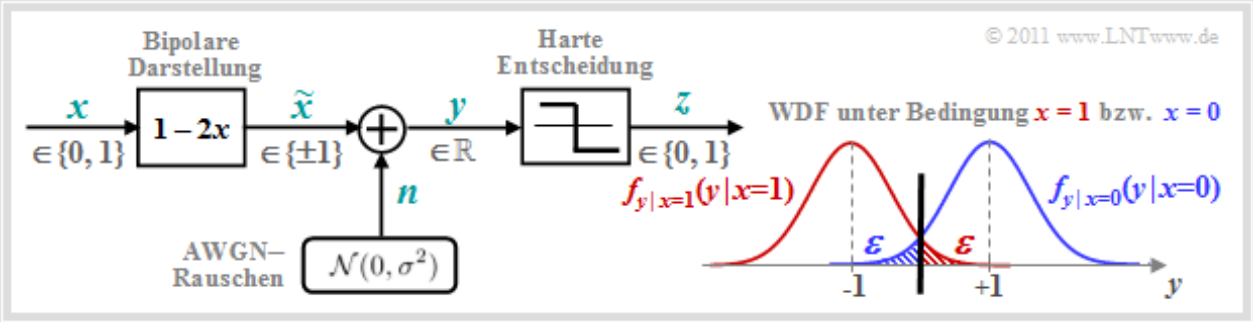
- Alle Signale werden durch die entsprechenden Symbolfolgen in Vektorschreibweise dargestellt. Beispielsweise kennzeichnet $\underline{u} = (u_1, u_2, \text{ ... }, u_k)$ die *Quellensymbolfolge* und $\underline{v} = (v_1, v_2, \text{ ... }, v_k)$ die *Sinkensymbolfolge*. Bisher wurden diese Symbolfolgen mit $\langle q_\nu \rangle$ bzw. $\langle v_\nu \rangle$ bezeichnet.
- Das zeitdiskrete Äquivalent zum *Sendesignal* $s(t)$ bzw. zum *Empfangssignal* $r(t)$ in anderen Büchern sind die Vektoren $\underline{x} = (x_1, x_2, \text{ ... }, x_n)$ und $\underline{y} = (y_1, y_2, \text{ ... }, y_n)$. Die *Coderate* ergibt sich zu $R = k/n$ (Werte zwischen 0 und 1 $\Rightarrow$ $n \ge k$), und $m = n - k$ gibt die Anzahl der Prüfbits an.
- Im Kapitel 1 sind die Elemente u_i und v_i (jeweils $i = 1, \text{ ... }, k$) der Vektoren $\underline{u}$ und $\underline{v}$ stets binär (0 oder 1), ebenso wie die n Elemente x_i des Codewortes $\underline{x}$. Bei digitalem Kanalmodell (BSC, BEC, BSEC $\Rightarrow$ Kapitel 1.2) gilt auch für die n Empfangswerte $y_i \in \{0, 1\}$.
- Das AWGN–Kanalmodell (erste Seite von Kapitel 1.2) ist durch reellwertige Ausgangswerte y_i gekennzeichnet. Der *Codewortschätzer* gewinnt daraus den binären Vektor $\underline{z} = (z_1, z_2, \text{ ... }, z_n)$, der mit dem Codewort $\underline{x}$ zu vergleichen ist.
- Der Übergang von $\underline{y}$ auf $\underline{z}$ erfolgt entweder durch Schwellenwertentscheidung $\Rightarrow$ *Hard Decision* oder nach dem MAP–Kriterium $\Rightarrow$ *Soft Decision*. Die Schätzung gemäß „Maximum Likelihood" führt bei gleichwahrscheinlichen Eingangssymbolen ebenfalls zur minimalen Fehlerrate.
- Im Zusammenhang mit dem AWGN–Modell macht es Sinn, die binären Codesymbole x_i bipolar (also als +1 oder –1) darzustellen. An den statistischen Eigenschaften ändert sich dadurch nichts. Wir kennzeichnen im Folgenden die bipolare Signalisierung durch eine Tilde. Dann gilt:

$$\tilde{x}_i = 1 - 2x_i = \begin{cases} +1 & \text{falls } x_i = 0\,, \\ -1 & \text{falls } x_i = 1\,. \end{cases}$$

AWGN–Kanal bei binärem Eingang

Wir betrachten das bekannte zeitdiskrete **AWGN–Kanalmodell** gemäß der unteren Grafik (links):

- Das binäre und zeitdiskrete Nachrichtensignal x nimmt mit gleicher Wahrscheinlichkeit die Werte 0 und 1 an, das heißt, es ist $\Pr(x = 0) = \Pr(\tilde{x} = +1) = 1/2$ sowie $\Pr(x = 1) = \Pr(\tilde{x} = -1) = 1/2$.
- Die Übertragung wird durch **additives weißes gaußverteiltes Rauschen** (AWGN) n mit der (normierten) Rauschleistung $\sigma^2 = N_0/(2E_S)$ beeinträchtigt. Die Streuung der Gauß–WDF ist σ.
- Aufgrund der Gaußschen WDF kann das Ausgangssignal $y = \tilde{x} + n$ alle reellen Werte annehmen. Der Signalwert y ist zwar wie $\tilde{x}$ zeitdiskret, im Gegensatz zu diesem aber wertkontinuierlich.



Die rechte Grafik zeigt die bedingten Wahrscheinlichkeitsdichtefunktionen (in blau bzw. rot):

$$f_{y|x=0}(y\,|\,x=0) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left[-\frac{(y-1)^2}{2\sigma^2}\right],$$

$$f_{y|x=1}(y\,|\,x=1) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left[-\frac{(y+1)^2}{2\sigma^2}\right].$$

Nicht dargestellt ist die gesamte (unbedingte) WDF, für die bei gleichwahrscheinlichen Symbolen gilt:

$$f_y(y) = 1/2\cdot\left[f_{y|x=0}(y\,|\,x=0) + f_{y|x=1}(y\,|\,x=1)\right].$$

Die beiden schraffierten Flächen (jeweils ε) markieren Entscheidungsfehler unter der Voraussetzung $x = 0 \;\Rightarrow\; \tilde{x} = +1$ (blau) bzw. $x = 1 \;\Rightarrow\; \tilde{x} = -1$ (rot), wenn harte Entscheidungen getroffen werden:

$$z = \begin{cases} 0 & \text{falls } y > 0, \\ 1 & \text{falls } y < 0. \end{cases}$$

Bei gleichwahrscheinlichen Eingangssymbolen ist dann die mittlere Bitfehlerwahrscheinlichkeit $\Pr(z \neq x)$ ebenfalls gleich ε. Mit dem **komplementären Gaußschen Fehlerintergral** $\mathrm{Q}(x)$ gilt dabei:

$$\varepsilon = \mathrm{Q}(1/\sigma) = \mathrm{Q}(\sqrt{\rho}) = \frac{1}{\sqrt{2\pi}}\cdot\int_{\sqrt{\rho}}^{\infty} \mathrm{e}^{-\alpha^2/2}\,\mathrm{d}\alpha.$$

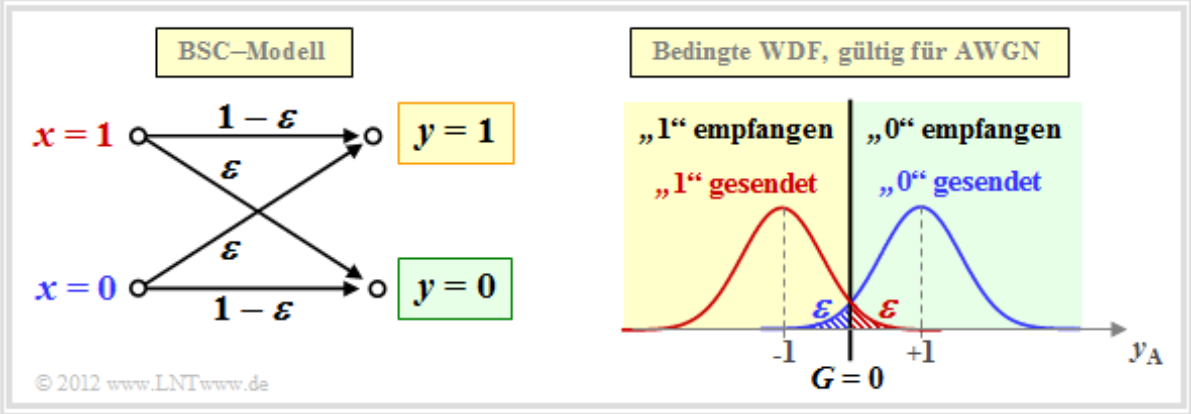
Hierbei bezeichnet $\rho = 1/\sigma^2 = 2\cdot E_S/N_0$ das Signal–zu–Rauschverhältnis (SNR) vor dem Entscheider, wobei folgende Systemgrößen verwendet werden:

- E_S ist die Signalenergie pro Symbol (ohne Codierung gleich E_B, also der Signalenergie pro Bit),
- N_0 bezeichnet die konstante (einseitige) Rauschleistungsdichte des AWGN–Kanals.

Wir verweisen hier auf das interaktive Flash–Modul **Fehlerwahrscheinlichkeit von Digitalsystemen.**

Binary Symmetric Channel – BSC

Das AWGN–Kanalmodell ist kein digitales Kanalmodell, wie wir es im **Kapitel 1.1** zur Beschreibung der Kanalcodierverfahren vorausgesetzt haben. Berücksichtigen wir aber eine harte Entscheidung, so kommen wir zum digitalen Modell *Binary Symmetric Channel* (BSC):



Wählt man die Verfälschungswahrscheinlichkeiten Pr($y = 1 \mid x = 0$) bzw. Pr($y = 0 \mid x = 1$) jeweils zu

$$\varepsilon = \mathrm{Q}(\sqrt{\rho}),$$

so ist der Zusammenhang zum **AWGN–Kanalmodell** hergestellt. Die Entscheidungsgrenze liegt dabei bei $G = 0$, wodurch auch die Eigenschaft „symmetrisch” begründet ist.

Hinweis: Beim AWGN–Modell haben wir die binäre Ausgangsgröße (nach Schwellenwertentscheidung) mit $z \in \{0, 1\}$ bezeichnet. Bei den digitalen Kanalmodellen (BSC, BEC, BSEC) bezeichnen wir nun den wertdiskreten Ausgang wieder mit y. Um Verwechslungen zu vermeiden, nennen wir das Ausgangssignal des AWGN –Modells nun y_A. Für das analoge Empfangssignal gilt $y_\mathrm{A} = \tilde{x} + n$.

Das BSC–Modell liefert eine statistisch unabhängige Fehlerfolge und eignet sich somit zur Modellierung gedächtnisloser rückkopplungsfreier Kanäle, die in diesem Buch ausnahmslos betrachtet werden.

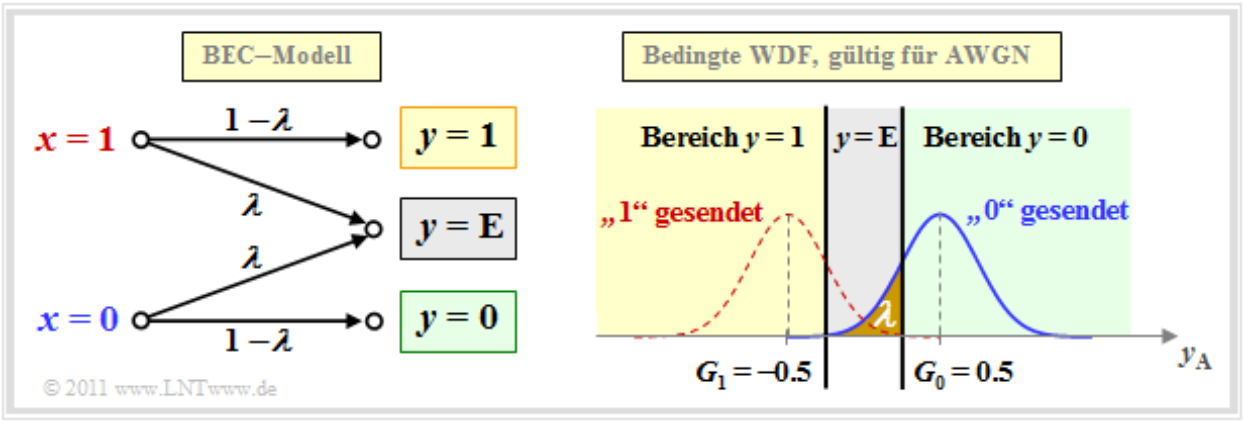
Zur Beschreibung gedächtnisbehafteter Kanäle müssen andere Modelle herangezogen werden, die im **Kapitel 5** des Buches „Digitalsignalübertragung” behandelt werden, zum Beispiel Bündelfehler nach

- dem **Gilbert–Elliott–Modell,**
- dem **McCullough–Kanalmodell.**

Die Abbildung zeigt statistisch unabhängige Fehler nach dem BSC–Modell (links) und so genannte Bündelfehler gemäß Gilbert–Elliott (rechts), wobei die Bitfehlerrate in beiden Fällen 10% beträgt. Aus der rechten Grafik ist zu erkennen, dass das Bild zeilenweise übertragen wird.

Binary Erasure Channel – BEC

Das BSC–Modell liefert nur die Aussagen „richtig” und „falsch”. Manche Empfänger – so zum Beispiel die so genannten **Soft–in Soft–out Decoder** – können jedoch auch gewisse Informationen über die Sicherheit der Entscheidung liefern, wobei sie natürlich darüber informiert werden müssen, welche ihrer Eingangswerte sicher sind und welche eher unsicher.



Der *Binary Erasure Channel* (BEC) liefert eine solche Information. Anhand der Grafik erkennt man:

- Das Eingangsalphabet des BEC–Kanalmodells ist binär $\Rightarrow x \in \{0, 1\}$ und das Ausgangsalphabet ternär $\Rightarrow y \in \{0, 1, \text{E}\}$. Ein „E” kennzeichnet eine unsichere Entscheidung. Dieses neue „Symbol” steht für *Erasure*, zu deutsch: Auslöschung.
- Bitfehler werden durch das BEC–Modell per se ausgeschlossen. Eine unsichere Entscheidung (E) wird mit der Wahrscheinlichkeit λ getroffen, während die Wahrscheinlichkeit für eine richtige (und gleichzeitig sichere) Entscheidung $1 – \lambda$ beträgt.
- Rechts oben ist der Zusammenhang zwischen BEC– und AWGN–Kanalmodell dargestellt, wobei das Erasure–Entscheidungsgebiet („E”) grau hinterlegt ist. Man erkennt, dass es im Gegensatz zum BSC–Modell ($G = 0$) nun zwei Entscheidungsgrenzen $G_0 = G$ und $G_1 = –G$ gibt, und es gilt:

$$\lambda = \text{Q}[\sqrt{\rho} \cdot (1 - G)] .$$

Wir weisen hier nochmals auf zwei interaktive Flash–Module hin:

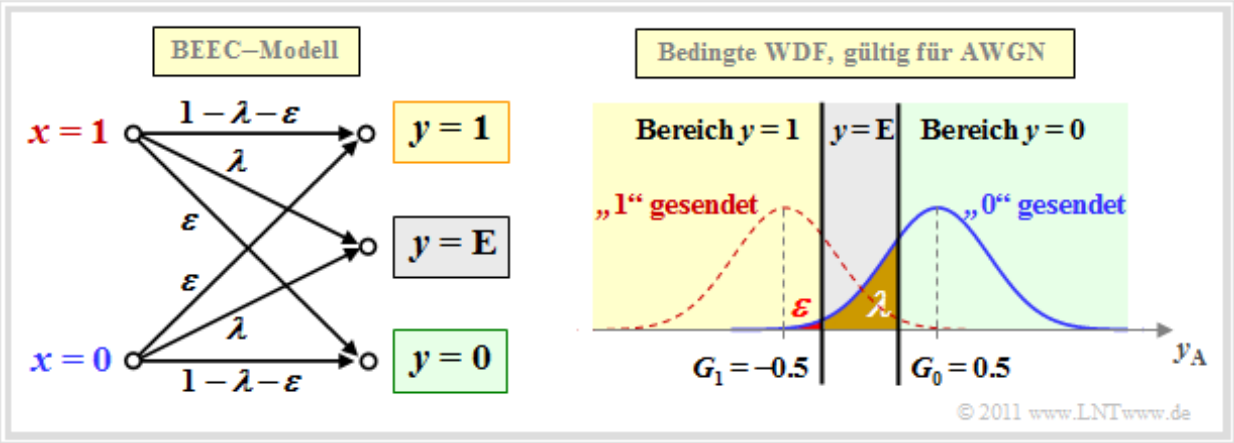
- **Fehlerwahrscheinlichkeit von Digitalsystemen**
- **Komplementäre Gaußsche Fehlerfunktion**

Binary Symmetric Error & Erasure Channel – BSEC

Das BEC–Modell ist aufgrund der Fehlerwahrscheinlichkeit 0 eher unrealistisch und nur eine Näherung für ein extrem großes Signal–zu–Rausch–Leistungsverhältnis (kurz SNR) ρ. Stärkere Störungen ⇒ ein kleineres ρ sollten besser durch den *Binary Symmetric Error & Erasure Channel* (BSEC) mit den zwei Parametern

- Verfälschungswahrscheinlichkeit $\varepsilon = \Pr(y = 1 \mid x = 0) = \Pr(y = 0 \mid x = 1)$,
- Erasure–Wahrscheinlichkeit $\lambda = \Pr(y = \mathrm{E} \mid x = 0) = \Pr(y = \mathrm{E} \mid x = 1)$

modelliert werden. Es gilt auch hier $x \in \{0, 1\}$ und $y \in \{0, 1, \mathrm{E}\}$.



Beispiel: Wir betrachten das BSEC–Modell mit den beiden Entscheidungsgeraden $G_0 = G = 0.5$ und $G_1 = -G$, dessen Parameter ε und λ durch das SNR $\rho = 1/\sigma^2$ des vergleichbaren AWGN–Kanals festgelegt sind. Dann gilt

- für $\sigma = 0.5 \;\Rightarrow\; \rho = 4$:

$$\varepsilon = \mathrm{Q}[\sqrt{\rho} \cdot (1 + G)] = \mathrm{Q}(3) \approx 0.14\%,$$
$$\lambda = \mathrm{Q}[\sqrt{\rho} \cdot (1 - G)] - \varepsilon = \mathrm{Q}(1) - \mathrm{Q}(3) \approx 15.87\% - 0.14\% = 15.73\%,$$

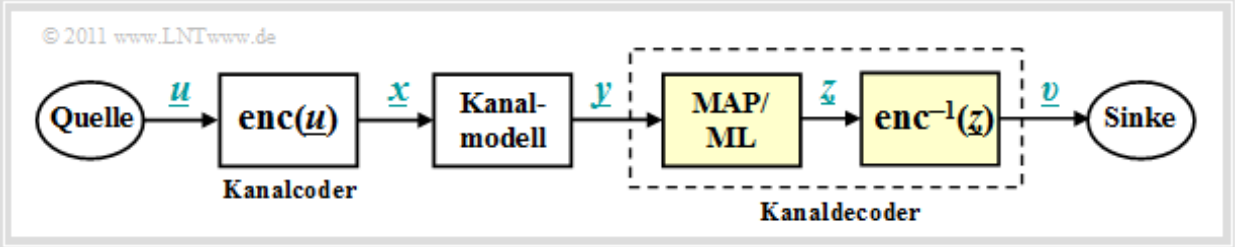
- für $\sigma = 0.25 \;\Rightarrow\; \rho = 16$:

$$\varepsilon = \mathrm{Q}(6) \approx 10^{-10}, \quad \lambda = \mathrm{Q}(2) \approx 2.27\%.$$

Für die rechts dargestellte WDF wurde $\rho = 4$ vorausgesetzt. Für $\rho = 16$ könnte das BSEC–Modell durch die BEC–Variante ersetzt werden, ohne dass es zu einer gravierenden Verfälschung kommt.

MAP– und ML–Kriterium (1)

Wir gehen nun von dem nachfolgend skizzierten Modell aus und wenden die bereits im **Kapitel 4.2** des Buches „Digitalsignalübertragung" genannten Entscheidungskriterien auf den Decodiervorgang an.



Aufgabe des Kanaldecoders ist es, den Ausgabevektor $\underline{v}$ so zu bestimmen, dass er „möglichst gut" mit dem Informationswort $\underline{u}$ übereinstimmt. Etwas genauer formuliert: Die **Blockfehlerwahrscheinlichkeit**

$$\Pr(\text{Blockfehler}) = \Pr(\underline{v} \neq \underline{u})$$

bezogen auf die Vektoren $\underline{u}$ und $\underline{v}$ der Länge k soll möglichst gering sein. Aufgrund der eindeutigen Zuordnung $\underline{x} = \text{enc}(\underline{u})$ durch den Kanalcoder bzw. empfängerseitig $\underline{v} = \text{enc}^{-1}(\underline{z})$ gilt in gleicher Weise:

$$\Pr(\text{Blockfehler}) = \Pr(\underline{z} \neq \underline{x}).$$

Der **Kanaldecoder** in obigem Modell besteht aus zwei Teilen:

- Der *Codewortschätzer* ermittelt aus dem Empfangsvektor $\underline{y}$ einen Schätzwert $\underline{z} \in C$ gemäß einem vorgegebenen Kriterium.
- Anschließend wird aus dem (empfangenen) Codewort $\underline{z}$ das (empfangene) Informationswort $\underline{v}$ durch *einfaches Mapping* ermittelt, das möglichst mit $\underline{u}$ übereinstimmen sollte.

Für den Codewortschätzer gibt es insgesamt vier unterschiedliche Varianten, nämlich

- den Maximum–a–posteriori–Empfänger (MAP–Empfänger) für das gesamte Codewort $\underline{x}$,
- den Maximum–a–posteriori–Empfänger (MAP–Empfänger) für die einzelnen Codebits x_i,
- den Maximum–Likelihood–Empfänger (ML–Empfänger) für das gesamte Codewort $\underline{x}$,
- den Maximum–Likelihood–Empfänger (ML–Empfänger) für die einzelnen Codebits x_i.

Deren Definitionen folgen auf der nächsten Seite. Vorab aber gleich die wichtige Information:

- Ein MAP–Empfängerberücksichtigt im Gegensatz zum ML–Empfänger auch unterschiedliche Auftrittswahrscheinlichkeiten für das gesamte Codewort bzw. für deren einzelne Bits.
- Sind alle Codeworte bzw. alle Codebits x_i der Codeworte gleichwahrscheinlich, so ist der einfachere ML–Empfänger zum entsprechenden MAP–Empfänger äquivalent.

MAP– und ML–Kriterium (2)

Definition: Der **Maximum–a–posteriori–Empfänger auf Blockebene** – kurz: **block–wise MAP** – entscheidet sich unter den 2^k zulässigen Codeworten $\underline{x}_i \in C$ für das Codewort mit der größten Rückschlusswahrscheinlichkeit (englisch: *a–posteriori probability*, APP):

$$\underline{z} = \arg \max_{\underline{x}_i \in C} \Pr(\underline{x}_i | \underline{y}) .$$

$\Pr(\underline{x}_i \,|\, \underline{y})$ ist die **bedingte Wahrscheinlichkeit,** dass $\underline{x}_i$ gesendet wurde, wenn $\underline{y}$ empfangen wird.

Nach dem **Satz von Bayes** kann die Rückschlusswahrscheinlichkeit wie folgt umgeformt werden:

$$\Pr(\underline{x}_i | \underline{y}) = \frac{\Pr(\underline{y} \,|\, \underline{x}_i) \cdot \Pr(\underline{x}_i)}{\Pr(\underline{y})} .$$

Die Wahrscheinlichkeit $\Pr(\underline{y})$ ist unabhängig von $\underline{x}_i$ und muss bei der Maximierung nicht berücksichtigt werden. Sind zudem alle 2^k Informationsworte $\underline{u}_i$ gleichwahrscheinlich, dann kann bei der Maximierung auch auf den Beitrag $\Pr(\underline{x}_i) = 2^{-k}$ im Zähler verzichtet werden.

Definition: Der **Maximum–Likelihood–Empfänger auf Blockebene** – kurz: **block–wise ML** – entscheidet sich unter den 2^k zulässigen Codeworten $\underline{x}_i \in C$ für das Codewort mit der größten Übergangswahrscheinlichkeit:

$$\underline{z} = \arg \max_{\underline{x}_i \in C} \Pr(\underline{y} \,|\, \underline{x}_i) .$$

Die bedingte Wahrscheinlichkeit $\Pr(\underline{y} \,|\, \underline{x}_i)$ ist nun in Vorwärtsrichtung zu verstehen, nämlich, dass der Vektor $\underline{y}$ empfangen wird, wenn das Codewort $\underline{x}_i$ gesendet wurde.

Im Folgenden verwenden wir auf Blockebene stets den ML–Empfänger. Aufgrund der vorausgesetzten gleichwahrscheinlichen Informationsworte liefert auch dieser stets die bestmögliche Entscheidung.

Anders sieht es jedoch auf Bitebene aus. Ziel einer iterativen Decodierung ist es gerade, für alle Codebits x_i ∈ {0, 1} Wahrscheinlichkeiten zu schätzen und diese an die nächste Stufe weiterzugeben. Hierzu benötigt man einen MAP–Empfänger.

Definition: Der **Maximum–a–posteriori–Empfänger auf Bitebene** (kurz: **bit–wise MAP**) wählt für jedes Codebit x_i den Wert (0 oder 1) mit der größten Rückschlusswahrscheinlichkeit $\Pr(x_i \,|\, \underline{y})$:

$$\underline{z} = \arg \max_{x_i \in \{0,1\}} \Pr(x_i | \underline{y}) .$$

ML–Entscheidung beim BSC–Kanal

Wenden wir nun das ML–Kriterium auf den gedächtnislosen **BSC–Kanal** an. Dann gilt:

$$\Pr(\underline{y}\,|\,\underline{x}_i) = \prod_{l=1}^{n} \Pr(y_l\,|\,x_l) \;\;\text{mit}\;\; \Pr(y_l\,|\,x_l) = \begin{cases} 1-\varepsilon & \text{falls } y_l = x_l\,, \\ \varepsilon & \text{falls } y_l \neq x_l\,. \end{cases}$$

$$\Rightarrow \quad \Pr(\underline{y}\,|\,\underline{x}_i) = \varepsilon^{d_{\rm H}(\underline{y},\,\underline{x}_i)} \cdot (1-\varepsilon)^{n-d_{\rm H}(\underline{y},\,\underline{x}_i)}\,.$$

Dies lässt sich wie folgt begründen:

- Die **Hamming–Distanz** $d_{\rm H}(\underline{y}, \underline{x}_i)$ gibt die Anzahl der Bitpositionen an, in denen sich die beiden Worte $\underline{y}$ und $\underline{x}_i$ mit jeweils n binären Elementen unterscheiden. Beispielsweise ist die Hamming–Distanz zwischen $\underline{y} = (0, 1, 0, 1, 0, 1, 1)$ und $\underline{x}_i = (0, 1, 0, 0, 1, 1, 1)$ gleich 2.
- In $n - d_{\rm H}(\underline{y}, \underline{x}_i)$ Positionen unterscheiden sich demnach die beiden Vektoren $\underline{y}$ und $\underline{x}_i$ nicht. Im obigen Beispiel sind 5 der $n = 7$ Bit identisch. Zu obiger Gleichung kommt man schließlich durch Einsetzen der Verfälschungswahrscheinlichkeit ε bzw. deren Ergänzung $1 - \varepsilon$.

Die Vorgehensweise bei der Maximum–Likelihood–Detektion ist, dasjenige Codewort $\underline{x}_i$ zu finden, das die Übergangswahrscheinlichkeit $\Pr(\underline{y} \,|\, \underline{x}_i)$ maximiert:

$$\underline{z} = \arg\max_{\underline{x}_i \in \mathcal{C}} \left[\varepsilon^{d_{\rm H}(\underline{y},\,\underline{x}_i)} \cdot (1-\varepsilon)^{n-d_{\rm H}(\underline{y},\,\underline{x}_i)}\right].$$

Da der Logarithmus eine monoton steigende Funktion ist, erhält man das gleiche Ergebnis nach folgender Maximierung:

$$\underline{z} = \arg\max_{\underline{x}_i \in \mathcal{C}} L(\underline{x}_i)$$

$$\begin{aligned} \text{mit } L(\underline{x}_i) &= \ln\left[\varepsilon^{d_{\rm H}(\underline{y},\,\underline{x}_i)} \cdot (1-\varepsilon)^{n-d_{\rm H}(\underline{y},\,\underline{x}_i)}\right] = \\ &= d_{\rm H}(\underline{y},\,\underline{x}_i) \cdot \ln\varepsilon + [n - d_{\rm H}(\underline{y},\,\underline{x}_i)] \cdot \ln(1-\varepsilon) = \\ &= \ln\frac{\varepsilon}{1-\varepsilon} \cdot d_{\rm H}(\underline{y},\,\underline{x}_i) + n \cdot \ln(1-\varepsilon)\,. \end{aligned}$$

Der zweite Term dieser Gleichung ist unabhängig von $\underline{x}_i$ und muss für die Maximierung nicht weiter betrachtet werden. Auch der Faktor vor der Hamming–Distanz ist für alle $\underline{x}_i$ gleich. Da aber $\ln \varepsilon/(1 - \varepsilon)$ negativ ist (zumindest für $\varepsilon < 0.5$, was ohne große Einschränkung vorausgestzt werden kann), wird aus der Maximierung eine Minimierung, und man erhält folgendes Endergebnis:

ML–Entscheidung beim BSC–Kanal: Wähle von den 2^k zulässigen Codeworten dasjenige mit der *geringsten Hamming–Distanz* $d_{\rm H}(\underline{y}, \underline{x}_i)$ zum Empfangsvektor $\underline{y}$ aus:

$$\underline{z} = \arg\min_{\underline{x}_i \in \mathcal{C}} d_{\rm H}(\underline{y},\,\underline{x}_i)\,,\;\; \underline{y} \in {\rm GF}(2^n)\,,\;\; \underline{x}_i \in {\rm GF}(2^n)\,.$$

Anwendungen der ML/BSC–Entscheidung finden Sie auf den folgenden Seiten:

- **Single Parity–check Code** (SPC)
- **Wiederholungscode** (englisch: *Repetition Code*, RC).

ML–Entscheidung beim AWGN–Kanal

Das AWGN–Modell für einen (n, k)–Blockcode unterscheidet sich vom **Modell** auf der ersten Seite dieses Kapitels dadurch, dass für x, $\tilde{x}$ und y nun die entsprechenden Vektoren $\underline{x}$, $\underline{\tilde{x}}$ und $\underline{y}$ verwendet werden müssen, jeweils bestehend aus n Elementen. Die Schritte zur Herleitung des ML–Entscheiders bei AWGN werden nachfolgend nur stichpunktartig angegeben:

- Der AWGN–Kanal ist per se **gedächtnislos** (hierfür steht das *White* im Namen), so dass für die bedingte Wahrscheinlichkeitsdichtefunktion geschrieben werden kann:

$$f(\underline{y}\,|\,\underline{\tilde{x}}) = \prod_{l=1}^{n} f(y_l\,|\,\tilde{x}_l)\,.$$

- Die bedingte WDF ist für jedes einzelne Codeelement ($l = 1, \ldots, n$) **gaußisch**. Damit genügt auch die gesamte WDF einer (eindimensionalen) Gaußverteilung:

$$f(y_l\,|\,\tilde{x}_l) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left[-\frac{(y_l-\tilde{x}_l)^2}{2\sigma^2}\right]$$

$$\Rightarrow\quad f(\underline{y}\,|\,\underline{\tilde{x}}) = \frac{1}{(2\pi)^{n/2}\cdot\sigma^n}\cdot\exp\left[-\frac{1}{2\sigma^2}\cdot\sum_{l=1}^{n}(y_l-\tilde{x}_l)^2\right].$$

- Da $\underline{y}$ nun nicht mehr wie beim BSC–Modell wertdiskret ist, sondern **wertkontinuierlich**, müssen jetzt entsprechend der ML–Entscheidungsregel *Wahrscheinlichkeitsdichten* untersucht werden und nicht mehr Wahrscheinlichkeiten. Das optimale Ergebnis lautet:

$$\underline{z} = \arg\max_{\underline{x}_i\,\in\,\mathcal{C}} f(\underline{y}\,|\,\underline{\tilde{x}}_i), \qquad \underline{y}\in R^n,\ \ \underline{x}_i\in \mathrm{GF}(2^n)\,.$$

- In der Algebra bezeichnet man den Abstand zweier Punkte $\underline{y}$ und $\tilde{x}$ im n–dimensionalen Raum als die **Euklidische Distanz**, benannt nach dem griechischen Mathematiker **Euklid,** der im dritten Jahrhundert vor Christus lebte:

$$d_{\rm E}(\underline{y},\,\underline{\tilde{x}}) = \sqrt{\sum_{l=1}^{n}(y_l-\tilde{x}_l)^2}\,, \qquad \underline{y}\in R^n,\ \ \underline{x}_i\in\mathcal{C}\,.$$

- Damit lautet die ML–Entscheidungsregel beim AWGN–Kanal für einen jeden Blockcode unter Berücksichtigung der Tatsache, dass der erste Faktor der WDF $f(\underline{y}\,|\,\underline{\tilde{x}}_i)$ konstant ist:

$$\underline{z} = \arg\max_{\underline{x}_i\,\in\,\mathcal{C}} \exp\left[-\frac{d_{\rm E}(\underline{y},\,\underline{\tilde{x}}_i)}{2\sigma^2}\right], \qquad \underline{y}\in R^n,\ \ \underline{x}_i\in \mathrm{GF}(2^n)\,.$$

Nach einigen weiteren Zwischenschritten kommt man zum Ergebnis:

ML–Entscheidung beim AWGN–Kanal: Wähle von den 2^k zulässigen Codeworten x_i dasjenige mit der *kleinsten Euklidischen Distanz* zum Empfangsvektor $\underline{y}$ aus:

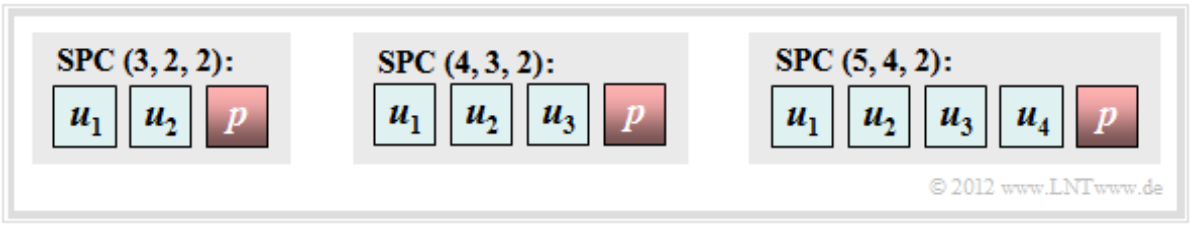
$$\underline{z} = \arg\min_{\underline{x}_i\,\in\,\mathcal{C}} d_{\rm E}(\underline{y},\,\underline{x}_i), \qquad \underline{y}\in R^n,\ \ \underline{x}_i\in \mathrm{GF}(2^n)\,.$$

Single Parity–check Codes (1)

Der *Single Parity–check Code* (SPC) fügt zu dem Informationsblock $\underline{u} = (u_1, u_2, \ \text{...} \ , u_k)$ ein Prüfbit (englisch: *Parity*) p hinzu:

$$\underline{u} = (u_1, u_2, \ \text{...} \ , u_k) \quad \Rightarrow \quad \underline{x} = (x_1, x_2, \ \text{...} \ , x_n) = (u_1, u_2, \ \text{...} \ , u_k, p) \, .$$

Die Grafik zeigt drei Beispiele solcher Codes mit $|C| = 4$ ($k = 2$), $|C| = 8$ ($k = 3$) und $|C| = 16$ ($k = 4$).



Dieser sehr einfache Code ist wie folgt charakterisiert:

- Aus $n = k + 1$ folgt für die **Coderate $R = k/n$** $= (n - 1)/n$ und für die Redundanz $1 - R = 1/n$. Für $k = 2$ ergibt sich zum Beispiel die Coderate 2/3 und die relative Redundanz beträgt 33.3%.
- Das Prüfbit erhält man durch die **Modulo–2–Addition**. Darunter versteht man die Addition im Galoisfeld zur Basis 2 $\Rightarrow$ GF(2), sodass $1 \oplus 1 = 0$ ergibt:

  $$p = u_1 \oplus u_2 \oplus \ \text{...} \ \oplus u_k \, .$$

- Damit enthält jedes gültige Codewort $\underline{x}$ eine gerade Anzahl von Einsen. Ausgedrückt mit $\oplus$ bzw. in vereinfachter Schreibweise entsprechend der zweiten Gleichung lautet diese Bedingung:

  $$x_1 \oplus x_2 \oplus \ \text{...} \ \oplus x_n = 0 \, , \quad \text{oder :} \ \sum_{i=1}^{n} x_i = 0 \, , \quad \text{Addition in GF(2)} \, .$$

- Für $k = 2 \Rightarrow n = 3$ ergeben sich die folgenden vier Codeworte, wobei in der ersten Zeile das Prüfbit jeweils durch einen kleinen Pfeil markiert ist:

  $$\underline{x}_0 = (0, 0 \rightarrow 0) \, , \ \underline{x}_1 = (0, 1 \rightarrow 1) \, , \ \underline{x}_2 = (1, 0 \rightarrow 1) \, , \ \underline{x}_3 = (1, 1 \rightarrow 0)$$
  $$\Rightarrow \quad \mathcal{C} = \{(0, 0, 0) \, , (0, 1, 1) \, , (1, 0, 1) \, , (1, 1, 0)\} \, .$$

- Es handelt sich um einen **linearen Code**, da die Summe zweier beliebiger Codeworte wieder ein gültiges Codewort ergibt, zum Beispiel $\underline{x}_1 \oplus \underline{x}_2 = \underline{x}_3$.
- Für beliebiges $k \Rightarrow n = k + 1$ unterscheidet sich jedes Codewort von allen anderen an einer geraden Anzahl von Positionen. Bei diesem Code ist die **minimale Distanz** $d_{\rm min} = 2$.

Mit der allgemeinen Codebezeichnung $(n, k, d_{\rm min})$ lässt sich jeder *Single Parity–check Code* auch mit **$(n, n - 1, 2)$** benennen. Die Grafik zeigt den SPC (3, 2, 2), den SPC (4, 3, 2) und den SPC (5, 4, 2).

Definition: Jeder **Single Parity–check Code** (SPC) lässt sich formal wie folgt beschreiben:

$$\mathcal{C} = \{\underline{x} \in \text{GF}(2^n) \text{: mit geradzahliger Anzahl von Einsen in } \underline{x}\} \, .$$

Single Parity–check Codes (2)

Der digitale Kanal ändert möglicherweise das Codewort $\underline{x} = (x_1, x_2, \ ... \ , x_n)$ in das Empfangswort $\underline{y} = (y_1, y_2, \ ... \ , y_n)$, wobei mit dem Fehlervektor $\underline{e} = (e_1, e_2, \ ... \ , e_n)$ gilt: $\underline{y} = \underline{x} \ \oplus \ \underline{e}$. Zur Decodierung des *Single Parity–check Codes* bildet man das sogenannte **Syndrom**:

$$s = y_1 \oplus y_2 \oplus \ ... \ \oplus y_n = \sum_{i=1}^{n} \ y_i \ \in \ \{0, 1\} \, .$$

Das Ergebnis „$s = 1$" weist dann auf (mindestens) einen Bitfehler innerhalb des Codewortes hin, während „$s = 0$" wie folgt zu interpretieren ist:

- Die Übertragung war fehlerfrei, oder:
- Die Anzahl der Bitfehler ist geradzahlig.

Beispiel: Wir betrachten den SPC (4, 3, 2) und gehen davon aus, dass das Nullwort gesendet wurde. Die Tabelle zeigt alle Möglichkeiten, dass f Bit verfälscht werden und gibt das jeweilige Syndrom s (entweder 0 oder 1) an.

$f = 0, s = 0$	$f = 1, s = 1$	$f = 2, s = 0$	$f = 3, s = 1$	$f = 4, s = 0$
0 0 0 0	0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0	0 0 1 1 0 1 0 1 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0 0	1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1	1 1 1 1

Für das **BSC–Modell** mit $\varepsilon = 1\%$ ergeben sich dann folgende Wahrscheinlichkeiten:

- Das Informationswort wird **richtig decodiert** (blaue Hinterlegung):

$$\Pr(\underline{v} = \underline{u}) = \Pr(\underline{y} = \underline{x}) = (1 - \varepsilon)^n = 0.99^4 \approx 96\,\% \, .$$

- Der Decoder **erkennt**, dass Übertragungsfehler aufgetreten sind (grüne Hinterlegung):

$$\Pr(s = 1) \ = \sum_{\substack{f=1 \\ f \ \text{ungerade}}}^{n} \binom{n}{f} \cdot \varepsilon^f \cdot (1 - \varepsilon)^{n-f} =$$

$$= \binom{4}{1} \cdot 0.01 \cdot 0.99^3 + \binom{4}{3} \cdot 0.01^3 \cdot 0.99 \approx 3.9\,\% \, .$$

- Das Informationswort wird **falsch decodiert** (rote Hinterlegung):

$$\Pr(\underline{v} \neq \underline{u}) \ = \sum_{\substack{f=2 \\ f \ \text{gerade}}}^{n} \binom{n}{f} \cdot \varepsilon^f \cdot (1 - \varepsilon)^{n-f} =$$

$$= \ 1 - \Pr(\underline{v} = \underline{u}) - \Pr(s = 1) \approx 0.1\,\% \, .$$

Wir verweisen hier auf das Modul **Ereigniswahrscheinlichkeiten der Binomialverteilung.**

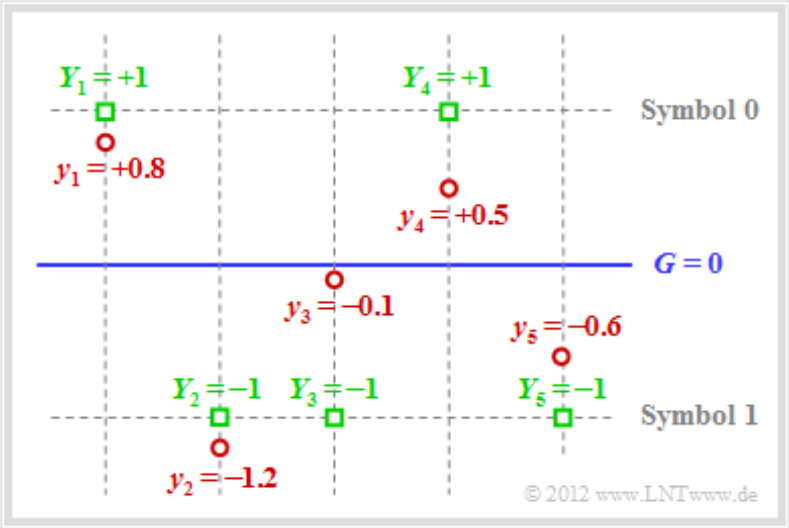
In der **Aufgabe A1.5** werden die hier gewonnenen Ergebnisse noch ausführlich diskutiert.

Single Parity–check Codes (3)

Eine Fehlerkorrektur des Single Parity–check Codes ist beim BSC–Modell nicht möglich im Unterschied zum **BEC–Kanal** (*Binary Erasure Channel*). Bei diesem werden Bitfehler ausgeschlossen. Ist nur ein Bit ausgelöscht (englisch: *Erasure*, E), so ist aufgrund der Tatsache „die Anzahl der Einsen im Codewort ist gerade" auch eine Fehlerkorrektur möglich, zum Beispiel für den SPC (5, 4):

$$\underline{y} = (1, 0, \mathrm{E}, 1, 1) \Rightarrow \underline{z} = (1, 0, 1, 1, 1) \Rightarrow \underline{v} = (1, 0, 1, 1) = \underline{u},$$
$$\underline{y} = (0, 1, 1, \mathrm{E}, 0) \Rightarrow \underline{z} = (0, 1, 1, 0, 0) \Rightarrow \underline{v} = (0, 1, 1, 0) = \underline{u},$$
$$\underline{y} = (0, 1, 0, 1, \mathrm{E}) \Rightarrow \underline{z} = (0, 1, 0, 1, 0) \Rightarrow \underline{v} = (0, 1, 0, 1) = \underline{u}.$$

Auch beim **AWGN–Kanal** ist Fehlerkorrektur möglich, wenn man *Soft Decision* anwendet. Für das Folgende gehen wir von bipolarer Signalisierung aus: $x = 0 \rightarrow \tilde{x} = +1, \quad x = 1 \rightarrow \tilde{x} = -1$.



Die Grafik verdeutlicht den hier dargelegten Sachverhalt:

- Beispielsweise lautet der Empfangsvektor (rote Punkte):

$$\underline{y} = (+0.8, -1.2, -0.1, +0.5, -0.6).$$

- Bei harter Entscheidung (Schwelle $G = 0$, nur die Vorzeichen werden ausgewertet) würde man zu folgendem binären Ergebnis kommen (grüne Quadrate $Y_i = y_i / |y_i|$):

$$\underline{Y} = (+1, -1, -1, +1, -1).$$

- In Symbolschreibweise ergibt sich daraus (0, 1, 1, 0, 1), was kein gültiges Codewort ist $\Rightarrow$ Syndrom $s = 1$. Also müssen ein, drei oder fünf Bit verfälscht worden sein.
- Die Wahrscheinlichkeit für drei oder fünf Bitfehler ist allerdings um Größenordnungen kleiner als diejenige für einen einzigen Fehler. Die Annahme „ein Bitfehler" ist deshalb nicht abwegig.
- Da der Empfangswert y_3 sehr nahe an der Schwelle $G = 0$ liegt, geht man davon aus, dass genau dieses Bit verfälscht wurde. Damit fällt bei *Soft Decision* die Entscheidung für $\underline{z} = (0, 1, 0, 0, 1)$ $\Rightarrow$ $\underline{v} = (0, 1, 0, 0)$. Die Blockfehlerwahrscheinlichkeit $\Pr(\underline{v} \neq \underline{u})$ ist so am geringsten.

Wiederholungscodes (1)

Definition: Man bezeichnet einen linearen binären (n, k)–Blockcode

$$\mathcal{C} = \{\underline{x} \in \mathrm{GF}(2^n) : x_i = x_j \text{ für alle } i, j = 1, \ldots, n\}$$

als **Wiederholungscode** (englisch: *Repetition Code*, RC) der Länge n. Es gilt also stets $\boldsymbol{k = 1}$. Entsprechend existieren nur zwei Codeworte (0, 0, ... , 0) und (1, 1, ... , 1), die sich in n Binärstellen unterscheiden. Daraus folgt für die minimale Distanz $\boldsymbol{d_{\rm min} = n}$.

Ein solcher $(n, 1, n)$–Blockcode besitzt nur die sehr kleine Coderate $R = 1/n$, ist aber auch sehr robust. Insbesondere beim BEC–Kanal genügt ein einziges richtig übertragenes Bit an beliebiger Position (alle anderen Bit können ausgelöscht sein), um das Informationswort richtig zu decodieren.

Betrachten wir zunächst die Decodierung und Fehlerwahrscheinlichkeiten beim BSC–Kanal.

Beispiel: Es gelte der **BSC–Kanal** mit $\varepsilon = 10\%$. Die Decodierung basiere auf dem Majoritätsprinzip. Bei ungeradem n können $e = n - 1$ Fehler erkannt und $t = (n - 1)/2$ Bitfehler korrigiert werden. Damit ergibt sich für die Wahrscheinlichkeit der korrekten Decodierung der Informationsbits u:

$$\Pr(v = u) = \sum_{f=0}^{(n-1)/2} \binom{n}{f} \cdot \varepsilon^f \cdot (1-\varepsilon)^{n-f}.$$

Die nachfolgenden Zahlenwerte gelten für $n = 5$. Das heißt: Es sind $t = 2$ Bitfehler korrigierbar:

$$\Pr(v = u) = (1-\varepsilon)^5 + 5 \cdot \varepsilon \cdot (1-\varepsilon)^4 + 10 \cdot \varepsilon^2 \cdot (1-\varepsilon)^3 \approx 99.15\,\%$$
$$\Rightarrow \quad \Pr(v \neq u) = 1 - \Pr(v = u) \approx 0.85\,\%.$$

Bei geradem n können dagegen nur $n/2 - 1$ Fehler korrigiert werden. Erhöht man n von 5 auf 6, so sind weiterhin auch nur zwei Bitfehler innerhalb eines Codewortes korrigierbar. Einen dritten Bitfehler kann man zwar nicht korrigieren, aber zumindest erkennen:

$$\Pr(\text{nicht korrigierbarer Fehler}) = \binom{6}{3} \cdot \varepsilon^3 \cdot (1-\varepsilon)^3 = 20 \cdot 0.1^3 \cdot 0.9^3 \approx 1.46\,\%.$$

Ein (unerkannter) Decodierfehler ($v \neq u$) ergibt sich erst, wenn innerhalb des 6 Bit–Wortes vier oder mehr Bit verfälscht wurden. Als Näherung unter der Annahme, dass fünf oder sechs Bitfehler sehr viel unwahrscheinlicher sind als vier, gilt:

$$\Pr(v \neq u) \approx \binom{6}{4} \cdot \varepsilon^4 \cdot (1-\varepsilon)^2 = 0.122\,\%.$$

Interessant ist, dass beim RC(6, 1, 6) die Wahrscheinlichkeit $\Pr(v = u)$ für eine mögliche und richtige Decodierung mit 98.42% kleiner ist als beim RC(5, 1, 5).

Wiederholungscodes (2)

Nun interessieren wir uns für die Leistungsfähigkeit des Wiederholungscodes beim **AWGN–Kanal.** Bei uncodierter Übertragung (oder dem Wiederholungscode mit $n = 1$) ist der Empfangswert $y = \tilde{x} + \eta$, wobei $\tilde{x} \in \{+1, -1\}$ das Informationsbit bei bipolarer Signalisierung bezeichnet und η den Rauschterm. Um Verwechslungen mit dem Codeparameter n zu vermeiden, wurde das Rauschen umbenannt: $n \rightarrow \eta$.

Für die Fehlerwahrscheinlichkeit gilt mit dem **komplementären Gaußschen Fehlerintegral** $\mathrm{Q}(x)$

$$\Pr(v \neq u) = \mathrm{Q}(\sqrt{\rho}),$$

wobei folgende physikalische Größen zu verwenden sind:

- das Signal–zu–Rauschleistungsverhältnis $\rho = 1/\sigma^2 = 2\,E_\mathrm{S}/N_0$,
- die Energie E_S pro Codesymbol $\Rightarrow$ Symbolenergie,
- die normierte Streuung σ des Rauschens, gültig für das Nutzsignal $\tilde{x} \in \{+1, -1\}$, und
- die konstante (einseitige) Rauschleistungsdichte N_0 des AWGN–Rauschens.

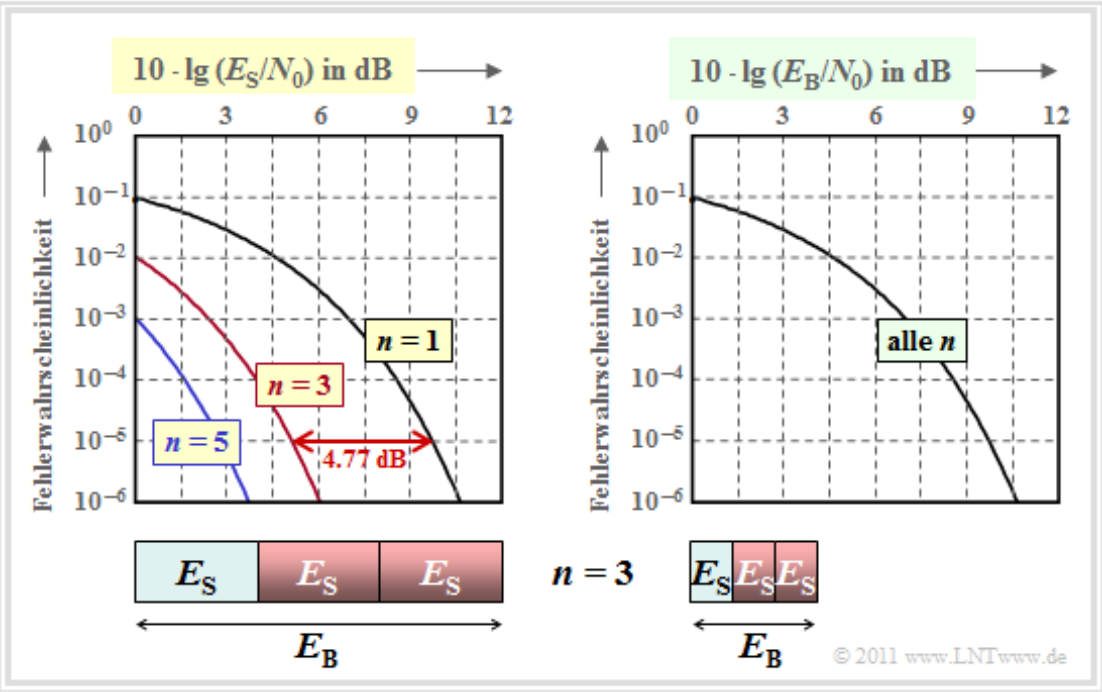
Bei einem $(n,\ 1,\ n)$–Wiederholungscode ergibt sich dagegen für den Eingangswert des ML–Decoders $y' = \tilde{x}' + \eta'$ mit folgenden Eigenschaften:

$$\tilde{x}' = \sum_{i=1}^{n} \tilde{x}_i \in \{+n, -n\} \;\Rightarrow\; n\text{–fache Amplitude} \;\Rightarrow\; n^2\text{–fache Leistung},$$

$$\eta' = \sum_{i=1}^{n} \eta_i \;\Rightarrow\; n\text{–fache Varianz}: \sigma^2 \rightarrow n \cdot \sigma^2,$$

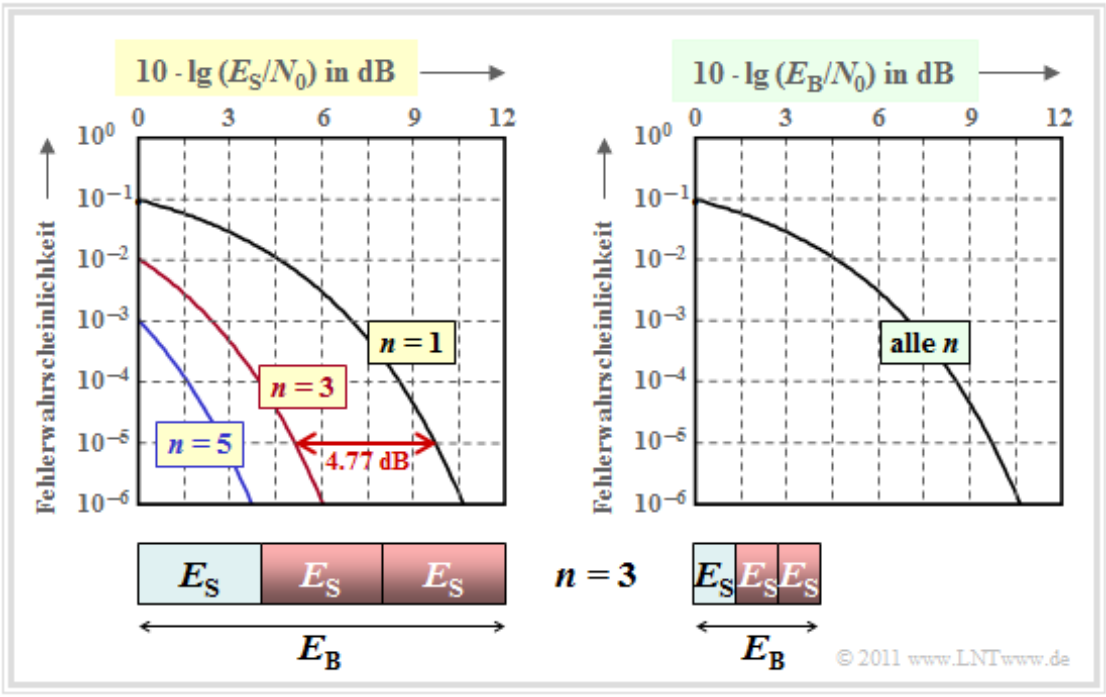
$$\rho' = \frac{n^2}{n \cdot \sigma^2} = n \cdot \rho \;\Rightarrow\; \Pr(v \neq u) = \mathrm{Q}\left(\sqrt{n \cdot \frac{2E_\mathrm{S}}{N_0}}\right).$$

Die linke Grafik zeigt die Fehlerwahrscheinlichkeit in doppelt logarithmischer Darstellung. Als Abszisse ist $10 \cdot \lg\,(E_\mathrm{S}/N_0)$ aufgetragen. Die Bildbeschreibung folgt auf der nächsten Seite.



Wiederholungscodes (3)

Es folgt nun die Interpretation der angegebenen Grafiken.



Die linke Grafik kann wie folgt interpretiert werden:

- Trägt man die Fehlerwahrscheinlichkeit über der Abszisse $10 \cdot \lg (E_S/N_0)$ auf, so ergibt sich durch n–fache Wiederholung gegenüber uncodierter Übertragung ($n = 1$) eine signifikante Verbesserung.
- Die Kurve für den Wiederholungsfaktor n erhält man durch Linksverschiebung um $10 \cdot \lg n$ (in dB) gegenüber der Vergleichskurve. Der Gewinn beträgt 4.77 dB ($n = 3$) bzw. ca. 7 dB ($n = 5$).
- Allerdings ist ein Vergleich bei konstantem E_S nicht fair, da man mit einem (5, 1) Repetition Code für die Übertragung eines Informationsbits eine um den Faktor n größere Energie aufwendet:

$$E_B = \frac{E_S}{R} = n \cdot E_S .$$

Aus der rechten Grafik erkennt man, dass alle Kurven genau übereinander liegen, wenn auf der Abszisse $10 \cdot \lg (E_B/N_0)$ aufgetragen wird. Daraus folgt:

Zusammenfassung der Ergebnisse für den Wiederholungscode bei AWGN

- Die Fehlerwahrscheinlichkeit ist bei fairem Vergleich unabhängig vom Wiederholungsfaktor n:

$$\Pr(v \neq u) = \mathrm{Q}\left(\sqrt{\frac{2E_B}{N_0}}\right) .$$

- Beim AWGN–Kanal ist durch einen Wiederholungscode kein **Codiergewinn** zu erzielen.

Hamming–Codes (1)

Richard Wesley Hamming hat 1962 eine Klasse binärer Blockcodes angegeben, die sich durch die Anzahl $m = 2, 3, ...$ der zugesetzten *Parity Bits* unterscheiden. Für diese Codeklasse gilt:

- Die Codelänge ergibt sich zu $n = 2^m - 1$. Möglich sind demzufolge beim Hamming–Code nur die Längen $n = 3$, $n = 7$, $n = 15$, $n = 31$, $n = 63$, $n = 127$, $n = 255$, usw.
- Ein Informationswort besteht aus $k = n - m$ Bit. Die Coderate ist somit gleich

$$R = \frac{k}{n} = \frac{2^m - 1 - m}{2^m - 1} \in \{1/3,\ 4/7,\ 11/15,\ 26/31,\ 57/63,\ 120/127,\ 247/255,\ ...\}\,.$$

- Alle Hamming–Codes weisen die minimale Distanz $\boldsymbol{d_{min} = 3}$ auf. Bei größerer Codelänge n erreicht man die minimale Distanz 3 schon mit weniger Redundanz, also bei größerer Coderate R.
- Aus der Angabe $d_{min} = 3$ folgt weiter, dass hier $e = d_{min} - 1 = 2$ Fehler erkannt werden können und $t = (d_{min} - 1)/2 = 1$ Fehler korrigiert werden kann.
- Der Hamming–Code (3, 1, 3) ist identisch mit dem Wiederholungscode (3, 1, 3):

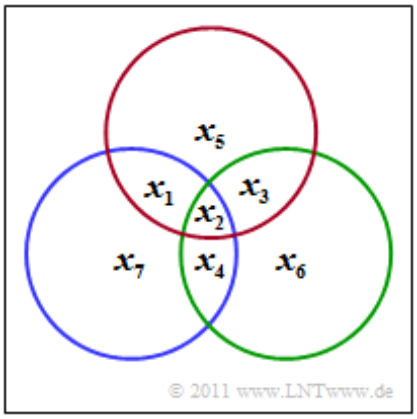
$$\mathcal{C} = \{(0, 0, 0)\,, (1, 1, 1)\}\,.$$

- Bei systematischer Codierung sind die ersten k Stellen eines jeden Codewortes identisch mit dem Informationswort. Anschließend folgen bei einem Hamming–Code die $m = n - k$ Paritätsbit:

$$\underline{x} = (x_1, x_2, \ldots, x_n) = (u_1, u_2, \ldots, u_k, p_1, p_2, \ldots, p_{n-k})\,.$$

Im Folgenden betrachten wir stets den **(7, 4, 3)–Hamming–Code**, der durch das folgende Schaubild verdeutlicht wird. Daraus lassen sich die drei Bedingungen ableiten:

$$\begin{aligned} x_1 \oplus x_2 \oplus x_3 \oplus x_5 &= 0\,,\\ x_2 \oplus x_3 \oplus x_4 \oplus x_6 &= 0\,,\\ x_1 \oplus x_2 \oplus x_4 \oplus x_7 &= 0\,. \end{aligned}$$



Im Schaubild kennzeichnet der rote Kreis die erste Prüfgleichung, der grüne die zweite und der blaue die letzte. In jedem Kreis muss die Anzahl der Einsen geradzahlig sein.

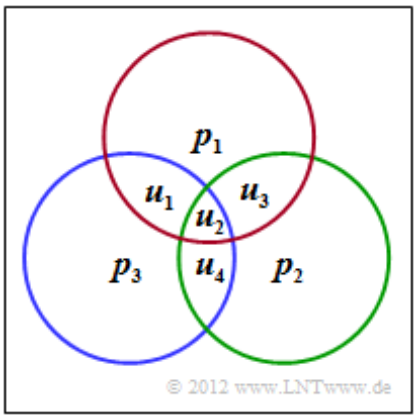
Hamming–Codes (2)

Bei systematischer Codierung des (7, 4, 3)–Hamming–Codes

$$x_1 = u_1, \quad x_2 = u_2, \quad x_3 = u_3, \quad x_4 = u_4,$$
$$x_5 = p_1, \quad x_6 = p_2, \quad x_7 = p_3$$

lauten die Bestimmungsgleichungen der drei Prüfbit, wie aus dem Schaubild rechts hervorgeht:

$$p_1 = u_1 \oplus u_2 \oplus u_3 ,$$
$$p_2 = u_2 \oplus u_3 \oplus u_4 ,$$
$$p_3 = u_1 \oplus u_2 \oplus u_4 .$$



Die nachfolgende Tabelle zeigt die $2^k = 16$ zulässigen Codeworte $\underline{x} = (u_1, u_2, u_3, u_4, p_1, p_2, p_3)$ des systematischen (7, 4, 3)–Codes. Das Informationswort $\underline{u} = (u_1, u_2, u_3, u_4)$ ist schwarz dargestellt und die Prüfbits p_1, p_2, p_3 rot. Man erkennt anhand dieser Tabelle, dass sich jeweils zwei der 16 möglichen Codeworte in mindestens $d_{\rm min} = 3$ Binärwerten unterscheiden.

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

Die **Decodierung linearer Blockcodes** wird im **Kapitel 1.5** ausführlich behandelt. Das nun folgende Beispiel soll die Decodierung des Hamming–Codes eher intuitiv erklären.

Beispiel: Wir gehen vom systematischen (7, 4, 3)–Code aus und betrachten das Empfangswort $\underline{y} = (y_1, y_2, y_3, y_4, y_5, y_6, y_7)$. Zur Decodierung bilden wir die drei Paritätsgleichungen

$$y_1 \oplus y_2 \oplus y_3 \oplus y_5 = 0, \quad \text{(I)}$$
$$y_2 \oplus y_3 \oplus y_4 \oplus y_6 = 0, \quad \text{(II)}$$
$$y_1 \oplus y_2 \oplus y_4 \oplus y_7 = 0. \quad \text{(III)}$$

Unter der Voraussetzung, dass in jedem Codewort höchstens ein Bit verfälscht wird, gelten dann die folgenden Aussagen. $\underline{v}$ bezeichnet das Decodierergebnis und sollte mit $\underline{u} = (1, 0, 1, 0)$ übereinstimmen:

- Das Empfangswort $\underline{y} = (1, 0, 1, 0, 0, 1, 1)$ erfüllt alle drei Paritätsgleichungen. Das heißt, dass kein einziger Übertragungsfehler aufgetreten ist $\Rightarrow$ $\underline{y} = \underline{x}$ $\Rightarrow$ $\underline{u} = (1, 0, 1, 0)$.
- Werden zwei der drei Paritätsgleichungen erfüllt wie zum Beispiel für das empfangene Wort $\underline{y} = (1, 0, 1, 0, 0, 1, 0)$, so wurde ein Paritätsbit verfälscht und es gilt auch hier $\underline{v} = (1, 0, 1, 0)$.
- Mit $\underline{y} = (1, 0, 1, 1, 0, 1, 1)$ wird nur die Gleichung (I) erfüllt und die beiden anderen nicht. Somit kann man die Verfälschung des vierten Binärsymbols korrigieren, und es gilt auch hier $\underline{v} = \underline{u}$.
- Ein Übertragungsfehler des zweiten Bits $\Rightarrow$ $\underline{y} = (1, 1, 1, 0, 0, 1, 1)$ führt dazu, dass alle drei Paritätsgleichungen nicht erfüllt werden. Auch dieser Fehler lässt sich eindeutig korrigieren.

Lineare Codes und zyklische Codes

Alle bisher behandelten Codes – *Single Parity–check Code*, *Repetition Code* und *Hamming–Code* – sind linear. Nun wird die für binäre Blockcodes gültige Definition von Linearität nachgereicht.

Definiton: Ein **linearer Blockcode** C ist ein Satz von 2^k Codeworten $\underline{x} = (x_1, x_2, \ldots, x_n)$, wobei die (Modulo–2)–Summe zweier beliebiger Codeworte $\underline{x}$ und $\underline{x}'$ wiederum ein gültiges Codewort ergibt:

$$\underline{x}, \underline{x}' \in \mathrm{GF}(2^n), \quad \underline{x}, \underline{x}' \in \mathcal{C} \quad \Rightarrow \quad \underline{x} + \underline{x}' \in \mathcal{C}\,.$$

Diese Bedingung muss auch für $\underline{x} = \underline{x}'$ erfüllt sein.

Hinweis: Die Modulo–Addition wird nun nicht mehr durch das Modulo–Additionszeichen ausgedrückt, sondern mit dem herkömmlichen Pluszeichen. Diese Vereinfachung der Schreibweise wird für den Rest dieses Buches beibehalten.

Beispiel: Wir betrachten zwei (3, 2)–Blockcodes:

$$\mathcal{C}_1 = \{(0,0,0)\,,(0,1,1)\,,(1,0,1)\,,(1,1,0)\}\,,$$

$$\mathcal{C}_2 = \{(0,0,0)\,,(0,1,1)\,,(1,1,0)\,,(1,1,1)\,.$$

Man erkennt:

- Der Code C_1 ist linear, da die Modulo–2–Addition zweier beliebiger Codeworte stets auch ein gültiges Codewort ergibt, zum Beispiel (0, 1, 1) + (1, 0, 1) = (1, 1, 0).
- Die obige Definition gilt auch für die Modulo–2–Addition eines Codewortes mit sich selbst, zum Beispiel (0, 1, 1) + (0, 1, 1) = (0, 0, 0) ⇒ Jeder lineare Code beinhaltet das Nullwort.
- Obwohl die letzte Voraussetzung erfüllt wird, ist C_2 kein linearer Code. Für diesen Code gilt nämlich beispielsweise: (0, 1, 1) + (1, 1, 0) = (1, 0, 1). Dies ist kein gültiges Codewort von C_2.

Im Folgenden beschränken wir uns ausschließlich auf lineare Codes, da nichtlineare Codes für die Praxis von untergeordneter Bedeutung sind.

Definition: Ein linearer Blockcode C heißt **zyklisch**, wenn eine jede zyklische Verschiebung eines Codewortes $\underline{x}$ (nach links oder rechts) wieder ein gültiges Codewort ergibt:

$$\underline{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{C} \quad \Rightarrow \quad \underline{x}' = (x_n, x_1, \ldots, x_{n-1}) \in \mathcal{C}\,.$$

Man erkennt aus der Tabelle für den HC (7, 4, 3), dass dieser linear und zyklisch ist (schwarz: 4 Informationsbit, rot: $n - k = 3$ Prüfbit).

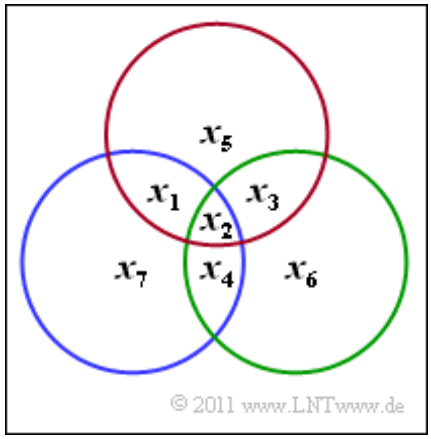
0000000	0100111	1000101	1100010
0001011	0101100	1001110	1101001
0010110	0110001	1010011	1110100
0011101	0111010	1011000	1111111

Außerdem ergibt sich auch dann ein gültiges Codewort, wenn man alle Bit invertiert: 0 ↔ 1. Auch das $\underline{0}$–Wort (n mal eine „0") und das $\underline{1}$–Wort (n mal eine „1") sind bei diesem Code zulässig.

Codefestlegung durch die Prüfmatrix

Wir betrachten den **(7, 4, 3)–Hamming–Code** mit Codeworten $\underline{x}$ der Länge $n = 7$, nämlich

- den $k = 4$ Informationsbits x_1, x_2, x_3, x_4 , und
- den $m = 3$ Prüfbits x_5, x_6, x_7.



Die Paritätsgleichungen lauten somit:

$$x_1 + x_2 + x_3 + x_5 = 0,$$
$$x_2 + x_3 + x_4 + x_6 = 0,$$
$$x_1 + x_2 + x_4 + x_7 = 0.$$

In der Matrixschreibweise lautet dieser Gleichungssatz:

$$\mathbf{H} \cdot \underline{x}^{\rm T} = \underline{0}^{\rm T}.$$

In dieser Gleichung werden verwendet:

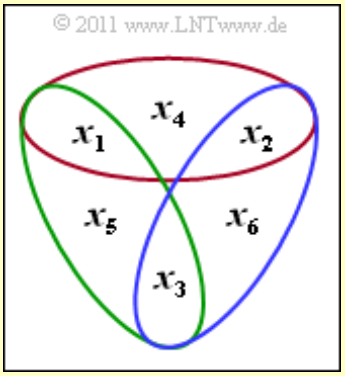
- die **Prüfmatrix** $\mathbf{H}$ mit $m = n - k = 3$ Zeilen und $n = 7$ Spalten:

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

- das **Codewort** $\underline{x} = (x_1, x_2, \ldots, x_7)$ der Länge $n = 7$,
- der **Nullvektor** $\underline{0} = (0, 0, 0)$ der Länge $m = 3$.

Durch Transponieren werden aus $\underline{x}$ und $\underline{0}$ die entsprechenden Spaltenvektoren $\underline{x}^{\rm T}$ und $\underline{0}^{\rm T}$.

Beispiel: Die Grafik illustriert die $m = 3$ Paritätsgleichungen eines Codes C mit den Codeparametern $n = 6$ und $k = 3$ in der Reihenfolge rot, grün und blau. Entsprechend $\mathbf{H} \cdot \underline{x}^{\rm T} = \underline{0}^{\rm T}$ lautet die Prüfmatrix:



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

Die $2^k = 8$ Worte einer systematischen Realisierung dieses Codes lauten (mit den Prüfbits rechts vom kleinen Pfeil):

$$\underline{x}_0 = (0, 0, 0 \underset{\rightarrow}{} 0, 0, 0), \quad \underline{x}_1 = (0, 0, 1 \underset{\rightarrow}{} 0, 1, 1), \quad \underline{x}_2 = (0, 1, 0 \underset{\rightarrow}{} 1, 0, 1),$$
$$\underline{x}_3 = (0, 1, 1 \underset{\rightarrow}{} 1, 1, 0), \quad \underline{x}_4 = (1, 0, 0 \underset{\rightarrow}{} 1, 1, 0), \quad \underline{x}_5 = (1, 0, 1 \underset{\rightarrow}{} 1, 0, 1),$$
$$\underline{x}_6 = (1, 1, 0 \underset{\rightarrow}{} 0, 1, 1), \quad \underline{x}_7 = (1, 1, 1 \underset{\rightarrow}{} 0, 0, 0).$$

Man erkennt aus diesen Angaben:

- Die Anzahl der Spalten von $\mathbf{H}$ ist gleich der Codelänge n.
- Die Anzahl der Zeilen von $\mathbf{H}$ ist gleich der Anzahl $m = n - k$ der Prüfgleichungen.
- Aus $\mathbf{H} \cdot \underline{x}^{\rm T} = \underline{0}$ folgt nicht, dass alle Codeworte eine gerade Anzahl von Einsen beinhalten.

Codefestlegung durch die Generatormatrix

Die Prüfmatrix **H** eines (n, k)–Blockcodes C hat $m = n - k$ Zeilen und n Spalten. Den gleichen Code kann man aber auch durch die Generatormatrix **G** mit ebenfalls n Spalten, aber k Zeilen beschreiben:

Definition: Ein **linearer Blockcode** C kann durch die Prüfmatrix **H** bzw. mit der Generatormatrix **G** wie folgt charakterisiert werden:

$$\mathcal{C} = \{\underline{x} \in \mathrm{GF}(2^n) : \mathbf{H} \cdot \underline{x}^{\mathrm{T}} = \underline{0}^{\mathrm{T}}\},$$
$$\mathcal{C} = \{\underline{x} \in \mathrm{GF}(2^n), \underline{u} \in \mathrm{GF}(2^k) : \underline{x} = \underline{u} \cdot \mathbf{G}\}.$$

Bevor wir uns den Eigenschaften der Generatormatrix zuwenden, beschreiben wir an einem Beispiel die Erzeugung der Codeworte.

Beispiel: Wir betrachten einen linearen (5, 3)–Blockcode mit der Generatormatrix

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \underline{g}_1 \\ \underline{g}_2 \\ \underline{g}_3 \end{pmatrix}.$$

Damit werden die Informationsworte $\underline{u} = (u_1, u_2, u_3)$ den Codeworten $\underline{x} = (x_1, x_2, x_3, x_4, x_5)$ gemäß der folgenden Tabelle mit 8 Einträgen zugeordnet. Es gilt $\underline{x} = \underline{u} \cdot \mathbf{G}$.

$$\begin{aligned}
\underline{u}_0 &= (0,0,0) \quad \rightarrow \quad \underline{x}_0 = (0,0,0,0,0), \\
\underline{u}_1 &= (0,0,1) \quad \rightarrow \quad \underline{x}_1 = (0,1,1,1,0) = \underline{g}_3, \\
\underline{u}_2 &= (0,1,0) \quad \rightarrow \quad \underline{x}_2 = (0,1,0,1,0) = \underline{g}_2, \\
\underline{u}_3 &= (0,1,1) \quad \rightarrow \quad \underline{x}_3 = (0,0,1,0,0) = \underline{g}_2 + \underline{g}_3, \\
\underline{u}_4 &= (1,0,0) \quad \rightarrow \quad \underline{x}_4 = (1,1,0,1,1) = \underline{g}_1, \\
\underline{u}_5 &= (1,0,1) \quad \rightarrow \quad \underline{x}_5 = (1,0,1,0,1) = \underline{g}_1 + \underline{g}_3, \\
\underline{u}_6 &= (1,1,0) \quad \rightarrow \quad \underline{x}_6 = (1,0,0,0,1) = \underline{g}_1 + \underline{g}_2, \\
\underline{u}_7 &= (1,1,1) \quad \rightarrow \quad \underline{x}_7 = (1,1,1,1,1) = \underline{g}_1 + \underline{g}_2 + \underline{g}_3.
\end{aligned}$$

Die hier zur Berechnung herangezogenen Basisvektoren $\underline{g}_1$, $\underline{g}_2$, $\underline{g}_3$ – jeweils mit der Länge $n = 5$ – entsprechen den $k = 3$ Zeilen der Generatormatrix **G**. Anzumerken ist ferner, dass dieser Code wegen $d_{\min} = 1$ weder zur Fehlerkorrektur noch zur Fehlererkennung geeignet ist. Trotzdem wird er auch auf den nächsten Seiten beispielhaft weiter betrachtet.

Hinweis: An dieser Stelle möchten wir Sie auf ein Interaktionsmodul zum Buch „Digitalsignalübertragung" aufmerksam machen, das die Bedeutung und Berechnung von Basisfunktionen vermittelt, wenn auch in völlig anderem Zusammenhang als in diesem Buch:

Gram–Schmidt–Verfahren (Dateigröße: 1.43 MB)

Systematische Codes (1)

Die im Beispiel auf der letzten Seite verwendeten Vektoren $\underline{g}_1, \underline{g}_2, \ \text{...} \ , \underline{g}_k$ sind die **Basisvektoren** des linearen Blockcodes C. Der Code selbst kann als k–dimensionaler *Untervektorraum* von GF(2^n) angesehen werden. Die Basisvektoren $\underline{g}_1, \underline{g}_2, \ \text{...} \ , \underline{g}_k$ sind linear unabhängig.

Der Untervektorraum C wird aber nicht nur durch die Basisvektoren

$$\underline{g}_1 = (1, 1, 0, 1, 1)\,, \quad \underline{g}_2 = (0, 1, 0, 1, 0)\,, \quad \underline{g}_3 = (0, 1, 1, 1, 0)$$

aufgespannt, sondern andere Basisvektoren $\underline{g}'_1$, $\underline{g}'_2$ und $\underline{g}'_3$ sind ebenso geeignet, so lange zwischen diesen die lineare Unabhängigkeit gewährleistet ist.

Beispiel: Wir vergleichen die beiden Codes C und C' mit den Generatormatrizen

$$\mathbf{G} = \begin{pmatrix} \underline{g}_1 \\ \underline{g}_2 \\ \underline{g}_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \quad \mathbf{G}' = \begin{pmatrix} \underline{g}'_1 \\ \underline{g}'_2 \\ \underline{g}'_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Die beiden Codes sind identisch: Sie beinhalten die genau gleichen Codeworte; es gilt nur eine andere Zuordnung. Bei dem Übergang von **G** auf **G'** wurden folgende erlaubte Operationen ausgeführt:

$$\underline{g}'_1 = \underline{g}_1 + \underline{g}_2\,, \quad \underline{g}'_2 = \underline{g}_2\,, \quad \underline{g}'_3 = \underline{g}_2 + \underline{g}_3\,.$$

Zum entsprechenden Code C' kommt man mit der Gleichung $\underline{x}' = \underline{u} \cdot \mathbf{G}$.

$$\begin{aligned}
\underline{u}_0 &= (0,0,0) \ \rightarrow \ \underline{x}'_0 = (0,0,0,0,0) = \underline{x}_0\,,\\
\underline{u}_1 &= (0,0,1) \ \rightarrow \ \underline{x}'_1 = (0,0,1,0,0) = \underline{x}_3\,,\\
\underline{u}_2 &= (0,1,0) \ \rightarrow \ \underline{x}'_2 = (0,1,0,1,0) = \underline{x}_2\,,\\
\underline{u}_3 &= (0,1,1) \ \rightarrow \ \underline{x}'_3 = (0,1,1,1,0) = \underline{x}_1\,,\\
\underline{u}_4 &= (1,0,0) \ \rightarrow \ \underline{x}'_4 = (1,0,0,0,1) = \underline{x}_6\,,\\
\underline{u}_5 &= (1,0,1) \ \rightarrow \ \underline{x}'_5 = (1,0,1,0,1) = \underline{x}_5\,,\\
\underline{u}_6 &= (1,1,0) \ \rightarrow \ \underline{x}'_6 = (1,1,0,1,1) = \underline{x}_4\,,\\
\underline{u}_7 &= (1,1,1) \ \rightarrow \ \underline{x}'_7 = (1,1,1,1,1) = \underline{x}_7\,.
\end{aligned}$$

Die Codeworte $\underline{x}_i$ beziehen sich auf die Generatormatrix **G**. Sie finden diese auf der **letzten Seite.**

Diese Codetabelle macht nochmals deutlich:

- C und C' beinhalten die genau gleichen Codeworte. Sie sind damit *identische Codes* und besitzen beide die gleiche Korrekturfähigkeit (siehe nächste Seite).
- C' ist aber nun ein systematischer Code, da die ersten k Binärstellen eines jeden Codewortes $\underline{x}'$ mit den Binärstellen des Informationswortes $\underline{u}$ übereinstimmen.

Systematische Codes (2)

Die Eigenschaft „systematisch" soll nun noch in mathematischer Form angegeben werden.

Definition: Bei einem **systematischen (n, k)–Blockcode** $C_{\rm sys}$ beinhaltet jedes Codewort $\underline{x}$ explizit das Informationswort $\underline{u}$, das heißt es gilt

$$\underline{u} = (u_1, u_2, \ldots, u_k) \quad \rightarrow \quad \underline{x} = (u_1, u_2, \ldots, u_k, x_{k+1}, \ldots, x_n),$$

und die Generatormatrix hat die spezifische Form

$$\mathbf{G_{sys}} = (\mathbf{I}_k \,;\, \mathbf{P})$$

mit der $k\times k$–Einheitsmatrix $\mathbf{I}_k$ und einer geeignet zu wählenden $(n-k)\times k$–Matrix $\mathbf{P}$.

Für das Beispiel auf der letzten Seite kann also auch geschrieben werden:

$$\mathbf{G_{sys}} = (\mathbf{I}_3 \,;\, \mathbf{P}) \quad \text{mit} \quad \mathbf{I_3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{und} \quad \mathbf{P} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Erfreulich aus Sicht der Kanalcodierung ist, dass für jeden Code C ein systematischer (identischer oder zumindest äquivalenter) Code $C_{\rm sys}$ gefunden werden kann.

Zum **identischen Code** (das heißt $\underline{x}$ und $\underline{x}_{\rm sys}$ haben die gleichen Codeworte, nur die Zuordnung $\underline{u} \rightarrow \underline{x}$ ist unterschiedlich) kommt man durch folgende Manipulationen bezüglich $\mathbf{G}$:

- Vertauschen oder Permutieren der Zeilen,
- Multiplizieren aller Zeilen mit einem konstanten Vektor ungleich $\underline{0}$,
- Ersetzen einer Zeile durch eine Linearkombination zwischen dieser Zeile und einer anderen.

Ein identischer systematischer Code kann immer dann gefunden werden, wenn zu einer Generatormatrix $\mathbf{G}$ eine Matrix $\mathbf{A}$ existiert, so dass $\mathbf{G}_{\rm sys} = \mathbf{A} \cdot \mathbf{G}$ gilt. Ist dies nicht möglich, so findet man zumindest durch Vertauschen oder Permutieren der Spalten von $\mathbf{G}$ einen **äquivalenten systematischen Code**:

$$\mathcal{C}_{\rm sys} = \pi(\mathcal{C}) \quad \text{mit} \quad \pi() : \ \text{Permutationsoperator}.$$

Die Codes C und $C_{\rm sys}$ beinhalten dann zwar andere Codeworte, aber sie zeigen gleiche Eigenschaften. Beispielsweise weist $C_{\rm sys}$ die gleiche minimale Hamming–Distanz $d_{\rm min}$ auf wie der Code C.

Beispiel: Wir betrachten die Generatormatrizen

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{und} \quad \mathbf{G_{sys}} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

$\mathbf{G}_{\rm sys}$ ergibt sich aus $\mathbf{G}$ durch Vertauschen der zweiten und dritten Spalte. Die zugehörigen Codes beinhalten unterschiedliche Codeworte und sind somit auch *nicht identisch*. Aber sie sind *äquivalent*: Es handelt sich in beiden Fällen um einen (4, 2, 2)–Blockcode $\Rightarrow$ $d_{\rm min} = 2$:

$$\mathcal{C} = \{(0,0,0,0), (0,0,1,1), (1,1,0,0), (1,1,1,1)\},$$
$$\mathcal{C}_{\rm sys} = \{(0,0,0,0), (0,1,0,1), (1,0,1,0), (1,1,1,1)\}.$$

Zusammenhang zwischen Generator– und Prüfmatrix (1)

Zur Definition dieser beiden Beschreibungsmatrizen gehen wir von folgenden Definitionsgleichungen aus:

$$\underline{x} = \underline{u} \cdot \mathbf{G} \quad \Rightarrow \quad \underline{x}^{\rm T} = \mathbf{G}^{\rm T} \cdot \underline{u}^{\rm T},$$

$$\mathbf{H} \cdot \underline{x}^{\rm T} = \mathbf{0}.$$

Verknüpft man diese zwei Gleichungen, so erhält man:

$$\mathbf{H} \cdot \mathbf{G}^{\rm T} \cdot \underline{u}^{\rm T} = \underline{0} \quad \forall\, \underline{u} \in {\rm GF}(2^k) \quad \Rightarrow \quad \mathbf{H} \cdot \mathbf{G}^{\rm T} = \mathbf{0}.$$

Anzumerken ist, dass in diesen Gleichungen $\underline{0}$ einen Zeilenvektor mit k Elementen bezeichnet und $\mathbf{0}$ eine Matrix mit m Zeilen und k Spalten. Alle Elemente von $\underline{0}$ und $\mathbf{0}$ sind identisch 0.

Beispiel: Wir betrachten wie im **Beispiel** auf der dritten Seite dieses Kapitels den (5, 3)–Blockcode

$$\begin{aligned}\mathcal{C} = \{&(0,0,0,0,0),\\ &(0,1,1,1,0),\\ &(0,1,0,1,0),\\ &(0,0,1,0,0),\\ &(1,1,0,1,1),\\ &(1,0,1,0,1),\\ &(1,0,0,0,1),\\ &(1,1,1,1,1)\}.\end{aligned}$$

Aus $n = 5$ und $k = 3$ folgt für die Anzahl der Prüfgleichungen $m = 2$. Durch Analyse der möglichen Codeworte erhält man folgende Ergebnisse:

$$\begin{aligned}x_1 \oplus x_5 &= 0,\\ x_2 \oplus x_4 &= 0\end{aligned} \quad \Rightarrow \quad \mathbf{H} = \begin{pmatrix}1 & 0 & 0 & 0 & 1\\ 0 & 1 & 0 & 1 & 0\end{pmatrix}$$

$$\Rightarrow \quad \mathbf{H} \cdot \mathbf{G}^{\rm T} = \begin{pmatrix}1 & 0 & 0 & 0 & 1\\ 0 & 1 & 0 & 1 & 0\end{pmatrix} \cdot \begin{pmatrix}1 & 0 & 0\\ 1 & 1 & 1\\ 0 & 0 & 1\\ 1 & 1 & 1\\ 1 & 0 & 0\end{pmatrix} = \begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}.$$

Die Nullmatrix besteht hier aus $m = 2$ Zeilen und $k = 3$ Spalten. Beispielsweise gilt für das Element in der ersten Zeile und der ersten Spalte:

$$1 \cdot 1 \oplus 0 \cdot 1 \oplus 0 \cdot 0 \oplus 0 \cdot 1 \oplus 1 \cdot 1 = 0.$$

Im allgemeinen Fall können $\mathbf{H}$ und $\mathbf{G}$ nicht direkt ineinander umgerechnet werden, schon allein aufgrund der unterschiedlichen Dimensionen von Prüfmatrix (m x n) und Generatormatrix (k x n).

Zusammenhang zwischen Generator– und Prüfmatrix (2)

Der Rechengang vereinfacht sich, wenn die $k \times n$–Generatormatrix in systematischer Form vorliegt:

$$\mathbf{G_{sys}} = (\mathbf{I}_k \,;\, \mathbf{P}) \,.$$

Dann folgt aus $\mathbf{H} \cdot \mathbf{G}^{\rm T} = \mathbf{0}$ für die $m \times n$–Prüfmatrix mit $m = n - k$:

$$\mathbf{H} = \left(-\mathbf{P}^{\rm T} \,;\, \mathbf{I}_m\right) = \left[\left(\mathbf{P}^{\rm T} \,;\, \mathbf{I}_m\right)\right]_{\rm binär} \,.$$

Die erste der beiden Gleichungen gilt allgemein. Da wir uns im gesamten Kapitel 1 auf binäre Codes beschränken $\Rightarrow C \in {\rm GF}(2^n)$, gilt $-\mathbf{P} = +\mathbf{P}$, und man erhält die Form, die wir im Weiteren verwenden.

Beispiel: Wir betrachten weiterhin den beispielhaften (5, 3)–Blockcode, gehen aber nun von der **systematischen Generatormatrix** $G_{\rm sys}$ aus:

$$\mathbf{G_{sys}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} = (\mathbf{I}_3 \,;\, \mathbf{P})$$

$$\Rightarrow \quad \mathbf{I_3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{P} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \Rightarrow \quad \mathbf{P}^{\rm T} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} .$$

Damit erhält man für die Prüfmatrix

$$\mathbf{H} = (\mathbf{P}^{\rm T} \,;\, \mathbf{I}_2) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} ,$$

und es ergibt sich folgende Codetabelle:

$$\begin{aligned}
\underline{u}_0 &= (0,0,0) \;\rightarrow\; \underline{x}_0 = (0,0,0,0,0)\,, & \qquad \underline{u}_4 &= (1,0,0) \;\rightarrow\; \underline{x}_4 = (1,0,0,0,1)\,,\\
\underline{u}_1 &= (0,0,1) \;\rightarrow\; \underline{x}_1 = (0,0,1,0,0)\,, & \qquad \underline{u}_5 &= (1,0,1) \;\rightarrow\; \underline{x}_5 = (1,0,1,0,1)\,,\\
\underline{u}_2 &= (0,1,0) \;\rightarrow\; \underline{x}_2 = (0,1,0,1,0)\,, & \qquad \underline{u}_6 &= (1,1,0) \;\rightarrow\; \underline{x}_6 = (1,1,0,1,1)\,,\\
\underline{u}_3 &= (0,1,1) \;\rightarrow\; \underline{x}_3 = (0,1,1,1,0)\,, & \qquad \underline{u}_7 &= (1,1,1) \;\rightarrow\; \underline{x}_7 = (1,1,1,1,1)\,.
\end{aligned}$$

Zusammen mit dem Vektor $\underline{x} = (u_1, u_2, u_3, p_1, p_2) = (x_1, x_2, x_3, x_4, x_5)$ lauten dann die Prüfbits

$$p_1 = u_2\,, \quad p_2 = u_1\,,$$

und die entsprechenden Prüfgleichungen des Decoders:

$$x_2 + x_4 = 0\,, \quad x_1 + x_5 = 0\,.$$

Man erkennt aus diesen Gleichungen und auch aus obiger Codetabelle, dass dieser Code gegenüber einem Übertragungsfehler hinsichtlich des dritten Bits ($x_3 = u_3$) keinen Schutz bieten. Damit ist natürlich weder eine Fehlererkennung und noch weniger Fehlerkorrektur möglich. Gleiches gilt aber auch für den nichtsystematischen Code auf der letzten Seite.

Darstellung von SPC und RC als duale Codes

Nun sollen für die bereits in **Kapitel 1.3** behandelten Codes noch jeweils die Generatormatrix **G** und die Prüfmatrix **H** angegeben werden. Die Codelänge sei für die folgenden Beispiele stets $n = 5$, doch lassen sich die Ergebnisse auch für andere Codelängen in gleicher Weise interpretieren. Es gilt für

- den **Single–Parity–check Code** ⇒ SPC (5, 4):

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

- den **Wiederholungscode** (*Repetition Code*) ⇒ RC (5,1):

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Die jeweils erste Gleichung lässt sich einfach aus der jeweiligen Definition herleiten und die abgeleitete Gleichung folgt aus der Beziehung $\mathbf{H} \cdot \mathbf{G}^{\rm T} = \mathbf{0}$. Aus den obigen Matrizen kann verallgemeinert werden:

- Die Generatormatrix des RC (5, 1) ist identisch mit der Prüfmatrix des SPC (5, 4). Es handelt sich jeweils um (5, 1)–Matrizen.
- Die Prüfmatrix des RC (5, 1) ist identisch mit der Generatormatrix des SPC (5, 4). Die beiden Matrizen haben jeweils 5 Spalten und 4 Zeilen.
- Dieser Sachverhalt ergibt sich, weil es sich bei den hier betrachteten Beispielen um duale Codes handelt. Zur Erklärung benötigen wir noch zwei Definitionen:

Definition 1: Zwei lineare Codes C und C', beide aus GF(2^n), sind orthogonal, wenn alle Codeworte $\underline{x} \in C$ zu allen Codeworten $\underline{x}' \in C'$ orthogonal sind. Man bezeichnet C und C' als **duale Codes**.

Definition 2: Zwei Codeworte $\underline{x} \in$ GF(2^n) und $x' \in$ GF(2^n) sind immer dann zueinander **orthogonal**, wenn das innere Produkt verschwindet:

$$\langle \underline{x} \cdot \underline{x}' \rangle = \sum_{i=1}^{n} x_i \cdot x_i' = 0, \quad \langle \underline{x} \cdot \underline{x}' \rangle \in \mathrm{GF}(2^n).$$

Wegen der Produktbildung in GF(2^n) sind auch folgende Codewort–Paare zueinander orthogonal:

$$\langle (0,1,1,0), \ (1,1,1,0) \rangle = 0, \ \ \langle (0,1,1,0), \ (0,1,1,0) \rangle = 0.$$

Der Code C spannt einen k–dimensionalen Untervektorraum in GF(2^n) auf. Der Untervektorraum des dualen Codes C' ist zu diesem orthogonal und weist die Dimension $n - k$ auf. Damit gilt:

$$\dim\{C\} + \dim\{C'\} = n.$$

Einige Eigenschaften des (7, 4, 3)–Hamming–Codes

Fassen wir die bisherigen Ergebnisse dieses Kapitels am Beispiel des systematischen Hamming–Codes nochmals zusammen, der bereits im **Kapitel 1.3** ausführlich beschrieben wurde. Dieser (7, 4, 3)–Code ist gekennzeichnet durch

- die Anzahl der Prüfgleichungen: $m = 3$,
- die Codelänge: $n = 2^m - 1 = 7$,
- die Informationswortlänge: $k = n - m = 4$,
- die Anzahl der Codeworte (Dimension): $2^k = 16$,
- die Rate $R = k/n = 4/7$,
- die minimale Distanz $d_{\rm min} = 3$ (unabhängig von m, n und k).

0000000	0100111	1000101	1100010
0001011	0101100	1001110	1101001
0010110	0110001	1010011	1110100
0011101	0111010	1011000	1111111

In der obigen Tabelle sind die $2^k = 16$ Codeworte angegeben (Informationsbit schwarz, Prüfbit rot). Man erkennt daraus, dass

- der Code sowohl das 0–Wort (0000000) als auch das 1–Wort (1111111) beinhaltet,
- es sieben Codeworte gibt, die sich aus (0001011) jeweils durch zyklische Verschiebung ergeben (alle gelb hinterlegt),
- es sieben Codeworte gibt, die sich aus (0011101) jeweils durch zyklische Verschiebung ergeben (alle grün hinterlegt),
- zu jedem Codewort auch das „negierte" Codewort existiert, zum Beispiel neben (0001011) auch das Codewort (1110100),
- die Prüfmatrix wie folgt geschrieben werden kann:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} = \left(\mathbf{P}^{\rm T}\,;\,\mathbf{I}_3\right)$$

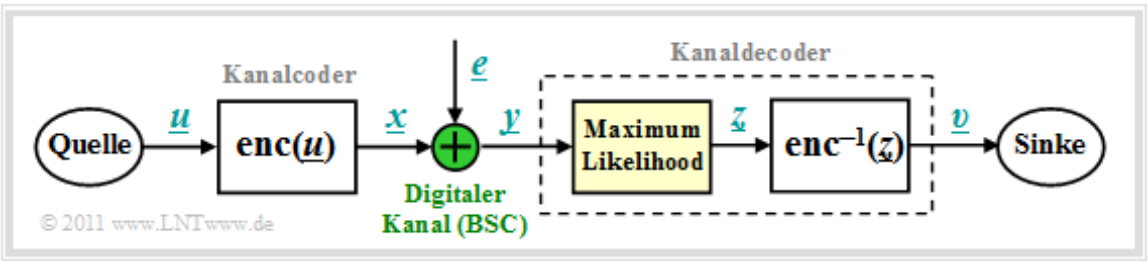
$$\Rightarrow \quad \mathbf{P}^{\rm T} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad \mathbf{I}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

- dementsprechend für die Generatormatrix gilt:

$$\mathbf{G} = \left(\mathbf{I}_4\,;\,\mathbf{P}\right) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Blockschaltbild und Voraussetzungen

Wir gehen von dem bereits im Kapitel 1.2 gezeigten Blockschaltbild aus, wobei als Kanalmodell meist der *Binary Symmetric Channel* (BSC) verwendet wird. Zur Codewortschätzung verwenden wir den *Maximum–Likelihood–Entscheider* (ML), der für binäre Codes $\Rightarrow \underline{x} \in \mathrm{GF}(2^n)$ das gleiche Ergebnis liefert wie der **MAP–Empfänger.**



Die Aufgabe des **Kanaldecoders** kann wie folgt beschrieben werden:

- Der Vektor $\underline{v}$ nach der Decodierung (an der Sinke) soll möglichst gut mit dem Informationswort $\underline{u}$ übereinstimmen. Das heißt: Die **Blockfehlerwahrscheinlichkeit** soll möglichst klein sein:

$$\Pr(\text{Blockfehler}) = \Pr(\underline{v} \neq \underline{u}) \stackrel{!}{=} \text{Minimum}\,.$$

- Aufgrund der deterministischen Zuweisungen $\underline{x} = \text{enc}(\underline{u})$ bzw. $\underline{v} = \text{enc}^{-1}(\underline{z})$ gilt aber auch:

$$\Pr(\text{Blockfehler}) = \Pr(\underline{z} \neq \underline{x}) \stackrel{!}{=} \text{Minimum}\,.$$

- Gesucht ist somit das zum gegebenen Empfangswort $\underline{y} = \underline{x} + \underline{e}$ am wahrscheinlichsten gesendete Codewort $\underline{x}_i$, das als Ergebnis $\underline{z}$ weiter gegeben wird:

$$\underline{z} = \arg \max_{\underline{x}_i \in \mathcal{C}} \Pr(\underline{x}_i \,|\, \underline{y})\,.$$

- Beim BSC–Kanal gilt sowohl $\underline{x}_i \in \mathrm{GF}(2^n)$ als auch $\underline{y} \in \mathrm{GF}(2^n)$, so dass die ML–Regel auch mit der **Hamming–Distanz** $d_{\rm H}(\underline{y}, \underline{x}_i)$ geschrieben werden kann:

$$\underline{z} = \arg \min_{\underline{x}_i \in \mathcal{C}} d_{\rm H}(\underline{y}\,,\, \underline{x}_i)\,.$$

Prinzip der Syndromdecodierung

Vorausgesetzt wird hier ein (n, k)–Blockcode mit der Prüfmatrix **H** und den systematischen Codeworten

$$\underline{x} = (x_1, x_2, \ldots, x_i, \ldots, x_n) = (u_1, u_2, \ldots, u_k, p_1, \ldots, p_{n-k}).$$

Mit dem Fehlervektor $\underline{e}$ gilt dann für das Empfangswort:

$$\underline{y} = \underline{x} + \underline{e}, \quad \underline{y} \in \mathrm{GF}(2^n), \ \underline{x} \in \mathrm{GF}(2^n), \ \underline{e} \in \mathrm{GF}(2^n).$$

Ein Bitfehler an der Position i, das heißt $y_i \neq x_i$, wird ausgedrückt durch den Fehlerkoeffizienten $e_i = 1$.

Definition: Das **Syndrom** $\underline{s} = (s_0, s_1, \ \ldots \ , s_{m-1})$ berechnet sich (als Zeilen– bzw. Spaltenvektor) aus dem Empfangswort $\underline{y}$ und der Prüfmatrix **H** in folgender Weise:

$$\underline{s} = \underline{y} \cdot \mathbf{H}^{\mathrm{T}} \quad \text{bzw.} \quad \underline{s}^{\mathrm{T}} = \mathbf{H} \cdot \underline{y}^{\mathrm{T}}.$$

Die Vektorlänge von $\underline{s}$ ist gleich $m = n - k$ (Zeilenzahl von **H**).

Das Syndrom $\underline{s}$ zeigt folgende Charakteristika:

- Wegen $\underline{x} \cdot \mathbf{H}^{\mathrm{T}} = \underline{0}$ hängt $\underline{s}$ nicht vom Codewort $\underline{x}$ ab, sondern allein vom Fehlervektor $\underline{e}$:

$$\underline{s} = \underline{y} \cdot \mathbf{H}^{\mathrm{T}} = \underline{x} \cdot \mathbf{H}^{\mathrm{T}} + \underline{e} \cdot \mathbf{H}^{\mathrm{T}} = \underline{e} \cdot \mathbf{H}^{\mathrm{T}}.$$

- Bei hinreichend wenig Bitfehlern liefert $\underline{s}$ einen eindeutigen Hinweis auf die Fehlerpositionen und ermöglicht so eine vollständige Fehlerkorrektur.

Beispiel: Ausgehend vom systematischen **(7, 4, 3)–Hamming–Code** erhält man beispielsweise für den Empfangsvektor $\underline{y} = (0, 1, 1, 1, 0, 0, 1)$ das folgende Ergebnis:

$$\mathbf{H} \cdot \underline{y}^{\mathrm{T}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \underline{s}^{\mathrm{T}}.$$

Vergleicht man das Syndrom mit den **Prüfgleichungen** des Hamming–Codes, so erkennt man, dass

- am wahrscheinlichsten das vierte Symbol ($x_4 = u_4$) des Codewortes verfälscht wurde,
- der Codewortschätzer somit das Ergebnis $\underline{z} = (0, 1, 1, 0, 0, 0, 1)$ liefern wird.
- die Entscheidung nur dann richtig ist, wenn bei der Übertragung nur ein Bit verfälscht wurde.

Nachfolgend sind die erforderlichen Korrekturen für den (7, 4, 3)–Hamming–Code angegeben, die sich aus dem errechneten Syndrom $\underline{s}$ entsprechend den Spalten der Prüfmatrix ergeben:

$$\begin{aligned}
&\underline{s} = (0,0,0) \Rightarrow \text{keine Korrektur}; \quad &&\underline{s} = (1,0,0) \Rightarrow p_1 \text{ invertieren}; \\
&\underline{s} = (0,0,1) \Rightarrow p_3 \text{ invertieren}; \quad &&\underline{s} = (1,0,1) \Rightarrow u_1 \text{ invertieren}; \\
&\underline{s} = (0,1,0) \Rightarrow p_2 \text{ invertieren}; \quad &&\underline{s} = (1,1,0) \Rightarrow u_3 \text{ invertieren}; \\
&\underline{s} = (0,1,1) \Rightarrow u_4 \text{ invertieren}; \quad &&\underline{s} = (1,1,1) \Rightarrow u_2 \text{ invertieren}.
\end{aligned}$$

Verallgemeinerung der Syndromdecodierung (1)

Wir fassen die Ergebnisse der letzten Seiten zusammen, wobei wir weiterhin vom BSC–Kanalmodell ausgehen. Das bedeutet: $\underline{y}$ und $\underline{e}$ sind Elemente von GF(2^n), während die möglichen Codeworte $\underline{x}_i$ zum Code C gehören, der einen $(n – k)$–dimensionalen Untervektorraum von GF(2^n) darstellt. Dann gilt:

- Die Syndromdecodierung ist eine Realisierungsmöglichkeit der Maximum–Likelihood–Detektion von Blockcodes. Man entscheidet sich für das Codewort mit der geringsten Hamming–Distanz zum Empfangswort:

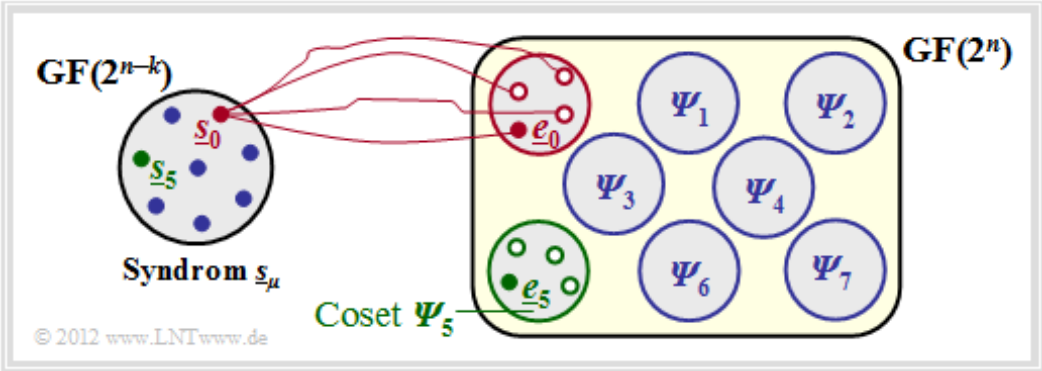
$$\underline{z} = \arg \min_{\underline{x}_i \in C} d_{\rm H}(\underline{y}, \underline{x}_i) .$$

- Die Syndromdecodierung ist aber auch die Suche nach dem wahrscheinlichsten Fehlervektor $\underline{e}$, der die Bedingung $\underline{e} \cdot \mathbf{H}^{\rm T} = \underline{s}$ erfüllt. Das *Syndrom* liegt dabei durch $\underline{s} = \underline{y} \cdot \mathbf{H}^{\rm T}$ fest.
- Mit dem **Hamming–Gewicht** $w_{\rm H}(\underline{e})$ kann die zweite Interpretation auch wie folgt mathematisch formuliert werden:

$$\underline{z} = \underline{y} + \arg \min_{\underline{e}_i \in {\rm GF}(2^n)} w_{\rm H}(\underline{e}_i) .$$

Zu beachten ist, dass der Fehlervektor $\underline{e}$ ebenso wie der Empfangsvektor $\underline{y}$ ein Element von GF(2^n) ist im Gegensatz zum Syndrom $\underline{s} \in$ GF(2^m) mit der Anzahl $m = n – k$ der Prüfgleichungen. Das bedeutet,

- dass die Zuordnung zwischen Syndrom $\underline{s}$ und Fehlervektor $\underline{e}$ nicht eindeutig ist, sondern
- dass jeweils 2^k Fehlervektoren zum gleichen Syndrom $\underline{s}$ führen, die man zu einer **Nebenklasse** (englisch: *Coset*) zusammenfasst.



Die Grafik verdeutlicht diesen Sachverhalt am Beispiel $n = 5$ und $k = 2 \Rightarrow m = n – k = 3$.

- Die insgesamt $2^n = 32$ möglichen Fehlervektoren $\underline{e}$ werden in $2^m = 8$ Nebenklassen Ψ_0, ... , Ψ_7 aufgeteilt, auch „*Cosets*" genannt. Explizit gezeichnet sind hier nur die Cosets Ψ_0 und Ψ_5.
- Alle $2^k = 4$ Fehlervektoren des Cosets Ψ_μ führen zum gleichen Syndrom $\underline{s}_\mu$. Zudem hat jede Nebenklasse einen Anführer $\underline{e}_\mu$, nämlich denjenigen mit dem minimalen Hamming–Gewicht.

Die Vorgehensweise bei der Syndromdecodierung wird auf den nächsten Seiten nochmals ausführlich an einem Beispiel erläutert.

Verallgemeinerung der Syndromdecodierung (2)

Die Syndromdecodierung wird hier am Beispiel eines systematischen (5, 2, 3)–Codes beschrieben:

$$\mathcal{C} = \{(0,0,0,0,0)\,,\ (0,1,0,1,1)\,,\ (1,0,1,1,0)\,,\ (1,1,1,0,1)\}\,.$$

Generatormatrix und Prüfmatrix lauten:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}\,,\quad \mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}\,.$$

Die Tabelle fasst das Endergebnis zusammen.

Index μ	Syndrom $\underline{s}_\mu$	Nebenklasse Ψ_μ inklusive „Anführer“ $\underline{e}_\mu$			
0	(0, 0, 0)	(0, 0, 0, 0, 0)	(1, 0, 1, 1, 0)	(0, 1, 0, 1, 1)	(1, 1, 1, 0, 1)
1	(0, 0, 1)	(0, 0, 0, 0, 1)	(1, 0, 1, 1, 1)	(0, 1, 0, 1, 0)	(1, 1, 1, 0, 0)
2	(0, 1, 0)	(0, 0, 0, 1, 0)	(1, 0, 1, 0, 0)	(0, 1, 0, 0, 1)	(1, 1, 1, 1, 1)
3	(1, 0, 0)	(0, 0, 1, 0, 0)	(1, 0, 0, 1, 0)	(0, 1, 1, 1, 1)	(1, 1, 0, 0, 1)
4	(0, 1, 1)	(0, 1, 0, 0, 0)	(1, 1, 1, 1, 0)	(0, 0, 0, 1, 1)	(1, 0, 1, 0, 1)
5	(1, 1, 0)	(1, 0, 0, 0, 0)	(0, 0, 1, 1, 0)	(1, 1, 0, 1, 1)	(0, 1, 1, 0, 1)
6	(1, 0, 1)	(1, 1, 0, 0, 0)	(0, 1, 1, 1, 0)	(1, 0, 0, 1, 1)	(0, 0, 1, 0, 1)
7	(1, 1, 1)	(0, 1, 1, 0, 0)	(1, 1, 0, 1, 0)	(0, 0, 1, 1, 1)	(1, 0, 0, 0, 1)

Zur Herleitung dieser Tabelle ist anzumerken:

- Die Zeile 1 bezieht sich auf das Syndrom $\underline{s}_0$ = (0, 0, 0) und die dazugehörige Nebenklasse Ψ_0. Am wahrscheinlichsten ist hier die Fehlerfolge (0, 0, 0, 0, 0) ⇒ kein Bitfehler, die wir als Nebenklassenanführer $\underline{e}_0$ bezeichnen. Auch die weiteren Einträge in der ersten Zeile, nämlich (1, 0, 1, 1, 0), (0, 1, 0, 1, 1) und (1, 1, 1, 0, 1), liefern jeweils das Syndrom $\underline{s}_0$ = (0, 0, 0), ergeben sich aber nur mit mindestens drei Bitfehlern und sind entsprechend unwahrscheinlich.
- In den Zeilen 2 bis 6 beinhaltet der jeweilige Nebenklassenanführer $\underline{e}_\mu$ genau eine einzige Eins. Dementsprechend ist $\underline{e}_\mu$ stets das wahrscheinlichste Fehlermuster der Klasse Ψ_μ (μ = 1, ... , 5). Die „Mitläufer” ergeben sich erst bei mindestens zwei Übertragungsfehlern.
- Das Syndrom $\underline{s}_6$ = (1, 0, 1) ist mit nur einem Bitfehler nicht möglich. Bei der Erstellung obiger Tabelle wurden daraufhin alle „5 über 2” = 10 Fehlermuster $\underline{e}$ mit Gewicht $w_{\rm H}(\underline{e})$ = 2 betrachtet. Die zuerst gefundene Folge mit Syndrom $\underline{s}_6$ = (1, 0, 1) wurde als $\underline{e}_6$ = (1, 1, 0, 0, 0) ausgewählt. Bei anderer Probierreihenfolge hätte sich auch die Folge (0, 0, 1, 0, 1) aus Ψ_6 ergeben können.
- Ähnlich wurde bei der Bestimmung des Anführers $\underline{e}_7$ = (0, 1, 1, 0, 0) der Nebenklasse Ψ_7 vorgegangen, die durch das einheitliche Syndrom $\underline{s}_7$ = (1, 1, 1) gekennzeichnet ist. Auch in der Klasse Ψ_7 gibt es eine weitere Folge mit Hamming–Gewicht $w_{\rm H}(\underline{e})$ = 2, nämlich (1, 0, 0, 0, 1).

Die Beschreibung wird auf der nächsten Seite fortgesetzt.

Verallgemeinerung der Syndromdecodierung (3)

Das Beispiel der letzten Seite wird fortgesetzt. Beachten Sie bitte bei der Tabelle, dass der Index μ nicht identisch ist mit dem Binärwert von s_μ. Die Reihenfolge ergibt sich vielmehr durch die Anzahl der Einsen im Nebenklassenanführer e_μ. So ergibt sich beispielsweise das Syndrom s_5 zu (1, 1, 0) und das Syndrom $s_6 = (1, 0, 1)$.

Index μ	Syndrom $\underline{s}_\mu$	Nebenklasse Ψ_μ inklusive „Anführer“ $\underline{e}_\mu$			
0	(0, 0, 0)	(0, 0, 0, 0, 0)	(1, 0, 1, 1, 0)	(0, 1, 0, 1, 1)	(1, 1, 1, 0, 1)
1	(0, 0, 1)	(0, 0, 0, 0, 1)	(1, 0, 1, 1, 1)	(0, 1, 0, 1, 0)	(1, 1, 1, 0, 0)
2	(0, 1, 0)	(0, 0, 0, 1, 0)	(1, 0, 1, 0, 0)	(0, 1, 0, 0, 1)	(1, 1, 1, 1, 1)
3	(1, 0, 0)	(0, 0, 1, 0, 0)	(1, 0, 0, 1, 0)	(0, 1, 1, 1, 1)	(1, 1, 0, 0, 1)
4	(0, 1, 1)	(0, 1, 0, 0, 0)	(1, 1, 1, 1, 0)	(0, 0, 0, 1, 1)	(1, 0, 1, 0, 1)
5	(1, 1, 0)	(1, 0, 0, 0, 0)	(0, 0, 1, 1, 0)	(1, 1, 0, 1, 1)	(0, 1, 1, 0, 1)
6	(1, 0, 1)	(1, 1, 0, 0, 0)	(0, 1, 1, 1, 0)	(1, 0, 0, 1, 1)	(0, 0, 1, 0, 1)
7	(1, 1, 1)	(0, 1, 1, 0, 0)	(1, 1, 0, 1, 0)	(0, 0, 1, 1, 1)	(1, 0, 0, 0, 1)

Die obige Tabelle muss nur einmal erstellt und kann beliebig oft genutzt werden. Lautet beispielsweise der Empfangsvektor $\underline{y} = (0, 1, 0, 0, 1)$, so muss zunächst das Syndrom ermittelt werden:

$$\underline{s} = \underline{y} \cdot \mathbf{H}^{\rm T} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} = \underline{s}_2 .$$

Mit dem Nebenklassenanführer $\underline{e}_2 = (0, 0, 0, 1, 0)$ aus obiger Tabelle (roter Eintrag für Index 2) gelangt man schließlich zum Decodierergebnis:

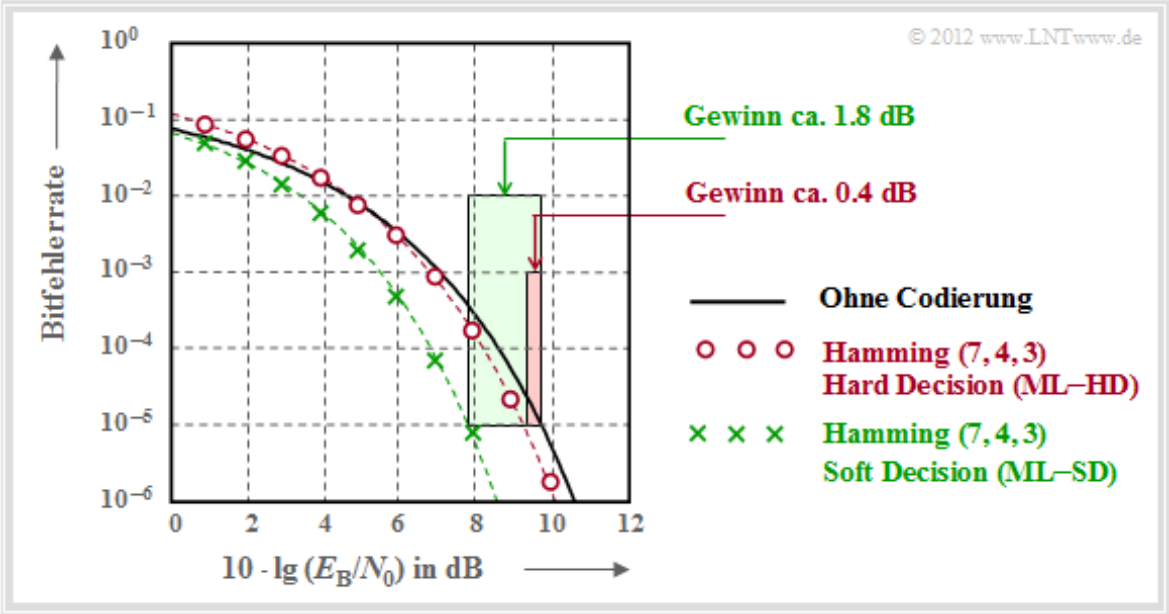
$$\underline{z} = \underline{y} + \underline{e}_2 = (0, 1, 0, 0, 1) + (0, 0, 0, 1, 0) = (0, 1, 0, 1, 1) .$$

Aus dieser Kurzzusammenfassung geht schon hervor, dass die Syndromdecodierung mit einem enormen Aufwand verbunden ist, wenn man nicht wie bei zyklischen Codes gewisse Eigenschaften nutzen kann. Bei großen Blockcodelängen versagt diese Methode vollständig. So müsste man zur Decodierung eines **BCH–Codes** – die Abkürzung steht für deren Erfinder Bose, Chaudhuri und Hocquenghem – mit den Codeparametern $n = 511$, $k = 259$ und $d_{\rm min} = 61$ genau $2^{511-259} \approx 10^{76}$ Fehlermuster der Länge 511 auswerten und abspeichern. Für diese Codes und auch für andere Codes großer Blocklänge gibt es aber spezielle Decodieralgorithmen, die mit weniger Aufwand zum Erfolg führen.

Codiergewinn – Bitfehlerrate bei AWGN

Wir betrachten nun die **Bitfehlerrate** (englisch: *Bit Error Rate*, BER) für folgende Konstellationen:

- Hamming–Code (7, 4, 3),
- AWGN–Kanal, gekennzeichnet durch den Quotienten E_B/N_0 (in dB),
- Maximum–Likelihood–Detektion (ML) mit *Hard Decision* bzw. *Soft Decision*.



Zu dieser Grafik ist anzumerken:

- Die schwarze Vergleichskurve gilt beispielsweise für die binäre Phasenmodulation (BPSK) ohne Codierung. Hierfür benötigt man für die Bitfehlerrate 10^{-5} etwa $10 \cdot \lg E_B/N_0 \approx 9.6$ dB.
- Die roten Kreise gelten für den (7, 4, 3)–Code und harte Entscheidungen des ML–Decoders (ML–HD). Die Syndromdecodierung ist hierfür eine mögliche Realisierungsform.
- Diese Systemkonfiguration bringt erst für $10 \cdot \lg E_B/N_0 > 6$ dB eine Verbesserung gegenüber dem Vergleichssystem. Für BER = 10^{-5} benötigt man nur noch $10 \cdot \lg E_B/N_0 \approx 9.2$ dB.
- Die grünen Kreuze für den Hamming–Code mit **Soft–Decision** (ML–SD) liegen im gesamten Bereich unterhalb der Vergleichskurve. Für BER = 10^{-5} ergibt sich $10 \cdot \lg (E_B/N_0) \approx 7.8$ dB.

Definition: Als **Codiergewinn** einer Systemkonfiguration (gekennzeichnet durch seinen Code und die Art der Decodierung) bezeichnen wir das gegenüber dem Vergleichssystem (ohne Codierung) kleinere $10 \cdot \lg (E_B/N_0)$, das für eine vorgegebene Bitfehlerrate (BER) erforderlich ist:

$$G_{\rm Code}({\rm System}\,|\,{\rm BER}) =$$
$$= 10 \cdot \lg E_B/N_0\ ({\rm ohne\ Codierung}\,|\,{\rm BER}) - 10 \cdot \lg E_B/N_0\ ({\rm System}\,|\,{\rm BER}).$$

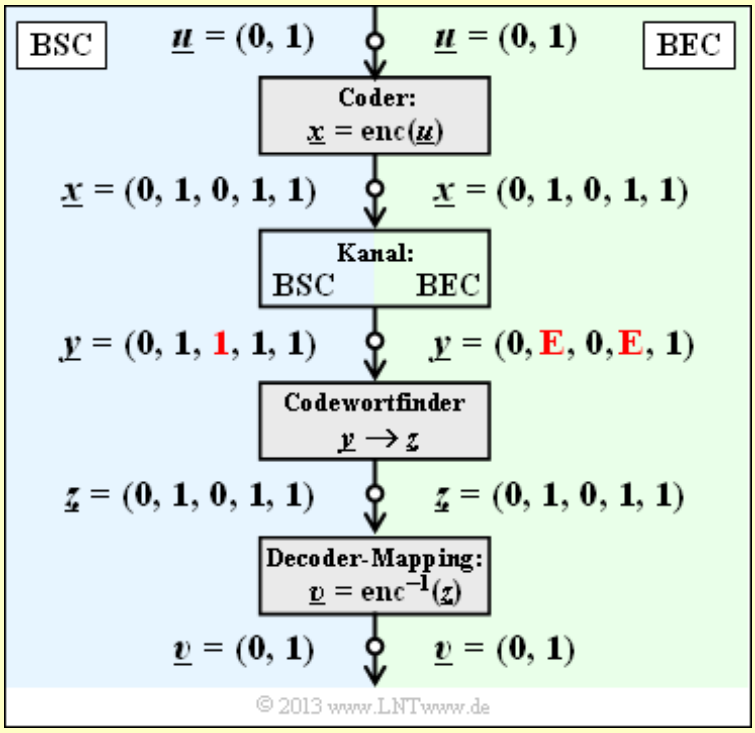
Angewendet auf obige Grafik erhält man:

$$G_{\rm Code}({\rm Hamming\ (7,4,3),\ ML-HD}\,|\,{\rm BER} = 10^{-5}) = 0.4\,{\rm dB},$$
$$G_{\rm Code}({\rm Hamming\ (7,4,3),\ ML-SD}\,|\,{\rm BER} = 10^{-5}) = 1.8\,{\rm dB}.$$

Decodierung beim Binary Erasure Channel (1)

Abschließend soll noch gezeigt werden, in wie weit der Decoder zu modifizieren ist, wenn anstelle des **BSC–Modells** *(Binary Symmetrie Channel)* das **BEC–Kanalmodell** *(Binary Erasure Channel)* zum Einsatz kommt, der keine Fehler produziert, sondern unsichere Bit als Auslöschungen markiert.

Beispiel: Nebenstehende Grafik zeigt das Systemmodell und gibt beispielhafte Werte für die einzelnen Vektoren wider. Der linke Bildteil (blau hinterlegt) gilt für das BSC–Modell (ein Bitfehler 0 → 1) und der rechte (grün hinterlegt) für das BEC–Modell (zwei *Erasures* 1 → E).



Ohne Einschränkung der Allgemeingültigkeit betrachten wir wie im **Beispiel auf Seite 3c** dieses Kapitels wieder den systematischen (5, 2, 3)–Blockcode mit den vier Codeworten

$$\mathcal{C} = \{(0,0,0,0,0)\,,\ (0,1,0,1,1)\,,\ (1,0,1,1,0)\,,\ (1,1,1,0,1)\}\,.$$

Man erkennt:

- Bei BSC kann wegen $d_{\rm min} = 3$ nur ein Bitfehler ($t = 1$) korrigiert werden (rot markiert). Beschränkt man sich auf Fehlererkennung, so funktioniert diese bis zu $e = d_{\rm min} - 1 = 2$ Bitfehler.
- Bei BEC macht Fehlererkennung keinen Sinn, denn bereits der Kanal lokalisiert ein unsicheres Bit als *Erasure* E (Auslöschung). Die Nullen und Einsen im BEC–Empfangswort $\underline{y}$ sind sicher. Deshalb funktioniert hier die Fehlerkorrektur bis zu $e = 2$ Auslöschungen mit Sicherheit.
- Auch $e = 3$ Auslöschungen sind manchmal noch korrigierbar. So kann $\underline{y}$ = (E, E, E, 1, 1) zu $\underline{z}$ = (0, 1, 0, 1, 1) korrigiert werden, da kein zweites Codewort mit zwei Einsen endet. Dagegen ist $\underline{y}$ = (0, E, 0, E, E) aufgrund des im Code erlaubten Nullwortes nicht korrigierbar.
- Wird sichergestellt, dass in keinem Empfangswort mehr als zwei Auslöschungen vorkommen, so ist die BEC–Blockfehlerwahrscheinlichkeit $\Pr(\underline{z} \neq \underline{x}) = \Pr(\underline{v} \neq \underline{u})$ identisch 0. Dagegen ist die entsprechende Blockfehlerwahrscheinlichkeit beim BSC–Modell stets größer als 0.

Da nach dem BEC ein jedes Empfangswort entweder richtig oder gar nicht decodiert wird, nennen wir hier den Block $y \rightarrow z$ zukünftig „Codewortfinder”. Eine „Schätzung” findet nur beim BSC–Modell statt.

Decodierung beim Binary Erasure Channel (2)

Wie funktioniert aber nun die Decodierung eines Empfangswortes $\underline{y}$ mit Auslöschungen algorithmisch? Ausgehend vom Empfangswort $\underline{y}$ = (0, E, 0, E, 1) des letzten Beispiels setzen wir den Ausgang des Codewortfinders formal zu $\underline{z} = (0, z_2, 0, z_4, 1)$, wobei die Symbole z_2 und z_4 (jeweils 0 oder 1) entsprechend der folgenden Gleichung zu bestimmen sind:

$$\underline{z} \cdot \mathbf{H}^{\rm T} = \underline{0} \quad \Rightarrow \quad \mathbf{H} \cdot \underline{z}^{\rm T} = \underline{0}^{\rm T}.$$

Es geht nun darum, diese Bestimmungsgleichung möglichst effizient umzusetzen.

Beispiel: Wir betrachten wieder den auf der **letzten Seite** skizzierten Fall mit dem Empfangsvektor $\underline{y}$ = (0, E, 0, E, 1). Damit ergeben sich folgende Rechenschritte:

- Mit der Prüfmatrix **H** des (5, 2, 3)–Blockcodes und dem Vektor $\underline{z} = (0, z_2, 0, z_4, 1)$ lautet die obige Bestimmungsgleichung:

$$\mathbf{H} \cdot \underline{z}^{\rm T} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ z_2 \\ 0 \\ z_4 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

- Wir fassen die sicheren (korrekten) Bit zum Vektor $\underline{z}_{\rm K}$ zusammen und die ausgelöschten Bit zum Vektor $\underline{z}_{\rm E}$. Dann teilen wir die Prüfmatrix **H** in die entsprechenden Teilmatrizen $\mathbf{H}_{\rm K}$ und $\mathbf{H}_{\rm E}$ auf:

$$\underline{z}_{\rm K} = (0, 0, 1), \quad \mathbf{H}_{\rm K} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \Rightarrow \quad \text{Spalten 1, 3 und 5 der Prüfmatrix},$$

$$\underline{z}_{\rm E} = (z_2, z_4), \quad \mathbf{H}_{\rm E} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \Rightarrow \quad \text{Spalten 2 und 4 der Prüfmatrix}.$$

- Unter Berücksichtigung der Tatsache, dass in GF(2) die Subtraktion gleich der Addition ist, lässt sich die obige Gleichung wie folgt darstellen:

$$\mathbf{H}_{\rm K} \cdot \underline{z}_{\rm K}^{\rm T} + \mathbf{H}_{\rm E} \cdot \underline{z}_{\rm E}^{\rm T} = \underline{0}^{\rm T} \quad \Rightarrow \quad \mathbf{H}_{\rm E} \cdot \underline{z}_{\rm E}^{\rm T} = \mathbf{H}_{\rm K} \cdot \underline{z}_{\rm K}^{\rm T}$$

$$\Rightarrow \quad \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} z_2 \\ z_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

- Dies führt zu einem linearen Gleichungssystem mit zwei Gleichungen für die beiden unbekannten z_2 und z_4 (jeweils 0 oder 1). Aus der letzten Zeile erhält man $z_2 = 1$ und aus der zweiten Zeile folgt somit $z_2 + z_4 = 0 \Rightarrow z_4 = 1$. Damit ergibt sich das zulässige Codewort $\underline{z}$ = (0, 1, 0, 1, 1).

Distanzspektrum eines linearen Codes (1)

Wir gehen weiterhin von einem linearen und binären (n, k)–Blockcode C aus. Ein wesentliches Ziel des Codedesigns ist es, die **Blockfehlerwahrscheinlichkeit** $\Pr(\underline{v} \neq \underline{u}) = \Pr(\underline{z} \neq \underline{x})$ möglichst gering zu halten. Dies erreicht man unter anderem dadurch, dass

- die minimale Distanz $d_{\rm min}$ zwischen zwei Codeworten $\underline{x}$ und $\underline{x}'$ möglichst groß ist, so dass man bis zu $t = \lfloor (d_{\rm min}-1)/2 \rfloor$ Bitfehler richtig korrigieren kann;
- gleichzeitig die minimale Distanz $d_{\rm min}$ $\Rightarrow$ *worst–case* möglichst selten auftritt, wenn man alle zulässigen Codeworte berücksichtigt.

Definition: Wir benennen die **Anzahl der Codeworte** $\underline{x}' \in C$ mit der Hamming–Distanz i vom betrachteten Codewort $\underline{x}$ des gleichen Codes C mit $W_i(\underline{x})$, wobei gilt:

$$W_i(\underline{x}) = \left| \{ \underline{x}, \underline{x}' \in \mathcal{C} \mid d_{\rm H}(\underline{x}, \underline{x}') = i \} \right| .$$

Man bezeichnet diesen Wert auch als **Vielfachheit** (englisch: *Multiplicity*).

Die Betragsstriche kennzeichnen hierbei die Anzahl der Codeworte $\underline{x}'$, die die Bedingung { ... } erfüllen.

Beispiel: Wir betrachten den (5, 2)–Blockcode C mit der Generatormatrix

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

Die nachfolgende Tabelle zeigt die Hamming–Distanzen zwischen allen Codeworten $\underline{x}$ mit den vier möglichen Bezugsworten $\underline{x}_0$, ... , $\underline{x}_3$.

$\underline{x} = (0, 0, 0, 0, 0) = \underline{x}_0$	$\underline{x} = (0, 1, 0, 1, 1) = \underline{x}_1$
$\underline{x}' = (0, 0, 0, 0, 0) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 0$ $\underline{x}' = (0, 1, 0, 1, 1) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 3$ $\underline{x}' = (1, 0, 1, 1, 0) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 3$ $\underline{x}' = (1, 1, 1, 0, 1) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 4$	$\underline{x}' = (0, 0, 0, 0, 0) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 3$ $\underline{x}' = (0, 1, 0, 1, 1) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 0$ $\underline{x}' = (1, 0, 1, 1, 0) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 4$ $\underline{x}' = (1, 1, 1, 0, 1) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 3$
$\underline{x} = (1, 0, 1, 1, 0) = \underline{x}_2$	$\underline{x} = (1, 1, 1, 0, 1) = \underline{x}_3$
$\underline{x}' = (0, 0, 0, 0, 0) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 3$ $\underline{x}' = (0, 1, 0, 1, 1) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 4$ $\underline{x}' = (1, 0, 1, 1, 0) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 0$ $\underline{x}' = (1, 1, 1, 0, 1) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 3$	$\underline{x}' = (0, 0, 0, 0, 0) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 4$ $\underline{x}' = (0, 1, 0, 1, 1) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 3$ $\underline{x}' = (1, 0, 1, 1, 0) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 3$ $\underline{x}' = (1, 1, 1, 0, 1) \Rightarrow d_{\rm H}(\underline{x}, \underline{x}') = 0$

Man erkennt, dass unabhängig vom Bezugswort $\underline{x}_i$ gilt:

$$W_0 = 1, \quad W_1 = W_2 = 0, \quad W_3 = 2, \quad W_4 = 1 \quad \Rightarrow \quad d_{\rm min} = 3.$$

Distanzspektrum eines linearen Codes (2)

Nicht nur im letzten Beispiel, sondern bei jedem linearen Code ergeben sich für jedes Codewort die gleichen Vielfachheiten W_i. Da zudem das Nullwort $\underline{0}$ = (0, 0, ..., 0) Bestandteil eines jeden linearen Binärcodes ist, lässt sich die Definition der letzten Seite auch wie folgt formulieren:

Definition: Das **Distanzspektrum** eines linearen binären (n, k)–Blockcodes ist die Menge $\{W_i\}$ mit $i = 0, 1, ..., n$. Hierbei gibt W_i die Anzahl der Codeworte $\underline{x} \in C$ mit Hamming–Gewicht $w_H(\underline{x}) = i$ an.

Oft beschreibt man die Menge $\{W_i\}$ auch als Polynom mit einer Pseudovariablen X:

$$\{W_i\} \quad \Leftrightarrow \quad W(X) = \sum_{i=0}^{n} W_i \cdot Xi^i = W_0 + W_1 \cdot X + W_2 \cdot X^2 + ... + W_n \cdot X^n .$$

Man bezeichnet $W(X)$ auch als **Gewichtsfunktion** (englisch: *Weight Enumerator Function*, WEF).

Beispielsweise lautet die Gewichtsfunktion des (5, 2)–Codes

$$\mathcal{C} = \{ (0,0,0,0,0),\ (0,1,0,1,1),\ (1,0,1,1,0),\ (1,1,1,0,1) \}$$

vom Beispiel auf der letzten Seite

$$W(X) = 1 + 2 \cdot X^3 + X^4 .$$

Wie aus der **Tabelle seiner Codeworte** hervorgeht, erhält man für den (7, 4, 3)–Hamming–Code:

$$W(X) = 1 + 7 \cdot X^3 + 7 \cdot X^4 + X^7 .$$

Die Überführung des Distanzspektrums $\{W_i\}$ in die Gewichtsfunktion $W(X)$ bietet zudem bei manchen Aufgabenstellungen große numerische Vorteile. Ist beispielsweise die *Weight Enumerator Function* $W(X)$ eines (n, k)–Blockcodes bekannt, so gilt für die WEF des hierzu dualen $(n, n–k)$–Codes C_{Dual}:

$$W_{Dual}(X) = \frac{(1+X)^n}{2^k} \cdot W\left(\frac{1-X}{1+X}\right) .$$

Beispiel: Gesucht ist die Gewichtsfunktion $W(X)$ des **Single Parity–check Codes** mit $n = 6$, $k = 5$ ⇒ **SPC (6, 5)**. Man erhält diese durch Vergleich aller $2^5 = 32$ Codeworte mit dem Nullwort:

$$W_{SPC(6,5)}(X) = 1 + 15 \cdot X^2 + 15 \cdot X^4 + X^6 .$$

Unter Berücksichtigung obiger Gleichung kommt man sehr viel schneller zum gleichen Ergebnis:

- Der zu SPC (6, 5) duale Code ist der **Repetition Code** RC (6, 1) mit nur zwei Codeworten (0, 0, 0, 0, 0, 0) und (1, 1, 1, 1, 1, 1), die sich in keiner bzw. allen Bitpositionen unterscheiden:

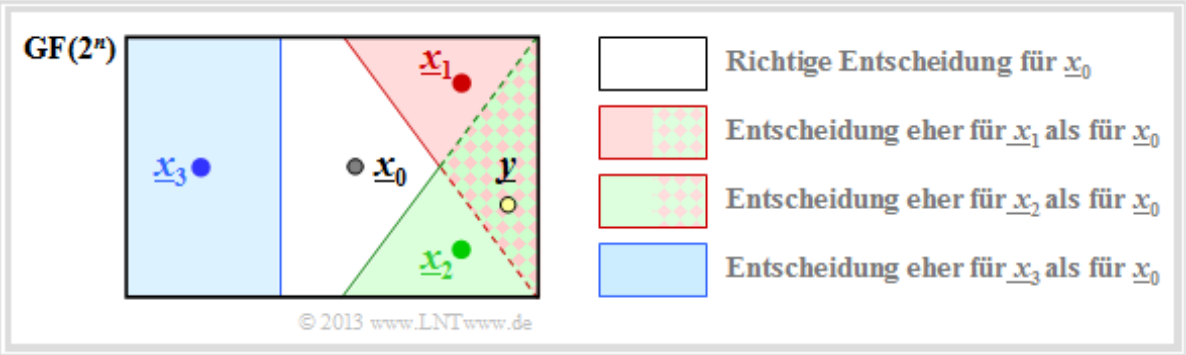
  $$W_{RC(6,1)}(X) = 1 + X^6 .$$

- Daraus folgt für die Gewichtsfunktion des SPC (6, 5) nach obiger Gleichung mit $k = 1$:

$$\begin{aligned} W_{SPC(6,5)}(X) &= \frac{(1+X)^6}{2^1} \cdot W\left[1 + ((1-X)/(1+X))^6\right] = \\ &= 1/2 \cdot \left[(1+X)^6 + (1-X)^6\right] = 1 + 15 \cdot X^2 + 15 \cdot X^4 + X^6 . \end{aligned}$$

Union Bound der Blockfehlerwahrscheinlichkeit

Wir betrachten wie im **Beispiel zum Distanzspektrum** den (5, 2)–Blockcode $C = \{\underline{x}_0, \underline{x}_1, \underline{x}_2, \underline{x}_3\}$ und setzen voraus, dass das Codewort $\underline{x}_0$ gesendet wurde. Die Grafik verdeutlicht den Sachverhalt).



Im fehlerfreien Fall würde dann der Codewortschätzer $\underline{z} = \underline{x}_0$ liefern. Andernfalls käme es zu einem Blockfehler (das heißt $\underline{z} \neq \underline{x}$ und dementsprechend $\underline{v} \neq \underline{u}$) mit der Wahrscheinlichkeit

$$\Pr(\text{Blockfehler}) = \Pr\left([\underline{x}_0 \mapsto \underline{x}_1] \cup [\underline{x}_0 \mapsto \underline{x}_2] \cup [\underline{x}_0 \mapsto \underline{x}_3]\right).$$

Das Ereignis „Verfälschung von $\underline{x}_0$ nach $\underline{x}_1$" tritt für ein gegebenes Empfangswort $\underline{y}$ entsprechend der ML–Entscheidungsregel genau dann ein, wenn für die bedingte Wahrscheinlichkeitsdichtefunktion gilt:

$$[\underline{x}_0 \mapsto \underline{x}_1] \quad \Longleftrightarrow \quad f(\underline{x}_0|\underline{y}) < f(\underline{x}_1|\underline{y}).$$

Da $[\underline{x}_0 \to \underline{x}_1]$, $[\underline{x}_0 \to \underline{x}_2]$, $[\underline{x}_0 \to \underline{x}_3]$ nicht notwendigerweise *disjunkte Ereignisse* sind (die sich somit gegenseitig ausschließen würden), ist die **Wahrscheinlichkeit der Vereinigungsmenge** kleiner oder gleich der Summe der Einzelwahrscheinlichkeiten:

$$\Pr(\text{Blockfehler}) \leq \Pr[\underline{x}_0 \mapsto \underline{x}_1] + \Pr[\underline{x}_0 \mapsto \underline{x}_2] + \Pr[\underline{x}_0 \mapsto \underline{x}_3].$$

Man nennt diese obere Schranke für die (Block–)Fehlerwahrscheinlichkeit die **Union Bound**. Diese wurde bereits im **Kapitel 4.3** des Buches „Digitalsignalübertragung" verwendet.

Verallgemeinern und formalisieren wir diese Ergebnisse (sowohl $\underline{x}$ als auch $\underline{x}'$ gehören zum Code C):

Blockfehlerwahrscheinlichkeit :

$$\Pr(\text{Blockfehler}) = \Pr\left(\bigcup_{\underline{x}' \neq \underline{x}} [\underline{x} \mapsto \underline{x}']\right),$$

Obere Schranke nach der **Union Bound**:

$$\Pr(\text{Union Bound}) \leq \sum_{\underline{x}' \neq \underline{x}} \Pr[\underline{x} \mapsto \underline{x}'],$$

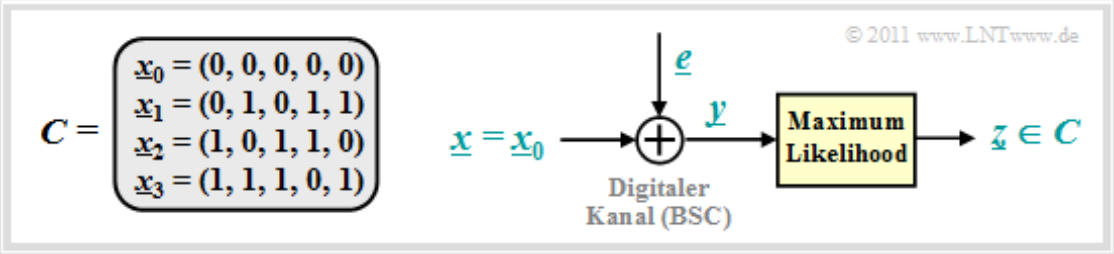
Paarweise Fehlerwahrscheinlichkeit (nach dem MAP– bzw. ML–Kriterium):

$$\Pr[\underline{x} \mapsto \underline{x}'] = \Pr\left[f(\underline{x}|\underline{y}) \leq f(\underline{x}'|\underline{y})\right].$$

Auf den nächsten Seiten werden diese Ergebnisse auf verschiedene Kanäle angewendet.

Union Bound für das BSC–Modell

Wir betrachten weiterhin den **beispielhaften (5, 2)–Code:**



Für den digitalen Kanal verwenden wir das **BSC–Modell** (*Binary Symmetric Channel*):

$$\Pr(y=1\,|\,x=0) = \Pr(y=0\,|\,x=1) = \Pr(e=1) = \varepsilon,$$
$$\Pr(y=0\,|\,x=0) = \Pr(y=1\,|\,x=1) = \Pr(e=0) = 1-\varepsilon.$$

Die beiden Codeworte $\underline{x}_0$ = (0, 0, 0, 0, 0) und $\underline{x}_1$ = (0, 1, 0, 1, 1) unterscheiden sich in genau d = 3 Bitpositionen, wobei d die Hamming–Distanz zwischen $\underline{x}_0$ und $\underline{x}_1$ angibt. Ein falsches Decodierergebnis $[\underline{x}_0 \rightarrow \underline{x}_1]$ erhält man immer dann, wenn mindestens zwei der drei Bit an den Bitpositionen 2, 4 und 5 verfälscht werden. Die Bitpositionen 1 und 3 spielen hier dagegen keine Rolle, da diese für $\underline{x}_0$ und $\underline{x}_1$ gleich sind. Da der betrachtete Code $t = \lfloor (d–1)/2 \rfloor = 1$ Fehler korrigieren kann, gilt:

$$\Pr[\underline{x}_0 \mapsto \underline{x}_1] = \sum_{i=t+1}^{d} \binom{d}{i} \cdot \varepsilon^i \cdot (1-\varepsilon)^{d-i} = \binom{3}{2} \cdot \varepsilon^2 \cdot (1-\varepsilon) + \binom{3}{3} \cdot \varepsilon^3 =$$
$$= 3 \cdot \varepsilon^2 \cdot (1-\varepsilon) + \varepsilon^3 = \Pr[\underline{x}_0 \mapsto \underline{x}_2].$$

Die Codeworte $\underline{x}_0$ und $\underline{x}_3$ unterscheiden sich in vier Bitpositionen. Zu einer falschen Decodierung des Blocks kommt es deshalb mit Sicherheit, wenn vier oder drei Bit verfälscht werden. Eine Verfälschung von zwei Bit hat mit 50–prozentiger Wahrscheinlichkeit ebenfalls einen Blockfehler zur Folge, wenn man hierfür eine Zufallsentscheidung voraussetzt:

$$\Pr[\underline{x}_0 \mapsto \underline{x}_3] = \varepsilon^4 + 4 \cdot \varepsilon^3 \cdot (1-\varepsilon) + 1/2 \cdot 6 \cdot \varepsilon^2 \cdot (1-\varepsilon)^2.$$

Daraus ergibt sich für die „Union Bound”:

$$\Pr(\text{Union Bound}) = \Pr[\underline{x}_0 \mapsto \underline{x}_1] + \Pr[\underline{x}_0 \mapsto \underline{x}_2] + \Pr[\underline{x}_0 \mapsto \underline{x}_3] \geq \Pr(\text{Blockfehler}).$$

In der Tabelle sind die Ergebnisse für verschiedene Werte des BSC–Parameters ε zusammengefasst:

ε	$\Pr[\underline{x}_0 \rightarrow \underline{x}_1]$	$\Pr[\underline{x}_0 \rightarrow \underline{x}_2]$	$\Pr[\underline{x}_0 \rightarrow \underline{x}_3]$	Union Bound
10^{-1}	$2.8 \cdot 10^{-2}$	$2.8 \cdot 10^{-2}$	$2.8 \cdot 10^{-2}$	$8.4 \cdot 10^{-2}$
10^{-2}	$2.98 \cdot 10^{-4}$	$2.98 \cdot 10^{-4}$	$2.98 \cdot 10^{-4}$	$8.94 \cdot 10^{-4}$
10^{-3}	$2.998 \cdot 10^{-6}$	$2.998 \cdot 10^{-6}$	$2.998 \cdot 10^{-6}$	$8.994 \cdot 10^{-6}$
10^{-4}	$2.9998 \cdot 10^{-8}$	$2.9998 \cdot 10^{-8}$	$2.9998 \cdot 10^{-8}$	$8.9994 \cdot 10^{-8}$

Zu erwähnen ist, dass die völlig unterschiedlich zu berechnenden Wahrscheinlichkeiten $\Pr(\underline{x}_0 \rightarrow \underline{x}_1)$ und $\Pr(\underline{x}_0 \rightarrow \underline{x}_3)$ genau das gleiche Ergebnis liefern.

Die obere Schranke nach Bhattacharyya (1)

Eine weitere obere Schranke für die Blockfehlerwahrscheinlichkeit wurde von Bhattacharyya angegeben:

$$\Pr(\text{Blockfehler}) \le W(X = \beta) - 1 = \Pr(\text{Bhattacharyya}).$$

Hierzu ist anzumerken:

- $W(X)$ ist die zu Beginn dieses Kapitels 1.6 definierte **Gewichtsfunktion,** die den verwendeten Kanalcode charakterisiert.
- Der *Bhattacharyya–Parameter* β kennzeichnet den digitalen Kanal. Beispielsweise gilt:

$$\beta = \begin{cases} \lambda & \text{für das BEC – Modell,} \\ \sqrt{4 \cdot \varepsilon \cdot (1-\varepsilon)} & \text{für das BSC – Modell,} \\ \exp[-R \cdot E_{\rm B}/N_0] & \text{für das AWGN – Modell.} \end{cases}$$

- Die Bhattacharyya–Schranke liegt stets (und meist deutlich) oberhalb der Kurve für die „Union Bound”. Mit dem Ziel, eine für alle Kanäle einheitliche Schranke zu finden, müssen hier sehr viel gröbere Abschätzungen vorgenommen werden als für die Herleitung der Union Bound.

Bhattacharyya–Schranke für das BSC–Modell

Für die paarweise Verfälschungswahrscheinlichkeit des BSC–Modells wurde vorne hergeleitet:

$$\Pr[\underline{x}_0 \mapsto \underline{x}_1] = \sum_{i=\lfloor (d-1)/2 \rfloor}^{d} \binom{d}{i} \cdot \varepsilon^i \cdot (1-\varepsilon)^{d-i} = \sum_{i=\lceil d/2 \rceil}^{d} \binom{d}{i} \cdot \varepsilon^i \cdot (1-\varepsilon)^{d-i}.$$

Hierbei kennzeichnet $\varepsilon = \Pr(y = 1 |\, x = 0) = \Pr(y = 0 |\, x = 1) < 0.5$ das BSC–Modell und $d = d_{\rm H}(\underline{x}_0, \underline{x}_1)$ gibt die Hamming–Distanz der betrachteten Codeworte an.

Um zur Bhattacharyya–Schranke zu kommen, müssen folgende Abschätzungen getroffen werden:

- Für alle $i < d$ gilt $\varepsilon^i \cdot (1-\varepsilon)^{d-i} \le [\varepsilon \cdot (1-\varepsilon)]^{d/2}$:

$$\Pr[\underline{x}_0 \mapsto \underline{x}_1] \le [\varepsilon \cdot (1-\varepsilon)]^{d/2} \cdot \sum_{i=\lceil d/2 \rceil}^{d} \binom{d}{i}.$$

- Änderung bezüglich der unteren Grenze der Laufvariablen i:

$$\sum_{i=\lceil d/2 \rceil}^{d} \binom{d}{i} < \sum_{i=0}^{d} \binom{d}{i} = 2^d, \quad \beta = 2 \cdot \sqrt{\varepsilon \cdot (1-\varepsilon)} \quad \Rightarrow \quad \Pr[\underline{x}_0 \mapsto \underline{x}_1] = \beta^d.$$

- Umsortierung gemäß den Hamming–Gewichten W_i (Hamming–Distanz $d = i$ kommt W_i mal vor):

$$\Pr(\text{Blockfehler}) \le \sum_{i=1}^{n} W_i \cdot \beta^i = 1 + W_1 \cdot \beta + W_2 \cdot \beta^2 + \ldots + W_n \cdot \beta^n.$$

- Mit der Gewichtsfunktion $W(X) = 1 + W_1 \cdot X + W_2 \cdot X^2 + \ldots + W_n \cdot X^n$:

$$\Pr(\text{Blockfehler}) \le W(X = \beta) - 1 = \Pr(\text{Bhattacharyya}).$$

Die obere Schranke nach Bhattacharyya (2)

In der Tabelle ist die Bhattacharyya–Schranke für verschiedene BSC–Parameter ε abgegeben, gültig für den **beispielhaften (5, 2)–Code.** Für diesen gilt:

$$W_0 = 1,\ W_1 = W_2 = 0,\ W_3 = 2,\ W_4 = 1 \quad \Rightarrow \quad W(X) = 1 + 2 \cdot X^3 + X^4$$

$$\Rightarrow \quad \Pr(\text{Blockfehler}) \le W(\beta) - 1 = 2 \cdot \beta^3 + \beta^4 = \Pr(\text{Bhattacharyya}).$$

ε	β	$2\beta^3$	β^4	Battacharyya	Union Bound
10^{-1}	0.600	$4.32 \cdot 10^{-1}$	$1.30 \cdot 10^{-1}$	$5.62 \cdot 10^{-1}$	$8.40 \cdot 10^{-2}$
10^{-2}	0.199	$1.58 \cdot 10^{-2}$	$1.57 \cdot 10^{-3}$	$1.74 \cdot 10^{-2}$	$8.94 \cdot 10^{-4}$
10^{-3}	0.063	$5.00 \cdot 10^{-4}$	$1.58 \cdot 10^{-5}$	$5.16 \cdot 10^{-4}$	$9.00 \cdot 10^{-6}$
10^{-4}	0.020	$1.60 \cdot 10^{-5}$	$1.60 \cdot 10^{-7}$	$1.62 \cdot 10^{-5}$	$9.00 \cdot 10^{-8}$

Basierend auf diesem Beispiel für den einfachen (5, 2)–Code, der allerdings wenig praxisrelevant ist, sowie dem Beispiel auf der nächsten Seite für den (7, 4, 3)–Hamming–Code kann man zusammenfassen:

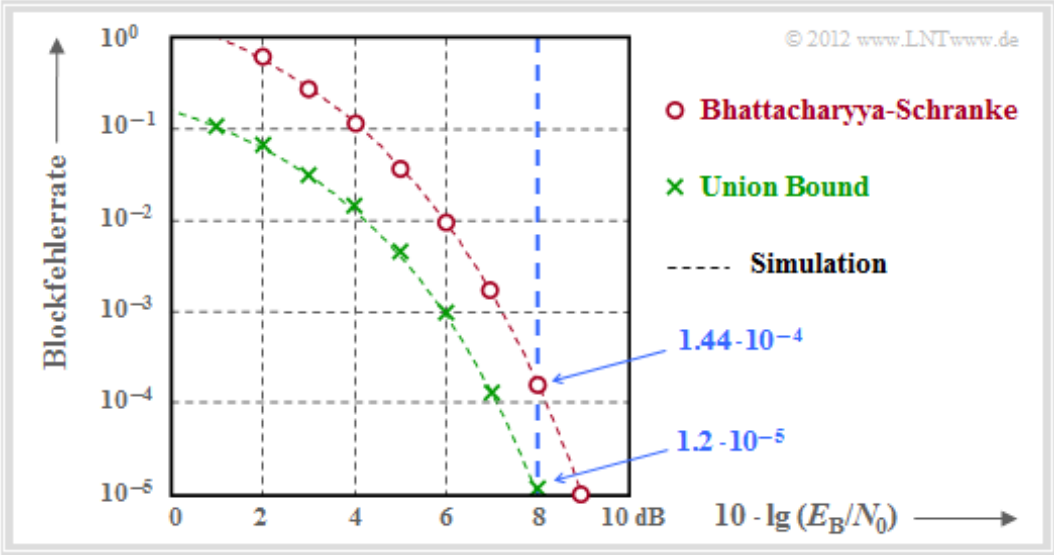
- Die Blockfehlerwahrscheinlichkeit eines Codiersystems ist oft analytisch nicht angebbar und muss per Simulation ermittelt werden. Gleiches gilt für die Bitfehlerwahrscheinlichkeit.
- Die *Union Bound* liefert eine obere Schranke für die Blockfehlerwahrscheinlichkeit. Bei vielen Anwendungen (insbesondere bei kurzen Codes) liegt sie nur geringfügig über dieser.
- Die Bhattacharyya–Schranke liegt beim BEC–Kanal etwa um den Faktor 2 oberhalb der *Union Bound* – siehe **Aufgabe A1.14.** Beim BSC– und beim AWGN–Kanal ist der Abstand zwischen beiden Schranken deutlich größer. Der Faktor 10 (oder mehr) ist keine Seltenheit.
- Die Bhattacharyya–Schranke $W(\beta) - 1$ wirkt auf den ersten Blick sehr einfach. Es sind einige Vereinfachungsschritte erforderlich, um auf diese Form zu kommen. Trotzdem benötigt man auch hier Kenntnis über die genaue Gewichtsfunktion $W(\xi)$ des Codes.
- Bei Kenntnis des Übertragungskanals (BEC, BSC, AWGN oder Abwandlungen hiervon) und dessen Parameter spricht vom Aufwand her nichts dagegen, die *Union Bound* als obere Schranke für die Blockfehlerwahrscheinlichkeit zu verwenden.

Schranken für den (7, 4, 3)–Hamming–Code beim AWGN–Kanal

Abschließend betrachten wir die Blockfehlerwahrscheinlichkeit und deren Schranken (*Union Bound* und Bhattacharyya–Schranke) für die folgende Konfiguration:

- AWGN–Kanal, gekennzeichnet durch den Quotienten E_B/N_0,
- Hamming–Code (7, 4) $\Rightarrow$ $R = 4/7$, $W(X) - 1 = 7 \cdot X^3 + 7 \cdot X^4 + X^7$,
- *Soft–Decision* nach dem ML–Kriterium.

Die Ergebnisse sind in der folgenden Grafik zusammengefasst. Im Gegensatz zur Grafik im **Kapitel 1.5** ist hier die Blockfehlerrate angegeben und nicht die Bitfehlerrate. Näherungsweise ist Letztere um den Faktor d_{min}/k kleiner, falls wie hier $d_{min} < k$ ist. Im vorliegenden Beispiel gilt $d_{min}/k = 0.75$.



Die folgenden Zahlenwerte gelten für $10 \cdot \lg E_B/N_0 = 8$ dB $\Rightarrow$ $E_B/N_0 = 6.31$ (blaue Markierungen):

- Die grünen Kreuze markieren die *Union Bound*. Nach dieser gilt:

$$\begin{aligned}\Pr(\text{Blockfehler}) \le & \sum_{i=d_{\min}}^{n} W_i \cdot \mathrm{Q}\left(\sqrt{i \cdot 2R \cdot E_B/N_0}\right) = \\ = \; & 7 \cdot \mathrm{Q}(4.65) + 7 \cdot \mathrm{Q}(5.37) + \mathrm{Q}(7.10) = \\ \approx \; & 7 \cdot 1.66 \cdot 10^{-6} + 7 \cdot 3.93 \cdot 10^{-8} + 10^{-9} = 1.2 \cdot 10^{-5}\,.\end{aligned}$$

- Die Zahlenwerte machen deutlich, dass die Union Bound durch den ersten Term bestimmt wird:

$$\Pr(\text{Union Bound}) \approx W_{d_{\min}} \cdot \mathrm{Q}\left(\sqrt{d_{\min} \cdot 2R \cdot E_B/N_0}\right) = 1.16 \cdot 10^{-5}\,.$$

  Allerdings ist diese sog. *Truncated Union Bound* nicht mehr bei allen Anwendungen eine echte Schranke für die Blockfehlerwahrscheinlichkeit, sondern ist eher als Näherung zu verstehen.

- Die *Bhattacharyya–Schranke* ist in der Grafik durch rote Punkte markiert. Diese Schranke liegt aufgrund der stark vereinfachten **Chernoff–Rubin Bound** $\mathrm{Q}(x) \le \exp(-x^2/2)$ deutlich über der Union Bound. Für $10 \cdot \lg E_B/N_0 = 8$ dB erhält man mit $\beta = \exp[-R \cdot E_B/N_0] \approx 0.027$:

$$\Pr(\text{Bhattacharyya}) = W(\beta) - 1 = 7 \cdot \beta^3 + 7 \cdot \beta^4 + \beta^7 \approx 1.44 \cdot 10^{-4}\,.$$

Kanalcodierungstheorem und Kanalkapazität

Wir betrachten weiterhin einen binären Blockcode mit k Informationsbits pro Block und Codeworte der Länge n, woraus sich die Coderate $R = k/n$ mit der Einheit Informationsbit/Codesymbol ergibt.

Der geniale Informationstheoretiker **Claude E. Shannon** hat sich schon 1948 sehr intensiv mit der Korrekturfähigkeit solcher Codes beschäftigt und hierfür für jeden Kanal eine Grenze angegeben, die sich allein aus informationstheoretischen Überlegungen ergibt. Bis heute wurde noch kein Code gefunden, der diese Grenze übersteigt, und dies wird auch so bleiben.

Shannons Kanalcodierungstheorem: Zu jedem Kanal mit der Kanalkapazität $C > 0$ existiert stets (mindestens) ein Code, dessen **Fehlerwahrscheinlichkeit gegen Null** geht, so lange die Coderate kleiner als die Kanalkapazität ist: $\boldsymbol{R < C}$. Voraussetzung hierfür ist allerdings, dass für die Blocklänge dieses Codes gilt: $n \to \infty$.

Die Umkehrung ist ebenfalls zutreffend und besagt:

Umkehrschluss: Ist die Coderate größer als die Kanalkapazität $\Rightarrow \boldsymbol{R > C}$, so kann eine beliebig kleine Fehlerwahrscheinlichkeit **auf keinen Fall** erreicht werden.

Zur Herleitung und Berechnung der Kanalkapazität gehen wir zunächst von einem digitalen Kanal mit M_X möglichen Eingangswerten x_i und M_Y möglichen Ausgangswerten y_j aus. Dann gilt für den mittleren Transinformationsgehalt – kurz die **Transinformation** (englisch: *Mutual Information*) – zwischen der Zufallsgröße x am Kanaleingang und der Zufallsgröße y am Ausgang:

$$I(x;\,y) = \sum_{i=1}^{M_X}\sum_{j=1}^{M_Y} \Pr(x_i, y_j) \cdot \log_2 \frac{\Pr(y_j \,|\, x_i)}{\Pr(y_j)} =$$

$$= \sum_{i=1}^{M_X}\sum_{j=1}^{M_Y} \Pr(y_j \,|\, x_i) \cdot \Pr(x_i) \cdot \log_2 \frac{\Pr(y_j \,|\, x_i)}{\sum_{k=1}^{M_X} \Pr(y_j \,|\, x_k) \cdot \Pr(x_k)}.$$

Beim Übergang von der ersten zur zweiten Gleichung wurden dabei der **Satz von Bayes** sowie der **Satz von der totalen Wahrscheinlichkeit** berücksichtigt. Weiter ist anzumerken, dass hier der *Logarithmus dualis* mit „$\log_2$" bezeichnet ist. Teilweise verwenden wir im LNTwww hierfür auch „ld".

Im Gegensatz zum Buch **Einführung in die Informationstheorie** unterscheiden wir im Folgenden auch nicht zwischen der Zufallsgröße (Großbuchstaben, X bzw. Y) und Realisierungen (Kleinbuchstaben, x bzw. y).

Definition: Die von Shannon eingeführte **Kanalkapazität** gibt die maximale Transinformation $I(x; y)$ zwischen der Eingangsgröße x und der Ausgangsgröße y an:

$$C = \max_{\Pr(x_i)} I(X;Y).$$

Hinzugefügt werden muss die Pseudo–Einheit „bit/Kanalzugriff".

Da die Maximierung der Transinformation über alle möglichen (diskreten) Eingangsverteilungen $\Pr(x_i)$ erfolgen muss, ist die Kanalkapazität unabhängig vom Eingang und damit eine reine Kanalkenngröße.

Kanalkapazität des BSC–Modells (1)

Wenden wir diese Definitionen auf das **BSC–Modell** an:

$$\begin{aligned} I(x;\,y) \;=\; &\Pr(y=0\,|\,x=0)\cdot\Pr(x=0)\cdot\log_2\,\frac{\Pr(y=0\,|\,x=0)}{\Pr(y=0)}\,+\\ &+\;\Pr(y=1\,|\,x=0)\cdot\Pr(x=0)\cdot\log_2\,\frac{\Pr(Y=1\,|\,x=0)}{\Pr(y=1)}\,+\\ &+\;\Pr(y=0\,|\,x=1)\cdot\Pr(x=1)\cdot\log_2\,\frac{\Pr(Y=0\,|\,x=1)}{\Pr(y=0)}\,+\\ &+\;\Pr(y=1\,|\,x=1)\cdot\Pr(x=1)\cdot\log_2\,\frac{\Pr(y=1\,|\,x=1)}{\Pr(y=1)}\,. \end{aligned}$$

Zur Kanalkapazität gelangt man durch folgende Überlegungen:

- Die Maximierung bezüglich der Eingangsverteilung führt auf gleichwahrscheinliche Symbole:

  $$\Pr(x=0)=\Pr(x=1)=1/2\,.$$

- Aufgrund der aus dem Modell erkennbaren Symmetrie gilt dann gleichzeitig:

  $$\Pr(y=0)=\Pr(y=1)=1/2\,.$$

- Berücksichtigt man weiter die BSC–Übergangswahrscheinlichkeiten

  $$\begin{aligned} \Pr(y=1\,|\,x=0) \;&=\; \Pr(y=0\,|\,x=1)=\varepsilon\,,\\ \Pr(y=0\,|\,x=0) \;&=\; \Pr(y=1\,|\,x=1)=1-\varepsilon\,, \end{aligned}$$

  so erhält man nach Zusammenfassen je zweier Terme:

  $$\begin{aligned} C \;&=\; 2\cdot 1/2\cdot\varepsilon\cdot\log_2\,\frac{\varepsilon}{1/2}+2\cdot 1/2\cdot(1-\varepsilon)\cdot\log_2\,\frac{1-\varepsilon}{1/2}\,=\\ &=\;\varepsilon\cdot\text{ld}\;2-\varepsilon\cdot\log_2\,\frac{1}{\varepsilon}+(1-\varepsilon)\cdot\text{ld}\;2-(1-\varepsilon)\cdot\log_2\,\frac{1}{1-\varepsilon}=1-H_{\rm bin}(\varepsilon) \end{aligned}$$

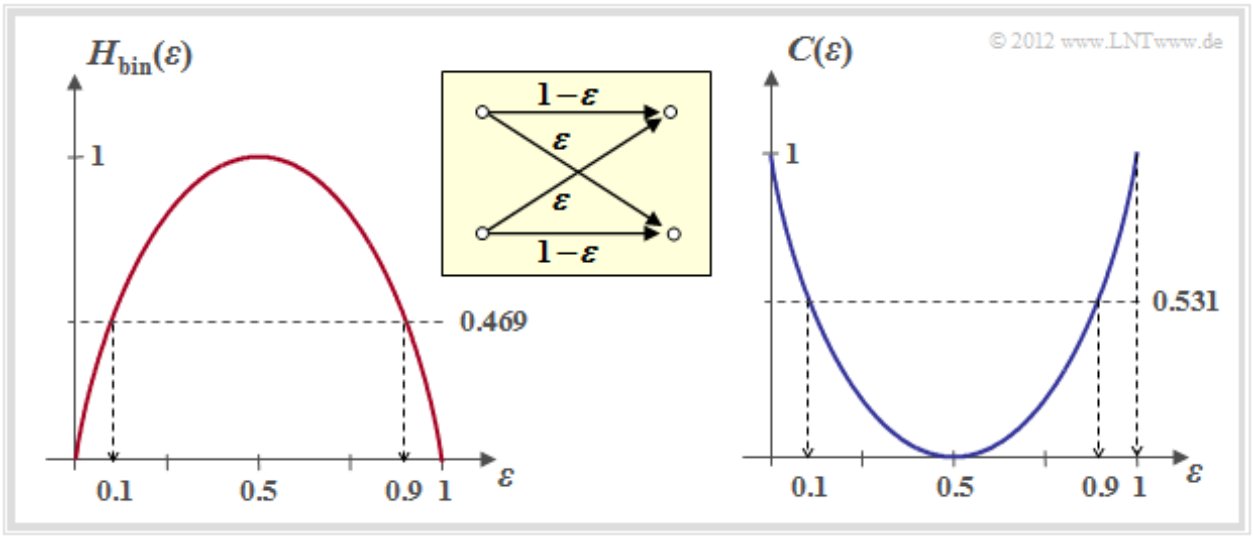
  mit

  $$H_{\rm bin}(\varepsilon)=\varepsilon\cdot\log_2\,\frac{1}{\varepsilon}+(1-\varepsilon)\cdot\log_2\,\frac{1}{1-\varepsilon}\,.$$

Das Ergebnis wird auf der nächsten Seite grafisch dargestellt.

Kanalkapazität des BSC–Modells (2)

Die rechte Grafik zeigt die BSC–Kanalkapazität abhängig von der Verfälschungswahrscheinlichkeit ε. Links ist zum Vergleich die binäre Entropiefunktion dargestellt, die bereits im **Kapitel 1.1** des Buches „Einführung in die Informationstheorie" definiert wurde.

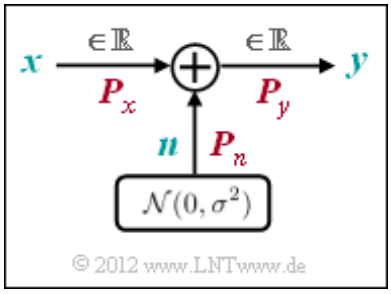


Man erkennt aus dieser Darstellung:

- Die Verfälschungswahrscheinlichkeit ε führt zur Kanalkapazität $C(\varepsilon)$. Eine fehlerfreie Decodierung nach bestmöglicher Codierung ist nach dem **Kanalcodierungstheorem** nur dann möglich, wenn die Coderate R kleiner ist als $C(\varepsilon)$.
- Mit $\varepsilon = 10\%$ ist wegen $C(0.1) = 0.531$ eine fehlerfreie Decodierung nicht möglich, wenn die Coderate $R \geq 0.531$ beträgt. Bei 50–prozentiger Verfälschung ist eine fehlerfreie Decodierung auch bei beliebig kleiner Coderate unmöglich: $C(0.5) = 0$.
- Aus informationstheoretischer Sicht ist $\varepsilon = 1$ (Invertierung aller Bits) gleich gut wie $\varepsilon = 0$ (fehlerfreie Übertragung). Ebenso ist $\varepsilon = 0.9$ äquivalent zu $\varepsilon = 0.1$. Eine fehlerfreie Decodierung erzielt man hier durch Vertauschen der Nullen und Einsen, also durch ein *Mapping*.

Kanalkapazität des AWGN–Modells (1)

Wir betrachten den AWGN–Kanal (*Additive White Gaussian Noise*). Hier gilt für das Ausgangssignal $y = x + n$, wobei n eine gaußverteilte Zufallsgröße beschreibt, und es gilt $E[n] = 0$ und $E[n^2] = P_n$.



Damit ergibt sich unabhängig vom Eingangssignal x ein kontinuierliches Ausgangssignal y, und in der **Gleichung** für die Transinformation ist $M_y \to \infty$ einzusetzen.

Die Berechnung der Kanalkapazität für den AWGN–Kanal wird hier nur in Stichworten angegeben. Die genaue Herleitung finden Sie im **Kapitel 4** des Buches „Einführung in die Informationstheorie":

- Die im Hinblick auf maximale Transinformation optimierte Eingangsgröße x wird mit Sicherheit wertkontinuierlich sein, das heißt, beim AWGN–Kanal gilt außer $M_y \to \infty$ auch $M_x \to \infty$.
- Während bei wertdiskretem Eingang über alle $\Pr(x_i)$ zu optimieren ist, erfolgt nun die Optimierung anhand der **WDF** $f_x(x)$ des Eingangssignals unter der Nebenbedingung **Leistungsbegrenzung:**

$$C = \max_{f_x(x)} I(x; y), \quad \text{wobei gelten muss : } \mathrm{E}\left[x^2\right] \le P_x .$$

- Die Optimierung liefert für die Eingangs–WDF ebenfalls eine Gaußverteilung $\Rightarrow$ x, n und y sind gaußverteilt gemäß den Dichtefunktionen $f_x(x)$, $f_n(n)$, $f_y(y)$ und den Leistungen P_x, P_n und P_y.
- Nach längerer Rechnung erhält man für die Kanalkapazität unter Verwendung des *Logarithmus dualis* $\log_2(.)$ – wiederum mit der Pseudo–Einheit „bit/Kanalzugriff":

$$C = \log_2 \sqrt{\frac{P_y}{P_n}} = \log_2 \sqrt{\frac{P_x + P_n}{P_n}} = \frac{1}{2} \cdot \log_2 \left(1 + \frac{P_x}{P_n}\right) .$$

- Beschreibt x ein zeitdiskretes Signal mit der Symbolrate $1/T_S$, so muss dieses auf $B = 1/(2T_S)$ bandbegrenzt sein, und die gleiche Bandbreite B muss man für das Rauschsignal n ansetzen:

$$P_X = \frac{E_S}{T_S}, \quad P_N = \frac{N_0}{2T_S}.$$

- Somit lässt sich die AWGN–Kanalkapazität auch durch die **Sendeenergie pro Symbol** (E_S) und die **Rauschleistungsdichte** (N_0) ausdrücken:

$$C = \frac{1}{2} \cdot \log_2 \left(1 + \frac{2E_S}{N_0}\right), \qquad \text{Einheit : } \frac{\text{bit}}{\text{Kanalzugriff}},$$

$$C^* = \frac{C}{T_S} = B \cdot \log_2 \left(1 + \frac{2E_S}{N_0}\right), \qquad \text{Einheit : } \frac{\text{bit}}{\text{Zeiteinheit}}.$$

Rechenbeispiel: Für das Zahlenverhältnis $E_S/N_0 = 7.5 \Rightarrow 10 \cdot \lg E_S/N_0 = 8.75$ dB erhält man die Kanalkapazität $\boldsymbol{C = 1/2 \cdot \log_2 (16) = 2}$ **bit/Kanalzugriff**. Bei einem Kanal mit der (physikalischen) Bandbreite $B = 4$ kHz, was der Abtastrate $f_A = 8$ kHz entspricht, gilt zudem $\boldsymbol{C^* = 16}$ **kbit/s**.

Kanalkapazität des AWGN–Modells (2)

Ein Vergleich verschiedener Codierverfahren bei konstantem E_S (Energie pro übertragenem Symbol) ist nicht fair. Vielmehr sollte man für diesen Vergleich die Energie E_B pro Nutzbit fest vorgeben. Dabei gilt:

$$E_S = R \cdot E_B \quad \Rightarrow \quad E_B = E_S/R\,.$$

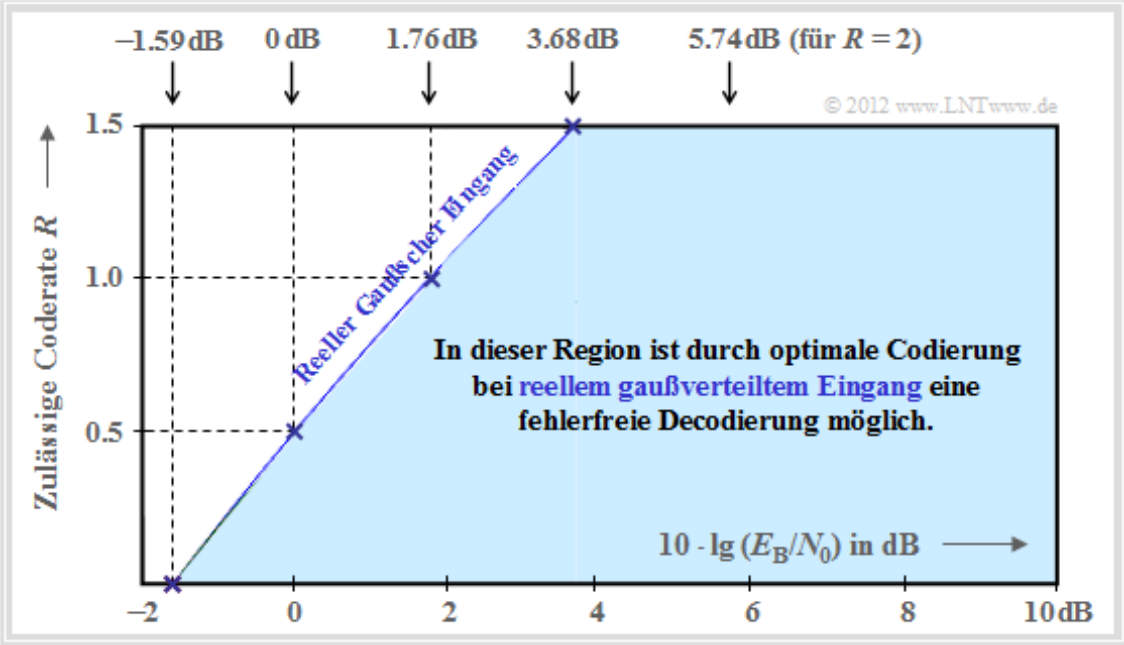
Damit lautet das *Kanalcodierungstheorem*: Eine fehlerfreie Decodierung (bei unendlich langen Blöcken) ist möglich, falls die Coderate R kleiner ist als die Kanalkapazität C:

$$R < C = 1/2 \cdot \log_2\left(1 + 2 \cdot R \cdot E_B/N_0\right).$$

Für jede Coderate R lässt sich damit das erforderliche E_B/N_0 des AWGN–Kanals ermitteln, damit eine fehlerfreie Decodierung möglich ist. Man erhält für den Grenzfall $R = C$:

$$E_B/N_0 > (2^{2R} - 1)/(2R).$$

Die Grafik fasst das Ergebnis zusammen, wobei die Ordinate R im linearen Maßstab und die Abszisse E_B/N_0 logarithmisch aufgetragen ist. Außerhalb der blauen Fläche ist eine fehlerfreie Codierung nicht möglich. Die blaue Grenzkurve gibt die Kanalkapazität C des AWGN–Kanals an.



Aus dieser Grafik und obiger Gleichung lässt sich Folgendes ableiten:

- Die Kanalkapazität C steigt etwas weniger als linear mit $10 \cdot \lg E_B/N_0$ an. In der Grafik sind einige ausgewählte Funktionswerte angegeben (blaue Kreuze).
- Ist $10 \cdot \lg (E_B/N_0)$ kleiner als –1.59 dB, so ist eine fehlerfreie Decodierung prinzipiell unmöglich. Beträgt die Coderate $R = 0.5$, so muss $10 \cdot \lg (E_B/N_0)$ größer als 0 dB $\Rightarrow$ $E_B > N_0$ sein.
- Für alle binären Codes gilt per se $0 \le R \le 1$. Coderaten $R > 1$ sind nur mit *nichtbinären* Codes möglich. Beispielsweise beträgt die maximal mögliche Coderate eines quaternären Codes $R = 2$.
- Alle eindimensionalen Modulationsarten – also solche Verfahren, die nur die Inphase– oder nur die Quadraturkomponente nutzen wie ASK, BPSK und FSK – müssen im blauen Bereich liegen.

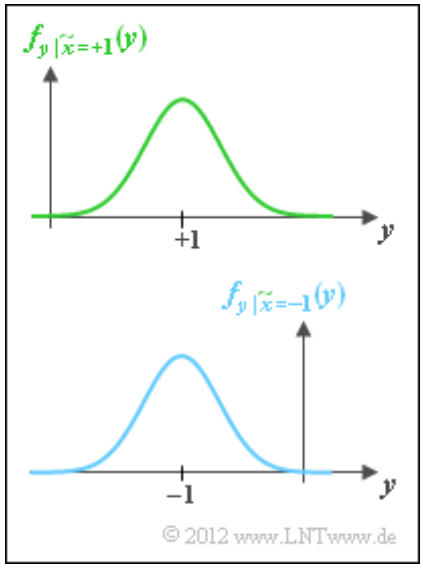
AWGN–Kanalkapazität für binäre Eingangssignale

In diesem Buch beschränken wir uns meist auf binäre Codes, also auf das Galoisfeld GF(2^n). Damit ist

- zum einen die Coderate auf den Bereich $R \le 1$ begrenzt,
- zum zweiten auch für $R \le 1$ nicht die gesamte blaue Region (siehe letzte Seite) verfügbar.

Die nun gültige Region ergibt sich aus der **allgemeinen Gleichung** der Transinformation durch

- die Parameter $M_x = 2$ und $M_y \to \infty$,
- bipolare Signalisierung $\Rightarrow$ $x = 0 \to \tilde{x} = +1$, $x = 1 \to \tilde{x} = -1$,
- den Übergang von bedingten Wahrscheinlichkeiten $\Pr(\tilde{x}_i)$ zu bedingten Wahrscheinlichkeitsdichtefunktionen,
- Ersetzen der Summe durch eine Integration.



Die Optimierung der Quelle führt auf gleichwahrscheinliche Symbole:

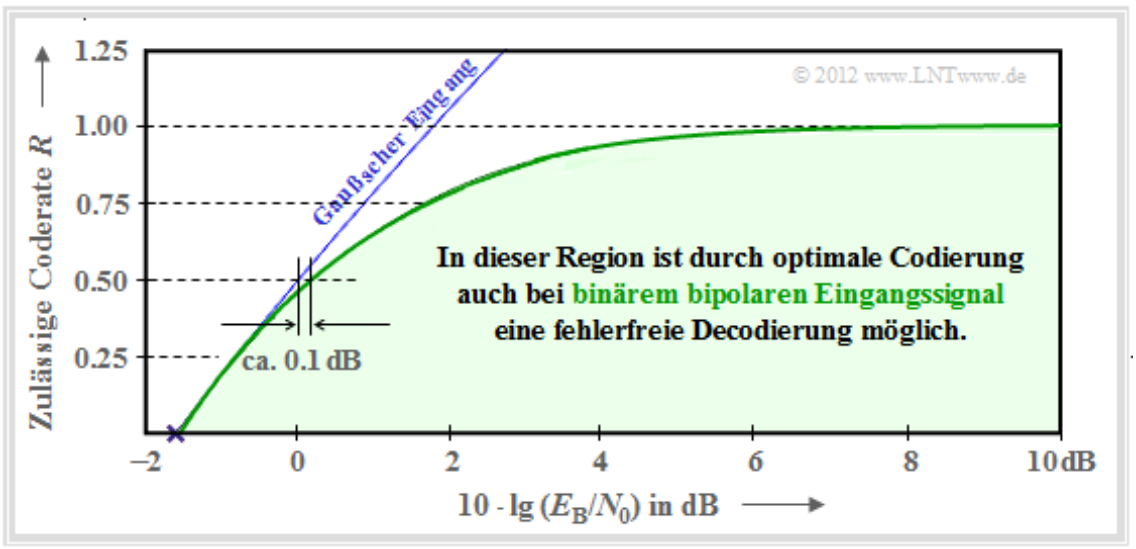
$$\Pr(\tilde{x} = +1) = \Pr(\tilde{x} = -1) = 1/2\,.$$

Damit erhält man bei binärem Eingang (±1) für die Kanalkapazität $\Rightarrow$ Maximum der Transinformation hinsichtlich $\Pr(\tilde{x}_i)$:

$$C = 1/2 \cdot \int_{-\infty}^{+\infty} \left[f_{y\,|\,\tilde{x}=+1}(y) \cdot \log_2 \frac{f_{y\,|\,\tilde{x}=+1}(y)}{f_y(y)} + f_{y\,|\,\tilde{x}=-1}(y) \cdot \log_2 \frac{f_{y\,|\,\tilde{x}=-1}(y)}{f_y(y)} \right] \mathrm{d}y\,.$$

Das Integral lässt sich nicht in mathematisch–geschlossener Form lösen, sondern kann nur numerisch ausgewertet werden. Die grüne Kurve zeigt das Ergebnis. Die blaue Kurve gibt zum Vergleich die auf der letzten Seite hergeleitete Kanalkapazität für gaußverteilte Eingangssignale an. Man erkennt:

- Für $10 \cdot \lg E_B/N_0 < 0$ unterscheiden sich die beiden Kapazitätskurven nur geringfügig. So muss man bei binärem bipolaren Eingang gegenüber dem Optimum (Gaußscher Eingang) die Kenngröße $10 \cdot \lg E_B/N_0$ nur etwa um 0.1 dB erhöhen, um ebenfalls die Coderate $R = 0.5$ zu ermöglichen.
- Ab etwa $10 \cdot \lg E_B/N_0 = 6$ dB ist die Kanalkapazität $C = 1$ bit/Kanalzugriff für binären Eingang (fast) erreicht. Dazwischen verläuft die grüne Grenzkurve annähernd exponentiell ansteigend.

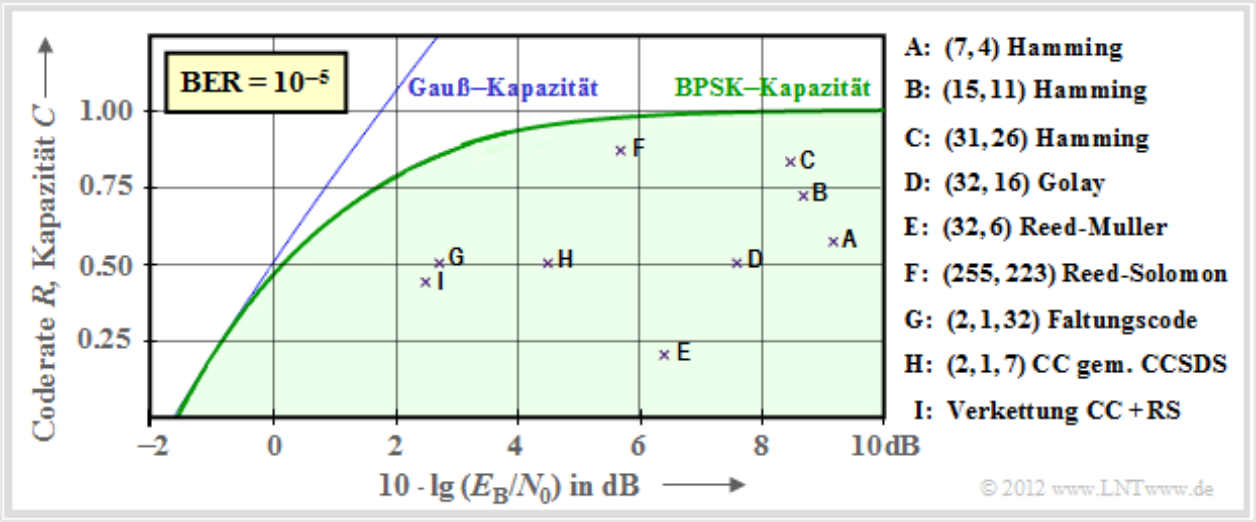


Gebräuchliche Kanalcodes im Vergleich zur Kanalkapazität (1)

Nun soll gezeigt werden, in wie weit sich etablierte Kanalcodes der BPSK–Kanalkapazität (grüne Kurve) annähern. In der Grafik ist als Ordinate die Rate $R = k/n$ dieser Codes bzw. die Kapazität C (mit der zusätzlichen Pseudo–Einheit „bit/Kanalzugriff") aufgetragen. Weiter ist vorausgesetzt:

- der AWGN–Kanal, gekennzeichnet durch $10 \cdot \lg E_B/N_0$ in dB, und
- für die durch Kreuze markierten Codes eine Bitfehlerrate (BER) von 10^{-5}.

Zu beachten ist, dass die Kanalkapazitätskurven stets für $n \to \infty$ und BER = 0 gelten. Würde man diese strenge Forderung „feherfrei" auch an die betrachteten Kanalcodes endlicher Codelänge n anlegen, so wäre hierfür stets $10 \cdot E_B/N_0 \to \infty$ erforderlich. Dies ist aber ein akademisches Problem, das für die Praxis wenig Bedeutung hat. Für BER = 10^{-10} ergäbe sich eine qualitativ ähnliche Grafik.



Es folgen einige Erläuterungen zu den Daten, die der Vorlesung **[Liv10]** entnommen wurden:

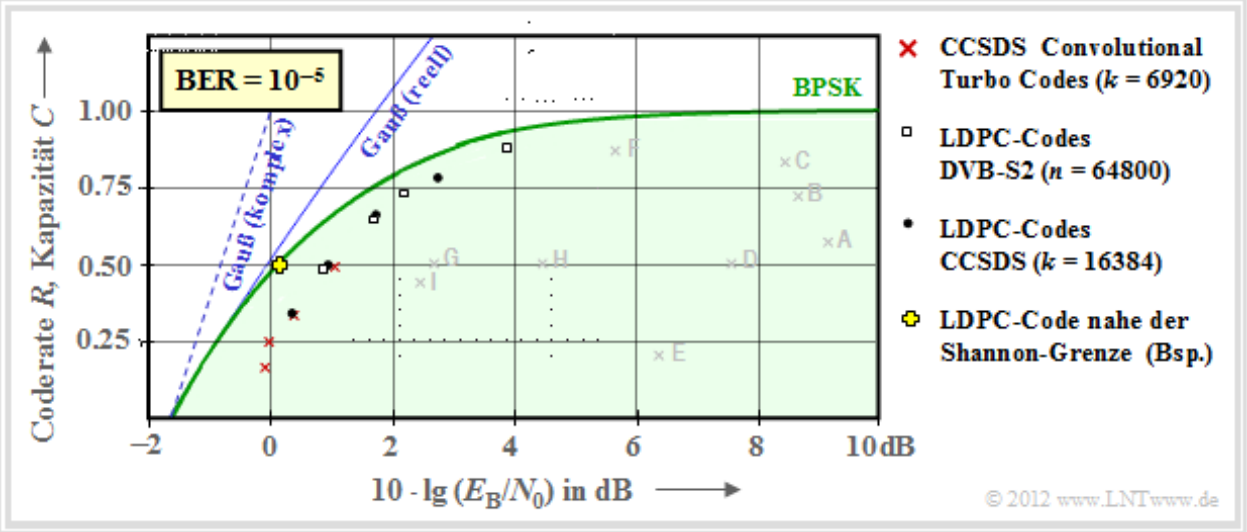
- Die Punkte **A**, **B** und **C** markieren Hamming–Codes unterschiedlicher Rate, die im **Kapitel 1.3** bereits ausführlich behandelt wurden. Sie alle benötigen $10 \cdot \lg E_B/N_0 > 8$ dBfür BER = 10^{-5}.
- **D** kennzeichnet den binären **Golay–Code** mit der Rate 1/2 und **E** einen **Reed–Muller–Code.** Dieser sehr niederratige Code kam bereits 1971 bei der Raumsonde Mariner 9 zum Einsatz.
- Die **Reed–Solomon–Codes** (RS–Codes) werden im Kapitel 2 noch ausführlich behandelt. Mit **F** markiert ist der RS–Code der Rate 223/255 > 0.9 und einem erforderlichen $E_B/N_0 < 6$ dB.
- Die Markierungen **G** und **H** bezeichnen beispielhafte **Faltungscodes** (englisch: *Convolutional Codes*, CC) mittlerer Rate. Ersterer wurde schon 1972 bei der Pioneer10–Mission eingesetzt.
- Die Kanalcodierung der Voyager–Mission Ende der 1970er Jahre ist mit **I** markiert. Es handelt sich um die Verkettung eines (2, 1, 7)–Faltungscodes mit einem Reed–Solomon–Code.

Anzumerken ist, dass bei den Faltungscodes insbesondere der dritte Parameter der Kennung eine andere Bedeutung hat als bei den Blockcodes. (2, 1, 32) weist beispielsweise auf das Memory $m = 32$ hin.

Auf der nächsten Seite folgen noch Kenndaten von Systemen mit iterativer Decodierung.

Gebräuchliche Kanalcodes im Vergleich zur Kanalkapazität (2)

Mit iterativer Decodierung lassen sich deutlich bessere Ergebnisse erzielen, wie die folgende Grafik zeigt. Das heißt: Die einzelnen Markierungspunkte liegen sehr viel näher an der Kapazitätskurve.



Die bisher mit „Gauß–Kapazität" beschriftete durchgezogene blaue Kurve wird hier „Gauß (reell)" genannt. Hier noch einige weitere Erläuterungen zu dieser Grafik:

- Rote Kreuze markieren sogenannte **Turbocodes** nach CCSDS (*Consultative Committee for Space Data Systems*) mit jeweils $k = 6920$ Informationsbits und unterschiedlichen Codelängen n. Diese von **Claude Berrou** um 1990 erfundenen Codes können iterativ decodiert werden. Die (roten) Markierungen liegen jeweils weniger als 1 dB von der Shannon–Grenze entfernt.
- Ähnliches Verhalten zeigen die durch weiße Rechtecke gekennzeichneten **LDPC–Codes** (*Low Density Parity–check Codes*), die seit 2006 bei DVB–S2 (*Digital Video Broadcast over Satellite*) eingesetzt werden. Diese eignen sich aufgrund der spärlichen Belegung der Prüfmatrix **H** (mit Einsen) sehr gut für die iterative Decodierung mittels **Faktor–Graphen** und **Exit Charts.**
- Die schwarzen Punkte markieren die von CCSDS spezifizierten LDPC–Codes, die alle von einer konstanten Anzahl von Informationsbits ausgehen ($k = 16384$). Dagegen ist bei allen weißen Rechtecken die Codelänge $n = 64800$ konstant, während sich die Anzahl k der Informationsbits entsprechend der Rate $R = k/n$ ändert.
- Um das Jahr 2000 hatten viele Forscher den Ehrgeiz, sich der Shannon–Grenze bis auf Bruchteile von einem dB anzunähern. Das gelbe Kreuz markiert ein solches Ergebnis aus [CFRU01] von 2001. Verwendet wurde ein irregulärer Rate–1/2–LDPC–Code der Codelänge $n = 2 \cdot 10^6$.

Festzuhalten ist, dass Shannon bereits 1948 erkannt und nachgewiesen hat, dass kein eindimensionales Modulationsverfahren links von der durchgehend eingezeichneten AWGN–Grenzkurve „Gauß (reell)" liegen kann. Für zweidimensionale Verfahren wie QAM und mehrstufige PSK gilt dagegen die Grenzkurve „Gauß (komplex)", die hier gestrichelt gezeichnet ist und stets links von der durchgezogenen Kurve liegt. Näheres hierzu im **Kapitel 4.3** des Buches „Einführung in die Informationstheorie".

Auch diese Grenzkurve wurde mit QAM und sehr langen Kanalcodes inzwischen nahezu erreicht, ohne dass sie jemals überschritten werden wird.