Computer Aided Applied Single Objective Optimization

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

Lecture - 21
Black-Box Optimization Problems

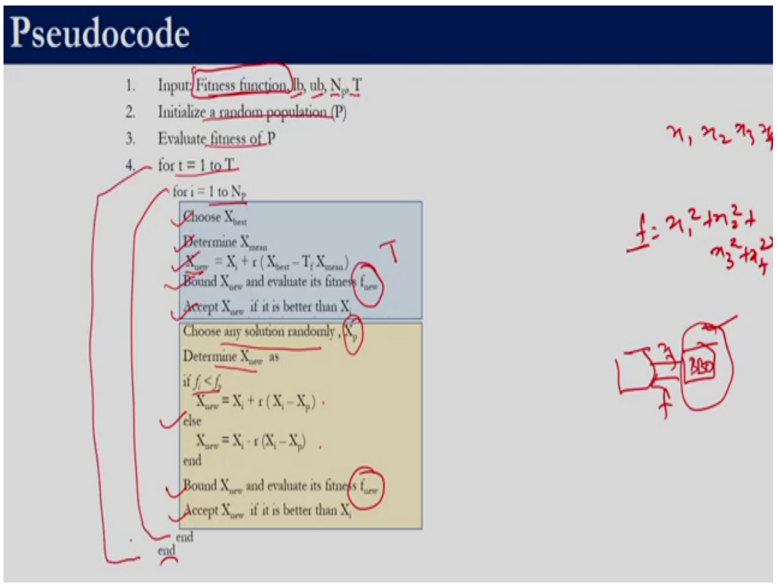
Welcome. In this session, we will be looking into what we will be calling as Black-Box Optimization Problem right. So, this we had already discussed, but here we thought we would just demonstrate it to you. So, that it stays back in our memory right. So, black box optimization problems, we are defining it as optimization problems where we do not know what is happening inside the problem right.

So, the problem is a black box. The only things that we know about the optimization problem is we know how many decision variables are there, what are the lower bounds and what are the upper bounds right. And we have a mechanism wherein if we give a set of decision variables that mechanism which we do not know right.

So, it may be an experiment or it may be a person, who is taking the values of the decision variable, but is able to give back the objective function value or the fitness function value right. So, we do not exactly know how that is being calculated, but we have this black box mechanism wherein given a set of decision variables, we are able to get the fitness function value. So, what we are discussing is true for all the five algorithms which we have discussed so far.

So, we will just demonstrate it on TLBO and it holds for all the other four algorithms. Before actually looking into it lets quickly revise the pseudo code of teaching learning based optimization right.

(Refer Slide Time: 01:45)



Pseudocode

```
1. Input: Fitness function, lb, ub, $N_p$, T
2. Initialize a random population (P)
3. Evaluate fitness of P
4. for t = 1 to T
     for i = 1 to $N_p$
       Choose $X_{best}$
       Determine $X_{mean}$
       $X_{new} = X_i + r(X_{best} - T_f X_{mean})$
       Bound $X_{new}$ and evaluate its fitness $f_{new}$
       Accept $X_{new}$ if it is better than $X_i$
       Choose any solution randomly, $X_p$
       Determine $X_{new}$ as
       if $f_i < f_p$
         $X_{new} = X_i + r(X_i - X_p)$
       else
         $X_{new} = X_i - r(X_i - X_p)$
       end
       Bound $X_{new}$ and evaluate its fitness $f_{new}$
       Accept $X_{new}$ if it is better than $X_i$
     end
   end
```

So, in teaching learning based optimization we have a fitness function right. Again this is what has become now black box right. So, previously we knew that given x 1 x 2 x 3 x 4 right the fitness function is x 1 square plus x 2 square plus x 3 square plus x 4 square. So, this is the problem that we had worked right. So, right now what we are going to do is that, we do not know how this f is calculated, but there is a mechanism to which if we give x, it will return back f right.

So, this is the black box optimization problem that we are talking about right. We need to know the lower bound and upper bound. Once we know the lower bounds and upper bounds we can; obviously, calculate how many decision variables are there. And now we need to also specify the population size and the number of iterations that we need to perform right. So, the first step is to initialize a random population and evaluate its fitness.

So, for evaluating its fitness, we have a mechanism which will give us the fitness. We do not know what is happening inside this black box optimization. Then we need to perform the t iterations right for t equal to 1 to T. So, that is this is the iterations loop and for each iteration each member is supposed to undergo the teacher phase as well as the learner phase right. So, for i is equal to 1 to N P.

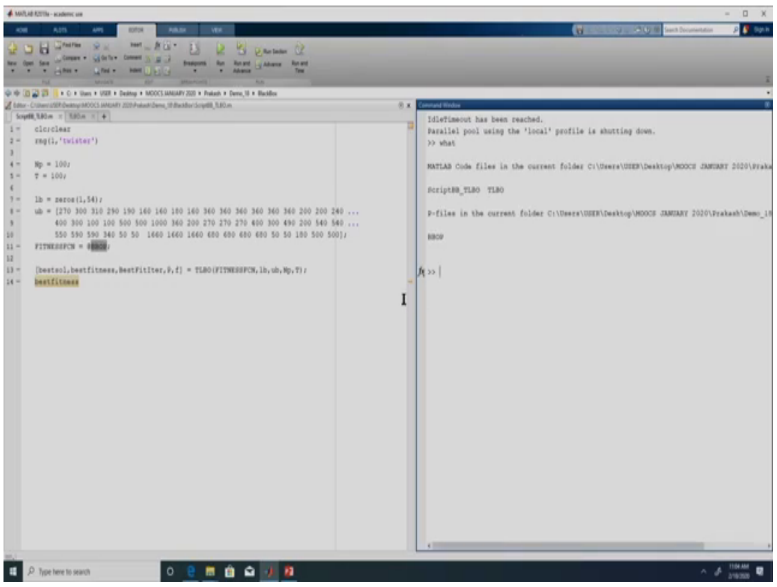
We select the best solution, we determine the mean of the population, we generate a new solution. This new solution is not guaranteed to be in the bounds. So, we employ a corner bounding strategy and evaluates its fitness function. So, again here we will be making use of this black box optimization problem. We will be accepting the new solution if it is better else, we will be discarding the new solution. So, the first member undergoes the teacher phase.

So, this is the teacher phase and then it undergoes the learner phase. So, for learners phase, we need to randomly select a member X P. So, this X P should not be the same as the ith member which is undergoing the population and we need to determine the new solution using one of this equation right. One of this equation get selected depending upon the fitness of the i th solution and the P'th partner which has been selected right.

So, once we generate a new solution from here, we again need to bound because this does not ensure that the new solution would be bounded. We need to bound and we need to evaluate the fitness function right. So, again when we want to evaluate the fitness function we will have to make use of this black box procedure right.

And we will be accepting the new solution if it is better than the i'th number which is undergoing the learner phase right. This has to be performed for all the N P members and this iteration has to be performed T times. So, this is the pseudo code of TLBO right.

(Refer Slide Time: 14:22)

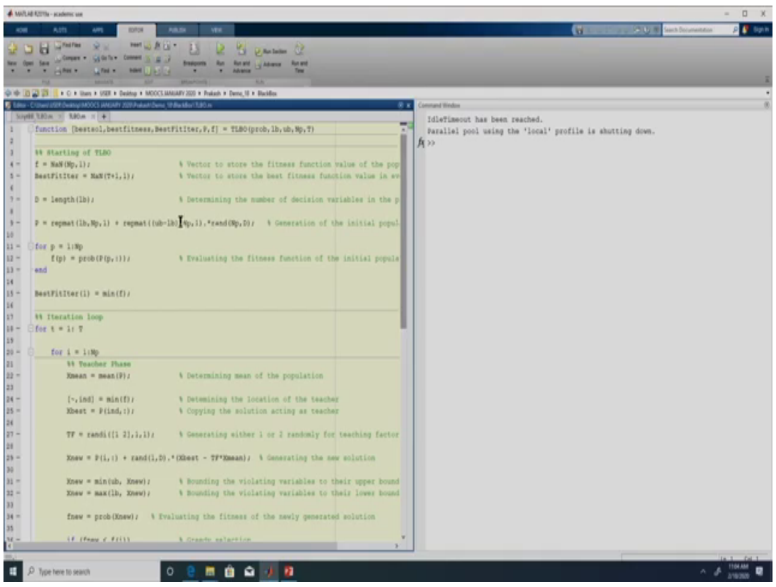


```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
    400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
    550 590 590 340 50 50  1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @RBOP;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
IdleTimeout has been reached.
Parallel pool using the 'local' profile is shutting down.
>> what

MATLAB Code files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Praka

ScriptBB_TLBO  TLBO

P-files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18

RBOP

>> 
```

So, now let us look into this right. So, here we have a the TLBO code which we have previously discussed right.

(Refer Slide Time: 04:28)

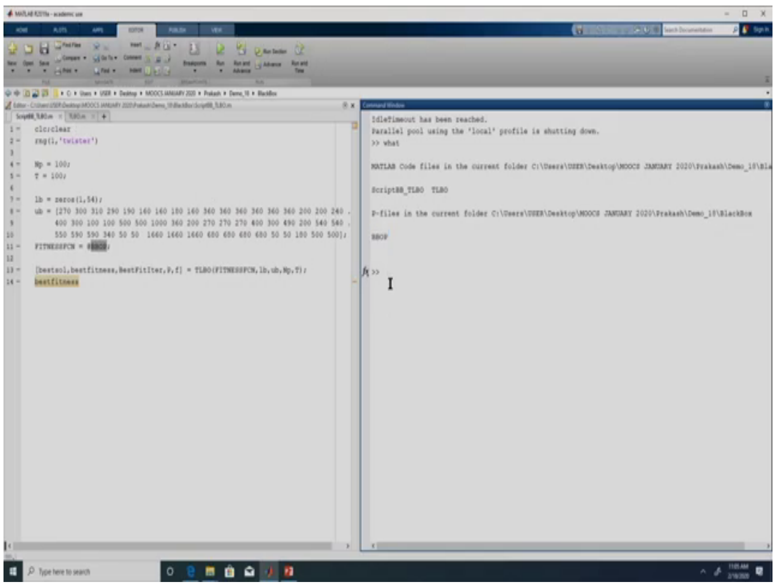


```
function [bestsol,bestfitness,BestFitIter,P,f] = TLBO(prob,lb,ub,Np,T)

%% Starting of TLBO
f = NaN(Np,1);                  % Vector to store the fitness function value of the pop
BestFitIter = NaN(T+1,1);       % Vector to store the best fitness function value in ev

D = length(lb);                 % Determining the number of decision variables in the p

P = repmat(lb,Np,1) + repmat((ub-lb),Np,1).*rand(Np,D);   % Generation of the initial popul

for p = 1:Np
    f(p) = prob(P(p,:));        % Evaluating the fitness function of the initial popula
end

BestFitIter(1) = min(f);

%% Iteration loop
for t = 1: T

    for i = 1:Np
        %% Teacher Phase
        Xmean = mean(P);        % Determining mean of the population

        [~,ind] = min(f);       % Detemining the location of the teacher
        Xbest = P(ind,:);       % Copying the solution acting as teacher

        TF = randi([1 2],1,1);  % Generating either 1 or 2 randomly for teaching factor

        Xnew = P(i,:) + rand(1,D).*(Xbest - TF*Xmean);  % Generating the new solution

        Xnew = min(ub, Xnew);   % Bounding the violating variables to their upper bound
        Xnew = max(lb, Xnew);   % Bounding the violating variables to their lower bound

        fnew = prob(Xnew);  % Evaluating the fitness of the newly generated solution

        if (fnew < f(i))        % Greedy selection
```

```
IdleTimeout has been reached.
Parallel pool using the 'local' profile is shutting down.
>>
```

So, we will not get into this because this we have already discussed right. So, here we have a problem right the problem is BBOP right, name of the file is BBOP. So, let us look at what are the files which are there in the current folder. So, in the current folder, we have three files this TLBO is the TLBO function which you can see over here. This script is this file right. So, this is a script file for black box optimization using TLBO right.

(Refer Slide Time: 05:06)



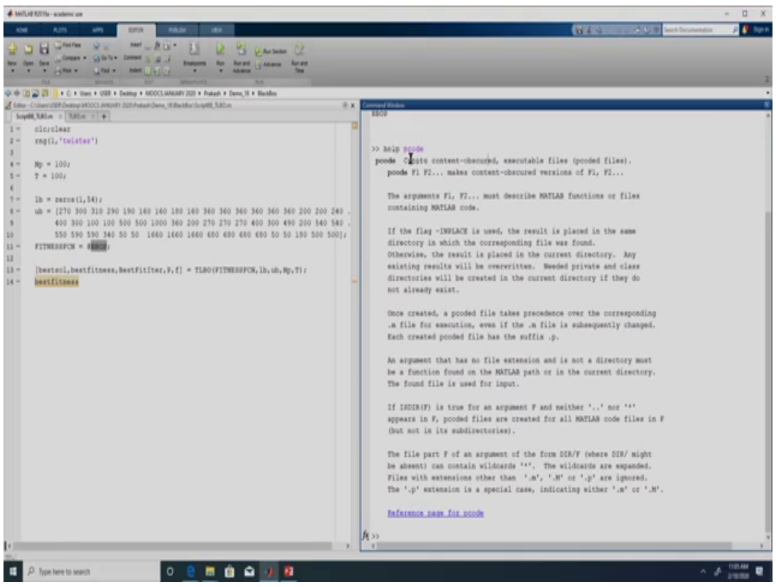
```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
      400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
      550 590 590 340 50 50  1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOP;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
IdleTimeout has been reached.
Parallel pool using the 'local' profile is shutting down.
>> what

MATLAB Code files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\Bla

ScriptBB_TLBO  TLBO

P-files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\BlackBox

BBOP

>>
```

So, this file so, both of this are m files. So, here if you see it is it is given MATLAB code files right. So, MATLAB Code files, we can open it over here and we can actually see what is happening inside right and we have this one p file named BBOP right. So, that is our fitness function. So, we do not know what is happening inside this BBOP. We cannot even look into it right.

Right now, we have this black box in terms of an p code. So, pcode if you want to look at the help of pcode, you can do help pcode.

(Refer Slide Time: 05:40)



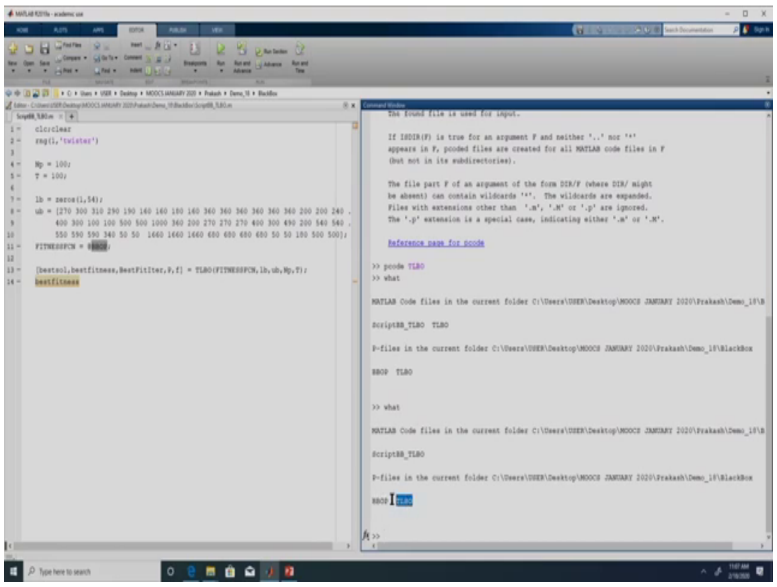
```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 .
      400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 .
      550 590 590 340 50 50  1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @EBOP;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
EBOP


>> help pcode
 pcode  Create content-obscured, executable files (pcoded files).
    pcode F1 F2... makes content-obscured versions of F1, F2...

    The arguments F1, F2... must describe MATLAB functions or files
    containing MATLAB code.

    If the flag -INPLACE is used, the result is placed in the same
    directory in which the corresponding file was found.
    Otherwise, the result is placed in the current directory.  Any
    existing results will be overwritten.  Needed private and class
    directories will be created in the current directory if they do
    not already exist.

    Once created, a pcoded file takes precedence over the corresponding
    .m file for execution, even if the .m file is subsequently changed.
    Each created pcoded file has the suffix .p.

    An argument that has no file extension and is not a directory must
    be a function found on the MATLAB path or in the current directory.
    The found file is used for input.

    If ISDIR(F) is true for an argument F and neither '..' nor '*'
    appears in F, pcoded files are created for all MATLAB code files in F
    (but not in its subdirectories).

    The file part F of an argument of the form DIR/F (where DIR/ might
    be absent) can contain wildcards '*'.  The wildcards are expanded.
    Files with extensions other than  '.m', '.M' or '.p' are ignored.
    The '.p' extension is a special case, indicating either '.m' or '.M'.

    Reference page for pcode

>>
```

So, pcode create content obscured executable files. Once a pcode is generated right so, a pcode can be generated from a m file right. Once a pcode has been generated if someone gets the pcode right, they will not be able to see what is inside it, but they can use it.

So, it is more like an executable file. So, if we can create pcode just by giving Pcode and name of whatever file we want to convert into pcode. So, for example, this TLBO ah; if we want to convert this TLBO into a pcode, we just need to give pcode TLBO. If we give this pcode, TLBO it will convert it into a pcode.

(Refer Slide Time: 06:26)

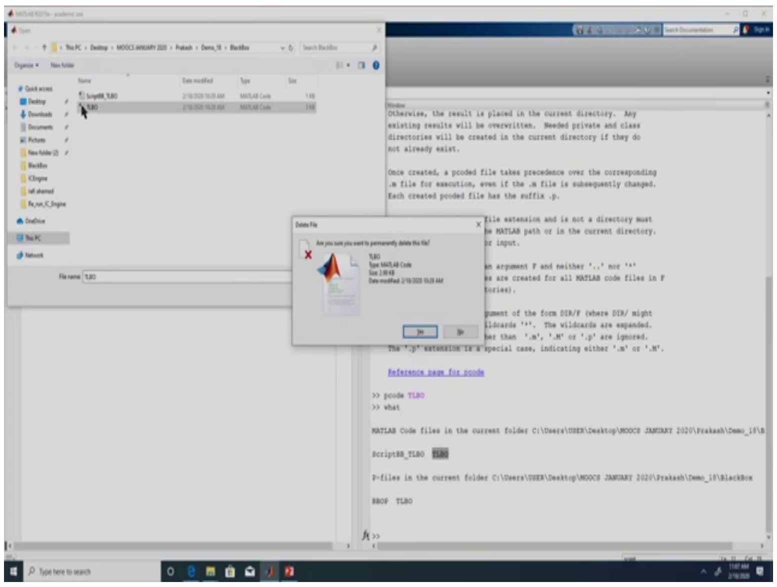


```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
      400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
      550 590 590 340 50 50  1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOP;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
    The found file is used for input.

    If ISDIR(F) is true for an argument F and neither '..' nor '*'
    appears in F, pcoded files are created for all MATLAB code files in F
    (but not in its subdirectories).

    The file part F of an argument of the form DIR/F (where DIR/ might
    be absent) can contain wildcards '*'.  The wildcards are expanded.
    Files with extensions other than  '.m', '.M' or '.p' are ignored.
    The '.p' extension is a special case, indicating either '.m' or '.M'.

    Reference page for pcode

>> pcode TLBO
>> what

MATLAB Code files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\B

ScriptBB_TLBO  TLBO

P-files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\BlackBox

BBOP  TLBO


>> what

MATLAB Code files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\B

ScriptBB_TLBO

P-files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\BlackBox

BBOP  TLBO

>>
```

So, here if we give what now if I remove this TLBO right from script from this folder right so, let us say open, let us cancel this, let me just close it first.

(Refer Slide Time: 06:39)



Are you sure you want to permanently delete this file?

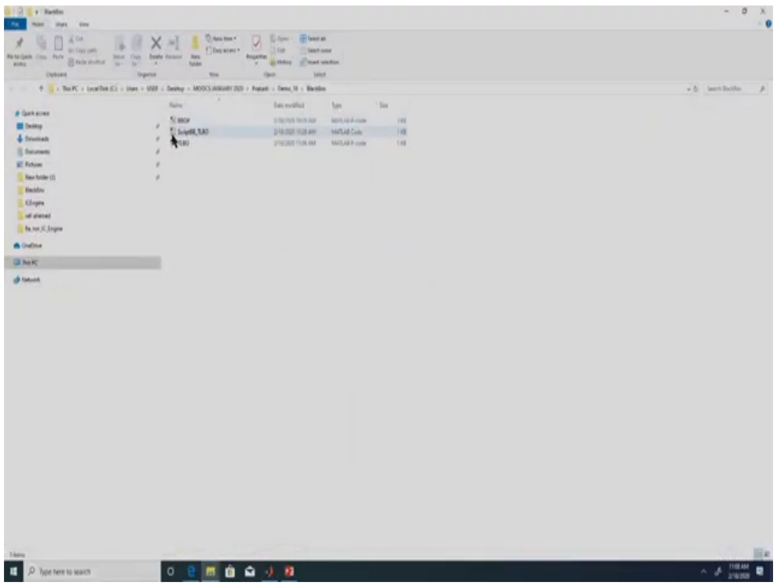
TLBO
Type: MATLAB Code
Size: 2.00 KB
Date modified: 2/18/2020 10:28 AM

```
Otherwise, the result is placed in the current directory.  Any
existing results will be overwritten.  Needed private and class
directories will be created in the current directory if they do
not already exist.

Once created, a pcoded file takes precedence over the corresponding
.m file for execution, even if the .m file is subsequently changed.
Each created pcoded file has the suffix .p.

file extension and is not a directory must
he MATLAB path or in the current directory.
r input.

n argument F and neither '..' nor '*'
s are created for all MATLAB code files in F
tories).

ument of the form DIR/F (where DIR/ might
ldcards '*'.  The wildcards are expanded.
her than  '.m', '.M' or '.p' are ignored.
The '.p' extension is a special case, indicating either '.m' or '.M'.

Reference page for pcode

>> pcode TLBO
>> what

MATLAB Code files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\B

ScriptBB_TLBO  TLBO

P-files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\BlackBox

BBOP  TLBO

>>
```

So, if I delete if I delete this right, now if we see what it has one script file. It has two pcode file; one is the algorithm and another one is the problem. So, here we showed you how to convert a file into pcode right. So, now, TLBO I can continue to use it, but I will not be able to see what is happening inside TLBO.

So, for example, over here if you see you can, you do not even get to see the pcode files this browser shows only the m files right. So, if you go to this folder and even if you try to open it with a notepad.

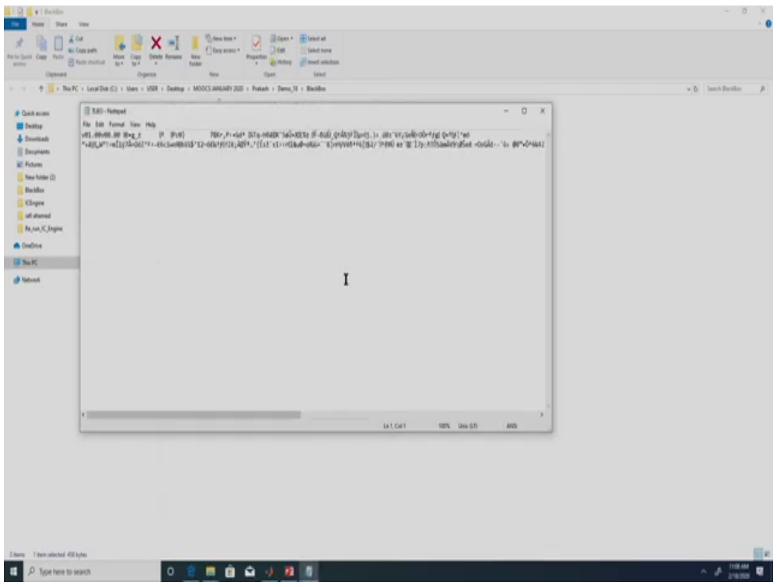
(Refer Slide Time: 07:21)



Name	Date modified	Type	Size
BBOP	[illegible]	MATLAB P-code	1 KB
ScriptBB_TLBO	2/18/2020 10:28 AM	MATLAB Code	1 KB
TLBO	2/18/2020 11:06 AM	MATLAB P-code	1 KB

It is a these two both are pcoded file. So, if I try to open it even with our notepad file, I will not be able to see what exactly is happening into it right.

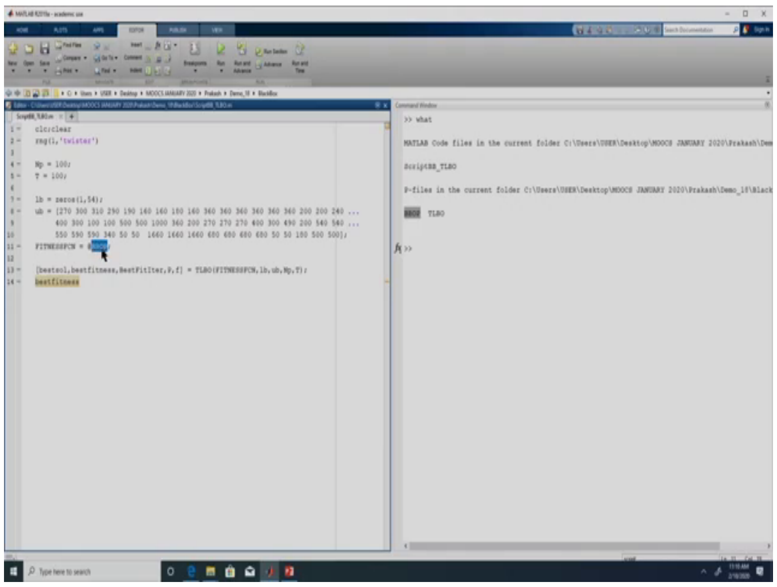
(Refer Slide Time: 07:37)



```
[illegible]
```

So, this is all that you can see. So, what we are trying to emphasize is let us say someone gives you an optimization problem. But is not willing to say what is the optimization problem, but is only willing to tell you the lower bound and upper bound. So, even then you can use one of these five materialistic technique to solve the optimization problem for them right.

(Refer Slide Time: 08:02)



```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
      400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
      550 590 590 340 50 50  1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOP;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
>> what

MATLAB Code files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Dem

ScriptBB_TLBO

P-files in the current folder C:\Users\USER\Desktop\MOOCS JANUARY 2020\Prakash\Demo_18\Black

BBOP  TLBO

>>
```

So, here currently we have this BBOP is the optimization problem that we are solving we converted into p code.

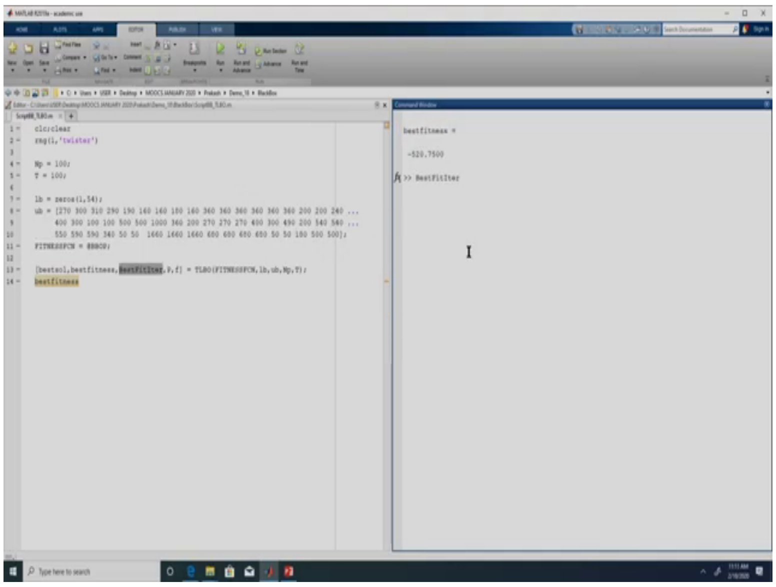
So, that you can understand how to create a p code and what are p files. Now the optimization problem BBOP is a black box for us we do not know what is happening inside that, but we know the lower bound and upper bound right. So, we can still use TLBO. So, we are calling the function TLBO and we know that the first argument has to be a function handle right which contains the problem statement.

The lower bound the upper bound the population size and the number of iterations. So, this is the input right and TLBO function if you remember or you can even download and see it. The output from TLBO function which we had written was the best solution the best fitness. So, at the end of the procedure what is the best solution that we have encountered about is its fitness function value and this is the best fitness in every iteration.

So, since here we are performing 100 iteration right. So, what is the best value at the end of every iteration that is stored in this BestFitIter. This p is the population at the end of last iteration and this is f is the fitness function corresponding to this population p right.

So, now if you understand this discussion, you would be able to appreciate that we are we do not know anything about BBOP what is happening inside a BBOP right.

(Refer Slide Time: 09:32)

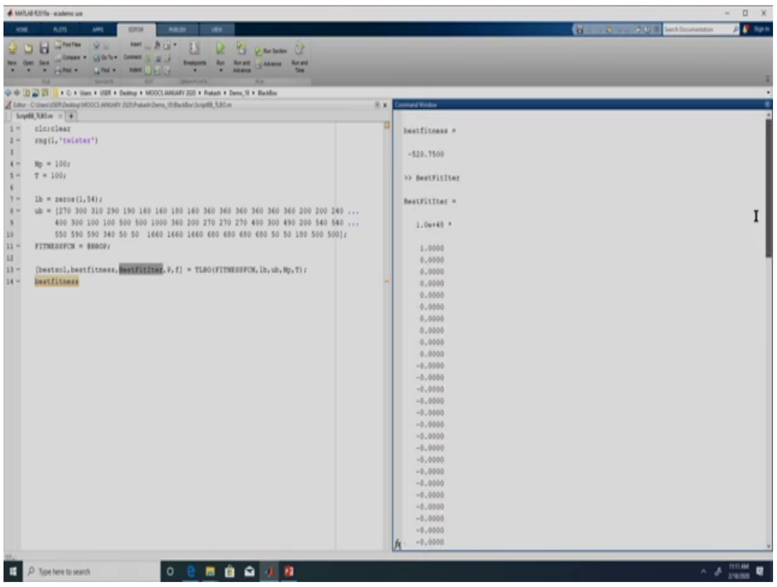


```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
      400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 580 ...
      550 590 590 340 50 50  1660 1660 1660 680 680 680 650 50 50 180 500 500];
FITNESSFCN = @BBOP;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
bestfitness =

  -520.7500

>> BestFitIter
```

Still if we run this code, you will be able to see that we will be able to optimize that problem right. So, that problem has 54 decision variables the lower bound for all the 54 decision variable is 0, the upper bound is as given over here. The best solution has a fitness function of a minus 520.75 if you see this BestFitIter right.

(Refer Slide Time: 09:48)

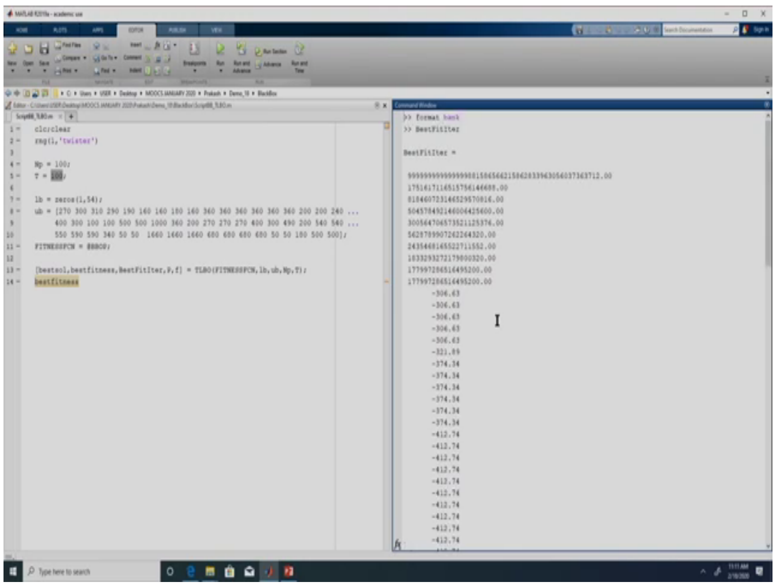


```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
    400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
    550 590 590 340 50 50  1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOF;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
bestfitness =

 -520.7500

>> BestFitIter

BestFitIter =

   1.0e+45 *

    1.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
   -0.0000
```

So, we had performed 100 iterations right. So, BestFitIter will have 101 values because of the initial population also right. So, let me put it in a better format to see all the values.

(Refer Slide Time: 10:01)

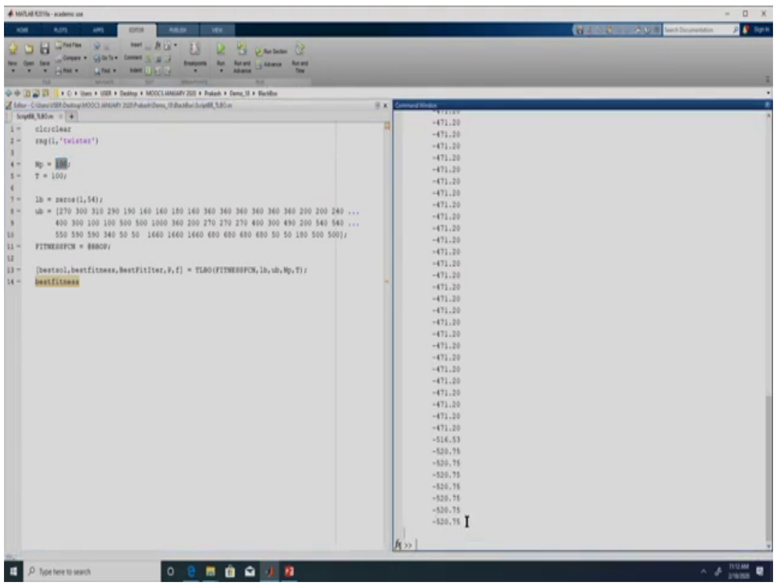


```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
      400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
      550 590 590 340 50 50  1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBO0;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
>> format bank
>> BestFitIter

BestFitIter =

 9999999999999999881586566215862833963056037363712.00
 17516171165157561466 88.00
 8184607231465295708 16.00
 5045784921460064256 00.00
 3005647065735211253 76.00
 562878990726226432 0.00
 243546816552271155 2.00
 183329327217900032 0.00
 177997286516495200.00
 177997286516495200.00
       -306.63
       -306.63
       -306.63
       -306.63
       -306.63
       -321.89
       -374.34
       -374.34
       -374.34
       -374.34
       -374.34
       -374.34
       -412.74
       -412.74
       -412.74
       -412.74
       -412.74
       -412.74
       -412.74
       -412.74
       -412.74
       -412.74
```

So, now if we see the initial population the best member had a fitness of this right and subsequently it got decreased right. So, as the algorithm progressed, we were able to get increasingly better solutions right. So, from these very large values, we came to minus 306, 374, 412, 449 minus 449 minus 471 and minus 516 minus 520 right.

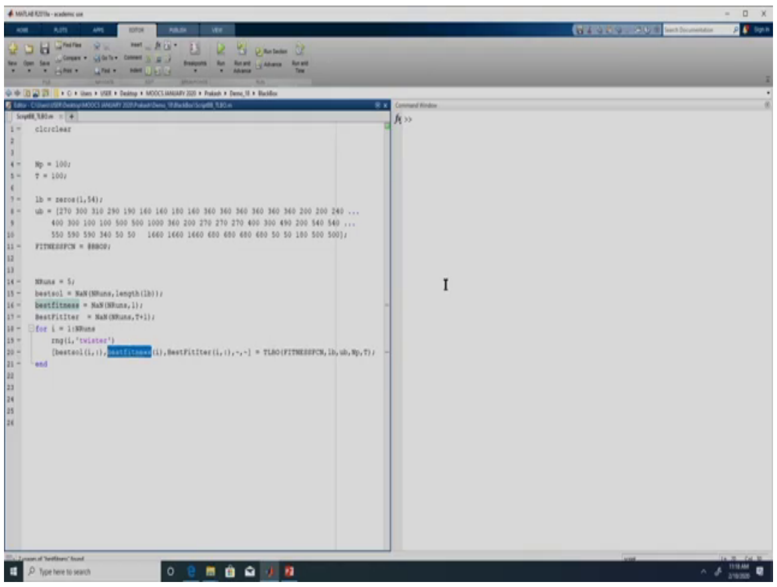
(Refer Slide Time: 10:25)



```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
     400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
     550 590 590 340 50 50  1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOB;

[bestsol,bestfitness,BestFitIter,P,f] = TLBO(FITNESSFCN,lb,ub,Np,T);
bestfitness
```

```
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-471.20
-516.53
-520.75
-520.75
-520.75
-520.75
-520.75
-520.75
-520.75

>>
```

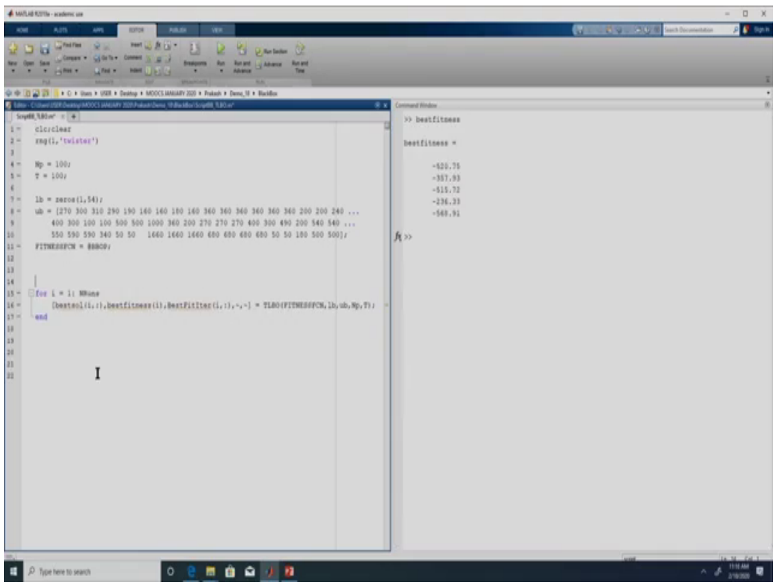
(Refer Slide Time: 11:00)



```
clc;clear


Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
    400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
    550 590 590 340 50 50   1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOB;


NRuns = 5;
bestsol = NaN(NRuns,length(lb));
bestfitness = NaN(NRuns,1);
BestFitIter  = NaN(NRuns,T+1);
for i = 1:NRuns
    rng(i,'twister')
    [bestsol(i,:),bestfitness(i),BestFitIter(i,:),~,~] = TLBO(FITNESSFCN,lb,ub,Np,T);
end
```

So, at the end of a 100 iterations with a population size of 100; the best solution that we could find was minus 520.75. Again we have run this only for one time, this we can again do the statistical analysis which we have discussed previously. So, for example, here we can say for i is equal to 1 to NRuns and over here.

(Refer Slide Time: 10:56)



```
clc;clear
rng(1,'twister')

Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
    400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
    550 590 590 340 50 50   1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOB;



for i = 1: NRuns
    [bestsol(i,:),bestfitness(i),BestFitIter(i,:),~,~] = TLBO(FITNESSFCN,lb,ub,Np,T);
end
```

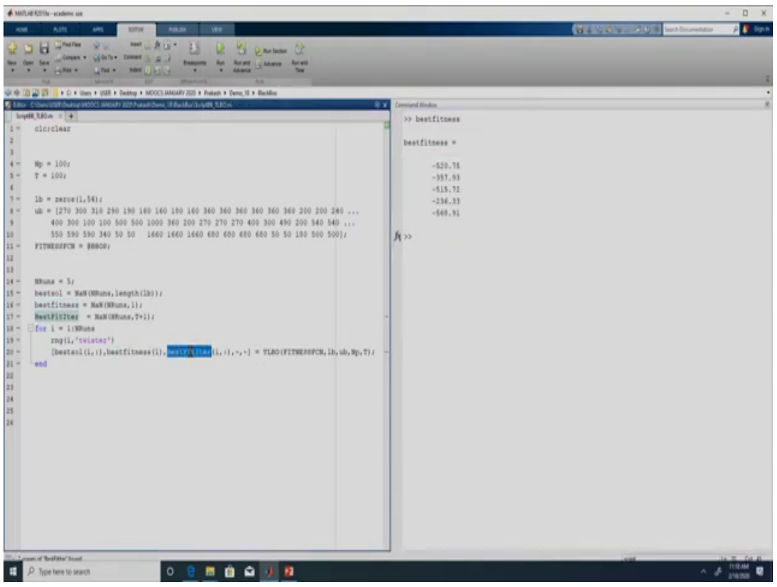
```
>> bestfitness

bestfitness =

       -520.75
       -357.93
       -515.72
       -236.33
       -568.91

>>
```

So, here if we define a number of runs is equal to file, let us say we want to implement 5 runs right. So, all this variables can be initialized right. So, right now we are initializing with a NaN right. So, right now we need to change the random number every time right. So, the (Refer Time: 11:13) we are going to change. All this we have discussed previously. If you do not remember, you will have to look into the statistical analysis video right.

So, over here every time we are executing with a different random (Refer Time: 11:32). So, this doing multiple runs we have seen previously right if you are not able to recollect you can go back and have a look at the video. So, what we are trying to show you over here is that I do not necessarily need to know what is happening inside the optimization problem right; even if I do not know I can still use this materialistic techniques right.

So, right now we have executed this whatever this optimization problem in BBOP is remember we did not even look at a what is BBOP right. The file itself was not we were not able to open right. It had this mechanism that if we give the set of decision variables, it will return us the fitness function value right. So, now, we can have a look at the best fitness right.

(Refer Slide Time: 12:09)

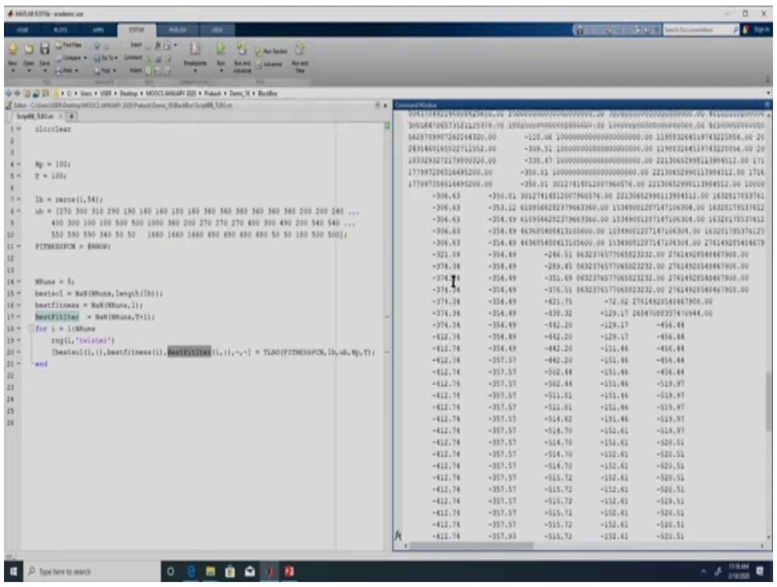


```
clc;clear


Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
    400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
    550 590 590 340 50 50   1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOP;


NRuns = 5;
bestsol = NaN(NRuns,length(lb));
bestfitness = NaN(NRuns,1);
BestFitIter  = NaN(NRuns,T+1);
for i = 1:NRuns
    rng(i,'twister')
    [bestsol(i,:),bestfitness(i),BestFitIter(i,:),~,~] = TLBO(FITNESSFCN,lb,ub,Np,T);
end
```

```
>> bestfitness

bestfitness =

       -520.75
       -357.93
       -515.72
       -236.33
       -568.91

>>
```

So, in the five runs, this is the best fitness we were able to obtain right. So, we can also have a look at this BestFitIter.

(Refer Slide Time: 12:22)

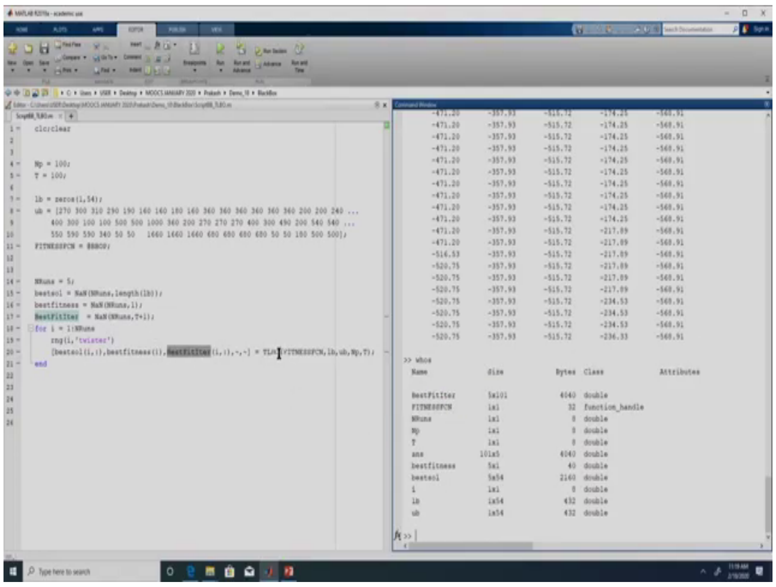


```
clc;clear


Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
    400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
    550 590 590 340 50 50   1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOF;


NRuns = 5;
bestsol = NaN(NRuns,length(lb));
bestfitness = NaN(NRuns,1);
BestFitIter  = NaN(NRuns,T+1);
for i = 1:NRuns
    rng(i,'twister')
    [bestsol(i,:),bestfitness(i),BestFitIter(i,:),~,~] = TLBO(FITNESSFCN,lb,ub,Np,T);
end
```

```
[illegible]
[illegible]
562879990726226 4320.00        -110.06 1000000000000000000000.00 1190832645197432250 56.00 20
2435460165522711552.00         -309.51 1000000000000000000000.00 1190832645197432250 56.00 20
1033293272179000320.00         -330.47 1000000000000000000000.00 2213065299811390 4512.00 171
177997206514495200.00         -350.01 1000000000000000000000.00 2213065299011390 4512.00 1716
177997206514495200.00         -350.01 3012781401200796 0576.00 2213065299011390 4512.00 10000
      -306.63       -350.01 3012781401200796 0576.00 2213065299011390 4512.00 1632017053761
      -306.63       -353.12 6189566282379663360.00 1534900120714710 6304.00 163201705 37612
      -306.63       -354.49 6189566282379663360.00 1534900120714710 6304.00 163201705 37612
      -306.63       -354.49 4636054084131056 00.00 1534900120714710 6304.00 16320178537612 5
      -306.63       -354.49 4636054084131056 00.00 1534900120714710 6304.00 27614928548467 9
      -321.09       -354.49       -246.51 8632376577065023232.00 27614928548467900.00
      -374.34       -354.49       -289.45 8632376577065023232.00 27614928548467900.00
      -374.34       -354.49       -351.69 8632376577065023232.00 27614928548467900.00
      -374.34       -354.49       -376.51 8632376577065023232.00 27614928548467900.00
      -374.34       -354.49       -421.75        -72.02 27614928548467900.00
      -374.34       -354.49       -430.32       -129.17 26047088357470944.00
      -374.34       -354.49       -442.20       -129.17       -456.44
      -412.74       -354.49       -442.20       -129.17       -456.44
      -412.74       -354.49       -442.20       -151.46       -456.44
      -412.74       -357.57       -442.20       -151.46       -456.44
      -412.74       -357.57       -502.44       -151.46       -456.44
      -412.74       -357.57       -502.44       -151.46       -519.97
      -412.74       -357.57       -511.01       -151.46       -519.97
      -412.74       -357.57       -511.01       -151.46       -519.97
      -412.74       -357.57       -514.62       -151.46       -519.97
      -412.74       -357.57       -514.70       -152.61       -519.97
      -412.74       -357.57       -514.70       -152.61       -520.51
      -412.74       -357.57       -514.70       -152.61       -520.51
      -412.74       -357.57       -514.70       -152.61       -520.51
      -412.74       -357.57       -515.72       -152.61       -520.51
      -412.74       -357.57       -515.72       -152.61       -520.51
      -412.74       -357.57       -515.72       -152.61       -520.51
      -412.74       -357.57       -515.72       -152.61       -520.51
      -412.74       -357.57       -515.72       -152.61       -520.51
      -412.74       -357.93       -515.72       -152.61       -520.51
```

So, let me just transpose it right. So, each column now indicates one run right and the number of values in BestFitIter would be 101 right.

(Refer Slide Time: 12:34)



```
clc;clear


Np = 100;
T = 100;

lb = zeros(1,54);
ub = [270 300 310 290 190 160 160 180 160 360 360 360 360 360 360 200 200 240 ...
    400 300 100 100 500 500 1000 360 200 270 270 270 400 300 490 200 540 540 ...
    550 590 590 340 50 50   1660 1660 1660 680 680 680 680 50 50 180 500 500];
FITNESSFCN = @BBOB;


NRuns = 5;
bestsol = NaN(NRuns,length(lb));
bestfitness = NaN(NRuns,1);
BestFitIter  = NaN(NRuns,T+1);
for i = 1:NRuns
    rng(i,'twister')
    [bestsol(i,:),bestfitness(i),BestFitIter(i,:),~,~] = TLBO(FITNESSFCN,lb,ub,Np,T);
end
```

```
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -174.25     -568.91
   -471.20     -357.93     -515.72     -217.09     -568.91
   -471.20     -357.93     -515.72     -217.09     -568.91
   -516.53     -357.93     -515.72     -217.09     -568.91
   -520.75     -357.93     -515.72     -217.09     -568.91
   -520.75     -357.93     -515.72     -217.09     -568.91
   -520.75     -357.93     -515.72     -217.09     -568.91
   -520.75     -357.93     -515.72     -234.53     -568.91
   -520.75     -357.93     -515.72     -234.53     -568.91
   -520.75     -357.93     -515.72     -234.53     -568.91
   -520.75     -357.93     -515.72     -236.33     -568.91

>> whos
  Name             Size             Bytes  Class              Attributes

  BestFitIter      5x101             4040  double
  FITNESSFCN       1x1                 32  function_handle
  NRuns            1x1                  8  double
  Np               1x1                  8  double
  T                1x1                  8  double
  ans            101x5               4040  double
  bestfitness      5x1                 40  double
  bestsol          5x54              2160  double
  i                1x1                  8  double
  lb               1x54               432  double
  ub               1x54               432  double

>>
```

So, if we do whos of this. So, BestFitIter it is y cross 1, 101 right. So, each of these we can plot. So, each of this is a convergence curve for the 5 runs and mean at every iteration and have a single convergence curve right.

So, all statistical analysis can be performed. So, here we have shown you how to solve a black box optimization problem using teaching learning based optimization right. The exactly same thing works for all the other four techniques which we have discussed. So, that is one of the best things about materialistic techniques is that even if you do not have knowledge about what is the optimization problem right.

We only know what are the lower bounds, what are the upper bounds and from that we know how many decision variables are there right. In addition to that if we have some mechanism right, we need not know what is the mechanism. But we have some mechanism which can be used so that we can get the fitness function value right.

So, over here the optimization problem can even be an experiment right. You get a set of decision variables, go and perform an experiment find out what is the fitness function and feed it back into the materialistic technique right or it can it can be other simulations also. So, for example, for every set of decision variables, you may have to run some other software. Let us say you have to run fluent or even a Simulink model which takes time to execute.

And once it gets executed, you get a fitness function value from it right. So, even in those cases this materialistic techniques can be easily used right. That concludes this session on black box optimization problems.

Thank you.