12 Turbo-Codes

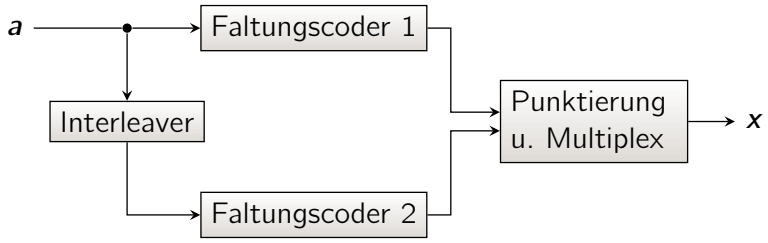
Faltungscodes

- lassen sich zu vorgegebenen Coderaten entwerfen
- erlauben beliebig lange Codewörter
- können mit dem Viterbi-Algorithmus effizient decodiert werden
- haben verhältnismäßig gute Fehlerkorrektureigenschaften
- sind aber empfindlich gegenüber Bündelfehlern.

Bündelfehler führen bei der Decodierung zu einer zeitlich begrenzten Abweichung vom korrekten Pfad durch das Trellis, also in den decodierten Nutzdaten wieder zu einem Bündelfehler.

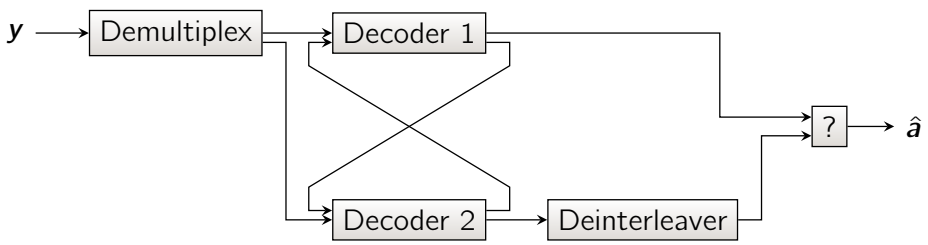
12.1 Codierung

Die Kernidee von Turbo-Codes ist die Verwendung von zwei parallel arbeitenden Faltungscodes.
Es kommen systematische Faltungscodes zum Einsatz; die in beiden Codewörtern enthaltenen Nutzdaten werden nur einmal übertragen. Durch zusätzliche Punktierung kann die Coderate weiter erhöht werden. (Typisch: Aus zwei halbratigen Faltungscodes wird ein halbratiger Turbo-Code.)
Damit Bündelfehler im Kanal nicht zu einer Störung beider Komponentencodes führen, werden die Nutzdaten vor der Codierung mit dem zweiten Faltungscoder durch einen Interleaver verwürfelt.



12.2 Das Turbo-Prinzip

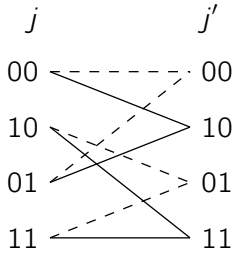
Werden beide Komponentencodes des Turbo-Codes einzeln decodiert, so werden – dank des Interleavens – Fehler bei der Decodierung mit hoher Wahrscheinlichkeit an verschiedenen Stellen auftreten.
Um zu entscheiden, welcher Decoder die endgültige Ausgabe bestimmt, müssen die Komponentendecoder "weiche Entscheidungen" liefern, also Wahrscheinlichkeiten statt entschiedenen Bit-Werten.
Eine weitere Verbesserung wird ereicht, wenn die Informationen des einen Decoders beim jeweils anderen verwendet werden. Dadurch wird den Decodern geholfen, im Bereich von Bündelfehlern den richtigen Pfad durch das Trellis zu finden.



Die Ähnlichkeit dieser Verwertung von Ausgangsprodukten zur Aufwertung des Eingangs mit Turboladern hat den Turbo-Codes ihren Namen gegeben.

12.3 Die Komponentendecoder

Die Komponentendecoder müssen zu jedem Nutzbit a_i die Wahrscheinlichkeit bestimmen, mit der dieses 1 ist.
Es sei das i-te Trellis-Segment betrachtet.



Daraus ergibt sich die Wahrscheinlichkeit

$$P(a_i = 1|\boldsymbol{y}) = \frac{P(a_i = 1, \boldsymbol{y})}{P(\boldsymbol{y})} = \frac{1}{P(\boldsymbol{y})} \sum_{(j,j')\Rightarrow 1} P(\boldsymbol{y}, j, j'). \tag{12.1}$$

Wir teilen die Empfangssequenz $\boldsymbol{y}$ auf in den Teil $\boldsymbol{y}_{<i}$ vor dem i-ten Schritt, den Teil $\boldsymbol{y}_{>i}$ nach dem i-ten Schritt und den Teil $\boldsymbol{y}_i$ zum i-ten Schritt.
Damit lässt sich umschreiben zu

$$P(a_i = 1|\boldsymbol{y}) = \frac{1}{P(\boldsymbol{y})} \sum_{(j,j')\Rightarrow 1} P(\boldsymbol{y}, j, j') = \frac{1}{P(\boldsymbol{y})} \sum_{(j,j')\Rightarrow 1} P(\boldsymbol{y}_{<i}, \boldsymbol{y}_i, \boldsymbol{y}_{>i}, j, j') \tag{12.2}$$

Nach der Bayes-Regel lässt sich der Term unter Summe umformen zu

$$P(\boldsymbol{y}_{<i}, \boldsymbol{y}_i, \boldsymbol{y}_{>i}, j, j') = P(\boldsymbol{y}_{>i}|\boldsymbol{y}_{<i}, \boldsymbol{y}_i, j, j') \cdot P(\boldsymbol{y}_{<i}, \boldsymbol{y}_i, j, j'). \tag{12.3}$$

Ist der Zustand j' nach dem i-ten Schritt bekannt, so ist $\boldsymbol{y}_{>i}$ von allen Vorgängen bis inklusive des i-ten Schritts unabhängig, also

$$P(\boldsymbol{y}_{<i}, \boldsymbol{y}_i, \boldsymbol{y}_{>i}, j, j') = P(\boldsymbol{y}_{>i}|j') \cdot P(\boldsymbol{y}_{<i}, \boldsymbol{y}_i, j, j'). \tag{12.4}$$

Nochmalige Anwendung der Bayes-Regel führt dann auf

$$P(\boldsymbol{y}_{<i}, \boldsymbol{y}_i, \boldsymbol{y}_{>i}, j, j') = P(\boldsymbol{y}_{>i}|j') \cdot P(\boldsymbol{y}_i, j'|\boldsymbol{y}_{<i}, j) \cdot P(\boldsymbol{y}_{<i}, j). \tag{12.5}$$

$$P(\boldsymbol{y}_{<i}, \boldsymbol{y}_i, \boldsymbol{y}_{>i}, j, j') = P(\boldsymbol{y}_{>i}|j') \cdot P(\boldsymbol{y}_i, j'|\boldsymbol{y}_{<i}, j) \cdot P(\boldsymbol{y}_{<i}, j).$$

Ist der Zustand j vor dem i-ten Schritt bekannt, so hängen j' und $\boldsymbol{y}_i$ nicht mehr von $\boldsymbol{y}_{<i}$ ab, also

$$P(\boldsymbol{y}_{<i}, \boldsymbol{y}_i, \boldsymbol{y}_{>i}, j, j') = \underbrace{P(\boldsymbol{y}_{>i}|j')}_{\beta_i(j')} \cdot \underbrace{P(\boldsymbol{y}_i, j'|j)}_{\gamma_i(j,j')} \cdot \underbrace{P(\boldsymbol{y}_{<i}, j)}_{\alpha_i(j)}. \tag{12.6}$$

Einsetzen in Gleichung (12.1) liefert

$$P(a_i = 1|\boldsymbol{y}) = \frac{1}{P(\boldsymbol{y})} \sum_{(j,j')\Rightarrow 1} \alpha_i(j) \cdot \gamma_i(j,j') \cdot \beta_i(j'). \tag{12.7}$$

Die Bestimmung und Eliminierung von $P(\boldsymbol{y})$ erfolgt, indem auf gleiche Weise $P(a_i = 0|\boldsymbol{y})$ bestimmt wird und $P(a_i = 1|\boldsymbol{y}) + P(a_i = 0|\boldsymbol{y}) = 1$ ausgenutzt wird.

Für $\gamma_i(j,j')$ ergibt sich

$$\gamma_i(j,j') = P(\boldsymbol{y}_i, j'|j) = P(\boldsymbol{y}_i|j', j) \cdot P(j'|j) = P\big(\boldsymbol{y}_i|\boldsymbol{x}(j,j')\big) \cdot P\big(a(j,j')\big), \tag{12.8}$$

wobei $\boldsymbol{x}(j,j')$ den beim Übergang von j zu j' ausgegeben Codewortteil, $a(j,j')$ das entsprechende Eingangsbit angibt.
$P\big(\boldsymbol{y}_i|\boldsymbol{x}(j,j')\big)$ ist durch den Kanal gegeben.
$P\big(a(j,j')\big)$ ist die A-Priori-Wahrscheinlichkeit des i-ten Nutzbits (ohne zusätzliche Information 1/2).

Für $\alpha_i(j)$ ergibt sich

$$\alpha_i(j) = P(\boldsymbol{y}_{<i}, j) = P(\boldsymbol{y}_{<i-1}, \boldsymbol{y}_{i-1}, j) = \sum_{\tilde{j}} P(\boldsymbol{y}_{<i-1}, \boldsymbol{y}_{i-1}, j, \tilde{j}), \tag{12.9}$$

wobei die Summe über die beiden möglichen Vorzustände $\tilde{j}$ von j läuft.
Mit der Bayes-Regel folgt

$$\alpha_i(j) = \sum_{\tilde{j}} P(\boldsymbol{y}_{<i-1}, \boldsymbol{y}_{i-1}, j, \tilde{j}) = \sum_{\tilde{j}} P(\boldsymbol{y}_{i-1}, j | \boldsymbol{y}_{<i-1}, \tilde{j}) \cdot P(\boldsymbol{y}_{<i-1}, \tilde{j}). \tag{12.10}$$

Ist der Vorzustand $\tilde{j}$ bekannt, so sind $\boldsymbol{y}_{i-1}$ und j von $\boldsymbol{y}_{<i-1}$ unabhängig, also

$$\alpha_i(j) = \sum_{\tilde{j}} P(\boldsymbol{y}_{i-1}, j | \boldsymbol{y}_{<i-1}, \tilde{j}) \cdot P(\boldsymbol{y}_{<i-1}, \tilde{j}) = \sum_{\tilde{j}} P(\boldsymbol{y}_{i-1}, j | \tilde{j}) \cdot P(\boldsymbol{y}_{<i-1}, \tilde{j}) \tag{12.11}$$

$$= \sum_{\tilde{j}} \gamma_{i-1}(\tilde{j}, j) \cdot \alpha_{i-1}(\tilde{j}). \tag{12.12}$$

Mit

$$\alpha_1(j) = \begin{cases} 1 & \text{für } j = 0 \\ 0 & \text{sonst} \end{cases} \tag{12.13}$$

lassen sich so sämtliche Werte für $\alpha_i(j)$ berechnen.

Für $\beta_i(j')$ ergibt sich auf ähnliche Weise

$$\beta_i(j') = P(\boldsymbol{y}_{>i} | j') = \frac{P(\boldsymbol{y}_{>i+1}, \boldsymbol{y}_{i+1}, j')}{P(j')} = \sum_{\tilde{j}} \frac{P(\boldsymbol{y}_{>i+1}, \boldsymbol{y}_{i+1}, \tilde{j}, j')}{P(j')}, \tag{12.14}$$

wobei die Summe über die beiden möglichen Folgezustände $\tilde{j}$ von j' läuft.
Mit der Bayes-Regel folgt

$$\beta_i(j') = \sum_{\tilde{j}} \frac{P(\boldsymbol{y}_{i+1}, j' | \boldsymbol{y}_{>i+1}, \tilde{j}) \cdot P(\boldsymbol{y}_{>i+1} | \tilde{j}) \cdot P(\tilde{j})}{P(j')} \tag{12.15}$$

Ist der Folgezustand $\tilde{j}$ bekannt, so sind $\boldsymbol{y}_{i+1}$ und j' von $\boldsymbol{y}_{>i+1}$ unabhängig, also

$$\beta_i(j') = \sum_{\tilde{j}} \frac{P(\boldsymbol{y}_{i+1}, j' | \tilde{j}) \cdot P(\boldsymbol{y}_{>i+1} | \tilde{j}) \cdot P(\tilde{j})}{P(j')} = \sum_{\tilde{j}} \frac{P(\boldsymbol{y}_{i+1}, j', \tilde{j}) \cdot P(\boldsymbol{y}_{>i+1} | \tilde{j})}{P(j')} \tag{12.16}$$

$$= \sum_{\tilde{j}} P(\boldsymbol{y}_{i+1}, \tilde{j} | j') \cdot P(\boldsymbol{y}_{>i+1} | \tilde{j}) = \sum_{\tilde{j}} \gamma_{i+1}(j', \tilde{j}) \cdot \beta_{i+1}(\tilde{j}). \tag{12.17}$$

Mit

$$\beta_n(j') = \begin{cases} 1 & \text{für } j' = 0 \\ 0 & \text{sonst} \end{cases} \tag{12.18}$$

lassen sich so sämtliche Werte für $\beta_i(j)$ berechnen.

Insgesamt ergibt sich damit folgender Decodieralgorithmus für die Komponentencodes:

1. Bestimme
$$\gamma_i(j,j') = P\big(\boldsymbol{y}_i|\boldsymbol{x}(j,j')\big) \cdot P\big(a(j,j')\big), \tag{12.19}$$
für $i = 1,\ldots,n$, $j = 0,\ldots,2^k-1$ und alle jeweils möglichen Folgezustände j'.
2. Bestimme
$$\alpha_i(j) = \sum_{\tilde{j}} \gamma_{i-1}(\tilde{j},j) \cdot \alpha_{i-1}(\tilde{j}). \tag{12.20}$$
für $i = 1,\ldots,n$, $j = 0,\ldots,2^k-1$ mit $\alpha_1(0) = 1$, $\alpha_1(j) = 0$ für $j \neq 0$.
3. Bestimme
$$\beta_i(j') = \sum_{\tilde{j}} \gamma_{i+1}(j',\tilde{j}) \cdot \beta_{i+1}(\tilde{j}). \tag{12.21}$$
für $i = n,\ldots,1$, $j = 0,\ldots,2^k-1$ mit $\beta_n(0) = 1$, $\beta_n(j) = 0$ für $j \neq 0$.
4. Bestimme
$$P(a_i = 1|\boldsymbol{y}) = \frac{1}{P(\boldsymbol{y})} \sum_{(j,j')\Rightarrow 1} \alpha_i(j) \cdot \gamma_i(j,j') \cdot \beta_i(j') \tag{12.22}$$
und analog $P(a_i = 0|\boldsymbol{y})$ um $P(\boldsymbol{y})$ zu eleminieren für $i = n,\ldots,1$.

12.4 Turbo-Decodierung

Für die Verwendung bei der Turbo-Decodierung ist der Term
$$\gamma_i(j,j') = P\big(\boldsymbol{y}_i|\boldsymbol{x}(j,j')\big) \cdot P\big(a(j,j')\big), \tag{12.23}$$
nochmals näher zu betrachten.
Dazu werden die Sende- und Empfangsdaten in den systematischen Anteil $x_{si} = a_i$ (y_{si}) und die vom ersten bzw. zweiten Faltungscoder hinzugefügten Bits x_{1i} (y_{1i}) bzw. x_{2i} (y_{2i}) aufgeteilt.
Damit folgt für den ersten Komponentendecoder
$$\gamma_i(j,j') = P\big(y_{si}|a(j,j')\big) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot P\big(a(j,j')\big) \tag{12.24}$$
und damit
$$P(a_i = 1|\boldsymbol{y}) = \frac{1}{P(\boldsymbol{y})} \sum_{(j,j')\Rightarrow 1} \alpha_i(j) \cdot P(y_{si}|a_i = 1) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot P(a_i = 1) \cdot \beta_i(j') \tag{12.25}$$
$$= \frac{1}{P(\boldsymbol{y})} P(y_{si}|a_i = 1) \cdot P(a_i = 1) \cdot \sum_{(j,j')\Rightarrow 1} \alpha_i(j) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot \beta_i(j'). \tag{12.26}$$

Das Ergebnis der Decodierung lässt sich also wie folgt aufteilen:

$$P(a_i = 1|\boldsymbol{y}) = \frac{1}{P(\boldsymbol{y})} P(y_{si}|a_i = 1) \cdot P(a_i = 1) \cdot \underbrace{\sum_{(j,j')\Rightarrow 1} \alpha_i(j) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot \beta_i(j')}_{}\,.$$

Ausgabe des Kanals bezüglich des i-ten systematischen Bits.

A-Priori-Wahrscheinlichkeit des i-ten Bits.

Die durch den ersten Komponenten-decoder hinzugewonnene Information über das i-te Bit.

Häufig findet man auch die Darstellung als Log-Likelihood-Ratio (LLR):

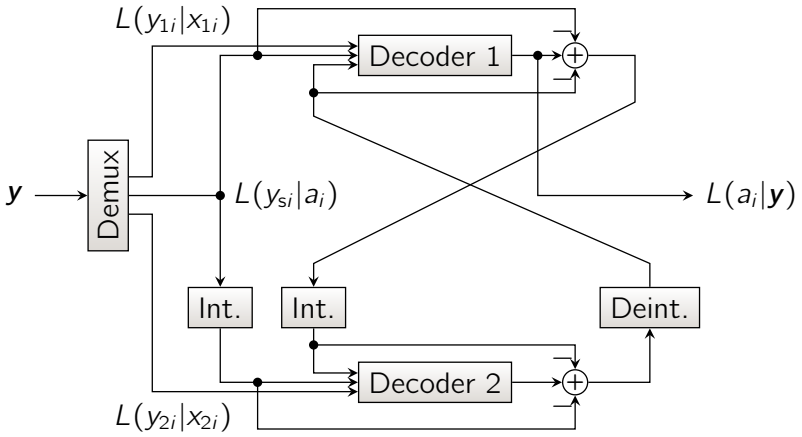
$$L(a_i|\boldsymbol{y}) = \log \frac{P(a_i = 1|\boldsymbol{y})}{P(a_i = 0|\boldsymbol{y})} \tag{12.27}$$

$$= \log \frac{P(y_{si}|a_i = 1) \cdot P(a_i = 1) \cdot \sum\limits_{(j,j')\Rightarrow 1} \alpha_i(j) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot \beta_i(j')}{P(y_{si}|a_i = 0) \cdot P(a_i = 0) \cdot \sum\limits_{(j,j')\Rightarrow 0} \alpha_i(j) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot \beta_i(j')} \tag{12.28}$$

$$= \log \frac{P(y_{si}|a_i = 1)}{P(y_{si}|a_i = 0)} + \log \frac{P(a_i = 1)}{P(a_i = 0)} + \log \frac{\sum\limits_{(j,j')\Rightarrow 1} \alpha_i(j) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot \beta_i(j')}{\sum\limits_{(j,j')\Rightarrow 0} \alpha_i(j) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot \beta_i(j')} \tag{12.29}$$

$$= L(y_{si}|a_i) + L(a_i) + \log \frac{\sum\limits_{(j,j')\Rightarrow 1} \alpha_i(j) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot \beta_i(j')}{\sum\limits_{(j,j')\Rightarrow 0} \alpha_i(j) \cdot P\big(y_{1i}|x_{1i}(j,j')\big) \cdot \beta_i(j')} \tag{12.30}$$

Bei der Turbo-Decodierung erhalten also beide Decodierer die LLRs der systematischen Bits, die LLRs der von jeweiligen Codierer eingefügten redundanten Bits sowie die A-Priori-LLRs der Nutzdaten.
Aus den LLRs an den Decodierer-Ausgängen lässt sich durch Subtraktion der LLRs der systematischen Bits und der A-Priori-LLRs die durch den jeweiligen Decodierer hinzugewonnen Information extrahieren.
Diese werden in einem zweiten Durchgang dem jeweils anderen Decodierer als A-Priori-LLRs eingespeist.
Dieser Vorgang wird iteriert, bis beide Decodierer das gleiche Ergebnis liefern, das dann als Endegebnis verwendet wird.



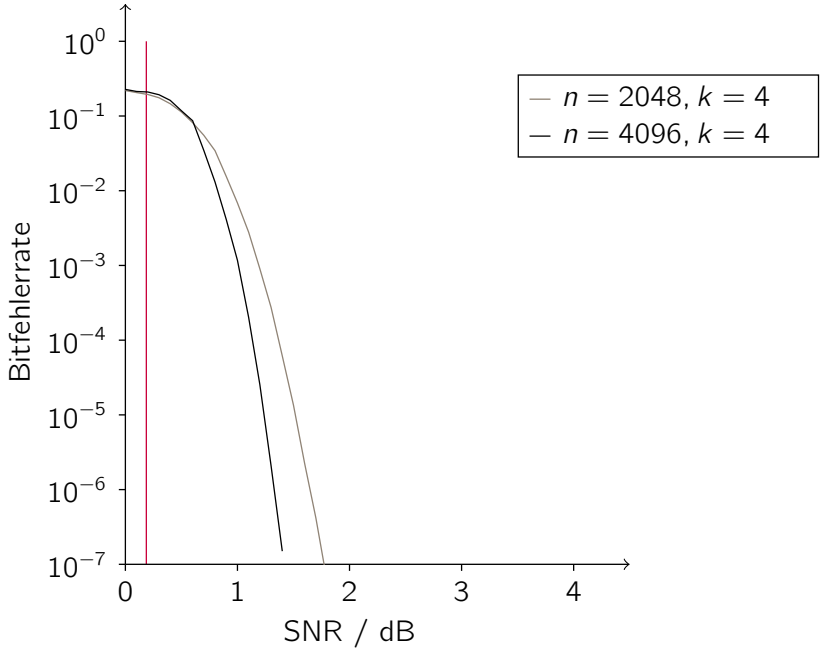
Typischerweise werden die Komponentendecodierer abwechselnd gerechnet, wobei jeweils die Ausgabe des anderen aus dem vorherigen Schritt mitberücksichtigt wird.
Insgesamt lautet der Algorithmus zur Turbo-Decodierung also:

1. Bestimme $L(a_i|\boldsymbol{y}_s, \boldsymbol{y}_1)$ mit dem ersten Komponentendecodierer, wobei $L(a_i) = 0$ angenommen wird.
2. Bestimme $L(a_i|\boldsymbol{y}_s, \boldsymbol{y}_2)$ mit dem zweiten Komponentendecodierer, wobei für $L(a_i)$ die vom ersten Komponentendecodierer zusätzlich gewonnene Information genutzt wird.
3. Bestimme $L(a_i|\boldsymbol{y}_s, \boldsymbol{y}_1)$ mit dem ersten Komponentendecodierer, wobei für $L(a_i)$ die vom zweiten Komponentendecodierer zusätzlich gewonnene Information genutzt wird.
4. Weiter bei 2, bis Einigkeit unter beiden Komponentendecodierern erreicht oder maximale Iterationszahl erreicht.

Die iterative Decodierung ähnelt in einem gewissen Maß derjenigen der LDPC-Codes; tatsächlich lässt sich auch die Turbo-Deocdierung als Belief-Propagation interpretieren.

Beispiel 12.1 (Turbo-Code für einen AWGN-Kanal)

Turbo-Code aus zwei halbratigen Faltungscodern, punktiert auf die Gesamtcoderate $R = 1/2$.



12.5 Zusammenfassung

- Turbo-Codes verwenden parallele Faltungscodes; durch Punktierung wird die gewünschte Code-Rate erreicht.
- Durch Interleaving wird erreicht, dass Übertragungsfehler die Komponentencodes an unterschiedlichen Stellen stören.
- Die Decoder der Komponentencodes liefern nicht nur das wahrscheinlichste Datenwort $\hat{\boldsymbol{a}}$, sondern die Wahrscheinlichkeiten $p(a_i|\boldsymbol{y})$ unter Berücksichtigung des jeweiligen Komponentencodes.
- Aus den Eingaben der Komponentendecoder (Wahrscheinlichkeiten am Kanalausgang und A-Priori-Wahrscheinlichkeiten der Nutzdaten) und der Ausgabe lässt sich leicht die durch die Decodierung hinzugewonnene Erkenntnis bestimmen.
- In einem iterativen Prozess lässt sich diese hinzugewonnene Erkenntnis dem jeweils anderen Komponentendecoder als A-Priori-Wahrscheinlichkeit zuführen, sodass sich die Komponentendecoder gegenseitig helfen.
- Die Turbo-Codes ermöglichen so eine Codierung nahe an der Shannon-Grenze.