Iterative Entzerrung

Michael Meyer, 2808666, Betreuer: Prof. Dr.–Ing. Joachim Hagenauer

***Abstract*— This paper is a summary of a talk given during the „Joint Advanced Students School 2005“ in St. Petersburg. Iterative Equalization or Turbo-Equalization is an iterative technique which concatenates the processes of equalization and decoding according to the Turbo-Principle[1]. In conventional receivers, these two processes are separated and only pass hard decisions instead of soft decisions. Iterative Equaliziation uses this soft information in a kind of feedback loop. Iterative Equalization can impressively reduce the bit error rate of communication systems which pass information through a channel with intersignal interference. In the following, Iterative Equalization will be explained using a concrete example.**

I. Einleitung

Wie in Abbildung 1 gezeigt, laufen in der digitalen Nachrichtenübertragung üblicherweise die folgenden Vorgänge ab. Im *Encoder* findet die Quellenkodierung der binären Datensequenz a_k in eine Sequenz b_k, die nicht nur die Daten, sondern auch redundante Informationen enthält, statt. Um Fehler zu vermeiden, die durch kurzfristige Störungen des Kanals verursacht werden, wird die Bitreihenfolge mit Hilfe eines *Interleavers* verändert. Dessen Aufgabe ist es vor allem, benachbarte Bits zu trennen. Die resultierende Sequenz c_k wird im *Mapper* den Kanalsymbolen x_k zugeordnet. Während der Übertragung der Symbole x_k über den Kanal werden diese u.a. durch additives Rauschen und *Intersignalinterferenz* (ISI) verändert. Die empfangenen Symbole y_k werden

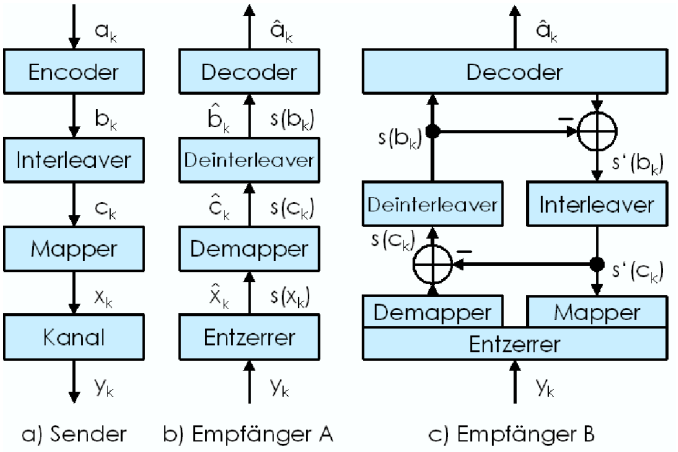

Fig. 1. Sender und 2 Empfänger-Strukturen. Empfänger A gibt binäre Entscheidungen oder Wahrscheinlichkeiten $s(\ldots)$ weiter, Empfänger B benutzt iterative Entzerrung.

im Empfänger zu einem Schätzwert $\hat{a}_k$ der gesendeten Datensequenz a_k verarbeitet. Üblicherweise geschieht dies, indem die zu Mapping, Interleaving und Quellenkodierung inversen Vorgänge der Reihe nach ausgeführt werden. Eine Entzerrung ist insbesondere bei ISI-behafteten Signalen nötig.

Während dieses Prozesses wird Information zerstört, da fortlaufend harte Entscheidungen getroffen werden, ohne die Wahrscheinlichkeit dieser Entscheidungen zu berücksichtigen. Diese zusätzliche *soft information* $s(\ldots)$ kann in Form von *log-likelyhood-ratios* (LLR), definiert als

$$L(a) = \ln \frac{P(a=0)}{P(a=1)}, \tag{1}$$

Schritt für Schritt weitergegeben werden. Diese *soft information* kann jedoch auch in Gegenrichtung fliessen. Der *Decoder*, der $s(b_k)$ verarbeitet und die Fehlererkennung und -korrektur durchführt, kann seine eigene *soft information* generieren, die eine Aussage über das *likelihood* jedes der übertragenen Bits zulässt. Wenn diese Information analog zum Sender verwürfelt und gemappt wird - und nach dem Turbo-Prinzip die extrinsische von der intrinsischen Information getrennt wird - kann eine Rückkopplungsschleife zwischen *Decoder* und *Entzerrer* aufgebaut werden.

Dieser Prozess wird auch *belief propagation* [2] genannt und ist verwandt mit Methoden, die in der künstlichen Intelligenz verwendet werden. Die eben beschriebene Rückkopplungsschleife ist typisch für die iterative Entzerrung, die nun anhand eines konkreten Beispiels aus [2] näher erläutert werden soll. Iterative Entzerrung wurde erstmals 1995 von Douillard et al. [3] als *Turbo Equalization* vorgeschlagen.

II. Verwendetes Kanalmodell

Im folgenden wird BPSK-Modulation und ein Kanalmodell

$$\tilde{\mathbf{H}} = \begin{bmatrix} h_2 & h_1 & h_0 & 0 & 0 & \ldots & 0 \\ 0 & h_2 & h_1 & h_0 & 0 & \ldots & 0 \\ & & & \ddots & \ddots & \ddots & \\ 0 & 0 & \ldots & 0 & h_2 & h_1 & h_0 \end{bmatrix} \tag{2}$$

der Länge L = 3 in einem AWGN-Kanal verwendet. Es gilt

$$\mathbf{y_k} = \tilde{\mathbf{H}}\mathbf{x_k} + \mathbf{n_k}. \tag{3}$$

Dieses Modell ist in Abb. 2 als FIR-Filter der Länge 3 dargestellt. Mit den Koeffizienten $h_0 = 0.407, h_1 = 0.815, h_2 = 0.407$ ergibt sich das Trellis-Diagramm in Abb. 3. Ein Ast eines Trellis ist ein 4-Tupel $(i, j, x_{i,j}, \nu_{i,j})$. Der Zustand $s_k = r_i$ zum Zeitpunkt k wird hierbei durch ein Eingangssignal $x_k = x_{i,j}$ und ein Ausgangssignal $\nu_k = \nu_{i,j}$ zum Zustand $s_{k+1} = r_j$ zum Zeitpunkt (k+1). ν_k ist das empfangene Symbol im störungsfreien Fall.

$$\nu_k = \sum_{l=0}^{L} h_l x_{k-l} \tag{4}$$

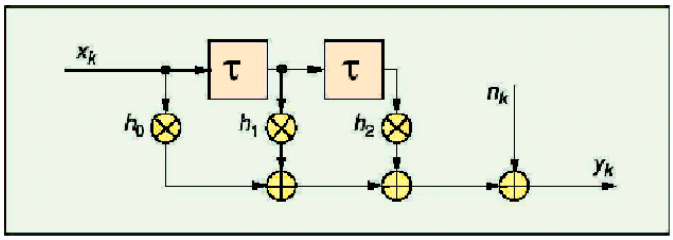

Fig. 2. Das verwendete Kanalmodell als FIR-Filter

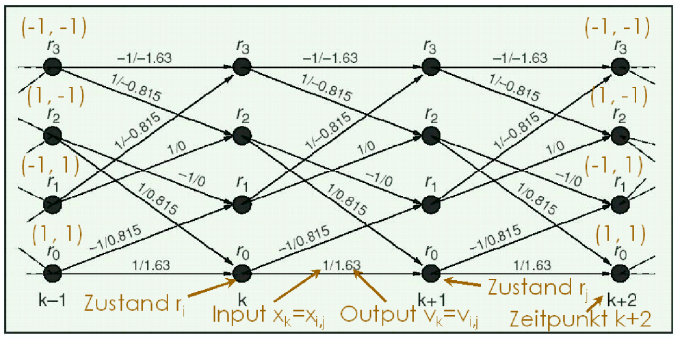

Fig. 3. Trellis-Diagramm zum verwendeten Kanalmodell

Die Knoten $r_0 = (1,1), r_1 = (-1,1), r_2 = (1,-1)$ und $r_3 = (-1,-1)$ beschreiben hierbei die möglichen Zustände der beiden Verzögerungselemente in Abb. 2.

III. Entzerrung und Demapping

Die beiden Schritte Entzerrung und Demapping aus Abb. 1 werden im folgenden zu einem Modul zusammengefasst. Da *soft information* weitergegeben wird, ist es sinnvoll, ein LLR $L(c_k|\mathbf{y})$ zu bilden. Dieses besteht aus extrinsischer Information $L_{ext}(c_k|\mathbf{y})$, die in der Kanalinformation $\mathbf{y}$ enthalten ist, und intrinsischer a priori Information $L(c_k)$. Dies wird später für das Turbo-Prinzip [1] wichtig. Nach dem Satz von Bayes gilt

$$P(c_k = c|\mathbf{y}) = \sum_{\forall \mathbf{c}: c_k = c} P(\mathbf{c}|\mathbf{y}) = \sum_{\forall \mathbf{c}: c_k = c} \frac{p(\mathbf{y}|\mathbf{c})P(\mathbf{c})}{p(\mathbf{y})} \quad (5)$$

und schließlich

$$\begin{aligned} L(c_k|\mathbf{y}) &= \ln \frac{\sum_{\forall \mathbf{c}: c_k=0} p(\mathbf{y}|\mathbf{c}) \prod_{i=1}^{K} P(c_i)}{\sum_{\forall \mathbf{c}: c_k=1} p(\mathbf{y}|\mathbf{c}) \prod_{i=1}^{K} P(c_i)} \\ &= \ln \frac{\sum_{\forall \mathbf{c}: c_k=0} p(\mathbf{y}|\mathbf{c}) \prod_{i=1: i\neq k}^{K} P(c_i)}{\sum_{\forall \mathbf{c}: c_k=1} p(\mathbf{y}|\mathbf{c}) \prod_{i=1: i\neq k}^{K} P(c_i)} + L(c_k) \\ &= L_{ext}(c_k|\mathbf{y}) + L(c_k). \end{aligned} \quad (6)$$

Das extrinisische Wissen wird vom Übertragungskanal bezogen, das a priori Wissen ist zusätzlich vorhanden. Wir werden es später vom Decoder und damit aus der Redundanz des verwendeten Codes erhalten.
Für die Übergangswahrscheinlichkeit $\gamma_k(s_k, s_{k+1})$, mit der ein Übergang vom Zustand s_k nach s_{k+1} stattfindet, gilt

$$\gamma_k(s_k, s_{k+1}) = P(s_{k+1}|s_k) \cdot p(y_k|s_k, s_{k+1}). \quad (7)$$

Auf das Trellis-Diagramm in Abb. 3 angewandt ergibt sich

$$\gamma_k(r_i, r_j) = \begin{cases} P(x_k = x_{i,j}) \cdot p(y_k|\nu_k = \nu_{i,j}) & (i\ j) \in \mathcal{B} \\ 0 & (i\ j) \notin \mathcal{B} \end{cases} \quad (8)$$

, wobei $\mathcal{B}$ die Menge aller erlaubten Übergänge von r_i nach r_j ist. $P(x_k = x_{i,j}$ stellt hierbei die a priori Information dar. Aus dem Kanalgesetz (3) und der Gauß-Verteilung des Rauschens wissen wir, dass

$$p(y_k|\nu_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(y_k - \nu_k)^2}{2\sigma^2}}. \quad (9)$$

Wenn man Schritt für Schritt durch das Trellis-Diagramm läuft und die Übergangswahrscheinlichkeiten jeweils multipliziert, erhält man den Term $\alpha_k(s)$, der die Wahrscheinlichkeit beschreibt, dass man auf genau diesen Weg durch die Trellis gelaufen ist. Dieser Term wird durch die Rekursionsformel

$$\alpha_k(s) = \sum \alpha_{k-1}(s') \cdot \gamma_{k-1}(s', s) \quad (10)$$

gebildet, wobei der Startwert $\alpha_0(s) = P(s_0 = s)$ die Wahrscheinlichkeit des Zustands s_0 zum Zeitpunkt $k = 0$ beschreibt. Wenn man statt in Vorwärtsrichtung (*forward*) rückwärts (*backward*) durch die Trellis läuft, erhält man die Rekursionsformel

$$\beta_k(s) = \sum \beta_{k+1}(s') \cdot \gamma_{k-1}(s, s') \quad (11)$$

mit dem Startwert $\beta_N(s) = 1$. Für den gesamten Pfad, also vom Zustand s_0 bis zu s_N, durch die Trellis gilt

$$p(s_k, s_k + 1, \mathbf{y}) = \alpha_k(s_k) \cdot \gamma_k(s_k, s_{k+1}) \cdot \beta_{k+1}(s_{k+1}). \quad (12)$$

Damit lässt sich schließlich das gewünschte LLR zu

$$L(c_k|\mathbf{y}) = \ln \frac{\sum_{\forall (i\ j) \in \mathcal{B}: x_{i,j}=+1} \alpha_k(r_i) \cdot \gamma_k(r_i, r_j) \cdot \beta_{k+1}(r_j)}{\sum_{\forall (i\ j) \in \mathcal{B}: x_{i,j}=-1} \alpha_k(r_i) \cdot \gamma_k(r_i, r_j) \cdot \beta_{k+1}(r_j)} \quad (13)$$

berechnen. Ein effizienter Weg, dieses LLR zu berechnen, ist die Verwendung des in Abb. 4 skizzierten *forward-backward-Algorithmus* (FBA) [4], dem die Matrizen $\mathbf{P}_k$ und $\mathbf{B}_k$ übergeben werden. Die Elemente von $\mathbf{P}_k$ ergeben sich aus den Übergangswahrscheinlichkeiten

$$\{\mathbf{P}_k\}_{i,j} = \gamma_k(r_i, r_j), \quad (14)$$

$\mathbf{B}_k$ berücksichtigt die möglichen Wege durch die Trellis:

$$\{\mathbf{A}(x)\}_{i,j} = \begin{cases} 1, & (i\ j)\ ist\ ein\ Zweig\ mit\ x_{i,j} = x \\ 0, & sonst \end{cases} \quad (15)$$

$$\{\mathbf{B}_k(x)\}_{i,j} = \{\mathbf{A}(x)\}_{i,j} \cdot \{\mathbf{P}_k\}_{i,j} \quad (16)$$

Für das Trellis-Diagramm in Abb. 3 ergibt sich

$$\mathbf{A}(+1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{A}(-1) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

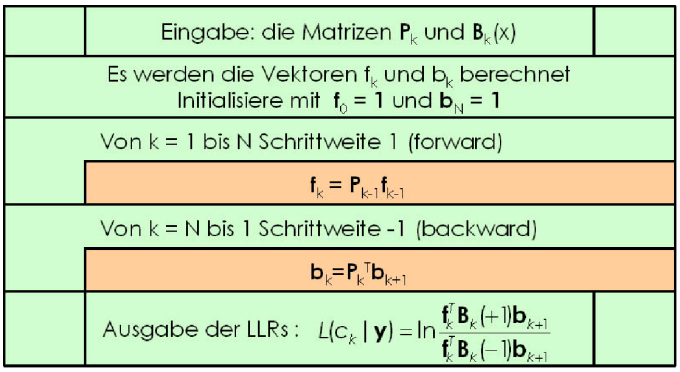

Fig. 4. Forward-Backward-Algorithmus für den Entzerrer

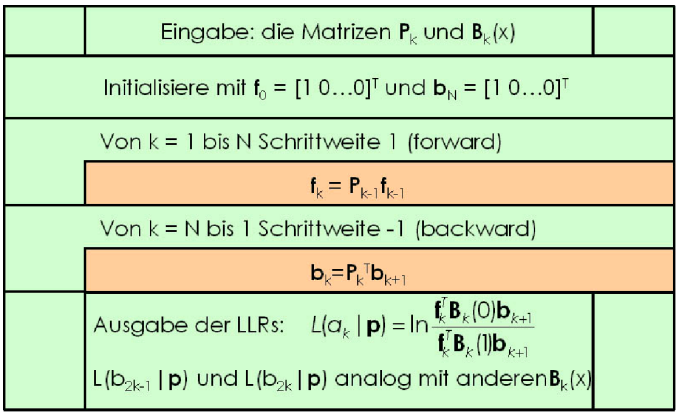

Fig. 7. Forward-Backward-Algorithmus für den Decoder

In diesem Abschnitt wurde ein MAP-Entzerrer behandelt. Für ZF- und MMSE-Entzerrer sei auf [2] verwiesen.

IV. Decodierung

Nach dem *Deinterleaving* muss die Information dekodiert werden. Im Beispiel ist der Code ein binärer Faltungscode. Der verwendete Faltungscode ist in Abb. 6 skizziert. Aus diesem Code ergibt sich das Trellisdiagramm in Abb.**??**.

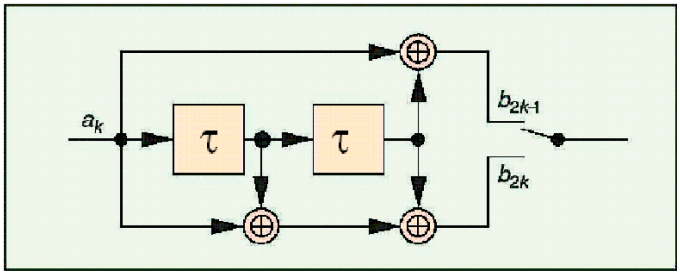

Fig. 5. Encoder eines Faltungscodes, bei dem jedes Eingangsbit a_k zwei Code-Bits $b_{2k-1} = a_k \oplus a_{k-2}$ und $b_{2k} = a_k \oplus a_{k-1} \oplus a_{k-2}$ erzeugt.

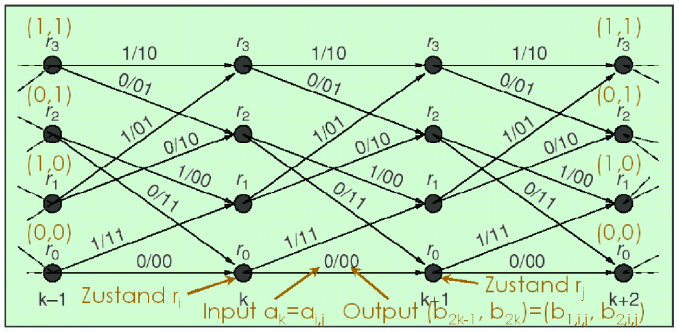

Fig. 6. Trellis-Diagramm zum verwendeten Faltungscode

Die Ähnlichkeit mit dem verwendeten Kanalmodell, Abb. 2, ist leicht zu erkennen. Wir können einen MAP-Decoder verwenden und erneut den *forward-backward-Algorithmus* mit $\gamma_k(r_i, r_j) =$

$$\begin{cases} P(a_k = a_{i,j}) & \cdot P(b_{2k} = b_{2,i,j} | \mathbf{y}) \cdot \\ & \cdot P(b_{2k-1} = b_{1,i,j} | \mathbf{y}), \quad wenn (i\ j) \in \mathcal{B} \\ 0, & wenn (i\ j) \notin \mathcal{B} \end{cases}, \tag{17}$$

$$\{\mathbf{A}(x)\}_{i,j} = \begin{cases} 1, & (i\ j)\ ist\ ein\ Zweig\ mit\ a_{i,j} = x \\ 0, & sonst \end{cases} \tag{18}$$

und den Gleichungen (14), (16) benutzen. Der konkrete Algorithmus ist in Abb. 7 skizziert.

Hierbei ist

$$\mathbf{p} = [P(b_1|\mathbf{y}) \quad P(b_2|\mathbf{y}) \quad \ldots \quad P(b_N|\mathbf{y})]^T \tag{19}$$

die Menge der Wahrscheinlichkeiten, die dem Decodier-Algorithmus übergeben werden. Die Wahrscheinlichkeiten $P(b_k|\mathbf{y})$ werden durch *deinterleaving* aus $P(c_k|\mathbf{y})$ gewonnen, die wiederum über

$$P(c_k = c|\mathbf{y}) = \frac{e^{-c \cdot L(c_k|\mathbf{y})}}{1 + e^{-L(c_k|\mathbf{y})}} \tag{20}$$

aus den LLRs gewonnen werden. Der *forward-backward-Algorithmus* gibt das LLR $L(a_k|\mathbf{p})$ aus. Mit Hilfe der einfachen Entscheidungsregel

$$\hat{a}_k = \begin{cases} 0, & L(a_k|\mathbf{p}) \geq 0 \\ 1, & L(a_k|\mathbf{p}) < 0 \end{cases} \tag{21}$$

kann daraus ein Schätzwert $\hat{a}_k$ für die gesendeten Datenbits a_k gebildet werden.

V. Das Turbo-Prinzip

Soweit entspricht die Vorgehensweise dem Empfänger A aus Abb. 1. Mit dem FBA können aber auch LLRs für $L(b_k|\mathbf{p})$ berechnet werden. Dazu ist nur die Matrix $\mathbf{A}(x)$ zu ändern. Um $L(b_{2k-1}|\mathbf{p})$ zu berechnen, wird

$$\{\mathbf{A}(x)\}_{i,j} = \begin{cases} 1, & (i\ j)\ ist\ ein\ Zweig\ mit\ b_{1,i,j} = x \\ 0, & sonst \end{cases} \tag{22}$$

gewählt, für $L(b_{2k}|\mathbf{p})$ wird die Matrix

$$\{\mathbf{A}(x)\}_{i,j} = \begin{cases} 1, & (i\ j)\ ist\ ein\ Zweig\ mit\ b_{2,i,j} = x \\ 0, & sonst \end{cases} \tag{23}$$

gewählt. Wenn mit Hilfe eines Interleavers die $L(b_k|\mathbf{p})$ wieder in in $L(c_k|\mathbf{p})$ gewandelt werden, kann der MAP-Entzerrer dies als a priori Information verwenden.

Die dahinter liegende Idee der Rückkopplung ist einfach und einleuchtend, es ist jedoch wichtig, dass diese Rückkopplung nicht zu stark wird, dass die Wahrscheinlichkeit eines Symbols, für die sich der Entzerrer einmal entschieden hat, nicht automatisch verstärkt wird. Dieses sogenannte *direct feedback* konvergiert zwar sehr schnell, in der Regel jedoch nicht gegen die optimale Lösung. Deshalb darf nur die intrinsische Information, nicht aber die extrinsische, zurückgeführt werden, wie es auch in Abb. 8 dargestellt ist.

In diesem Zusammenhang spielen die Interleaver ebenfalls

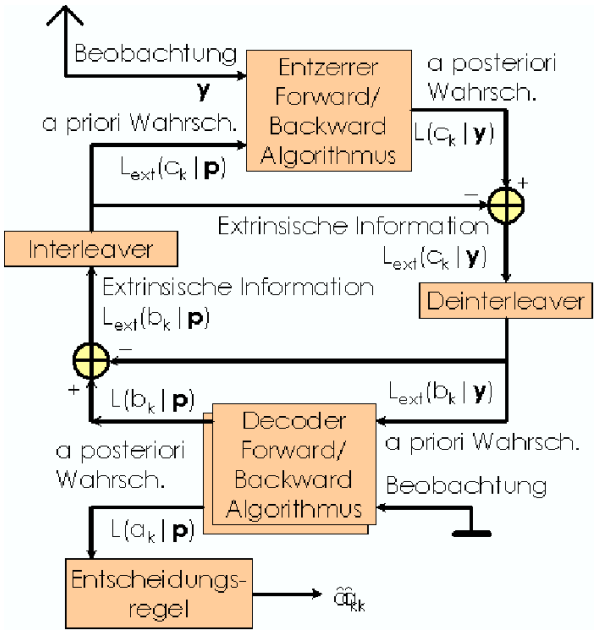

Fig. 8. Schaubild zur iterativen Entzerrung.

eine wichtige Rolle. Die durch den FBA gewonnen LLRs sind lokal stark korreliert. Diese Korrelationen zwischen benachbarten Symbolen werden durch das Interleaving Stark unterdrückt. Im Beispiel wurde ein *16-random-interleaver* verwendet, der benachbarte Bits mindestens durch einen Abstand von 16 Codebits trennt.

Der mit Hilfe des Turbo-Prinzips erzielte Performance-Gewinn des hier beschriebenen Empfängers ist in Abb. 9 dargestellt. Bereits nach einmaligem Durchlaufen der Rückkopplungsschleife reduziert sich die Bitfehlerrate signifikant. Der verwendete Algorithmus ist in Abb. 10 zusammengefasst.

VI. Zusammenfassung und Ausblick

In dieser Ausarbeitung wurde das Prinzip der iterativen Entzerrung aufgezeigt und anhand eines konkreten Beispiels erklärt. Damit iterative Entzerrung nach dem Turbo-Prinzip funktionieren kann, ist es nötig, zwischen extrinsischer und intrinsischer Information zu unterscheiden und nur die extrinsische Information rückzukoppeln. Der deutliche Performance-

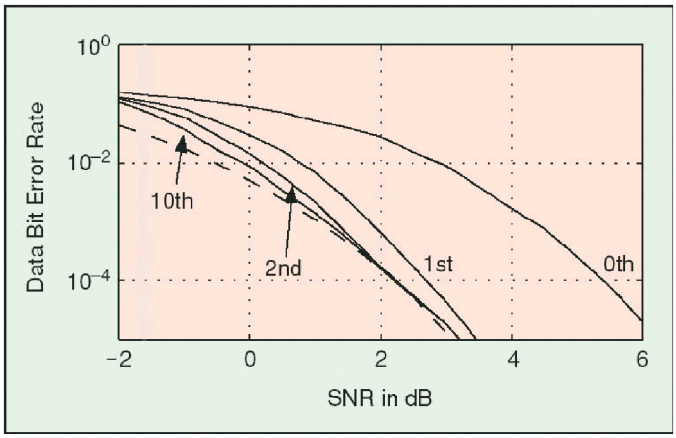

Fig. 9. Performance des hier beschriebenen Empfängers nach 0, 1, 2 und 10 Iterationen. Es wurden K = 512 Datenbits übertragen.

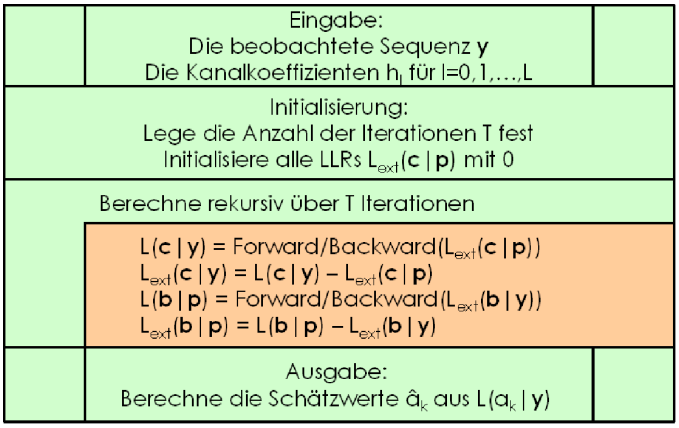

Fig. 10. Zusammenfassung des verwendeten Algorithmus'.

Gewinn der iterativen Entzerrung auch bei wenigen Iterationen gegenüber konventioneller Entzerrung wurde dargestellt.

Für die Implementierung von iterativer Entzerrung ist es nicht zwingend nötig, wie hier beschrieben, einen MAP-Entzerrer und einen MAP-Decoder zu verwenden. Grundsätzlich ist jeder Entzerrer und Decoder geeignet, der *soft information*, zum Beispiel in Form von *log likelihood ratios* verarbeitet und generiert.

Interessant ist es ferner, iterative Entzerrung im Zusammenhang mit Turbo-Codes oder anderen iterativen Dekodieralgorithmen zu verwenden. In diesem Fall liegen zwei Rückkopplungsschleifen für *soft information* vor, die auf die eine oder andere Weise miteinander zu verknüpfen sind.

References

[1] J. Hagenauer: The Turbo Principle in Communications. - In *Joint Advanced Student School (JASS 2005)*, St. Petersburg, Russia, März 2005, www.lnt.e-technik.tu-muenchen.de

[2] R. Koetter, A. Singer, M. Tüchler: Turbo Equalization. - In *IEEE Signal Processing Magazine*, pp. 67-80, Vol. 21, Nr. 1, January 2004

[3] C. Douillard et al.: Iterative correction of intersymbol interference: Turbo equalization - In *European Trans. Telecomm.*, pp. 507-511, Vol. 6, Sept.-Okt. 1995

[4] L. Rabiner: A tutorial on hidden Markov models and selected applications in speech recognition - In *Proc. IEEE*, pp. 257-286, Vol. 77, Feb. 1989