Advanced Operations Research

Prof. G. Srinivasan

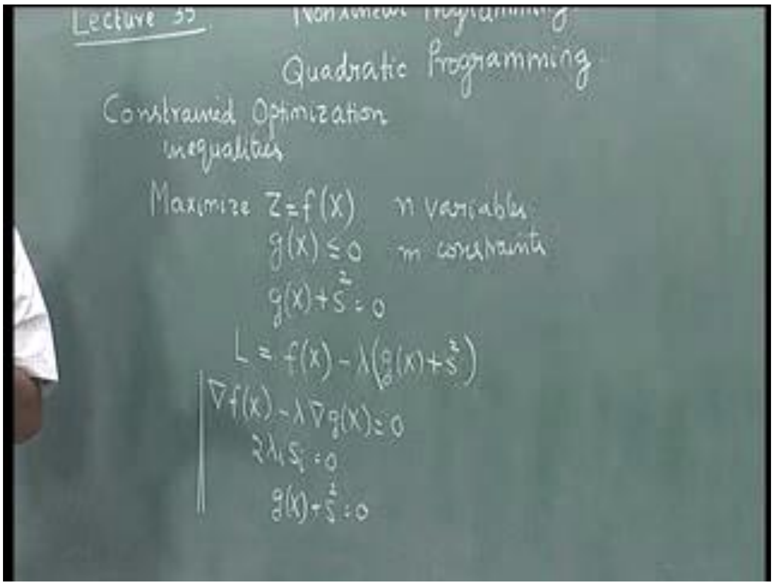
Department of Management Studies

Indian Institute of Technology, Madras

Lecture - 35
Quadratic Programming

In this lecture, we continue our discussion on nonlinear programming. We mentioned in the earlier lecture that the focus is primarily on quadratic programming but in order to understand the principles we need to look at the basics of nonlinear programming.

(Refer Slide Time: 00:36)



In the previous lecture, we were looking at a constrained optimization problem with inequalities as constraints. The constrained optimization problem with inequalities would be of the form, maximize Z equal to f of X subject to g of X less than or equal to 0. We have already mentioned, particularly in nonlinear optimization, we do not have an explicit condition that X has to be greater than or equal to 0. If there is a restriction that X has to be a greater than equal to 0, then it has to be included as a constraint in the problem. Therefore, all these constrains can be written in the form g of X less than or equal to 0.

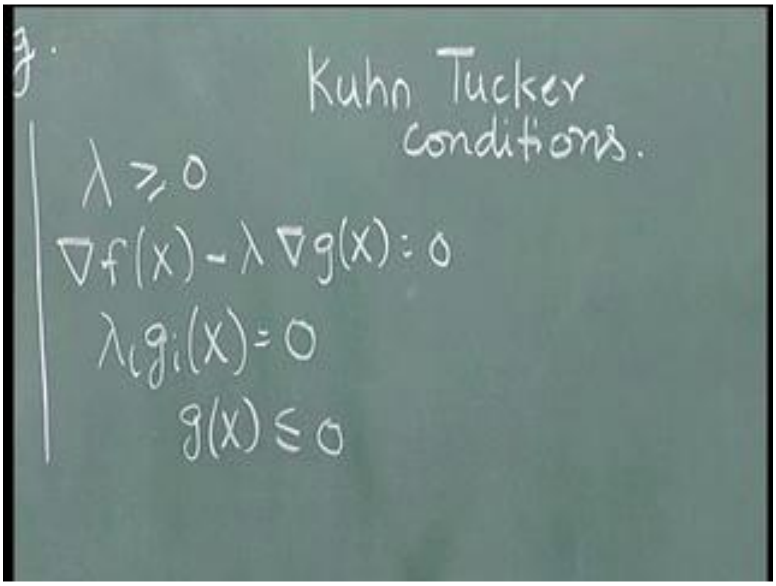
We take the case where we wish to maximize and we derived the conditions for it. Every minimization problem can be written suitably as a maximization problem, with suitable changes in the objective function. The guiding principle is first to convert this inequality into an equation. Then, using the method of Lagrangian multipliers and take it into the objective functions. This inequality is written as an equation by using g of X plus S square is equal to 0. The S square comes, because we do not have an explicit restriction that the S should be greater than or equal to 0. If it were linear programming, we would have written this as g of X plus S equal to 0 and said S greater than equal to 0. Here, we will write it as g of X plus S square equal to 0, so that, S can be positive or negative or 0, S can be less than or equal to greater than or equal to or equation. But S square will be greater than or equal to 0, so that g of X plus S square is equal to 0. Remember that, this g of x plus S square equal to 0 is a set of constraints. Similarly, Z equal to f of X is a function that involves more than one variable. We would have X_1 to X_n as variables. In general, we say n variables and m constraints for this problem.

Now we take this to the objective function by introducing Lagrangian multipliers, as many multipliers as a number of constraints. This is also called dualizing the constraint and bringing into the objective function. We now define the Lagrangian function L, which will be f of X and because it is a maximization problem you would put it as minus lambda into g of X plus S square. Partial differentiation of this with respect X, lambda and S would give us the conditions for the optimum.

Therefore there are three sets of variables: X, which are the given decision variables in the problem; S which are slack variables so we will have as many slack variables as the number of constraints because each constraint is written as g of X plus S_i square equal to 0. Then we take it into the objective function by introducing lambdas so we will have as many Lagrangian multipliers as the number of constraints.

We have this L which is the Lagrangian function. Now, dow L by dow X equal to 0 would give us del f of X, minus lambda del g of X equal to 0, dow L by dow S which is a slack variable would give us 2 lambda_i S_i equal to 0. Here we differentiate with respect to S so this becomes 2 times lambda S equal to 0, so 2 lambda_i S_i equal to 0. Differentiating with respect to lambda would give us g of X plus S square equal to 0. We modify these three conditions, we also use lambda greater than or equal to 0 for this, under the principle that, when we brought it into the Lagrangian, we relax the constraint. Therefore, the restricted problem, a maximization problem with the addition of the constraint will only have its objective function value reduced so lambda will be greater than or equal to 0.

(Refer Slide Time: 05:46)



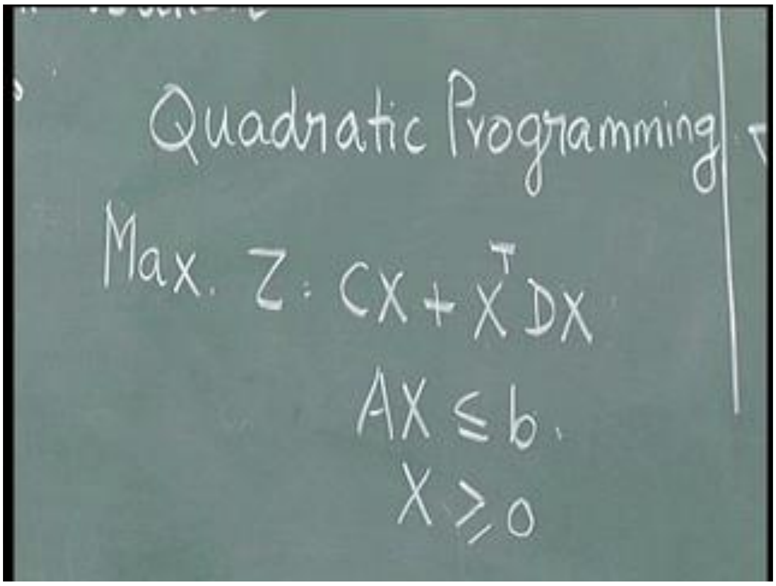
The conditions that we have are lambda greater than or equal to 0. First derivative del of X minus lambda del g of X equal to 0, is exactly written there, then, lambda_i g_i of X equal to 0. This comes from 2 times lambda_i S_i equal to 0. This is g of X plus S square equal to 0, so 2 lambda_i S_i equal to 0. This would give us either lambda_i equal to 0 or S_i equal to 0.

Lambda_i equal to 0, we can simply write lambda_i g_i of X equal to 0. When S is equal to 0, then we get g of X equal to 0. We can write it as lambda_i g_i of X equal to 0. This form actually captures 2 lambda_i S_i equal to 0, except for the fact that, we have eliminated the variable which is the slack variable that we actually added into this problem. Then, g of X plus S square equal to 0, automatically become g of X less than or equal to 0. The reason for writing it in this form is the original problem did not have the inequalities. The slack variable was added to convert the inequality into an equation and then we used the method of multipliers. Therefore, at the end this is to be written in the form, which does not have the S square. Now, use of lambda is inevitable, because we have to convert this inequality into an equation. We eliminate the variable S that we have introduced here and write it in this form. Where we write it in terms of lambda and in terms of X, we leave out the S as such. This form is called the Kuhn Tucker conditions, also called Karush Kuhn Tucker conditions depending on the contribution of the three people.

This provides us with a general set of equations and inequalities to solve any nonlinear optimization problem, with multiple variables, particularly, when the constraints are inequalities. This would give us a set of inequalities and equations. For example, you have a set of inequalities here, we have this is an explicit lambda greater than or equal to 0, you have a set of equations here and these can be linear or nonlinear or can have any power depending on f of X, again g of X less than equal to 0, can have product form or quadratic form of the variables, depending on g of X, for general non linear programming problem. Nevertheless, once we write these conditions and write the resulting equations and inequalities, all we need is to solve this using suitable methods to get the correct value of X and lambda.

With this background we move to the central theme of this lecture series which is the quadratic programming. Central theme in the sense, that aspect of nonlinear programming that we are going to cover in this lecture series is quadratic programming. We use these Kuhn Tucker conditions and then write down the corresponding Kuhn Tucker equations for a quadratic programming problem. Then, we show that the resultant system can actually be solved as a linear programming problem, with some additional restrictions. To that extent, we are going to do two things: one is we are going to drive the specific Kuhn Tucker conditions for a quadratic programming problem; in part two, we are going to trying to show that this resultant system can be solved using linear programming. Therefore, this will be shown as a linear programming application.

(Refer Slide Time: 10:00)



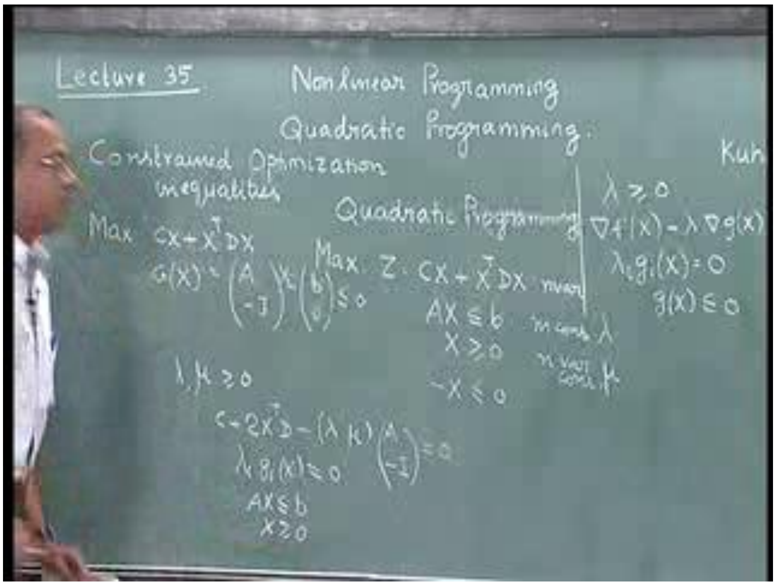
The quadratic programming problem, can be written as maximize or minimize Z equal to CX plus X transpose DX subject to, AX less than or equal to b, X greater than or equal to 0. We note a couple of things, X equal to X_1, X_2, to X_n, are a set of decision variables. The objective function is quadratic that happens, because of the form X transpose DX, which gives us the quadratic form. There is also a linear portion in the objective function, which is CX. There can be a constant, which can be ignored temporally. Z equals to CX plus X transpose DX, represents the quadratic objective function.

Remember that the constraints are linear, so it is written in the form AX less than or equal to b. All these constraints are linear, if the constraint is of the greater than or equal to type, it can automatically be written in this form. This can also handle equations and there is no difficulty about it. What is all the more important is X, is explicitly stated to be greater than or equal to 0. In general, nonlinear programming, whenever we have X greater than or equal to 0, it is treated as a constraint. In quadratic programming, we explicitly have all X greater than or equal to 0 which is actually pulled out of the resultant system. This is a general form of a quadratic programming problem. This has several applications and one of the common applications that, one can think of is actually to look at a portfolio problem.

(Refer Slide Time: 11:54)

If someone has an investment, say portfolio of shares and other investments, ordinarily, there are two things associated with this. One is called the return and the other is called the risk. Here, the objectively decision variable will be, how much proportion do I spend on each of these securities that, form part of the portfolio. Such that, there is some expected return, expected return is greater than or equal to some R and minimize the risk, subject to, of course, the condition that sigma X_i equal to 1. Sum of the proportions of the investments should add up to 1. This problem is modeled as a quadratic programming problem, because this risk is seen as a variance or a covariance of the returns and therefore it is a quadratic form. The objective function is quadratic, subject to 2 constraints: one is a greater than or equal to constraint; the other is the sum of proportions are equal to 1; third is an explicit condition that, X greater than or equal to 0, because we do not want negative proportions of investment. This is an application of quadratic programming to a real life situation.

(Refer Slide Time: 13:18)



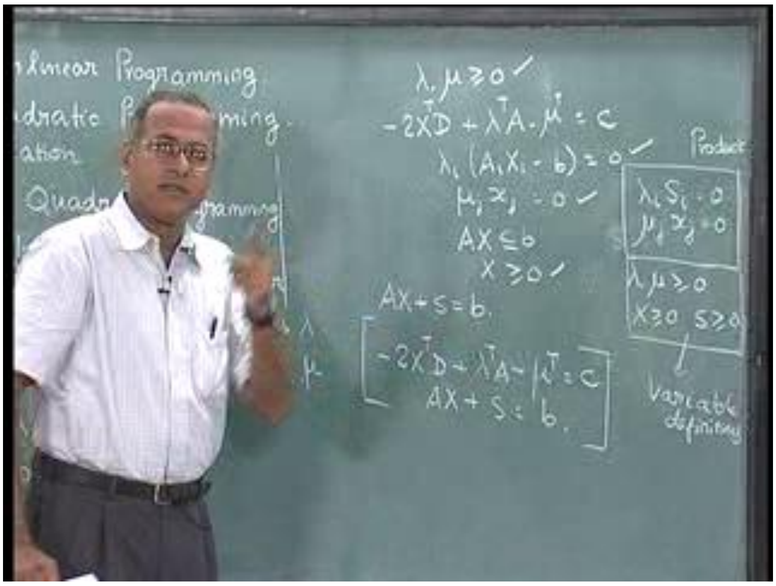
If we have a quadratic programming problem like this, then we write the Kuhn Tucker conditions here, for maximization. The first thing that, we will have is lambda greater than or equal to 0, there are as many Lagrangian multipliers lambda here. In fact before we do that we also try and write it. The quadratic programming problem has an explicit condition that X is greater than or equal to 0, but the Kuhn Tucker conditions are derived for a general nonlinear programming problem. Therefore in order to write down the Kuhn Tucker conditions, we actually write this also as constraint and then write the Kuhn Tucker conditions. When we do that, we now get rid of this form maximize CX plus X transpose DX, subject to the condition we have some G of X is equal to A minus I X minus b 0 is less than or equal to 0. The Kuhn Tucker conditions were derived for G of X less than or equal to 0. Therefore, AX minus b automatically qualifies, because it is of the less than or equal to type, X greater than or equal to 0 is written as minus X less than or equal to 0. Therefore, these two put together now forms the constrained set G of X. Therefore, it is written as A X minus b less than or equal to 0 minus X less than or equal to 0. This is the general set here.

Again, we have n variables m given constraints and again n variables here therefore n constraints. Because this X is greater than or equal to 0 are treated now as an explicit constraint. Therefore now when we start writing the Kuhn Tucker conditions for this, we introduce Lagrangian multipliers lambda for this, we also introduce Lagrangian multipliers mu for this. There are as many lambdas as the number of constraints and there are as many mus as the number of variables and we have as many X as the number of variables so we have n X_1 to X_n, mu_1 to mu_n, $lambda_1$ to $lambda_m$, this is what we introduce here. Then we write the Kuhn Tucker conditions for this, so this would mean, both lambda mu greater than or equal to 0 because this represents all the Lagrangian multipliers, both lambda mu greater than or equal to 0.

Hence, del f X minus lambda del g X would give partial derivative of this which is C. Then this del f X will be, C plus 2X transpose D derivative of this del of X minus lambda del g X equal to 0, so this will become minus lambda mu because the set of Lagrangian multipliers and del g X will be AX on differentiation so you get A minus I equal to 0. Thus, del f X minus lambda dell g X equal to 0 would give this equation for us. Then, $lambda_i$ g_i of X equal to 0 would give us $lambda_i$ g_i of X equal to 0.

Then we have g of X less than or equal to 0 which is given by AX less than or equal to b and X greater than or equal to 0 because g of X less than equal to 0 is the same set of constraints that repeat here so that is given by AX less than equal to b X greater than equal to 0. This we are explicitly written exactly it this way and then we will make some modifications to this.

(Refer Slide Time: 18:14)



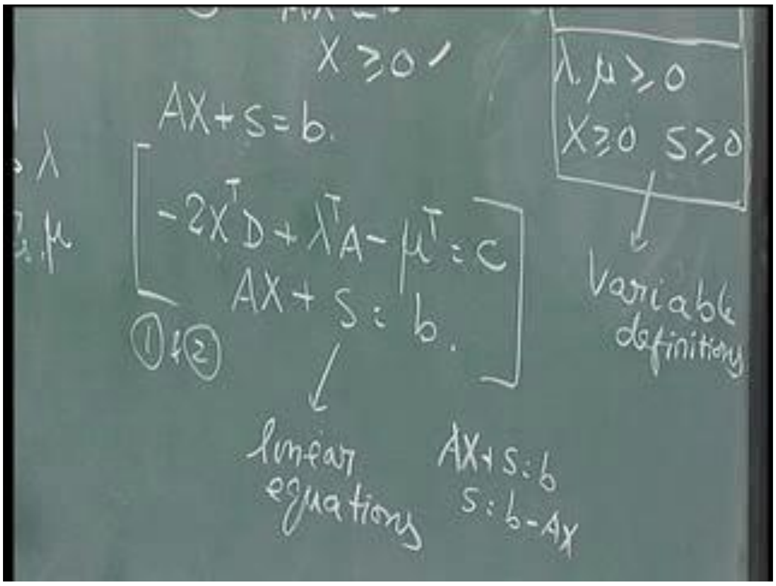
So, we rewrite this and then make some modifications. Hence Lambda mu greater than or equal to 0 which comes from here. Now this is written as; we keep this C on this side and take the rest of them on the other side because this involves a variable X, these involve variables lambda and mu, now C is the only one that is the constant. Hence we keep C on this side and take the rest of them on the other side then we would get minus 2X transpose D plus lambda transpose A minus mu transpose equal to C. This lambda A goes on this side, minus mu into minus I would give you a plus here, and when this goes to the other side you get a minus. Therefore this equation is written that way.

This one, lambda$_i$ g$_i$ of X equal to 0, is now written like this. If we have for this set, please remember, that this is for a general Kuhn Tucker form. This has to be written for both lambda and mu that we talk about. This will be written as lambda times, each one of these constraints will become A_i X_i minus b. This will be written as lambda$_i$ into A_i X_i minus b equal to 0. This will be written as mu into X is equal to 0. This will be written as mu$_j$ X_j equal to 0 and then we have AX less than or equal to b. We have X greater than or equal to 0. I repeat again, this one is has been written from the Kuhn Tucker form. This has to be written for both set of constraints, this constraint as well as this constraint. When we have this lambda$_i$, actually represents both lambda and mu, lambda for this set and mu for this set. This will become some lambda$_i$ into A_i X_i minus b for the ith constraint here; this will become mu into X equal to 0.

What we do here, is we now start introducing; this is a Kuhn Tucker condition that is written. Since, we have AX less than or equal to b, we can always introduce a slack variable S, such that, AX equal to b or AX plus S equal to b. This is written as AX plus S equal to b. Therefore, this becomes lambda$_i$ S_i equal to 0. This becomes lambda$_i$ S_i equal to 0. Here we have a mu$_j$ X_j equal to 0 and because AX is less than or equal to b and X greater than or equal to 0, we can always write this as AX plus S_i equal to b or AX plus S equal to b. The Kuhn Tucker conditions will become, lambda mu greater than or equal to 0, comes from here. X greater than or equal to 0 is already here, S greater than or equal to 0 has happened, because we have introduced this S, such that AX plus S equal to b. We have now used this, we have now used this, we have now used this, and we have now used this. The Kuhn Tucker conditions, now reduced to minus 2X transpose D plus lambda transpose A minus mu transpose equal to C and we have AX plus S equal to b. Kuhn Tucker conditions reduce to these two things, plus these two sets of things, plus we have these. Kuhn Tucker conditions, now reduces to this, plus this, plus this. We will look at all these carefully, now we realise that, these are only variable definitions. We are okay with these being greater than or equal to 0. Lambda$_i$ S_i and mu$_j$ X_j are not linear; they represent a product form here, so these are some product form. Remember, how we got these two, now mu$_j$ X_j equal to 0, is got automatically from the Kuhn Tucker condition, which comes from this form. This was the original Kuhn Tucker equation, for every constraint there are two sets of constraints, this and this. This is written for both lambda and mu. Therefore, for lambda we got lambda$_i$ into A_i X_i minus b equal to 0. For mu we got mu$_j$ X_j equal to 0. We also have from here AX less than or equal to b X greater than or equal to 0. So we simply introduced the slack variable S.

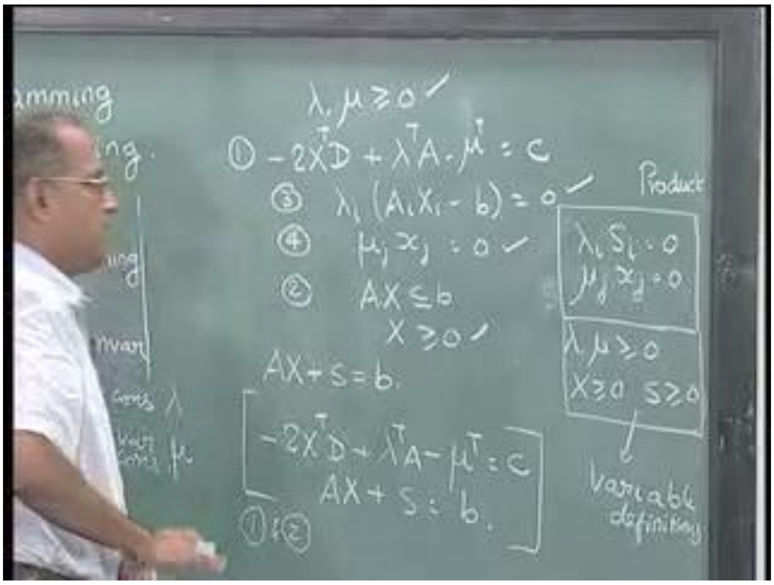
Please note, the difference that this slack variable S, I am talking about here, was actually different from the S that was introduced to get the Kuhn Tucker conditions. We have first applied the Kuhn Tucker conditions. After applying the Kuhn Tucker conditions, since we get AX less than or equal to b and we know that X is greater than or equal to 0, we could always write this as AX plus S equal to b and define a new slack variable S greater than or equal to 0. Therefore, in quadratic programming we will have slack variable S greater than or equal to 0.

(Refer Slide Time: 24:12)



After writing this, we realize that we can write it actually in three different pieces or three different sets. The first set are these equations which are got from this and this, this modified to AX plus S equal to b. So, these two would give us, I call this as 1 and call this as 2, so this is 1 and 2. Then, we get into lambda$_i$ S_i mu$_j$ X_j equal to 0. I am going to call this as 3 and this as 4 (Refer Slide Time: 24: 46) because S is defined as AX plus S equal to b, since this is AX plus S equal to b, S equal to b minus AX, this would give us minus $lambda_i$ S_i equal to 0 from which $lambda_i$ S_i equal to 0. hence these two (Refer Slide Time: 25:15) comes from 3 and 4. The rest of them are here lambda mu greater than or equal to 0, X greater than or equal to 0 and S greater than or equal to 0.

(Refer Slide Time: 25:20)



Therefore the Kuhn Tucker conditions reduce to solving these three sets. We also said that this is like a product form; this interestingly, is a set of linear equations. Linear equations, because from the given 1, Z is the only one that is quadratic, therefore, the partial derivative of Z, with respect to X, will give us a linear set of equations, which are here. AX less than or equal to b represents a set of linear inequalities, therefore, AX plus S equal to b would represent a set of linear equations. All we need to do now is to solve for 1 2 3 4 5 6, all these six things put together.

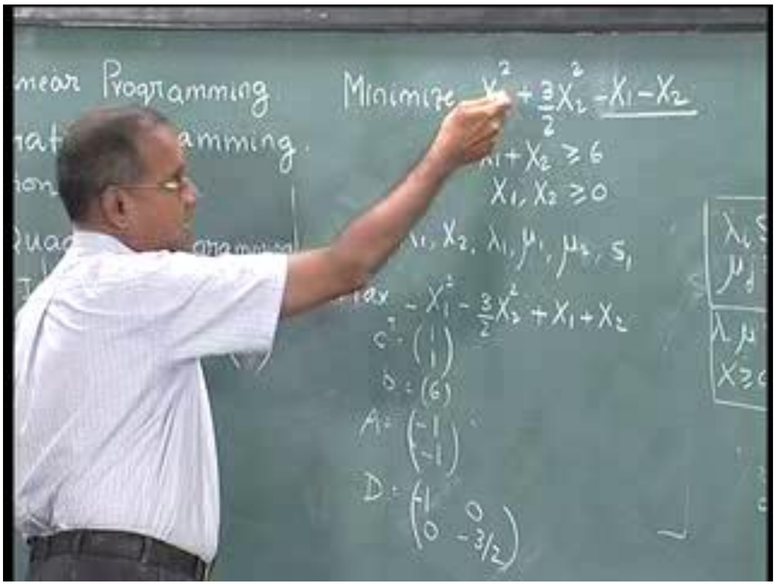
Each one, for example, this is a partial derivative with respect to X. Therefore, this will have as many equations as the number of variables in the problem. This is as many equations as the number of constraints in the problem. This product form is $lambda_i$ S_i, both depend on the number of constraints mu_j X_j, both depend on the number of variables and lambda mu X are all variable. We have two times m plus n number of variables, where n is the actual number of decision variables, because both mu and X are from 1 to n. Lambda and S are from 1 to m, the number of constraints. This will have the number of variables plus number of constraints.

This will be m plus n. Here, we have m product, n product. Again m plus n sets, here it is m plus n, plus m plus n, so two times m plus n. So, finding out X, lambda, mu, S, such that it satisfies all this, would give us the maximum or the minimum, assuming that the maximum or the minimum exists and it is unique here. We are not at the moment stressing on the second derivative to actually prove that it is a maximum or minimum. We restrict ourselves to solving this problem, understanding that, whatever we get by the solution of this, by the unique solution of this will be the corresponding maximum or minimum.

It boils down to solving these and how do we solve this. (Refer Slide Time: 27:42) In order to solve this, these can be written in a slightly different way, minus 2D A -I 0, A 0 0 I into X lambda mu S equal to c b. The first can be written as minus 2 D x, which is this form, plus lambda A, which is this form, multiplying by transpose, we get minus mu, which is here equal to C, which is here. The other one is written as AX plus S equal to b, so AX plus S equal to b. This is the nice representation of this set of linear equations, which are given here, plus of course, we have this and we have this. If we momentarily leave out this, then add, for example, X, lambda, mu, S greater than equal to 0, this is like solving a set of equation subject to the condition that the variables X, lambda, mu, and S are greater than or equal to 0. These equations are linear equations.

We also have learnt much earlier in the lecture series that, solving a set of equations subject to the condition that all variables greater or equal to 0 can be modeled as a linear programming problem. This quadratic programming problem, whose optimality conditions reduce or after applying the Kuhn Tucker conditions, reduces to a set of linear equations, a set of non-negative restrictions on the variables and a set of product relationship among these variables, if we temporarily leave out this set of equations or the product form, we get a set of linear equation, subject to this, it can be solved as a linear programming problem. In some way, linear programming can be used provided, the linear programming iterations are suitably carried out to make sure that this is not violated. Essentially, we relax this solve as a linear programming and make sure that at every iteration these constraints are satisfied, so that, the solution of the linear programming with the additional restriction of these is obviously optimal to the system, where we try to solve this, plus this greater than equal to 0. All these we try and show through a numeric examples, so that, all these computations come out quite clearly.

(Refer Slide Time: 31:13)

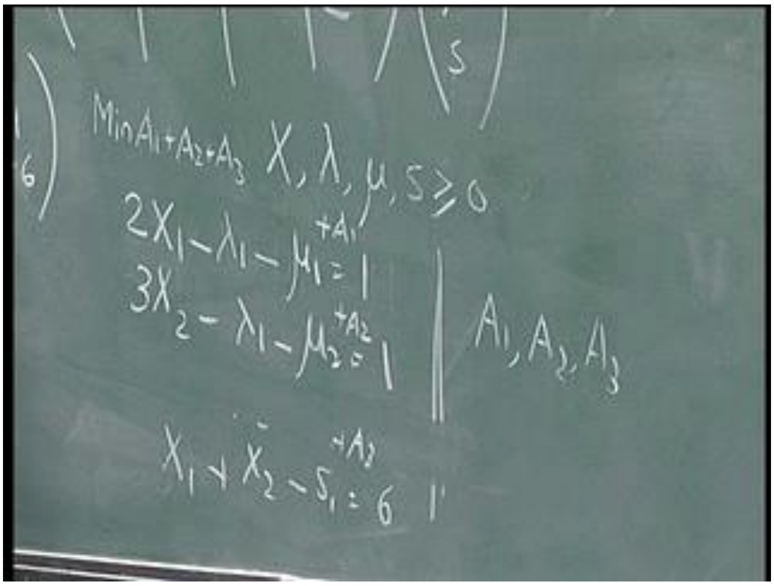


So the numerical example that we would have is this. Minimize X_1 square, plus 3 by 2 X_2 square, minus X_1, minus X_2, subject to the condition X_1 plus X_2 greater than or equal to 6 and X_1 X_2 greater than equal to 0. There are two variables, so X_1, and X2 that we have; there is a single constraint, so there is a lambda_1. Since, there are two variables X_1 and X_2; there are two Lagrangian multipliers mu_1 and mu_2. Since, there is only one constraint here, you get a S_1. These will be the set of variables that we will have for X, lambda, mu, S, as X_1 X_2 lambda_1 mu_1 mu_2 S_1.

Again, this is the linear portion, so this is your C. This first is written as maximize, because this table was set up for maximization. We get maximize minus X_1 square, minus 3 by 2 X_2 square, plus X_1, plus X_2, so that C or C transpose is 1 1. Then b, which is the right-hand side, is 6. The constraint is X_1 plus X_2 greater than equal to 6. This is a greater than or equal to a constraint, but the Kuhn Tucker conditions were derived under the condition that G_i f X is less than or equal to 0. Therefore, this will become X_1plus X_2 is less than or equal to minus 6. A will become minus 1 and minus 1. This will become X_1 plus X_2 less than or equal to minus 6, so A will become minus 1 minus 1.

Now this, from which we have to write 2D, so D will become 1 0, it is a maximization, so minus 1 0, 0 minus 3 by 2. So that X transpose DX would give us minus X_1 square 0 X_1 X_2 minus 3 by 2 X_2 square. We do not have an X_1, X_2 term here. If we had an X_1, X_2 term, say with a positive coefficient here, then this would come as something with a negative coefficient here. This divided by 2, should appear here, because the multiplication actually happens twice. Right now we do not have an X_1, X_2 term, so we can leave this as it is. We do not have any difficulty about this particular thing.

(Refer Slide Time: 34:25)



We already know C, we know b, we know A, we know D, so we can write it in this form and see how it looks like. Right now, minus 2D will become 2 0, 0 3, A is minus 1 minus, I will become minus 1 0 0 minus 1 and 0 is 0 0. The third one will the last one, will be minus 1 minus 1, which is A, 0 and then this 0 will be 0 0, I will be plus 1. This multiplied by X_1, X_2, lambda$_1$, mu$_1$, mu$_2$, S_1 is equal to, we have C, which is 1 1 and we have b, which is a minus 6, because this is written in the form, minus X_1 minus X_2 is less than or equal to minus 6.

When we expand this, we would get $2X_1$ minus lambda$_1$ minus mu$_1$ is equal to 1. We have 3 X_2 minus lambda$_1$ minus mu$_2$ equal to 1. Then, we have minus X_1 minus X_2 plus S_1 is equal to minus 6, this is what we have here. We need to solve this set of three equations, subject to the condition that lambda mu X S greater are equal to 0 and lambda$_i$ S_i equal to mu$_j$ X_j equal to 0. When we write this, we also mentioned that this can be solved as a linear programming problem.

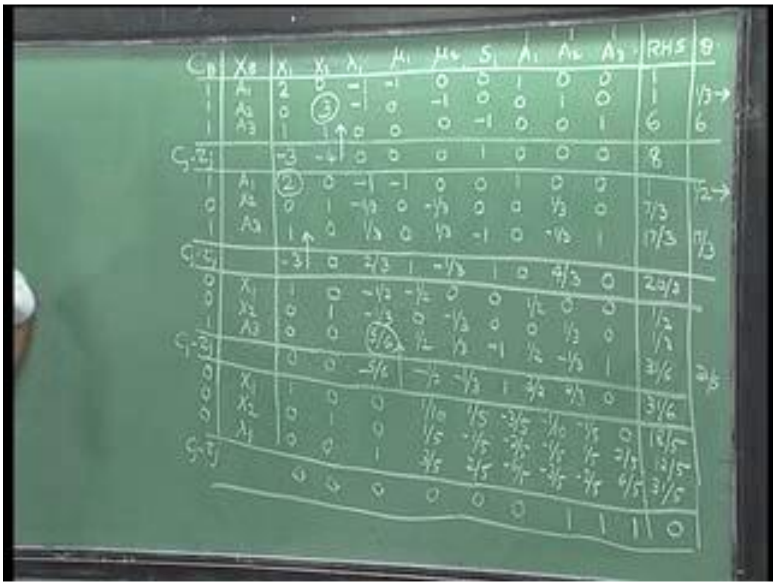
The first thing that we have to do if we were to solve it as a linear programming problem is to ensure that the right-hand side values are non-negative, so this is multiplied again with a minus 1 to get X_1 plus X_2 minus S_1 equal to 6. It gets X_1 plus X_2 minus S_1, which comes from this inequality. So X_1 plus X_2 minus S_1 equal to 6. We can leave out this particular equation, which is now rewritten as this (Refer Slide Time: 37:10). We actually solve for six variables here, three equations here, subject to the condition that these two are satisfied and these are also satisfied. If we want to write it as a linear programming problem, then we have all equations here.

The first thing we see or we verify is whether we can identify a basic variable. When we expand this in this form, we realize X_1 does not have an identity matrix, because X_1 is here, as well it does not have an identity column, because X_1 is here and X_1 is here. X_2 does not have an identity column, because it is present here and it is present here. $Lambda_1$ does not qualify because it has a minus 1 and a minus 1. Then, mu_1 seems okay, but mu_1 has a negative coefficient, therefore it does not qualify, mu_2 also does not qualify, S_1 also does not qualify.

In order to solve this as a linear programming problem, since we have equations, we introduce three artificial variables A_1, A_2 and A_3, such that A_1, A_2, A_3 now form the initial basis and the objective function automatically shifts. We add an A_1 here, so this will become plus A_1equal to 1 plus, A_2 equal to 1, plus A_3 equal to 6. Then we say, minimize A_1 plus A_2 plus A_3. This is like the two phase method where finally, if we get a solution with Z equal to 0; A_1, A_2, A_3 which are currently the beginning basic variables are out of the basis. Some three other variables get into the basis. Since all X, lambda, mu and S is greater than equal to 0, the moment we get a solution with Z equal to 0, we have reached the optimum.

What we do now is we take these three equations, we add the artificial variables A_1, A_2, A_3. Then, we set up the corresponding simplex table so that we solve this system, minimize A_1 plus A_2 plus A_3, subject to, this plus, this plus, this with three artificial variables introduced into the problem.

(Refer Slide Time: 39:25)



That we show here as part of this table. This is the expanded simplex table, where we have X_1 X_2 lambda$_1$ mu$_1$ mu$_2$ and S_1, which are these six original variables. As I mentioned, we need three more artificial variables A_1, A_2 and A_3, which are also shown here as A_1, A_2 and A_3. In the first equation, which is written here as $2X_1$ minus lambda$_1$ minus mu$_1$ plus A_1 equal to 1 2 X_1 minus lambda$_1$ minus mu$_1$ plus A_1 equal to 1. Similarly, the second one is $3X_2$ minus lambda$_1$ minus mu$_2$ plus A_2 equal to 1. The third is X_1 plus X_2 minus S_1 plus A_3 equal to 6.

This is written here, these A_1, A_2, A_3 being artificial variables, have an objective function contribution equal to 1. The initial simplex stable will look like this minus 3, minus 4, 0 0 etc., we should also remember that, while we have to solve for this set, we also cannot ignore this and this. This has been taken care of by the fact that we have formulated a linear programming problem.

This not only minimizes A_1 plus A_2 plus A_3, this is going to have X lambda mu S greater than or equal to 0. We have left this out, so we temporarily relax this. When we do this simplex iteration, now this is for C_j minus Z_j and this is for a minimization problem. The most negative C_j minus Z_j will enter, so variable X_2 enters the basis. Whenever a variable enters the basis, we now invoke this condition and make sure that this condition is not violated. X_2 enters the basis; the corresponding leaving variable is A_2. Our condition is from here, X_2 into mu$_2$ should be equal to 0. Right now mu$_2$ is not in the basis. Therefore, we can comfortably enter X_2 into the basis. Only when mu$_2$ is in the basis and X_2 tries to enter, we can enter X_2

only when mu_2 leaves, otherwise we cannot do that. We have to look at another entering variable, such that these conditions are not violated.

Hence there is absolutely no difficulty in entering X_2 here. So X_2 enters the basis, A_2 leaves the basis so we do one simplex iteration to get A_1, X_2 and A_3 as a new set of basic variables. The original one, since the objective function values are 1, the objective function value is 8 here. We can follow the simplex. We have 4 and 1 by 3, so 4 by 3, so 8 minus 4 by 3, is 20 by 3 which has come here. Then, at the end of simplex iterations, we still realize that, these two artificial variables are in the basis. We also see that the most negative C_j minus Z_j will enter, so variable X_1 enters. When variable X_1 enters, we try to invoke this condition. The corresponding leaving variable is artificial variable A_1. Entry of variable X_1 does not affect this, because mu_1 is currently non-basic at 0. X into mu is equal to 0 is satisfied.

We enter X_1, now A_1 leaves, we perform one more simplex iteration to get X_1, X_2 and A_3. We have still not reached the optimum because variable $lambda_1$ can enter the basis with a negative value. In some sense we have not reached the optimum with respect to the quadratic programming, because we still have not got 0 objective function values. Now, $lambda_1$ tries to enter here, with a negative value. Again we do not want to invoke this condition; $lambda_i$ S_i should be equal to 0. S_1 is not in the basis; therefore, $lambda_1$ can comfortably enter the basis. So, $lambda_1$ enters the basis and there is 1 leaving variable, which is A_3. Now A_3 leaves the basis.

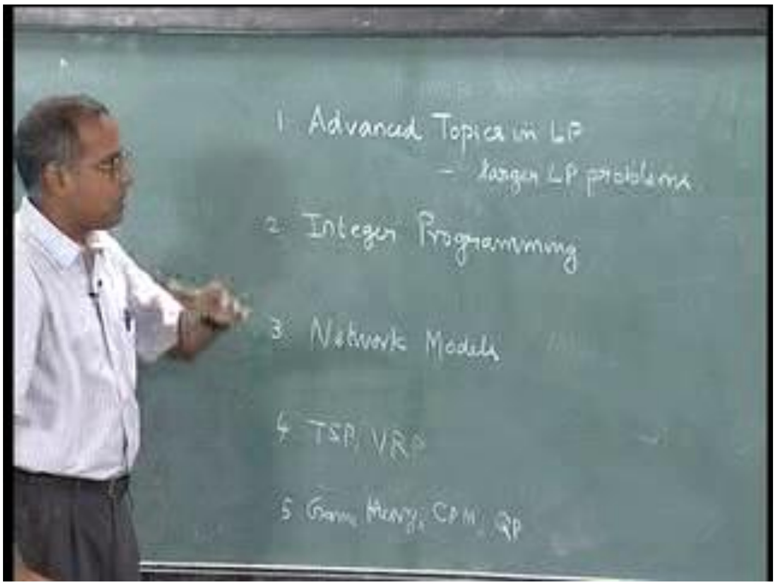
At the end of this iteration, we have X_1 equal to 18 by 5 X_2 equal to 12 by 5. $Lambda_1$ equal to 31 by 5 with Z equal to 0 and there is no entering variable. There is no candidate for entering variable. The simplex algorithm terminates and in this simplex we have also ensured that these are not violated. Therefore, the solution that we have here, with X_1 equal to 18 by 5 and X_2 equal to 12 by 5 is optimal with respect to the given quadratic programming problem. The corresponding Z can be calculated for X_1 equal to 18 by 5 and X_2 equal to 12 by 5. That is how a quadratic programming problem is solved. We can go back and now calculate, the Z for X_1 square plus 3 by $2X_2$ square minus X_1 minus X_2 and get the corresponding value of the objective function. This is how the quadratic programming problem is solved to optimality.

We wish to go back and show that, this portion is like an LP application because the quadratic nature of the quadratic programming problem. When we apply the Kuhn Tucker conditions, we end up getting a set of equations, a set of linear equations along with this, plus this. The set of linear equations with the non-negativity of the restriction of the variable is now solved as a linear programming and shown as an LP application to this problem. The other important thing there is in all the simplex iterations, we have to ensure that, $lambda_i$ S_i equal to 0, mu_j X_j equal to 0. Whenever a lambda or an X or a mu or an S enters the basis or tries to enter the basis, based on the largest coefficient rule, we have to make sure that, first find out the leaving variable and then make sure that the pair and the entry of this does not violate any of these.

If it violates then choose another variable which can enter such that there is a corresponding leaving variable and these conditions are not violated. In a way, it is not like, just feeding a linear programming problem say into a solver and getting the answer, because at every iteration we have to make sure that these are not violated. Therefore, it boils down to looking at every iteration. It is almost like doing it by hand or putting another condition into the linear programming and solving it. Nevertheless, we can use the very fact that the objective function being quadratic, the only place where we have the nonlinearity coming is this form. This is very convenient with respect to simplex because these variables are defined as basic and non-basic variables.

A linear programming based approach to solving a quadratic programming problem is actually in place. That is called the Wolfe's method to solve a quadratic programming problem. With this we come to the end of the discussion of quadratic programming. As we have already mentioned, we are only going to see one part of nonlinear programming, which is the quadratic programming. It is now time to recap, what are all the things that we have seen in this lecture series of advanced operation research.

(Refer Slide Time: 47:00)



We started off with advanced topics in linear programming, we looked at integer programming, we looked at network models, we looked at traveling salesman problem and vehicle routing problem and then, we also looked at a little bit of game theory, CPM and quadratic programming. These are all the things that we have seen in this course.

Here, we saw several topics which largely help us to solve slightly larger linear programming problems. We use certain features of simplex or certain ideas such that, we are able to expand the scope of solving linear programming to slightly larger problems, slightly specific problems and problems that have certain structures. We looked at even some varieties like goal programming here, where we tried to look at multiple objectives and then come up with a meaningful way of trying to solve them. One could also see that there was a lot of linear programming related stuff about complementary slackness conditions and so on.

We also looked at column generation and solved cutting stock problem, so that large problems can be solved. We solved using decomposition algorithm to solve different types of linear programming problems. The integer programming provided us with a separate solution methodology but again largely based on linear programming, the idea either in the branch and bound or in the cutting plane we solved problems that have an explicit integer restriction on the variables, both the large focus was on cutting plane and branch and bound and both of them in some sense used ideas from linear programming. We also said that some ideas from here, some ideas from efficient matrix multiplication can also be used in solving here. This in particular has tremendous real life application and potential.

Then we moved to network models, where the unimodularity nature of the constraints helped us to develop LP based algorithms, even though they were integer programming problems. Each one of them used certain special feature of the network model to get faster and better algorithms, than simply solving it by the simplex method. Here, we solved the shortest path problem, the maximum flow problem and the minimum cost flow problem and certain various versions of this, which were also, solved using principles and ideas from linear programming suitably modified for these types of problems. Then we moved to a set of problems traveling salesman and vehicle routing, which were difficult problems, which were closer to integer programming. They were difficult problems, they all had integer programming formulations, and their exact solutions depended largely on branch and bound methods, which we studied in integer programming.

Then, we also had to look at heuristics, so that whenever we get into solving difficult problems, large sizes, it is absolutely necessary to provide good approximate solutions. Then, we saw a bit of game theory along with CPM and a little bit of quadratic programming. Each one is an area in its own right. Then, we introduced some queueing theory. We simply introduced some basics of each one of them. Queueing theory was the only topic that we looked at in this course, where we looked at some probabilistic kind of model. The rest of them could be seen as linear programming applications, even though each one has a lot of practical applications and each one is a separate field in its own right.

We complete this lecture series on advanced topics in operations research, providing the user with a set of tools, which can be used extensively to solve real life practical problems. We hope this lecture series is useful in understanding the tools of operations research and learn more and more tools and techniques to solve real life problems. Thank you.