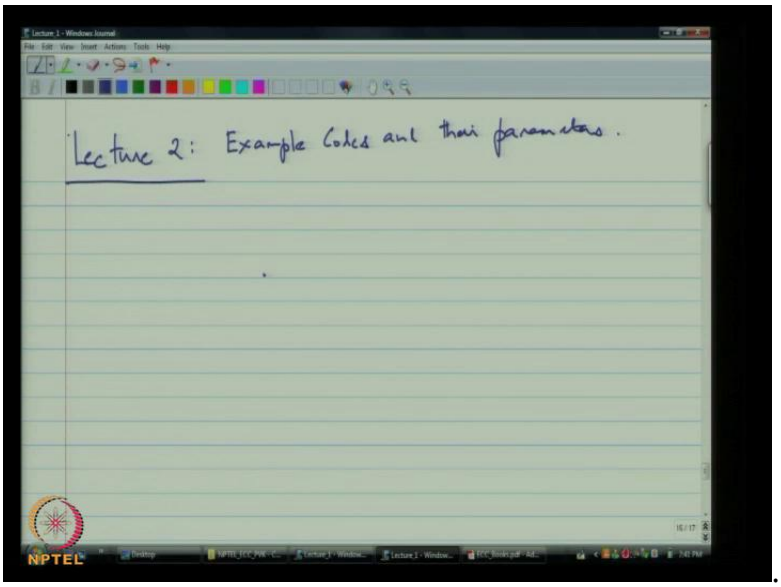
Error Correcting Codes
Dr. P. Vijay Kumar
Electrical Communication Engineering
Indian Institute of Technology, Bangalore

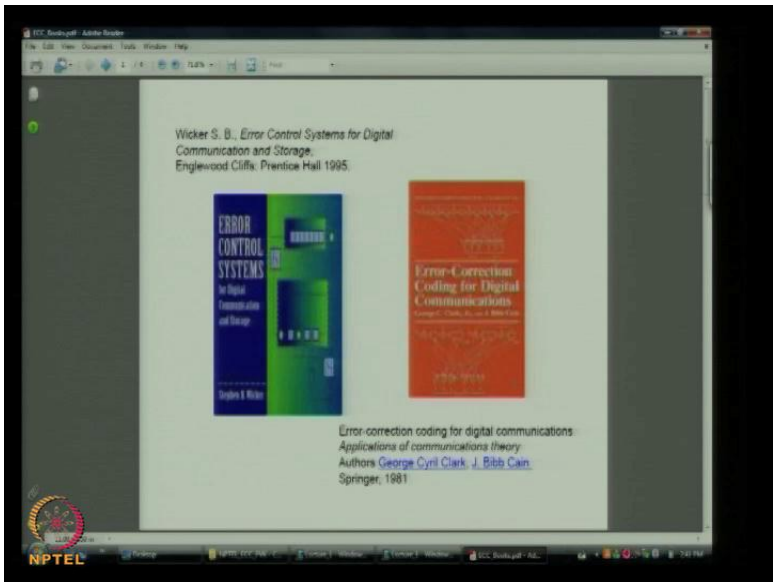
Lecture No. # 02
Examples Codes and their Parameters

(Refer Slide Time: 00:37)



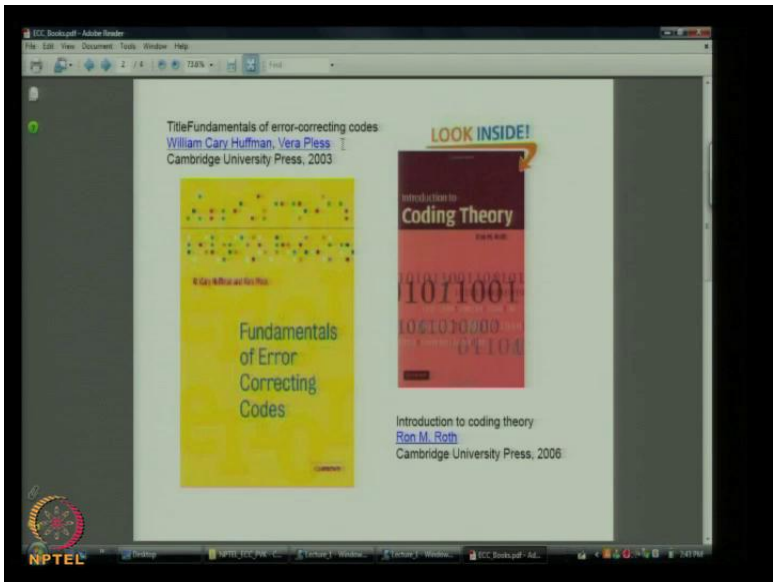
Second lecture on error correcting codes, and let me just bring you to the title of our lecture today. So, our lecture today is entitled error correcting example codes and their parameters. But normally what I do is before I start every lecture, I give an overview of what we covered in the previous lecture, and I will also do that today. But there is other thing that I actually wanted to do which was to show you some of text books that you might want to take look at just to get just to use them as a reference.

(Refer Slide Time: 01:16)



So, let me just quickly do that, then I will come back, I will summaries the last lecture and then we will continue with our lecture. So, here now these text books I must say are not in any particular order, it is a random order, there is one book Steven Wicker Error Control Systems for Digital Communication and Storage, this is 1995. So, not quite latest text book, but still not bad. So, this is one of the text books. It is a, it is a it is a well written text book, it is quite balanced, little on the older side. To the right is a text book, which is even dates back even further back to 1981, but one of the things that I particularly liked about this text book is that it is extremely well written, and it is easy to read the authors avoid, they avoid formulae or equation wherever possible in yet convey much of the meaning. But of course, in since 1981 this is clearly dated, but still it is also written from a rather practical point of view, because these authors are actually practicing engineers. So, it is little bit different because most of the other text books are written by people, who are professors in some university or the other.

(Refer Slide Time: 02:37)

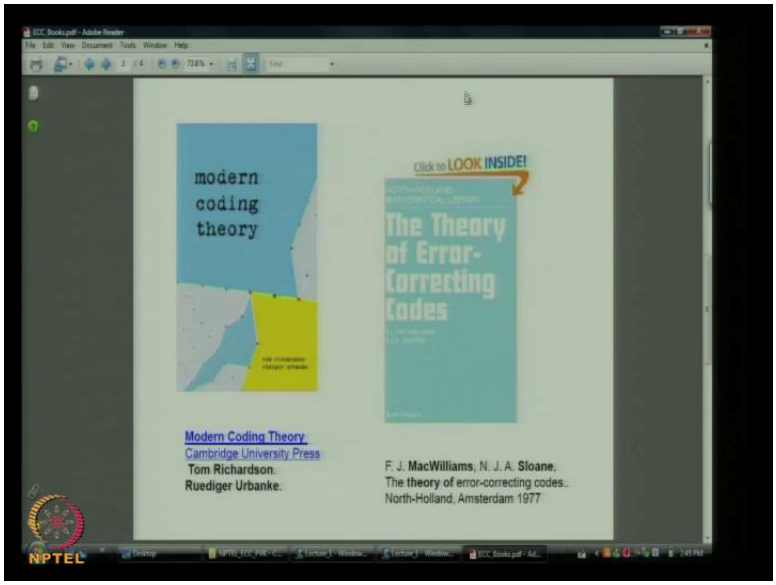


This one by contrast is this one on the left over here, is this one here is more recent text book. The Fundamental of Error Correcting Codes by Huffman and Vera Pless, it is 2003, very well written. These same authors actually put together handbook on coding theory, which is about several thousand pages pages long, and they got contributing contributions from the leading coding theories around the world. Then subsequently, they brought out their own book and this book is very well written, very carefully written. I think at some extent, the text books define their coverage of the topics. For example; one author who is done research in a particular topic, might actually emphasize that topic in relation to the others. So, that often is the difference between textbooks.

Now the other comment I guess, I should make is versus I mean this issue of the Classical Coding Theory, which tends to be algebraic versus the modern, which tends to the probabilistic. From that point of view this book, for example; this book is algebraic, it does not consider the modern theory, coding theory, we point simply because it was written a little before Modern Coding Theory took over, took its current place. Similarly, the same is true of this book except the it is written as I just explained in an not so mathematical manner. This one it does write to include some of the modern coding theory, but really the authors are are experts in algebra and (( )) and the book reflects that. Similarly, here if you look at Ron M. Roth he is classical coding theorist although he does some of the modern

aspects as well, but again you can actually as you browse through the contents of this book, you see that the authors personal research preferences show up in terms of the emphasis on different topics.

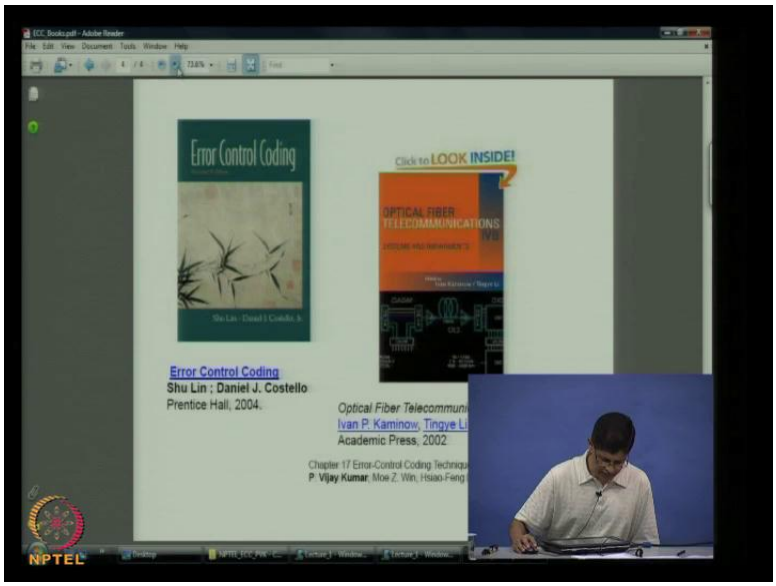
(Refer Slide Time: 04:38)



Now first let me begin with this book on the right, the book this book on the right was the classic, it was it is a 1977 book, so very much dated, but in its time it was considered a Bible. It is it is the Bible from an algebraic coding theory point of view. It is like an encyclopedia anything that you want it could actually be found there, it is beauty beautifully written, but very compact. So, every word there is no extra word in this entire text book. But also it is restricted to the algebraic coding theory view point. And even there for example; it is not treat the topic of convolutional codes. So, it is rather specialize that given that it is superb text book.

On the left, you have kind of the opposite model, which is the Modern Coding Theory text book. I think, it is a year or two old, and it is completely different from all the other text books, because it takes predominantly the modern coding theory view point, which is that it is very probabilistic and it emphasizes low density parity check codes or or or turbo codes and so on. Very nicely written, the authors very are experts in this area.

(Refer Slide Time: 05:54)

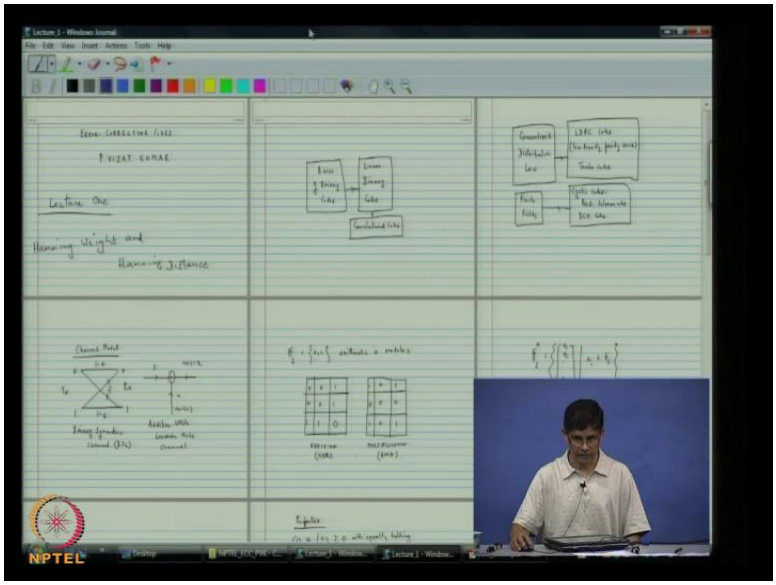


The text book on the left here maybe I will just magnify this little bit. This text book on the left here, Error Control Coding by Shu Line and Daniel J Castello, this was the default reference text book for many years. And basically, because all topics were covered and they two authors were experts respectively in block coding and convolutional coding and so it made for balance. Now now that with regard to modern coding theory, the authors have done some work in this area particularly the first author. So, they do pay some attention to that as well in their topics. So, basically what they have done is they taken that text book and then they are upgraded it to include the modern coding theory view point. It is its for main people will actually like if we, if we have to use the single text book, you might actually think of considering this book. To the right to the right is book actually on Optical Fiber Tele Communication as you can see here. So, we might say well what is an Optical Fiber Tele Communication text book doing here?

As I mentioned last time that in this hand book published in 2002, a bunch of us had occasion to write to a chapter on error control coding techniques; and and my my discussion the topics will be in the same sequence as is covered in the chapter. So, this might be the useful reference for you. So, there are 8 text books here, but in reality, but if you look out in the market there are lots of text books on coding theory many, many text books have come out in the recent few years. So, this is, by no means exhaustive, but just give an idea of some

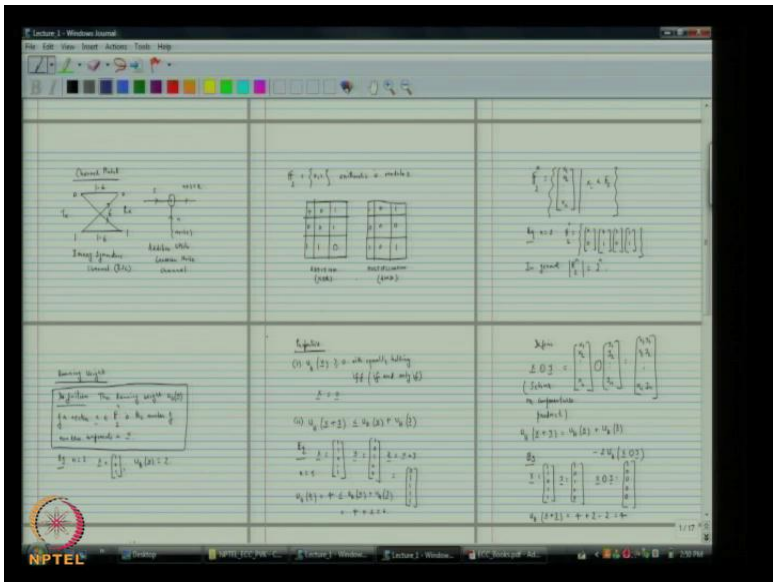
of the text books that you are likely to encounter; so that I will close on this. And then we will move on to talking about the lecture. The lecture 2 is example codes and their performance. So, let us quickly preview the previous lecture.

(Refer Slide Time: 08:06)



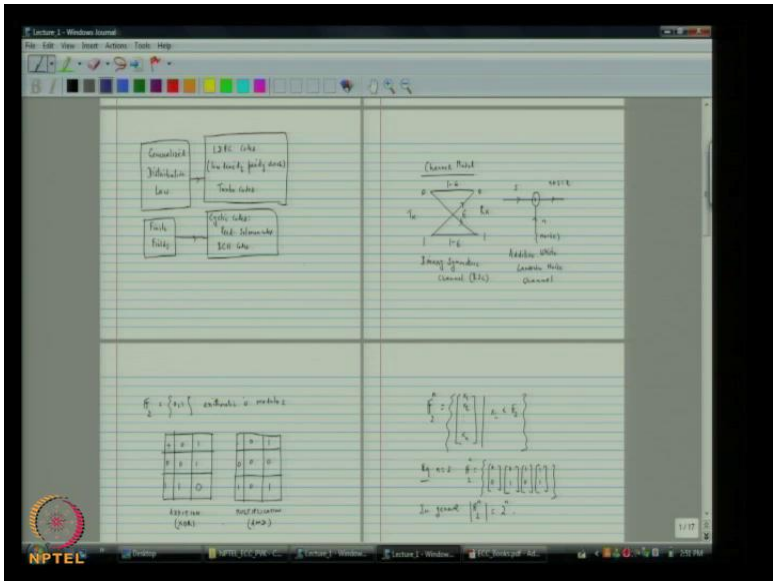
Just for an overview purpose I think that this level of magnification should suffice, what I did last time was basically said I am going to introduce the course in terms of flow charts. So, here what you see here is like a flow chart. So, I discuss, I discuss over here in the second slide, let us see if we can see this (( )) there you go. So, on this side what I do is I discuss in this slide. I tell you how we are going to start of the course by talking about basics of binary codes; then we will move on to linear binary codes. And then as a side topic we consider the topic of convolutional codes. And then after convolutional codes, we will move on to to the modern view point, which is, which as I said is probabilistic, but we will approach it from the point of view of an algorithm that is way efficient in terms of minimizing competitions. And that algorithm has its bases in something called generalized distributive law.

(Refer Slide Time: 09:43)



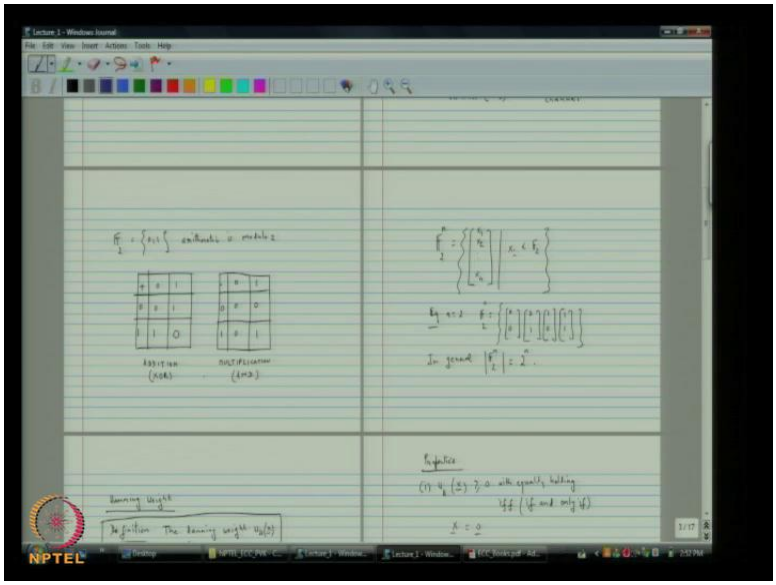
So, we will cover that, we will spend three-four lectures on it, and then go on to talking about the modern codes, which are examples of which are low density parity check codes, and then turbo codes. And after that the end we will actually start talking about the algebraic view point, which will begin with finite fields and then want to talking about a specific classes of codes, which make use the algebraic view point.

(Refer Slide Time: 09:53)



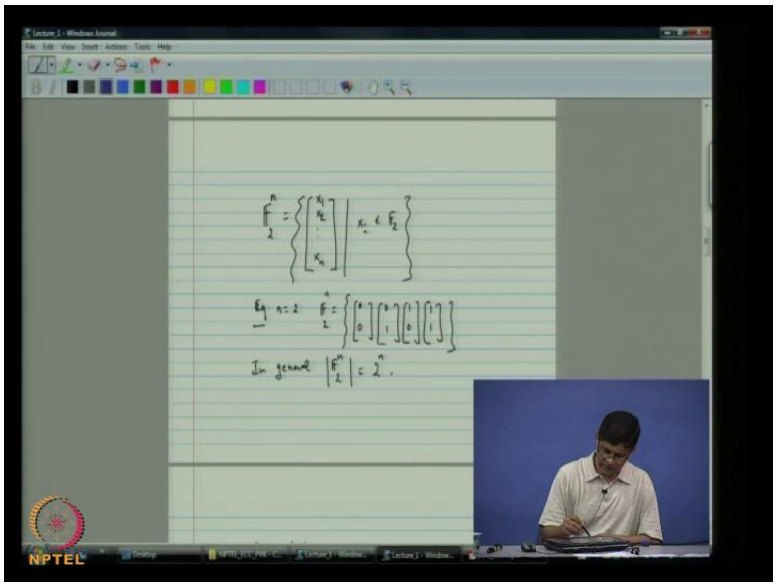
Then next thing that I did, was to actually talks so maybe I can access this zoom out little bit; so that was the first part. Then the next thing that I went on to do is to consider to discuss the, to discuss channel models here. And the channel particular channel module that we will be dealing with for the first few lectures is so called binary symmetric channel model. And what I did there was I explained how a second channel called (( )) channel and which perhaps is closer to what you might imagine a channel looks like, how that in approximation of that leads to the binary symmetric channel. The binary symmetric channel is the one that we will focus on. So, the input is both binary the input and output are both binary so, that leads to question how do you do a arithmetic working with just binary set 0 and 1.

(Refer Slide Time: 10:47)

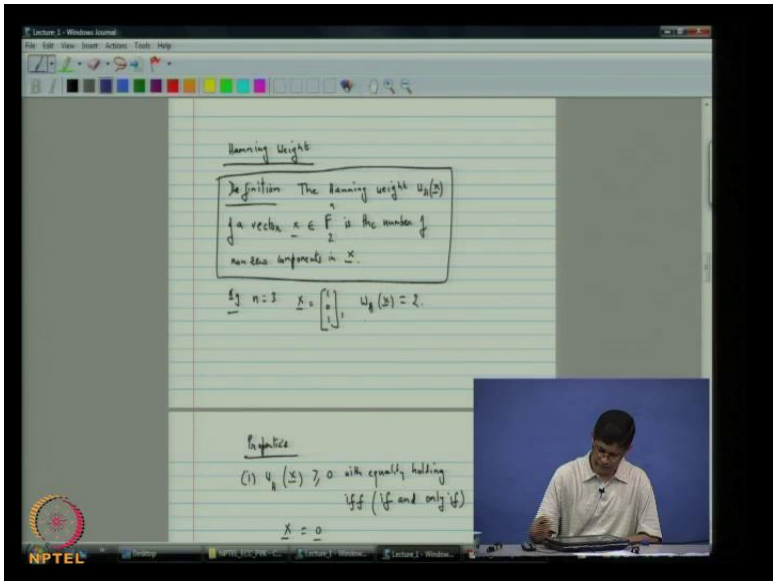


So, we talk about addition and multiplication and how this are carried out, when we have set is 0 1 and then we actually say how do we extended for the case, when you have vectors, where the symbols individually come from 0 and 1. Then we introduce some terminology, we introduce what is meant by the hamming weight, hamming weight of the vector is the number of non-zero components in the vector. So, here you see the space of vectors they the symbols come from f 2. I give you an example.

(Refer Slide Time: 11:10)

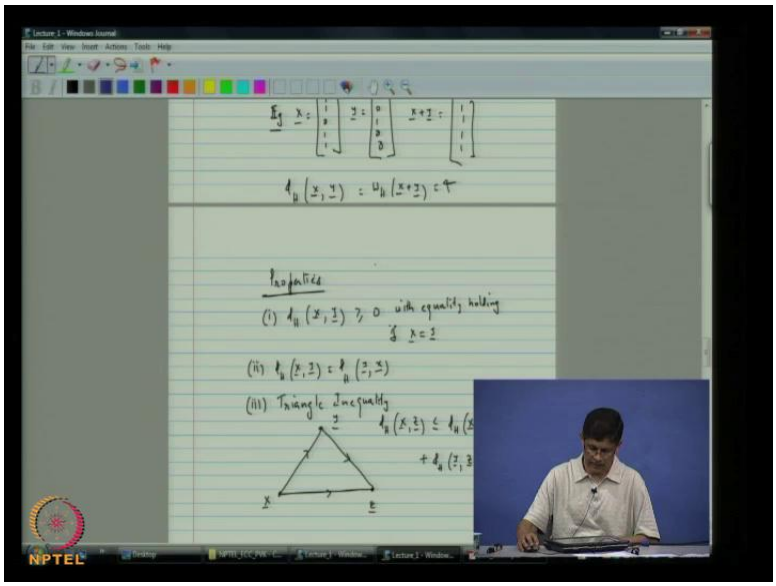


(Refer Slide Time: 11:18)



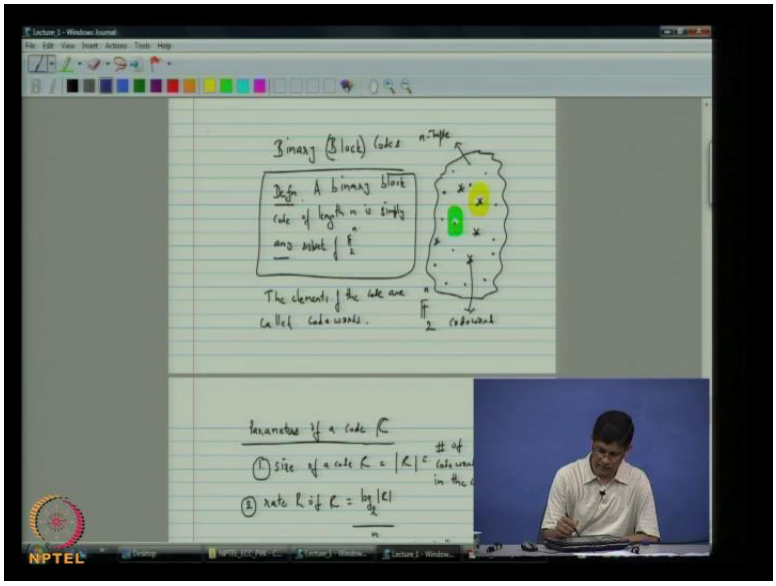
The hamming weight is the number of non zero components and the hamming weight function has certain properties, which we actually discussed, and this led naturally to discussion of the hamming distance between two vectors..

(Refer Slide Time: 11:40)



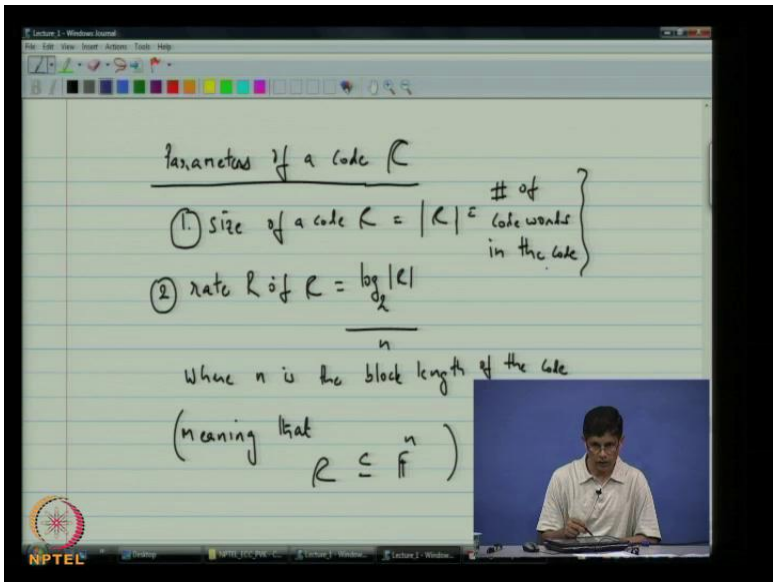
And we define the hamming distance as simply the hamming weight of the of the some vector and again there are some properties, one of them being the triangle in the quality which corresponds to this figure. After that I said, I think we are now have the set up where actually tell you what in a very general way a binary block code is?

(Refer Slide Time: 12:07)



So, binary block code is simply any subset or the set of all n tuples; and here what we see in this figure are collection of n tuples, and certain selected ones among these correspond to the code words. And error correction takes place, because given that you transmitted the code word and given that the channel has perturb the code word to another vector that close by, you can often correct that errors simply by saying well this is transmitted this is received, but since this is closest to this most likely that was transmitted. So you guess what the transmitted code word, and that is how error correction takes place.

(Refer Slide Time: 12:56)

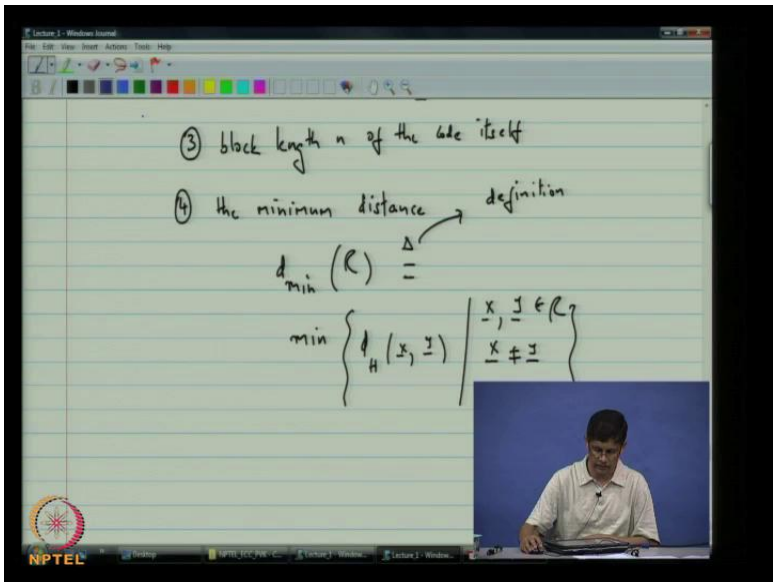


And then I started talking about the parameters the different parameters of a block code, one is of course, the size which in some sense measures how much information. So, one parameter of course, is this size of a code, which is simply a count number of code words in the code. The second is the rate of the code, the rate of the code if you taken information theory is a major of how much information you are sending along with the code word per channel use. If the code word has the length n, then in reality what you are doing is, you are using up n channel uses. So, the amount of information when you average per channel use, terms out from an information theory point of view.

And the units in this case are bits by channel use, bit being a very clearly define quantity in information theory. So, its log of the size of the code to be use to divided by n that is number

of bits the channel use it have transmitting; so the bigger the code the more the information that you are sending.

(Refer Slide Time: 13:47)

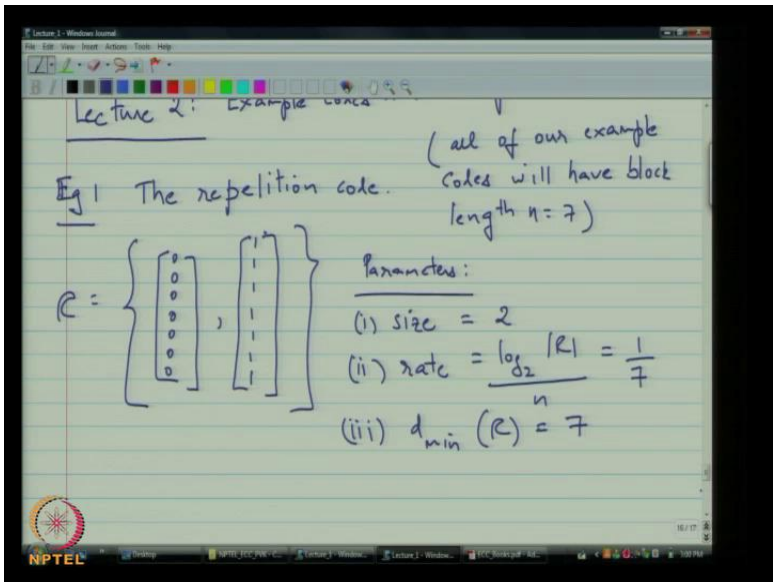


And the third then there is of course, the block length of the code itself, which is here simply the number of components in each vector and n there is the minimum hamming distance; the minimum hamming distance is the is is really a measure which tells you how close to, too close two code words come together, because this smaller the distance the lesser the error correction capability of the code. So, what that tells us is that building an error correction code is really all about trying to actually within the set of n tuples picking certain vectors as code words; and then number of them is the function of how much information you want actually convey but how much error correction capability you get is really depended upon how well spaced these vectors are in this space.

So, having introduced the parameters, so what will do today and that is the title of our talk is we look at some example codes and their parameters. Now, typically what I do is, I ask questions in the class then I say, does anybody know of an error correcting code? And the and usually we do get responses. And the most common responses are very good examples to begin this theory with. So, I will actually start with those. So, the first example that you

might actually think of is that repetition code. Now all of our example codes will have block length 7. So, let me make a note on this side of on that.

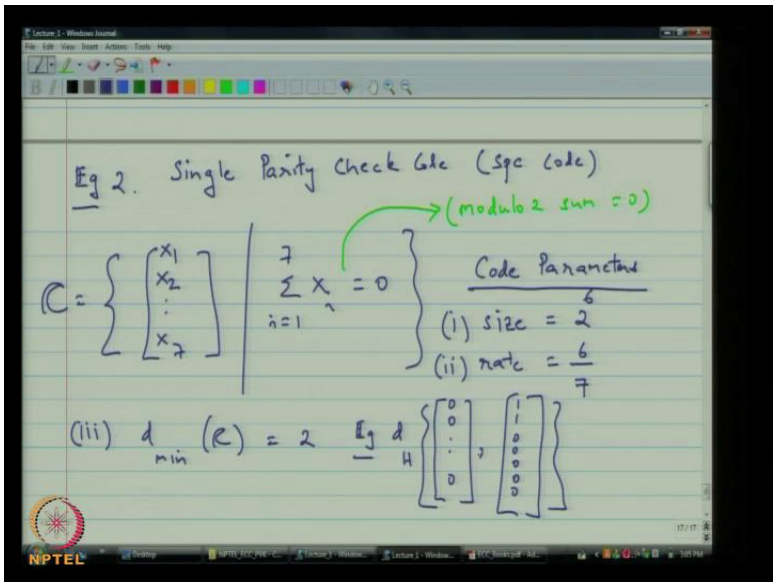
(Refer Slide Time: 15:17)



So, the repetition code in this case will just simply consist of two code words, the code word beings the all 0 code vector as well as the all 1 code vector. Now let us go about seeing what the parameters of this code might be so the size, so the size of the code is of course 2, the rate of the code is remember it is log to base 2 of the size of the code divided by n. So, in this case that would be 1 by 7. The third parameter is the block length, but I have already pointed out that all of our example codes will have block length n equal to 7. So, the last parameters the minimum distance of the code. So the last parameter is the minimum distance of the code; that is what is the minimum spacing between a pair of code words in the code, distinct code words in the code? And you can see in this this case the the minimum distance is 7.

So, this of course, since the vectors have length 7, I am sorry this one type here, this 0 should have been 1, let me just quickly correct that. So, there are, in this case, the code size is small, but the separation is large, in fact the separation is as large as can be.

(Refer Slide Time: 18:12)



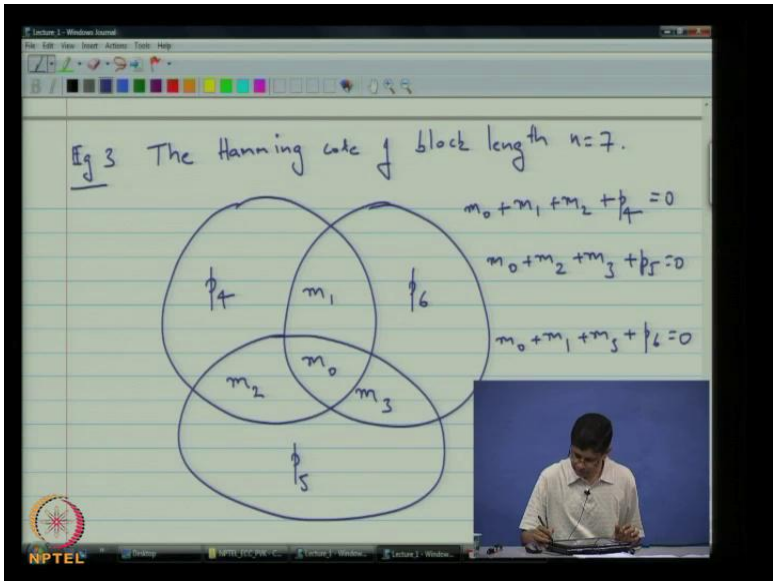
Let us go to our next example; so, example 2 is so called single parity check code and we will abbreviate this as s p c code, single parity check code. And the way you define this is that c is the set of all vectors of the form under the constraint that summation is equal to 0. Now just reminder here and I want discuss this many times, but whenever we talk about any arithmetic, we will always mean modulo to arithmetic. So for example, when I write down the condition that the sum of the symbols must be 0, what you mean is that this sum of after it is computed modulo 2. So, the modulo 2 sum is equal to 0 is the way you should actually interpret this. But I want to mention that in the sequel we will always assume that to be the case.

Now again with regard to code parameters, you have that this size of the code is 2 is 2 to the 6, and does not have to see because after all you can choose the first six components arbitrarily and this seventh symbol gets fixed by the parity constraint. The block length of course, is 7, we chosen that the rate of the code is we are going to take the log of the code size to be 2 divide by 7, so that will be 6 by 7. And then that brings us to the minimum distance of the code. The minimum distance of the code is the minimum distance between a pair of code words; and if you think about it, this another way you can actually try to answer the question is to, what the minimum distance for code might be is just to ask yourself the question. If let us that X 1 through X 7 is a code word then how many of this symbols must I

flow in order to get a second code word; and it is not actually had to see that if you flip one symbol, then the parity condition will be violated and therefore no two code words can be one symbol can have a hamming distance of 1. We cannot define just one symbol. But you can certainly flip two symbols and satisfy the even parity constraint, thus the minimum distance of the code is actually equal to 2 in this case.

And if you want a small example of that you can it is not at all hard to construct for example, you can consider the hamming distance between the vector of all zeros and the vector whose first two components are one and the remaining components are all 0 so, hamming distance is 2. And both cases I know that their code words, because after all the parity check constraints is actually satisfy so that is our second example.

(Refer Slide Time: 22:23)



Now, let us look at our third example, the third example will be the hamming code. It turns out that the hamming code is really a chain of error correcting codes and these codes can have different lengths, there always of the form 2 to some power minus 1. So, now we are looking at the code whose length is 2 to the 3 minus 1, there are you can present the hamming code in a number of ways. Some years back I listen to a seminar by well known coding theorist Bob Macklis and he presented it from a rather interesting point of view and he was saying, you know I can explain the hamming code to a 5 year old. And we were

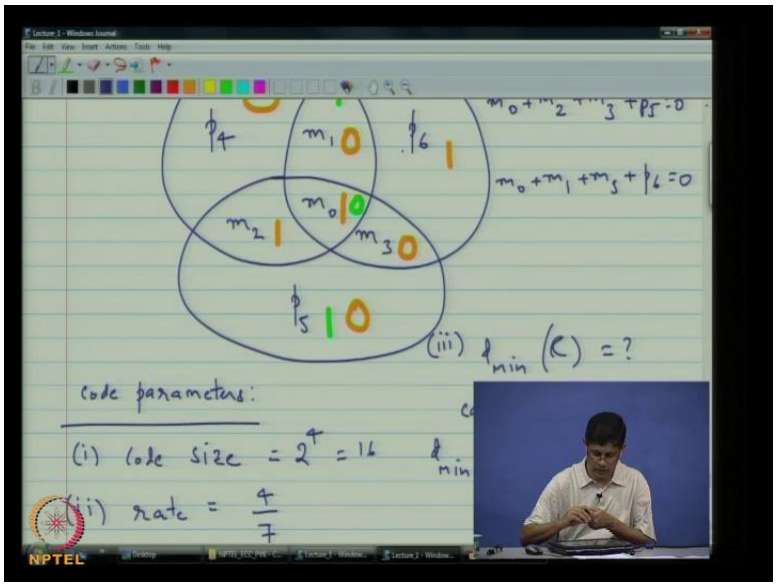
spectacle it first, but, as he gave the lecture, we can clear yes of course, you can do that. So, I am going to present that few point to you and afterwards we will will view it from a different perspective, which will allows to generalize it to a large class of course.

I have myself presented this few point to some, some children, who were fifth graders so, not quite 5 year old, they were 10 years old and they seem to understand perfectly well. And going to put down some symbols and I actually I called it the magic of 3 circles when I actually presented it to them. So, let me just introduce some notations. So m 0, m 1, m 2, m 3, so each of this represents a symbol and it turns out that there in the hamming code that you can pick this symbols anyway you like, but of course, this is the binary code, so you can only make it either 0 or actually 1. So, there only two possibilities, so that means, the total number of code words is therefore, 2 times, 2 times, 2 times, 2, which is16.

So, the hamming code will actually contain sixteen code words. Let us go about seeing how after all the code is block length 7 that means, there are 3 more symbols to this code words, let us go about introducing them. So, the code symbols are I will put down p 4, p 5 and p 6. So, there are these three symbols and the way the hamming code is going to impose the requirement that in any in any circle the parity must be even. So for example, let us see arbitrarily chose the message symbols m 0, m 1, m 2 and m 3, then the parity symbols must be chosen in such way that every circle you actually have even parity. So, in particular that means that the following equations must be satisfied namely that m 0 plus m 1 plus m 2 plus p 4 equals 0, m 0 plus m 2 plus m 3 plus p 5 equal 0, m 0 plus m 1 plus m 3 plus p 6 is equal to 0. So, for each of the circles you actually write down a parity check equation. So now you can see, why I call the first example a single parity check code, because there is just a single parity check that had to be satisfied there, whereas, here you actually have to have satisfy three parity checks one corresponding to each circle.

Now of course, the question is how how is it that this code corrects errors etcetera, but we come back to that little bit later, right now I am just giving you an example of a code in terms of just saying how is it constructed.

(Refer Slide Time: 27:18)



So, let us now look at code parameters so one the size of the code, is the size of the code as I told you earlier sixteen because you free to pick the message symbols, and then after that the parity symbols are all fixed. So, the size of the code is 2 to the 4 which is 16. And as the consequence is this the rate of the code the rate of the code is the log of this to this 2 divided by 7 so, the rate of the code is therefore, 4 by 7. The third parameter is a minimum distance of the code. What is the minimum distance of the code? So the minimum distance really is asking the question, let us assume that someone put down a certain set of symbols. So, let me take take a particular example I will put it down in red.

So, let us say for example that I chose the message symbols to be 1, 0, 1 and 1. See the message symbols can be chosen arbitrarily so let us assume that in a particular instance they are, I am going make this little bit thicker, so that you can see this clearer in a more clear way. So let us say that again that this is 1, this is 0, this is 1 and this is 0; and these are your choices of the message symbols. Now you must make sure that the parity satisfied in each of these; and as a result for example, what you will do is you have your force to make this equal to 0, and here you have force to make this equal to 1 and here 0. So, you pick the four message symbols and the parity symbols are determined after that.

Now the question is so, that gives this is an example, this 7 tuple is an example of a code word in the hamming code. The the question with regard to minimum distance is, what is the minimum distance between a pair of code words or asked in different way how many symbols must I flip at a minimum in a code word to get a second code word. And let us try to this code word reference to the question is how many of these symbols can I flip and still get a code word? Now let us consider the various possibilities. For example, I might flip this symbol in this (( )) but, if I flip this symbol, now it is a part of three circles, so it is clear that if I flip this, then I have to flip at least one other symbol. Let us say there I flip this, because I have to make sure the parity in all three circles is sets, so if I change this to a 0, then I might try to balance out the parity in both of these by changing this to 1, but then I am still I still have to do something about this circle.

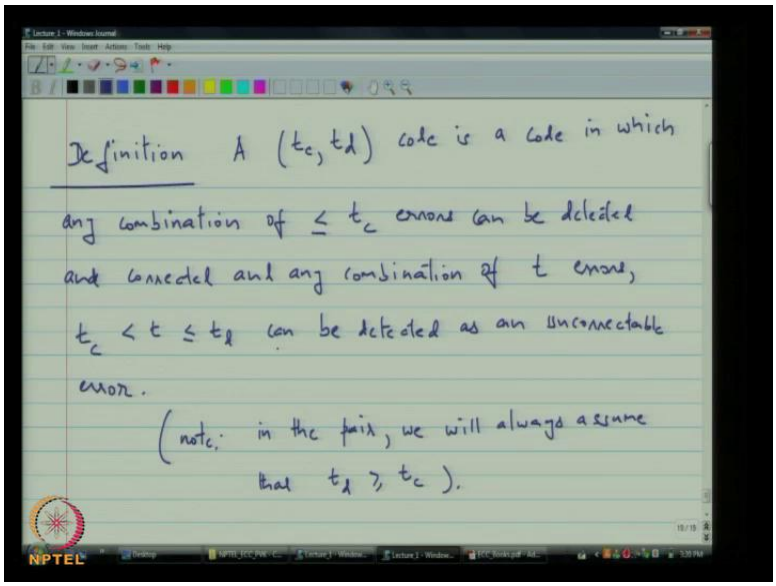
And since I do not want to interfere with what is happening here, I am forced to flip this. So, you saw there in this particular instance, you had to flip at least three symbols before you went from one code word to another. So, once again let me just repeat, so for example, if I tried let me try picking slightly different color see if we can show you this. So if I tried let us see changing this to a 0 and trying to make sure that the parity here was balanced, I am making that 1, now balanced parity in both of these circles, but then I am left with this circle, I do not want to change this or this because that will upset the parity in the two circles that are I have already balanced, so I tried to change this one, so I do this. So you see that I was force to put three green entries here, which meant that three symbols had to be changed, that is the indication that the minimum distance of the code is actually 3, that is not a complete proof, but I will let you complete the proof on your own the rather ways of showing later on. So, we will just put down here of now, that the minimum distance of the code is 3.

And I leave this is an exercise for you to prove that, no matter what your starting code word was that you can never change from one code word to another by just changing three symbols. So that is an important point because here what we did was we took a particular code word, and saw how many symbols we had to flip to get an another one, but of course, to prove this you have to show that no matter what code word you started out with that the same you will be in the same situation. No matter what you did? And also they were other

choices available to us, but no matter which situation you consider, you will always end up with this. So, the minimum distance of the code is 3. Now we actually finished looking at some example codes.

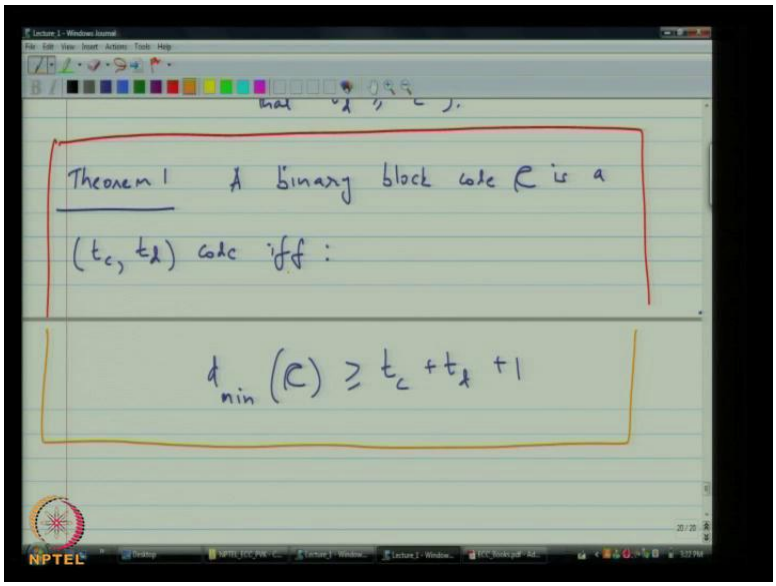
Of course the burning question is that is fine, so you told us what the code is, you told us what some parameters is the code are, but what I really like to know how do you use this codes for actually correcting errors? So for that we will we are going to need some definitions so let me get to that.

(Refer Slide Time: 33:09)



A t sub c, t sub d code is a code in which any combination, any combination of less than or equal to t sub c errors can be detected and corrected. And any combination of t errors, where t is greater than t sub c, less than or equal to t sub d can be detected as in an correctable error. This automatically implies that t sub d is always greater than or equal to t sub c in any code, so perhaps let me just in the pair. We always assume that t sub d is greater than equal to t sub c. So that is fine, but still the question is about how these codes correct errors. And will actually the next theorem will make this clear, because what we will do is we will tie in the minimum distance of a code, which we will able to compute to these parameters t sub c and t sub d.

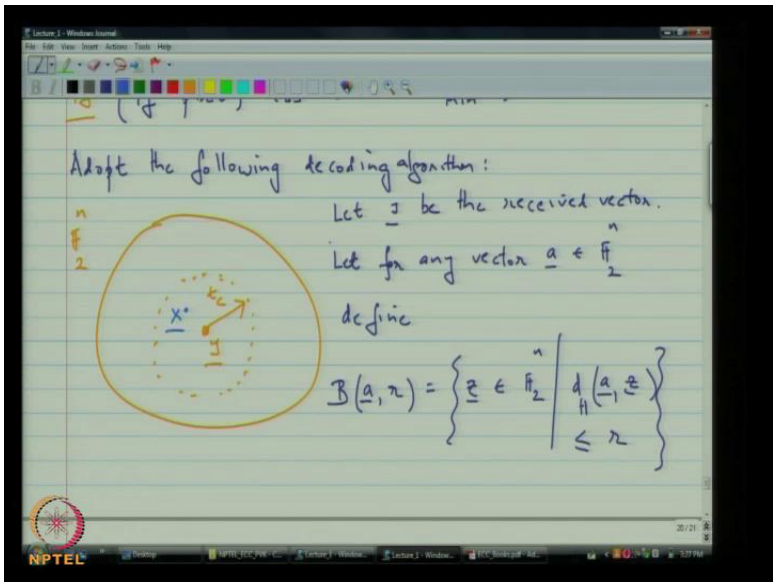
(Refer Slide Time: 36:40)



So here is our first theorem, binary block code C is a t sub c, t sub d code, if and only if the following is very straight, namely that the d min of the code is greater than or equal to t sub c plus t sub d plus 1. So only if this conditions satisfied is code t sub c, t sub d code, so how do you prove this? So, what we actually do is we will provide, so by the way this is if and only if, so that means that we have to show two things, one is that if this equation is satisfied then in fact the code can be used to correct this sub c errors and detect t sub d errors.

On the other hand to show only if we have to show that if this condition is violated, then it is not possible. And to show the if part, we will actually do this by exhibiting very simple algorithm.

(Refer Slide Time: 38:36)



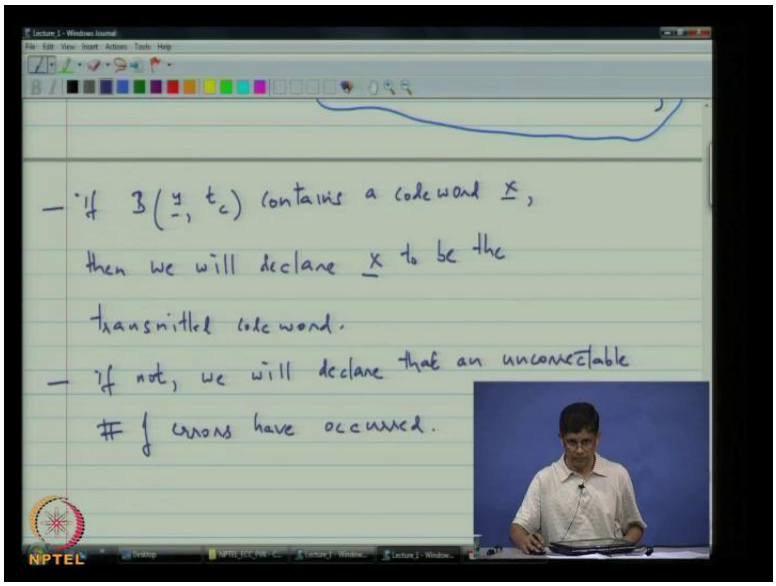
So here proof, so this will be for the if part, so here assume that d min is greater than or equal to t sub c plus t sub d plus 1. We will adopt the following decoding algorithm, so I am I am going to draw a picture that goes a long width, when I am going to describe, so in the picture this is an abstract depiction of the set of all n tuples, and here let us say that you have the received vector y.

And so what we are going to do is, we are going to draw, we are going to consider in the neighborhood of this vector a ball of radius t sub c and if there is code word in this ball, so let us say that there is a code word in this ball, which is located here let us say. Then we will declare that x was a transmitted code word. If there is no ball, if there is no code word in this ball, then we will actually say that the number of errors exceeded t sub c and we will detect an uncorrectable error. So, let me just write some of that down, let y be the received vector and now for for any vector a in F 2 to the n, let us define let us define B of a, r to be the set of all vectors of the form is call that z, z belonging to F 2 to the n. Such that the hamming distance between a, so a is a vector and z is less than or equal to r. This is a general definition.

So, for given any vector a, we define we can define this ball like this. So, when I was earlier talking here about a ball always referring to one such ball except that here it is going to be

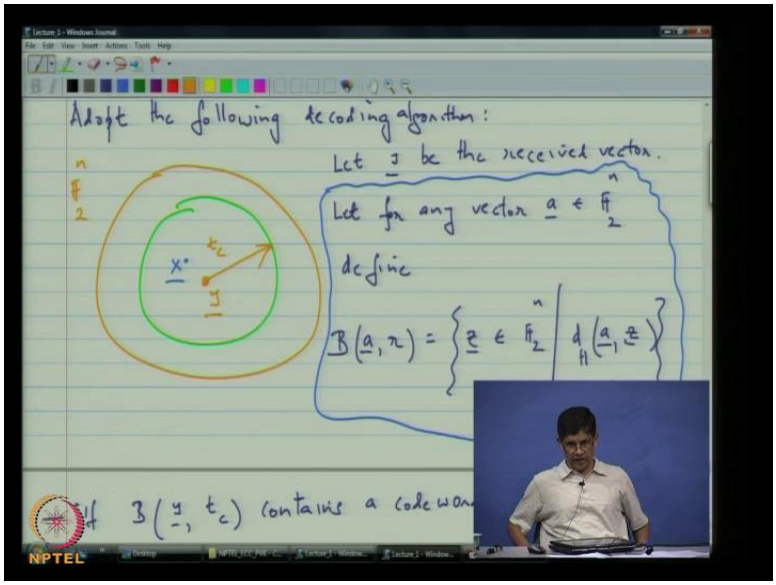
centered around y. So, our decoding algorithm will say let us look at the ball of radius t sub c centered at y and look to see, if there is a code word in there.

(Refer Slide Time: 43:13)



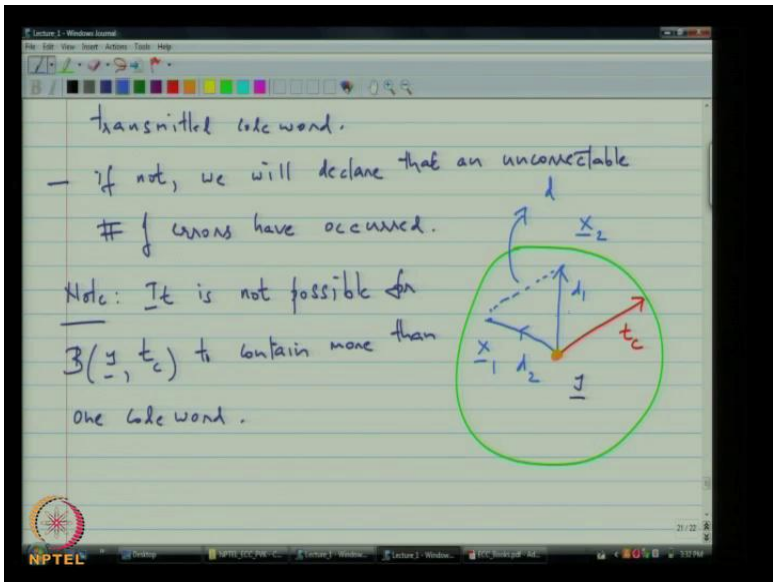
So, let me write that down, if B sub y t sub c contains a code word x, then we will declare x to be the transmitted code word. If not if not, we will declare that an uncorrectable; that an uncorrectable number of errors have occurred. So that is our algorithm.

(Refer Slide Time: 44:50)



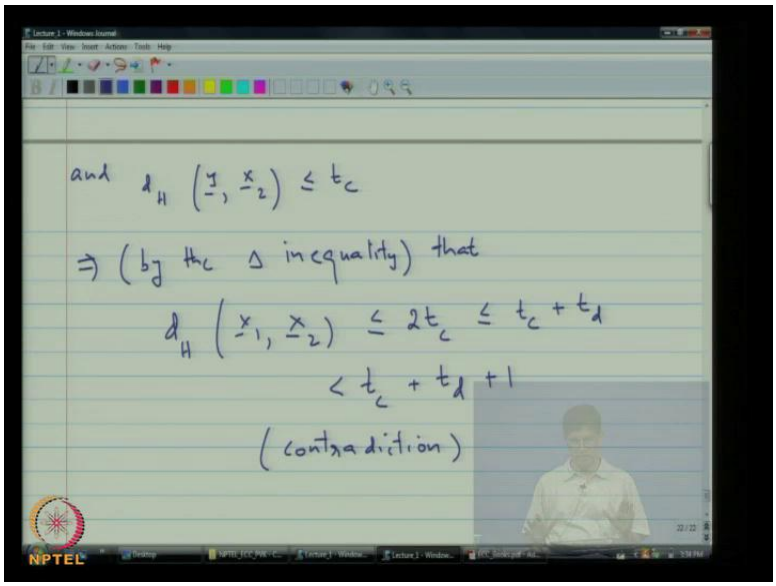
Let me just do one thing here, since this dotted line is not so clear, let me try to draw this circle in perhaps different color here. So, this is the same circle that I was talking about last time. So, this is your received vector, you are going to look in this ball and look to see if there is a code word in this ball. Now, often students say this point say sir, that may be true, but what you have not said explained to us is what will happen if this more than one code word in this ball? And what I want to point out is that it is not possible for more than one code word to belong to this ball.

(Refer Slide Time: 46:00)



Why is that? Because let me just draw that so we will make that note, so it is not possible for B y t sub c to contain more than one code word, the reason being so let me draw this circle again here. So, let us say that they were two code words whose distances were, let us say that their distances were d 1 and d 2. So, now within this ball if let us say that there are two code words and both of them belong to the code, then what you would have is that the distance between these two code words the two code words themselves. Let us call this distance d and what that would implies that the distance between these two code words by the triangular inequality is less than or equal to the sum of these distances, which means that is less than or equal to d 1 plus d 2.

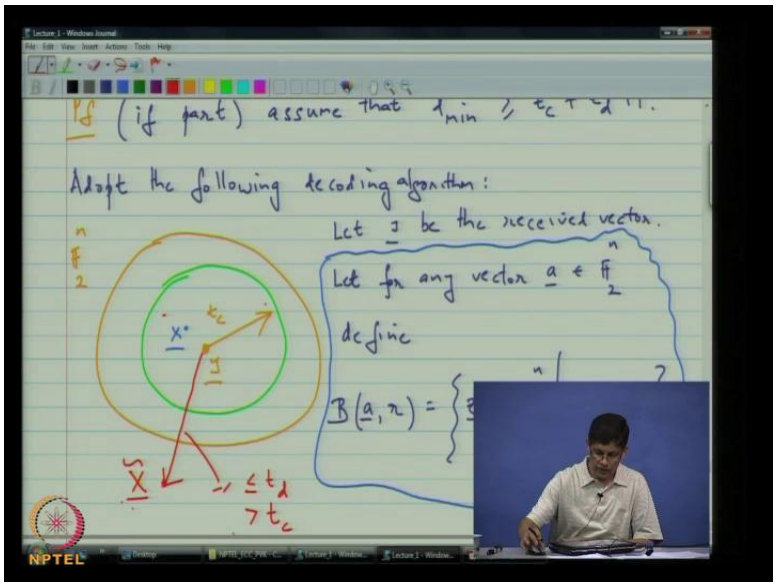
(Refer Slide Time: 48:22)



So, if so if if the hamming distance between y and x 1 is less than or equal to t sub c, and the hamming distance between y and x sub 2 is less than t sub c, this would imply by the triangular inequality, that the hamming distance between x 1 and x 2 is less than or equal to 2 times t sub d c, but this is less than or equal to t sub c plus t sub d, which is strictly less than t sub c plus t sub d plus 1. So, that is a contradiction.

So, what that means is that we ruled out the possibility that the ball of radius t sub c around y can contain more than one code word. So, the only possibility is therefore, are that either the ball contains no code word so, we are back here either this ball does not contain a code word or it contains exactly one. Now let us so, what I would like to claim is that the algorithm does what it promises, there is if the number of errors is less than or equal to t sub c, it will give you back the correct code word; and if the number of errors is greater than t sub c, but less than or equal to t sub d, then it will tell you this, there is an error, but I cannot correct it.

(Refer Slide Time: 50:25)



Now let us let us say for instance, that let us say that let me just clean this up a little bit, so let us say that in a particular instance there was a code word, and you decode it to that code word. So, the question is when how could you go wrong? You could only go wrong if there was some other code word that was transmitted; and so let us say that there was some other code word that was transmitted, which is let us say, let us call that other code word, let us put it in a different color, let us say let me see if yellow shows up in a screen here. Let us say that I will call this I guess, yellow is not a good choice. So, I may try different color, let us go with red. Let us say that there was another code word here, which is let us put x with a waggle on it; that this let us assume that this was actually the transmitted code word, but when you found another code word in this column. Now we know that this cannot belong inside the ball so that is what actually happen is that perhaps, this was the transmitted code word and that this was received with the number of errors being less than or equal to t sub d.

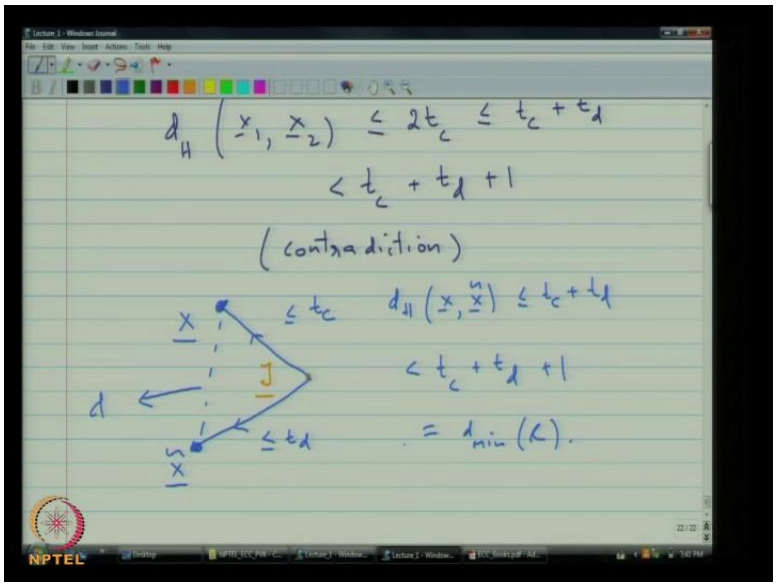
So, may be that has happened, and which case you would be an error, because what you should have done was you should have detected that there was an error, which was uncorrectable. And as this figure shows we have already agreed that x from cannot belong in this ball. So, the distance must be less than or equal to t sub c d, but greater than t sub c so, in this case what we should have done was you should have declare an uncorrectable error. Instead what you did was you declared x to be the the actual transmitted code word, which

means you made a mistake, which you should not have, but that is assuming this picture is true. But this picture really cannot be true, because if you look at this here, you cannot have a pair of code words x and x (( )), which with this set of distances and I will explain that.

So, once again but before I do that let just to summarize that, we are in a situation, where the vector y has been received, and you declared x to be the transmitted code word. So, only question is could you have gone wrong, now x itself was the transmitted code word then there is no problem. And we know that whatever if if x was not the transmitted code word the transmitted code word must be somewhere in this place. And we are only interested in the situations when the distance from the transmitted code word is less than or equal to t sub d, wise that because all that here algorithm is promising is that its saying that if the number of errors is less than or equal to t sub c, then I then I will correctly correct the errors. But if it is greater than that, but, less than or equal to t sub d, I will declare it uncorrectable error. What happens beyond that there are no promises there are no guarantees, if the number of errors is greater than t sub d, then you just throw up your hands and give up. And that is perfectly fine, because we are not promising anything that region.

So, for that reason we are when we are trying to actually if there are performance of the code, we only need to consider this two cases, the case when the transmitted code word is within this ball, which will shown cannot happen or the case when it is outside the ball, but its distance less than or equal to t sub d. And the question is, could this happen? And as we will see for the same reason, because of the triangular inequality even that that cannot actually happen.

(Refer Slide Time: 54:44)



So, here is your received vector and our concern is that on the one hand there is on the one hand, there is one code word here; whereas, the true code word is over here. And the distances is that you have are that these was less than or equal to t sub c and this distance was less than or equal to t sub d. But again that the triangular inequality that would imply that the distance d between x and x tilde would be t H of x x tilde would then be less than or equal to t sub c plus t sub d would be less than or equal to t sub c plus t sub d, which would be strictly less, excuse me, which would be strictly less than t sub c, this t sub d plus 1, which is again a contradiction. Because which is thank you, which is the minimum distance of the code.

So, that we have shown is that you will not have an error even in this case because even this situation cannot error occur. So, what we shown is that if there is a code word within the ball you are going to make an error. Now what we need to actually come back and show is that what if there is no code word in that ball, then what you are suppose to do is you see you suppose to declare an uncorrectable error, and what we will do in the next class is we will actually show in there in that case, if you do declare an uncorrectable error, then in fact that is exactly what is happened, provided the number of errors is not greater than or equal to t sub d.

So, I think this may be a good place to actually stop. So, just to quickly recap what we done in this lecture, we looked at three example codes, there are predation code, the single parity check code and the hamming code. And we looked at the parameters, then I actually explained, so now we are in the process of making the connection, the important connection between the code parameters and the error correction capabilities of the code and that is captured in this theorem. And we are in the middle of proving the theorem, we are almost done. We should be able to do that rap that up in the first part of the next lecture. So, let me stop here; and I will catch up with you all in your next lecture. Thank you.