Principles of Digital Communications
Prof. Shabbir N. Merchant
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Lecture - 63
Channel Coding : Hamming Codes

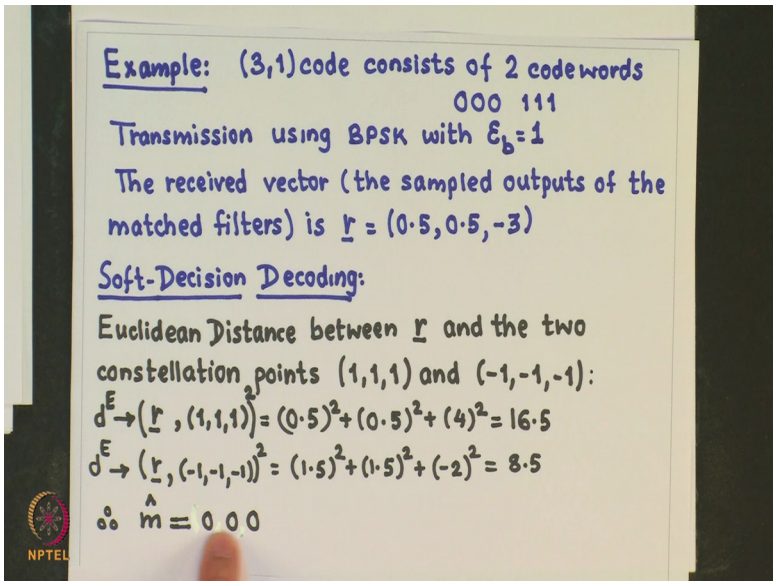
For any n comma k linear code, codeword is a sequence of symbol 0 and 1 of length n. Now for transmission of the codeword onto a physical channel, we need to convert this codeword vector into physical signals or waveforms. So, let us assume that we are using a radio channel and deploy BPSK modulation for the transmission of the symbols of this codeword. So, each symbol would be transmitted using a sinusoidal pulse of a particular duration, which say positive polarity for symbol 1 and negative polarity for symbol 0.

Now, so, this message codeword is now converted to a sequence of sinusoidal pulse waveforms or sinusoidal pulse signals. On the channel this will get corrupted by additive white Gaussian noise as usual. Now at the receiver assuming that all the messages are equiprobable, we will deploy minimum distance receiver. What this means that the output of the channel will pass through the match filters and sample to obtain the received vector. Now we will choose that message signal vector for which this received vector is the closest in the Euclidean distance sense and the decision will be in favor of that message signal or message signal vector this is known as soft decision decoding.

There is something what is known as hard decision decoding. So, it operates as follows. Once we get the received vector which is the sampled version of the outputs of the match filters each component of this received vector is thresholded to converted to binary symbols in our case 0 or 1, because we are restricting our a discussion to binary codes. So, once you threshold each component of this received vector; we generate what we call it as an estimate received codeword. Now this estimate received codeword is compared with the codewords in the code in the hamming distance sense and the to whichever codeword this estimate receive vector is closes we decide in its favor.

So, this can be made more clear with the help of an example. So, let us say we have three comma 1 code, which will consists of 2 codewords 0 0 0 and 11 1 and we transmitted using BPSK with the bit energy equal to 1.

(Refer Slide Time: 04:36)

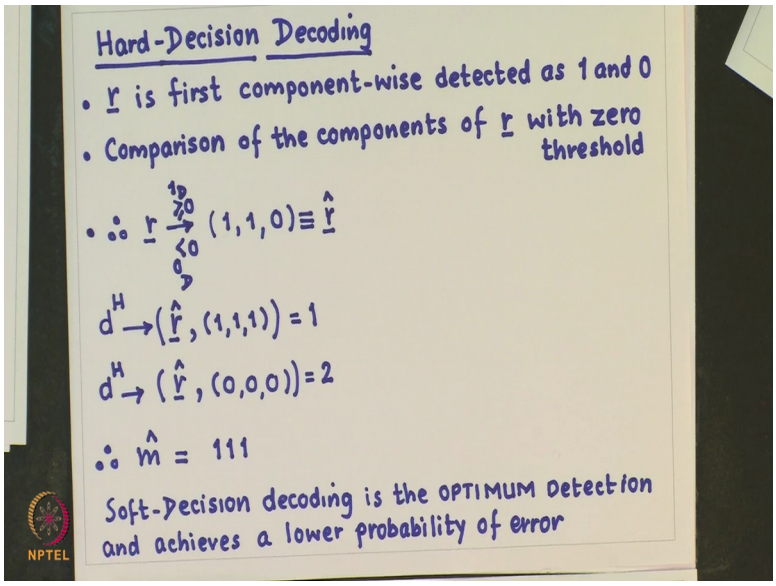


So, the received vector which is the sampled outputs of the match filters is let us assume to be as given here; these are the three components of the received vector. So, let us see what would be the decision if we use what is known as soft decision decoding.

So, in this decoding we evaluate the Euclidean distance between this received vector and the 2 message vectors or 2 signal vectors or 2 constellation points which are 1 1 1 and minus 1 minus 1 minus 1. This may be for 0 0 0 and this is for 1 1 1, these are the signals which we have transmitted. So, if we compute the Euclidean distance. So, we have the dc vector r and compute the Euclidean distance with respect to 1 1 1 this is the message vector or signal vector we take the square of that we get this value based on the given information.

And similarly we compute the Euclidean distance between the received vector and the other message vector or the signal vector which is minus 1 minus 1 minus 1 and the square of that distance turns out to be this value. So, now, since this is a minimum Euclidean distance we decide in the favor of this message vector or signal vector or indirectly we decide in the favor of the codeword 0 0 0. Now let us deploy hard decision decoding.

(Refer Slide Time: 07:09)



So, in the hard decision decoding once we obtain the vector r, which is the sampled outputs of the match filter, this vector is first component wise detected as symbol 1 or 0 by thresholding it to 0 threshold correct assuming that the messages are equiprobable.

So, if we do that this value 0.5; 0.5 minus 3 would be converted this would be converted to 1 1 and 0. So, whenever component is greater than or equal to 0, we decide in the favor of 1 and when it is less than 0, we decide in the favor of 0. So, this is what it shows correct. I get this is my estimate of the received codeword and now we compute the hamming distance between this received codeword and the message codeword 1 1 1 and we get to be equal to 1 because it differs only in one place.

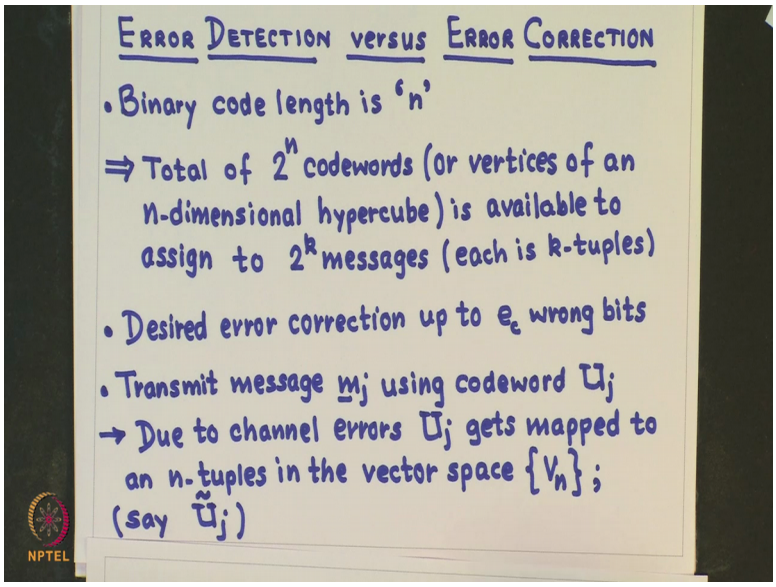
And similarly we compute the hamming distance between the risk estimate received codeword and the codeword in the code which is 0 0 0 and we find that it differs in 2 positions. So, the hamming distance is 2. So, in this case we decide that the message transmitted was 1 1 1 ok. So, we see that in this case the decision is different than what we got earlier. But important to note that soft decision decoding is the optimum detection and it achieves a lower probability of error.

Now, we can show that in practice the difference between the soft decision decoding and hard decision decoding may not be very significant, and it is easy to implement the hard decision decoding. So, in our discussions we will assume that we are using hard decision

decoding. Now let us try to understand the error detection and error correction properties of a particular n comma k linear code.

Now, if we have n comma k linear code, what it implies that the codeword is binary codeword and it has a length of n. Now this implies that there are totally 2 raised to n codewords or vertices of an n dimensional hypercube, which is available to assign to 2 raised to k messages each of this 2 raised to k messages is k tuples.

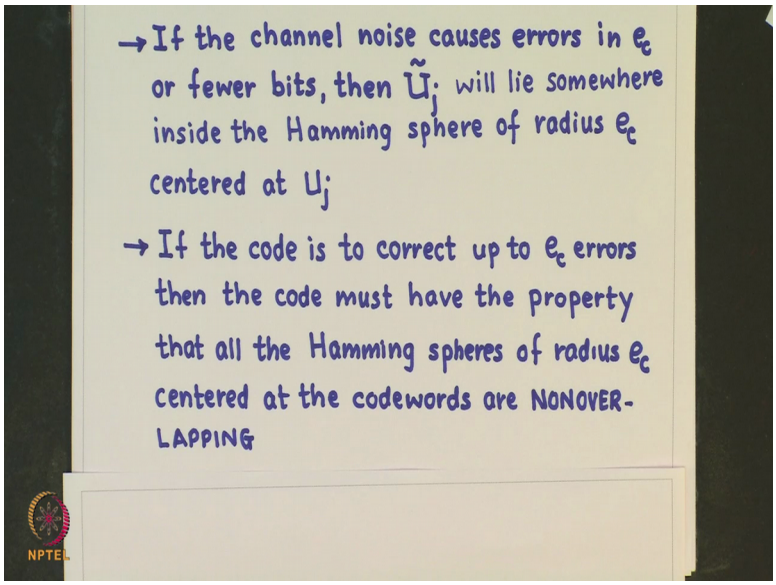
(Refer Slide Time: 10:48)



So, now let us assume that it is desired that this code is able to correct up to ec wrong bits. So, what this implies is as follows; that if I transmit message say nj using a codeword U j, then due to channel errors U j gets mapped to another n-tuple let us call it say u tilt j in the vector space of n-tuples.

So, if the channel noise causes error in ec of fewer bits, then your received u tilt j n-tuple will lie somewhere inside the hamming sphere of radius ec, which is centered at the codeword U j.

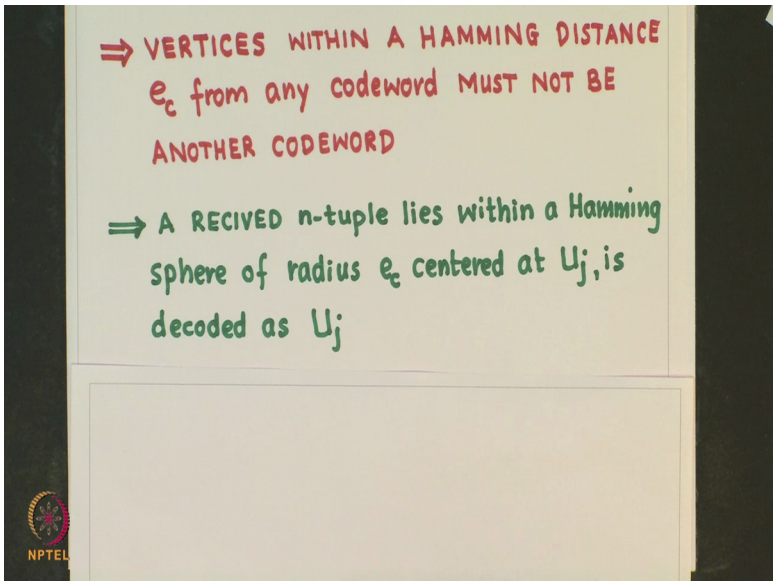
(Refer Slide Time: 11:59)



Please remember that this hamming sphere is not in the geometrical sense hyper sphere because your hamming distance is also not a geometric distance. So, now, if the code is to correct up to ec errors, then the code must have the property that all the hamming spheres of radius ec centered at the codewords are non overlapping correct.

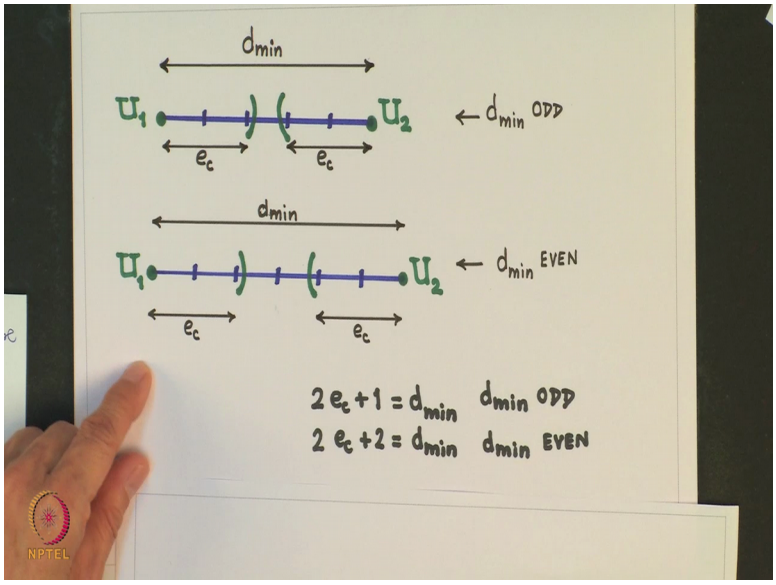
So, what this means that vertices within a hamming distance ec from any codeword must not be another codeword correct.

(Refer Slide Time: 13:00)



So, this implies that a received u tilde j which is n tuples, if it lies with a hamming sphere of radius ec centered at U j that is a codeword, then it is decoded as the codeword uj. So, this will be very clear in the figure given here.

(Refer Slide Time: 13:32)

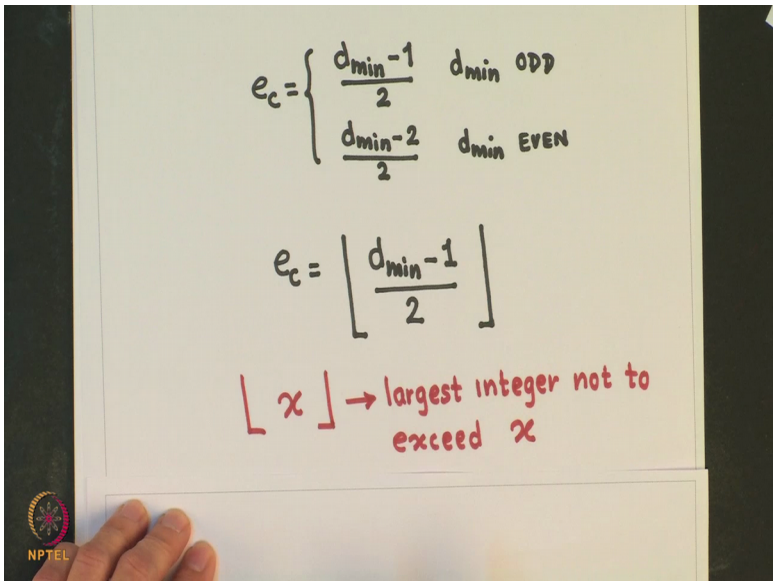


So, for explanation I have taken 2 codewords U 1 and U 2, which have the minimum distance in this code. So, there are 2 cases have been considered where d minimum is odd. So, you can check 1, 2, 3, 4, 5. So, d minimum here is 5 whereas here d minimum is 6.

So, d minimum is even; so as shown here each codeword U 1 and U 2 here. Around this there is a hamming sphere of radius ec this is what is being shown, where ec denotes the number of correctable errors. So, similarly it is shown here. Now as long as these spheres are disjoint the code is capable of correcting ec errors. So, a little thinking shows that the condition for non overlapping spheres for this 2 cases is given here. So, when it is odd this condition should be satisfied 2 ec plus 1 should be equal to d minimum and here 2 ec plus 2 should be equal to d minimum correct because this cannot be included because if it is here, then we do not know whether it is coming from U 2 or it is coming from U 1 there will be ambiguity.

So, using this 2 relationship, we can solve it as follows.

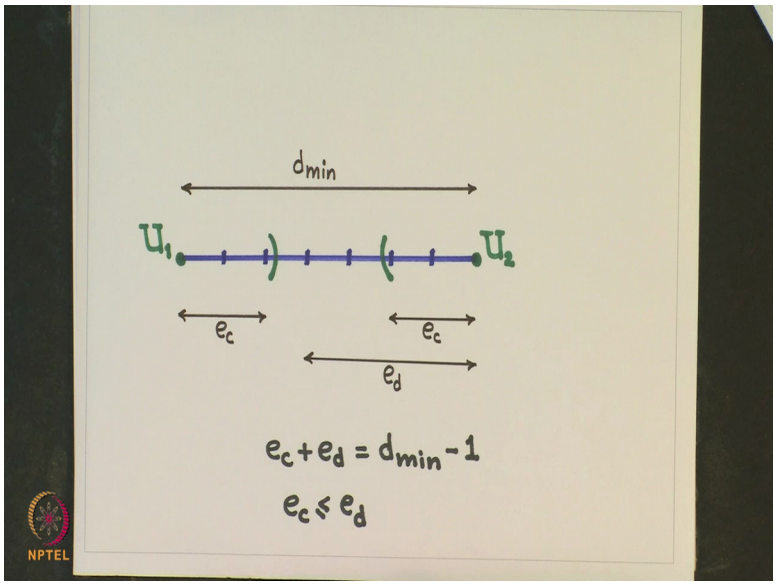
(Refer Slide Time: 15:18)



So, ec becomes equal to what is shown in the right hand side here for d minimum odd and d minimum even. Now from here we can summarize that this is equivalent to ec being the lower flow of d minimum minus 1 by 2. What I mean by this is that lower flow of a quantity x is the largest integer which does not exceed x. So, using this definition you see that, we can find out the error correcting capability of the code if I know the d minimum of the code.

In some cases we are interested in decoding procedures that can detect errors, but not correct the errors. For example, this could happen if I have a communication link from the receiver to the transmitter. So, if the receiver detects an error it communicates via reverse channel to the transmitter then there is an error; so retransmitted. So, in this case I am not interested in correcting the errors, but only detecting the errors. So, in that case we could use this figure to explain the same.

(Refer Slide Time: 16:44)



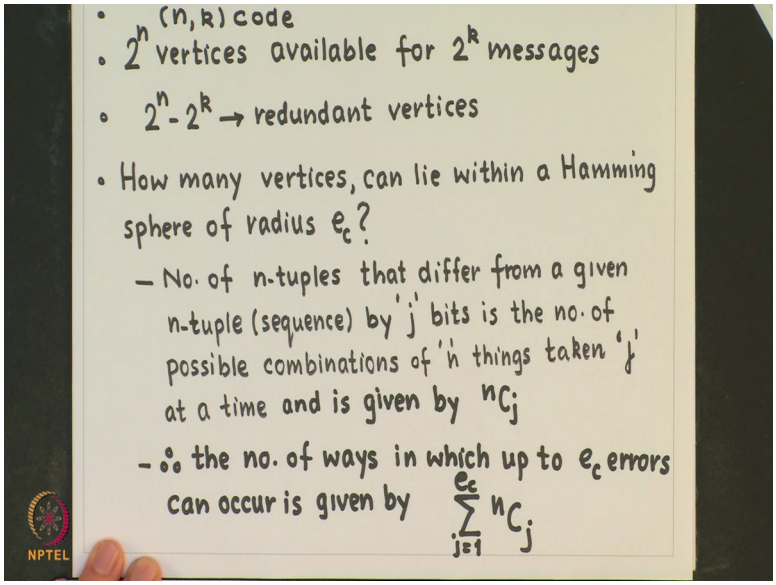
So, now here you look I have again 2 codewords, which are separated by this d minimum without loss of generality we have call this U 1 and U 2 it could be any 2 codewords which has that minimum distance. So, here if we denote the error detection capability of the code by e d then obviously, in the absence of error correction ed is equal to d minimum minus 1; because if d minimum minus 1 that is ed or fewer errors occurred the transmitted codeword will be converted to a non codeword sequence and therefore, an error is detected.

So, in this case we will get our ed to be equal to d minimum minus 1. Now if you are interested in both error correction and error detection as shown in this figure, now here we are interested in error correction up to ec. So, we draw the hamming spheres of radius ec around this 2 codewords as the centers and then we are interested in error detection as ed. So, in this case the error detection can happen up to this point if I assume that I have transmitted U 2 without loss of generality.

So, now from this figure it is very clear that ec plus ed should be equal to d minimum minus 1 with the condition that ec is less than or equal to ed correct. So, we get all the characteristics of the error correction or detection of the code by knowing the d minimum of the code. So, now, let us see whether there is any relationship between the values n, k and the error correcting capability of a code. In the sense we would like to find out that if we are interested in correcting errors up to certain number of bits, then

what kind of relationship should hold between this n k and the error correcting capability and we will try to find the answer to that.

(Refer Slide Time: 19:46)

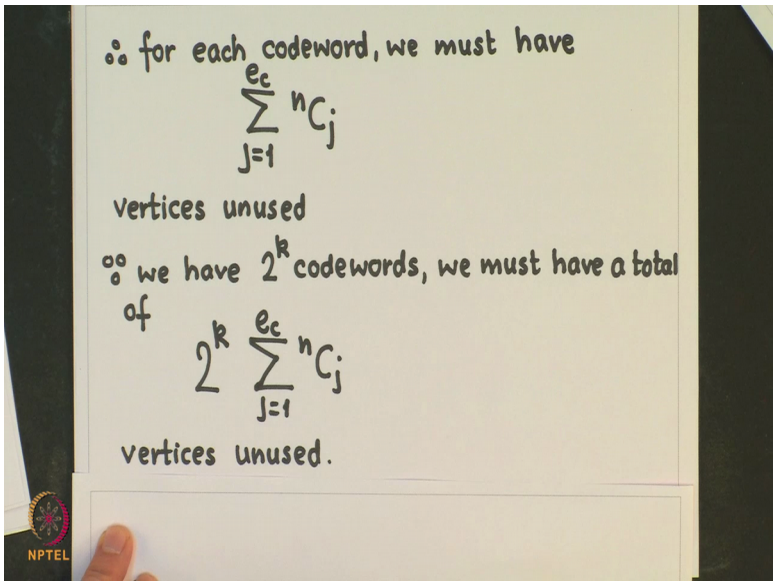


So, assume that we have n comma k linear code. So, what this implies is that we have 2 raised to n vertices available for 2 raised to k messages correct. And 2 raised to n minus 2 raised 2 k are redundant vertices, which we have not used in the vector space of n tuples. So, the question is now, how many vertices can lie within a hamming sphere of radius ec, which is the desired error correcting capability of my n comma k code correct.

So, now number of n tuples that differ from a given n-tuple that is a sequence of length, n by j bits is the number of possible combinations of n things taken g at a time and this is given by n C j correct. So, therefore, the number of ways in which up to ec errors can occur is given by the summation of this expression correct. So, n C j could be from 1 up to ec. So, you form all the combinations with 1 error, 2 error, 3 errors up to the correctable desired errors which is ec.

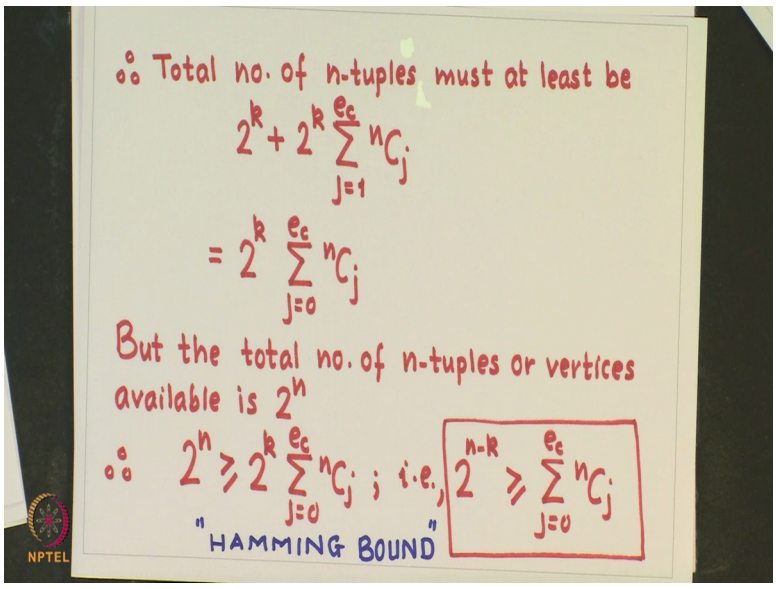
So, what this implies that for each codeword we must have this many number of vertices which cannot be used.

(Refer Slide Time: 21:46)



So, now, there are 2 raised to k codewords. So, what this implies that we must have a total of 2 raised to k multiplied by this quantity vertices unused for 2 raised to k codewords. So, from this we can conclude that the total number of n tuples must be such that it should be at least equal to this quantity given here.

(Refer Slide Time: 22:25)

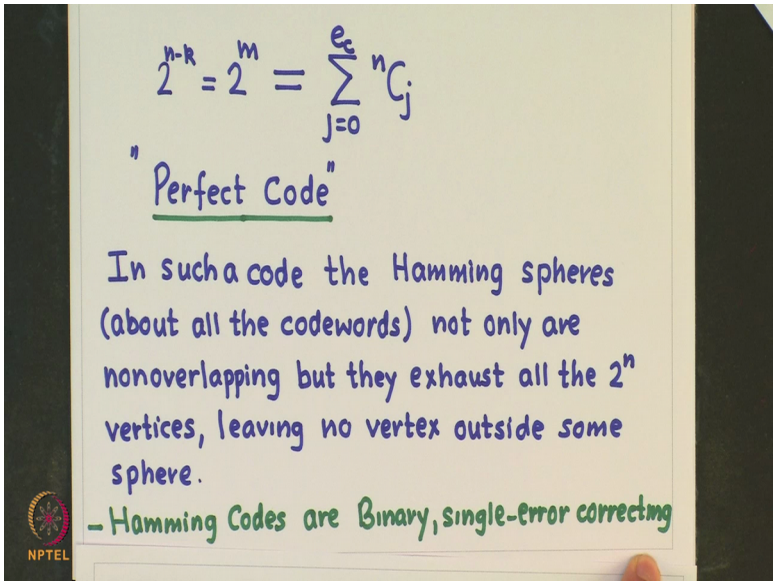


So, for each codeword I cannot use this many number of vertices because so many vertices would be lying in the hamming sphere of radius ec.

There are 2 raised to k codewords. So, so many number of vertices cannot be used for all the 2 raised to k codewords and we also have to have 2 raised to k codewords. So, the total number of n tuples must be at least this quantity now this can be written as here. Now we know that the total number of n tuples or vertices, which is available in a vector space of n-tuple is equal to 2 raised to n. So, from this we conclude that 2 raised to n this quantity the total number which is available, should be larger or equal to this quantity and from this we get this inequality which is known as hamming bound.

So, this hamming bound has to be satisfied if I want to get error correction up to ec wrong bits. Now there are some codes for which this inequality would become equal equality.

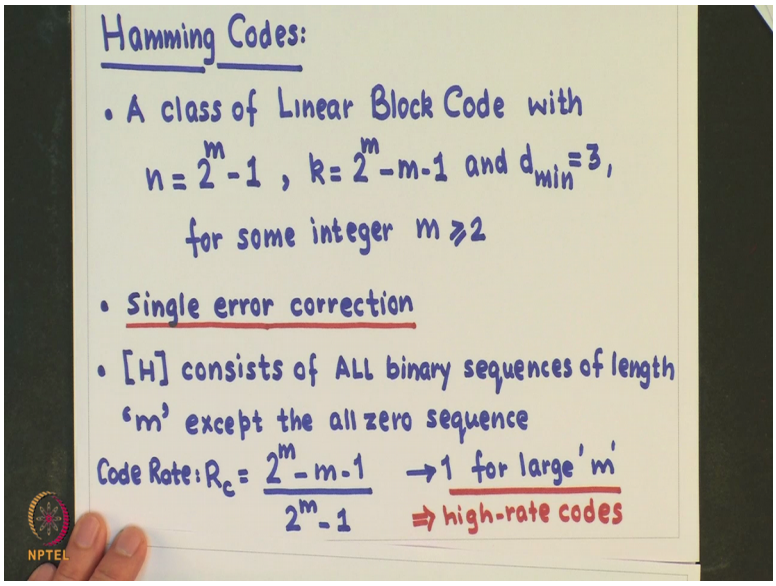
(Refer Slide Time: 23:57)



So, what this implies is follows; this is known as a perfect code and in a such a code the hamming spheres about all the codewords not only are non overlapping, but they exhaust all the 2 raised to n vertices living no vertex outside some sphere correct. An example of a perfect code is the hamming code, which are binary single error correcting codes.

So, the hamming codes which are systematic linear codes and they are perfect code have the following property.

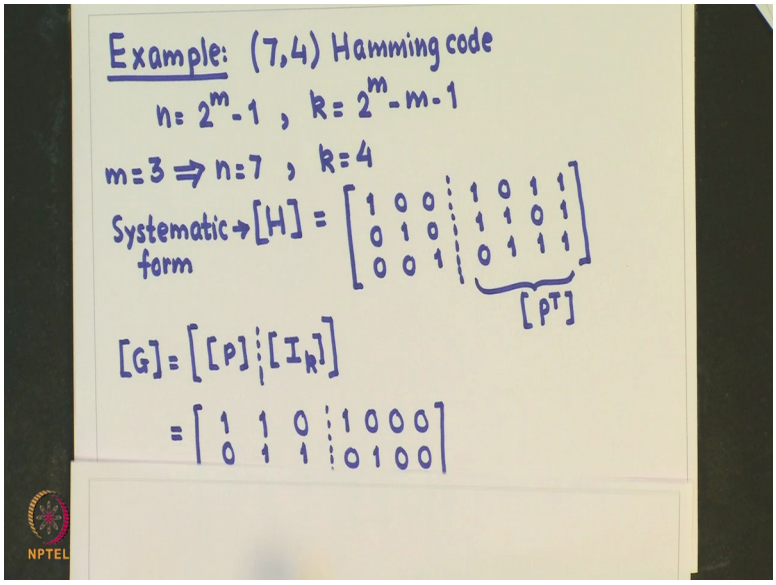
(Refer Slide Time: 24:50)



So, it is a class of linear block codes, which satisfies this equalities n is equal to 2 raised to m minus 1 k is given by this expression and d minimum for all the hamming code is equal to 3 what this implies that, its error correcting capability is just equal to 1. So, this is single error correction only possible and to get the hamming code, we can easily form what is known as parity check matrix and this consists of all binary sequences of length m except the all 0 sequence.

And the code rate for this which is defined as k by n is given by this relationship, which is obtained from here. So, and this tends to 1 for large m correct. So, we want in practice the code rate to be high, but the disadvantage is that it has only single error correction capability.

(Refer Slide Time: 26:10)



So, let just take 1 popular hamming code, which is 7 comma 4 hamming code for which if you substitute this value 7 4 into this requirement for the perfect code, we see that it satisfies and the systematic form of the parity check matrix is as shown here.

So, we have used there are 7 columns out here except for the 0 sequence of 3 tuple, all other sequence of 3 tuples have been used to form this parity check matrix. So, once we know this I know my parity array from this, because this corresponds to the identity matrix we are assuming systematic form. So, from this we can get the generator matrix as follows. So, code could have error correcting capability or just error detecting capability or both.

Now, error detection decoding is quite simple; the received codeword is multiplied by the transpose of the parity check matrix to obtain the syndrome and if the syndrome is nonzero, this implies that the received codeword is non codeword. So, in this case the receiver via the reverse communication channel informs the transmitter for retransmission, but how do you achieve error correction decoding? We will try to find that answer to this question next time.

Thank you.