Besser sehen, besser hören!
Fehlerkorrigierende Codes

Ringvorlesung
Technische Mathematik
20. Mai 2010

Hermann Kautschitsch

Institut für Mathematik
Universität Klagenfurt

Vorwort

Der stetig zunehmende Einsatz von elektronischen Instrumenten und des Computers zur Datengewinnung und Datenerzeugung verlangt nach rascher und korrekter Übertragung dieser Daten[1]. Jeder Übertragungskanal ist durch ein bestimmtes Maß an Störungen (Rauschen) bestimmt, welches die Datenübertragung beeinträchtigt. Hören wir Radio oder lesen ein Telegramm, so können wir die beim diesem Rauschen entstandenen Fehler meist vernachlässigen. Auch wenn wir miteinander sprechen, so müssen wir nicht jedes Wort exakt verstehen und können trotzdem den Sinn eines Satzes rekonstruieren. Unsere Sprache ist redundant. Anders, wenn wir Daten von einem Satelliten empfangen oder über das Internet versenden. Hier liegt im Allgemeinen keine Redundanz vor. Wie können wir den praktisch unvermeidlichen Übertragungsfehlern begegnen? Hier hilft, wie in vielen anderen Fragestellungen, die Mathematik mit ihren vielfältigen Werkzeugen. Ein Lösungsansatz besteht, im Sinne der Mathematik, darin, Daten in einen Code „einzupacken". Mit Hilfe dieses Codes können Fehler erkannt (error detecting code) und in vielen Fällen auch korrigiert (error correcting codee) werden. Mit solchen Codes wollen wir uns – wenn wir von den ersten einführenden Beispielen absehen - beschäftigen.

Einleitung

Computer, Handy, DVD, CD, Internet etc. speichern und übertragen Daten (wir werden diese Daten auch **Klartext** nennen) in binärer Form (0 und 1). Insofern werden wir uns auf binäre Codes konzentrieren. Tritt bei einer Signalübertragung ein Fehler auf, so wird aus 0 eine 1 bzw. aus 1 eine 0. Wir nehmen an, dass die entsprechenden Wahrscheinlichkeiten gleich sind. Man sagt, dass entsprechende Übertragungskanäle binär symmetrisch sind

1 Elemente der Codierungstheorie

Wir beschreiben nun an Beispielen die digitale Übermittlung von Text, Bild und Ton.

Beispiel: Quellencodierung eines Textes.

Wir senden „ICH KOMME". Zunächst werden Buchstaben und Sonderzeichen durch Paare von Ziffern ersetzt. Solche Ersetzungen nennt man Codierungen. Wir verwenden den ASCII Code[2]. Mit I->73, C->67 ….folgt:
„ICH KOMME"->73 67 72 75 79 77 77 69. Im nächsten Schritt werden die Ziffernpaare durch binäre 8-er Blöcke (8-Tupel aus 0 und 1) ersetzt (codiert). D.h., wir schreiben das Ziffernpaar z -> $b_7b_6b_5\ldots b_1b_0$ falls $z=b_7 2^7+b_6 2^6+\ldots+b_0$ mit $b_i \in \{0,1\}$ für i=0,1,2,..,7. Weil $73=0\ 2^7+1\ 2^6+0\ 2^5+0\ 2^4+1\ 2^3+0\ 2^2+0\ 2^1+1\ 2^0$ wird 73 durch 01001001 ersetzt. So fortfahrend lautet unsere Nachricht:

01001001 01000011 01001000 01001011 01001111 01001101 01001101 01000101

Diese Ersetzung eines Textes in eine 0,1-Folge nennt man **Quellencodierung**.

[1] Vgl. Gilbert, 286 ff
[2] American Standard Code for Information Interchange (ASCII, alternativ US-ASCII, oft [æski] ausgesprochen) ist eine 7-Bit-Zeichenkodierung und bildet die US-Variante von ISO 646 sowie die Grundlage für spätere mehrbittige Zeichensätze und -kodierungen.

Wie wird ein Bild in eine 0,1-Folge übersetzt also quellencodiert?

Beispiel: Quellencodierung eines Bildes.

Wir senden folgendes Bild:

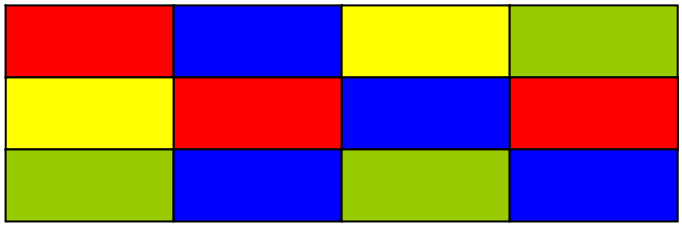


Abbildung 1

Zuerst ordnen wir den Feldern des Bildes ganzzahlige Koordinaten (i,j) mit $0 \leq i \leq 2$ und $0 \leq j \leq 3$ zu und codieren i und j jeweils binär. Den Farbinhalt eines Feldes wird festgelegt durch rot=(0,0), gelb=(0,1), grün=(1,0), blau=(1,1). In unserem Bild wird dritte Feld in der zweiten Zeile ersetzt durch (1,2, blau). Wegen 1=01 und 2=10 und blau=(1,1) ist dieses Feld durch das 6-Tupel
(01 10 11) festgelegt. Insgesamt wird obiges Bild ersetzt durch

000000 000111 001001 001110 010001 010100 011011 011100 110010 110111 111010 101111

Zum Beispiel wurde bei den Aufnahmen durch die Mariner-Sonden (Marsexpeditionen) jedes Bild in 658240 Felder (Bildpunkte, Pixel) zerlegt und jedes Pixel hatte eine Helligkeitsabstufung zwischen 1 und 2^8=256, welche durch r=8Bits wieder gegeben wurde, sodass pro Aufnahme etwa 5 Millionen Bits an Information enthalten waren[3].

Die Quellencodierung eines Tonsignales ist wesentlich aufwändiger.

Beispiel: Quellencodierung einer Audioinformation

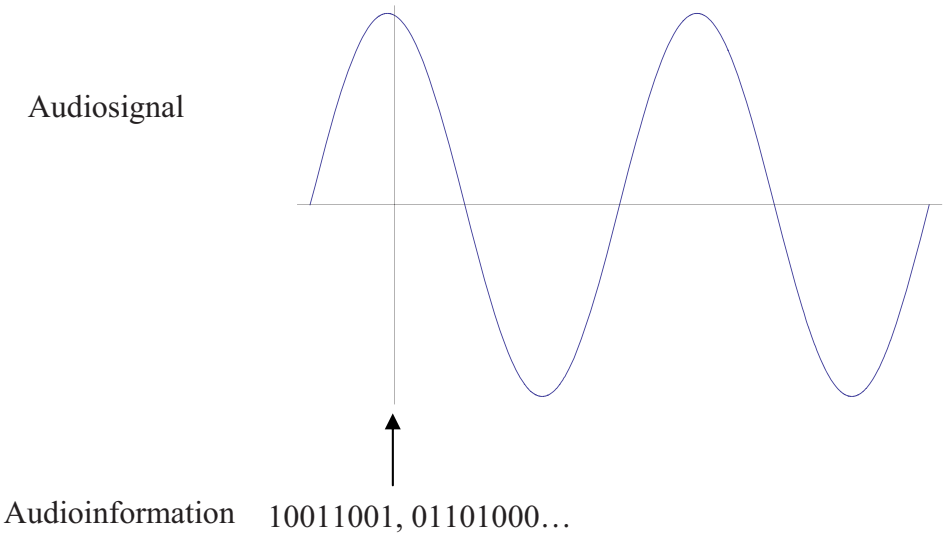


Abbildung 2

[3] Vgl. Dorninger, 1996, S. 90.

Wir erkennen, dass in allen Fällen (Text, Ton und Bild) die ursprüngliche Information durch eine 0,1-Folge ersetzt. Zur leichteren Lesbarkeit werden die im Allgemeinen sehr langen 0,1-Folgen in Blöcke unterteilt. Dieser Vorgang entspricht der Verwendung von Worten in der gesprochenen Sprache über dem Alphabet {A,.., Z}. Bei der digitalen Dateninformation entsprechen den Worten Blöcke über dem Alphabet {0,1}. Allerdings sind diese Blöcke immer gleich lang.

Bei einer Datenübertragung treten in der Regel Fehler auf.

Abbildung 3

Abbildung 3 zeigt ein Bild, das den übersandten Daten einer Marsexpedition entspricht. Details sind nicht erkennbar. Die Daten sind in dieser Form unbrauchbar. Erst eine Fehlerkorrektur führt zum Bild in Abbildung 4.

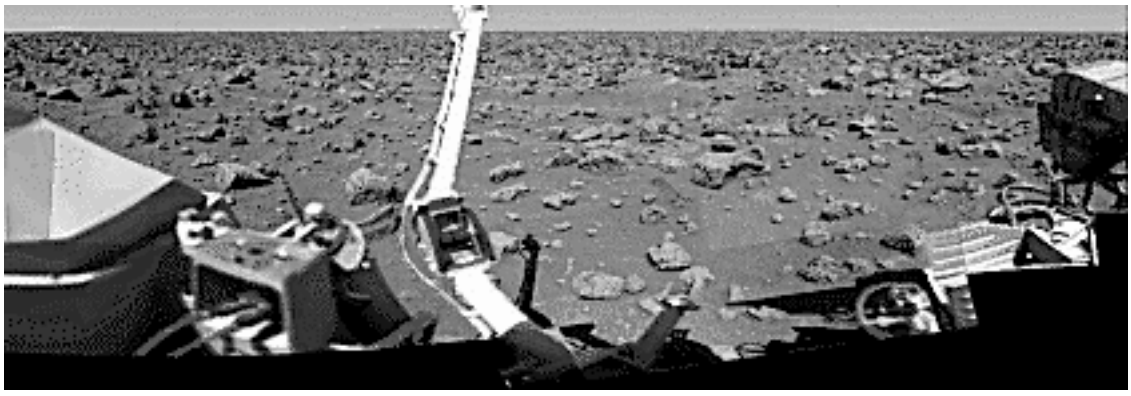


Abbildung 4

Wie können diese Fehler entdeckt oder sogar korrigiert werden?

Codieren mit Prüfzeichen bzw. Prüfziffern

Bei einer Datenübertragung haben wir folgende Worte empfangen. Offensichtlich sind Fehler entstanden.

Aufgabe 1 : Was könnten die folgenden Wörter bedeuten: MARHEMATAK?
KLIGENFAST?

Um diese Frage auf systematische Weise bearbeiten zu können, wollen wir den Begriff der Hamming-Distanz[4] definieren.

[4] US-amerikanischer Mathematiker Richard Wesley Hamming (1915–1998)

Definition 1: Unter der **Hamming-Distanz** d zweier Wörter mit gleicher Buchstabenanzahl versteht man die Anzahl der Stellen, an denen sich die beiden Wörter unterscheiden.

Dazu nun die folgende

Aufgabe 2: Berechne die Hamming-Distanzen:

d(MARHEMATAK, MATHEMATIK) = 2
d(KLIGENFAST, KLAGENFURT) = 3

Wir vermuten, dass je kleiner die Hamming-Distanz d ist, desto eher kann man ein empfangenes Wort entschlüsseln (**decodieren**), also den Klartext ermitteln. In dieser 2. Aufgabe leistet unsere Sprachkenntnis bzw. unser geografisches Wissen die erfolgreiche Dekodierung der vorliegenden Wörter. Wie sollen wir vorgehen, wenn entsprechendes Wissen nicht vorliegt? Wir formulieren damit die zentrale Frage:

Wie kann man Übertragungsfehler **entdecken** ? Eventuell sogar **korrigieren** ?

Beispiel: Wiederholungscode

Nehmen wir an, unsere Originalnachricht lautet WANN. Eine erste einfache **Strategie** zur Erkennung von Fehlern besteht in der wiederholten Sendung der Originalnachricht. Wie senden also den Code WANN|WANN. Der Empfänger erhält z.B. die Buchstabenfolge WANNDANN. Weil auch er weiß, dass gleich lange Blöcke übertragen werden, in unserem Fall 4-Blöcke, liest er WANN|DANN. Damit ist für ihn klar: Mindestens ein Fehler ist aufgetreten. Aber wo?

Aus der einfachen Wiederholung kann die Position des Fehlers nicht bestimmt werden. Als Originalnachricht käme sowohl WANN als auch DANN in Frage. Eine mögliche Verbesserung besteht im Vorschlag, die Originalnachricht mindestens zweimal zu wiederholen. Also:
Gesendetes Wort (zweimalige Wiederholung): WANN|WANN|WANN
Empfangenes Wort (unter der Annahme, dass nur ein Fehler aufgetreten ist):
WANN|DANN|WANN.

Aufgabe 3 :
i) Decodierung des obigen Wortes: …….
ii) Empfangenes Wort: DANNWANNDANN
Decodierung: …….
iii) Finde ähnliche Beispiele!

Wir halten fest, dass ein Fehler entdeckt und korrigiert werden kann, sofern man von der Wiederholungsstrategie Kenntnis hat! Der Nachteil dieses elementaren Codierungsverfahrens liegt in der Menge der gesandten Daten. Die übertragenen Worte werden sehr lang!

00110 und 00100 → Hamming-Abstand=1
12345 und 13344 → Hamming-Abstand=2
Haus und Baum → Hamming-Abstand=2
(vgl. http://de.wikipedia.org/wiki/Hamming-Abstand, 24. 08. 2009).

Im Sinne der eingangs vorgestellten Quellencodierung wollen uns auf die Übermittlung von 0,1-Folgen beschränken. Wie wir oben gesehen haben, ermöglicht die

Verlängerung einer übermittelten Nachricht durch Anfügen von 0 und 1 das Entdecken und sogar das Korrigieren von Fehlern.

Daraus ergibt für uns folgende Strategie:

1. Unterteile die 0,1-Folge in Blöcke zu je k Elementen ($k \in \mathbf{N}$).
   Diese k-elementigen Blöcke nennt man **Nachrichtenwörter**.´

2. Hänge an jeden Block n-k weitere Zeichen (Prüfzeichen, Prüfstellen) aus dem verwendeten Alphabet A (in unserem Fall A={0,1}). Dadurch entstehen Blöcke der Länge n=k+(n-k). Diese n-elementigen Wörter nennt man **Codewörter**.

3. Verlängere die Nachrichtenwörter so geschickt, dass
   - entweder alle Mengen von t Fehlern entdeckt werden
   (man spricht von einem **t-Fehler entdeckender Code**)
   - oder alle Mengen von t Fehlern korrigiert werden
   (man spricht von einem **t-Fehler korrigierender Code**) und dabei die Wörter nur in einem vertretbaren Ausmaß länger werden.

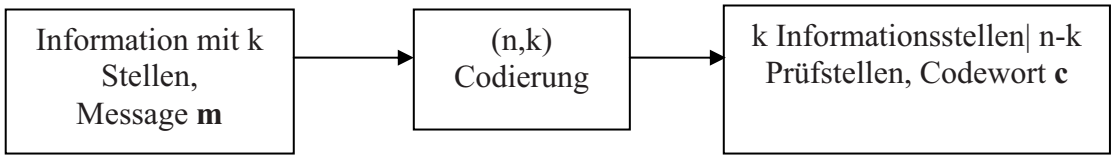


Abbildung 5

Codieren bedeutet: **Verlängere** Nachrichten $m=(m_1,m_2,\ldots,m_k)$ der Länge k durch Hinzufügen von (n-k) Prüfzeichen $(x_1,x_2,\ldots,x_{n-k})$ auf ein Wort der Länge n. Jedem k-stelligen Nachrichtenwort über A wird also für die Übertragung ein n-stelliges Codewort über A zugeordnet. Diese Zuordnung heißt **Kanalcodierung**. Die verwendete Zuordnungsvorschrift heißt **Codierungsfunktion**.

Die Menge aller Codewörter, welche als Funktionswerte der Codierungsfunktion auftreten, wird (n,k)-Code genannt.

Verwendet man als Alphabet A={0,1}, so spricht man von einem **binären Code**. Bezeichnet man die Codierungsfunktion eines binären (n,k)-Codes mit f, dann ist im Sinne der Mathematik f eine injektive Abbildung aus der Menge der k-Tupel $\mathbf{m}=(m_1,\ldots,m_k)$ in die Menge der n-Tupel $\mathbf{c}=(c_1,\ldots,c_n)$ jeweils über $\{0,1\}$[5]. Symbolisch:

$$f: \{0,1\}^k \mapsto \{0,1\}^n$$

$$f(\mathbf{m})=\mathbf{c} \text{ bzw. } f(m_1,\ldots,m_k)= (c_1,\ldots,c_n)$$

[5] **m**…Message; **c**…Code

Beispiel: Wiederholungscode (repeating code) mit mathematischen Symbolen.
Wähle n = rk mit r ≥ 2. Die Codierungsfunktion f wird festgesetzt durch
$f(m_1,\ldots,m_k):=(m_1,\ldots,m_k, m_1,\ldots,m_k,\ldots, m_1,\ldots,m_k)$.
Also ist $c_1=m_1,\ldots,c_k=m_k,c_{k+1}=m_1,\ldots,c_{2k}=m_k,c_{2k+1}=m_1,\ldots,c_{rk}=c_n=m_k$.
Das Codewort **c** entsteht aus dem Nachrichtenwort **m** durch r-fache Wiederholung.
Ist z.B. k=2 und r=3, so ist

$$f(0,0)=(0,0,0,0,0,0),$$
$$f(0,1)=(0,1,0,1,0,1),$$
$$f(1,0)=(1,0,1,0,1,0),$$
$$f(1,1)=(1,1,1,1,1,1).$$

Dieser Wiederholungscode $\mathbf{C_w}$ ist damit ein (6,2)-Code, der aus vier Codewörtern besteht, weil $|\{0,1\}^2|=4$ gilt. Die Anzahl der Codewörter eines (n,k)-Codes ist immer gleich der Mächtigkeit von $\{0,1\}^k$, also gleich 2^k. Der (n,k)-Code ist immer eine echte Teilmenge der Menge aller n-Tupel.

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

Wird z.B. das Wort **u**=(1,1,0,1,0,1) empfangen, so ist mindestens ein Fehler aufgetreten, weil dieses Wort nicht in $\mathbf{C_w}$ enthalten ist. Es ist anzunehmen, dass es vom Nachrichtenwort **m**=(0,1) stammt. Dessen Codewort ist (0,1,0,1,0,1) und unterscheidet sich vom empfangenen Wort nur an einer Stelle. Die Hammingdistanz d((0,1,0,1,0,1), (1,1,0,1,0,1)) dieser beiden Wörter ist 1. Alle anderen Hammingdistanzen von **u** zu den restlichen Codewörtern sind größer als 1.

Dies wird in unseren nachfolgenden Ausführungen die allgemeine Strategie zur Decodierung sein:

Suche im Code **C** jenes Wort, welches zum empfangenen Wort **u**
die kleinste Hammingdistanz d besitzt (Maximum Likelihood Decodierung[6], MLD) .

Es kann auch geschehen, dass es in **C** mehr als nur ein Wort mit „kleinster“ Hammingdistanz gibt.

Beispiel: Quersummenprüfcode (parity check code)

Wähle n=k+1. Es wird also zu den k Stellen des Nachrichtenwortes eine einzige Stelle hinzugefügt. Der Wert an dieser Stelle ist die Summe aller Eintragungen des Nachrichtenwortes. Die Codierungsfunktion f ist also definiert durch:

$$f(m_1,\ldots,m_k):=(m_1,\ldots,m_k, \sum_{i=1}^{k} m_i)$$

Also ist $c_1=m_1,\ldots,c_k=m_k,c_{k+1}= m_1+m_2+\ldots+m_k= \sum_{i=1}^{k} m_i$.

[6] Man nimmt an, dass es am wahrscheinlichsten ist, dass wenig Fehler auftreten.

Dabei wird die modulare Arithmetik verwendet, d.h., 1+1=0. Sonst gelten die üblichen Rechenregeln:

+	0	1
0	0	1
1	1	0

.	0	1
0	0	0
1	0	1

Die Menge der binären Symbole {0,1} bildet bezüglich dieser Operationen einen **Körper**, d.h. vereinfacht gesprochen, dass die üblichen Rechenregeln gelten wie in $\mathbb{Q}$ oder $\mathbb{R}$. Man kann also in {0,1} die vier Grundrechnungsarten ausüben. Insbesondere gilt: -1=1 und an Stelle des Subtrahierens wird addiert:
1-1=1+1=0
1-1+1-1-1=1+1+1+1+1=0+0+1=1.

Bemerkung: Den Körper mit den Elementen {0,1} und der modularen Arithmetik bezeichnet man F_2 **(**F von engl. field für Körper). Dieser Bezeichnung entsprechend wird die Menge der binären k-Tupel $\{0,1\}^k$ zu $(F_2)^k$.

Der Quersummenprüfcode ist also ein (n,k)-Code = (k+1,k)-Code. Ist k=3, dann lautet die Codierungsfunktion:
f(0,0,0)=(0,0,0,0), f(0,0,1)=(0,0,1,1), f(0,1,0)=(0,1,0,1), f(0,1,1)=(0,1,1,0), f(1,0,0)=(1,0,0,1), f(1,0,1)=(1,0,1,0), f(1,1,0)=(1,1,0,0), f(1,1,1)=(1,1,1,1)

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

Die Mächtigkeit von $\mathbf{C_Q}$ ist $8=2^3$. Die Summe aller 1 ist in jedem Codewort gleich 0, weil $(m_1+\ldots+m_k)+(m_1+\ldots+m_k)=(m_1+m_1)+\ldots+(m_k+m_k)=0+\ldots+0=0$.
Wird z.B. **u**=(1,0,1,1) empfangen, dann ist mindestens ein Fehler aufgetreten, weil **u** nicht in $\mathbf{C_Q}$ liegt.
Ist höchstens ein Fehler aufgetreten, so stammt **u** von (1,0,1) oder von (0,0,1) oder von (1,0,0). Daher ist der Fehler nicht korrigierbar.
Mehr als ein Fehler können als Folge der modularen Arithmetik nicht erkannt werden.
Als Anwendung wird der Quersummenprüfcode bei der Datenübertragung innerhalb des Computers verwendet. Falls bei der Übertragung höchstens ein Fehler auftritt, so kann dieser erkannt und die Übertragung wiederholt werden.

Bei beiden Codes (Wiederholungscode, Quersummenprüfcode) gilt für die Codierungsfunktion f die Eigenschaft

$$f(\mathbf{m_1}+\mathbf{m_2})=f(\mathbf{m_1})+f(\mathbf{m_2}) \text{ für alle } \mathbf{m_1},\mathbf{m_2} \in \{0,1\}^k$$

sowie

$$f(\lambda\,\mathbf{m})=\lambda\, f(\mathbf{m}) \text{ für alle } \mathbf{m}\in\{0,1\}^k \text{ und alle } \lambda \in \{0,1\}$$

Man sagt, dass f mit der Addition und Vervielfachung mit einem Skalar verträglich ist. Codes mit einer solchen Codierungsfunktion nennt man **Linearcodes**.

Aufgabe 4: Prüfe diese Eigenschaft für die Codefunktionen von Wiederholungscode und Quersummenprüfcode:

f[(1,0)+(1,1)] = f[(0,1)] = (0,1,0,1,0,1) und
f[(1,0)]+f[(1,1)] = (1,0,1,0,1,0) + (1,1,1,1,1,1) = (0,1,0,1,0,1)

f[(1,0,1)+(1,1,1)] = f[(0,1,0)] = (0,1,0,1) und
f[(1,0,1)]+f[(1,1,1)] = (1,0,1,0)+(1,1,1,1) = (0,1,0,1)

Eine wichtige Eigenschaft von linearen Codes **C** ist:

Satz 1: Ist **C** ein linearer Code, dann enthält **C** mit je zwei Codewörtern auch deren Summe und Differenz sowie jedes Vielfache. Symbolisch:

$$\mathbf{c_1}, \mathbf{c_2} \in \mathbf{C} \Rightarrow \mathbf{c_1} \pm \mathbf{c_2} \in \mathbf{C}$$
$$\mathbf{c} \in \mathbf{C} \Rightarrow \lambda\, \mathbf{c} \in C$$

Beweis:

$c_1, c_2 \in C \Rightarrow \exists m_1, m_2 \in \{0,1\}^k$ mit: $c_1 = f(m_1)$ und $c_2 = f(m_2)$

Wegen der Linearität von f gilt: $c_1 \pm c_2 = f(m_1) \pm f(m_2) = f(m_1 \pm m_2)$

Also ist $c_1 \pm c_2$ Bild eines Nachrichtenwortes unter f und damit ein Codewort aus C.

Analog: $\lambda c = \lambda f(m) = f(\lambda m)$. □

Zusammenfassung: Durch die Quellencodierung wird eine Text-, Bild- oder Toninformation in eine 0,1-Folge transformiert. Diese wird im nächsten Schritt in Worte gleicher Länge zerlegt. Durch die Kanalcodierung entstehen aus diesen Worten durch Anhängen von jeweils gleich vielen Prüfzeichen die Codeworte. Dies ist der Vorgang des Codierens.

Wird das Codewort **c** gesendet, so wird als Folge von Störungen bei der Sendung das von **c** verschiedene Wort **u** empfangen.

Symbolisch: **u** = **c** + **e** (**e** nennt man **Fehlerwort** (e von engl. error), **e** ist ein n-Tupel)

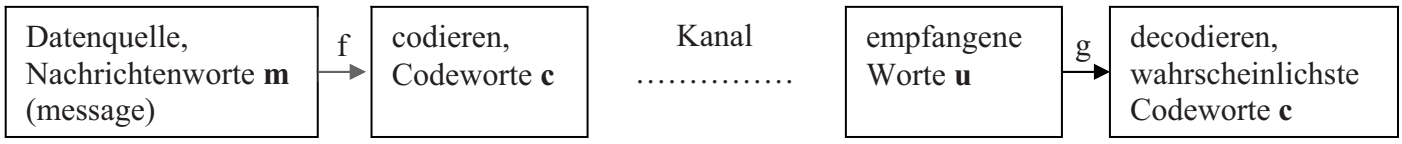
Beispiel: c = (0,1,0,1,1); **u** = (1,1,0,0,1); also (1,1,0,0,1) = (0,1,0,1,1)+(1,0,0,1,0).
Also ist das Fehlerwort **e** = (1,0,0,1,0).

Für binäre Codes gilt wegen der modularen Arithmetik:

$$\mathbf{u} = \mathbf{c} + \mathbf{e} \Leftrightarrow \mathbf{c} = \mathbf{u} + \mathbf{e} \Leftrightarrow \mathbf{e} = \mathbf{u} + \mathbf{c}$$

Beim Decodieren muss entschieden werden, welches Codewort ursprünglich gesendet wurde. Wir treffen folgende Vereinbarung (Maximum Likelihood Decodierung, abgekürzt MLD):

*Als Codewort **c**, das wahrscheinlich ursprünglich gesendet wurde, wird jenes gewählt, welches sich am wenigsten vom empfangenen Wort **u** unterscheidet. Also wird **c** so gewählt, dass die Hammingdistanz d(**c**,**u**) minimal ist.*



m → f(**m**) = **c** …… **u** = **c** + **e** g(**u**)=$\mathbf{c}_w$

f…….Codierungsfunktion
g……. Decodierungsfunktion
e……. Fehlerwort (error)
$\mathbf{c}_w$..... wahrscheinliches Codewort

Es gilt nach MLD: Die Hammingdistanz d(**u**,$\mathbf{c}_w$) ist minimal. Im Idealfall ist $\mathbf{c}_w$ = **c**.

Definition: d_{min} bezeichnet die minimale Hammingdistanz zwischen allen Worten eines Codes **C**. Symbolisch: $d_{min} := \min(\{d(\mathbf{x},\mathbf{y})|\mathbf{x},\mathbf{y}\in\mathbf{C}\})$

Um d(**x,y**) bestimmen zu können, benötigen wir einige Eigenschaften der Hammingdistanz.

Satz 2: Es seien **x**,**y**,**z** binäre Worte. Dann gilt:

i) $d(\mathbf{x},\mathbf{y}) \geq 0$ und $d(\mathbf{x},\mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$ (Positivität)

ii) $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$ (Symmetrie)

iii) $d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{z}) + d(\mathbf{z},\mathbf{y}) \Leftrightarrow d(\mathbf{x},\mathbf{z}) \geq d(\mathbf{x},\mathbf{y}) - d(\mathbf{z},\mathbf{y})$ (Dreiecksungleichung)

Beweis: i) und ii) gelten offensichtlich.
zu iii): Sei d(**x**,**y**)=r, d.h., dass sich **x** und **y** an r Stellen unterscheiden. Wir nehmen, dass sie sich an den ersten r Stellen unterscheiden. Die restlichen Stellen seien gleich. **z** unterscheide sich von **x** in den ersten r Stellen an r_1 Stellen und in den restlichen an r_2 Stellen. Dann unterscheidet sich **z** von **y** in den ersten r Stellen an $r-r_1$ Stellen und in den restlichen um r_2 Stellen. Es ist also $d(\mathbf{x},\mathbf{z})=r_1+r_2$ und $d(\mathbf{y},\mathbf{z})=(r-r_1)+r_2$. Damit erhält man
$d(\mathbf{x},\mathbf{z})+ d(\mathbf{y},\mathbf{z})= (r_1+r_2)+((r-r_1)+r_2)=r+2r_2\geq r=d(\mathbf{x},\mathbf{y})$□

Aufgabe 5: Überprüfe die Dreiecksungleichung an: **x** = 100110, **y** = 010111, **z** = 001010

x = 100110 **x** = 100110 **z** = 001010
y = 010111 **z** = 001010 **y** = 010111

d(**x**,**y**) = 3 und d(**x**,**z**) = 3, d(**z**,**y**) = 4, also 3≤ 3 + 4.

Satz 2 zeigt, dass sich die unanschauliche Hammingdistanz d wie ein **anschaulicher Abstand** verhält!

Fehlerbehaftete Worte kann man leichter decodieren, wenn sich in ihrer **Umgebung** nur wenige andere Worte befinden.

Definition: $U_t(\mathbf{x}) := \{\mathbf{y} \in (F_2)^n \mid d(\mathbf{x},\mathbf{y}) \leq t\}$ heißt Umgebung von **x** mit Radius t.

Aufgabe 6: Berechne $U_1(110100)$

	Fehler an					
0 Fehler	1.Stelle	2.Stelle	3.Stelle	4.Stelle	5.Stelle	6.Stelle

$U_1(110100) = \{110100 , 010100, 100100, 111100, 110000, 110110, 110101\}$

Wie muss der Code gestaltet sein, damit Fehler entdeckt werden können?

Satz 3 (**Fehlererkennung**): Ein Code **C** kann genau dann höchstens t Fehler erkennen, wenn seine minimale Hammingdistanz zwischen allen Codeworten mindestens t+1 ist, also $d_{min} \geq (t+1)$.

Beweis: Treten bei der Übertragung des Wortes **c** höchsten t Fehler auf, dann liegt das empfangene Wort **u** in $U_t(\mathbf{c})$, also gilt $d(\mathbf{c},\mathbf{u}) \leq t$. Ist die minimale Hammingdistanz zwischen allen Codeworten größer als t, dann kann **u** kein Codewort sein. Also ist **u** fehlerbehaftet. Wird **u** als ein mit höchstens t Fehlern behaftetes Wort erkannt, dann muss $d_{min} \geq t+1$ sein. Wäre $d_{min} \leq t$, dann könnte **u** ein Codewort sein. □

Aufgabe 7: Gegeben sei der Code (Worte wollen wir ab nun der leichteren Lesbarkeit wegen ohne Beistriche schreiben)

C = {000000, 110100, 011010, 111001, 101110, 001101, 100011, 010111}.

Um seine Fehlererkennungskapazität zu bestimmen, muss man d_{min} berechnen.
Dies erfordert die Berechnung von 28 Hammingdistanzen ($28 = \binom{8}{2}$, „zwei aus acht wählen“). Mit dem Begriff des Hamming-Gewichtes geht es leichter.

Definition: Das **Hamming-Gewicht g(x)** eines Wortes **x ≠ 0** ist die Anzahl der Einser in **x**

Aufgabe 8: Berechne die Hamming-Gewichte des Codes **C** und bestimme einige Codewörter-Abstände.
g(110100)=3, g(111001)=4, g(001101)=3,…
d(110100,101110)=3, d(111001, 100011)=3,…

Satz 4: Sei **C** ein linearer Code. Dann gilt
$d_{min}(\mathbf{C})$ = kleinstes Gewicht seiner Codewörter $\mathbf{c} \neq \mathbf{0}$.

Beweis:
Mit $\mathbf{c_1}$ und $\mathbf{c_2}$ ist auch $\mathbf{c_1} \pm \mathbf{c_2}$ ein Codewort. Auch $\mathbf{0}$ = 000000…0 ist ein Codewort, weil $\mathbf{c_1} - \mathbf{c_1} = \mathbf{0}$. Die Hamming-Distanz d(**c**,**0**) = Anzahl der Einser in **c**. □

Mit Satz 4 können wir nun die Fehler-Erkennungskapazität des Codes
C = {000000, 110100, 011010, 111001, 101110, 001101, 100011, 010111}
bestimmen:

1.Es werde **c**=110100 mit einem Fehler übertragen, es sei **u**= 110110 das empfangene Wort. u liegt nicht in **C,** also muss bei der Übertragung ein Fehler aufgetreten sein, 1 Fehler wird erkannt.
2. Es werde **c**=110100 mit zwei Fehlern übertragen, es sei **u**= 101100 das empfangene Wort. u liegt nicht in **C**, also wird es als fehlerhaft erkannt, 2 Fehler werden erkannt.
3.1. Es werde **c**=110100 mit drei Fehlern übertragen, es sei z.B. **u**= 111001 das empfangene Wort, es liegt in C, wird also nicht als fehlerhaft erkannt.
3.2. Wird **c** als $\mathbf{u_1}$=110011 empfangen, wird es als fehlerhaft erkannt, aber nicht jedes mit 3 Fehlern behaftetes Wort wird als fehlerhaft erkannt (wie 3.1.demonstriert).
3.3. Wird **c** als $\mathbf{u_2}$=010111 empfangen, wird es als nicht fehlerhaft erkannt, weil es in C vorkommt.

Wie muss der Code gestaltet sein, damit Fehler korrigiert werden können?

Kommt bei Übertragung von **c** das Wort **u** an und unterscheidet sich **u** an höchstens t Stellen von **c**, so kann man das ursprüngliche Codewort wiedererkennen, wenn **c** das einzige zulässige Codewort in der Umgebung $U_t(u)$ ist.

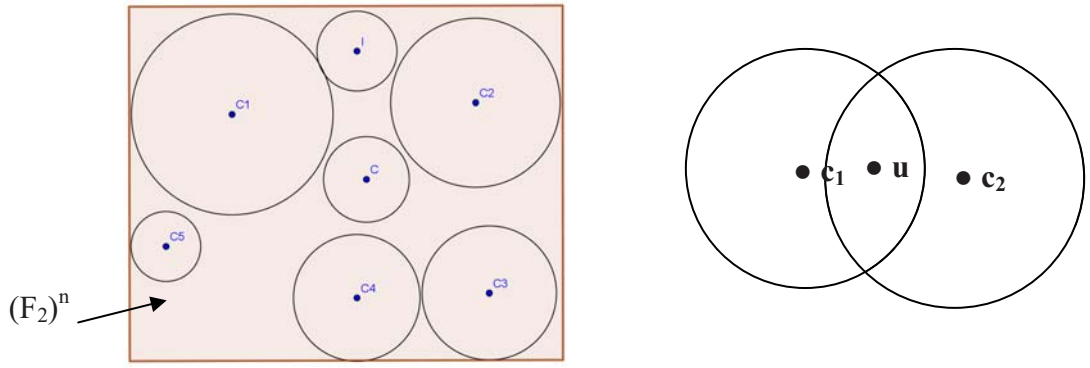


Abbildung 6
Liegt in einer Umgebung vom Radius t um **u** nur **ein** zulässiges Codewort **c**, dann kann man **u** zu **c** decodieren.

Satz 5 (Fehlerkorrektur): Ein Code **C** kann genau dann t oder weniger Fehler korrigieren, wenn $d_{min}(\mathbf{C}) \geq 2t+1$ ist.

Beweis: **c** wird gesendet, **u** empfangen mit höchstens t Fehlern. Man kann das ursprünglich gesendete Wort **c** wieder erkennen, wenn es das **einzige** zulässige Codewort in $U_t(\mathbf{u})$ ist. Wann ist das der Fall? Genau dann, wenn je zwei Codeworte einen Abstand von mindestens 2t+1 besitzen (vlg. Abb. 7): Sind **c** und **c'** zwei zulässige Codewörter, dann gilt nach der Dreiecksungleichung:
$d(c',u)+d(u,c) \geq d(c',c)$ also $d(c',u) \geq d(c',c) - d(c,u)$ und damit
$d(c',u) \geq (2t+1) - t = t+1$. Also: $c' \notin U_t(\mathbf{u})$. Damit ist **c** das einzige Codewort in $U_t(\mathbf{u})$.□

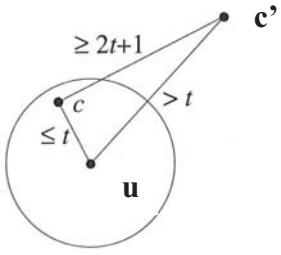


Abbildung 7

Aufgabe 9: Gegeben sei der (6,3)-Code
C = {000000, 110100, 011010, 111001, 101110, 001101, 100011, 010111}.
Entdecke und korrigiere empfangene Wörter mit nur einem Fehler.

Lösung: Wegen $d_{min}(\mathbf{C})=3$ kann **C** nur (3-1)/2 = 1 Fehler korrigieren. Wir berechnen daher nur die Umgebungen vom Radius 1 der 8 Codewörter von **C** :

	1.	2.	3.	4.	5.	6.
$U_1(000000)$ =	{000000,	100000,	010000,	001000,	000100,	000010,	000001}
$U_1(110100)$ =	{110100,	010100,	100100,	111100,	110000,	110110,	110101}
$U_1(011010)$ =	{011010,	111010,	001010,	010010,	011110,	011000,	011011}
$U_1(111001)$ =	{111001,	011001,	101001,	110001,	111101,	111011,	111000}
$U_1(101110)$ =	{101110,	001110,	111110,	100110,	101010,	101100,	101111}
$U_1(001101)$ =	{001101,	101010,	011101,	000101,	001001,	001111,	001100}
$U_1(100011)$ =	{100011,	000011,	110011,	101011,	100101,	100010,	100010}
$U_1(010111)$ =	{010111,	110111,	000111,	011111,	010011,	010101,	010110}

Nehmen wir an, dass bei der Übertragung nur ein Fehler aufgetreten ist. Dann unterscheiden sich die empfangenen Worte **u** an höchstens einer Stelle vom ursprünglichen Wort, also $d(\mathbf{u},\mathbf{c})\leq 1$ oder **u** liegt in $U_1(\mathbf{c})$.
1.Wird $\mathbf{u_2}$=110111 empfangen, so liegt es in U_1(010111) wird also zu 010111 decodiert.
2.Decodiere $\mathbf{u_3}$**= 111110.** Dieses Wort liegt in U_1(101110), das decodierte Wort lautet daher **c=101110.**
3.Decodiere $\mathbf{u_4}$**=001011**. Dieses Wort kommt in keiner Umgebung vor, es kann nicht decodiert werde. Es gilt $d(\mathbf{u_4},\mathbf{c})\geq 2$ für alle Codeworte **c**.
4. Decodiere $\mathbf{u_5}$**=111011.** Unter der Annahme, dass nur 1 Fehler aufgetaucht ist, wird es zu **c=111001** decodiert. Nimmt man an, dass 2 Fehler auftauchen können, dann kann es nicht eindeutig decodiert werden, sowohl **c**=011010 als auch c=100011 sind möglich. Der Code C ist zu schwach, um 2 Fehler korrigieren zu können.

Beachte: Die Vereinigung aller Umgebungen enthält 7*8 = 56 von 2^6 = 64 möglichen Worten der Länge 6, daher gibt es 8 Wörter, die nicht decodiert werden können.
Wegen $t\leq d_{min}-1$ bzw. $t\leq (d_{min}-1)/2$ wird man bestrebt sein, einen Code so zu konstruieren, dass d_{min} möglichst groß wird, damit man möglichst viele Fehler entdecken bzw. korrigieren kann. Andererseits dürfen sich die Umgebungen nicht oder zumindest nur wenig überlappen, aber es sollen auch nur wenige Zwischenräume übrig bleiben. Es ist schwierig, gute Codes zu konstruieren, wegen der leichteren Handhabung sollen sie auch noch linear sein.

2 Maschinelle Codierung – Polynomcodierung

Für große Wortlängen ist das Abspeichern und Suchen von Codewörtern zeitlich zu aufwändig. Wörter können nun leicht durch Polynome dargestellt werden:

Das binäre Wort $a_0a_1\ldots a_{n-1}$ wird durch das Polynom p(x) vom Grad (n-1) mit Koeffizienten aus F_2 dargestellt. Symbolisch: $p(x) := a_0 + a_1x + \ldots a_{n-1}x^{n-1} \in \mathbf{F_2[x]}$

$$\text{Also: } \mathbf{a_0a_1\ldots a_{n-1} \Leftrightarrow a_0 + a_1x + \ldots + a_{n-1}x^{n-1}}$$

Beachte: Das k-Tupel $(a_1,a_2,\ldots,a_k)$ wird in der Form $(a_0,a_1,\ldots,a_{k-1})$ geschrieben, d.h., man beginnt die Plätze der Zählung bei 0. Desgleichen entspricht jedem Codewort $\mathbf{c} = (c_0,c_1,\ldots,c_{n-1})$ ein Polynom $c(x)=c_0+c_1x+\ldots+c_{n-1}x^{n-1}$ über F_2 vom Grad (n-1).

Beispiel: Nachricht 101 = $1 + 0x + 1x^2 = 1+x^2$
Ein Wort der Länge 3 wird also durch ein Polynom vom Grad 2 dargestellt.

Wir erinnern, dass wegen der modularen Arithmetik in F_2 gilt:
$p_1(x)-p_2(x)=p_1(x)+p_2(x)$. Damit kann die Division von Polynomen in F_2 wie folgt durchgeführt werden:

$x^5 + x^3$: $x^3 + x + 1 = x^2$ oder $x^5 + x^3 = (x^3 + x + 1)*(x^2) + x^2$
$\underline{x^5 + x^3 + x^2}$
x^2 Rest r(x)

Wie codiert man nun eine Nachricht $\mathbf{m} \in (F_2)^k$ mit einem Polynom g(x), das in diesem Zusammenhang Generatorpolynom genannt wird?
Die Polynomcodierung eines (n,k)-Codes mit einem Generatorpolynom g(x) vom Grad n-k wollen wir an einem Beispiel mit k=3 und n=6 und mit $\mathbf{g(x)=1+x+x^3}$ demonstrieren:

- Das Nachrichtenwort **m** möge die Länge k haben: **101**
  ⇒ das **Nachrichtenpolynom** m(x) hat den Grad < k: $1+x^2$

- Codeworte **c** soll die Länge n haben und Nachrichtenzeichen m_i sollen am rechten Ende stehen[7].
  ⇒Multiplikation von m(x) mit x^{n-k}: $x^{n-k}\cdot m(x)$: $x^3 + x^5$

- Das Generatorpolynom ist $\mathbf{g(x)=1+x+x^3}$
  Bestimme den Rest r(x) von $x^{n-k}\cdot m(x)$ bei Division durch g(x) (siehe Beispiel oben) x^2

- Addiere den Rest: $r(x)+ x^{n-k}\cdot m(x)$: $x^2 + x^3 + x^5$
  ⇒ **Codepolynom** $c(x)= r(x)+ x^{n-k}\cdot m(x)$ $0+0x+1x^2+1x^3+0x^4+1x^5$

⇒ Codewort **c**: **001101**
C ≅ Koeffizienten des Codepolynoms Prüfzeichen|Nachricht

Bemerkung: Im Gegensatz zu vorhin stehen die Prüfzeichen im Codewort **c** nun **vor** den Nachrichtenzeichen des Nachrichtenwortes **m**. Sie sind genau die Koeffizienten des Restpolynoms r(x).

[7] Dies nennt man shiften von **m** an das rechte Ende des n-Tupels **c**.

Wir halten fest: Prüfziffern = Koeffizienten des Restes.

Die Codierungsfunktion f eines (n,k)-Polynomcodes mit dem Generatorpolynom g(x) vom Grad (n-k) ist eine eindeutige Abbildung aus $(F_2)^k$ in $(F_2)^n$, welche Koeffizienten eines Polynoms vom Grad kleiner gleich (k-1) die Koeffizienten eines Polynoms vom Grad kleiner gleich (n-1) zuordnet.
Symbolisch:

$$(F_2)^k \rightarrow (F_2)^n$$
$$f : F_2[x] \rightarrow F_2[x] \, mit$$
$$f(\mathrm{m}) = (\text{Koeffizienten von } x^{n-k}\ m(x) + r(x))$$
$$\text{mit } r(x) \text{ aus: } x^{n-k}\ m(x) = g(x)\ q(x) + r(x)$$

Satz 6: Die Codewörter eine (n,k)-Polynomcodes sind die Koeffizienten genau jener Polynome vom Grad < n, die durch das Generatorpolynom teilbar sind.

Beweis: Sei $c(x) = r(x) + x^{n-k} \cdot m(x)$ und g(x) das Generatorpolynom.
r(x) war das Restpolynom von $(x^{n-k} \cdot m(x)) : (g(x))$, also
$x^{n-k} \cdot m(x) = q(x) \cdot g(x) + r(x)$ oder $r(x) = x^{n-k} \cdot m(x) - q(x) \cdot g(x)$.

Einsetzen in c(x) ergibt, wenn man berücksichtigt, dass in F_2 gilt: r(x) = -r(x).
$c(x) = r(x) + x^{n-k} \cdot m(x) = -r(x) + x^{n-k} \cdot m(x) =$
$-x^{n-k} \cdot m(x) + q(x) \cdot g(x) + x^{n-k} \cdot m(x) = q(x) \cdot g(x)$.
Also ist g(x) ein Teiler von c(x).□

Zusammenfassend ergibt sich folgende Möglichkeit für eine **Fehler-Entdeckung**:

- Wandle das empfangene Wort in ein Polynom um und dividiere es durch das Generatorpolynom.
- Ist der Rest ≠ **0**, dann ist bei der Übertragung ein Fehler aufgetreten Allerdings kennt man seine Position nicht.

Unser Code **C** ist ein Polynomcode mit dem Generatorpolynom $1+x+x^3$.

Aufgabe 10: Bestimme alle Codeworte des Polynomcodes mit dem Generatorpolynom $g(x) = 1+x+x^3$ für Nachrichtenworte der Länge k=3.

Anleitung: Weil grad(g(x))=3 gibt es 3 Prüfzeichen und weil die Länge der Nachrichtenworte k=3 gilt, werden die Codeworte die Länge n=3+3=6 besitzen. Die Anzahl der Nachrichtenworte ist $2^k=2^3=8$.
Berechne bzw. überprüfe einige Beispiele folgender Tabelle und verwende die Anleitung:

Nachricht			Codewort					
			Prüfzeichen			Nachrichtenzeichen		
0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0
0	1	0	0	1	1	0	1	0
0	0	1	1	1	1	0	0	1
1	1	0	1	0	1	1	1	0
1	0	1	0	0	1	1	0	1
0	1	1	1	0	0	0	1	1
1	1	1	0	1	0	1	1	1
↑	↑	↑	↑	↑	↑	↑	↑	↑
1	x	x^2	1	x	x^2	x^3	x^4	x^5

Tabelle 1

Anleitung: Wir nehmen an, dass die Nachricht 110 lautet. Dieser Nachricht entspricht das Polynom m(x)=1+x. Die Prüfzeichen sind die Koeffizienten des Restpolynoms, das bei Division von $(x^3 \cdot m(x)):g(x) = (x^3 \cdot (1+x)):(1+x+x^3)$ entsteht. Das Codepolynom lautet dann $r(x)+x^3 \cdot m(x) = 1+x^2+x^3+x^4$. Damit ergibt sich das Codewort 101110.

Wir sehen, dass der von $1+x+x^3$ erzeugte Polynomcode gerade der Code **C** aus Aufgabe 7 ist.

Aufgabe 11: Prüfe welche der folgenden empfangenen Worte entdeckbare Fehler enthalten unter Verwendung des (6,3)-Codes erzeugt von $1+x+x^3$:
(i) 100011 (ii) 100110 (iii) 101000

Ein Vorteil der Verwendung von Polynomcodes besteht darin, dass man die Grunddrechnungsarten in $F_2[x]$ mit Hilfe einfacher Schaltungen sehr leicht technisch realisieren kann (shift-Register).
Auch unsere eingangs vorgestellten Codes (Wiederholungs- und Quersummenprüfcode) sind Polynomcodes.

Beispiel: Das Polynom g(x)=1+x erzeugt den (n,n-1)- Quersummenprüfcode.

Beispiel: Das Polynom $g(x)=1+x+x^2$ erzeugt den (3,1) – Wiederholungscode.

Abschließend halten wir fest, dass Polynomcodes lineare Codes sind. Beweis im Anhang.
Dies wird uns die Möglichkeit eröffnen, Fehler auch zu korrigieren.

Beweis: siehe Anhang.

3. Korrigieren von Fehlern mit Polynomcodes

Bisher haben wir Fähigkeit von Polynomcodes zur Entdeckung von Fehlern ausgenutzt. Nun wollen die entdeckten Fehler auch korrigieren. Dabei benützen wir, dass Polynomcodes lineare Codes sind. D.h. nach Satz 1, dass der Code mit je zwei Codeworte auch ihre Summe, Differenz und Vielfache enthält. Der Code besitzt also eine „Struktur“ so wie die natürliche Sprache durch die Grammatik eine Struktur erhält und damit Fehler in der Sprache von uns automatische korrigiert werden. Z.B. wird „Ich gibt dich Geld“ auf Grund der Grammatik zu „Ich gebe dir Geld“ korrigiert. Zunächst werden wir lineare Codes mittels „Matrizen“ darstellen und damit Codeworte und Fehlerworte generieren.

Matrixdarstellung von linearen Codes

So wie wir bei Polynomcodes mit Hilfe eines Generatorpolynomes g(x) die Codeworte erzeugt haben (nach Satz 6 erhielten wie die Codeworte aus Vielfachen des Generatorpolynoms, also c(x)=g(x)·q(x)), wollen wir nun die Codeworte mit Hilfe einer **Generatormatrix G** aus den Nachrichtenworten erzeugen. Wir suchen also eine Matrix G, sodass gilt: G**m** = **c**. Weil $\mathbf{m}\in(F_2)^k$ und $\mathbf{c}\in(F_2)^n$ ist, muss G eine n×k Matrix mit Eintragungen F_2 sein. Wegen der Linearitätseigenschaften muss z.B. für k=3 gelten:

$$\text{Ist } m := \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = m_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + m_2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + m_3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix};$$

$$\text{dann ist } Gm = G\left[m_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + m_2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + m_3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] = m_1 G \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + m_2 G \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + m_3 G \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\text{Nun ist (siehe Anhang) } G \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \text{erste Spalte von G} = \text{Codewort der Nachricht } \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \text{ usw.}$$

Nachrichten der Form $\mathbf{e_i}$=(00…1…0) mit 1 an der i-ten Stelle nennt man i-te Einheitsnachricht.
Wir halten fest:

> Die Spalten der Generatormatrix eines linearen (n,k)-Codes sind die Codeworte der „Einheitsnachrichten“.

Beispiel: Die Generatormatrix des (6,3)-Codes mit Generatorpolynom $g(x)=1+x+x^3$ ist nach Tabelle 1 die Matrix G mit

$$G=\begin{pmatrix}1&0&1\\1&1&1\\0&1&1\\1&0&0\\0&1&0\\0&0&1\end{pmatrix}=\left(\begin{array}{ccc}1&0&1\\1&1&1\\0&1&1\\\hline 1&0&0\\0&1&0\\0&0&1\end{array}\right)=\begin{pmatrix}P\\\hline I_3\end{pmatrix}$$

Wir beobachten: Soll das Codewort **c** eines (n,k)-Codes die Nachrichtenzeichen in den letzten k Positionen des n-Tupels **c** enthalten, dann muss die Generatormatrix G von der Form

$$G=\begin{pmatrix}P\\\hline I_k\end{pmatrix}$$

mit P (n-k)×k Matrix und I_k k×k Einheitsmatrix.

Eine Nachricht **m** wird durch Berechnung von G**m** codiert. Wird nun ein Wort **u** empfangen, so es unter Verwendung der Generatormatrix G nicht leicht festzustellen, ob **u** ein Codewort ist. Man müsste nämlich ein **m** so finden, dass **u**=G**m** gilt. Wir wollen nun eine Matrix H, erzeugt aus G, so bestimmen, dass dies leicht geht.

$$H:=\left(I_{n-k}\,|\,P\right),\ \text{wenn}\ G=\begin{pmatrix}P\\\hline I_k\end{pmatrix}$$

Für diese Matrix H gilt (siehe Anhang)

Satz 8: **c** ist ein Codewort genau dann, wenn H**c** = **0** gilt.
G**m** = **c** ⇔ H**c** = 0

Ein empfangenes Wort ist also genau dann ein Codewort, wenn seine Multiplikation mit H das Nullwort **0** = (00…0) ergibt. H nennt man deshalb **Kontrollmatrix** des (n,k)-Codes.

Zusammenfassung: Ist G = $\begin{pmatrix}P\\\hline I_k\end{pmatrix}$ die Matrix des (n,k)-Codes, dann ist H = $\left(I_{n-k}\,|\,P\right)$ die Kontroll- oder parity-check-Matrix. Eine Nachricht **m** wird durch G**m** codiert. Die Fehlerentdeckung eines empfangenen Wortes **u** erfolgt durch H**u**.

Fehlerkorrektur durch Nebenklassenfindung
Eine einfache Decodierungsregel (mit Fehlerkorrektur) besteht in folgender Vorgehensweise: Wird das Wort **u** empfangen, so suche jenes Codewort **c**, das **u** am nächsten kommt (Nächste-Nachbar-Decodierung bzw. Minimalabstandsdecodierung bzw. MLD). Man müsste also die Hamming-Distanz zwischen **u** und jedem Codewort **c** berechnen. Bei großen Nachrichtenlängen ist dies äußerst aufwendig und nicht praktikabel.

Für lineare Codes, wie z.B. die Polynomcodes, benutzen wir die aus Satz 1 folgende Eigenschaft, dass ein linearer Code **C** eine Untergruppe in $((F_2)^n, \oplus)$ bezüglich der modularen Arithmetik $\oplus$ ist. Unter Verwendung einer Untergruppe ist es möglich, Nebenklassen nach einem Wort **a** zu bilden (siehe Anhang):

$$\mathbf{a}+\mathbf{C}:=\{\mathbf{a}+\mathbf{c} \mid \mathbf{c}\in\mathbf{C}\}$$

Dies ist die Menge jener Worte, die aus **C** durch Addition von **a** zu jedem Codewort **c** entsteht.

Symbolisch können wir die Nächste-Nachbar-Decodierung folgend beschreiben:
u = **c** + **e**, **e** …Fehlerwort mit kleinstem Gewicht, also ein Wort mit möglichst wenig 1er als Eintragungen. Wegen der modularen Arithmetik gilt:

$$\mathbf{u} = \mathbf{c} + \mathbf{e} \Leftrightarrow \mathbf{c} = \mathbf{u} + \mathbf{e} \Leftrightarrow \mathbf{e} = \mathbf{u} + \mathbf{c}.$$

Also liegt das Fehlerwort **e** in der Nebenklasse **u** + **C**: $\mathbf{e}\in \mathbf{u} + \mathbf{C}$.

Allgemeine Decodierungsregel:

Bei Empfang eines Wortes **u** sind die möglichen Fehlerworte **e** genau die Worte in jener Nebenklasse, welche **u** enthält. Der **wahrscheinlichste** Fehler ist der Fehler **e** mit Minimalgewicht in der Nebenklasse **u** + **C**. Wir nennen dieses Fehlerwort $\mathbf{e_{min}}$. Nach der MLD Methode wird **u** zu $\mathbf{c} := \mathbf{u} + \mathbf{e_{min}}$ decodiert.

Bemerkung: Gibt es mehrere Worte mit gleichem Minimalgewicht, so wählt man jenes Fehlerwort, in denen 1er gebündelt auftreten (Fehler treten technisch eher unmittelbar hintereinander auf).

$\mathbf{c_1}$	$\mathbf{c_2}$	$\mathbf{c_3}$	…	$\mathbf{c_j}$	…	$\mathbf{c_k}$
$\mathbf{a_1}+\mathbf{c_1}$	$\mathbf{a_1}+\mathbf{c_2}$	$\mathbf{a_1}+\mathbf{c_3}$	…	$\mathbf{a_1}+\mathbf{c_j}$	…	$\mathbf{a_1}+\mathbf{c_k}$
…	…	…	…	…	…	…
$\mathbf{a_i}+\mathbf{c_1}$	$\mathbf{a_i}+\mathbf{c_2}$	$\mathbf{a_i}+\mathbf{c_3}$	…	$\mathbf{a_i}+\mathbf{c_j}=\mathbf{u}$	…	$\mathbf{a_i}+\mathbf{c_k}$
…	…	…	…	…	…	…
$\mathbf{a_t}+\mathbf{c_1}$	$\mathbf{a_t}+\mathbf{c_2}$	$\mathbf{a_t}+\mathbf{c_3}$	…	$\mathbf{a_t}+\mathbf{c_j}$	…	$\mathbf{a_t}+\mathbf{c_k}$

In der ersten Spalte stehen die **Nebenklassenführer**: Wir wissen, jedes Fehlerwort muss in einer Nebenklasse liegen. Jenes Fehlerwort, das am wahrscheinlichsten auftritt (wahrscheinlichstes Fehlermuster besitzt, also möglichst wenig 1er), wird in die erste Spalte geschrieben. Dies sind die Nebenklassenführer. An der Position Zeile1/Spalte1 steht das Fehlerwort **0**. In den nächsten Zeilen stehen die Fehlerworte mit einem 1er. Jedes empfangene Wort **u** in dieser Nebenklasse enthält einen Fehler (**u** = **c** + **e**). Danach kommen die Zeilen mit zwei 1ern. Am wahrscheinlichsten treten diese beiden 1er hintereinander auf. Also ist 000110 wahrscheinlicher als 101000 oder 010001. usw.

Beispiel: Nebenklassentabelle unseres (6,3)-Codes.

Nebenklassen-führer	Worte						
000000	110100	011010	111001	101110	001101	100011	010111
100000	010100	111010	011001	001110	101101	000011	110111
010000	100100	001010	101001	111110	011101	110011	000111
001000	111100	010010	110001	100110	000101	101011	011111
000100	110000	011110	111101	101010	001001	100111	010011
000010	110110	011000	111011	101100	001111	100001	010101
000001	110101	011011	111000	101111	001100	100010	010110
000110	110010	011100	111111	101000	001011	100101	010001

Algorithmus für das Decodieren mit elementarer Suche:

1. Das Wort **u** wird empfangen. Stelle fest, in welcher Nebenklasse $\mathbf{a}_i+\mathbf{C}$ liegt.
2. Angenommen, $\mathbf{u} = \mathbf{a}_i+\mathbf{c}_j$. Dann ist der Fehlervektor **e** = Nebenklassenführer $\mathbf{a}_i$.
3. Das am wahrscheinlichsten gesendete Wort (decodierte Wort von **u**):
   $\mathbf{c} = \mathbf{u}+\mathbf{e} = \mathbf{u}+\mathbf{a}_i = \mathbf{a}_i+\mathbf{c}_j+\mathbf{a}_i = \mathbf{c}_j+2\mathbf{a}_i = \mathbf{c}_j$.
4. Also: Decodiertes Wort = erstes Element in der Spalte mit **u**.

Beispiel: Decodiere das Wort 111101 mit Hilfe des (6,3)-Codes.

Nebenklassen-führer	Worte						
000000	110100	011010	111001	101110	001101	100011	010111
100000	010100	111010	011001	001110	101101	000011	110111
010000	100100	001010	101001	111110	011101	110011	000111
001000	111100	010010	110001	100110	000101	101011	011111
000100	110000	011110	111101	101010	001001	100111	010011
000010	110110	011000	111011	101100	001111	100001	010101
000001	110101	011011	111000	101111	001100	100010	010110
000110	110010	011100	111111	101000	001011	100101	010001

Das Wort 111001 wird decodiert zu 111**001**. Die ursprüngliche Nachricht lautet also **001**.

Aufgabe 12: Decodiere das Wort 101100.-->**10**1110, ursprüngliche Nachricht 100
Beispiel: Sende das Wort 110. Nimm an, dass bei der Übertragung an der zweiten Stelle ein Fehler auftritt. Wie lautet das empfangene Wort **u**? Decodiere dieses Wort **u**.

Lösung: $\mathbf{c} = G\mathbf{m} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \mathbf{u}$

Dieses empfangene Wort u liegt an der Position 3.Zeile/4.Spalte. Es wird zum ersten Element der 4. Spalte = 101110 decodiert. Also war 110 das gesendete Wort. Stimmt!

Wie diese Aufgaben zeigen, ist die Suche nach der Nebenklasse aufwendig. Darüber hinaus ist für großes (n,k) die Nebenklassentabelle nicht sinnvoll speicherbar. Ein binärer linearer (50,20)-Code besitzt $2^{n-k}=2^{50-20}=2^{30}\approx 10^9$ Nebenklassen. Wir wollen nun zeigen, wie man die Nebenklasse, in der das empfangene Wort **u** liegt, durch Berechnung bestimmen kann.

Nebenklassenfindung durch Berechnung der Syndrome

In der Medizin: Syndrom ist die Menge von Symptomen, die zur Diagnose einer Krankheit herangezogen werden.
In der Codierungstheorie: Der Krankheit entspricht das Fehlerwort. Die Symptome entsprechen den verletzten Kontrollgleichungen H**u**, wobei H die Kontrollmatrix des Codes ist.

Definition: Sei H die Kontrollmatrix des linearen Codes **C**.

$S(\mathbf{u}) := H\mathbf{u} \in (F_2)^{n-k}$ heißt **Syndrom von u**.

Satz 9: $S(\mathbf{u})=\mathbf{0} \Leftrightarrow \mathbf{u}\in\mathbf{C}$.
D.h.: Nur die Codeworte besitzen das Syndrom **0**, also keine verletzten Kontrollgleichungen.

Beweis:

1.) $S(\mathbf{u})=\mathbf{0} \Rightarrow H\mathbf{u}=\mathbf{0} \Rightarrow$ (Satz 8) $G\mathbf{m}=\mathbf{u} \Rightarrow \mathbf{u}\in\mathbf{C}$.
2.) $\mathbf{u}\in\mathbf{C} \Rightarrow \exists\, \mathbf{m}$: $G\mathbf{m}=\mathbf{u} \Rightarrow$ (Satz 8) $H\mathbf{u}=\mathbf{0}$.□

Damit können wir zeigen, dass alle Worte einer Nebenklasse $\mathbf{a}_i+\mathbf{C}$ dasselbe Syndrom besitzen.

Satz 10: $S(\mathbf{u_1})= S(\mathbf{u_2}) \Leftrightarrow \mathbf{u}_1+\mathbf{C}= \mathbf{u}_2+\mathbf{C}$
Jede Nebenklasse ist durch ihr Syndrom charakterisiert.

Beweis: $S(\mathbf{u_1})= S(\mathbf{u_2}) \Leftrightarrow H\mathbf{u_1}= H\mathbf{u_2} \Leftrightarrow H\mathbf{u_1}- H\mathbf{u_2}=\mathbf{0} \Leftrightarrow H(\mathbf{u_1}- \mathbf{u_2})=\mathbf{0} \Leftrightarrow$(Satz oben) $(\mathbf{u_1}- \mathbf{u_2}) \in\mathbf{C} \Leftrightarrow \mathbf{u}_1+\mathbf{C}= \mathbf{u}_2+\mathbf{C}$. Dabei haben wir benützt: Zwei Nebenklassen nach **C** sind genau dann gleich, wenn die Differenz ihrer Vertreter in **C** liegt.

Beispiel: Berechnung der Syndrome unseres (6,3)-Codes.

Syndrom	Nebenklassen-führer	Worte						
000	000000	110100	011010	111001	101110	001101	100011	010111
100	100000	010100	111010	011001	001110	101101	000011	110111
010	010000	100100	001010	101001	111110	011101	110011	000111
001	001000	111100	010010	110001	100110	000101	101011	011111
110	000100	110000	011110	111101	101010	001001	100111	010011
011	000010	110110	011000	111011	101100	001111	100001	010101
111	000001	110101	011011	111000	101111	001100	100010	010110
101	000110	110010	011100	111111	101000	001011	100101	010001

Satz 11: Das empfangene Wort und sein Fehler liegen in derselben Nebenklasse.

Beweis: $\mathbf{u}=\mathbf{c}+\mathbf{e} \Rightarrow S(\mathbf{u})=H\mathbf{u}=H(\mathbf{c}+\mathbf{e})=H\mathbf{c}+H\mathbf{e}=\mathbf{0}+H\mathbf{e}=H\mathbf{e}=S(\mathbf{e})$. Also besitzen **u** dassselbe Syndrom, liegen nach Satz von oben in derselben Nebenklasse.□

Algorithmus für das Decodieren mit Syndromen:

1. Das Wort **u** werde empfangen. Berechne das Syndrom S(**u**).
2. Bestimme den Nebenklassenführer **e** mit dem Syndrom S(**u**).
3. Dann wurde höchstwahrscheinlich **c**=**u**+**e** gesendet.

Beispiel: Decodiere das empfangene Wort **u**=111110 mit Hilfe seines Syndroms.

Lösung: $S(\mathbf{u})=H\mathbf{u}=\begin{pmatrix}1&0&0&1&0&1\\0&1&0&1&1&1\\0&0&1&0&1&1\end{pmatrix}\begin{pmatrix}1\\1\\1\\1\\1\\0\end{pmatrix}=\begin{pmatrix}0\\1\\0\end{pmatrix}$

Der zugehörige Nebenklassenführer lautet **e**=010000. Also wurde **c**=**u**+**e**=111110+010000=101110 gesendet. Die ursprüngliche Nachricht war 110. Stimmt!

BCH Codes

Wie wir gesehen haben, war die Fehlerkorrektur sehr aufwendig und die Speicherung der 2^n Nebenklassen umfangreich. Um 1960 haben Hocquenghem, Bose und Chaudhari die Konstruktion eines Generatorpolynoms g(x) angegeben, mit dem man t oder weniger Fehler korrigieren kann. Diese Codes nennt man nach ihren Erfindern BCH-Codes. Die Fehler in speziellen BCH-Codes können durch algebraische Methoden ohne Speicherung einer Tabelle von Syndromen und Nebenklassenführern entdeckt und korrigiert werden. Es gibt effiziente Algorithmen, mit denen solche „Fehlerpolynome“ $\Sigma e_i X^i$ berechnet werden können, deren X-Potenzen die Lage der Fehler und die Koeffizienten e_i die Größe der Fehler angeben (vgl. Jungnickel, S.158). Diese Codes nennt man Reed-Solomon Codes (RS-Codes).
Um die BCH-Codes bzw. das entsprechende Generatorpolynom konstruieren zu können, weicht man auf endliche Körper mit 2^m Elementen aus. In solchen Körpern, die man mit $GF(2^m)$, Galois-Feld 2^m, bezeichnet, lassen sich alle Elemente ungleich 0 als Potenz eines einzigen Elementes α darstellen. In diesem Zusammenhang nennt man α **primitives** Element des endlichen Körpers $GF(2^m)$[8]. Dieses primitive Element α ist Nullstelle eines irreduziblen Polynomes aus $F_2[x]$, erfüllt also eine Beziehung m-ten Grades: $\alpha^m+a^1\alpha^{m-1}+\ldots+a^0=0$. Für große m ist die Bestimmung irreduzibler Polynome sehr schwierig.

Codierung auf einer Musik CD

Beim Codieren und Decodieren verwendet man RS (255,5)-Codes über GF(256) mit dem Generatorpolynom $g(x)= (x-\alpha)(x-\alpha)^2(x-\alpha)^3(x-\alpha)^4$, wobei α jenes primitive Element von GF(256) ist, das die Beziehung $1+\alpha^2+\alpha^3+\alpha^4+\alpha^8=0$ erfüllt.

[8] In diesem Zusammenhang gilt: $F_2=GF(2^1)=GF(2)$.

Zur Wiedergabe der Musik genügt allerdings nicht nur die korrigierte (0,1)-Folge, sondern auch deren Umwandlung in ein Audio Signal.
Die Musik auf einer CD wird in Digitalform als eine 5km lange spiralförmige Spur aus Vertiefungen (pits) und normalen Zonen (lands) aufgezeichnet.

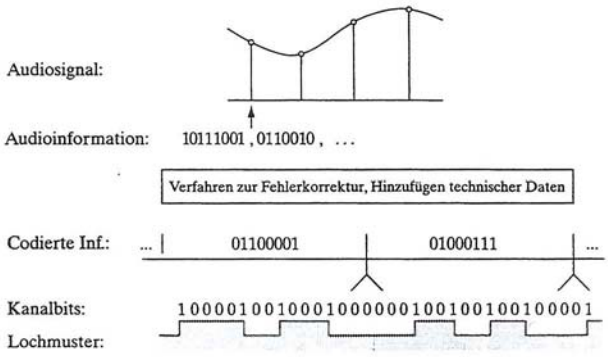


Abbildung 8 (vgl. Dorninger, 1996, S. 92)

Bei der Quellencodierung wird das Audiosignal 44100 mal pro Sekunde abgetastet. Der bei Abtasten gefundene Wert wird in eines von 2^{16}=65536 Niveaus eingeordnet.
Die CD wird beim Abspielen durch einen Laserstrahl ausgelesen. Fällt der Lichtstrahl auf eine Erhebung, so wird das Licht fast völlig reflektiert und der Strahl erreicht in fas voller Stärke eine Photodiode. Fällt der Lichtstrahl in eine Vertiefung (Tiefe ≈ ¼ der Wellenlänge des Lichts), so wird er nur sehr wenig reflektiert. Jedes mal, wenn ein Wechsel zwischen einer Vertiefung und einer Erhebung stattfindet, d.h., wenn von der Photodiode ein Wechsel zwischen sehr starker und geringfügiger Reflexion registriert wird, wird eine Eins angenommen, dazwischen Nullen. Auf den dabei entstehenden Datenstrom wird die Codierung und Decodierung mittels RS-Codes angewandt. Schließlich wird der digitale Informationsstrom wieder in Audiosignale umgewandelt.

Literatur:

Dorninger, D. (1996). Algebraische Codierungstheorie und Comapct Disc. *Elemente der Mathematik, 51*(3), 89-101.

Gilbert, W. J. (1976). *Modern Algebra with Applications*. New York, London, Sydney, Toronto: John Wiley & Sons.

Jungnickel, D. (1995). Codierungstheorie. Heidelberg, Berlin,Oxford:Spektrum,Akademischer Verlag

Anhang:

Matrizen

Matrizen sind rechteckige Zahlenschemata, welche in beinahe allen Bereichen der Mathematik Anwendung finden. Eine Aufzählung der entsprechenden Einsatzgebiete wäre wohl zu lang. Wir begnügen uns mit wenigen Beispielen und geben dann die Definition einer Matrix über einem Körper.

$$A=\begin{pmatrix}3 & 0\\ -1 & 2\end{pmatrix};E=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix};B=\begin{pmatrix}a & n & l & u\\ b & g & k & j\end{pmatrix}$$

$$C=\begin{pmatrix}3 & \sqrt{2} & 1/5 & -\pi\end{pmatrix};D=\begin{pmatrix}\alpha\\ \beta\\ \gamma\end{pmatrix}$$

Aus diesen Beispielen lesen wir die ersten definierenden Eigenschaften von Matrizen: Matrizen besitzen Zeilen und Spalten: A ist eine 2 Zeilen/2 Spalten Matrix, C ist eine 1 Zeilem/4 Spalten Matrix. Besitzt eine Matrix gleich viele Zeilen wie Spalten, so nennen wir sie quadratisch. Quadratische Matrizen, welche in der Diagonale Einser und sonst nur Nullen als Einträge besitzen, nennt man Einheitsmatrizen. Die Matrix C nennen wir auch Zeilenvektor, die Matrix D ist ein Spaltenvektor.

Definition: Sei ein Körper K gegeben (z.B. K=R) Unter einer Matrix A mit m Zeilen und n Spalten über K versteht man das Zahlenschema der Form:

$$A=\begin{pmatrix}a_{11} & a_{12} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n}\\ \cdots & \cdots & \cdots & \cdots\\ a_{m1} & a_{m2} & \cdots & a_{mn}\end{pmatrix}$$

$a_{ij} \in K$ mit 1≤i≤m und 1≤j≤n ist ein beliebiger Eintrag an der Stelle (i,j). Das Symbol a_{ij} verwendet man auch, um die Matrix A zu bezeichnen. Man schreibt $A=(a_{ij})$.
Das Zahlenpaar (m,n) nennt man die Dimension der Matrix A.

Rechnen mit Matrizen

Es seien $A=(a_{ij})$ und $B=(b_{ij})$ zwei Matrizen gegeben (jeweils gleiche Anzahl von Zeilen bzw. Spalten). Wir definieren:

$$A+B=(a_{ij})+(b_{ij}):=\begin{pmatrix}a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n}\\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n}\\ \cdots & \cdots & \cdots & \cdots\\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn}\end{pmatrix}$$

Beachten wir diese Definition, so können wir mit Matrizen bezüglich + rechnen, so als ob sie „Zahlen“ eines Körpers wären. Leicht prüft man nach, dass Matrizen gleicher Dimension bezüglich der Addition eine kommutative Gruppe bilden.

Die Multiplikation einer Matrix $A = (a_{ij})$ mit einem Skalar k aus dem zugrunde gelegten Körper K definiert man folgend:

$$k\,A := \begin{pmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{pmatrix}$$

Etwas aufwendiger gestaltet sich die Matrizenmultiplikation. Wir wollen sie zuerst an Beispielen demonstrieren und dann einen Definition geben.

Beispiel: Erinnern wir uns an das skalare Produkt zweier Vektoren:

$$(a_1, a_2, \ldots, a_n)\begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{pmatrix} := a_1b_1 + a_2b_2 + \ldots + a_nb_n = \sum_{k=1}^{n} a_ib_i$$

$$(8, -4, 5)\begin{pmatrix} 3 \\ 2 \\ -1 \end{pmatrix} = 24 - 8 - 5 = 11$$

Wir wollen diese Vorgangsweise nun verwenden, um zwei Matrizen zu multiplizieren, die nicht nur eine Zeile bzw. eine Spalte besitzen.

Beispiel:
$A = \begin{pmatrix} 1 & 3 \\ 2 & -1 \end{pmatrix}, B = \begin{pmatrix} 2 & 0 & -4 \\ 3 & -2 & 6 \end{pmatrix}$; nehmen wir die erste Zeile der Matrix A und deuten ihn als Zeilenvektor. Diese Zeile besitzt gleich viele Komponenten wie die Spalten der Matrix B. Nennen wir die Produktmatrix *C*, also $C := A\,B$. Gehen wir dem skalaren Produkte zweier Vektoren vor. Wir berechnen die erste Zeile von C. Die Eintragungen sind jeweils das Skalarprodukt der ersten Zeile von A mit den jeweiligen Spalten von B:

$$C = A\,B = \begin{pmatrix} 1 & 3 \\ 2 & -1 \end{pmatrix}\begin{pmatrix} 2 & 0 & -4 \\ 3 & -2 & 6 \end{pmatrix} := \begin{pmatrix} 1\cdot 2 + 3\cdot 3 & 1\cdot 0 + 3\cdot(-2) & 1\cdot(-4) + 3\cdot 6 \\ \ldots & \ldots & \ldots \end{pmatrix} = \begin{pmatrix} 11 & -6 & 14 \\ \ldots & \ldots & \ldots \end{pmatrix}$$

Nun die zweite Zeile von C:

$$C = A\,B = \begin{pmatrix} 1 & 3 \\ 2 & -1 \end{pmatrix}\begin{pmatrix} 2 & 0 & -4 \\ 3 & -2 & 6 \end{pmatrix} := \begin{pmatrix} 11 & -6 & 14 \\ 2\cdot 2 + (-1)\cdot 3 & 2\cdot 0 + (-1)\cdot(-2) & 2\cdot(-4) + (-1)\cdot 6 \end{pmatrix} = \begin{pmatrix} 11 & -6 & 14 \\ 1 & 2 & -14 \end{pmatrix}$$

Man bildet wieder das skalare Produkt der zweiten Zeile von *A* wird mit den jeweiligen Spalten von *B*. Beachte, dass die Matrix *C* zwei Zeilen und drei Spalten besitzt. Das Produkt einer 2×2 Matrix (sprich: 2 kreuz 2 Matrix) mit einer 2×3 Matrix ergibt eine 2×3 Matrix. Beachte: Die Anzahl der Spalten von *A* entspricht der Anzahl der Zeilen von *B*.

Definition: Es seien A eine m×p Matrix und B eine p×n Matrix über einem Körper K. Unter dem Produkt AB versteht man die m×n Matrix C, mit:

$$A\,B=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mp} \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_{p1} & b_{p2} & \cdots & b_{pn} \end{pmatrix}:=\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{pmatrix}=:C$$

$$\text{mit } c_{ij}:=a_{i1}b_{1j}+a_{i2}b_{2j}+\ldots+a_{ip}b_{pj}=\sum_{k=1}^{p}a_{ik}b_{kj}$$

Beispiel:

a.) Prüfe, ob folgendes Matrizenprodukt richtig ist.

$$\begin{pmatrix} 2 & 3 & -1 \\ 4 & -2 & 5 \end{pmatrix}\begin{pmatrix} 2 & -1 & 0 & 6 \\ 1 & 3 & -5 & 1 \\ 4 & 1 & -2 & 2 \end{pmatrix}=\begin{pmatrix} 3 & 6 & -13 & 13 \\ 26 & -5 & 0 & 32 \end{pmatrix}$$

b.) Berechne folgende Produkte, sofern definiert:

$$\begin{pmatrix} 1 & 6 \\ -3 & 5 \end{pmatrix}\begin{pmatrix} 2 \\ -7 \end{pmatrix};\begin{pmatrix} 2 \\ -7 \end{pmatrix}\begin{pmatrix} 1 & 6 \\ -3 & 5 \end{pmatrix};\begin{pmatrix} 2 & -7 \end{pmatrix}\begin{pmatrix} 1 & 6 \\ -3 & 5 \end{pmatrix},$$

Ein besonderer Fall: Wir multiplizieren die Matrix A mit einem Einheitsvektor. Die Produktmatrix ist die jeweilige Spalte der Matrix A.

$$A=\begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix};B=\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix};C=\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$AB=\begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}=\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$AC=\begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}=\begin{pmatrix} g \\ h \\ i \end{pmatrix}$$

Allgemein: Sei E_i Einheitsvektor mit 1 an der i-ten Stelle sonst 0 und sei die Multiplikation mit der Matrix M möglich:

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mp} \end{pmatrix};E_i=\begin{pmatrix} 0 \\ \cdots \\ 0 \\ 1 \\ 0 \\ \cdots \\ 0 \end{pmatrix}\quad AE_i=\begin{pmatrix} a_{1i} \\ a_{2i} \\ \cdots \\ a_{mi} \end{pmatrix}$$

Multiplikation von Blockmatrizen:

$$A=\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right); B=\left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right)$$

A und B seien in Untermatrizen zerlegt. Dabei setzen wir
voraus, dass die entsprechenden Matrizenprodukte definiert sind. Dann folgt aus
den Rechenregeln für das Multiplizieren von Matrizen (hier für 2×2 Blockmatrizen angegeben):

$$AB=\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right)\left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right)=\left(\begin{array}{c|c} A_{11}B_{11}+A_{12}B_{21} & A_{11}B_{12}+A_{12}B_{22} \\ \hline A_{21}B_{11}+A_{22}B_{21} & A_{21}B_{12}+A_{22}B_{22} \end{array}\right)$$

Beispiel: Berechne das Produkt!

$$A=\left(\begin{array}{cc|cc} -1 & 2 & 1 & 5 \\ 0 & -3 & 4 & 2 \\ \hline 1 & 5 & 6 & 1 \end{array}\right); B=\left(\begin{array}{cc|c} 2 & 1 & 4 \\ -3 & 5 & 2 \\ \hline 7 & -1 & 5 \\ 0 & 3 & -3 \end{array}\right)$$

Für die Kontroll- Matrix H gilt: siehe S:19

Satz 8: **c** ist ein Codewort genau dann, wenn H**c** = **0** gilt.
G**m** = **c** $\Leftrightarrow$ H**c** = 0

Beweis:

Aus $G\mathrm{m}=\left(\dfrac{P}{I_k}\right)\mathrm{m}=\left(\dfrac{P\mathrm{m}}{I_k\mathrm{m}}\right)=\left(\dfrac{P\mathrm{m}}{\mathrm{m}}\right)=\mathrm{c}=\begin{pmatrix} c_0 \\ \vdots \\ \dfrac{c_{n-k}}{c_{n-k+1}} \\ \vdots \\ c_n \end{pmatrix}$ folgt

$\begin{pmatrix} c_0 \\ \vdots \\ c_{n-k} \end{pmatrix}=P\mathrm{m}$ und $\begin{pmatrix} c_{n-k+1} \\ \vdots \\ c_n \end{pmatrix}=\mathrm{m}$.

Damit ist $\mathrm{H}\mathbf{c}=\left(I_{n-k}\,|\,P\right)\begin{pmatrix} c_0 \\ \vdots \\ \dfrac{c_{n-k}}{c_{n-k+1}} \\ \vdots \\ c_n \end{pmatrix}=I_{n-k}\begin{pmatrix} c_0 \\ \vdots \\ c_{n-k} \end{pmatrix}+P\begin{pmatrix} c_{n-k+1} \\ \vdots \\ c_n \end{pmatrix}=\begin{pmatrix} c_0 \\ \vdots \\ c_{n-k} \end{pmatrix}+P\mathrm{m}=P\mathrm{m}+P\mathrm{m}=0$.

Das letzte „=“ gilt wegen der modularen Arithmetik.

Nun zeigen wir die Umkehrung: Aus H**c** = **0** folgt G**m** = **c**.

Aus H**c**=**0** folgt: $\left(I_{n-k} \mid P\right)\begin{pmatrix} c_0 \\ \vdots \\ c_{n-k} \\ c_{n-k+1} \\ \vdots \\ c_n \end{pmatrix} = I_{n-k}\begin{pmatrix} c_0 \\ \vdots \\ c_{n-k} \end{pmatrix} + P\begin{pmatrix} c_{n-k+1} \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_{n-k} \end{pmatrix} + P\mathrm{m} = 0$

Also gilt wegen der modularen Arithmetik P**m**=$\begin{pmatrix} c_0 \\ \vdots \\ c_{n-k} \end{pmatrix}$ und damit folgt

G**m**=$\left(\frac{P}{I_k}\right)\mathrm{m} = \left(\frac{P\mathrm{m}}{I_k\mathrm{m}}\right) = \left(\frac{P\mathrm{m}}{\mathrm{m}}\right) = \begin{pmatrix} c_0 \\ \vdots \\ c_{n-k} \\ c_{n-k+1} \\ \vdots \\ c_n \end{pmatrix} = \mathbf{c}$. □

Rechnen mit Polynomen

In der Menge der ganzen Zahlen können wir die Division mit Rest durchführen. Wir wollen nun Polynome, wie sie auch im Unterricht besprochen werden, dividieren. Dass dies möglich ist, sichert die Algebra. Wir halten einleitend folgende Eigenschaften fest:

1. Besitzt das Polynom $f \in K[x]$ (verkürzend gesprochen: f ist ein Polynom über dem Körper K in der Variablen x) eine Nullstelle x_0, so ist f ohne Rest durch $(x-x_0)$ teilbar. $f(x)=g(x)(x-x_0)$.
2. Zwei Polynome sind gleich, falls sie in allen Koeffizienten übereinstimmen.
3. Unter dem Grad eines Polynoms verstehen wir die höchste nicht verschwindende Potenz.
4. Es sei K ein Körper. Dann heißt ein Polynom p(x) aus K[x] in der Variablen x irreduzibel, wenn p(x) nicht konstant ist und es keine nichtkonstanten Polynome q(x), r(x) aus K[x] gibt, so dass $p(x)=q(x)\cdot r(x)$ gilt. Falls solche Polynome existieren, so heißt p(x) auch reduzibel oder zerlegbar.
5. Ein vom Nullpolynom verschiedenes Polynom f vom Grad n besitzt höchstens n Nullstellen
6. Besitzt ein Polynom $f \in K[x]$ vom Grad n über dem Körper K genau n Nullstellen, so kann das Polynom als Produkt seiner Linearfaktoren dargestellt werden.

Dividieren wir nun Polynome über den reellen Zahlen. Dabei gehen wir analog der Division in den ganzen Zahlen vor[9]:

	$(2x^5+5x^3+x^2+3x+1)$: $(2x^2+1)$ = x^3+2x+1
$-g\,x^3$	$-(2x^5+x^3)$
$f_1:$	$4x^3+x^2+3x+1$
$-g\,2x$	$-(4x^3+2x)$
$f_2:$	x^2+x+1
$-g\,\frac{1}{2}$	$-(x^2+\frac{1}{2})$
$f_3:$	$x+\frac{1}{2}$

Also: $2x^5+5x^3+x^2+3x+1=(2x^2+1)(x^3+2x+1)+(x+1/2)$

In endlichen Körpern z.B. in Z_3 gehen wir in gleicher Weise vor. Notieren wir zuerst die Verknüpfungstafeln.

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

.	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Nun dividieren wir unter Verwendung dieser Tafeln die folgenden Polynome aus $Z_3[X]$:

[9] Vgl. Hartmann, 2006, S. 88ff

$$
\begin{array}{lll}
(2x^4+2x^2+x+1) & : & (x+2) \quad = \quad 2x^3+2x^2+x+2 \\
\quad -(2x^4+x^3) & & \\
\hline
\quad 2x^3+2x^2+x+1 & & \\
\quad -(2x^3+x^2) & & \\
\hline
\qquad x^2+x+1 & & \\
\quad -(x^2+2x) & & \\
\hline
\qquad 2x+1 & & \\
\quad -(2x+1) & &
\end{array}
$$

Beachte, dass wir in einem endlichen Körper rechnen! Hier gilt:

$x\,2x^2 = 2x^3, 2\,(2x^3) = (2\,2x^3) = 1x^3$

$0x^3 - 1x^3 = 0x^3 + 2x^3 = 2x^3$

$x - 2x = x + x = 2x$

Zuletzt halten wir fest: Sind p(x) und q(x) zwei Polynome. Aus dem Euklidischen Algorithmus folgt, dass der bei der Division p(x):q(x) das Restpolynom eindeutig bestimmt ist.

Zu S.17:

Satz 7: Sei g(x) ein Polynom vom Grad (n-k). Dann ist der von g(x) erzeugte binäre (n,k)-Code ein linearer Code.

Beweis: Sei $f:(F_2)^k \longrightarrow (F_2)^n$ die Codierungsfunktion, die durch das Generatorpolynom g(x) bestimmt ist. Sind $m_1(x)$ und $m_2(x)$ die entsprechenden Nachrichtenpolynome der Nachrichten $\mathbf{m}_1$ und $\mathbf{m}_2$ vom Grad < k. Dann sind die Polynome $c_i(x) = r_i(x)+x^{n-k}\cdot m_i(x)$ für i=1,2 die Codepolynome der Codeworte $\mathbf{c}_1$ und $\mathbf{c}_2$. Damit ist $f(\mathbf{m}_1)$= (Koeffizienten von $r_1(x)+x^{n-k}\cdot m_1(x)$) und
$f(\mathbf{m}_2)$= (Koeffizienten von $r_2(x)+x^{n-k}\cdot m_2(x)$).
Wie lautet nun das Bild von $\mathbf{m}_1+\mathbf{m}_2$ unter dieser Abbildung f?
Nach Definition: Die Koeffizienten des Polynoms $r(x)+x^{n-k}\cdot(m_1(x)+m_2(x))$
Also müssen wir das Polynom r(x) bestimmen.
Die folgende Rechnung ergibt dazu:
$c_1(x)+c_2(x)= r_1(x)+x^{n-k}\cdot m_1(x)+ r_2(x)+x^{n-k}\cdot m_2(x)=$
$=r_1(x)+ r_2(x)+ x^{n-k}\cdot[m_1(x)+m_2(x)]$ (*)
Dabei ist der Grad von $r_1(x)+ r_2(x)$ kleiner als (n-k). Wegen der Eindeutigkeit des Restpolynoms ist daher $r_1(x)+ r_2(x)$ das Restpolynom von $(x^{n-k}\cdot m_1(x)+x^{n-k}\cdot m_2(x)):g(x)$. Also ist $r(x)=r_1(x)+ r_2(x)$. Wir erhalten folgende Schlusskette:
$f(\mathbf{m}_1 + \mathbf{m}_2)$ = (Koeffizienten von $r(x)+x^{n-k}\cdot(m_1(x)+m_2(x))$) = (Koeffizienten von $r_1(x)+r_2(x)+x^{n-k}\cdot(m_1(x)+m_2(x))$ =[10] (Koeffizienten von $r_1(x)+x^{n-k}\cdot m_1(x)$) + (Koeffizienten von $r_2(x)+x^{n-k}\cdot m_2(x)$) = $f(\mathbf{m}_1) + f(\mathbf{m}_2)$.
Damit ist die erste Eigenschaft der Linearität gezeigt.
Die zweite Eigenschaft der Linearität $f(\lambda\mathbf{m})= \lambda f(\mathbf{m})$ gilt offensichtlich, weil λ bei binären Codes nur 0 oder 1 sein kann.□

[10] wegen (*)

Etwas über Gruppen

Definition: Sei G eine nicht leere Menge und $\circ$ eine binäre Verknüpfung. Man nennt G zusammen mit $\circ$ eine Gruppe, wenn folgende Axiome erfüllt sind:

1. Es gilt $(a \circ b) \circ c = a \circ (b \circ c)$ für alle a,b,c aus G
2. Es existiert ein e aus G mit $a \circ e = e \circ a = a$ für alle a aus G (Existenz des neutrales Element)
3. Zu jedem a aus G existiert ein a^{-1} aus G mit $a \circ a^{-1} = a^{-1} \circ a = e$ (Existenz des inversen Elementes)

Ist die Verknüpfung kommutativ, so nennt man $(G, \circ)$ eine kommutative (abelsche) Gruppe.

Folgerungen (ohne Beweis): Seien a,b,c aus einer Gruppe G

1. Das neutrale Element einer Gruppe G ist eindeutig bestimmt
2. Zu jedem Element einer Gruppe gibt es nur ein inverses Element.
3. In einer Gruppe gilt die Kürzungsregel: $a \circ b = a \circ c \Rightarrow b = c$
4. Die Gleichungen $a \circ x = b$ und $y \circ a = b$ sind eindeutig lösbar.
5. Das inverse Element eines inversen Elementes ist das Element selbst.
6. $(a \circ b)^{-1} = b^{-1} \circ a^{-1}$

Vereinbarung: mit m aus N schreiben wir: $a^m := a \circ a \circ \ldots \circ a$ („m-malige Verknüpfung“)
sowie $a^{-m} := (a^{-1})^m$
Additiv geschrieben: $m \cdot a = a+a+\ldots+a$ sowie $m(-a) = (-a)+(-a)+\ldots+(-a)$

Definition: Die Anzahl der Elemente einer Gruppe G nennt man die Ordnung von G. Man schreibt dafür: $o(G)=|G|$.

Beispiele: (Z,+) ist von nicht endlicher Ordnung. $(Z_2,+)$ besitzt die Ordnung 2.

Untergruppe

Definition: Sei $(G, \circ)$ eine Gruppe. Eine nicht leere Teilmenge G’ von G nennt man genau dann Untergruppe von G, falls $(G', \circ)$ wieder eine Gruppe ist (bezüglich derselben Verknüpfung!).
Man schreibt: $G' \lhd G$ bzw. $G' \trianglelefteq G$.

Es gilt: Der Durchschnitt zweier Untergruppen ist wieder eine Untergruppe.

Untergruppenkriterien:

Für G’ könnte man zur Überprüfung „ist Untergruppe“ alle Gruppenaxiome für die in Frage kommenden Elemente prüfen, oder Untergruppenkriterien verwenden.

Kriterium 1: Eine nicht leere Teilmenge G’ eine Gruppe G mit der Verknüpfung $\circ$ ist genau dann eine Untergruppe von G, falls folgende Bedingungen erfüllt sind:

- G’ ist bezüglich $\circ$ abgeschlossen

- die inversen Elemente jedes Elementes von G' liegen wieder in G'

Kriterium 2: Eine nicht leere Teilmenge G' einer Gruppe G mit der Verknüpfung$\circ$ ist genau dann eine Untergruppe von G, wenn mit a,b aus G' stets $a^{-1} \circ b$ in G' liegt.

Beispiel:
Ist a aus einer Gruppe G. Dann gilt: G':=$\{a^n | n \text{ aus } Z\}$ ist eine Untergruppe von G. Man sagt: Die Menge aller ganzzahligen Potenzen von a aus G ist eine Untergruppe von G.

Nebenklassen:

Definition: Es sei (G, $\circ$) eine Gruppe und H eine Untergruppe von G. Sei a ein beliebiges Element von G.
Ha:=$\{h \circ a | h \text{ aus } H\}$ nennt man die von a erzeugte Rechtsnebenklasse von H in G.
aH:=$\{a \circ h | h \text{ aus } H\}$ nenne man die von a erzeugte Linksnebenklasse von H in G.

Untersuchen wir einige Eigenschaften am Beispiel der Rechtsnebenklassen. Für Linksnebenklassen gelten diese Eigenschaften analog.

1. Es gilt $a \in Ha$, weil H und G dasselbe neutrale Element e besitzen.
2. Gilt o(H)=m, dann enthält auch Ha m Elemente.
   Denn: Ha kann höchstens m Elemente enthalten. Gäbe es nun in H mehr Elemente als in Ha, dann muss es mindestens zwei Elemente $h_1, h_2 \in H$ geben, mit $h_1 \neq h_2$ und $h_1 \circ a = h_2 \circ a$. Aus der Kürzungsregel folgt der Widerspruch $h_1 = h_2$.
3. Ist C_r die Menge aller Rechtsnebenklassen von H bezüglich G, so liegt auch H in C_r.
   Denn: Mit $a \in H$ folgt Ha=H.
4. Zwei Rechtsnebenklassen Ha und Hb von H bezüglich G und $a \neq b$ sind entweder disjunkt oder identisch.
   Denn: Besitzen Ha und Hb ein gemeinsames Element c, dann gibt es $h_1, h_2 \in H$ mit $c = h_1 \circ a = h_2 \circ b$. Wegen $a = h_1^{-1} \circ (h_2 \circ b) = (h_1^{-1} \circ h_2) \circ b$ und $(h_1^{-1} \circ h_2) \in H$ folgt dann $a \in Hb$. Damit gilt Ha=Hb. Damit ist C_r eine Menge paarweiser disjunkter Rechtsnebenklassen in G und weil jedes $a \in G$ in wenigstens einer Rechtsnebenklasse enthalten ist, ist C_r eine Zerlegung von G. Zerlegung von G bedeutet also: Die Teilmengen von G sind disjunkt und deren Vereinigung ergibt G.

Satz von Lagrange: Es sei G eine endliche Gruppe der Ordnung n. H sei eine Untergruppe der Ordnung m. Dann ist die Ordnung von H ein Teiler der Ordnung von G.

Beweis: Bilden wir die Rechtsnebenklassen von G bezüglich H. Jede enthält genau m=|H| Elemente. Da sie eine Zerlegung von G bilden, folgt:
$n = |G| = |Ha_1 \cup Ha_2 \cup \ldots \cup Ha_r| = |Ha_1| + |Ha_2| + \ldots + |Ha_r| = |H| + \ldots + |H| = r|H| = r \cdot m$. Also gilt die Behauptung.□