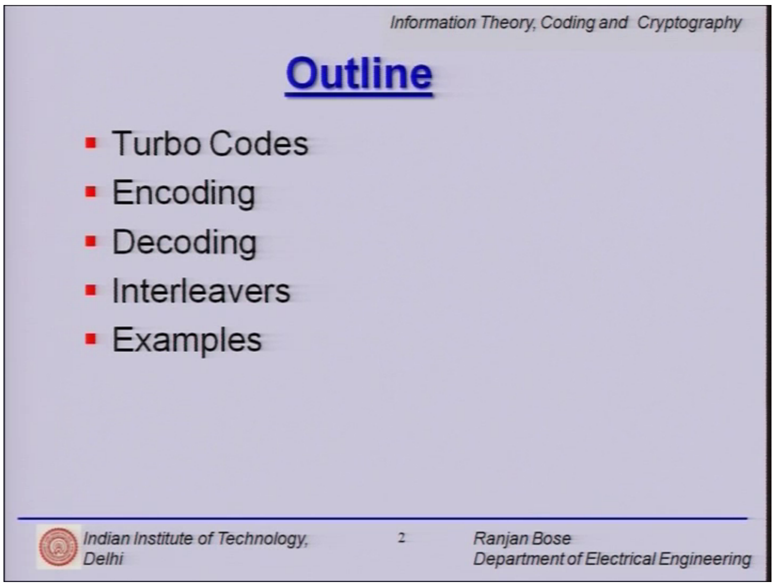
Information Theory, Coding and Cryptography

Dr. Ranjan Bose
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Module - 29
Turbo Codes
Lecture - 29

Hello and welcome to this lecture on Turbo Codes. Let us start with the outline of today's talk.

(Refer Slide Time: 00:34)



Information Theory, Coding and Cryptography

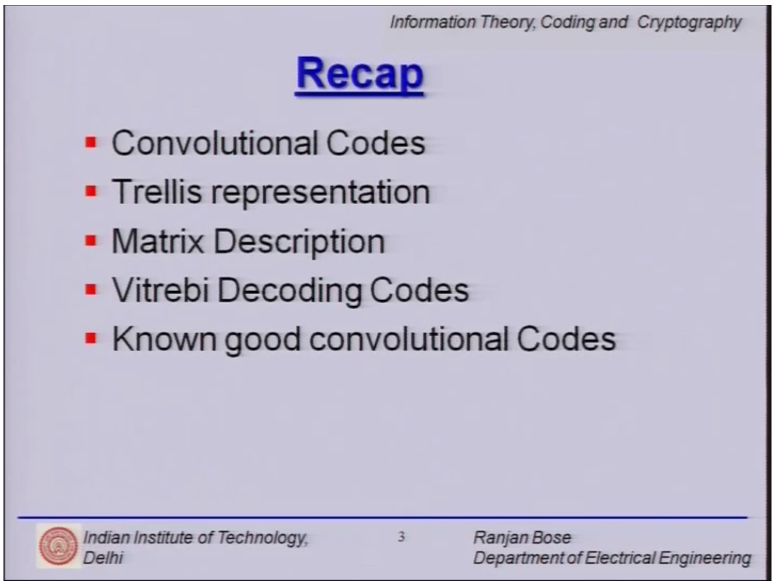
Outline

- Turbo Codes
- Encoding
- Decoding
- Interleavers
- Examples

Indian Institute of Technology, Delhi   2   Ranjan Bose, Department of Electrical Engineering

We would have a quick introduction to turbo codes followed by the encoding process. Then what makes turbo code very interesting is the decoding process from which it also derives its name. Then we look at a very interesting component of these turbo codes called interleavers. And finally, we will look at some examples, so that is the outline for today's talk.

(Refer Slide Time: 00:57)


Information Theory, Coding and Cryptography

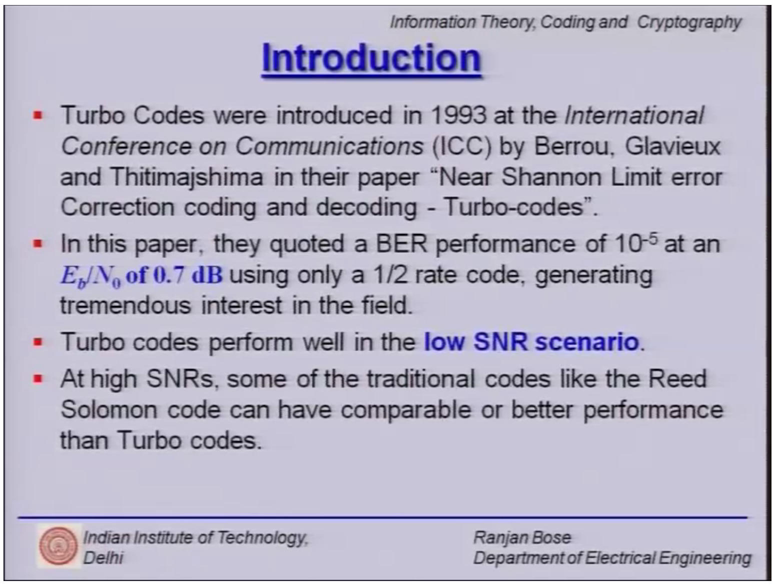
Recap

- Convolutional Codes
- Trellis representation
- Matrix Description
- Vitrebi Decoding Codes
- Known good convolutional Codes

Indian Institute of Technology, Delhi 3 Ranjan Bose Department of Electrical Engineering

What we have done so far we can have a quick recap. We were looking at convolutional codes and this is the logical end to the topic of convolutional codes, where we looked at the trellis representation, the matrix description, and finally, the popular Vitrebi decoding algorithm for convolutional codes. We also looked at some of the known good convolutional codes. And now we change gears and go onto study the next version which is the turbo codes.

(Refer Slide Time: 01:28)


Information Theory, Coding and Cryptography

Introduction

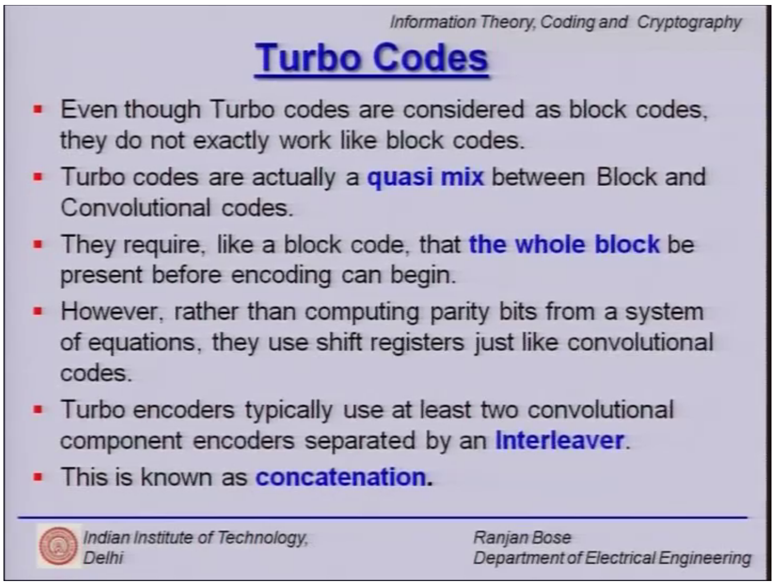
- Turbo Codes were introduced in 1993 at the *International Conference on Communications* (ICC) by Berrou, Glavieux and Thitimajshima in their paper "Near Shannon Limit error Correction coding and decoding - Turbo-codes".
- In this paper, they quoted a BER performance of 10^{-5} at an **E_b/N_0 of 0.7 dB** using only a 1/2 rate code, generating tremendous interest in the field.
- Turbo codes perform well in the **low SNR scenario**.
- At high SNRs, some of the traditional codes like the Reed Solomon code can have comparable or better performance than Turbo codes.

Indian Institute of Technology, Delhi Ranjan Bose Department of Electrical Engineering

So, let us have a quick introduction. So, turbo codes are not very old, but not very new either. They were first proposed in 1993 by 3 inventors. And the title of the paper was Near Shannon Limit error correction coding and decoding. So, as the title suggests that these were supposed to work close to the Shannon's limit, and that is why it generated a lot of curiosity. And they also showed a bit error rate performance of 10 raised to power minus 5 at very low E b over N naught of 0.7 dB, and they used only 1 by 2 rate code. And this was the starting point for turbo codes.

So, they perform well in low SNR scenarios, and that is where we see them touching the Shannon's limit. But, at the other end of the spectrum, if you look at high SNR scenarios, the traditional codes such as Reed Solomon codes that we have studied can outperform turbo codes. So, turbo codes are supposed to be used, when we are working in low SNR scenarios, which we cannot have enough of we do have cases, and we would like to preserve battery life. So, we can always lower the transmit power, and come into the regime, where turbo codes become effective.

(Refer Slide Time: 03:00)


Information Theory, Coding and Cryptography

Turbo Codes

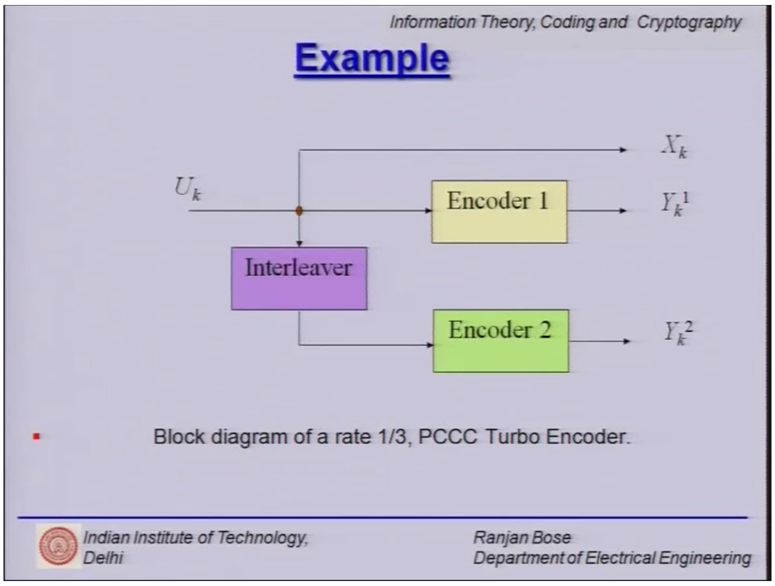
- Even though Turbo codes are considered as block codes, they do not exactly work like block codes.
- Turbo codes are actually a **quasi mix** between Block and Convolutional codes.
- They require, like a block code, that **the whole block** be present before encoding can begin.
- However, rather than computing parity bits from a system of equations, they use shift registers just like convolutional codes.
- Turbo encoders typically use at least two convolutional component encoders separated by an **Interleaver**.
- This is known as **concatenation.**

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, you will shortly see that turbo codes behave like block codes to some extent, but they are kind of an in between convolutional codes and block codes. And it is kind of a quasi mix so to say,. They require the whole block to be received before, decoding can start to happen. But, rather than computing the parity bits from a system of equations, they use shift registers just like convolutional codes. In fact, turbo codes are a combination of two or more convolutional code separated by an interleaver as we will shortly see. So, they typically use two or more convolutional component encoders, which are separated by an interleaver. This is called concatenation. So, we can have a series, concatenation or a parallel concatenation.

(Refer Slide Time: 04:00)


Information Theory, Coding and Cryptography
U_k
X_k
Encoder 1
Y_k^1
Interleaver
Encoder 2
Y_k^2
Indian Institute of Technology, Delhi
Ranjan Bose
Department of Electrical Engineering

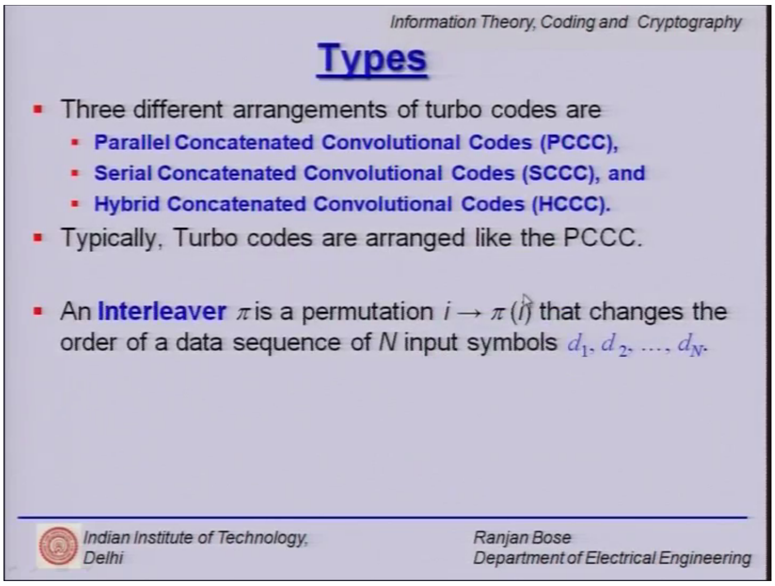
Example

- Block diagram of a rate 1/3, PCCC Turbo Encoder.

So, this is the basic structure of turbo encoder. As you can see, this is the input. And this input can go to the first convolutional encoder called encoder 1, and it gets some give some output. But, the same input goes to an interleaver which is basically shuffling operation, and is fed to another convolutional encoder. So, both encoder 1, and encoded 2 are convolutional encoder, very very hardware friendly. And we have output coming from encoder 2.

These three, so it is some kind of a systematic code, you can say that the input has two parity symbols coming out from encoder 1 and encoded 2, and this together forms the output of this turbo encoder. Whereas, you can see that these two encoders, the convolution encoders can be in several configurations.

(Refer Slide Time: 05:08)



Information Theory, Coding and Cryptography

Types

- Three different arrangements of turbo codes are
  - **Parallel Concatenated Convolutional Codes (PCCC),**
  - **Serial Concatenated Convolutional Codes (SCCC), and**
  - **Hybrid Concatenated Convolutional Codes (HCCC).**
- Typically, Turbo codes are arranged like the PCCC.
- An **Interleaver** π is a permutation $i \rightarrow \pi(i)$ that changes the order of a data sequence of N input symbols $d_1, d_2, \ldots, d_N$.

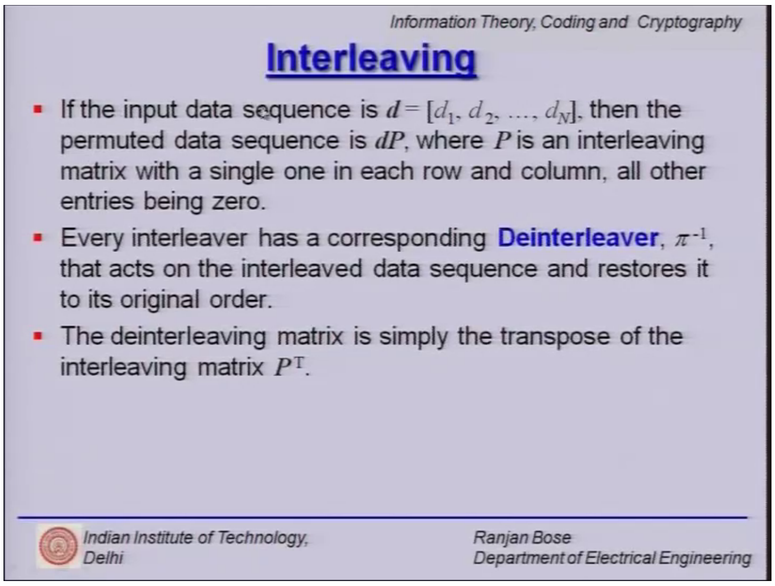
Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, what are the obvious configurations, we can have the parallel concatenated conversion encoder, which we just now see. We have seen that this encoder 1, encoder 2 are in parallel, but nothing stops them from putting an encoder 1 then an interleaver in series, and then an encoder 2. So, you can have a serial concatenated convolutional code, so SCCC as opposed to PCCC. And you can always have a hybrid a combination of a serial and parallel, you know giving us the name hybrid concatenated convolutional encoder or HCCC.

But in most of the cases, we will be working with PCCC ok. And what is it interleaver, it is denoted by pi, it is nothing but a permutation i goes to pi i, which changes an order of the data sequence of N input symbols d 1, d 2, up to d N. So, it is just a permutation that happens, because of the interleaver. Earlier interleavers were used to ensure that block codes can work, where we have a run of errors, burst errors that happen. So, interleaver were used to separate out the burst errors into random errors, and then the block codes could work. Here, the interleaver is being used for a very different purpose as we will shortly point out.

(Refer Slide Time: 06:38)



Information Theory, Coding and Cryptography

Interleaving

- If the input data sequence is $\boldsymbol{d} = [d_1, d_2, \ldots, d_N]$, then the permuted data sequence is $\boldsymbol{d}P$, where P is an interleaving matrix with a single one in each row and column, all other entries being zero.
- Every interleaver has a corresponding **Deinterleaver**, π^{-1}, that acts on the interleaved data sequence and restores it to its original order.
- The deinterleaving matrix is simply the transpose of the interleaving matrix P^{T}.

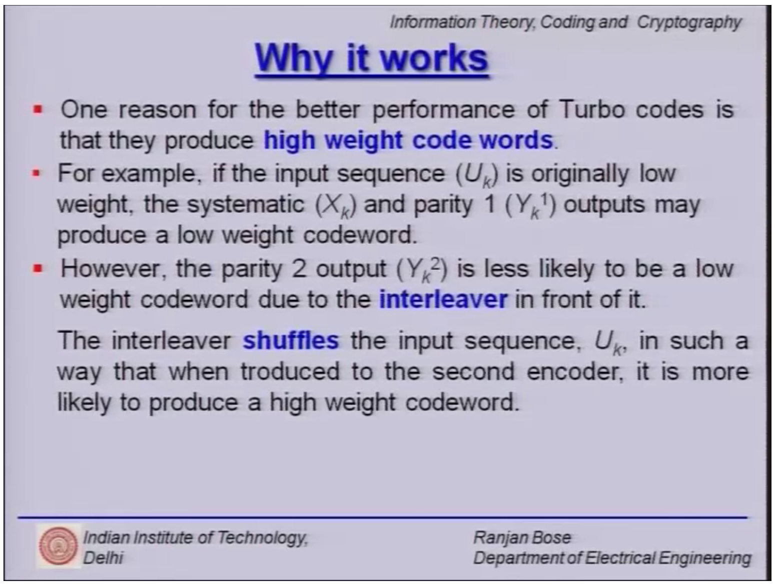
Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, the if the input data sequence d is d 1, d 2 up to d N, then the permutated data sequence is d P, where P is the interleaving matrix, and it has a single one in each row and column on other entries are zero. So, the sparse matrix, and it is very simply an interleaving matrix multiplied by this vector gives you an interleaved vector.

Obviously, I can have a deinterleaver, which undoes the operation of the interleaver, and so we denoted by pi inverse, and it restores back the interleaved sequence back to its original sequence. So, the interleaving deinterleaving matrix is simply the transpose of the interleaving matrix P transpose. So, using algebra, we can easily show the process of interleaving and deinterleaving.

(Refer Slide Time: 07:40)


Information Theory, Coding and Cryptography

Why it works

- One reason for the better performance of Turbo codes is that they produce **high weight code words**.
- For example, if the input sequence (U_k) is originally low weight, the systematic (X_k) and parity 1 (Y_k^1) outputs may produce a low weight codeword.
- However, the parity 2 output (Y_k^2) is less likely to be a low weight codeword due to the **interleaver** in front of it.

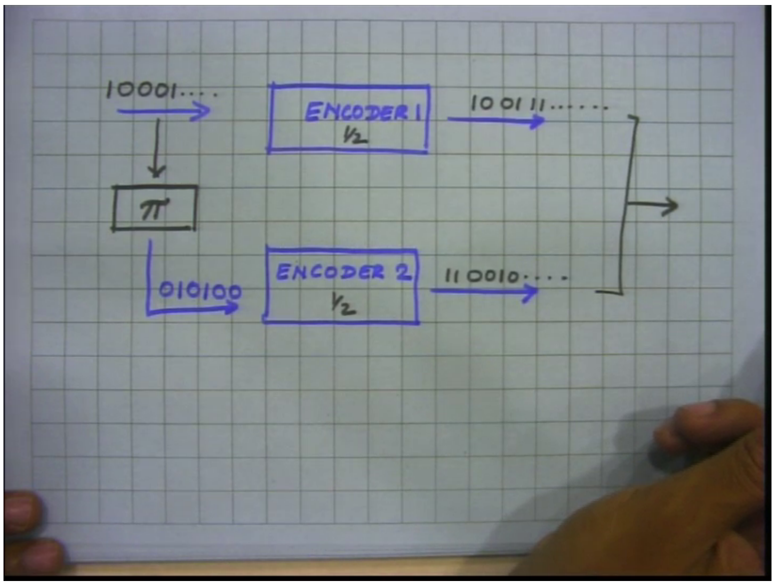
The interleaver **shuffles** the input sequence, U_k, in such a way that when troduced to the second encoder, it is more likely to produce a high weight codeword.

Indian Institute of Technology, Delhi
Ranjan Bose
Department of Electrical Engineering

So, we need to ask what is it that the interleaver is doing. Well, the interleaver as the name suggests shuffles the input bits, in such a way that it the input to the second interleaver, the second encoder is different from the first one. So, let us see why with this will work. This will result in high weight code words. If you remember, the simple convolutional encoder was a linear encoder, so the weight of the code, kind of has a direct correspondence with the d free.

So, we are concerned with high weight code words leading to better error correcting capability. Now, you must know that d free corresponds to the minimum free distance, which means that we are looking at the worst case scenario. It does not mean that all sequences have poor weight characteristics, some of them are good in terms of the distance properties, and hence a higher weight and some of them are poor. What the turbo code does is it mixes and matches to ensure that you do not end up with a poor weight all the times say. So, the interleaver 2, which has the parity, which requires an interleaver for the second encoder will result in a different kind of a output bit stream.

(Refer Slide Time: 09:22)



So, let us look at it in a slightly more detailed manner. So, as you know, you have a convolutional encoder, we call it encoder 1, and it should give a sequence. Suppose, an input sequence was coming in, and this encoder it could be rate 1 by 2. So, for every bit it gives you 2 bits of input.

Now, if we put the same encoder here, but we label is at encoder 2, and it is again rate 1 by 2. If we had given the same input here, I would get the same output, but we do not want that. So, we put it through an interleaver. And what it does is it gives to this encoder something, which is permit it. Now, clearly the output of the second encoder will be different from what was received earlier.

Student: Is the (Refer Time: 10:54) random.

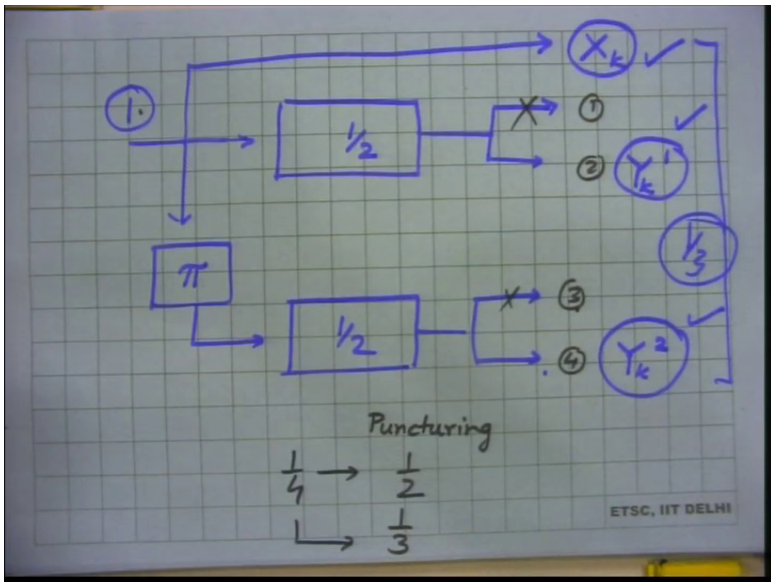
Question being asked is the interleaver random, random interleaver is one type of several possible interleaver. And at the end of today's class, we will talk about the different kinds of interleaver. But, random interleavers also work very well. So, right now, let us say it is just a random interleaver. So, there is no constrain it randomly permutes this combination. But, what is resulted in this one. The output, if this was rate 1 by 2 for every 1 bit that comes in 2 bits go out ok.

Similarly, 1 bit comes in, 2 bits go out. So, the output here is a path in the trellis, because this is a convolutional encoder. Now, this path in the trellis, will have some weight. Now, clearly this encoder, with a different input has another path in the trellis, because the input bit stream decides, which path in the trellis you give. And therefore, the output is a different bit stream.

Now, if one of the paths corresponds to the d free, it is a poor weight. The d free is resulting from the 1, which has low weight. Then obviously, this being an random interleaver, this output sequence will be different from the first one. So, if this is low or lowest possible weight, this will not be infinite, this could be very most like a high weight. So, if we can combine these 2 on an average, we will come out of winner. So, a low weight path gets combined with a high weight path, and together they balance out, and we do not kind of have a path, which has low weight, because effectively, the worst case analysis leads to the d free ok. But there are many paths, which are high weight.

So, a chance of them combining leading to a path with a good weight is very high, and that is why a random interleaver just works.

(Refer Slide Time: 13:15)

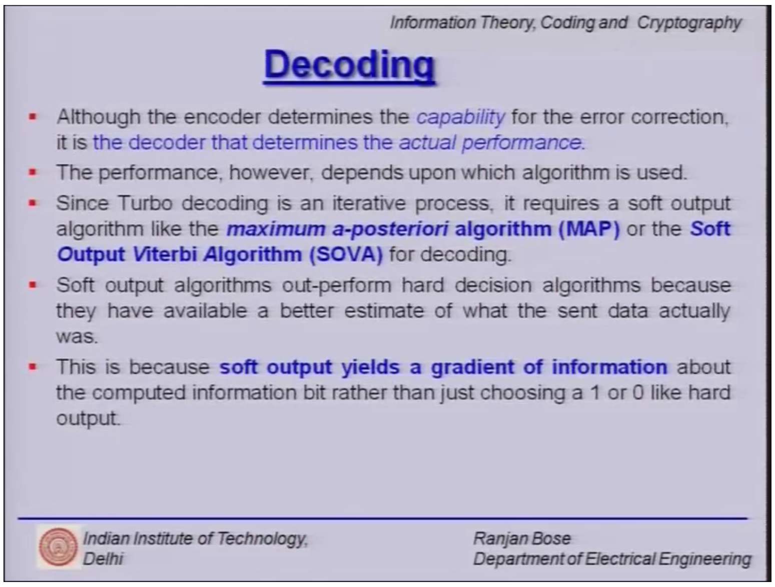


So, if you look at this a little bit more carefully, you have now an input to an encoder, which is supposed rate 1 by 2, so you will have an output, which has 2 bits out. But, this goes through an interleaver, and goes through another encoder again read by 2. Again it has 2 bits coming out of here.

So, you have 1 bit going in, and you have 1, 2, 3, 4 bits. And it is not a surprise, because you have rate 1 by 2, and rate 1 by 2 leading to an effective rate 1 by 4. Now, this is pretty expensive in terms of the extra bits we are padding, because we are resulting in rate 1 by 4. So, it is possible to so to say, drop 1 bit at a random, which is called puncturing.

So, we can possibly drop 1 of them. And if really this is this guy is actually a systematic encoder, then this first bit is identical to this one, but not this one, because this comes from an interleaved version. So, you can drop any one of them. So, I can lead this rate 1 by 4 to back to 1 by 2, if I cancel out both of them, or I can take it to 1 by 3, if I just cancel just drop one of them. This is the puncturing process. So, I am introducing errors by myself, but what the strong encoder, decoder can do is to recover from this error, self introduced error.

(Refer Slide Time: 15:26)



Information Theory, Coding and Cryptography

Decoding

- Although the encoder determines the *capability* for the error correction, it is the decoder that determines the *actual performance*.
- The performance, however, depends upon which algorithm is used.
- Since Turbo decoding is an iterative process, it requires a soft output algorithm like the ***maximum a-posteriori* algorithm (MAP)** or the **Soft Output Viterbi Algorithm (SOVA)** for decoding.
- Soft output algorithms out-perform hard decision algorithms because they have available a better estimate of what the sent data actually was.
- This is because **soft output yields a gradient of information** about the computed information bit rather than just choosing a 1 or 0 like hard output.

Indian Institute of Technology, Delhi

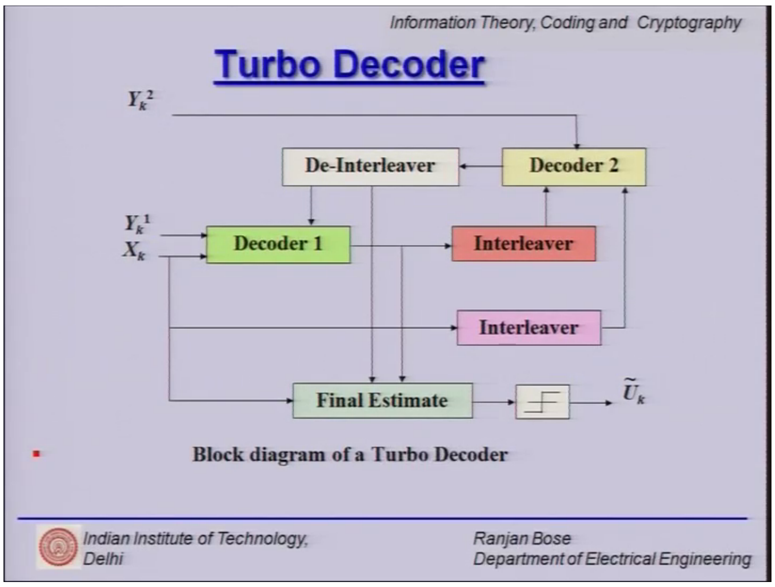
Ranjan Bose
Department of Electrical Engineering

So, we come back to the slides. And we now focus our attention on the decoding part, because we have just now understood that encoding is very easy it is fairly hardware friendly. So, the encoder determines the capability for error correction, it is a decoder that determines the performance, there is a difference. There is a difference between the capability and actual performance.

So, you can have several kinds of decoders, and we will discuss two of them briefly at the end of this lecture. So, there is a decoder, which will decide the performance. So, the performance depends on the algorithm used. And turbo decoding is an iterative decoding process, please note this is a basic difference between decoding a Viterbi based convolutional code, and this turbo codes. Turbo codes require an iterative decoding process for example, the maximum a-posteriori algorithm MAP or the soft output Viterbi algorithm SOVA for decoding.

So, soft output algorithms, out-perform hard decision decoding algorithms, because they have available a better estimate of what the data sent was actually. And soft output yields a gradient of information as opposed to hard decision decoding, which is like choosing 1 or 0 right in the beginning, leading to the loss of information.

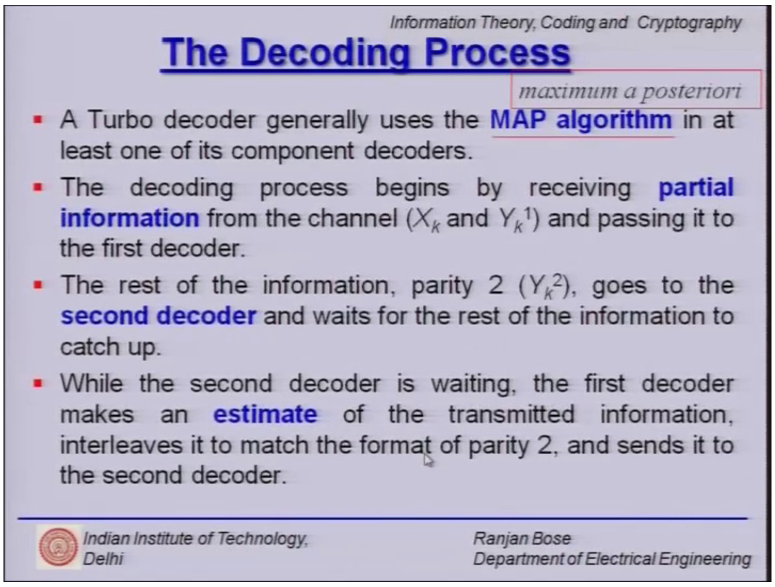
(Refer Slide Time: 17:00)


Information Theory, Coding and Cryptography
Turbo Decoder
Y_k^2
De-Interleaver
Decoder 2
Y_k^1
X_k
Decoder 1
Interleaver
Interleaver
Final Estimate
$\tilde{U}_k$
Block diagram of a Turbo Decoder
Indian Institute of Technology, Delhi
Ranjan Bose
Department of Electrical Engineering

So, let us look at the structure of the turbo decoder all right. So, here if you see, we have the three output bits coming into the decoder. And if you look at the various blocks within the decoder, you have a decoder 1, you have a decoded 2, even interleaver, and also a de-interleaver, and then again there is an interleaver.

So, first we have to understand how it works in a very basic manner. The job of the decoder is to guess what would have been the real thing, and mimic the encoding process, so you have an interleaver itself. So, so encoder, interleaver, encoder, and it compares it with the received vector, finds the difference, adjust the estimate. And then again does encoding process, compares with the received vector, compares the difference. And then does it several times rate converges or a threshold is met, and then you get the final estimate, and declare the result. So, it has to be an iterative guessing process.

(Refer Slide Time: 18:24)



Information Theory, Coding and Cryptography

The Decoding Process

maximum a posteriori

- A Turbo decoder generally uses the **MAP algorithm** in at least one of its component decoders.
- The decoding process begins by receiving **partial information** from the channel (X_k and Y_k^1) and passing it to the first decoder.
- The rest of the information, parity 2 (Y_k^2), goes to the **second decoder** and waits for the rest of the information to catch up.
- While the second decoder is waiting, the first decoder makes an **estimate** of the transmitted information, interleaves it to match the format of parity 2, and sends it to the second decoder.

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, let us talk quickly about the decoding process. A turbo decoder generally uses an MAP algorithm in at least 1 of his component decoders. What is MAP? It is a maximum a-posterior algorithm. The decoding process begins by receiving the partial information from the channel, and passing it to the first decoder, as we saw. The rest of the information, which is the parity bit 2 goes to the second decoder.

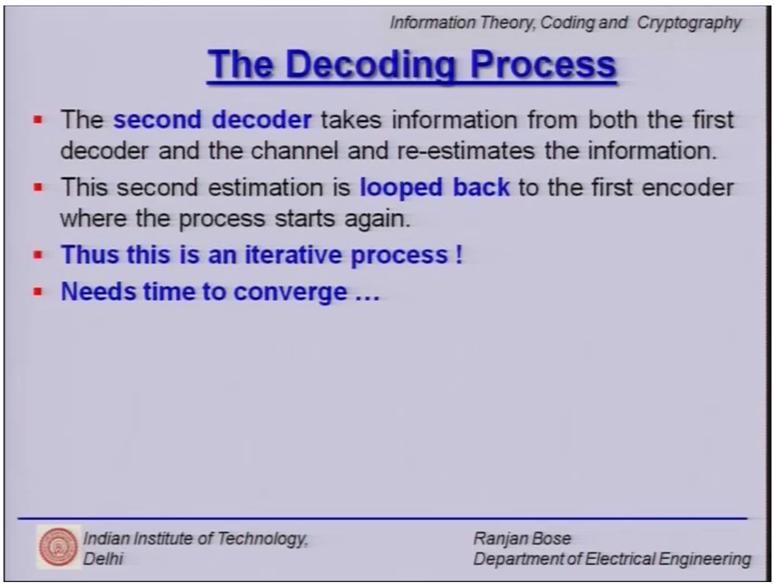
So, if you recall, what we have done is we have got. If you go back to the drawing board, we have the original bit coming in, so it is a systematic. And then I discard one of them, and I get Y k 1. And I discard this 1, and I get Y k 2. So, yes I had a rate 1 by 2, I threw out 1, I am confident I will be able to recover from this self imposed error, I discarded 1 here, I got back here. So, I have on this side, I rate 1 by 3. So, I am going to work with these 3 bits at the decoder.

Now, please note, X k, Y k 1, will be fed, this called the partial information, this will be fed to decoder 1. And this really corresponds to encoder 2, so it will be fed to decoded 2. And we will have a message information passing algorithm, which should they will share the information together, try to guess what could have been sent, so as to give me X k, Y k 1, and Y k 2 that is the strategy.

So, we go back to our original slide, and we see that the decoding process begins by receiving the partial information X k and Y k 1 and passing to the first decoder within the turbo decoder. And the rest of the information parity 2, which is Y k 2 goes to the second decoder, and waits for the rest of the information to catch up.

While the second decoder is waiting, the first decoder makes an estimate of the transmitted information, because it knows what the encoder was encoder 1 was, and it makes an estimates right. And then interleaves it to match, the format for parity 2, because the decoder 2 can work with only an interleaved data. So, it makes an estimate, hoping it is right, does the interleaving, and gives it to decoded 2, to generate its output.

(Refer Slide Time: 21:20)



Information Theory, Coding and Cryptography

The Decoding Process

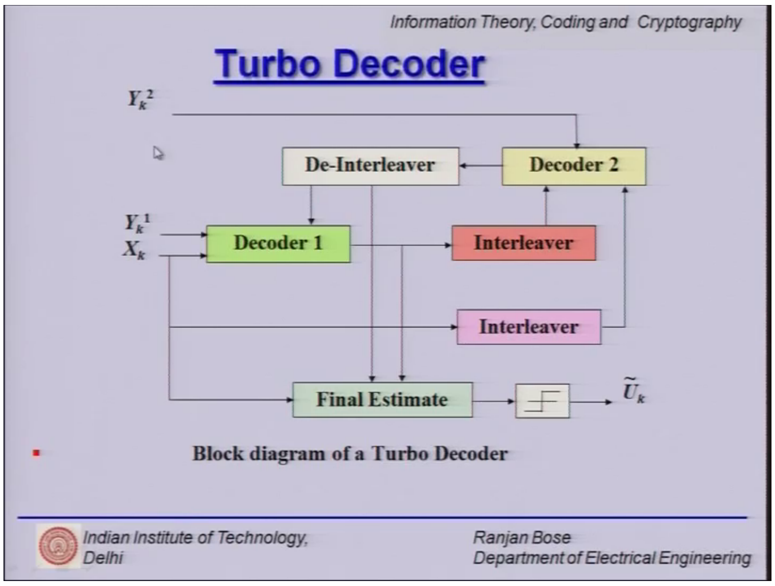
- The **second decoder** takes information from both the first decoder and the channel and re-estimates the information.
- This second estimation is **looped back** to the first encoder where the process starts again.
- **Thus this is an iterative process !**
- **Needs time to converge ...**

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

Now, the decoder 2, the decoder 2, the second decoder takes the information both from the first decoder, and the channel, and re-estimates the information. This second estimation is looped back to the first encoder, where the process starts again, so it is message passing. Therefore, it is clearly an iterative process, and we keep doing it, till we converge. And we know that ok, this is the best bet I had in terms of the receive message. So, clearly it needs some time to converge.

(Refer Slide Time: 21:55)


Information Theory, Coding and Cryptography

Turbo Decoder

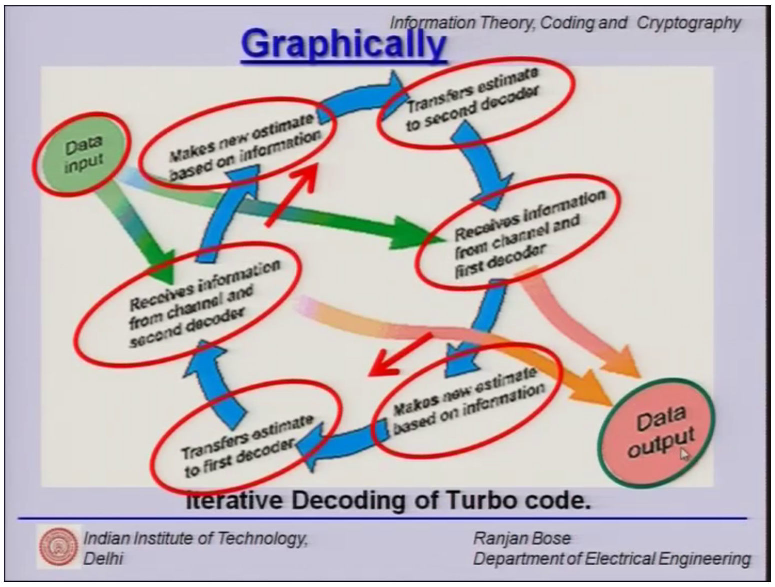
Block diagram of a Turbo Decoder

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, this is what we had sent, this Y k goes to Y k 2 goes to decoder 2, Y k 1 and X k 1 X k goes to decoder 1, it gives an estimate, so it interleaves, and passes on to decoder 2; so that it can mimic, the encoder 2, and then it together does a deal interleaving and passes the message. So, it goes on and on.

(Refer Slide Time: 22:22)


Information Theory, Coding and Cryptography

Graphically

Data input
Makes new estimate based on information
Transfers estimate to second decoder
Receives information from channel and first decoder
Receives information from channel and second decoder
Makes new estimate based on information
Transfers estimate to first decoder
Data output

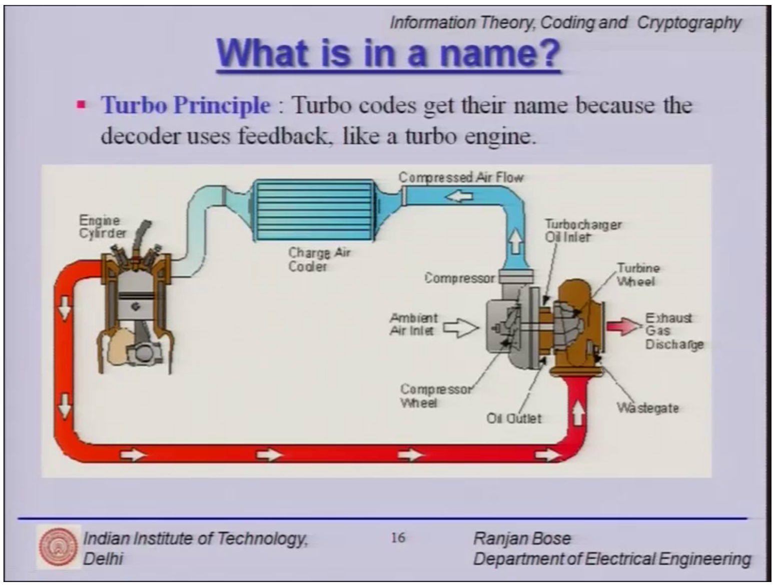
Iterative Decoding of Turbo code.

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, graphically let us see what is happening. Let us start with the input data. And input data is sent from the channel to the first decoder, we have seen that, which makes an estimate based on the information, and then it transfer the estimates to the first decoder, receive the information for the channel to the second decoder, makes a new estimate based on the information, and transfer the estimate to the second decoder. The reason this is exactly what we are discussed, why we have put it in this way is to show an iterative decoding process ok. And finally, when we are happy or in the convergence happens, we show the output. And this loop must keep on going, till we are ready to declare the results.

(Refer Slide Time: 23:15)



Information Theory, Coding and Cryptography

What is in a name?

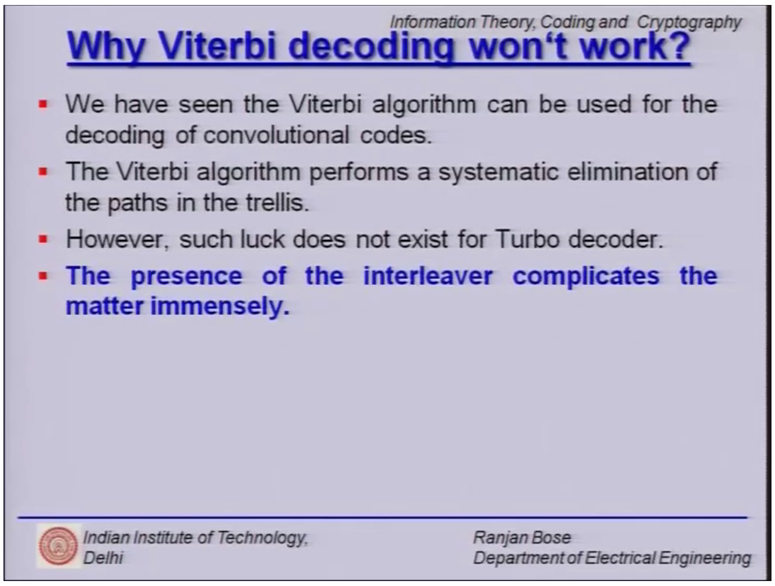
- **Turbo Principle** : Turbo codes get their name because the decoder uses feedback, like a turbo engine.

Compressed Air Flow
Engine Cylinder
Charge Air Cooler
Turbocharger Oil Inlet
Turbine Wheel
Compressor
Ambient Air Inlet
Exhaust Gas Discharge
Compressor Wheel
Oil Outlet
Wastegate

Indian Institute of Technology, Delhi | 16 | Ranjan Bose, Department of Electrical Engineering

So, why is it called a turbo decoder, well if you see the turbo engine, it also has an engine here, and exhaust gas discharge here, and it somehow has a loop like this, which works. And therefore, this is called the turbo decoder. This is how the name has come why is it called a turbo decoder. It has a, or it is some similarity with a little bit of imagination with this turbo engine.

(Refer Slide Time: 23:50)



Information Theory, Coding and Cryptography

Why Viterbi decoding won't work?

- We have seen the Viterbi algorithm can be used for the decoding of convolutional codes.
- The Viterbi algorithm performs a systematic elimination of the paths in the trellis.
- However, such luck does not exist for Turbo decoder.
- **The presence of the interleaver complicates the matter immensely.**

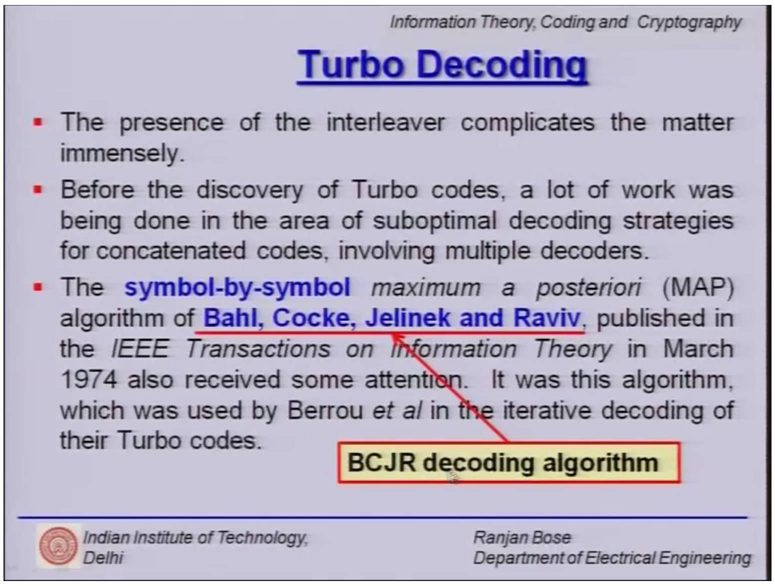
Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

Question now is we had this very nice Viterbi decoding algorithm for conversion coder. Here we have two conversion encoders, why cannot we use Viterbi to decode. Well, the problem is that we do not have the luxury of a single trellis anymore. Why, because even though the first encoder uses a trellis diagram, this input to the second one is based on a permitted sequence, and therefore, it corresponds to some other trellis.

So, the Viterbi algorithm works by systematic elimination of paths in the trellis, but the question is which trellis ok. But, we do not have this luck for turbo decoder and therefore, that iterative decoding must work. So, we must appreciate that Viterbi can work, if you specify a deterministic trellis, not that a trellis that keeps changing. So, this presence of this interleaver has really complicated this decoding process.

(Refer Slide Time: 25:00)


Information Theory, Coding and Cryptography

Turbo Decoding

- The presence of the interleaver complicates the matter immensely.
- Before the discovery of Turbo codes, a lot of work was being done in the area of suboptimal decoding strategies for concatenated codes, involving multiple decoders.
- The **symbol-by-symbol** *maximum a posteriori* (MAP) algorithm of **Bahl, Cocke, Jelinek and Raviv**, published in the *IEEE Transactions on Information Theory* in March 1974 also received some attention. It was this algorithm, which was used by Berrou *et al* in the iterative decoding of their Turbo codes.

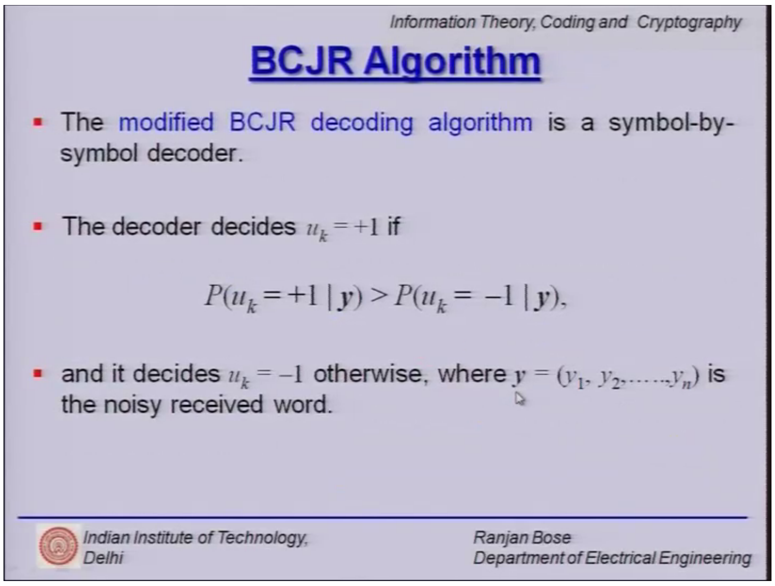
BCJR decoding algorithm

Indian Institute of Technology, Delhi
Ranjan Bose
Department of Electrical Engineering

So, let us just now spend some time looking at 2 popular decoding algorithms. So, before the discovery of turbo codes, a lot of work was done in the area of suboptimal decoding strategies for concatenated codes. So, anyway the work was going on.

Now, the symbol-by-symbol maximum a posteriori MAP algorithm by Bahl, Cocke, Jelinek and Raviv, which is called the BCJR algorithm, was published in 1974. But, it was there on the library shelves not much was being done, till in 1993, it was reused by Berrou et al to do turbo decoding. So, what we will discuss today is this BCJR, actually a modified BCJR decoding algorithm for turbo decoding.

(Refer Slide Time: 26:00)



Information Theory, Coding and Cryptography

BCJR Algorithm

- The modified BCJR decoding algorithm is a symbol-by-symbol decoder.
- The decoder decides $u_k = +1$ if

$$P(u_k = +1 \mid \boldsymbol{y}) > P(u_k = -1 \mid \boldsymbol{y}),$$

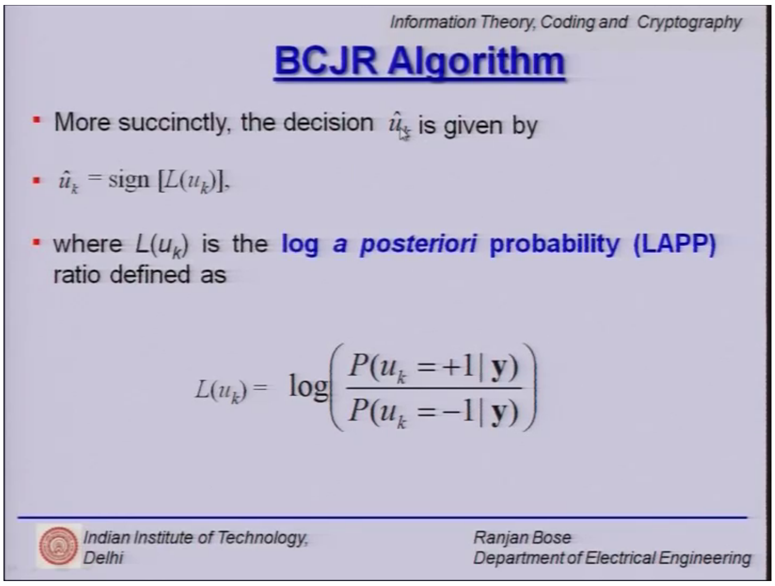
- and it decides $u_k = -1$ otherwise, where $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ is the noisy received word.

Indian Institute of Technology, Delhi
Ranjan Bose
Department of Electrical Engineering

So, what is this algorithm? It is just in the next few slides will put into mathematics, the intuitive understanding that we have developed for this turbo decoding that information passing between encoder 1 and encoder 2. So, first of all, this modified BCJR decoding algorithm is a symbol-by-symbol decoder.

So, the decoder decides that U k is plus 1. So, let us talk about binary, so plus 1 and minus 1 instead of 1 and 0. So, the decoder decides U k is plus 1, if probability of U k is plus 1 given Y the received vector greater than probability the U k is equal to minus 1 given the received vector. So, this Y volt phase is Y 1, Y 2, Y n is the noisy received word it could have errors in it.

(Refer Slide Time: 27:00)



Information Theory, Coding and Cryptography

BCJR Algorithm

- More succinctly, the decision $\hat{u}_k$ is given by
- $\hat{u}_k = \text{sign}\,[L(u_k)]$,
- where $L(u_k)$ is the **log *a posteriori* probability (LAPP)** ratio defined as

$$L(u_k) = \log\left(\frac{P(u_k = +1 \mid \mathbf{y})}{P(u_k = -1 \mid \mathbf{y})}\right)$$

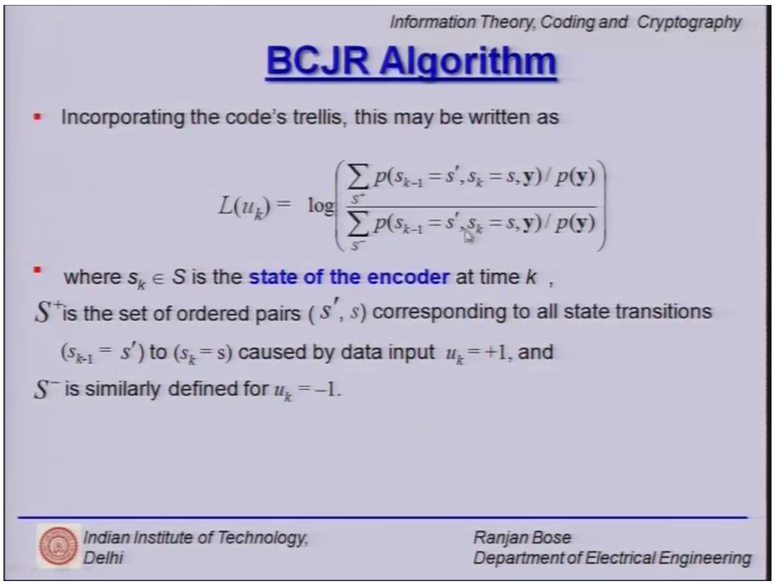
Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, what we do is a decision U k hat, is the sign of L u k. What is L u k, is a log a posterior probability. So, LAPP, we will use this as a way, which is a log likelihood ratio, which we have just now seen the probability of U k is equal to 1 given the noisy stream Y versus U k is equal to minus 1 giving the received vector. And if it is now a question of a sign, if the numerator is larger, the sign is positive, the numerator is less, probability is less; the sign is negative.

So, we are now only working with the sign ok. So, it is a very smart way to convert this log a posteriori probability ratio. So, we are now going to work with this ratio and the signs itself. So, it is pretty decent to do this. So, now we just go into the trellis, and for the encoder 1, and encoder 2. And we look at these probabilities. So, we are trying to find out given this vector Y, what is the probability that U k is 1.

(Refer Slide Time: 28:19)



Information Theory, Coding and Cryptography

BCJR Algorithm

- Incorporating the code's trellis, this may be written as

$$L(u_k) = \log\left(\frac{\sum_{S^+} p(s_{k-1}=s', s_k=s, \mathbf{y})/p(\mathbf{y})}{\sum_{S^-} p(s_{k-1}=s', s_k=s, \mathbf{y})/p(\mathbf{y})}\right)$$

- where $s_k \in S$ is the **state of the encoder** at time k,

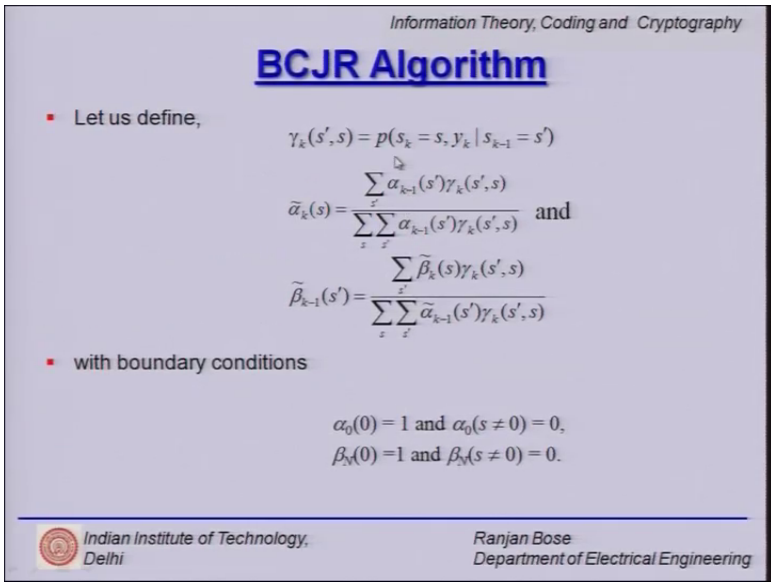
S^+ is the set of ordered pairs (s', s) corresponding to all state transitions ($s_{k-1} = s'$) to ($s_k = s$) caused by data input $u_k = +1$, and

S^- is similarly defined for $u_k = -1$.

Indian Institute of Technology, Delhi — Ranjan Bose, Department of Electrical Engineering

And similarly, we define these two ratios, and in this in terms of the state of the encoder right.

(Refer Slide Time: 28:30)



Information Theory, Coding and Cryptography

BCJR Algorithm

- Let us define,

$$\gamma_k(s',s) = p(s_k = s, y_k \mid s_{k-1} = s')$$

$$\tilde{\alpha}_k(s) = \frac{\sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)}{\sum_s \sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)} \text{ and}$$

$$\tilde{\beta}_{k-1}(s') = \frac{\sum_{s'} \tilde{\beta}_k(s)\gamma_k(s',s)}{\sum_s \sum_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s',s)}$$

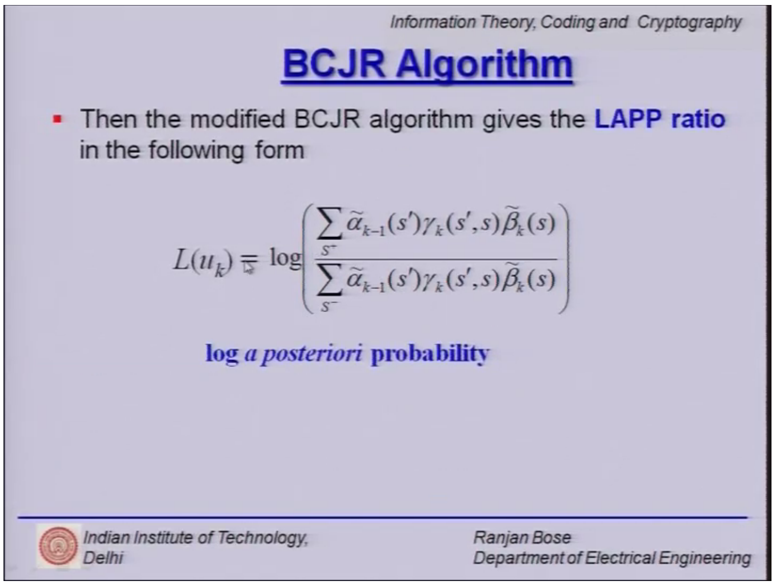
- with boundary conditions

$$\alpha_0(0) = 1 \text{ and } \alpha_0(s \neq 0) = 0,$$
$$\beta_N(0) = 1 \text{ and } \beta_N(s \neq 0) = 0.$$

Indian Institute of Technology, Delhi — Ranjan Bose, Department of Electrical Engineering

So, we can do a little bit of maths, and we have these forward and backward gains, which are intermediatory terms as alpha, tilde, and beta, tilde with some boundary conditions. So, we can go into the details of this BCJR algorithm.

(Refer Slide Time: 28:48)



Information Theory, Coding and Cryptography

BCJR Algorithm

- Then the modified BCJR algorithm gives the LAPP ratio in the following form

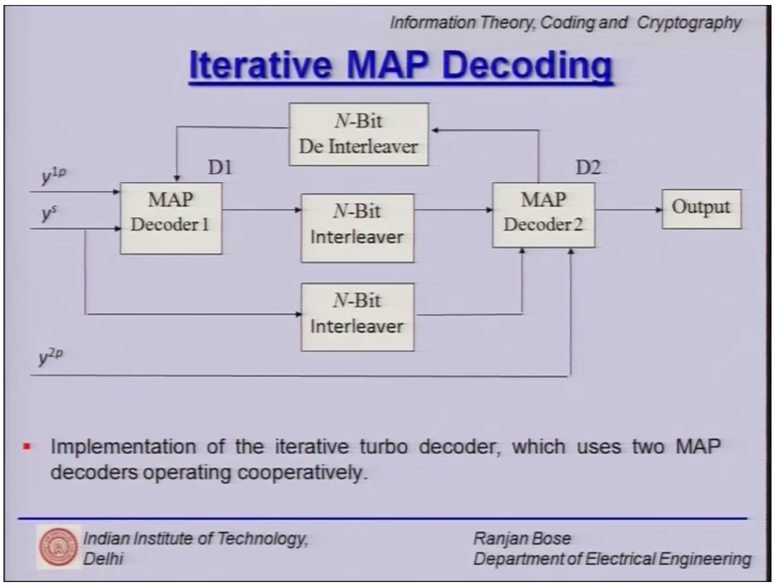
$$L(u_k) \triangleq \log\left(\frac{\sum_{S^+} \tilde{\alpha}_{k-1}(s')\gamma_k(s',s)\tilde{\beta}_k(s)}{\sum_{S^-} \tilde{\alpha}_{k-1}(s')\gamma_k(s',s)\tilde{\beta}_k(s)}\right)$$

log *a posteriori* probability

Indian Institute of Technology, Delhi   Ranjan Bose Department of Electrical Engineering

Basically, to define exactly what is the log a posterior probability in terms of these alphas and betas. So, this is the LAPP ratio.

(Refer Slide Time: 29:00)



Information Theory, Coding and Cryptography

Iterative MAP Decoding

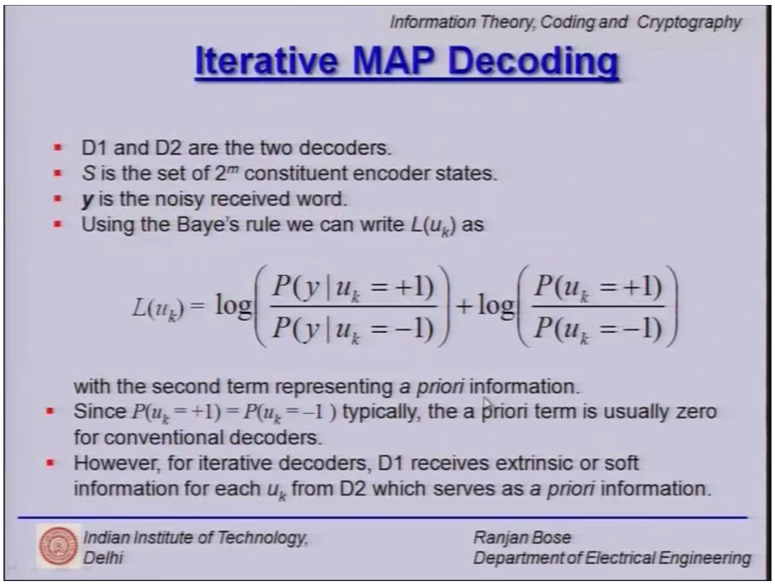
- Implementation of the iterative turbo decoder, which uses two MAP decoders operating cooperatively.

Indian Institute of Technology, Delhi   Ranjan Bose Department of Electrical Engineering

So, based on the sign, then the decoding can happen right, that is how you do the estimate. Now, we look at the second decoding algorithm, I will again give you a hint as to how it is done, this is called the iterative MAP decoding, is different from the BCJR algorithm that we studied before.

And here, let us look at the methodology. So, you have these three inputs that are coming in. And we have now, the MAP decoder 1, then we have an interleaver giving in to MAP decoder 2. So, you can estimate this capital D 1 and capital D 2, and then we have a de interleaver, which passes to decoder 1. We have looked at this block diagram earlier, and then through the interleaver you look at 2 and so and so forth. And finally, you give the output. So, we are using two MAP decoders, operating cooperatively. That is the iterative MAP decoding algorithm.

(Refer Slide Time: 30:00)



Information Theory, Coding and Cryptography

Iterative MAP Decoding

- D1 and D2 are the two decoders.
- *S* is the set of 2^m constituent encoder states.
- ***y*** is the noisy received word.
- Using the Baye's rule we can write $L(u_k)$ as

$$L(u_k) = \log\left(\frac{P(y \mid u_k = +1)}{P(y \mid u_k = -1)}\right) + \log\left(\frac{P(u_k = +1)}{P(u_k = -1)}\right)$$

with the second term representing *a priori* information.

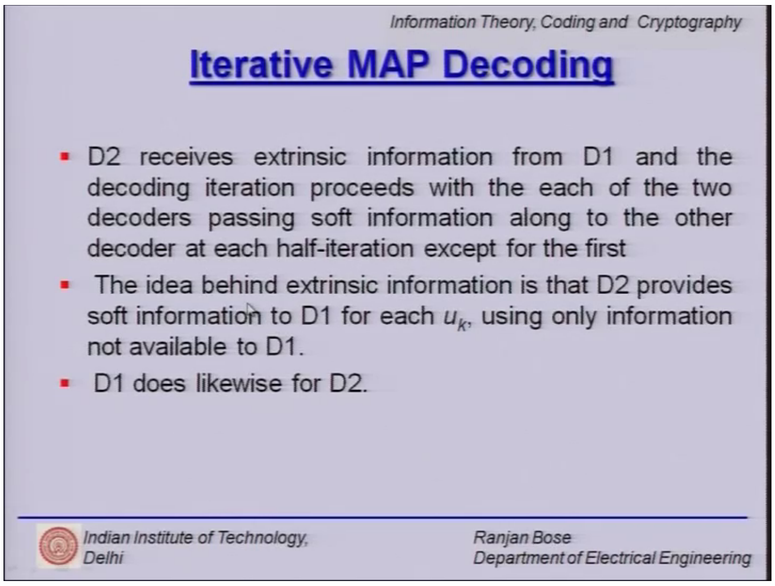
- Since $P(u_k = +1) = P(u_k = -1)$ typically, the a priori term is usually zero for conventional decoders.
- However, for iterative decoders, D1 receives extrinsic or soft information for each u_k from D2 which serves as *a priori* information.

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

Again we will have to use some kind of a likelihood function. So, D 1 and D 2 are two decoders. And we have S is the set of 2 raised power m constituent encoder states. Y vector is the noisy received word. Then we can use the Baye's rule to write L U k as follows and second term representing the a priori information here. So since, probability of U k being 1 and U k equal to minus 1 typically is equal then the a priori term is usually zero for conventional decoders, ok.

(Refer Slide Time: 30:45)



Information Theory, Coding and Cryptography

Iterative MAP Decoding

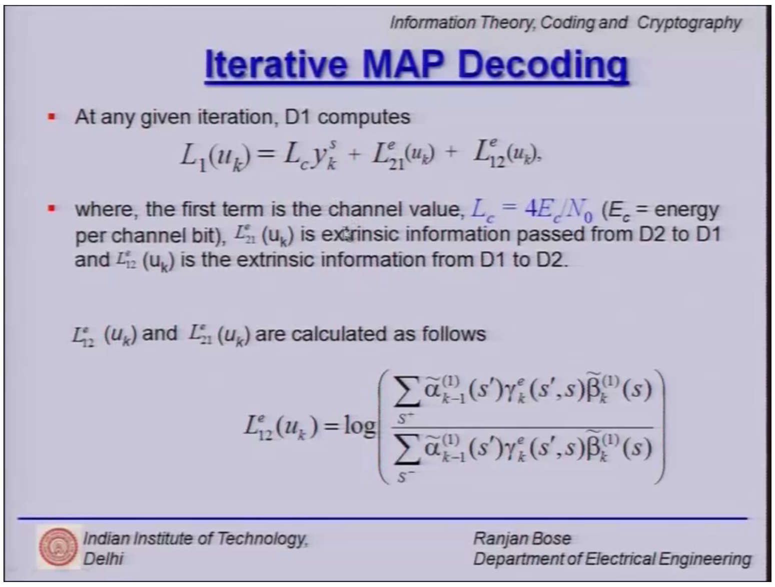
- D2 receives extrinsic information from D1 and the decoding iteration proceeds with the each of the two decoders passing soft information along to the other decoder at each half-iteration except for the first
- The idea behind extrinsic information is that D2 provides soft information to D1 for each u_k, using only information not available to D1.
- D1 does likewise for D2.

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, what D 2, second decoder, does it receives the extrinsic information from D 1, and the decoder decoding iteration proceeds with each of the two decoders passing the soft information along to the other decoder at each half-iteration. So, the idea behind this extrinsic information is that D 2 provides soft information to D 1 for each U k, using only information not available to the decoder 1. Similarly, D 1 does for D 2. So, they keep passing partial information without declaring the result till the convergence is reached.

(Refer Slide Time: 31:26)



Information Theory, Coding and Cryptography

Iterative MAP Decoding

- At any given iteration, D1 computes

$$L_1(u_k) = L_c y_k^s + L_{21}^e(u_k) + L_{12}^e(u_k),$$

- where, the first term is the channel value, $L_c = 4E_c/N_0$ (E_c = energy per channel bit), L_{21}^e (u_k) is extrinsic information passed from D2 to D1 and L_{12}^e (u_k) is the extrinsic information from D1 to D2.

L_{12}^e (u_k) and L_{21}^e (u_k) are calculated as follows

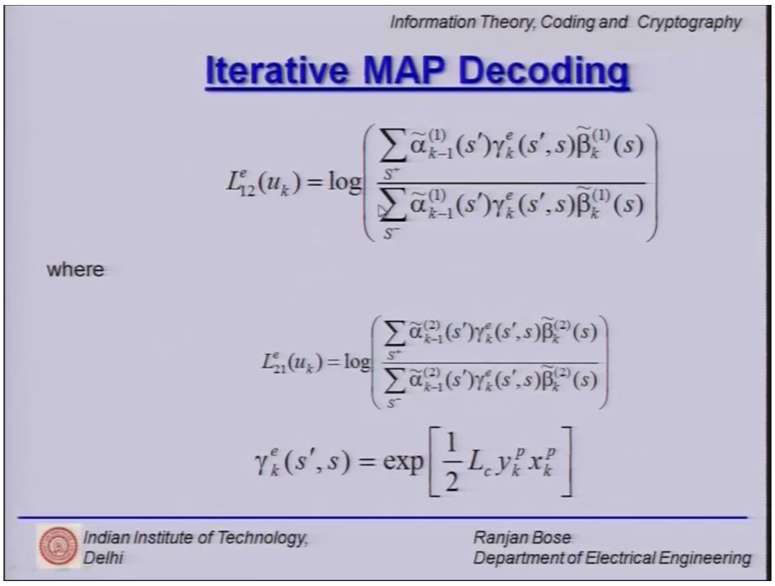
$$L_{12}^e(u_k) = \log\left(\frac{\sum_{S^+} \tilde{\alpha}_{k-1}^{(1)}(s')\gamma_k^e(s',s)\tilde{\beta}_k^{(1)}(s)}{\sum_{S^-} \tilde{\alpha}_{k-1}^{(1)}(s')\gamma_k^e(s',s)\tilde{\beta}_k^{(1)}(s)} \right)$$

Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, we can define again, the intermediary terms will not go into details, as to how these message passing is actually done. But, again we have a ratio depending on the states of the algorithm that we have studied earlier. And it is quite similar to that we have done for the BCJR algorithm.

(Refer Slide Time: 31:48)



Iterative MAP Decoding

$$L_{12}^{e}(u_k) = \log\left(\frac{\sum_{S^+} \tilde{\alpha}_{k-1}^{(1)}(s')\gamma_k^{e}(s',s)\tilde{\beta}_k^{(1)}(s)}{\sum_{S^-} \tilde{\alpha}_{k-1}^{(1)}(s')\gamma_k^{e}(s',s)\tilde{\beta}_k^{(1)}(s)}\right)$$

where

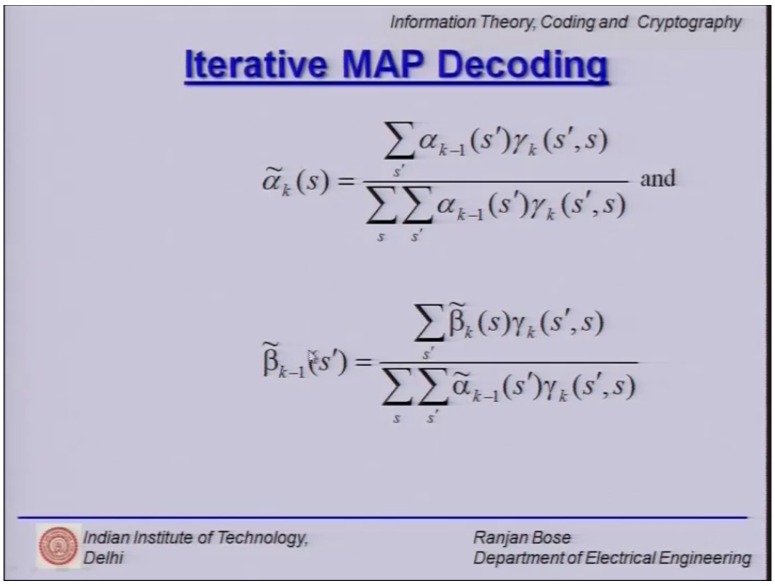
$$L_{21}^{e}(u_k) = \log\left(\frac{\sum_{S^+} \tilde{\alpha}_{k-1}^{(2)}(s')\gamma_k^{e}(s',s)\tilde{\beta}_k^{(2)}(s)}{\sum_{S^-} \tilde{\alpha}_{k-1}^{(2)}(s')\gamma_k^{e}(s',s)\tilde{\beta}_k^{(2)}(s)}\right)$$

$$\gamma_k^{e}(s',s) = \exp\left[\frac{1}{2}L_c y_k^p x_k^p\right]$$

Indian Institute of Technology, Delhi — Ranjan Bose, Department of Electrical Engineering

So, you can work this out, and you have these expressions for conducting the iterative MAP decoding.

(Refer Slide Time: 32:00)



Iterative MAP Decoding

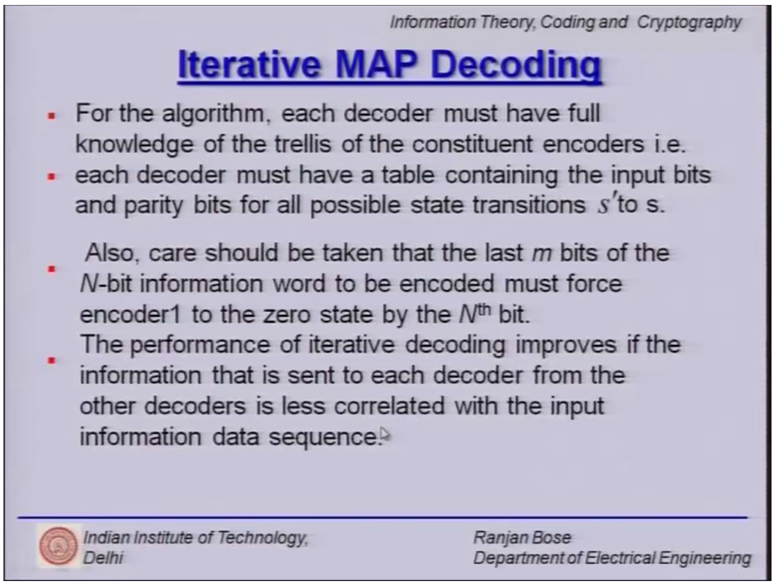
$$\tilde{\alpha}_k(s) = \frac{\sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)}{\sum_{s}\sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)} \quad \text{and}$$

$$\tilde{\beta}_{k-1}(s') = \frac{\sum_{s'} \tilde{\beta}_k(s)\gamma_k(s',s)}{\sum_{s}\sum_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s',s)}$$

Indian Institute of Technology, Delhi — Ranjan Bose, Department of Electrical Engineering

And you have this forward and backward parameters alpha k tilde and beta k tilde.

(Refer Slide Time: 32:06)


Information Theory, Coding and Cryptography

Iterative MAP Decoding

- For the algorithm, each decoder must have full knowledge of the trellis of the constituent encoders i.e.
- each decoder must have a table containing the input bits and parity bits for all possible state transitions s' to s.
- Also, care should be taken that the last m bits of the N-bit information word to be encoded must force encoder1 to the zero state by the N^{th} bit.
- The performance of iterative decoding improves if the information that is sent to each decoder from the other decoders is less correlated with the input information data sequence.

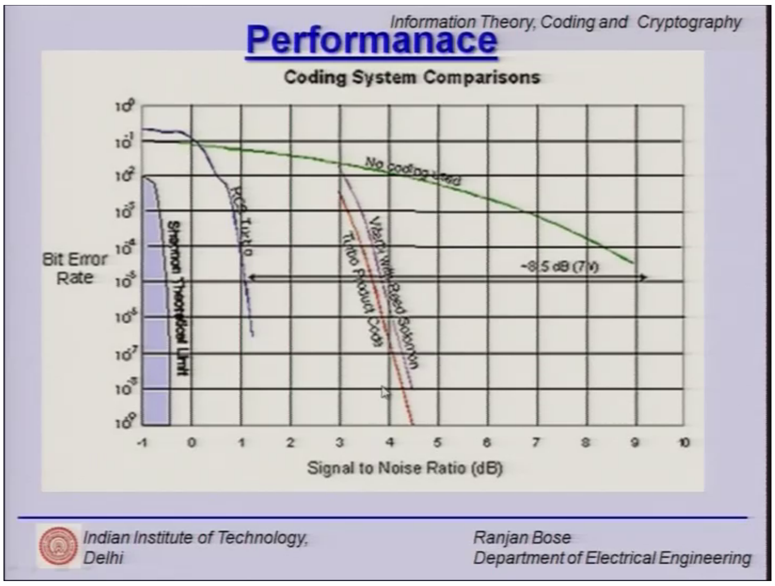
Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, for the algorithm, if you go back to the intuitive understanding of how it works, each decoder must have full knowledge of the trellis of the constituent encoders. But, since transmitter and receiver are friends it is not a difficult thing to have. So, you must know that trellis states for the decoding to happen. So, each decoder must have a table containing the input bits and the parity bits for all possible state transition ok. So, the state transition diagram must be available, and based on that the alpha and beta are calculated.

Also care should be taken that the last N bits of the N-bit information word to be encoded must force encoder 1 to state 0, so that for the next block we can start a fresh ok. There should be no ripple effect. The performance of iterative decoding improves, if the information that is sent to each decoder from the other decoder is less correlated with the input information data sequence, this is just an observation.

(Refer Slide Time: 33:15)


Information Theory, Coding and Cryptography

Performanace

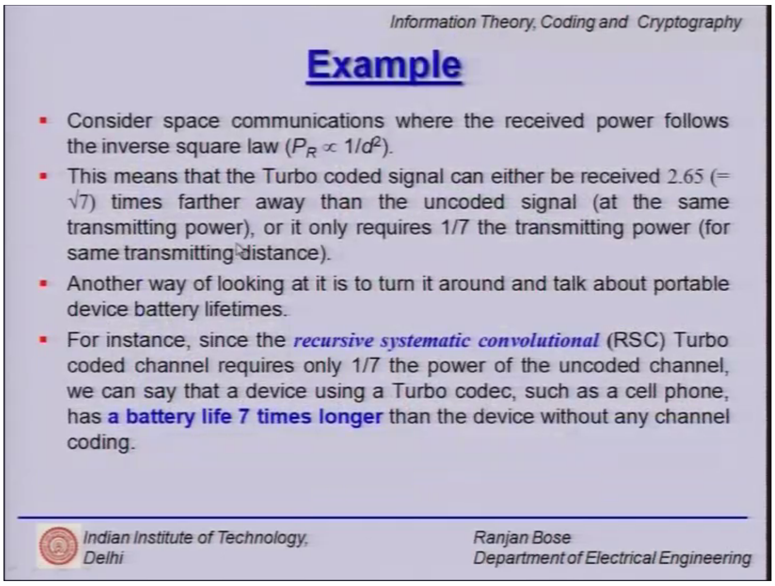
Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, just let us compare, how turbo decoding performs. So, if let us put the axis in place, so the x axis represents the signal to noise ratio in dB, the Y axis represents the bit error rate. So, if you see, the greenish curve is no coding is used. So, depending upon the signal to noise ratio, you have a raw error rate, which is really unacceptable. And then if you use this reed Solomon code, you are somewhere here, depending upon what is your signal to noise ratio. So, as you can see that this x axis is SNR, so reed Solomon codes work well at reasonably high SNR then.

If you look at turbo product codes, they are slightly here. But, if you look at thus recursive systematic convolutional turbo code, this is the performance cover that it really works at low SNR. And here is the Shannon’s limit by the way. So, relatively speaking, it is much closer to the Shannon’s limit, and you can get a remarkable performance. Suppose, you are working at 10 raised power minus 5, this is the 10 raised power minus 5. Then with respect to uncoded, you can have a humongous 8.5 dB coding gain, because of this RCS turbo encoder.

(Refer Slide Time: 34:52)



Information Theory, Coding and Cryptography

Example

- Consider space communications where the received power follows the inverse square law ($P_R \propto 1/d^2$).
- This means that the Turbo coded signal can either be received 2.65 (= $\sqrt{7}$) times farther away than the uncoded signal (at the same transmitting power), or it only requires 1/7 the transmitting power (for same transmitting distance).
- Another way of looking at it is to turn it around and talk about portable device battery lifetimes.
- For instance, since the ***recursive systematic convolutional*** (RSC) Turbo coded channel requires only 1/7 the power of the uncoded channel, we can say that a device using a Turbo codec, such as a cell phone, has **a battery life 7 times longer** than the device without any channel coding.

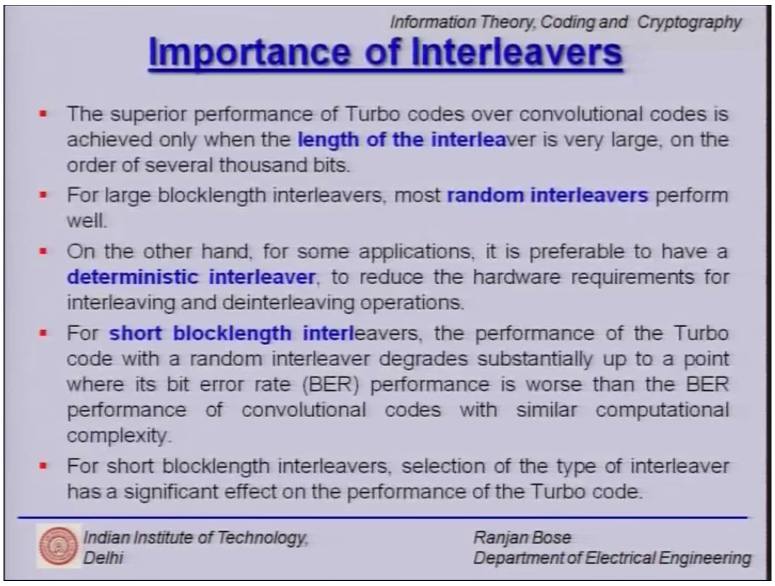
Indian Institute of Technology, Delhi

Ranjan Bose
Department of Electrical Engineering

So, let us look at an example, how effective it will be in real life. Suppose, we are doing deep space communication where we are having this inverse square law being followed with respect to the distance, so this means that the turbo coded signal can be received with 2.65 that is under root 7 times farther away from with respect to the uncoded signals, because it requires 1-7th of the transmit power for the same transmitting distance.

So, even if you have large distances are use of a turbo coder, would really require you to use much lesser power. And these are power limited channels, so it really makes sense. As we saw, in this previous diagram of a recursive systematic convolutional RSC turbo encoder, this is the diagram for a bit error rate of 10 raise power minus 5; we have this minus 8.5 dB. So, this is the coding gain, which is like under root 7, and which gives you thus humongous saving in power. So, the battery life tentatively lasts 7 times longer ok, or from the distance it is almost 2.6 times, the distance that can be sent the signal from, so as to get the same performance. So, it is pretty impressive in terms of the performance.

(Refer Slide Time: 36:30)



Information Theory, Coding and Cryptography

Importance of Interleavers

- The superior performance of Turbo codes over convolutional codes is achieved only when the **length of the interleaver** is very large, on the order of several thousand bits.
- For large blocklength interleavers, most **random interleavers** perform well.
- On the other hand, for some applications, it is preferable to have a **deterministic interleaver**, to reduce the hardware requirements for interleaving and deinterleaving operations.
- For **short blocklength interleavers**, the performance of the Turbo code with a random interleaver degrades substantially up to a point where its bit error rate (BER) performance is worse than the BER performance of convolutional codes with similar computational complexity.
- For short blocklength interleavers, selection of the type of interleaver has a significant effect on the performance of the Turbo code.

Indian Institute of Technology, Delhi

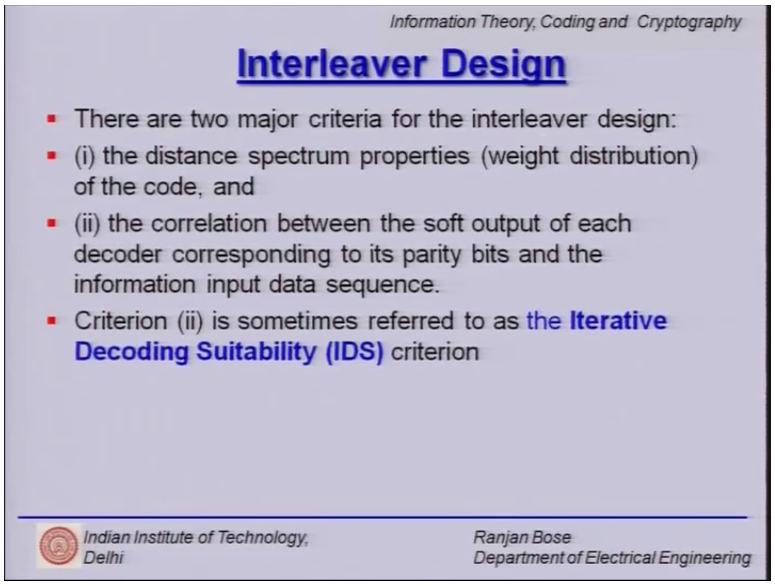
Ranjan Bose
Department of Electrical Engineering

Now, we come to this interleaver, which has made a life difficult for decoding purposes,. But, let us talk briefly about the interleavers, what is the importance, what are the types, how can we design them. So, superior performance of turbo codes over convolutional codes is achieved only when the length of the interleaver is quite large. So since, the interleaver must be present, so the interleaver works with a block. Therefore, this convolutional code lies between block codes and convolutional codes. We worked with a block size at a time.

So, most random interleavers work well. On the other hand, for some applications, it is preferable to have a deterministic interleaver, to reduce the hardware requirements for the interleaving and deinterleaving process. For short block length interleavers, the performance of turbo codes with a random interleaver degrades substantially up to a point, when its BER performance is worse, then the BER performs with convolutional code standard convolutional codes with similar computational complexity.

So, please note that the short block interleaver right that the block length interleaver must be large enough to give you the gain. Otherwise, we will go back to normal, and the advantage will wear off, because you are actually puncturing and throwing out some of the bits, right. So, for short block length interleaves, the selection of the type of interleaver does have a significant effect on the performance.

(Refer Slide Time: 38:11)


Information Theory, Coding and Cryptography

Interleaver Design

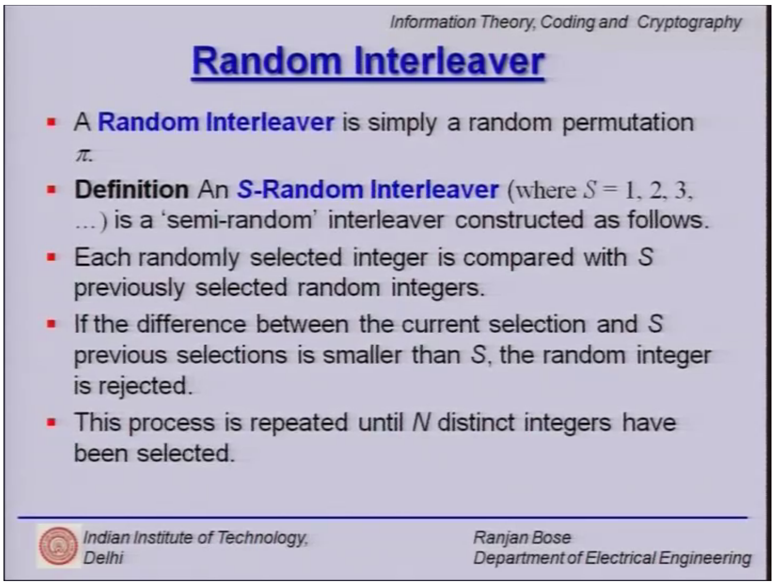
- There are two major criteria for the interleaver design:
- (i) the distance spectrum properties (weight distribution) of the code, and
- (ii) the correlation between the soft output of each decoder corresponding to its parity bits and the information input data sequence.
- Criterion (ii) is sometimes referred to as **the Iterative Decoding Suitability (IDS)** criterion

Indian Institute of Technology, Delhi
Ranjan Bose
Department of Electrical Engineering

So, let us quickly spends some time on the interleaver designs. The two major criteria for interleaver design. One is the distance spectrum properties, which is the weight distribution of the code; I mean that is how we started off with why are we adding an interleaver, because for the same input bit stream, if the input bit stream leads to a low weight sequence, then the interleaver should shuffle the bit stream, such that the resultant output should be a high weight output. Number two criterion is the correlation between the soft output of each decoder corresponding to its parity bits, and the information input information sequence should be such that the advantage of decoding is achieved.

So, the second criteria of the correlation between soft output of each decoder and the information sequence, is called the iterative decoding suitability criteria. So, that is how we design interleavers. So, they have been designed by trial and error or there are some design rules. And once you fix an interleaver, you are good to go.

(Refer Slide Time: 39:30)



Information Theory, Coding and Cryptography

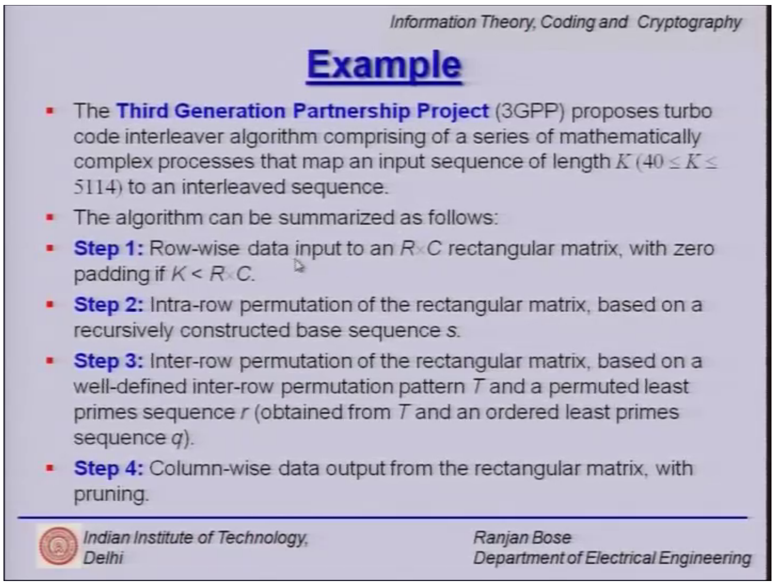
Random Interleaver

- A **Random Interleaver** is simply a random permutation π.
- **Definition** An ***S*-Random Interleaver** (where $S = 1, 2, 3, \ldots$) is a 'semi-random' interleaver constructed as follows.
- Each randomly selected integer is compared with *S* previously selected random integers.
- If the difference between the current selection and *S* previous selections is smaller than *S*, the random integer is rejected.
- This process is repeated until *N* distinct integers have been selected.

Indian Institute of Technology, Delhi | Ranjan Bose, Department of Electrical Engineering

What about the random interleaver. Well, you do not have any effort in designing it is simply a random permutation pi. So, if you define S-random interleaver, is a 'semi-random' interleaver, which is constructed as follows. So, it is a compromise between a random interleaver deterministic. Each randomly selected integer is compared with S, where S could be 1, 2, or 3 or and so on and so forth, previously selected random integers. If the difference between the current selection and S previous selections is smaller than S, the random integer is rejected. So, it is a way of picking random data. This process is repeated until N distinct integers have been selected. And thereby, you end up with a S random interleaver.

(Refer Slide Time: 40:14)



Information Theory, Coding and Cryptography

Example

- The **Third Generation Partnership Project** (3GPP) proposes turbo code interleaver algorithm comprising of a series of mathematically complex processes that map an input sequence of length K ($40 \leq K \leq 5114$) to an interleaved sequence.
- The algorithm can be summarized as follows:
- **Step 1:** Row-wise data input to an $R \times C$ rectangular matrix, with zero padding if $K < R \times C$.
- **Step 2:** Intra-row permutation of the rectangular matrix, based on a recursively constructed base sequence s.
- **Step 3:** Inter-row permutation of the rectangular matrix, based on a well-defined inter-row permutation pattern T and a permuted least primes sequence r (obtained from T and an ordered least primes sequence q).
- **Step 4:** Column-wise data output from the rectangular matrix, with pruning.

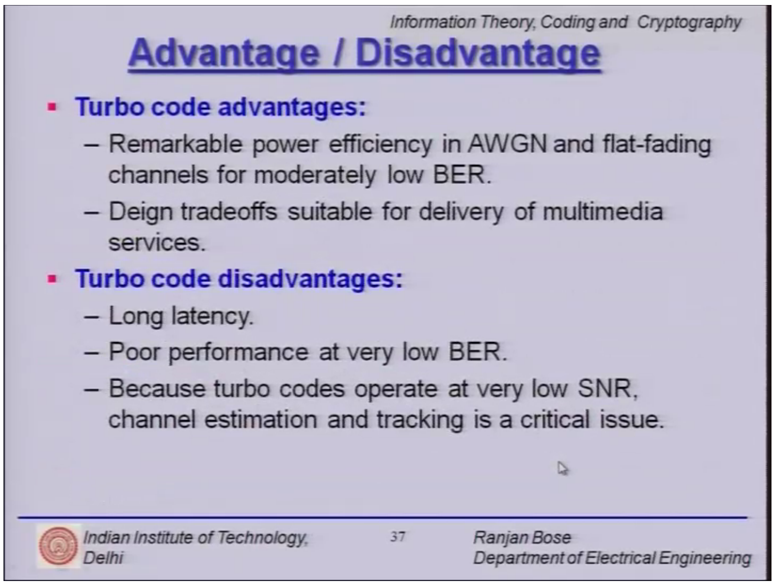
Indian Institute of Technology, Delhi   Ranjan Bose, Department of Electrical Engineering

Let us look at an example, how it is used in real life. So, the 3GPP, which is the third generation partnership project proposes, the use of turbo code interleaver and algorithm is given in this slide. So, the input sequence can be starting from 40 to over 5000 ok, so we have to input this large size of a sequence.

So, let us summarize the algorithm. So, step 1, you have row-wise data input to an R cross C rectangular matrix, with zero padding right, and K is less than R cross C. So, I have to fit in those many bits into a matrix. So, R and C represents the rows and columns. Now, intra-row permutation of the rectangular matrix is carried out, based on recursively reconstructed base sequence S. Then step 3 is inter-row permutation of the rectangular matrix, based on a well defined inter-row permutation pattern T, so you can specify T.

And then finally, a column-wise data output of the rectangular matrix. If you remember, the most basic form of an interleaver is, you feed in the data row-wise, and read it out column-wise right. But, here we put in the data row-wise, and then do row permutations of two types, and then read out the data column-wise.

(Refer Slide Time: 41:52)



Information Theory, Coding and Cryptography

Advantage / Disadvantage

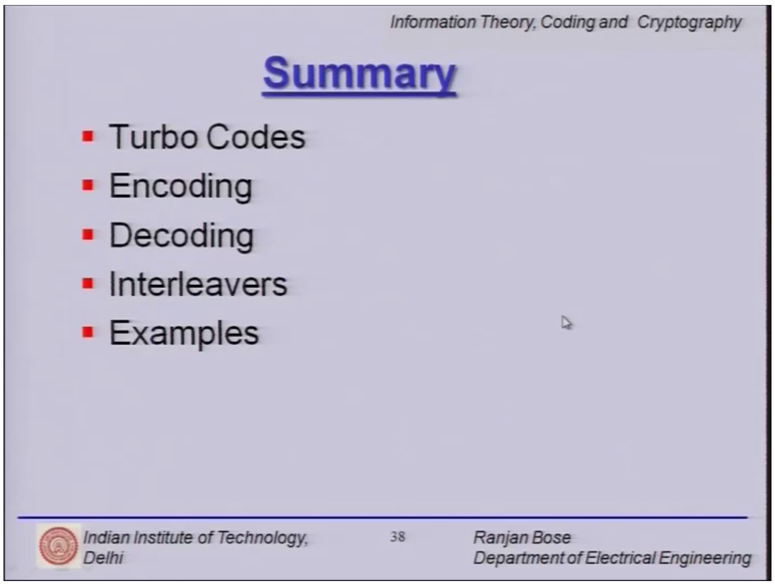
- **Turbo code advantages:**
  - Remarkable power efficiency in AWGN and flat-fading channels for moderately low BER.
  - Deign tradeoffs suitable for delivery of multimedia services.
- **Turbo code disadvantages:**
  - Long latency.
  - Poor performance at very low BER.
  - Because turbo codes operate at very low SNR, channel estimation and tracking is a critical issue.

Indian Institute of Technology, Delhi 37 Ranjan Bose Department of Electrical Engineering

Finally, what are the advantages and disadvantages of the turbo codes? Well, let us start with advantages; remarkable power efficiency in Additive White Gaussian Noise, and flat-fading channels for moderately low bit error rate. Please note, moderately low, because if you put the BER too low, then the convergence does not happen, and we do not get good results. And that design tradeoffs suitable for delivery of multimedia data services.

What are the disadvantages, long latency, because it is an iterative algorithm? We have to wait till the entire block is received iterations happen and decoding is declared. Poor performance at very low BER, so BER cannot be below a threshold, otherwise we will keep iterating and not converge. Because, turbo codes operated at very low SNR, channel estimation tracking is critical to the decoding process, because the efficiency of this turbo codes is at low SNR suddenly. So, yes that is kind of a disadvantage.

(Refer Slide Time: 43:00)



Information Theory, Coding and Cryptography

Summary

- Turbo Codes
- Encoding
- Decoding
- Interleavers
- Examples

Indian Institute of Technology, Delhi 38 Ranjan Bose Department of Electrical Engineering

Now, let us summarize what we have done today. We started off with a brief introduction to turbo codes, we looked at what is the structure of turbo codes, how concatenated convolutional encoders can be converted into turbo codes. We looked at the encoding process. And finally, we looked at two decoders, and then looked at the importance of interleavers for turbo decoders. Finally, we looked at some examples, to show the efficacy of this turbo encoders and decoders.

With that we come to the end of this lecture.