Turbo Codes

Manjunatha. P

manjup.jnnce@gmail.com

Professor

Dept. of ECE

J.N.N. College of Engineering, Shimoga

June 29, 2013

[1, 2, 3, 4, 5, 6]

Note:

- Slides are prepared to use in class room purpose, may be used as a reference material
- All the slides are prepared based on the reference material
- Most of the figures used in this material are redrawn, some of the figures/pictures are downloaded from the Internet.
- This material is not for commercial purpose.
- This material is prepared based on Advanced Digital Communication for DECS M Tech course as per Visvesvaraya Technological University (VTU) syllabus (Karnataka State, India).

Introduction

[1, 2, 3, 4, 5, 6]

- Concatenated coding schemes were first proposed by Forney as a method for achieving large coding gains by combining two or more relatively simple building block or component codes
- The resulting codes had the error-correction capability of much longer codes, and they were endowed with a structure that permitted relatively easy to moderately complex decoding
- A turbo code can be thought of as a refinement of the concatenated encoding structure plus an iterative algorithm for decoding the associated code sequence
- Turbo codes were first introduced in 1993 by Berrou, Glavieux, and Thitimajshima
- a scheme is described that achieves a bit-error probability of 10-5 using a rate 1/2 code over an additive white Gaussian noise (AWGN) channel and BPSK modulation at an Eb/N0 of 0.7 dB

- The codes are constructed by using two or more component codes on different interleaved versions of the same information sequence
- For a system with two component codes, the concept behind turbo decoding is to pass soft decisions from the output of one decoder to the input of the other decoder, and to iterate this process several times so as to produce more reliable decisions.

Turbo code concepts

Likelihood functions

- For communications engineering, where applications involving an AWGN channel are of great interest, the most useful form of Bayes theorem expresses the a posteriori Fundamentals of Turbo Codes probability (APP) of a decision in terms of a continuous-valued random variable x in the following form

$$P\,(d=i|x)=\frac{p\,(x|d=i)\;\;P(d=i)}{p\,(x)}\qquad i=1,\ldots,M \tag{1}$$

$$p(x)=\sum_{i=1}^{M} p\,(x|d=i)\;\;P(d=i) \tag{2}$$

- P(d = i), called the a priori probability is the probability of occurrence of the ith signal class
- p(x) is the pdf of the received signal x, yielding the test statistic over the entire space of signal classes

The two-signal class case

- Let the binary logical elements 1 and 0 be represented electronically by voltages +1 and -1, respectively. The variable d is used to represent the transmitted data bit, whether it appears as a voltage or as a logical element
- Let the binary 0 (or the voltage value -1) be the null element under addition. For signal transmission over an AWGN channel, Figure 1 shows the conditional pdfs referred to as likelihood functions
- The rightmost function, p(x—d = +1), shows the pdf of the random variable x conditioned on d = +1 being transmitted
- The leftmost function, p(x—d = -1), illustrates a similar pdf conditioned on d = -1 being transmitted
- The abscissa represents the full range of possible values of the test statistic x generated at the receiver.
- one such arbitrary value xk is shown, where the index denotes an observation in the kth time interval

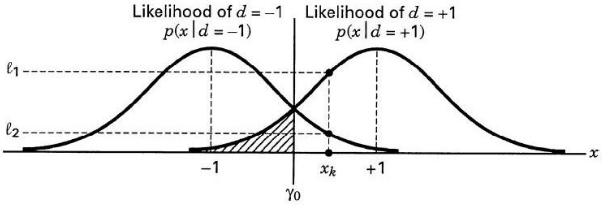


Figure: likelihood functions

- A line subtended from xk intercepts the two likelihood functions, yielding two likelihood values .1 = p(xk—dk = +1) and .2 = p(xk—dk = -1)
- A well-known hard decision rule, known as maximum likelihood, is to choose the data dk = +1 or dk = -1 associated with the larger of the two intercept values, l1 or l2, respectively.
- A similar decision rule, known as maximum a posteriori (MAP), which can be shown to be a minimum probability of error rule, takes into account the a priori probabilities of the data.

$$P(d = +1|x) \underset{H_2}{\overset{H_1}{\gtrless}} P(d = -1|x) \qquad (3)$$

- Equation (3) states that you should choose the hypothesis H1, (d = +1), if the APP P(d = +1—x), is greater than the APP P(d = -1—x) else you should choose hypothesis H2, (d = -1)
- Equation (3) can be rewritten using Baye's theorem as

$$p(x|d = +1)\ P(d = +1) \underset{H_2}{\overset{H_1}{\gtrless}} p(x|d = -1)\ P(d = -1) \qquad (4)$$

- equation 4 is generally expressed in terms of ratio, yielding likelihood ratio test and is given by

$$\frac{p(x|d=+1)}{p(x|d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} \frac{P(d=-1)}{P(d=+1)} \quad \text{or} \quad \frac{p(x|d=+1)\ P(d=+1)}{p(x|d=-1)\ P(d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} 1 \qquad (5)$$

Log-Likelihood Ratio

- By taking the logarithm of the likelihood ratio developed in Equations (3) through (5), we obtain a useful metric called the log-likelihood ratio (LLR)
- It is a real number representing a soft decision out of a detector, designated by as follows

$$L(d|x) = \log\left[\frac{P(d=+1|x)}{P(d=-1|x)}\right] = \log\left[\frac{p(x|d=+1)\ P(d=+1)}{p(x|d=-1)\ P(d=-1)}\right] \qquad (6)$$

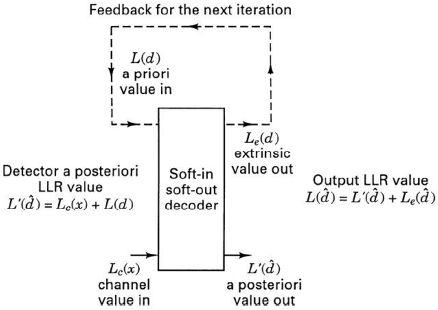
$$L(d|x) = \log\left[\frac{p(x|d=+1)}{p(x|d=-1)}\right] + \log\left[\frac{P(d=+1)}{P(d=-1)}\right] \qquad (7)$$

$$L(d|x) = L(x|d) + L(d) \qquad (8)$$

- where L(x—d) is the LLR of the test statistic x obtained by measurements of the channel output x under the alternate conditions that d = +1 or d = -1 may have been transmitted, and L(d) is the a priori LLR of the data bit d.
- the output LLR L(d) of the decoder is

$$L(\hat{d}) = L_c(x) + L(d) + L_e(\hat{d})$$

- The sign of $L(d)$ denotes the hard decision; that is, for positive values of $L(d)$ decide that $d = +1$, and for negative values decide that $d = -1$. The magnitude of $L(d)$ denotes the reliability of that decision. Often, the value of $Le(d)$ due to the decoding has the same sign as $Lc(x) + L(d)$, and therefore acts to improve the reliability of $L(d)$.
  Principles of Iterative (Turbo) Decoding
- In a typical communications receiver, a demodulator is often designed to produce soft decisions, which are then transferred to a decoder
- The error-performance improvement of systems utilizing such soft decisions compared to hard decisions are typically approximated as 2 dB in AWGN.Such a decoder could be called a soft input/hard output decoder
- With turbo codes, where two or more component codes are used, and decoding involves feeding outputs from one decoder to the inputs of other decoders in an iterative fashion, a hard-output decoder would not be suitable
- for the decoding of turbo codes soft input/soft output decoder is needed


Feedback for the next iteration
$L(d)$
a priori
value in
$L_e(d)$
extrinsic
value out
Detector a posteriori
LLR value
$L'(\hat{d}) = L_c(x) + L(d)$
Soft-in
soft-out
decoder
Output LLR value
$L(\hat{d}) = L'(\hat{d}) + L_e(\hat{d})$
$L_c(x)$
channel
value in
$L'(\hat{d})$
a posteriori
value out

Recursive Systematic codes

- Implementation of turbo codes that are formed by the parallel concatenation of component convolutional codes
- Consider a simple binary rate 1/2 convolutional encoders with constraint length K and memory K-1. The input to the encoder at time k is a bit d_k, and the corresponding codeword is the bit pair (u_k, v_k) are given in equations 1 and 2

$$u_k = \sum_{i=0}^{K-1} g_{1i} d_{k-i} \bmod -2, g_{1i} = 0, 1 \tag{1}$$

$$v_k = \sum_{i=0}^{K-1} g_{2i} d_{k-i} \bmod -2, g_{2i} = 0, 1 \tag{2}$$

- $G_1 = \{g_{1i}\}$ and $G_2 = \{g_{2i}\}$ are the code generators, and d_k is represented as a binary digit
- The nonsystematic convolutional (NSC) code is shown in figure

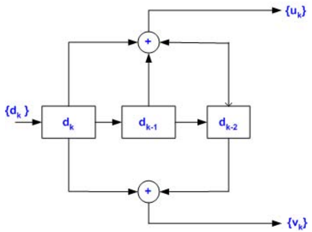


Figure: Nonsystematic Convolutional code(NSC)

- The error performance of a NSC is better only at large E_b/N_0 values
- A class of infinite impulse response(IIR) convolution codes are the building blocks of a turbo code. These blocks are referred to as recursive systematic convolutional (RSC) codes because previously encoded information bits are continually fed back to the encoder's input
- RSC codes result in better error performance at any value of E_b/N_0
- Figure illustrates an example of an RSC code, with K=3, where a_k is recursively calculated as given in equation **??**

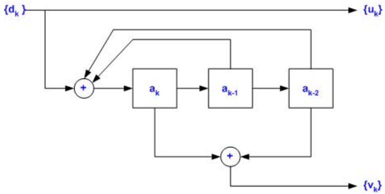


Figure: Recursive Systematic Convolutional code(RSC)

$$a_k = d_k + \sum_{i=1}^{K-1} g'_i a_{k-i} \bmod -2 \tag{3}$$

- The free distance and trellis structures are identical for the RSC code and the NSC code
- The two output sequences $\{u_k\}$ and $\{v_k\}$ do not correspond to the same input sequence $\{d_k\}$ for RSC and NSC codes

The details given in table verifies the trellis diagram

Table: Encoding procedure table

Input bit $d_k = u_k$	Current bit a_k	Starting state a_{k-1}	$ak - 2$	Code bits u_k	v_k	Ending state a_k	a_{k-1}
0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	1
0	1	0	1	0	0	1	0
0	0	1	1	0	1	0	1
1	1	0	0	1	1	1	0
1	0	1	0	1	0	0	1
1	0	0	1	1	1	0	0
1	1	1	1	1	0	1	1

The step by step encoding procedure is as given below

- At any input-bit time, k, the (starting) state of a transition, is denoted by the contents of the two rightmost stages in the register, namely a_{k-1} and a_{k-2}
- For any row on the table, the contents of the a_k stage is found by the modulo-2 addition of bits d_k, a_{k-1} and a_{k-2} on that row
- The output code-bit sequence, $u_k v_k$, for each possible starting state(i.e., a=00, b=10, c=01, d=11) is found by appending the modulo-2 addition of a_k and a_{k-2} to the current data bit $d_k = u_k$
- An interesting property of the recursive shift registers used as component codes for turbo encoders is that the two transitions entering a state should not correspond to the same input bit value

The Trellis structure of RSC encoder of figure 3 is as shown in the figure

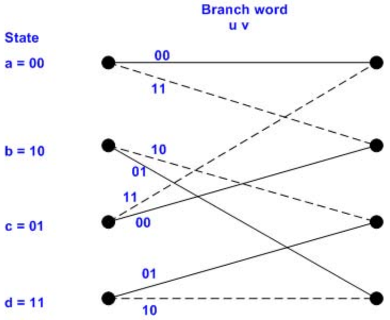


Figure: Trellis Structure

Encoding the input data sequence $\{d_k\}$=1 1 1 0 involves the following steps and is given in table

1. At any instant of time, a data bit d_k becomes transformed to a_k by summing it to the bits a_{k-1} and a_{k-2} on the same row
2. For example, at time k=2, the data bit d_k=1 is transformed to $a_k = 0$ by summing it to the bits a_{k-1} and a_{k-2} on the same row

- The resulting output, $u_k v_k$=10 dictated by the encoder logic circuitry, is the coded-bit sequence associated with time k=2
- At time k=2, the contents, 10, of the rightmost two stages, a_{k-1} a_{k-2} represents the state of the machine at the start of that transition
- The state at the end of that transition is seen as the contents, 01, in the two leftmost stages, a_k a_{k-1} on that same row
- Each row can be described in the same way. Thus the encoded sequence is 1 1 1 0 1 1 0 0

Table: Encoding a bit sequence 1110

Time k	Input bit $d_k = u_k$	First Stage a_k	State at time k a_{k-1}	$ak-2$	Code bits u_k	v_k
1	1	1	0	0	1	1
1	1	1	0	0	1	1
2	1	0	1	0	1	0
3	1	0	0	1	1	1
4	0	0	0	0	0	0
5			0	0		

Consider the parallel concatenation of two RSC encoders shown in Figure

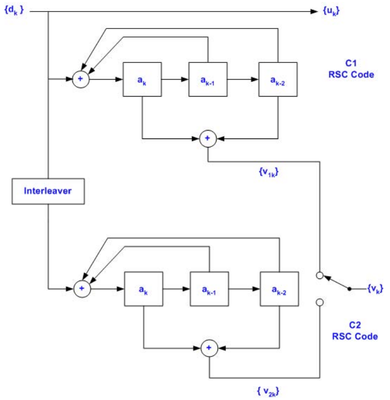


Figure: Parallel concatenation of two RSC encoders

- The switch yielding v_k makes the overall code rate 1/2, otherwise the code rate would be 1/3
- There is no limit to the number of encoders that may be concatenated, and, in general, the component codes need not be identical with regard to constraint length and rate
- The goal in designing turbo codes is to choose the best component codes by maximizing the effective free distance of the code, at low values of E_b/N_0 the weight distribution of the codewords must be optimized

- The weight distribution for the codewords depends on how the codewords from one of the component encoders are combined with codewords from the other encoder
- The pairing of low weight codewords can be avoided by proper design of the interleaver
- However the interleaver would not influence the output codeword weight distribution if the component codes were not recursive

A Feedback Decoder

- A decoded bit $d_k = i$ can be derived from the joint probability as given in equation 4

$$\lambda_k^{i,m} = P\left\{d_k = i, S_k = m | R_1^N\right\} \tag{4}$$

where $S_k = m$ is the encoder state at time k, and R_1^N is a received binary sequence from time k=1 through some time N

- Thus the APP that a decoded data bit $d_k = i$, represented as a binary digit, is obtained by summing the joint probability over all states as given in equation 5

$$P\left\{d_k = i | R_1^N\right\} = \sum_m \lambda_k^{i,m}, i = 0, 1 \tag{5}$$

- The log-likelihood ratio(LLR) is written as the logarithm of the ratio of APPs, as given in equation 6

$$L\left(\hat{d}_k\right) = \log\left[\frac{\sum_m \lambda_k^{1,m}}{\sum_m \lambda_k^{0,m}}\right] \tag{6}$$

- The decoder makes a decision, known as the maximum a posteriori(MAP) decision rule, by comparing $L\left(\overset{\wedge}{d_k}\right)$ to a zero threshold as given in equation 7 and 8

$$\overset{\wedge}{d_k} = 1 \; if \; L\left(\overset{\wedge}{d_k}\right) > 0 \tag{7}$$

$$\overset{\wedge}{d_k} = 0 \; if \; L\left(\overset{\wedge}{d_k}\right) < 0 \tag{8}$$

- For a systematic code, the LLR $L\left(\overset{\wedge}{d_k}\right)$ associated with each decoder bit $\overset{\wedge}{d_k}$ can be described as the sum of the LLR of $\overset{\wedge}{d_k}$, out of the demodulator and of other LLRs generated by the decoder
- Consider the detection of a noisy data sequence that comes from the encoder shown in figure 5 with the use of a decoder shown in figure

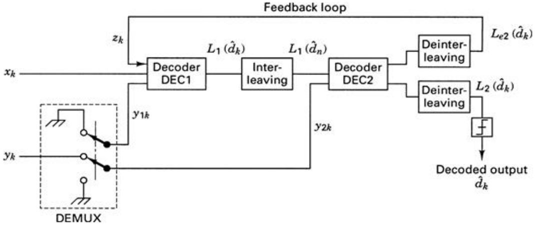


Figure: Feedback Decoder

- The decoder input is made up of a set R_k of two random variables x_k and y_k and are expressed as in equation 9and 10

$$x_k = (2d_k - 1) + i_k \tag{9}$$

$$y_k = (2v_k - 1) + q_k \tag{10}$$

- i_k and q_k are two statistically independent variables accounting for the noise contribution
- The redundant information, y_k, is demultiplexed and sent to decoder DEC1 as y_{1k} when $v_k = v_{1k}$, and to decoder DEC2 as y_{2k} when $v_k = v_{2k}$
- The output of DEC1 has an interleaver structure identical to the one used at the transmitter between the two encoders. This is because the information processed by DEC1 is the noninterleaved output of C1 (corrupted by channel noise). Conversely, the information processed by DEC2 is the noisy output of C2 whose input is the same data going into C1, however permuted by the interleaver.

Decoding with a feedback loop

- The soft-decision output at the decoder is given by equation 11

$$L\left(\overset{\wedge}{d_k}\right) = L_c\left(x_k\right) + L_e\left(\overset{\wedge}{d_k}\right) \tag{11}$$

$$L\left(\overset{\wedge}{d_k}\right) = \log\left[\frac{p\left(x_k|d_k = 1\right)}{p\left(x_k|d_k = 0\right)}\right] + L_e\left(\overset{\wedge}{d_k}\right) \tag{12}$$

- $L_c\left(x_k\right)$ and $L_e\left(\overset{\wedge}{d_k}\right)$ are corrupted by uncorrelated noise, and thus $L_e\left(\overset{\wedge}{d_k}\right)$ may be used as a new observation of d_k by another decoder and the principle here is that a decoder should never be supplied with information that stems from its own input

- The LLR can be rewritten as

$$L_c\left(x_k\right) = -\frac{1}{2}\left(\frac{x_k - 1}{\sigma}\right)^2 + \frac{1}{2}\left(\frac{x_k + 1}{\sigma}\right)^2 = \frac{2}{\sigma^2}x_k \tag{13}$$

- If the inputs $L_1\left(\hat{d}_k\right)$ and y_{2k} to decoder DEC2 are statistically independent, then the LLR $L_2\left(\hat{d}_k\right)$ at the output of DEC2 is

$$L_2\left(\hat{d}_k\right) = f\left[L_1\left(\hat{d}_k\right)\right] + L_{e2}\left(\hat{d}_k\right) \tag{14}$$

with

$$L_1\left(\hat{d}_k\right) = \frac{2}{\sigma_0^2}x_k + L_{e1}\left(\hat{d}_k\right) \tag{15}$$

- where $f\left[\bullet\right]$ indicates a functional relationship
- Due to the interleaving between DEC1 and DEC2, the extrinsic information $L_{e2}\left(\hat{d}_k\right)$ and the observations x_k and y_{1k} are weak correlated. Therefore, they can be jointly used for the decoding
- $L_{e2}\left(\hat{d}_k\right)$ will have the same sign as d_k which increases the LLR and thereby improves the reliability

Turbo code error performance example

Performance results have been presented for a rate 1/2, K = 5 encoder implemented with generators G1 = 1 1 1 1 1 and G2 = 1 0 0 0 1, using parallel concatenation and a 256X256 array interleaver. The modified Bahl algorithm was used with a data block length of 65,536 bits. After 18 decoder iterations, the bit-error probability PB was less than 10^{-5} at $E_b/N_0 = 0.7$ dB. The error-performance improvement as a function of the number of decoder iterations is seen in Figure 7

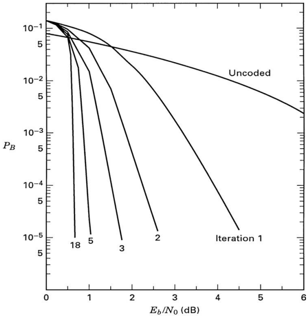


Figure: Bit-error probability as a function of E_b/N_0 and multiple iterations

Introduction

- In traditional decoding, the demodulator makes hard decision of the received symbol.
  -Its disadvantage is that, if the value of some bit is determined with greater certainty, the decoder cannot make use of this information
- In tubo codes Soft In Soft Out(SISO) decoding is used.
- Since turbo coding has two encoders, there will be two decoders for o/p from both encoders.
- The decoder outputs for each data bit an estimate expressing the probability of transmitted data(i.e., a posteriori probability,APP) for a number of iterations
- At each round, decoder re-evaluates this estimates using information from the other decoder and only in the final stage hard decisions will be made.

The MAP(Maximum A Posteriori) algorithm

- In this algorithm the process of decoding starts with
  - formation of a posteriori probabilities(APPs) for each data bit, followed by
  - choosing the data bit value that corresponds to the maximum a posteriori(MAP) probability for that data bit.
- Let us derive the MAP decoding algorithm for convolutional codes assuming an AWGN
- Starting with ratio of APPs, known as likelihood ratio $\Lambda(\hat{d}_k)$, or its logarithm, $L(\hat{d}_k)$ called the log-likelihood ratio given by eqn 16 and 17

$$\Lambda(\hat{d}_k) = \frac{\sum\limits_m \lambda_k^{1,m}}{\sum\limits_m \lambda_k^{0,m}} \tag{16}$$

$$L(\hat{d}_k) = \log\left[\frac{\sum\limits_m \lambda_k^{1,m}}{\sum\limits_m \lambda_k^{0,m}}\right] \tag{17}$$

- where $\lambda_k^{i,m}$ is the joint probability that $d_k = i$ and state $S_k = m$ conditioned on the received binary sequence R_1^N given by eqn 18

$$\lambda_k^{i,m} = P(d_k = i, S_k = m | R_1^N) \tag{18}$$

- R_1^N represents a corrupted code bit sequence given by eqn 19.
- The output sequence from the modulator is presented to the decoder as a block of N bits at a time

$$R_1^N = \{R_1^{k-1}, R_k, R_{k+1}^N\} \tag{19}$$

- Substituting eqn 19 in eqn 18 and applying Bayes' rule,

$$\lambda_k^{i,m} = P(\underbrace{d_k = i, S_k = m}_{A} \mid \underbrace{R_1^{k-1},}_{B} \underbrace{R_k}_{C}, \underbrace{R_{k+1}^N}_{D}) \tag{20}$$

- From the Bayes' rule we know that,

$$P(A|B,C,D) = \frac{P(A,B,C,D)}{P(B,C,D)} = \frac{P(B|A,C,D)P(A,C,D)}{P(B,C,D)} \tag{21}$$

$$= \frac{P(B|A,C,D)P(D|A,C)P(A,C)}{P(B,C,D)} \tag{22}$$

Applying the above rule to eqn 20 gives,

$$\lambda_k^{i,m} = \frac{P(R_1^{k-1}|d_k = i, S_k = m, R_k^N)P(R_{k+1}^N|d_k = i, S_k = m, R_k)P(d_k = i, S_k = m, R_k)}{P(R_1^k)} \tag{23}$$

where $R_k^N = \{R_k, R_{k+1}^N\}$. In the next section the 3 numerator factors on the right side of eqn 23 will be defined and developped as Forward state metric, Reverse state metric and The branch metric.

The State Metrics and The Branch Metric

- Defining the first numerator factor of eqn 23 as the forward state metric at time k and state m, as α_k^m

$$P(R_1^{k-1}| \overbrace{d_k = i,}^{IRRELEVANT} \quad S_k = m, \quad \overbrace{R_k^N}^{IRRELEVANT} \quad) = P(R_1^{k-1}|S_k = m) \triangleq \alpha_k^m \tag{24}$$

- In eqn 24 the two terms are irrelevant because, past is not affected by the future ; i.e., $P(R_1^{k-1})$ is independent of $d_k = i$ and the sequence R_k^N.
- Since the encoder has memory, the state $S_k = m$ is based on the past, so this term is relevant.
- The second numerator factor of eqn 23 represents a reverse state metric β_k^m at time k and state m, given by,

$$P(R_{k+1}^N|d_k = i, S_k = m, R_k) = P(R_{k+1}^N|S_{k+1} = f(i,m)) \triangleq \beta_{k+1}^{f(i,m)} \tag{25}$$

where f(i,m) is the next state, given an input i and state m, and $\beta_{k+1}^{f(i,m)}$ is the reverse state metric at time k+1 and state f(i,m).

- The third numerator factor of eqn 23 represents a branch metric $\delta_k^{i,m}$ at time k and state m, given by

$$P(d_k = i, S_k = m, R_k) \triangleq \delta_k^{i,m} \tag{26}$$

- Substituting eqns 24 through 26 into eqn 23 gives a more compact expression for the joint probability, as follows

$$\lambda_k^{i,m} = \frac{\alpha_k^m \delta_k^{i,m} \beta_{k+1}^{f(i,m)}}{P(R_1^N)} \tag{27}$$

- substituting eqn 27 in eqns 16 and 17,

$$\Lambda(\hat{d}_k) = \frac{\sum_m \alpha_k^m \delta_k^{1,m} \beta_{k+1}^{f(1,m)}}{\sum_m \alpha_k^m \delta_k^{0,m} \beta_{k+1}^{f(0,m)}} \tag{28}$$

and

$$L(\hat{d}_k) = \log_e \left[\frac{\sum_m \alpha_k^m \delta_k^{1,m} \beta_{k+1}^{f(1,m)}}{\sum_m \alpha_k^m \delta_k^{0,m} \beta_{k+1}^{f(0,m)}} \right] \tag{29}$$

- where $\Lambda(\hat{d}_k)$ and $L(\hat{d}_k)$ are the likelihood ratio and the log-likelihood ratio of the k-th data bit respectively.

Calculating the Forward State Metric

- Starting from eqn 24 α_k^m can be expressed as summation of all possible transition probabilities from time k-1,

$$\alpha_k^m = \sum_{m'} \sum_{j=0}^{1} P(d_{k-1} = j, S_{k-1} = m', R_1^{k-1} | S_k = m) \tag{30}$$

- Rewriting R_1^{k-1} as $\{R_1^{k-2}, R_{k-1}\}$ and from Bayes' Rule,

$$\begin{aligned} \alpha_k^m &= \sum_{m'} \sum_{j=0}^{1} P(R_1^{k-2} | S_k = m, d_{k-1} = j, S_{k-1} = m', R_{k-1}) && (31) \\ &= XP(d_{k-1} = j, S_{k-1} = m', R_{k-1} | S_k = m) && (32) \end{aligned}$$

$$= \sum_{j=0}^{1} P(R_1^{k-2} | S_{k-1} = b(j, m)) P(d_{k-1} = j, S_{k-1} = b(j, m), R_{k-1}) \tag{33}$$

- where b(j,m) is the state going backwards in time from state m, via the previous branch corresponding to input j.

- Using eqns 24 and 26 in eqn 33 gives,

$$\alpha_k^m = \sum_{j=0}^{1} \alpha_{k-1}^{b(j,m)} \delta_{k-1}^{j,b(j,m)} \tag{34}$$

- Eqn 34 indicates that a new forward state metric at time k and state m is obtained by summing two weighted state metrics from time k-1, corresponding to data bits 0 and 1.

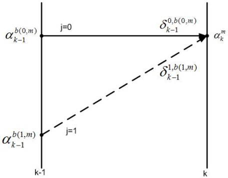


Figure: Forward state metric

- The two possible transitions from the previous time terminate on the same state m at time k as shown in fig **??**

Calculating the Reverse State Metric

- Starting from eqn 25 where, $\beta_{k+1}^{f(i,m)} = P(R_{k+1}^N | S_{k+1} = f(i,m))$ we have,

$$\beta_k^m = P(R_k^N | S_k = m) = P(R_k, R_{k+1}^N | S_k = m) \tag{35}$$

- We can express β_k^m as the summation of possible transition probabilities to time k+1, as

$$\beta_k^m = \sum_{m'} \sum_{j=0}^{1} P(d_k = j, S_{k+1} = m', R_k, R_{k+1}^N | S_k = m,) \tag{36}$$

- Using Bayes' Rule,

$$\begin{aligned} \beta_k^m &= \sum_{m'} \sum_{j=0}^{1} P(R_{k+1}^N | S_k = m, d_k = j, S_{k+1} = m', R_k) &(37) \\ &= XP(d_k = j, S_{k+1} = m', R_k | S_k = m) &(38) \end{aligned}$$

- $S_k = m$ and $d_k = j$ in the first term of eqn **??** defines the path resulting in $S_{k+1} = f(j,m)$, the next state given an input j and state m.

- So replacing $S_{k+1} = m'$ with $S_k = m$ in the second term of eqn **??**

$$\beta_k^m = \sum_{j=0}^{1} P(R_{k+1}^N | S_{k+1} = f(j,m)) P(d_k = j, S_k = m, R_k) \quad (39)$$

$$= \sum_{j=0}^{1} \delta_k^{j,m} \beta_{k+1}^{f(j,m)} \quad (40)$$

- Eqn **??** indicates that a new reverse state metric at time k and state m is obtained by summing two weighted state metrics from time k+1 associated with transitions corresponding to data bits 0 and 1 as shown in fig **??**

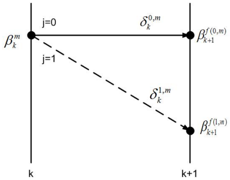


Figure: Reverse state metric

Calculating the Branch Metric

- Starting with eqn 26 ,

$$\delta_k^{i,m} = P(d_k = i, S_k = m, R_k) \tag{41}$$

$$= P(R_k | d_k = i, S_k = m) P(S_k = m | d_k = i) P(d_k = i) \tag{42}$$

- Where R_k represents the sequence x_k, y_k
  x_k is the noisy received bit and
  y_k is the corresponding noisy received parity bit.
- Noise affecting the data and parity are independent $\Rightarrow$ current state is independent of current input, and it can be any one of 2^v states, where v is the no. of memory elements.

$$P(S_k = m | d_k = i) = \frac{1}{2^v} \tag{43}$$

$$\delta_k^{i,m} = P(x_k | d_k = i, S_k = m) P(y_k | d_k = i, S_k = m) \frac{\pi_k^i}{2^v} \tag{44}$$

- Where π_k^i is defined as $P(d_k = i)$, the a posteriori probability of d_k.

- The probability $P(X_k = x_k)$ of a random variable, X_k is related to the pdf $p_{x_k}(x_k)$ as follows,

$$P(X_k = x_k) = p_{x_k}(x_k)dx_k \tag{45}$$

- For an AWGN channel,where the noise has zero mean and variance σ^2, using eqn 45 to replace the probability terms in eqn **??** with their pdf equivalents,

$$\delta_k^{i,m} = \frac{\pi_k^i}{2^v\sqrt{2\pi\sigma}} \exp\left[-\frac{1}{2}\left(\frac{x_k - u_k^i}{\sigma}\right)^2\right] dx_k \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{1}{2}\left(\frac{y_k - v_k^{i,m}}{\sigma}\right)^2\right] dy_k \tag{46}$$

- where u_k and v_k represent the transmitted data bits and parity bits, dx_k and dy_k are the differentials of x_k and y_k and get absorbed into the constant A_k.
- u_k^i is independent of state m, and $v_k^{i,m}$ is dependent on state m.
- Simplifying the eqn 46, we get

$$\delta_k^{i,m} = A_k \pi_k^i \exp\left[\frac{1}{\sigma^2}\left(x_k u_k^i + y_k v_k^{i,m}\right)\right] \tag{47}$$

- Substituting eqn 47 in eqn 28 we get,

$$\Lambda(\hat{d}_k) = \pi_k \exp\left(\frac{2x_k}{\sigma^2}\right) \frac{\sum_m \alpha_k^m \exp\left(\frac{y_k v_k^{1,m}}{\sigma^2}\right) \beta_{k+1}^{f(1,m)}}{\sum_m \alpha_k^m \exp\left(\frac{y_k v_k^{0,m}}{\sigma^2}\right) \beta_{k+1}^{f(0,m)}} \tag{48}$$

$$= \pi_k \exp\left(\frac{2x_k}{\sigma^2}\right) \pi_k^e \tag{49}$$

$$L(\hat{d}_k) = L(d_k) + L_c(x_k) + L_e(\hat{d}_k) \tag{50}$$

where $\pi_k = \pi_k^1/\pi_k^0$ is the input a priori probability ratio and π_k^e is the output extrinsic likelihood, which is the correction term.

- Eqn 50 gives the a priori LLR, the channel measurement LLR, and the extrinsic LLR.
- The MAP algorithm can be implemented in terms of a likelihood ratio using eqns 48 and 49.

Thank You

References

B. Sklar and P. K. Ray., *Digital communications: fundamentals and applications*, 2nd ed. Prentice Halls International editions, 2001.

S. Lin and D. J. C. Jr., *Error Control Coding*, 2nd ed. Pearson / Prentice Hall, 2004.

R. Blahut, *Theory and Practice of Error Control Codes*, 2nd ed. Addison Wesley, 1984.

Forouzan and A. Behrouz, *Data Communications and Networking*, 4th ed. Tata-McGraw-Hill, 2007.

J. G. Proakis, *Digital communications*, 4th ed. Prentice Hall, 2001.

J. G. Proakis and M. Salehi, *Communication Systems Engineering*, 2nd ed. Prentice Hall, 2002.