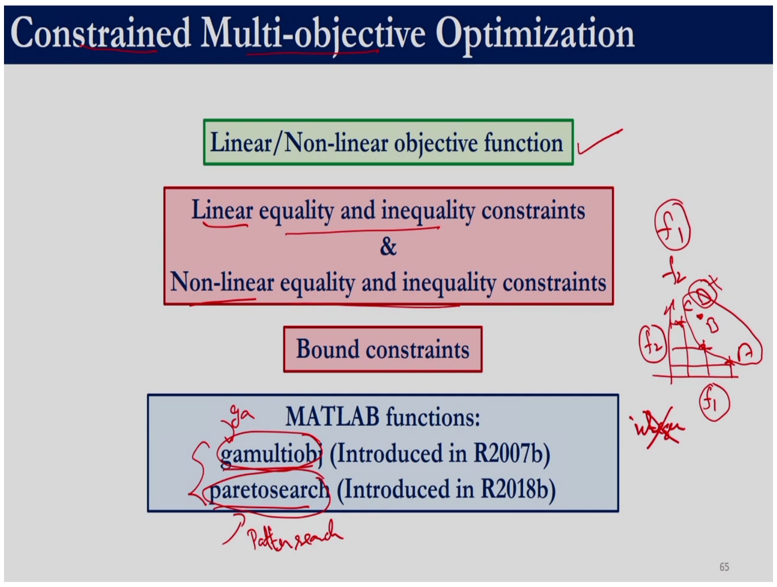
# Computer Aided Applied Single Objective Optimization

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

## Lecture – 30
## MATLAB Inbuilt Functions: Multi-objective Optimization

Welcome back, in this session we will look into how to solve Multi-objective Optimization problem using the Inbuilt Functions of MATLAB.

(Refer Slide Time: 00:39)



### Constrained Multi-objective Optimization

Linear/Non-linear objective function

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

Bound constraints

MATLAB functions:
gamultiobj (Introduced in R2007b)
paretosearch (Introduced in R2018b)

ga
Patternsearch
$f_1$
$f_2$
$f_1$

So, MATLAB has these two functions; gamultiobj and paretosearch right. So, gamultiobj is based on genetic algorithm whereas, this paretosearch is based on pattern search right.

So, both of this functions can solve multi objective constrained optimization problems, but neither of these functions support integer variables right. So, problems which do not have integer variables and are multi objective in nature can be solved by gamultiobj or paretosearch. So, here the objective function can be linear or non-linear, the constraints can be linear or non-linear though the title of this course is single objective optimization.

We are just demonstrating, how to solve multi objective optimization problem. We are not looking into the theory of it, but should you have any multi objective optimization; you can use one of these inbuilt functions to solve it right. So, just to help you recollect what is multi objective optimization. In this we have more than one objective let us say f 1 and f 2 right.

So, both of these functions are conflicting in nature. So, let us say if you want to improve f 1, the solutions have a poor value of f 2. But if we improve on f 2, the solutions have poor values on f 1 right. So, if you remember, we have seen an example in the introduction lecture; you can refer back to it right.

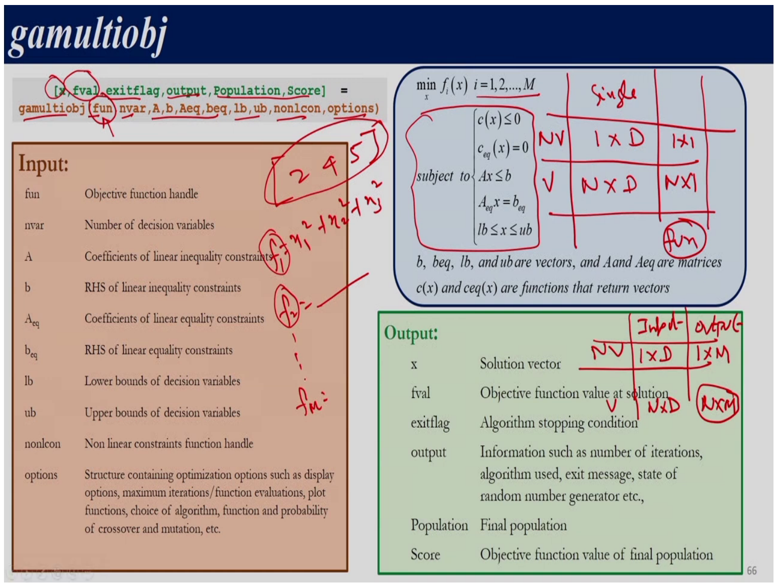
So, here what we are interested is in the set of non dominated point or the pareto points right. So, if both the objectives are to be minimized, f 1 and f 2. Remember, this is not the decision variable space; this is now the objective function space. Let us say we have these three points. So, if you analyze none of these three points are inferior to one another.

So, for example, let us say this is point A, this is point B and this is point C. So, point A has a very good f 2 value because, we are minimizing both the objectives. It has a very good f 2 value, but it has a poor f 1. Similarly, B if you see, it has a f 2 which is poor to 1 right, but it has an f 1 which is better than A.

Similarly, f C if we see it has an f 1 which is better than A as well as B, but has a f 2 value which is poor than both A and B. In this case, all these three solutions are equally good right. So, they constitute the pareto solutions and are commonly known as non dominated solutions, if we have a point over here. Let us say this is point D. So, this point D would not be a pareto point, because one of these three points A B C dominate it right.

So, we would never select a point D if we have points A B and C. So, in this case, if you see C has a better f 1 value as well as f 2 value then the solution D right. So, D is dominated by C, here our aim is to find out the set of non dominated points.

(Refer Slide Time: 03:30)



## gamultiobj

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

**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 |
| 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. |

$[2\ 4\ 5]$

$f_1 : x_1^2 + x_2^2 + x_3^2$

$f_2 :$

$f_M :$

$$\min_x f_i(x) \quad i = 1, 2, \ldots, M$$

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

| | Single | |
|---|---|---|
| NV | 1 x D | 1x1 |
| V | N x D | Nx1 |
| | | fun |

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

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

**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 |

| | Input | output |
|---|---|---|
| NV | 1xD | 1xM |
| V | NxD | NxM |

66

So, the syntax of ga multi objective is given over here right. So, we need to supply the function handle the number of decision variables details about the linear equalities and inequalities, the lower and upper bound, the function handle for non-linear constrains and options right.

So, if you see this is similar to what we have discussed for the ga in built function for single objective optimization and what we can expect from ga multi objective is the decision variable, the fitness function values of the decision variable exit flag which tells us the reason for termination output which gives us additional detail about the solution procedure, the final population and its score right.

So, the difference between single objective and multi objective optimization is going to be primarily in terms of this file which we are supplying right. So, in case of single objective optimization, let us say we have a non vectorized code. In this case, what we would send is, we would send a 1 cross D vector right. One solution right, if we have 3 D variables what we might be sending would be similar to 2, 4, 5 and let us say we have an objective function which is x 1 square plus x 2square plus x 3 square.

So, what the fitness function would return is a scalar value, this one. So, it will do 2 square plus 4 square plus 5 square and it will return that value. So, what we would be getting is, 1 cross 1 right. So, this is for single objective optimization. If it is a victorized code what we would send is a N cross D and what we would receive is a N cross 1.

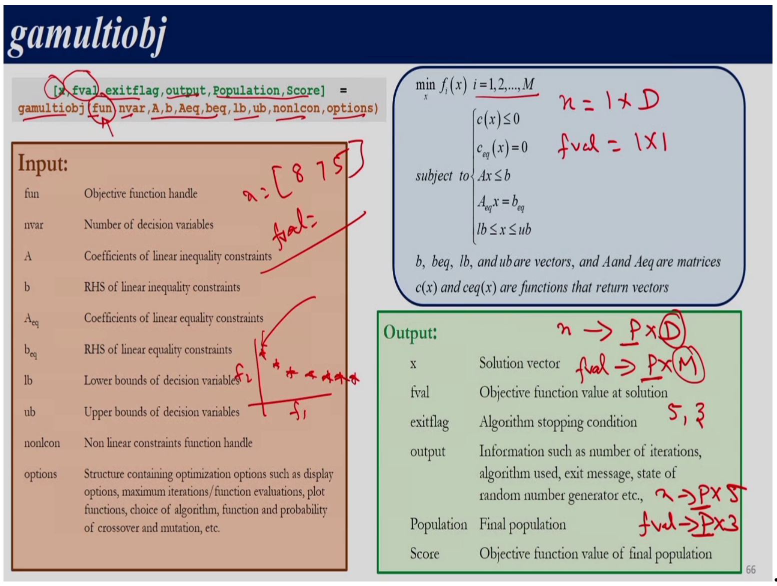
So, this is what the function file would do in terms of a single objective optimization. In case of multi objective optimization, we have more than one objective right. So, here this part of the optimization problem you are familiar with right, here we have M objective functions right. So, let us say M capital M is equal to 2 so; that means, we have 2 objective function right.

In case of multi objective optimization right, If it is a non vectorized code. We would be sending a 1 cross D and what we would be getting back is a 1 cross M. Let us say this is f 1 and let us say we have f 2 is something else. So, for these 2, 4, 5, we need to calculate the value of f 1 as well as the value of f 2 right. So, similarly if there are M objective functions, there would be m sets scalar values for this one solution right.

So, this is input to the objective function and this is output from the objective function. If we have a vectorized code right so, what we would be sending is N cross D and what will be getting back from the objective function is, N cross M. For each solution, we will have m objectives. So, for N solution we will have matrix N cross M which is being written by the objective function file.

Given a solution; our function file should be capable of sending values of both the objectives, if we have two objectives or M values if we have a M objective problem. So, that is the difference in terms of the objective function file, which we need to provide right. With respect to output right in single objective optimization x which is written by the algorithm would be a 1 cross D vector right.

(Refer Slide Time: 06:40)



## gamultiobj

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

**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 |
| 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. |

$x = [8\ 7\ 5]$

$fval =$

$f_2$ $f_1$

$$\min_x f_i(x) \quad \underline{i=1,2,\ldots,M}$$

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

$x = 1 \times D$

$fval = 1 \times 1$

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

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

**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 |

$x \rightarrow \underline{P} \times \boxed{D}$

$fval \Rightarrow \underline{P} \times \boxed{M}$

5, 3

$x \rightarrow \underline{P} \times 5$

$fval \rightarrow \underline{P} \times 3$

66

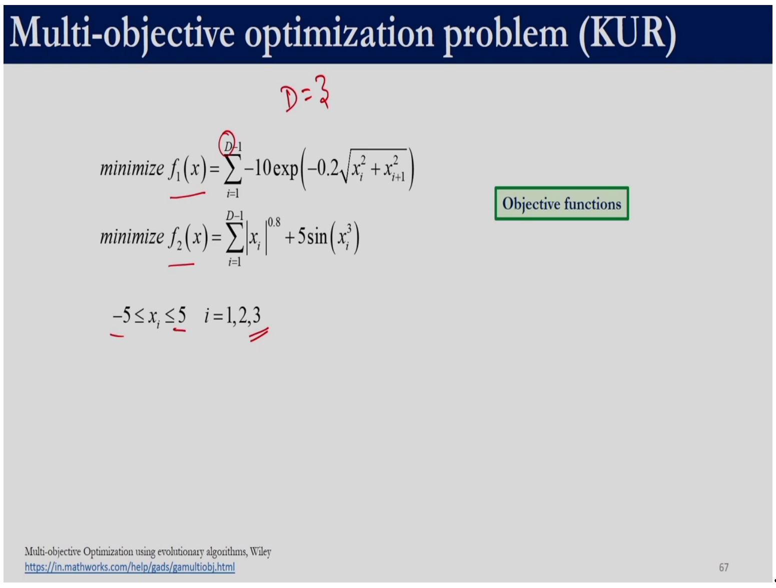
And fval which is written by the optimization algorithm would be a 1 cross 1 value right. So, if we have a three variable problem we will get let us say x is equal to 8, 7, 5 and the fitness function value corresponding to this solution would be in fval. So, that is what we will get in terms of output in case of a single objective optimization.

In case of a multi objective optimization; this x maybe a P cross D, D is the number of decision variables right and P is the number of pareto points. So, remember, we discuss about pareto points depending upon what your problem is and the performance of the algorithm you may get any number of pareto points.

So, this is f 1 and f 2, each of this point on the pareto front can be realized by a set of decision variables. So, if there are P such points on the pareto front right. So, we will get x to be a P cross D vector whereas, fval would be a P cross M matrix right. P because, the number of pareto points is P and M because the number of objectives are M right.

So, in case of let us say, if we have a 5 variable problem and let us say 3 objectives. So, the number of pareto points cannot be known a priori. So, x would be P cross 5 right whereas, fval would be P cross 3. So, this P is the number of pareto points determined by the algorithm right. So, that is the difference between single objective optimization and multi objective optimization, with respect to what we provide and what we will get from the algorithm.

(Refer Slide Time: 08:23)



## Multi-objective optimization problem (KUR)

$D=3$

$$\text{minimize } \underline{f_1(x)} = \sum_{i=1}^{\boxed{D}-1} -10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)$$

$$\text{minimize } \underline{f_2(x)} = \sum_{i=1}^{D-1} |x_i|^{0.8} + 5\sin\left(x_i^3\right)$$

$$\underline{-5} \le x_i \le \underline{5} \quad i = 1,2,\underline{3}$$

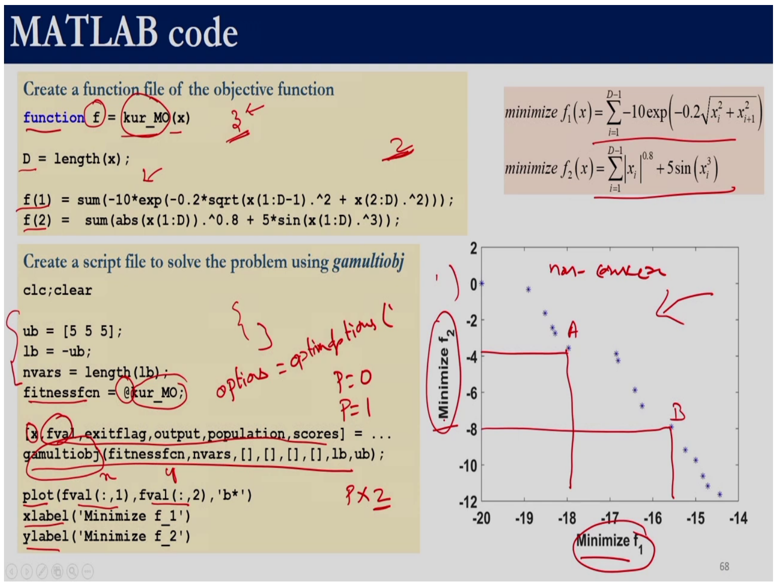
Objective functions

Multi-objective Optimization using evolutionary algorithms, Wiley
https://in.mathworks.com/help/gads/gamultiobj.html

67

So, here we have taken an example right. So, it is a three variable problem, the lower and upper bound for all the three variables are same right. So, minus 5 to 5 this is an unconstrained optimization. The two objective functions are as given over here right. This D is the number of decision variables. So, in this case we have taken three variables. So, D is equal to 3, before solving this multi objective optimization problem, we need to first define both the objectives right.

(Refer Slide Time: 08:47)



## MATLAB code

Create a function file of the objective function

```
function f = kur_MO(x)

D = length(x);

f(1) = sum(-10*exp(-0.2*sqrt(x(1:D-1).^2 + x(2:D).^2)));
f(2) =  sum(abs(x(1:D)).^0.8 + 5*sin(x(1:D).^3));
```

$$minimize\ f_1(x) = \sum_{i=1}^{D-1} -10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)$$

$$minimize\ f_2(x) = \sum_{i=1}^{D-1} |x_i|^{0.8} + 5\sin\left(x_i^3\right)$$

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

```
clc;clear

ub = [5 5 5];
lb = -ub;
nvars = length(lb);
fitnessfcn = @kur_MO;

[x,fval,exitflag,output,population,scores] = ...
gamultiobj(fitnessfcn,nvars,[],[],[],[],lb,ub);

plot(fval(:,1),fval(:,2),'b*')
xlabel('Minimize f_1')
ylabel('Minimize f_2')
```

So, here we are creating a function file kur underscore MO right. So, it will receive decision variables right and we are returning f right. So, f has two values f of 1 f of 2. So, f of 1 is determined by this expression and f of 2 is determined by this expression right. So, here we are just measuring the length of x to compute f of 1 and f of 2.

So, every time this function is called it will receive 3 elements right and it will send back 2 elements right. So, it will receive 3 elements because there are 3 decision variable and it will return 2 values because there are 2 objective functions right.

So, the rest of it is going to be similar to what we have discussed. So, here we defined the lower and upper bound the number of decision variables. We define a variable fitnessfcn which is nothing, but a function handle right. So, we refer to this file kur underscore MO and since it refers to a function file, we have this at the rate operator in front of it right and then over here we solve with the default options.

So, over here also you can change the options right. So, you can use options is equal to optim options right and the name of the solver in this case ga multi objective and whatever options you want to change you can change that and you can also pass those options to ga multi objective right.

So, over here, so the output would be x fval. So, fval remember it will be a P cross M matrix or in this case it will be a P cross 2, because there are 2 objectives. We would not a priori know the number of paeto points right. So, fval will have two columns.

So, the first column will contain the value of the first objective the second column will contain the value of the second objective. So, here we have first objective and the second objective. So, after solving it, we are plotting the first objective on the x axis and the second objective on the y axis right. So, wherever there is a point we are putting a blue color star right and then we are just adding the labels x label y label.

So, this is the plot that we would get right. So, f 1 is in the x axis and f 2 is in the y axis. So, for using this ga multi objective right. So, the optimization problem should be in terms of a minimization problem. So, the both the objectives need to be minimized right. So, here this is the pareto front. If we take any two points in this pareto front and if you analyze you will see that none of them are inferior to one another.

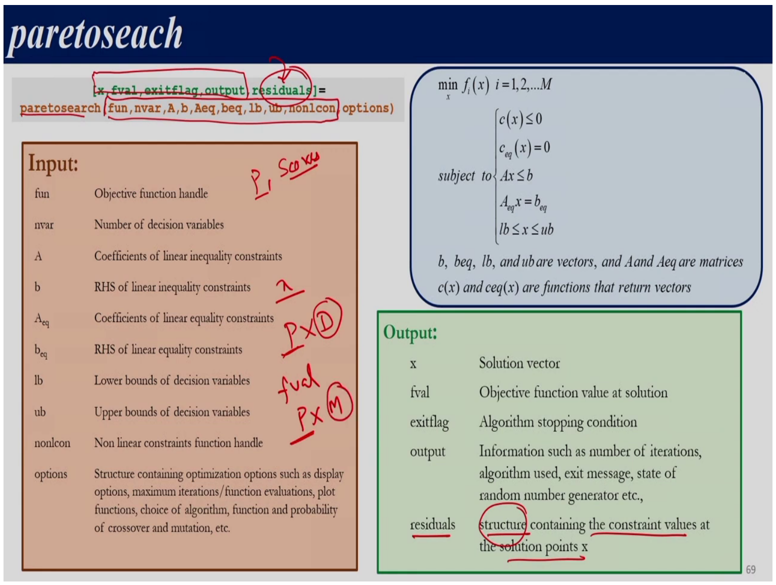
So, for example, if we take this point and let us say this point right so, let me call this as B and this as A. So, A has a better f 1 right whereas, B has a better f 2 right. So, similarly, you can analyze all of this. So, remember as we discussed earlier; the pareto front need not be convex. So, the problems can have non- convex pareto front right.

Similarly, it is not necessary that a pareto front be continuous right though it is desired that the pareto front be continuous depending upon the nature of your problem and the performance of the algorithm, you may get a discontinuous pareto front right. So, it need not be convex, it need not be continuous. So, here we did not have constraints. So, if you have constraints right and if the constraints are conflicting then they would not even be a single feasible solution.

So, if there is not a single feasible solution itself, then there is would not be any pareto point also because the pareto points need to be feasible right. So, you can have a case where in the number of pareto points is 0. So, for example, problems which do not have any feasible solution, you can have a pareto front in which the number of point is only 1.

So, a trivial example is where in the number of feasible solution itself is 1 right and the common misconception is that if you have two objectives you will get only 2 pareto points. So, that is also not true.

(Refer Slide Time: 12:46)



## paretoseach

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

$$\min_x f_i(x) \quad i=1,2,\dots M$$

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

*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 |
| 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:**

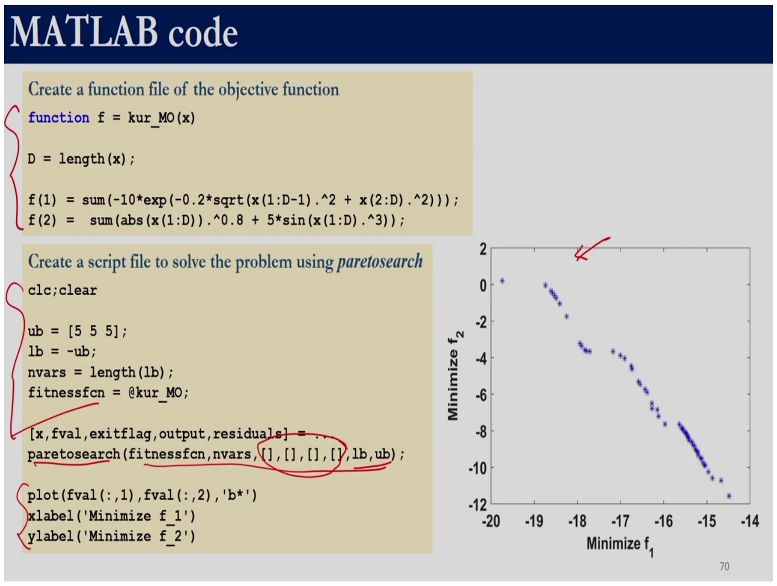
| | |
|---|---|
| x | Solution vector |
| fval | Objective function value at solution |
| exitflag | Algorithm stopping condition |
| output | Information such as number of iterations, algorithm used, exit message, state of random number generator etc., |
| residuals | structure containing the constraint values at the solution points x |

69

The other algorithm that MATLAB has for solving multi objective problems is paretosearch right. So, its uses similar to gamultiobj right, so here the name of the function is paretosearch right. The input is identical to ga multi objective output these four values are identical to ga multi objective. In gamultiobj, in addition to this 4 what we got was the final population and their scores.

So, scores are nothing, but the objective function values over here we will get residuals. So, this residuals is a structure right. So, here we chose to give a name residuals, you can give any variable name. So, it would be a structure and it will contain the constraint values at the solution point x and the dimension of x would be P cross D, where P is the number of pareto points and D is the number of decision variables in your optimization problem. fval would be P cross M right where P is the number of pareto points and M is the number of objectives that our problem has.

(Refer Slide Time: 13:47)



## MATLAB code

Create a function file of the objective function

```
function f = kur_MO(x)

D = length(x);

f(1) = sum(-10*exp(-0.2*sqrt(x(1:D-1).^2 + x(2:D).^2)));
f(2) =  sum(abs(x(1:D)).^0.8 + 5*sin(x(1:D).^3));
```

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

```
clc;clear

ub = [5 5 5];
lb = -ub;
nvars = length(lb);
fitnessfcn = @kur_MO;

[x,fval,exitflag,output,residuals] = ...
paretosearch(fitnessfcn,nvars,[],[],[],[],lb,ub);

plot(fval(:,1),fval(:,2),'b*')
xlabel('Minimize f_1')
ylabel('Minimize f_2')
```

So, the solution procedure is also same, this is the same function file which we explained earlier. So, this part of the code should also be clear to you it is exactly the same which we discuss here we call the paretosearch function and supply the necessary inputs. So, there are no constraints right. So, we give empty brackets, we have lower and upper bound of minus 5 and 5. So, that is given over here right and similar to last time we also plot the pareto points. So, this is the same problem if you recollect the previous figure and look at this figure you can see that they are not identical right.

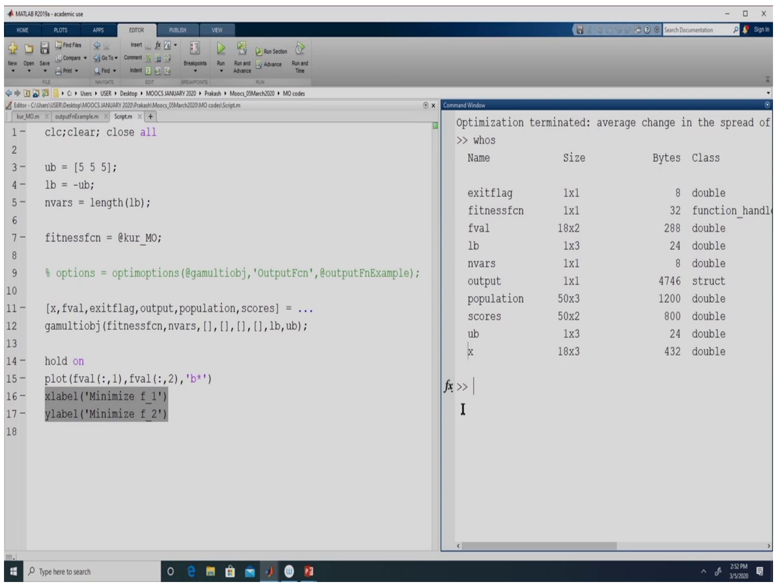
So, the performance of paretosearch and the performance of gamultiobj would be different, just like for single objective optimization gamultiobj is also a stochastic technique which is based on genetic algorithm right. So, it has to be run multiple times. So, every time we run it we may get a different pareto front. So, let us say if we run for 10 times, we will get 10 pareto fronts right. So, all of these 10 pareto fronts can be combined and we can identify the points which are actually non dominated.

So, for every time we solve, we will get a set of non dominated points, when all these points are clubbed all of them need not be non-dominated right; because we are combining the results of different runs. So, all the points can be combined and a non dominated sorting can be done. So, there are in build functions in MATLAB which can help you do non dominated sorting and that is how you can get the final pareto front from different number of runs right.

So, similar to statistical analysis that we did for single objective optimization. We can also do statistical analysis for multi objective optimization. However, for doing statistical analysis for multi objective optimization; we need additional measures right that you can look into standard multi objective optimization textbooks. We will not be discussing that over here because, the scope of our course is primarily single objective optimization. Since MATLAB has a inbuilt functions to solve multi objective optimization problems

We just discuss how to use those functions to solve multi objective optimization problems. Now we will look into the implementation of multi objective optimization.

(Refer Slide Time: 15:54)

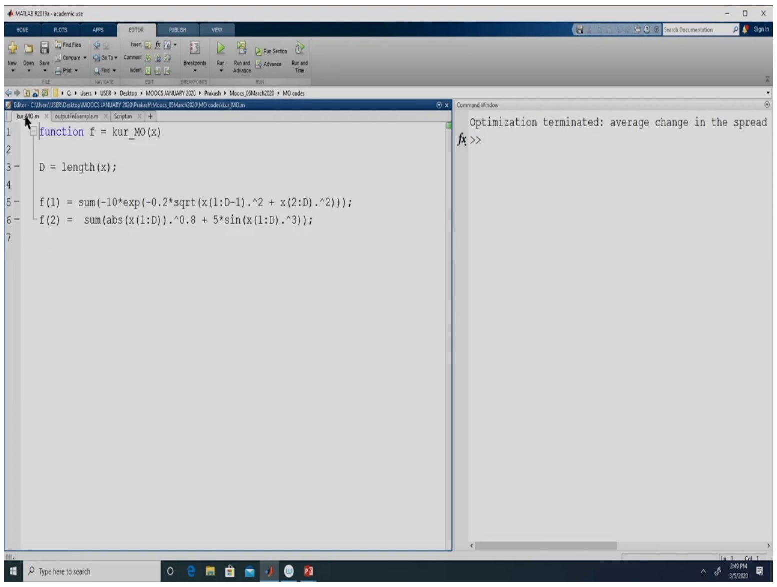


```
1  clc;clear; close all
2
3  ub = [5 5 5];
4  lb = -ub;
5  nvars = length(lb);
6
7  fitnessfcn = @kur_MO;
8
9  % options = optimoptions(@gamultiobj,'OutputFcn',@outputFnExample);
10
11 [x,fval,exitflag,output,population,scores] = ...
12 gamultiobj(fitnessfcn,nvars,[],[],[],[],lb,ub);
13
14 hold on
15 plot(fval(:,1),fval(:,2),'b*')
16 xlabel('Minimize f_1')
17 ylabel('Minimize f_2')
18
```

```
Optimization terminated: average change in the spread of
>> whos
  Name            Size            Bytes  Class

  exitflag        1x1                 8  double
  fitnessfcn      1x1                32  function_handl
  fval           18x2               288  double
  lb              1x3                24  double
  nvars           1x1                 8  double
  output          1x1              4746  struct
  population     50x3              1200  double
  scores         50x2               800  double
  ub              1x3                24  double
  x              18x3               432  double

>>
```

So, this is the script file which we have written over here right. So, we are clearing the command window, the workspace and closing all the figure files if they are open. We are defining the upper and lower bound in line 3 and 4 and then in line 5, we are just determining the number of variables, in line 7, we are assigning kur underscore MO right.

(Refer Slide Time: 06:25)



```
function f = kur_MO(x)

D = length(x);

f(1) = sum(-10*exp(-0.2*sqrt(x(1:D-1).^2 + x(2:D).^2)));
f(2) =  sum(abs(x(1:D)).^0.8 + 5*sin(x(1:D).^3));

```

```
Optimization terminated: average change in the spread
>>
```

So, that is the objective function file which we have written for this problem right. So, that we are assigning to fitness function right. So, this fitnessfcn is a function handle. So, now, let us look into this fitness function files, so kur underscore MO right. So, here we receive the decision variables and we pass the objective function values right.

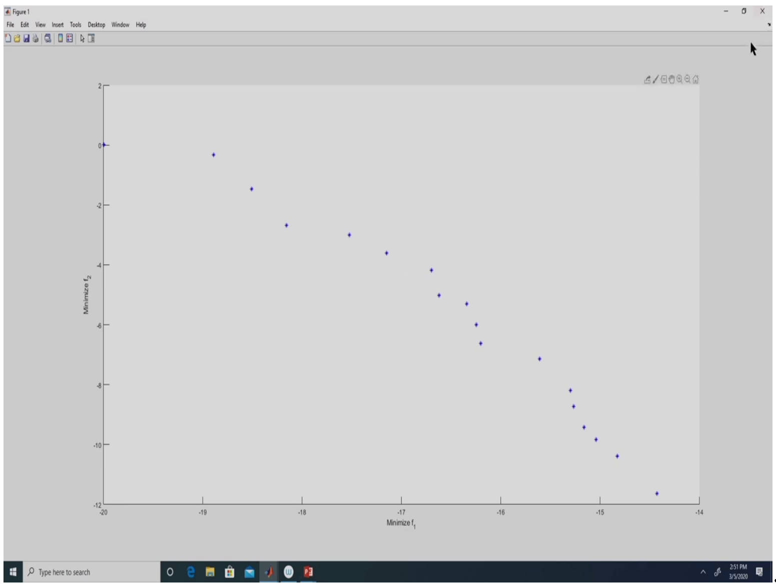
So, these are the two objective functions which we had seen earlier. We calculate f of 1 and we calculate f of 2. So, to calculate f of 1 and f of 2, we need the value of this D, has to how many decision variables are there. So, that is why we determine the length of x. So, every time, the length of x is determined and f of 1 and f of 2 are calculated. And it is returned back to genetic algorithm in this case, here we are using gamultiobj to solve the problem right.

So, this is the name of the inbuilt function gamultiobj. We are providing the problem that is to be solved right. We are supplying this name fitnessfcn. So, the number of variable in this case is 3 right. So, that we have calculated here, here we do not have linear inequalities or linear equalities, so we have given empty brackets, the lower and upper bound and we do not have any non-linear constraints right. So, we have given empty bracket.

So, this is not even required. So, as we discuss the size of this x is going to be P cross D where P is the number of pareto points which we do not know a priori only after the solution; we would know what is the number of pareto points.

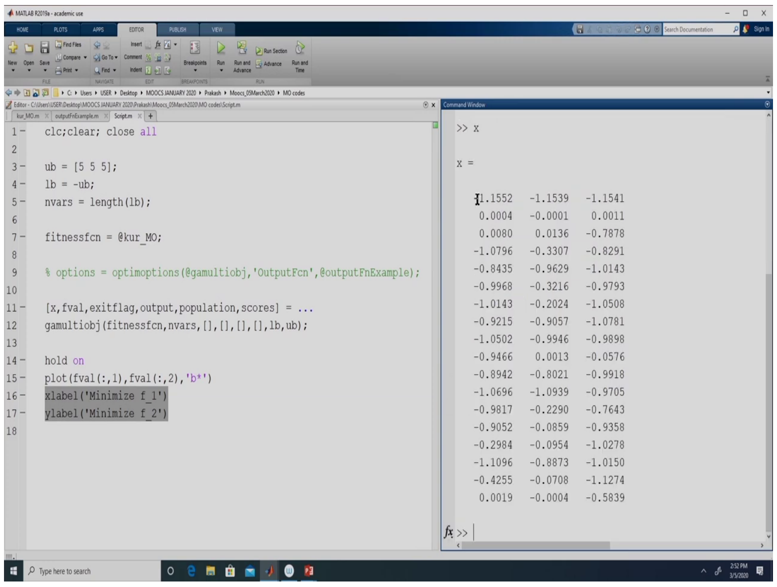
fval in this case will be P cross 2, because there are two objective functions right. Since we have two objective functions right. So, here we plot that right, so fval of all rows first column comma fval of all rows second column right. So, basically we are plotting f 1 and f 2. So, wherever we have a data point, we want to mark it with a with a blue color star right and this is just adding the x label and y label.

(Refer Slide Time: 18:05)



So, now if we execute this. So, we get this pareto front. So, this is the pareto front which we had shown you earlier right. So, it is fairly simple to solve a multi objective optimization using this function right. So, here if we look at the variables, so over here f val if we see it is 18 cross 2 and x is 18 cross 3. In this case it has determined 18 pareto points and since there are 18 pareto points and the number of decision variable is 3, we get 18 cross 3 for x right.

(Refer Slide Time: 18:37)

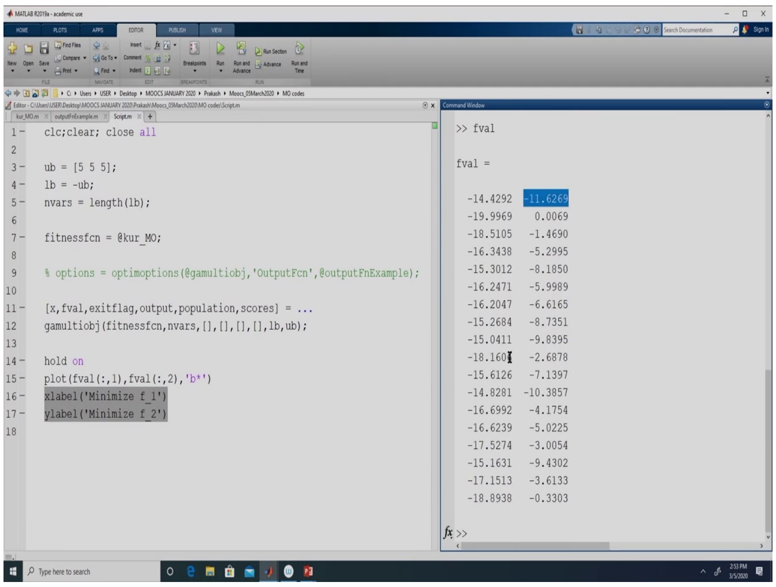


```
clc;clear; close all

ub = [5 5 5];
lb = -ub;
nvars = length(lb);

fitnessfcn = @kur_MO;

% options = optimoptions(@gamultiobj,'OutputFcn',@outputFnExample);

[x,fval,exitflag,output,population,scores] = ...
gamultiobj(fitnessfcn,nvars,[],[],[],[],lb,ub);

hold on
plot(fval(:,1),fval(:,2),'b*')
xlabel('Minimize f_1')
ylabel('Minimize f_2')
```

```
>> x

x =

   -1.1552   -1.1539   -1.1541
    0.0004   -0.0001    0.0011
    0.0080    0.0136   -0.7878
   -1.0796   -0.3307   -0.8291
   -0.8435   -0.9629   -1.0143
   -0.9968   -0.3216   -0.9793
   -1.0143   -0.2024   -1.0508
   -0.9215   -0.9057   -1.0781
   -1.0502   -0.9946   -0.9898
   -0.9466    0.0013   -0.0576
   -0.8942   -0.8021   -0.9918
   -1.0696   -1.0939   -0.9705
   -0.9817   -0.2290   -0.7643
   -0.9052   -0.0859   -0.9358
   -0.2984   -0.0954   -1.0278
   -1.1096   -0.8873   -1.0150
   -0.4255   -0.0708   -1.1274
    0.0019   -0.0004   -0.5839

fx >>
```

So, these are the decision variables. So, the fitness function corresponding to each of the solution is in fval right. So, this is the fitness function. So, for the first solution which is minus 1.1552, minus 1.1539 minus 1.1541.

(Refer Slide Time: 18:53)

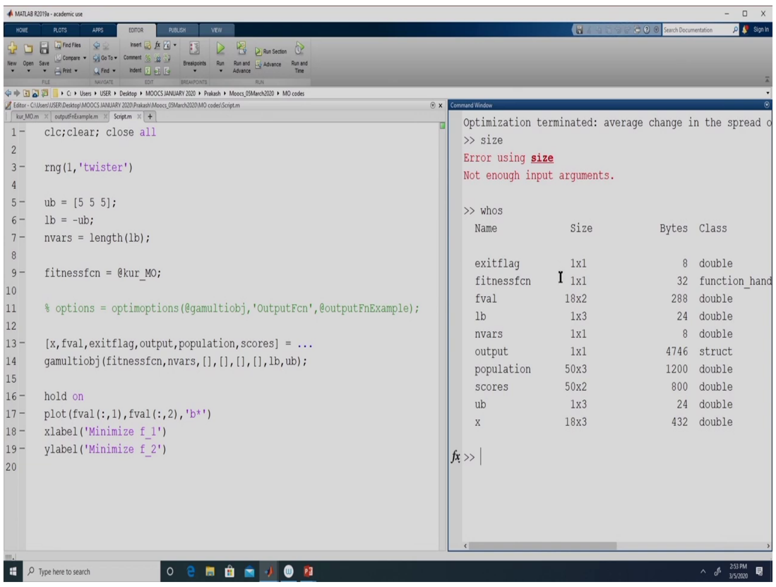


```
clc;clear; close all

ub = [5 5 5];
lb = -ub;
nvars = length(lb);

fitnessfcn = @kur_MO;

% options = optimoptions(@gamultiobj,'OutputFcn',@outputFnExample);

[x,fval,exitflag,output,population,scores] = ...
gamultiobj(fitnessfcn,nvars,[],[],[],[],lb,ub);

hold on
plot(fval(:,1),fval(:,2),'b*')
xlabel('Minimize f_1')
ylabel('Minimize f_2')
```

```
>> fval

fval =

  -14.4292  -11.6269
  -19.9969    0.0069
  -18.5105   -1.4690
  -16.3438   -5.2995
  -15.3012   -8.1850
  -16.2471   -5.9989
  -16.2047   -6.6165
  -15.2684   -8.7351
  -15.0411   -9.8395
  -18.1609   -2.6878
  -15.6126   -7.1397
  -14.8281  -10.3857
  -16.6992   -4.1754
  -16.6239   -5.0225
  -17.5274   -3.0054
  -15.1631   -9.4302
  -17.1513   -3.6133
  -18.8938   -0.3303

>>
```

The first fit objective function value is minus 14.42 the second one is minus 11.62. Similarly, we can interpret the rest of the solutions and their objective function.

(Refer Slide Time: 19:06)

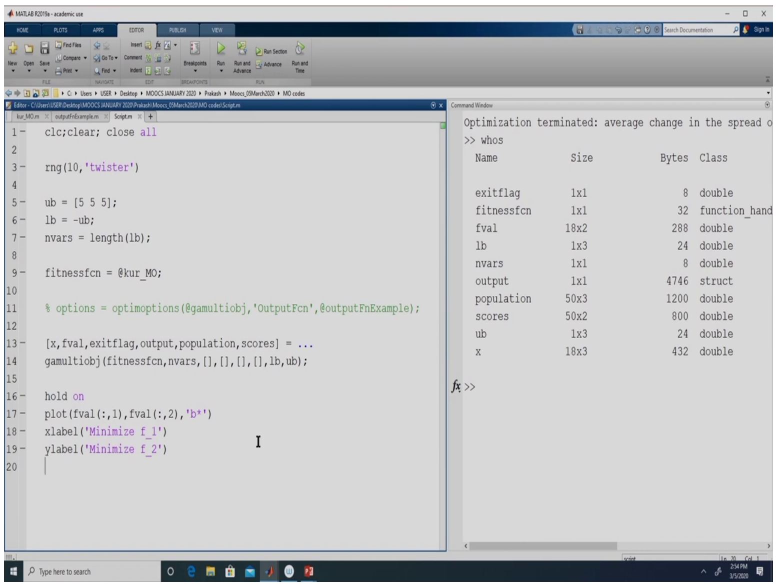


```
clc;clear; close all

rng(1,'twister')

ub = [5 5 5];
lb = -ub;
nvars = length(lb);

fitnessfcn = @kur_MO;

% options = optimoptions(@gamultiobj,'OutputFcn',@outputFnExample);

[x,fval,exitflag,output,population,scores] = ...
gamultiobj(fitnessfcn,nvars,[],[],[],[],lb,ub);

hold on
plot(fval(:,1),fval(:,2),'b*')
xlabel('Minimize f_1')
ylabel('Minimize f_2')
```

```
Optimization terminated: average change in the spread o
>> size
Error using size
Not enough input arguments.

>> whos
  Name            Size            Bytes  Class

  exitflag        1x1                 8  double
  fitnessfcn      1x1                32  function_hand
  fval           18x2               288  double
  lb              1x3                24  double
  nvars           1x1                 8  double
  output          1x1              4746  struct
  population     50x3              1200  double
  scores         50x2               800  double
  ub              1x3                24  double
  x              18x3               432  double

>>
```

So, again since this is a stochastic technique, we need to fix the random number generator to be the twister algorithm and let us say the cds 1. So, now, if we execute let us see how many points we get, right. So, we get 18 points. Now, what we are ideally supposed to do is; we are supposed to run it multiple times right again we need to put a for loop over here and run it 10 times and every time change the seed and see what is the pareto front.

(Refer Slide Time: 19:34)



```
clc;clear; close all

rng(10,'twister')

ub = [5 5 5];
lb = -ub;
nvars = length(lb);

fitnessfcn = @kur_MO;

% options = optimoptions(@gamultiobj,'OutputFcn',@outputFnExample);

[x,fval,exitflag,output,population,scores] = ...
gamultiobj(fitnessfcn,nvars,[],[],[],[],lb,ub);

hold on
plot(fval(:,1),fval(:,2),'b*')
xlabel('Minimize f_1')
ylabel('Minimize f_2')
```

```
Optimization terminated: average change in the spread o
>> whos
  Name            Size            Bytes  Class

  exitflag        1x1                 8  double
  fitnessfcn      1x1                32  function_hand
  fval           18x2               288  double
  lb              1x3                24  double
  nvars           1x1                 8  double
  output          1x1              4746  struct
  population     50x3              1200  double
  scores         50x2               800  double
  ub              1x3                24  double
  x              18x3               432  double

fx >>
```

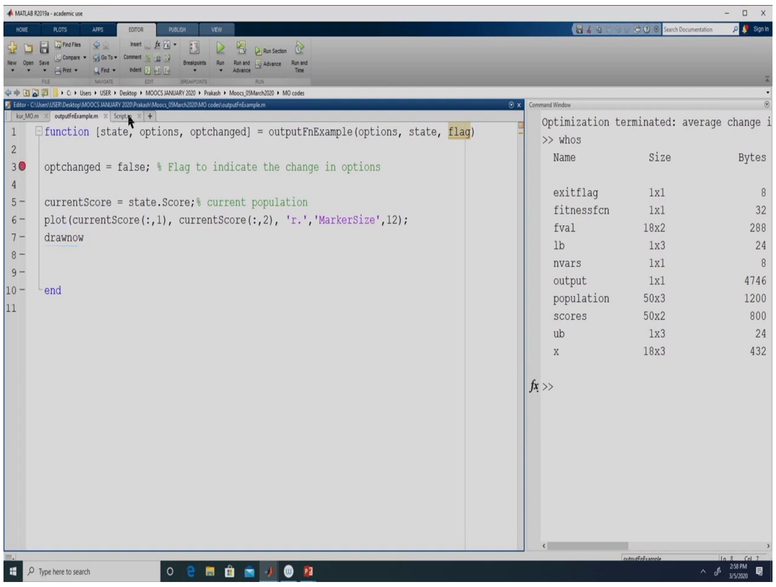
So, let us say with a seed of 10 how many pareto points do we get? So, even in this case we have got 18 pareto points right. So, it is not necessarily that we get the same number of pareto points every time. As we had discussed for single objective optimization right for multi objective optimization also we can use this optimoptions right.

So, let me uncomment this right. So, what we are going to do now is; we are going to use the output function feature of MATLAB right for solving optimization problems with respect to this function gamultiobj right. So, optimoptions, the name of the solver, the key word and this is a function file that we have written.

So, we want MATLAB to do something at the end of every iteration. So, what we want to now see is the pareto front being developed like for at the end of every iteration we want to see what is the pareto front right. So, since we want to do that we can implement it using the output function feature of MATLAB. We are using this variable name options and using this inbuilt function optimoptions to make sure that ga multi objective after every iteration executes what is given in this function file right.

So, here since we do not have a non-linear constraints, we need to put empty bracket right and remember ga multi objective does not support integer variables. So, we do not need to give a empty bracket right. So, over here we can give directly option. So, now, when MATLAB solves it will take this options which are defined over here right and here now let us look into this function file.

(Refer Slide Time: 21:11)



```
function [state, options, optchanged] = outputFnExample(options, state, flag)

optchanged = false; % Flag to indicate the change in options

currentScore = state.Score;% current population
plot(currentScore(:,1), currentScore(:,2), 'r.','MarkerSize',12);
drawnow


end
```

```
Optimization terminated: average change i
>> whos
  Name            Size            Bytes

  exitflag        1x1                 8
  fitnessfcn      1x1                32
  fval           18x2               288
  lb              1x3                24
  nvars           1x1                 8
  output          1x1              4746
  population     50x3              1200
  scores         50x2               800
  ub              1x3                24
  x              18x3               432

>>
```

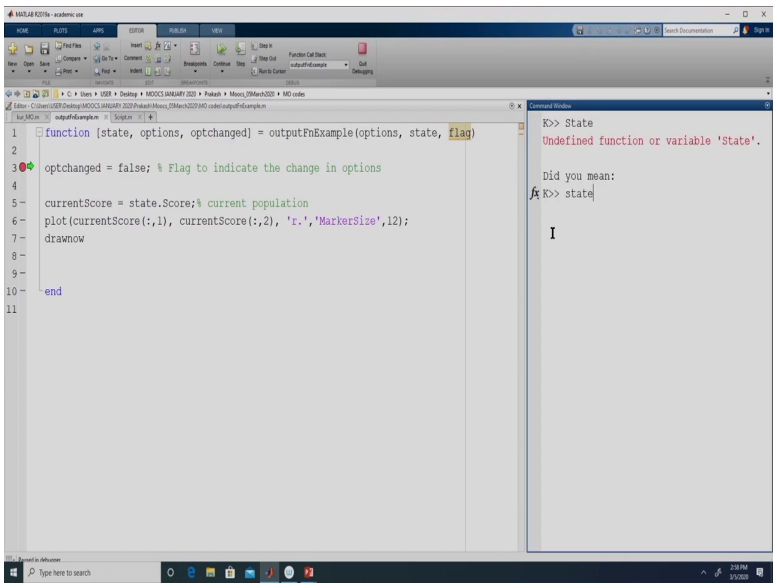
So, similar to single objective optimization; in multi objective optimization also when this function is being accessed by ga multi objective, it supplies options state and flag right. So, if we change the options in this file we need to return flag as true right. So, in this case we are not going to change the flag. So, we are just saying optchanged is equal to false and is optchanged is pass back to ga multi objective.

So, this state contains information about solutions, the objective function values can be accessed by state dot score. So, what we are doing is, we are accessing state dot Score and we are saving it in another variable currentScore right. So, it will have two columns state dot Score will have two columns. And as many rows as the number of pareto points discovered at the end of that particular iteration right.

So, just like we plotted previously, here also we plot of currentScore of the first column and currentScore of the second column. So, what we are doing is, just plotting both the objective functions, here we are plotting it with red color right and since this dot has a very small size we are increasing the Marker size to 12 and then we are asking MATLAB to immediately plotted right.

So, that we can see the pareto front actually moving as the iterations proceed, similar to last time let us just put a break point over here and execute this file.

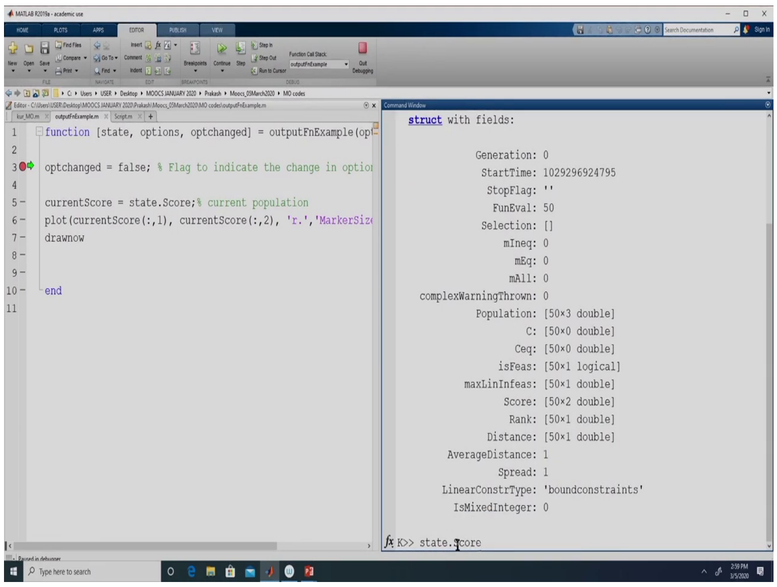
(Refer Slide Time: 22:43)



```
function [state, options, optchanged] = outputFnExample(options, state, flag)

optchanged = false; % Flag to indicate the change in options

currentScore = state.Score;% current population
plot(currentScore(:,1), currentScore(:,2), 'r.','MarkerSize',12);
drawnow


end
```

```
K>> State
Undefined function or variable 'State'.

Did you mean:
K>> state
```

(Refer Slide Time: 22:47)

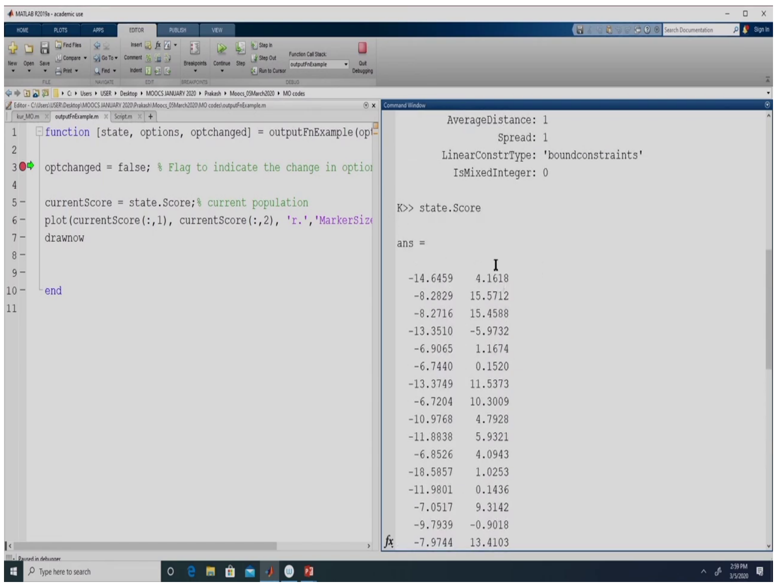


```
function [state, options, optchanged] = outputFnExample(op

optchanged = false; % Flag to indicate the change in optio

currentScore = state.Score;% current population
plot(currentScore(:,1), currentScore(:,2), 'r.','MarkerSiz
drawnow



end
```

```
  struct with fields:

              Generation: 0
               StartTime: 1029296924795
                StopFlag: ''
                 FunEval: 50
               Selection: []
                   mIneq: 0
                     mEq: 0
                    mAll: 0
    complexWarningThrown: 0
              Population: [50×3 double]
                       C: [50×0 double]
                     Ceq: [50×0 double]
                  isFeas: [50×1 logical]
             maxLinInfeas: [50×1 double]
                   Score: [50×2 double]
                    Rank: [50×1 double]
                Distance: [50×1 double]
         AverageDistance: 1
                  Spread: 1
        LinearConstrType: 'boundconstraints'
          IsMixedInteger: 0

K>> state.Score
```

So, if you look at state right. So, this is a 0 th generation right and it has this Score right. So, Score is the objective function values. So, this is the Population there are 50 members and for each of the member, we have 2 values for objective function that is why this score is 50 cross 2. So, we can access Score by just writing state dot Score. So, state is a structure and Score is a field in that structure right.

(Refer Slide Time: 23:14)

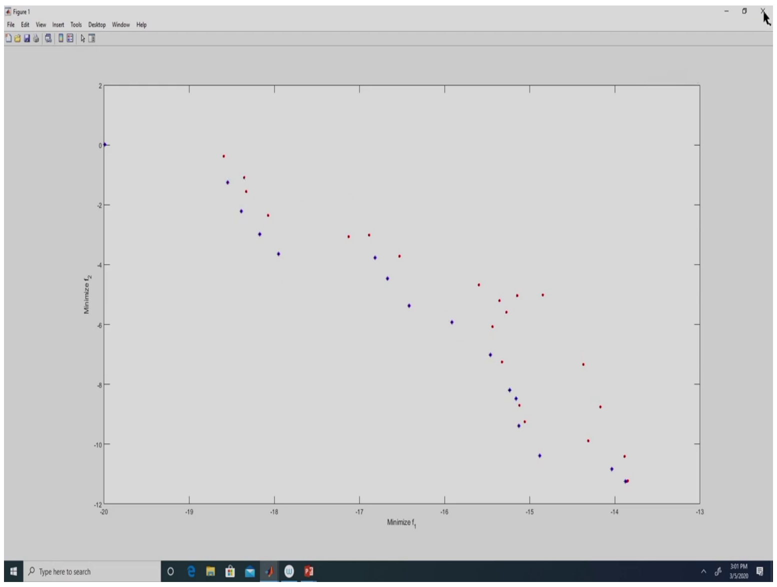


```
function [state, options, optchanged] = outputFnExample(opt

optchanged = false; % Flag to indicate the change in optio

currentScore = state.Score;% current population
plot(currentScore(:,1), currentScore(:,2), 'r.','MarkerSize
drawnow



end
```

```
         AverageDistance: 1
                  Spread: 1
         LinearConstrType: 'boundconstraints'
           IsMixedInteger: 0

K>> state.Score

ans =

  -14.6459    4.1618
   -8.2829   15.5712
   -8.2716   15.4588
  -13.3510   -5.9732
   -6.9065    1.1674
   -6.7440    0.1520
  -13.3749   11.5373
   -6.7204   10.3009
  -10.9768    4.7928
  -11.8838    5.9321
   -6.8526    4.0943
  -18.5857    1.0253
  -11.9801    0.1436
   -7.0517    9.3142
   -9.7939   -0.9018
   -7.9744   13.4103
```

So, state dot Score gives us all the values. It is just these values which we are plotting. So, minus 14.6459 in the x axis and y axis 4.1618 so, that is the point we are plotting right. So, similarly we will plot all these 50 points. So, now, let me just remove this break point and provide continue right.

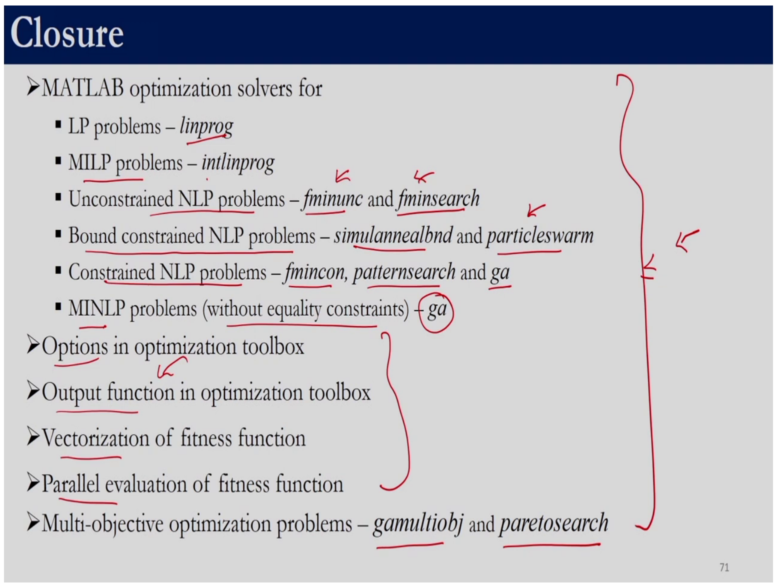
(Refer Slide Time: 23:33)



So, here if you see the pareto points are moving right. So, this is not the decision variable space, this is the objective function space, this is at the end of all the iterations right. So, the blue colour points indicate the final pareto front; whereas, the red colour dots are solutions which are dominated. Since we work with a population size of 50 which is the default population size for ga multi objective, we will have 50 points right.

So, the all these points are dominated right. So, the red color points are inferior points where is the blue color points are the trade of solutions right and they constitute the non dominated points or the pareto points. That is how we can solve multi objective optimization problems using MATLAB right.

(Refer Slide Time: 24:24)



## Closure

- MATLAB optimization solvers for
  - LP problems – *linprog*
  - MILP problems – *intlinprog*
  - Unconstrained NLP problems – *fminunc* and *fminsearch*
  - Bound constrained NLP problems – *simulannealbnd* and *particleswarm*
  - Constrained NLP problems – *fmincon, patternsearch* and *ga*
  - MINLP problems (without equality constraints) – *ga*
- Options in optimization toolbox
- Output function in optimization toolbox
- Vectorization of fitness function
- Parallel evaluation of fitness function
- Multi-objective optimization problems – *gamultiobj* and *paretosearch*

71

So, again all the features which we discuss for single objective optimization can be employed for multi objective optimization also. With that we will be concluding our discussion on inbuilt optimization functions which are available in MATLAB right.

So, as part of this, we had seen function linprog to solve linear programming problem intlinprog to solve MILP problems. We had looked into fminunc and fminsearch right. So, it was used to solve unconstrained non-linear programming problems. Remember, it does not support bounce on decision variables right.

So, for only bound constrained NLP problems like there are no linear or non-linear constraints. In that case we had discussed simulated annealing and particles from optimization.

The inbuilt function of MATLAB particleswam does not explicitly support handling of the constraint.

But as discussed in the earlier lectures; you can use a penalty approach to incorporate the constraints in the objective function and come up with a fitness function right and for constrained non-linear programming problems we had looked into fmincon patternsearch and ga; for a MINLP problems again remember only if there are no equality constraints right.

So, if there is MINLP problem with equality constraint, we cannot use ga, but we have MINLP problems, which has only inequality constraints linear as well as non-linear we can use ga to solve MINLP problems.

And we also looked into this four features available in MATLAB with respect to optimization solvers right. So, we looked how to over write the default options, we also discussed on how to have a output function right. So, the output function is called at the end of every iteration right.

So, if you want to modify something at the end of every iteration; we can write it in a function file and use this output function feature of a MATLAB. We also discussed on vectorization and parallel computing. Finally, we looked into these two functions of MATLAB which can be used to solve multi objective optimization problems these functions can be used only for minimization problems.

So, if we have a maximization problem; we need to convert it into a minimization problem and then use the appropriate function. With that we will conclude our discussion on MATLAB inbuilt solvers. We will now move on to two other software; GAMA and IMB ILOG CPLEX Optimization Studio.

Thank you.