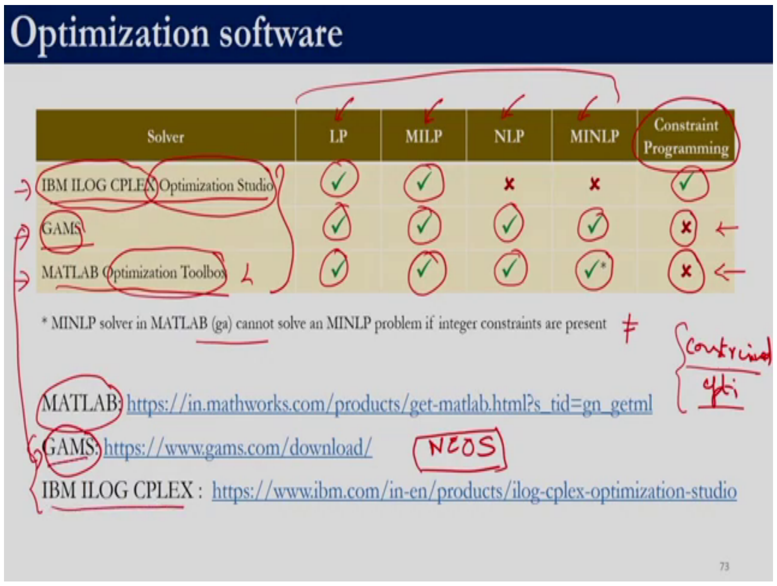
Computer Aided Applied Single Objective Optimization

Dr. Prakash Kotecha
Department of Chemical Engineering
Indian Institute of Technology, Guwahati

Lecture - 27
MATLAB Inbuilt Functions:
Linear & Mixed Integer Linear Programming

Welcome in the next few sessions we will be looking at 3 different optimization software.

(Refer Slide Time: 00:36)



Optimization software

Solver	LP	MILP	NLP	MINLP	Constraint Programming
IBM ILOG CPLEX Optimization Studio	✓	✓	×	×	✓
GAMS	✓	✓	✓	✓	×
MATLAB Optimization Toolbox	✓	✓	✓	✓*	×

* MINLP solver in MATLAB (ga) cannot solve an MINLP problem if integer constraints are present $\neq$

Constrained opti

MATLAB: https://in.mathworks.com/products/get-matlab.html?s_tid=gn_getml

GAMS: https://www.gams.com/download/ NEOS

IBM ILOG CPLEX : https://www.ibm.com/in-en/products/ilog-cplex-optimization-studio

73

We will be starting with MATLAB Optimization Toolbox. So, it can be used to solve linear programming, mixed integer linear programming, it has inbuilt functions to solve non-linear programming problem and it can solve mixed integer non-linear programming problems right. To the best of our knowledge there is only one function in MATLAB which can solve mixed integer non-linear programming problem. Again, over here it can solve mixed integer non-linear programming problems provided they do not have equality constraint.

So, that is the restriction that MATLAB has with respect to solving mixed integer non-linear programming problems right. We will also be looking into GAMS right. So, GAMS can be used to solve linear programming problems, mixed integer linear programming problems, non-linear programming problems, mixed integer non-linear programming problems and there are various other types of problem which we can solve. But as part of this course we are only classifying an optimization problem to be one of this 4 right.

So, GAMS can be used to solve any of them right. IBM ILOG CPLEX Optimization Studio can be used to solve linear programming as well as mixed integer linear programming problem. It also supports quadratic programming, in addition it supports what is called as constraint programming. So, this constraint programming is an optimization technique right.

So, constraint programming is not to be confused with constrained optimization. So, when we say constrained optimization it means there are constraints right. So, a linear programming problem can have constraint mixed integer linear programming problem can have constraint. Similarly, non-linear programming and mixed integer non-linear programming problem can also have constraints right.

So, when we say constraint optimization we are specifying something about the structure of the problem, that the problem has constraint. Whereas this constraint programming is actually an optimization technique right. So, to the best of our knowledge there are no inbuilt functions in MATLAB which can be used to solve problems using constraint programming.

Similarly, to the best of our knowledge GAMS does not seem to have anything on constraint programming. Again, remember these are software which are continuously being developed right. So, you can always go and have a look at their websites to see if they also include solution using constraint programming.

So, we have chosen these three software for various reasons. So, for example, MATLAB optimization toolbox is available free of cost a part of this course right and MATLAB is more commonly available in many of the institutes right. So, that is why we have chosen to include MATLAB optimization toolbox. So, IBM ILOG CPLEX optimization studio for academic purposes the full version can be downloaded.

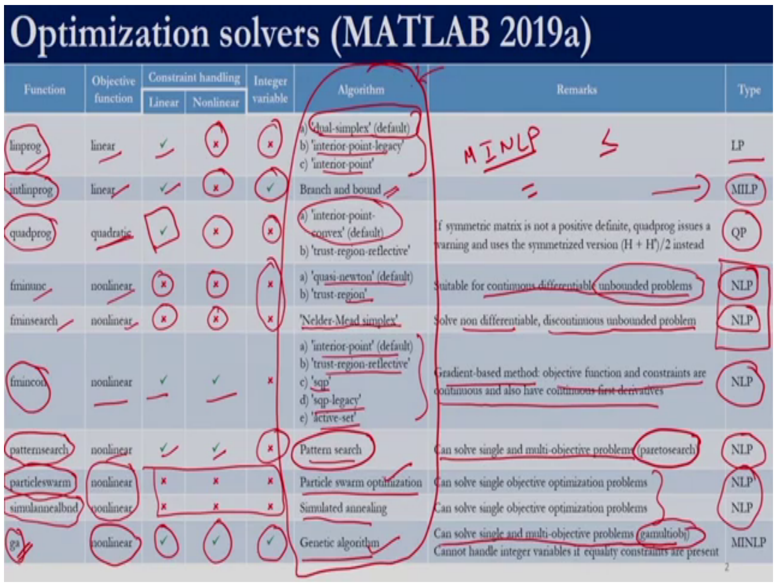
So, it is available as part of IBM ILOG academic initiative right so and on top of that it is the only software which supports Constraint Programming. Again, as part of this course we have not looked into constraint programming, but still we are talking about constraint programming because we want to know that there is something called as constraint programming and you can go and explore on your own right.

This GAMS if you see it is the only software which can solve a linear programming problem, mixed integer linear programming problem, non-linear programming problem and mixed integer non-linear programming problem right. So, that is why we have included GAMS right, though it does not support constraint programming right. So, based on the popularity availability and it is ability to solve problems, we have chosen these three different optimization software.

As far as accessibility is concerned MATLAB optimization toolbox is available to you this can be freely downloaded for academic purposes right. So, GAMS can be downloaded from their website right. So, you can download the demo version, so demo version works for a limited number of variables and constraints right. So, this GAMS is essentially a modeling language. So, this GAMS can be used along with NEOS solver right. So, we can model the problem using GAMS and then upload the code in NEOS solver right and we can upload the code into NEOS solver and get the solution right. So, this NEOS solver comes without any type of charges right.

So, in that sense all these 3 software are available to you right. We will start with MATLAB because we presume that you have been able to install MATLAB which was given to us part of this course right. Then we will subsequently move on to these two software by the time we complete MATLAB. We expect that you have already installed GAMS and IBM ILOG CPLEX. So now, will move on to the inbuilt functions of MATLAB which can be used to solve different type of optimization problems.

(Refer Slide Time: 05:10)



Optimization solvers (MATLAB 2019a)

Function	Objective function	Constraint handling		Integer variable	Algorithm	Remarks	Type
		Linear	Nonlinear				
linprog	linear	✓	✗	✗	a) 'dual-simplex' (default) b) 'interior-point-legacy' c) 'interior-point'	MINLP 5	LP
intlinprog	linear	✓	✗	✓	Branch and bound		MILP
quadprog	quadratic	✓	✗	✗	a) 'interior-point-convex' (default) b) 'trust-region-reflective'	If symmetric matrix is not a positive definite, quadprog issues a warning and uses the symmetrized version (H + H')/2 instead	QP
fminunc	nonlinear	✗	✗	✗	a) 'quasi-newton' (default) b) 'trust-region'	Suitable for continuous differentiable unbounded problems	NLP
fminsearch	nonlinear	✗	✗	✗	'Nelder-Mead simplex'	Solve non differentiable, discontinuous unbounded problem	NLP
fmincon	nonlinear	✓	✓	✗	a) 'interior-point' (default) b) 'trust-region-reflective' c) 'sqp' d) 'sqp-legacy' e) 'active-set'	Gradient-based method: objective function and constraints are continuous and also have continuous first derivatives	NLP
patternsearch	nonlinear	✓	✓	✗	Pattern search	Can solve single and multi-objective problems (paretosearch)	NLP
particleswarm	nonlinear	✗	✗	✗	Particle swarm optimization	Can solve single objective optimization problems	NLP
simulannealbnd	nonlinear	✗	✗	✗	Simulated annealing	Can solve single objective optimization problems	NLP
ga	nonlinear	✓	✓	✓	Genetic algorithm	Can solve single and multi-objective problems (gamultiobj) Cannot handle integer variables if equality constraints are present	MINLP

2

MATLAB as a large number of inbuilt functions, we will be seeing a few of them right. So, whatever we are going to discuss is based on MATLAB 2019 a right. So, the first function that we will be looking at is linprog. In linprog the objective function has to be linear the constraints have to be linear it does not support non-linear constraints and it does not support integer variable right. So, that is nothing but a linear programming problem.

If you have a linear programming problem then linprog can be used to solve that problem right. So, again remember linprog is not the name of any algorithm right. So, linprog is the name of a function right. So, with that function we can use any of these 3 algorithm dual simplex or interior point legacy or interior point method can be used with this function linprog to solve linear programming problems right.

So, by default dual simplex is what is selected, if you want to override the default option and if you want to solve by any of these two methods it can be done. So, we will show you one example wherein we can change the default setting of any of this function right. The second function is intlinprog right. So, as the name indicates linprog, so the objective function has to be linear the constraints have to be linear, it does not support non-linear constraints it can have integer variables right.

So, here we are talking about mixed integer linear programming problem, some variables can be integers some variables can be continuous right, but the objective function as well as the constraints are linear right. So, if we have a mixed integer linear programming problem we will be solving in it using intlinprog. So, for MATLAB intlinprog it uses the branch and bound method.

Then we have quadprog, so this is for solving quadratic problem or QP problem in which the constraints need to be linear, the objective function can be quadratic it does not support non-linear constraints, it does not support integer variables right. So, again there are 2 algorithms which can be used to solve quadratic programming problem; one is interior point convex method the other trust region reflective. So, the default option is interior point convex method.

Then we have these two functions fminunc and fminsearch right, for both of these functions this is for unconstrained optimization problem right. So, no constraints are allowed no linear constraints no non-linear constraints no linear constraints no non-linear constraints and integer variables are also not allow. The objective function can be non-linear. So, for fminunc the two methods which are available is Quasi Newton Method which is the default method and it also has trust region method. Whereas, for fminsearch the underline algorithm is Nelder Mead Simplex Method right.

So, fminunc is suitable for continuous differentiable unbounded problems right, so there should not be any bounds also right. Whereas, fminsearch can be used for solving non differentiable discontinuous unbounded problems. So, both of them fall under non-linear programming, but unconstrained non-linear programming. The other widely used function as fmincon right. So, fmincon can support linear constraints non-linear constraints and non-linear objective function right. It has these 5 methods interior point default, trust region reflective, sqp, sqp legacy and active set right.

So, this is a gradient based method right. So, the objective function and constraints are continuous and should also have continuous first derivatives. In those cases we can use function fmincon to solve non-linear constrained optimization problem. So, here also we have written NLP right, so but here constraints are allowed. Whereas, in these two constraints were not allowed. The other function that we have is pattern search in MATLAB right. So, it can also support linear constraint, non-linear constraint it cannot handle integer variables. The objective function can be linear or non-linear the underline algorithm is pattern search right.

So, this pattern search can be used to solve single as well as multi objective problems right. Though multi objective optimization is not part of this course we will show you how to solve one multi objective optimization problem using pattern search right. So, here also it is a non-linear programming problem right. So, if there are integer variables non-linear constraints it cannot solve it. Even if everything was linear and if there is at least one integer variable, then again pattern search cannot be used to solve those problems right. So, it supports only non-linear programming no integer variable.

So far whatever functions we have seen only this intlinprog supports integer variables. But again there the constraints and objective function has to be linear, other two functions which will be looking is particle swarm. So, this is based on particle swarm optimization which we have already discussed in this scores right and there is another technique called as simulated annealing right. So, the function for that is simulannealbnd right. So, both of these can be used to solve non-linear problem. But they do not allow for any constraints, both of them are for unconstrained optimization problems right.

So, if does not support integer variable size, set particle swarm is based on particle swarm optimization simulannealbnd is based on simulated annealing right. So, both of this algorithm is again restricted to solving single objective optimization problem and then we have the inbuilt MATLAB function ga right. So, here it can support non-linear objective function it can support linear constraints non-linear constraints as well as integer variable. So, the underlying algorithm is genetic algorithm, which we have previously seen in this course right.

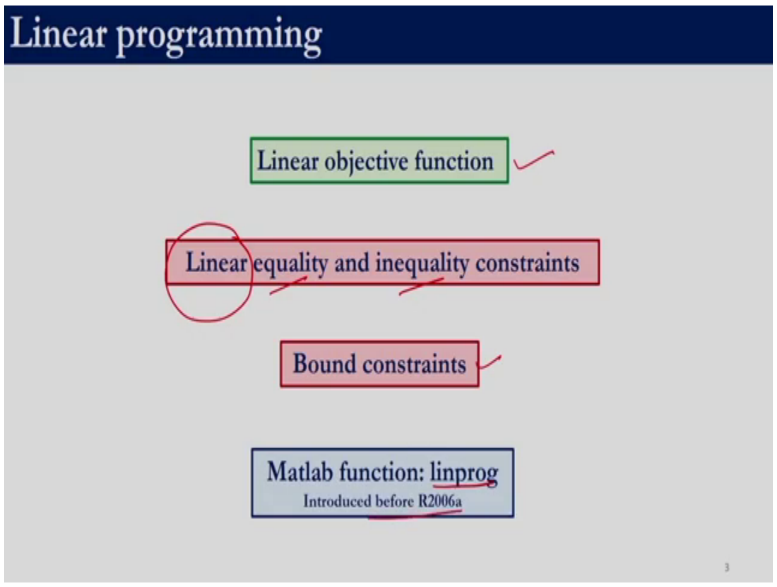
So, it can solve single as well as multi objective optimization problems right. So, if you have to use multi objective optimization then the function name is a gamultiobj. For single objective the name of the function ga, similarly here in pattern search also pattern search are search supports only single objective. But the algorithm pattern search can be used to solve multi objective problems using this function paretosearch. So, if we see genetic algorithm is able to support integer variables linear constraints as well as non-linear constraints right.

So, it can solve mixed integer non-linear programming problems as long as the constraints are inequalities right. It cannot support integer variables as well as equality constraints. If there are no integer variables it can support equality constraints, but if there are integer variables it cannot support equality constraints right. So, we will be looking into these 10 functions right, there are a few additional functions which will list out at the end.

So, we expect that you look at the help of MATLAB and try to use them as an when necessary. But for this course we will just demonstrate the working of these 10 functions. Again, remember we have not looked into many of these algorithms over here right. So, we have only looked at particle swarm optimization, genetic algorithm and we will be looking at simplex method right. But otherwise we have not looked into how is the underlying algorithm performing right.

So, there are various courses under NPTEL itself which covers many of this algorithms right. So, if you are interested you can go and have a look at those videos. Now, we look into these functions which we have selected as part of this course right, to begin with we will discuss linprog and intlinprog.

(Refer Slide Time: 12:24)



Linear programming

Linear objective function

Linear equality and inequality constraints

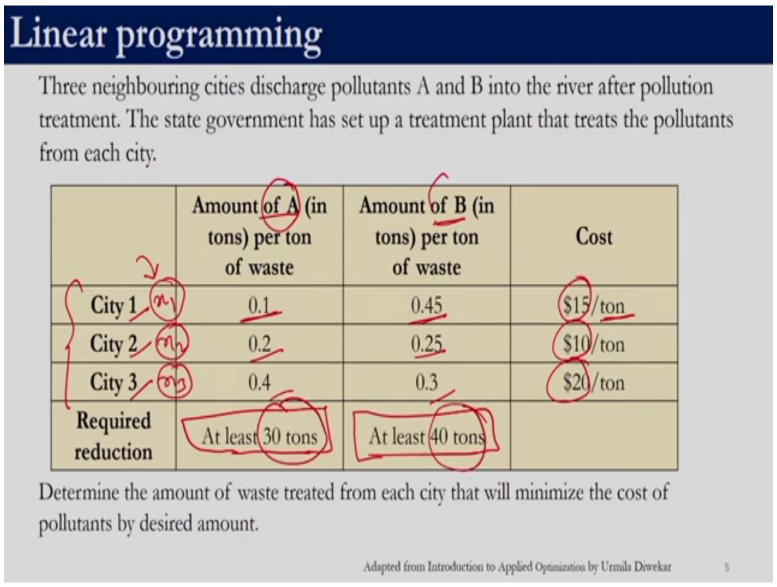
Bound constraints

Matlab function: linprog

Introduced before R2006a

So, in the session we will be looking at how to solve Linear programming problems using MATLAB. So, when we say linear programming problem the objective function has to be linear, there can be equality as well as inequality constraints. But they need to be linear right. Similarly, there can be Bound constraints, bound constraints as in the decision variables are restricted between a lower bound and upper bound. The MATLAB function to solve linear programming problem is linprog right. So, linprog has been there in MATLAB for quite some time.

(Refer Slide Time: 12:52)



Linear programming

Three neighbouring cities discharge pollutants A and B into the river after pollution treatment. The state government has set up a treatment plant that treats the pollutants from each city.

		Amount of A (in tons) per ton of waste	Amount of B (in tons) per ton of waste	Cost
City 1	x_1	0.1	0.45	\$15/ton
City 2	x_2	0.2	0.25	\$10/ton
City 3	x_3	0.4	0.3	\$20/ton
Required reduction		At least 30 tons	At least 40 tons	

Determine the amount of waste treated from each city that will minimize the cost of pollutants by desired amount.

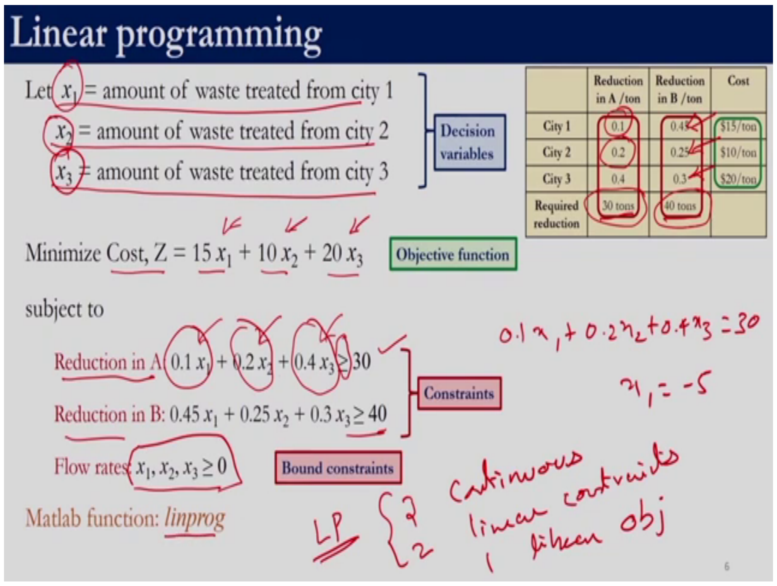
Adapted from Introduction to Applied Optimization by Urmila Diwekar 5

Let us look into a Linear programming problem. So, in this problem there are 3 cities City 1 City 2 City 3 right. So, there is a plant which is being set up right. So, it has to be decided as to how much stream which is coming out from city 1, city 2 and city 3 is to be treated right, such that a certain amount of pollutant is removed right. If a process let us say x 1 amount of stream from city 1 then the amount of A which is a pollutant that we can remove from that stream is 0.1. The amount B that will be able to remove by processing x 1 amount of stream coming out of city 1 is 0.45 into x 1. Similarly, this values have been given for city 2 and city 3 right.

So, this is let us assume that we will be processing x 2 and let us assume that we will be processing x 3, so this is not known. So, the question is how much of stream coming out of city 1 has to be process, how much of stream coming out of city 2 has to be process and how much of stream coming out of city 3 has to be processed right. Given this is the pollutant content right.

So, to process 1 ton of stream which is coming out of city 1 it requires 15 dollars. Whereas, it requires only 10 dollars to process 1 ton of stream coming from city 2 and it requires 20 dollar to process 1 ton of stream coming from city 3 right. So, we need to decide on how much from each city it has to be processed, such that the total cost is minimum right. There are 2 constraints at least 30 tons of pollutant A has to be removed and 40 tons of pollutants B has to be removed.

(Refer Slide Time: 14:33)



Linear programming

Let x_1 = amount of waste treated from city 1

x_2 = amount of waste treated from city 2

x_3 = amount of waste treated from city 3

Decision variables

	Reduction in A /ton	Reduction in B /ton	Cost
City 1	0.1	0.45	\$15/ton
City 2	0.2	0.25	\$10/ton
City 3	0.4	0.3	\$20/ton
Required reduction	30 tons	40 tons	

Minimize Cost, $Z = 15\,x_1 + 10\,x_2 + 20\,x_3$ Objective function

subject to

Reduction in A: $0.1\,x_1 + 0.2\,x_2 + 0.4\,x_3 \geq 30$

Reduction in B: $0.45\,x_1 + 0.25\,x_2 + 0.3\,x_3 \geq 40$

Constraints

Flow rates: $x_1, x_2, x_3 \geq 0$ Bound constraints

Matlab function: *linprog*

$0.1x_1 + 0.2x_2 + 0.4x_3 = 30$

$x_1 = -5$

LP $\begin{cases} ? \text{ Continuous} \\ 2 \text{ linear constraints} \\ 1 \text{ linear obj} \end{cases}$

6

So, if you assume that x 1 is the amount of waste treated from city 1, x 2 is the amount of waste treated from city 2 and x 3 is the amount of waste treated from city 3 right. And if this is the cost right 15 dollars 10 dollars and 20 dollars. So, the total cost that is required is 15 x 1 because for processing 1 ton of waste from city 1 we required 15 dollars right.

So, if we process x 1 quantity then we required 15 x 1, similarly 10 x 2 and 20 x 3. So, the total cost is 15 x 1 plus 10 x 2 plus 20 x 3 right. So, this is the total cost. So now, this cost has to be minimized right, if you think about it x 1 is equal to 0 x 2 is equal to 0 x 3 is equal to 0 gives the cost of 0 right.

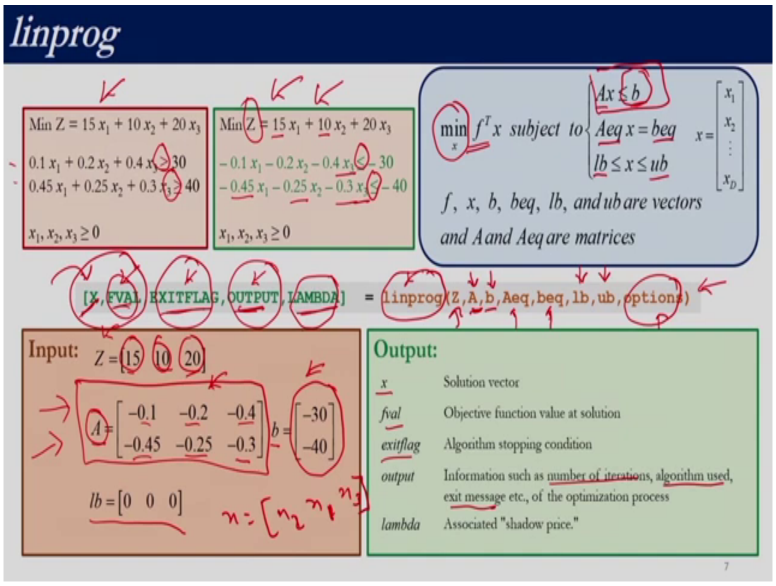
Since x 1 x 2 x 3 cannot take negative values, the lowest values that they can take is 0 right. So, the optimal cost is 0 right, but then that will not ensure this removal that 30 tons of A and 40 tons of B should to be removed right. So now, we need to include constraints such that this phenomenon is capture. So, if we treat x 1 quantity of waste coming from city 1 the total amount of A that we are removing is 0.1 x 1, the total amount of A which we are removing by processing by processing x 2 quantity of waste from city 2 is 0.2 x 2 similarly 0.4 x 3. So, this is the A that can be removed, if we process x 1 from city 1 x 2 from city 2 and x 3 from city 3. So, it is said that we need to remove at least 30 tons. So, that is why we have this greater than equal to symbol right.

So, it can be greater than 30 or it can be equal to 30, it need not be exactly equal to 30 right. So, writing this as 0.1 x 1 plus 0.2 x 2 plus 0.4 x 3 is equal to 30 is not correct right, it does not capture what is required in the problem. It says we need to remove at least 30, if we you are removing more than 30 that is not a problem right. So, this constraint captures only the reduction in pollutant A. So, the reduction in pollutant B similarly can be done by 0.45 x 1 plus 0.25 x 2 plus 0.3 x 3. Should be greater than or equal to 40, again the problem requires that at least 40 tons B removed.

So, these 2 are constraints, so here if you see the objective function is linear right, both the constants are linear and the variables are continuous variables. So, x 1 x 2 x 3 are continuous variable the lower bound is 0 right. So, it does not make sense if we say that we will process minus 5 tons from city A right, it has to be a positive quantity right. So, that is why we have this x 1 comma x 2 comma x 3 is greater than or equal to 0.

So now, we have continuous variables, all the 3 variables are continuous and we have 2 constraints which are linear right and the objective function is also linear. This problem if you classify it will be linear programming. So, we will use the MATLAB function linprog to solve this problem right.

(Refer Slide Time: 17:37)



linprog

Min Z = $15x_1 + 10x_2 + 20x_3$

$0.1x_1 + 0.2x_2 + 0.4x_3 \geq 30$

$0.45x_1 + 0.25x_2 + 0.3x_3 \geq 40$

$x_1, x_2, x_3 \geq 0$

Min Z = $15x_1 + 10x_2 + 20x_3$

$-0.1x_1 - 0.2x_2 - 0.4x_3 \leq -30$

$-0.45x_1 - 0.25x_2 - 0.3x_3 \leq -40$

$x_1, x_2, x_3 \geq 0$

$$\min_x f^T x \text{ subject to } \begin{cases} Ax \leq b \\ Aeq\, x = beq \\ lb \leq x \leq ub \end{cases} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

f, x, b, beq, lb, and ub are vectors and A and Aeq are matrices

```
[X,FVAL,EXITFLAG,OUTPUT,LAMBDA] = linprog(Z,A,b,Aeq,beq,lb,ub,options)
```

Input: $Z = [15\ 10\ 20]$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -40 \end{bmatrix}$$

$lb = [0\ 0\ 0]$

$x = [x_2\ x_1\ x_3]$

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message etc., of the optimization process
lambda	Associated "shadow price."

7

So, if you do help space linprog on the command window it will give help on the function. So, linprog requires the problem in this format right. So, the constraint should be in this less than equal to form right. If we have greater than equal to form it cannot be directly be given to MATLAB right. It has to be converted into less than equal to form right and it can solve only minimization problems right.

So, if we have a maximization problem, we have discuss how to converted into a minimization problem right. We need to covert the maximization to problem minimization problem and only then we will be able to you MATLAB to solve it right. So, we look into what is this A equality and b equality with the example that we have considered lb and ub are the lower and upper bound of the decision variables right.

So, right now what the problem that we had was this one minimize Z is equal to 15 x 1 plus 10 x 2 plus 20 x 3 and these two were constraints which you had this greater than or equal to symbol. First, we need to convert this problem into this one were in the constraints are in the less than equal to form right. If the constraints are in the less than equal to form only then we can use MATLAB to solve it right. So, that is why we multiply by a minus sign on both the sides and we have these two constraints right.

So, now this modified problem right which is exactly the same as the original problem, but it is in a format that is required for solving using MATLAB. So, for this problem we need to provide the coefficients of the decision variable in the objective function, in this case the coefficients are 15, 10 and 20 right.

So, we will define that as a variable Z this variable Z is what is to be given or this variable f right. So, it is just a different variable name that we are using right. So, this Z is the coefficients of the decision variables in the objective function right and this A is the coefficient matrix of the inequality constraints right, A and b are two variable names right. They should contain the coefficients of the decision variables in the constraints right.

So, for example, the first constraint is minus 0.1 x 1 minus 0.2 x 2 minus 0.4 x 3 right. The second constraint is minus 0.45 x 1 minus 0.25 x 2 and minus 0.3 x 3. So, this A contains the coefficient matrix right. Whereas, the vector b is the right-hand side of those two constraints, so b is minus 30 and minus 40. The lower bound for this problem is 0 0 0 right.

So, here you need to remember that we need to first decide how we are going to arrange the variables right. So, when we write this we have implicitly assumed that our variable are arrange in x 1 x 2 x 3 right. It is not necessary that we always arrange like this we could have even arranged it does x 2 x 1 and x 3 right. If we have done that then this A matrix should be appropriately rearranged right.

So, here the assumption is that A into x is less than equal to b right. So, A into x is less than equal to b. So, x if we do not consider x 1 x 2 x 3, if we consider x 2 x 1 x 3 this a matrix will appropriately change right. First fix the order in which we are going to arrange the decision variable and then supply coefficients right. So, for the function linprog the input that we need to give is this variable Z right, so which should contain the coefficients of the objective function right.

So, it can be any variable, so for example instead of Z over here I could have used y and I should use the same variable y right. The first variable should contain the coefficients of the objective function right. The second variable and the third variable represent the inequality constraints, the second variable should contain the coefficient matrix of the inequality constraint.

So, the coefficient of the inequality constraints for this problem is given over here right. So, that is A b is that the right hand side values of the inequality constraints right. So, in this case we have minus 30 and minus 40, so b will be minus 30 minus 40. If we had equality constraints just like we derived the coefficient of this inequality constraint, we can also derive the coefficients of equality constraint, so that has to be given in subsequent variables right.

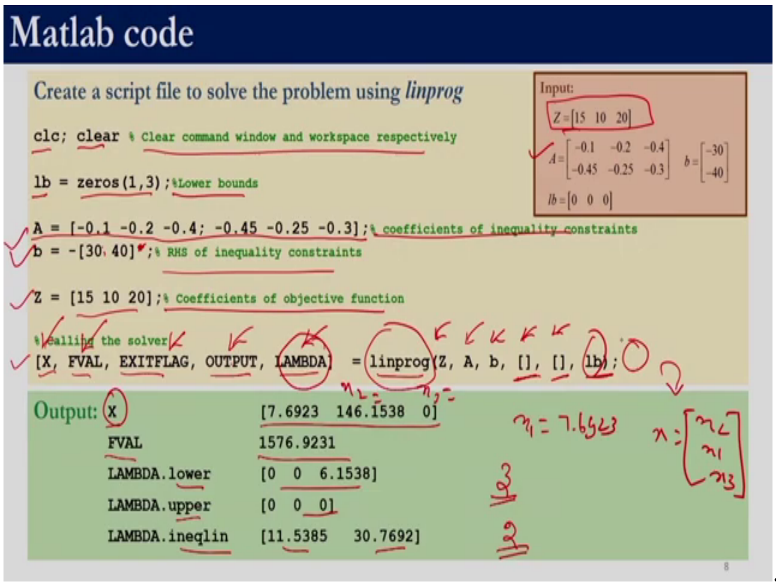
Then we need to provide the lower bound we need to provide the upper bound and then we need to provide the options right. Options is usually provided when we want to override the default options of linprog. So, remember all functions require some kind of user defined input right, if nothing else then you need to at least select which is the algorithm that linprog is going to do. But for all of them MATLAB by default select some value, if we want to override the default value we can use this options to do that right. But right now we will not be looking into those options right so, but that can be done right, the output would be the decision variables right.

So, here for this problem we have x 1 x 2 x 3. So, this x will be a vector of 1 by 3, fval is the value of the objective function at the optimal decision variable right. So, what we get x if we substitute those values of x in this objective function, what will be the value of the objective function that can also be obtained from this linprog function right. So, the second variable would be fval right. The third variable exitflag tells us as to when the algorithm stopped what is the reason for it stopping right, so the exitflag will give us that right.

This variable output is actually a structure; it will give a lot of information including number of iterations what was the algorithm use, why did the algorithm stop which is there in this exitflag that also it will give right. It also gives the what is called us lagrange multiplier or shadow prices right. So, this gives additional information about the solution of the problem right. As of now just know that we will be getting this 5 outputs right.

So, analyzing these two straightforward x is the solution which linprog has been able to get right, fval is the objective function value at that point right and exitflag is the reason for why it is stopping? Output will contain additional information about the solution right and this lambda is the lagrange multipliers or the shadow prices. We have written this x over here below min to indicate that x is the decision variable right. This is the script file which can be used to solve the problem right.

(Refer Slide Time: 24:09)



Matlab code

Create a script file to solve the problem using *linprog*

```
clc; clear % Clear command window and workspace respectively

lb = zeros(1,3);%Lower bounds

A = [-0.1 -0.2 -0.4; -0.45 -0.25 -0.3];% coefficients of inequality constraints
b = -[30, 40]';% RHS of inequality constraints

Z = [15 10 20];% Coefficients of objective function

%Calling the solver
[X, FVAL, EXITFLAG, OUTPUT, LAMBDA]  = linprog(Z, A, b, [], [], lb);
```

Input:

$Z = [15 \;\; 10 \;\; 20]$

$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -40 \end{bmatrix}$

$lb = [0 \;\; 0 \;\; 0]$

Output:

```
X                 [7.6923   146.1538   0]
FVAL              1576.9231
LAMBDA.lower      [0   0   6.1538]
LAMBDA.upper      [0   0   0]
LAMBDA.ineqlin    [11.5385    30.7692]
```

$x_2 =$ $x_3 =$

$x_1 = 7.6923$

$x : \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix}$

$\underline{\underline{3}}$

$\underline{\underline{2}}$

8

So, clc and clear you know we are defining the lower bound 0s of 1 comma 3. So, it will create 1 row 3 columns, because we have 3 variables right. We do not need to create ub because for this problem ub is infinity we need to define this A matrix. So, that is what is being done over here right. So, A is equal to within square brackets minus 0.1 minus 0.2 minus 0.4 semi colon, so that goes onto that next row minus 0.45 minus 0.25 and minus 0.3 right. So, that will help us to define A and this B is the right hand side vector.

So, b is equal to minus of 30 space 40 or transpose or we can remove this transpose and just say minus 30 semi colon 40 right. And this Z is the vector 15 10 20 which is the coefficient of objective function. So, anything that starts after this percentage symbol or comments right it is not required for solving the problem. So, linprog we are accessing the function linprog the order is important.0020

So, the first one should be the coefficient of the objective function the second variable is coefficient matrix of the inequality constraints, the next one is the right hand side vector of the inequality constraints. Since we do not have inequality constraints we do not have a coefficient matrix and we do not have a right hand side vector for inequality constraint. So, this is empty bracket.

And we do have lower bound right, so lower bound is to be supplied right. So, here if you see we have not given comma ub comma options right, because ub for this problem is infinity. As of now we are not changing the default options by which this linprog will solve this problem right, we are solving with default options. So, that is why we have not given lb comma ub comma options right.

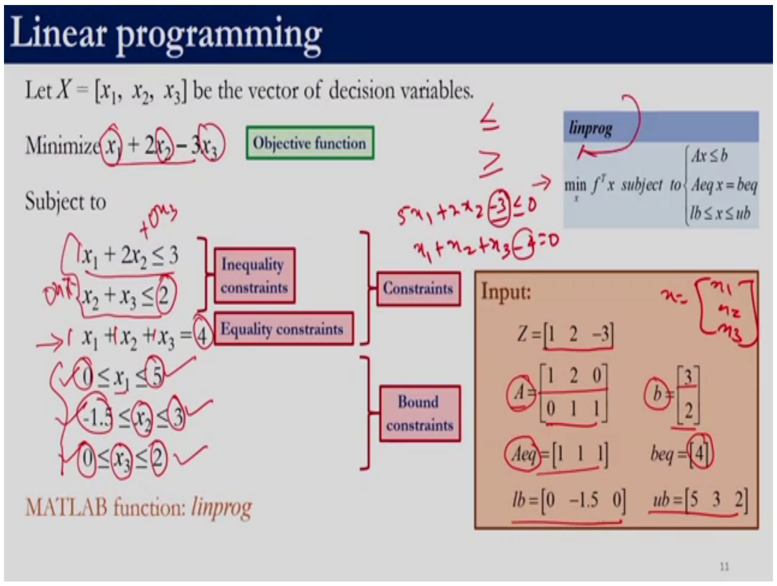
So, when we are solving this we are seeking these 5 values, the decision variable the value of the objective function, the reason why it is terminating in exitflag, output and the shadow prices. So, if we do that this is what we will get, we will get the decision variable 7.6923 146.1538 0. So, this means x 1 is 7.6923 x 2 is this and x 3 is this. Because this is the way in which we had arranged the variables right.

If our x was x 2 x 1 x 3 right then this z would I have not been 15 10 20, it would have been 10 15 20 similarly this A would I have changed right. So, in that case the answer which we would received is 146.15 would have been the first value right. So, the order in which we are supplying the input has to be consistent for the constraints as well as the objective function and the lower and upper bound.

So, the solution that we obtain will be in the same format right. For this decision variable the value of the objective function is 1576.9231 right and this lambda will be a structure right. The structure will contain 3 values, the lower, upper and inequality. So, if we say lambda dot lower we will get 3 values, if we say lambda dot upper we will get 3 values, if we say lambda dot ineqlin we will get 2 values right. So, there are 3 values over here and over here because we have 3 decision variables. There are 2 values over here, because we have 2 inequality constraints right.

So, we will see how to interpret this, but as of now we are just seeing how to access the values right. Once we access the values, we will talk about how to interpret it right. So, in this case you would need to remember that these are just variable names, it need not be x fval exitflag output lambda it can be any 5 variable names right. But lambda would be a structure, output will also be a structure, exitflag will be a scalar fval will be a scalar because it is the value of the objective function. And the size of x is going to be the same as size of our optimization problem or the size of lower and upper bound.

(Refer Slide Time: 28:09)



Linear programming

Let $X = [x_1, x_2, x_3]$ be the vector of decision variables.

Minimize $x_1 + 2x_2 - 3x_3$ Objective function

Subject to

$x_1 + 2x_2 \le 3$, $x_2 + x_3 \le 2$: Inequality constraints

$x_1 + x_2 + x_3 = 4$: Equality constraints

Constraints

$0 \le x_1 \le 5$, $-1.5 \le x_2 \le 3$, $0 \le x_3 \le 2$: Bound constraints

MATLAB function: *linprog*

linprog

$$\min_x f^T x \text{ subject to } \begin{cases} Ax \le b \\ Aeq\,x = beq \\ lb \le x \le ub \end{cases}$$

Input:

$Z = [1\ \ 2\ \ -3]$

$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix}$ $b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$Aeq = [1\ \ 1\ \ 1]$ $beq = 4$

$lb = [0\ \ -1.5\ \ 0]$ $ub = [5\ \ 3\ \ 2]$

$\le$ $\ge$

$5x_1 + 2x_2 - 3 \le 0$

$x_1 + x_2 + x_3 - 4 = 0$

$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

$+0x_3$ $0x_1$

11

The previous problem which we solve using linprog did not have inequality constraint right. So, we will look at an example wherein we have inequality constraint right. So, the objective function that we have over here is x 1 plus 2 x 2 minus 3 x 3 right. So, this is an optimization problem involving 3 variables. The decision variables are x 1 x 2 and x 3 right. We have 2 inequality constraints right both of them are linear right.

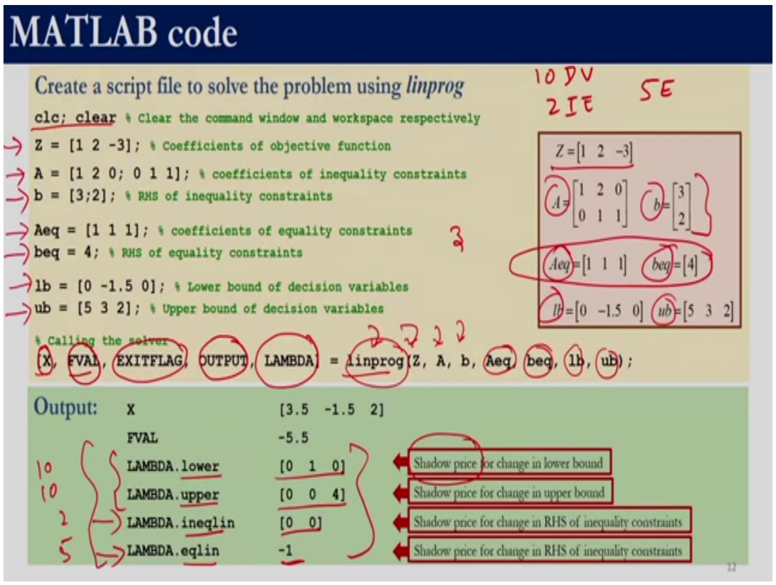
So, x 1 plus 2 x 2 is less than equal to 3 x 2 plus x 3 is less than equal to 2 and we have 1 equality constraint x 1 plus x 2 plus x 3 equal to 4. The lower and upper bounds are as given over here right. So, the variable x 1 has a lower bound of 0 and upper bound of 5, variable x 2 has a lower bound of minus 1.5 and an upper bound of 3 x 3 has a lower bound of 0 and an upper bound of 2 right. So, this is the problem that linprog can solve.

So, here we will define X to be x 1 x 2 x 3 right. So, the coefficient of x 1 x 2 x 3 is 1 2 and minus 3, when we have Z into x we will get x 1 plus 2 x 2 minus 3 x 3 right. So, these 2 constraints we need to find out the coefficient of x 1 x 2 x 3 right. So, first equation if we see one x 1 plus 2 x 2 plus 0 x 3 is less than equal to 3. So, that is why we have 1 to 0 over here and right hand side is 3. Similarly, the second equation is 0 x 1 plus 1 x 2 plus 1 x 3 0 1 1 and then this right hand side value is 2, this is A this is b right and then we define another variable right Aeq right for equality constraints. So, we have only 1 equality constraints right. So, here the coefficient of x 1 x 2 x 3 is 1 1 1.

So, we have 1 1 1 over here and the right hand side value is 4. So, we have 4 over here. So, lb is used to define the lower bound, so the lower bound is 0 minus 1.5 and 0 and the upper bounds are 5 3 and 2, so this is the upper bound right. So now, we have transformed this problem to the format required by a MATLAB right. So, again remember it only supports less then equal to constraints. So, if you have a greater than or equal to constraint, we need to multiply by a minus sign.

Similarly, if we have equations like this f 1 x 1 plus 2 x 2 minus 3 is less than equal to 0, then we need to take this minus 3 to the right and side. Similarly, for equality constraint if it is x 1 plus x 2 plus x 3 minus 4 is equal to 0. Then we need to take this minus 4 to the right hand side, because that is the format which is required for linprog.

(Refer Slide Time: 30:51)



MATLAB code

Create a script file to solve the problem using *linprog*

```
clc; clear % Clear the command window and workspace respectively
Z = [1 2 -3]; % Coefficients of objective function
A = [1 2 0; 0 1 1]; % coefficients of inequality constraints
b = [3;2]; % RHS of inequality constraints
Aeq = [1 1 1]; % coefficients of equality constraints
beq = 4; % RHS of equality constraints
lb = [0 -1.5 0]; % Lower bound of decision variables
ub = [5 3 2]; % Upper bound of decision variables
% Calling the solver
[X, FVAL, EXITFLAG, OUTPUT, LAMBDA] = linprog(Z, A, b, Aeq, beq, lb, ub);
```

$Z=[1\ \ 2\ \ -3]$

$A=\begin{bmatrix}1 & 2 & 0\\ 0 & 1 & 1\end{bmatrix}$ $b=\begin{bmatrix}3\\ 2\end{bmatrix}$

$Aeq=[1\ \ 1\ \ 1]$ $beq=[4]$

$lb=[0\ \ -1.5\ \ 0]$ $ub=[5\ \ 3\ \ 2]$

Output:

```
X               [3.5  -1.5  2]
FVAL            -5.5
LAMBDA.lower    [0  1  0]     Shadow price for change in lower bound
LAMBDA.upper    [0  0  4]     Shadow price for change in upper bound
LAMBDA.ineqlin  [0  0]        Shadow price for change in RHS of inequality constraints
LAMBDA.eqlin    -1            Shadow price for change in RHS of inequality constraints
```

So, here we defined clc and clear to clear the command window and MATLAB workspace. We defined Z right as 1 2 minus 3 A and b as defined over here right. Aimilarly A quality and b equality this A quality and b quality the lower bound and upper bound. So, the lower bound and upper bound right and then we supply that information in the specified order right.

So, the first value has to be the coefficient vector, the second one has to be the coefficient matrix of the inequality constraint, third one has to be the right hand side value of the inequality constraint fourth and fifth one is about the equality constraint lower bound and then upper bound. Here we are solving with the default options of linprog, so we are not giving any options. So, the output we expect from this function is the values of the decision variable right. So, in this case it will be 3 values, because we have 3 decision variable. This will be a scalar value because this is the value of the objective function at this point.

So, that would be a scalar exitflag again will be a scalar it will tell us what is the reason for termination of linprog, output is a structure right which will have several fields which will tell various information on the solution of this problem using linprog and then we have the Lagrange multipliers right. So, Lagrange multiplier in this case since there are 3 decision variables.

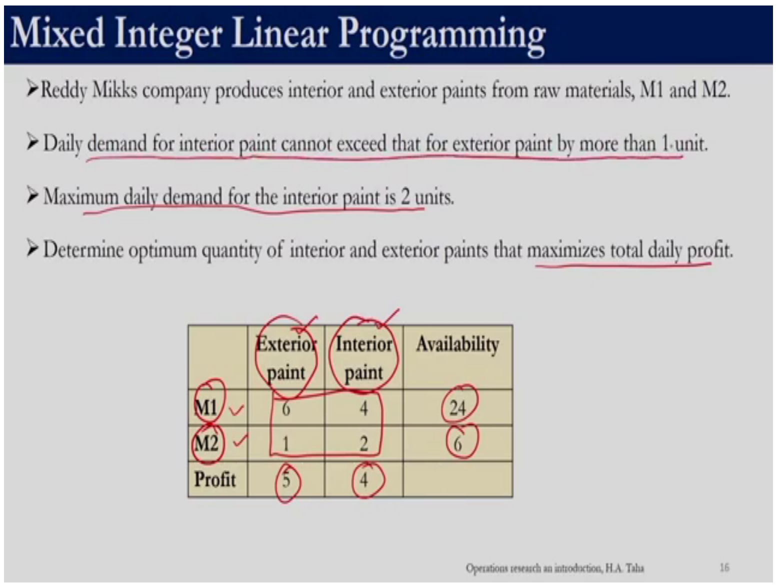
So, we will have lambda dot lower and lambda dot upper each of this will contain 3 values, because there are three decision variables. Lambda dot inequality will contain 2 values because, we have 2 inequality constraints and lambda dot eqlin right will contain only 1 value, because we have only 1 equality constraint. If we have 10 decision variables 2 inequality constraints and 5 equality constraint right.

So, then lambda dot lower and lambda dot approach each would have 10 values right, lambda dot inequality would have 2 values and lambda dot equality would have 5 values. This lambda denotes the Lagrange multiplier or shadow price right. So, we will discuss on what is the significance of this Lagrange multiplier with an example.

Now that we have seen how to solve a linear programming problem using linprog, let us move on to intlinprog. So, intlinprog is used to solve mixed integer linear programming problems, it can solve problems in which some of the decision variables are integer and some of the decision variables are continuous right. But still the objective function and the constraints need to be linear.

So, in integer linear programming as you might remember the objective function is linear, the constraints both it can support linear equalities and linear inequalities right. And it can also have bound constraints as well as integer variables right, so the function that we will be looking at this intlinprog.

(Refer Slide Time: 33:41)



Mixed Integer Linear Programming

- Reddy Mikks company produces interior and exterior paints from raw materials, M1 and M2.
- Daily demand for interior paint cannot exceed that for exterior paint by more than 1 unit.
- Maximum daily demand for the interior paint is 2 units.
- Determine optimum quantity of interior and exterior paints that maximizes total daily profit.

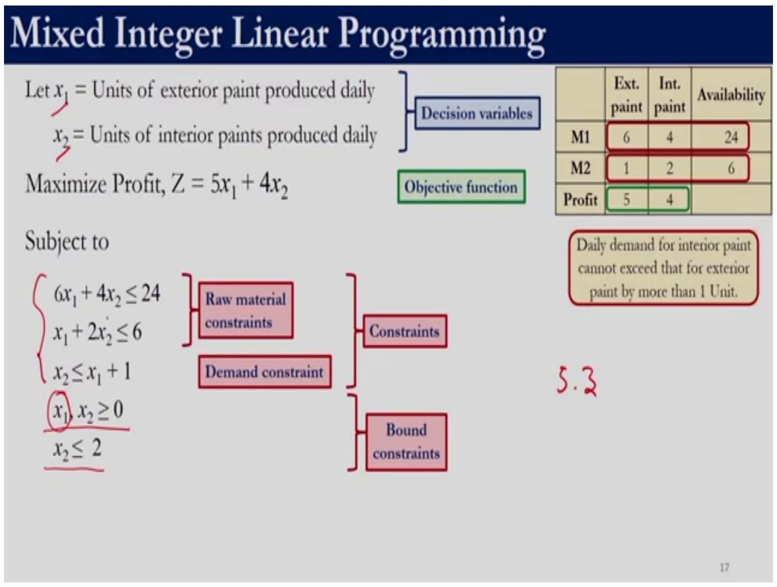
	Exterior paint	Interior paint	Availability
M1 ✓	6	4	24
M2 ✓	1	2	6
Profit	5	4	

Operations research an introduction, H.A. Taha 16

So, this is a problem which we have previously discussed the paint problem, that these are two raw materials M1 and M2 and 2 types of paints can be produced Interior paint and Exterior paint. So, these 4 values indicate the amount of raw material M1 and M2 required for manufacturing paint 1 and paint 2 right. And the availability of raw material M1 and M2 is 24 and 6. The profit realized by selling one unit of exterior paint is 5 the profit realized by selling 1 unit of interior paint is 4.

So, we need to decide on how many units of exterior paint and how many units of interior paint need to be manufactured. So, that we maximize the total daily profit right. There is an additional constraint that maximum daily demand for the interior paint is 2 units and the daily demand for interior paint cannot exceed that for exterior paint by more than 1 unit.

(Refer Slide Time: 34:33)



Mixed Integer Linear Programming

Let x_1 = Units of exterior paint produced daily

x_2 = Units of interior paints produced daily

Decision variables

Maximize Profit, $Z = 5x_1 + 4x_2$

Objective function

Subject to

$6x_1 + 4x_2 \leq 24$

$x_1 + 2x_2 \leq 6$

Raw material constraints

$x_2 \leq x_1 + 1$

Demand constraint

Constraints

$\boxed{x_1}, x_2 \geq 0$

$x_2 \leq 2$

Bound constraints

	Ext. paint	Int. paint	Availability
M1	6	4	24
M2	1	2	6
Profit	5	4	

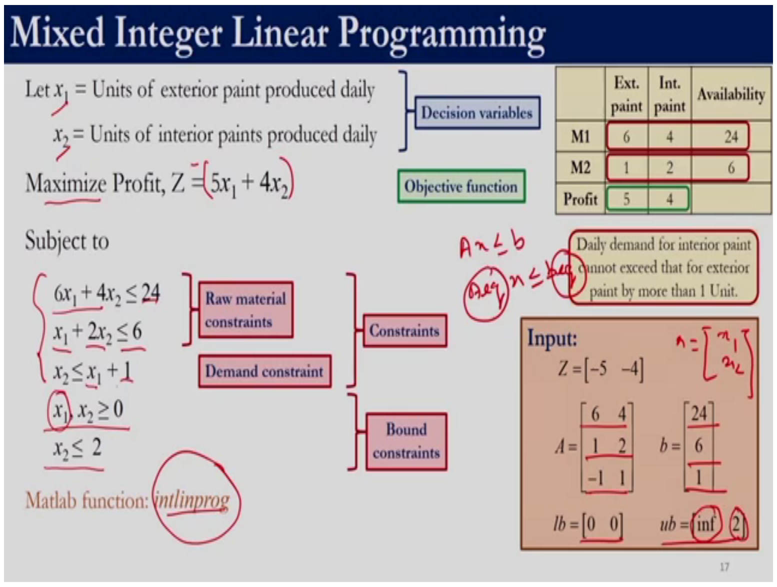
Daily demand for interior paint cannot exceed that for exterior paint by more than 1 Unit.

5.3

17

So, we had previously a model to this right, so we will not get into it again. So, these are our constraints right. We have 3 constraints and 2 decision variables right. In this case both the decision variables are integer, we want to determine the number of units right. So, 5.3 units is not a valid answer it has to be an integer, this x 1 and x 2 has to be an integer right. So, the lower bound is 0 for both the variables, where is the upper bound for x 2 is 2 there is no direct upper bound for x 1. The bound for x 1 will be determined by this constraints right.

(Refer Slide Time: 35:08)



Mixed Integer Linear Programming

Let x_1 = Units of exterior paint produced daily

x_2 = Units of interior paints produced daily

Decision variables

Maximize Profit, $Z = -(5x_1 + 4x_2)$

Objective function

	Ext. paint	Int. paint	Availability
M1	6	4	24
M2	1	2	6
Profit	5	4	

Subject to

$6x_1 + 4x_2 \le 24$

$x_1 + 2x_2 \le 6$

Raw material constraints

$x_2 \le x_1 + 1$

Demand constraint

Constraints

$x_1, x_2 \ge 0$

$x_2 \le 2$

Bound constraints

Matlab function: intlinprog

$Ax \le b$

$A_{eq}x \le b_{eq}$

Daily demand for interior paint cannot exceed that for exterior paint by more than 1 Unit.

Input:

$Z = [-5 \quad -4]$

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ -1 & 1 \end{bmatrix}$ $b = \begin{bmatrix} 24 \\ 6 \\ 1 \end{bmatrix}$

$lb = [0 \quad 0]$ $ub = [inf \quad 2]$

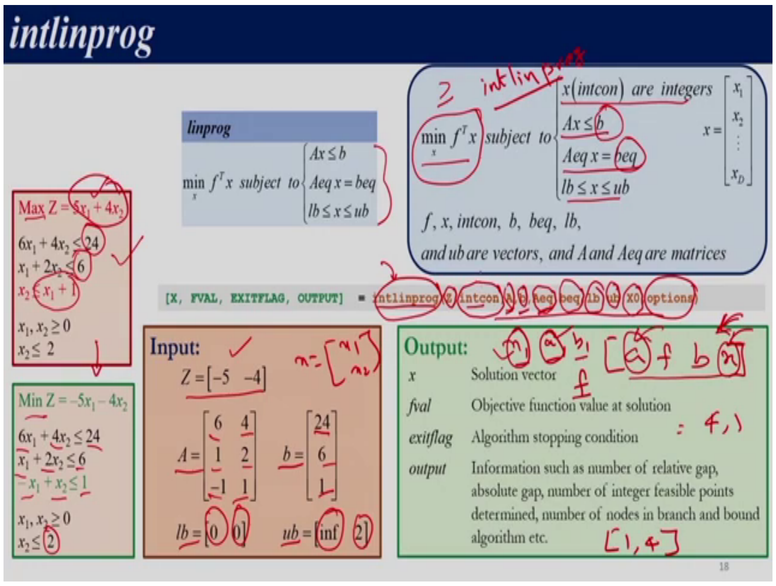
17

So, the MATLAB function which we are going to use here is intlinprog right. So, similar to linear programming we need to put it in the appropriate format right. So, this is a maximization problem and intlinprog similar to linprog solves only minimization problem.

So, we need to multiply this objective function by a negative sign and we need to only supply the coefficients right. So, the coefficient are minus 5 minus 4 right. So, when we write the coefficient in this order minus 5 minus 4, we have decided that our x vector will be x 1 x 2 that is when Z x will give us minus 5 x 1 minus 4 x 2. Similarly, we need to determine the coefficients of the inequality constraints right. So, here we have 3 inequality constraints, so 6 4 24 1 2 and 6 1 2 and 6 right. This x 1 has to be taken to the left hand side, because MATLAB supports constraints of the form A x is less than equal to b.

So, here we need to take x 1 to the left hand side, so we will have minus x 1 plus x 2. So, minus x 1 plus x 2 on the right hand side we will have 1 which is over here. The lower bound for both the variables are 0 0 and the upper bound for the first variable is infinity. So, infinity in MATLAB is given by inf and upper bound for the second variable is 2.

(Refer Slide Time: 36:24)



intlinprog

Max Z = 5x1 + 4x2
$6x_1 + 4x_2 \le 24$
$x_1 + 2x_2 \le 6$
$x_2 \le x_1 + 1$
$x_1, x_2 \ge 0$
$x_2 \le 2$

Min Z = –5x1 – 4x2
$6x_1 + 4x_2 \le 24$
$x_1 + 2x_2 \le 6$
$-x_1 + x_2 \le 1$
$x_1, x_2 \ge 0$
$x_2 \le 2$

linprog

$$\min_x f^T x \text{ subject to } \begin{cases} Ax \le b \\ Aeq\,x = beq \\ lb \le x \le ub \end{cases}$$

$$\min_x f^T x \text{ subject to } \begin{cases} x(intcon) \text{ are integers} \\ Ax \le b \\ Aeq\,x = beq \\ lb \le x \le ub \end{cases} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

$f, x, intcon, b, beq, lb,$ and ub are vectors, and A and Aeq are matrices

```
[X, FVAL, EXITFLAG, OUTPUT] = intlinprog(Z,intcon,A,b,Aeq,beq,lb,ub,X0,options)
```

Input:

$Z = [-5 \quad -4]$

$A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 24 \\ 6 \\ 1 \end{bmatrix}$

$lb = [0 \quad 0] \quad ub = [inf \quad 2]$

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of relative gap, absolute gap, number of integer feasible points determined, number of nodes in branch and bound algorithm etc.

18

So, this is what intlinprog solves right. So, minimization problem we need to provide the constraint in this form right. So, it does not support a constraint of the form greater than or equal to, it supports only of the form less than or equal to and we need supply the lower and upper bounds right.

So, if you compare this intlinprog with linprog right. So, this is for intlinprog, previously we had discussed linprog right. So, if you look at it is similar except for the fact that we also need to indicate the variables which need to take only integer values right. So, that is the difference between linprog and intlinprog. So, here we have converted that maximization problem into a minimization problem right and we have determined this input over here Z, A, b, lb ub. So, if you see the coefficient of this is 6 4 24 right 6 4 and 24. The second constraint 1 2 and 6 1 2 and 6 over here minus 1 1 minus 1 1 and again 1 on the right hand side, the lower bound of both the variables are 0.

So, that is why we have a 0, the upper bound is infinity for the first variables right and upper bound for the second variable is 2, that is what we have given right. So, here this transformation you might be knowing right. So, this is a maximization problem that is converted into a minimization problem, because intlinprog can solve only minimization problems right. And over here if you see on the right hand side we are expected to have only constraint coefficients right.

So, over here we have a constraint coefficient over here we have a constraint coefficient, but over here we do not have a constraint coefficient. So, x 1 on the right hand side has to be taken to the left hand side. So, that way we converted this problem into this standard format right. So, the input to intlinprog is similar to linprog right. So, this is the name of the inbuilt function intlinprog remember it is in lowercase.

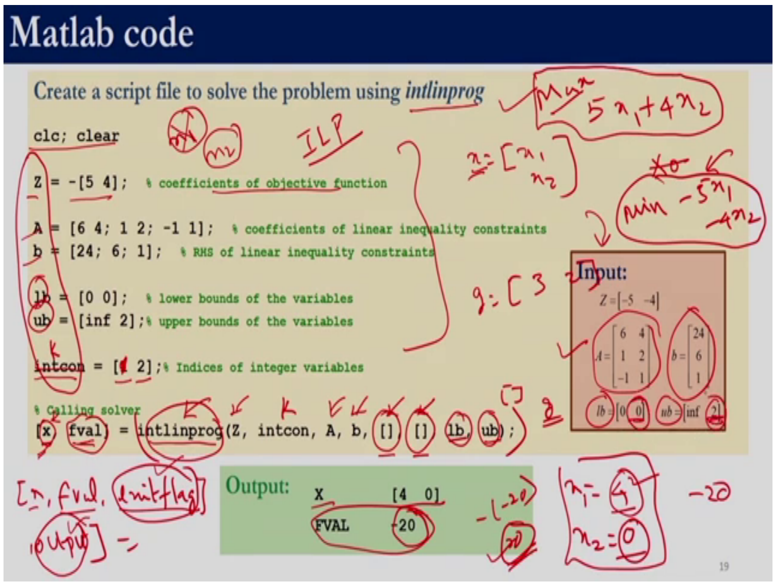
So, we need to give the variable which has the coefficient values of the objective function right. We will come to this intcon in a while right. So, this A b is the coefficient matrix of the inequality constraint, b is the right hand side. The vector of the inequality constraint A equality is the coefficient matrix of the equality constraint b equality is the right hand side of the equality constraint, lower bound upper bound of the variables X naught is the initial guess right.

So, for any problem if required we can supply the initial guess. So, that is a name of a variable X naught right and additionally we can give this options if we want to override any of the default options of intlinprog. So, much of it is similar to what we discussed for linprog right.

So, only difference is this intcon. So, over here for the intcon we need to specify the indices of the variables which are integer. Let us say I have 3 variables x, a, b and f for some reason let us say I have these 4 variables and I decide to arrange in this format a, f, b, x right, these are 4 decision variables in our optimization problem. And out of these 4 variables let us say a and x are integer variables right. So, we have four decision variable x a b f in this x and a are integer variables and let us say we have decided to arrange the variables like this right. So, first we have to decide the way in which we are going to arrange the deficient variable, only then we will be able to right coefficient matrix.

So, for example, over here this minus 5 minus 4 captures this 5 x 1 plus 4 x 2, if and only if x is x 1 and x 2. If it is x 2 and x 1 then it is minus 5 x 2 minus 4 x 1 which is not the case that we have. So, let us say we decide to arrange the variables as a f b x. Now, in this x and a are integer right. So, what does the position of x? So, the position of x is 4 and the position of a is 1 these two are the index of the 2 decision variables which are integer. So, in this case intcon has to be 1 comma 4. So, this will help the inbuilt function MATLAB to realize that the 1st variable and the 4th variable are integer. So, the 1st variable is a and the 4th variable is x right. So, that is what this intcon indicates, it indicates the indices of the integer variables.

(Refer Slide Time: 40:57)



Matlab code

Create a script file to solve the problem using *intlinprog*

```
clc; clear

Z = -[5 4];  % coefficients of objective function

A = [6 4; 1 2; -1 1];  % coefficients of linear inequality constraints
b = [24; 6; 1];  % RHS of linear inequality constraints

lb = [0 0];  % lower bounds of the variables
ub = [inf 2];% upper bounds of the variables

intcon = [1 2];% Indices of integer variables

% Calling solver
[x fval] = intlinprog(Z, intcon, A, b, [], [], lb, ub);
```

Input:

$Z = [-5 \ \ -4]$

$A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ -1 & 1 \end{bmatrix}$ $b = \begin{bmatrix} 24 \\ 6 \\ 1 \end{bmatrix}$

$lb = [0 \ \ 0]$ $ub = [\inf \ \ 2]$

Output:

```
X        [4   0]
FVAL     -20
```

So, now let us look at the script file for this problem right. So, this is the problem which we have we do not have any equality constraint, we are not going to give any initial guess right. So, you know what a clc and clear right. So, we define the variable Z as minus of 5 and 4 right. So, this is the coefficient of the objective function and then we use this variable A and this variable b to define these values. And then we use the variable lb and ub to define the lower and upper bound which we have for this current problem. Remember all these are names of variables, so it can be anything and appropriately that variable name has to be used over here right.

So, it is not necessary that this variable should always be intcon, let us say this variable is k right, then we will have to appropriately use k over here right. So, here in this case we have 2 variables both of them are integer right, so here we give 1 and 2. So, so far we have defined the necessary variables which can help us to define our problem right. So now, we are solving it with intlinprog. So, for intlinprog we need to provide Z intcon or k A b we do not have any equality constraint right. So, both of those would be empty brackets right and then we need to give lower bound and upper bound.

So, beyond upper bound we do not need to give anything because we do not have. Since we do not have an initial guess in this case, we do not need to give empty brackets, because beyond ub we are not using anything right. Over here we had to give this empty bracket because we are still using this variable lb and ub, which can be defined only after the equality constraints are defined right. So, that is why we were forced to give this empty brackets over here.

So, the output of this function would be x and fval the size of x will be the same as the size of lower bound or the upper bound right. So, we have two variables in this case, so x will contain 2 values; the first value will be x 1 the second value will be x 2, because that was the convention we had followed to define all these values right. So, all these values are correct only if x is in this format right.

So, the result will also be in the same format. So, there will be two values of x the first one will correspond to the optimal value of x 1 and second one will correspond to the value of the objective function at this particular x right. So, in this case we get x as 4 and 0 right. So, that is x 1 is equal to 4 and x 2 is equal to 0 right. So, our objective function was 5 x 1 plus 4 x 2 this was to be maximized right and what we did was minimize minus 5 x 1 minus 4 x 2 right.

So, if you substitute this 4 and 0 in this expression because this is what was given to intlinprog. So, what we will get is minus 20, so that is nothing but this fval right. So, but our problem was maximization right. So, the profit is whatever value we need to get we need to multiply it by a minus sign, because we have converted the maximization problem into a minimization problem.

So, now the answer whatever is given has to be multiplied with minus 1 to get the objective function value of the maximization problem. So, in this case it will be minus of minus 20 so which is 20 and the amount of exterior paint we need to generate is 4 and amount of interior paint that we need to generate a 0 right. You can check this solution also satisfies the constraints right.

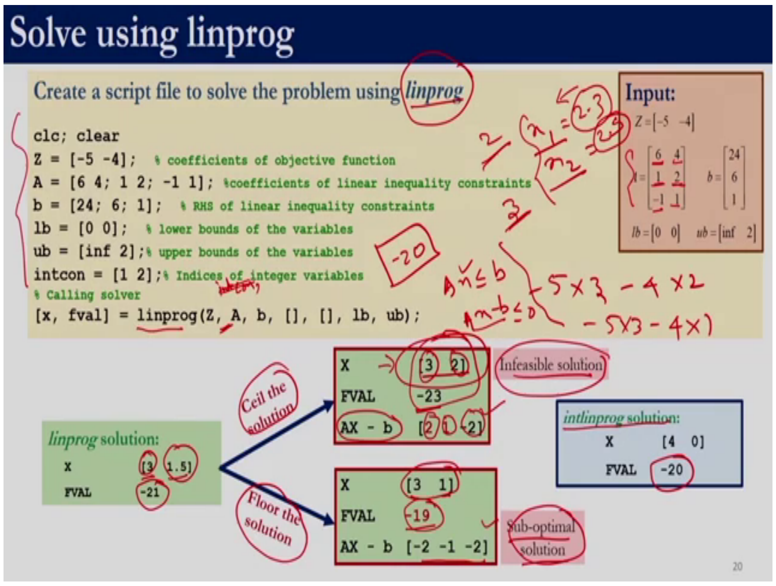
So, over here we did not use exitflag right. So, what we could have done is instead of just stopping with x comma fval, we could have given x comma fval comma exitflag right equal to this right hand side as such right. So, in this case we would have given us the value of x the value of fval and also it would have given us a number with which we can find out the reason for termination of the algorithm. So, as in when intlinprog terminator right, what was the reason for which it terminated did it find the optimal solution and they did terminate or did we reach any resource limitations or it terminated for any other reason right. So, that can be determined using this exitflag.

You can also give x comma fval comma exitflag comma output. So, again if you look into this output variable it would be a structure, it would contain additional information about the solution of this problem using intlinprog. So, we leave it to you to explore what is exitflag and what is the information that you get from output. So, for this problem let us say we had an initial guess of let us say 3 comma 2. So, then what we could have done is we could have set g is equal to 3 and 2 and over here lb comma ub comma g could have given right. So, this way you can use intlinprog to solve on mixed integer linear programming problem.

So, the example which we showed is actually ILP, we had two variables both of them were integer variables. But it is not necessary that all the variables need to be integer for us to use intlinprog, even for this problem let us say x 1 was not integer only x 2 was integer right. So, in that case what we could have done is this intcon we could have defined it has just 2. So, in that case x 1 will be treated as a continuous variable between 0 and infinity and x 2 would be treated as a integer variable which can take either the value of 0 1 or 2, because the lower bound over here is 0 and upper bound over here is 2.

So, the question is what would have happened if we had solved this problem with linprog right and whatever values we had got for the decision variables what if he had rounded it of. So, let us solve the same problem by linprog right.

(Refer Slide Time: 46:50)



Solve using linprog

Create a script file to solve the problem using *linprog*

```
clc; clear
Z = [-5 -4];  % coefficients of objective function
A = [6 4; 1 2; -1 1]; %coefficients of linear inequality constraints
b = [24; 6; 1];  % RHS of linear inequality constraints
lb = [0 0];  % lower bounds of the variables
ub = [inf 2];% upper bounds of the variables
intcon = [1 2];% Indices of integer variables
% Calling solver
[x, fval] = linprog(Z, A, b, [], [], lb, ub);
```

Input:

$Z = [-5 \ \ -4]$

$A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 24 \\ 6 \\ 1 \end{bmatrix}$

$lb = [0 \ \ 0] \quad ub = [\text{inf} \ \ 2]$

linprog solution:

```
X      [3 1.5]
FVAL   -21
```

Ceil the solution

```
X        [3  2]
FVAL     -23
AX - b   [2 1 -2]
```

Infeasible solution

Floor the solution

```
X        [3  1]
FVAL     -19
AX - b   [-2 -1 -2]
```

Sub-optimal solution

intlinprog solution:

```
X      [4  0]
FVAL   -20
```

20

So, linprog; so remember we have been told that x 1 and x 2 are integer variables right. But still we choose it to solve with linprog, we are under the assumption. That we can solve it using linprog and then whatever values we get we will round it of right. So, let us say x 1 if we get 2.3 and x 2 if we get 2.9, then we will round off the solution as 2.3 or 2 2 and 2.9 as 3. So, very often this is a misconception that this can be done right. So, for this problem let us see what will happen, if we solve it linprog right.

So, this piece of code you will be able to understand, just instead of intlinprog we are using linprog and over here we are not defining that intcon right. So, otherwise we would have given the intcon over here between Z and A. So, here we are not giving linprog that is not required right. So, if we solve this problem right, so the value of x 1 we would get this 3 and the value of x 2 is 1.5 right and the objective function value is minus 21.

So, minus 21 is actually better than minus 20 right, which we had got a using intlinprog right. So, the only thing is that these 2 values are not integers. So, we cannot use these values, so let us see what will happen if we round this off. So, let us say we decide to ceil this solution right 3 and 1.5 we decided to ceil this solution. So, if we ceil the solution, we will get 3 and 2 and the objective function value corresponding to it would be minus 5 into 3 minus 4 into 2 right. So, this if you workout it will come out to minus 23 right. So, by ceiling we get a better solution right.

But the question is this solution a feasible solution, does it satisfy all the constraint right. So, what we can do is we can substitute this value of x 1 and x 2 into these constraints right. So, what we can do is we can find out this A x minus b, remember our constraint is A x is less than equal to b. Now, we have x given by this three two we have this A.

So, we multiply A with x minus B should be less than equal to 0. So, this condition has to be satisfied. But if you plug in this x with this A matrix and find out A x minus b you will see that you will get to 1 and minus 2. So, the all these 3 values are supposed to be less than equal to 0, but only the last value is less than equal to 0 the first two values are positive numbers right.

So that means, this solution 3 and 2 which has a better objective function right, it is actually an infeasible solution right. So, ceiling the solution can give a better value of the objective function. But then it may lead to infeasibility. So, here the question is we ceil the solution what happens if we take the floor of the solution. So, the floor of the solution is 3 and 1. If we calculate the objective function for 3 and 1 that is minus 5 into 3 minus 4 into 1, so this will be minus 19 right.

Again, we can calculate for this solution A x minus b. So, in this case all the 3 values are less than 0 right. So, this is a feasible solution, but it has an objective function of minus 19. If you remember from intlinprog we had got a solution of minus 20 right. Whereas if we use linprog and then floor the solution in this case we end up with a suboptimal solution right, so that is why it is necessary to use the appropriate function for the problem at hand.

If there are no integer variables and if there are only continuous variables, then it is better to solve it using linprog. If there are integer variables and the constraints are linear right, then we should use intlinprog right ceiling of the solution. And the flooring the solution may either take us to a suboptimal solution or it can lead to a infeasible solution. So, with that we will conclude the session. In this session we have looked into how to solve linear programming problems as well as how to solve mixed integer linear programming problems. For linear programming problems we use linprog program and for mixed integer linear programming problems we use intlinprog.

So, we have not looked into few things such as how to change the options as well as how to interpret the lambda values, that we will be doing subsequently. As that part is common to most of the inbuilt functions which we use for solving optimization problems in MATLAB, with that we conclude this session.

Thank you.