Digital Communication
By Professor Surendra Prasad
Department of Electrical Engineering
Indian Institute of Technology, Delhi
Module 1
Lecture 38
Elements of Error Correcting Codes (continued)

Started to talk about error correcting codes yesterday, as I said we will be only able to do a very brief coverage of this topic and yesterday I tried to tell you the basic concept behind linear block codes we said that they are essentially two most important kinds of channel coding schemes are also called block coding schemes and the convolution coding schemes all in which we will be very briefly be able to touch the linear block codes. We also saw that channel coding is essentially distinct from coding the purpose of source coding is to remove redundancy whereas the purpose of channel coding is to introduce redundancy in a controlled manner so as to be able to achieve error free communication at least as close to error free communication as feasible.

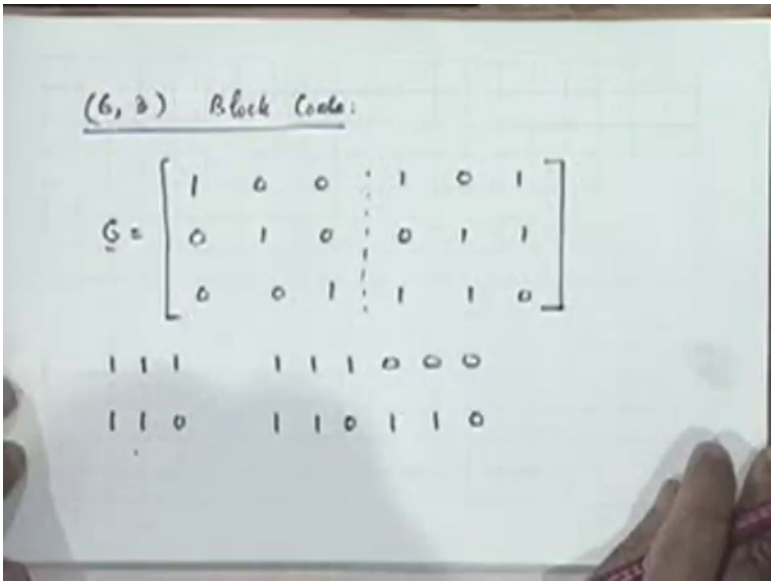
In any case we worked out certain relationships between the n and k parameters of an n, k block code, right? And n, k block code consist of a mapping scheme in which a group of k information bits are coded into a group n code bits, right? And the relationship between n and k that we saw was the so called hamming bound, right? And that hamming bound requires us to be able to first specify what properties you want from your error correcting codes and this feature is typically specified by specifying how many errors we would like to be able to detect or correct, right?

So the hamming bound is defined in terms of three parameters, the value n, the value k and a value t where t is the number of errors that you want to be able to correct, okay. So based on that we obtain the hamming which tells us how to choose the third given any two of them, right? No all the three are related n, k and t are related to the hamming bound, right? That is all I can say. So given if for any particular reason you have a reason for choosing two of them to be specific values than third will be limited by that, right?

And then we introduced the linear block codes through the generator matrix g in particular we are considering what we called systematic linear block codes, right? In which the first k bits of the code are precisely the same as the k information bits or data bits that we are encoding, all

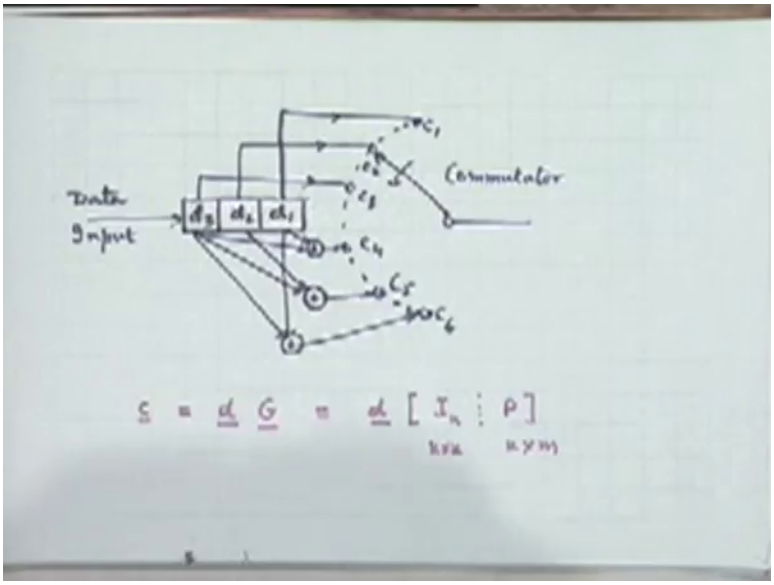
right? And the next n minus k bits or m bits are called the check bits or parity bits and they are generated by taking linear combinations linear combinations of the information bits or data bits, right?

(Refer Slide Time: 4:18)



And this whole this is specified by a generator matrix g, for example this was example we considered for a 6, 3 block code, okay in which this was a generator matrix g this is the identity matrix which tells me that the first three bits will be the same as the information bits and this is what is called the parity check matrix which specifies the check bits of parity bits in terms of data bits, right? This is an example if your input sequence input information bits sequence is this your output coded sequence encoded sequence this 6 bit sequence, right? When it is this it is this and so on, I think this is where we stopped yesterday.

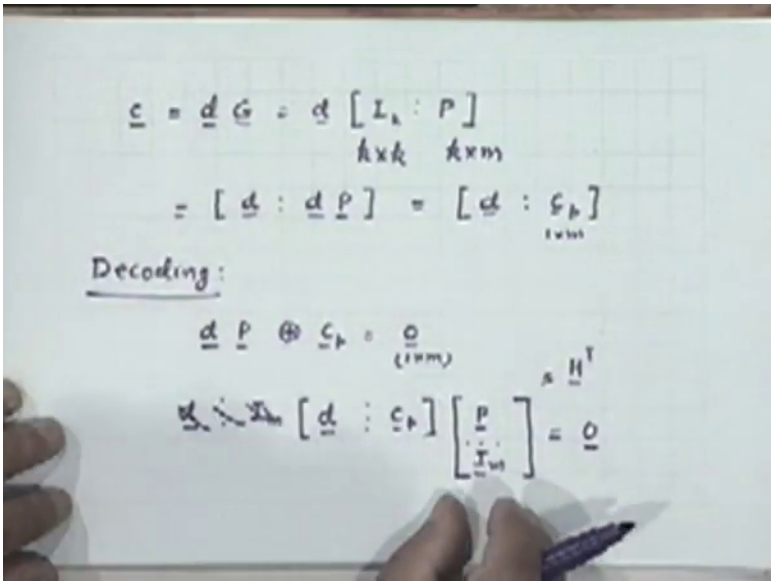
(Refer Slide Time: 5:20)



So I have done a very quick revision of what we did yesterday. Now schematically I can represent this implementation of this generator matrix implementation like this, your data is coming in which is being stored in a some kind of a shift register or something some memory device and then basically d1 is being picked up first by this commutator, d2 is being picked up next this is the first code bit, this is the next code bit, this is the third code bit and the fourth code bit is obtained by taking linear combination appropriate linear combination of this bits the fifth also a different linear combination sixth also a different linear (())(5:54), right? So essentially it generates this and transmits them one after another, right? So this could be a schematic representation for the generation of linear block codes, okay. To proceed further let us recapitulate that C the code word vector is related to the data vector the information bit vector d through the liner transformation by G the linear transformation G that is the vector C is the vector D times G where C and d are what kind of vectors, row vectors not column vectors if those column vectors then (())(6:49), okay.

Now G in turn has a specific structure, right? What is a structure? It consist of it can be partition into two parts and identity matrix I sub k where k is the length of your information sequence and a parity check matrix of size k into n minus k, right? k into m this is k into k this is k into m this is (())(7:27) on the next at least you can write this on the next page I will use the different color I will return to my normal color that is purple, is it fine?

(Refer Slide Time: 7:50)



Now so, okay I will rewrite this just in case you are having problem C is equal to this is okay this is visible? Fine? Now let us take this d inside this if I multiply this d with I sub k the identity matrix I will get the vector d itself, right? And multiplying with this will give me some vector d times p which I will write as let us say C sub p, what does this tell me? It tells me very clearly what we have already understood that the code vector C is comprised of two components , right? The first k bits of the code vector was it is nothing but the data vector itself.

And the next m bits are the this is 1 into m row vector because this was k into m P was k into m this is 1 into k when you multiply that will get a 1 into m vector. So this is a vector m vector of dimension m consisting of the so called parity bits or the check bits, right? That is a structure that comes out also from the structure of G which we have chosen for a systematic (())(9:19), right? So for a systematic code this is a structure of the code word.

So as far as encoding is concerned I think you have understood what linear how linear block codes are generated given a specific generator matrix G, okay. Now let us see suppose we want to decode a receive sequence which has been generated like this and then transmitted through a noisy channel decoding of such coded sequences. For that we will like to first study certain basic properties of the generator matrix to start with we may you collect that from any binary sequence when added to itself exclusive add bits itself will produce a all 0 sequence, right? I will start

from there, now dP and Cp are really the same thing, right? Because C sub p is equal to d multiplied by p.

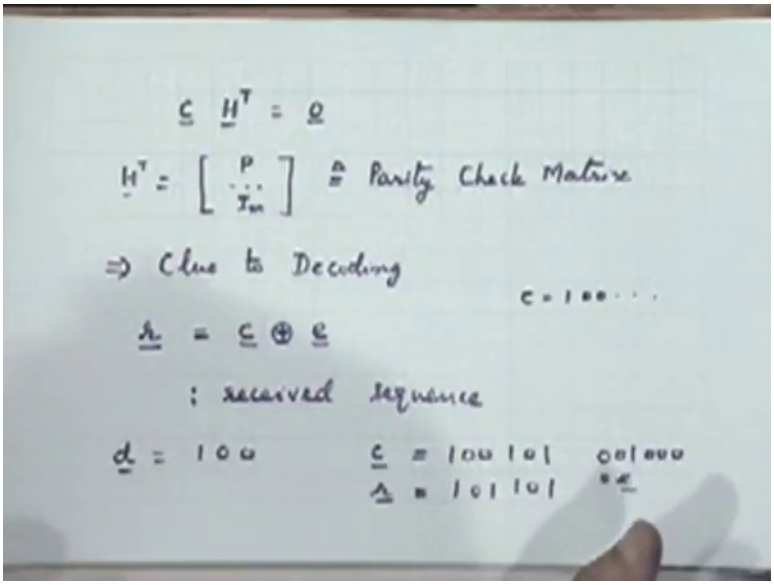
So if I write like this you will all agree with this this will be equal to a vector which is consisting of all 0's, right? To m because C sub p was a vector of length m, okay which I can write this relation again I can write in matrix form this is a when I say linear combination you were absent yesterday I think this basically we are working in a special field we are not working the real field of numbers we are working in a (())(11:07) field of numbers in which are essentially in this case binary numbers, right? The addition operation defined here is the exclusive OR operation or the modular 2 operation, right? Modular two addition. So we are working in that number system here.

So in matrix form we can write this as a d this I am sorry I have to rewrite it we know this is our code vector it consist of d and Cp, right? This equation I can rewrite like this d is being multiplied by the matrix p, right? And Cp is being multiplied by the matrix what is it being multiplied with? Identity matrix and what is the dimension of this identity matrix? Not k by k it is m by m because this is a vector of dimension 1 into n, right? And this results in a m dimensional row vector 0, right?

So I am just rewriting this equation in a matrix form, not very clear this is just matrix multiplication, is this clear this equation is okay? Now I am just writing this equivalently as equal to this because if I do a block multiplication of this vector and this matrix I will l get a same equation again d into p plus Cp into Im which is Cp is equal to 0 you are taking a row vector multiplying by column matrix on a block by block cases, okay simple matrix multiplication really nothing else, okay.

Now this matrix also has a very important purpose to serve as we can easily appreciate and therefore it is got a special notation for itself it is denoted usually by h transpose, okay this matrix that usual interrupt by h transpose, what is d along with Cp this is your the code vector C, right? In places of information bits followed by check bits, so what we are saying is that every code vector C which is generated by this data matrix G satisfies this equation, right?

(Refer Slide Time: 14:01)



You know what is the equation that we are really talking about is C multiplied with h transpose is equal to 0, right? Where h transpose is this matrix and it is defined to be the parity check matrix and the reason why it is so called is quite obvious though, is it obvious?

The way it works is it helps us is if we are given a specific sequence, right? If you multiply with h transpose if the result is 0 then you know that given sequence was invalid code word generated by that matrix G. On the other hand if it does not produce a 0 vector here that means it is not a valid code sequence, okay that is why the matrix h is called a parity check matrix as it is able to tell me where are the parity bits and the data bits that are coming along they are the same kind of relationship with respect to each other which is implied by the structure of the G matrix, right? It is the check on the structure of the G matrix and therefore we call it a parity check matrix and this forms our basic clue to decoding, right?

Of course we are trying to decode a receive sequence that receive sequence may or may not be a valid code word because if errors are taken place of the receiver due to channel noise and various other reasons at the moment of course we are only talking about channel noise we will have a receive sequence r which is not going to be C but associated with some let us say error pattern e, right? For example the error pattern by contain all 0's expect a single 1 somewhere that will correspond to and error at that specific bit location.

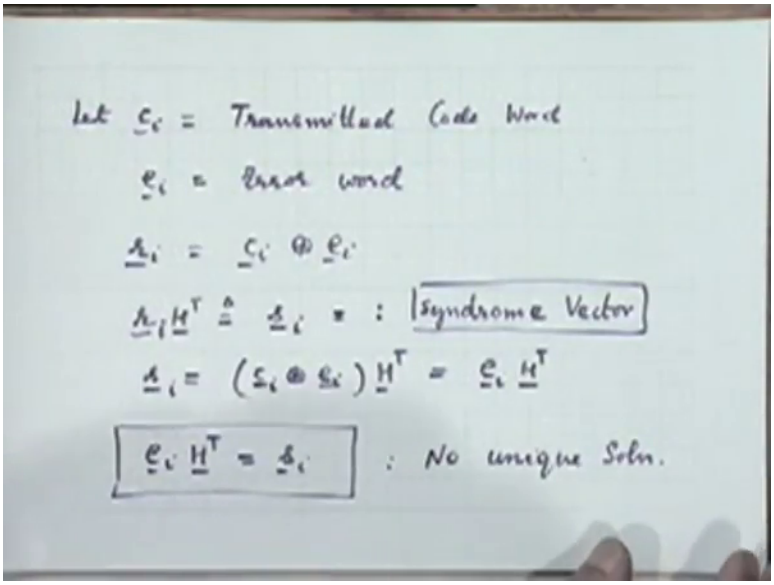
Suppose a first bit of this error pattern is 1 and the rest are all 0, right? That means the first bit has been anonymously decoded in the received sequence I mean the received sequence contains a first bit to be to be anonymous, right? For a different error pattern we will have a different receive sequence so you might have transmitted C but at the output of your let us say batch filter receiver after your thresholding you are getting a sequence r which is different from C, right? And this is what now the decoder has to do it has to take in this input sequence and hopefully take it back to C so that there are no errors, right? That is a functionality decode that is what decoder is required to do.

So this is your received sequence, what are the dimensions of vector C and e? Both are n dimensional vectors C is an n dimensional code vector e is an n dimensional error vector most of it is expect to be 0's in fact if there is no error then e is perfectly 0 is consisting only of 0 bits I mean all bits being equal to 0. For example if d you know for the same 6, 3 code that you are considering let us say the data bits were 1, 0, 0 and the corresponding code word you can check that will come out is this from that generator matrix which we were considering a few minutes ago, right?

If you take this as a generator matrix if you feed this as a data vector input vector the code vector that will come out is this, okay and maybe your received sequence is this instead of what you transmitted, right? When I say received we are really talking about the output of your threshold device at the receiver because it is there that the decoder will be fitted. So what is the error pattern associated with this received sequence? 001000 that is your e, okay do you understood that? All right.

Now, yes we are coming to that I am not yet given you the decoding procedure, I have to still I am only trying to tell you that the received sequence can be modeled as the sum of the codes actual code sequence that you transmitted and an appropriate error sequence, right? Depending on how many errors have taken place where, right? That is all I have said so far.

(Refer Slide Time: 19:17)



Now let us say C sub i is the actually transmitted code word and let us say e sub i is the corresponding error word that you get, okay if e sub i is 0 of course there is no error and you already seen that r sub i the corresponding receive sequence is C sub i plus e sub i, okay.

And if you compute r H transpose and if it turns out to be 0 that will imply that ei e sub i is 0, right? In general it will not turn out to be 0 it will turn out to be some non-zero vector, right? Instead of getting a vector of 0's m 0's we will get a vector of an n dimensional vector of 0's and 1's, right? In general, if it is not 0 it has m components some of them will be 0 some of them will be 1's, right? That is (())(20:31) expected situation.

Precisely speaking this s will be the vector s that you will get you can again write C plus ei into h transpose but Ch transpose is 0, right? It is nothing but ei h transpose, okay I am writing s is equal i h transpose r is Ci plus ei I should probably write s ri write this si, right? Let me okay, it is okay? So this kind of might give you the feeling that there exist a mapping between this s that you can compute from your given r you have received this sequence r sub i, right? From which you can compute s sub i my multiplying with h transpose, right?

So it might give you the impression that given this s sub i which I can compute from the received sequence I should be able to obtain e sub i, right? Because there seems to be a linear relationship between these two, right? And once I have the e sub i I know my transmitted code word because I know where errors are taken place and therefore I can correct those errors, right? But things are

not that easy as we will try to appreciate. The reason is this equation is e sub i h transpose can you give me the reason what is the difficulty, this equation does not have a unique solution, okay and can you appreciate the reason why it is not a unique solution?

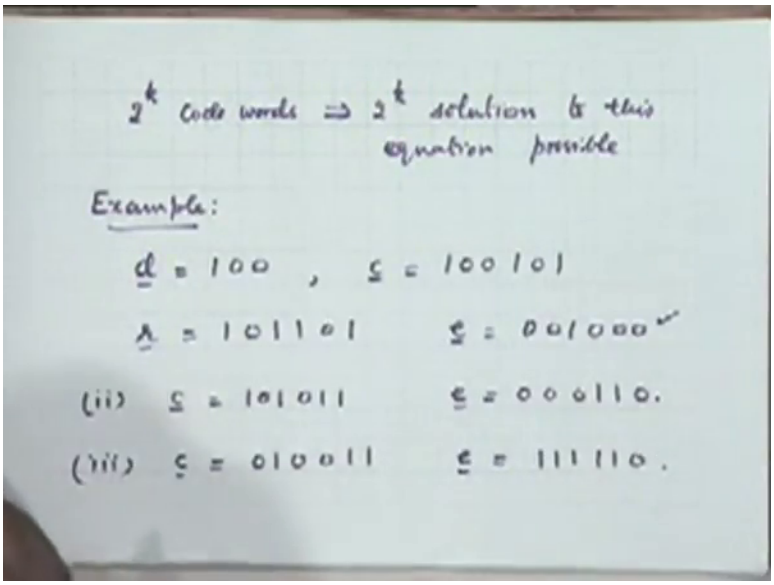
The reason is as follows, corresponding to every C that you might have transmitted, listen to this carefully corresponding to n you see you might have transmitted you can always think of some other code word and some other ei some other corresponding error pattern which would have produced the same received sequence and therefore would have produced the syndrome, this incidentally this vector s sub i I think I forgot to mention this is called the syndrome vector or simply the syndrome, okay so do you appreciate that given a particular syndrome vector it may not be easy to go from there to e sub i because I can produce a different e sub i corresponding to a I may had different there may be many different e sub i's which will produce the same syndrome corresponding to the fact that for each code word I can identify a different e sub i which will produce a same r, that is right so we will come to that, right?

But by itself it does not contain (())(24:13), not only that we would like to what we like to do is, no matter what structure of h you might have this fact is not related to structure of h this fact is related to the fact that si has been generated from what, from r sub i, right? If I have a given r sub i s sub i is fixed there is no confusion about, right? Because s that is being generated like this but I could get this r sub i by different combinations of code words and error patterns you might have transmitted the specific code word but at the receiver I do not know what you transmitted so I can think as if some other code word is transmitted and some other error pattern was associated with it to give me the specific r sub i that I have obtained, yeah so the uniqueness will come by imposing some other but have you understood this point I am coming to your point in a few minutes but is this point all clear that this does not contain a unique solution, right?

This equation does not have a unique solution is this point understood, right? From there I can proceed on and come to the point such you are making, yes but at the moment I have gone away from the sphere interpretation but yes basically that is what you have been, right? Because demodulation is already been done we have already got r sub i which is a binary sequence, right? Yes of course it has to it has to know h and to know h it has to know p, how will I know communication system is hopefully being designed by a set of individuals who are designing both transmitters as well as receiver, okay so there is no what they are (())(26:29).

So that is easy that is part when you design encoder there is a specific decoder structure also implied, okay that is right you cannot have an arbitrary decoder when you without knowing what your coder was, right? Somebody had use a cyclic code and you trying to decode it in a different way it is not at all relevant to cyclic codes then obviously being something wrong, okay. So the parity check matrix is known to the decoder, all right.

(Refer Slide Time: 27:16)



So the point is that there are 2 to the power k code words there can be 2 to the power k solutions to this equation, let me illustrate by again an example let me take an example I think you have understood this point, isn't it? That there are 2 to the power k different possible solutions, right? You understood this point because for every valid code word I can think of an error sequence which would produce that r sub i, isn't it? And therefore it would have produced that specific syndrome that has been produced, that is the only answer for every specific code word and how many code words are there 2 to the power k I can find some error sequence such that that code word plus that error sequence will produce that specific received sequence that I have got, right?

And therefore it will produce that same syndrome because the received sequence is same correspondingly, right? Let me give you an example let us take d equal to 1, 0, 0 again the example in the context of the same 6, 3 code that you have been talking about C corresponding C is 1, 0, 0, 1, 0, 1 and we said r was 1, 0, 1, 1, 0, 1 and I actual error pattern is 0, 0, 1, 0, 0, 0, okay this is the actual error pattern corresponding to this situation.

Consider C equal to 1, 0, 1 that is when the data sequence is not 1, 0, 0 but data sequence is 1, 0, 1 then the corresponding code word will turn out to be this, right? And it is very easy for you to check that if take e like this then I will get the same r, okay or, isn't it? This is 1, 0, 1 as such and 1, 1, 0 add a two 0 1, 1 gives rise to 1, 0, 1. Similarly if I take the code word 0, 1, 0, 0, 1, 1, right? And the error pattern 1, 1, 1, 1, 1, 0 I will again get the same received sequence r you add this two you will get the same r, in fact it true for every possible valid code word and also now we know what we should do here is a situation where there is a single error, here is a situation where there are two errors, here is a situation where 5 errors which is a 1 it is most likely to be the case the single error case.

So the answer is find the solution for the error pattern which is associated with the least weight least hamming weight, hamming weight is defined to be the number of 1's in sequence, right? Which has a least hamming weight there will be some least, okay so least is what you have to check, if you have that choice then of course you cannot do much you can choose any one of them unfortunately but usually that no theoretically there is a possibility of that happening, no that is no possible because no no it is may not be possible if your code is properly designed if your located your code words sufficiently far apart that is it takes a large number of errors to go from one code word to another code word, right?

If your code words are properly designed that will never happen in fact that is what we discussed last time, you are forgetting our discussion yesterday and the time before that where we did a geometrical pasteurization of this code word code book designed where we said that if you want to design a code book it should be designed in such a way that along every code word if you want to correct up to t errors then you have to have the hamming spheres of radius t around each code word to be non-overlapping with respect to each other, right?

If you have that kind of situation you will always have one unique solution, right? Because corresponding to one error there will be only one code word which is nearest to it, right? And basically that is what you are saying you are finding that code word which is nearest it (()) (32:45) least error, right? That is right, no here even that does not come into the picture here the only thing that is coming to the picture is choose the error sequence which instead of which kind of implies minimum number of errors, right? Because smaller number of errors is more likely to happen then a larger number of errors in a normal channel therefore you give more weightage to

this possibility then to this possibility or this possibility mind you this is possible if not this event is not possible it is possible that you actually transmitted this and you received, sorry you actually transmitted this and you received this and therefore two errors took place, right?

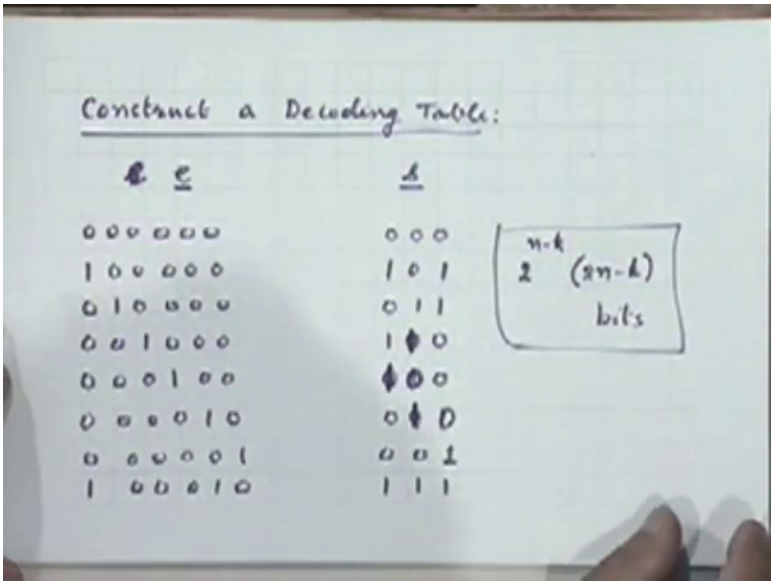
But there is no the point is the possibility of that event is less than that you transmitted this and received this, it does not matter where, right? No that will imply a different receive sequence that will imply a different receive sequence if you transmit this and your error pattern is 100000 then your corresponding received sequence will be different, what is your confusion can you specify your confusion? Let me (())(34:31) again yes yes that is implied of course no see that t may be a large number and you may have several solutions less than t then what you say, then why do you have to go through a two-step procedure, right?

(())(35:01) choose straight away the situation which corresponds to minimum we of course saying that minimum should be less than t, right? But that is secondary, one at a time, yes but within that sphere there can be many patterns with one error sure but they will correspond to a different receive sequences, isn't it? There can be many error patterns corresponding to a given transmitted code word resulting in different receive sequences for A received sequence there is only one unique way by which you can go from there to its nearest code word, right? That situation is hypothetical it is not worth discussing what is you are saying that because you never have a least which is greater than t, right? The least will have to be less than C, no no the question is you can please listen to me for a given received sequence I can construct error patterns containing single errors containing double errors containing triple errors and so on and so forth, right? Is it clear?

Now for a given syndrome you find out which are the possible error patterns, right? Maybe you get one error pattern of a single error and one error pattern of a double error, one error pattern of triple error then you choose a single error one that is all we are saying nothing more will (()) (36:57), obviously yes, is what always non-zero? No it maybe 0 in that case we will get a syndrome to be all 0 that is how, right? And then we know that we are fine we do not have to do any correction, right? If we get the syndrome vector to be all 0 then you are there no further effort is required, alright.

So basically what we are saying is use majority logic decoder, right? Essentially it binds down to that that is find the code word which is nearest to the received sequence basically that is what you are saying, all right.

(Refer Slide Time: 38:00)



Therefore what you really do is for decoding you construct a decoding table, see there is a further problem here how many different error patterns I can generate? 2 to the power n, right? Obviously I have because that is why I have to put a limit I cannot correct all these error patterns, right? The every code will have a limit up to which it can correct depending on how it is designed depending on what is a minimum distance between code words this was a crucial parameter we discussed if you want to correct up to t errors what is a required minimum distance between the core words, 2t plus 1, right? Every code word must be 2t plus 1 away from the other code words or maybe other code.

Now therefore I cannot possible hope to be able to correct any kind of error patterns that might take place, right? That also can be appreciated for fact that your, the vector s what is the dimension of vector s? Its dimension is m m is n minus k what is a number of different possible syndromes I can have, 2 to the power m 2 to the power n minus k so the maximum number of error patterns that I can correct are 2 to the power n minus k, right? The syndrome vector has a dimension of m the vector s this vector or this vector has a dimension m because this is a 1 into k vector this is k into m matrix this is 1 into m m is a number of check bits, right?

So the dimension of the vector s sub i or s is m where m is a number of check bits. So the number of different pattern syndrome patterns I can have is 2 to the power m, right? Therefore if I want a unique decoding procedure that is for 1 syndrome vector one unique error pattern associated with it, then how many error patterns I can possible take care of? Only 2 to the power m one corresponding to each different syndrome vector, right? Therefore the procedure consists of going through a mapping or a table construction in which every possible syndrome vector you associate a corresponding error vector, right?

And obviously the error vector should such that find a minimum weight error vector corresponding to that syndrome, right? So and the best way to do it is so for example let us take the same example 6, 3 code what is a value of m, d how many different syndrome patterns I can generate? 8, right? Now I have to therefore I can take care of only 8 error patterns whatever they might be and I have decided that I will give more weightage to error patterns with single errors then maybe perhaps if I have error correction possibility even beyond that then I can take care of double errors and so on, right?

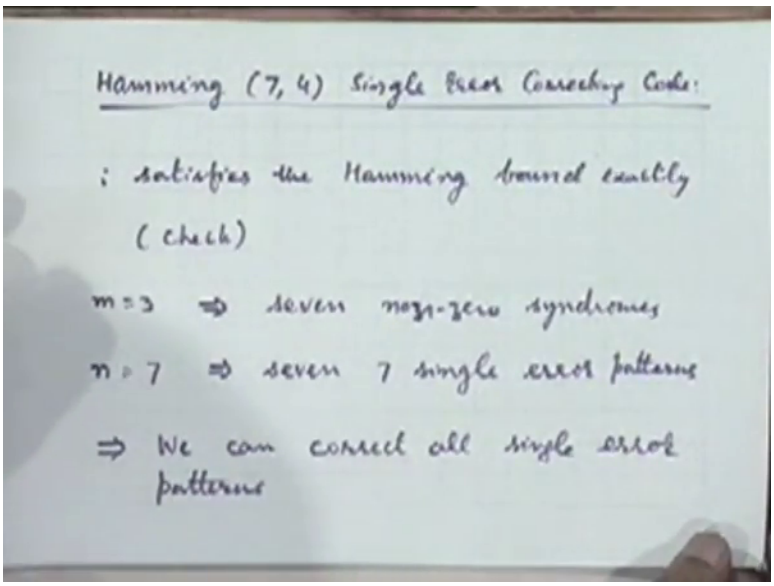
So what I will do therefore is I will list out first all single error all the 0 error patterns that correspond of course to the syndrome all 0 that is trivial then I have list out all error patterns with single error, right? Start with 100000 compute what is a corresponding implied syndrome how will you compute that, eh transpose, right? 101, right? Then take the next single error pattern this will turn out to be this then you take the next one, right? This will turn out to be this, next one turns out to be 010 next one turns out to be 001, sorry I think I have made some mistake 101 011 this is 100 here, no this was okay I am sorry for this this is 100 and this was 010, okay you are left with one more this gives you 001, how many patterns are exhausted? 7 there is one more pattern left so if you want I can take care of 1 error pattern with two errors, right? Maybe this one you have to find out which error pattern of two errors will produce this syndrome it turns out to be this, okay.

So now I have a decoding table, I will look at the syndrome find the minimum weight error pattern from here add that error pattern to the received word and I have my decoded sequence, is the procedure clear? So you will have the received sequence coming in that is operated by the parity check matrix to generate the syndrome and then the syndrome output vector maybe is given to a look up table in a ROM or something to produce the output coded output code

sequence error pattern sequence from which you can construct the codes you could as well have the code sequence outputted as the, okay what is the storage required here? You require to store how many patterns in this table? 2 to the power n minus k each with how many bits n plus m minus k, right? 2n minus k bits this is required size of the ROM that you may (())(45:03), okay you required to store 2 to the power n minus k words each of length 2n minus k bits, to store this table, right? So that if you adjust the ROM using this address you will get this output sequence to which you can add your receive sequence and get your code sequence.

Now we have only 5 minutes but I can briefly discuss one this about how do you choose because we are not really going to design of codes as I said that is a full-fledged subject by itself but I like to offer a few comments how do you choose the entries of the g matrix or the entries of h matrix such that we are able to correct all this single errors, alright? How do you really do that? But that is not an easy job that is quite really no systematic way of designing or generated matrices like that but for a specific kind of codes which are called hamming codes hamming single error correcting codes it is relatively easy to do this design and I will just like to go over that way, are you familiar with the hamming code in some other context in any of your other courses? No, okay.

(Refer Slide Time: 46:38)



So we will discuss briefly the hamming 7, 4 single error correcting code, okay yes that is right each of this error parent sequences multiplied with h transpose that is the entry here, this is h

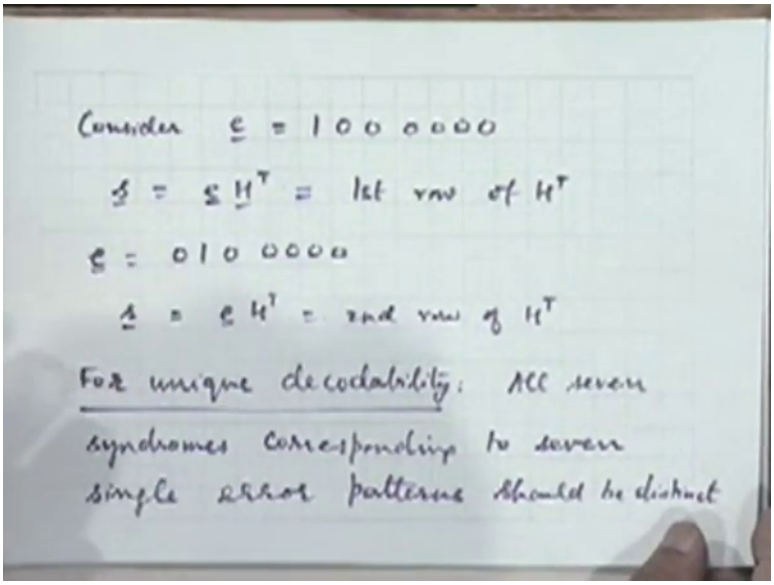
matrix h transpose this is m by m and this is m into sorry this will be k by m, no how it can be k by k? right? You this is what I try to clarify right at that stage this is how we got this h matrix, right? Let us go through each dimension in case there is a confusion this is 1 into k this is 1 into m this whole thing becomes 1 into n, right? Now this is 1 into k this matrix has to be k into m, right? And this matrix this Cp is being multiplied by Im which has to be identity matrix of dimension m into n, is it clear? Yeah, let me the hamming bound is satisfied exactly for 7, 4 not (())(48:41), okay this 7, 4 comes from the hamming bound it turns out that the two sides will precisely be equal if you take n equal to 7 and k equal to 4 and for t equal to 1, right? You can substitute and then find out, right?

So you cannot do it with k equal to sorry n equal to 6 this is what you are saying, isn't it? What is 3 here? k yeah but I am doing something better than that, isn't it? Alright? You give me few minutes more, as I said you can easily verify that it satisfies the hamming bound exactly please check, okay. Now what is a value of m? It is 3 3 check bits are implied in a 7, 4 code which implies how many non-zero syndromes? Total is 8 out of which 7 are non-zero, right?

So 7 non-zero syndromes, okay. What is a value of n? 7 how many error patterns are there with exactly a single error? 7 error patterns, why 6 when code is of length 7 you can have a error at each point each location, right? Separately, so there are exactly 7 error patterns with single errors, okay. So we can correct it seems to be possible to associate every syndrome sequence with a corresponding error pattern sequence of different kind but corresponding to a single error, so why did they choose our h matrix or g matrix properly, right? The possibility is there because you have exactly 7 syndrome sequences and exactly 7 single error pattern sequences the question is how to do it.

So it seems to be possible to correct all single error sequences all single error patterns and of course no more than that because we do not have any syndromes left after that.

(Refer Slide Time: 52:00)



Let us consider the situation in the error sequence is 10000 what should be your s equal to eh transpose what will it correspond to? It will be nothing but the first row of h transpose, isn't it? Is it clear, alright one more is it fine is nothing but the first row of h transpose not column because syndrome sequence is m dimensional vector it cannot be the column, column is an n dimensional vector, alright?

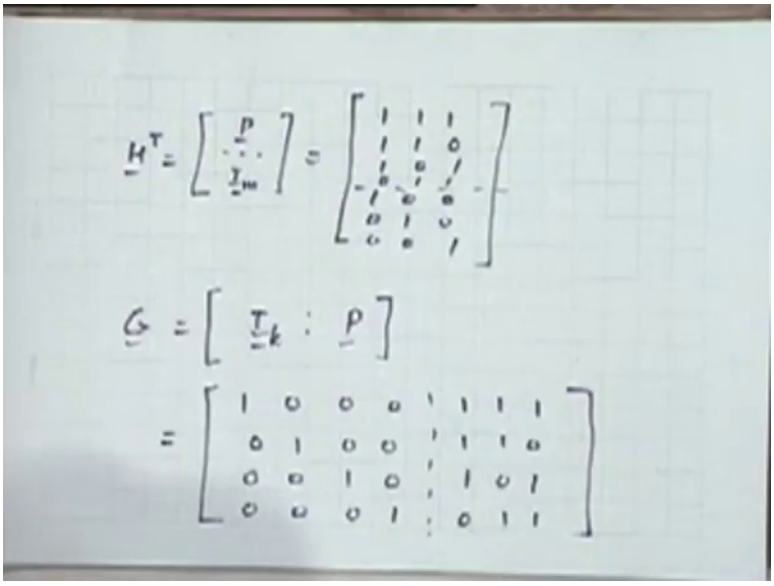
Similarly if I take the sequence e equal to 0100000 what will be s, again it will be (())(53:05), right? Therefore what does it mean? If I want all this 7 syndromes sequences to be associated with 7 different error patterns 7 distinct error patterns then all I have to ensure is that the all different 7 rows of h transpose should be distinct, right? Is it clear? If I want each of this syndrome patterns to be associated with are different distinct single error patterns then the way to ensure that is that these h matrix rows should be all distinct because that is what I will get a syndromes if 1000 I get a syndrome sequence for 01000 I should get a different syndrome sequence which is lying in the next row of h transpose, for the next third error I should get a different syndrome sequence which lies in the third row in so on and so forth.

So if all the rows of h transpose are different I am (())(54:06) 1 to 1 mapping between each of this single error patterns and a corresponding syndrome sequence, clear? And that is very easy to do because we can construct exactly 7 non-zero sequence with 3 bits m is equal to 3 so all I have

to do is put them in any order I like except to ensure that the lowest part has to be parity matrix that is how, so that is the generation of a hamming code.

So for unique decodability all 7 syndromes corresponding to 7 single error patterns should be distinct this is a point and the only way to do that is to make sure that h transpose contains all distinct rows, okay.

(Refer Slide Time: 56:12)



So one possibility of a choosing h transpose is possibly like this you know that top part of it is p and the lower part of it is I sub m so the lower part is specified to be 100010001 this is specified, right? As far as the top part is concerned you have only 4 other non-zero sequences you can put them in any order you like, right? Maybe 111 110 101, right? 011, alright?

So that is the hamming code for you corresponding g will be constructed by like this that will turn out to be and P you know its 111 110 101 011, right? So you have a generator matrix and you can generate the code and then transmit. So the generator matrix and the check matrix are closely linked by virtue of design, right? So without if you do not know at the decoder what check matrix is then that means you are not working as on a single corresponding encoder decoder pairs, okay.

I think with that we will you an generalize this to other single error correcting hamming codes not necessarily 7, 4 but for arbitrary values of k and n, right? In fact the same procedure can be

fit in. So for hamming codes single error correcting codes of this kind is really no problem in design but in general to design code with specific number of errors capability error correcting capability is a non-trivial job and is a subject of full-fledged course in coding theory. I think this is a good point at which we can stop.