FH Frankfurt am Main
University of Applied Sciences
Fachbereich 2
Dr. Ing. Wilfried Dankmeier

Eppstein im Taunus, 24.03.2013

Codierungsverfahren

Versuch eines „Roten Fadens“
Stand: 24.03.2013

Hinweis: Viele der hier genannten Begriffe sind in den begleitenden Unterlagen genauer erläutert, insbesondere auch im „Grundkurs Codierung“. Auf eine nochmalige Detaillierung an dieser Stelle wird daher verzichtet.

Fehlerbeseitigung

1. **Codierungstechnik/Codierungstheorie** ist ein Teilgebiet der Informatik und behandelt die 3 Bereiche Fehlerbeseitigung, Verschlüsselung und Datenkompression.
2. **Fehlerbeseitigung** untersucht die Ursachen für die Entstehung von Fehlerbits in **Informationsströmen** und stellt Verfahren bereit, um diese Fehler so gering wie möglich zu halten.
3. Informationsströme bestehen logisch aus Folgen von 0/1-Bits, virtuell unterteilt in passende Blöcke **u**, z. B. in je ein Byte zu 8 Bits.
4. Zur physikalischen Übertragung auf Kanälen (Leitungen, Funkstrecken) oder zur Speicherung werden die **Info-Bits** in physikalisch darstellbare Zustandsformen **us** abgebildet (= **moduliert**). Ein einfaches Modulationsverfahren ist die Basisband-Modulation, bei der den logischen Bits Spannungs- oder Strompegel zugewiesen werden, etwa 0 → 1 Volt, 1 → -1 Volt.
5. Unvermeidliche physikalische Störeinflüsse auf dem Kanal, die sich dem Info-Strom z. B. als **Rauschen rs** überlagern, verändern den Info-Strom **us** zum Empfangssignal **ws = us + rs**.
6. Bei der **Demodulation** des Empfangssignals mit der Hard Decision-Methode (**HD**) oder auch bei dessen direkter Weiterverwendung in einem Soft Decision-Verfahren (**SD**) können beim Empfänger fehlerhafte Info-Bits zurückbleiben.
7. Ohne Zusatzmaßnahmen ist es im allgemeinen nicht möglich, die fehlerhaften Info-Bits zu identifizieren. Für eine rechentechnische Behandlung kann man das aus der Demodulation entstehende logische Empfangssignal **w = f(ws)** aus dem Info-Strom u und dem hypothetischen Fehler-Strom e als Summe **w = u + e** (immer MOD 2) darstellen. Das hypothetische Fehlersignal e hat dann an den - bis hierhin noch unbekannten - Fehlerpositionen den logischen Wert 1, an den unverfälschten Stellen den Wert 0, z. B. **e = 0001000**.
8. Grundlage für die Fehlerbeseitigung ist die Berechnung geeigneter **Prüf-Bits y** zu gegebenen Info-Bits, beide Anteile bilden bei systematischen Codes (siehe Punkt 34) die **Codewörter v = (u | y)**. Damit enthalten die Codewörter gezielt **redundante Anteile** zu den Info-Bits, mit deren Hilfe die fehlerhaften Bits eventuell rekonstruiert werden können.
9. Das Empfangssignal ist nun die Summe aus Sendesignal vs und Rauchen rs, also **ws = vs + rs**.
10. Damit man sich nicht in abstrakten Verfahren verliert, ist es hilfreich, Kriterien einzubeziehen, mit denen sich die Eignung der vielen möglichen Wege zur Fehlerbeseitigung beurteilen und vergleichen lassen. Ein nützliches Kriterium bietet die Betrachtung der **Restfehlerrate p_{err}** gegen die Fehlerrate bei uncodierten Informationsströmen. Für einen „fairen“ Vergleich ist es angebracht, nicht die Code-Bit-Geschwindigkeiten (also Info-Bits **und** Prüf-Bits), sondern die **Info-Bit-Geschwindigkeiten** zu betrachten, da der Empfänger nur an diesen interessiert ist.
11. Die **Fehlerrate** wird durch das **Kanalrauschen**, genauer durch das Verhältnis der Nutzsignalleistung zur Rauschleistung bestimmt. Dies ist der **Störabstand SNR**. Je kleiner SNR, umso stärker die Fehlerrate.
12. Für das Hard Decision-Demodulationsverfahren (**HD-Verfahren**) bei bipolaren Sendesignalen gibt für mittelwertfreie, **normalverteilte** Rauschsignale das **Gauss'sche Fehlerintegral** die Fehlerrate für einen uncodierten Info-Bitstrom an.

13. Die Fehlerrate (Punkt 11) wird meistens in einem Diagramm als Funktion von SNR dargestellt. Da für die praktische Verwendung von Codes nur deren Verhalten bei sehr geringen Restfehlerraten (Punkt 10) - das bedeutet $p_{err} << 10^{-6}$ - interessant ist, benutzt man für die Darstellung **logarithmische Achsen**, um diese kleinwertigen Bereiche deutlicher erkennen zu können.
14. Ein wesentlicher Einflussparameter ist die **Informationsrate R = k/n** mit k als Anzahl der Info-Bits im Codewort v, und n als Gesamtstellenzahl von v. Der Wert von R liegt zwischen 0 und 1.
15. Damit die Infobits verschiedener Codes mit gleicher Geschwindigkeit übertragen werden, muss man die Geschwindigkeit für die Codewort-Bits um 1/R anheben. Dazu ist die **Kanalbandbreite B** ebenfalls um 1/R zu vergrößern, was wiederum die Rauschleistung um 1/R erhöht und damit der Störabstand auf den schlechteren Wert SNR·R sinkt (→ mehr Fehler als bei SNR).
16. Für die vergleichende Darstellung der Restfehlerraten verschiedener Codes (Punkt 4) bezieht man die Ergebnisse auf den Geschwindigkeits-normierten Wert **SNR' = SNR/R**. Die Einzelergebnisse sind dann aber jeweils für den Störabstand SNR ermittelt worden. Die Diagramme zeigen also die Funktionsverläufe p_{err} (SNR').
17. Die Korrekturfähigkeit eines Codes bei HD-Demodulation der **Empfangswörter w** wird durch den **Mindestabstand** (genauer Mindest-Hammingabstand)

$$d_{min} = 2 \cdot t_{kor} + 1$$

beschrieben, t_{kor} ist die Anzahl der korrigierbaren Fehler-Bits pro Codewort. Der Mindestabstand ist diejenige Anzahl von Codewort-Bits, in denen sich jedes Codewort gegen jedes andere unterscheidet.

18. Der einzige Code, bei dem der Mindestabstand unmittelbar zur Konstruktion verwendet werden kann, ist der **Wiederholungscode**. Im binären Fall besteht ein solcher Code aus 2 Codewörtern, jeweils eines für das Info-Bit 0 und das Info-Bit 1. Die Prüfbits wiederholen das Info-Bit, so dass die beiden Codewörter den gewünschten Mindestabstand (Punkt 11) aufweisen.
19. Der Wiederholungscode hat wegen seiner ungünstigen Inforate $R=1/d_{min}$ keine praktische Bedeutung (so genannter **asymptotisch schlechter Code**).
20. Da Codes für geringere Restfehlerraten mehr Prüf-Bits haben müssen, dies aber zugleich die Inforate R verkleinert und damit wiederum den Störabstand verschlechtert, müssen diese Codes geeignet sein, auch die dadurch bedingte nochmalige Erhöhung der Fehler mit abzufangen. **C. E. Shannon** hat in einem genialen Aufsatz gezeigt, dass es deshalb einen Grenzwert für den Geschwindigkeits-normierten Störabstand SNR' gibt. Für R -> 0 liegt dieser bei SNR' -> 1,386 (die berühmte **Shannon-Grenze** für R ->0, was zugleich sehr lange Codewörter bedeutet).
21. Für jeden Wert der Inforate R gibt es einen eigenen Wert SNR' (R) der Shannon-Grenze.
22. Für Werte SNR' oberhalb der jeweiligen Shannon-Grenze existieren Codes, mit denen eine **beliebig kleine Restfehlerrate** erreichbar ist, jedoch kennt man kein Bildungsgesetz für solche Codes. Sie müssen durch „Probieren“ gefunden werden.
23. In der Forschung hat man sich der Shannon-Grenze schon sehr stark angenähert (Hagenauer an der TU München). Die dort gefundenen Codes sind allerdings noch sehr aufwändig.
24. Neben der grundsätzlichen Korrekturleistung eines Codes ist auch der **Aufwand für Codierung und Dekodierung** von Interesse.
25. Häufig gibt man die Parameter Codewortlänge n, Anzahl der Info-Bits k und Mindestabstand d_{min} von Codes abkürzend als **(n, k, d_{min})** an.
26. Ein auch heute noch viel verwendeter Code zur Korrektur von 1-Bitfehlern in Speicherchips ist der **Hammingcode**. Bei ihm wird die Anzahl m der Prüf-Bits vorgegeben, so dass daraus die Kurzbeschreibung **$(2^m - 1, 2^m - m - 1, 3)$** entsteht.
27. Für praktisch ausgeführte Codes ist manchmal die erforderliche Anzahl der Info-Bits k gegeben. Man muss dann unter Umständen einen Code für mehr Info-Bits wählen und lässt im realen Codewort die

überzähligen Info-Stellen weg. Z. B. benötigt man zur Fehlerabsicherung von Speicherworten Info-Bitlängen in ganzzahligen Vielfachen von Bytes. Für 64-Bit-Speicherworte ist ein (71, 64, 3)-Hammingcode geeignet (warum?).

28. Bei jedem Code kann durch Hinzufügen eines weiteren Prüfbits auf gerade Parität eine **zusätzliche 1-Bit Fehlererkennung** bewirkt werden. Bei einem Hammingcode lässt sich damit neben der 1-Bit-Fehlerkorrektur noch eine 2-Bitfehler-Erkennung erreichen, was für den praktischen Betrieb von Speicherchips sehr nützlich ist (Warum?).

29. Neben der **Bitfehler-Häufigkeit** (= Fehlerrate) kann auch die **Wortfehler-Häufigkeit** betrachtet werden. Sie gibt an, wie viele empfangene Worte unkorrigiert oder korrigiert in Abhängigkeit von SNR fehlerhaft sind. Mit den Hilfsmitteln der Wahrscheinlichkeitsrechnung lassen sich aus den Bitfehler-Häufigkeiten die Wortfehler-Häufigkeiten für Empfangswörter der Länge n bestimmen. Man erhält so die Angabe, wie viele Worte im Mittel fehlerfrei sind, wie viele einen Fehler aufweisen usw. Der Nutzen liegt darin, theoretische Vorhersagen für die Korrekturleistungen verschiedener Codes machen zu können.

30. Die Korrekturfähigkeit eines Codes wird durch **notwendige** und **hinreichende Bedingungen** bestimmt. Sollen bei HD-Demodulation der Empfangswörter t_{kor}-Fehler korrigiert werden können, so ist zunächst

$$\text{Anzahl der Fehlermuster} = \sum_{i=1}^{t_{kor}} \binom{n}{t_{kor}}$$

Da für jedes Prüf-Bit in einem Codewort eine Bestimmungsgleichung aus einer geeigneten Kombination der Info-Bits aufgestellt werden muss, können bei m Prüf-Bits die aus dem Empfangswort w ermittelten Bit-Werte zu 2^m Kombinationen erfüllter und nicht erfüllter Prüf-Bit-Gleichungen (= **Syndromgleichungen**) führen, je nach dem tatsächlich eingetretenen Fehlermuster. Als Voraussetzung, dass ein Code überhaupt t_{kor} – Fehler korrigieren kann, muss also **$2^m - 1 \geq$ „Anzahl der Fehlermuster“** sein. Damit ist aber noch nicht gewährleistet, dass dann tatsächlich alle Fehlermuster korrigierbar sind, dies ist nur notwendig. Als hinreichende Bedingung muss sich jede Kombinationen der Syndromgleichungen auch noch **eindeutig** genau einem Fehlermuster zuordnen lassen.

31. Die einzelnen Codierungsverfahren stellen mit ihren Rechenvorschriften zur Bestimmung der m Prüfbits die beiden Forderungen aus Punkt 24 sicher.

32. Eine Variante des 1-Bitfehler korrigierbaren Hamming-Codes ist der **(23, 11, 7)-Golay-Code**.

33. Hamming-Code und Golay-Code gehören zu den **perfekten Codes**. Bei diesen ist $2^m - 1$ genau gleich der Anzahl der möglichen Fehlermuster.

34. **Systematische Codes** sind alle, bei denen die Info-Bits eine feste Stelle im Codewort besitzen, vorzugsweise, aber nicht zwingend, z. B. immer alle links. Bei **nicht systematischen Codes** sind die Info-Bits nicht unmittelbar aus dem Codewort ersichtlich. Sie müssen nach der Korrektur erst wieder aus dem Codewort ermittelt werden, bei systematischen Codes entfällt dieser Schritt. Viele Codes, aber nicht alle, lassen sich sowohl als systematische als auch nicht systematische Codes gestalten, die systematische Form wird aber bevorzugt.

35. **Zyklische Codes** bilden eine eigene Klasse. Sie sind sowohl zur **Mehrbitfehler-Erkennung** als auch für **Mehrbitfehler-Korrektur** geeignet. Da es zyklische Codes auch zur 1-Bitfehler-Korrektur gibt und diese ausserdem perfekt sind, bezeichnet man sie als zyklische Hamming-Codes.

36. Zyklische Codes für Mehrbitfehler-Korrektur sind nicht perfekt.

37. Zyklische Codes nutzen die Eigenschaften von **Galoisfeldern**, siehe auch Blatt „Tabellen zu GF(2^3) und GF(2^4). Bei binären zyklischen Codes sind es die Galoisfelder $GF(2^m)$, m ist der Grad des definierenden, über Z_2 irrduziblen Polynoms (z. B. für m = 3: $f(x) = x^3 + x^2 + 1$). Dieses Polynom erzeugt mit seiner Nullstelle $x = \alpha$ mit $f(\alpha) = \alpha^3 + \alpha^2 + 1 = 0$, das **primitve Element** α aus $\alpha^3 = \alpha^2 + 1$. Aus diesem entstehen durch Potenzierung alle $2^m - 1$ weiteren, von Null verschiedenen Elemente des Galoisfeldes.

38. Die **Ordnung** ord eines primitiven Elements ist die Anzahl seiner Potenzierungen, bis es sich wiederholt. Geeignete Nullstellenlieferanten sind nur solche irreduziblen Polynome, für die die Ordnung des primitiven Elements **ord = $2^m - 1$** und damit maximal ist.

39. Das **Generator-Polynom g(x)** bestimmt den Codewortaufbau eines zyklischen Codes.

40. Das Generatorpolynom für einen zyklischen Code zur Korrektur von t_{kor} Einzel-Bitfehlern besteht aus dem Produkt des irreduziblen Polynoms $g_1(x) = f(x)$, siehe Punkt 37, und $t_{kor} - 1$ weiterer irreduzibler Polynome $g_2(x)$, $g_3(x)$, ..., $g_{tkor}(x)$, die ihrerseits Potenzen des primitiven Elements als Nullstellen haben:

    $$\mathbf{g(x) = g_1(x) \cdot g_2(x) \cdot \ldots \cdot g_{tkor}(x)}$$

    Es sind nur **bestimmte Kombinationen irreduzibler Polynome** zum Aufbau solcher Generatorpolynome geeignet, nicht beliebige. Die Begründung hängt wieder mit den in Punkt 30 genannten notwendigen und hinreichenden Bedingungen zusammen.

41. Die Codewort-Polynome v(x) eines nicht systematischen zyklischen Codes sind das Produkt des Info-Polynoms u(x) und des Generator-Polynoms g(x), also $v(x) = u(x) \cdot g(x)$. Diese Variante wird wegen des geringfügig höheren Aufwands bei der Decodierung im allgemeinen nicht verwendet.

42. Zur Erzeugung der Codewörter von systematischen zyklischen Codes wird das Info-Polynom durch Multiplikation mit $x^{grad\ g(x)}$ von rechts aus um soviel Stellen mit Nullen aufgefüllt, dass die benötigten Prüfbits der Anzahl grad g(x) im Codewort v(x) Platz finden. Das Codewort wird über

    $$\mathbf{v(x) = u(x) \cdot x^{grad\ g(x)} + [u(x) \cdot x^{grad\ g(x)}]\ MOD\ g(x)}$$

    gebildet. Durch diese Konstruktion bleibt der Teil mit den Infobits links unverändert und – genauso wichtig – **jedes Codewort-Polynom v(x) ist ein Vielfaches von g(x)**.

43. Wegen Punkt 42 hat jedes Codewort-Polynom v(x) seinerseits alle Nullstellen von g(x), z. B. ist $v(\alpha) = 0$.

44. Jedes HD-demodulierte Empfangswort w kann als **Empfangswort-Polynom w(x)** dargestellt werden.

45. Das Empfangswort-Polynom lässt sich theoretisch als Summe des Codewort-Polynoms v(x) und eines Fehler-Polynoms e(x) auffassen (das der Empfänger zunächst nicht kennt):

    $$\mathbf{w(x) = v(x) + e(x).}$$

46. Die Aufgabe der Dekodierung ist die Bestimmung von e(x).

47. Durch Einsetzen der Nullstellen β der einzelnen Teilpolynome des Generator-Polynoms, siehe Punkt 40, erhält man t_{kor} **Syndromwerte $s(\beta)$, $s(\beta^3)$, ...** und damit t_{kor} Gleichungen zur Ermittlung der maximal t_{kor} Fehlerpositionen im Empfangswort:

    $$\mathbf{s(\beta) = w(\beta) = v(\beta) + e(\beta) = 0 + e(\beta) = e(\beta)}$$
    $$\mathbf{s(\beta^3) = w(\beta^3) = \ldots = e(\beta^3)}$$
    $$\mathbf{\ldots}$$

    Ein **Beispiel** für $t_{kor} = 3$ ist im Blatt „Beispiel für BCH-Code (15,5,7)“ dargestellt.

48. Die Nullstellen sind jeweils die ungeraden Potenzen **$\beta, \beta^3, \ldots$** des primitiven Elements.

49. Bei $t_{kor} = 3$ kann $e(x) = x^i + x^j + x^k$ sein, die Potenzen i, j und k geben die Fehlerpositionen an.

50. Einsetzen der Nullstellen gemäss Punkt 47 in die Beziehung nach Punkt 49 ergibt die 3 Gleichungen zur Bestimmung der 3 Fehlerpositionen:

    $$\mathbf{e(\beta) = \beta^i + \beta^j + \beta^k = s(\beta)}$$ usw.

51. Ein sehr uneffektiver, aber dennoch korrekter Lösungsweg wäre es, nacheinander alle i, j und k von 0 bis 14 einzusetzen (warum?) und zu prüfen, wann alle rechten Seiten erfüllt sind (**„brute force“-**

Methode). Das ist dann die Lösung. Wesentlich effektiver sind die beiden im „Grundkurs Codierung“ in den Unterkapiteln 3.7.7 und 3.8.3 beschriebenen Verfahren.

52. Der **Reed Solomon-Code** (RS-Code) ist als zyklischer Code ein Sonderfall des BCH-Codes. Das Generatorpolynom enthält als Nullstellen die Elemente des zugeordneten Galoisfeldes GF(2^m) in aufsteigender Reihenfolge der Potenzen, für das Galoisfeld GF(2^4) zum Beispiel:

    $$g(x)=(x-\beta)\cdot(x-\beta^2)\cdot(x-\beta^3)\cdot(x-\beta^4)\cdot\ \ldots..\ \cdot(x-\beta^{2\cdot t_{kor}-1})\cdot(x-\beta^{2\cdot t_{kor}})\quad .$$

    Für jeden zu korrigierenden Fehler sind zwei Linearfaktoren anzusetzen.

53. Das Generatorpolynom g(x) besitzt als Koeffizienten nun nicht mehr wie beim BCH-Code nur „0“ und „1“, sondern alle Elemente des Galoisfeldes GF(2^m), für GF(2^4) und t_{kor}=1 z. B.

    $$g(x)=(x-\beta)\cdot(x-\beta^2)=x^2-(\beta+\beta^2)\cdot x+\beta\cdot\beta^2=x^2+\beta^5\cdot x+\beta^3$$

54. Mit der Formel zur Bildung der Codewörter für zyklische Codes gemäß Punkt 42 entstehen mit dem Generatorpolynom Codewortpolynome v(x), die als Koeffizienten die Elemente des GF(2^m) besitzen. Diese Koeffizienten haben zunächst keinen Bezug zu den logischen „0/1“-Elementen realer Informationsquellen.

55. Über die abgekürzte (Koeffizienten-) Schreibweise für die Elemente des Galoisfeldes, die selbst wieder nur aus „0“ und „1“-Elementen besteht, ist eine **Zuordnung der GF(2^m)-Elemente zu Blöcken von m Bits** möglich. Der Parameter m ist hier der Grad des definierenden irreduziblen Polynoms für das Galoisfeld, für GF(2^4) also z. B. f(x) = x^4 + x +1. Im GF(2^4) wird die Folge 0100 z. B. dem Element β^2 zugeordnet.

56. Nach Zuordnung der m-Bit breiten Blöcke zu den Elementen des GF(2^m) werden alle weiteren Rechnungen im Galoisfeld gemacht, die Ergebnisse werden wieder den m-Bit breiten „0/1“-Blöcken zugewiesen.

57. Der Reed Solomon-Code ist besonders für die Korrektur von m-Bit breiten Fehlerbündeln geeignet, wie sie bevorzugt bei Massenspeichergeräten (Platten, CDs, ...) auftreten.

58. Für die Decodierung des RS-Codes muss neben der **Fehlerposition** im Codewort (jede Codewortstelle repräsentiert einen m-Bit breiten „0/1“-Block) auch der **Fehlerwert** (= Bitmuster des m-Bit breiten Fehlermusters) bestimmt werden. Diese Bestimmung erfolgt ähnlich wie beim BCH-Code.

59. Reed Solomon-Codes weisen als vorteilhafte Besonderheit ein **asymptotisch gutes Verhalten** auf. Das bedeutet, dass die RS-Codes bei Vergrößerung ihrer Länge keine Minderung der Korrekturleistung erleiden. Die meisten übrigen Codes sind asymptotisch schlecht.

60. Die Darstellung der Restfehlerraten als Funktion des Störabstandes wird zur Vergleichbarkeit verschiedener Codierungsverfahren meistens über den auf SNR' = SNR/R Geschwindigkeits-normierten Störabstand vorgenommen, siehe Punkt 16. Das geniale Kanalkapazitätstheorem von Shannon stellt einen Zusammenhang zwischen Kanalkapazität (= pro Sekunde fehlerfrei übertragbare Info-Bits) Bandbreite des Kanals (in Hz) und Störabstand SNR her:

    $$C=B\log_2(1+SNR)=B\log_2\left(1+\frac{SNR\cdot R}{R}\right)=B\log_2(1+SNR'\cdot R)$$

    In dieser einfachen Form gilt das Theorem für den theoretischen Fall normalverteilter Rausch- und Nutzsignale. Es leitet sich aus Entropie-Betrachtungen ab, die Shannon zunächst für ganz allgemeine Verteilungen aufgestellt hat, siehe Unterkapitel 3.5.6 im GKC. Danach ist die Übertragungsrate R_c fehlerfreier Info-Bits

    $$R_c=H(ws)-H_{vs}(ws)\quad ,$$

    woraus sich der oben genannte einfache Ausdruck für C ergibt.

61. Wir betrachten zwar den Fall normalverteilter Rauschsignale, aber **gleichverteilter, bipolarer** Nutzsignale. Hierfür liefert die obige Entropiebetrachtung für R_c einen Ausdruck, der nicht mehr als einfache Formel dargestellt werden kann. Die numerische Auswertung ist im GkC auf Seite 87 in Bild 3.7

gezeigt. Für gute (=große) Störabstände läuft die Kapazität gegen den erwarteten Wert 2·B, für die eigentlich interessanten schlechten (= kleinen) Störabstände gegen den Funktionsverlauf für normalverteilte Nutzsignale. Die Übertragungsrate R_c als Rate der fehlerfreien Info-Bits stellt bezogen auf ein gesendetes Codewort-Bit pro Sekunde die Info-Rate R dar:

$$\frac{R_c}{1s} = R = \frac{1}{2} \cdot \log_2(1 + SNR' \cdot R) \quad \rightarrow \quad \frac{2^{2 \cdot R} - 1}{R} = SNR'$$

Dieser Ausdruck ist nur für bestimmte Wertepaare von R und SNR' erfüllbar. Für sehr kleine Info-Raten R → 0, die bei stark gestörten Kanälen wegen der vielen notwendigen Prüfbits zwangsläufig erforderlich werden, läuft SNR' gegen 1.38, die berühmte **Shannongrenze**.

62. In der Literatur findet man für den Geschwindigkeits-normierten Störabstand SNR' häufig den gleichwertigen Ausdruck

$$SNR' = \frac{2 \cdot E_b}{N_{0,}} \quad \text{oder} \quad \frac{E_b}{N_0} = \frac{SNR'}{2} \quad .$$

Hierin bezeichnet E_b die **auf die Info-Rate bezogene Energie für jedes gesendete Signal-Bit E_b**, $E_b = E_s/R$, der Parameter N_0 ist als **Rauschleistungsdichte** eine spezifische Konstante des verwendeten Kanals.

63. Die Korrekturleistung von Codes lässt sich steigern, wenn man **Soft Decision-Verfahren** in Verbindung mit **Produkt-Codes** (= verschachtelte Codes, „Kreuzworträtsel“) einsetzt. Besonders gut arbeiten **Turboprodukt-Codes** (TPC), mit denen man der Shannongrenze nahe kommt. Ihr Einsatz erfordert allerdings einen hohen Rechenaufwand, siehe GkC, Unterkapitel 3.11 und 3.12. Zum Beispiel stellt das Unternehmen AHA komplette Chips her, die eine Fehlerkorrektur nach dem TPC-Verfahren bei Geschwindigkeiten von einigen -zig Megabit/s liefern.

64. Für die besonderen Anforderungen im Mobilfunk wurden **Faltungscodes** (= Convolutional Codes) entwickelt, siehe GkC, Unterkapitel 3.13. Bei diesen lassen sich Soft-Decision-Verfahren (BCJR) besonders einfach gestalten, weshalb sie sich für die Kompaktheitsanforderungen in Handys eignen.

65. Faltungscodes entstehen aus den Info-Bits über Schieberegisterfolgen. Der zeitliche Ablauf der Codewort-Erzeugung lässt sich übersichtlich in Form eines **Trellisdiagramms** darstellen.

Verschlüsselung

66. Während bei Fehlerkorrekur-Codes aus den Info-Bits Prüfbits berechnet und den Info-Bits als redundante Zusatzinformationen „angehängt“ werden und sich damit das Volumen vergrößert, werden bei der **symmetrischen Verschlüsselung** die Info-Bits durch Addition eines **Schlüssels S** so verändert,

$$\text{Klartext K} \overset{\text{Schlüssel}}{\longleftrightarrow} \text{Geheimtext G} \qquad \text{(symmetrische Verschlüsselung)} \quad ,$$

dass im Idealfall nur die Schlüsselbesitzer aus dem **Geheimtext G** in akzeptabler Zeit den **Klartext K** wiederherstellen können. Das Volumen des Geheimtextes bleibt dabei im Vergleich zum Klartext unverändert.

67. Dies funktioniert auch mit der **asymmetrischen Verschlüsselung**, bei der die Verschlüsselung mit einem anderen Schlüssel, dem **öffentlichen Schlüssel e**, erfolgt, als die Entschlüsselung mit dem **geheimen Schlüssel d**:

$$\text{Klartext K} \rightarrow \overset{\text{öffentlicher Schlüssel e}}{\rightarrow} \text{Geheimtext G}$$
$$\text{Geheimtext G} \rightarrow \overset{\text{geheimer Schlüssel d}}{\rightarrow} \text{Klartext K}$$

68. Wird für alle Klartextzeichen immer derselbe Schlüssel verwendet, so bildet sich die statistische

Häufigkeitsverteilung der Klartextzeichen direkt in die der Geheimtextzeichen ab. Dies erleichtert die Bestimmung des Klartextes aus dem Geheimtext, wenn der Schlüssel S oder die Schlüsselpaare (e,d) nicht bekannt sind.

69. Die Häufigkeitsverteilung der Klartextzeichen, insbesondere der von natürlichen Sprachen, muss wegen Punkt 68 also möglichst flach gemacht werden. Dies erreicht man durch Zusammenfassung einzelner Klartextzeichen zu Blöcken von „Superzeichen“. Beim DES-Verfahren sind es in der 64 Bit-Urform 8 ASCII-Zeichen, beim RSA-Algorithmus wären es bei einer Schlüssellänge von 192 dann 24.

70. Eine auch im mathematischen Sinn bewiesenen **perfekte Verschlüsselung** erreicht man, wenn jedes Klartextzeichen mit einem anderen zufällig gewähltem Schlüssel verschlüsselt wird (**One Time Pad**). Bei Gleichverteilung der Schlüssel nähert sich dann die Verteilung der Geheimtextzeichen ebenfalls einer Gleichverteilung, aber selbst ein eventuell noch verbleibendes Verteilungsprofil gäbe keine brauchbaren Hinweise auf den Klartext, da zu jedem Geheimtextzeichen mehrere verschiedene Klartextzeichen gehören können.

71. Selbst wenn der Angreifer beim One Time Pad-Verfahren mit der „Brute Force-“ Methode alle Schlüsselkombinationen durchprobiert, hilft das nicht, weil er eine Fülle sinnvoller und darunter auch sehr ähnliche Klartexte erhält, unter denen er sich noch entscheiden müsste.

72. Eine **große Schwachstelle** aller symmetrischen und – eingeschränkt – auch asymmetrischen Verfahren ist die sichere Schlüsselübergabe. Insbesondere beim One Time Pad-Verfahren muss ein ebenso langer Schlüssel wie der Klartext übergeben werden, weshalb der Eindruck besteht, dass die direkte Übergabe des Klartextes nicht schwieriger wäre. Das trifft zwar nicht zu, da es auch für die Erzeugung langer Schlüssel Verfahren gibt, bei denen mit kurzen deterministischen Schlüsseln lange pseudo-statistische, also Zufalls-ähnliche Schlüsselfolgen (siehe Unterkapitel 4.3 „Zufallserzeugung von Schlüsselwörtern“) entstehen, doch das grundsätzliche Problem bleibt damit weiter ungelöst.

73. Die **vollkommene Beseitigung** der Schwachstelle „Schlüsselübergabe“ wird voraussichtlich mit Hilfe **quantenphysikalischer Effekte** gelingen. Dies basiert darauf, dass Photonen (= Lichtteilchen) polarisiert werden können. Polarisiert der Sender die Photonen senkrecht oder waagerecht und hat der Empfänger seinen Polarisator ebenfalls senkrecht oder waagerecht eingestellt, so empfängt er das Photon exakt so, wie es gesendet wurde. Hat er seinen Polarisator dagegen um 45 Grad schräg orientiert, so ist die Zuordnung zur gesendeten Polarisation im Mittel nur zur Hälfte richtig (entsprechend bei Aussendung mit „schräger“ und Empfang mit „gerader“ Polarisation).

    Der Sender schickt die zufällig gewählten 0/1-Bits des Schlüssels also mit einer zufällig gewählten Folge von Polarisationen. Nach Übertragungsende gibt der Sender dem Empfänger **öffentlich** die Polarisator-Stellungen bekannt, dieser streicht diejenigen Bits heraus, bei denen er seinen Polarisator falsch eingestellt hatte und es bleiben im Mittel 50% richtige Schlüsselbits übrig, mit denen nun die Verschlüsselung durchgeführt werden kann. Diese richtigen Schlüsselbits sind im Übrigen nur dem Sender und dem Empfänger bekannt.

    Ein **unberechtigter Abhörer** im Leitungsweg ist zunächst in der gleichen Lage wie der berechtigte Empfänger – mit einem entscheidenden Unterschied: Wenn er die Leitung anzapft und ein Photon empfängt, ist es verbraucht. Er muss nun durch „Würfeln“ entscheiden, welche Sende-Polarisation gewählt wurde und schickt mit dieser – im Mittel nur zur Hälfte richtigen – Polarisation ein neues Photon an den Empfänger weiter. Nach Übertragungsende und Abgleich mit dem Sender stellt der Empfänger im Mittel also nur 25% richtige Polarisationsrichtungen fest. Der Kanal wird als unsicher angesehen, der Schlüssel verworfen und ein anderer Kanal gewählt.

74. Der Urvater der praktisch auch noch heute kommerziell verwendeten symmetrischen Verschlüsselungsverfahren ist der **DES** (= Data Encryption Standard). Er gilt wegen seiner kurzen Block- und Schlüssellänge als nicht mehr ausreichend sicher, wird aber als Tripel-DES immer noch eingesetzt. Sein Nachfolger ist seit wenigen Jahren der **AES** (= Advanced Encryption Standard), der ähnlich arbeitet, aber mehr Flexibilität bei der Wahl der Schlüssellänge bietet und insgesamt als sicherer angesehen wird.

75. Der Urvater der asymmetrischen Verfahren ist der **RSA-Algorithmus** (von Rivest, Shamir und Adelman), siehe Punkt 67. Seine Sicherheit beruht auf der Unmöglichkeit, auch mit den schnellsten Rechnern und Verfahren eine gegebene ganze Zahl in akzeptabler Zeit (= in wenigen Tagen) in ihre **Primfaktoren** zu zerlegen. Die technologische Entwicklung zu schnelleren Rechnern lässt sich dabei immer durch eine Vergrößerung der Schlüssellänge ausgleichen. Sollte es jedoch einmal Wege geben, die eine um Größenordnungen schnellere Primzahlenzerlegung gestatten (z. B. mit **Quantencomputern**), so sind diese Verfahren „tot". Dann bliebe das One Time Pad als vollwertige Alternative.

76. Das RSA-Verfahren kann in **zweierlei** Hinsicht genutzt werden:

    - Als **Verschlüsselung** eines Klartextes K in den Geheimtext G mit dem öffentlichen Schlüssel e des Empfängers und Entschlüsselung durch ihn mit seinem geheimen Schlüssel d
    - Als **Authentifizierungsverfahren** dafür, dass ein öffentlich zugängliches Dokument zweifelsfrei vom Autor (= Besitzer des geheimen Schlüssels d) stammt. Der Autor verschlüsselt in diesem Fall den Klartext K mit seinem geheimen Schlüssel d zum Pseudo-Geheimtext G. Jeder kann diesen nun mit dem öffentlichen Schlüssel e des Autors lesbar machen (= elektronische Unterschrift). **Aber Vorsicht:** Damit ist in keiner Weise sichergestellt, dass der tatsächliche Autor mit dem „erwarteten" Autor identisch ist. Dies kann nur durch zusätzliche Massnahmen abgesichert werden, z. B. durch **Zertifizierungsverfahren**.

77. Der RSA-Algorithmus erfordert gegenüber der symmetrischen Verschlüsselung (DES, AES) etwa die **10-fache Rechenzeit**. Daher wird er in kommerziellen Anwendungen heute noch nicht ausschließlich eingesetzt.

78. Die Verschlüsselung nach dem **PGP** (Pritty Good Privacy)-. oder **GNUPP**-Verfahren versucht, die Vorteile der symmetrischen und asymmetrischen Lösungen zu verbinden. Dies ist insbesondere in Hinsicht auf die Schwachstelle der Schlüsselübergabe interessant:

    - Der Sender erzeugt für eine aktuelle Session ein öffentlich/geheimes RSA-Schlüsselpaar (e,d) und schickt den öffentlichen Schlüssel e an den Empfänger. Dieser verschlüsselt damit einen bei ihm gebildeten temporären Schlüssel für ein symmetrisches Verfahren (z. B. AES) und schickt ihn an den Sender. Mit diesem Schlüssel läuft nun eine schnelle, verschlüsselte Kommunikation ab. Nach Ende verfällt der temporäre Schlüssel. Dies entspricht grob dem Ablauf bei Aufruf eines **https-Protokolls** z. B. bei Aufbau einer gesicherten Homebanking-Verbindung.
    - Die Unsicherheit der Schlüsselübergabe ist damit entschärft (das „**Man in the Middle**"- Problem bedarf allerdings auch hier einer äußerst sorgfältigen Betrachtung), aber nicht vollständig beseitigt.
    - Als Problem bleibt weiterhin die Authentifizierung der Partner: Wie kann der Bankkunde sicher sein, dass der Bank-Server wirklich der von seiner Bank ist? Und wie kann die Bank sicher sein, dass es sich bei ihm wirklich um den bei ihr registrierten Kunden handelt?
    - Man versucht, diese Unsicherheit durch **Zertifizierungen** zu mildern. ebay fordert seine Nutzer z. B. auf, die Qualität eines Anbieters zu bewerten. Je mehr gute Bewertungen zusammen kommen, desto mehr wächst die Sicherheit, dass man es mit einem seriösen Partner zu tun hat (ohne jemals völlige Sicherheit zu erlangen). Mit den abstrakteren **Sicherheits-Zertifikaten** im Internet wird ein ähnlicher Weg beschritten.

79. Das Gebiet der Verschlüsselung behandelt neben der eigentlichen „klassischen" Verschlüsselung heute in gleich bedeutender Weise auch die damit zusammenhängenden Verfahren

    - der **elektronischen Unterschrift** (RSA, DSA und weitere),
    - des **Integritäts-Nachweises** mit Hilfe von Hash-Summen (= Prüfsummen) oder MACs (=Message Authenication Codes), womit überprüft werden kann, ob ein Dokument verändert wurde,
    - der **sicheren Schlüsselübergabe** (Diffie-Hellman-Schlüsseltausch),
    - der **Authentifizierung** (= Berechtigung) eines Anwenders (Zero Knowledge Protokolle, Fiat-Sha-

mir-Verfahren), bei der festgestellt wird, ob er im Besitz einer geheimen Information ist, ohne diese bekannt geben zu müssen.

- und weitere, wie elektronisches Geld u. a. (siehe die sehr empfehlenswerten Bücher von **Albrecht Beutelsbacher**, Vieweg-Verlag).

80. Bei der Verwendung von **Hash-Summen** zur Überprüfung einer Datei auf technisch oder kriminell hervorgerufenen Veränderungn muss sicher gestellt sein, dass die Hash-Summe selbst **vertrauenswürdig** ist. Ein Angreifer könnte ein Dokument verändern und die passende Hash-Summe dazu stellen, was vom berechtigten Empfänger dann unbemerkt bliebe (z. B. können die Open Office Produkte mit den auf der offiziellen Homepage veröffentlichten MACs geprüft werden. Ein Angreifer müsste beim Empfänger die Open Office Homepage nachbilden, was prinzipiell ginge, aber vermutlich auch schnell entdeckt würde).
81. Überhaupt ist für die Sicherheit eines Verschlüsselungs-Verfahrens nicht nur die mathematische Sicherheit zu beachten, sondern die **Sicherheit des Gesamtablaufs** im praktischen Gebrauch (= **sichere Protokolle**). Dafür müssen alle möglichen Angriffs-Szenarios betrachtet werden. Eine Schwachstelle ist oft der „**Man in the Middle**“, also ein Angreifer, der sich unbemerkt in den Kommunikationsstrom einklinkt und im entscheidenden Augenblick die in seinem Sinn veränderten Informationen an Sender und Empfänger schickt.
82. Beim **Fiat-Shamir-Protokoll** kann ein Angreifer versuchen, sich als Berechtigter zu erweisen. Er hat aber in jedem Durchlauf des Authentikations-Protokolls nur eine **50-prozentige Chance**, erfolgreich zu sein. Wird das Verfahren z. B. 20-mal durchlaufen, sinkt die Wahrscheinlichkeit, erfolgreich zu sein, bereits auf 0.5^{20}. Es wäre sehr unwahrscheinlich, dass er hier ungeschoren durchkommt.

Datenkompression

83. Im Gegensatz zur Fehlerkorrektur, bei der an die Info-Bits zusätzliche Prüfbits angehängt werden und sich dadurch die Redundanz einer Nachricht vergrößert, besteht die Aufgabe der Kompression darin, Redundanz aus der Information heraus zu nehmen und damit das Volumen zu verkleinern. Hierdurch lassen sich Übertragungszeiten und Speicherkosten senken.
84. Die für den Menschen zu dessen unmittelbarer Verwendung bestimmten Informationsmengen (gesprochene Worte, Schriftstücke, Tabellen, Grafiken, Bilder, Filme, Programmcodes, Musikstücke und ähnliches) lassen sich technisch in Form von Bitströmen, z. B. unterteilt in Bytes, darstellen, übertragen und speichern. Unbehandelt werden hierzu mehr Bits benötigt, als zur vollständigen Abbildung des Originals notwendig wären. Die Differenz ist eine Folge der **Redundanz**, einem Maß für den Anteil mehrfacher Information in einem Bitstrom.
85. Durch **Kompression** wird ein Teil - oder im Idealfall sämtliche - Redundanz aus einer Nachricht entfernt. Der Vorgang ist vollständig oder teilweise umkehrbar, so dass sich die für die unmittelbare Verwendung durch den Menschen benötigte Originalform wieder herstellen lässt.
86. Redundanzfreie Nachrichten sind im allgemeinen nicht zweckmäßig, da sie selbst durch kleine Veränderungen bei Störungen nicht mehr den originalen Informationsgehalt aufweisen und sich nicht vollständig wiederherstellen lassen. Z. B. ermöglicht uns die hohe Redundanz der gesprochenen Sprache, unsere Partner auch bei ungünstigen, Lärm erfüllten Umgebungsbedingungen noch zu verstehen.
87. Die **Kernaufgabe für die Datenkompression** besteht darin, Wege zu finden, über die die redundanten Anteile in einer Nachricht identifiziert und entfernt und später wieder eingefügt werden können (damit sie für den Menschen „lesbar“ ist).
88. Der technische Ablauf einer Nachrichtenübertragung oder Speicherung wird im allgemeinen aus mehreren Teilen bestehen:

    - Kompression des Originals
    - Hinzufügen von Redundanz zum Zweck der Fehlerkorrektur

- Senden über im allgemeinen verrauschte Kanäle
- Durchführen der Fehlerkorrektur
- Wiederherstellen des Originals

89. Zusätzlich kann in den Ablauf gemäß Punkt 85 nach dem Kompressionsschritt noch ein Sicherungsverfahren zum Verhindern unberechtigter Veränderungen oder zur Verschlüsselung eingefügt werden (elektronische Unterschrift, Geheimhaltung). Dieser Schritt würde beim Empfänger nach der Fehlerkorrektur dann wieder rückgängig gemacht. Insgesamt ist dies aber nicht mit einer Redundanzveränderung verbunden.
90. Es gibt zwei unterschiedliche Vorgehensweisen zur **Redundanzverringerung:**
    - **Verlustfreie Kompression** (lossless compression) = die Wiederherstellung der originalen Nachricht ist möglich und unbedingt erwünscht (Texte, Bankdaten, Programmcode und ähnliches)
    - **Verlustbehaftete Kompression** (lossy compression) = die Wiederherstellung der originalen Nachricht ist nicht möglich und auch nicht erforderlich (Bilddaten, akustische Daten). Dies ist eng mit den physiologischen und psychologischen Merkmalen der menschlischen Sinneswahrnehmung verbunden.
91. Die verlustfreie Kompression nutzt entweder bestimmte Strukturen einer Nachricht, z. B. die Aufeinanderfolge gleicher Zeichen/Bytes, oder die unterschiedliche Häufigkeitsverteilung der Zeichen/Bytes.
92. Beim RLE-Verfahren (= Run Length Encoding) werden aufeinander folgende gleiche Bytes durch die Angabe des Bytes selbst und seiner Vielfachheit (Zählerinhalt) ersetzt. Da hierfür auch ein Zählerkennungszeichen erforderlich ist, lohnt sich dieses Vorgehen ab 4 gleichen Bytes. Als Zählerkennungszeichen eignet sich jedes Byte, welches im Original nur selten vorkommt, in praktisch angewendeten RLE's ist es oft $90_{HEX} = 144_{DEZ}$. Wenn es im Original doch auftritt, wird es durch das nachfolgende Zählerkennungszeichen und den Zählerinhalt „0“ ergänzt.

    Das RLE-Verfahren eignet sich praktisch nicht für Textdokumente, wohl aber für die Kompression von Bild-Dateien, z. B. wenn größere, gleichartige Farbflächen vorkommen. Auch als Zwischenschritt bei mehrstufigen Kompressionen (JPEG) ist es vorteilhaft.
93. Wenn die **statistische Häufigkeitsverteilung der Bytes** in einer Nachricht bekannt ist (etwa die Verteilung der Buchstaben in einem Text), so kann man den häufig vorkommenden Bytes kurze Codes mit wenigen Bits und den seltenen Bytes lange Codes zuordnen. Im Mittel erhält man dadurch eine reduziertes Gesamtvolumen.
94. Weitere Reduzierungen (= Redundanzverringerungen) sind möglich, wenn man nicht nur die Zeichen selbst, sondern **Zeichenfolgen und deren Häufigkeiten** zugrunde legt. Z. B. kommt in der deutschen Sprache die Folge „ei“ häufiger als „qy“ vor. Ähnliches gilt für Folgen von 3 Zeichen wie bei „sch“ und mehr. Allerdings steigt der Aufwand dabei auch stark an (Tabellen werden exponentiell größer), so dass man einen Kompromiss eingehen wird.
95. Verlustfreie Verfahren, welche die statistischen Häufigkeitsverteilungen nutzen, sind
    - **Huffman-Codierung** oder **Fano-Codierung** (optimal für Häufigkeiten, die in Potenzen von 0.5 auftreten, was praktisch kaum der Fall ist, aber dennoch brauchbar für beliebige Verteilungen)
    - **arithmetische Codierung** (optimal für beliebige Häufigkeitsverteilungen, aber aufwändiger als die Huffman-Codierung)
    - **LZW-Codierung** (= Lempel-Ziv-Welch). Im Gegensatz zur Huffman-, Fano- oder arithmetischen Codierung, bei der die Verteilungstabellen vorher bekannt sein - oder ermittelt und zusätzlich übertragen – werden müssen, erzeugt das LZW-Verfahren die notwendigen statistischen Informationen selbst ad hoc während des Übertragungsvorgangs am Sende- und Empfangsort. Diese Art der Kompression ist daher unabhängig vom Typ der Nachricht (= Dokumenten-transparent).
96. Für bestimmte Informationstypen wie Bilddokumente oder Sprach/Musik-Dateien ist eine verlustfreie

Kompression nicht unbedingt erforderlich. Die spezifischen Eigenschaften unseres Gesichts- und Gehörsinnes erlauben das Weglassen bestimmter Informationsanteile, ohne dass uns dieses stört oder wir es überhaupt bemerken. Wendet man dann Verfahren zur verlustbehafteten Kompression an, so lässt sich ein höherer Kompressionsgewinn als mit verlustfreien Verfahren erzielen. Allerdings ist die vollständige Wiederherstellung des Originals unmöglich.

97. Die **Aufgabe der verlustbehafteten Kompression** besteht

    - bei Grafik- oder Audio-Dateien entweder in der Identifizierung derjenigen Informationsanteile, die ohne störende Folgen verringert oder sogar weggelassen werden können (wie bei JPEG, MPEG, MP3). Hierfür eignen sich insbesondere die Techniken der **Fourier- und der Wavelet-Transformation**.
    - oder bei Grafikdateien auch in der Suche nach **selbstähnlichen Bildelementen**. Hierbei werden die Bilder zunächst in einzelne Felder (z. B. Quadrate) zerlegt, ein Feld wird als **Referenzfeld** ausgewählt und weitere Felder unter zugrunde Legung eines **Ähnlichkeitsmaßes** darauf untersucht, wie gut sie sich im Sinne dieses Maßes durch einfache **Veränderungsoperationen** wie **Drehung**, **Verschiebung**, **Streckung** oder **Stauchung** mit dem Referenzfeld „zur Deckung“ bringen lassen. In den Fällen, in denen das zum Erfolg führt, braucht man dann statt des Feldes selbst nur noch die wenigen Veränderungsparameter zu übertragen (man kann dies als Erweiterung von RLE auffassen). Als Technik hierfür eignen sich die **Fraktal-Methode**. Ist z. B. in einem Bild ein Himmelsausschnitt mit Schleierwolken dargestellt, so kann man sich vorstellen, diesen gut durch die Fraktaltechnik komprimieren zu können. Bei stark unterschiedlichen Feldern wird dies nicht erfolgreich sein.

98. Die **Fourier-Transformation** ist ein mathematisches Verfahren, um eine Orts- oder Zeitfunktion (Pixel-Verlauf in einem Bild, Spannungsverlauf einer Mikrofonaufnahme) in eine **Summe von Sinus– und Cosinus-Funktionen** zu wandeln. Diese Sinus- und Cosinus-Funktionen bestehen aus einer „**Grundschwingung**“ der Frequenz f_0 und deren „**Oberwellen**“ mit den ganzzahligen Vielfachen von f_0. Fourier hat gezeigt, dass – mit kleinen Einschränkungen - beliebige Funktionen durch solche Sinus/Cosinus-Schwingungen in Näherung gut beschrieben werden können, wenn man die **Amplituden** dieser Schwingungen geeignet wählt.

    Die Berechnung dieser geeigneten Amplituden aus den Orts- oder Zeitfunktionen s(x) bzw. s(t) wird durch eben die Fourier-Transformation beschrieben, siehe Beispiel in Punkt 99.

    Die tiefen Frequenzanteile in der Nähe von f_0 beschreiben dabei die „Hauptinformation“ in einem Bild oder einer akustischen Aufnahme. Z. B. gibt eine stark gerasterte Schwarz-Weiss-Darstellung eines fein aufgelösten Farbfotos meist schon den wesentlichen Bildinhalt wieder. Die höheren Frequenzanteile dienen dann eher noch der „kosmetischen“ Verfeinerung. Wenn man diese also gezielt absenkt oder gar weg lässt, erreicht man die gewünschte Verringerung des Darstellungsvolumens.

    Aber Achtung: Die **Fourier-Transformation selbst ist verlustfrei**, sie dient nur der Vorbereitung für die eigentliche Kompression.

99. Eine häufig genutzte Variante der Fourier-Transformation kann dann verwendet werden, wenn das räumliche Bildsignal s(x) oder das zeitliche Audio-Signal s(t) die maximale Frequenzkomponente f_{max} enthält (= **bandbegrenztes Signal**). Sie wird als **diskrete Fourier-Transformation** bezeichnet. Entnimmt man dem Signalverlauf s(x) oder s(t) über die „Zeit“ T eine Anzahl von 2N äquidistanten Messungen im Abstand T/(2N), so können daraus genau 2N Fourier-Koeffizienten (die Amplituden der N Sinus- und Cosinusschwingungen) bestimmt werden:

$$s(\nu \cdot \Delta T) = \sum_{\lambda=0}^{N} a_\lambda \cos\lambda\frac{\nu\pi}{N} + b_\lambda \sin\lambda\frac{\nu\pi}{N}, \qquad \nu = 0,\ 1,\ 2,\ \ldots.,\ 2N-1 \quad \text{mit}$$

$$a_\lambda = \frac{1}{2N}\sum_{\nu=0}^{2N-1} s(\nu\cdot\Delta T)\cos\lambda\frac{\nu\pi}{N}, \qquad b_\lambda = \frac{1}{2N}\sum_{\nu=0}^{2N-1} s(\nu\cdot\Delta T)\sin\lambda\frac{\nu\pi}{N}, \qquad \lambda = 0,\ 1,\ 2,\ \ldots,\ N$$

Hierbei wird immer $b_0 = 0$ und $b_N = 0$, so dass den 2N „Abtastwerten“ $s(\nu\cdot\Delta T)$ genau 2N Fourierkoeffizienten gegenüberstehen und umgekehrt. Auch daran sieht man, dass die Fourier-Transformation selbst verlustfrei wirkt.

100. Beim **JPEG-Verfahren** (= Joined Photographic Expert Group) wird die Bilddatei mit Hilfe der (zweidimensionalen) Fourier-Transformation zunächst in eine Summe von Grundschwingungen und deren Oberwellen zerlegt. Das Ergebnis ist eine **quadratische Tabelle** mit den Amplituden, wobei die zur Grundschwingung gehörende (= genauer der Gleichanteil) links oben, die zu den hochfrequenten Schwingungen gehörenden rechts unten stehen. Die Kompression durchläuft nun folgende weiteren Schritte:

- Die absoluten Zahlen werden entsprechend der gewählten Bit-Tiefe **gerundet**, z. B. auf 8 Bit-Ganzzahlen. Dabei ergibt sich schon ein kleiner Kompressionseffekt, der allerdings nicht rückgängig gemacht werden kann.
- Nun senkt man die Amplituden der hochfrequenten Anteile mit Teilerfaktoren (**Quantisierungstabelle**) gegen die Amplitude der Grundschwingung ab und dies um so stärker, je höher die Frequenz ist. Die Ergebnisse sind nochmals gerundet, der Bit-Bedarf zur Darstellung der auftretenden Zahlen verkleinert sich.
- Durch die **Rundung** erhält man im Mittel mehrfach auftretende gleiche Zahlen, insbesondere Nullen.
- Die Zahlen der so behandelte quadratische Amplitudentabelle werden durch einen **Zick-Zack-Durchlauf** von links oben nach rechts unten in eine neue Reihenfolge gebracht. Dabei ordnen sich gleiche Werte oft so, dass sie ijn Blöcken auftreten. Insbesondere entstehen auch „0“-Folgen.
- Diese Zahlenfolge durchläuft die verlustfreie RLE-Kompression.
- Das Ergebnis wird nochmals mit der Huffman- oder arithmetischen Codierung komprimiert, der Vorgang ist damit beendet. Der Kompressionsgewinn beträgt bis 98 % (1:50)
- Zur Wiederherstellung der Originaldatei durchläuft man die genannten Schritte rückwärts. Wegen der Rundungen und Quantisierungen gelingt dies allerdings nicht vollständig (**Kompressionsverlust**).

101. Beim **MPEG-Verfahren** (= Moving Picture Expert Group) werden ausgewählte Einzelbilder eines Videos ähnlich komprimiert wie bei JPEG. Folgebilder unterscheiden sich häufig nur geringfügig, so dass es ausreicht, die dann meist **kleinen Differenzzahlen** zu übertragen. Dies ergibt eine weiteren erheblichen Einspareffekt. Allerdings muss zwischendurch immer wieder ein Vollbild eingeschaltet werden, da bei Übertragungsstörungen die Differenzangaben zu unbrauchbaren Ergebnissen führen.

102. Die Fourier-Transformation stellt zwar ein gutes Hilfsmittel zur Vorbereitung der eigentlichen Kompression dar, hat aber auch **Grenzen**. Insbesondere erfordert die Darstellung von orts- oder Zeit-Funktionen mit abrupten, z. B. **scharfkantigen Bildelementen** oder **hart einsetzenden Klangereignissen** (Schlagzeug) die Berücksichtigung unverhältnismäßig vieler hochfrequenter Anteile in der Summe der Sinus- und Cosinus-Funktionen. Und selbst dann ist eine beliebig genaue Darstellung scharfer Übergänge aus prinzipiellen Gründen unmöglich (Grund: „weicher“ Verlauf der Sinus- und Cosinus-Funktion, **Gibbs'sches Phänomen**).

103. Die Vermeidung der unter Punkt 102 genannten Nachteile der Fourier-Transformation gelingt, wenn man die Summe nicht aus den „weichen“ Sinus- und Cosinusfunktionen sondern aus solchen zusammensetzt, die eine direktere Darstellung scharfkantiger (= **unstetiger**) Übergänge gestatten. Dies führt auf die **Wavelet-Transformation**. Von der Vielzahl möglicher **Basis-Funktionstypen** (sie müssen allerdings ganz bestimmte Eigenschaften aufweisen) eignen sich z. B. auch solche, die

einen rechteckförmigen Anteil haben. Daran wird klar, dass sich damit scharfe Übergänge eher gut modellieren lassen als mit Sinus- oder Cosinus-Funktionen.

Die **Wavelet-Basis-Funktionen** werden ähnlich wie bei der Fourier-Transformation in der Höhe angepasst, zusätzlich jedoch auch verschoben, gestaucht oder gestreckt. Die damit zusammenhängenden Veränderungsparameter bilden die Wavelet-Koeffizienten, mit denen ihrerseits die vollständige Wiederherstellung des Originals möglich ist.

Die Wavelet-Transformation gestattet im Mittel höhere Kompressionsgewinne als die Fouriertransformation. Sie liegen z. Z. bei 99 % (1:100). Sie ist inzwischen zusätzlicher Bestandteil von JPEG 2000. Da die gängigen Browser dies aber noch nicht unterstützen, kann man die Vorteile flächendeckend allerdings nicht nutzen.

104. Bei **Audio-Dateien** könnte man theoretisch einen ähnlichen Weg zur Kompression gehen wie bei Bilddokumenten. Die Absenkung der hochfrequenten Anteile würde aber ein unerwünschtes, weil **dumpferes Klangbild** ergeben. Daher löst man die Aufgabe hier anders. Unser Gehörsinn ist durch ein hohes Maß an Vorverarbeitung über das Gehirn gekennzeichnet, ehe ein **Klangereignis** an unsere bewusste Wahrnehmung übergeben wird. Insbesondere verdecken starke (= laute) Frequenzanteile die in unmittelbarer zeitlicher Nachbarschaft liegenden schwächeren, sowohl wenige Millisekunden vorher (!), als auch einige -zig Millisekunden danach. Diese Anteile werden dem Bewusstsein gar nicht weiter geleitet und lassen sich ersatzlos streichen.

Die Aufgabe besteht hier also darin, herauszufinden, welche „**Masken**“ in welchen Frequenzbereichen angewendet werden können. Die Masken selbst ermittelt man durch Messungen an Versuchspersonen, es entsteht das **psychoakustische Modell.** Das dabei gewonnene Wissen dient später zum gezielten Weglassen nicht hörbarer Frequenzbeiträge. Was nicht übertragen werden muss, verbraucht auch keine Bits, weshalb der wesentliche Kompressionsgewinn hieraus entsteht.

Das **MP3-Verfahren** (= MPEG 1 Layer 3) legt das Vorgehen fest. Auch hierzu ist eine Fourier-Transformation erforderlich, um die Frequenzanteile in einem Zeitfenster von jeweils wenigen Millisekunden zu ermitteln. Diese werden zunächst in 32 Frequenzbereiche aufgeteilt und danach in einem aufwendigen Schritt durch Anwendung der Maskenfunktionen weiter verarbeitet, dies alles zudem in Echtzeit. Bei der Wiedergabe der MP3-Files entfällt der zugehörige rückwärtige Schritt, so dass die Wiederherstellung des hörbaren Audio-Ereignisses im MP3-Player einfacher ist.

Das MP3-Verfahren gestattet gegenüber der Abbildung in .wav-Dateien einen Kompressionsgewinn von bis zu 92 % (1:12).

105. Bei der unter Punkt 97 erwähnten Kompression mit dem **Fraktal-Verfahren** lassen sich Kompressionsgewinne von 99.5 % (1:200) erzielen, sie sind damit etwa doppelt so hoch wie bei Wavelet-Transformationen. Allerdings ist der Rechenaufwand immens. Die Forschung arbeitet an Zeitoptimierungen und sonstigen Verbesserungen, um die Fraktal-Verfahren auch für den flächendeckenden Gebrauch einsetzbar zu machen.