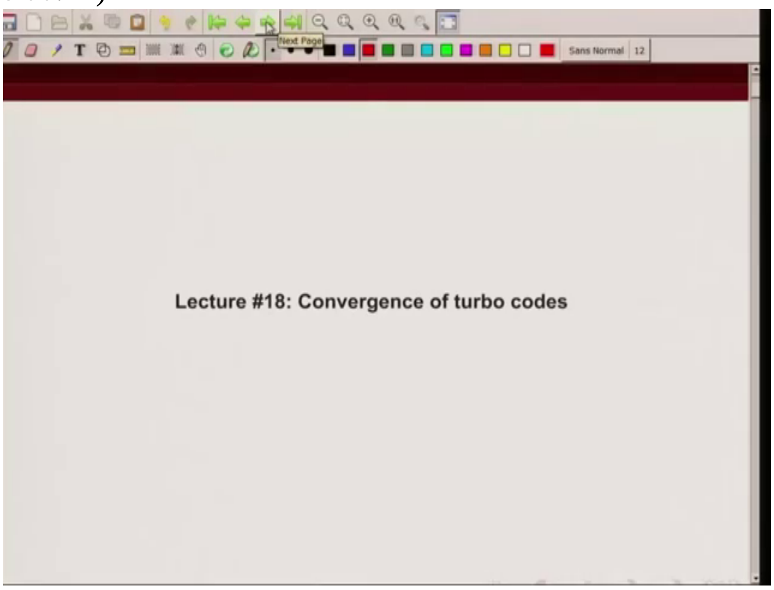
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 08
Lecture Number 31
Convergence of turbo codes

(Refer Slide Time 00:13)

Today we are going to talk about how to analyze the performance of turbo code in low S N R.

(Refer Slide Time 00:22)

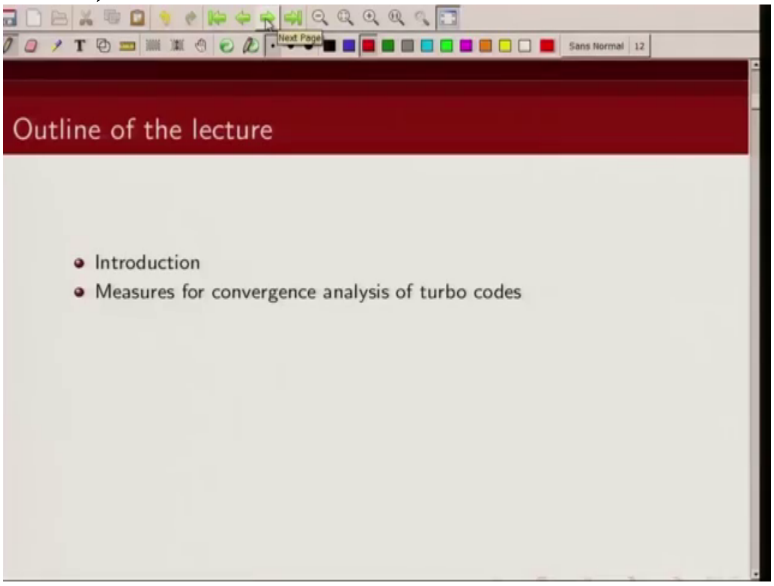


So we are going to talk about convergence, how to track the convergence

(Refer Slide Time 00:28)

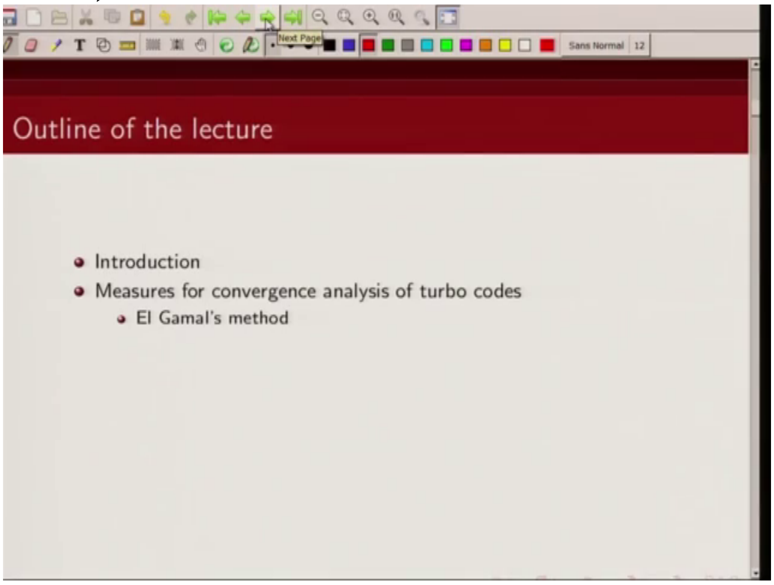
of turbo iterative decoding algorithm and that's the topic of our discussion, convergence of turbo codes. So with brief introduction, we will talk about

(Refer Slide Time 00:40)



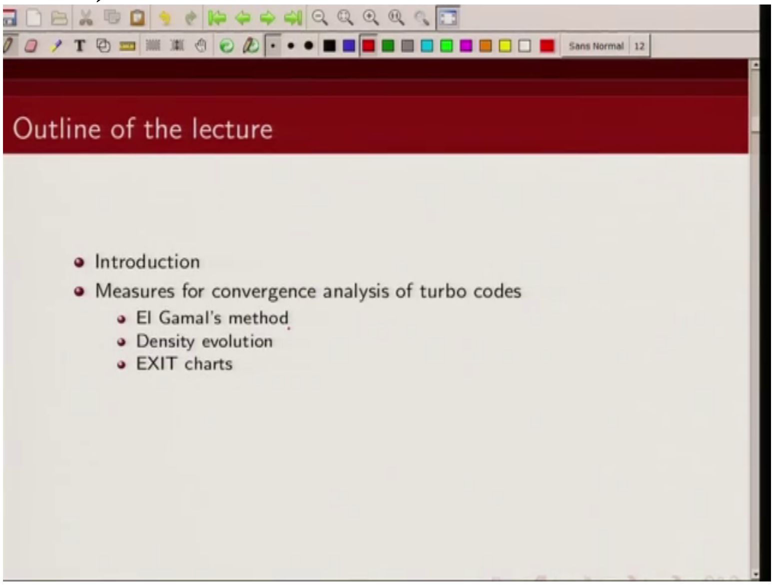
what are the various measures for convergence analysis

(Refer Slide Time 00:45)



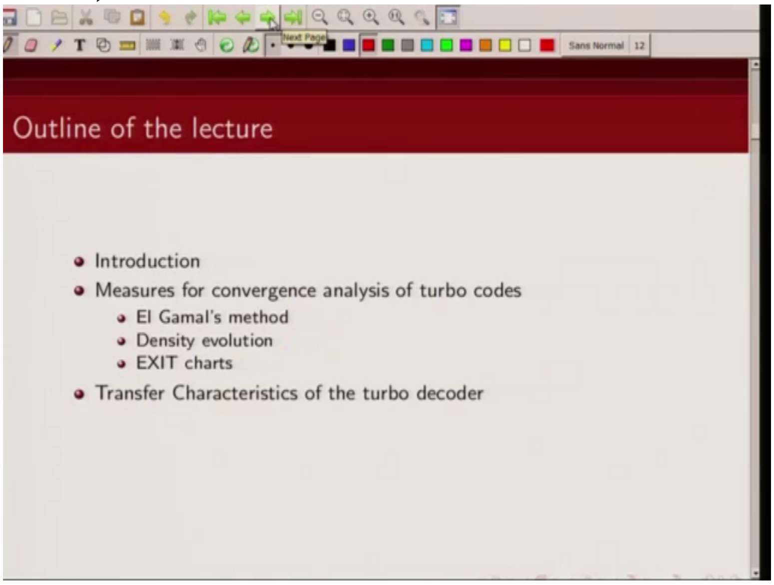
and in particular we will, are going to talk about these three methods, the first method

(Refer Slide Time 00:50)



which is based on Gaussian approximation and which involves tracking the mean of the extrinsic values, a method proposed by El Gamal. Next we will talk about a method which is proposed by Divsalar and others using density evolution and then a method which is based on mutual information, tracking mutual information proposed by ten Brink.

(Refer Slide Time 01:20)

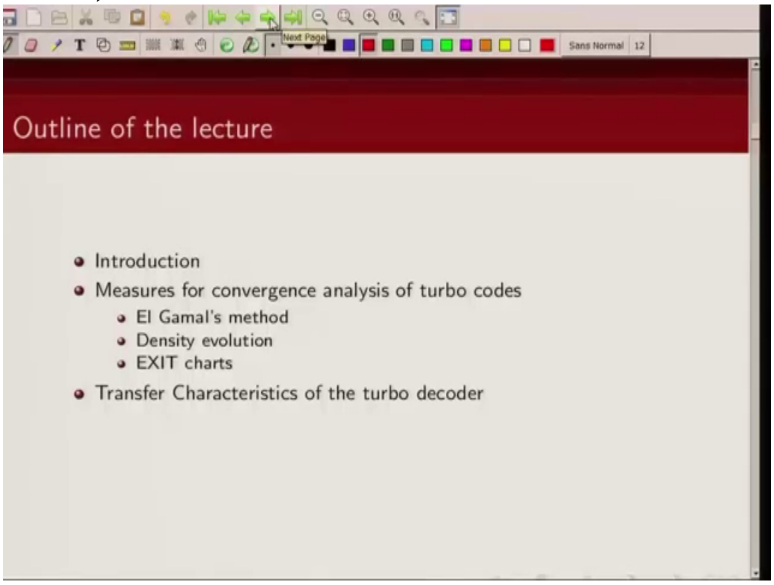


And then we will talk about what do we mean by a

(Refer Slide Time 01:24)

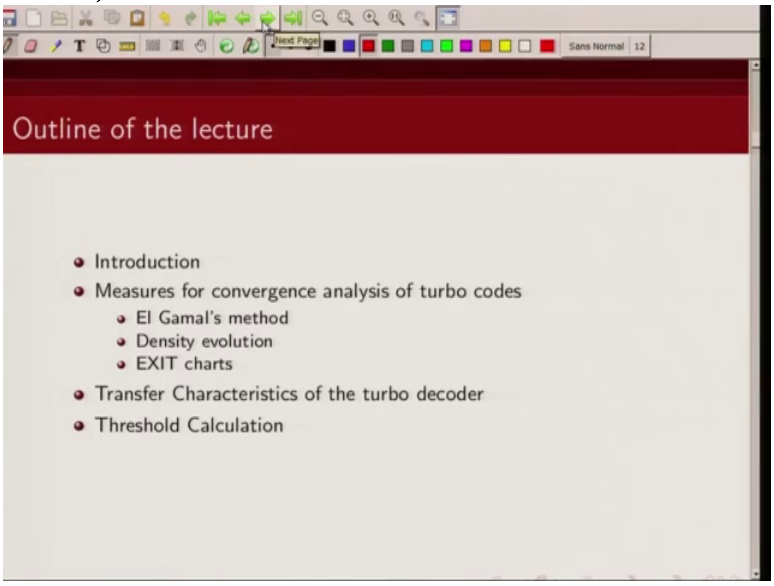
transfer characteristic of a turbo

(Refer Slide Time 01:26)



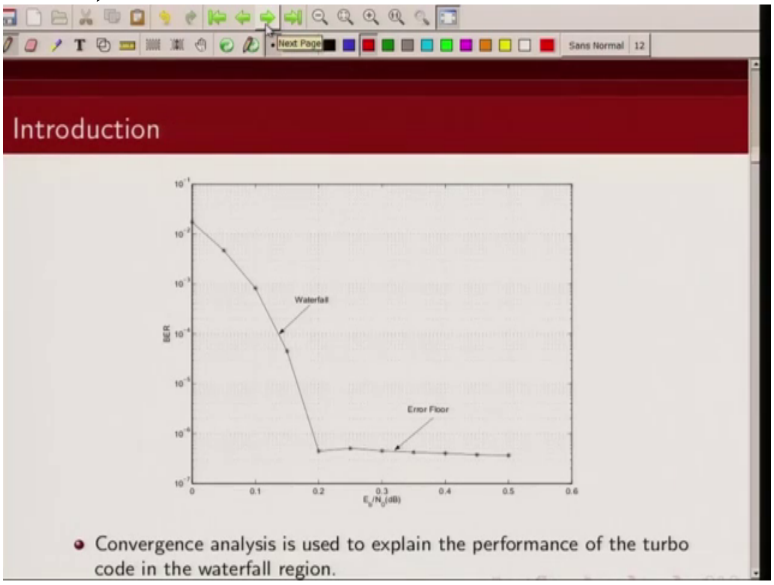
decoder and how we can use it to compute the convergence threshold

(Refer Slide Time 01:30)



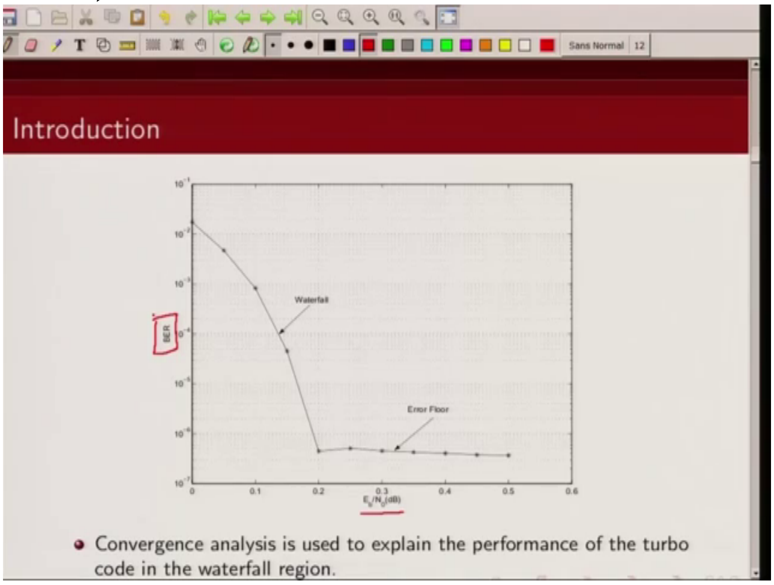
of a turbo code.

(Refer Slide Time 01:33)



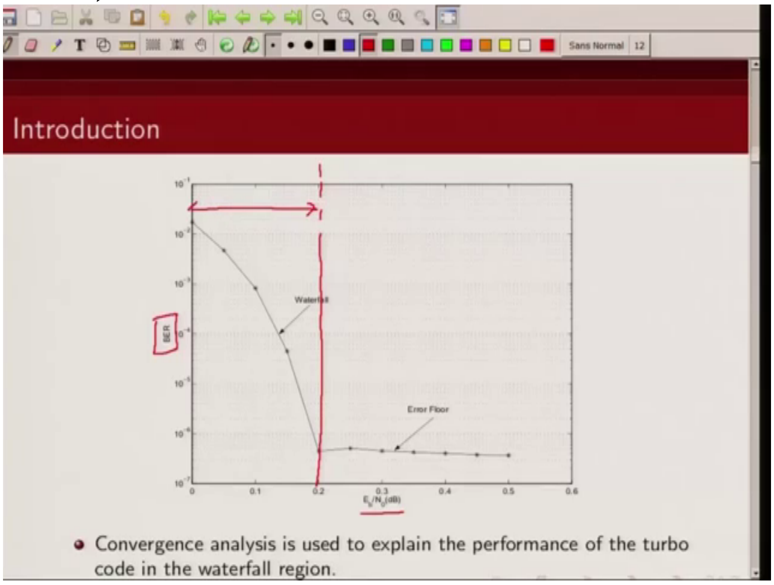
So this is a typical performance of a turbo code. If we take a larger block size, this is for a block size, I think 65000 plus, so if you take a large block size, this is typical performance of a turbo code. On x axis, I have signal to noise ratio and on the y axis, I have plotted

(Refer Slide Time 01:55)



bit error rate. Now you will see there is a region, so this region which we are calling

(Refer Slide Time 02:03)

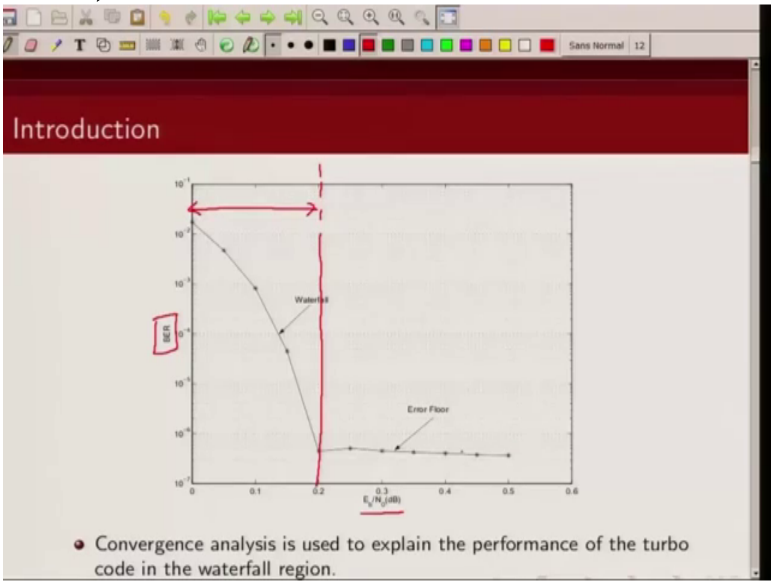


waterfall region where there is a steep fall

(Refer Slide Time 02:06)

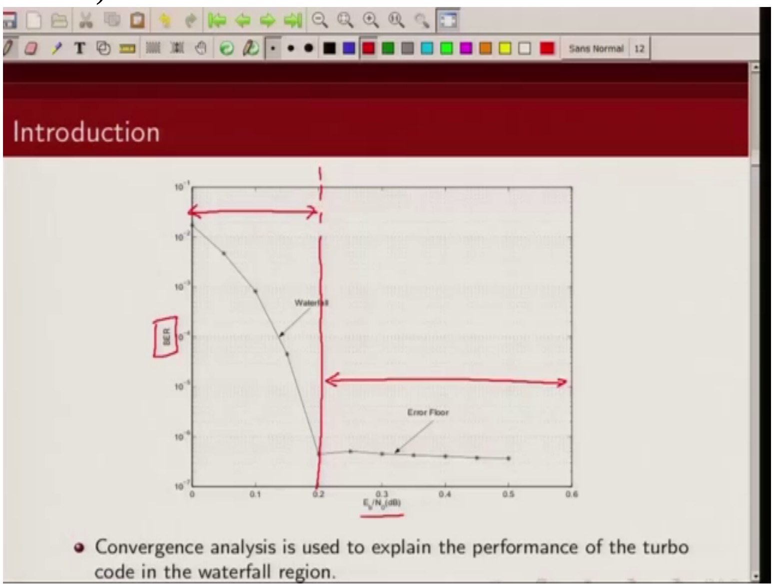
in bit error rate performance and there

(Refer Slide Time 02:08)



is a region, we call it error flow region

(Refer Slide Time 02:12)

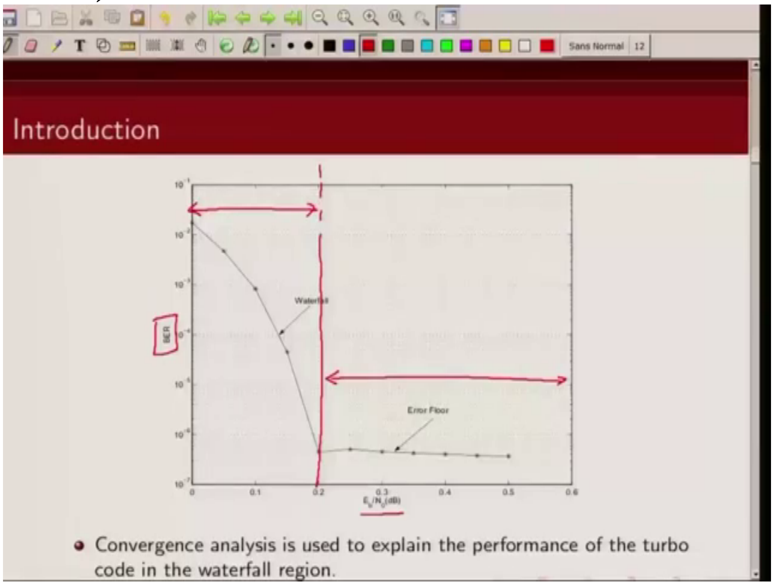


where

(Refer Slide Time 02:14)

the b e r does not improve much. So today's

(Refer Slide Time 02:20)

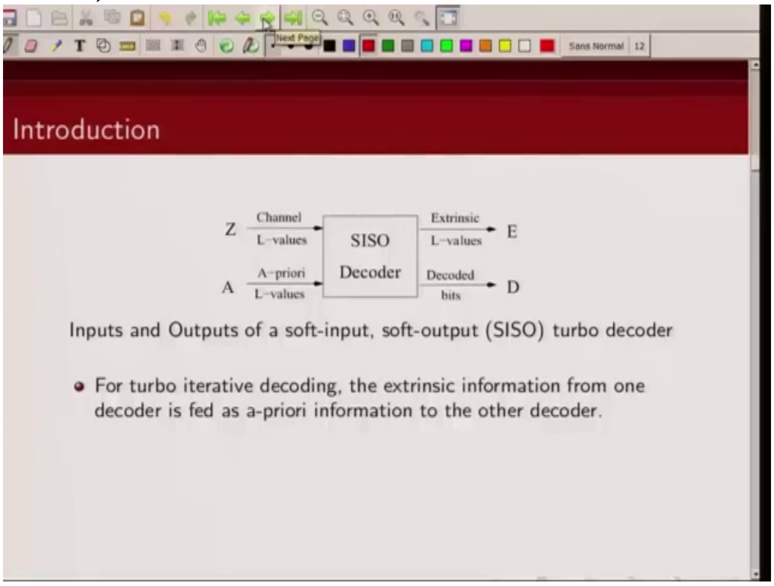


topic of discussion is this waterfall region. What determines

(Refer Slide Time 02:25)

the performance of turbo code in this region where it falls sharply and how can we get some guidelines on how to choose constituent encoders so that we get a steep fall like this.

(Refer Slide Time 02:40)

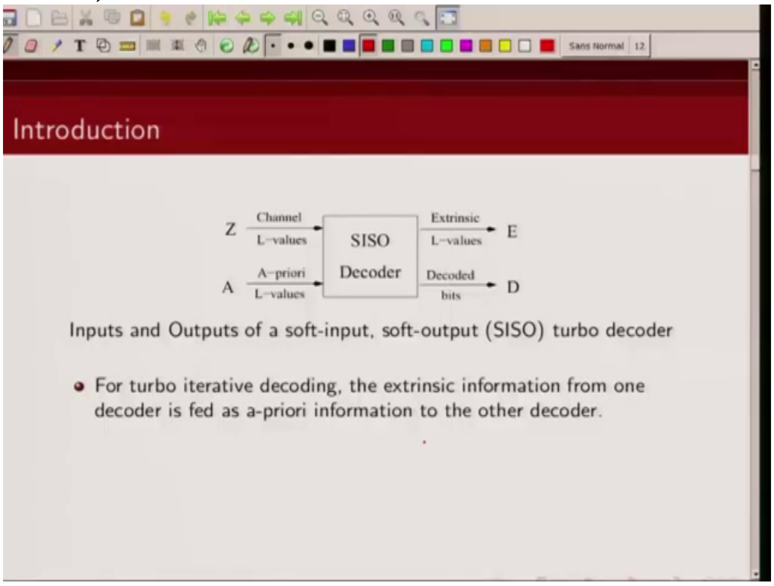


So before we study the convergence

(Refer Slide Time 02:44)

analysis, convergence of turbo code, let's pay close attention to the basic block diagram of our turbo decoder.

(Refer Slide Time 02:55)

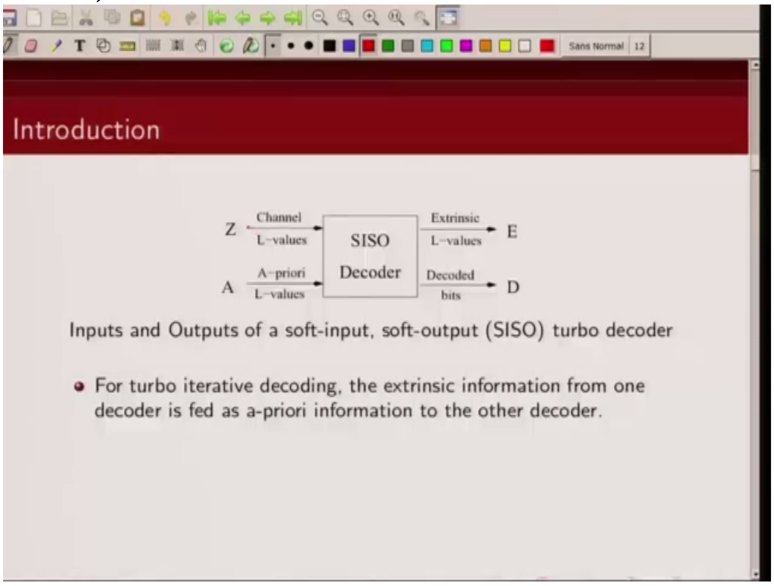


The heart of the turbo decoder is the soft input soft output decoder and if you recall this soft input soft output

(Refer Slide Time 03:03)

decoder takes in as input the channel

(Refer Slide Time 03:07)

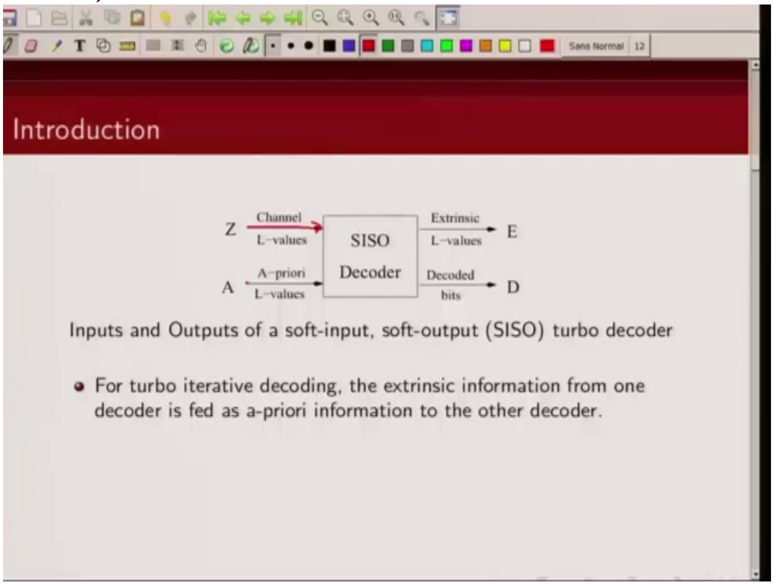


received values corresponding to the information

(Refer Slide Time 03:10)

and parity bits,

(Refer Slide Time 03:12)

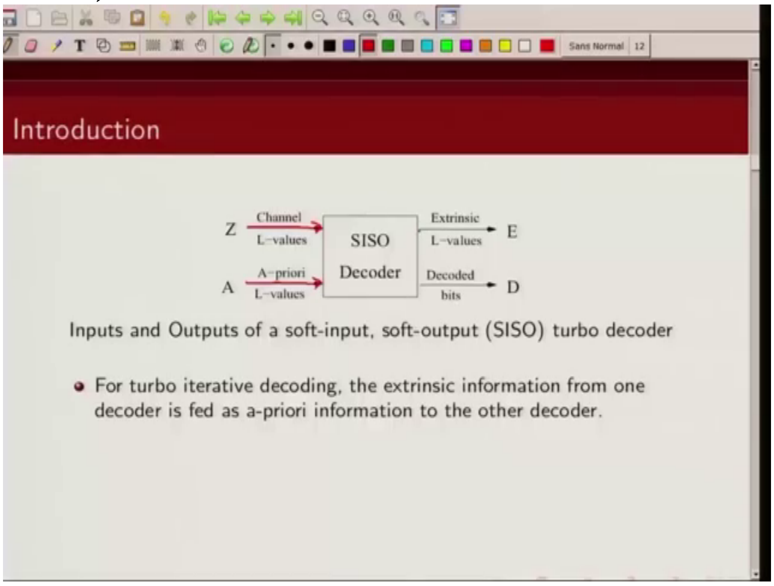


a priori value which

(Refer Slide Time 03:15)

it receives from the other decoder, which are the extrinsic values passed on to the other decoder and it

(Refer Slide Time 03:23)

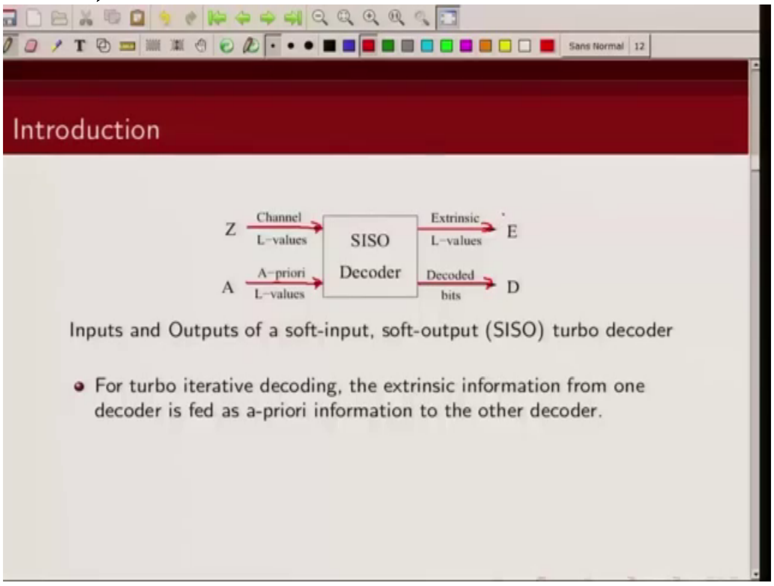


computes extrinsic values as well as A P P L values

(Refer Slide Time 03:28)

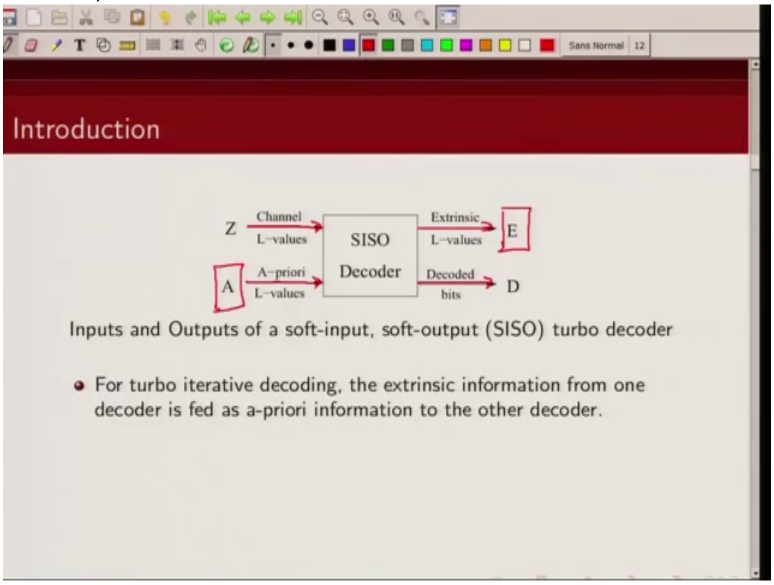
where you take a hard decision to get back your decoded bits. So if you look at a turbo decoder, this is the heart of the turbo decoder. There are two such soft input soft output decoder and if you look for a particular signal to noise ratio, if we look at turbo decoder as a function of iteration you will notice the only thing changing with iteration is this

(Refer Slide Time 03:58)



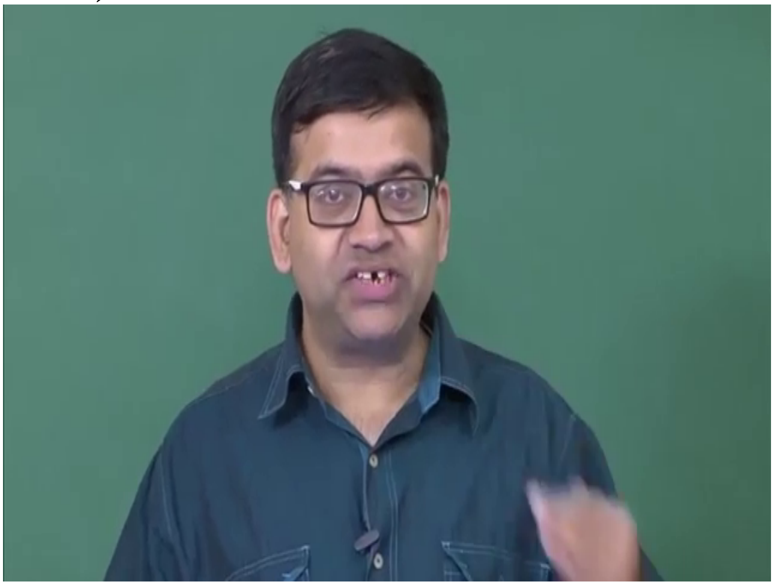
extrinsic value and

(Refer Slide Time 04:01)



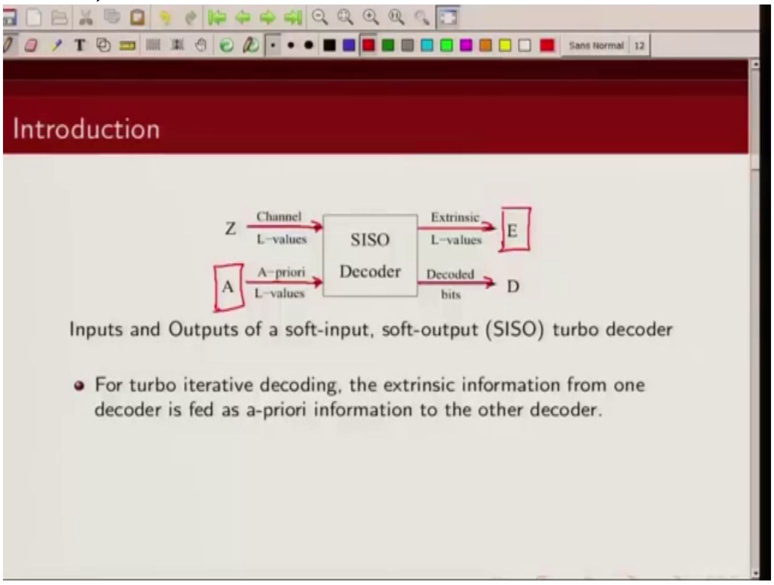
a priori value. So with iteration,

(Refer Slide Time 04:04)



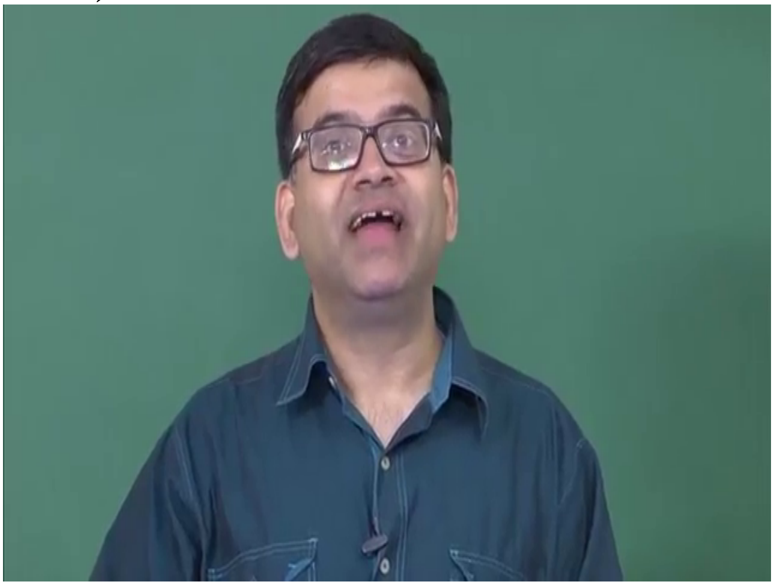
your, initially you do not have any estimate on a priori value, you assume that the bits are equally likely to be zero and 1 but subsequently with iteration when your extrinsic values are generated, those are passed on as a priori value. Now the channel L values remain same for a fixed signal to noise ratio; for a received bit, the channel L value remains same. Only thing changing with iteration are these two quantities,

(Refer Slide Time 04:35)



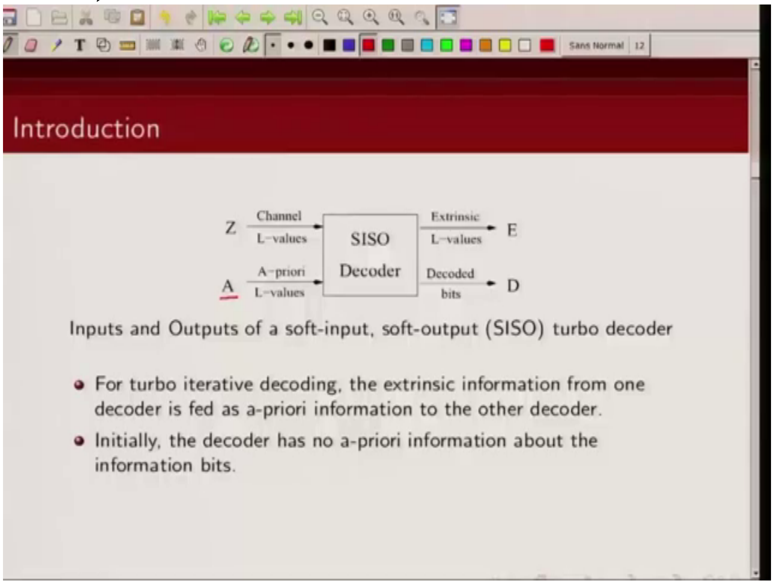
this a priori value and the

(Refer Slide Time 04:37)



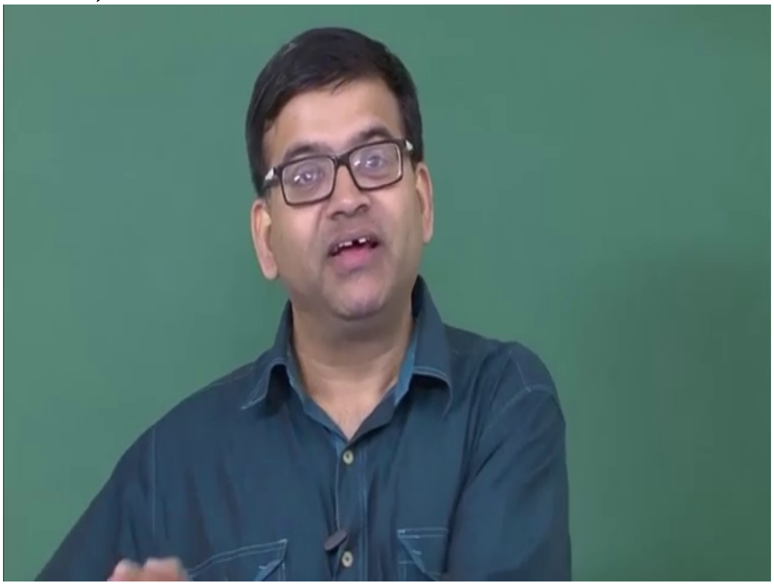
extrinsic value. So if we can track with iteration how our extrinsic information is growing with this a priori information, that will give us some clue about the performance of turbo code at waterfall region.

(Refer Slide Time 04:56)



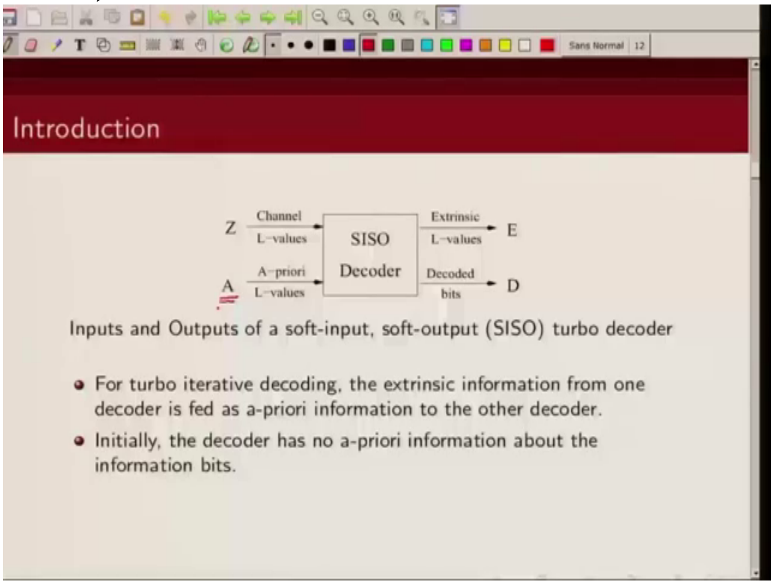
So as I said, initially we do not have any a priori value but subsequently after one half iteration

(Refer Slide Time 05:05)



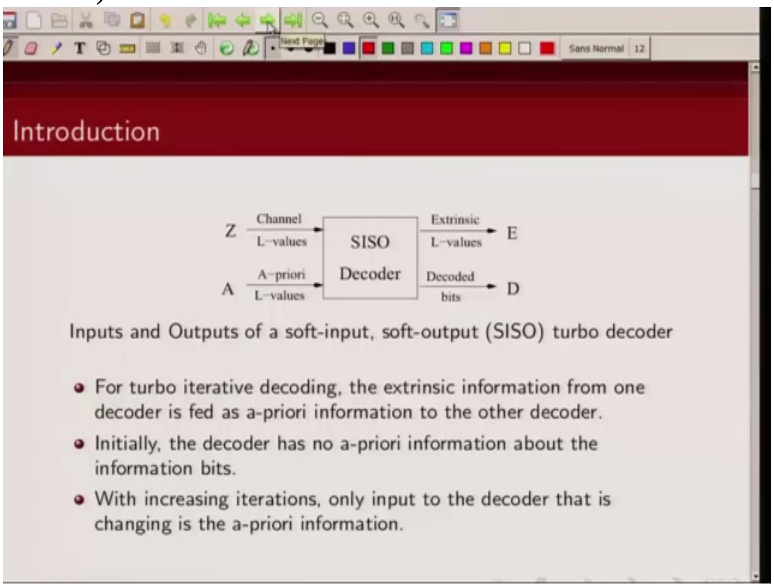
extrinsic information are generated and that's passed on as a priori value

(Refer Slide Time 05:10)



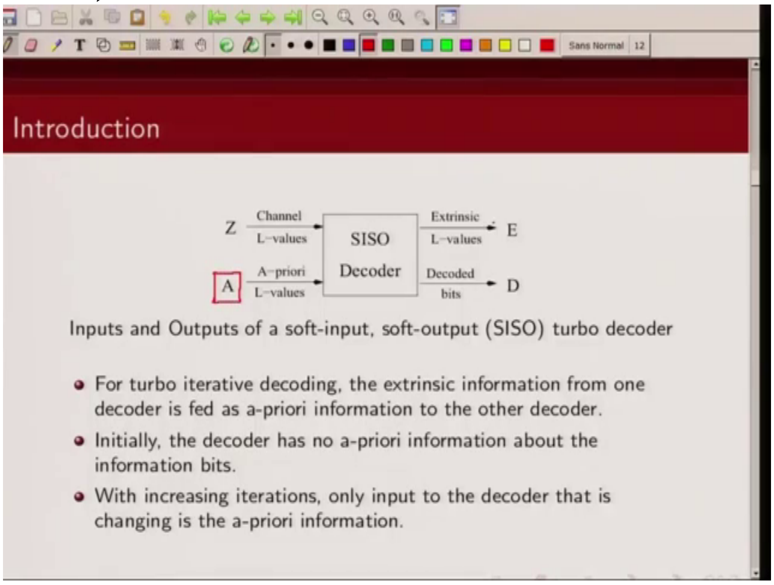
to this soft input soft output

(Refer Slide Time 05:12)



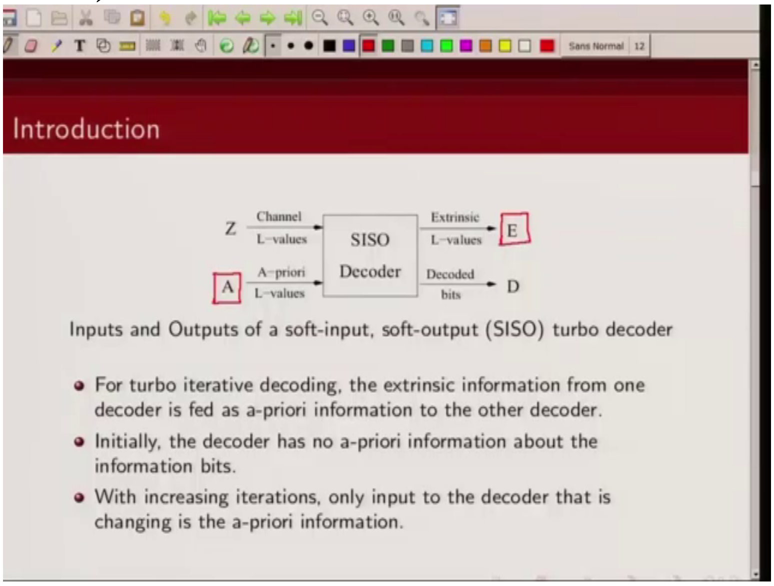
decoder. And again I emphasize, the only thing changing with iteration are

(Refer Slide Time 05:21)



these extrinsic values and

(Refer Slide Time 05:23)

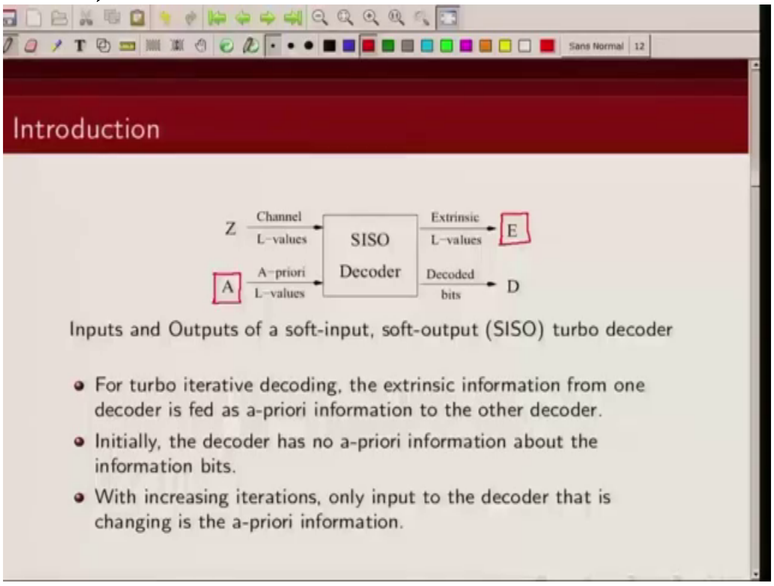


a priori values. So if

(Refer Slide Time 05:26)

you want to track how your turbo decoder is working with iteration, you need to track these two quantities

(Refer Slide Time 05:34)

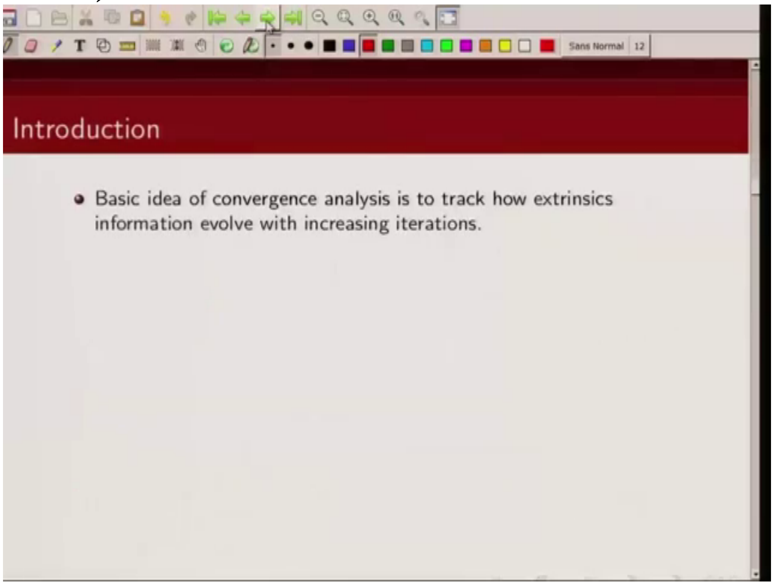


and we are going to talk about what are the various measures

(Refer Slide Time 05:38)

that we can use to track these two quantities.

(Refer Slide Time 05:41)

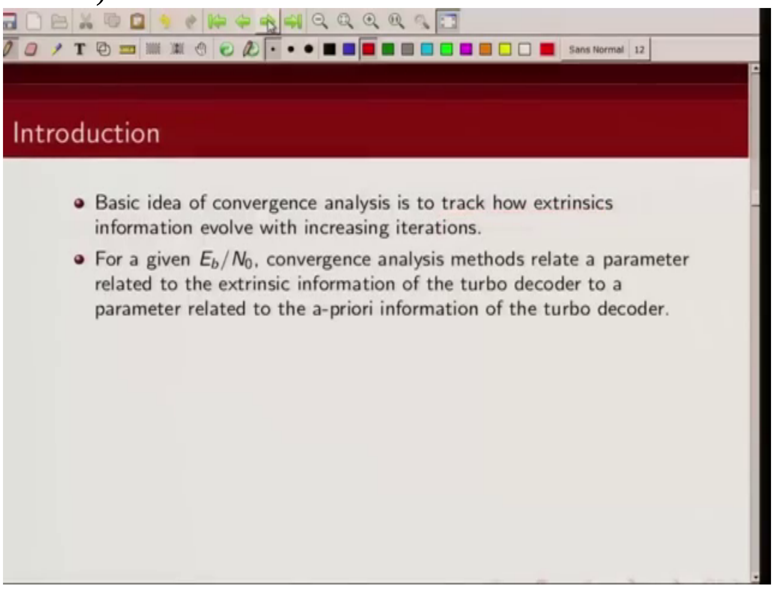


So basic idea of convergence of turbo code, convergence analysis of turbo code is to track how these extrinsic information are evolving with increased iteration. So if you feed in

(Refer Slide Time 06:00)

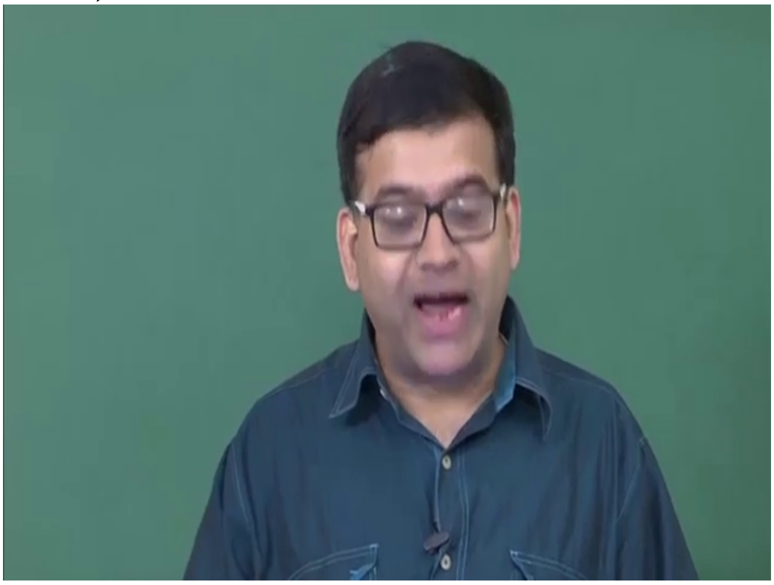
better a priori value, how is your extrinsic information

(Refer Slide Time 06:06)



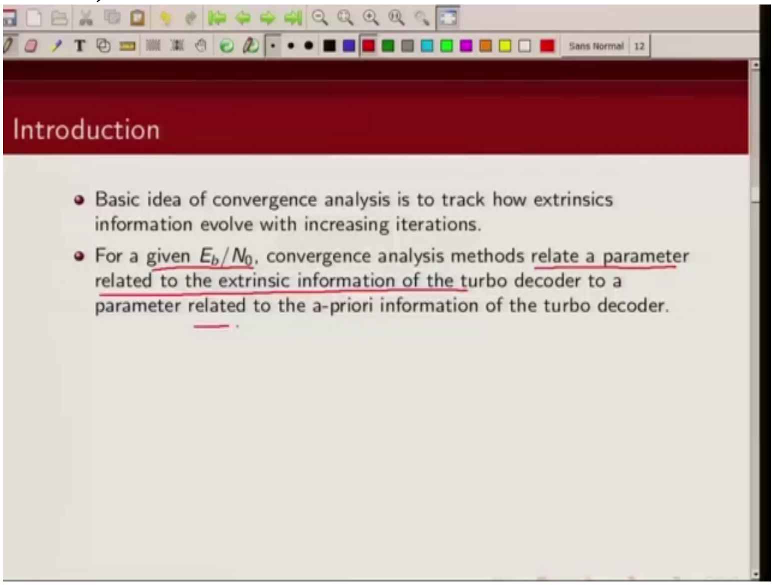
evolving? So what we do is, for a fixed signal to noise ratio we have a set of received values. So what we do is we try to relate a parameter which is related to the extrinsic information of the turbo decoder and

(Refer Slide Time 06:25)



we try to relate it to the parameter

(Refer Slide Time 06:27)

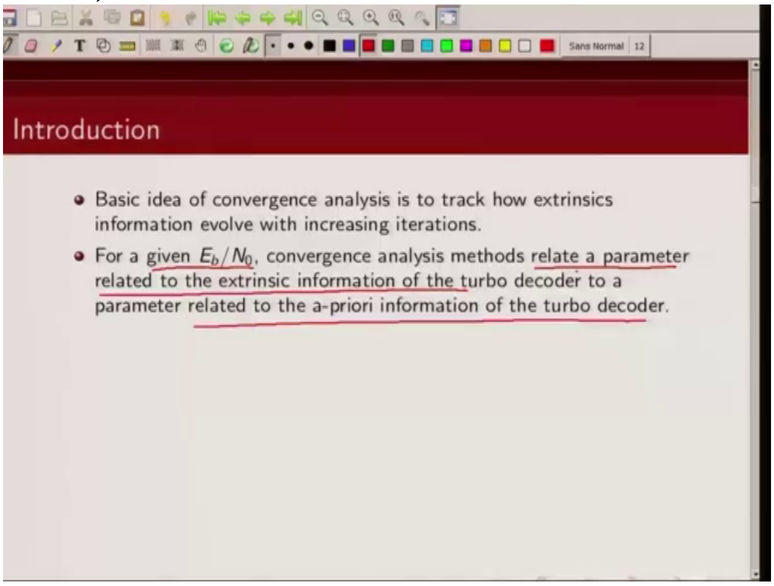


which is related to the a priori information. As

(Refer Slide Time 06:33)

I said in this soft input soft output decoder, only thing changing is this a priori information and this extrinsic information. So we want to track how these extrinsic information and a priori information are growing with iteration. So what we are going to do in this convergence analysis is we are going

(Refer Slide Time 06:56)

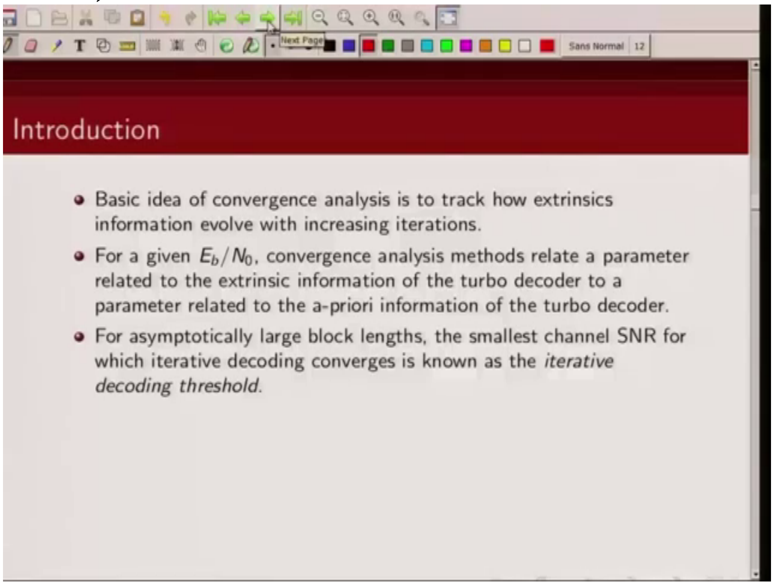


to track a parameter which is related to extrinsic

(Refer Slide Time 06:59)

information and we will see how that parameter will change when the parameter at the input side which is a priori value is also changed.

(Refer Slide Time 07:10)

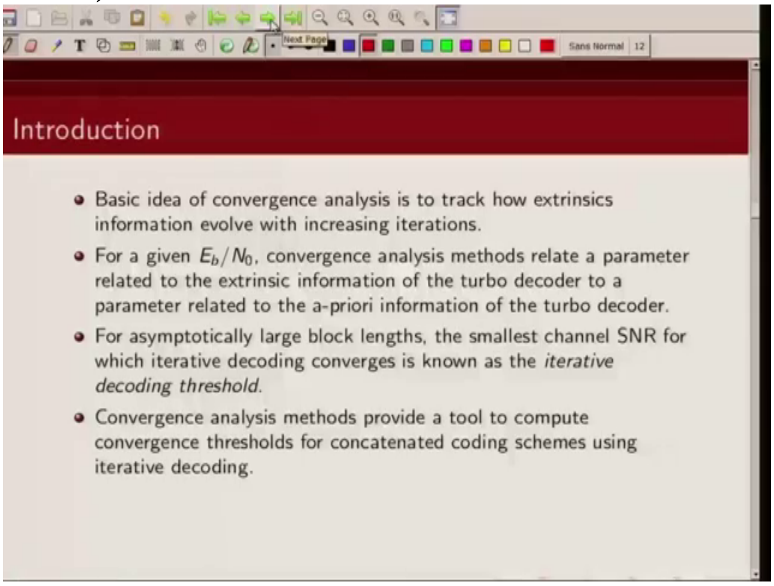


And for an asymptotically large block size the smallest channel S N R for which iterative decoding algorithm converges is known as decoding threshold. So this iterative

(Refer Slide Time 07:28)

decoding threshold will be away from your channel capacity, typically.

(Refer Slide Time 07:33)

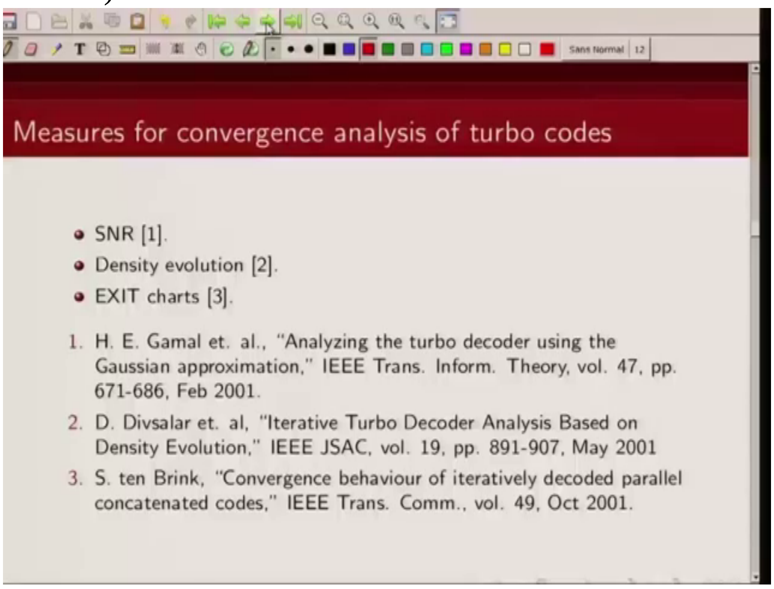


Now this convergence analysis tool is a very, very powerful tool to analyze these kinds of

(Refer Slide Time 07:44)

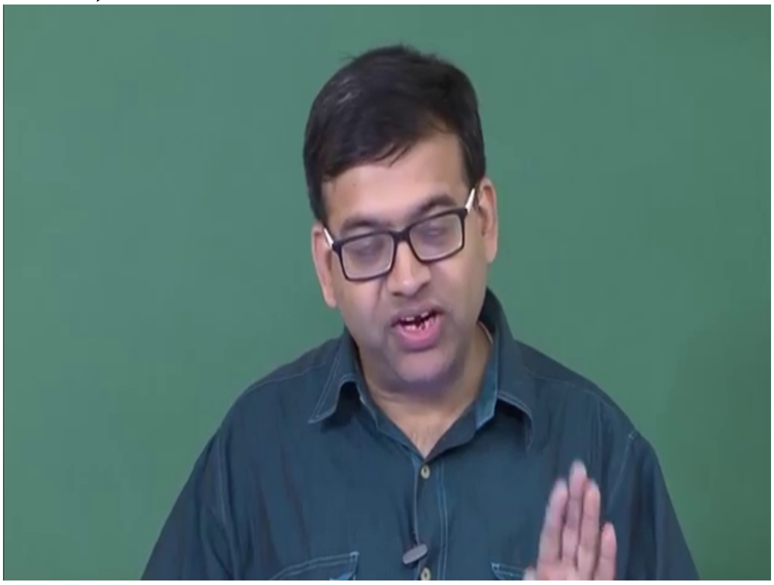
iterative decoding algorithms. It gives us tool to analyze the performance of concatenated schemes that use iterative decoding algorithm. It gives us tool to design our constituent encoders. It gives us tool to design our puncturing pattern, uh so it is a very, very interesting tool for analysis in the waterfall region.

(Refer Slide Time 08:07)



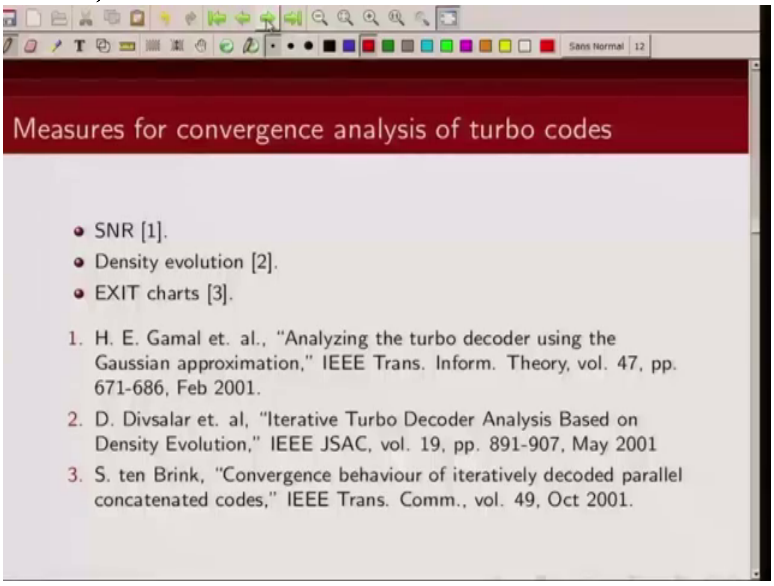
So as I said there are three

(Refer Slide Time 08:13)



popularly known techniques for

(Refer Slide Time 08:19)

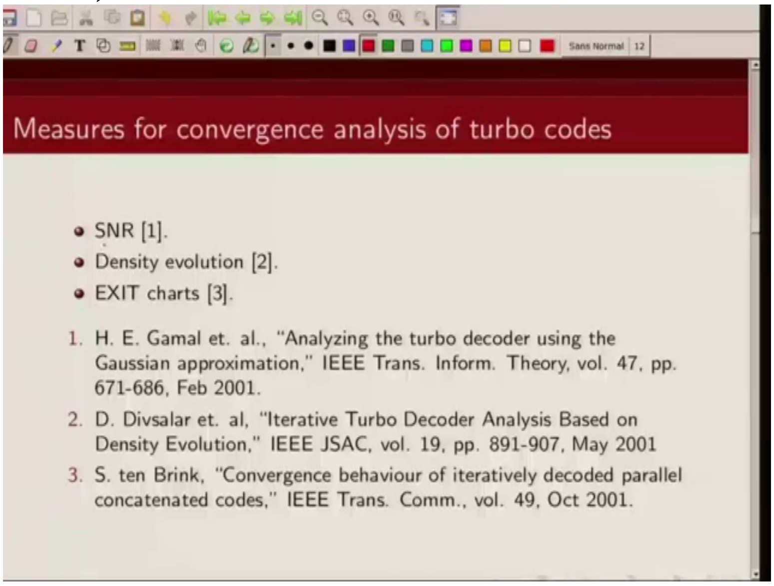


convergence analysis and as I said the idea

(Refer Slide Time 08:22)

of these techniques is track one parameter which is related to the extrinsic information and track the same parameter related to the a priori information. So this technique by El Gamal

(Refer Slide Time 08:35)

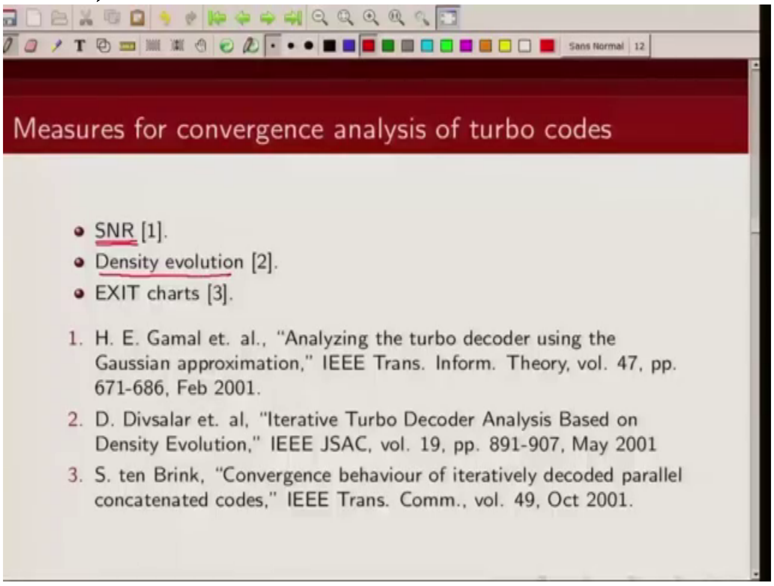


makes use of Gaussian approximation and it tracks the signal to noise ratio, so it tracks the signal to noise

(Refer Slide Time 08:41)

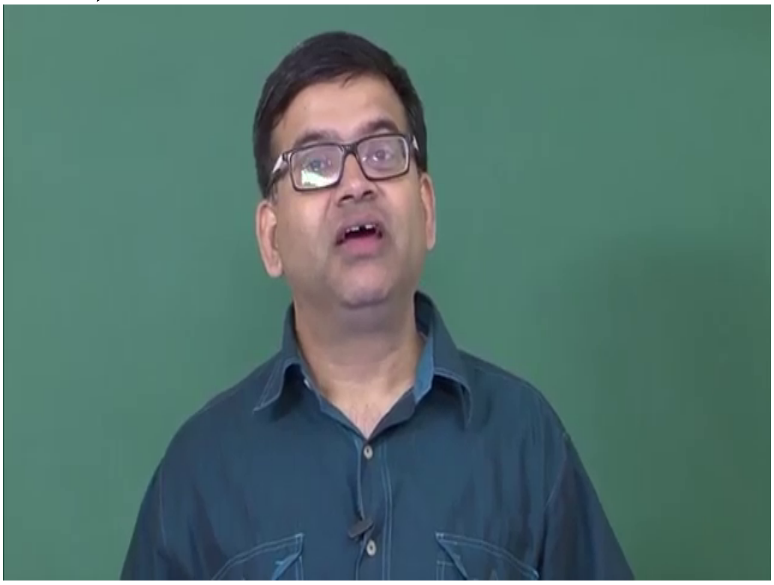
ratio of the extrinsic information and observes how this S N R extrinsic information grows when you change the S N R of the a priori information. In the density

(Refer Slide Time 09:00)



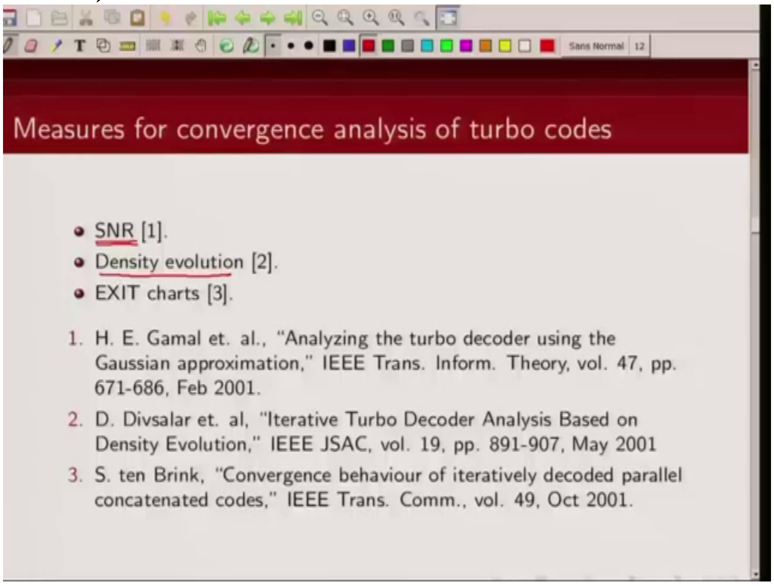
evolution method by Divsalar and others they actually see the

(Refer Slide Time 09:05)



density of this extrinsic information, how does it grow with iteration and this

(Refer Slide Time 09:12)

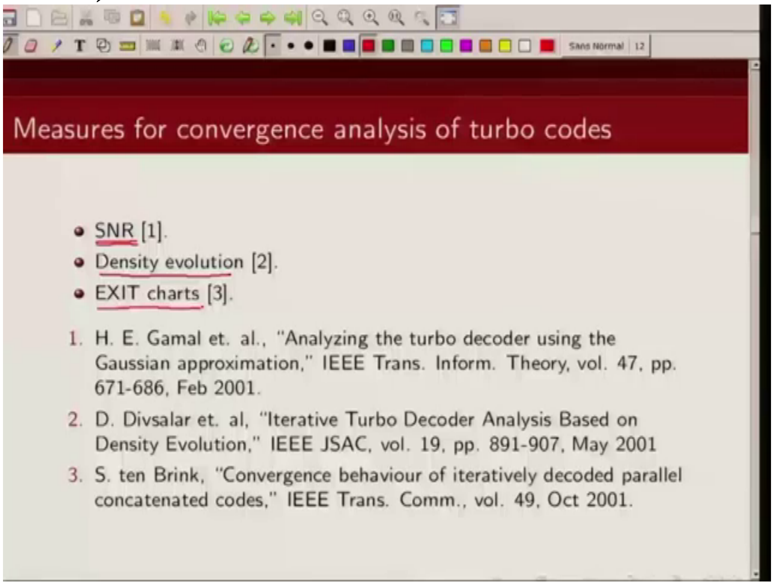


approach of ten Brink which is known as extrinsic information transfer chart, it

(Refer Slide Time 09:21)

uses mutual information as a parameter to

(Refer Slide Time 09:26)

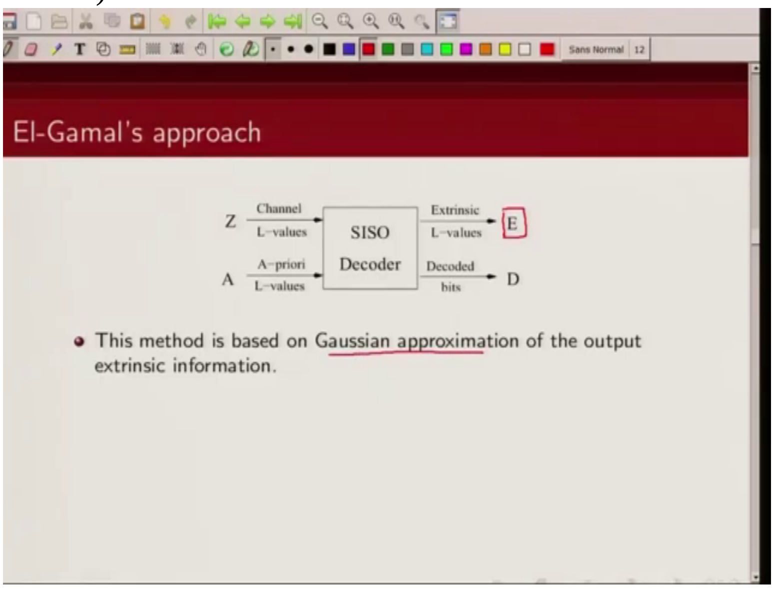


observe how, with iteration your extrinsic information is growing. And these are the three references, the first one corresponding to this S N R technique, the second one corresponding to this density evolution technique and third corresponds to this EXIT chart technique.

(Refer Slide Time 09:45)

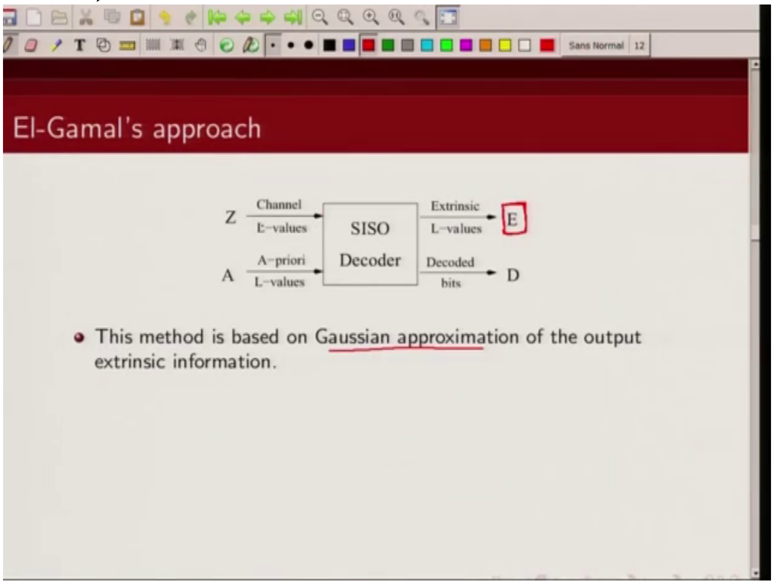
So the El Gamal approach is based on Gaussian

(Refer Slide Time 09:50)



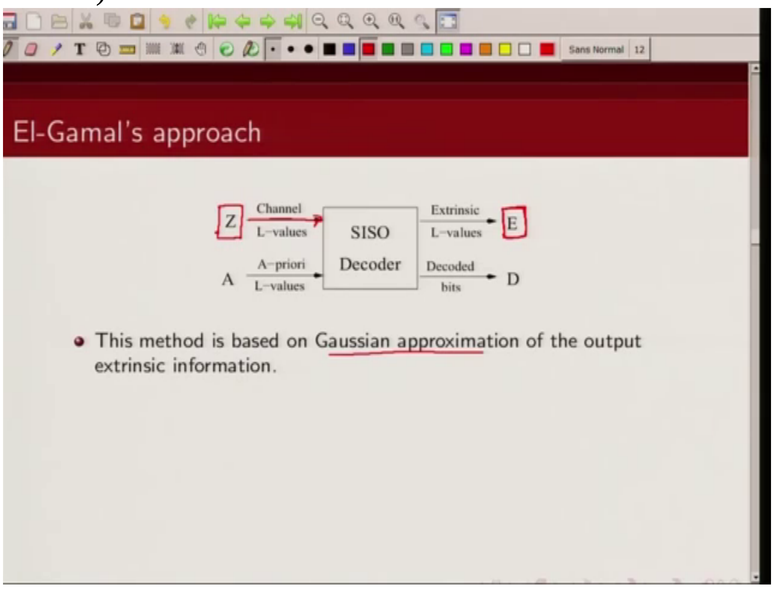
approximation of this output extrinsic information. So note,

(Refer Slide Time 09:57)



there are 2 inputs to my soft input soft output decoder; one which I am referring by Z

(Refer Slide Time 10:04)

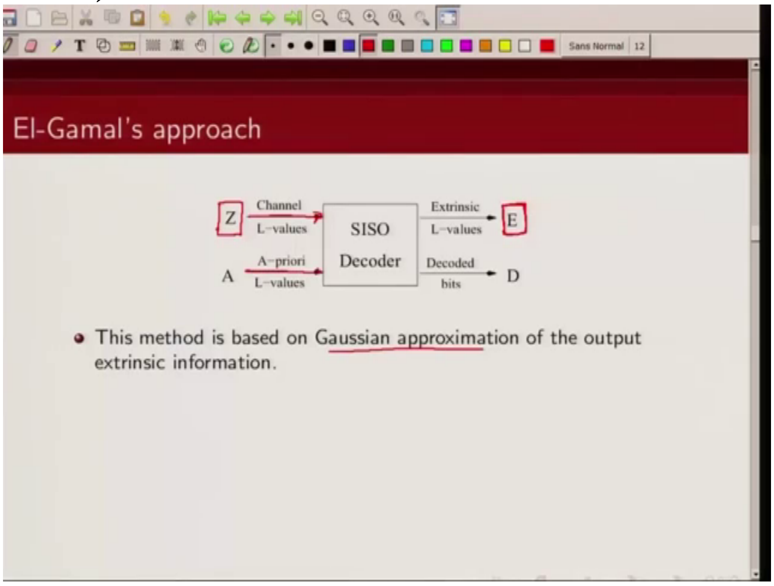


which is just channel received L

(Refer Slide Time 10:07)

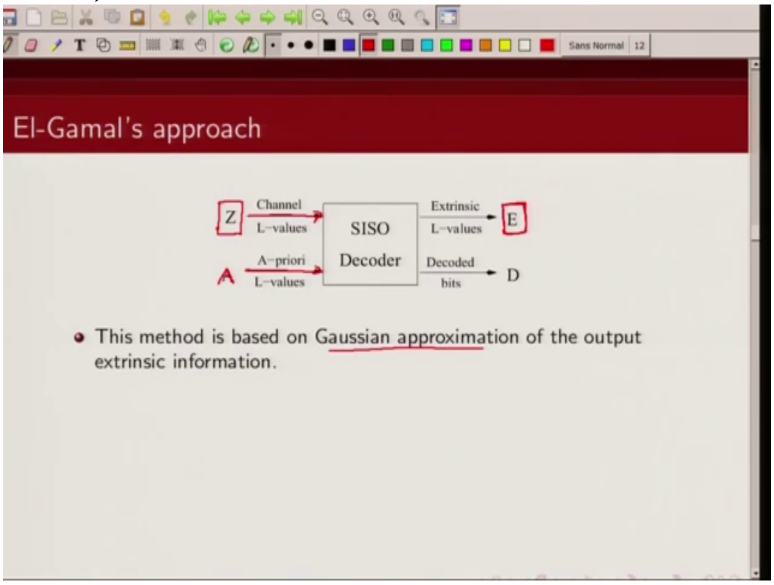
values. The second

(Refer Slide Time 10:08)



one is this a priori

(Refer Slide Time 10:11)

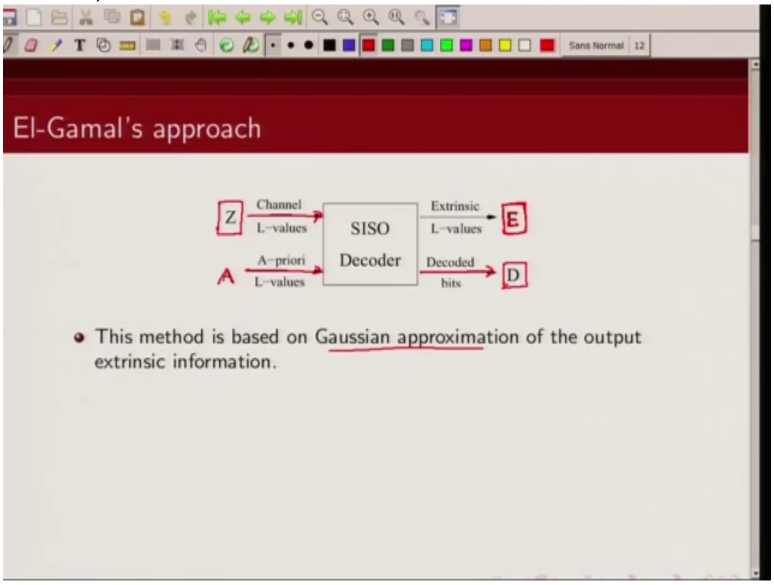


values and there are 2 outputs, one is this extrinsic information and other one is A P P L values, if I take a hard decision

(Refer Slide Time 10:22)

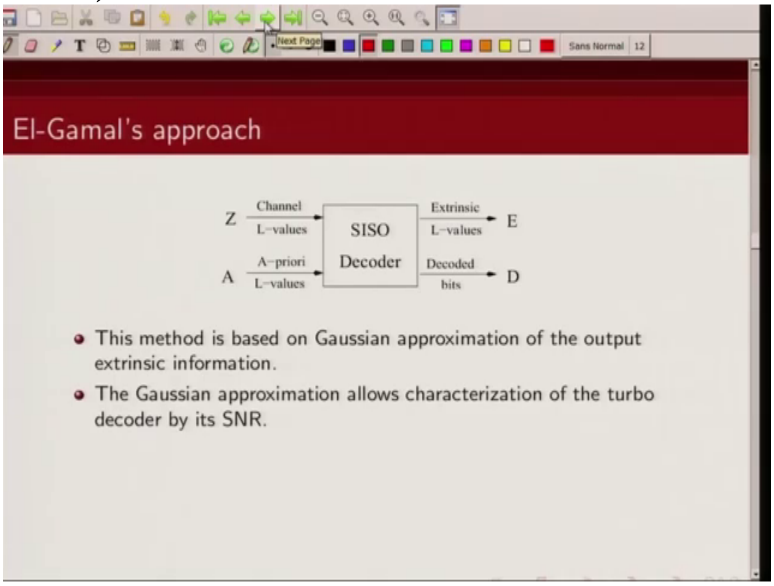
on that, what I get is my decoded bits.

(Refer Slide Time 10:26)



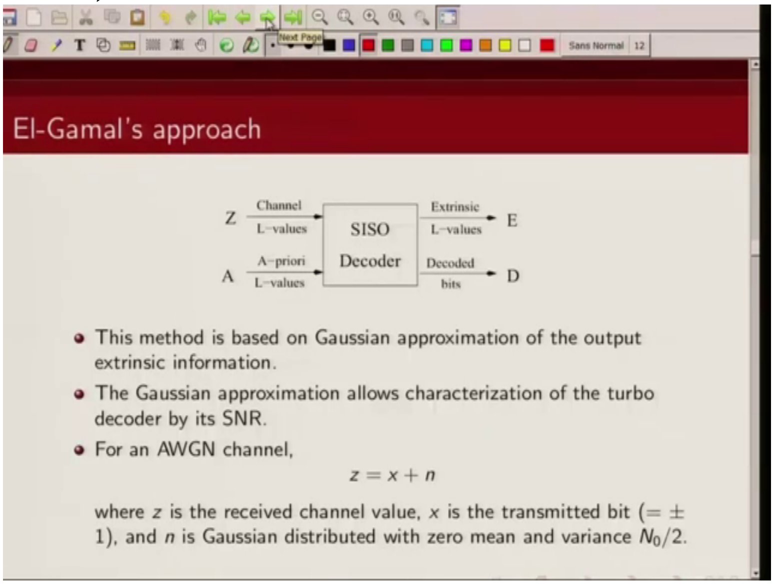
Now

(Refer Slide Time 10:28)



we are using this Gaussian

(Refer Slide Time 10:31)

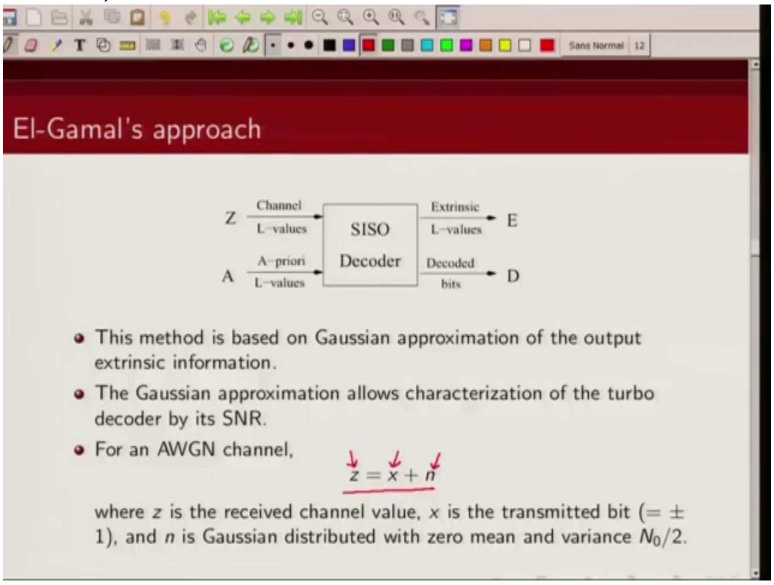


approximation so assume, so we have Gaussian channel. So if x was your modulated signal and n is my Gaussian noise, so what I receive is

(Refer Slide Time 10:42)

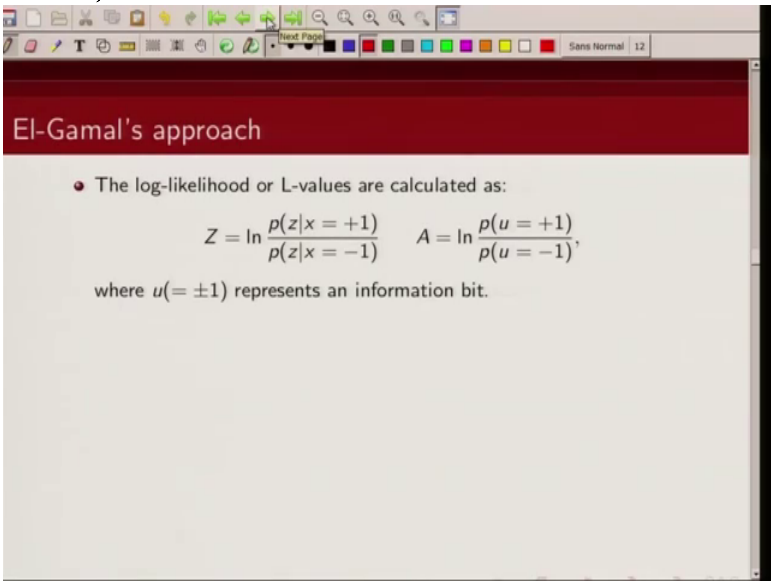
Z.

(Refer Slide Time 10:44)



Now

(Refer Slide Time 10:45)

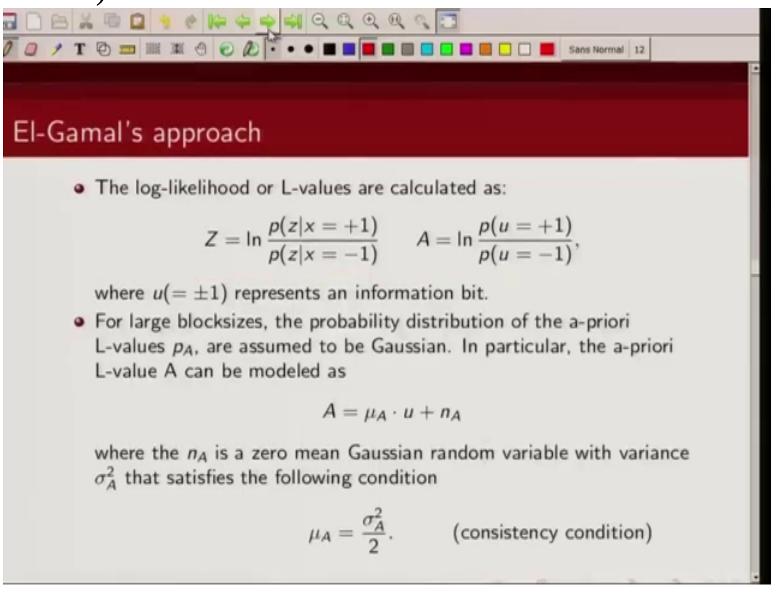


the likelihood ratio of Z we can write it like this, similarly this a priori information, the L value of that I can write

(Refer Slide Time 10:55)

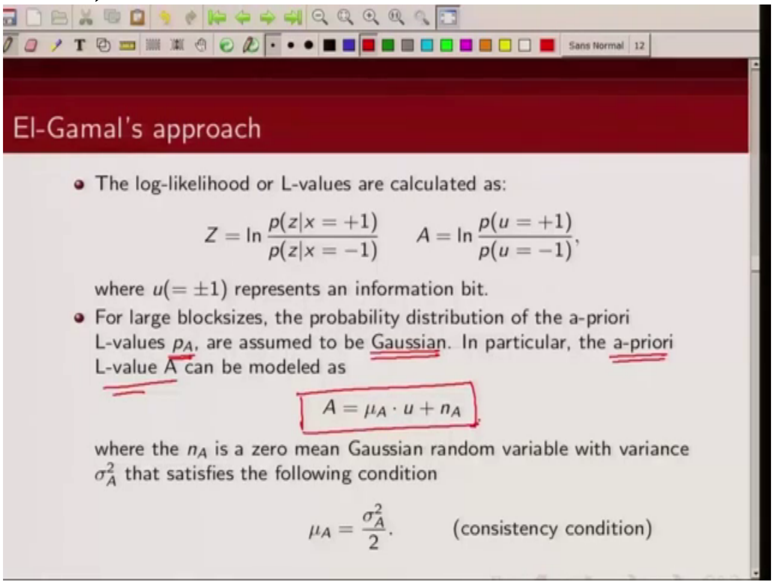
it like this.

(Refer Slide Time 10:56)



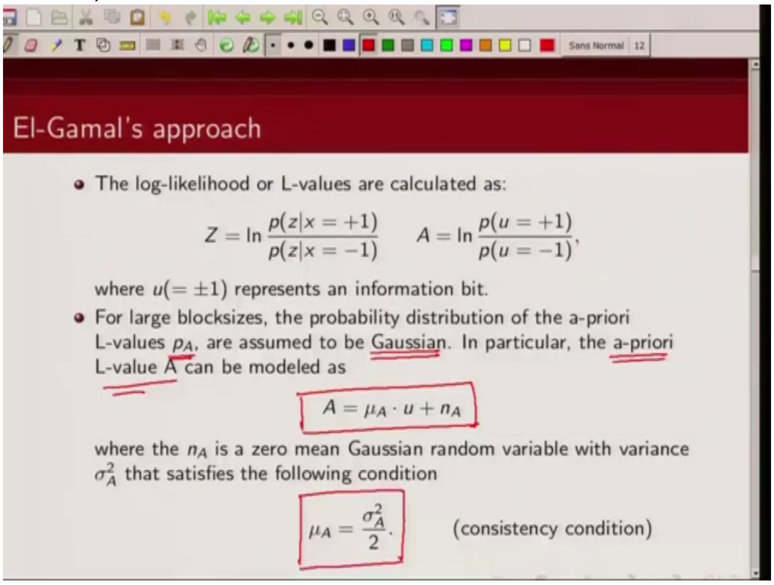
Now for large block sizes this a priori distribution is assumed to be Gaussian. So we model this a priori L value in this particular way in this

(Refer Slide Time 11:16)



El Gamal's approach. So in El Gamal's approach we modeled our a priori information as Gaussian and we generated like this, A is mu A times input plus some Gaussian noise and they have also observed what they call consistency condition.

(Refer Slide Time 11:38)

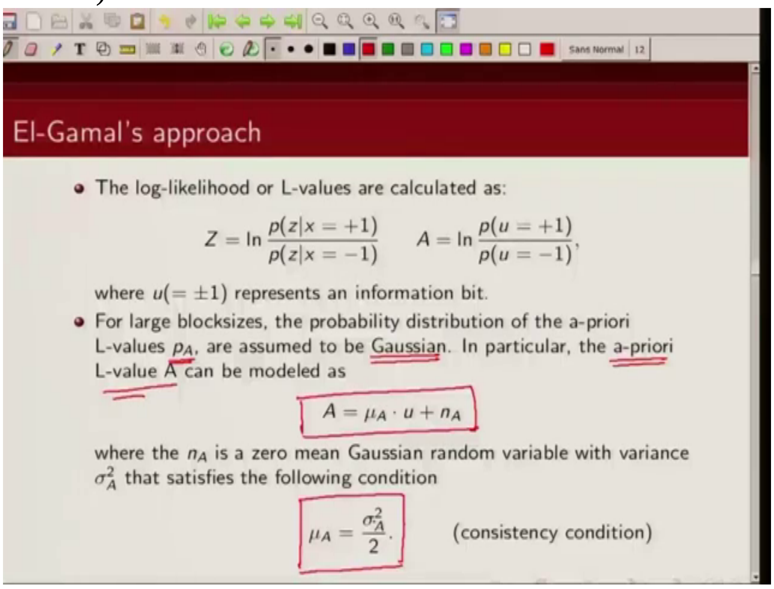


So they assume the mean and variance are related in this particular fashion. So what happens is if you make this Gaussian assumption and you make this assumption that mean and variance are related, then you essentially need to track only

(Refer Slide Time 11:55)

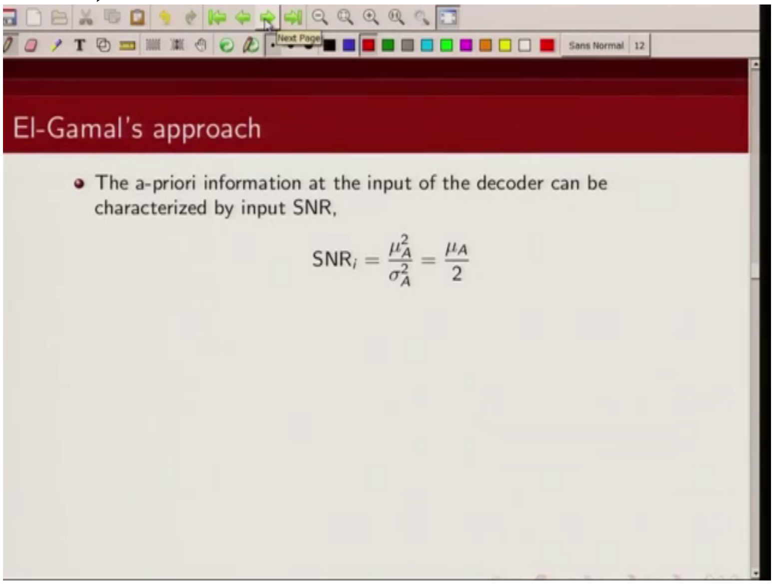
one parameter. So you, for example, with just the mean you can track your

(Refer Slide Time 12:03)



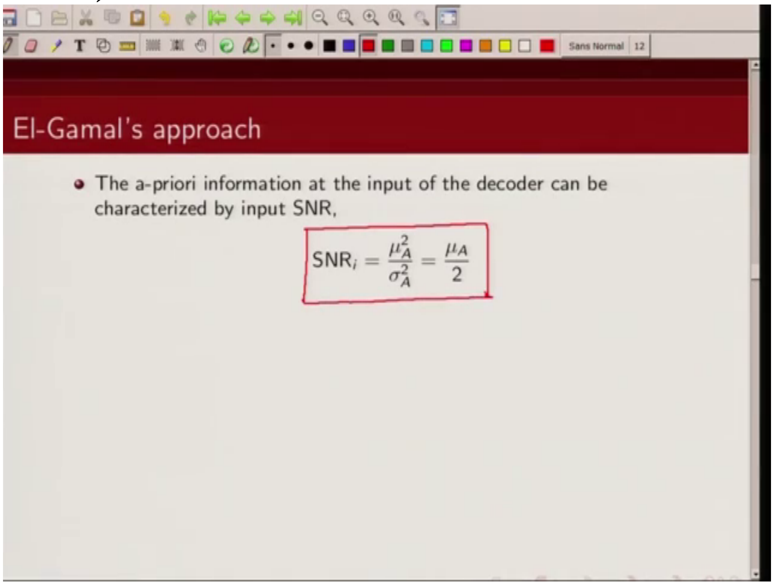
Gaussian distribution because mean and variance are related.

(Refer Slide Time 12:07)



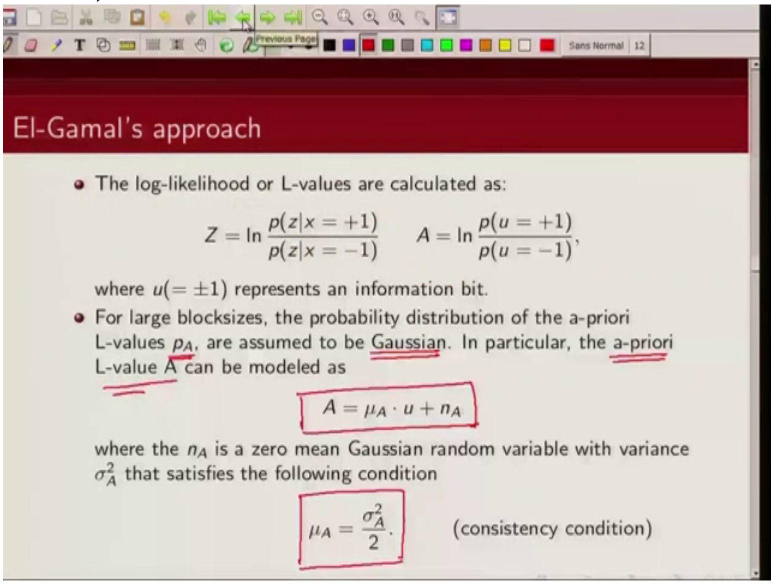
Now similarly we can define input S N R of the a priori information.

(Refer Slide Time 12:14)



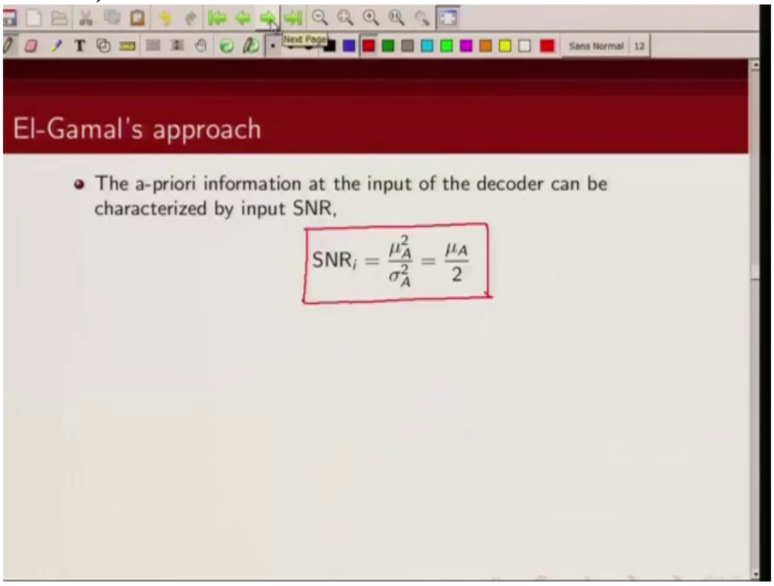
This is mu A square by sigma square. Now sigma square by 2

(Refer Slide Time 12:19)



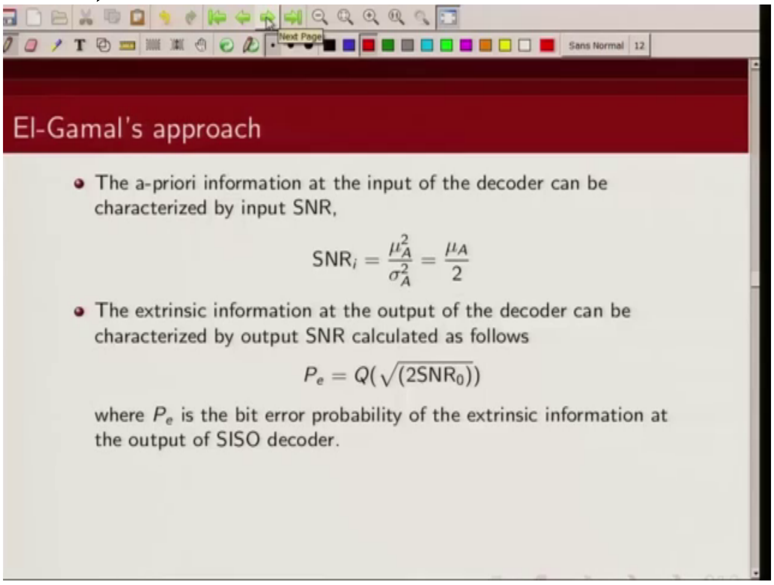
is mu A. So our

(Refer Slide Time 12:21)



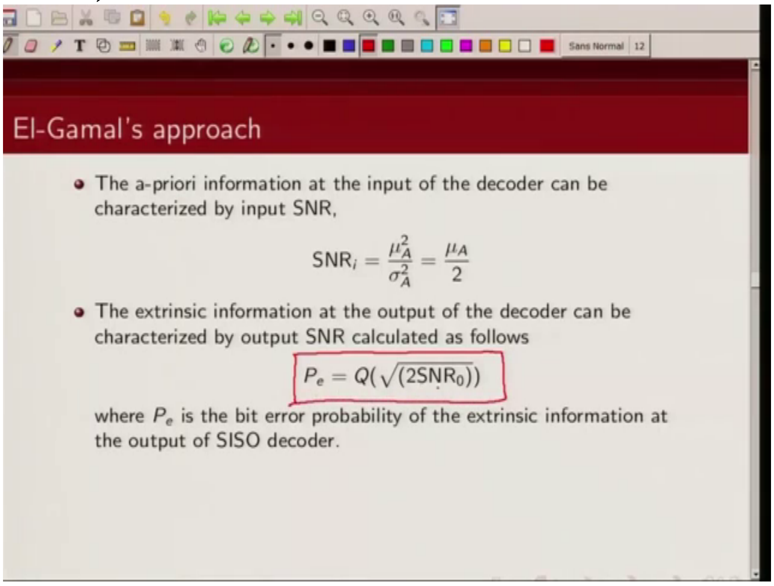
input S N R is given by the mean of the a priori information divided by 2.

(Refer Slide Time 12:32)



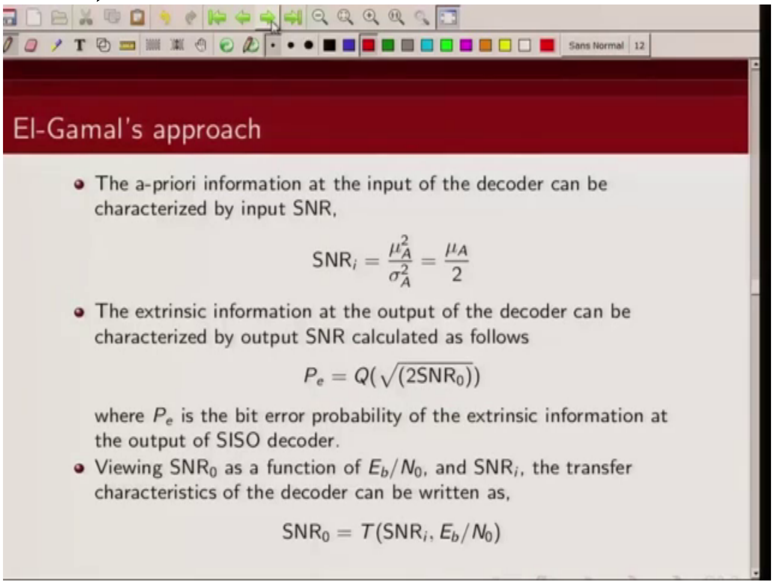
And since our output is approximated as Gaussian, so we can calculate the output probability of error as a function of

(Refer Slide Time 12:43)



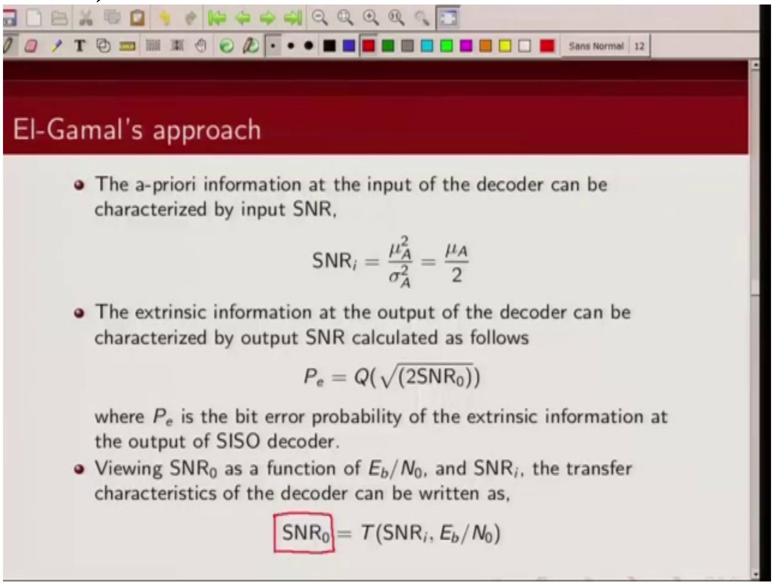
output S N R and they are related to the, using this Q function. Now,

(Refer Slide Time 12:53)



so what we can do is we can write this output S N R

(Refer Slide Time 13:00)

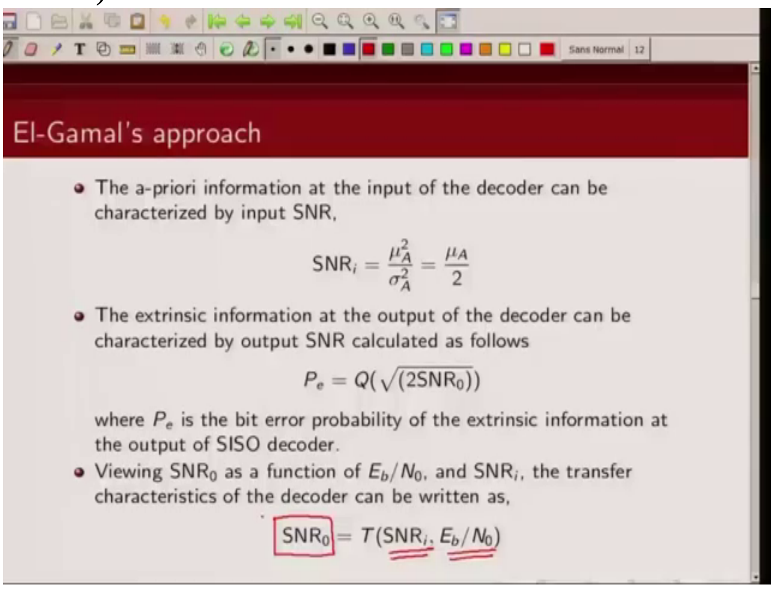


in terms of input S N R and our operating signal to noise ratio. So what we can do is we can view the output S N R of the extrinsic

(Refer Slide Time 13:14)

information as a function of input S N R of a priori information as well as the channel operating signal to noise ratio. So

(Refer Slide Time 13:27)

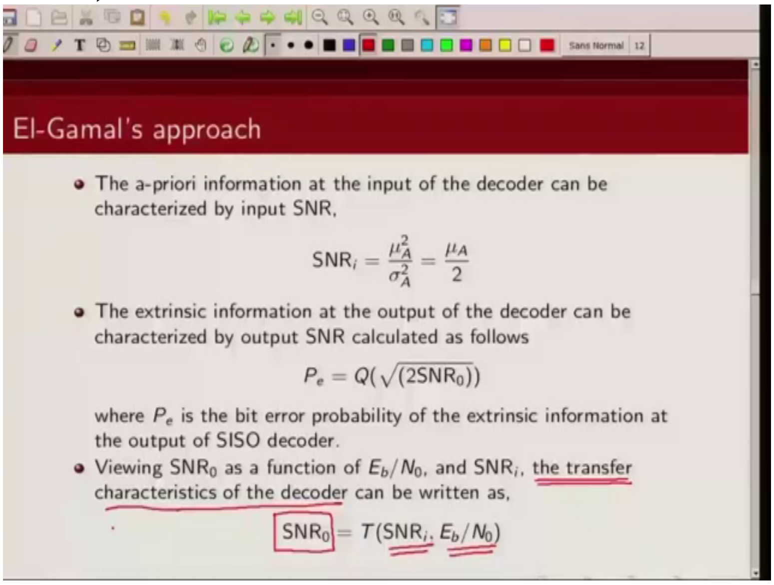


this is crucial, so this is basically what I call the transfer characteristics of the decoder. Because my decoder is a function of

(Refer Slide Time 13:41)

a priori inputs as well as channel received values. Now channel received value is the function of channel operating S N R and what I get, a priori information is the function of a priori input S N R. So I can view S N R of the extrinsic information, I can

(Refer Slide Time 14:02)

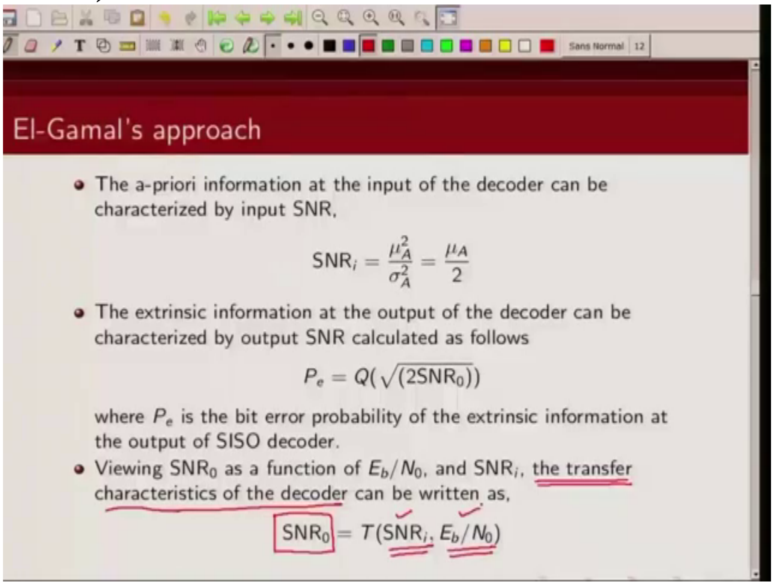


view it as a function of input S N R of a priori values as well as channel, operating channel signal to noise ratio. So this relation characterizes how my decoder will behave. Because remember with iteration your extrinsic information is changing as a function of

(Refer Slide Time 14:24)

a priori value and what is your operating channel S N R. So this transfer function will give

(Refer Slide Time 14:30)

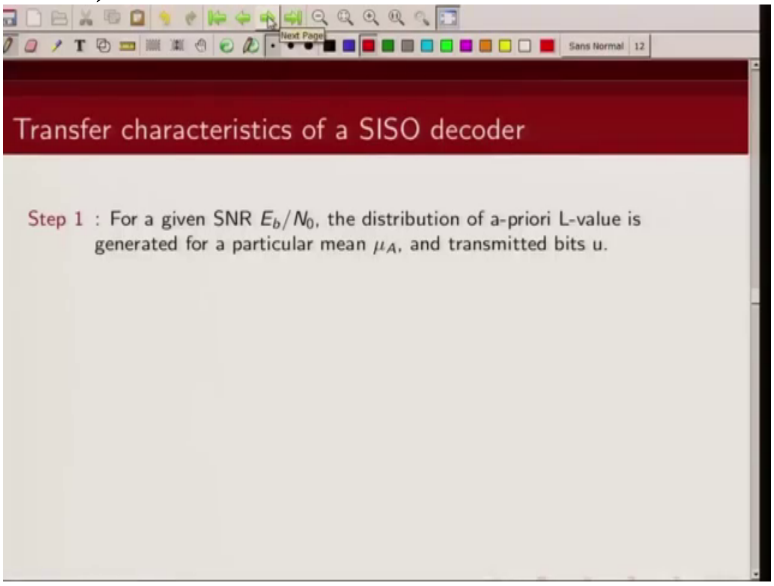


me how my decoder, this soft input soft output

(Refer Slide Time 14:35)

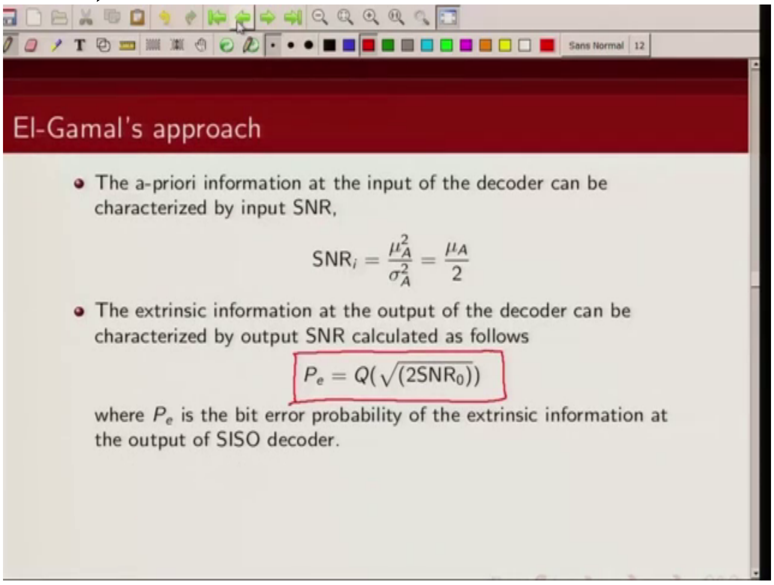
decoder, how it will perform as a function of a priori value and the channel operating S N R.

(Refer Slide Time 14:44)



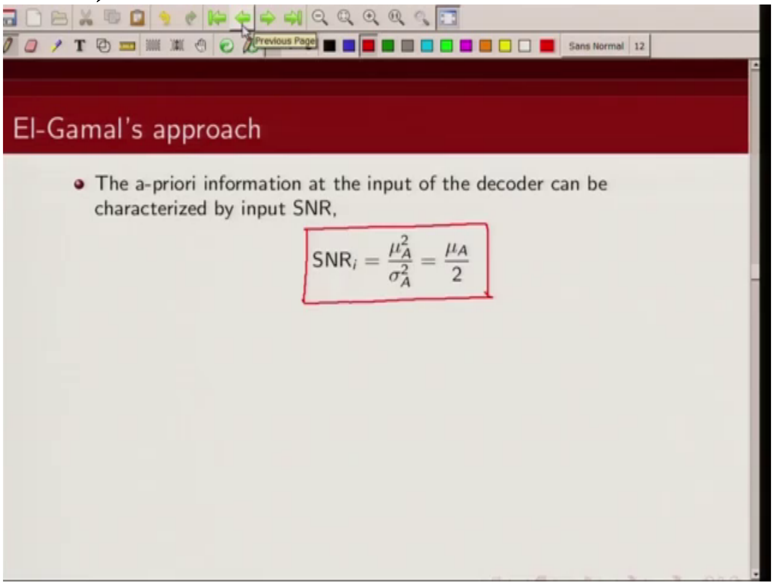
So then how do we draw the transfer characteristics? For a given signal to noise ratio, the distribution of a priori L values is generated for a particular mean mu a and transmitted bit u. How?

(Refer Slide Time 15:03)



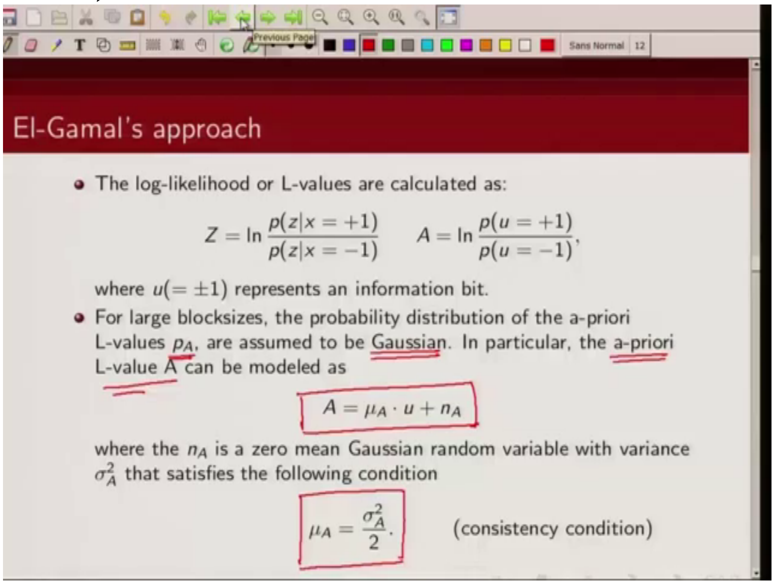
We know that we are modeling

(Refer Slide Time 15:05)



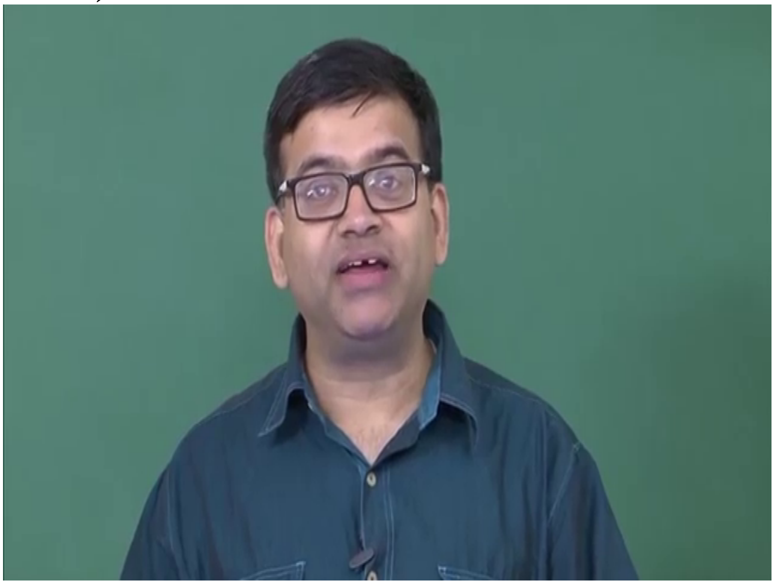
our a priori

(Refer Slide Time 15:06)



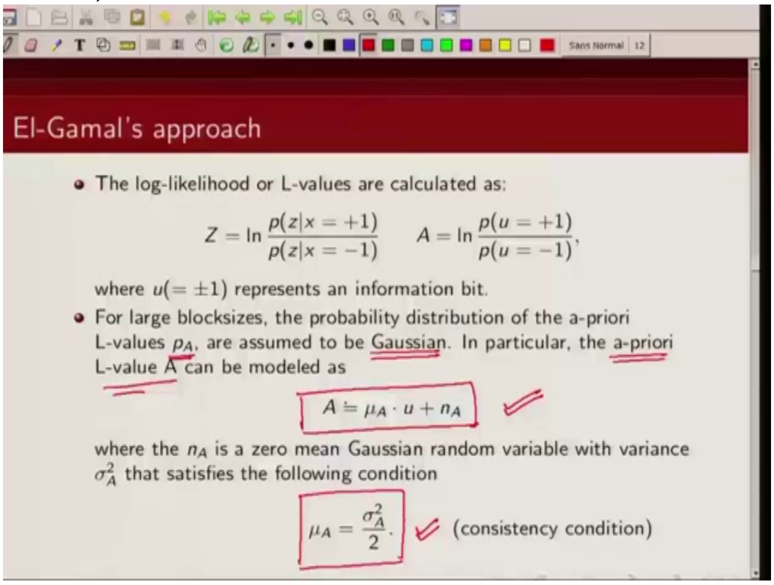
information like this. And of course we are assuming consistency condition so the mean and

(Refer Slide Time 15:14)



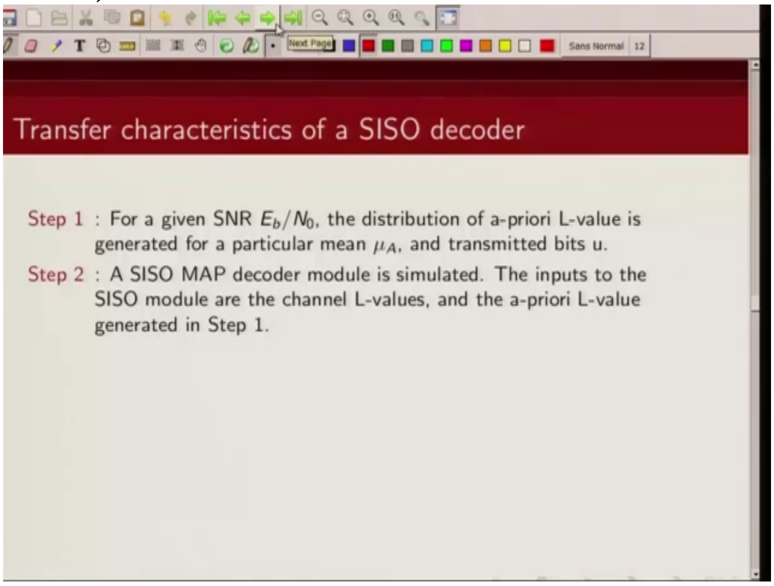
variance of the mutual, the a priori information is related like this.

(Refer Slide Time 15:22)



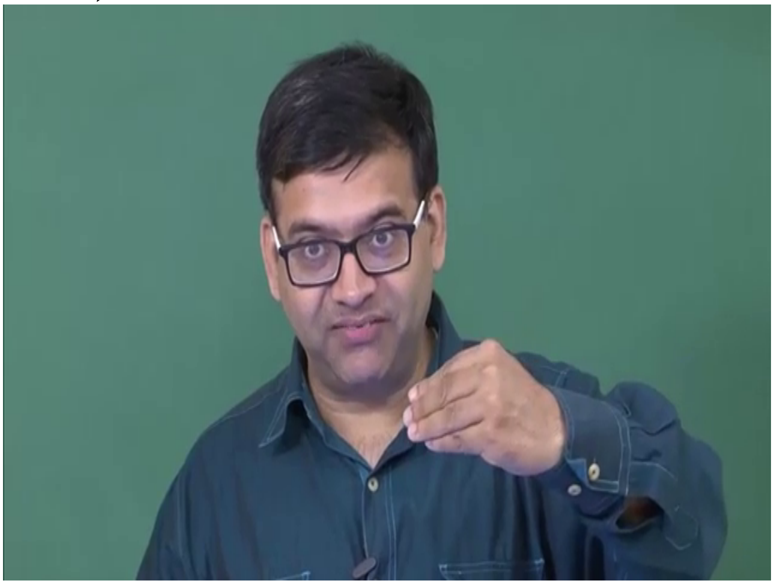
So next

(Refer Slide Time 15:24)



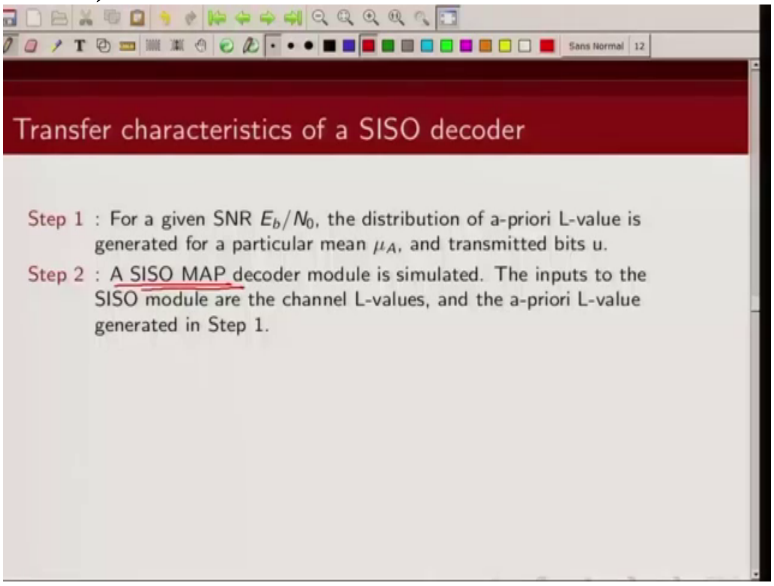
step is we simulate a soft input soft output decoder. So we feed in these two input. One is this channel received

(Refer Slide Time 15:33)



S N R and other is this a priori information which

(Refer Slide Time 15:38)

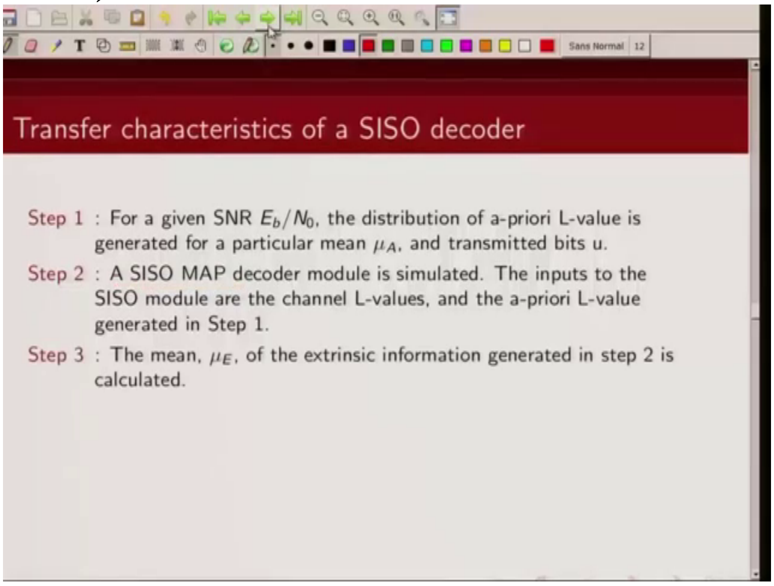


we modeled as Gaussian. We feed these two inputs to the decoder and what comes out as output

(Refer Slide Time 15:47)

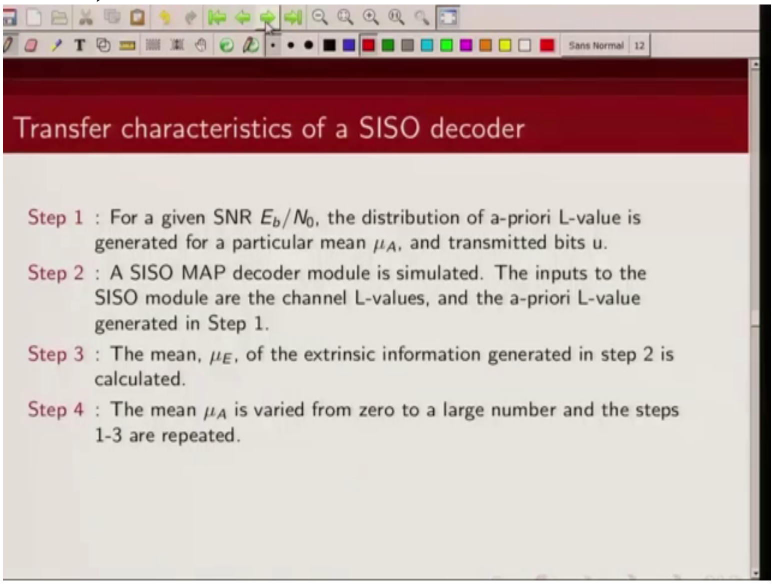
are these extrinsic values.

(Refer Slide Time 15:50)



And we compute the mean of the extrinsic values.

(Refer Slide Time 15:58)

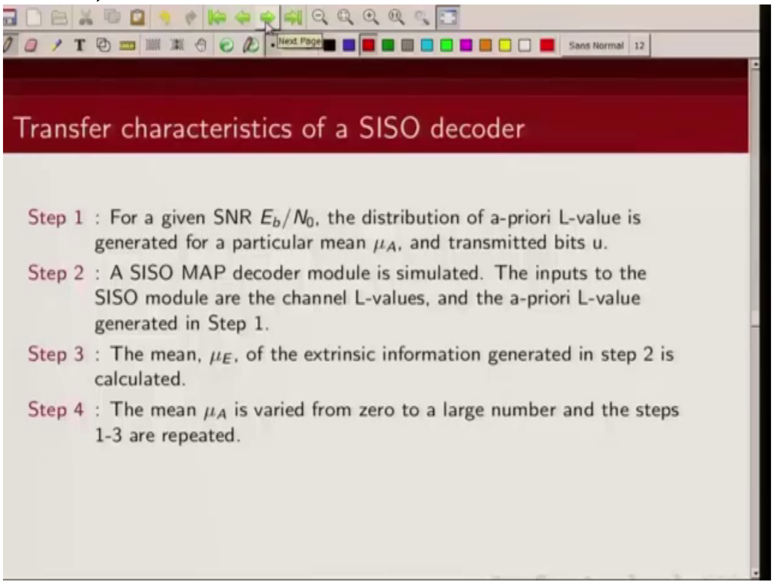


Now we know that our signal to noise ratio, because we are making

(Refer Slide Time 16:04)

Gaussian assumption, our signal to noise ratio is related to the mean. Now as I said with iteration, my a priori information is changing. So now we are going to

(Refer Slide Time 16:16)

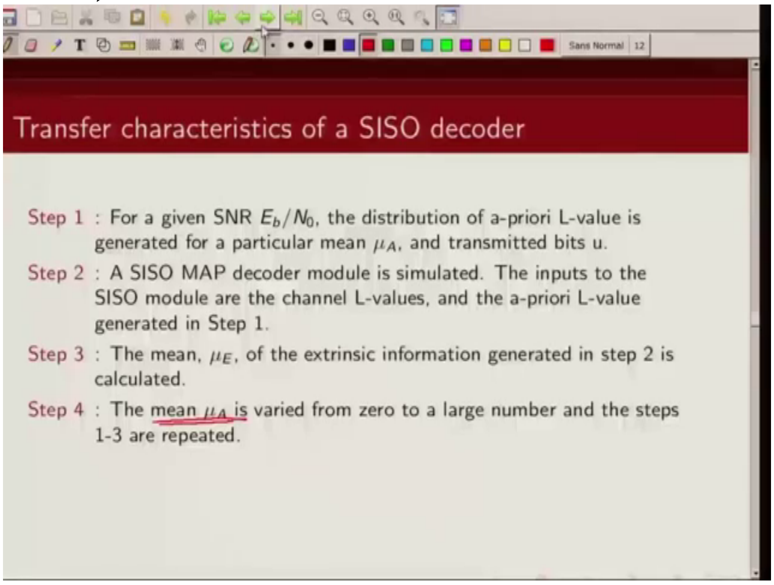


change the mean of the a priori information. And then we will again simulate

(Refer Slide Time 16:23)

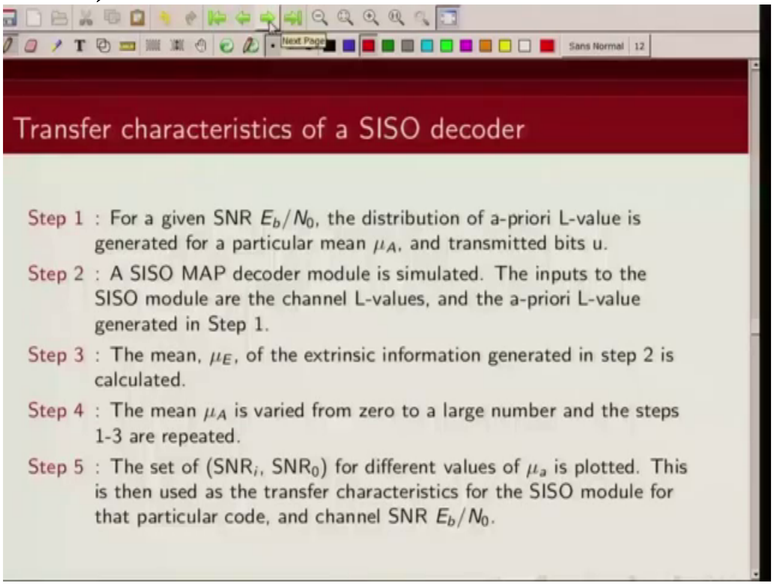
this soft input soft output decoder and we will try to see what happens to the extrinsic information mean. How much it is growing with change in input a priori information mean?

(Refer Slide Time 16:41)



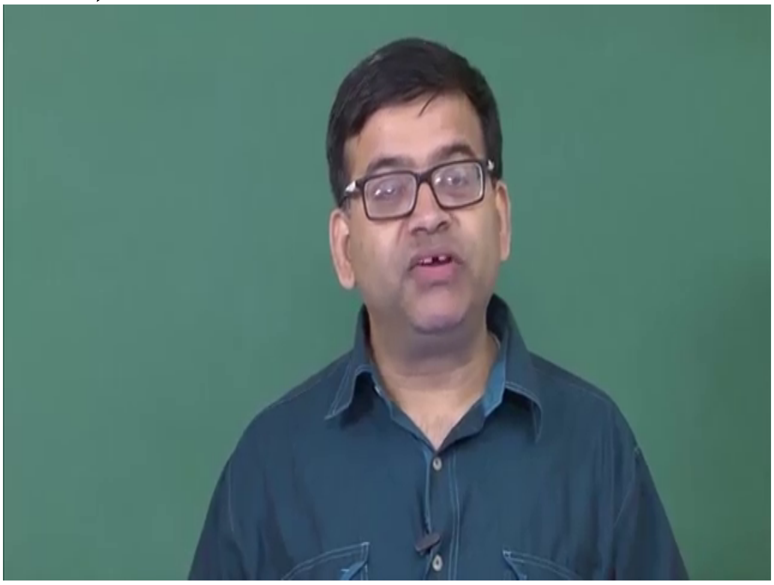
So this process is done. So we repeat this by varying our a priori information mean.

(Refer Slide Time 16:53)



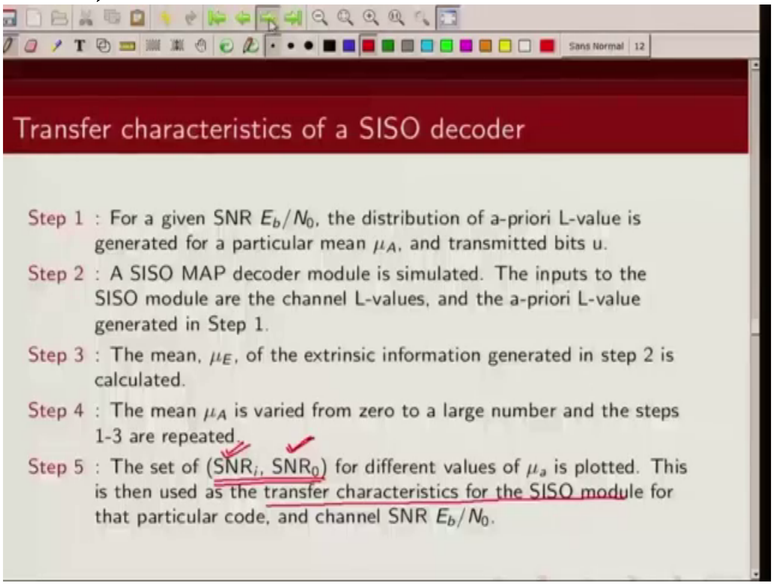
And finally what we do, we plot this input output relation for a particular channel S N R. So this is my input a priori S N R, this is the extrinsic information S N R. We plot it for a particular value of signal to noise ratio and this is my transfer characteristic for that particular decoder

(Refer Slide Time 17:23)

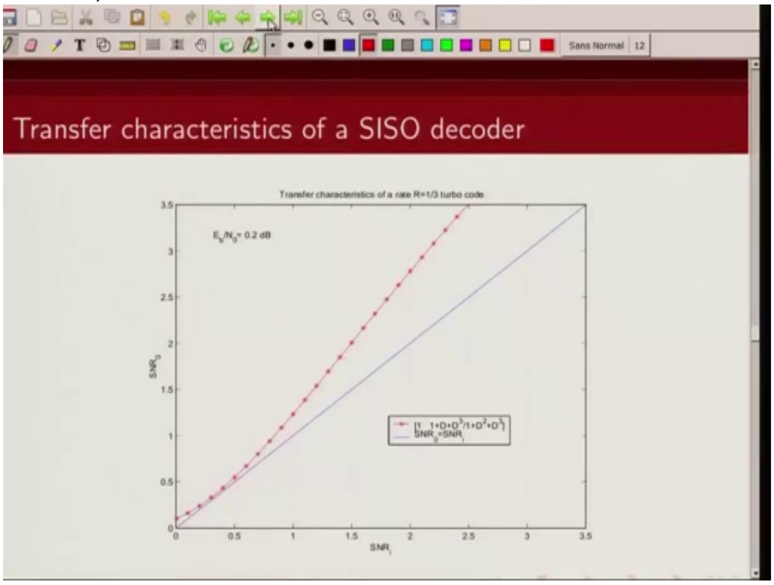


which is a function of channel operating S N R and of course it is the function of the constituent encoders that I have used.

(Refer Slide Time 17:31)

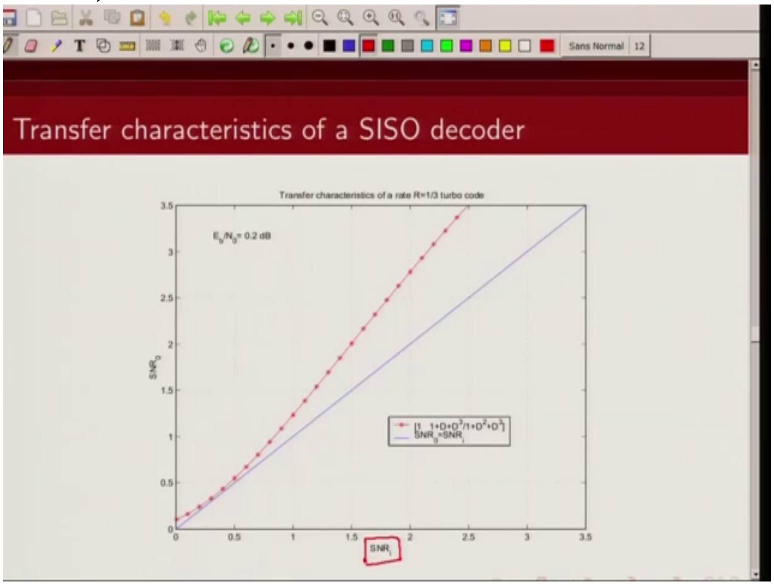


(Refer Slide Time 17:31)



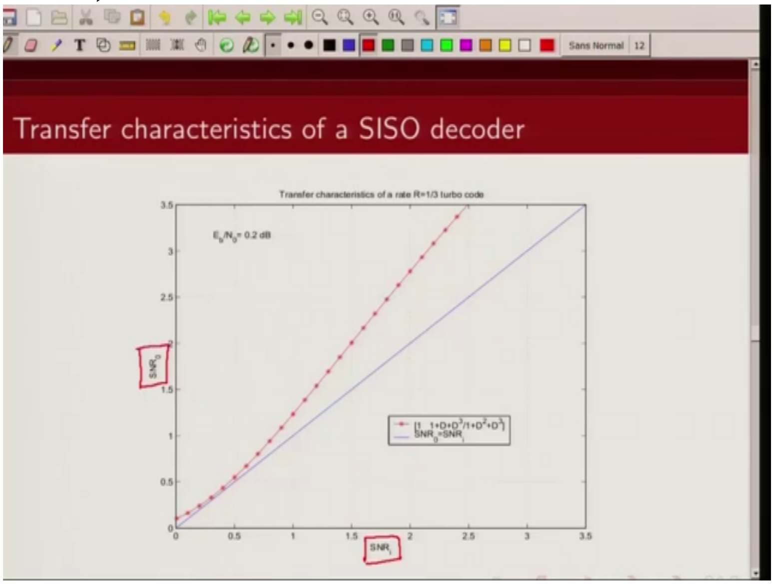
So here basically I have plotted, with red curve I have plotted transfer characteristics of one such code. It is a 8 state code. What I have here at the input side is

(Refer Slide Time 17:48)



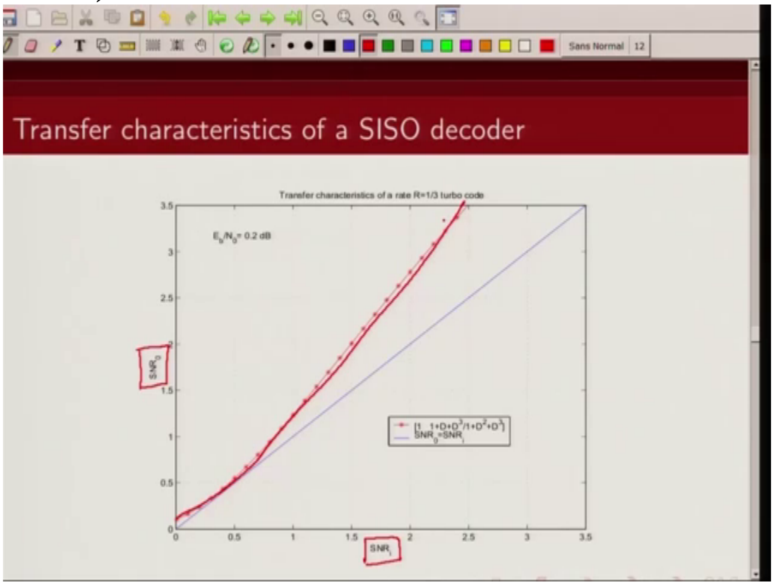
S N R of a priori information and what I have here on the output side is

(Refer Slide Time 17:55)



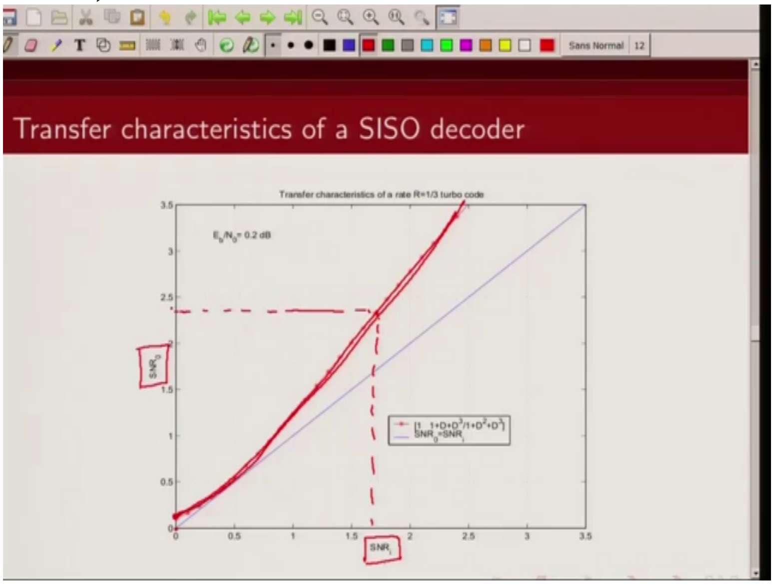
S N R of the extrinsic information. And this is how my; so initially

(Refer Slide Time 18:02)



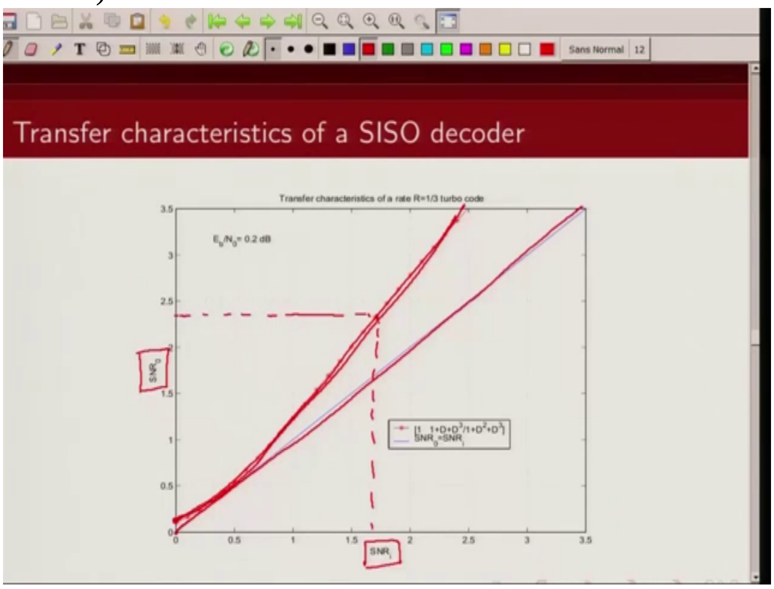
I don't have any a priori knowledge, the extrinsic information will, this is the amount of extrinsic information which is generated. So this transfer characteristics will tell me, if I have a particular input a priori information then what is the corresponding

(Refer Slide Time 18:20)



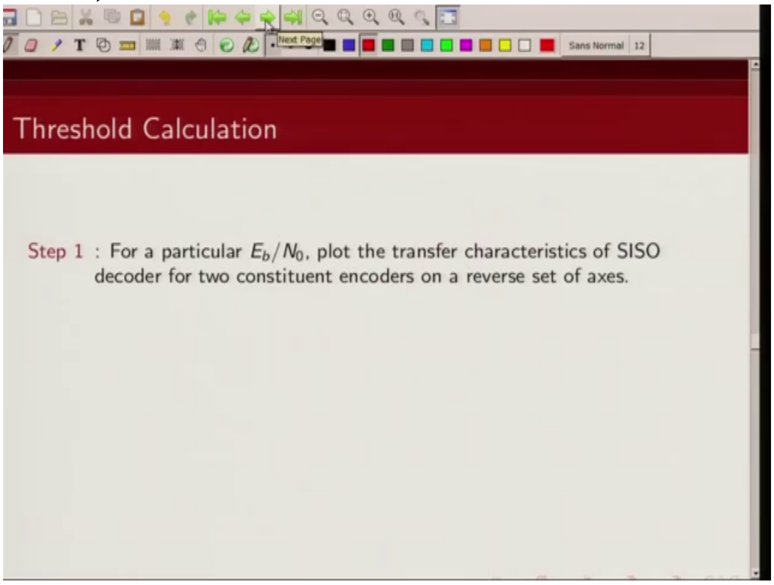
extrinsic information S N R. And for comparison sake I have drawn this line which is the S N R in

(Refer Slide Time 18:28)



equal to S N R out. Now if you have a symmetric turbo code, you obviously would like your transfer characteristics to be above this line.

(Refer Slide Time 18:41)

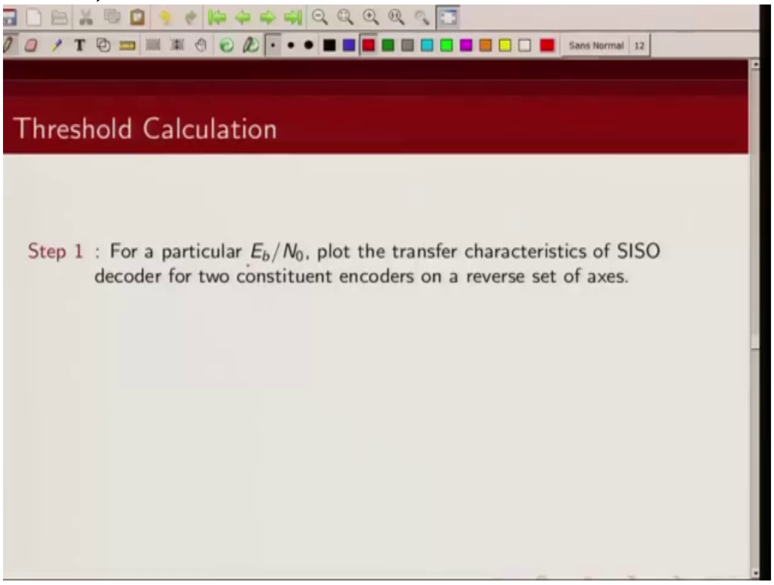


Now how do we compute, how do we use these

(Refer Slide Time 18:46)

transfer characteristics to compute the decoding threshold? So how do we find out the S N R, minimum S N R under which our iterative algorithm will converge? For that we need to do this threshold computation. So how do we do this threshold computation? So for a

(Refer Slide Time 19:08)

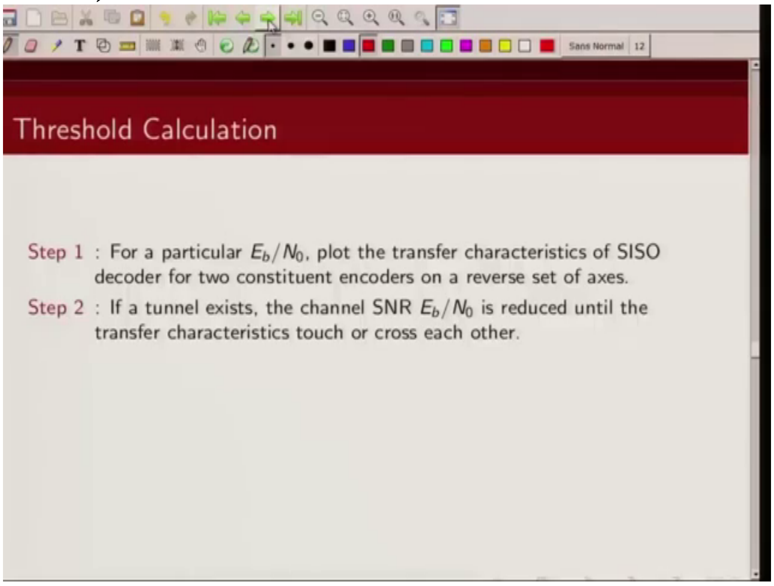


particular signal to noise ratio, we plot the transfer characteristics of this soft input soft output decoder. We plot them on reverse set of axes. Now what do I mean by reverse set of axes? So for the first, my S N R in is on x axis, and

(Refer Slide Time 19:30)

S N R out is on the y axis. Now for the second decoder, my S N R in is on the y axis and S N R out is on the x axis. Now why do I do this? Because the extrinsic information of first decoder is input to the second decoder. So S N R out of the first decoder becomes S N R in of the second decoder. And that's why I put the S N R in of the second decoder as y axis and the S N R out of the second decoder is S N R in for the first decoder because the extrinsic information from the second decoder is coming as input to the, as a priori input to the first decoder. And that is the reason I plot these transfer characteristics on reverse axes.

(Refer Slide Time 20:35)

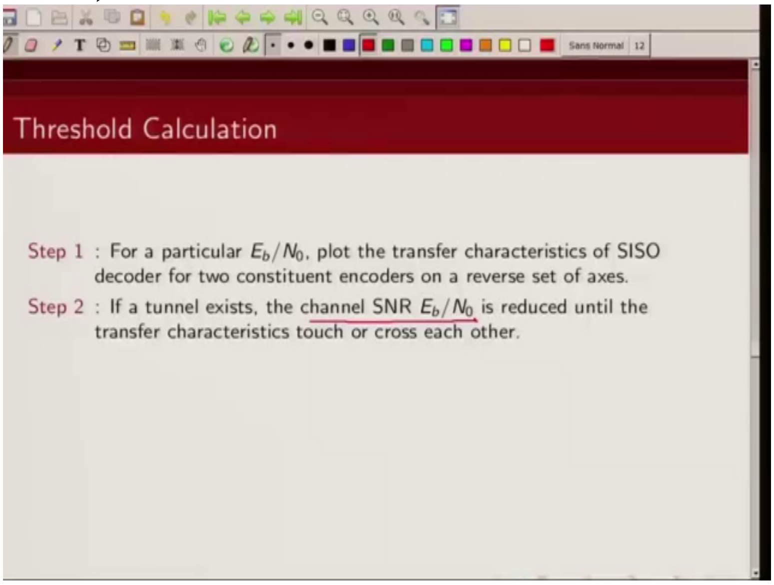


Now if these transfer characteristics do not cross, there is a tunnel in the sense they do not touch each other, then what we do is the channel, operating channel S N R is reduced until these transfer characteristics just about touch.

(Refer Slide Time 21:01)

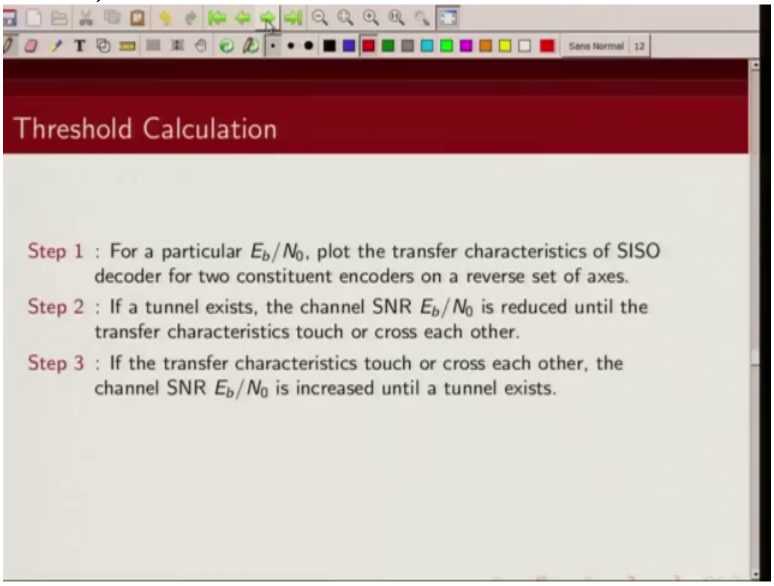
So what is the effect of channel S N R? So as you reduce the channel S N Rs these transfer characteristics which have been plot on reverse axes, they come closer when you reduce the channel S N R. So the smallest S N R for which there is still a tunnel, that's your decoding threshold for that particular

(Refer Slide Time 21:28)



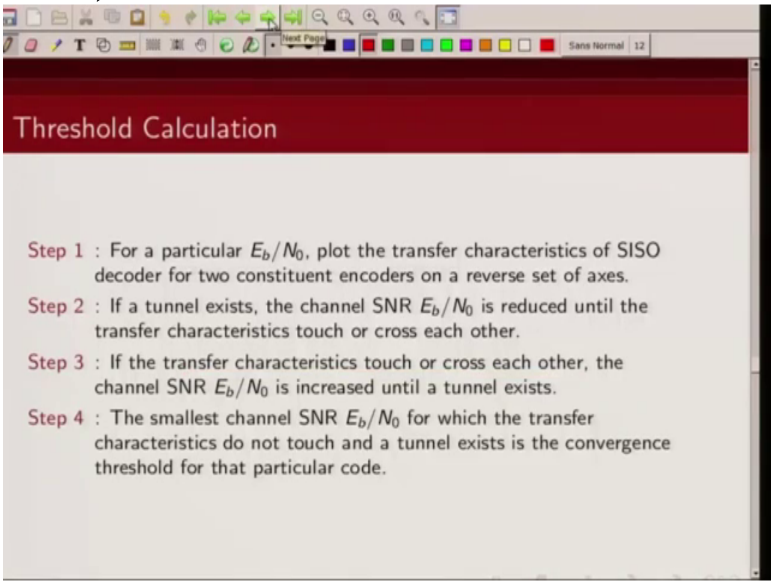
code.

(Refer Slide Time 21:29)



So if the transfer characteristics touch or cross each other, what we need to do is we need to increase the S N R until there is a tunnel, still a tunnel.

(Refer Slide Time 21:41)

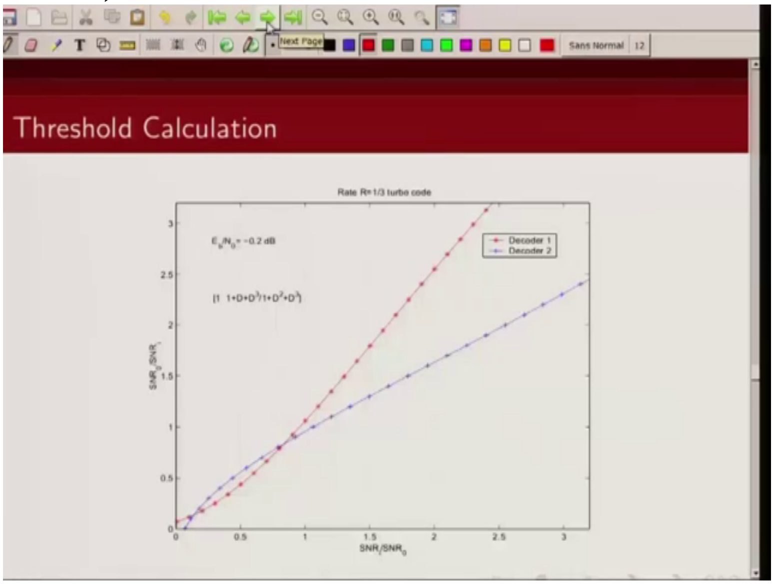


So the smallest channel S N R for which these two transfer characteristics which have been plotted on reverse axes, they do not touch and a tunnel exist is basically the convergence threshold for that particular code. So that would give the S N R, minimum S N R under which that particular code will converge and it will have a

(Refer Slide Time 22:11)

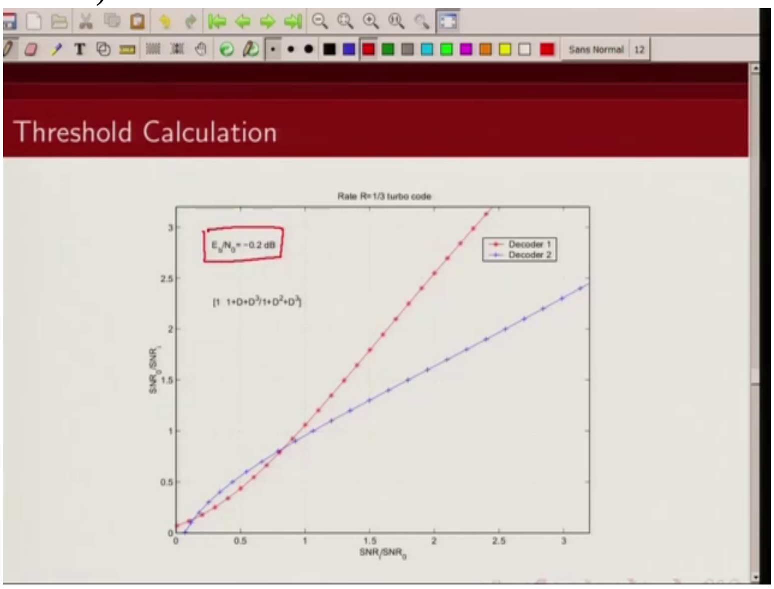
waterfall kind of behavior if you take large enough block size.

(Refer Slide Time 22:16)



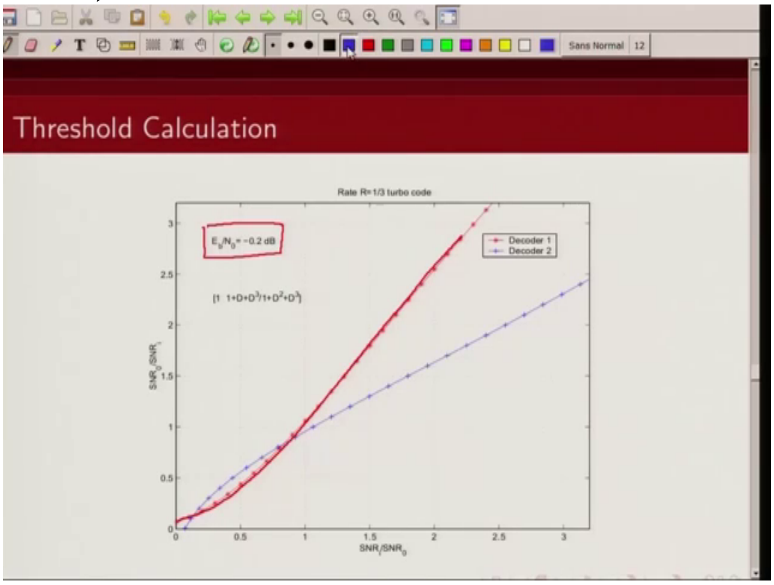
This is one example. Now note here, this is plotted for channel operating

(Refer Slide Time 22:25)



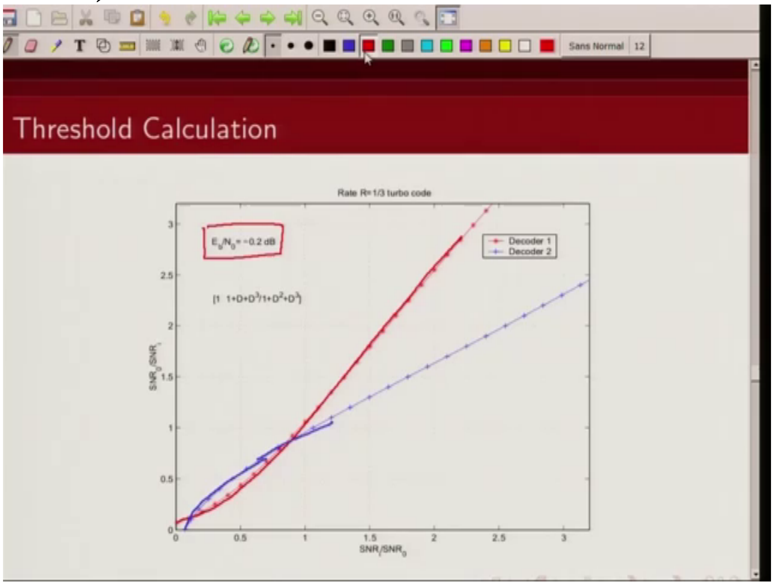
S N R of minus point 2 d B so this is, in red curve is my decoder 1 and

(Refer Slide Time 22:34)



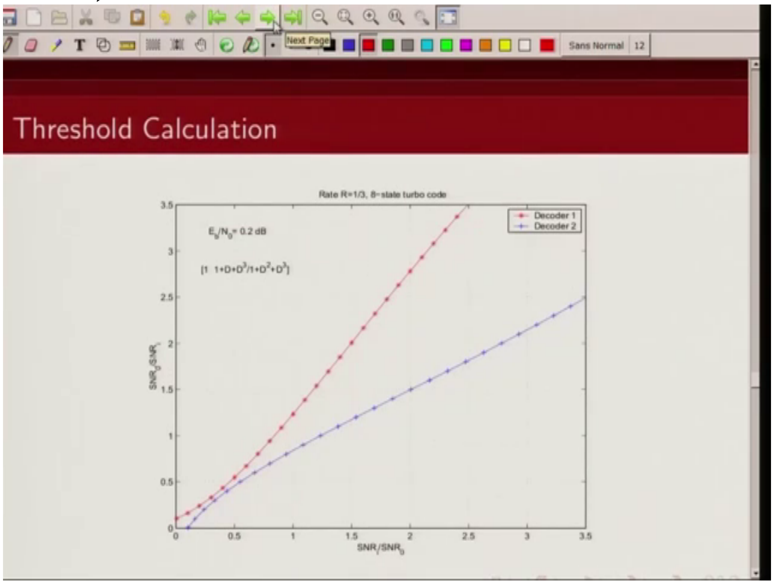
in blue curve I have decoder 2. Note that these 2 are crossing each other so there is no tunnel.

(Refer Slide Time 22:41)



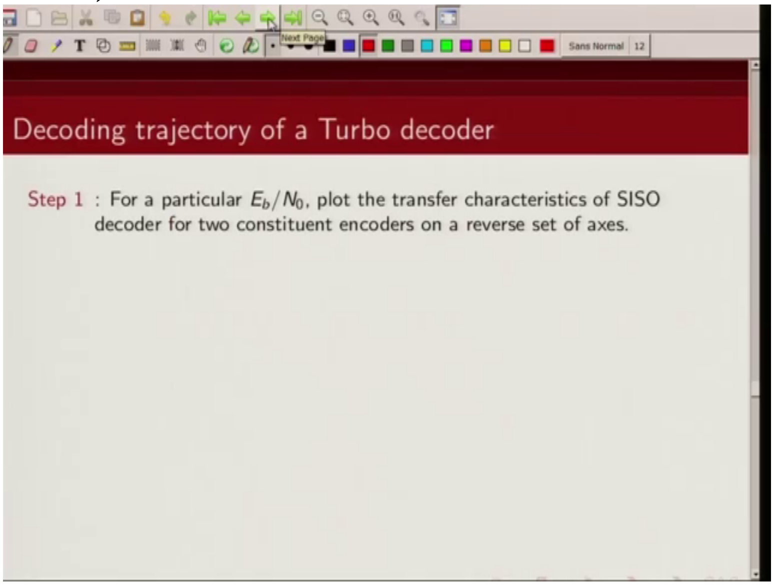
Now

(Refer Slide Time 22:42)



same code, now I increase my S N R and I have made it point 2 d B. Now you can see there is a tunnel between them. There is a tunnel, Ok.

(Refer Slide Time 22:59)

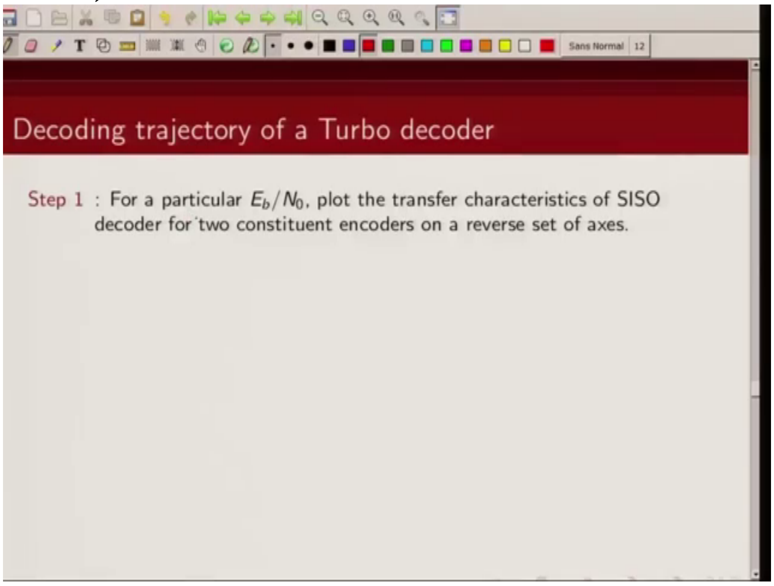


Now let us see how

(Refer Slide Time 23:01)

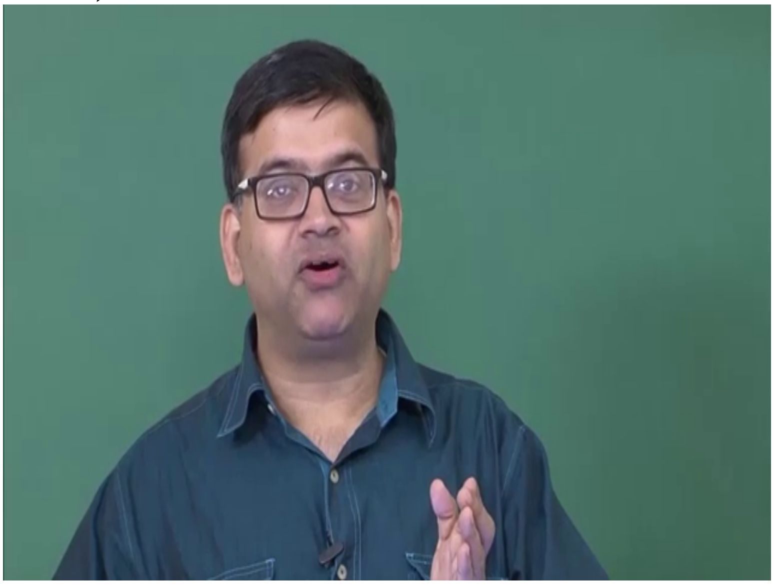
we can draw a decoding trajectory of a turbo decoder with the help of these transfer characteristics. So

(Refer Slide Time 23:11)



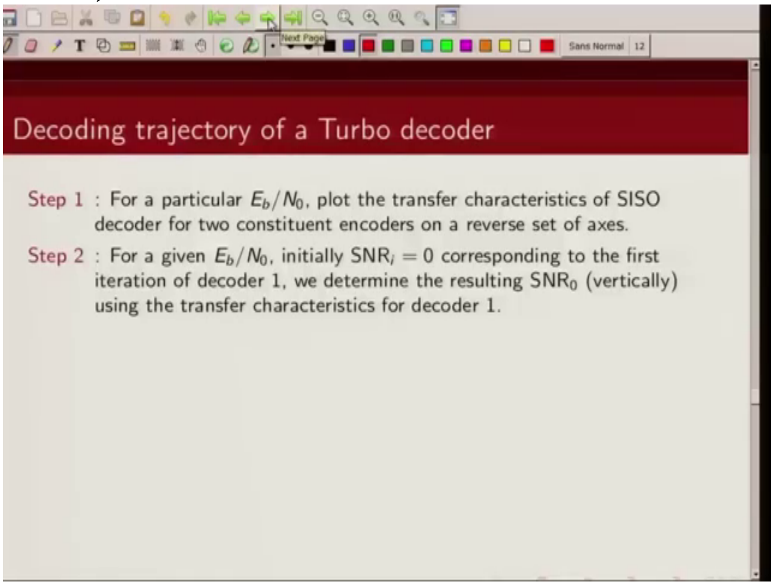
what we do is for a particular signal to noise ratio as I said, we plot these transfer characteristics of two constituent encoders on reverse set of axes. So for decoder 1, S N R in will be on

(Refer Slide Time 23:29)



x axis, S N R out will be on y axis, where as for decoder 2, S N R in will be on y axis and S N R out will be on x axis.

(Refer Slide Time 23:41)

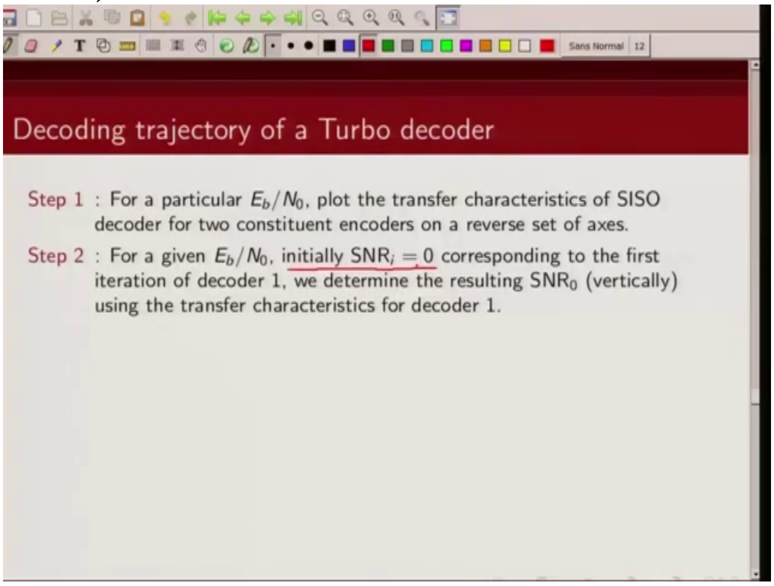


So initially, because you don't have any a priori knowledge about the information bits, so initially the

(Refer Slide Time 23:51)

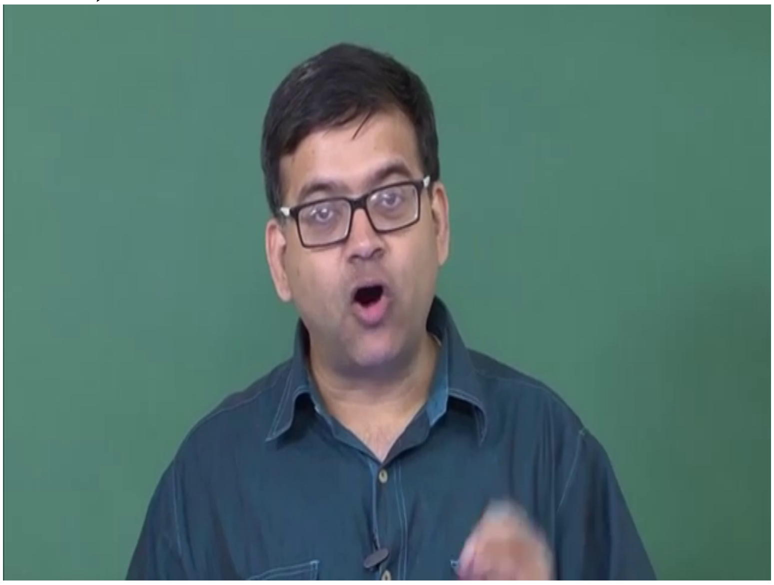
a priori S N R is zero. And this

(Refer Slide Time 23:54)



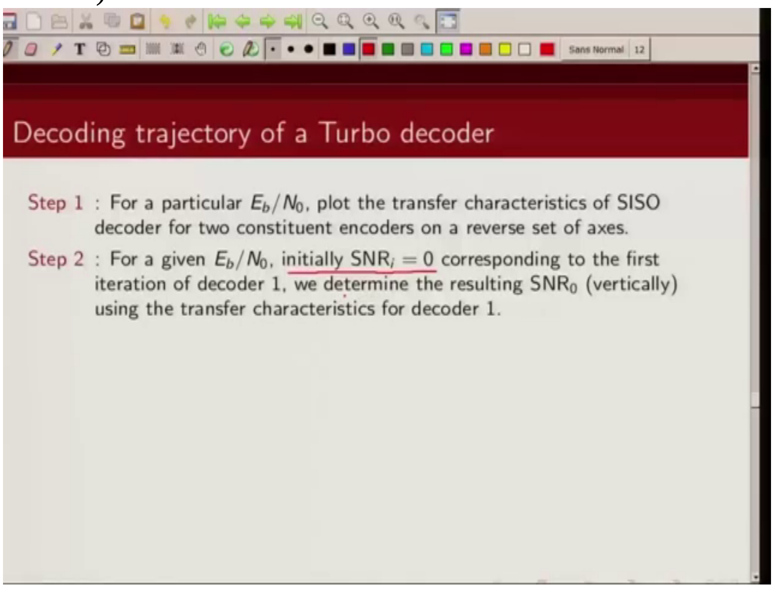
corresponds to, and so we are first going to look at the transfer characteristics of the first decoder. So input we will get zero, so we will try to see what is the

(Refer Slide Time 24:06)



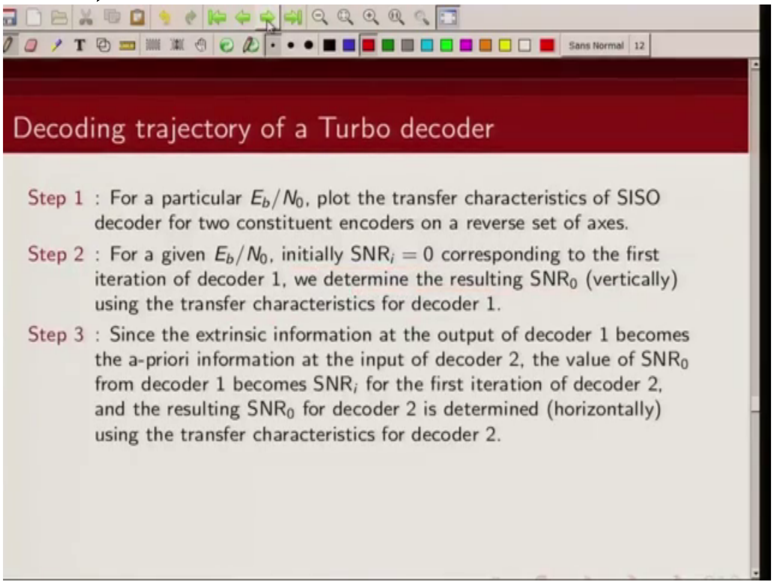
output S N R corresponding to this decoder 1. So we determine

(Refer Slide Time 24:13)



the resulting output S N R which we look vertically for using the transfer characteristics for decoder 1.

(Refer Slide Time 24:23)

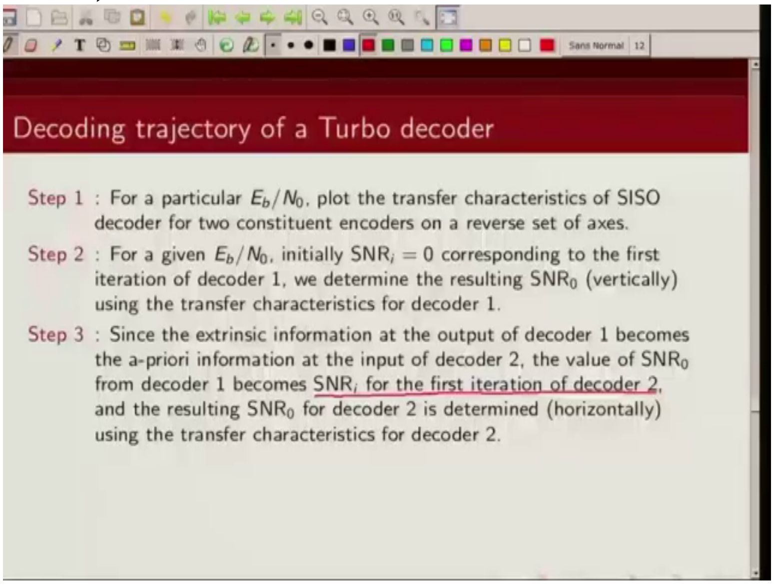


Now as I said, since the extrinsic information from the first decoder is actually a priori value for the second decoder, so what we are going to do is that particular extrinsic information will now become S N R in for the decoder 2. So the

(Refer Slide Time 24:51)

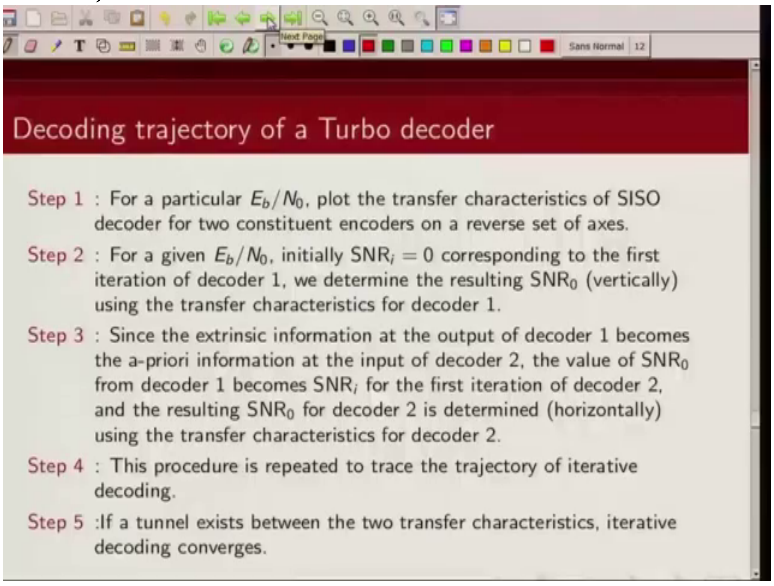
S N R out that we got from the transfer characteristics of decoder 1, that is our new a priori S N R in for decoder 2. Now we are going to look at the transfer characteristics of decoder 2 and we are going to go horizontal and find a point corresponding to that particular a priori S N R what is the output S N R.

(Refer Slide Time 25:20)



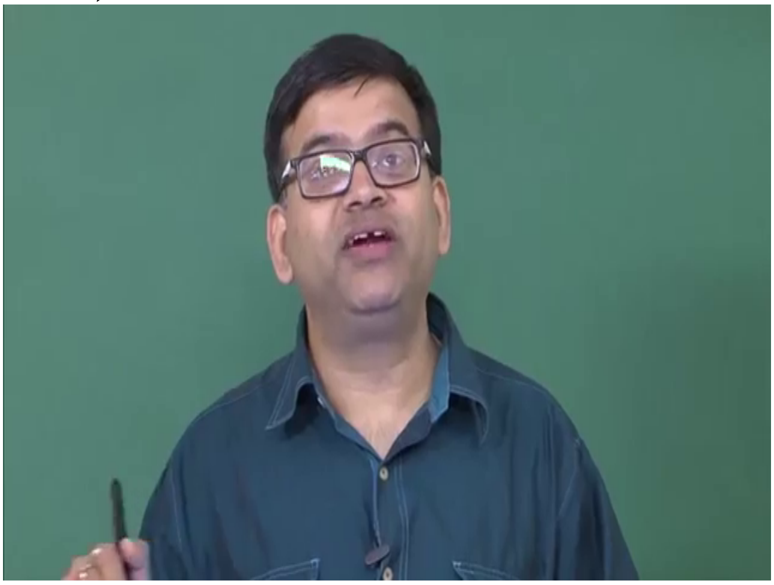
And this process we are going to repeat to draw the decoding trajectory of turbo decoder.

(Refer Slide Time 25:30)



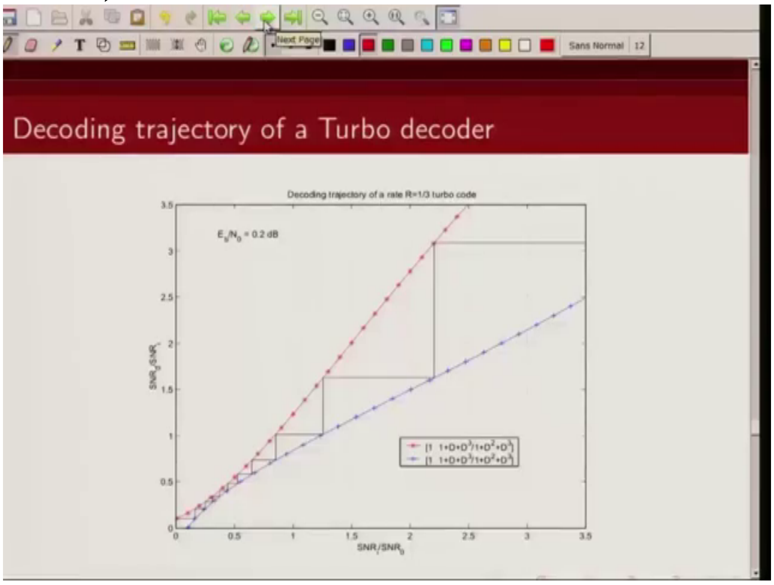
If while drawing this decoding trajectory, our

(Refer Slide Time 25:36)



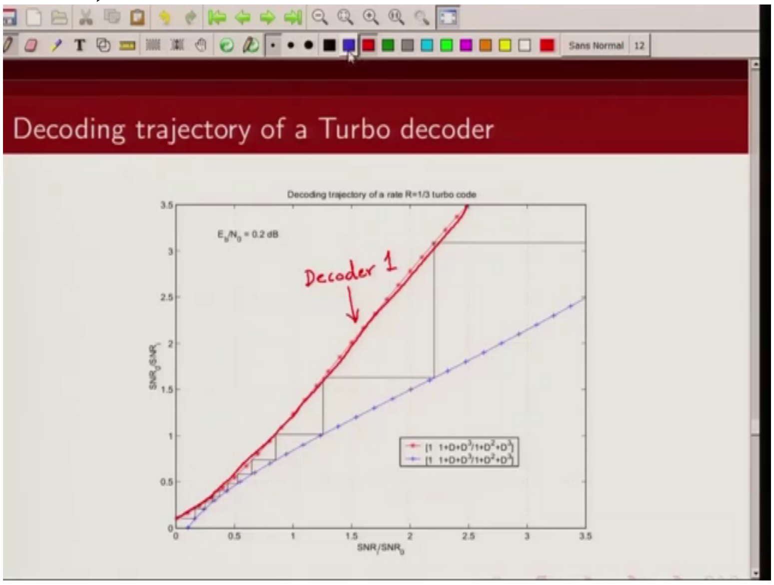
decoding trajectory does not get stuck, our decoding trajectory will not get stuck if there is a tunnel and if there is these transfer characteristics cross each other, then our decoding trajectory will get stuck.

(Refer Slide Time 25:50)



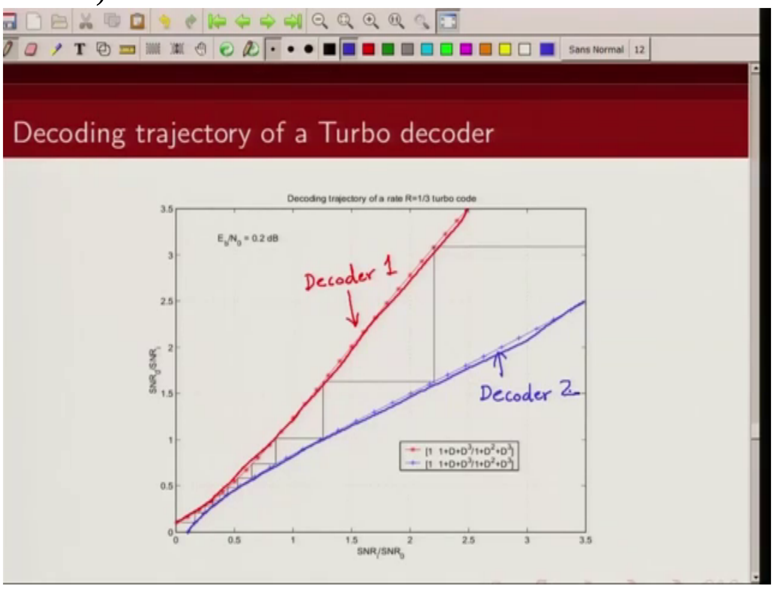
So this is an example. So I have this with red that you see, that is the transfer characteristics of the first decoder. This is decoder 1. This is transfer characteristics of decoder 1.

(Refer Slide Time 26:12)



And what you see in blue is the transfer characteristics of decoder 2. They are the

(Refer Slide Time 26:23)

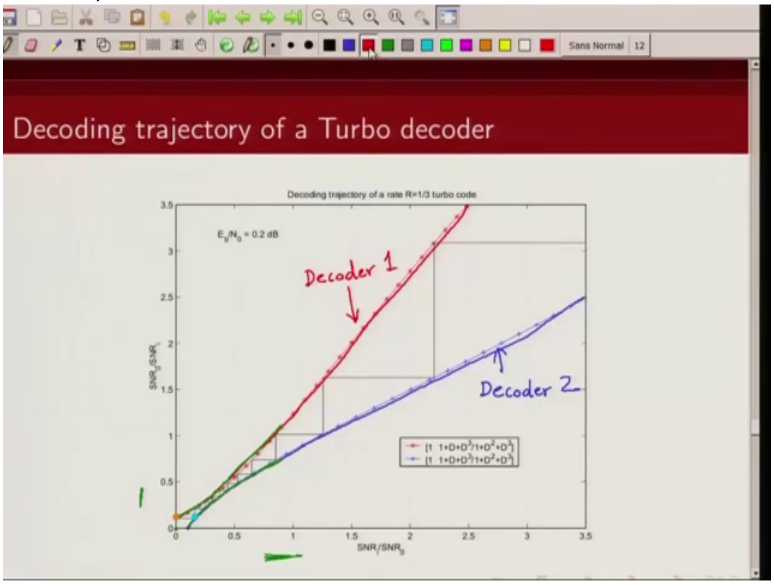


same encoder; this is the symmetric turbo code I am considering. So how do I start? So initially I will look at the transfer characteristics of the first decoder. This is where I will look. So initially I don't have any a priori knowledge. So I will start from this point and I am looking at this curve. So this is my extrinsic S N R corresponding to zero input. Now note that this extrinsic information that we are getting from decoder 1 is going to be the a priori information for decoder 2. So then what we will do? So we will now look at this curve which is transfer characteristics of decoder 2. For decoder 2, this side is input and this side is output, this is input and this is output. So we will look here and we will look horizontally. So this is the point. So this is the point corresponding to S N R out corresponding to decoder 2. Now note this extrinsic information is getting fed as a priori information to

(Refer Slide Time 27:44)

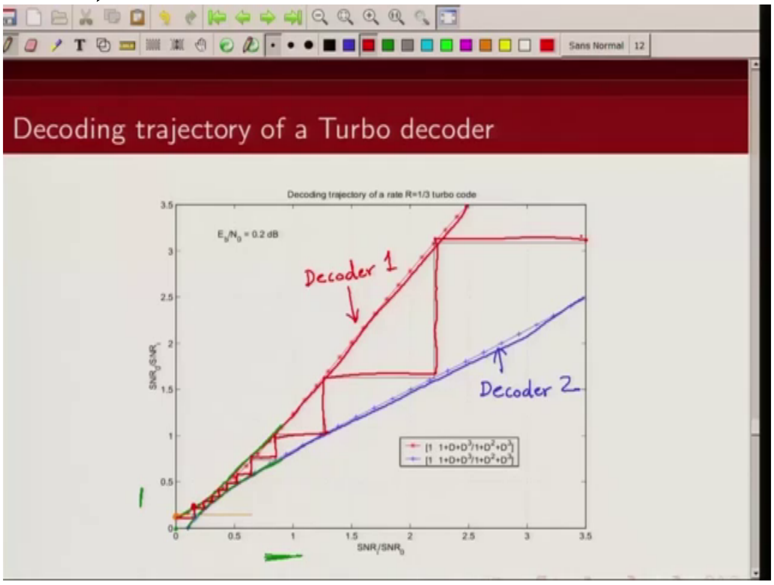
decoder 1. So we will look at

(Refer Slide Time 27:47)



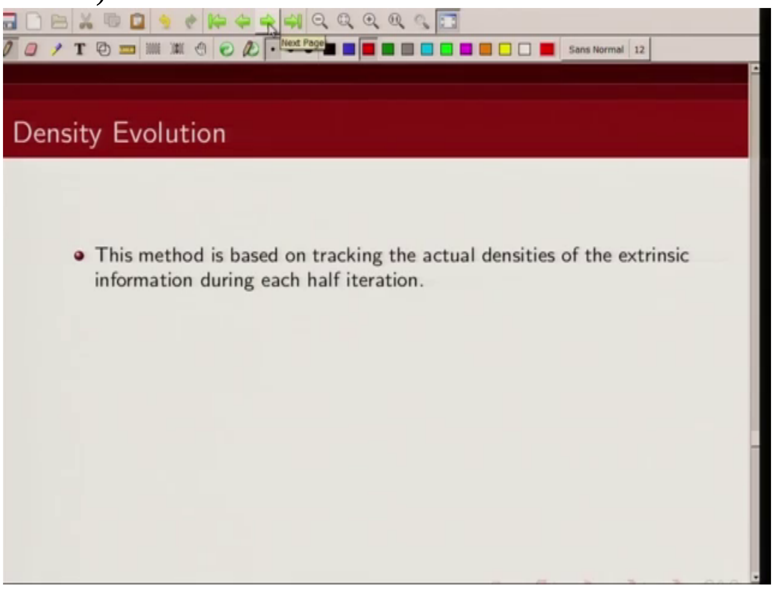
decoder 1 transfer characteristics and this is the point. So you can see I am going like this. You see

(Refer Slide Time 28:06)



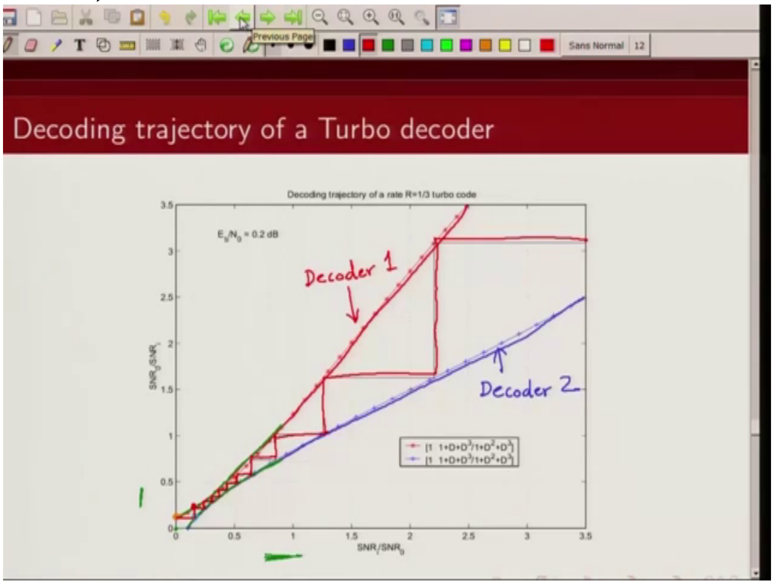
this is how basically my decoding trajectory of my turbo decoder is happening.

(Refer Slide Time 28:14)



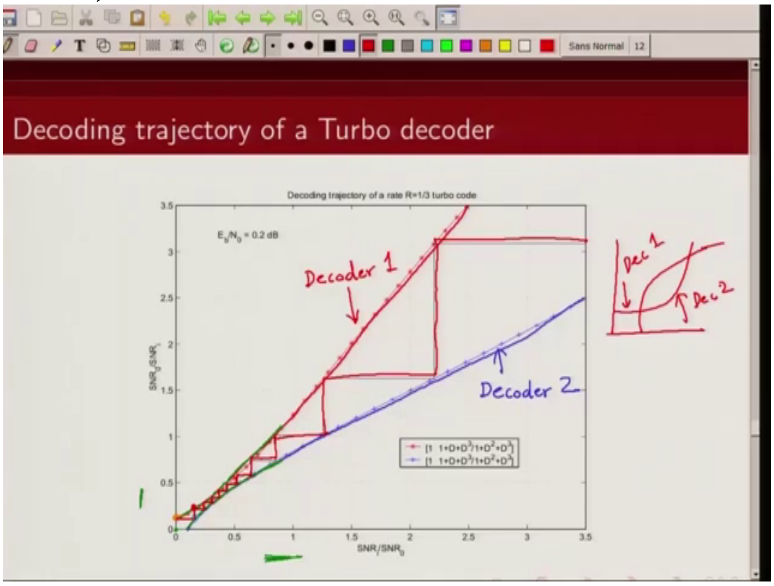
Now what would have happened

(Refer Slide Time 28:17)



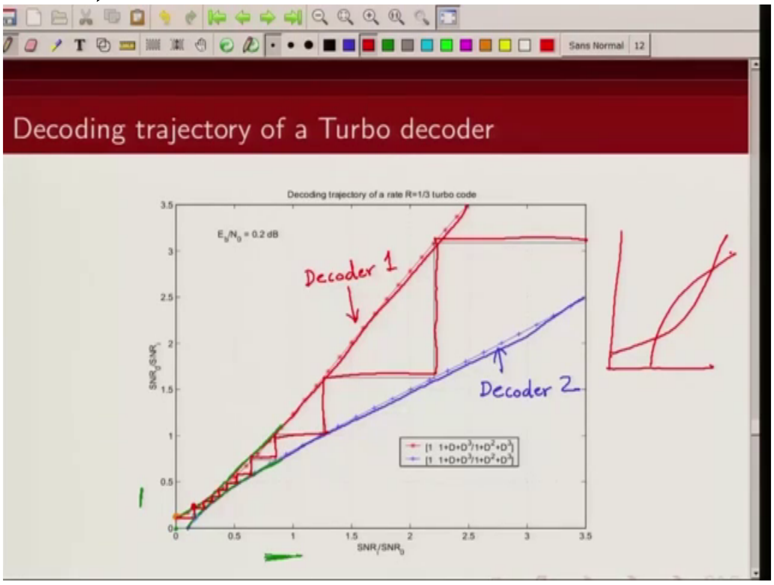
if these curves would have got crossed? So let's look at scenario. Let us say I had some curves which are like this. So let's say this is my decoder 1 and this is my decoder 2.

(Refer Slide Time 28:34)



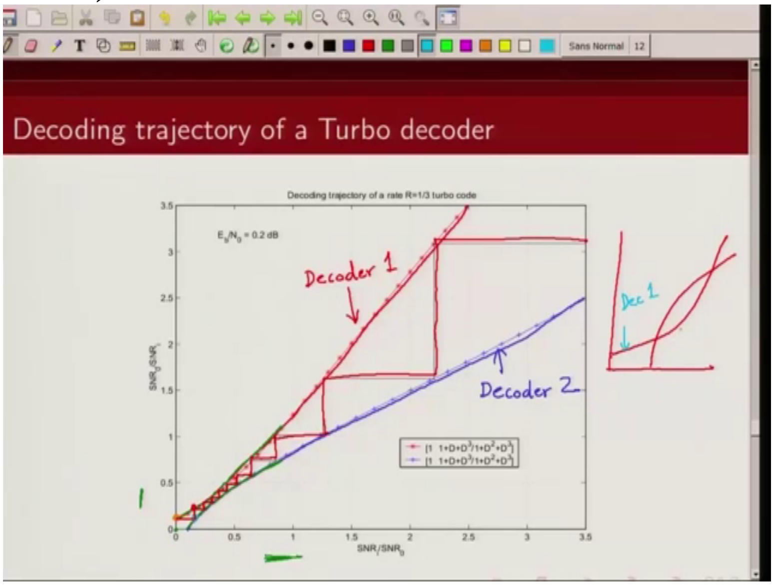
Then what would have happened is, so I would have initially started with zero, I have got this, then I got this. Let me draw slightly better transfer characteristics. So (()) second. So you draw it, basically you draw it like this, Ok. Now

(Refer Slide Time 28:59)



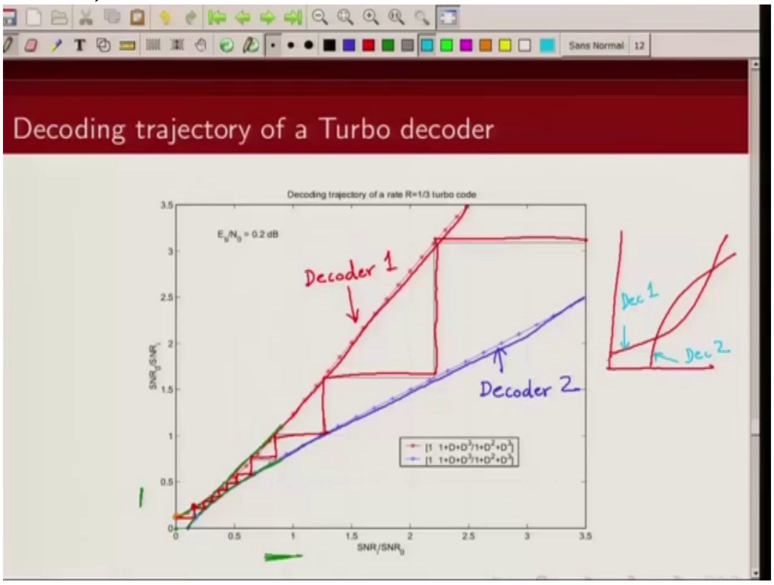
let's draw the decoding. So this is transfer characteristics of decoder 1

(Refer Slide Time 29:06)



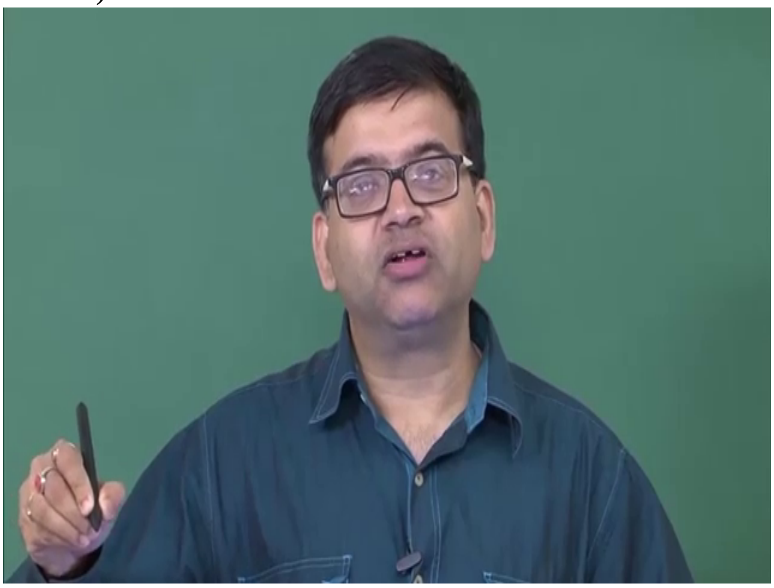
and this is transfer characteristics of decoder 2. So what

(Refer Slide Time 29:11)



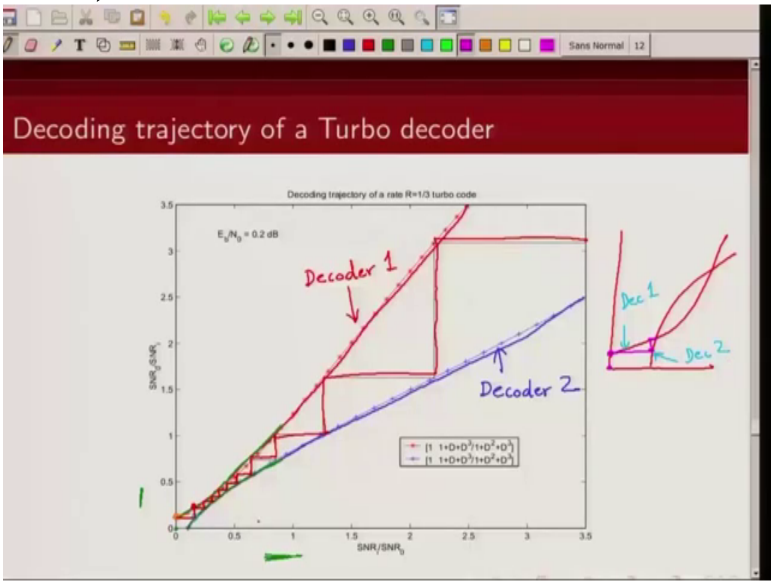
happens here? So you start off with S N R 0 point, you are getting this output S N R from the decoder 1. Now this is input to decoder 2. So you will get to this point. Then from here you will get to this point. Then you get to this point. And then here you are stuck because these 2 graphs cross each other. So what you will notice is if there is no tunnel then your decoding algorithm

(Refer Slide Time 29:38)



will get sruck and the extrinsic values will not improve whereas if there is a tunnel existing

(Refer Slide Time 29:45)

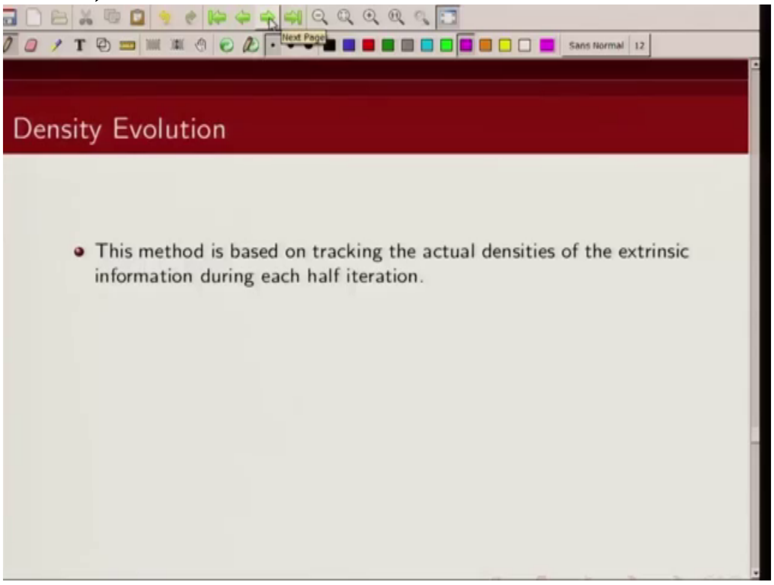


like in this particular case, you saw that, with iterations your extrinsic information is growing. And that's what we would like. So we would like to choose our encoders in such a way such that they match up in a way that there is a tunnel if we plot

(Refer Slide Time 30:04)

the decoding trajectories on reverse axes.

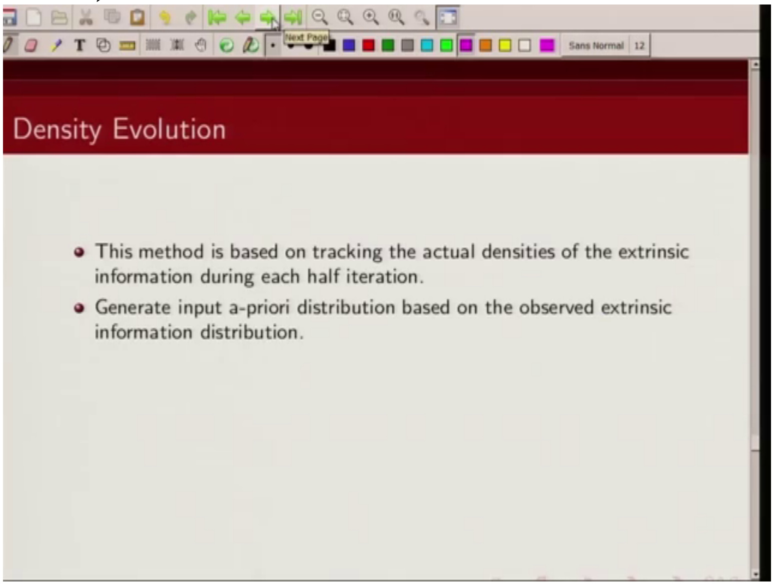
(Refer Slide Time 30:11)



This was the method of El Gamal.

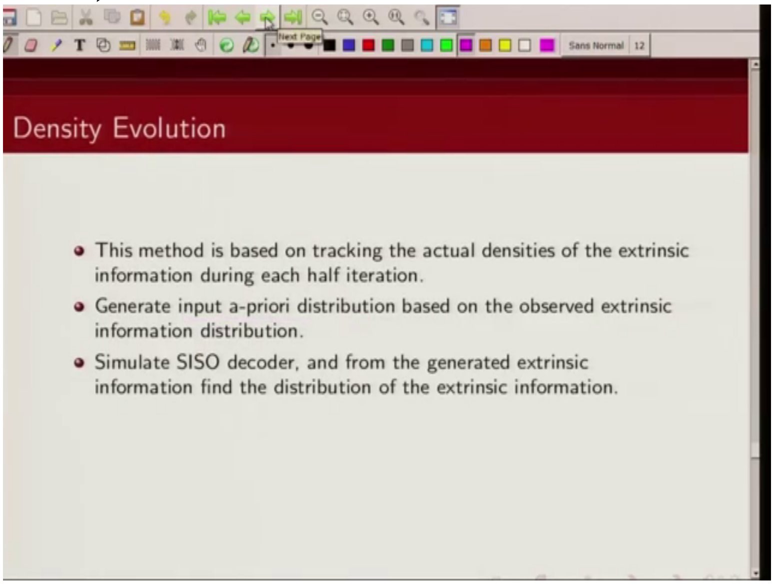
Now the method of Divsalar, they actually used the actual densities of the extrinsic information and they track it for finding

(Refer Slide Time 30:27)



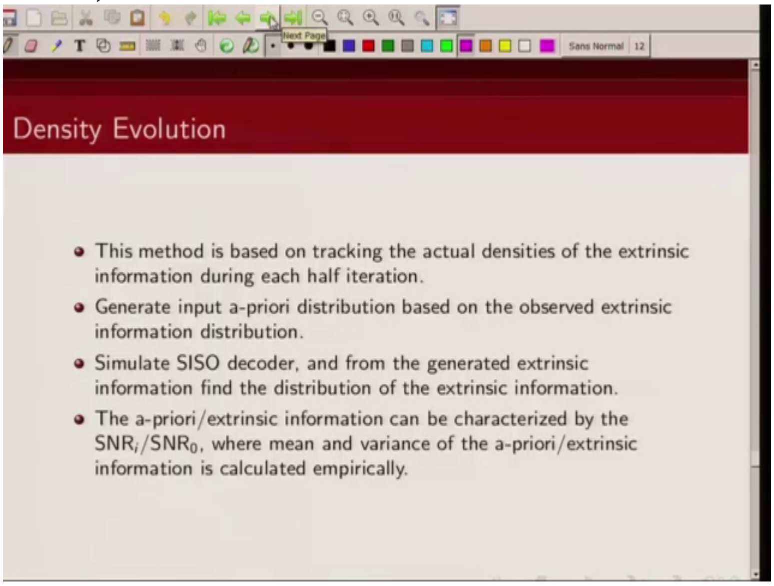
out how it is growing for iteration. So they generated some input a priori distribution based on observed extrinsic information and then they

(Refer Slide Time 30:38)



simulate this soft input soft output decoder using this generated distribution of a priori information and they find out the distribution of extrinsic

(Refer Slide Time 30:54)

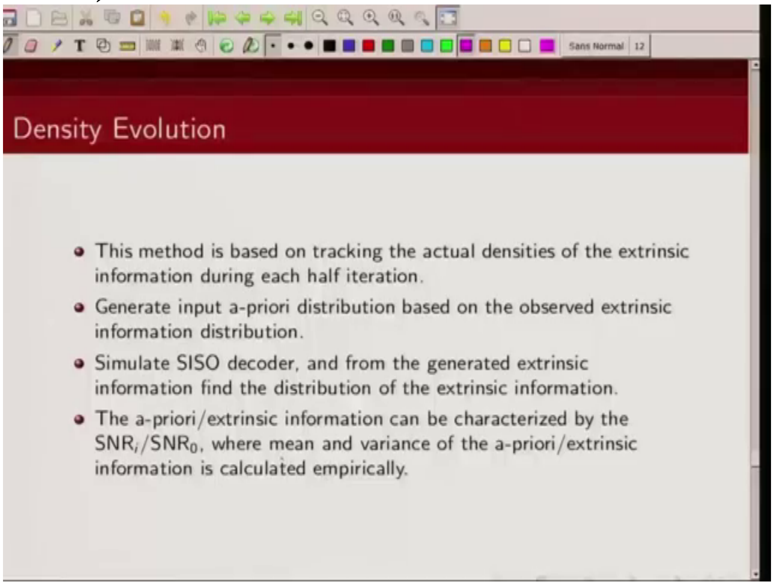


information. And similarly they characterized

(Refer Slide Time 31:00)

the S N R of the input distribution as well as the output distribution using mean and

(Refer Slide Time 31:13)

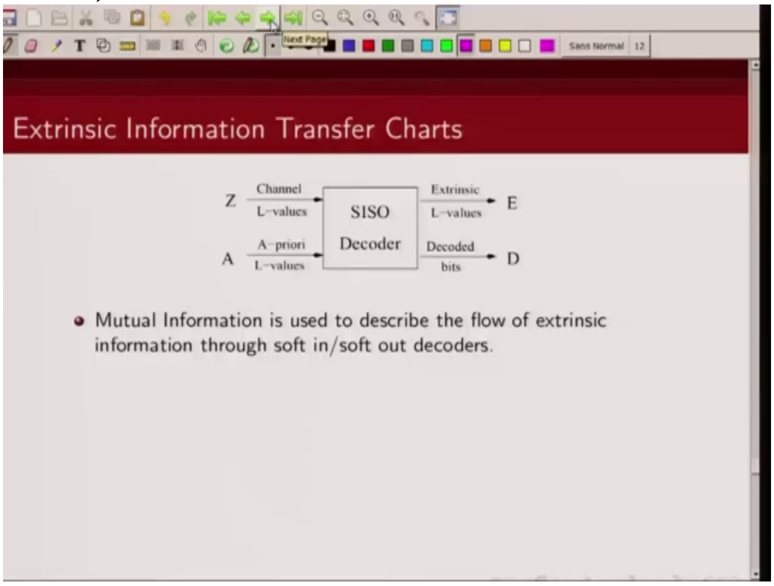


variance which was empirically computed. So they did not assume that consistency criteria which El Gamal and others did, they actually

(Refer Slide Time 31:26)

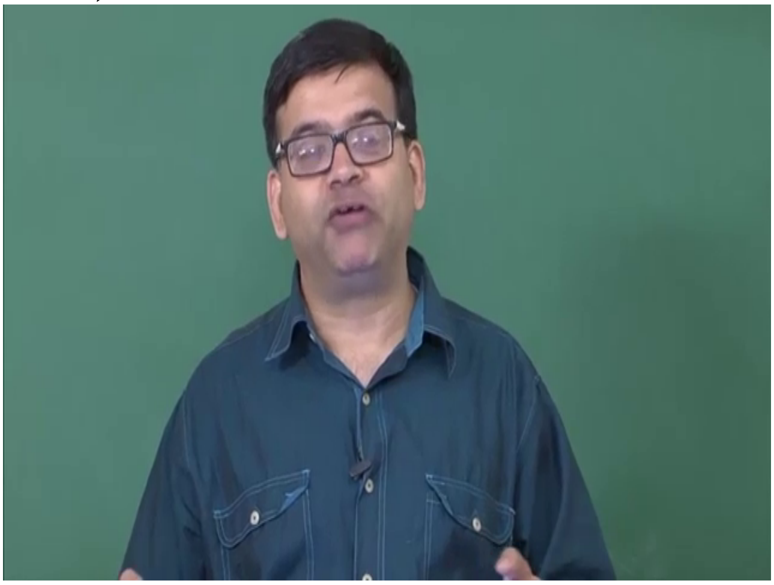
used the observed density. They generated a priori information based on the observed distribution of the extrinsic information.

(Refer Slide Time 31:36)



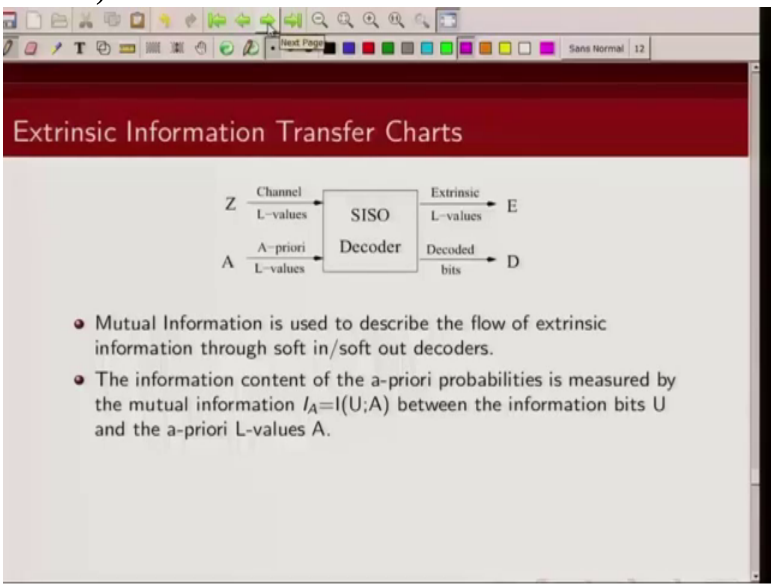
The third method which was proposed is based on mutual information. So mutual information was used to describe the flow

(Refer Slide Time 31:47)



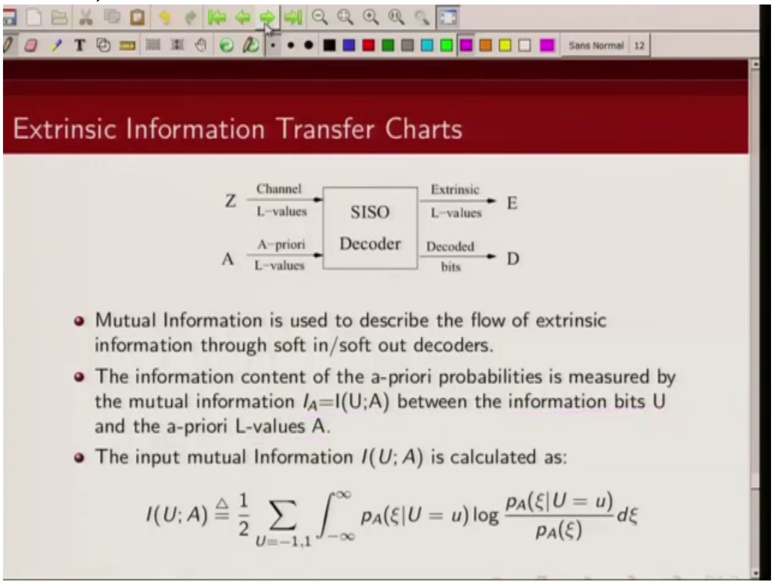
of information through this soft input soft output decoder. So there

(Refer Slide Time 31:51)



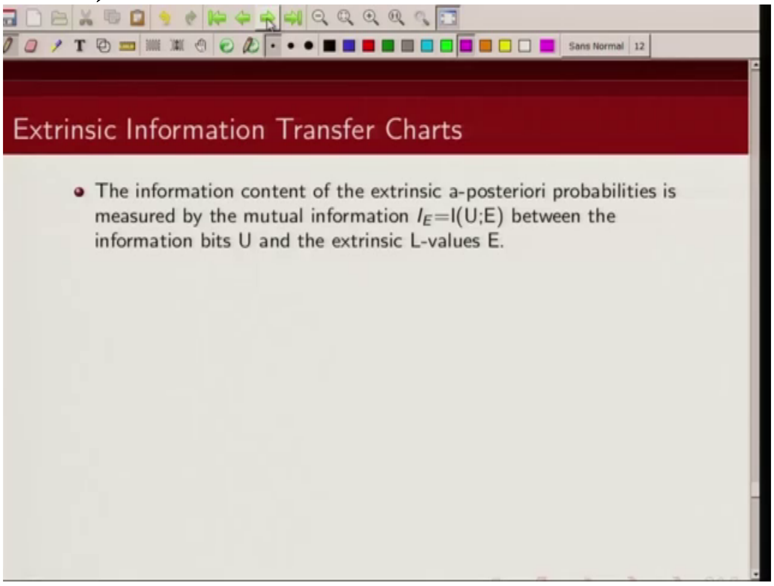
were 2 quantities which were described here. Basically one was this input mutual information which is the mutual information between the information bits and the a priori value and the second

(Refer Slide Time 32:04)



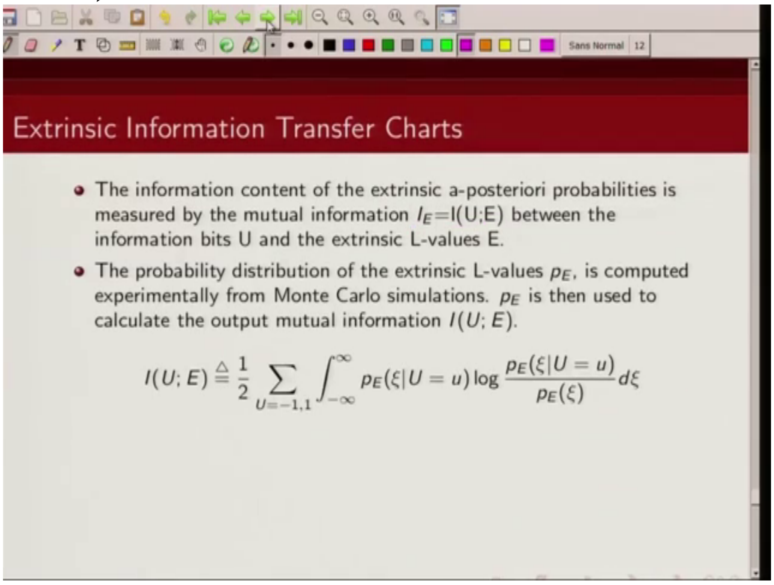
term which was defined here was

(Refer Slide Time 32:07)



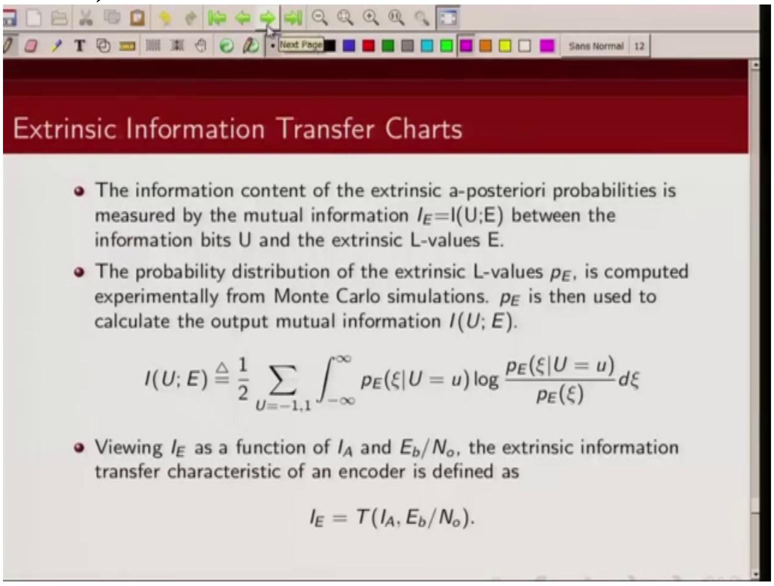
the extrinsic mutual information which is the mutual information between the input bits and the extrinsic values. So

(Refer Slide Time 32:17)



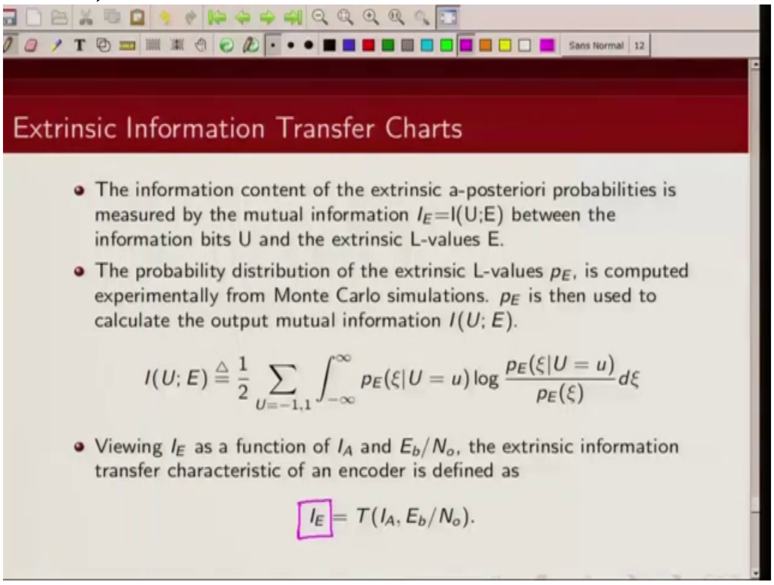
what was done in

(Refer Slide Time 32:19)



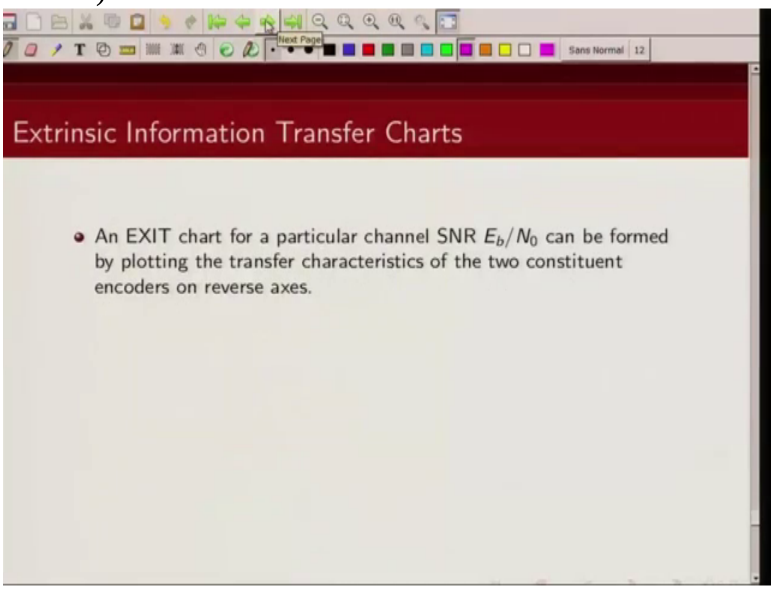
this technique was you can view the mutual information corresponding

(Refer Slide Time 32:24)



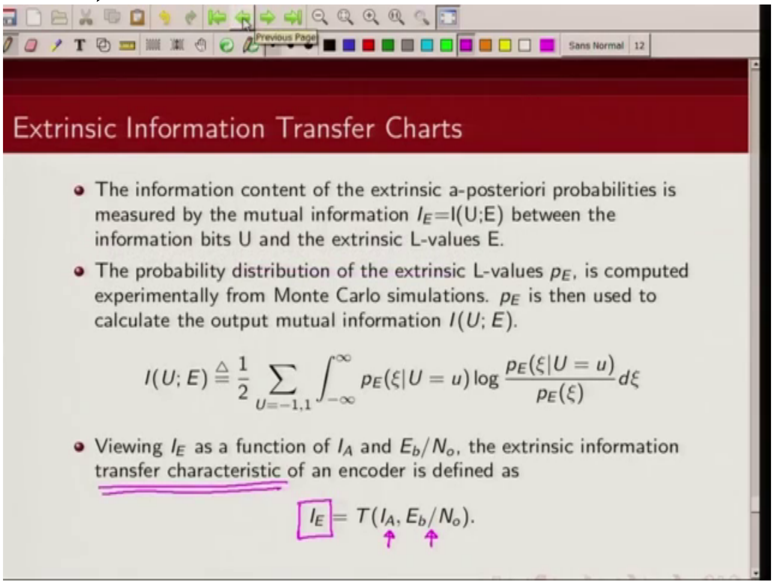
to the input and extrinsic value as a function of mutual information of a priori values and information bits and operating signal to noise ratio. So this was the transfer function which was considered in this extrinsic information chart. That viewing the output mutual information between the extrinsic information and the information bit as a function of mutual information between the a priori and the information bits and signal to noise ratio.

(Refer Slide Time 33:04)



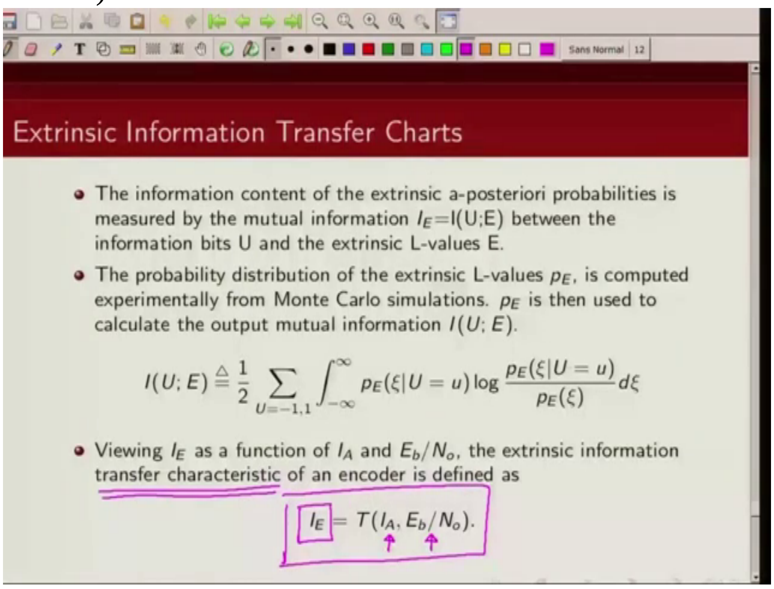
So how was EXIT chart created? So they plotted these transfer characteristics which was given by this.

(Refer Slide Time 33:15)

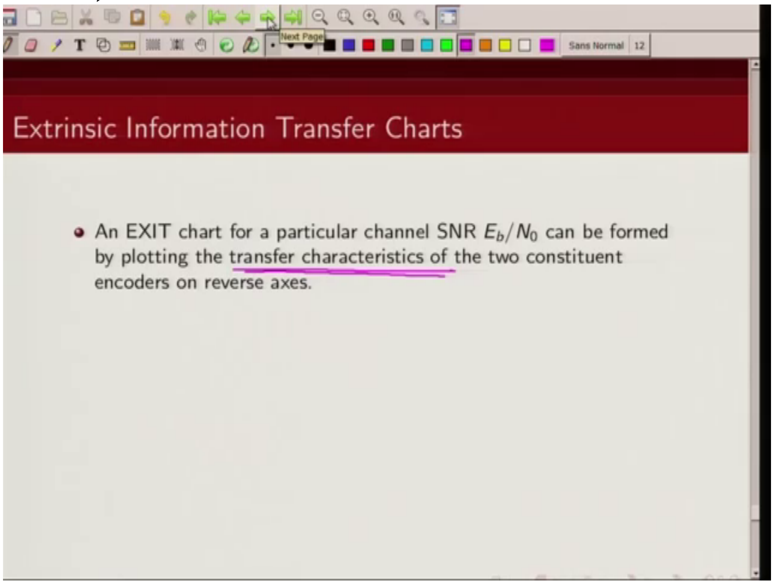


They plotted these transfer characteristics

(Refer Slide Time 33:19)

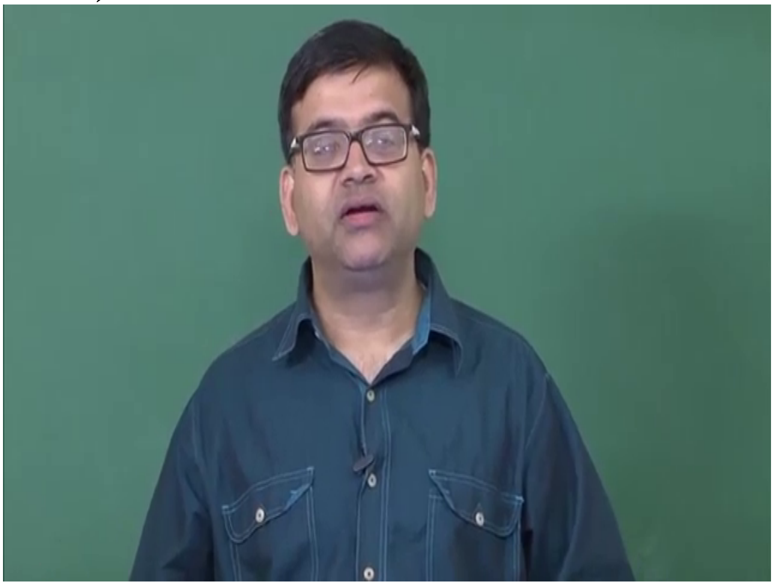


(Refer Slide Time 33:21)



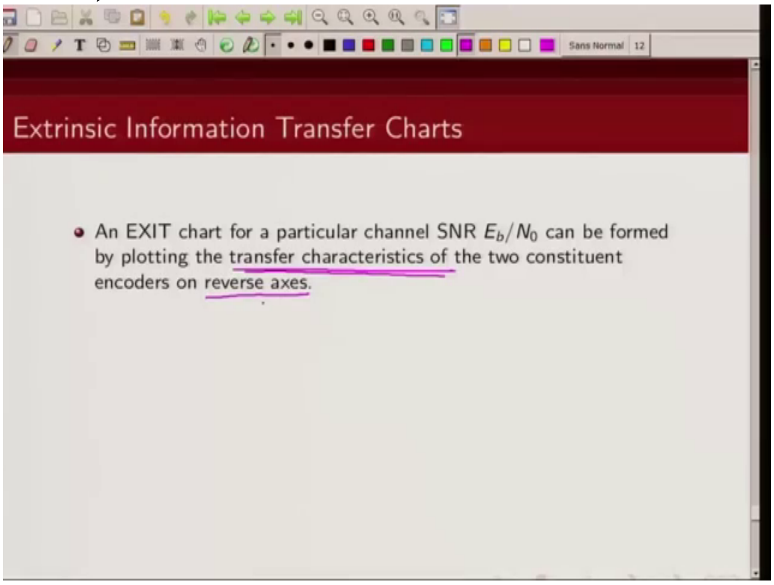
for two constituent

(Refer Slide Time 33:23)



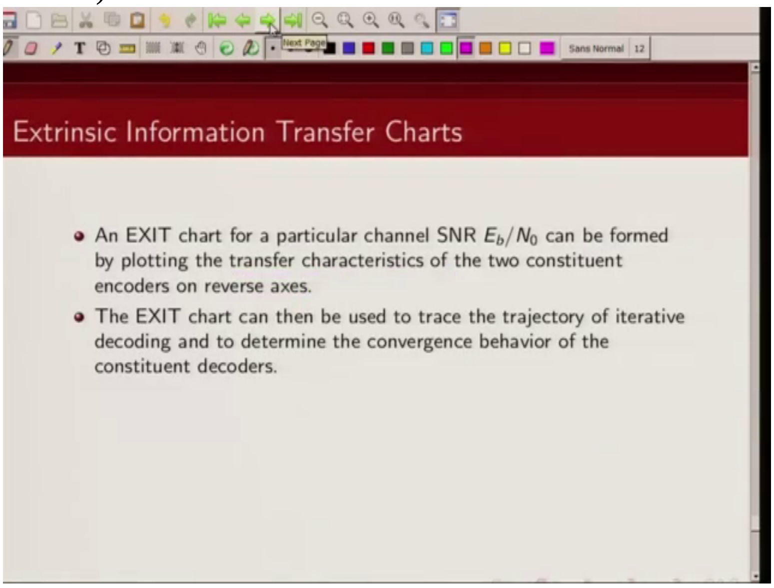
decoders on reverse axes

(Refer Slide Time 33:27)



similar to El Gamal's technique, the difference is

(Refer Slide Time 33:30)

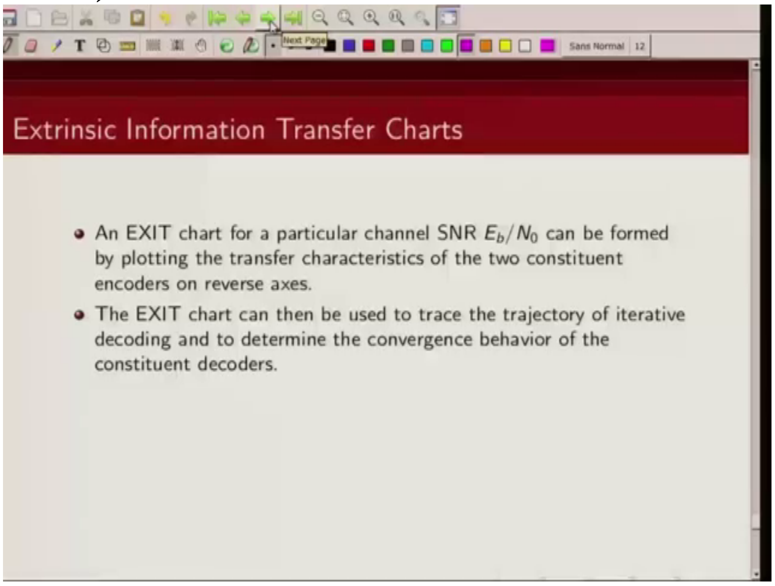


El Gamal used mean as

(Refer Slide Time 33:33)

S N R, here they used mutual information.

(Refer Slide Time 33:38)

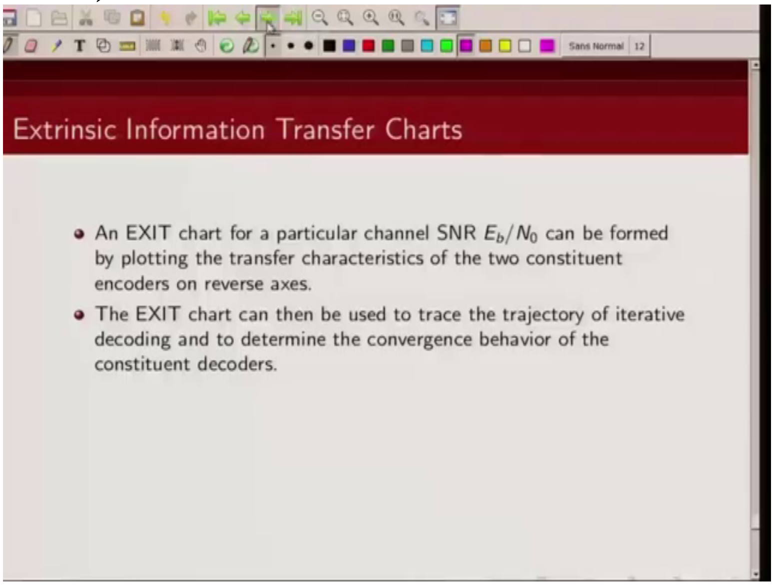


So very similar idea, so

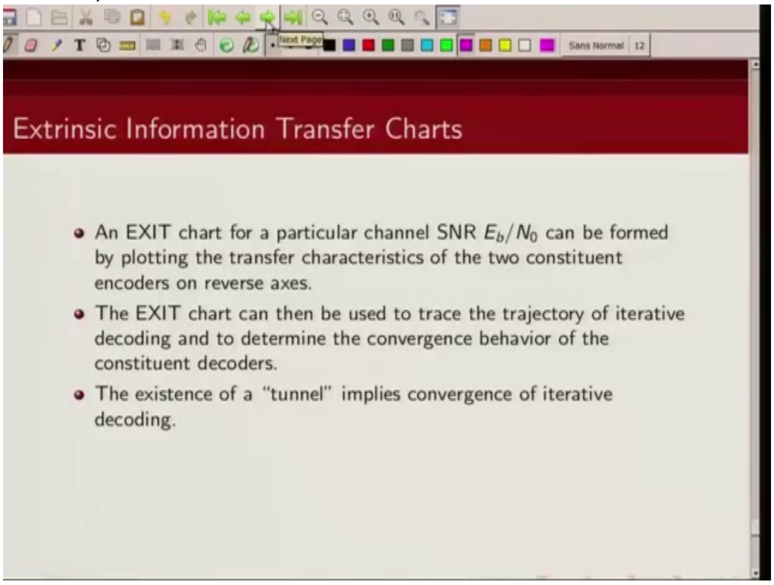
(Refer Slide Time 33:42)

these transfer functions were plotted on reverse axes. Initially you don't have any a priori knowledge, so the input a priori mutual information is zero. And then after one half iteration, you get some extrinsic information. So you have some positive mutual information. And then you pass that as input to second decoder. And the decoding will progress if there is a tunnel otherwise it will get stuck.

(Refer Slide Time 34:14)

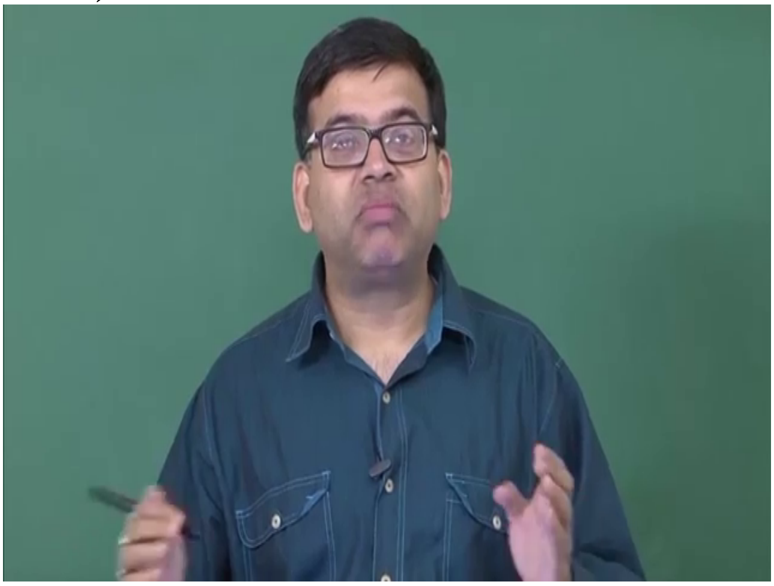


(Refer Slide Time 34:15)



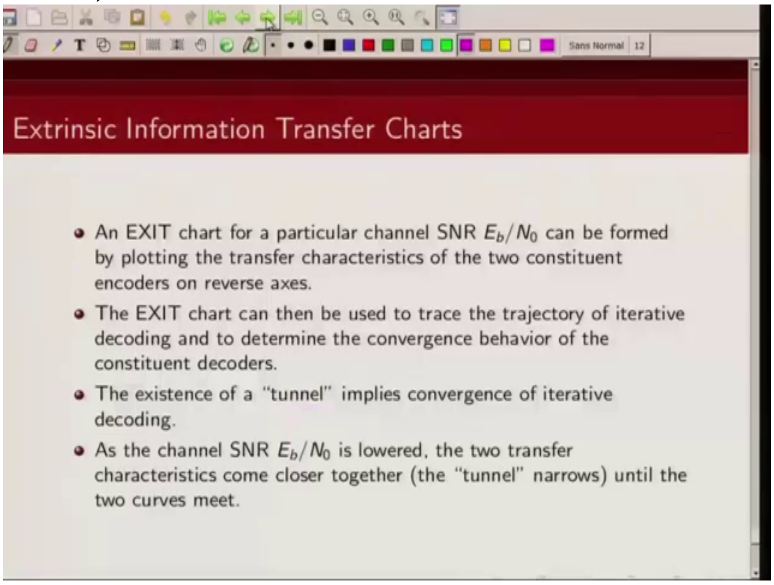
So as I have said, whether the decoding algorithm will converge or not, is, can be viewed by

(Refer Slide Time 34:27)



plotting these transfer characteristics on reverse axes and seeing whether a tunnel exists between them or not.

(Refer Slide Time 34:36)



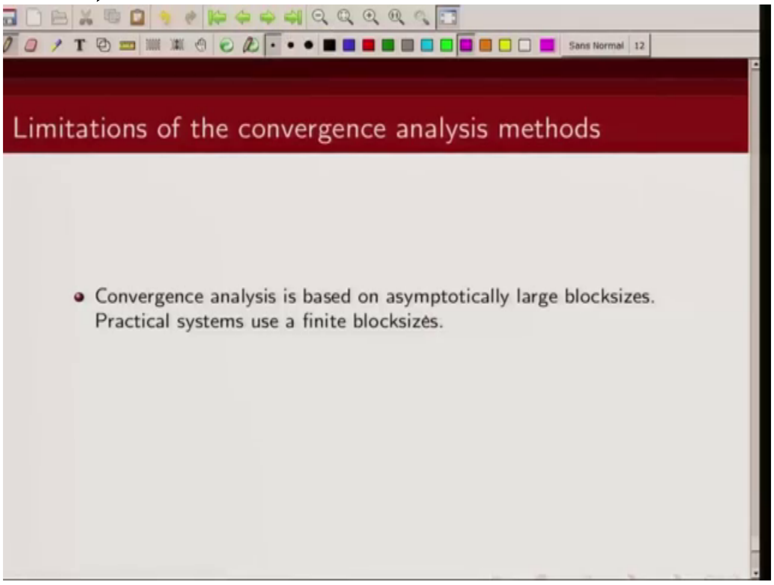
Now what happens if we reduce the channel operating S N R? If we reduce channel operating S N R, then these curves come closer until a point will come when they will barely touch or they will touch and cross each other. So the point, the minimum S N R

(Refer Slide Time 34:56)

where there still is a tunnel that's your threshold, decoding threshold.

So we have specified various methods for tracking the mutual information, tracking the extrinsic information and a priori information and this can be used to see how our constituent encoders will behave, how the turbo code, how the turbo decoder will behave under iterative decoding algorithm. Now what are the limitations of this analysis approach? Now this approach assumes that

(Refer Slide Time 35:42)



we have very large block sizes. So these convergence analysis results hold for very large block

(Refer Slide Time 35:50)

sizes but in practical systems we use small size block sizes so the thresholds predicted by this method may not be consistent when we use small block sizes and of course there are some assumptions, for example in El Gamal's technique we use Gaussian assumptions, we made assumption of consistency conditions. Those conditions may or may not hold, Ok. So with this I will conclude this discussion on convergence analysis of turbo codes, thank you.