INDIAN INSTITUTE OF TECHNOLOGY
ROORKEE

NATIONAL PROGRAMME ON TECHNOLOGY
ENHANCED LEARNING
(NPTEL)

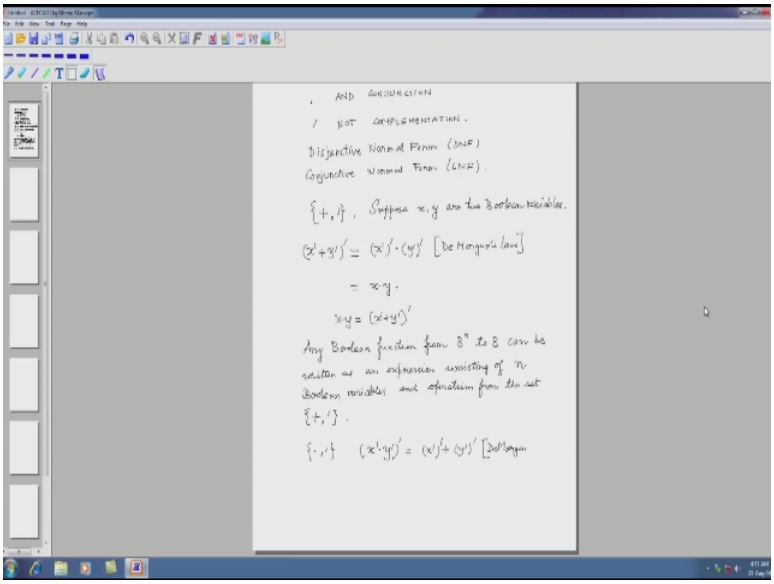
Discrete Mathematics

Module-08
Boolean Algebra and Boolean function
Lecture-03
Boolean functions (2)

With
Dr. Sugata Gangopadhyay
Department of Mathematics
IIT Roorkee

In today's lecture we continue our discussions on Boolean functions.

(Refer Slide Time: 00:56)



What we have seen so far that a function from BN to B that is called a Boolean function. And we have also seen that a Boolean function can be represented by Boolean expressions. Now a Boolean expression we mean an expression consisting of Boolean variables and the operations plus, dot and complementation. Of course, there are other names of these operations the plus is

sometimes called OR or disjunction, this dot is called AND or conjunction, and this prime is called NOT or complementation.

Now we have also seen that there are some standard forms of expressing Boolean functions. Two most important of those standard forms are disjunctive normal form and conjunctive normal form, in short written as DNF and CNF respectively. So we have disjunctive normal form written DNF, and conjunctive normal form written CNF. At this point we ask a question that can we represent a Boolean function by expressions consisting of something less than the Boolean variables and plus, dot and complementation.

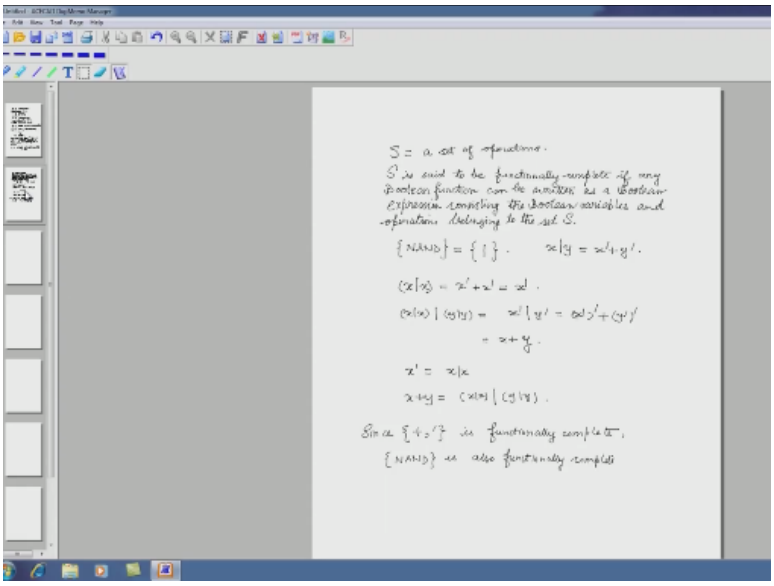
The answer is yes, and first example is just the set consisting of plus and dot, I am sorry plus and complementation. Suppose, we have two Boolean variables and consider the expression X complement plus Y complement and complement of this whole expression which gives me X complement, complement of that dot, complement of Y complement by DeMorgan's law which in turn gives me X.Y.

Now this shows me that the X.Y or X and Y can be written in terms of plus and complement. Therefore, if we consider the disjunctive normal form of a function and then just replace all the products by using the relation X.Y=X complement, Y complement and complement of that, then we can write the disjunctive normal form purely in terms of plus and complementation. And therefore, we can say that any Boolean function can be written in terms of the Boolean variables and the operations from the set plus and complementation.

So I write like this, any Boolean function from BN to B can be written as an expression consisting of N Boolean variables and operations from the set plus N complementation. Now we consider another set which is dot and complementation. Here also we see that if X and Y are Boolean variables we can take X complement dot Y complement and complement of that whole expression and by DeMorgan's law this becomes X complement, complement plus Y complement, complement this is by DeMorgan's law which is equal to X+Y.

Thus, in working the same argument we can say that a Boolean function can be written as a Boolean expression consisting of the Boolean variables and the operations dot and complement.

(Refer Slide Time: 09:56)



In general if we have a set of operations let us say S, we will call this set S functionally complete or universal if any Boolean function can be written as expressions consisting of the Boolean variables and the operations from the set S. So let us write it down formally S is said to be functionally complete if any Boolean function can be written as a Boolean expression consisting of the Boolean variables and operations belonging to the set S.

Now we ask a question, can there be a singleton set which is functionally complete and surprisingly the answer is yes. In fact we will check two singleton sets which are functionally complete. The first one consists of an operation which is called NAND. So I write NAND inside third bracket to designate that we are considering a set consisting of a single operation called NAND. And this NAND is sometimes denoted by a single stroke, a vertical stroke.

NAND is defined as X NAND Y = X complement + Y complement. Now in order to show that NAND is functionally complete we have to show that we can express complementation OR and AND all three in terms of NAND. In order to check complementation we see that X NAND X gives me X complement + X complement which is of course X complement. Then we see that if we consider X NAND X and then NAND of Y NAND Y, then we get X complement NAND Y complement which in turn by definition of NAND is X complement, complement of that plus complement of Y complement which is X+Y.

Therefore, we see that we can construct the complementation operation of Boolean variable by NAND which is X complement is equal to X NAND X and the OR operation which is X NAND

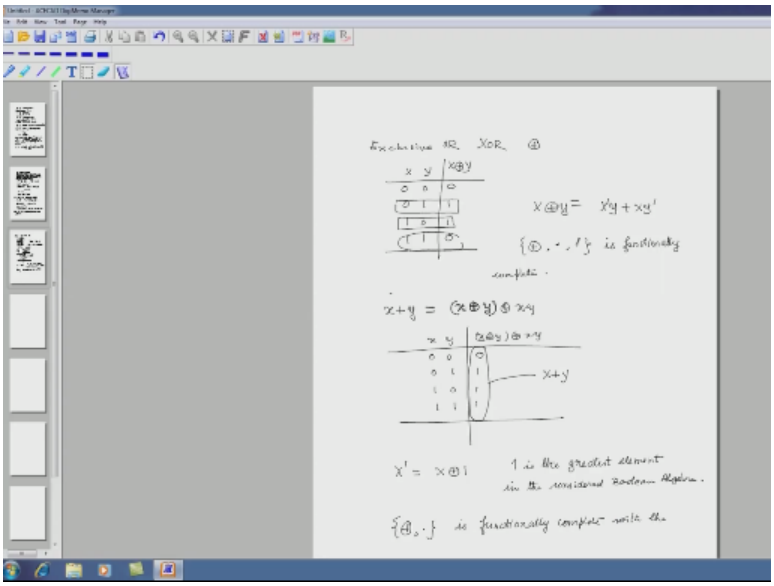
X NAND, Y NAND Y. At this point we realize that we do not have to show that we can write AND operation in terms of NAND, because we have already proved that the set plus AND complement is functionally complete, we have p roved that.

Therefore, we can write since the set plus and complement is functionally complete. The set consisting the single operation NAND is also functionally complete. Now we move to another operation which is called NOR and this is given by the symbol of an arrow directed downward, and NOR is defined as X NOR Y = X complement . Y complement. Now let us check whether NOR is functionally complete or not.

We consider juts like before X NOR X which is X complement and X complement which is of course X complement. Then we construct X NOR X NOR of Y NOR Y = X complement NOR Y complement = X complement, complement of that and complement of Y complement which is XY. Thus, again we see that the operations dot and complement can be generated from NOR. Therefore, we can write as before since dot and complement is functionally complete.

So is the singleton set NOR and that is all. So the basic strategy of proving such results is that when we are given a set of operations and ask to s how whether that set is functionally complete or not. Somehow, we should try to write plus and complement or dot and complement in terms of those operations. If we can do that, then it is direct that the original set is functionally complete.

(Refer Slide Time: 21:08)



At this point we will introduce another operation which is called exclusive OR or simply XOR. Now I will write the table corresponding to XOR, suppose we are considering two variables X and Y, the possible values of these two variables are 00, 01, 10, 11 and X or Y, by the way this is the symbol corresponding to XOR is 0, when both X and Y are 0s, it is 1 when X is 0 and Y is 1, it is again 1 when X is 1 and Y is 0.

But unlike OR XOR is 0 when both X and Y are 1s. This is an operation which is used in several applications or Boolean functions. Now we can consider this operation as a Boolean function itself and try to write down the disjunctive normal form or DNF of this operation. We want to do that, then we will find that the DNF is X, X or Y is equal to we have to only consider this row and this one.

So here we have got X complement Y, and for this row we will have Y complement, and this is the expression for XOR in terms of complement AND and OR. It can be proved fairly easily that OR and AND complementation together is also functionally complete. XOR and AND complementation is functionally complete. This is because I can write X+Y as, so X+Y=X OR Y OR XY.

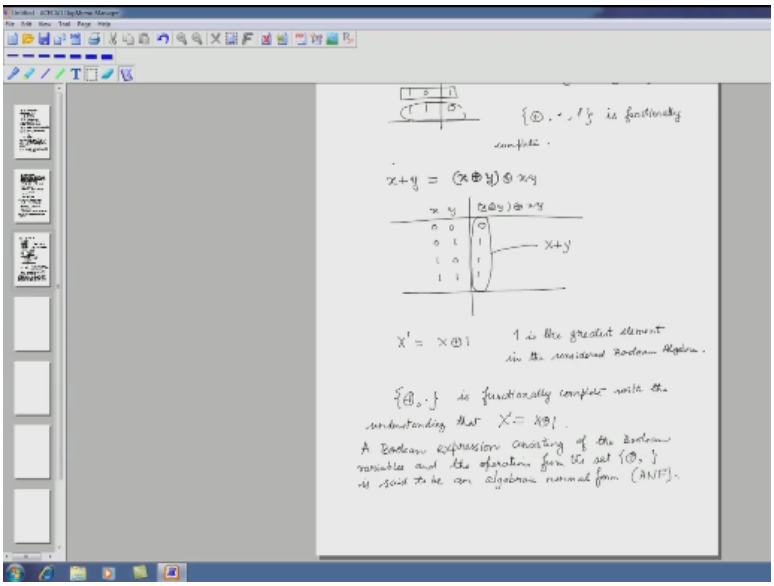
If you want to check that we have to check the truth table of this function X OR Y OR X.Y. Let us consider all the input patterns for this expression and then evaluate. So when X and Y, both are 0s, we have the output as 0, when Y is 1 we see that X OR Y gives me 1, X OR X.Y gives me 0, so I have got 1, when X is 1 and Y is 0, then I will also have 1. And now when X and Y both are 1s, then X OR Y gives me 1, I am sorry, X OR Y gives me 0, because of this.

But X.Y gives me 1, so all together I have got 0 XOR 1 which gives me 1. And this pattern in the extreme right hand column is exactly the pattern corresponding to X+Y or X OR Y. Therefore, we see that by combining XOR then the AND complementation we can generate OR. And since OR and AND complementation is functionally complete, XOR and AND complementation is also functionally complete.

At this point we can also do something even further, we can write the complement of X as X XOR 1, where 1 is the greatest element in the Boolean algebra that we are considering. 1 is the greatest element in the considered Boolean algebra. So what happens then, this means that I can replace even complement, if I denote the greatest element by 1 and 0, and of course their

existence is guaranteed by the basic definition of Boolean algebra. Therefore, we can even say that XOR and dot is functionally complete.

(Refer Slide Time: 28:50)

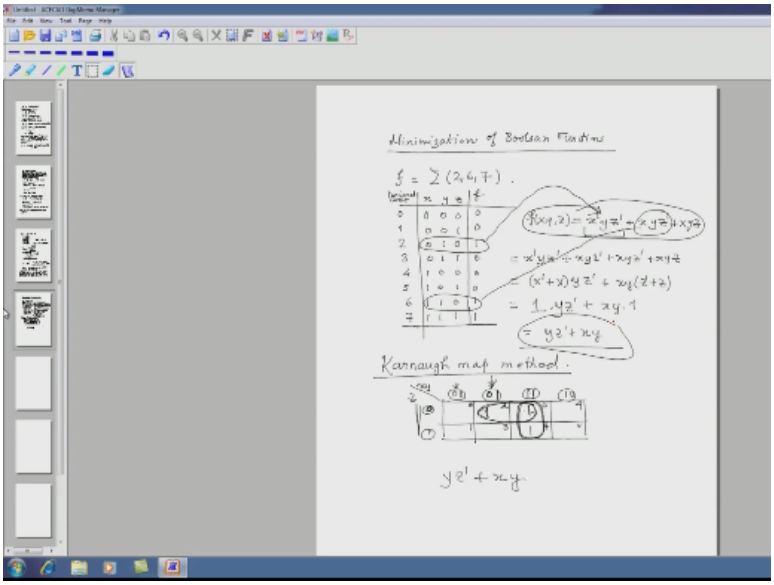


With the understanding that further we can write a Boolean function by using simply XOR and dot, and we can see that because we can always replace the plus by XOR and dot and what we can do is that every complement we can write as a sum with the original and the 1, by sum I mean XOR. Now if we do that then we will be getting a Boolean expression written in terms of the Boolean variables XOR and dot or AND whatever we call it.

A Boolean expression of this type is called an algebraic normal form or ANF. A Boolean expression consisting of the Boolean variables and the operations from the set consisting of XOR and dot is said to be an algebraic normal form, in short it is written as ANF. And of course, I need

not explain further that any Boolean function can be written in terms of an algebraic normal form.

(Refer Slide Time: 32:21)



Next we move on to another topic which is extremely important in the context of Boolean functions, and that is called minimization of Boolean functions. By this we will mean that suppose we know that a Boolean function can be written in terms of the Boolean variables and some operations. We would like to write the same function by using minimum number of variables and in such a way that the operations are used minimum number of times.

Now in general this is a very vast problem and a current area of research. However, we can deal with small Boolean functions and consider its expression in terms of OR and NOT and try to reduce it to something smaller. Now we start with an expression on three variables. Suppose we have a function F which is given by (2, 6, 7) this is a form that we have already discussed before and let us suppose the variables are designated as (x, y, z).

Therefore, we first write the truth table (x, y, z) are the variables corresponding to the coordinates of the Boolean algebra B3 and we write the possible values of (x, y, z). So the first

consist of 000, the second 001, then 010, and the fourth one is 011, and we move further down to 100, 101, 110, and 111. We now write down the corresponding decimal code, the first entry is 0, the second is 1, then 2, then 3, then 4, then 5, 6 and lastly 7.

My function says (2, 6, 7) that means according to our convention the function is 1 at 2, then it is 1 at 6, and then it is 1 at 7. And at the rest of the places the functional value is 0s, we have got, we can write F over here. And now we are in a position to write down the algebraic normal, I am sorry the disjunctive normal form of this function. So I am going to write down (x, y, z) as X complement Y, Z complement + XY Z complement and + XY and Z.

Now the question occurs is that can we write this in terms of something which is shorter than this. What we can do here is X complement Y Z complement and split this the second term as XY Z complement + XYZ complement, because we know that if we take OR of the same variable we get back the same variable and (x, y, z). And then if we observe here we see that you can use distributive law to write X complement + X within the bracket.

Then followed by Y Z complement + and here again I can write XY and Z complement Z. Now we recall the results that we derived extremely thoroughly in a lecture previous to this one where we have proved that X+X complement gives me 1 and Z+Z complement gives me 1, well that is the definition of complement. So I have got 1 Y Z complement and XY1, so I get YZ complement + X and Y.

The final term is interesting because this gives the same function, but this uses lesser number of variables at each term and also of course, lesser number of operations. The question is that can they do it systematically. There is a tabular form that is used for functions with smaller number of variables which is the case here that I would like to discuss in this lecture. This is called Karnaugh map method.

Here we will consider a table which is best understood by using examples, I write XY over here and Z below it. So I will keep on writing all possible values of XY in these positions and all possible values of Z in these positions. I will write 00, and then write 01, but instead of writing 10, I will write 11 and then lastly I will write 10. There is a pattern here that is when we move from one cell to another there is only one change in the variable values.

If I change from 01 to 10 that means 0 will change to 1 and 1 will change to 0 which is not acceptable to us. We will change from 01 to 11 and then the last one we will make 0, so I will have 10. And in case of the rows I will just write 0 and 1. Now see that we can write the decimal codes in a corner of each cell. So this cell corresponds to 000 which is 0, then this cell corresponds to 010, now 010 is 1, then I am sorry, 010 is not 1, 010 is 2.

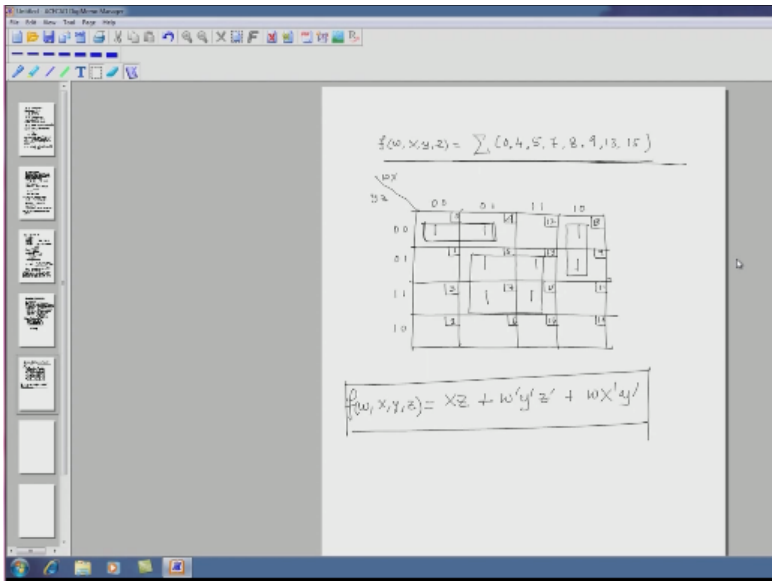
And then I have got 110 which is 6 and then I have got 100 which is 4, so I am just writing the decimal codes over here, and I have got 001, 001 is 1, 011 which is 3, 111 which is 7, and 101 which is 5. Now I would look at the disjunctive normal form of the function, the first entry of the disjunctive normal form is X complement Y Z complement. Now when I encounter X complement, then I will consider X=0.

So I come to either this cell or this cell, but I have Y along with that, but Y corresponds to 1, so I come to this cell and then, I have got Z complement, Z complement means Z=0. In fact I could look into the entry 010 corresponding to which I have the mean term X complement Y Z complement. So 010, here I put a 1, similarly if I look at the other mean term 110 which corresponds to this mean term 110, I will put a 1 over here, and last one is 111, I put a 1 over here.

After this I will join the adjacent cells like this by some rectangles. Now if you consider this two cells you will find that the value of X changes, value of Y remains in tact and value of Z remains in tact, only the value of X changes, so we cut down X. So what we do is that corresponding to this too much cell I write Y Z complement. Then I see that I can merge also these two cells and when I do this the value of Z changes from 0 to 1, value of X and Y remains as it is.

So I will just write +X and Y removing the variable value which changes. And this happens to be the reduced expression. What we see is that this is exactly what we got by using algebra. The problem of this method is that when the variables start increasing then this whole system can become very, very complicated, but we can do the same map method for functions having four variables which is the last topic that we will discuss in this lecture.

(Refer Slide Time: 47:15)



Now let us consider a function with four variables F(w, x, y and z) which is given by (0, 4, 5, 7, 8, 9, 13, and 15). As I have already told that Karnaugh map method is best understood by examples. Now we have a function at hand, we will consider a table in this case and like the previous table we will have two variables leveling the columns as before, but two more variables leveling the rows.

So I draw the table like this, I pick first two variables (w and x) to level the columns and the last two variables (y and z) to level the rows. Well, there will be four possible columns and four rows, I make the greed and here the values will be 00, 01, 11, this is a note of caution and 10. In the rows we will have 00, 01, 11, and 10. So when we are reading of the cells, the upper left hand corner cell we will read 00, 00 which corresponds to 0, I can write over here.

The next one is 01, 00 which corresponds to 4 right. Then we have 1100 this corresponds to 12 and here we have 1000 which corresponds to 8. Now then we have 0001 which corresponds to 1, and 0101 which corresponds to 5, and similarly 1101 which corresponds to 13, and lastly corresponds to 9. Here again, the next one is 0011 it corresponds to 3, then we have 7, then we have 15, and we have 11.

And the fourth row will be 2, 6, 14, and lastly 10, 1010 which is 10. Now we can read off from the formula and write 1 in appropriate places in 0 we have got 1, in 4 we have got 1, and then in 5 we have got 1, in 7 we have got 1, in 8 we have got 1, in 9 also we have got 1, then lastly 13

and 15. Now our goal will be to cover these 1s by rectangles in such a way that we can cover all of them with minimal number of rectangles.

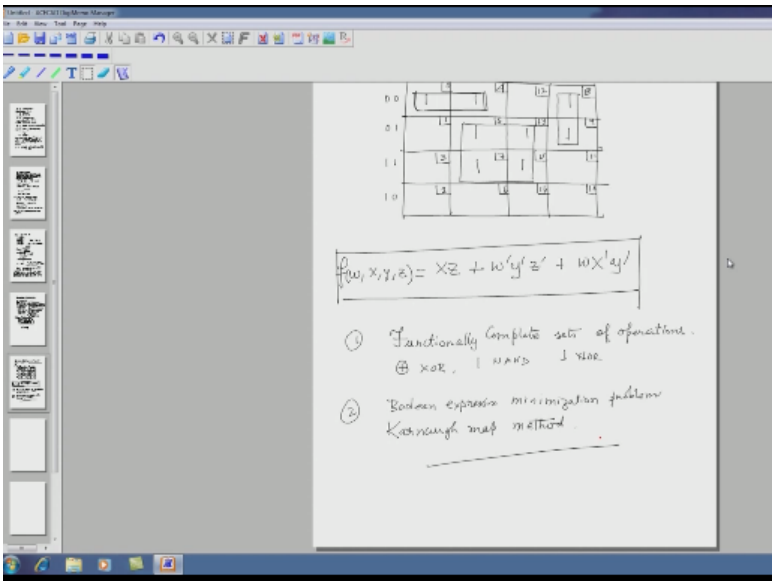
If we try to do that we will see that we have a big rectangle which is in fact a square covering these 4 1s and then 1 like this and another like this. Now we first check the middle rectangle or the square, an d try to see that when I move around how many variables are changing to see if I move around W is changing and X is keeping the value 1, therefore I will put X over here. And when I move in the vertical direction I see that Y is changing but Z is remaining in the same value 1.

Therefore, Y will be deleted and I will write Z. When I move to this rectangle in the left hand corner I see that X varies, but Y, Z and W remains same, but all at the state 0, so I will put W complement, Y complement and Z complement. And similarly for the last one we see that Z changes, but W is in the state 1, X is in the state 0, therefore X complement Z changes therefore I remove Z, and Y does not change betterments in the state 0, therefore I just put Y complement.

And that is going to be the reduced expression of the function FW, x, y, z. Now what we can do is that we can write the disjunctive normal form of the function F that I have stated over here, and then try to do algebraic manipulation, and then also we will come to the same expression as this one. This is an example of minimization of Boolean expressions by using the Karnaugh map method.

So in this lecture we have started with the idea of functionally complete sets of operations. In this we have checked apart from the usual operations, three new operations one, is called XOR, another is NAND, and the other one which is NOR.

(Refer Slide Time: 56:36)



Secondly we have briefly discussed the Boolean expression minimization problem by using the Karnaugh map method. That is for today, thank you.

Educational Technology Cell
Indian Institute of Technology Roorkee

Production for NPTEL
Ministry of Human Resource Development
Government of India

For Further Details **Contact**

Coordinate, Educational Technology Cell
Indian Institute of Technology Roorkee
Roorkee-247667
E Mail: etcell@iitr.ernet.in, etcell.iitrke@gmail.com
Website: www.nptel.iim.ac.in

Acknowledgement
Prof. Pradipta Banerji
Director, IIT Roorkee

Subject Expert & Script
Dr. Sugata Gangopadhyay
Dept of Mathematics
IIT Roorkee

Production Team
Neetesh Kumar
Jitender Kumar

Pankaj Saini
Meenakshi Chauhan

Camera
Sarath Koovery
Younus Salim

Online Editing
Jithin.k

Graphics
Binoy.V.P

NPTEL Coordinator
Prof.Bikash Mohanty

An Educational Technology Cell
IIT Roorkee Production