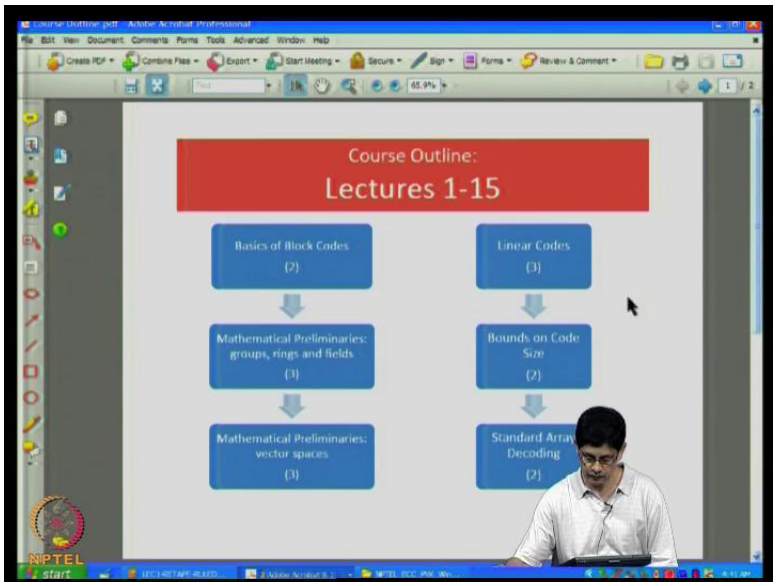
Error Correcting Codes
Prof. Dr. P Vijay Kumar
Department of Electrical Communication Engineering
Indian Institute of Science, Bangalore

Lecture No. # 01
Course Overview and Basics

Good morning or afternoon depending upon when you are listening to the set of lectures. My name is Vijaykumar; I am here at electrical communication engineering department of IIC, Bangalore, and I will be leading you through a course on error correcting codes. So, this is a course; that is a first introductory graduate level course.

That is the title of the course, and this is our first lecture. So, I am going to call this first lecture course overview and basics. So, I am taping this first lecture having taped remaining lectures in the course, so that gives me opportunity to tell you about the course with better knowledge of how thinks unfolds here onwards. So as it turns out, what we done is, we gone head and taped about 42 lectures.

(Refer Slide Time: 02:08)



And let me just quickly take you to through the course as it going to evolve. So, here this course outline and what I have done here is that… So, these are the first 15 out of the 42 lectures, and you see the various title. So, the way this course actually proceeds is vertically

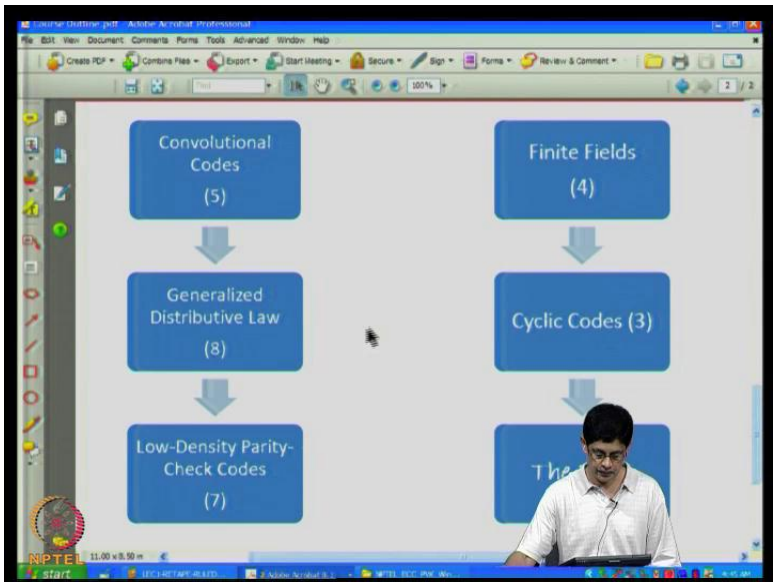
down, and then over here. So, we start with basics of block codes. So, just by way of clarification, when we talk about codes here, we are going to mean codes that I use for error correction. So, this is distinct from say source coding or cryptographic coding. So, we are interested in codes primarily for the purposes of error correction, and so some of these titles may not meant too much to you. We will first talk about at this moment, but you can revisit this later. So, will go through the basics of block codes starting from today after that so this is I think one of the I feel it is one of the good think about the course that actually take you through the mathematical preliminaries; that is I because I do not really assume too much by way of background, but you will need background as you progress you course.

So, we cover groups; will cover rings; will cover fields; will also talk about vector spaces so that is into linear algebra; will talk about vector spaces; will talk about notions of linear independent, basis and dimension, because these are concept that will drop on throughout the course, and I did want to make course self-contained. So as you can see here, we spend about three lectures talking about groups, rings and fields. So, maybe I can enlarge it little bit better. So, we do spend about three lectures talking about groups, rings and fields that you can see, this is the block here. And then after that, we gone to talking about vector spaces, so that is the linear algebra, and the number at the bottom in bracket represents the number of lectures that we will actually end up spending on each of its topics. Then after that, we will talk about sub class of course, and as in any other subject, the linear case is the one that more extractable and as it turns out it also requires useful. So, we will actually spend fair amount of time and linear course.

Then, will talk about bounds on code size. It turns out that you can rank codes according to certain performance criterion and one of them is a code size, and there is another parameter that will come up later. So, this is code size in terms of other parameter. How best; how good can your codes get? So, we will spend two lectures on that. Then after that, will turn to the topic of decoding, and this is a way to actually optimally decode linear code, it is called standard array decoding. Now, throughout this part of the course we will actually be working with binary codes; that is will be working with codes simple alphabet is be the 0 or 1. So I think, in other feature that I am going that these lectures keep in mind is, we introduce the math only when you needed. So, finite fields are used to discuss block codes,

but what I am going to do is I am going to introduce finite fields towards the tail of the class rather than now, because these concepts can be passed on without need for finite fields.

(Refer Slide Time: 05:45)



Then after that, now moving to the lot of course, lectures 16 to 42, what will do is, we will talk about first talk about convolutional codes. So, these are distinct from block codes turned out to be very interesting useful in practice for example, these codes used in these space machine also, so what block codes, but convolutional codes had some aspects in which take perform better than block codes not at all. Then after that so we send 5 lectures on convolutional codes, then we spend time on something which I have given in analysis distributive law the reason if called it that because taken a title from certain general article.

Now this is I think perhaps unique feature of this course here what... So, how did this come about? This came about because coding theory actually transition from time period in which the techniques that will employed to build codes were largely algebraic in nature with perhaps this expression of convolutional codes. But then sometime in the 90s around the early 90s, there was a new class of codes that came into the picture, which tend into excellent performance. But whose description was in a sense more probabilistic and using probabilistic techniques as a post algebraic, so it is not to say the device no algebra erase algebra, but it was more probabilistic nature in comparison with earlier codes. So, what I am

speaking of is difference between turbo codes which are raise in dimension compare to let say b c h or read Solomon code which were the traditional class of codes.

Now these codes the strength of those codes was not due a better codes in terms of there in inherence structure, there was better codes in the sense that they could be decoded more easily. So, in practice this in great deal of emphasis on is of decoding, and this is where turbo codes and there success as known as ldpc codes, ldpc standing for low density parity check codes score heavily. Now at least now this is the personal decision, I found that theoretical underpinning's our became a little bit clearer, when you adopt the view point that is taken in a very nice article in title is the generalized distributive law, which basically tells you that there is a method of decoding, which follows a I mean, there are very many different names that are associated with this style of decoding, someone might say this is an iterative decoding, other might say this is message passing decoding; other might say this is belief propagation, but whatever that phenomenon is a method of decoding is.

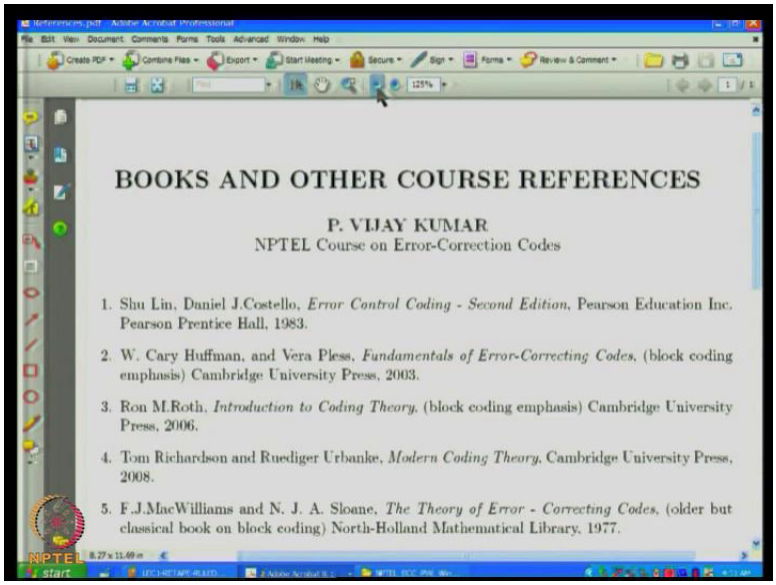
This title which is somewhat mathematical and in nature, and might scare you actually gives you very nice introduction, and it is what I provides what I think of theoretical underpinning for methods of decoding that I used today. Then after that we talk about the class of codes that use this particular distributive law, there are two classes primarily although there are other turbo codes and ldpc codes. Here we will cover ldpc codes, they seen to cover be fast becoming the ubiquitous code of the present day, there is there anyone, who needs the codes for any application will often the first code that they will think of our of ldpc codes are some variation of them. Now so in other words this class of codes really represents the modern class of codes. In fact, there is book on coding theory which is devoted the theory of to the theory of codes such as this.

Then after that, so will spend 7 lectures on that then we go back to the algebraic point, and we now talk about codes whose algebraic structure draws upon both theory then will covered to this point, so we actually telling the theory of finite fields, and finite fields are good for constructing very many class of codes, what we will covered and this class our class of codes known as cyclic codes, and although the number 3 might looks small here actually this lecture are somewhat lengthy. So as number count is probably stands for more

than 3 lectures, because I actually use written material and I go through it a little faster than in the earlier lectures. So this fair amount of material in this block, so with that we actually end the codes.

So, again so we start with basics, which involves block codes, then in the middle will go of these codes having the probabilistic description, and then come back again to block codes, and discuss the one's that involves more algebra; so that is the quick outline. Now before forget this code thus have a list of homework problems, there are not really tiled in to the lectures as in this lecture requires you to solve this problem, so it is just to list of problem. And as you roll through the course you can look at problem, and see if you can try them, so that might give you that might help you in getting a feel for how value understood course.

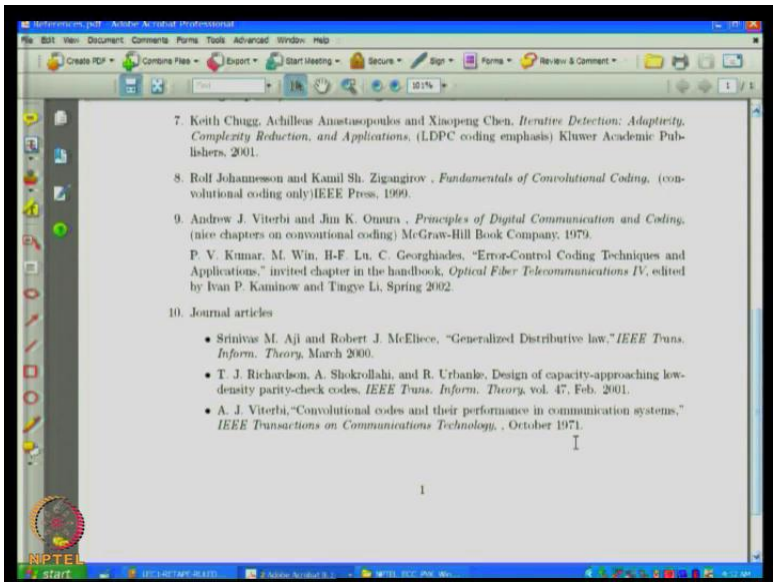
(Refer Slide Time: 11:33)



Now in other thing which you might find useful is here in a file; and I think I will actually come back in a later lecture to show you actual text book in the class. But I thought it my useful show to you actually, look at list of text books. Now there are large number of books on coding theory this is not case some years ago, but in the recent year they have been a large number, and many of them very good, so it is matter of taste, and some emphasize algebraic aspect of the subject, some and a few emphasize the probabilistic aspects. And but I will try to list and this list is by no may exhaustive, there many very many good book that I

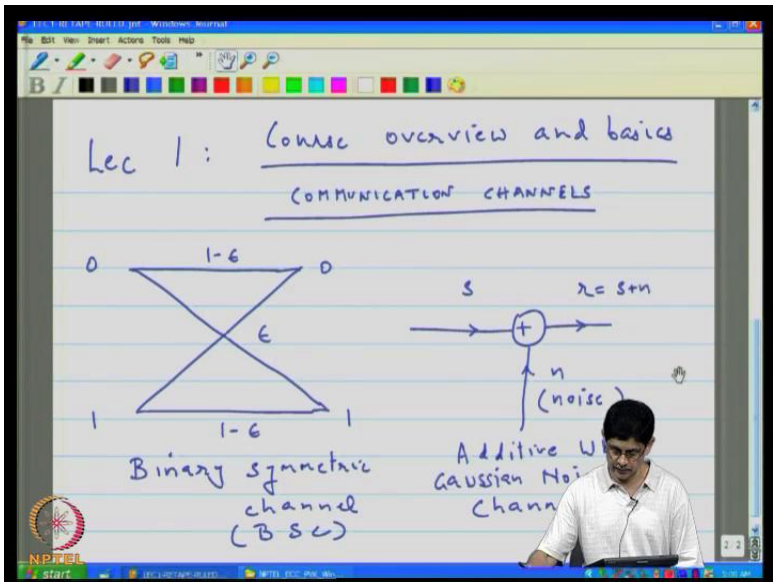
am not listed here, and I have also listed the chapter that we actually had written on error correction, which applies which appears in a hand book. And this course follows somewhat along the lines of that chapter, and here some three of the journal article that I am actually going to draw upon in this class, and including the article on the generalized distributive law.

(Refer Slide Time: 12:35)



So that is here references and also told you about homework. So with that what I am going to do is I am going to start taking you through the basics, preliminaries that allow you to better understand binary codes, but first few words about where this codes there actually going to use, so these codes will be used over communication channels.

(Refer Slide Time: 13:27)



And so let me put that down here, so there are two examples communication channels that I will put down here, so there are two examples communication channels there are presented here. And this here is the binary symmetric channel, so the and this is may be familiar to you as may this, so there are two inputs which you can regard as 0 and 1, and the outputs are also 0 and 1, this epsilon is the cross over probability, there is it is a probability of transmitting a 0 and receiving a 1 or vice versa. Then the second channel is what is called an additive white Gaussian noise channel, here the input of the channel is what is transmitted by the transmitter which is s. And then noise is add it to it which is n and what you actually receiver r equal to s plus n, this is called an additive white Gaussian noise channel, the additive is clear from the picture.

Gaussian because the noise is assumed to be have Gaussian statistics, in the deep space channel for example, you might and count a channels like this. And although channel may not exactly look like this, this is channel for which codes are most often designed. Now you might ask where would this channel come in, you could argue that for example, in magnetic recording channels the channel might be better model has being this type of a channel is suppose to this, because you might be recording 0s or 1s. But there is an another connection that is that if you quantize the output to two levels, then what you received could be

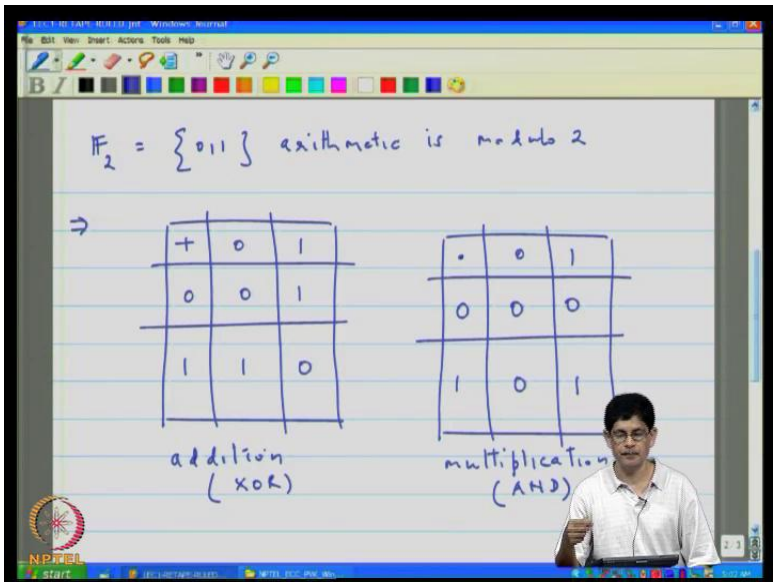
construed as being 0 and 1. Now as for the input, well when you send the message typically going to use some kind of modification.

So, even though this channel can accept any real input it is customary from the point of view of simplifying hardware to use a modulation with finite small alphabet, with some example, popular alphabets being binary phase shift keying bpsk, or quantitative phase shift keying qpsk, or you mightily use 16 quantitative for that matter. So, these are example modulation there may be some slight differences because I shown real channel here, whereas typically when you talk about quam cancellation you would use a complex version of this, but that is more a matter of technical detail. But consider a case when the input is bpsk when you... Now this channel model abstract some details that you would see for example, explain in a course on digital communication, so this is the base band equivalent baseband model.

So, in the equivalent baseband you have transmitted signal would be either plus or minus 1, so that is why you are binary input would come from, and if you quantize the output it get binary output, and if you reliable binary input is 0 and 1 and reliable binary output of the quantization is 0and 1 again, then what you have is the binary symmetric channel. And in the early days coding theory focussed on this binary symmetric channel. And you do lose something in this because so the input is I am showing is not too bad, because you might actually be transmitting binary input, but the output if when you quantize the received signal to extract the binary output, you lose some information in the process and that consumes something.

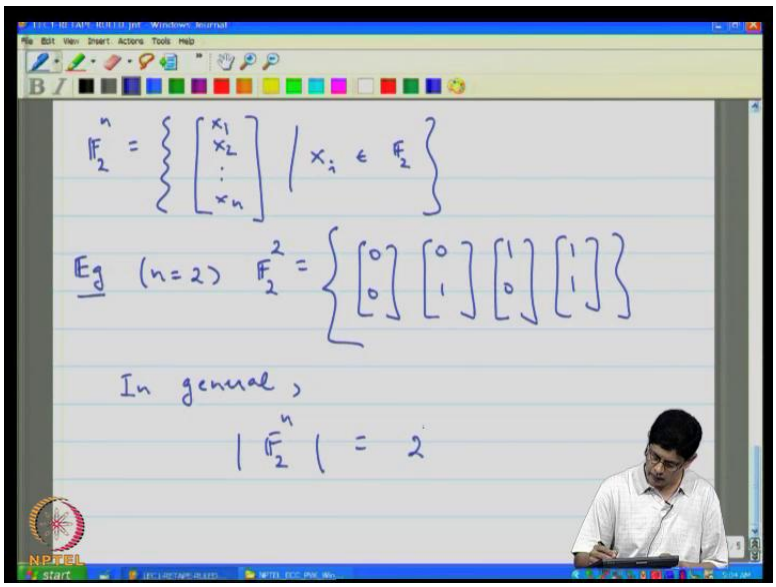
And so in the early days we made the quantization assumption just to make the problem on attractable, but they did pay that penalty, and one of the benefits of this consequent probabilistic approach is that big that approach enabled them to work with codes that could work of the real output of a channel is to the quantize. So, the algebra was well tuned this channel whereas the probabilistic methods can also work with this channel that give them an important advantage in practice. Now, what I would like to do is just go through some basic math that you would need to work with bandwidth signals, so I will introduce first some notation.

(Refer Slide Time: 20:30)



By the way in other point I want to in mention is that so I am this lectures are going to be return out on windows using windows journal on that tablet pc.

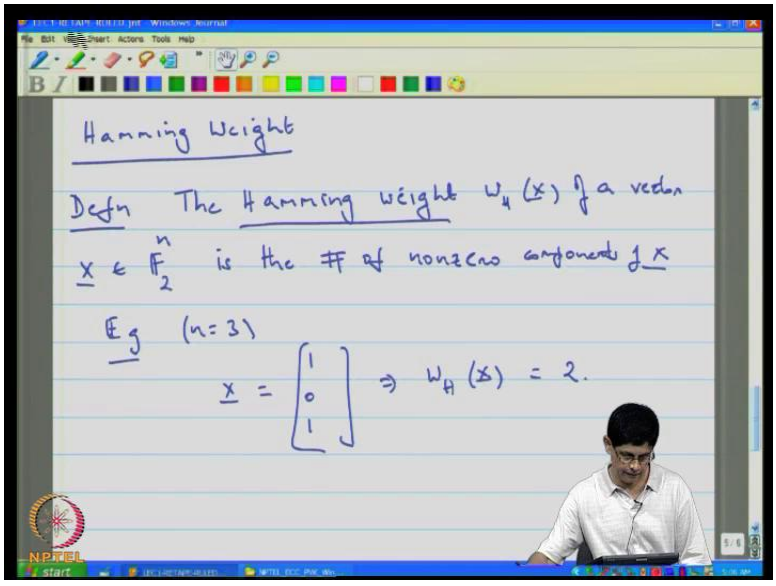
(Refer Slide Time: 23:24)



And so the plan is to make available on the website for you find the course video pdf file, which has all the lectures in pdf form, so that I think could be useful to mind please keep their mind look for it on the website they should be a file which has all the lectures

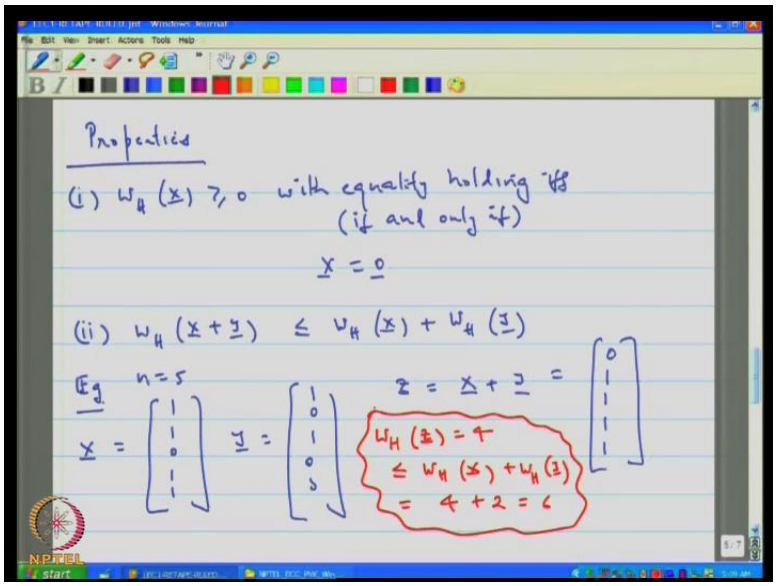
appearing in the single pdf file. Then I will try to append to it, the course outline, the list of references and homework is well. So, saying back through that math here, so here what will see are rules that we will use to add and multiply, when we alphabet just consist of 0 and 1, so this is addition which correspond to exclusive or addition modulo 2, and this is multiplication which corresponds to the and operation. We will write f 2 for the alphabet 0 1. And now I want to introduce in other notation f 2 to the n, which is the collection of n tuples that is vector with n components. And so we use the super step here to denote that for example, when n is 2, then f 2 to the 2 consists of four vectors, so in general from this it is clear, the size of f 2 to the n is 2 to the n.

(Refer Slide Time: 24:33)



So, we will need the simple notion of hamming weight. So, supposing you are given an n tuple that is the vector with n components x then the hamming weight of that particular is simply the number of non 0 components. Now remember that the components of x either 1 or 1, so for example, if x is 1; 0 1 then hamming weight is 2.

(Refer Slide Time: 26:20)



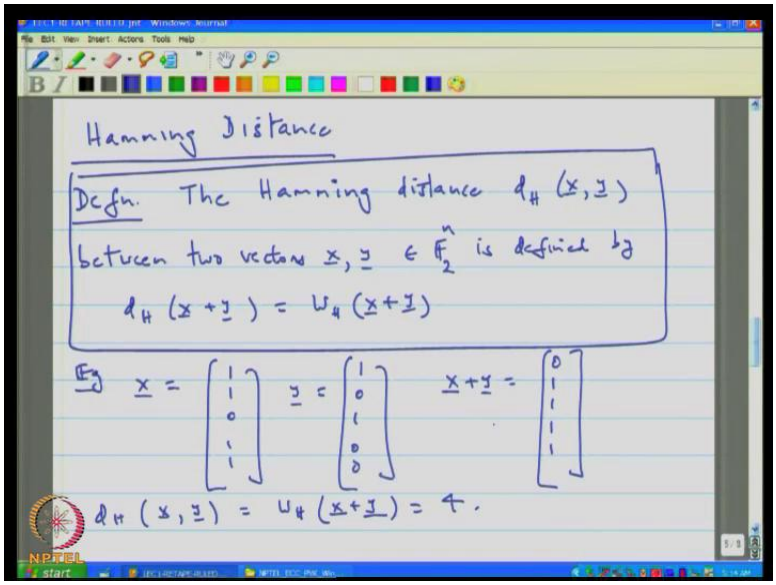
So, here where looking at properties of the hamming weight function, so the first property is that hamming weight, since it simply counts the number of non zero symbols, it is always going to be greater than or equal to 0, and only way it can equal 0 is if the vector is the identically zero vector, that is the vector where n tuple with all the components are 0.

The second property is that if you add, if you take the sum of x plus y, then the hamming weight of that sum is less than or equal to the sum of the hamming weight of x plus the hamming weight of y. And that is actually not hard to see why that is true first let us take look at an examples, so in this example x is a vector 1 1 0 1 1, y is 1 0 1 0 0, and if you add x and y, you get z which is 0 1, see 1 plus 1 is 0, 1 plus 0 is 1, 0 plus 1 is 1, 1 plus 0 is 1, and 1 plus 0 is 1. So here the hamming weight is 4; this is hamming weight 4; this is hamming weight 2, but z has hamming weight 4. And the reason for that is clear, because whenever you have two 1s phase in each other in the same position, and you carry out addition then their sum is 0.

So, in some sense you lose two 1s, so the hamming weight of this sum can never exceed the sum of the hamming weights, but typically it will be less by a factor or term, which is equal to twice the number of 1s, they have in common for example, the projection in which these two vectors have 1s in common is just the first position. So, the weight is 4 plus 2 minus two

times 1, which is 6 minus 2 which is 4. But anyway in think that you way often keep in mind is just that the hamming weight of the sum of x plus y is less than the sum of hamming weight of x and hamming weight of y.

(Refer Slide Time: 31:05)

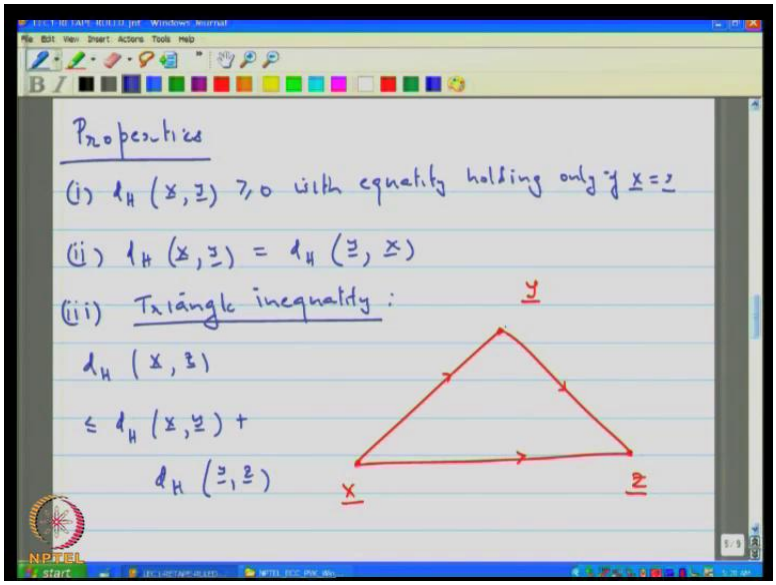


Next we will introduce related concept that of hamming distance. So, next concept is that of hamming distance, and the hamming distance of two vectors x and y and both let say are binary vectors having the same number of components, then the hamming distance between them is formally speaking it is the hamming weight of their sum. But most people think of it as simply saying hamming distance between x and y is simply the number of components in which there actually differ, because that will end up being the hamming weight of x plus y, because hamming weight of x plus y counts the number of non zero symbols in x plus y, and that precisely the symbols corresponding to location where x and y disagree.

So, in this example, so we are back to the same example x and y same as before, and here we saw that x plus y was 0 1 1 1 1, so the hamming weight of this is 4. Now the hamming distance is between them according to this definition is the hamming weight of this, but you can also just see from inspection that if you look across these two vectors then the basically agree in just the first position everywhere else they differ, so the hamming distance is 4 and that is typically the way people think about it. So, we introduce the motion of hamming

weight, and we look at properties of that hamming weight function, so now we will introduce hamming distance, so we will again investigate properties of the hamming distance function.

(Refer Slide Time: 34:55)

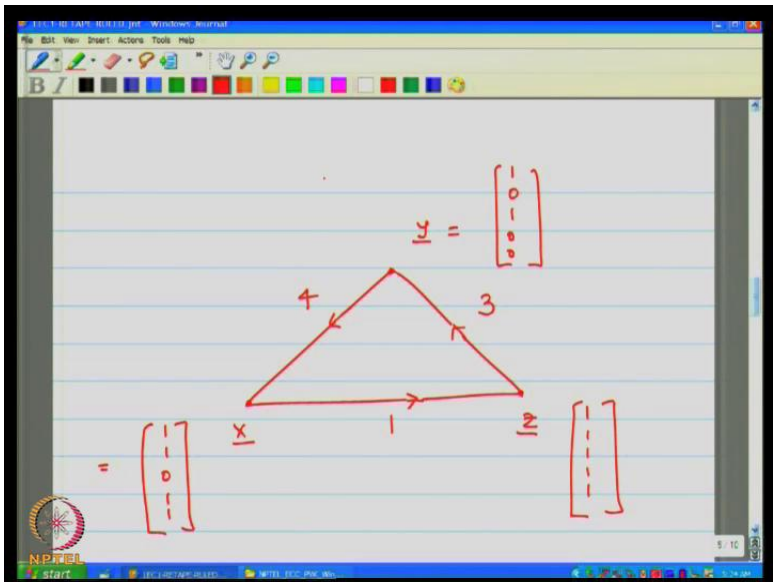


So, the properties that the hamming distance function actually satisfies, or that the hamming distance between any two vectors is strictly greater than or equal to 0 with equality holding only if In fact if and only if x equal to y, because the only way that x and y... Well the distance greater than being greater than or equal to 0 it is obvious, because we are counting the number of location in which two vectors differ, so that is going to be a number its greater than 0, and if it is 0 that means, there are no projection in which are differ, which means that agree everywhere so vectors have to be the same, so the first property is clear. The second property says that whether you look at the hamming distance between x and y or y and x makes no difference, and that again is obvious, because you just looking at the pair and saying in every location is pair the same or different, so the order is relevant.

The third one more interesting is called the triangle inequality, and triangle inequality states well from a mathematical point of view just says that the hamming distance between x and z is less than or equal to hamming distance between x and y plus the hamming distance between y and z. But pictorial you could actually look at like this that is that just like the

distance in the real world in everyday world from x to z is going to be no more than the distance x to y and y to z the same is true of the hamming distance function. So the hamming distance agrees with r equilibrium motion of distance, and it turns out that this is useful, because it allows you to carryover your intuition from everyday life into the hamming world at least to some extension, and the reason for this actually very clear, but I am going to illustrate this on the next page with an example.

(Refer Slide Time: 38:57)



So, let us take our vectors x and y to be as before, but now in this example there is third vector let us say that is z. And let us look at hamming distances between three pairs, so the distance between x and y counts the number of symbols in which they differ, so they agree on the first; they disagree in 2, 3, 4 and 5, so from that, we say that the hamming distance here is equal to 4. Now between y and z, this has two 1s this as all 1s, so it is clear that the distance is 3, the distance between x and z on the other hand is just 1, because the only differ under this.

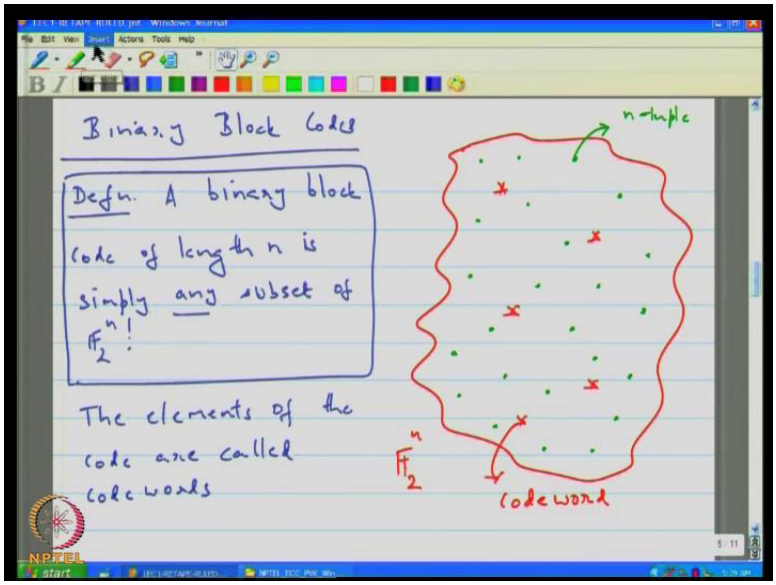
So, now the here and just that the distance between x and z is less than the sum of the distance from x to y and y to z, so that since like a grows over estimate in this case. But sometimes you could have equality, and reason for this not hot to see, and reason why this distance between less, because you could go from x to z by first flipping some of the

symbols in x which will take it y for example, you can flip then last 4 symbols and get to y. And then you can flip the 2nd the 4^{th} and the 5th symbols to get to z, but some symbols you could flip twice in fact, 3 of them you flip write this for no reason, and for this reason that in some sense that wasted effect, you could have just flip one go to from x to z, so from this it is clear to the hamming distance between x and z is going to be the less than or equal to some of the distance x to y and y to z.

Now I have shown the arrows this way, but you could argue in just about any regardless of the orientation of the arrows. So for example, if you look at this way, then this would tell you hamming distance between y and x is actually equal to the sum of the distance from x to z and z to y, so here some instance which you actually have equality. So, that ends our mathematical preliminaries in terms of hamming weight and hamming distance.

Now we want to talk about block codes; that is we want to talk about objects which can actually use be used for the purposes of error correction, we will begin with a very surprisingly simple definition, and then I explain why it is that works; we may not and then will see examples in the next lecture, which will make that concept clearer.

(Refer Slide Time: 42:38)

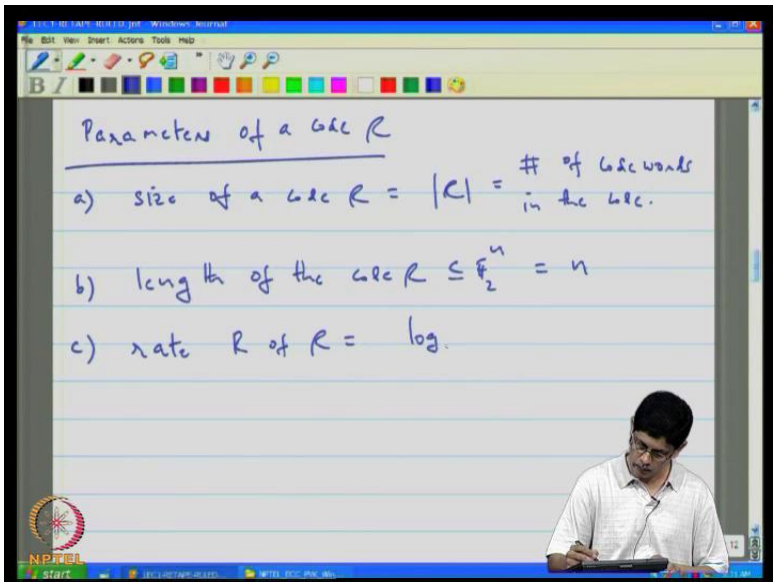


So, now we have as a said a simple definition of a binary block codes, but first when we talk about block code there is motion of length that is the number of components, this block code is going to be composed of vectors the number of components in which vector the parameter which is called a length. So, binary block code of length n is simply any subset of f 2 to the n for example, if n is 5, then there are 2 to the 5 32 n tuples; 5 tuples, if you take any subset of them, then you are justify calling that to be a code.

Now you might somewhat surprised with this definition, but just keep in mind, there are did not say, I did not tell you little be a good code. So for example, and from just pictorial, abstract pictorial depiction, so let us say that this represents set of all n tuples, so this entire picture here represents the collection of binary n tuples. So the red vectors here are the code words, and the green dots are the other n tuples, so these are the subset that forms the code and so the elements of the code are called code word. So, you have in this particular case, 5 code words.

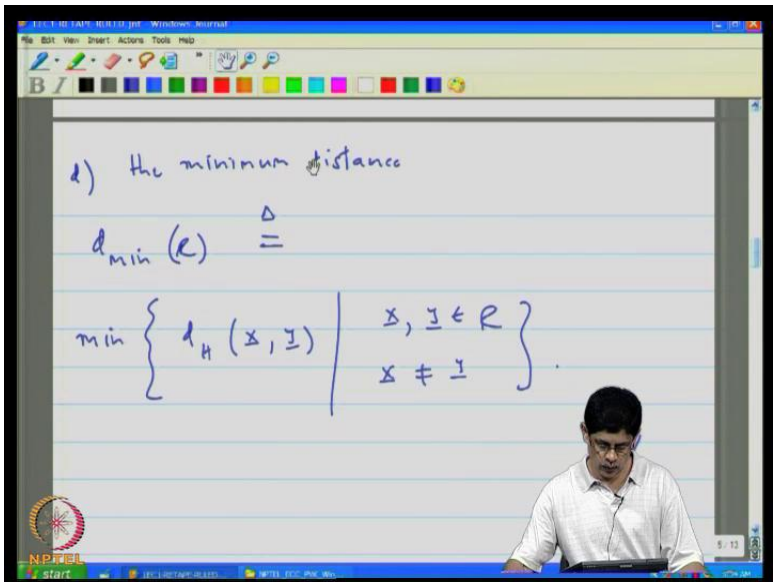
Now just a quick perhaps the results I was going to talk about error correction, but we can see that if you transmit code word, and during the transmission it is corrupted, and what you receive is a neighbouring n tuple. Then by going back to the nearest code word you can correct errors, so that very quickly is how an error correction code words at least in an abstraction. But I want to end this picture talking about in parameters of a code that we will need in the next lecture.

(Refer Slide Time: 47:20)



So, there are 4 parameters of a code c, which I used to actually judge both to identify as well as judge the of the code, 1 is the size of the code that simply is the number of code words in the code, the other is the length but I have already told you about what it means talk about length; that means that in a code all the vectors have same number of components, and that component is a number of length. The rate of code now that is defined as a log to the base 2 of the size of code divided by n, so that turns out to be major of how much information is transmitted or how much uncertainty is dispelled when actually receive the code over the channel and decoded correctly. So that is the definition that actually comes from information theory, and I think you will get a better feel for it once you look at examples, so I going to spend time on that for now, and we will close with just the 4th and last parameter.

(Refer Slide Time: 49:52)



So, the last parameter of the code is the hamming the minimum distance, and the minimum distance the notation for that is d min; d min of c this is script c standing for a code, d min of the code is the minimum distance between a pair of distinct code words, that is you list of the code words all of them r n tuples, binary n tuples. And you look at the once there are closest together n terms hamming distance, and that minimum hamming distance is the minimum distance of the code.

Now it start hard to see why that will be important an error correction, because so for example, if you look here, here are the code words, and you can see that if noise perturbs the code word, and then noise if noise raising from this channel, then you will start with the code word but you will end up with n tuple. And if the code words are too closed to each other, then it will tell which code word it came from, in fact it may be possible that noise would draw you one code word to the next of that to close. So you would like to keep them separated, but acting on opposition to this is your desire to send large a amount of information, so that is the rate of the code. So the rate of the code grows up with size of the code, so these two act in opposition at each other, once is there I want to more and more code words into the space in order that I can send more information the other is well wait a minute, I want to correct error and for that I would like to space the code words apart.

So, there is the trade off, so what you would like to do in coding theory is build code words which for a given side size as far apart as possible, or for a given distance between minimum distance between code words have as large a code word as possible. But all of these become clear on the next lecture, when we look at some examples or become clearer. So just to recap what we did was I gave you a course outline, and then I pointed out what some of the resources that are available to accompany this course are, and then will looked at basics, and terms of how to deal with binary vectors, hamming weight, hamming distance, and then we defined in error correcting code, so with that I would like to close, so look out seeing you in our next class. Thank you.