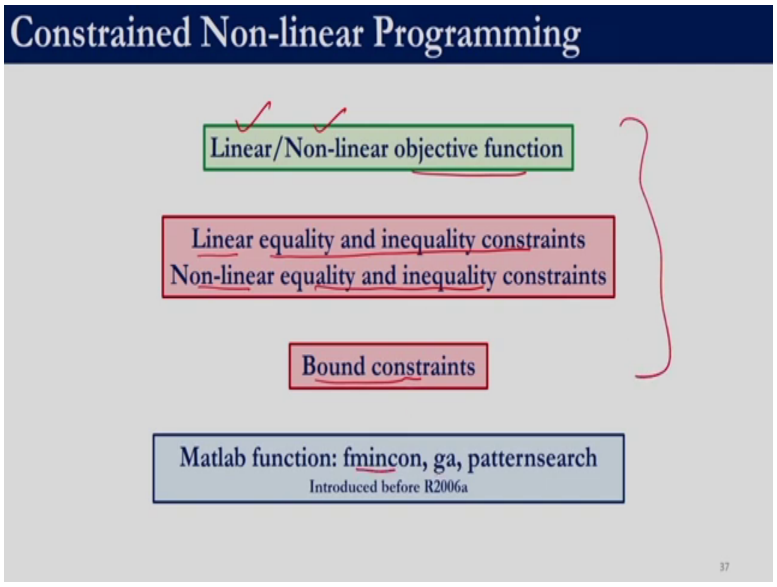
Computer Aided Applied Single Objective Optimization
Dr. Prakash Kotecha
Department of Chemical Engineering
Indian Institute of Technology, Guwahati

Lecture – 28
MATLAB Inbuilt Functions: Nonlinear & Mixed Integer Nonlinear Programming

Welcome back. In the previous session, we had looked into how to solve linear programming problems and integer linear programming problems. In this session, we will be looking into how to solve Non-linear programming problems as well as Mixed Integer linear Programming problems.

(Refer Slide Time: 00:45)



Constrained Non-linear Programming

Linear/Non-linear objective function

Linear equality and inequality constraints
Non-linear equality and inequality constraints

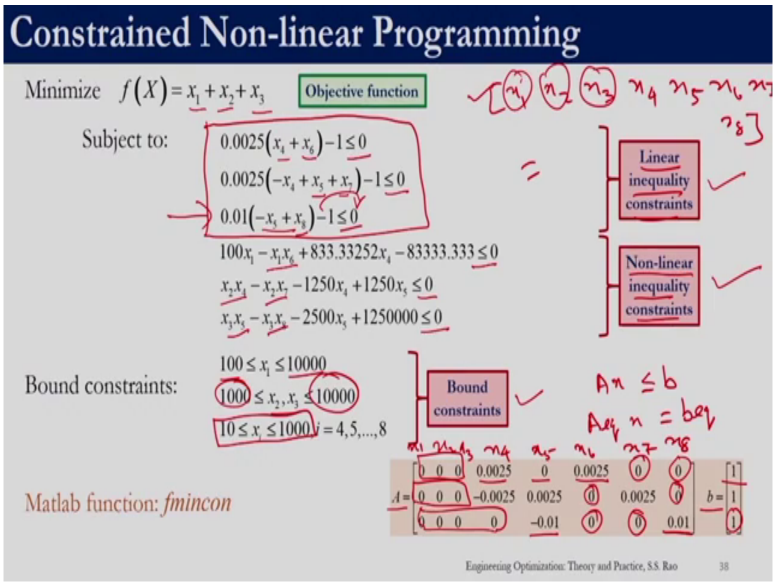
Bound constraints

Matlab function: fmincon, ga, patternsearch
Introduced before R2006a

37

So, for non-linear constrained optimization the objective function can be either linear or non-linear. There can be equality and inequality constraints these constraints can either be linear or non-linear right. And there are bound constraints that is the decision variables are bounded by a lower and upper bound right. The functions which can be used to solve these problems are fmincon, ga, and pattern search.

(Refer Slide Time: 01:12)



Constrained Non-linear Programming

Minimize $f(X) = x_1 + x_2 + x_3$ Objective function

$[x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8]$

Subject to:

$$0.0025(x_4 + x_6) - 1 \le 0$$
$$0.0025(-x_4 + x_5 + x_7) - 1 \le 0$$
$$0.01(-x_5 + x_8) - 1 \le 0$$

Linear inequality constraints

$$100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0$$
$$x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0$$
$$x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0$$

Non-linear inequality constraints

Bound constraints:

$$100 \le x_1 \le 10000$$
$$1000 \le x_2, x_3 \le 10000$$
$$10 \le x_i \le 1000\ i = 4,5,\ldots,8$$

Bound constraints

$Ax \le b$

$A_{eq}x = b_{eq}$

Matlab function: *fmincon*

$$A = \begin{array}{cccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ \hline 0 & 0 & 0 & 0.0025 & 0 & 0.0025 & 0 & 0 \\ 0 & 0 & 0 & -0.0025 & 0.0025 & 0 & 0.0025 & 0 \\ 0 & 0 & 0 & 0 & -0.01 & 0 & 0 & 0.01 \end{array} \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Engineering Optimization: Theory and Practice, S.S. Rao 38

So, the example that we will take to solve fmincon is objective function is x 1 plus x 2 plus x 3, so the objective function is linear right. We have 3 inequality constraints right and all of these 3 inequality constraints are linear. So, the decision variables are x 1, x 2, x 3, x 4, x 5, x 6, x 7, and x 8 right. So, these are linear inequality constraints, we also have non-linear inequality constraints.

So, over here these three constraints are less than equal to 0, over here if we see we have this non-linear terms over here right and we have bound constraints over here. So, the lower and upper bound of x 1 is given over here, the lower and upper bound of the variables x 2 and x 3 are same as given over here. The lower bounds of the 4th variable, 5th variable, 6th, 7th, and 8th variable is given over here right.

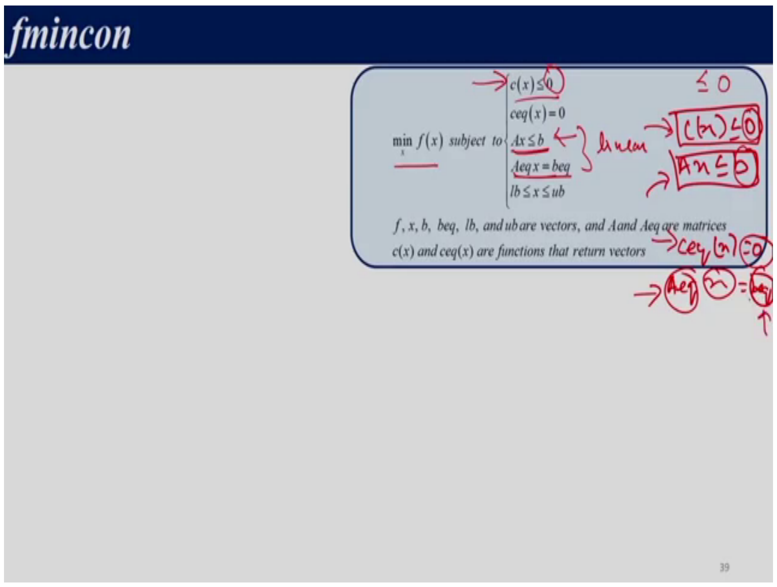
So, we have inequality linear constraints, we have inequality non-linear constraints and we also have bound constraints. So, in this example we do not have equality constraints right. Similar to the other functions first we need to decide as to how we are going to arrange the variables right. So, we will take this pattern x 1, x 2, x 3, x 4, x 5, x 6, x 7, and then x 8. So, this is how we are going to stack the variables right. So, this is important to define this convention because only based on this convention, we will be able to write the variables which are required.

So, these three linear inequalities we can write it like this as given over here right, so the matrix A the variable x 1, x 2, x 3 are not in these equation right. So, we have 3 0's right, the coefficient of x 4 is 0.0025 the coefficient of x 5 is 0 because it does not appear in this equation. The coefficient of x 6 the coefficient of x 6 is 0.0025 the coefficient of x 7 and x 8 is 0 and 0 right.

So, similar to the other functions which we have discussed previously this minus 1 can be brought to the right hand side right, so if we bring this minus 1 to the right hand side it will be 1. So, fmincon requires the constraints to be of this form A x is less than equal to b, and A equality x is equal to b equality right. So, that is why we have brought this to the right hand side. So, for the second constraint if we see again the variables 1, 2, 3 are not there right 4, 5 and 7, so 6 over here is 0.

So, here if you want you can write x 1, x 2, x 3, x 4, x 5, x 6, x 7, and x 8. So, the coefficient of x 6 in second equation is 0 similarly the coefficient of x 8 is 0. In the final equation only 5 and 8 appear right, so 1, 2, 3, 4 are 0. The coefficient of 5 is minus 0.01, the coefficient of x 8 is 0.01 right this 1 is again to be taken to the right hand side. So, we have one over here, this equation does not involve x 6 and x 7, so we have 0 over here. So, this is how we define the coefficient matrix for this problem right.

(Refer Slide Time: 04:15)



fmincon

$$\min_x f(x) \text{ subject to } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ Ax \le b \\ Aeq\,x = beq \\ lb \le x \le ub \end{cases}$$

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

$c(x)$ and $ceq(x)$ are functions that return vectors

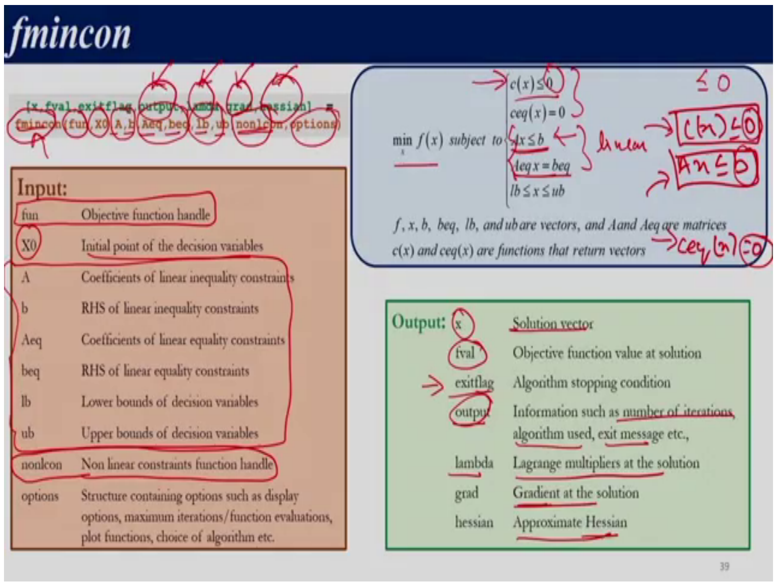
≤ 0
linear
c(x) ≤ 0
Ax ≤ b
ceq(x) = 0
Aeq (x) = beq
39

As we discuss fmincon requires the problem to be in this form. This we have previously discussed inequality and equality this denotes linear constraints right. So, non-linear constraints should be of this form, non-linear inequalities should have less than equal to form and the right hand side has to be 0 for non-linear constraints right. This is also inequality this one is also inequality right.

So, let me just write C of x less than equal to 0 and A x less than equal to b. So, this one is for linear and this one is for non-linear. So, for linear inequalities the right hand side should be a constant value right whereas, for non-linear inequalities the right hand side should be 0. Similarly, over here if you see C equality of x equal to 0 and A equality. So, the name of the variable is Aeq into x.

So, this is a separate variable this is a separate variable should be equal to b equality again this is this the name of a variable right. So, this is again for linear this is for non-linear. So, for non-linear equality constraint should have 0 as their right hand side whereas, for linear equality constraint the right hand side should be a constant right. It is not necessary that this b equality would be 0.

(Refer Slide Time: 05:34)



fmincon

```
[x,fval,exitflag,output,lambda,grad,hessian] =
fmincon(fun,X0,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

$$\min_x f(x) \text{ subject to } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ Ax \le b \\ Aeq\,x = beq \\ lb \le x \le ub \end{cases}$$

f, x, b, beq, lb, and ub are vectors, and A and Aeq are matrices
c(x) and ceq(x) are functions that return vectors

Input:

fun	Objective function handle
X0	Initial point of the decision variables
A	Coefficients of linear inequality constraints
b	RHS of linear inequality constraints
Aeq	Coefficients of linear equality constraints
beq	RHS of linear equality constraints
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
nonlcon	Non linear constraints function handle
options	Structure containing options such as display options, maximum iterations/function evaluations, plot functions, choice of algorithm etc.

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.,
lambda	Lagrange multipliers at the solution
grad	Gradient at the solution
hessian	Approximate Hessian

39

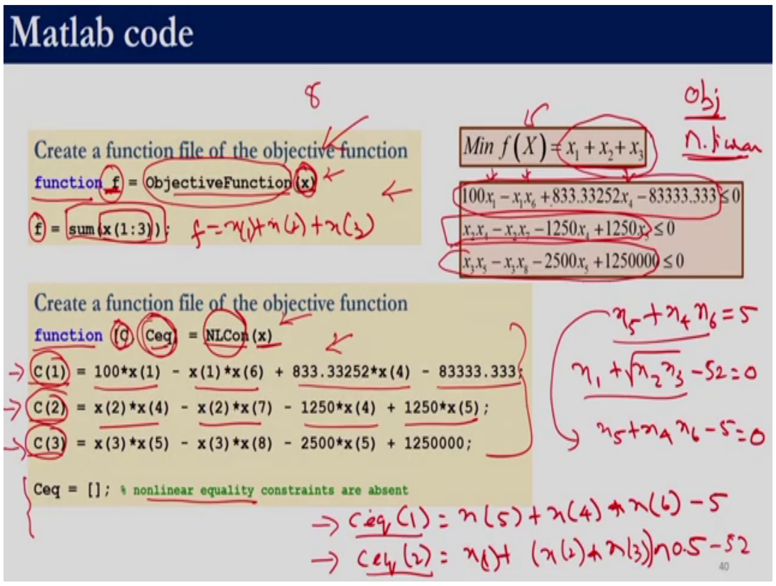
fmincon is the inbuilt function right. So, fun can be a function handle right. So, we can write the non-linear objective function in a function file and give the name of the function file over here. So, fun is an objective function handle right, x naught is the initial point of the decision variable. So, for non-linear problems we are supposed to give x naught right.

So, this A b A equality, b equality, lower bound, upper bound is similar to what we have discussed previously right. The coefficient matrix and the right hand side vector of the inequality constraint the coefficient matrix and the right hand side of the equality constraint lower and upper bound right.

So, these non-linear constraints can be written in a function file and that handle can be given as input right over here. And then if we want to override any of the default options of fmincon, we can provide the options over here right. So, depending upon what are we interested in the output of the function can be the solution vector fval; fval is the value of the objective function at this solution vector exit flag as usual we will tell us the reason for the stopping of the algorithm right.

The variable output is a structure right, it will have lot of information such as number of iteration used, which is the algorithm use, the exit flag which indicates the reason for termination. It can also give us the gradient at the solution as well as approximate hessian right. And lambda is the Lagrange multipliers at the solution.

(Refer Slide Time: 07:11)



Matlab code

Create a function file of the objective function

```
function f = ObjectiveFunction(x)
f = sum(x(1:3));
```

$Min\ f(X) = x_1 + x_2 + x_3$

$100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0$

$x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0$

$x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0$

Create a function file of the objective function

```
function [C, Ceq] = NLCon(x)
C(1) = 100*x(1) - x(1)*x(6) + 833.33252*x(4) - 83333.333;
C(2) = x(2)*x(4) - x(2)*x(7) - 1250*x(4) + 1250*x(5);
C(3) = x(3)*x(5) - x(3)*x(8) - 2500*x(5) + 1250000;

Ceq = []; % nonlinear equality constraints are absent
```

So, in this case our objective function is x 1 plus x 2 plus x 3 right. So, what we will do is, we will create a function file function f is equal to the name of the file we have chosen it as objective function you can choose whatever you want right. And the input to this objective function is x right; x is the set of decision variables right. And what we are returning from this function is the value of the objective function. So, the objective function in this case is x 1 plus x 2 plus x 3.

So, what this function is supposed to do is it will receive the values of all the decision variables. So, for example, the problem that we are solving has 8 decision variables, but the objective function has only 3 decision variables right. But, still if you look at this x it will have 8 values. All the values of the decision variable will be passed on by the algorithm to this objective function file and this file is to evaluate the value of the objective function.

So, in this case we could have just written x of 1 plus x of 2 plus x of 3 is equal to f right or we could have written this the variables x 1, 2, 3 we are only accessing that and then we are summing it up right. That is being stored in the variable f and this is the variable which is getting passed on right. Let us say our objective function was x 1 square plus sin of x 2 right. So, in this case f will be x of 1 square plus sin of x of 2. So, whatever is the nonlinearity that can be captured in this objective function.

So, now we will create another function file for the non-linear constraints right. So, remember the non-linear objective and the non-linear constraints can be passed through a function files right. In this case we are again creating a function file whose name is NLCon right this is a name that we have chosen NLCon right. So, this non-linear constraint file can receive the decision variables the values of the decision variable and it is supposed to return 2 variables C and C equality right.

In this case we have chosen variable names as C equality, but you can choose any variable right. C is supposed to be the value of the expression on the left hand side. So, in this case the left hand side expression is this one right 100 x 1 minus x 1 x 2 so on, so that is the expression. So, if we substitute the values of x 1, x 1, x 6, x 4 in this equation what would be this left hand side right. So, that is supposed to be returned.

So, that is true for all the inequality constraints that we have. So, in this case we have 3 inequality constraint, so C of 1, C of 2, and C of 3 right. So, we can pass 2 variables C and C equality right. So, the first variable C is for the inequality constraints. So, we have 3 inequality constraints, so we will have 3 values of C C of 1, C of 2, and C of 3. So, C of 1 the expression on the left hand side of the equation is 100 into x of 1 right minus x 1 into x 6 plus 833.33252 x of 4 minus 83333.333 right, so that is the first constraint.

The second constraint is x 2 into x 4 minus x 2 into x 7 minus 1250 x 4 plus 1250 x 5 right. Similarly, you can write the third constraint also. So, whatever nonlinearities are there in the non-linear inequality constraints can be captured through this function file right. So, here in this case we did not have equality constraint, let us say we had these two equality constraint x

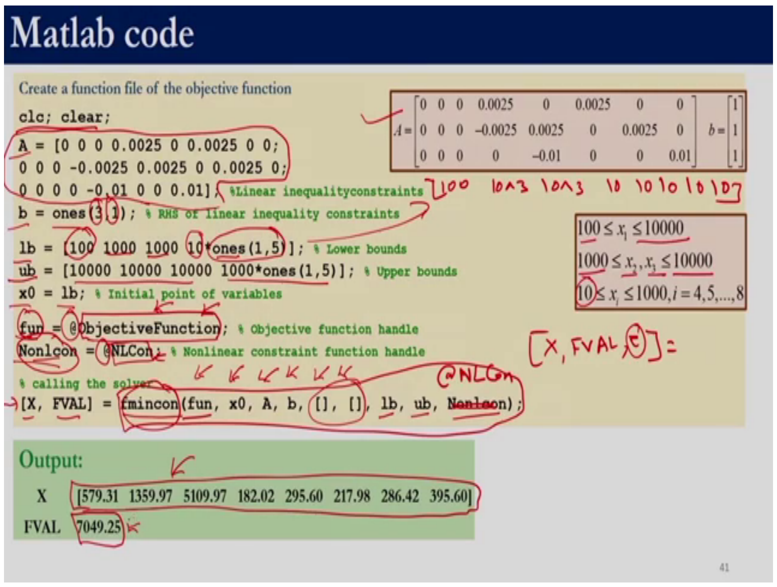
5 plus x 4 into x 6 equal to 5. And let us say we had another equality constraint x 1 plus root of x 2, x 3 minus 52 is equal to 0.

Let us say if we had these 2 equality constraint right, remember for the function fmincon right; all the equalities involving nonlinearities right. So, this is also nonlinearities this is also a non-linear constraint the right hand side has to be 0. So, first of all we need to convert this one into the appropriate form which will be x 5 plus x 4 x 6 minus 5 equal to 0. The second constraint is already in the same form right. So, here what we will do is C equality of 1 is x of 5 plus x of 4 into x of 6 minus 5.

C equality of 2 is x 1 plus x of 2 into x of 3 to the power point 5 right minus 52. If we had non-linear equalities, we could have defined C equality of 1 and C equality of 2 over here. But, since in this current problem we do not have non-linear equality constraint we just give empty bracket over here. So, here for the objective function file as well as the non-linear constraint file you need to remember that the input would be the decision variables right.

Does not matter how many decision variables are used over here the values of all the decision variables will be passed to these two functions by the algorithm right. And what the algorithm expects is the value of the objective function f right and the value of the expression on the left-hand side of the inequality constraint and the equality constraints. Now, that we have defined both the files which we will need we can now solve the problem right.

(Refer Slide Time: 13:02)



Matlab code

Create a function file of the objective function

```
clc; clear;
A = [0 0 0 0.0025 0 0.0025 0 0;
0 0 0 -0.0025 0.0025 0 0.0025 0;
0 0 0 0 -0.01 0 0 0.01]; %Linear inequalityconstraints
b = ones(3,1); % RHS of linear inequality constraints

lb = [100 1000 1000 10*ones(1,5)]; % Lower bounds
ub = [10000 10000 10000 1000*ones(1,5)]; % Upper bounds
x0 = lb; % Initial point of variables

fun = @ObjectiveFunction; % Objective function handle
Nonlcon = @NLCon; % Nonlinear constraint function handle

% calling the solver
[X, FVAL] = fmincon(fun, x0, A, b, [], [], lb, ub, ~~Nonlcon~~);
```

$$A = \begin{bmatrix} 0 & 0 & 0 & 0.0025 & 0 & 0.0025 & 0 & 0 \\ 0 & 0 & 0 & -0.0025 & 0.0025 & 0 & 0.0025 & 0 \\ 0 & 0 & 0 & 0 & -0.01 & 0 & 0 & 0.01 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

[100 10^3 10^3 10 10 10 10 10]

$100 \le x_1 \le 10000$

$1000 \le x_2, x_3 \le 10000$

$10 \le x_i \le 1000, i = 4,5,\ldots,8$

[X, FVAL, c] =

@NLCon

Output:

```
X      [579.31  1359.97  5109.97  182.02  295.60  217.98  286.42  395.60]
FVAL   7049.25
```

41

So, clc clear is took just clear the command window and the MATLAB workspace right. So, this we have discussed A which comes from the linear inequalities of this problem right, so that is what is defined over here right. So, this defines we have 3 constraints right, so we will have 3 rows and will have 8 variables, so there would be 8 columns. So, this is the coefficient matrix of the linear inequalities.

This is the right hand side vector of the linear inequalities right. So, here we have 3 1s. So, once is an inbuilt function of MATLAB right we want 3 rows and one column right, so this will give 1 1 1 right. So, the lower bound for the variable is the first variable lower bound is 100, so that is given over here. The second and the variable have a lower bound of 1000, so 1000 1000 and the rest of the 5 variables have a lower bound of 10. We could have chosen to write 100 right 10 power 3 twice and then 10 5 times right.

So, this is the upper bound right or we could just use this functions ones of MATLAB. So, we want one row 5 columns this will give 5 columns with the value of one right we are multiplying it by 10 right. So, we will ultimately end up getting this one right. Similarly, we defined the upper bound, so the first 3 variables have the same upper bound and the last 5 variables have an upper bound of 1000

So, for this problem we are taking the initial guess x naught is equal to the lower bound itself right, we are defining these 2 variables which are function handles. So, the function which we wrote for the objective function is objective function and the function that we wrote for non-linear constraint is NLcon right. So, you need to have this at the rate symbol over here. So, that whenever this variable is accessed right we are actually accessing this function file.

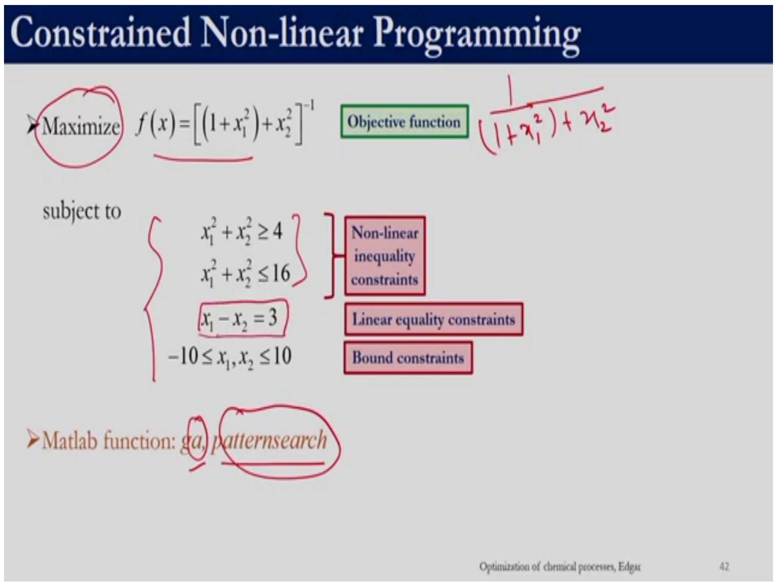
So, till now we have only defined whatever is require to solve the problem. So, in this line we call the function fmincon we provide in the same order first the objective function right so, fun; then the initial point the coefficient matrix and the right hand side of the linear inequalities. The coefficient matrix and the right hand side of the equalities in this case we do not have them, so we are giving empty brackets right.

So, the lower bound, the upper bound, and the file for the non-linear constraints right. If we do not want to use this variable, we can directly give at NLCon. So, we are directly providing the name of the file; similarly, instead of fun we could have directly provided the name of the file right. So, when we solve this, we will get the value of the decision variable. So, these are the values of the 8 variables right, and this is the objective function value corresponding to this solution right.

So, now that we have got this solution we expect you to go back and execute this program and find out what is the reason for termination as and when it terminated what was the reason for termination right. So, you need to give X comma FVAL comma exitflag right. So, let us say you give this variable e and then use this same part right it will give you a number right. So, look at the help of fmincon and see the reason for which the algorithm got terminated right.

And also we expect you to have a look at whether this solution satisfies all the constraints right. Similarly, you can have a look at the output, the values of lambda, the gradient, and the hessian right, so that we leave it to you to find out. Now, we look into ga and patternsearch right both ga and pattern search can solve non-linear programming problems right.

(Refer Slide Time: 16:58)



Constrained Non-linear Programming

- Maximize $f(x) = \left[\left(1+x_1^2\right)+x_2^2\right]^{-1}$ Objective function $\frac{1}{(1+x_1^2)+x_2^2}$

subject to

$$x_1^2 + x_2^2 \geq 4$$
$$x_1^2 + x_2^2 \leq 16$$
Non-linear inequality constraints

$$x_1 - x_2 = 3$$
Linear equality constraints

$$-10 \leq x_1, x_2 \leq 10$$
Bound constraints

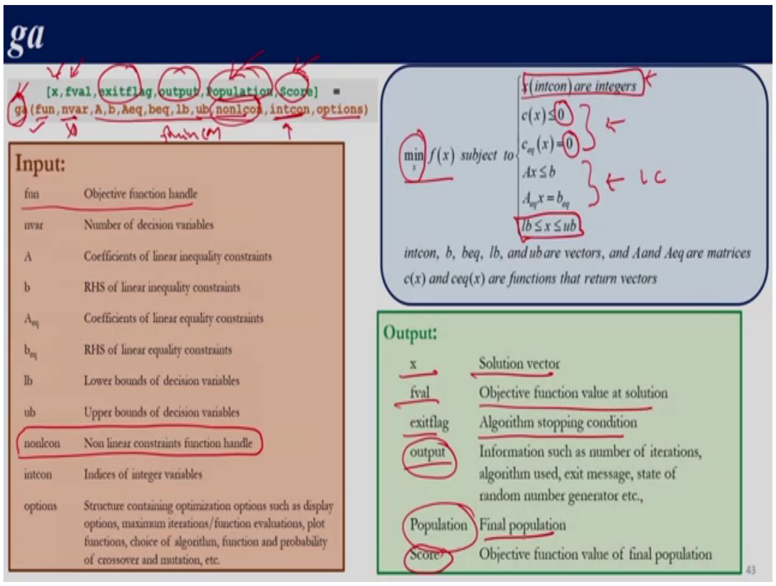
- Matlab function: ga, patternsearch

Optimization of chemical processes, Edgar 42

So, for ga and patternsearch we will use this example right. There is a reason that we have chosen a new problem to demonstrate ga and patternsearch ideally we could have used the same previous example to demonstrate ga and patternsearch right. So, there is a reason why we are not doing it at the end of this session you would know how to solve a non-linear programming problem with ga and patternsearch right.

What we expect you is to solve the previous problem right? The problem which we use for fmincon take the same problem and try to solve with ga and patternsearch and let us know your experience about solving that problem with ga and patternsearch right. So, here the objective function we have is a maximization problem right objective function is a non-linear function right. So, the objective function is 1 by 1 plus x 1 square right plus x 2 square right. We have 3 constraints 2 of them are non-linear right and they are inequality constraint this is a linear equality.

(Refer Slide Time: 17:57).



ga

```
[x,fval,exitflag,output,Population,Score] =
ga(fun,nvar,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

$$\min_x f(x) \text{ subject to } \begin{cases} x(\text{intcon}) \text{ are integers} \\ c(x) \le 0 \\ c_{eq}(x) = 0 \\ Ax \le b \\ A_{eq}x = b_{eq} \\ lb \le x \le ub \end{cases}$$

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

c(x) and ceq(x) are functions that return vectors

Input:

fun	Objective function handle
nvar	Number of decision variables
A	Coefficients of linear inequality constraints
b	RHS of linear inequality constraints
A_{eq}	Coefficients of linear equality constraints
b_{eq}	RHS of linear equality constraints
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
nonlcon	Non linear constraints function handle
intcon	Indices of integer variables
options	Structure containing optimization options such as display options, maximum iterations/function evaluations, plot functions, choice of algorithm, function and probability of crossover and mutation, etc.

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, state of random number generator etc.,
Population	Final population
Score	Objective function value of final population

43

So, first we look into the function ga right. It can solve only minimization problems right. So, the problem which we have is a maximization problem. So, first thing is that we will have to convert into a minimization problem the non-linear constraints and the linear constraints are to be provided in the same form as we did for fmincon right.

So, for fmincon also for linear constraints these 2 are linear constraints, the right hand side should be a constant value the left hand side will not have any constant value. Whatever the constant value is there on the left hand side has to be brought to the right hand side that is for linear constraints. For non-linear constraints whatever constant terms appear on the right hand side they need to be brought to the left hand side right. The values should be 0 over here for non-linear constraint and this is the lower and upper bounds right.

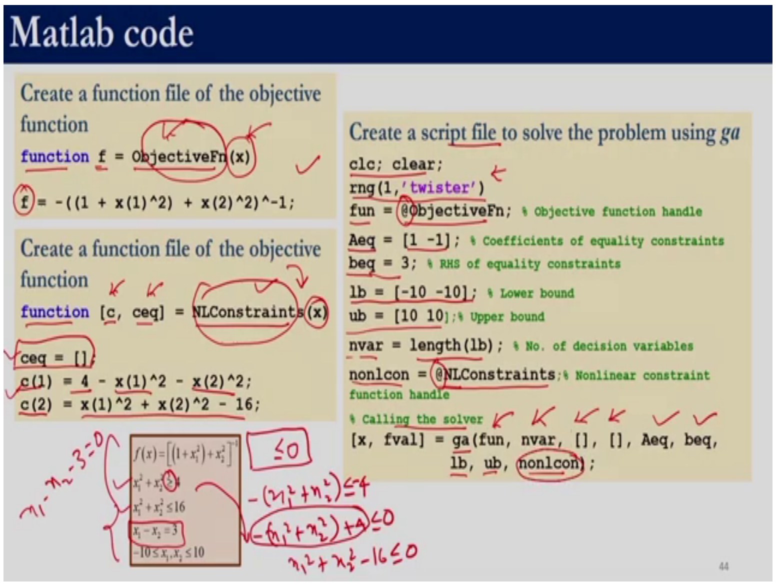
So, in addition to this right ga can also handle integer variables for the current problem we do not have any integer variables. So, we will not discuss about this right now. For using ga, ga is the inbuilt function again lower case right. Similar to fmincon to provide the objective function you can use a function file in this case we will be providing a function handle right. We need to provide number of decision variables. Remember for fmincon fun was the same thing, in fmincon, we did not have to provide the number of variable it was the initial guess over here we need to provide the number of variables right.

This is the same thing which we discussed for fmincon right. Similar, for non-linear constraint we will be defining the non-linear constraints in a function file and will provide the function handle over here right. So, this is identical to fmincon right ga additionally can take in a variable which can be used to provide indices of the decision variables which are integer right.

So, if we have integer variables then we can provide the indices of that which is similar to intlinprog which we have discussed previously. And we can also provide options to override any of the default options of ga right. The output from the ga is the solution vector right the value of the objective function at the solution vector, the exitflag as to what was the reason the algorithm terminated. A structure output which has various information about the algorithm; it can also provide us the final population right. So, ga is meta heuristic technique which we have discussed in the course, so there we work with population right.

So, at the end of all iterations the final population whatever we have that can also be accessed right. So, this variable population if we use over here right will provide us the final population and the objective function value or the fitness value of the final population. So, corresponding to this population what are their objective function values, so that is given in score right.

(Refer Slide Time: 20:45)



Matlab code

Create a function file of the objective function

```
function f = ObjectiveFn(x)

f = -((1 + x(1)^2) + x(2)^2)^-1;
```

Create a function file of the objective function

```
function [c, ceq] = NLConstraints(x)

ceq = [];
c(1) = 4 - x(1)^2 - x(2)^2;
c(2) = x(1)^2 + x(2)^2 - 16;
```

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

```
clc; clear;
rng(1,'twister')
fun = @ObjectiveFn; % Objective function handle
Aeq = [1 -1]; % Coefficients of equality constraints
beq = 3; % RHS of equality constraints
lb = [-10 -10]; % Lower bound
ub = [10 10];% Upper bound
nvar = length(lb); % No. of decision variables
nonlcon = @NLConstraints;% Nonlinear constraint
function handle
% Calling the solver
[x, fval] = ga(fun, nvar, [], [], Aeq, beq,
           lb, ub, nonlcon);
```

$f(x) = \left[(1+x_1^2)+x_2^2\right]^{-1}$

$x_1^2 + x_2^2 \geq 4$

$x_1^2 + x_2^2 \leq 16$

$x_1 - x_2 = 3$

$-10 \leq x_1, x_2 \leq 10$

44

So, we need to first create the objective function file, the name of the objective function file we are using is objectiveFn right. So, you can give any other name that you choose. So, to this function also the algorithm will pass the decision variables x. So, this function file is required to send the value of the objective function at this decision variable.

So, in this case the objective function is 1 plus x of 1 whole square plus x 2 square to the power minus 1, so this is the objective function. Similarly, we can define the constraints in another file right, so the name of the function is NLConstraints right. So, this is a function file just like fmincon we will have to use c and c equality right c is for the inequality constraint and c equality is for equality constraints. So, in this problem we do not have any equality constraint which is non-linear. So, there is this equality constraint, but it is linear, linear constraints can be provided separately.

Over here we need to provide only non-linear constraints right, so we do not have any non-linear equality right. So, we give c equality is equal to empty bracket right and we have these two constraints remember these are non-linear constraints. So, non-linear constraints need to be of the form less than equal to 0. So, this constraint is of the form greater than or equal to right. So, we need to multiply by a minus sign on both sides. So, minus x 1 square plus x 2 square is less than equal to minus 4.

So, right now we have only converted this greater than or equal 2 2 less than or equal 2 this has to be brought to the left hand side because for non-linear constraints we require the right hand side to be 0 right. So, this has to be minus x 1 square plus x 2 square right plus 4 is less than equal to 0 right. The other constraint is straight forward that it is x 1 square plus x 2 square minus 16 is less than equal to 0.

So, the constraints need to be first converted into the appropriate format right. So, here if you see minus x 1 square minus x 2 square plus 4 that is the left hand side over here. Similarly, the other one has been defined x 1 square plus x 2 square minus 16, so c of 1 c of 2. Every time the algorithm passes the value of the decision variable, this file will return the values of the constraints right.

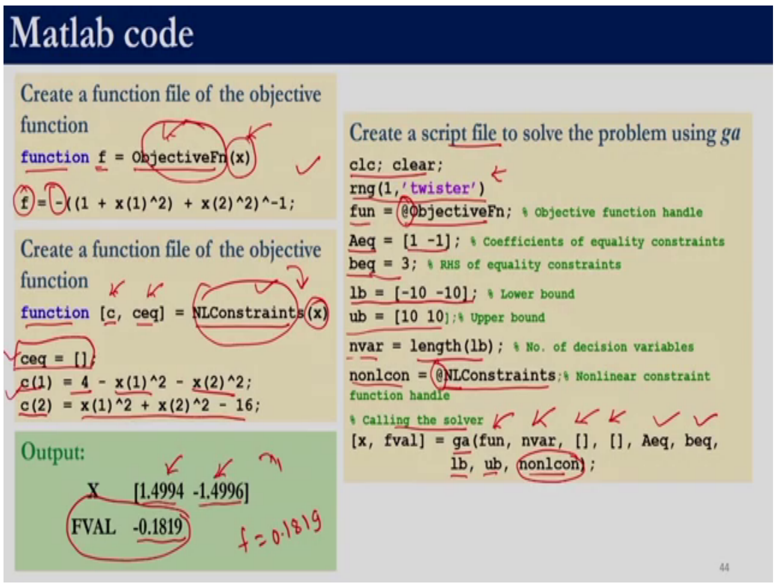
So, these 2 file is similar to what we have written previously for fmincon. Now, we have created both the function file which is required right. So, we can write this script file, so clc clear you know right. So, since it is a stochastic algorithm every time we run we may get a different solution right, so we do not want that to happen. So, we are fixing the seed to generate random numbers.

So, every time we run we will get the same answer once we see that it is running well for one algorithm then you know how to change this seed and run for multiple times and do statistical analysis that we have discussed previously as part of this course right. So, we are defining fun which is a function handle right; so, at objective fun because that is the name of the function file which we created.

So, we have equality constraint right, so we define Aeq, beq the coefficient of x 1 is 1 the coefficient of x 2 is minus 01. So, we give 1 and minus 1 the right hand side vector is 3 right, remember this equation is not to be converted into x 1, minus x 2, minus 3 is equal to 0 right. So, for linear constraints the constant term should be on the right hand side. So, for this problem both the variables are bounded between minus 10 and 10. So, the lower bound is minus 10 minus 10 the upper bound is 10 10. So, the number of variables we could either write to directly or we can just say length of l b.

Then we are defining this function handle with the name of the file which has the non-linear constraints. So, here it is NLConstraints, so we are defining that right. Now, we can call the solver right, so ga we need to pass the function handle which has the objective function value number of decision variables linear inequalities. In this problem we do not have any linear inequalities we have one equality constraints. So, we need to pass on these 2 variable the lower bound, upper bound, and the name of the file which has the non-linear constraints.

(Refer Slide Time: 25:02)



Matlab code

Create a function file of the objective function

```
function f = ObjectiveFn(x)
f = -((1 + x(1)^2) + x(2)^2)^-1;
```

Create a function file of the objective function

```
function [c, ceq] = NLConstraints(x)
ceq = [];
c(1) = 4 - x(1)^2 - x(2)^2;
c(2) = x(1)^2 + x(2)^2 - 16;
```

Output:

```
X       [1.4994  -1.4996]
FVAL    -0.1819
```

f = 0.1819

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

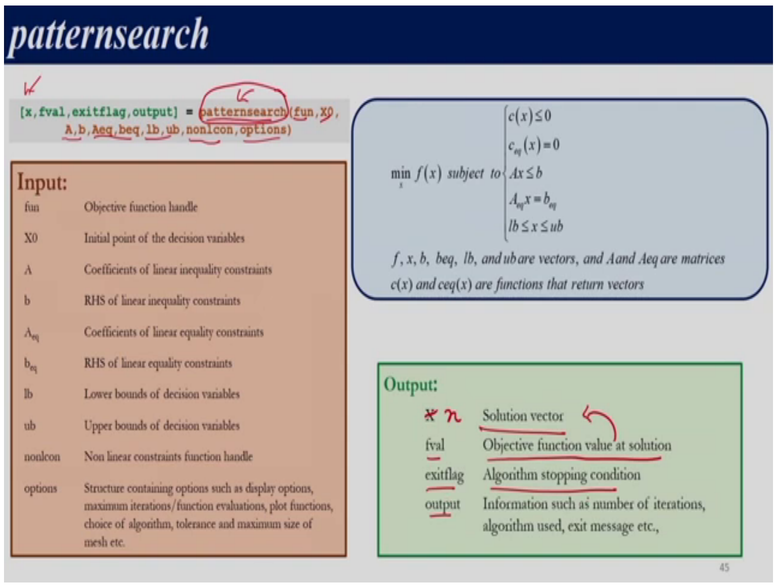
```
clc; clear;
rng(1,'twister')
fun = @ObjectiveFn; % Objective function handle
Aeq = [1 -1]; % Coefficients of equality constraints
beq = 3; % RHS of equality constraints
lb = [-10 -10]; % Lower bound
ub = [10 10];% Upper bound
nvar = length(lb); % No. of decision variables
nonlcon = @NLConstraints;% Nonlinear constraint
function handle
% Calling the solver
[x, fval] = ga(fun, nvar, [], [], Aeq, beq,
            lb, ub, nonlcon);
```

44

This is what we get as solution right 1. 4994 minus 1; 4996 and the value of the objective function at this point is minus 0.1819. Remember we had a maximization problem right that is why we had a negative sign over here to convert it into a minimization problem. So, whatever value we get only for the objective function right not for the decision variable only for the objective function we need to multiply it with a negative sign.

So, the value of the objective function is 0.1819, because our problem is maximization problem. The decision variable are as given over here this is x 1 this is x 2 because that is the way we had choose to arrange the variable. So, in addition to x and FVAL right which is what we accessed as of now the algorithm can also provide exit flag, output, population, and score right. So, we expect you to execute the same problem right and see these values and you can also read the help given in MATLAB.

(Refer Slide Time: 26:08)



patternsearch

```
[x,fval,exitflag,output] = patternsearch(fun,X0,
      A,b,Aeq,beq,lb,ub,nonlcon,options)
```

$$\min_x f(x) \text{ subject to } \begin{cases} c(x) \le 0 \\ c_{eq}(x) = 0 \\ Ax \le b \\ A_{eq}x = b_{eq} \\ lb \le x \le ub \end{cases}$$

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

c(x) and ceq(x) are functions that return vectors

Input:

fun	Objective function handle
X0	Initial point of the decision variables
A	Coefficients of linear inequality constraints
b	RHS of linear inequality constraints
A_{eq}	Coefficients of linear equality constraints
b_{eq}	RHS of linear equality constraints
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
nonlcon	Non linear constraints function handle
options	Structure containing options such as display options, maximum iterations/function evaluations, plot functions, choice of algorithm, tolerance and maximum size of mesh etc.

Output:

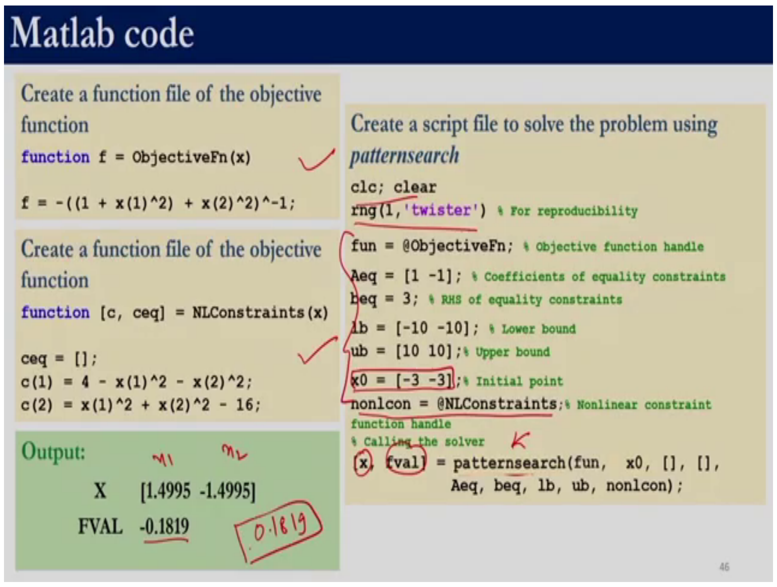
~~x~~ x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message etc.,

45

So, now we will solve the same problem with patternsearch right. So, for patternsearch the name of the function is patternsearch this is the inbuilt function right. The input is similar to fmincon right, the objective function, the initial point, linear inequalities, linear equalities, upper and lower bound. Function file which will contain the non-linear constraints and options which can be used to override the default options of patternsearch.

So, the output from patternsearch would be x the solution vector right, fval which is the value of the objective function at the solution given by x. Exit flag that will help us to determine the reason for which the algorithm terminated. And it will also give this output structure right it will have lot of other variables which can give information about various other things.

(Refer Slide Time: 26:57)



Matlab code

Create a function file of the objective function

```
function f = ObjectiveFn(x)

f = -((1 + x(1)^2) + x(2)^2)^-1;
```

Create a function file of the objective function

```
function [c, ceq] = NLConstraints(x)

ceq = [];
c(1) = 4 - x(1)^2 - x(2)^2;
c(2) = x(1)^2 + x(2)^2 - 16;
```

Output: x_1 x_2

```
      X    [1.4995  -1.4995]
   FVAL    -0.1819
```

$\boxed{0.1819}$

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

```
clc; clear
rng(1,'twister') % For reproducibility
fun = @ObjectiveFn; % Objective function handle
Aeq = [1 -1]; % Coefficients of equality constraints
beq = 3; % RHS of equality constraints
lb = [-10 -10]; % Lower bound
ub = [10 10];% Upper bound
x0 = [-3 -3];% Initial point
nonlcon = @NLConstraints;% Nonlinear constraint
function handle
% Calling the solver
[x, fval] = patternsearch(fun,  x0, [], [],
             Aeq, beq, lb, ub, nonlcon);
```

46

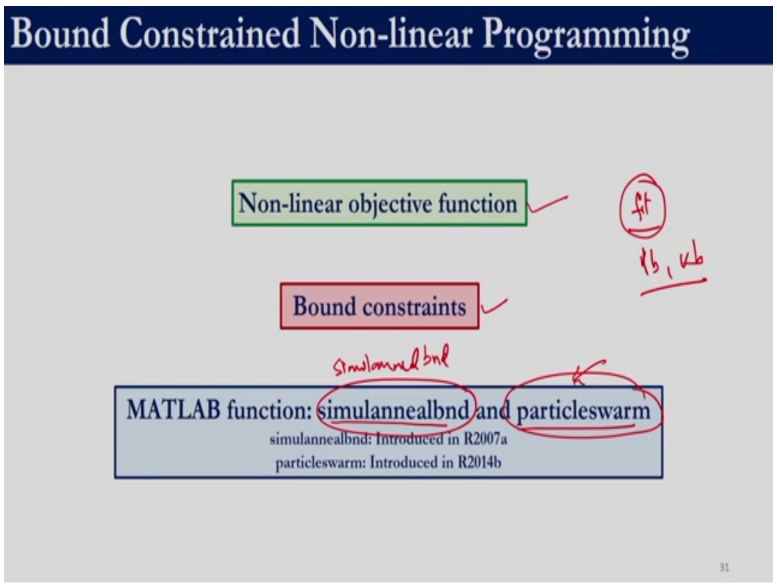
So, similar to ga we need to construct these two files right. So, this has non-linear constraints this has the objective function right and over here patternsearch is also a stochastic technique right. So, we need to fix the seat this part remains the same which we discussed previously right.

Over here we need to give an initial guess whereas, in ga we did not provide an initial guess right. So, x naught is equal to minus 3 minus 3 is the initial point we are giving, this is again the function handle for non-linear constraint. So, we are solving it by patternsearch right. So, if we solve it by patternsearch we will get the value of x as well as fval.

In this case x 1 is 1.4995 and x 2 is minus 1.4995 the value of the objective function reported is minus 0.1819. But, since it us a maximization problem which we are solving the value of the objective function would be 0.1819. So, we have seen three functions right fmincon, ga, and patternsearch all these three are inbuilt functions of MATLAB.

So, we showed you how to solve a problem with fmincon and we choose a different problem for demonstrating ga and patternsearch right. So, we expect you to look into the other output which is given by all these 3 functions and try to interpret them. Next we will be looking into how to solve bound constraint non-linear optimization problem using particleswarm optimization and simulated annealing.

(Refer Slide Time: 28:28)



Bound Constrained Non-linear Programming

Non-linear objective function ✓

fit

lb, ub

Bound constraints ✓

simulannealbnd

MATLAB function: simulannealbnd and particleswarm

simulannealbnd: Introduced in R2007a

particleswarm: Introduced in R2014b

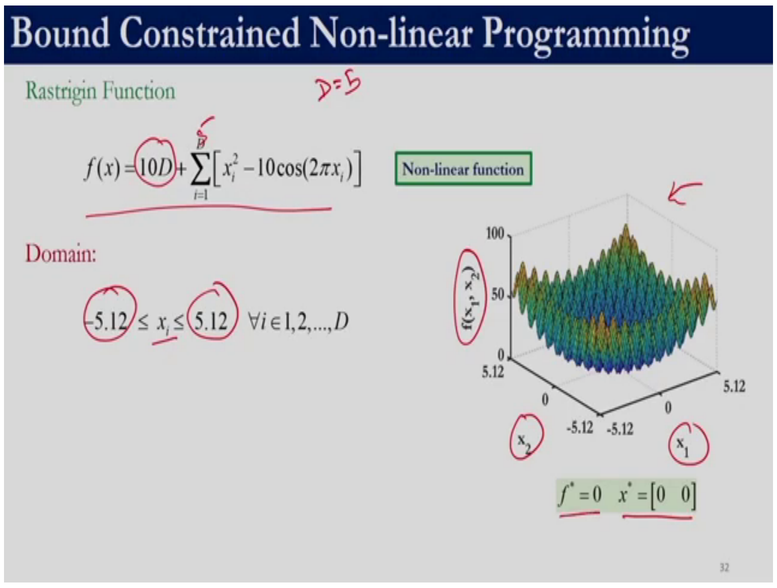
31

We will be looking into two functions one is based on particleswarm optimization which we discussed as part of this course, another one is based on another materialistic technique called as simulated annealing. So, the inbuilt function in MATLAB for simulated annealing is simulannealbnd right. So, this simul for simulated, and then anneal we have for annealing, and then this supports bound constraints.

So, we have this inbuilt function in MATLAB particleswarm optimization is something that we have discussed in this course right. So, both of this function can be used to solve optimization problems which have non-linear objective function. It can also support linear objective function right, but when we have linear objective function we usually prefer linprog in MATLAB to solve this right and it can also support bound constraints right.

So, that is what we had discussed in particleswarm optimization right that we need to provide the fitness function, the lower bound, and the upper bound right So, it does not matter what is the nature of this fitness function whether it is linear, non-linear all of them could be solved using these 2 inbuilt functions right.

(Refer Slide Time: 29:33)



Bound Constrained Non-linear Programming

Rastrigin Function

$D=5$

$$f(x) = 10D + \sum_{i=1}^{\overset{5}{D}} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Non-linear function

Domain:

$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,\ldots,D$$

$f^* = 0 \quad x^* = [0 \quad 0]$

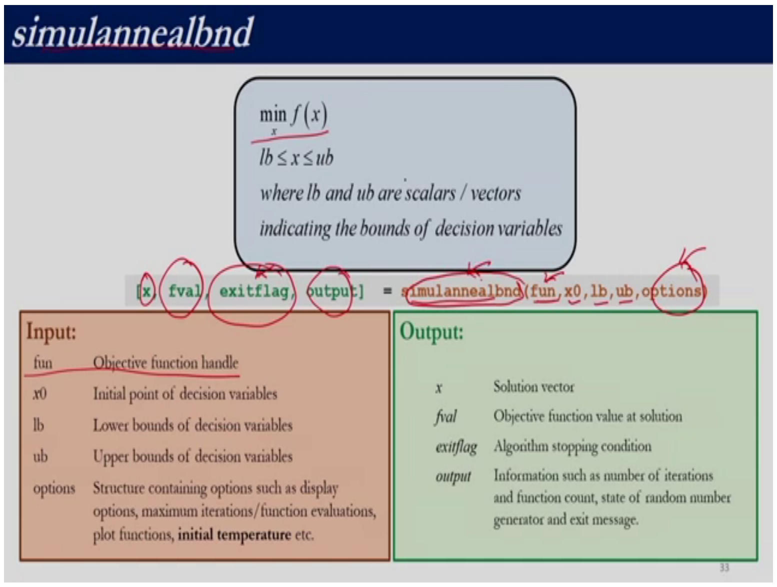
32

$f(x_1, x_2)$
100
50
0
5.12
0
-5.12
-5.12
0
5.12
x_2
x_1

So, here we will take the rastrigin function, rastrigin function is given over here. So, it is a scalable function as in like we need to fix the number of variables right. So, if it is D is equal to 5. So, this will be 10 into 5 right plus summation going from 1 to 5 right x 1 square minus 10 cos 2 pi x 1 plus x 2 square minus 10 cos 2 pi plus x 3 square minus 10 cos 2 pi x 3 all the way up to 5 right.

And the domain of the decision variables are between minus 5.12 to 5.12 this is the surface plot of the rastrigin function right. So, x 1 over here x 2 over here and the objective function value. So, as you can see there are several peaks and valley the algorithm is expected to search in this space and come up with the best optimal value right. So, for this problem the global optima is known, it occurs at x one is equal to 0 x 2 is equal to 0 and the value of the objective function is 0 at this point.

(Refer Slide Time: 30:31)



simulannealbnd

$$\min_x f(x)$$
$$lb \le x \le ub$$
where lb and ub are scalars / vectors indicating the bounds of decision variables

```
[x, fval, exitflag, output] = simulannealbnd(fun,x0,lb,ub,options)
```

Input:

fun	Objective function handle
x0	Initial point of decision variables
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
options	Structure containing options such as display options, maximum iterations/function evaluations, plot functions, **initial temperature** etc.

Output:

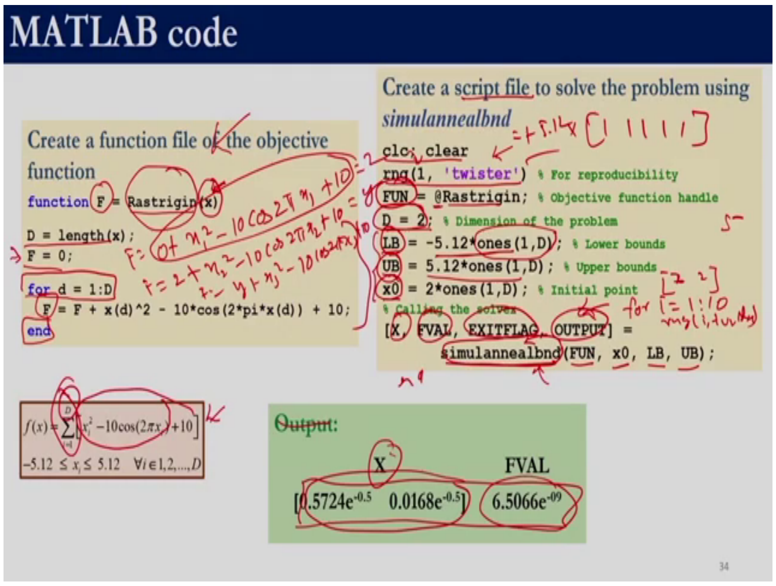
x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations and function count, state of random number generator and exit message.

33

For us to use this MATLAB inbuilt function for simulated annealing the optimization problem has to be necessarily a minimization problem. If we have a maximization problem we need to convert it to into a minimization problem right. So, the syntax for using this function is the name of the inbuilt function we need to provide the function. So, in this case we will be providing it using a function handle.

We will write a function file which will contain the objective function right. And we will supply the name of that file over here right and we can give a starting point lower bound upper bound and this options is required if we want to overwrite any of the default options of this function. The output from this function would be the decision variables the value of the objective function, the reason for termination right. It will be indicated by a number and you can then look into the help of this function. And decode what is the reason for termination using the value indicated by exit flag output is a structure right. So, which will contain information on several things related to simulated annealing.

(Refer Slide Time: 31:36)



MATLAB code

Create a function file of the objective function

```
function F = Rastrigin(x)

D = length(x);
F = 0;

for d = 1:D
  F = F + x(d)^2 - 10*cos(2*pi*x(d)) + 10;
end
```

$$f(x) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,\ldots,D$$

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

```
clc; clear
rng(1, 'twister') % For reproducibility
FUN = @Rastrigin; % Objective function handle
D = 2; % Dimension of the problem
LB = -5.12*ones(1,D); % Lower bounds
UB = 5.12*ones(1,D); % Upper bounds
x0 = 2*ones(1,D); % Initial point
% Calling the solver
[X, FVAL, EXITFLAG, OUTPUT] =
      simulannealbnd(FUN, x0, LB, UB);
```

~~Output:~~

```
            X                        FVAL
[0.5724e-0.5   0.0168e-0.5]     6.5066e-09
```

34

$F = 0 + x_1^2 - 10\cos 2\pi x_1 + 10 = 2$
$F = 2 + x_2^2 - 10\cos 2\pi x_2 + 10 = y$
$F = y + x_3^2 - 10\cos 2\pi x_3 + 10$
$= +5.14 \times [1\ 1\ 1\ 1\ 1]$
5
[2 2]
for i = 1:10
rng(i, twister)

So, for this objective function right we need to first write a function file. So, here we are writing a function file whose name is rastrigin right and the input to this function file would be x right only one variable irrespective of any number of decision variable right. So, this variable would be a vector if there is more than one decision variable right.

So, if there are five decision variable x will contain 5 values. So, what we are trying to write over here is right objective function file which will work for any number of variables right. So, this is a scalable function. So, what we are doing is first we find out what is the length of x right. So, that would be D then we assign a variable f is equal to 0 and then we run a loop for the summation we run a loop for D is equal to 1 to upper case D right.

So, what we do is every time we calculate this term for that particular variable and then sum it with F right. And this f will contain the value still the previous term. So, initially it will be 0 plus x 1 square minus 10 cos 2 pi x 1 right plus 10 right. So, this will be f in the first instant next instant the value of this would be stored in f right. So, let us say the value turns out to be 2. So, this will be f is equal to 2 plus x 2 square minus 10 cos 2 pi x 2 plus 10.

And the third time it will be f plus whatever value we get over here let us say y y plus x 3 square minus 10 cos 2 pi x 3 plus 10 right. So, at the end of it we will have this as a scalar value which is what this function passes to simulated annealing. So, this is what we can have in the script file right. So, clc clear is to just clear the command window and MATLAB workspace, again simulated annealing is a stochastic technique right.

So, to help us to reproduce the results we are fixing the random number generator right. So, rng we are using the twister algorithm with a seed of 1 and then we are defining this variable FUN which is a function handle right. So, we have this at the rate symbol and then the name of the objective function file over here right. So, whenever we want to access this particular function file we can just call FUN and over here we are defining D is equal to 2 for a 2 variable problem the lower and upper bound is defined using these two lines right.

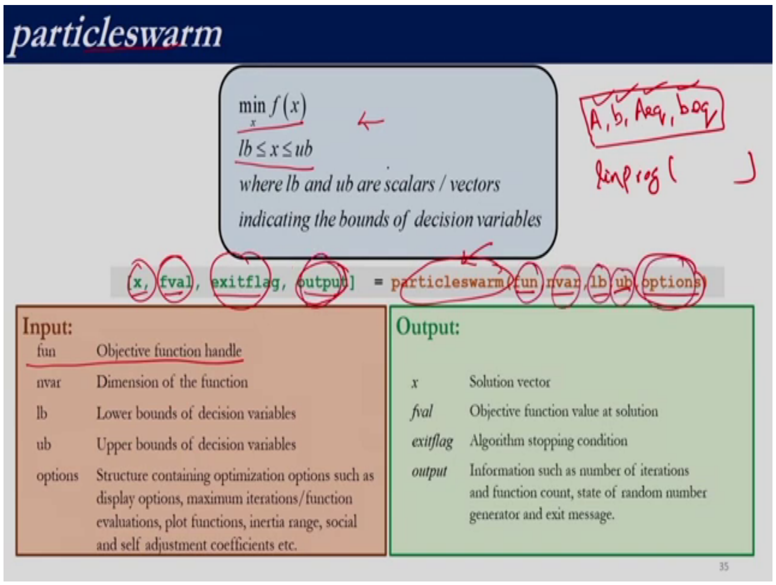
So, we first create a vector of onse right, so if there are 5 decision variable. We create 1 1 1 1 1 and then multiply it with minus 5 point one 2. So, this will give us the lower bound and for upper bound we have this plus 5 one 2 into onse of D x naught is the initial starting point right. So, we are giving just 2 2 as the starting point x 1 is equal to 2 x 2 is equal to 2 if d is equal to 2 right. So, if we change D to b let us say 5 the lower bound upper bound and this x naught would automatically get adjusted right.

So, over here we are calling the function simulated annealing we are providing this variable fun right which is a function handle the initial guess which we have defined right the lower bound and the upper bound. So, we are solving with default options of simulated annealing. So, we are not providing any options right. So, what it can return is x fval exitflag and output this output does not correspond to this output right.

So, this output in MATLAB would have several fields it is a structure it will have several fields when you execute this program you can have a look at it right. The final values of x and fval reported by MATLAB for this particular problem is as given over here right. So, these are the decision variables and this is the objective function value at this decision variable right. So, since this is a stochastic technique we need to run it for multiple times.

So, as usual we can put a for loop over here for I is equal to let us say 1 to 10 and then bring this line over here right. So, rng of I comma twister and then run I comma twister and then run this for 10 or 15 times and then do a statistical analysis as shown earlier for the 5 algorithms which we have discussed as part of this course.

(Refer Slide Time: 36:08)



particleswarm

$$\min_{x} f(x)$$

$$lb \le x \le ub$$

where lb and ub are scalars / vectors indicating the bounds of decision variables

A, b, Aeq, beq

linprog (   )

```
[x, fval, exitflag, output] = particleswarm(fun, nvar, lb, ub, options)
```

Input:

fun	Objective function handle
nvar	Dimension of the function
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
options	Structure containing optimization options such as display options, maximum iterations/function evaluations, plot functions, inertia range, social and self adjustment coefficients etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations and function count, state of random number generator and exit message.

35

So, the next function which we will discuss is particleswarm. So, the inbuilt function for particleswarm optimization in MATLAB is particleswarm like all other functions of MATLAB it can only solve minimization problems right. And it does not explicitly support constraints right and it can have bounds on the decision variables lower and upper bound.

So, when we say it does not explicitly support constraint it does not mean that particleswarm optimization cannot be used to solve constrained optimization problem. But, MATLAB does not allow to specify the constraints right, but we can include the constraints in the objective function file which we have done in the first half of the course right.

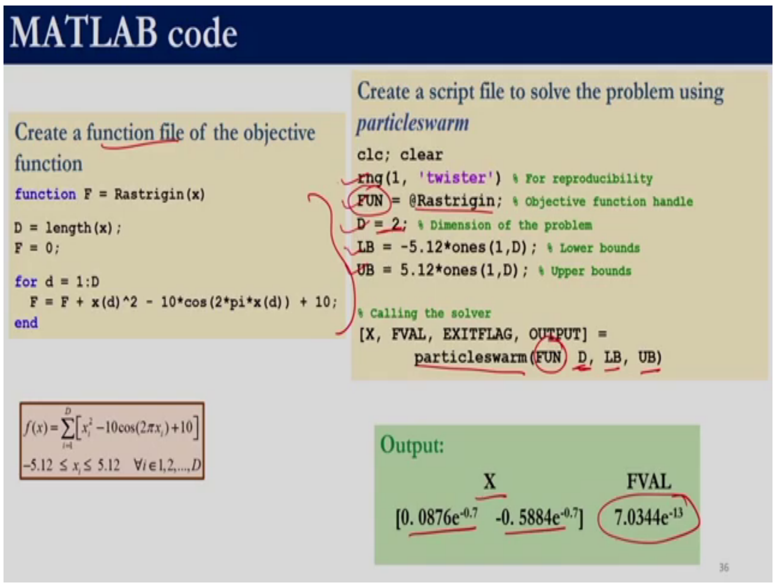
Wherein we had the objective function we evaluate the violation and then add penalty to the objective function. So, that way we can solve constrained optimization using particles warm, but MATLAB as such does not have the feature to explicitly define the constraints right. So, for example, if you remember linprog we were able to define A B A equality B equality the linear constraints where explicitly defined right and given to the function linprog right.

So, over here particleswarm function of MATLAB does not explicitly take these constraints. So, the syntax of particleswarm is the name of the function particleswarm the name of the function which has the objective function value. We will write a function file which we will contain the objective function and we will pass on that as a function handle.

Nvar is the number of decision variable, lb and ub are the lower and upper bounds of the decision variables. And options can be used if we want to overwrite any of the default setting parameters of particleswarm. The output is x which is the decision variable the value of the objective function at the decision variable, exitflag would be a integer which will tell us the reason for the termination of the function particleswarm.

And we will have a structure right which will have several fields right. So, the name of the structure that we have given over here is output right. So, this output will have several fields which will provide various information about this particleswarm optimization right. So, again all these 4 are variable names right, we can give any other variable name similarly all of these are variable names.

(Refer Slide Time: 38:19)



MATLAB code

Create a function file of the objective function

```
function F = Rastrigin(x)

D = length(x);
F = 0;

for d = 1:D
  F = F + x(d)^2 - 10*cos(2*pi*x(d)) + 10;
end
```

$$f(x)=\sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$$
$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,\dots,D$$

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

```
clc; clear
rng(1, 'twister') % For reproducibility
FUN = @Rastrigin; % Objective function handle
D = 2; % Dimension of the problem
LB = -5.12*ones(1,D); % Lower bounds
UB = 5.12*ones(1,D); % Upper bounds

% Calling the solver
[X, FVAL, EXITFLAG, OUTPUT] =
      particleswarm(FUN, D, LB, UB)
```

Output:

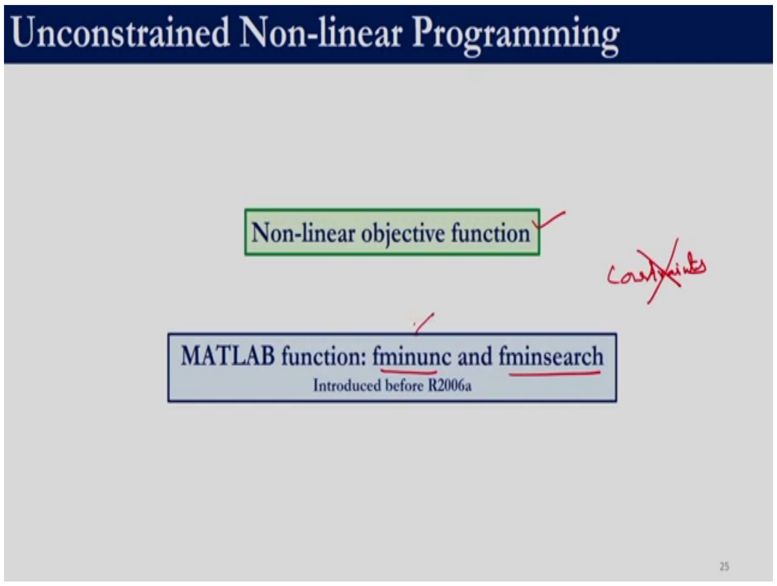
```
              X                      FVAL
[0. 0876e-0.7  -0. 5884e-0.7]    7.0344e-13
```

36

Here we will use the same function file which we built for simulated annealing right. So, this is the same file. So, here for solving we fix the random number generator we define this function handle, we define d is equal to 2 because we are solving a 2 variable problem lb and ub are defined like this which we have discussed previously right. So, here we call particleswarm provide FUN which basically refers to the function file which has the objective function D is the number of variables right.

So, 2 in this case the lower bound and the upper bound right. So, the output from this would be x 1 is equal to this and x 2 is equal to this and at this value of x 1 and x 2 the objective function has a value of 7.0 into 10 power minus 13. So, now we have seen simulated annealing and particle swarm optimization. So, both of this can be used to solve linear and non-linear objective function, but they cannot support constraints right it is only for unconstrained optimization problem.

(Refer Slide Time: 39:32)



Unconstrained Non-linear Programming

Non-linear objective function ✓

~~Constraints~~

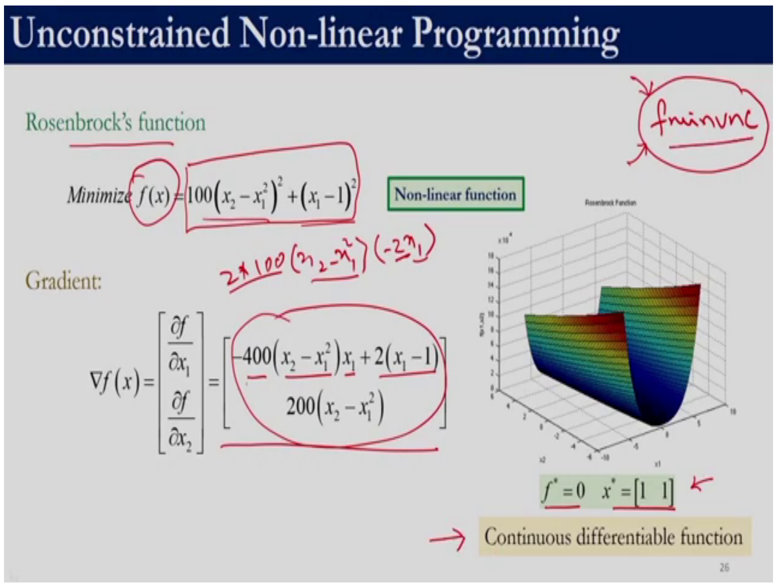
MATLAB function: fminunc and fminsearch

Introduced before R2006a

25

But, the decision variables can have their own bounds. Now, we look into 2 inbuilt functions of MATLAB fminsearch and fminunc which will help us to solve unconstrained non-linear optimization problems. So, first will begin with these 2 functions fminunc and fminsearch these are 2 inbuilt MATLAB functions. So, it can be used for problems which have a non-linear objective function and there should not be any constraints. So, constraints are not allowed right neither linear nor non-linear even bound constraints are not allowed in fminunc and fminsearch.

(Refer Slide Time: 39:53)



Unconstrained Non-linear Programming

fminunc

Rosenbrock's function

$$Minimize\ f(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

Non-linear function

$$2*100(x_2 - x_1^2)(-2x_1)$$

Gradient:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -400(x_2 - x_1^2)x_1 + 2(x_1 - 1) \\ 200(x_2 - x_1^2) \end{bmatrix}$$

Rosenbrock Function

$f^* = 0 \quad x^* = [1\ \ 1]$

Continuous differentiable function

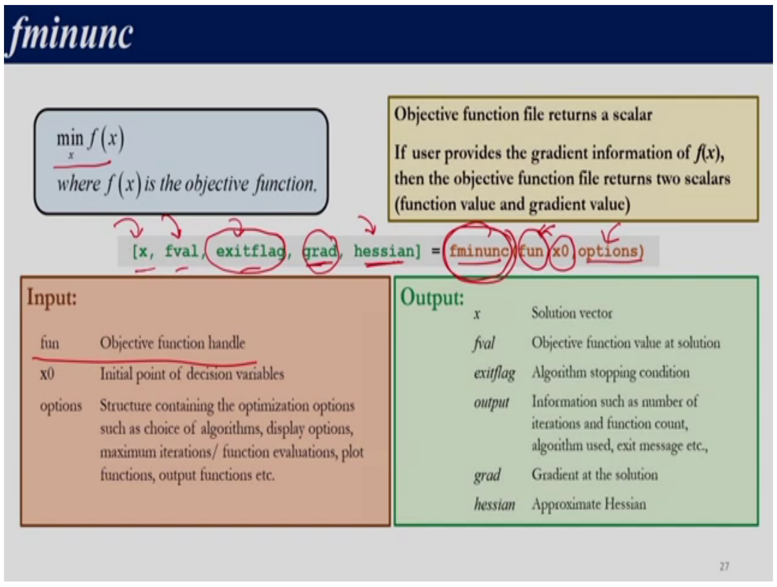
26

So, the example that we will be taking is rosenbrocks function right we are taking 2 variables right. So, we are supposed to find out the values of x 1 and x 2 at which the function has a minimum value. So, for this problem the optima actually lies at 1 comma 1 that x 1 is equal to 1 x 2 is equal to one and at that point the value of the objective function will be 0 right.

So, this is the global optima which is known for rosenbrock function right. So, first we will be discussing fminunc right, so for this we can also supply the gradient right and the function has to be continuous. So, we can find out what is dou f by dou x 1 dou f by dou x 2 and this information can also be used by fminunc it not only can take the objective function value, but it can also take the gradient right. So, in this case dou f by dou x 1 will work out to be 2 into 100 x 2 minus x 1 square minus 2 x 1 right.

So, that is why 200 400 minus 400 x 2 minus x 1 square into x 1 plus 2 times x 1 minus 1 right; similarly, dou f by dou x 2 can be calculated. So, to this function we will not only provide this objective function value, but we are also be providing this gradient expression for dou f by dou x one and dou f by dou x 2 right. For us to use fminunc the function has to be a continuous differentiable function.

(Refer Slide Time: 41:19)



fminunc

$\min_x f(x)$

where $f(x)$ is the objective function.

Objective function file returns a scalar

If user provides the gradient information of *f*(*x*), then the objective function file returns two scalars (function value and gradient value)

```
[x, fval, exitflag, grad, hessian] = fminunc(fun, x0, options)
```

Input:

fun	Objective function handle
x0	Initial point of decision variables
options	Structure containing the optimization options such as choice of algorithms, display options, maximum iterations/ function evaluations, plot functions, output functions etc.

Output:

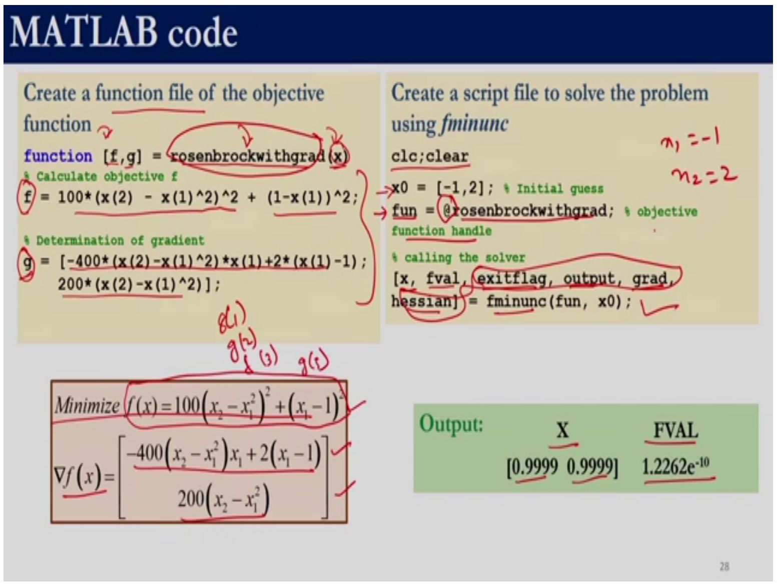
x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations and function count, algorithm used, exit message etc.,
grad	Gradient at the solution
hessian	Approximate Hessian

27

So, fminunc solves a minimization problem right. So, if I have a maximization problem we are supposed to convert it into a minimization problem, so the input to fminunc is the objective function right. So, fun is a objective function handle it can also contain the gradient as we will show you it can also contain the gradient. Or it can just contain the objective function value we can give an initial guess and if we want to override any of the default option of fminunc we can also give this option. So, the output from fminunc would be the decision variables right the value of the objective function at the decision variable.

Exitflag will be a integer right which will tell us the reason for the termination of fminunc and it can also give us the gradient and hessian of the function right. So, here remember that all these are variable names right. We can give any variable name just that the fourth value returned by fminunc would correspond to gradient. The fifth value would correspond to hessian, the third value would correspond to exit flag, the second value would correspond to fval, and the first value will correspond to the decision variables.

(Refer Slide Time: 42:27)



MATLAB code

Create a function file of the objective function

```
function [f,g] = rosenbrockwithgrad(x)
% Calculate objective f
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;

% Determination of gradient
g = [-400*(x(2)-x(1)^2)*x(1)+2*(x(1)-1);
    200*(x(2)-x(1)^2)];
```

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

```
clc;clear
x0 = [-1,2]; % Initial guess
fun = @rosenbrockwithgrad; % objective function handle
% calling the solver
[x, fval, exitflag, output, grad,
hessian] = fminunc(fun, x0);
```

$$\text{Minimize } f(x) = 100\left(x_2 - x_1^2\right)^2 + \left(x_1 - 1\right)^2$$

$$\nabla f(x) = \begin{bmatrix} -400\left(x_2 - x_1^2\right)x_1 + 2\left(x_1 - 1\right) \\ 200\left(x_2 - x_1^2\right) \end{bmatrix}$$

```
Output:    X                FVAL
      [0.9999  0.9999]   1.2262e-10
```

So, this was our objective function right and this was our gradient right. So, we are writing this file right, so rosenbrock with grad this is the name of the file again you can give whatever name you want. The input to the function is the value of the decision variables every time the solver would call this file multiple times and every time it calls it will give us the decision variables and we can return the value of the objective function and the value of the gradients. So, f is the value of the objective function, so the first value which we are supposed to return is the objective function value.

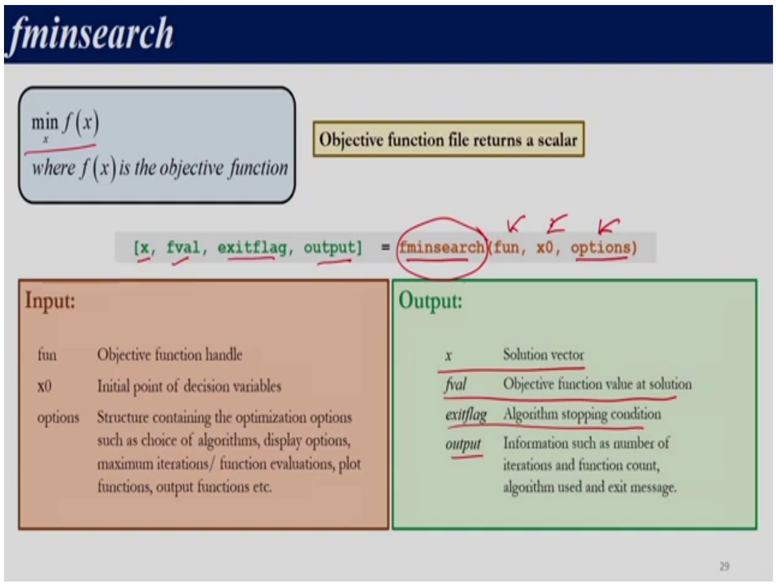
So, we have written f is equal to 100 x 2 minus x 1 square the whole square plus one minus x 1 the whole square or x 1 minus 1 the whole square both are same. So, the gradient will have 2 values g of 1 and g of 2 right. So, g of 1 is this expression g of 2 is this expression, so we have this first row and then we have a semicolon and then the second row. So, this will contain 2 values because we have 2 decision variables right, so x 1 and x 2. So, what this piece of code does is it will plug in the value of x 1 and x 2 in these 3 expressions and return them back right.

So, objective function value is written separately as f where as the gradient is returned as a vector if this were to be a 5 variable problem this g will have 5 values g of 1, g of 2, g of 3 and so on all the way up to g of 5. So, once we have created this function file we can now solve the problem right. So, clc and clear will clear the command window and the MATLAB workspace right we are defining an initial guess x naught is equal to minus 1 2. So, we are solving the problem with an initial guess of x 1 is equal to minus 1 and x 2 is equal to 2.

We are defining a variable fun which is a function handle right and we are assigning the name of this file to this variable, so fun becomes the function handle. So, the output from fminunc which we want is the value of the decision variable the objective function value at that x exitflag output grad and hessian right. So, we could have just given x 1, x 2, x 3, x 4, x 5, x 6; x 1 would indicate the decision variables, x 2 would indicate fval, x 3 would indicate exitflag, x 4 would indicate output, x 5 would indicate grad, and x 6 would indicate hessian.

You can go back and have a look at what would these values be right, but here we are only showing what is the value of x and what is the value of fval. So, fminunc has been able to find the value of x 1 is equal to 0.999 x 2 is equal to 0.9999 and the value of the objective function is has given over here. So, this is how you use fminunc to solve an unconstrained optimization problem, now we look into fminsearch.

(Refer Slide Time: 45:19)



fminsearch

$\min_{x} f(x)$

where $f(x)$ is the objective function

Objective function file returns a scalar

```
[x, fval, exitflag, output] = fminsearch(fun, x0, options)
```

Input:

fun	Objective function handle
x0	Initial point of decision variables
options	Structure containing the optimization options such as choice of algorithms, display options, maximum iterations/ function evaluations, plot functions, output functions etc.

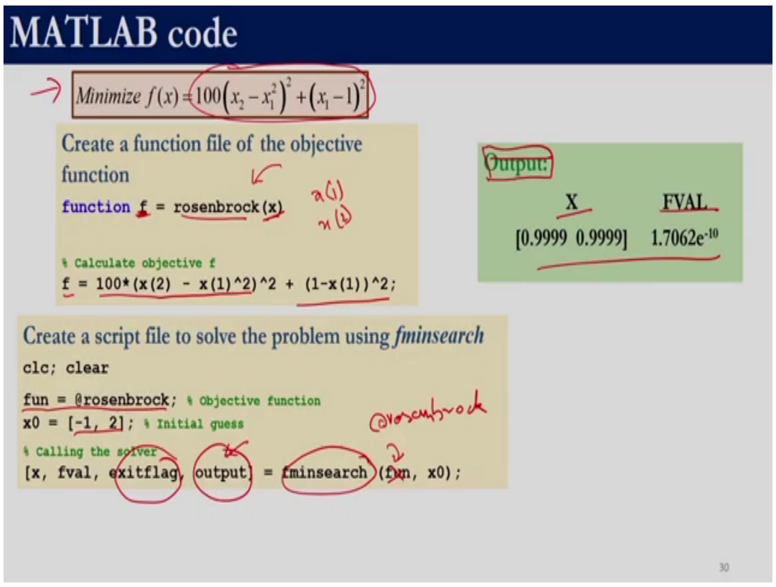
Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations and function count, algorithm used and exit message.

29

So, for fminsearch also MATLAB supports only minimization problem right. So, if you have a maximization problem you need to convert it into a minimization problem. For the function fminsearch in MATLAB, we need to provide the objective function a starting point. And if we want to override any of this options of fminsearch we need to use options. The output which we can have from fminsearch is x which is the value of the decision variables after solving by fminsearch fval is the value of the objective function at the decision variable. Exitflag is a number which will tell us the reason why fminsearch terminated and output contains more information about the solution process.

(Refer Slide Time: 46:00)



MATLAB code

Minimize $f(x) = 100\left(x_2 - x_1^2\right)^2 + \left(x_1 - 1\right)^2$

Create a function file of the objective function

```
function f = rosenbrock(x)

% Calculate objective f
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
```

x(1)
x(2)

Output:

```
      X               FVAL
[0.9999 0.9999]    1.7062e-10
```

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

```
clc; clear
fun = @rosenbrock; % Objective function
x0 = [-1, 2]; % Initial guess
% Calling the solver
[x, fval, exitflag, output] = fminsearch(fun, x0);
```

@rosenbrock

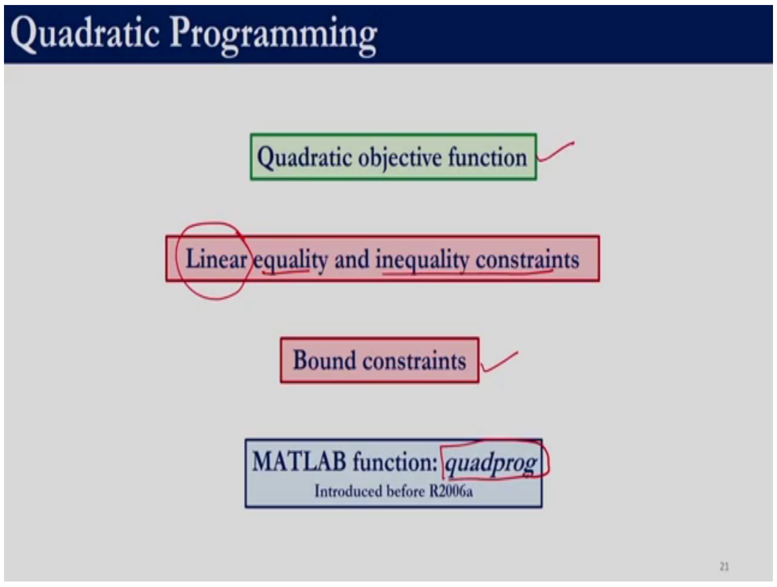
30

So, we will use the same optimization problem which we used right for fminunc. So, the objective is to minimize f of x is equal to 100 into x 2 minus x 1 square the whole square plus x 1 minus 1 the whole square right. So, here we need to only pass the objective function value we cannot pass gradient for fminsearch. We create this function file which will return the scalar f the name of the function is rosenbrock and x is the input to this function.

So, fminsearch when it solves it will call this function multiple times every time it will send the value of x right. So, x will contain x of 1 x of 2 because we have 2 decision variables for this problem and this function is expected to return a scalar value right. So, the scalar value is nothing but the evaluation of the objective function at the given point x. So, this is straightforward we call the function fminsearch we provide this fun right, so fun is function handle. So, either we can use fun over here or we can directly give at rosenbrock over here comma x naught.

So, again we are starting with the point minus 1 comma x 2 in this case the value of the decision variable that we obtain and the objective function is given over here right. We expect you to go back and look at the exitflag and output right. So, remember this output is not the same as this output right. So, this output is just to show you that this is the output from this function whereas, this MATLAB variable output is a structure right. You will be having multiple fields into that, so you can look into the values of each of them.

(Refer Slide Time: 47:33)



Quadratic Programming

Quadratic objective function

Linear equality and inequality constraints

Bound constraints

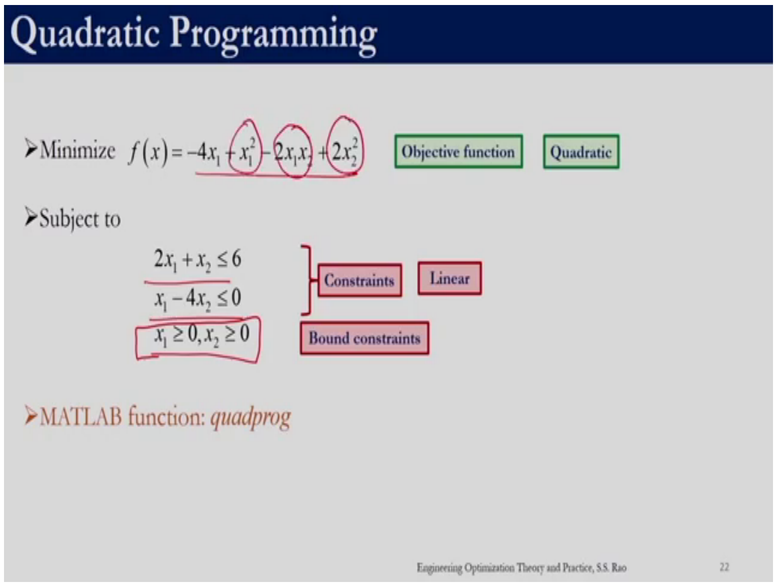
MATLAB function: *quadprog*
Introduced before R2006a

21

So, now we look into a special type of non-linear problem which is called as quadratic programming. In quadratic programming the objective function is quadratic whereas, the constraints are supposed to be linear for the inbuilt function quadprog of MATLAB. So, this quadratic programming can be solved using the inbuilt function quadprog in MATLAB right.

So, here we can have bound constraints, so the decision variables can have their own bounds lower and upper bounds. This function quadprog can support equality as well as inequality constraints, but they need to be linear. If the constraints are not linear then we cannot use quadprog and the objective function as we discuss has to be quadratic in nature.

(Refer Slide Time: 48:10)



Quadratic Programming

- Minimize $f(x) = -4x_1 + x_1^2 - 2x_1x_2 + 2x_2^2$ Objective function Quadratic
- Subject to

$$2x_1 + x_2 \le 6$$
$$x_1 - 4x_2 \le 0$$
$$x_1 \ge 0, x_2 \ge 0$$

Constraints Linear

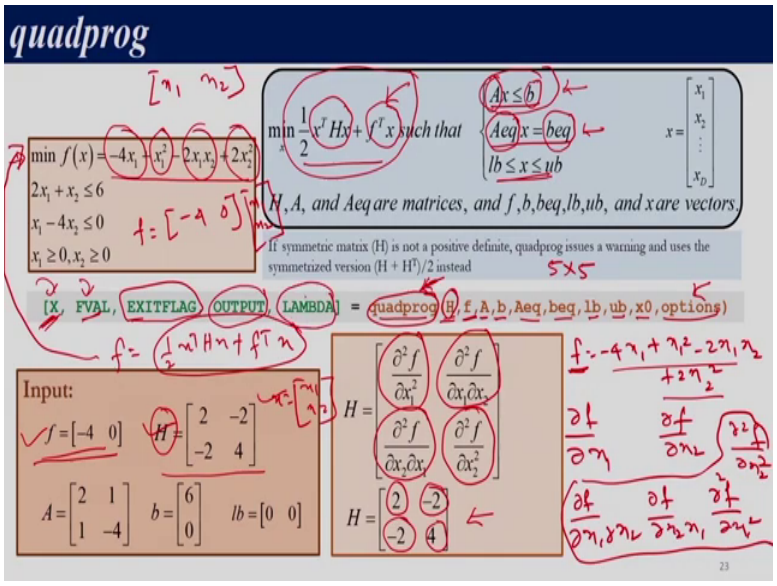
Bound constraints

- MATLAB function: *quadprog*

Engineering Optimization Theory and Practice, S.S. Rao 22

So, this is an example of quadratic programming right. So, the constraints if you see all those constraints are linear the lower bound for the decision variables are 0 the upper bounds are not given which means the upper bound is infinity. And this objective function has quadratic term right this is a quadratic term quadratic term and a the quadratic term over here. For solving this problem we can use the MATLAB function quadprog.

(Refer Slide Time: 48:32)



quadprog

$[x_1 \quad x_2]$

$$\min_x \frac{1}{2} x^T H x + f^T x \text{ such that } \begin{cases} A x \le b \\ Aeq\, x = beq \\ lb \le x \le ub \end{cases} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

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

If symmetric matrix (H) is not a positive definite, quadprog issues a warning and uses the symmetrized version $(H + H^T)/2$ instead

5x5

$$\min f(x) = -4x_1 + x_1^2 - 2x_1x_2 + 2x_2^2$$
$$2x_1 + x_2 \le 6$$
$$x_1 - 4x_2 \le 0$$
$$x_1 \ge 0, x_2 \ge 0$$

$f = [-4 \quad 0]$

```
[X, FVAL, EXITFLAG, OUTPUT, LAMBDA] = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)
```

$f = \frac{1}{2} x^T H x + f^T x$

Input:

$f = [-4 \quad 0]$ $\quad H = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$ $\quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$A = \begin{bmatrix} 2 & 1 \\ 1 & -4 \end{bmatrix}$ $\quad b = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$ $\quad lb = [0 \quad 0]$

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

$$H = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$$

$f = -4x_1 + x_1^2 - 2x_1x_2 + 2x_2^2$

$\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \frac{\partial^2 f}{\partial x_2^2}$

$\frac{\partial f}{\partial x_1 \partial x_2} \quad \frac{\partial f}{\partial x_2 x_1} \quad \frac{\partial^2 f}{\partial x_2^2}$

23

So, this is how MATLAB requires the input, so the objective function is supposed to be in this format right. So, all the quadratic terms are captured over here and all the other linear terms which are there in the objective function are captured over here right. So, for example, consider this problem minus 4 x one plus x one square minus 2 x 1 x 2 plus 2 x 2 square. So, here we have these 3 quadratic terms this is a quadratic term, this is a quadratic term, this is a quadratic term. So, only minus 4 x 1 falls under this second part right.

So, here what we will do is first we need to decide the order in which we are going to arrange the decision variables right, so let us say we will arrange x 1 and x 2. So, in this case if we compare the objective function which is given to us and this objective function which is the form that MATLAB requires right. So, then f has to be minus 4 and 0 right and x would be x 1 x 2 right; so, minus 4 x 1 plus 0 x 2 because we do not have an x 2 term which is linear in the objective function.

So, that is going to be our f this H is the hessian matrix right, hessian matrix is the second derivative of the objective functions. Dou square f by dou x one square is 2 dou square f by dou x 1, dou x 2 is minus 2. Similarly, dou x square f, dou x 2, dou x 1 is minus 2 and dou square f by dou x 2 square is 4. This will always be a matrix its size will be number of decision variables by number of decision variables. So, if you have a quadratic objective function which involves 5 decision variables right.

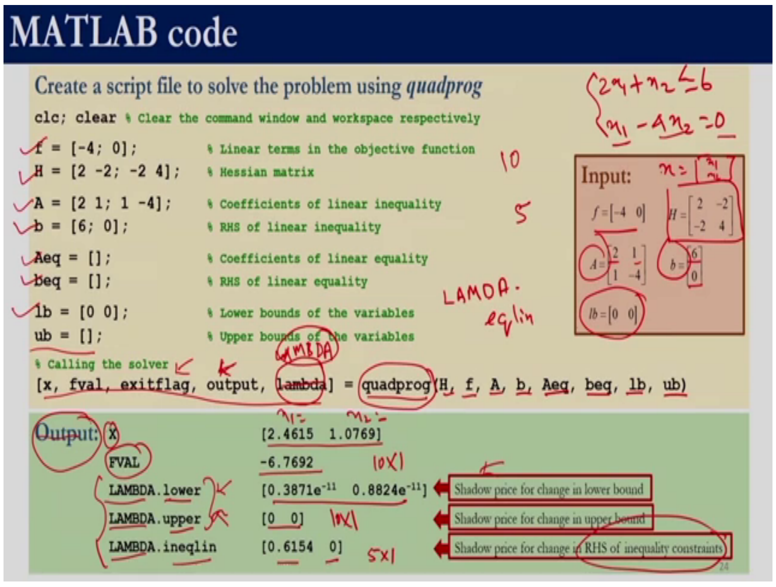
So, then you will have a 5 cross 5 matrix then the hessian matrix will be 5 cross 5. So, this hessian matrix has to be calculated by us and it has to be provided to quadprog function right, so this is H right. So, now if you do this operation half x transpose H x plus f transpose x right with the x is equal to x one x 2 right. And this f and this h and if you calculate this expression it will come down to what we have over here. The first step in solving an objective function which is quadratic in nature using quadprog is to determine the hessian matrix right.

So, once we have the hessian matrix then we can proceed ahead right. So, quadprog requires us to provide the hessian matrix and the constraints need to be of this form right. So, whatever constraints we have has to be converted into this form right. So, remember that quadprog can only support linear constraints right. So, whatever linear inequalities we have A would be the coefficient matrix b would be the right hand side vector this is for the inequality constraints. And for equality constraint the right hand side would be the constant values whereas, A equality is actually the coefficient matrix of the decision variables for linear equalities.

And then we can give this lower and upper bound right. So, the syntax for quadratic programming is quadprog is the name of the inbuilt function right. We need to provide the hessian value H right we need to provide this vector f followed by the coefficient matrix. And the right hand side vector of the linear inequalities the coefficient matrix and the right side vector of the linear equalities. The lower bound the upper bound initial guess and options we need to provide if we want to override any of the default options of quad prog.

So, the output from quadprog can be x which contains the decision variable. FVAL which contains the value of the objective function at the decision variables which quadprog has reported to be optimal. Exit flag which will tell us the reason for termination of quadprog, output which will contain additional information about the solution process and the lagrange multipliers.

(Refer Slide Time: 52:18)



MATLAB code

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

```
clc; clear % Clear the command window and workspace respectively
f = [-4; 0];              % Linear terms in the objective function
H = [2 -2; -2 4];         % Hessian matrix
A = [2 1; 1 -4];          % Coefficients of linear inequality
b = [6; 0];               % RHS of linear inequality
Aeq = [];                 % Coefficients of linear equality
beq = [];                 % RHS of linear equality
lb = [0 0];               % Lower bounds of the variables
ub = [];                  % Upper bounds of the variables
% Calling the solver
[x, fval, exitflag, output, lambda] = quadprog(H, f, A, b, Aeq, beq, lb, ub)
```

Input: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $f = [-4 \;\; 0]$ $H = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$ $A = \begin{bmatrix} 2 & 1 \\ 1 & -4 \end{bmatrix}$ $b = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$ $lb = [0 \;\; 0]$

$\begin{cases} 2x_1 + x_2 \le 6 \\ x_1 - 4x_2 = 0 \end{cases}$

10

5

LAMDA.

eqlin

LAMBDA

Output:
```
X              [2.4615  1.0769]
FVAL           -6.7692
LAMBDA.lower   [0.3871e-11  0.8824e-11]   ← Shadow price for change in lower bound
LAMBDA.upper   [0   0]                    ← Shadow price for change in upper bound
LAMBDA.ineqlin [0.6154  0]                ← Shadow price for change in RHS of inequality constraints
```

$x_1 =$ $x_2 =$ 10x1 10x1 5x1

24

So, for the problem which we showed you a couple of slides back right, so this is f this is the hessian matrix right. So, the constraints were 2 x 1 plus x 2 is less than equal to 6 x 1 minus 4 x 2 is equal to 0. So, these 2 were our constraints the coefficient matrix is 2 1 because our decision variables are x 1 and x 2 and the right-hand side value is 6 and then 0 right.

We were only given the lower bound, so lower bound is 0 0 right. So, here we need to define f H right A and b as defined over here right. We do not have any equality constraint, so we give empty matrix right. The lower bound is 0 0, so we are defining lb is equal to 0 0, so ub is an empty matrix right. So, if it is an empty matrix the upper bound is taken to be infinity right. So, here we call quadprog we provide H f A b, A equality, b equality, lower bound, upper bound which we have defined over here right. And the output that we are expecting is all these five variables which we have discussed previously.

So, in this case we get x to be 2.4615 and 1.0769 right. This means x 1 is equal to this and x 2 is equal to this and at this value of x 1 and x 2 the value of the objective function is minus 6.7692 right. So, exit flag and output we have not given over here right. So, this output is not the same as this output, this output is just to show you what is the output of MATLAB. This variable output will be a structure which will contain lot of fields it will provide us with various information about the quadprog solution.

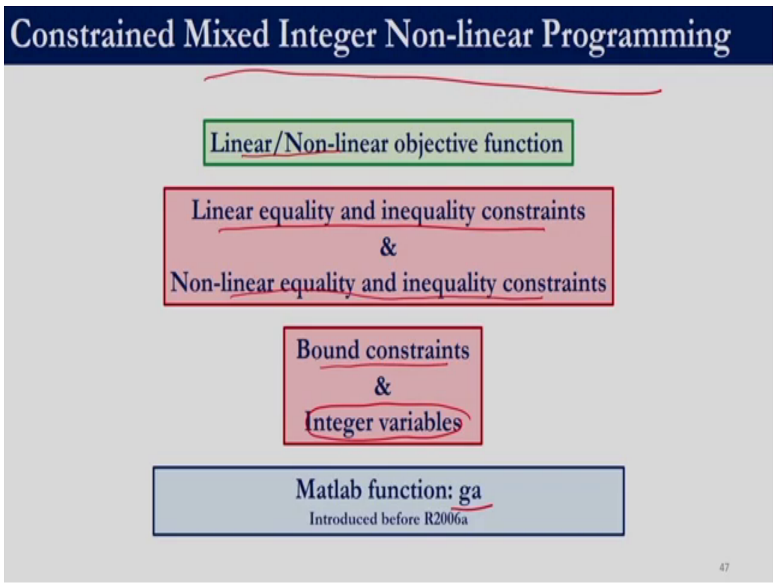
Lambda is a structure right, so you would get this only when this is in uppercase right LAMB DA right. So, lambda dot lower, lambda dot upper, lambda dot inequality right. So, this can be any variable name whatever the variable name is there that variable name would appear over here. So, we have dot lower, dot upper, and dot inequality and these two corresponds to the decision variable with respect to the lower and upper bounds; whereas, this corresponds to the inequality constraint.

So, here we have two decision variables, so this lambda dot lower will contain 2 values, and lambda dot upper will contain 2 values, and we have 2 inequality constraints, so this will also contain 2 values right. So, if we had let us say 10 variables and let us say 5 linear inequalities. In that case this would have been 5 cross 1 and these 2 would have been 10 cross 1 and 10 cross 1 right. So, the interpretation of this Lagrange multiplier is similar to the discussion which we had for linprog. And those are the shadow prices for the changes in a lower bound, upper bound, and right hand side of the inequality constraints right.

So, here if we had equality constraints then we would get something as lambda dot equivalent right which will also give us information about the lagrange multipliers with respect to the equality constraints. Now, we will look into how to solve mixed integer non-linear programming problem using the inbuilt function of MATLAB. The only function in MATLAB which supports solving mixed integer non-linear programming is ga.

When we say the inbuilt function of MATLAB can solve mixed integer non-linear programming problems using genetic algorithm. We need to remember that it can either support integer variables or it can either support equality constraint it cannot solve the problem which involves both. So, for example, if you have a mixed integer non-linear programming problem if it does not have equality only then ga can solve it.

(Refer Slide Time: 55:57)



Constrained Mixed Integer Non-linear Programming

Linear/Non-linear objective function

Linear equality and inequality constraints
&
Non-linear equality and inequality constraints

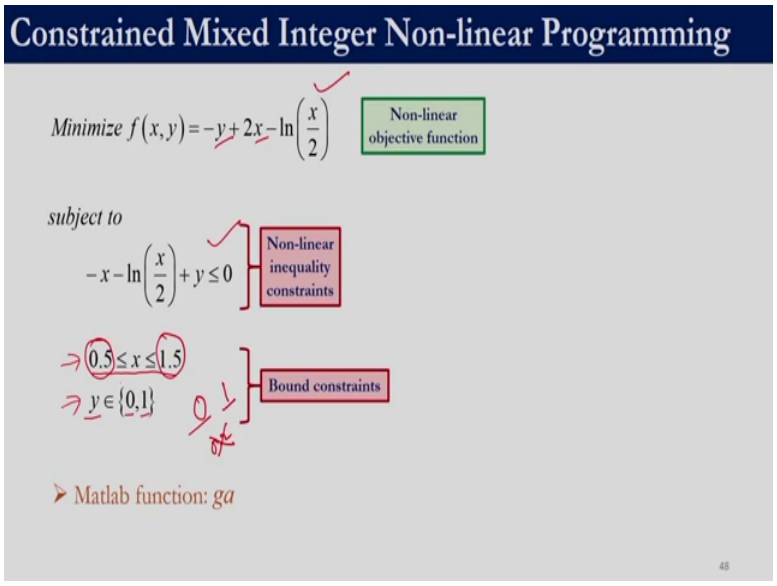
Bound constraints
&
Integer variables

Matlab function: ga
Introduced before R2006a

47

So, the function that we are looking at is ga right to solve mixed integer non-linear programming problems right. So, it can support linear equality and inequality constraint non-linear equality, and inequality the constraint. It can support linear and non-linear objective function, it can support bound constraints right, it can support integer variables only if there are no equality constraint; if there are equality constraint it cannot support integer variables.

(Refer Slide Time: 56:28)



Constrained Mixed Integer Non-linear Programming

$$\text{Minimize } f(x,y) = -y + 2x - \ln\left(\frac{x}{2}\right)$$ Non-linear objective function

subject to

$$-x - \ln\left(\frac{x}{2}\right) + y \le 0$$ Non-linear inequality constraints

$$0.5 \le x \le 1.5$$
$$y \in \{0,1\}$$ Bound constraints

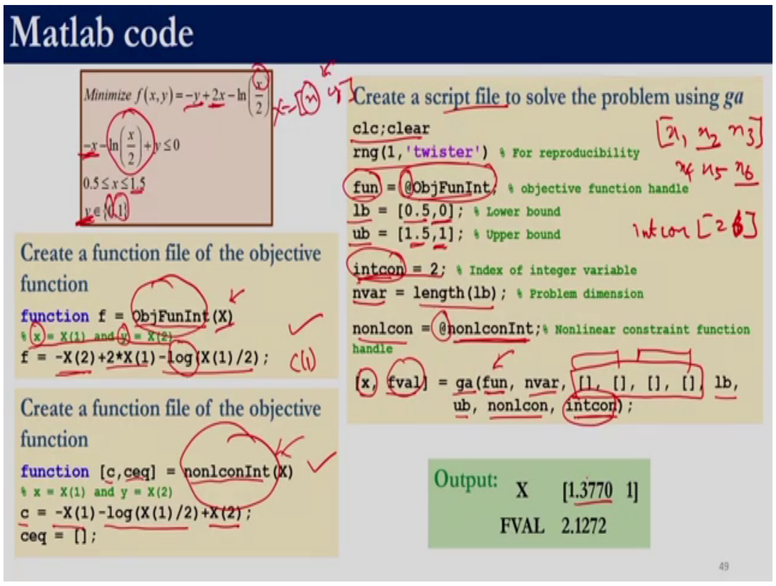
➤ Matlab function: *ga*

48

The problem that we will take over here is two variable problem. So, the decision variables are x and y x is a continuous variable the lower bound is 0.5 and the upper bound is 1.5. Whereas, y is a binary variable, so y can take either 0 or 1 it cannot take any other value apart from 0 and 1.

So, it is a binary variable, so a value of point 2 is not acceptable though it is between 0 and one that is not acceptable. So, the objective function over here is non-linear right, the only constraint that we have over here is non-linear constraint it is an inequality constraint and there are bound constraints lower and upper bound of both the variables are known.

(Refer Slide Time: 57:08)



Matlab code

Minimize $f(x,y) = -y + 2x - \ln\left(\frac{x}{2}\right)$

$-x - \ln\left(\frac{x}{2}\right) + y \le 0$

$0.5 \le x \le 1.5$

$y \in \{0,1\}$

Create a function file of the objective function

```
function f = ObjFunInt(X)
% x = X(1) and y = X(2)
f = -X(2)+2*X(1)-log(X(1)/2);
```

Create a function file of the objective function

```
function [c,ceq] = nonlconInt(X)
% x = X(1) and y = X(2)
c = -X(1)-log(X(1)/2)+X(2);
ceq = [];
```

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

```
clc;clear
rng(1,'twister') % For reproducibility
fun = @ObjFunInt; % objective function handle
lb = [0.5,0]; % Lower bound
ub = [1.5,1]; % Upper bound
intcon = 2; % Index of integer variable
nvar = length(lb); % Problem dimension
nonlcon = @nonlconInt;% Nonlinear constraint function handle
[x, fval] = ga(fun, nvar, [], [], [], [], lb,
              ub, nonlcon, intcon);
```

```
Output:  X      [1.3770  1]
         FVAL   2.1272
```

49

So, as we had seen previously for ga we need to construct these two files, so in this case what we are doing is it’s a two-variable problem. So, we are choosing a notation of x and y. So, the first value will indicate the value of x right and the second value will indicate the value of y, the name of the objective function file which we have chosen here is objfunint right.

So, the input to this would be the decision variable let us say capital x it can be any variable name right we have chosen capital x right. So, and according to our convention that the first value of x is the lower-case x and the second value of the upper-case x is the value of y. So, this is the objective function file which we require. Similarly, we can write this file nonlconint the decision values are passed by the algorithm to this function right. So, this function is supposed to return c and c equality where c and c equality is similar to what we have discussed previously for solving non-linear programming problems right.

So, here c is equal to right we have only one constraint, so that is why we are not doing c of 1, so c is equal to minus x right. So, minus x of one because the first variable indicates the x value right minus log of x 1 by 2 minus log of x 1 by 2 plus x of 2. Remember we do not have y right y is stored in this second position right. So, we are accessing x of 2 and we do not have any equality constraint. So, c equality is equal to empty bracket, so this is the script file right.

So, clc clear you know again we are fixing the random numbers right. So, we are defining the variable fun right, it has the name of the objective function file over here right since we want fun to be a function handle, we need to have this at the rate right. So, the lower bound and upper bound lower bound of the first variable is 0.5. So, 0.5 the lower bound of the second variable which is actually the integer variable is 0 right.

So, over here it is 0 the upper bound of the first variable that is x is 1.5 the upper bound of the second variable is one. So, here since we have an integer variable we need to specify which is the variable that is integer. So, in this case the second variable is integer, so we only give 2 over here this is similar to intlinprog which we have discussed previously.

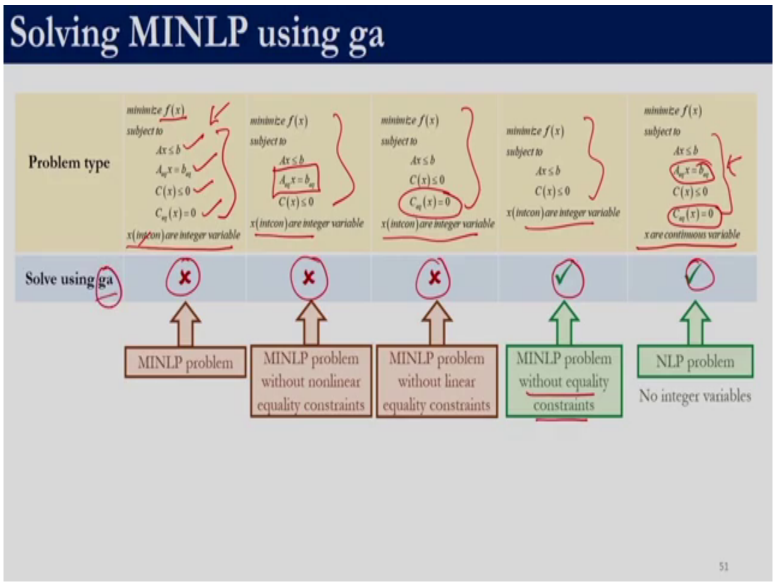
Let us say we had 3 variables let us say x 1, x 2, and x 3 and let us say the second and third variable were integers right. And we choose to arrange the variables in this order right then in that case intcon would have been 2 3. Or let us say we had x 1, x 2, x 3, x 4, x 5 and let us say x 6 and if only the second and the sixth variable had been integer, so this will be 2 and 6. So, in this case only the second variable is integer. So, we give intcon is equal to 2 the number of variables is nothing but length of lower bound or we could have just given 2.

This is a variable name right which is a function handle it refers to the file in which we have written the non-linear constraints. Again we need to specify this symbol at the rate right, so here we are solving with ga right. So, ga we need to provide the details of the objective function file. So, either we can write this one over here instead of fun or we can directly give fun nvar for the number of variables. We do not have any linear inequalities or equalities, these two are for inequalities these two are for equalit right.

So, we do not have them, so we have given empty brackets the lower bound the upper bound and the name of the file wherein we have the non-linear constraints right. So, in this case we have given nonlcon. So, it could be any variable name as long as it refers to this particular function. And previously when we had used ga to solve non-linear programming problem we had stopped over here since we now have integer variables right.

So, we also need to specify the integer variables right, so we had used this variable intcon to specify the location of the integer variables, so we provide that intcon over here right. So, as output we have only accessed the decision variable and the objective function file. We expect you to access the rest of the output provided by MATLAB and try to understand that right. So, the value of the decision variable reported by MATLAB for solving this problem is x 1 is equal to 1.3770. So, which is nothing, but the value of x right and the value of y is one over here. At this decision variable the value of the objective function is 2.1272.

(Refer Slide Time: 61:58)



Solving MINLP using ga

Problem type	minimize $f(x)$ subject to $Ax \le b$, $A_{eq}x = b_{eq}$, $C(x) \le 0$, $C_{eq}(x) = 0$, $x(intcon)$ are integer variable	minimize $f(x)$ subject to $Ax \le b$, $A_{eq}x = b_{eq}$, $C(x) \le 0$, $x(intcon)$ are integer variable	minimize $f(x)$ subject to $Ax \le b$, $C(x) \le 0$, $C_{eq}(x) = 0$, $x(intcon)$ are integer variable	minimize $f(x)$ subject to $Ax \le b$, $C(x) \le 0$, $x(intcon)$ are integer variable	minimize $f(x)$ subject to $Ax \le b$, $A_{eq}x = b_{eq}$, $C(x) \le 0$, $C_{eq}(x) = 0$, x are continuous variable
Solve using ga	✗	✗	✗	✓	✓
	MINLP problem	MINLP problem without nonlinear equality constraints	MINLP problem without linear equality constraints	MINLP problem without equality constraints	NLP problem No integer variables

51

Ah this slide is to just help you consolidate right. So, this is a typical MINLP problem right. When we say an MINLP problem this is what we mean right there are linear inequalities, there are linear equalities the objective function its not in a specific pattern right. So, it can be non-linear also we have non-linear inequalities non-linear equalities and some of the variables are integer right.

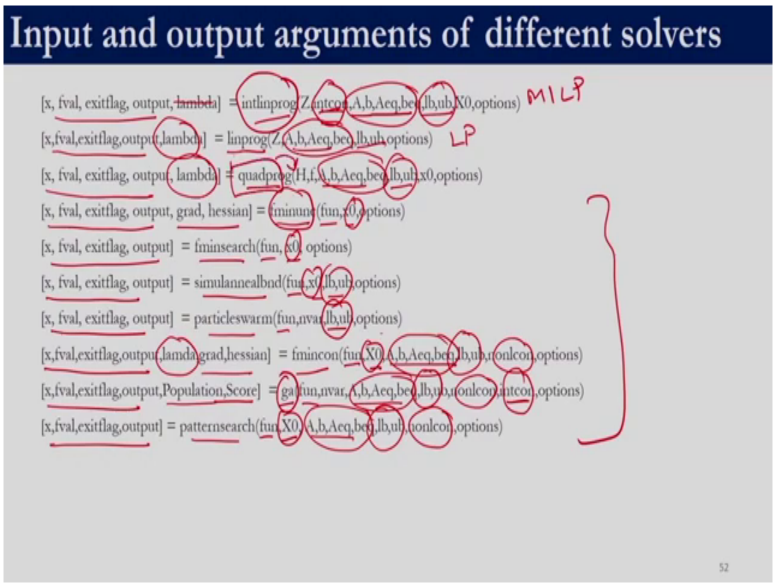
So, this intcon is written just because we are discussing about the ga function. If our problem contains all of this right then MATLABs ga function cannot solve this right. So, it cannot support integer variable with equality constraints right. So, over here if you see this is also an MINLP problem right because some of the variables are integer and there may be nonlinearities involved. But, MATLAB ga cannot solve this because we have linear equalities.

So, this is also an MINLP problem because we have nonlinearities and some variables are integer right, so here non-linear equalities are there right.

So, that is why MATLAB ga cannot solve right and if there are no integer variables if all the variables are continuous then MATLAB can solve it right. It can even support equality constraints both linear as well as non-linear because this is then NLP problem right. A non-linear programming problem there are no integer variables right this MATLAB ga function can solve, so it can solve an MINLP problem right. So, technically as per the definition of MINLP which we use for this course right. So, some variables are integer and there is some nonlinearity right

So, that way this problem also gets classified as an MINLP problem, but this is an MINLP problem without equality constraints right, so without equality constraints right. So, only this problem can be solved by the inbuilt function of MATLAB right. To the best of our knowledge there is no inbuilt function in MATLAB which can solve a really MINLP problem involving equality constraints right. Whether it is linear or non-linear does not matter if there is a equality then integer variables are not supported by MATLABs inbuilt function ga.

(Refer Slide Time: 64:01)



Input and output arguments of different solvers

```
[x, fval, exitflag, output, ~~lambda~~] = intlinprog(Z,intcon,A,b,Aeq,beq,lb,ub,X0,options)   MILP
[x,fval,exitflag,output,lambda] = linprog(Z,A,b,Aeq,beq,lb,ub,options)   LP
[x, fval, exitflag, output, lambda] = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)
[x, fval, exitflag, output, grad, hessian] = fminunc(fun,x0,options)
[x, fval, exitflag, output] = fminsearch(fun, x0, options)
[x, fval, exitflag, output] = simulannealbnd(fun,x0,lb,ub,options)
[x, fval, exitflag, output] = particleswarm(fun,nvar,lb,ub,options)
[x,fval,exitflag,output,lamda,grad,hessian] = fmincon(fun,X0,A,b,Aeq,beq,lb,ub,nonlcon,options)
[x,fval,exitflag,output,Population,Score] = ga(fun,nvar,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
[x,fval,exitflag,output] = patternsearch(fun,X0,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

52

This slide provides all the functions which we have discussed right along with its input and output right. So, linprog is the first function we discussed this is for linear programming in intprog for MINLP right. Quadprog is for quadratic programming, fminunc and fminsearch are for unconstrained optimization problem. And they do not even support bound constraints right the variables can vary between minus infinity to plus infinity only. Simulated annealing and particle swarm optimization again no constraints constraints are not allowed over here.

But the domain can have constrained lb and ub right, fmincon is used to solve non-linear optimization problems right. So, only integer variables are not allowed ga can solve NLP problems as well as MINLP problems only thing is that if there are integer variables then there cannot be equality constraints right. If integer variables are not there then equality constraints are supported, but if integer variables are there then equalities are not supported and then we had this patternsearch right.

So, the output if you see there is a similarity among all the functions right. The first 4 values are similar the decision variable, the objective function value, the reason for termination indicated by a integer number right. And output will contain additional details about the solution procedure right. So, the lagrangian multiplier can be obtained only in linprog right, quadprog and in, fmincon right. So, only there we can get the lagrangian multipliers right over here lagrangian multiplier is not returned by intlinprog. In addition to that we get gradient and hessian only in these two functions fmincon and fminunc right.

And for genetic algorithm we can additionally get the final population and the score right. So, if you look at the input right, so all of this are for non-linear. So, the objective function we supplied as a function handle right. And here we can give initial guess also right, so in this cases we can provide the initial guess. Over here there are only two functions which will support integer variables right; one is this intlinprog and the other is ga right. In intlinprog the intcon is the second variable, so in ga it is given after the non-linear constraint right.

So, only these two functions support integer variables right. So, for intcon we need to give the index of the variable which is an integer. So, this A b A equality b equality is for linear equalities and linear inequalities right. This lb ub is the lower and upper bound of the decision variables. In this course, we gave non-linear constraint through a function file right. So, we need to write the non-linear constraints in a function file and need to provide the name of the function.

Only this quadprog requires us to provide the hessian matrix. In this session we looked into how to solve non-linear programming problems as well as mixed integer non-linear programming problem using the inbuilt functions of MATLAB with that we will conclude this session.

Thank you.