Chapter 7 Turbo Codes

References

1. Berrou ,C. , Glavieux ,A., and Thitimajshima ,P. , ” Near Shannon-Limit Error Correcting Coding and Decoding : Turbo Codes ,” Proc. 1993 IEEE Int. Conf. on Communications , pp.1064-1070 , Geneva , Switzerland , May 1993.

2 .Berrou ,C. and A. Glavieux ,” Near Optimum Error Correcting coding and decoding : Turbo Codes ,” IEEE Trans. Commun. Vol.44, No.10, pp.1261-1271, Oct.1996.

3. Bahl, L.R. , Cocke,J., Jelinek , F.,and Raviv, J. ,” Optimal Decoding of Linear Codes for Minimizing Symbol Error Rates ,” IEEE Trans. Inform. Theory , Vol. IT-20 Pp. 284-287 , Mar. 1974.

4. Hanzo , L., Woodward, J.P. and Robertson, P.l , “ Turbo Decoding and Detection for Wireless Applications “ , Proc. IEEE , Vol. No.95 ,No.6 , pp.1178-1200, June 2007.

5. Sivio A. Abrantes,” From BCJR to turbo decoding : MAP algorithms made easier “ Universidade de Porto , Porto , Portugal ,April 2004.

6.Robertson, P., Villebrun,E. and Hoeher,P., “ A comparison of optimal and suboptimal MAP decoding algorithm operating in the log domain, “ 1995 IEEE Int. Conf. On Communications, pp.1009-1013.

7. ten Brink , S., “Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes ,: IEEE Trans. Commun. , vol.49, no.10, pp. 1727-1737, Oct.2001

8. Lin,S .& Costello , Jr. , D.J. , Error Control Coding , Prentice-Hall, 2004

7.1 Turbo Codes

7.1.1 Shannon Limit on Performance

- **The capacity of an AWGN channel is given by**

  $C = W \log_2 (1 + S / N)$

  where W is the bandwidth of the channel in Hz , S is the average power of the signal and N is the noise power.

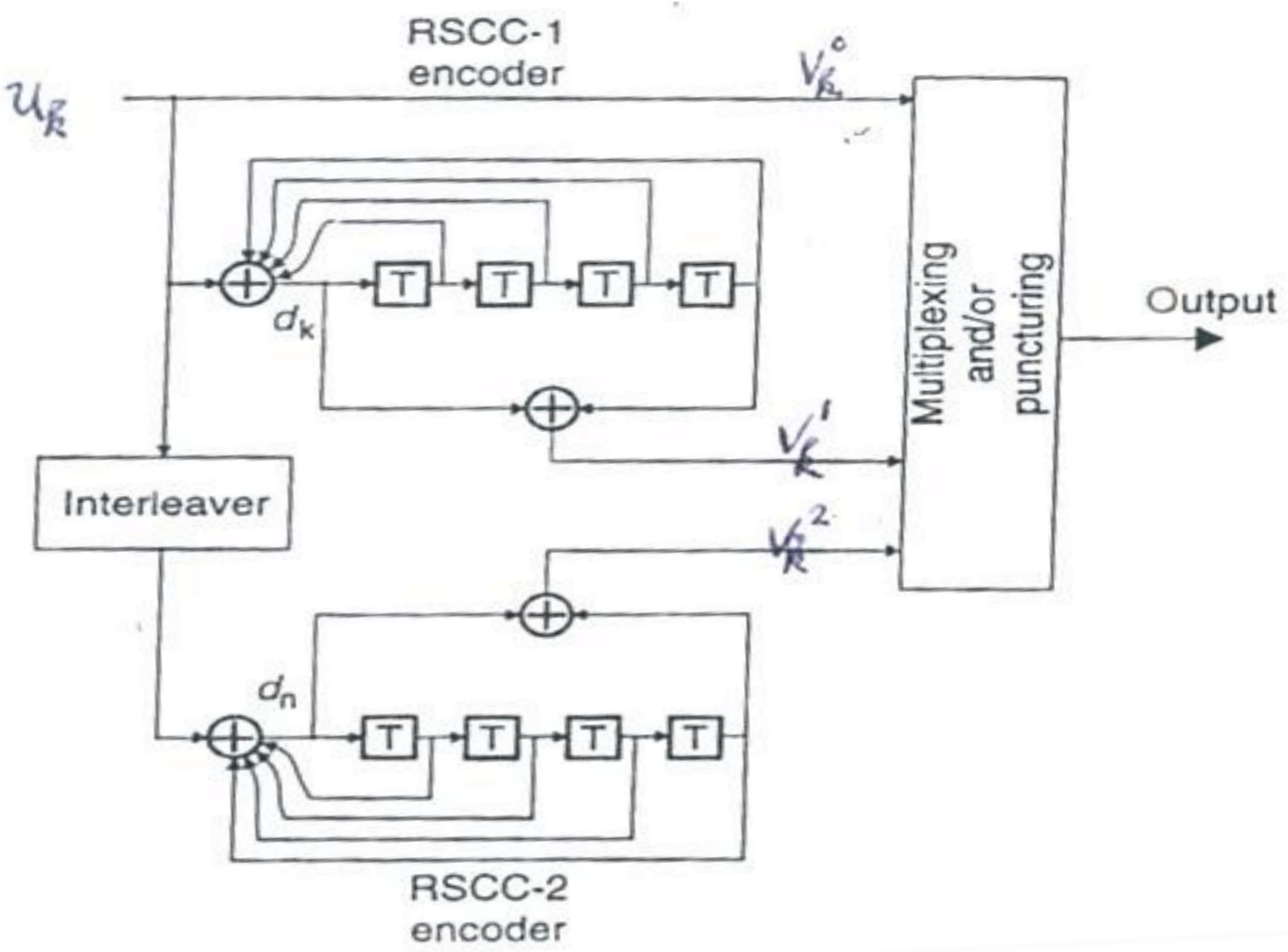
- **Shannon bound**
- **In the limit as the signal is allowed to occupy an infinity amount of bandwidth, one obtains the Shannon bound**

  $E_b / N_0 = -1.59$ **dB**

  for reliable transmission.

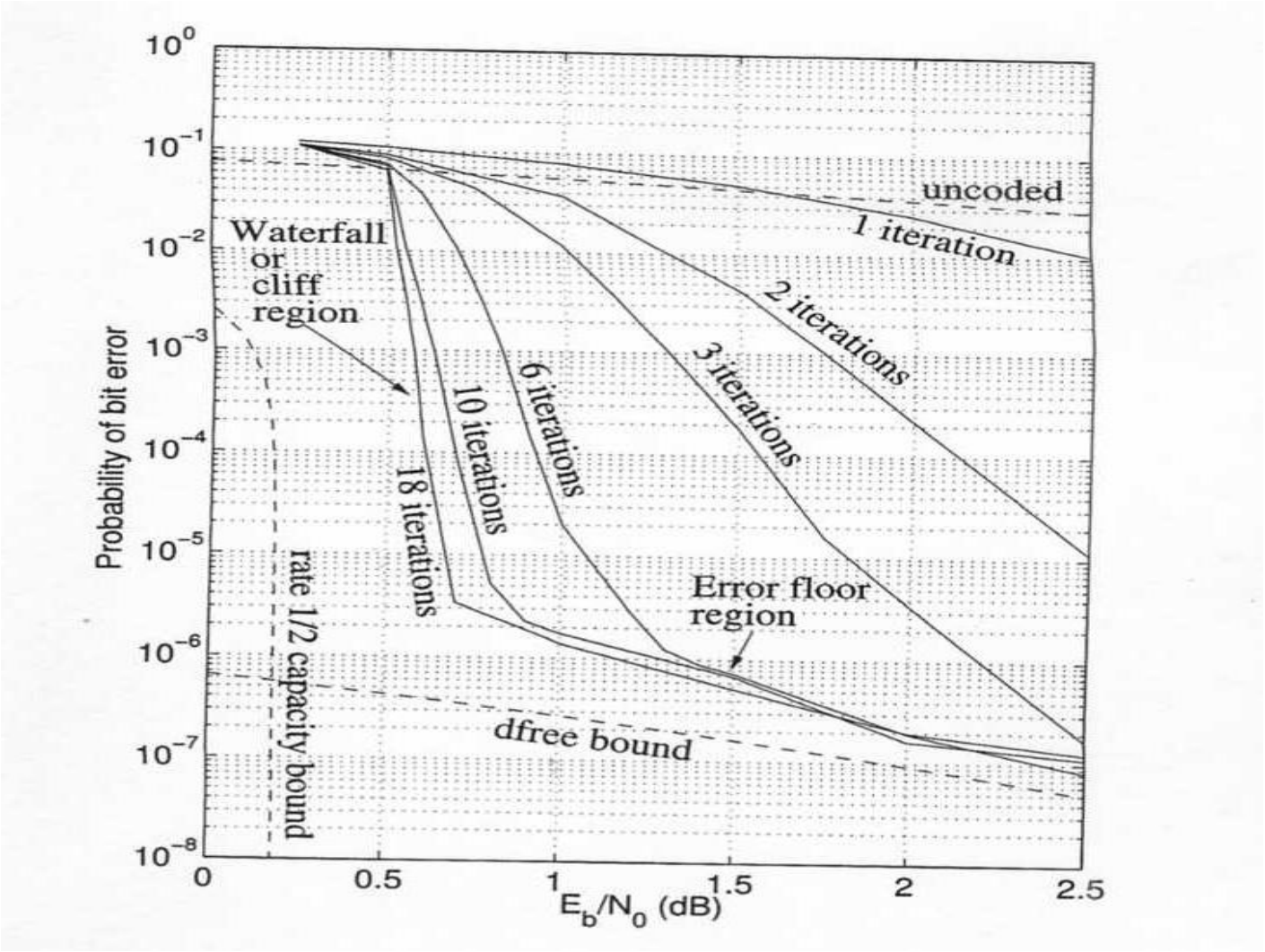
- **In 1948, Shannon defined the concept of channel capacity. He shown that, as long as the rate at which information is transmitted is less than the channel capacity, there exists error control codes that can provide arbitrarily high levels of reliability at the receiver output**
- **The 1993 paper by C. Berrou, A. Glavieux, and P. Thitimasjshima entitled ‘’Near Shannon Limit Error Correcting Coding and Decoding : Turbo Codes’’ has brought us within a hair’s breadth of achieving Shannon’s promise. This paper was presented at 1993 IEEE International Conference on Communications in Geneva.**
- **In their presentation, Berrou et al claimed that a combination of parallel concatenation and iterative decoding can provide reliable communication at a SNR that is within a few tenth of a dB of the Shannon limit.**

Turbo code encoder (Berrou *et al* , 1993)

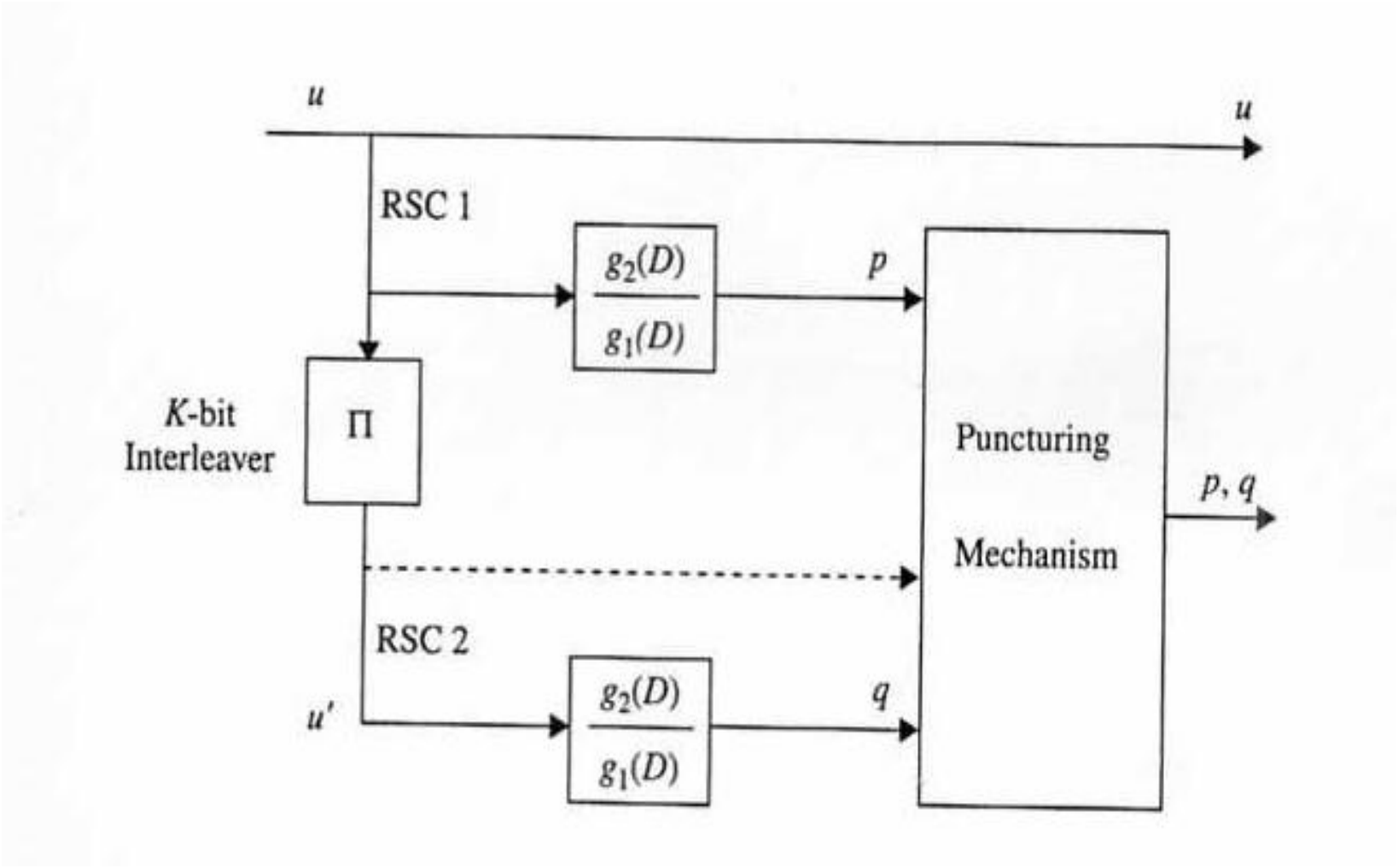


Decoding result of (37,21 ,65536) turbo code

Note : Octal numbers 37 =11 111 , 21 = 10001 .Block length N =65536



Standard structure for turbo code encoder



- **One key to the effectiveness of turbo coding systems is the interleaver.**
- **The interleaver allows the constituent decoders to generate separate estimates of the *a posteriori probability* (APP) for a given information symbol based on data sources that are not highly corrected.**
- **The pseudorandom interleaver in this circuit is used to permute the input bits in such a manner that the two encoders operate on the same set of input bits, but with different input sequence to the encoders.**

7.1.3 Decoding of Turbo Codes

- **The iterative decoder uses a soft-in/soft-out maximum *a posteriori* probability (MAP) decoding algorithm.**
- **This algorithm was first applied to convolutional codes by BCJR algorithm , introduced by Bahl, Cocke, Jelinek, and Raviv in 1974.**
- **The BCJR algorithm differs from the Viterbi algorithm in the sense that it produces soft outputs . The soft output weights the confidence or log-likelihood of each bit estimate.**
- **The BCJR algorithm attempts to minimize the bit-error-rate by estimating the a posteriori probability (APP) of the individual bits of the codeword, rather than the ML estimate of the transmitted codeword .**

- **For encoder given in Fig.8.4, there are two elementary decoders that are interconnected as shown in Fig. 8.xx**
- **The BCJR algorithm is employed in each decoder.**

The data ($r^{(0)}$, $r^{(1)}$) associated with the first encoder are fed to Decoder 1. This decoder initially uses uniform priors on the transmitted bits and produces probabilities of the bits conditioned on the observed data. These probabilities are called the extrinsic probabilities. The output probabilities of Decoder 1 are interleaved and passed to Decoder 2 , where they are used as " prior " probabilities in the decoder , along with the data associated the second encoder , which is $r^{(0)}$ (interleaved) and $r^{(2)}$.

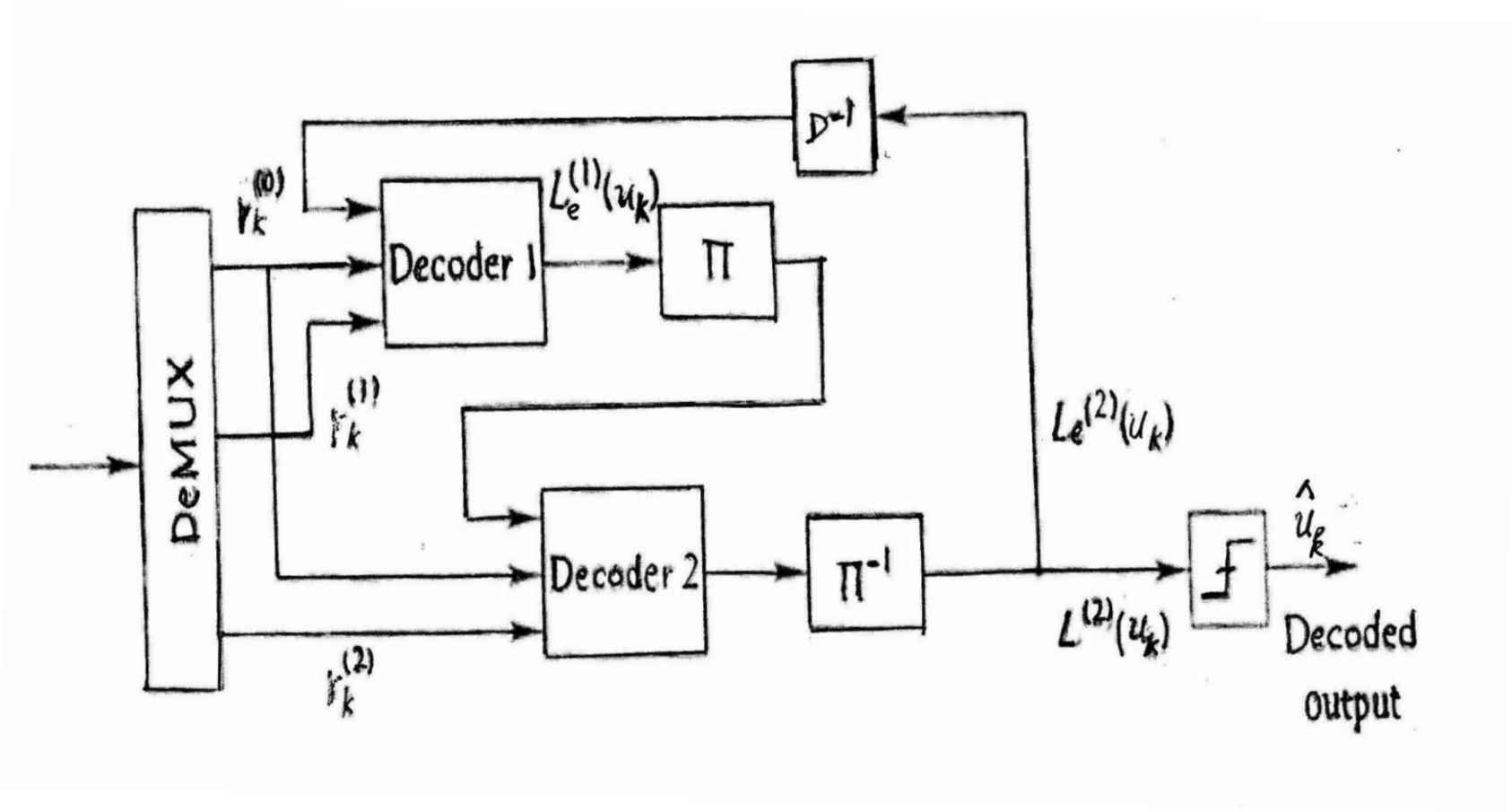
- **The extrinsic output probabilities of Decoder II are deinterleaved and passed back to become prior probabilities to Decoder 1.**

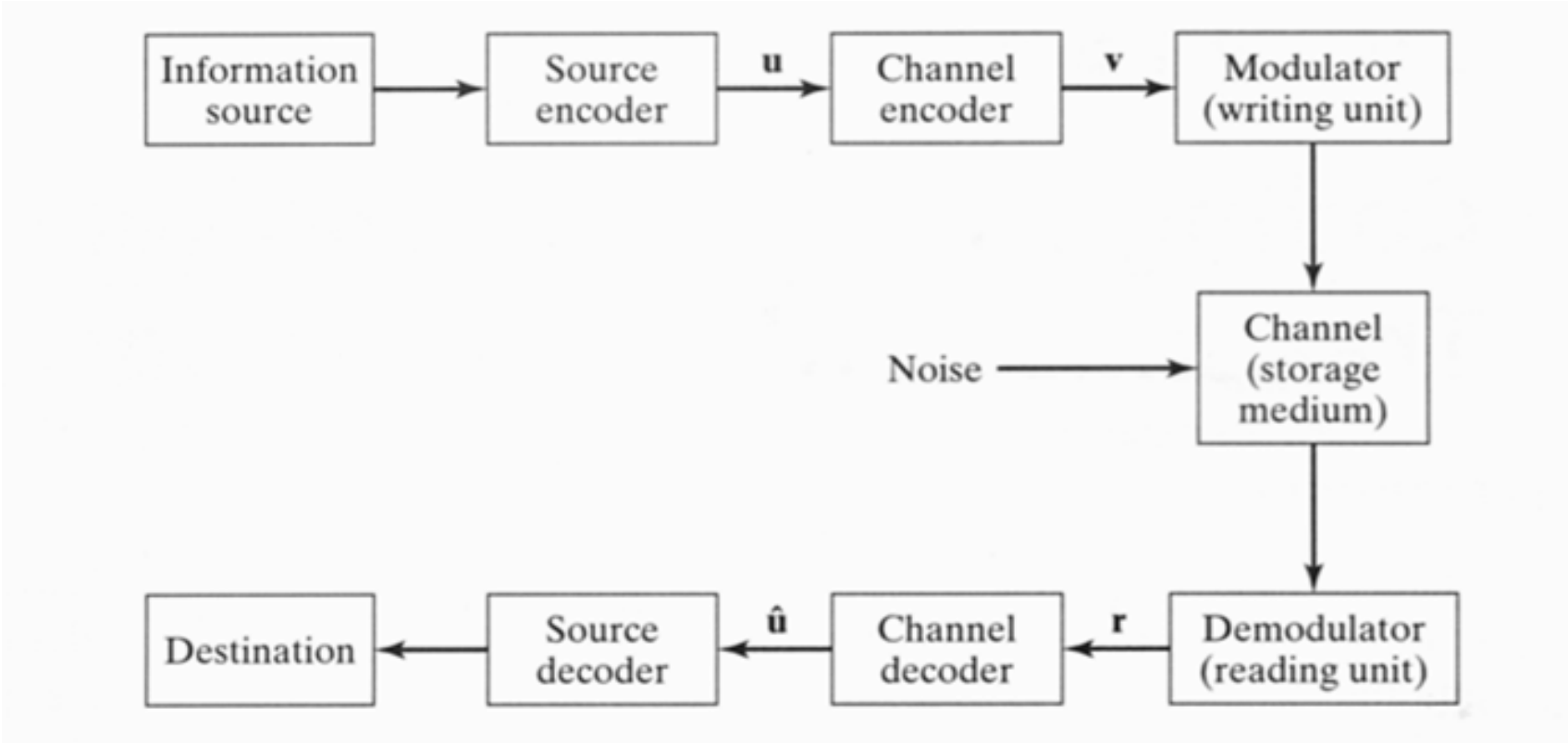
  The process of passing probability information back and forth continues until the decoder determines (somehow) that the process has converged, or until some maximum number of iteration has reached.

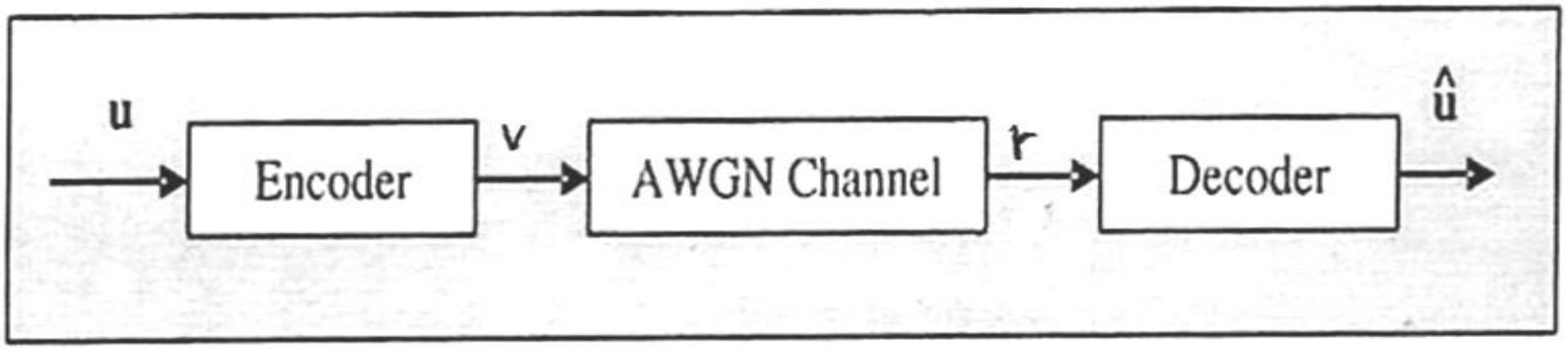
- **When iterative decoding is employed (e.g. turbo decoding), and the *a posteriori* probabilities of the information bits change from iteration to iteration , a MAP decoder gives the best performance.**

  The term *turbo* in the turbo coding has more to do with decoding than encoding. Indeed , it is successive feedback of extrinsic information from the SISO decoders in the iterative decoding process that mimics the feedback of exhaust gasses in a turbocharged engine.

A simplified Turbo decoder structure




Information source
Source encoder
u
Channel encoder
v
Modulator (writing unit)
Noise
Channel (storage medium)
Destination
Source decoder
û
Channel decoder
r
Demodulator (reading unit)


u
Encoder
v
AWGN Channel
r
Decoder
û

7.2 BCJR Algorithm

7.2.1 Log-likelihood Ratio

- We consider transmitting message over a binary- input continuous output memoryless channel . The input u_k equals to $+1$ or -1.
- The log-likelihood ratio (LLR) of a transmitted symbol u_k ,with the received sequence r , is defined as

$$L(u_k) = \ln [\, p(u_k = +1 \mid r) / p(u_k = -1 \mid r) \,] \qquad (7.1)$$

$L(u_k)$ is also denoted as *a posteriori* probability (APP) L-value .

- The *a priori* L-value of an information bit u_k is defined as

$$L_a(u_k) = \ln [\, p(u_k = +1) / p(u_k = -1) \,] \qquad (7.2)$$

- **An L-value can be interpreted as a measure of reliability for a binary random variable .**

  For example, assuming that the *a priori* probabilities of the code bits u are equally likely , that is,

  $p(u=1) = p(u = -1) = 1/2$ **, then, using Bayes' rule , we have**

$$L(r) = \ln [\, p(r \mid u = +1) / p(r \mid u = -1) \,]$$

$$= \ln [p(u = +1 \mid r) / p(u = -1 \mid r)] \quad (7.3)$$

  From this equation, we see that given the received symbol r , a large positive value of $L(r)$ indicates a high reliability that $u = +1$, a large negative value of $L(r)$ indicates a high reliability that $u = -1$.

7.2.2 BCJR Algorithm

- **Here we describe the BCJR algorithm for the case of rate $R = 1/n$ convolutional codes used on a binary-input continuous-output AWGN channel and on a discrete memberless channel (DMC).**

  The decoder inputs are the received sequence r and the *a priori* L-values of the information bits $L_a(u_k)$,

  $k = 0, 1, 2 \ldots, K-1$.

  The algorithm calculates the APP L-values

$$L(u_k) = \ln [\, p(u_k = +1 \mid r) / p(u_k = -1 \mid r) \,] \qquad (7.4)$$

  of each information bit.

  The decoder output is given by

$$u_k^{\wedge} = \begin{cases} +1 & \text{if } L(u_k) > 0 \\ -1 & \text{if } L(u_k) < 0 \end{cases}, \qquad k = 0,1,2,\ldots,K-1 . \qquad (7.5)$$

- **The APP value** $p(u_k = +1 \mid r)$ **can be rewritten as**

$$p(u_k = +1 \mid r) = p(u_k = +1, r) / p(r)$$

$$= \Sigma_{U+} \; p(s_k = s', s_{k+1} = s, r) / p(r) \qquad (7,6)$$

where U+ is the set of transition from state s_k **to the state** s_{k+1} **that can occur if the input bit** $u_k = +1$ **.**

Similarly, we can rewritten $p(u_l = -1 \mid r)$ **as**

$$p(u_k = -1 \mid r) = p(u_k = -1, r) / p(r)$$

$$= \Sigma_{U-} \; p(s_k = s', s_{k+1} = s, r) / p(r) \qquad (7.7)$$

where U- is the set of transition from state s_k **to the state** s_{k+1} **that can occur if the input bit** $u_k = -1$ **.**

- **The APP L-value can be expressed as**

$$L(u_k) = \ln \{\Sigma_{U+}\, p(s_k = s', s_{k+1} = s, r) \,/\, \Sigma_{U-}\, p(s_k = s', s_{k+1} = s, r)\} \qquad (7.8)$$

- **Now we will show how the joint pdf $p(s',s,r)$ can be evaluated recursively.**

Note that $p(s', s, r) = p(s_k = s', s_{k+1} = s, r)$.

The convolutional code introduces a Markov property into the probability structure:

Knowledge of the state at time $t+1$ renders irrelevant knowledge of the state at time t or previous time.

The observation r is divided into three portions : $r_{t<k}$, r_k , $r_{t>k}$, where $r_{t<k}$ represents the portion of the received sequence before time k , and $r_{t>k}$ represents the portion of the received sequence y after time k .

$$r = r_1\, r_2 \ldots r_{k-1}\, r_k\, r_{k+1} \ldots r_N = r_{t<k}\, r_k\, r_{t>k} \qquad (7.9)$$

- We then have $p(s',s,r) = p(s',s, r_{t<k}, r_k, r_{t>k})$ and applying the Bayes's rule we obtain

$$p(s',s,y) = p(r_{t>k} \mid s',s, r_{t<k}, r_k)\ p(s',s, r_{t<k}, r_k)$$
$$= p(r_{t>k} \mid s',s, r_{t<k}, r_k)\ p(s, r_t \mid s', r_{t<k},)\ P(s', r_{t<k})$$
$$= p(r_{t>k} \mid s)\ P(s, r_k \mid s',)p(s', r_{t<k}) \qquad (7.10)$$

where the last equality follows from the fact that the probability of the received branch at time k depends only on the state and input bit at time k.

- Defining

$$\alpha_k(s') \equiv p(s', r_{<k}) \qquad (7.11)$$
$$\gamma_k(s', s) \equiv p(s, r_k \mid s',) \qquad (7.12)$$
$$\beta_{k+1}(s) \equiv p(r_{t>k} \mid s) \qquad (7.13)$$

we then can write $P(s', s, r)$ by the equation

$$p(s',s,r) = \alpha_k(s')\,\gamma_k(s', s)\,\beta_{k+1}(s) \qquad (7.14)$$

- **Forward Recursion**

  We can now rewrite $\alpha_k(s)$ as

$$\alpha_{k+1}(s) \equiv p(s, \mathbf{r}_{t<k+1}) = \Sigma_{s'} \; p(s', s, \mathbf{r}_{t<k}, \mathbf{r}_k)$$

$$= \Sigma_{s'} \; p(s, \mathbf{r}_k \mid s', \mathbf{r}_{t<k}) \, p(s', \mathbf{r}_{t<k})$$

$$= \Sigma_{s'} \; p(s, \mathbf{r}_k \mid s') \, p(s', \mathbf{r}_{t<k})$$

$$= \Sigma_{s'} \, \alpha_k(s') \, \gamma_k(s', s) \qquad (7.15)$$

where the summation is taken over all states at time k.

Thus, once the $\gamma_k(s', s)$ values are known, the $\alpha_{k+1}(s)$ can be calculated recursively.

- **Similarly, we can write the expression for the probability $\beta_{k-1}(s')$ as**

$$\beta_k(s') = \Sigma_s \beta_{k+1}(s)\, \gamma_k(s', s) \qquad (7.16)$$

where *the summation is taken over* all states at time $k+1$, and we can compute a backward metric $\beta_k(s')$ for each state s' at time k using the backward recursion (7.16).

- **The forward recursion begins at time $k = 0$ with the initial condition**

$$\alpha_0(s) = \begin{cases} 1, & s = 0 \\ 0, & s \neq 0 \end{cases} \qquad (7.17)$$

since the encoder starts in the all-zero state $s=0$, and we use (7.15) to recursively compute $\alpha_k(s)$, $k = 0, 1, \ldots, K-1$, where $K = h+m$ is the input sequence length ., m is the memory length of the code.

- **Similarly , the backward recursion begins at time $k=K$ with the initial condition**

$$\beta_K(s) = \begin{cases} 1 & s = 0 \\ 0 & s \neq 0 \end{cases} \qquad (7.18)$$

since the decoder also ends in all-zero state, and we use (7.18) to recursively compute $\beta_k(s)$, $k =, K-1, \ldots, 1, 0$

- **We can write the branch transition probability**

$$\gamma_k(s', s) \equiv p(s, r_k \mid s',) = P(s' s, r_k,) / P(s')$$

$$= p(r_k \mid s', s)\, P(s \mid s')$$
$$= p(r_k \mid s', s) P(u_k)$$
$$= p(r_k \mid v_k)\, P(u_k) \qquad (7.19)$$

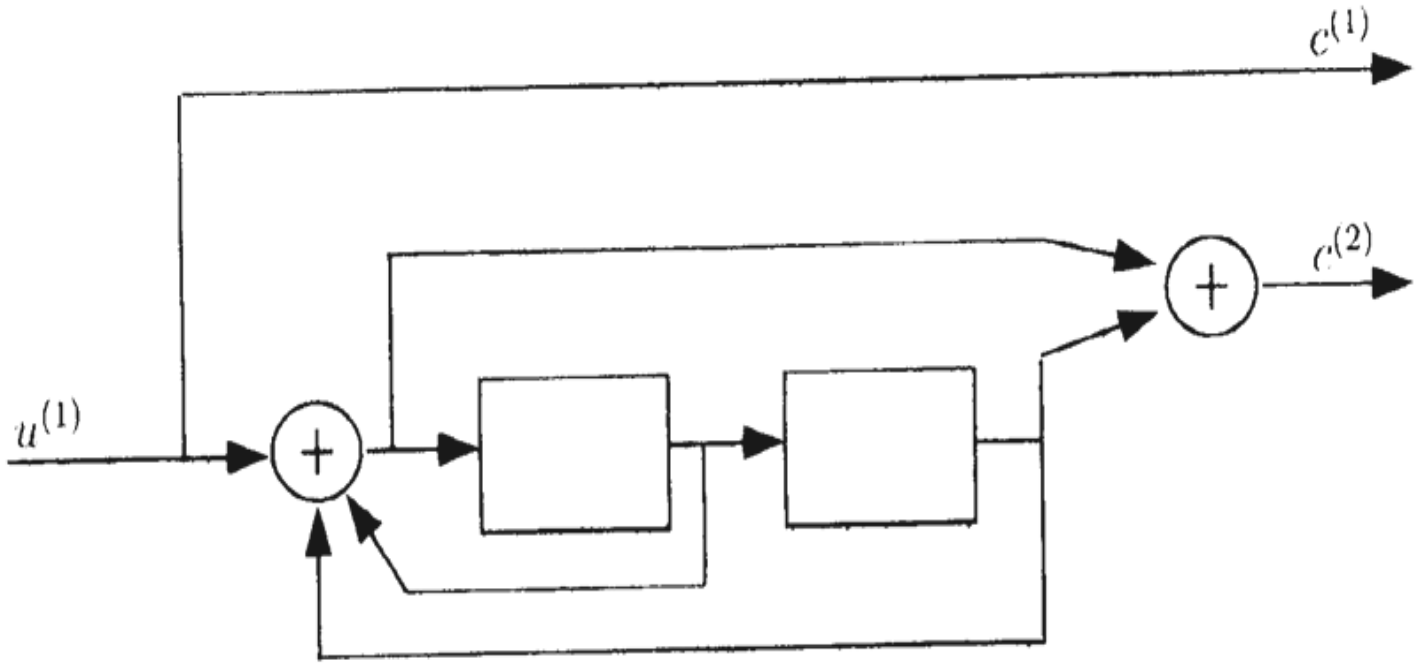
where u_k is the input bit necessary to cause the transition from sate s' to state s , $P(u_k)$ is the *a priori probability* of this bit; v_k is the transmitted codeword associated with this transition $s' \rightarrow s$.

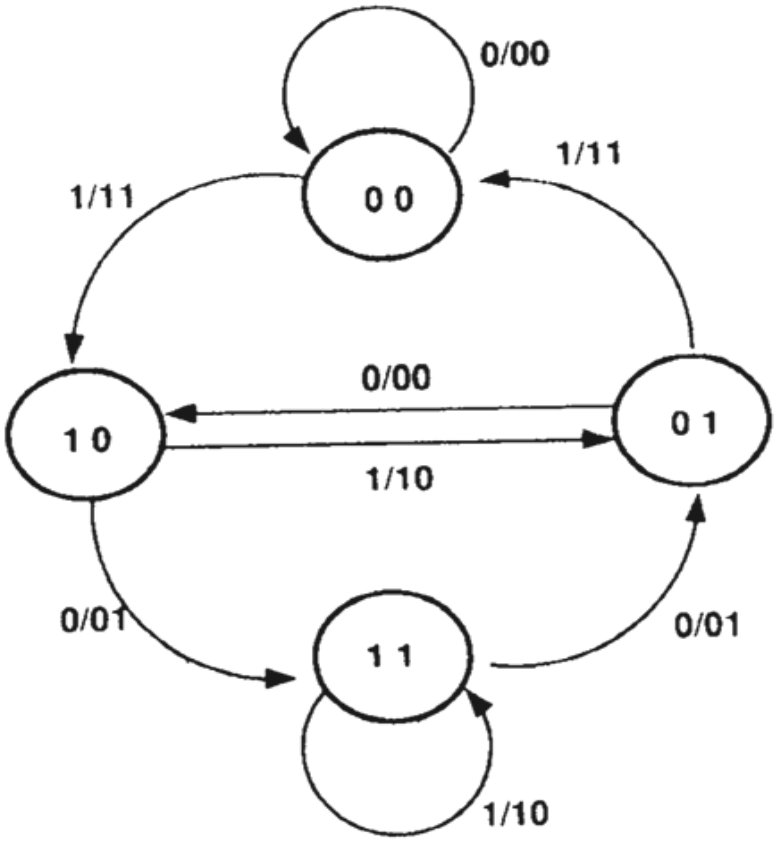
- **Hence , the transition probability density $\gamma_k(s', s)$ is given by the product of the *a priori* probability of the input bit u_k necessary for the transition and the conditional density of the received channel sequence for the value r_k given that the codeword v_k associated with the transition was transmitted.**
- **The *a priori* probability $p(u_k)$ is derived in an iterative decoder from the output of the previous component decoder.**

- **In the following , trellis representation of a convolutional code is used to illustrate the algorithm.**

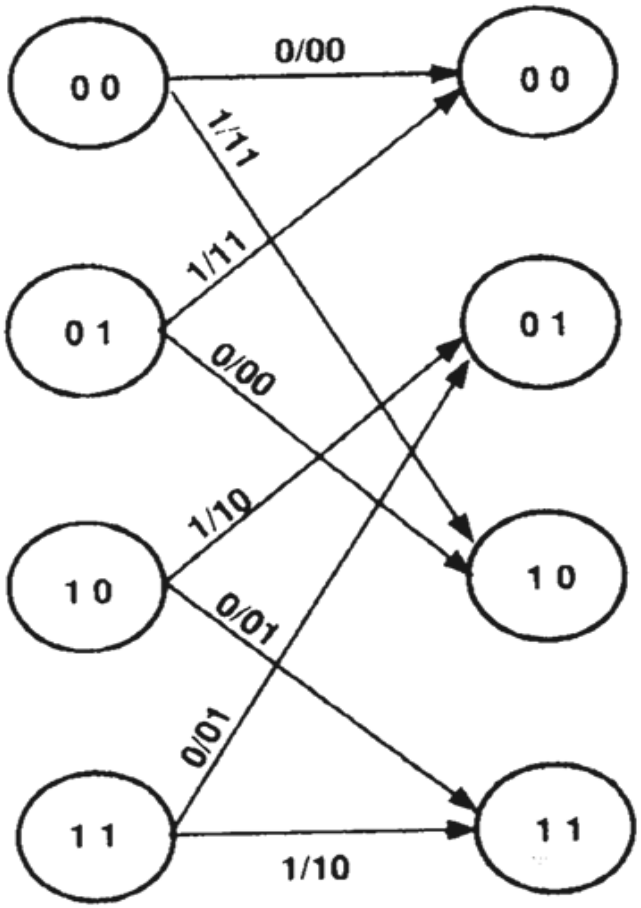
  The functional block diagram of a (2,1,2) recursive convolutional code encoder is shown in ig.8.xx(a) .

  $$G = [\ 1 \quad (1+D^2)/\ 1+D+D^2\ )\ ]$$


$c^{(1)}$
$c^{(2)}$
$u^{(1)}$

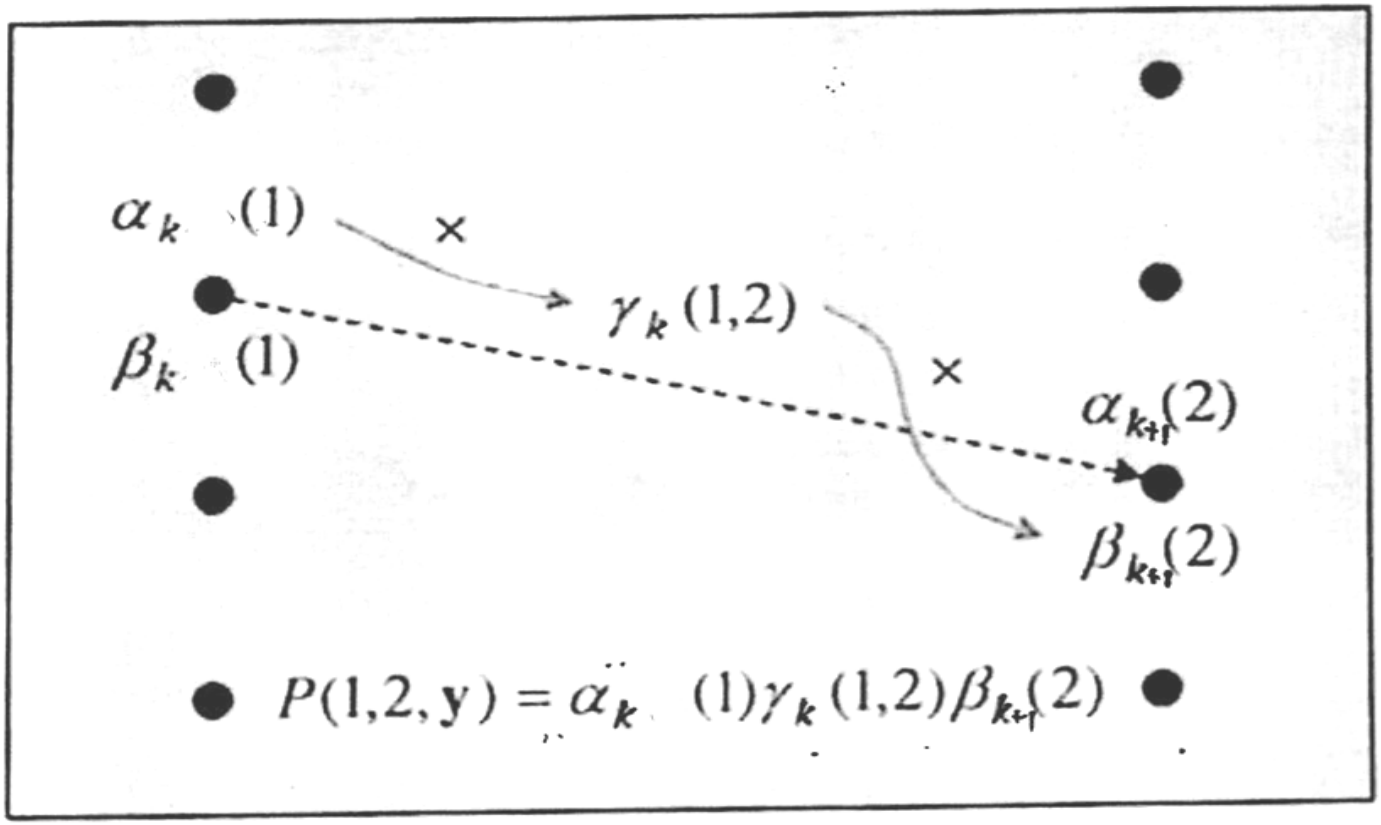

0/00
1/11
1/11
0 0
0/00
1 0
0 1
1/10
0/01
0/01
1 1
1/10

(a)


0 0
0/00
0 0
1/11
1/11
0 1
0 1
0/00
1/10
1 0
1 0
0/01
0/01
1 1
1 1
1/10

(b)

***P*(s', s, y) can be expressed as the three-factor product $\alpha_{k-1}(s')\ \gamma_k(s', s)\ \beta_k(s)$**



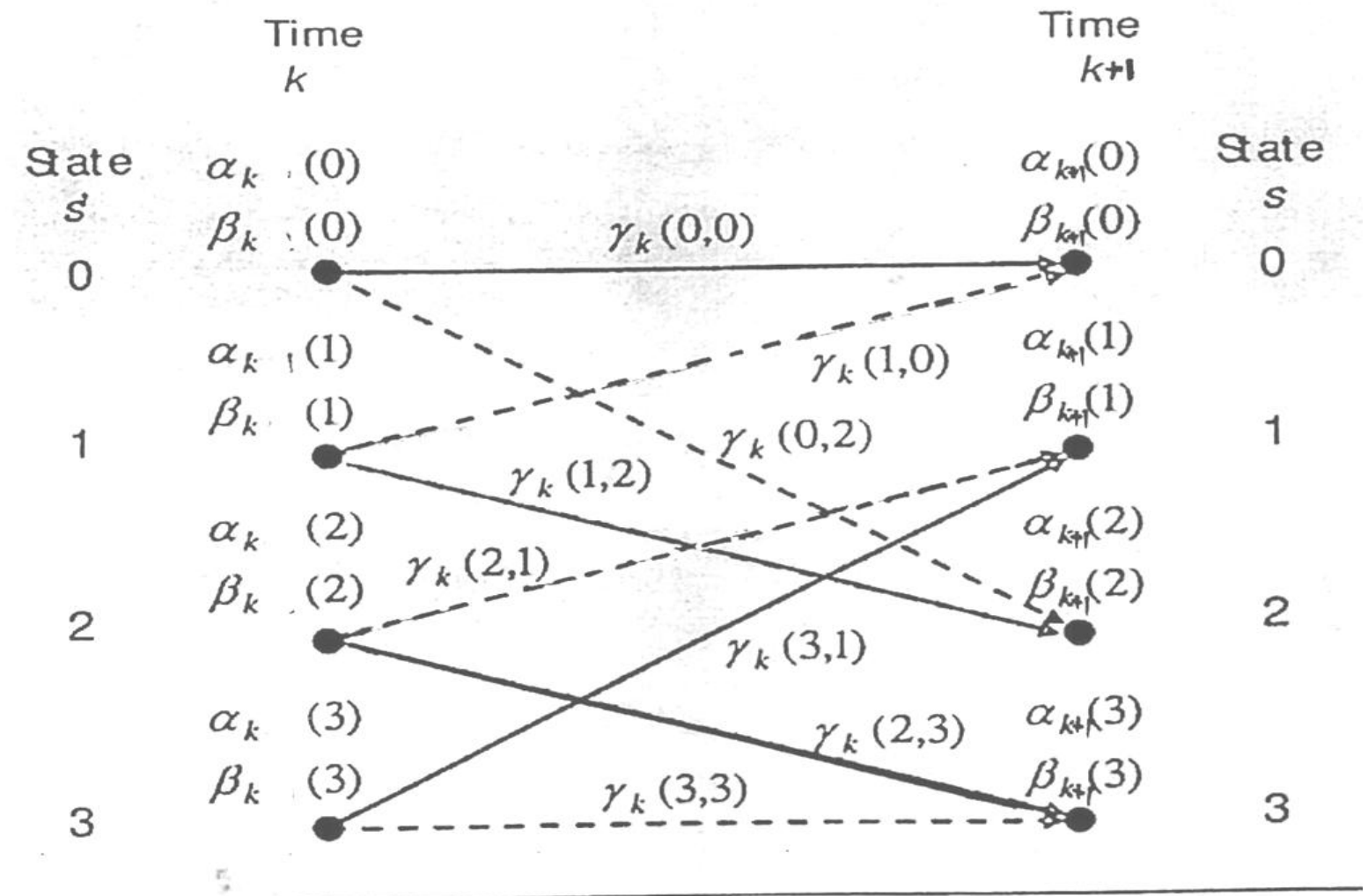
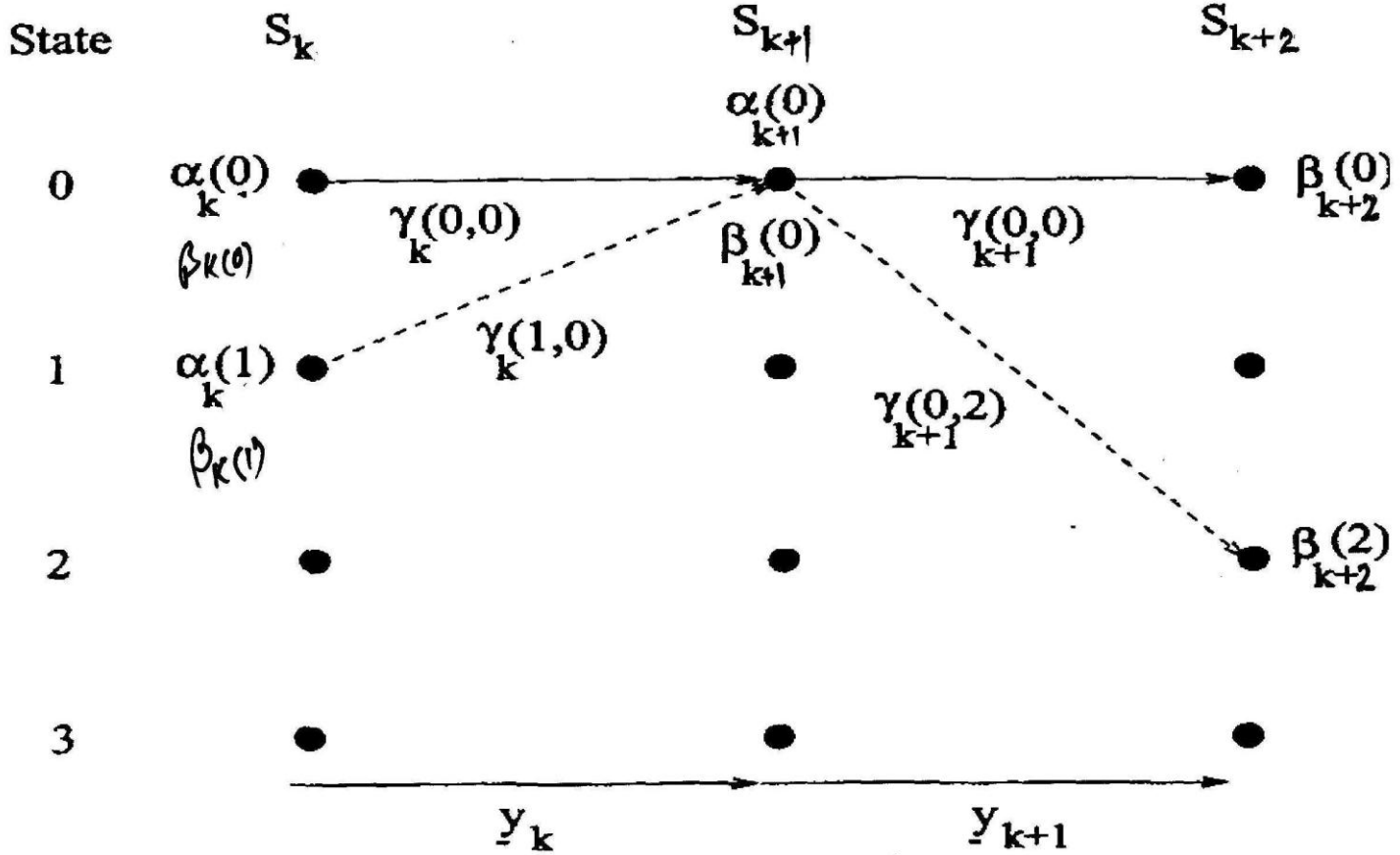


Fig. 3 α, β and γ as trellis labels



$$\alpha_{k+1}(0) = \alpha_k(0)\ \gamma_k(0,0) + \alpha_k(1)\ \gamma_k(1,0)$$

$$\beta_{k+1}(0) = \beta_{k+2}(0)\ \gamma_{k+1}(0,0) + \beta_{k+2}(2)\ \gamma_{k+1}(0,2)$$

Fig. 4. *Recursive calculation of* $\alpha_k(0)$ *and* $\beta_k(0)$.

- **From (7.8) and (7.10) , we can compute the conditional LLR of u_k , given the received value r_k .**
- **The MAP algorithm finds**

  $\alpha_k(s)$ and $\beta_{k+1}(s)$ for all states s throughout the trellis, i.e., *for* $k = 0,1,2,\ldots, K-1$, and $\gamma_k(s', s)$ for all possible transitions from state $s_k = s'$ to state $s_{k+1} = s$, and again for $k = 0,1,2,\ldots, , K-1$

  These values are then used to compute the conditional LLR $L(u_k)$ that the MAP decoder delivers.

- **Summary of BCJR (MAP) Algorithm :**

The MAP decoding of a received sequence r to give the *a posteriori* LLR $L(u_k)$ can carried out as follows.

(1) Initialize forward and backward recursions , $\alpha_0(s)$, *and* $\beta_K(s')$.

(2)As the channel values r_{ki} are received, they and the *a priori* LLRs $L_a(u_k)$,which are provided in an iterative turbo decoder by the other components , are used to calculate $\gamma_k(s', s)$, according to (7-12).

(3) As the channel values $y_{k\,l}$ *and the* $\gamma_k(s', s)$ values are calculated , the forward recursion from (7.15) can be used to calculate $\alpha_k(s)$ *based on* $\alpha_{k-1}(s)$.

(4) Once all the channel values have been received , and $\gamma_k(s', s)$ has been calculated for all $k = 0,1, 2,\ldots,K-1$, the backward recursion from (7.16) can be used to calculate the $\beta_{k-1}(s')$ values based on the $\beta_k(s')$.

(5) Finally, all the calculated values of $\alpha_k(s)$, , $\gamma_k(s', s)$, and $\beta_{k+1}(s')$ are used to calculate the values of $L(u_k)$.

Note : The complexity is about three times of Viterbi algorithm

7.3 Logarithmic BCJR Algorithm :

Log-MAP and Max-Log-MAP

- The log-MAP algorithm was proposed by Robertson *et al* in 1995 . It has performance close to that of the MAP algorithm, but at a fraction of its complexity.
  Also, it is easier to implement and numerically more stable .
- In the log-MAP algorithm, additions substitute the BCJR algorithm multiplications with the aid of the Jacobian logarithm (The max operation).

$$\max^*(a,b) = \ln(e^a + e^b)$$

$$= \max(a,b) + \ln(1 + e^{-|a-b|}) \qquad (7.20)$$

The term $\ln(1 + e^{-|a-b|})$ is usually small which can be stored in a simple look-up table .

- The term $\ln(1 + e^{-|a-b|})$ is negligible in most practical applications. Thus , $\max^*(a,b)$ is replaced by max (a,b) when $|\max(a,b)| \geqq 7$.

- **New variables are defined :**

$$A_k(s) = \ln \alpha_k(s)$$
$$B_k(s) = \ln \beta_k(s)$$
$$\Gamma_k(s', s) = \ln \gamma_k(s', s) \qquad (7.21)$$

With some mathematical manipulations, we can obtain the following equations :

$$\Gamma_k(s', s) = u_k L_a(u_k) / 2 + (L_c / 2) \Sigma_{i=1}^{n} v_{ki} r_{ki} \quad k = 0, 1, \ldots, h-1$$
$$\Sigma_{j=1}^{n} v_{ki} r_{ki} \quad k = h, h+1, \ldots, K-1$$
$$(7.22)$$

$$A_k(s) = \max^{*}_{S'} [A_{k-1}(s') + \Gamma_k(s', s)], \quad k = 0, 1, \ldots, K-1$$
$$(7.23)$$

$$B_k(s) = \max^*_S [B_{k+1}(s) + \Gamma_k(s', s)]$$

$$k = K-1, K-2, \ldots 1, 0 \qquad (7.24)$$

The initial values of A and B in the terminal trellis are

$$A_0(s) = \begin{cases} 0 & s = 0 \\ -\infty & s \neq 0 \end{cases} \qquad B_K(s) = \begin{cases} 0 & s = 0 \\ -\infty & s \neq 0 \end{cases}$$

(7.25)

Consequently ,we have

$$L(u_k) = \max^*_{U+} [A_k(s') + \Gamma_k(s', s) + B_{k+1}(s)] - \max^*_{U-} [A_k(s') + \Gamma_k(s', s) + B_{k+1}(s)]$$

(7.26)

Summary :

The log-BCJR Algorithm :

Step1. Initialize the forward and backward metrics $A_0(s)$ *and* $B_K(s)$ using (7.25) .

Step2. Compute the branch metrics $\Gamma_k(s', s)$ using (7.22) .

Step3. Compute the forward metrics $A_{k-1}(s)$ using (7.23) .

Step 4. Compute the backward metrics $B_{k-1}(s)$,

$k = K, K-1, \ldots, 1$, using (7.24)

Step 5 . Compute the APP L-values $L(u_k)$ using (7.26) .

Step 6. (optional) Compute the hard-decision u_k

- **The algorithm that compute $L(u_k)$ using Eq. (7.26) is called the log-MAP algorithm ; if $\max^*$ function is replaced by max function , then the algorithm is called Max-log- MAP algorithm.**

- **The max-log-MAP algorithm relies on the MAP algorithm but significantly reduces the complexity by neglecting the correction term in (7.20) , that is,**

$$\ln (e^{a} + e^{b}) \doteqdot \max^{*} (a,b)$$

Remarks :

It was shown by Fossorier et al. in 1998 that the performance of a modified SOVA (soft-output Viterbi algorithm) is equivalent to the max-log-MAP algorithm if the soft-update rule is extended.

Fossorirt, M.P. , Bunkert, F., Lin,S., and Hagenauer,J., " On the Equivalence Between SOVA and Max-Log-MAP Decoding ," IEEE Comm. Letters , vol. 2 , no.5 , pp. 137-139 , May 1998.

- **Note 1 :**

From $\ln (e^{a} + e^{b}) = \max^{*} (a,b) = \max (a,b) + \ln (1 + e^{- | a-b |})$

we obtain the formula

$$\ln \Sigma_{i=1}^{n} e^{a\, i} = \ln (e^{a1} + e^{a2} + \ldots + e^{an})$$

$$= \ln (\Delta + e^{an})$$

with $\Delta = e^{a1} + e^{a2} + \ldots + e^{a\,(n-1)} = e^{\delta}$

Therefore ,

$$\ln \Sigma\, e^{a\, I} = \ln (e^{\delta} + e^{an})$$

$$= \max (\delta + a_n) + \ln (1 + e^{- | \delta - an |})$$

$$= \ldots$$

$$= A_{M} + \ln [1 + \Sigma\, e^{(Ai - AM)}]$$

where $A_{M} = \max (a_i) \quad i \neq m$

- **Note 2:**

max* (x,y,z) = max*(max*(x,y), z)

7.4 BCJR for Decoding BPSK Signal over AWGN Channel

- **For a system with memoryless AWGN channel and BPSK modulation, The conditional received sequence probability density $p(r_k \mid v_k)$ is given by**

$$p(r_k \mid v_k) = \prod_{i=1}^{n} p(r_{ki} \mid v_{ki})$$

$$= \prod_{i=1}^{n} \sqrt{(E_c / \pi N_0)} \exp\{-(E_c/N_0)(r_{ki} - a\, v_{ki})^2$$

(7.27)

where

$r_{k\,i}$ **and** v_{ki} **are the individual bits within the received and transmitted codeword, respectively,**

n **is the number of these bits in the codeword, and** a **is the channel gain .**

a =1 for non-fading AWGN channel.

E_c **is the transmitted energy per coded bit ,** $N_0/2$ **is the power spectral**

density of the noise .

$E_c = R_c E_b$ **, and** E_b **is the energy of a message bit .**

- $(r_k - v_k)^2 = \sum_i (r_{k\,i} - a\, v_{ki})^2$ **is the squared Euclidean distance between the received branch and transmitted branch at time** k **.**

- **For a continuous –output AWGN channel, if s' → s is a valid state transition ,**

$$\gamma_k(s', s) = p(r_k \mid v_k)\, p(u_k)$$
$$= P(u_k)\ \left[\sqrt{(E_c / \pi N_0)}\right]^n$$
$$\exp\left[-(E_s/N_0)\ \| (r_k - v_k) \|^2\right]$$

The constant factor $\left[\sqrt{(E_c / \pi N_0)}\right]^n$ is usually dropped for simplicity.

Thus, we have a modified branch metric

$$\gamma_k(s', s) = p(u_k)\ \exp\left(-\ E_c/N_0\ \| (r_k - v_k) \|^2\right) \qquad (7.28)$$

- $L_a(u_k) = \ln\left[p(u_k = +1) / p(u_k = -1)\right]$ **(7.29)**

  is the *a priori* L-value of the bit u_k.

- **We can express the a priori probabilities $P(u_k = \pm 1)$ as exponential terms by written**

$$p(u_k = \pm 1) = [\, p(u_k = +1) / p(u_k = -1) \,]^{\pm 1} / \{\, 1 + [\, P(u_k = +1) / P(u_k = -1) \,]^{\pm 1} \,\}$$

$$= \exp [\, L_a(u_k)]^{\pm 1} / \{\, 1 + \exp [\, L_a(u_k)]^{\pm 1} \,\}$$

$$= \exp [-L_a(u_k)/2\,] \exp [u_k L_a(u_k)/2\,] / \{\, 1 + \exp [-L_a(u_k)\,] \,\}$$

$$= C_{1k} \exp [\, u_k L_a(u_k) / 2\,] \qquad (7.30)$$

$$k = 0,1,\ldots.h-1$$

where, since L-values do not depend on the value of their argument , the parameter A_k is independent of the actual value of u_k .

$$C_{1k} = \exp [-L_a(u_k)/2\,] / \{\, 1 + \exp [-L_a(u_k)\,] \,\}$$

- For termination bits u_k , $k = h, \ldots, h+m-1 = K-1$, where $p(u_k) = 1$ and, $L_a(u_k) = \pm\infty$ for each valid state transition we simply use (7.28) .

  Thus we can write the modified branch metric as

$$\gamma_k(s', s) = C_{1k} \exp[u_k L_a(u_k)/2] \exp\left[-(E_c/N_0) \| (r_k - v_k) \|^2\right] = \ldots$$

$$= C_{1k} C_{2k} \exp[u_k L_a(u_k)/2] \exp\left[(L_c/2)(r_k . v_k)\right]$$

$$k = 0, 1, \ldots, h-1 \qquad (7.31)$$

$$\gamma_k(s', s) = p(u_k) \exp\left[-(E_c/N_0) \| (y_k - x_k) \|^2\right] = \ldots$$

$$= C_{2k} \exp\left[(L_c/2)(r_k . v_k)\right]$$

$$k = h, \ldots, K-1 \qquad (7.32)$$

where $C_{2k} = \| r_k \|^2 + n$ is a constant independent of the codeword v_k, and $L_c = 4 E_s/N_0$ is the channel reliability factor.

- **Thus in the calculation of $p(s', s, y)$, the factors $\Pi_{k=0}^{h-1} C_{1k}$ and $\Pi_{k=0}^{K-1} C_{2k}$ are both contained in the numerators and denominators summations, we can drop these factors and use the exponential function**

$$\gamma_k(s', s) = \exp[u_k L_a(u_k)/2] \exp[(L_c/2)(r_k \cdot v_k)] \quad k = 0,1,\ldots, h-1 \qquad (7.33a)$$

$$\gamma_k(s', s) = \exp[(L_c/2)(r_k \cdot v_k)] \quad k = h, h+1,\ldots, K-1 \qquad (7.33b)$$

as a simplified branch metric .

- **Note that when the input bits are equally likely, $L^a(u_k) = 0$, and the simplified branch metric is given by**

$$\gamma_k(s', s) = \exp[(L_c/2)(r_k \cdot v_k)] \quad k = 0,1,\ldots h, h+1,\ldots, K-1 \qquad (7.34)$$

Example (Lin & Costello, p. 572)

BCJR Decoding of a (2,1,1) systematic recursive convolutional code on an AWGN Channel.

-- The generator matrix of the (2,1,1) systematic recursive convolutional code code is given by

$G(D) = [\ 1 \quad 1/(1+D)\ ]$

The block diagram of the encoder and the corresponding trellis diagram are shown in the following figure .

-- The input sequence has a length of 4 and mapping rule is

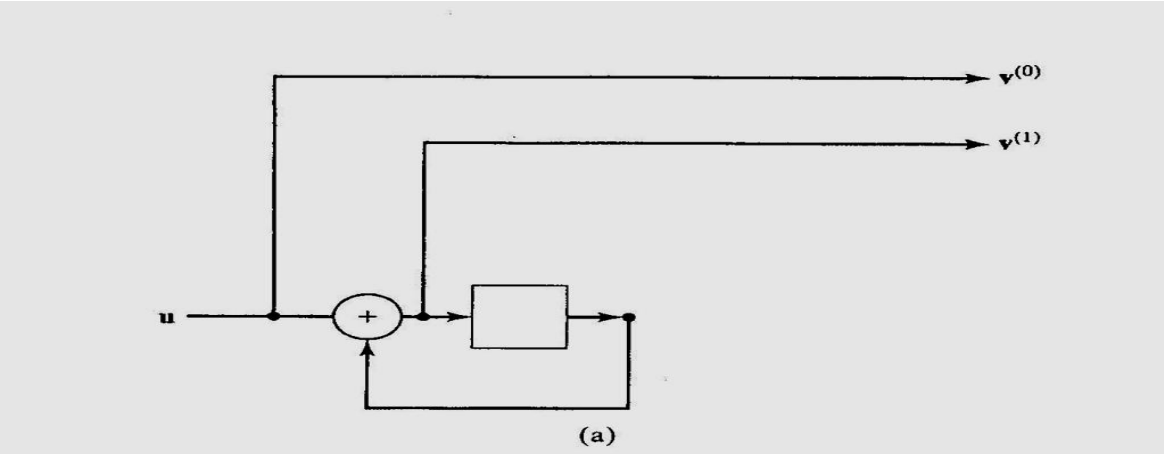
0 --> -1 , 1--> +1

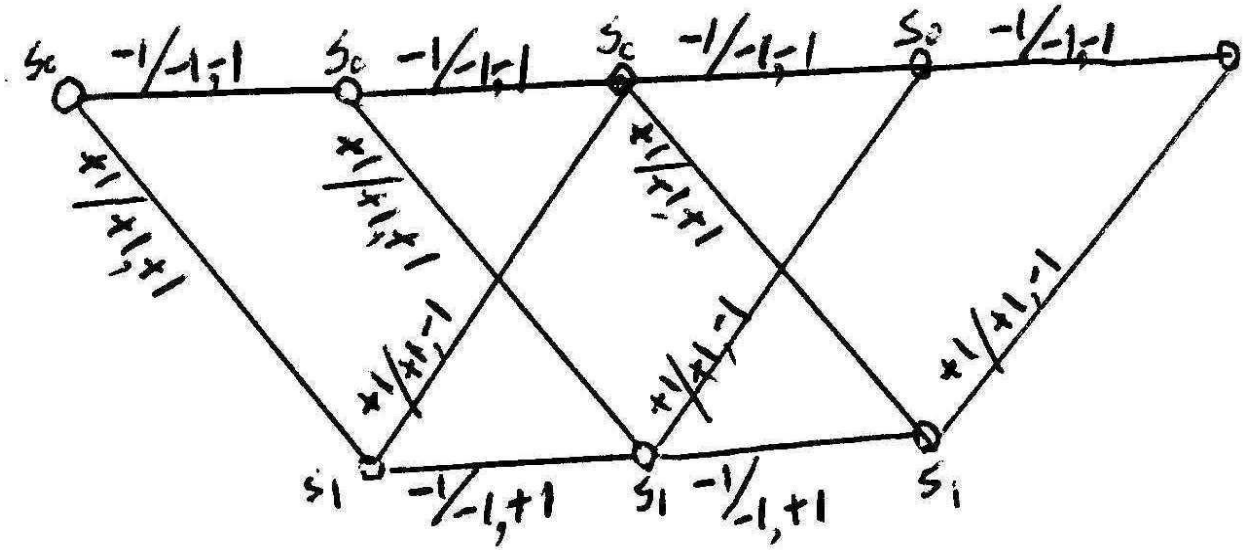
-- Assume that the channel has SNR of $E_s/N_0 = ¼$ (-6.02 dB) and a (normalized by $\sqrt{E_s}$) receiver vector

$R = (r_0 \quad r_1 \quad r_2 \quad r_3) = (r_1^{(1)}\ r_1^{(2)}\ \ r_1^{(1)}\ r_1^{(2)}\ \ r_2^{(1)}\ r_2^{(2)}\ \ r_3^{(1)}\ r_3^{(2)})$

$= (0.8 \ \ 0.1 \ \ 1.0 \ \ -0.5 \ \ -1.8 \ \ 1.1 \ \ 1.6 \ \ -1.6)$

Note : $E_b/N_0 = E_s/(3/8)N_0 = 2/3$


u
+
v(0)
v(1)
(a)


Sc
Sc
Sc
Sc
-1/-1,-1
-1/-1,-1
-1/-1,-1
-1/-1,-1
+1/+1,+1
+1/+1,+1
+1/+1,-1
+1/+1,-1
+1/+1,+1
+1/+1,-1
+1/+1,-1
S1
S1
Si
-1/-1,+1
-1/-1,+1

- **Assuming that the a priori probability of the information bits are equally likely , i.e. $L_a(u_k) = 0$, $k = 1,2,..$,**
  the log-domain branch metrics are calculated as follows.

$$\Gamma_0 (s_0 , s_0) = (-1/2) L_a(u_0) + (1/2) (r_0 . v_0)$$
$$= (1/2) (- 0.8-0.1) = - 0.45$$
$$\Gamma_0 (s_0 , s_1) = (1/2) L_a(u_0) + (1/2) (r_0 . v_0)$$
$$= (1/2) (0.8 +0.1) = 0.45$$
$$\Gamma_1 (s_0 , s_0) = (-1/2) L_a(u_1) + (1/2) (r_1 . v_1)$$
$$= (1/2) (-1.0 + 0.5) = - 0.25$$
$$\Gamma_1 (s_0 , s_1) = (1/2) L_a(u_1) + (1/2) (r_1 . V_1)$$
$$= (1/2) (1.0-0.5) = 0.25$$
$$\Gamma_1 (s_1 , s_0) = (1/2) L_a(u_1) + (1/2) (r_1 . v_1)$$
$$= (1/2) (-1.0 + 0.5) = 0.75$$
$$\Gamma_1 (s_1 , s_1) = (-/2) L_a(u_1) + (1/2) (r_1 . v_1)$$
$$= (1/2) (-1.0 - 0.5) = - 0.75$$

$$\Gamma_2 (s_0 , s_0) = (-1/2) L_a(u_2) + (1/2) (r_2 . v_2)$$
$$= (1/2) (1.8 - 1.1) = 0.35$$
$$\Gamma_2 (s_0 , s_1) = (1/2) L_a(u_2) + (1/2) (r_2 . v_2)$$
$$= (1/2) (-1.8+1.1) = -0.35$$
$$\Gamma_2 (s_1 , s_0) = (1/2) L_a(u_2) + (1/2) (r_2 . v_2)$$
$$= (1/2) (-1.8-1.1) = -1.45$$
$$\Gamma_2 (s_1 , s_1) = (-1/2) L_a(u_2) + (1/2) (r_2 . v_2)$$
$$= (1/2) (1.8+1.1) = 1.45$$

$$\Gamma_3 (s_0 , s_0) = (1/2) (r_3 . v_3)$$
$$= (1/2) (-1.6+1.6) = 0$$
$$\Gamma_3 (s_1 , s_0) = (1/2) (r_3 . v_3)$$
$$= (1/2) (1.6+1.6) = 1.60$$

- **Then we calculate log-domain metrics *using* (*7.23*) as follows.**

$A_1(s_0) = \Gamma_0(s_0, s_0) + A_0(s_0) = -0.45 + 0 = -0.45$

$A_1(s_1) = \Gamma_0(s_0, s_1) + A_0(s_0) = 0.45 + 0 = 0.45$

$A_2(s_0) = \max^*\{[\Gamma_1(s_0, s_0) + A_1(s_0)], [\Gamma_1(s_1, s_0) + A_1(s_1)]$

$= \max^*\{[-0.25 + (-0.45)], [(0.75) + (=0.45)]\}$

$= \max^*(-0.70, +1.20)$

$= 1.20 + \ln(1 + e^{-1.9}) = 1.34$

$A_2(s_1) = \max^*\{[\Gamma_1(s_0, s_1) + A_1(s_0)], [\Gamma_1(s_1, s_1) + A_1(s_1)]$

$= \max^*(-0.20, -0.30) = 0.44$

- **Similarly , we calculate log-doman backward metrics as follows.**

$B_3(s_0) = \Gamma_3(s_0, s_0) + B_4(s_0) = 0 + 0 = 0$

$B_3(s_1) = \Gamma_3(s_1, s_0) + B_4(s_0) = 1.60 + 0 = 1.60$

$B_2(s_0) = \max^*\{[\Gamma_2(s_0, s_0) + B_3(s_0)], [\Gamma_2(s_0, s_1) + B_3(s_1)]\}$

$= \max^*\{[0.35 + 0], [-0.35 + 1.60]\}$

$= \max^*(0.35, 1.25) = 1.25 + \ln(1 + e^{-0.90}) = 1.59$

$B_2(s_1) = \max^*\{[\Gamma_2(s_1, s_0) + B_3(s_0)], [\Gamma_2(s_1, s_1) + B_3(s_1)]\}$

$= \max^*(-1.45, 3.05) = 3.06$

$$B_1(s_0) = \max^*\{ [\Gamma_1 (s_0 , s_0) + B_2(s_0)] ,[\Gamma_1(s_0 , s_1) + B_2(s_1)]\}$$
$$= \max^*(1.34 ,3.31) = 3.44$$
$$B_1(s_1) = \max^*\{ [\Gamma_1 (s_1 , s_0) + B_2(s_0)] ,[\Gamma_1(s_0 , s_1) + B_2(s_1)]\}$$
$$= \max^*(2.34 , 2.31) = 3.02$$

- Finally , we calculate the APP L-values for the three information bits as follows.

$$L(u_0) = [B_1(s_1) + \Gamma_0 (s_0 , s_1) + A_0 (s_0)]$$
$$- [B_1(s_0) + \Gamma_0 (s_0 , s_0) + A_0 (s_0)]$$
$$= 3.47 - 2.99 = 0.48$$
$$L(u_1) = \max^* \{ [B_2(s_0) + \Gamma_1 (s_1 , s_0) + A_1 (s_1)], [B_2(s_1) +$$
$$\Gamma_1 (s_0 , s_1) + A_1 (s_0)]$$
$$- [B_2(s_0) + \Gamma_1 (s_0 , s_0) + A_1(s_0)] + [B_2(s_1) +$$
$$\Gamma_1 (s_1 , s_1) + A_1 (s_1)] \}$$
$$= \max^* (2.79, 2.86) - \max^* (0.89, 2.76) = 0.62$$

$$L(u_2) = \max^* \{ [B_3(s_0) + \Gamma_2(s_1, s_0) + A_2(s_1)], [B_3(s_1) + \Gamma_2(s_0, s_1) + A_2(s_0)] - [B_3(s_0) + \Gamma_2(s_0, s_0) + A_2(s_0)] + [B_3(s_1) + \Gamma_2(s_1, s_1) + A_2(s_1)] \}$$

$$= \max^*(-1.01, 2.59) - \max^*(1.69, 3.49) = -1.02$$

- The hard-decision outputs of the BCJR decoder for the three information bits are

$$\hat{u} = (1, 1, -1)$$

7. 5 Application of BCJR Algorithm :

Iterative Decoding of Turbo Codes

- Consider a rate *1 / 3* systematic convolutional encoder in which the first coded bit , v_{k0} , is equal to the information bit u_k .

  In this case, the *a posteriori* log-likelihood ratio $L(u_k)$ can be generally decomposed into a sum of three elements :

$$L(u_k) = L_c r_k + L_a(u_k) + L_e(u_k) \qquad (7.35)$$

The first two terms are related with the information bit u_k .

The third term, $L_e(u_k)$ is the extrinsic information provided by the decoder based on both the received sequence and on the *a priori* information , excluding both the received sample representing the systematic bit u_k and the *a priori* information $L_a(u_k)$. Derivation is given in Appendix 7A

- **The basic structure of a turbo decoder is shown in Fig. 8.xx .**

  Here, we assume a rate 1/3 parallel concatenated code without puncturing. It uses two MAP decoders.

- **At each time unit k , three output values are received from the channel , one for the information bit u_k , denoted by $r_k^{(0)}$, and two for the parity bits , denoted by $r_k^{(1)}$ and $r_k^{(1)}$.**

  The received sequence can be expressed by a 3K-dimensional vector r as

$$\mathbf{r} = (r_0^{(0)}\, r_0^{(1)}\, r_0^{(2)}\, , r_1^{(0)}\, r_1^{(1)}\, r_1^{(2)}\, , \ldots , r_{K-1}^{(0)}\, r_{K-1}^{(1)}\, r_{K-1}^{(2)}\,)$$

$$= (\mathbf{r}^{(0)}\ \mathbf{r}^{(1)}\ \mathbf{r}^{(2)}\ ) \qquad (7.36)$$

Also, let each transmitted bit be represented using the mapping “0” $\longrightarrow$ -1 and “ 1” $\longrightarrow$ +1 .

- For an AWGN channel with soft (unquantized) outputs , the LLR of a transmitted information bit u_k , denoted as the $L(u_k \mid r_k^{(0)})$, is expressed by

$$L(u_k \mid r_k^{(0)}) = \ln \{ p(u_k = +1 \mid r_k^{(0)}) / p(u_k = -1) \mid r_k^{(0)}) \}$$

$$= \ln \{ p(r_k^{(0)} \mid u_k = +1) \, p(u_k = +1) / p(r_k^{(0)} \mid u_k = -1) \, p(u_k = -1) \}$$

$$= \ln \{ \exp [-(E_s/N_0)((r_k^{(0)} - 1)^2] / \exp [-(E_s/N_0)((r_k^{(0)} + 1)^2] \} + \ln \{ p(u_k = +1) / p(u_k = -1) \}$$

$$= (4 E_s/N_0) r_k^{(0)} + L_a(u_k)$$

$$= L_c r_k^{(0)} + L_a(u_k) \qquad (7.37)$$

where $L_c = 4 E_s/N_0$ is the channel reliability factor and $L_a(u_k)$ is the a priori L-value of the bit u_k .

- **In the case of a transmitted parity bit $v_k^{(j)}$, giving the received value $r_k^{(j)}$, $j = 1,2$, the L-value (before decoding) is given by**

$$L(v_k^{(j)} \mid r_k^{(j)}) = L_c r_k^{(j)} + L_a(v_k^{(j)})$$
$$= L_c r_k^{(j)}, \quad j = 1,2 \qquad (7.38)$$

since in a linear code with equally likely information bits , the parity bits are also equally to be +1 or -1 , and thus the a priori L-values of the parity bits are 0 ;that is ,

$$L_a(v_k^{(j)}) = 0, \quad j = 1,2 \qquad (7.39)$$

Note that $L_a(u_k)$ also equals 0 for the first iteration of the decoder 1 but that thereafter the *a priori* L-values of the information bits ate replaced by extrinsic l-values from the other decoder (say , decoder 2).

- **Iterative decoding Process**

a. The received soft channel L-values $L_c\, r_k^{(0)}$ and $L_c\ r_k^{(1)}$ enter decoder 1 , and $L_c\, r_k^{(0)}$ and the properly interleaved soft channel L-values $L_c\, r_k^{(2)}$ enter decoder 2 .

The output od decoder 1 contains two parts :

(1) $$L^{(1)}(u_k) = \ln \{ \; p(u_k = +1 \mid \mathbf{r}^{(0)}, \mathbf{r}^{(1)}; L_a^{(1)}) / p(u_k = -1 \mid \mathbf{r}^{(0)}, \mathbf{r}^{(1)}; L_a^{(1)}) \}$$ **(7.40)**

(2) $$L_e^{(1)}(u_k) = L^{(1)}(u_k) - [\, L_c\, r_k^{(0)} + L_e^{(1)}(u_k)$$ **(7.41)**

where $\mathbf{L}_a^{(1)} = [\, L_a^{(1)}(u_0), L_a^{(1)}(u_1), ..., L_a^{(1)}(u_{K-1}) \,]$ **is *a priori* input vector for decoder 1**

The extrinsic information $L_e^{(1)}(u_k)$, after interleaving , is then passed to the input of decoder 2 as *a priori* value $L_2^{(2)}(u_k)$.

It is noted that we assume $L_a^{(1)}(u_k) = 0$ in the first iteration .

b. The output of decoder 2 contains two parts :

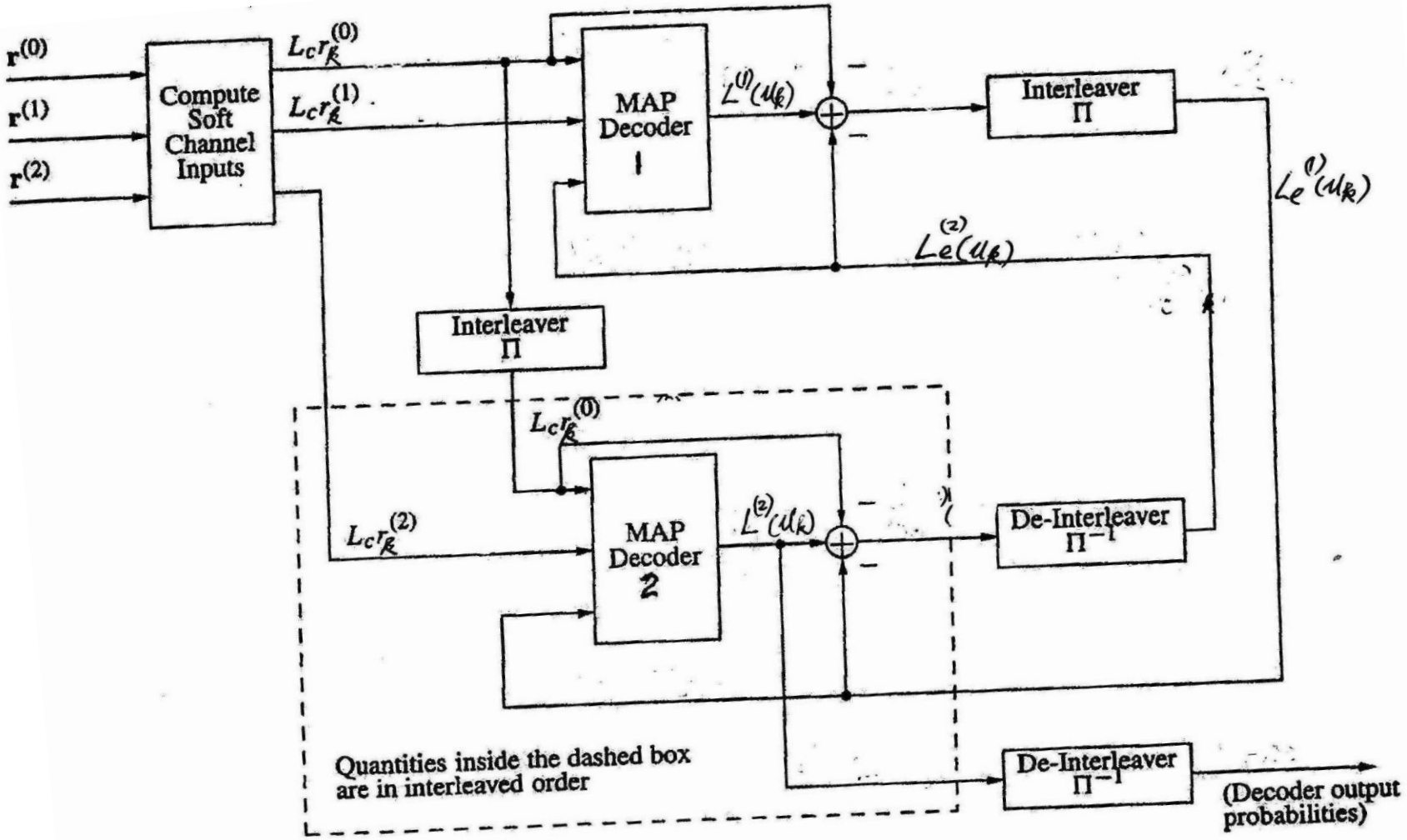
(1) $L^{(2)}(u_k) = \ln \{ p(u_k = +1 \mid \mathbf{r}^{(0)}, \mathbf{r}^{(2)}; \mathbf{L}_a^{(2)}) / p(u_k = -1) \mid \mathbf{r}^{(0)}, \mathbf{r}^{(2)}; \mathbf{L}_a^{(2)}) \}$ **(7.42)**

(2) $L_e^{(2)}(u_k) = L^{(2)}(u_k) - [L_c r_k^{(0)} + L_e^{(1)}(u_k)]$ **(7.43)**

The extrinsic information $L_e^{(2)}(u_k)$ **, after interleaving , is then passed to the input of decoder 2 as** ***a priori*** **value** $L_a^{(1)}(u_k)$**.**

c. Decoding then proceeds iteratively. With each deco0der passing its respectively extrinsic L-values back to the other decoder . This results in a turbo effect in which each estimate becomes successively more reliable. After a sufficient number of iterations , the decoded information bits are determined from the ***a posteriori*** **L-values** $L^{(2)}(u_k)$ **,** $k = 0,1,2,\ldots, K-1$ **.**

Fig. 8. Block diagram of a turbo decoder



Appendix 7A :

BPSK signal transmitted over AWGN channel

- The probability $p(r_k \mid v_k)$ that n values $r_k = r_{k1}\, r_{k2} \ldots r_{kn}$ are received given L values $v_k = v_{k1}\, v_{k2} \ldots v_{kn}$ were transmitted will be equal to the product of the individual probabilities $p(r_{ki} \mid v_{ki})$,$i = 1,2,\ldots, n$.

  In a memoryless channel , the successive transmissions are statistically independent .

$$p(r_k \mid v_k) = \prod_{i=1}^{n} p(r_{k\,i} \mid v_{ki}) \qquad \text{(A-1)}$$

- With BPSK modulation , the transmitted signals have amplitudes $v_{ki} E_c$, *where* $v_{ki} = +1$ *or* -1 , and E_c is the energy transmitted per code bit .
- Let us consider an AWGN channel with noise power spectral density $N_0/2$ and fading amplitude a .

- **At the receiver's matched filter output , the signal amplitude is now $r'_{ki} = \pm a \ (\sqrt{E_c}) + w'$, where w' is a sample of Gaussian noise with zero mean and variance**

  $\sigma_w^2 = N_0/2$.

  Normalizing amplitudes in the receiver we get

  $r_{ki} = r'_{ki} / (\sqrt{E_c}) = a\, v_{ki} + w$

  where the noise w has variance $\sigma_w^2 = N_0 / (2E_c)$

  Finally we have

  $$p(r_{ki} \mid v_{ki}) = \sqrt{(E_c/\pi N_0)} \exp\{-(E_c/N_0)(r_{ki} - a\, v_{ki})^2 \qquad \text{(A-2)}$$

  and then

  $$p(r_k \mid v_k) = [\sqrt{(\sqrt{(E_c/\pi N_0)})}]^n \exp\{-(E_c/N_0)[\textstyle\sum_{i=1}^{n} r_{ki}^2 + a^2 \sum_{i=1}^{L} v_{ki}^2]\} \exp\{2a(E_c/N_0)\textstyle\sum_{i=1}^{n} r_{ki} v_{ki}\}$$
  $$= C_{2i} \exp\{2a(E_c/N_0)\textstyle\sum_{i=1}^{n} r_{ki} v_{kI}\} \qquad \text{(A-3)}$$

The product factors C_{2i} do not depend either on the u_k *sign* or the codeword v_k .

Appendix 7B :

Derivation of $L(u_k) = L^a(u_k) + L_c y_{k1} + L^e(u_k)$ (7.35)

- For a *1/n* systematic convolutional code , the first coded bit x_{k1} is equal to the first coded bit y_{k1} .

From the relation

$$\gamma_k(s', s) = \exp[u_k L^a(u_k)/2] \exp[(L_c/2)(y_k . x_k)]$$

$$k=1,2,\ldots, K \qquad \text{(A-4)}$$

where $L_c = 4a R_c E_b / N_0$. Then , we have

$$\gamma_k(s', s) = \exp[u_k L^a(u_k)/2] \exp[(L_c/2) \Sigma_{j=2}^{n} (y_{kj} x_{kj})] \qquad \text{(A-5)}$$

Let us define

$$\chi_k(s', s) = \exp[(L_c/2) \Sigma_{j=2}^{n} (y_{kj} x_{kj})] \qquad \text{(A-6)}$$

- If $n = 2$ (e.g. , the (2,1,2) RSC as the constituent code for turbo code), then $\chi_k(s' ,s) = \exp [(L_c (y_{k2} x_{k2})/ 2]$

(A-7)

Then we get

$\gamma_k (s', s) = \exp\{ [u_k L^a(u_k) + L_c (y_{k2} x_{k2})] /2 \} \chi_k(s' ,s)$

(A-8)

The APP L-value from (A-2)

$$L(u_k) = \ln \{\Sigma_{U+} p (s_{k-1}=s' , s_k = s , y) / \Sigma_{U--} (s_{k-1}=s' ,s_k= s ,y) \}$$

$$= \ln \{\Sigma_{U+} \beta_k (s) \chi_k (s', s) \alpha_{k-1} (s') \exp [u_k L^a(u_k) + L_c (y_{k1})] /2 \} /$$

$$\Sigma_{U--} \beta_k (s) \chi_k (s', s) \alpha_{k-1} (s') \exp [u_k L^a(u_k) + L_c (y_{k1})] /2 \}$$

(A-9)

Then

$$L(u_k) = L^a(u_k) + L_c y_{k1} + \Sigma_{U+} \beta_k(s) \chi_k(s', s) \alpha_{k-1}(s') /$$

$$\Sigma_{U--} \beta_k(s) \chi_k(s', s) \alpha_{k-1}(s') \qquad \text{(A-10)}$$

We now define

$$L^e(u_k) = \Sigma_{U+} \beta_k(s) \chi_k(s', s) \alpha_{k-1}(s') /$$

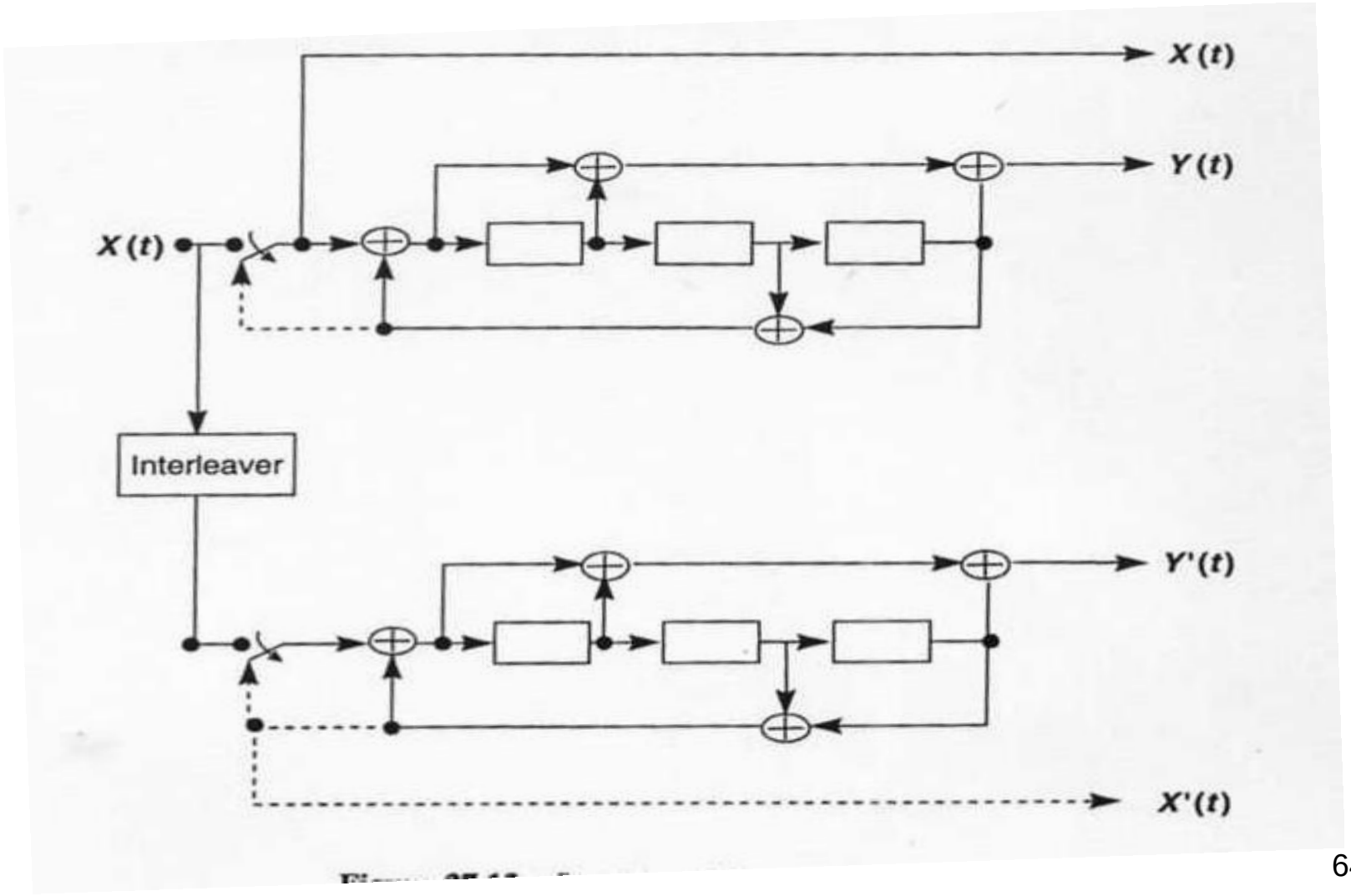
$$\Sigma_{U--} \beta_k(s) \chi_k(s', s) \alpha_{k-1}(s') \qquad \text{(A-11}$$

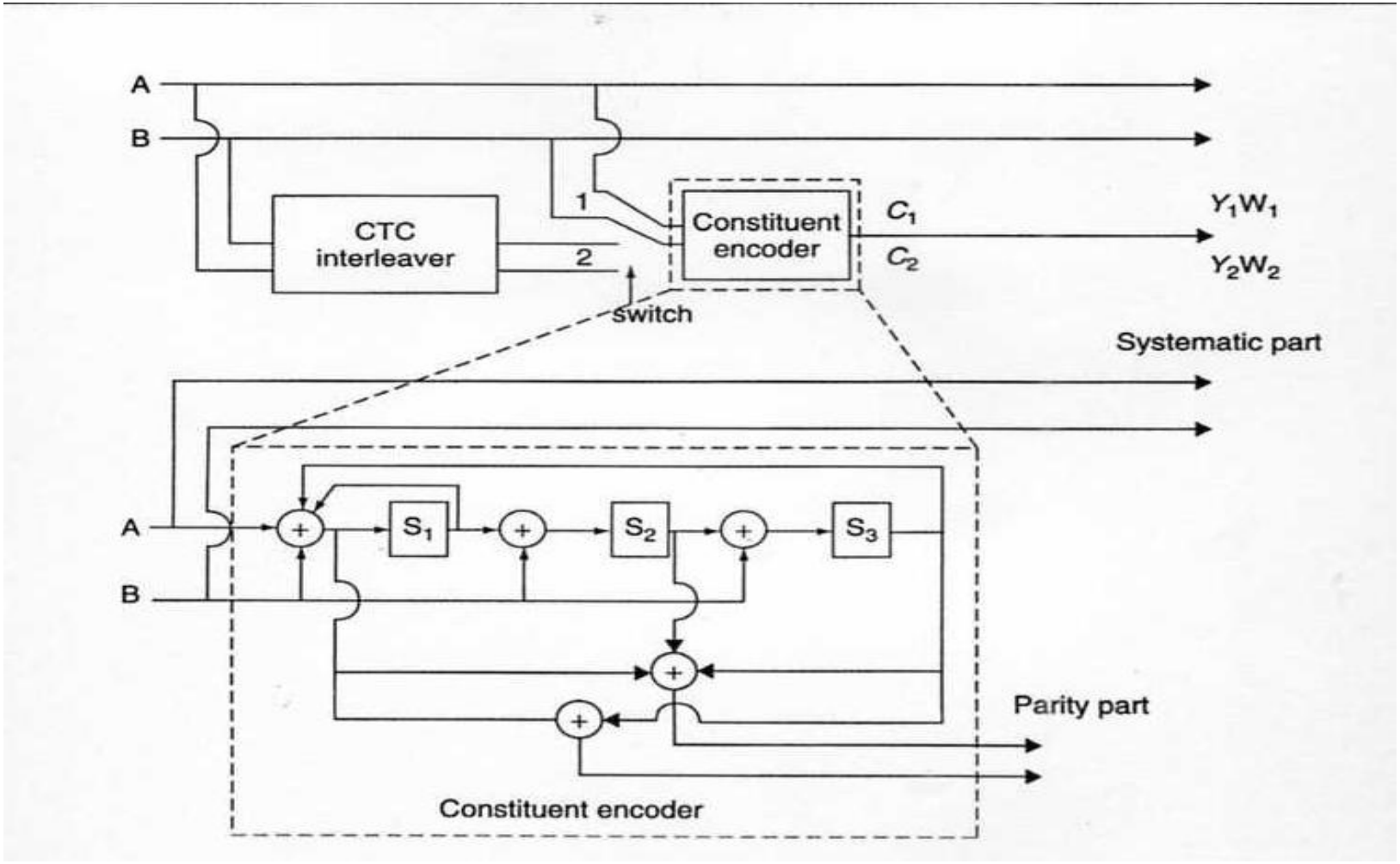
Finally we get

$$L(u_k) = L^a(u_k) + L_c y_{k1} + L^e(u_k) \qquad \text{(A-12)}$$

Appendix 7 C : Turbo code in LTE

$g_2(D) = 1+D+D^3$ $g_1(D) = 1+D^2+D^3$




A
B
CTC
interleaver
1
2
Constituent
encoder
C_1
C_2
Y_1W_1
Y_2W_2
switch
Systematic part
A
B
S_1
S_2
S_3
Parity part
Constituent encoder

Turbo code in WiMAX (IEEE 802.16-e)