11 Faltungscodes

11 Faltungscodes

Die nicht-systematische Codierung für zyklische Codes erfolgt durch Multiplikation

$$x(\chi) = a(\chi) \cdot g(\chi) \tag{11.1}$$

$$\sum_{i=0}^{n-1} x_i \chi^i = \left(\sum_{i=0}^{l-1} a_i \chi^i \right) \cdot \left(\sum_{i=0}^{k} g_i \chi^i \right). \tag{11.2}$$

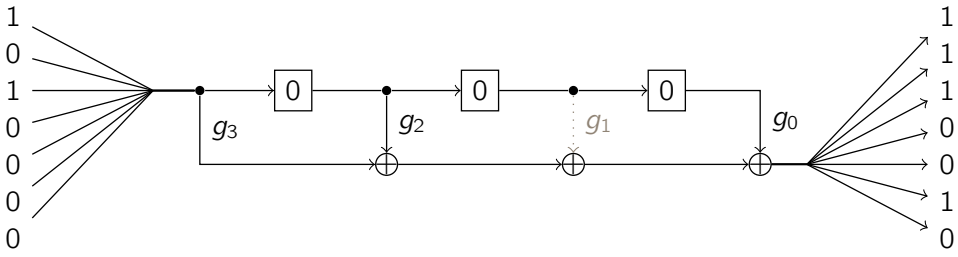
Die Codewort-Bits ergeben sich dementsprechend gemäß

$$x_j = \sum_{i=0}^{k} a_{j-i} g_i, \tag{11.3}$$

wenn $a_i = 0$ für $i < 0$ bzw. $i \geq l$ gesetzt wird, was einer Faltungsoperation entspricht. Die Codierung kann also erfolgen, indem eine Sequenz aus den l Nutzbits a_i und k Nullen gebildet wird, und diese mit g_i gefaltet wird.

Beispiel 11.1 (Codierung durch Faltung)

Es soll das Nutzdatenwort $\boldsymbol{a} = 1010$ mit dem Generatorpolynom $g(\chi) = \chi^3 + \chi^2 + 1$ codiert werden.



Das Codewort lautet somit 1110010.

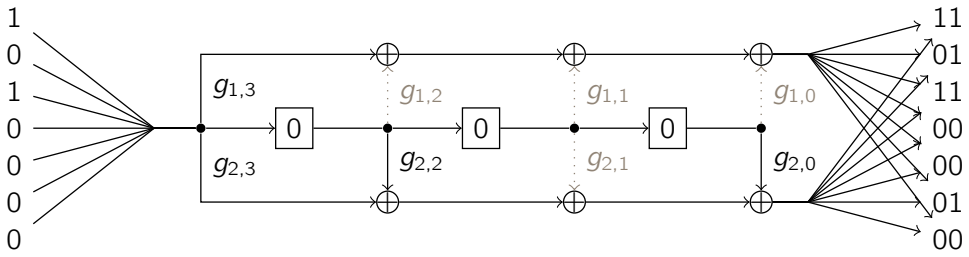
Der Übergang von den zyklischen Codes zu Faltungscodes umfasst zwei Schritte:

1. Es wird $l \gg k$ gewählt, womit die Coderate R gegen eins geht, die Fehlerkorrektureigenschaften also beliebig schlecht werden. (Es ist sogar $l \to \infty$, also eine blockfreie Codierung möglich.)
   Nichtsdestoweniger wird durch die Faltungsoperation gezielt eine Abhängigkeit aufeinanderfolgender Codebits erzeugt.
2. Es wird mit m (typisch zwei bis drei) Polynomen parallel gefaltet und die jeweiligen Teilergebnisse zu einem Codewort zusammengefasst. So ergibt sich eine Coderate von $R \approx \frac{1}{m} < 1$, die eine Fehlerkorrektur ermöglicht.

Wird eins der Polynome als χ^k gewählt, erhält man einen systematischen Code.

Beispiel 11.2

Es soll das Nutzdatenwort $\boldsymbol{a} = 1010$ mit den Generatorpolynomen $g_1(\chi) = \chi^3$ und $g_2(\chi) = \chi^3 + \chi^2 + 1$ codiert werden.



Das Codewort lautet somit 11 01 11 00 00 01 00.
Aus Gründen der Übersichtlichkeit müssen sich die Beispiele auf kleine l beschränken, weshalb die Coderate hier mit $R = \frac{4}{14}$ deutlich unter $R \approx \frac{1}{2}$ für große l liegt.

11.1 Freier Abstand und Darstellung als Zustandsdiagramm

Für die Beurteilung der Fehlerkorrekturfähigkeit eines Codes C ist der minimale Abstand

$$d_{\min}(C) = \min_{\substack{\boldsymbol{x},\boldsymbol{x}' \in C \\ \boldsymbol{x} \neq \boldsymbol{x}'}} \mathrm{d}(\boldsymbol{x}, \boldsymbol{x}') \tag{11.4}$$

von Interesse. (Im Zusammenhang mit Faltungscodes spricht man auch vom freien Abstand.)
Da Faltungscodes lineare Codes sind, reicht es aus, das Codewort minimalen Gewichts zu ermitteln, also den minimalen Abstand gemäß

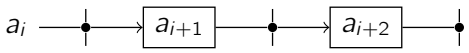
$$d_{\min}(C) = \min_{\substack{\boldsymbol{x} \in C \\ \boldsymbol{x} \neq \boldsymbol{0}}} \mathrm{d}(\boldsymbol{x}, \boldsymbol{0}) \tag{11.5}$$

zu bestimmen.
Zur Bestimmung des freien Abstands ist die Betrachtung des Zustandsdiagramms zweckmäßig.

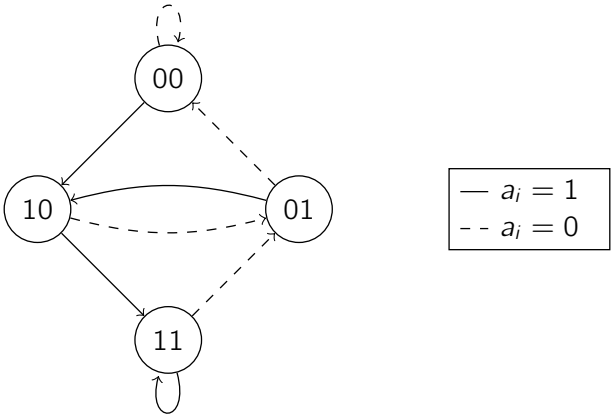
Ein Faltungscoder besitzt k binäre Zustandsvariablen, befindet sich also in einem von 2^k Zuständen. Das Einspeisen eines Bits führt zu einem Zustandsübergang und der Ausgabe von m Codebits.
Die Zustandsübergänge hängen nur von k, nicht von den Generatorpolynomen ab.

Beispiel 11.3 (Zustandsdiagramm für $k = 2$)

Zu dem Schieberegister

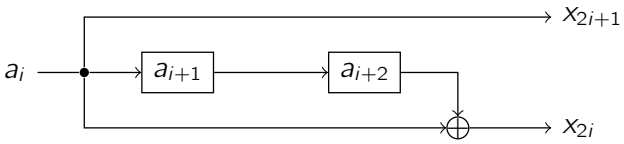


mit $k = 2$ Zustandvariablen gehört das Zustandsdiagramm

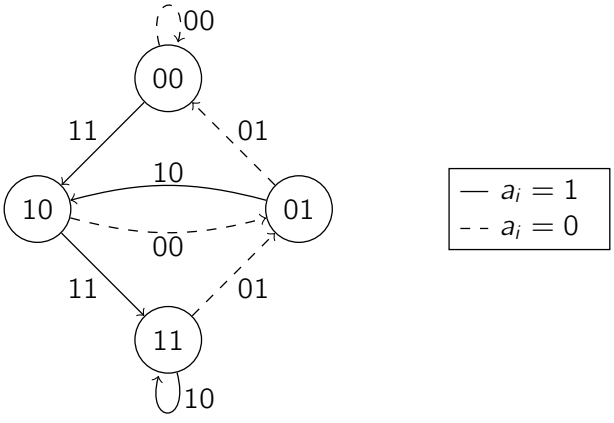


Beispiel 11.4 (Zustandsdiagramm mit Ausgabebits)

Zu dem Faltungscode mit den Generatorpolynomen $g_1(X) = X^2$ und $g_2(X) = X^2 + 1$, also



gehört das Zustandsdiagramm

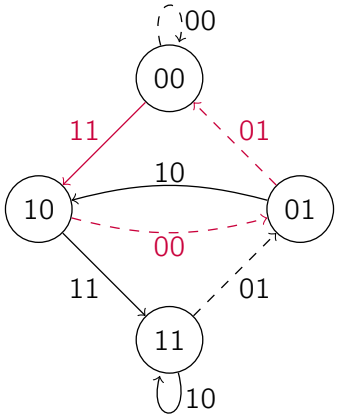


Mit Hilfe des Zustandsdiagramms kann der freie Abstand besimmt werden. Dazu ist im Zustandsdiagramm derjenige Pfad zu bestimmen, der

- den Nullzustand mit dem Nullzustand verbindet
- mindestens einen anderen Zustand enthält
- die geringste Anzahl Einsen in der Ausgabe erzeugt.

Diese Anzahl an Einsen ist dann die minimale Anzahl an Einsen in einem (von **0** verschiedenen) Codewort, also der freie Abstand.

Beispiel 11.5 (Freier Abstand des Codes aus Beispiel 11.4)



Das Codewort mit der minimalen Anzahl an Einsen lautet also
00 . . . 00 11 00 01 00 . . . 00, der freie Abstand ist folglich 3.

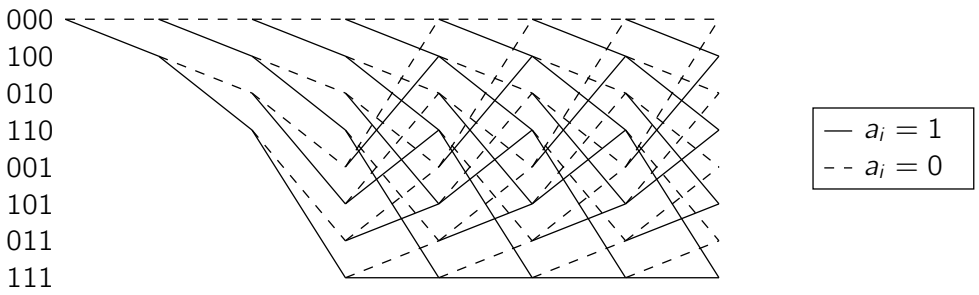
Der freie Abstand (und damit die Anzahl korrigierbarer Fehler) von Faltungscodes hängt von den Generatorpolynomen ab, nicht jedoch von der Codewortlänge n. Je länger die Codewörter sind, desto geringer muss demnach die Fehlerrate des Kanals sein, um eine fehlerfreie Decodierung zu erlauben. *Sind Faltungscodes also für große Codewortlängen n unbrauchbar?*
Codewörter mit wenigen Einsen entsprechen immer einem Pfad im Zustandsdiagramm, der nur relativ wenige Übergänge außerhalb des Nullzustands macht[4]. Nur auf diesen liegen die Eins-Bits, im Codewort sind sie folglich auf einen kleinen Bereich konzentriert.
Faltungscodes können also in der Regel deutlich mehr Fehler korrigieren, als der freie Abstand erwarten lässt, vorausgesetzt, die Fehler sind über die gesamte Länge des Codeworts verteilt.
Nichtsdestotrotz ist der freie Abstand eins der wichtigsten Entwurfskriterien für Faltungscodes. Leider gibt es kein systematisches Vorgehen zum Entwurf von Codes, die bei gegebenem m und k den freien Abstand maximieren. In der Literatur finden sich aber Tabellen, die die jeweils besten, durch exhaustive Suche gefundenen, Codes auflisten.

[4] Dies setzt voraus, dass das Zustandsdiagramm keinen Zyklus aufweist, der ausschließlich Nullen ausgibt, mit Ausnahme des Übergangs vom Nullzustand zu sich selbst. Ein Code, der einen solchen Zyklus besitzt, wird *katastrophal* genannt und sollte vermieden werden.

11.2 Darstellung als Trellis

Das Trellisdiagramm stellt wie das Zustandsdiagramm die möglichen Zuständsübergänge dar, bildet aber explizit die zeitliche Dimension mit ab. Dazu werden die Zustände auf der Vertikalen, die zeitliche Abfolge auf der Horizontalen aufgetragen werden.

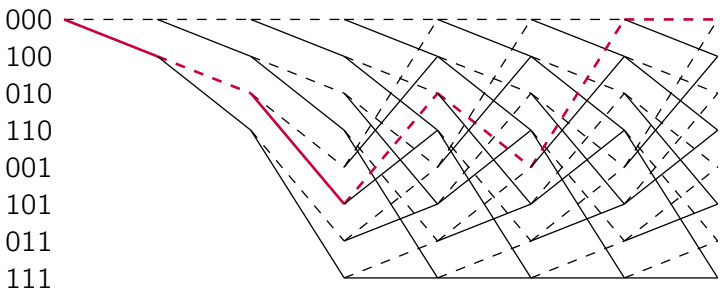
Beispiel 11.6 (Trellis für $k = 3$)



Jedes Nutzdatenwort entspricht einem Pfad durch das Trellis.

Beispiel 11.7

Das Wort $\boldsymbol{a} = 1010$ (gefolgt von drei Nullbits) entspricht folgendem Pfad im Trellis:



11.3 Decodierung mit dem Viterbi-Algorithmus

Für eine fehlerkorrigierende Decodierung ist der Pfad durch das Trellis zu bestimmen, dessen Codewort $\boldsymbol{x}$ den geringsten Abstand zum Empfangswort $\boldsymbol{y}$ aufweist. Aus den damit verbundenen Zustandsübergängen lässt sich das Datenwort $\boldsymbol{a}$ gewinnen.
Auch wenn berücksichtigt wird, dass die letzten k codierten Bits Nullen sind, verbleiben 2^l zu überprüfende Pfade, was für große l schnell unpraktikabel wird.
Dem Viterbi-Algorithmus liegt folgende Überlegung zu Grunde, um den Rechenaufwand zu reduzieren:

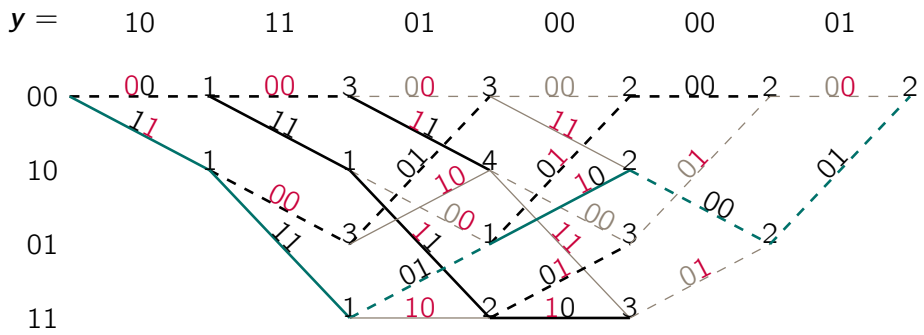
- Wird ein Pfad $\boldsymbol{x}$ durch das Trellis an einem beliebigen auf ihm liegenden Zustand σ geteilt, so ist der Gesamtabstand zu $\boldsymbol{y}$ die Summe der beiden Abstände des linken und des rechten Teilpfades zu den entsprechenden Abschnitten von $\boldsymbol{y}$.
- Ist $\boldsymbol{x}$ der Pfad mit dem geringsten Abstand zu $\boldsymbol{y}$, so ist also der Teilpfad bis zum Zustand σ auch derjenige unter allen Pfaden bis σ, der den geringsten Abstand zum entsprechenden Abschnitt von $\boldsymbol{y}$ hat.
- Wird bei der Decodierung schrittweise durch das Trellis gegangen, muss daher pro Zustand nur der bis dahin beste Pfad beibehalten werden, bei 2^k Zuständen also auch nur 2^k Pfade. Da $k \ll l$ bedeutet dies eine deutliche Reduktion des Rechenaufwands.

Beispiel 11.8

Es soll das Wort $\boldsymbol{a} = 1101$ mit $g_1(\chi) = \chi^2$ und $g_2(\chi) = \chi^2 + 1$ codiert werden.

Eingabe	1	1	0	1	0	0
Ausgabe 1	1	1	0	1	0	0
Ausgabe 2	1	1	1	0	0	1

Es wird also das Codewort $\boldsymbol{x} = 11\,11\,01\,10\,00\,01$ gebildet.
Empfangen werde das gestörte Codewort $\boldsymbol{y} = 10\,11\,01\,00\,00\,01$, das mit dem Virterbi-Algorithmus decodiert werden soll:



Es wird also korrekt zu $\hat{\boldsymbol{x}} = 11\,11\,01\,10\,00\,01$ bzw. $\hat{\boldsymbol{a}} = 1101$ decodiert.

Der Viterbi-Algorithmus lässt sich also wie folgt formulieren:

1. Initialisiere $\hat{\boldsymbol{a}}_{1,1} = \epsilon$ (leere Folge), $d_{1,1} = 0$, $i = 1$.
2. Erhöhe i um eins.
3. Bestimme für jeden möglichen Zustand $j = 1, \ldots, 2^k$ denjenigen Vorzustand

$$\hat{j} = \arg\min_{j' \in \Sigma_j} d_{j',i-1} + \mathrm{d}(\boldsymbol{y_i}, \boldsymbol{x}_{j',j}), \tag{11.6}$$

   unter den beiden möglichen Vorzustängen Σ_j, sodass die Summe aus Abstand $d_{\hat{j},i-1}$ bis zu diesem Vorzustand und der Abstand des i-ten Abschnitts $\boldsymbol{y_i}$ der Empfangsfolge zur Ausgabe $\boldsymbol{x}_{\hat{j},j}$ des Zustandsübergangs von $\hat{j}$ nach j minimal wird.
   Das Abtandsmaß $\mathrm{d}(\cdot)$ kann
   - die Hamming-Distanz sein
   - erkannte Fehler berücksichtigen (Abstand zu ? immer Null)
   - oder sich auf das analoge Signal beziehen (z.B. euklidscher Abstand), also implizit eine "weiche" Entscheidung durchführen.

   Bestimme mit diesem $\hat{j}$ für jedes j

$$d_{j,i} = d_{\hat{j},i-1} + \mathrm{d}(\boldsymbol{y_i}, \boldsymbol{x}_{\hat{j},j}), \tag{11.7}$$

   und bilde $\hat{a}_{j,i}$ aus $\hat{a}_{\hat{j},i-1}$ durch anhängen einer 0 oder 1, entsprechend dem gewählten Zustandsübergang.
4. Wiederhole ab 2, bis Codewortende erreicht.

Die Laufzeit des Viterbi-Algorithmus steigt

- exponentiell mit dem Grad k der verwendeten Polynome (da für 2^k Zustände Berechnungen durchgeführt werden müssen
- linear mit der Codewortlänge n.

Bei konstantem k bleibt der Rechenaufwand pro Bit also konstant, die Codewortlänge wird nur durch den benötigten Speicher oder andere Anforderungen begrenzt.
Leider erlaubt die Wahl ausreichend großer n nicht das Erreichen der Shannon-Grenze, da Faltungscodes empfindlich gegenüber Bündelfehlern sind: Treten zu viele Fehler in einem kurzem Abschnitt auf, so kann der optimale Pfad durch das Trellis in diesem Bereich leicht falsch sein.

11.4 Punktierung

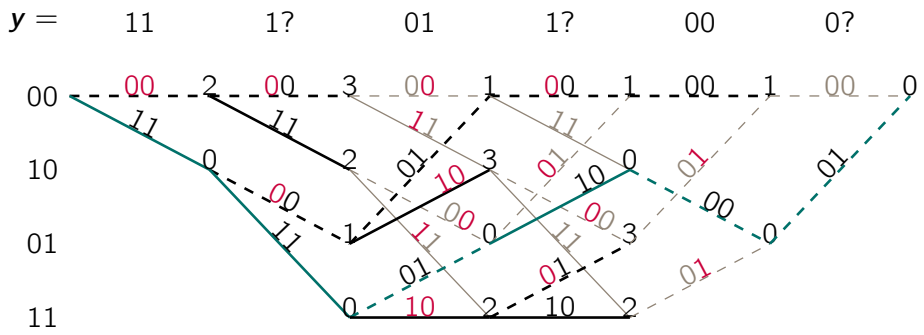
Die Rate eines Faltungscodes mit $m \geq 2$ Polynomen vom Grad k ergibt sich zu

$$R = \frac{l}{(l+k)\cdot m} \approx \frac{1}{m}. \tag{11.8}$$

Eine flexiblere Wahl der Coderate, insbesondere auch von Raten $> \frac{1}{2}$, lässt sich durch eine Punktierung erreichen.
Bei der Punktierung werden aus dem Codewort vor der Übertragung einzelne Bits entfernt, um so die Codewortlänge auf das gewünscht Maß zu reduzieren.
Bei Decodierung werden die entfernten Bits wie erkannte Fehler gehandhabt, also ? an den entsprechenden Stellen eingefügt.

Beispiel 11.9

Es soll wieder das Wort $\boldsymbol{a} = 1101$ mit $g_1(\chi) = \chi^2$ und $g_2(\chi) = \chi^2 + 1$ codiert werden. Wie in Beispiel 11.8 gezeigt, führt dies auf das Codewort $\boldsymbol{x} = 11\,11\,01\,10\,00\,01$. Durch eine Punktierung jedes vierten Bits entsteht dadurch das verkürzte Codewort $11\,1\,01\,1\,00\,0$, das gesendet wird.
Angenommen, es treten keine Übertragungsfehler auf, so muss der Decoder $\boldsymbol{y} = 11\,1?\,01\,1?\,00\,0?$ decodieren. Der Viterbi-Algorithmus liefert:



Es wird also korrekt zu $\hat{\boldsymbol{x}} = 11\,11\,01\,10\,00\,01$ bzw. $\hat{\boldsymbol{a}} = 1101$ decodiert.

11.5 Interleaving

Faltungscodes

- können in der Regel wesentlich mehr Fehler korrigieren, als ihr freier Abstand vermuten lässt, solange die Fehler gleichmäßig über das gesamte Codewort verteilt sind,
- sind aber empfindlich gegenüber Bündelfehlern.

Leider treten Bündelfehler in der praktischen Anwendung häufig auf (z.B. Fading-Kanäle). Daher ist es wünschenswert, Faltungscodes gegen Bündelfehler robuster zu machen; dies kann über sogenanntes Interleaving erreicht werden. Dabei wird das Codewort vor der Übertragung durch Umsortierung "verwürfelt", vor der Decodierung wird die ursprüngliche Reihenfolge wiederhergestellt.

Beispiel 11.10

Eine sehr einfache Möglichkeit, einen Interleaver zu konstruieren, besteht darin, die Daten zeilenweise in eine Matrix geeigneter Größe zu schreiben und spaltenweise wieder auszulesen. Diese Art des Interleavens führt für $\boldsymbol{x} = 111101100001$ auf:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

also 100110110101.
Angenommen, nach einer Übertragung und Decodierung führt ein Bündelfehler auf 101000110101. Nach dem De-Interleaven wird daraus 101100101001, die Fehler sind also über das Wort verteilt.

11.6 Zusammenfassung

- Faltungscodes führen die Codierung durch eine fortlaufende Faltung mit mehreren Generatorpolynomen durch.
- Damit kann die Codewortlänge n beliebig groß gewählt werden. Die Coderate hängt im wesentlichen von der Anzahl m der verwendeten Polynome ab; für $n \to \infty$ gilt $R = \frac{1}{m}$.
- Durch Punktierung kann die Coderate noch flexibler gewählt werden.
- Der minimale Abstand (freie Abstand) bei Faltungscodes ist von der Codewortlänge n unabhängig, sodass auch bei sehr langen Codewörtern verhältnismäßig wenige Fehler ausreichen, um einen Decodierfehler zu verursachen. Dazu müssen die Fehler aber konzentriert in einem kleinen Bereich des Codeworts auftreten (Bündelfehler); andernfalls sind die Korrektureigenschaften von Faltungscodes deutlich besser.
- Die auf dem Trellis-Diagramm beruhende Viterbi-Decodierung von Faltungscodes hat eine Rechenzeit, die zwar exponentiell vom Grad k der Generatorpolynome, aber nur linear von der Codewortlänge n abhängt. Bei festem k bleibt der Rechenaufwand pro Bit also konstant.