Überlegungen zum Wyner-Ash-Code

Fehlerabstand $d = 4$, Fehler auf den Verknüpfungsstellen:

Daten:	0 0 0 0 0 0 0 0 0 0	
Codiert:	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
2-Fehler:	0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	
Decodiert:	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	→ 2 Stellen verändert
	0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0	→ 4 Stellen verändert
	0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0	→ 2 Stellen verändert

➔ Es gibt 2 Fälle, deren Korrekturaufwand gleich ist, in denen der Codierer also per Zufall entscheiden müsste, welche Korrekt ist. Der Abstand von 4 würde diese Situation auch vorgeben, weil es einen Zustand gibt, welcher nicht korrigiert werden kann. Diesen sieht man in diesem Beispiel.

Fehlerabstand $d = 5$:

2-Fehler:	0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0	
Decodiert:	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	→ 2 Stellen verändert
	0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0	→ 4 Stellen verändert

➔ In diesem Fall sind die Fehler weit genug voneinander entfernt, sodass es eine eindeutig beste Lösung zur Korrektur gibt.

Fehlerabstand $d = 4$, Fehler auf den Bitstellen:

2-Fehler:	0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0	
Decodiert:	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	→ 2 Stellen verändert
	0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0	→ 4 Stellen verändert

➔ In diesem Fall funktioniert die Korrektur und es kann ein eindeutiger Pfad gefunden werden.

➔ Fehler dürfen sich im Viterbi-Decoder nicht zu nah kommen, weil dann nicht mehr eindeutig Fehler erkannt und dann entsprechend korrigiert werden können.

Simulation des Viterbi-Decoders in Matlab:

- Verschiedene Betrachtungen zeigen, dass nicht immer alles an Fehlern vom Decoder beseitigt wird. Der Idealfall von Fehlern für den Viterbi-Decoder sind gleichmäßig verteilte Fehler mit genügend Abstand zu einander
- In der Realität liegen Fehler mal weiter aus einander, mal enger zusammen
- die Codierung passt gut zum statistischen Rauschen im Kanal

➔ Längere Fehlerfolgen können nicht korrigiert werden → in der Praxis wird diese Codierung z.B. auf Satellitenstrecken angewendet (ca. 120 dB Streckendämpfung)

Ergebnisse aus der Simulation:

Generatorpolynome	[101] [111]	[1001] [1111]
Ordnung	2	3
Coderate	$\frac{1}{2}$	$\frac{1}{2}$
aus 10 % Fehlerrate	6,2 %	7,2 %
aus 5 % Fehlerrate	0,8 %	0,6 %
aus 1 % Fehlerrate	0,0 %	0,0 %

➔ Man benötigt im Kanal eine gewisse Mindestqualität, damit der Decodierer besser arbeiten kann.

➔ Längere Polynome wirken sich nur dann vorteilhaft aus, wenn diese Mindestqualität gegeben ist.

Aufgabe: Wie wirken sich größere Gedächtnisordnungen und das Hinzufügen von Polynomen auf die folgenden Aspekte aus: Korrektureigenschaften, Rechenaufwand/Speicher, Bandbreitenbedarf ?

	Korrektureigenschaften	Rechenaufwand	Bandbreitenbedarf
Erhöhen der Anzahl der Polynome	Größere Wegmarken → größere Distanz ➔ Verbesserung	wächst leicht linear an (mehr Stellen im Trellisdiagramm)*	erhöht → wächst linear
Erhöhen der Gedächtnisordnungen	Größere Verteilung → Bündelfehler können besser korrigiert werden ➔ Verbesserung	+1 Punkt → doppelte Anzahl Möglichkeiten im Trellisdiagramm → exp. Anstieg	bleibt gleich

* Es sind zwar mehr Stellen im Trellisdiagramm, die Wegmöglichkeiten sind aber dieselben und der Rechenaufwand wird nur durch einige Vergleiche, die mehr gemacht werden müssen, erhöht.

Punktierung von Faltungscodes:

- Coderate von z.B. 1/2 in Richtung 1 verändern → durch Punktieren

➔ Man streicht Codebits anhand einer Maskierung heraus

$k = 8 \rightarrow n = 2 \cdot 8$ (nicht punktiert) → $R = \frac{1}{2}$

$k = 8 \rightarrow n = 2 \cdot 5$ (punktiert) → $R = \frac{4}{5}$

- Der Code lässt sich skalieren, allerdings ist die Auswahl der genutzten Stellen variabel
- Im Trellisdiagramm bleibt die gestrichene Stelle frei → es gibt weder Punkte für korrekt erkannte Stelle noch für fehlerhafte Stelle

➔ Die Metrik für den korrekten Pfad wird kleiner, die Abstände zu den anderen nicht korrekten Pfaden bleiben im Idealfall erhalten (man geht davon aus, dass es alle Pfade gleich trifft, was nicht immer der Realität entspricht)

➔ Clusterbetrachtung: Telco und Broadcast fortgeschritten, Productivity in der Anfangsphase

Reed-Solomon-Codes

- Sichere Speicherung von Daten durch Verfahren mit Kanalcodierung → leistungsfähige Codes werden benötigt
- RS-Code ist ein leistungsfähiger Code
- Verwendung eines Gallois-Feldes mit den Eigenschaften eines festen Zahlenraums

➔ Nullstellen werden als signifikante Punkte der Codierung genutzt

Galloisfeld:

- Begrenztes „finites" Feld mit q verschiedenen Elementen
- Bei Addition & Multiplikation → Anwendung auf 2 Feldelemente ergibt wieder ein Feldelement (Überschreiten der Grenzen des Feldes wird durch z.B. Modulo Rechnung auf das Feld zurück projetziert)
- Neutrale Elemente: Addition → 0, Multiplikation → 1
- Zu jedem Element existiert ein Inverses für Addition und Multiplikation
- Man nutzt Polynome des Grads < w für ein Galloisfeld $GF\left(2^w\right)$ mit Koeffizienten 0 oder 1
- Definition:
  - Erstes Element ist eine Nullfolge → 0
  - Alle weiteren werden durch das Generatorpolynom und ein Primitives Element festgelegt
- Beispiel: Erstellung eines Galloisfeldes

$$g\left(x\right)=x^3+x+1,\quad \alpha=x\left(\textit{primitives Element}\right)\Rightarrow GF\left(2^3\right)$$

0	000
$\alpha^0=1$	001
$\alpha^1=x$	010
$\alpha^2=x^2$	100
$\alpha^3=x^3=x+1$	011
$\alpha^4=x^4=x^2+x$	110
$\alpha^5=x^5=x^2+x+1$	111
$\alpha^6=x^6=x^2+1$	101

Die Werte für α^n ergeben sich aus der Division der Polynome mit dem Generatorpolynom. Weil 2^3 festgelegt wurde, sind im Feld nur Polynome des Grades < 3 zu finden.

- Rechenregeln:
  - Addition → XOR Verknüpfung der binären Werte → 011 XOR 110 = 101 ($x^3+x^4=x^6$)
  - Multiplikation → Addition der Exponenten mit anschließender Modulo 7 Rechnung (Wert der Modulo-Rechnung ist die Feldgröße -1, das Nullelement kann nicht erreicht werden) → $x^4\cdot x^5\rightarrow 4+5=9 \bmod 7=2\rightarrow x^2$
- Jeder Koeffizient ist ein Polynom in einem übergeordneten Polynom (→ Mehrwertigkeit)
- Daten kommen so wie sie kommen → werden in gleiche Abschnitte unterteilt (hier 3-er Abschnitte)
- Es liegen im übergeordneten Polynom evtl. Nullstellen vor, welche als Orientierung dienen können

Übung: Ermittlung der Nullstellen eines übergeordneten Polynoms

Datenstrom:	100	011	101	000	111	010	110
	$\alpha^2 \cdot X^6$	$\alpha^3 \cdot X^5$	$\alpha^6 \cdot X^4$	$0 \cdot X^3$	$\alpha^5 \cdot X^2$	$\alpha^1 \cdot X$	α^4
$X = \alpha$	α^1	α^1	α^3	0	α^0	α^2	α^4
$X = \alpha^2$	α^0	α^6	α^0	0	α^2	α^3	α^4

Rechnung für $X = \alpha$:

$$\begin{array}{r} 010 \\ +\,010 \\ +\,011 \\ +\,000 \\ +\,001 \\ +\,100 \\ \underline{+\,110} \\ 000 \end{array}$$

Rechnung für $X = \alpha^2$:

$$\begin{array}{r} 001 \\ +\,101 \\ +\,001 \\ +\,000 \\ +\,100 \\ +\,011 \\ \underline{+\,110} \\ 100 \end{array}$$

- Potenzen werden der Reihe nach auf Nullstellen geprüft ($\alpha^0 - \alpha^6$)
- Berechnung ist eine Teilberechnung der DFT, allerdings nur für eine spektrale Komponente
  $A_l = \sum_{i=0}^{N-1} a_i \cdot \alpha^{il}$ für verschiedene Werte von l durchrechnen wäre die DFT
- Systematisches Durcharbeiten zur Ermittlung von Nullstellen (z.B. Nullstelle bei l = 1 → α^1)

➔ Nullstellen werden als Gesamtaussage genutzt → alle Anteil in den Polynomen sind beteiligt → jede Stelle des Codewortes trägt dazu bei

➔ Optimierung: Es werden Nullstellen als Redundanz hinzugefügt

- Datenwort + Nullstellen wird als Spektralbereich angesehen → IDFT wird durchgeführt → Codewort im Zeitbereich (Nullstellen sind garantiert vorhanden)
- Nach dem Empfang und der „Rücktransformation“ (dies ist in diesem Fall die DFT) wird erwartet, das wieder genauso viele Nullstellen vorhanden sind, wie hinzugefügt worden sind

➔ Es interessieren für die Betrachtung nur jene Nullstellen, die hinzugefügt worden sind

- Es können natürlich im Datenstrom auch Nullstellen vorliegen, dies ist aber eher selten der Fall (→ es werden größere Blöcke betrachtet)
- Bsp. zum Verfahren der Kanalcodierung mit RS-Code
  255 Byte zusammen betrachtet → 16 Byte Nullstellen, d.h. man kann 8 Byte korrigieren

➔ Es können 8 Symbole korrigiert werden, dabei spielt keine Rolle, wie viele Bits im Symbol falsch sind

- RS-Code nutzt die beim Faltungscode nicht funktionierenden, eng auf einander folgenden Fehler und korrigiert diese entsprechend gut
- Fehler, die verteilt sind (gut für den Viterbi-Coder), sind nicht gut für den RS-Coder geeignet

➔ RS-Code für singuläre Events (z.B. nach der Anwendung eines Faltungscoders)