Faltungscodes

- Lineare Abbildung (→ Blockcode) einer Menge an Informationswörtern auf eine Menge von Codewörtern
  - ➔ Aber unendlich lange Code- & Informationswörter → Rahmen ist „unendlich“ lang
- Warum setzt man Faltungscodes ein?
  - ➔ Fehler treten nicht am Stück auf, sondern einzeln an verschiedenen Stellen → Blockcode ist für dieses Fehlermuster nicht gut geeignet
- Coder kann als LTI System mit k Eingängen und n Ausgängen
  - ➔ Coderate $R = \frac{k}{n} < 1$ (wäre es = 1, gäbe es keine Redundanz, die für Kanalcodierung nötig ist)
  - ➔ K = Beeinflussungslänge → Anzahl der verwendeten Stellen im Coder
- Bsp. Wyner-Ash-Code:
  - o $K = 2$ → jedes Bit wird selbst gesendet, danach wird die EXOR Verknüpfung des Bits mit dem Folgebit gesendet usw.
  - o $k = 1, R = \frac{1}{2}$ (es kommen doppelt so viele Wörter aus dem Coder, wie Informationswörter eingehen)
- Bsp. GSM:
  - o Verwendung von verschiedenen Polynomen zur Erzeugung der Wörter im Coder
  - o $K = 5, k = 1, R = \frac{1}{2}$
  - o $G_0(d) = d^4 + d^3 + 1 = 11001, G_1(d) = d^4 + d^3 + d + 1 = 11011$
  - o Coder stellt eine entsprechende Verknüpfung her, dieses passiert auf zwei Polynomstrecken
  - ➔ Nutzdaten werden nie einzeln gesendet, aber einzelne Bits werden durch weiterschieben in den Folgen an verschiedenen Stellen auftauchen
  - o Coderate 1/2 →Es kommt die doppelte Anzahl an Bits aus dem Coder heraus
- Standardtechnik für alle Systeme (Broadcast, Telco), welche Antennen nutzen
- 2. Polynom dienst als Redundanz (könnte auch einfaches Senden der Originaldaten sein)
  - ➔ Dirac-Impuls als Eingangssignal würde die Polynome ausgeben (→ Vgl. FIR Filter, Dirac gibt die Filterkoeffizienten aus)
- Fehler, welche einzeln auftreten sind eher geeignet als Fehler, die in Blöcken auftreten
- Coderate sagt etwas über jene Frequenz am Ende des Coders aus → R = 1/2 → doppelte Frequenz
- Lernbeispiel → EXOR – Verknüpfung durchführen → Prüfung mit verschiedenen Polynomen
- Polynome sind auf beiden Seiten (Encoder und Decoder) vereinbarte Parameter
- Übergangsdiagramm:
  - o Zeigt die gesendeten Worte, welche bei den Übergängen von einem Zustand des Coders auf einen anderen gesendet werden („Wegmarken“)
  - o Man muss die Schreibweise umkehren, wenn man den Ablauf im Signaldiagramm zeigen möchte (ansonsten wird immer anhand der Polynome sortiert)
  - o Ein Übergang kann immer nur eine neue „0“ oder „1“ im Coder bedeuten
  - o Die Übergangswerte haben maximalen Abstand (invertieren ergibt den jeweils anderen Wert) → maximale Hammingdistanz

Fehlererkennung / Fehlerkorrektur

- Durch die festgelegten Zustandsübergange kann der Decoder herausfinden, ob ein Fehler vorliegt (dieser würde sich durch einen Übergangszustand äußern, der momentan nicht vorliegen kann)
- Tritt ein Fehler auf, werden beide möglichen Wege, die zum nachfolgenden korrekten Zustand führen
  - ➔ Man wartet auf das nächste Wort und erhält dadurch eine Info über den möglichen Zwischenverlauf
  - ➔ Daraus kann eine Aussage über den wahrscheinlichsten Weg gemacht werden, der Fehler lässt sich korrigieren
- Bei 2 Fehlern, die auf einander folgen wird verfahren wie bei einem Fehler mit dem Unterschied, dass es nun mehr Wege gibt, die in Frage kommen, der Coder also länger warten muss, um die Fehler zu korrigieren
- Bei 3 Fehlern müssen schon 4 Schritte gegangen werden, um den Weg zu rekonstruieren, hinzu kommt noch, dass es wieder mehrere Wege gibt, die funktionieren
  - ➔ 3 Fehler können nicht zweifelsfrei überwunden werden

Viterbi-Algorithmus

- Suche nach Codefolge x, die von der empfangenen Codefolge y die geringste Hamming-Distanz hat

  $x = \{x_1, x_2, x_3, ..., x_n\}$, $y = \{y_1, y_2, y_3, ..., y_n\}$

  $$\lambda_i = \begin{cases} 0 & für \;\; x_i \neq y_i \\ 1 & für \;\; x_i = y_i \end{cases}$$

  - ➔ Vergleich der einzelnen Symbole, ob diese in beiden Folgen vorkommen
  - ➔ Es werden alle Folgen durchgespielt und geprüft, welche die größte Übereinstimmung hat
- Bsp. mit Codierung von 3 Bit in 2 Bit (Polynome: $G_1(d) = d^2 + 1,\;\; G_2(d) = d^2 + d + 1$)
  - Darstellung im Trellis-Diagramm → Versuch, einen Decoder nachzustellen
  - Nutzung der durch den Viterbi-Algorithmus vorgegebenen Metriken zur Bestimmung der kürzesten Strecke
  - Die Wege mit den niedrigsten Metriken an einem Knotenpunkt werden ausgenommen
  - Sollten beide Strecken dieselbe Metrik haben, streicht man beliebig eine der beiden
  - Bsp. 7 Werte werden gesendet → kürzester Weg ohne Fehler hätte die Metrik 14
  - Bsp. der Favorit hat die Metrik 12, daher sind 2 Fehler vorhanden → Korrektur kann durch die Position (Erhöhung der Metrik um einen Wert < 2) eingeleitet werden
- In der Praxis nutzt man einen Faktor 10 – 100 von K (Beeinflussungslänge), bevor entschieden wird, welcher Weg der Favorit ist (Gedächtnis + Verzögerung im System)
- Fehlerdichte → Veranschaulichung in Matlab:
  - 2. Und 5. Stelle von 14 erhalten einen Fehler → Beeinflusst nicht, kann korrigiert werden
  - Weiterer Fehler an Stelle 12 → Beeinflusst nicht, kann korrigiert werden
  - Fehler von Stelle 12 auf Stelle 6 → Beeinflussung beim ersten Wert sichtbar, danach keine Beeinflussung

➔ Es ist nicht die Fehlerrate entscheidend, sondern die Dichte der Fehler

- Vgl. Blockcode → es wird ein falsches Wort als richtiges erkannt, statt einen Fehler zu erkennen
- Modellbildung mit zufällig verteilten Fehlern → Veranschaulichung in Matlab:
  ➔ Viterbi Decoder kann bei ausreichendem Abstand zwischen den Fehlern reparieren/rekonstruieren, bei zu dichten Fehlern kommen nicht korrigierbare Wörter heraus
- In der Praxis wird zu gegebenen Systemen ein Fehlermodell erstellt, an welches die Codierung angepasst wird → Fehlermodelle entsprechen in den meisten Fällen nicht der Realität (→ stochastische Verteilung) → jede Strecke hat ihre eigenen Fehlerquellen und daher auch eine andere Fehlerquote