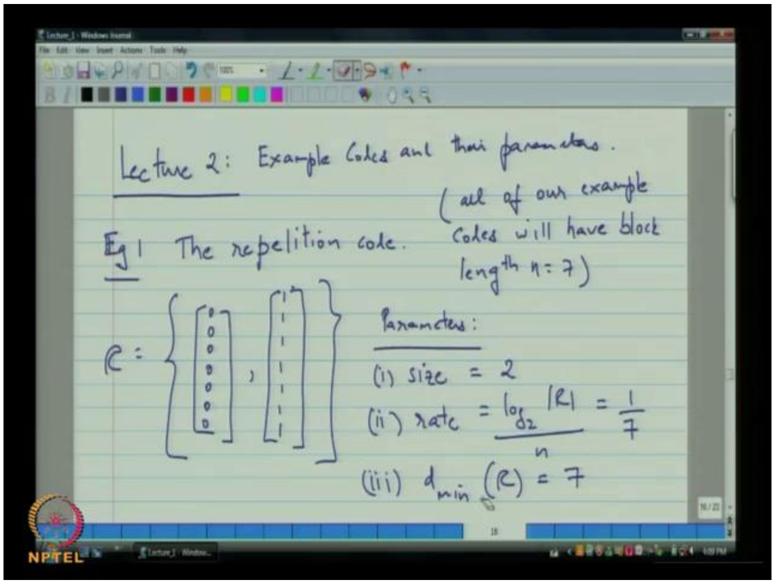
**Error Correcting Codes**
**Prof. Dr. P. Vijay Kumar**
**Department of Electrical Communication Engineering**
**Indian Institute of Science, Bangalore**

**Lecture No. # 03**
**Mathematical Preliminaries: Groups**

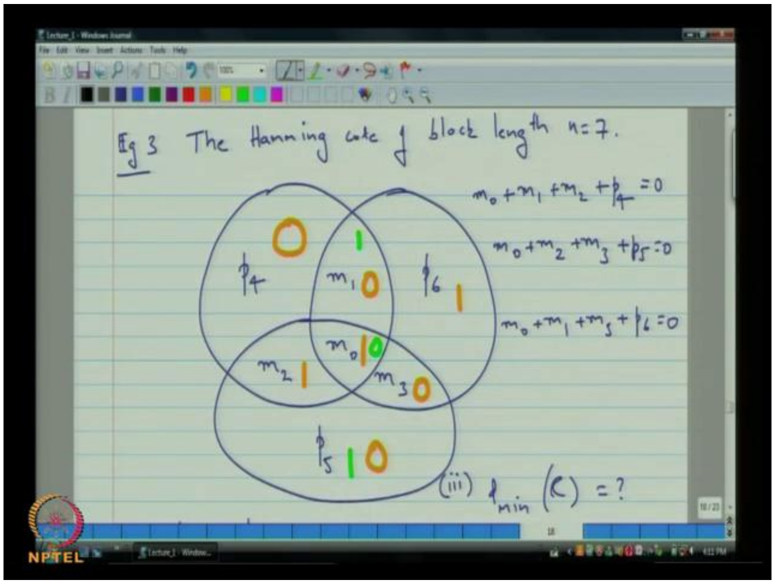
(Refer Slide Time: 00:15)




This is the third lecture in our series on Error Correcting Codes, and as before what I would like to do is, I begin with recap of what we did last time. In a nutshell, what we did last time was we look at some examples codes in that parameters, and then towards the end of the last lecture, we try to say well, supposing we have a code whose minimum distance is such and such, then what is how does that relate to the error correcting capability of the code. And we were in the middle of that proof, so we will pick up from that point onwards. First, what I will do is, I will just go over the materials relating to example codes in the parameters. We will go over that quickly, and then after that, I will come back and talk about and continue where we left of in the proof.

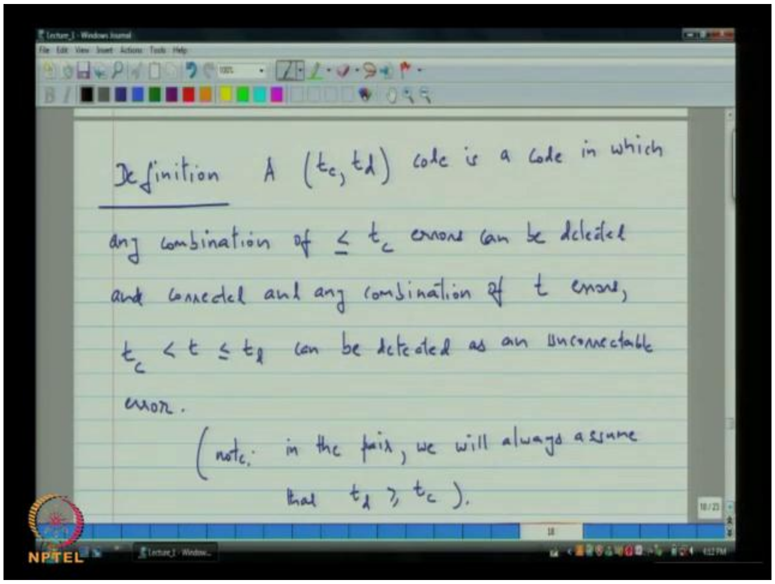
Here, if you go down to the part here, you see that example codes in there parameter, I first introduce the repetition code. In the repetition code, all the symbols are the same. The parameters and the codes are the following, the size is 2; the rate of the code is 1 by 7 and the minimum distance of the code is 7. The next code was the single parity check code, and here the condition on the code is that, the sum of the symbols may actually be zero, and as a result you find that the size of the code is 2 to the 6; the rate is 6 by 7, and the minimum distance is 2 and these are two example code words whose distance is 2.

(Refer Slide Time: 01:53)



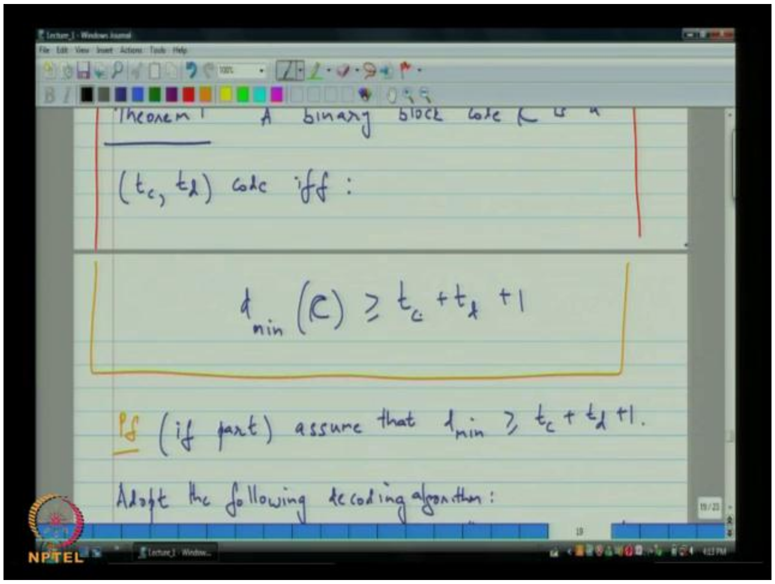

Then the hamming code; hamming code is defined a little bit differently explained it in terms of these three circles, and I said the way it is defined by introducing certain message symbol m 0, m 1, m 2, and m 3, and then filling in the corresponding parity, and that gives rise to this equation. And from these equations, it is possible to actually determine the parameters of the code, which in this case turns out to be, so the size of the code is 16, because there are four message symbols; the rate is 4 by 7, the minimum distance you can show 3 and left it to you was an exercise. Later on, we will actually see an alternative, and perhaps simpler way of actually proving that the minimum distance of this code is actually 3.

(Refer Slide Time: 02:45)



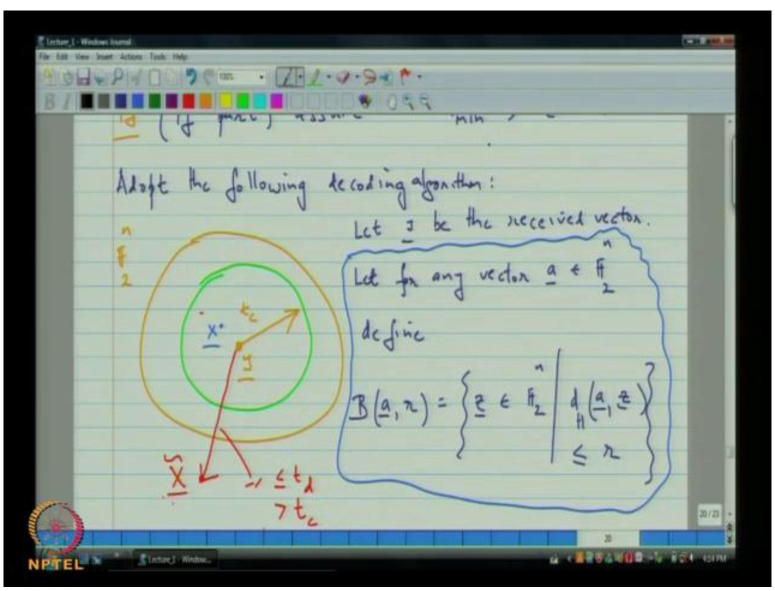

Now, next I am moving on to talking about the relationship that exists between the minimum distance of a code and the error correction capability of the code. So that, we begin with definition which says that t c, t d code is a code, in which any combination of less than or equal to t sub c are us can be detected and corrected and any combination of two errors, where t lies between t sub c and t sub d can be detected as an uncorrectable error. And in this pair, you will always assume that t sub d is greater than or equal to t sub c. So, whenever we define this, it is analytically been implied that while this can be the same as this; the second parameter t sub d can be the same as the first, in general it is larger.

(Refer Slide Time: 03:38)




And the theorem that we were in the mixed of rolling is that the binary block code c is a t c, t d code if and only if the minimum distance of the code is greater than or equal to t sub c plus t sub d plus 1. And now, i f f here means if and only if, so which means we have to prove two things; one is we have to show that if this condition is satisfied, then the code is in fact capable of correcting t sub c errors and detecting t sub d errors.

(Refer Slide Time: 04:09)



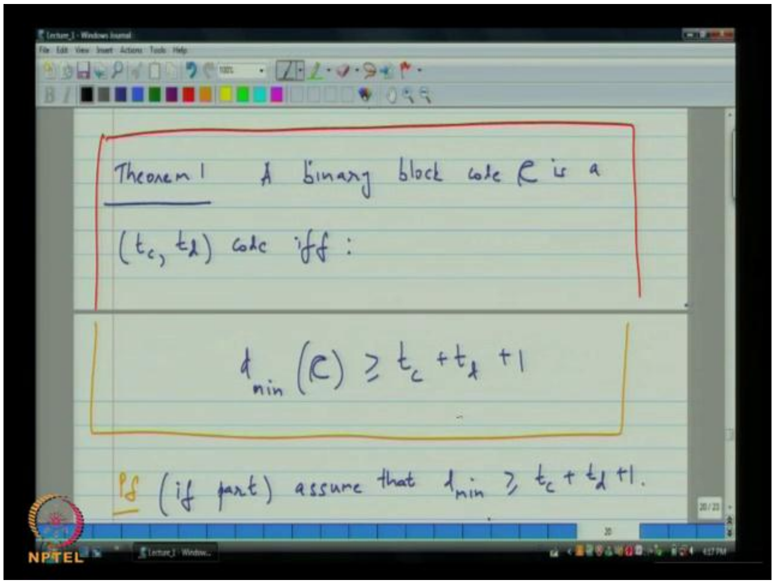
We went through the proof of if part I will go through that once more again, and then pick up for the only if part. So, let us assume that in fact you have given that the minimum distance of the code satisfies the condition like d minus greater than or equal to t sub c plus t sub d plus 1. And now we are going to show that the code can correct the sub c errors and detect t sub d by exhibiting a particular decoding algorithm.

The decoding algorithm runs as follows, so by the way one definition that we use here is that of a ball given any vector a and given a real number r, which you think of the is a radius. The ball surrounding the vector, the set of all n tuples were the Hamming distance between a and z is less than or equal to r that is what we defined as a ball and what we done. And what we shown in this figure here, is that here is your receive vector y and decoding algorithm proceeds as follows. We will first of all examine the ball at centered at y whose radius is t sub c, and that is the green circle that you see on your screen. Now, given that ball, what we are going to do is we are going to look to see, if there is a codeword that is contained within the ball.

Now, it could be that there is a codeword; it could be that there is more than one codeword or it could be that there is no codeword. At the outside these are the possibilities we narrow down the possibilities as we go long. So, we will first examine the ball or the neighborhood of the vector y whose radius is t sub c; if there is a single codeword in the ball, then we actually declare that x was the transmitted codeword and decode x in the decoding algorithm ends there. On the other hand, if we look at this ball and we find that there is no vector are contained in this ball, then we will say that an uncorrectable error has taken place.

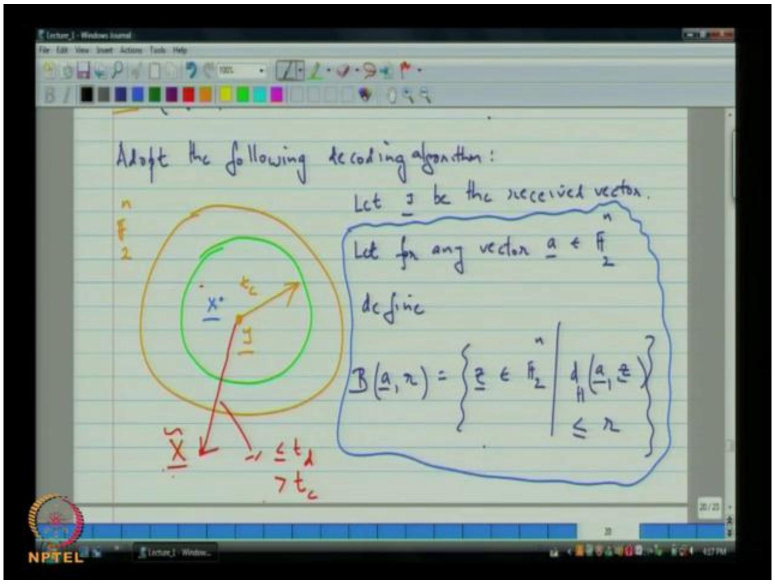
This is very simple. There are two cases and we take different actions depending upon which of the two is actually true, now to show the actually work. Let us examine, what could possibly go wrong? Now, there is another possibility that is maybe; that is running through your mind. What if this ball contains more than one codeword but as will see in the proof that is not going to be possible. I will not discuss that at from this is a special case. Here, now supposing you look at the ball at y of radius t sub c, and you happen to find that the there is a vector accent, and then what you do is you declare that x was a transmitted codeword, what could go wrong? The only thing that can go wrong is, if x was not the transmitted codeword, which means there was some other vector, which was the transmitted codeword. Let us, put x tilde x with a vigil on top. It pronounced x tilde was actually the transmitted codeword.

(Refer Slide Time: 07:33)



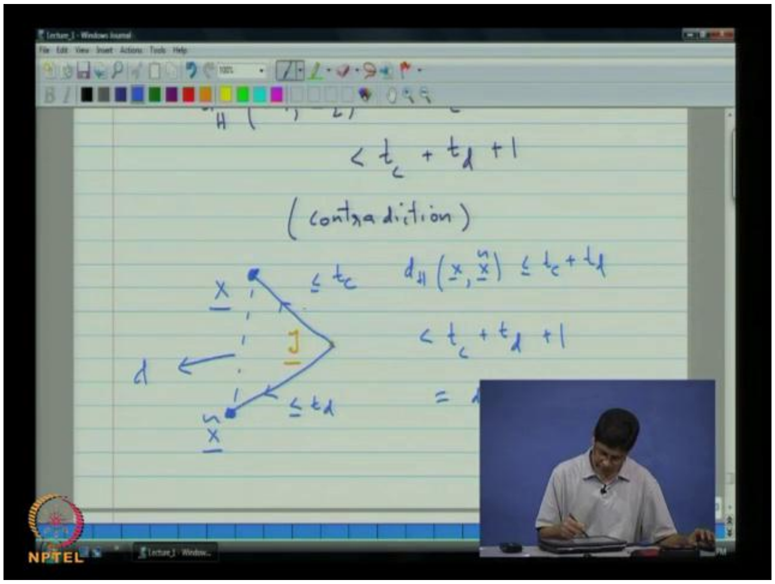

But one think to keep in mind is that R promises with respect to this code are only and so for as correcting two sub errors and detecting t sub d errors. If in fact, it turns out that the number of errors is greater than t sub d, then we do not guarantee anything; we do not promise anything; we do not required to deliver anything. So for that reason, it is perfectly for the code to break down if the number of errors is more than t sub d, so we will keep that in mind.

(Refer Slide Time: 08:00)




Here for that reason, we can restrict our attention to the case, when in fact, the number of errors is less than or equal to t sub d. Let say that here is your transmitted, actual transmitted codeword x tilde, and if it turns out that was not equal to x then you made an error. Now my goal is to show, but this cannot happen, because and the simple reason for that is if you take a look the distance between x and y; the distance between y and x is less than or equal to t sub c, simply because it belongs to the ball. On the other hand, the Hamming distance between y and x tilde is less than or equal to t sub d. The reason being that your only confining your attention to these situations.

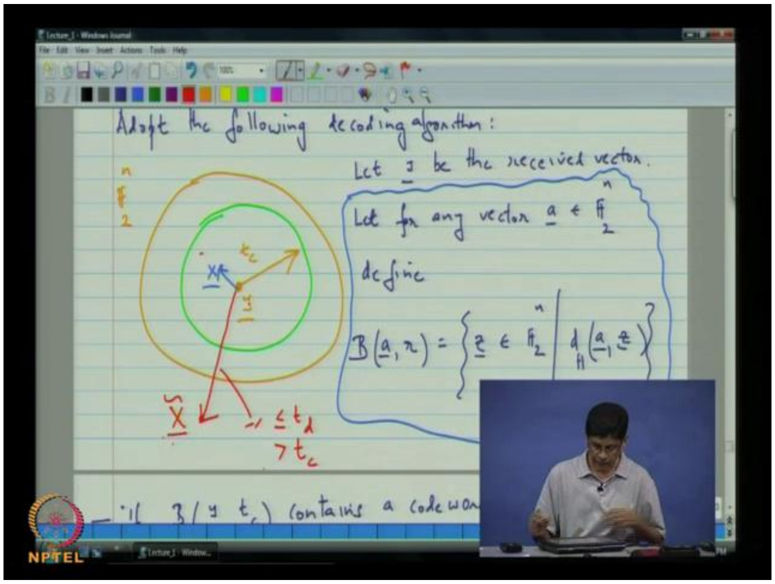
(Refer Slide Time: 08:55)



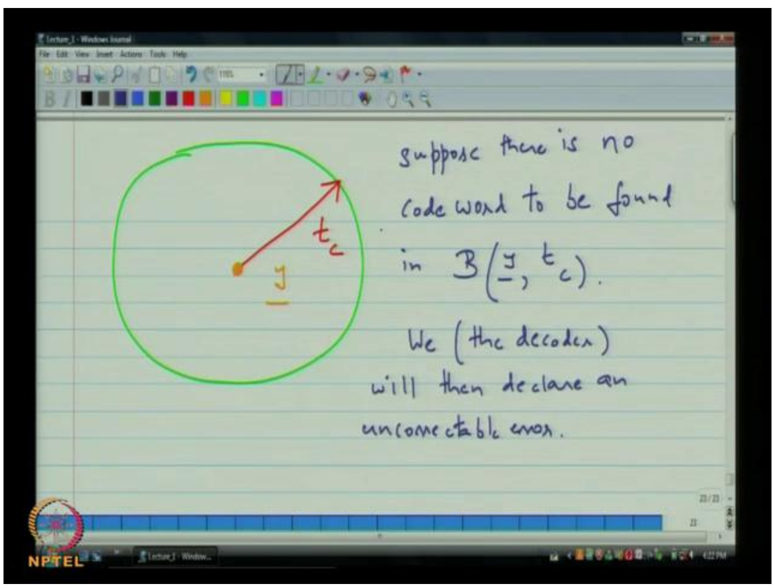

All right now, if you put the two together then what you actually have is that here, at so after some discussion in which basically, I expand and what I have just said, you have the situation over here, where this thing here is your vector y received. We have x whose Hamming distance is less than or equal to t sub c, and have x tilde whose Hamming distance is less than or equal to t sub d. But that would imply that there are two codeword in the code, whose Hamming distance is less than or equal to t c plus t d sub d, but that would imply that there are two code words in the code whose Hamming distance is less than or equal to t c plus t d, because this distance plus this distance is greater than or equal to the distance along a straight line path.

Let me just do one thing here, so the point is that the distance from here to here is less than or equal to the sum of these two distances, which much mean as, if you look at the argument on the right? Here, right naught in blue the Hamming distance between x and x tilde is less than or equal to t sub c plus t sub d plus 1 strictly less, which is the minimum distance of the code. So that is a contradiction, because after all, no two code words in the code can be separated by less than the minimum distance. That proves that this situation is pictured that we showed our front is in fact not possible, it cannot be that the geometric picture looks like this.

(Refer Slide Time: 10:18)



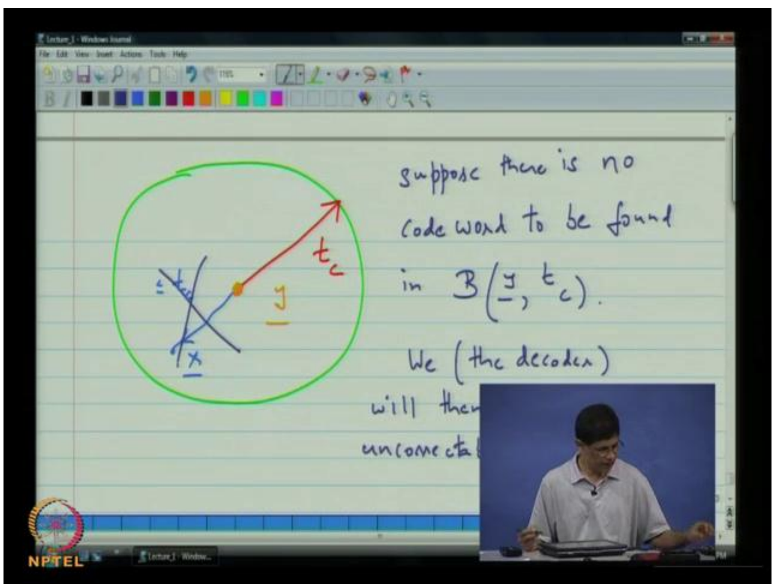

(Refer Slide Time: 10:35)




This picture is not a true picture in other words. Now, that leads as to the other situation which is, what if, what if and let me repeat my argument; what if I have my received vector y. And I draw a ball, whose radius is let say t sub c and this time and in the middle of decoding algorithm and I find that there is no codeword in this ball another question is what can go wrong? Now, let me put that down. Suppose, there is no codeword to be found in the ball centered at y, whose radius is t sub c, and then the question is what can go wrong? The only thing that can go wrong is because now what you are going to do is; you are going to actually declare an uncorrectable error. We now, we remind the decoder will then declare an uncorrectable error.

The only thing that can possibly go wrong is, if in fact there was a correctable error; the only way we can wrong is, if there was a correctable error, but what does that mean? That means the only way we could have wrong gone wrong is if there was a vector whose distance the only way in which you could have gone wrong is if there was a codeword is which was correctable there was a codeword, whose which was correctable; there was a codeword and there was a number of errors which was correctable which would imply that there was a codeword whose Hamming distance was less than or equal to t sub c. But there was clearly not possible right, because we started off we look to the ball we found that there was no codeword in there, so there is no question of there been the other possibilities.

(Refer Slide Time: 13:00)

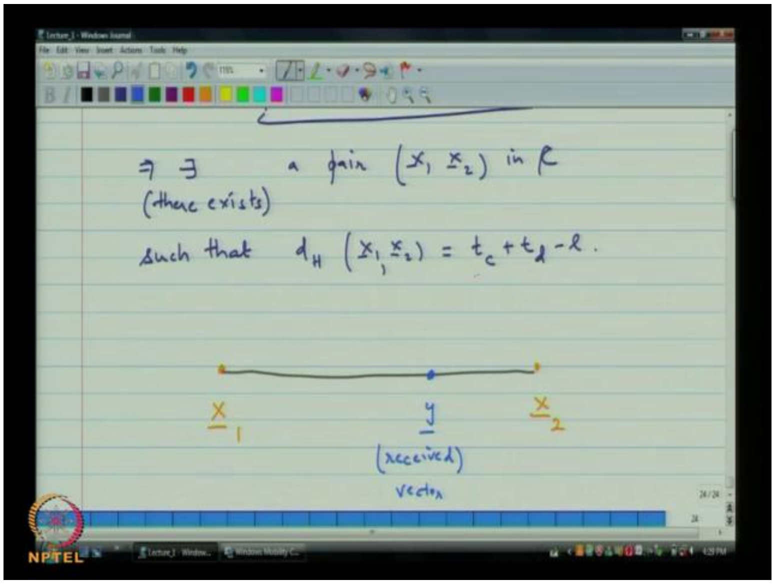



I am just going to drawn this but I am going to rule this out just to emphasize this situation is not possible and clearly, we cannot go wrong. The conclusion is that we cannot in this case go wrong. Clearly, the decoder will be correct, since the only way it could possibly go wrong is if there was a correctable error, that is a codeword within Hamming distance t c that of I, but this is impossible by our initial assumption that the ball was empty; that concludes the proof of a if part.

And is possible that the first time you go through this, you are still a little bit puzzled, but you just have to write this throughout on your own.

Now, I am going to proceed to the only if part, so the proof of the only if part, so here, we actually show that the in equality that we are treated down earlier was in fact necessary. We have to show that d min greater than or equal to t sub c plus t sub d plus 1 is in fact, necessary. Suppose not, which implies that d min was less is less than strictly less than t sub c plus t sub d plus 1. Let us say, d min is equal to t sub c plus t sub d minus l say, l could be 0 or larger than 0. Let us, assume that this was the case; and now, but the fact that the minimum distance of the code is given by an expression like this implies the existing of a pair of code word separated by such a Hamming distance implies, there exist.

(Refer Slide Time: 17:30)




This symbol means there exist, a pair of code words x 1 x 2 in the code c, such that the Hamming distance between x 1 and x 2 is equal to t sub c plus t sub d minus l. And now, you let us look at the following picture; you have on the one hand, one side, you have x 1, and then here, you have x 2. And I am going to draw a line between these two; and let say that now let say I am going to put down a vector in the middle here, somewhere in the middle, and I am going to call that y. And by y, I am using the symbol y as before to denote the received vector. Now, there is there is something to prove here, I mean you first of all, what do I mean by vector lying on this line; what I really mean is that the distance from end to end on this line is t sub c plus t sub d minus l.

Let us, choose a vector in such a way that the distance from here to here from x 1 to y is in fact t sub d. This distance let us say is t sub d, and at the remaining distance is let us say t sub c minus l. Now there is something to prove here, because you have to show that given two vectors x 1 and x 2, whose Hamming distance separation is t sub c plus t sub d minus l that in fact, you can actually find a vector which is kind of in between them was such that the sum of the distance is from x 1 to y, and from y to x 2 adds up to t sub c plus t sub d minus l. But that well easy to see, because the meaning of Hamming distance or an alternative interpretation of the Hamming distance is simply the number of symbols that you need to change to go from x 1 to x 2.

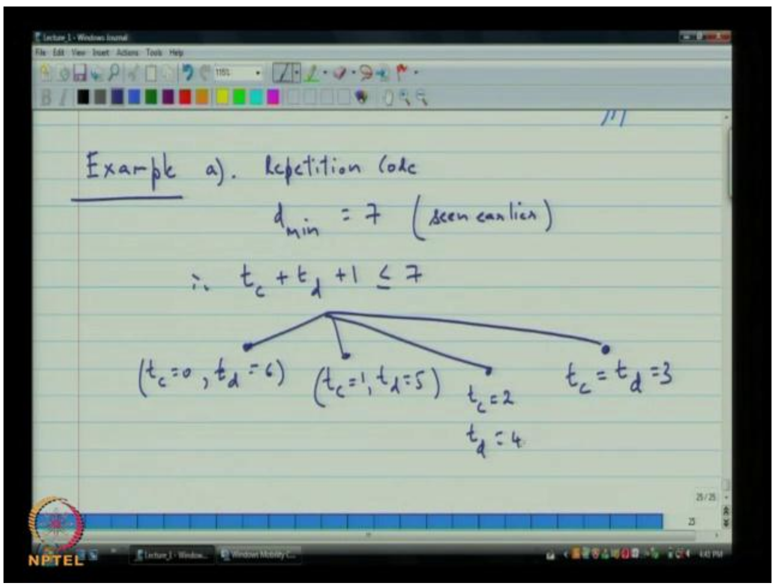
You need to change, t sub c plus t sub d minus l symbols to go from x 1 to x 2; so supposing I change t sub d of them and stop in the middle, then I will be at the vector y, and I meet t sub c minus l more changes to get to x 2. This diagram, in other words, this figure makes sense, and it is perfectly possible that x 1 that you have a code in where x 1 and x 2 are the code words, and where y is the received vector.

And the problem here is that this situation presents the decoder with dilemma that cannot be resolved, because of what can the receiver do, if it receives y? It is like the same which says that I am donned, if I do and donned if I do not. The matter what the receiver does in this situation, it is going to potentially been error, the received the decoder being deterministic given y it has to say something, it has to either declare but there is an error and corrected or it has to declare an uncorrectable error. Now, because the distance from x 2 to y is t sub c minus l, which is less than or equal to t sub c, it is possible that y was the result of x 2 been transmitted. So, with that in mind the decoder to take care of that case should given that y was the received vector should have decode to x 2.

But similarly, it perfectly possible that x 1 was the transmitted vector, and they were t sub d errors; so both possibilities exist and they call for two different actions in the receiver, in order for the receiver we correct. It is impossible that the matter how you design the decoder? You are going to be an error in one of the two cases; you cannot be wrote write in both cases, and that is why you cannot correct t sub c and detect t sub d errors. If the minimum distance was insufficient, so that concludes the proof.

I will just make a brief note along the lines of what I have just said, so will say that the situation about presents the receiver with a dilemma that cannot be resolved. For the case, when y is the received, so thus when d min is less than t sub c plus t sub d plus 1, a code cannot be a t sub c t sub d code and that concludes this proof. And that was a lengthy proof and perhaps you got last in a couple of in an argument along the way, but just keep at this the result in mind. Namely, that in order for it to be able to be a t c it means this inequality to be satisfied.

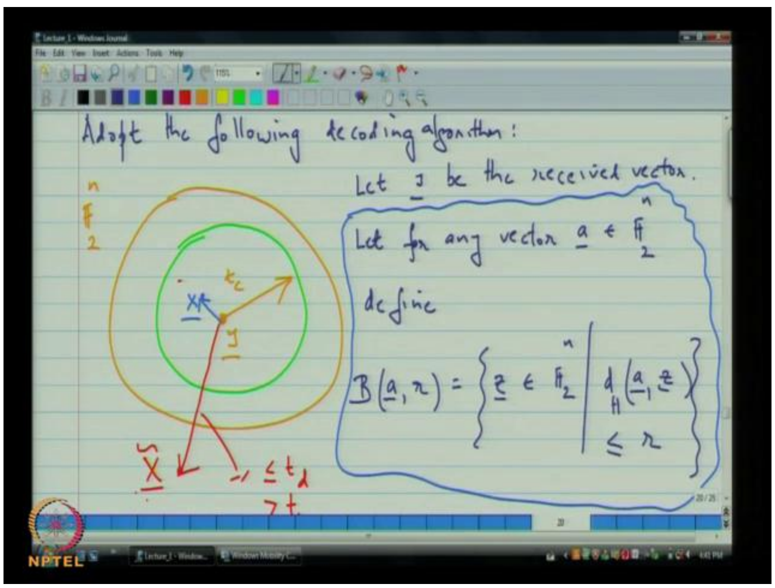
(Refer Slide Time: 24:48)




And now, let us go to our example codes. The first code been the repetition code here, the minimum distance of the code was seven so, we say this earlier, and therefore we have the inequality t sub c plus t sub d plus one ok that must be less than or equal to seven. So, that code is t c t d code for all pair which satisfies the in equality now the interesting thing is that this inequality is all that is required. You can use the same code at to provide different measures of error correcting capability. So, from this, you can actually use the code on the one hand as a code in which t sub c is equal to 0 and t sub d is equal to 6. You could use it as a code for which t sub c is equal to 1 t sub d is equal to 5. And then at the other end of the spectrum, you can also use it as a code for which t sub c is equal to t sub d is equal to 3. You can use this code in these different ways; I guess the only one that is really missing in this list. Let me put that down as well is the case, when t sub c is equal to 2 and t sub d is equal to 4, you can user this code in four different ways.

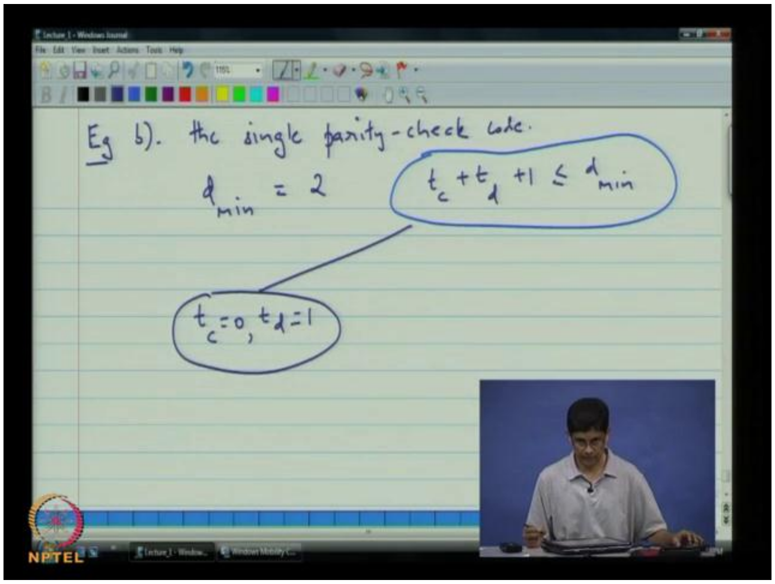
The one thing that might puzzle you is how is it possible to use the same code to provide different measures of error correcting capability. And the answer is because remember that you are decoding algorithm was centered around, it involved using the value of t sub c, because remember we had that you would look in a ball.

(Refer Slide Time: 27:50)



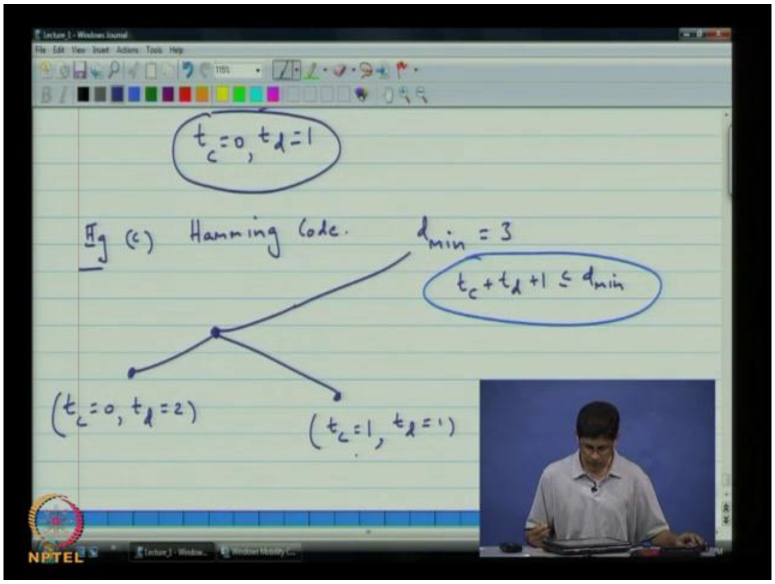

I can find that for you, so in our decoding algorithm what we actually did was we took the received vector drover ball of radius t sub c around it and look for a codeword right. Now, t sub c was larger or smaller this ball will get bigger or smaller, the decoding algorithm changes with the parameter choose said that you actually choose, and this is what allows the code to provide different measures of error correcting capability. This is for the case of the repetition code.

(Refer Slide Time: 28:20)




Now, let us look at the some of the other codes example b. The single parity check code, here, for this code d min if you will recall was two so since the minimum distance is small that does not leave a whole lot of room for selecting a parameters. According to so let us put down the equation again that t sub c plus t sub d plus 1 must be that is b. So, it does not leave very much to play. In fact, the only thing that you can really can do in this case is choose t sub c equal to zero t sub d equal to one. So, the code can only be used for detecting single errors, it cannot be use for error correction.

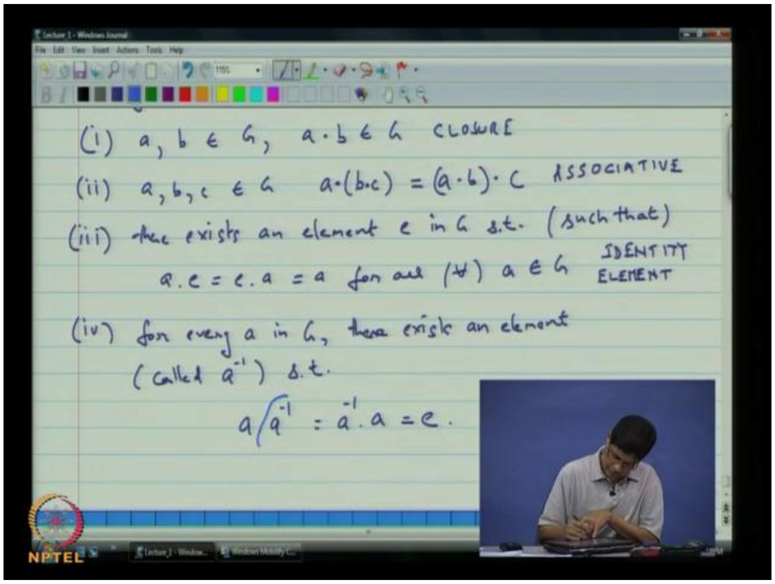
(Refer Slide Time: 29:31)




Moving on to our third code example c, which is the Hamming code. So, in the hamming code, the minimum distance of this code was equal to 3. From this, it is clear that the code can be used again with respect to our equation which says that this code can be used in one of two ways actually; in one way you could use it as a t sub c equal to 0 t sub d equal to 2. You can detect whereas, but not correct any errors. The other alternative is to correct a single error, and this corresponds to the case when both t sub c and t sub d are both equal to 1. You will see in three examples of how of the codes minimum distance and the associated error correction capabilities.

Now, I would like to just say few words about where we are going to go next. We talked about binary codes in general, and now what I would like to do is focus on this sub class of binary codes, which are the so called linear codes. Now linear codes essentially make use of linear algebra but there is also a little bit of group theory, and since I assume that some of you may not have a background in these two areas. What I will do over the course of the next two lectures or so is to actually try to fill in that background. So, we will make a slight detour, we leave coding theory aside for the movement, and we will talk first about groups and sub groups and later on, will talk about rings fields and vector spaces that is going require some passions on your part, to sit through a couple of lectures where we talk about a elementary a very elementary abstract algebra and then you come back to coding theory. Of course, the benefit of these lectures is you have the background you can just skip quite ahead and catch up couple of lectures later. I am going to go on to discuss some background on groups.

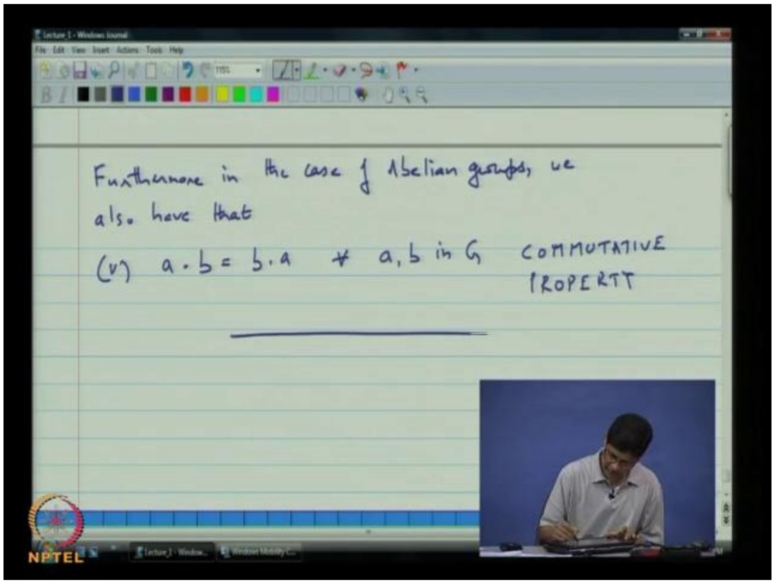
(Refer Slide Time: 32:44)




I will entitle these mathematical preliminaries. So, we will talk firstly about groups, definition: a group G comma dot is a set G along with an operation, which is dot, under which following conditions are satisfied. First of all, if a, b belong to the group, then a dot b must belong to the group, this is called the operation of closure. If you have three elements a, b, c, then and if you are interested in multiplying the three together, in the way in which group them prior to multiplying does not make a difference. Then a times b c ,b dot c, so a dot b dot c is equal to a dot b dot c. So you can multiply b and c together first, and then bring an a, why you can multiply a and b and then bring an c and it does not matter. So, this is called the associative property.

The third property is there exist an element e in g, such that so when I write as dot t dot you should read that as, such that a times e is equal to e times a is equal to a for all, and I will often write is this symbol to denote all. For all a in g, then four so this axiom is called the axiom of the identity element. The fourth axiom says that, for every a in g there exist an element, which is called a inverse such that, a times a inverse is equal to a inverse times a is equal to e and so, this is pronounced a inverse and this is called the axiom of the inverse. Now, this is four axioms collectively make up a group.

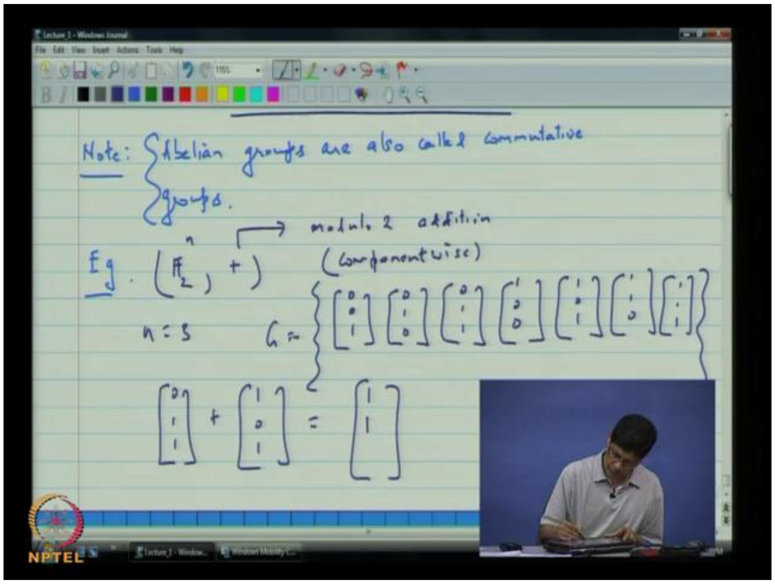
(Refer Slide Time: 37:52)




Often, the groups that we will deal with will be abelian groups, further more in the case of Abelian groups, they also have that a b is equal to b a for all a b in G and this is called the commutative property, other commutative axiom. And Abelian groups, so that those are the five axioms that you define a groups and this make up quick remark that note Abelian groups are also called commutative groups and it is not surprise. So, once again just quickly go through the axioms. There are five is them you need closure that is if… Now you can think of this operation is either multiplication or radiation right now, it is just a binary operation; in our example it will often correspond radiation.

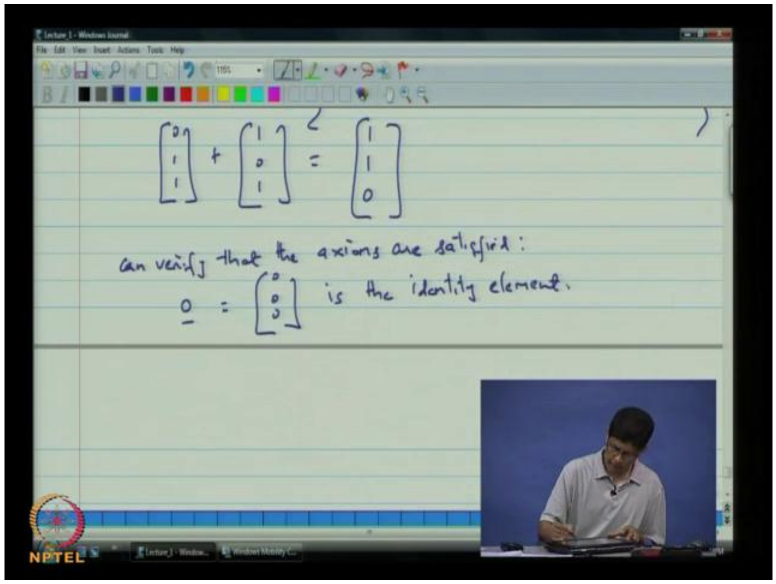
What the first one is saying is that if one operates on two elements you get a third element in the group and this is saying in order which you multiply does not make a difference. There is an identity element, which preserves elements upon multiplication, every element has an inverse, and then case of Abelian groups and often will be with abelian groups, you also have the additional axiom that the elements commute under this operation, which means that the odd m which you write them does not make a difference.

(Refer Slide Time: 41:02)



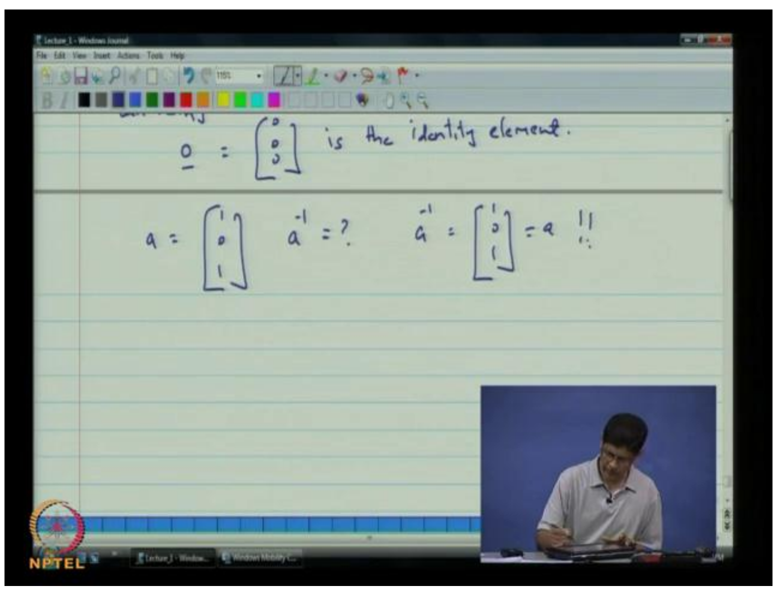

So put a some examples, the first example is a consider the set of all n tuples, where the operation is plus, and by plus what we will mean here is modular two addition component wise. For if let say that n is equal to 3, in which case your group your group three, the group G will be comprised of eight vectors. So, your group our G will be comprised of these eight elements and the way in which you actually add is component wise modular tree. So, 0 1 1 plus 1 0 1 would be 0 plus 1, which is 1 1 plus 0, which is 1 1 plus 1 which is 0, because you are doing modular two arithmetic, and you can check that, the axioms has satisfied.

(Refer Slide Time: 42:54)



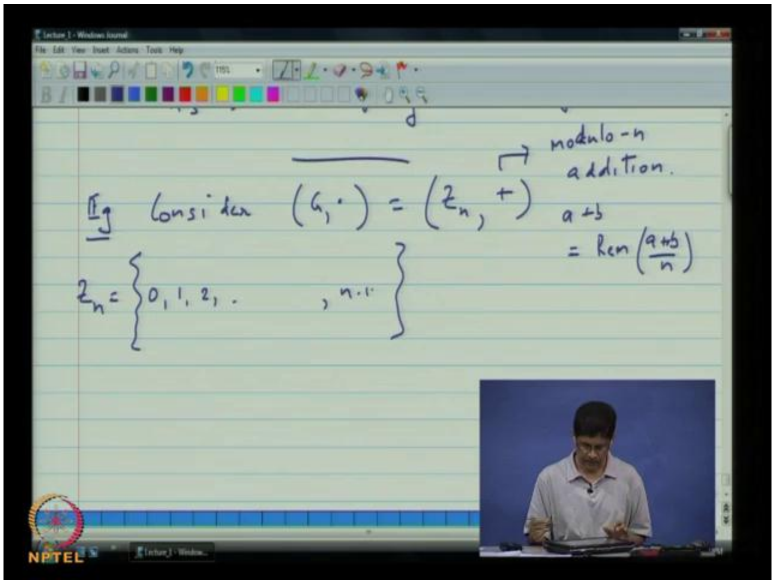

For example, so can verify that in particular zero, which is the vector whose components are all zero plays the role of the identity element. So, this is the identity element.

(Refer Slide Time: 43:37)



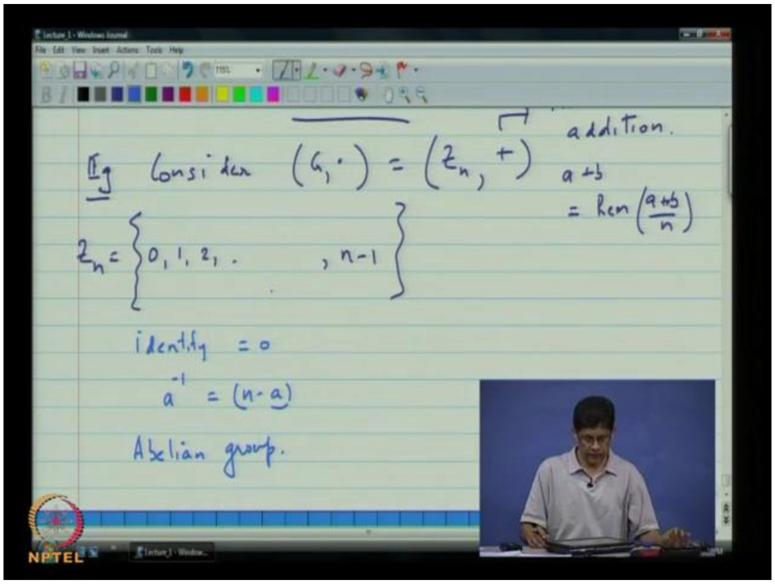

And, if you want, if you thinking in terms of the inverse, let say, a is the element 1 0 1 and you might be wondering what is a inverse? Remember that a inverse is some element that has to be added back to a to give you 0. So in this case, since you are doing modular two arithmetic a inverse is in fact a itself. So, it is equal to a, so that c s for this particular case, and of course addition is commutative here, so this is an example of an Abelian group.

(Refer Slide Time: 44:53)



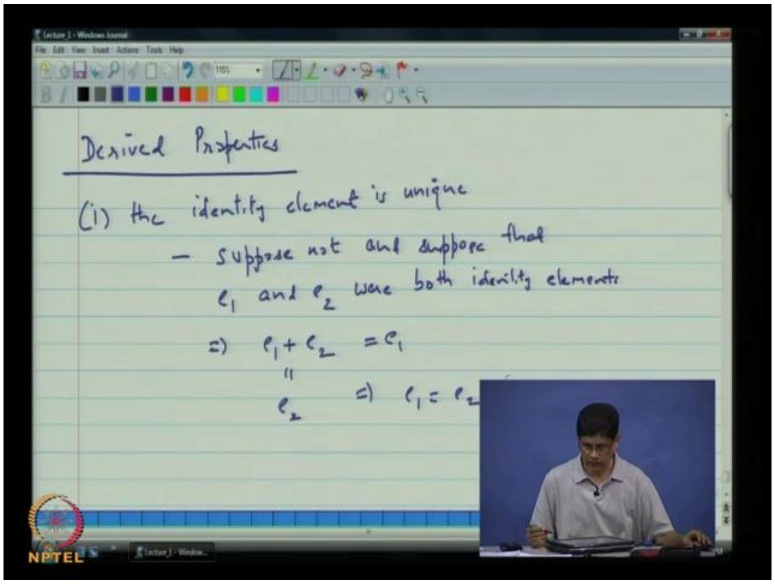

Let us look at another example, consider G dot to be z n and a plus. While, this time plus denotes modulo n addition. So, what does modulo n addition mean? It means that a plus b a plus b is really the remainder after you divide a plus b by n. And z sub n is the set of integer is modulo n and strictly speaking, it is a collection of equivalence classes, but we will choose representatives. So, we will pretend that this is just, this particular set, the set of integers ranging from 0 to n minus 1. So, there are n elements in the set.

(Refer Slide Time: 46:55)



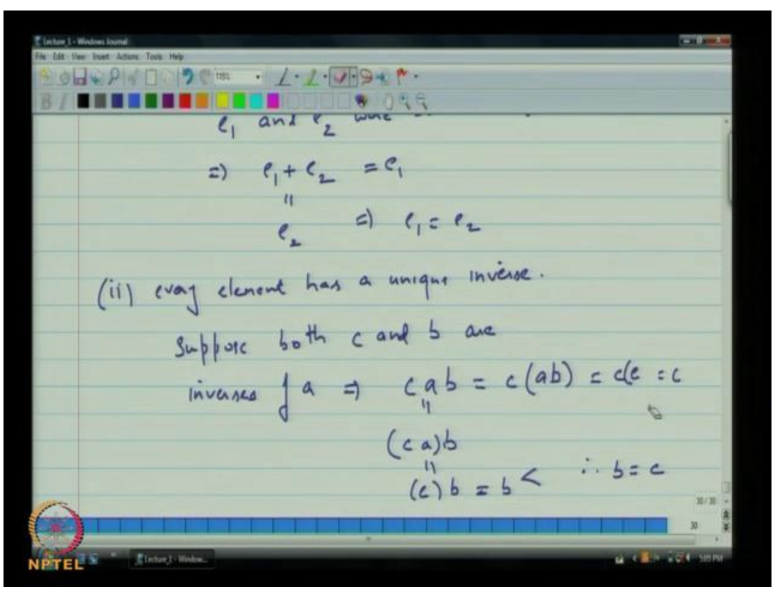

And, once again you can check that the axioms are satisfied. For example, the identity element the identity is zero, if you take two inverse for example, then or more generally, if you take n inverse; if you look at and ask the question, what is a inverse? a inverse is nothing but n minus a, and it is an Abelian group. If you are uncomfortable, with these sets and operations; you might want to go through the axioms and see for yourself, and convince yourself that the axioms actually hold for this particular case. Now, there are some consequences of the axioms that is groups have certain property that follow from your axioms and therefore, which do not need to be stated separately along with the axioms. We will call these derived properties.

(Refer Slide Time: 47:47)



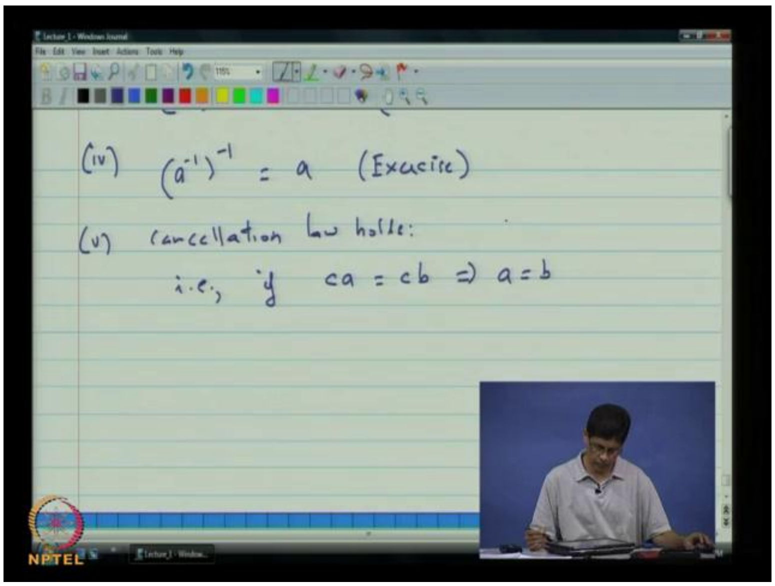

The derived properties are these first of all, the identity element is unique, because suppose not and suppose that e 1 and e 2 were both identity elements. This would imply that e 1 plus e 2 is equal to e 1, because after all e 2 is the identity, but on the other hand since, e 1 is the identity that is also equal to e 2. So, this implies that e 1 equals e 2 and therefore, that shows that the identity element is unique.

(Refer Slide Time: 49:05)



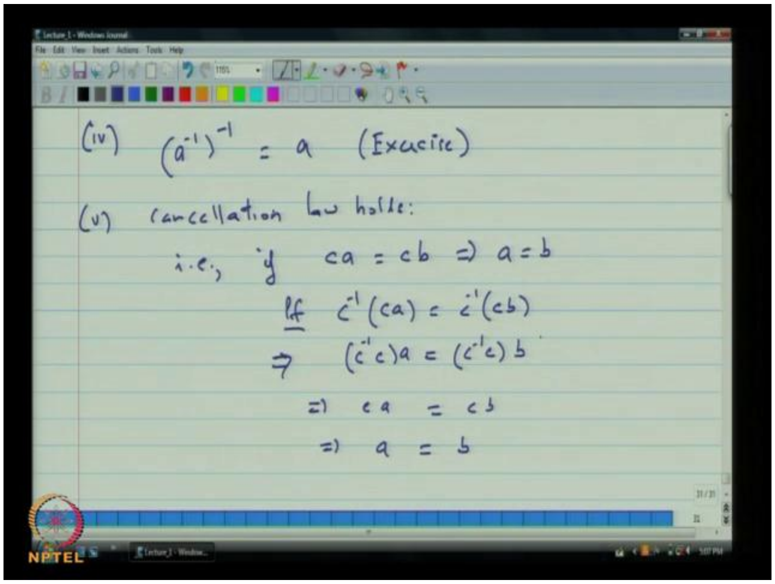
The second property shows that, even the inverse is a unique. Every element has a unique inverse, how do we see that? So let say, again suppose not. Suppose, both c and b inverses of a, but this implies if you consider c times a times b. On the one hand, you can say that this is c times a b that because b is the inverse of a that c times the identity, which is c. On the other hand, we can wrote this and say that this is c times a times b, and because our c is the identity this is a times b ah sorry this is e times b this is e times b, which is b. So, that proves that therefore, by comparing n results. We see that, b is actually equal to c.

(Refer Slide Time: 51:26)



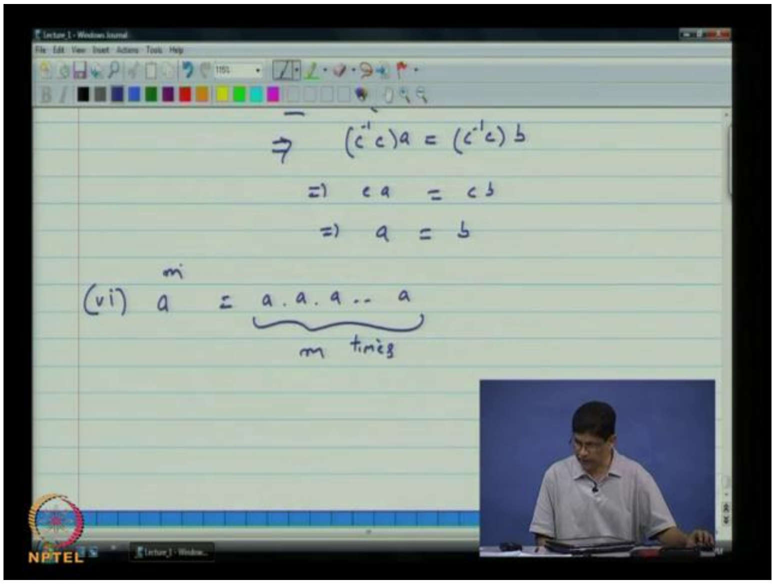

Then another property, which you can I leave it to you as an exercise is 3 a b inverse is equal to b inverse a inverse so, very quick exercise. This is like matrices; it is like this is also case of matrices, but also hole here. Then four, if you take the inverse of the inverse, then you will actually get the element back also an exercise.

(Refer Slide Time: 53:20)



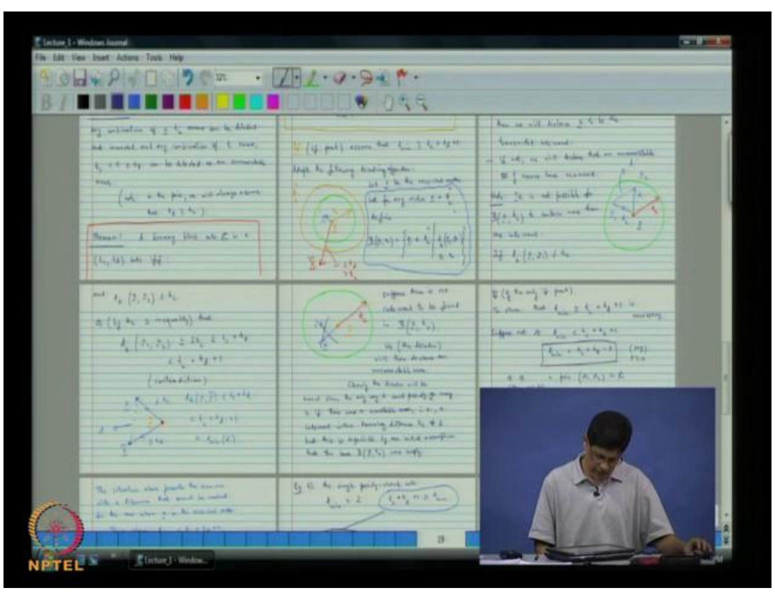

Then, the cancellation law holds what is that mean? This means, that is, if c times a is equal to c times b that implies a is equal to b. You can sort of cancel c and both sides and the proof is because c inverse times c a these two elements are equal of course, they will remain equal even if you multiply on the lefts by c inverse, but this implies that c inverse c times a, implies, is equal to c inverse c inverse c times b implies that e times a is equal to e times b implies that a is equal to b. So, the cancellation law also holds.

(Refer Slide Time: 53:36)



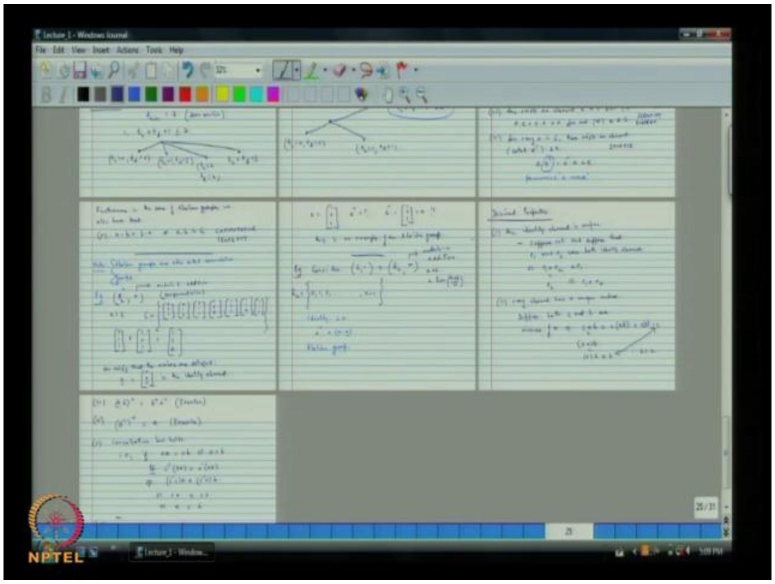

If you take and we are going to define, when we write a to the m this you mean a times a times a times a m times. Thus one more property in relation to this, but I see that we are running out of time. What I will do is rather than go through that, I continue that next time, I will just summarize what we have actually covered in this lecture. Let me see, if I can zoom out so, that we can see more of the lecture on the slide.

(Refer Slide Time: 53:23)



We first completed the proof that a code is a t sub c t sub d code if and only if, the inequality t sub c plus t sub d plus 1 is less than or equal to the minimum distance of the code we completed that proof. Then, we went on to talking about mathematical preliminaries to define the axioms that going to making up a group.

(Refer Slide Time: 54:54)



Some examples and then we stated some derived properties. With that, I would like to close and will see in the next time whereas, I mention before for the next two classes will be working on mathematical preliminaries.