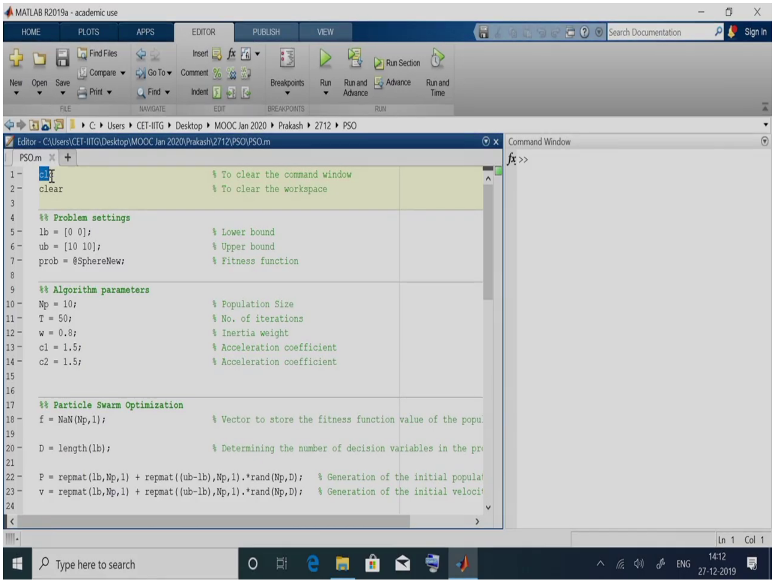
Computer Aided Applied Single Objective Optimization

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

Lecture – 11
Implementation of PSO in MATLAB

So, now we will Implement Particle Swarm Optimization on MATLAB. As we did with TLBO, first I will walk you through the code of particle swarm optimization which we already have and then we will get into the debug mode and see the execution of the code line by line, right.

(Refer Slide Time: 00:47)

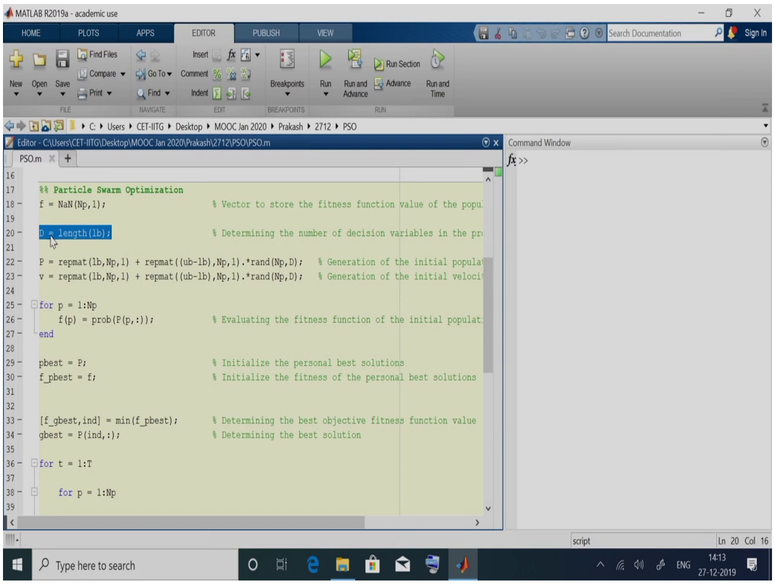


The first two lines are similar to what we did in a teaching learning based optimization that, we are clearing the command window and clearing the workspace, MATLAB workspace, right. And then we need to give the details with regards to the problem that we are solving.

So, we need to specify the lower bound. So, that we are using the variable l b to provide the lower bound.

The variable u b is used to provide the upper bound and prob is a function handle, right. So, the functions sphere new is assigned to this variable prob, and since we are using this at the rate symbol; it indicates that prob is a function handle, as and when we access prob, we will be actually accessing sphere new. Those are the three things which we require from the user as part of the problem.

(Refer Slide Time: 01:31)



And then we also require these settings to be given by the user. So, number of particles here we have taken it as 10 particles for T is equal to 50 indicates the number of iterations we want to perform, w is the inertia weight which we have initially set to 0.8, c 1 and c 2 are the acceleration coefficients with respect to the social and cognitive part, right.

So, then we define this variable f, right. So, where this variable f will be used to store the fitness function values of the each particle. So, since we have N p particles, we are creating a vector f which will have N p values; it will be a column vector and it will have N p rows. All the values are initially filled with N a N not a number, right. And then to create the initial population and the velocity we would require the number of decision variables. So, we determine the number of decision variables using this line 20, D is equal to length of l b.

So, if there are 10 variables D will be 10; if we are solving a 100 variable problem, then we would have specified 100 values for the lower bound and D will be 100.

(Refer Slide Time: 02:37)

So, P is the particle position or the solutions which we generate, we are using the same procedure that we used in TLBO, right. So, we are using the repmat function of MATLAB; obviously, there are multiple ways to do it, here we have used the repmat function. So, repmat will create N p copies of lower bound right and then again this repmat is used to create l b minus u b which is the range N p times. And then we are doing that element to element multiplication with random number generator between 0 and 1, right.

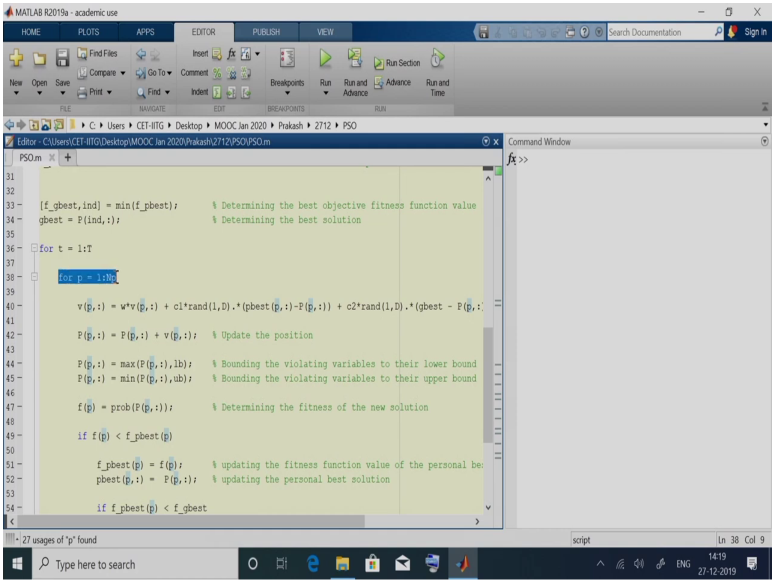
So, we will be creating N p cross D random numbers; one random number is required for every decision variable. We have D decision variable and N p population, right. So, this will this section, this second section will also be a matrix of N p cross D; this rand will also be a matrix of N p cross D. So, we are doing elemental multiplication and then we are adding to the lower bound to make sure that whatever we are generating is above the lower bound. So, that is how we generate the initial position of the particles or the initial population. We employ this similar strategy to generate the velocity; again velocity is to be generated within the bounds of the decision variables. So, we use the same strategy as in line 22, right.

So, velocity is not the same as the particle positions. So, velocity we are randomly generating within the lower and upper bounds. So, once we have generated the particles and their respective velocity, we move on to calculating the fitness of the particle, right. So, till this if you see it is similar to what we have done in teaching learning based optimization. So, here we are employing a for loop, which will run from 1 to N p; that means, from 1 to the total number of particles. For each particle we are determining the fitness using this line 26, right. To determine the fitness, we need to use the fitness function.

So, in our case the fitness function is sphere new, which is saved in the function handle prob, right. So, to prob we are sending the entire population vector. So, the entire member has to be sent. So, if there are 100 decision variable; the first row, the entire first row has to be sent. So, that is why we are sending P, the uppercase P indicates the population and the lower case p indicates the current member, right. So, p will vary from 1 to N p. So, we are sending the position or the population member one by one and determining their respective fitness.

Once we have determined the fitness of the individual particles; for the first iteration remember, we had assigned the p best to be the positions themselves, right. Same thing f of p best was nothing, but the objective function value or the fitness value itself, right. So, that is what we are doing in line 29 and line 30. So, as of now the particles do not have any previous best, right. So, the current position are assigned as best position determined the particle so far. So, we are assigning the value of p to p best. Similarly we are assigning the value of f to f of p best, right. So, p best and f of p best in the first iteration, before the beginning of the first iteration will be same as our population and the fitness function value.

(Refer Slide Time: 05:52)



So, once we have done this, in line 33 we identify the minimum objective function value or the fitness value, right.

So, when we identify the minimum value using this function min, we get the value right the minimum value in the vector f underscore p best and it is also its location. So, our g best solution is nothing, but the population located at that particular index. So, we are using this ind variable to extract that population member and are assigning it to g best, right. When we say extract, it is just that we are making a copy and the member is not removed from the population. So, once we have determined the f best and g best, we are ready to begin the iterative loop of particle swarm optimization, right

So, it is a very simple code. So, what we are doing is for t is equal to 1 to capital T, where capital T as we define is our number of iterations. So, in this case we have taken 50 to be the number of iteration, we have defined it as a user defined variable. So, whatever is there below this line 36 and till line 62 will be executed T times capital T times, right. So, for every iteration, every particle has to undergo the position and velocity update, right. So, that is why we have the second loop for p is equal to 1 to N p; for each iteration, every particle is supposed to undergo velocity and position update.

So, the next step in line 40 is to generate the new velocity. So, if you remember this was our equation, right. So, velocity of the pth particle right is equal to inertia weight into velocity of that particle right plus c 1 into random r 1, we had used r 1 over there. So, again as we had stressed upon over there, r 1 is not one particular random variable, but D variables; where D is the problem dimension. So, we create this rand of 1 comma D. So, it will give us D random numbers between 0 and 1 and that we do a elemental multiplication with p best minus p, right. So, p best is a matrix, right.

We need to extract the p'th member. So, we are saying p best of p comma colon. So, this will give us the p'th entire p'th row, right. Similarly we are using the p'th member of the population, right. So, this difference is multiplied with the acceleration co efficient c 1 and the random numbers r 1, right. So, this is the equation that we have written. Similarly c 2 again is a acceleration coefficient, which we have fixed to be 1.5; rand 1 comma D will again give us D random numbers between 0 and 1. And this g best minus p will give the difference between the global best which we have determined before beginning the iteration loop and the current population member, right.

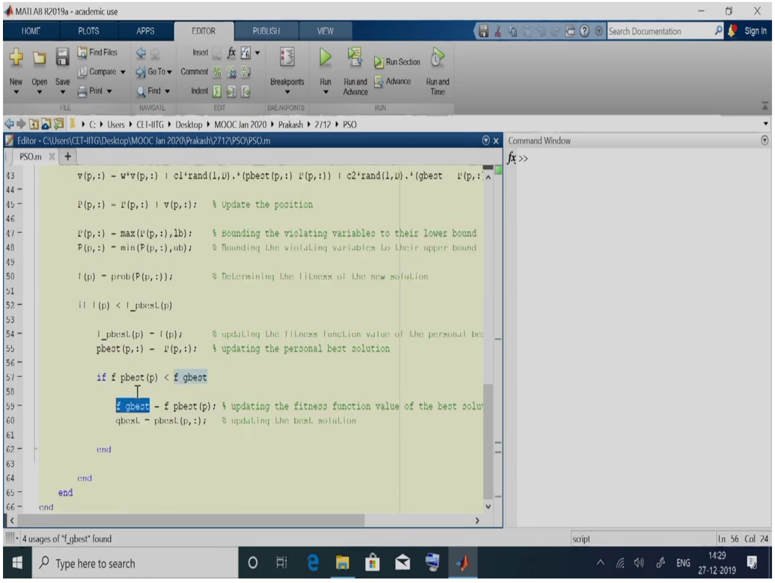
So, this equation is what we had seen in while we were doing particle swarm optimization. Once we have determined the velocity, we need to update the position, right. So, position of the p'th member now is its previous position plus the newly determined velocity, right. So, over here if you see, we are directly saving the this new solution whatever we are finding on the right hand side of line 42; directly we are including it in the population, because PSO does not employ a greedy selection strategy while updating the population, right. So, whatever population member we are generating is directly incorporated into the population matrix, right.

Once we have done that, right we need to check for the upper and lower bounds, right. So, in TLBO if you remember, we would have called it as x new; because we did not know whether that member will survive greedy selection strategy and will it go inside the population or not. But here since greedy selection is not involved for updating the population, we are directly saving it in this p matrix, right. Once we have done that we need to check the bounds, whether the newly generated population member is within the bounds or not, so again we employ the max and min function. So, line 44 will ensure that the p'th member is not violating its lower bound; whereas, line 45 will ensure that the pth member is not violating its upper bound. If it is violating, it is set to the lower and upper bound respectively, right.

So, at the end of line 45 we would have generated the new population member which is within the bounds of the decision variables, right. So, once we have generated the population member, our task is to find out it is fitness. So, we say f of p that, for this p'th member which has entered the population right; the objective function has to be evaluated. So, again we make use of the variable prob right, which is the function handle; in this case it has the sphere new function right and we pass the currently generated population member.

So, P of p comma colon right p comma colon is important, because the entire set of decision variable has to be sent to the objective function, right. So, that will return the fitness function of the p'th variable.

(Refer Slide Time: 11:12)



Once we have the fitness function of the p'th variable, we need to check whether the newly generated member is better than it is previous personal best. So, we are comparing is f of p which was currently generated is better than the best position of that particles. So, we are currently working with the p'th particle, right.

So, this p is going to vary from 1 to N p; currently we are working with the p'th member. So, we are comparing the f of p best of the pth particle with the newly generated fitness. If this conditions is valid, it executes these two conditions. So, in these two condition, we are updating the f of p best and the p best solution itself. So, if this condition is executed, we update the p best solution as well as the fitness function value of the p best, right. So, here we are assigning f of p to f of p best of p; unlike global best which is just one value f g best, here we have multiple f of p best, right. For each particle member we have it is own p best and for each p best, we have its own objective function value. We need to be careful over here and replace the p'th value.

Similarly we are replacing the p'th solution right; the newly generated solution is stored in P of p comma colon. So, that has to be assigned to f best of p comma colon, right. So, this will make sure that, we have updated the fitness function of the p best solution as well as the p best solution itself, right. So, once we are done with it, we need to check whether the newly generated solution right is better than the global best, right. So, since we have already updated f of p best right, we just check if f of p best of the p'th solution is less than g best.

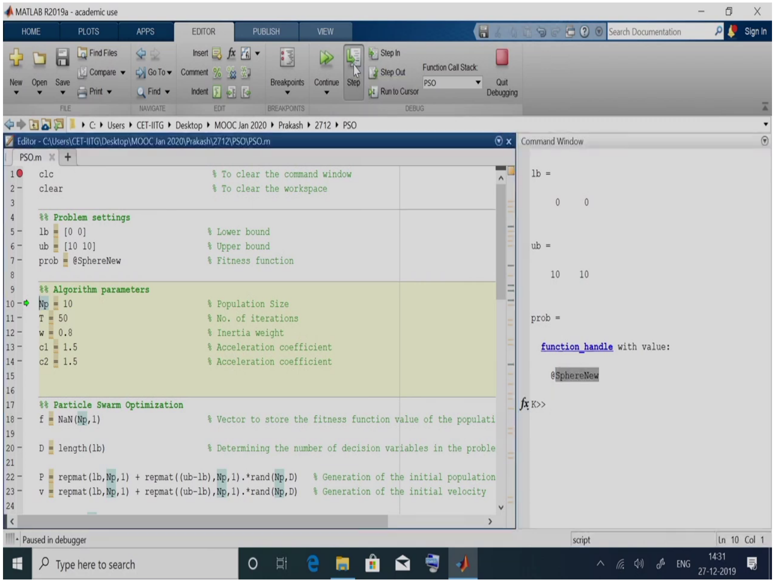
So, if that condition is valid, then again similar to updating p best; here we update the g best right, the function value as well as the solution value. So, here if you see in line 56 and 57, we do not have f underscore the g best of p and we do not have g best of p comma colon right; because f of g best and g best are only the global best, right. So, there is nothing for each individual particle, as the name itself suggests it is the global best; whereas, p best and f of p best is assigned for each particle. So, that is why over here we have this p and here p comma colon, here we do not have any index, right.

So, since the p best pth solution is to be assigned, we use the index p over here, right. So, once we check for these conditions; once we update the p best and the g best right, so that completes the implementation of particle swarm optimization. As you can see, it is a very simple algorithm right, it can be quickly implemented; though we have implemented it on MATLAB, you can implement it in any other programming language. Since these lines are in a for loop, they are going to be executed p times; and this for loop itself is going to be executed for t times because of this iteration loop, right.

So, at the end of iteration when all iterations are completed; the best fitness obtained by this algorithm right is nothing, but what is stored in f of g best, right. F of g best is the global at any given point of iteration, right. So, at the end of the iteration, f of g best is assigned to best fitness and best sol is g best, right. So, these two lines are actually not required, we have written it, so that it is consistent with TLBO; wherein if you remember we had found the fitness function value right, min of fitness function and then we had accessed that particular solution. Here we do not need to find out the minimum of f right; because if it had been better, it would have been updated in this g best and f of g best, right.

So, that completes the implementation of particle swarm optimization. So, I have removed all the semicolons. So, the execution of every line is going to be printed on the command window. So, let us go into the debug mode, let me put a breakpoint over here and let us execute this program, right.

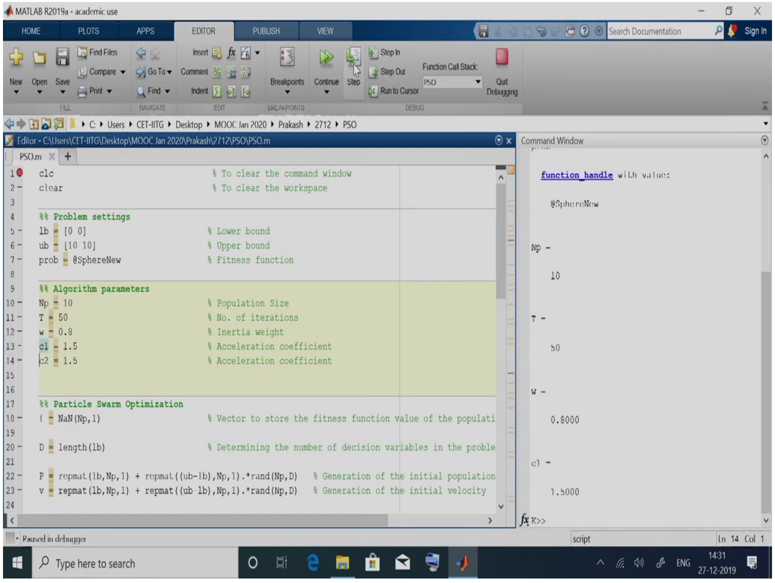
(Refer Slide Time: 15:26)



So, clc as you know will clear the command window right, clear will clear the workspace MATLAB workspace, right. We are defining the lower bound to be 0, 0; the upper bound to be 10, 10 right and the problem is a function handle. So, here if we see it says function handle with a value at the rate sphere new, right.

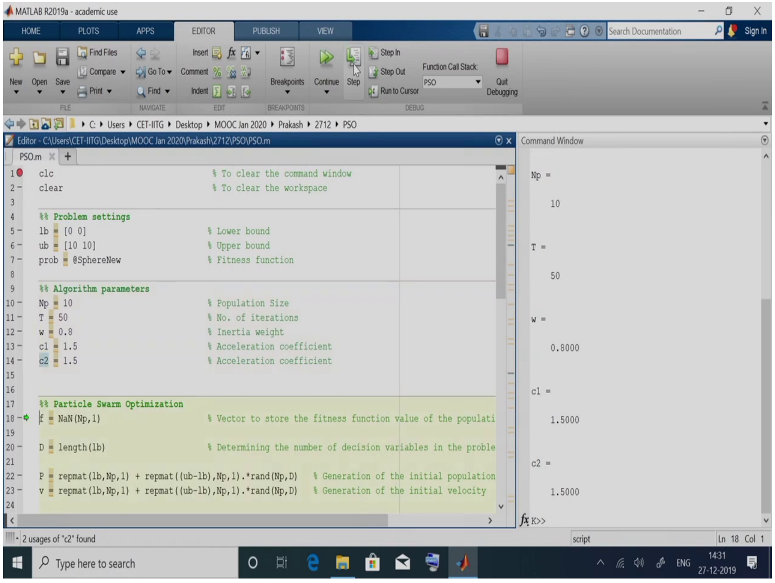
So, sphere new is actually a function name, right. So, whenever we are accessing prob, we will be actually accessing sphere new right.

(Refer Slide Time: 15:59)



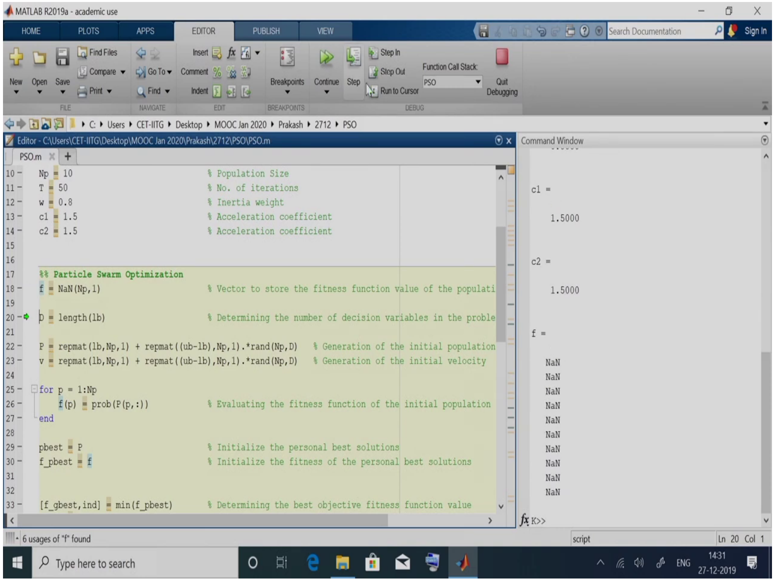
And then we are defining the population size, the number of iterations, the inertia weight, the acceleration coefficient c 1, the acceleration coefficient c 2, right.

(Refer Slide Time: 16:08)



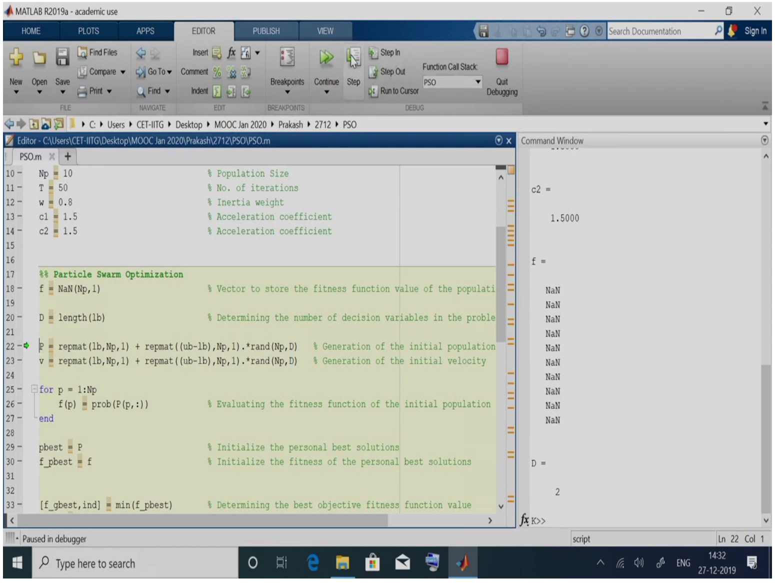
And then we are initializing the vector f, right.

(Refer Slide Time: 16:13)



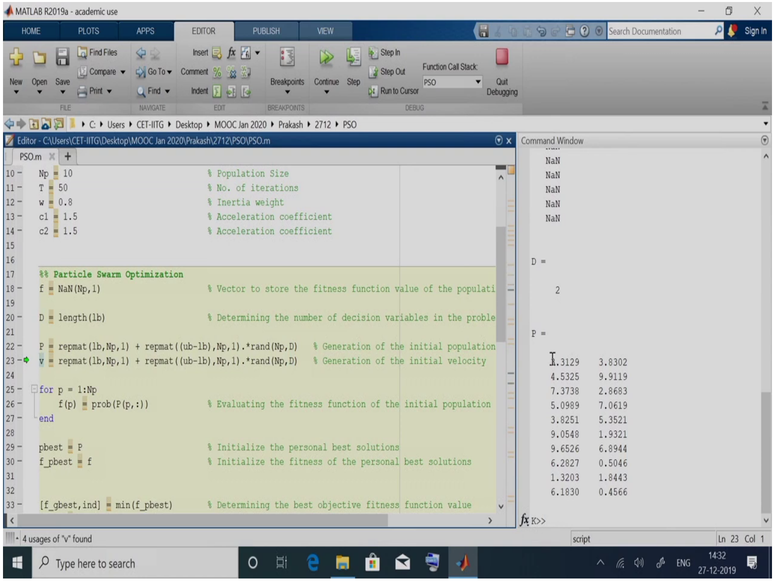
So, f if you see right now it is N a N 10 times; because we have 10 particles right. And then the next line is to determine the length of the decision variables, right.

(Refer Slide Time: 16:25)



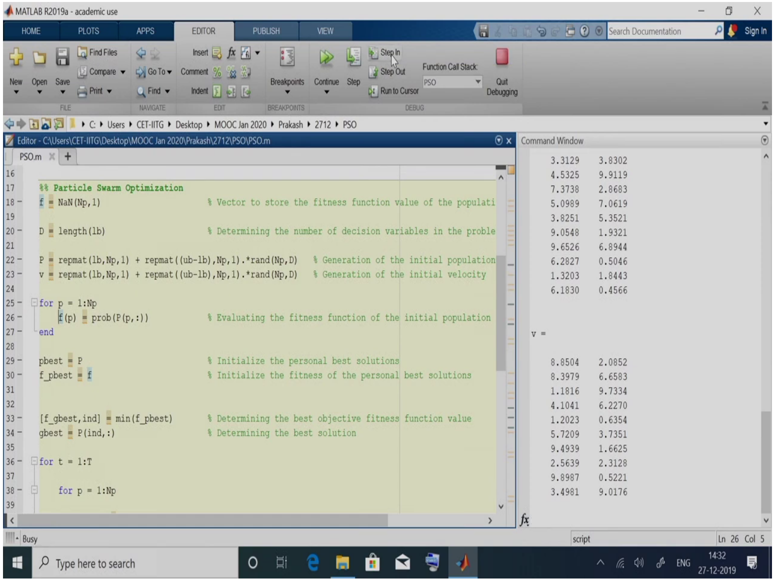
So, in this case we have to, so D capital D is 2, right. Now if we execute this line.

(Refer Slide Time: 16:32)



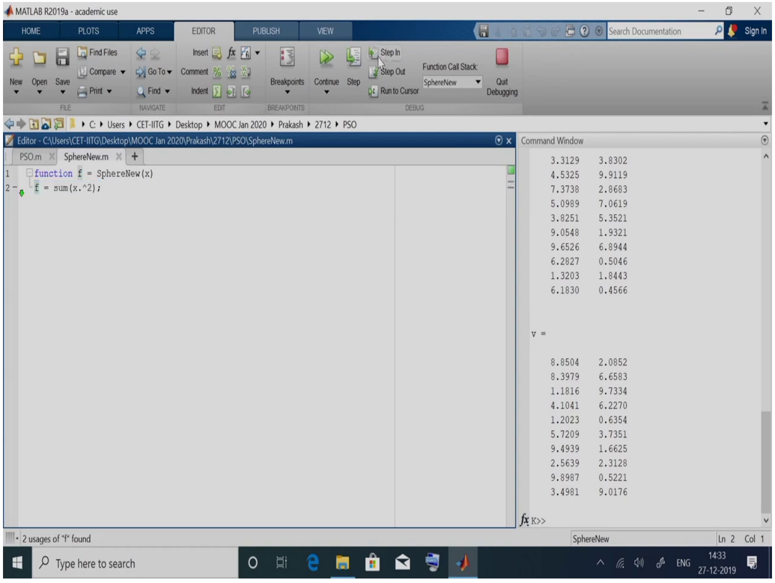
Line 22 has helped us to generate 10 population members; all of them are between 0 and 10, right. So, the working of line 22, we have discussed in detail during the implementation of TLBO.

(Refer Slide Time: 16:46)



So, similarly velocity is being randomly generated between the upper and lower bounds, right. So, the next step is to calculate the fitness function value. So, we are passing one member after the other, right.

(Refer Slide Time: 16:58)



So, if I do step in, it actually comes to the sphere function right, x right now is 3.312, 3.8302.

So, we are passing the population not the velocity, right. So, that is something that you will have to carefully remember that, the fitness function of the positions are to be determined and not of the velocity. So, this will calculate 3.3129 square plus 3.8302 square, because of this line, right. So, if we give step in and then again step in. So, if we move to the next line. So, the first objective function value or the fitness function value in this case is 25.6457, right.

So, this we can keep executing 10 times. So, every time if we, if you see in the command window; one value is getting populated, right. So, for all the ten members, this objective function value is being determined. Once we have determined the objective function value, the next is to assign p best and f of p best, right. So, this line will nearly create p best right, variable p best with values same as population. So, p what we had created was; here if we see

3.3129, 3.8302. So, the same value is assigned to this variable p best, right. So, once we assign that, similarly we need to assign the fitness function value corresponding to the each p best solution, right.

So, if we step that, so this is nothing, but nearly another assignment; wherein we have taken the value of f and assigned it to f of p best, right. So, the next step is to identify the minimum over here. So, the minimum over here if we see it is 5.1448, place where it is located at the 9th position right. Because our population size is 10, so we can see that it is located at the 9th position, right. So, if we step over here. So, as expected right the f of g best is 5.1448 and it is located at the 9th position, right. So, now, we need to extract the member the solution, the value of the decision variable corresponding to the 9th position right, and assign it to g best.

So, 9th position if we see, the variables are 1.3203 and 1.8443. So, line 34 will help us to do that, the decision variable corresponding to the best objective function value is taken right and it is stored in g best. So, now, we are in the iteration loop. So, for t is equal to 1, right. So, let us say step, right. So, here if we see t is 1 as of right now right and p will also be 1, ok. So, and the next step is to calculate the velocity.

So, since I have removed the semicolon at the end of this line, it has updated the velocity of one right; the rest of the nine values are same, because it will change when p is equal to 2. Right now we have changed the velocity of only the first members. So, initially the velocity was, if you see the initial velocity of the first member was 8.8504 and 2.0852; that has now changed to 4.5934 and minus 0.5773, right. So, the rest the other nine numbers would be the same.

So, if we scroll up 8.397, 6.6583. So, all of them would be the same, because only the velocity of the first particle has been updated. So, when this loop is getting executed in p times, all of these velocities would get updated accordingly. So, now, that we have determined the velocity, the next step is to update the position, right. So, if we do step, right. So, the position would have changed. So, initially the position of the first particle was 3.3129 and 3.8302, right. So, right now it has been updated, it has been updated using its previous position right and this velocity.

So, now our new particle is 7.90363 and 3.2529; as expected it has been directly plugged into the population right, there is no greedy selection involved over here. So, it is directly plugged into this, right. So, remember we did not bound the velocity. Once the iteration starts, the velocity need not be within the lower and upper bound right, it is the position which has to be in the upper and lower bound. So, in this case the value that we have obtained is within the bound 7.9063 and 3.2529 is between 0 and 10, right

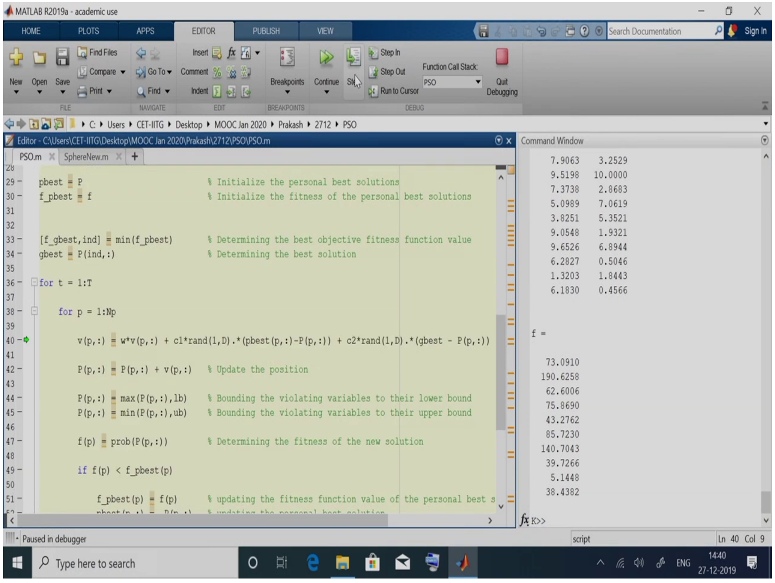
So, this line 44 and 45 we do not expect that to do anything, because the variables are anyway within the bounds. So, here if we see, this was printed in the command window when we generated p; this is after bounding it for the lower bound and this is after bounding it for the upper bound, only the first solution is being bounded. So, the next step is to evaluate the fitness, right. So, if we do step. So, it evaluated the fitness and now it has obtained the solution 73.09, right. So, previously the first solution had a fitness of 25.6457, right.

So, now the solution which we have generated has a inferior fitness, right. So, we do not expect it to go inside this if condition right, let us just see, right. So, it did not go into the if condition; because our current f of p, so with p equal to 1, we have 73.09. For this particle the best value previously was 25.6457. Since this condition is not met, none of these conditions are to be executed and that loop gets completed right. Completed in the sense for p equal to 1 and then when we do it this is p equal to 2, right. So, for p is equal to 2, the velocity is updated right, the position is updated.

So, right now if you see for p is equal to 2, the second decision variable is actually violating the bounds. So, this line 44 will not make any change; because it is not violating the lower bound right, the lower bound is zero. But line 45 will bring back this 14.1633 to 10, right. So, if we do step. So, as expected this has been brought back to the upper bound right; because it was violating the upper bound, right. Again we need to determine the fitness function value. So, the fitness function value in this case happens to be 190.6258.

Here if we see for the second member 190 and p best previously we had was 118.79, right. So, again this condition is not satisfied, so step.

(Refer Slide Time: 23:51)



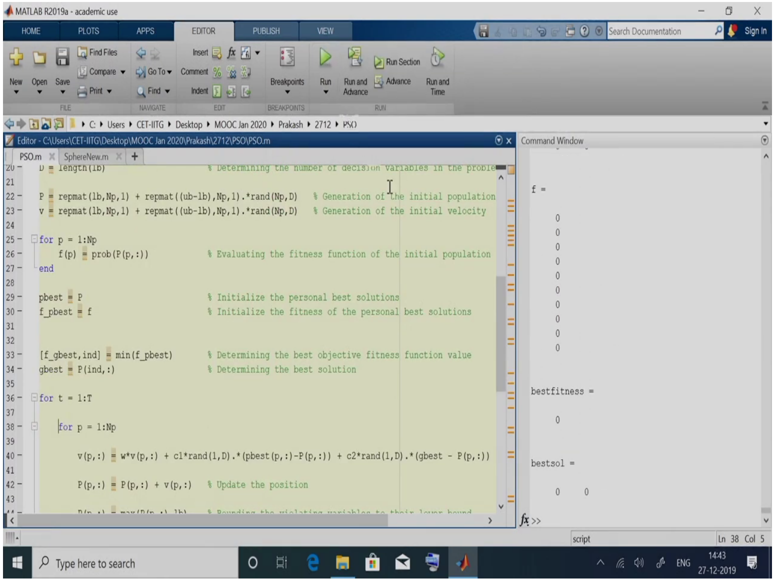
Now it is performing for p is equal to 3, right. So, p is equal to 3, again let us not we will not discuss in detail it is same thing right; let us say for p is equal to 4, it still did not go inside that loop, right. So, for p is equal to 5, right. So, f of 5th member is 37.4157 and the previous best for that was 43, right. So, this was 37 the fifth value.

You can see first value is 73, second value is 190, third value is 110, fourth value is 132 and the fifth value is 37.4157. So, that 37 has to be compared with this fifth value. So, fifth value in this case is 43. So, this condition is satisfied, right. So, since this condition is satisfied, it is going to update the p best for the fifth particle, right. So, if we step in. So, the fifth particle if you see that has been updated, this is the personal best right and same thing the fitness function as well as the solution would have got updated.

So, this condition would still not be satisfied, because the best of the p'th particle is fifth particle is 37; whereas the global best is 5.1448. So, this condition is we expected to fail, right. So, it did not update the f g best and the g best, right. So, this we can keep continuing, right. So, this is for the sixth member, right. So, again for sixth member it was not able to update the g best right; for the seventh member it was able to obtain a better p best right, but not better than the g best so, eighth member, ok. So, for eighth member it did not even update the personal best, right.

So, ninth member also it did not update the personal best; because it did not find a better solution, right. So, that completes one iteration. So, now, t is equal to 2. So, this loop is going to be executed 50 times. So, let me just click on this continue, so that it will complete the entire procedure, right.

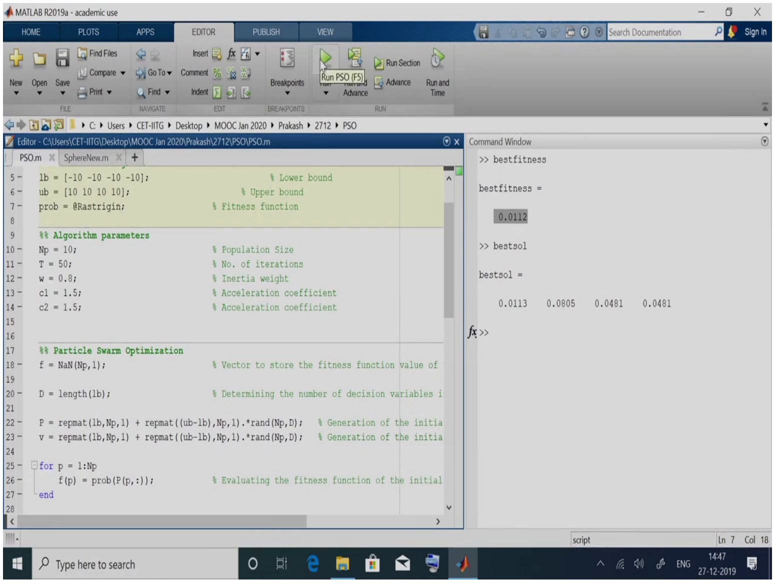
(Refer Slide Time: 26:10)



At the end of the entire procedure, since we have not given a semicolon in at the end of line 65 and 66; it is displaying the best fitness and the best solution. So, at the end of all the iterations for all the members, the best solution that we get is 0 comma 0 for the sphere function; and the best fitness function value corresponding to this 0 comma 0 is 0, right.

So, let me just quickly put the semicolon back, right. So, now, let us just tweak the lower bounds; like instead of 0, 0 let us say if the lower bounds is minus 10, minus 10 right and let us say it is a four variable problem, right. So, let us say the bounds are minus 10 and 10 for all the four decision variables.

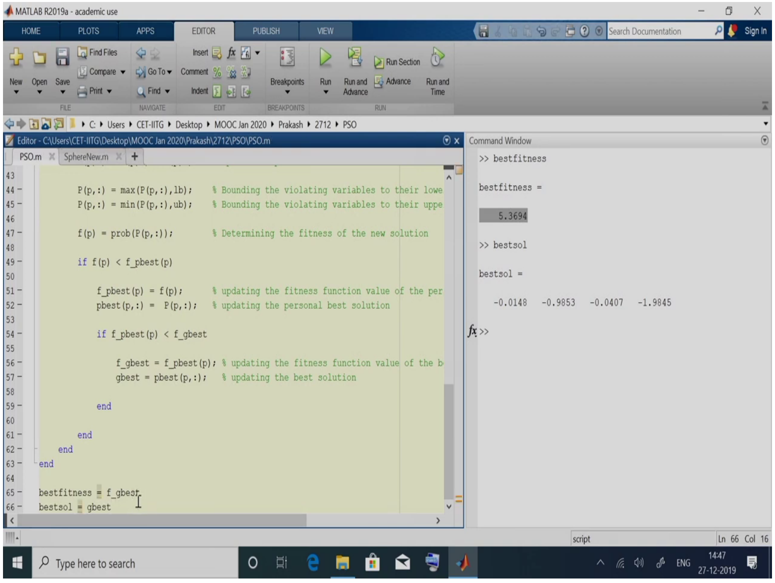
(Refer Slide Time: 27:11)



So, now, if we execute this, and if we the best solutions are best fitness, right. So, right now we do not get a 0, right; and best sol is this thing, right. So, for the sphere function we a priory knew the optimal solution right; the optimal solution was x 1 is equal to 0, x 2 is equal to 0, x 3 is equal to 0, x 4 is equal to 0.

When we are solving a four variable problem PSO is with these settings right, it is not able to determine the globally optimal solution; the best that it has determined is 0.0112, right. So, similarly we can run this for other objective functions also. So, let me see Rastrigin. So, let us see what happens with Rastrigin function.

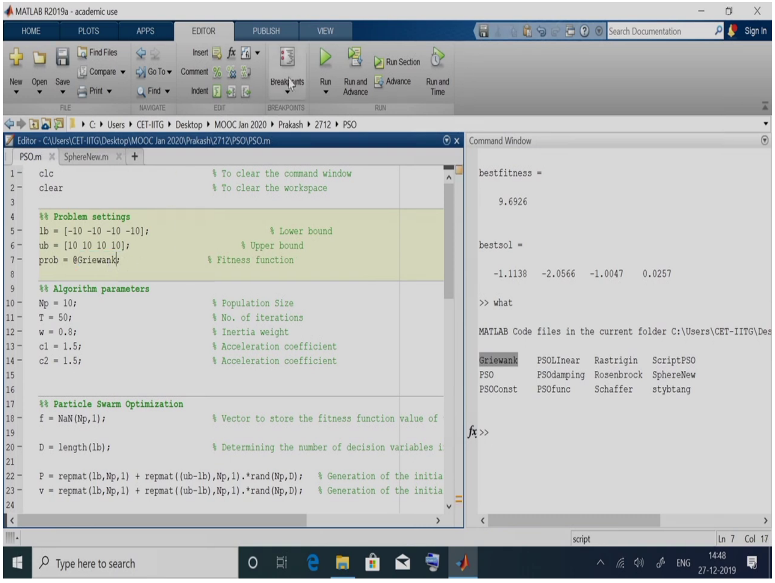
(Refer Slide Time: 28:09)



So, here again best fitness, best sol; So, again the best fitness for this problem is 0, right; but it is not able to obtain the optimal solution with these settings, right. So, obviously, these settings have their own impact on the performance of algorithm.

So, let me just remove this semicolon, so that I do not need to type every time to see the solution.

(Refer Slide Time: 28:40)



So, now if we see every time we run right, we get a different solution; that is because of the stochasticity of the algorithm, that is why as we saw in TLBO the algorithm has to be run multiple times, right. So, the objective function sometimes is as bad as 18.84. So, every time we run, we get a different this thing; we can test it for other functions also. So, let me see what other functions are there. So, this Griewank function is there. So, for Griewank function let us see whether it is. So, for Griewank function it seems to be performing reasonably well right; because the global optimal solution is located at zero, right. It is able to consistently get closer to zero; unlike in Rastingin function where it was significantly away from zero; here it is more or less close to zero.

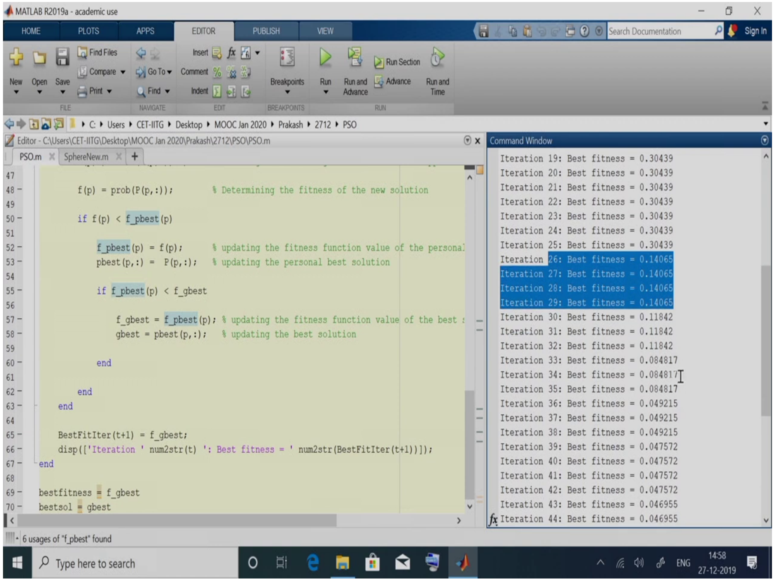
Since we have wrote the code in a generic fashion right; wherein we have kept the problem details away from the actual algorithm, it is easy to test with different problems; different problem, different lower and upper bounds, right. So, now, what we will do is, we will try to plot the convergence curve similar to TLBO. So, let me define best Fit Iter right; I am going to store N a N in it right, the number of columns is going to be 1, right and the number of rows is going to be T plus 1. Because I will also be storing what is happening in the initial the best value in the initial population, right.

So, this thing, so let me just take this best Fit Iter right and best Fit Iter for the first time right; remember MATLAB indexing starts with 1, not with 0, right. So, let me just put this. So, let me just write best Fit Iter over here. So, the first value is nothing, but f underscore g best; and then at the end of every iteration, best Fit Iter at the end of every iteration. So, t plus 1, because the first where we have already used the first index for saving the best in the initial population, right. So, t plus 1 is equal to f underscore g best, right.

So, here I have chosen to store the fitness function of the best solution known so far, right. So, this will ensure that we get a monotonic convergence curve, right. So, because the population as we know, does not employ a greedy selection strategy, right. So, we are not just plotting the minimum of f right; the best solution that has been obtained so far is actually located in f underscore g best. So, that is why I have chosen to store f underscore g best right. In addition to storing the best fitness obtained in every iteration right; let us also display it using this statement, right.

So, this statement we had previously seen in TLBO also, right. We want to display the best fitness value in every iteration, right. So, this num 2 str will help us to convert this variability into a string, right. And we also want to display the word iteration, so it is within single codes. And then we have num 2 string of t and then we have this which within this single codes which will be displayed as it is. And then again we are using the num 2 string function to display the best fitness function value in every iteration, right. So, this is inside this iteration loop.

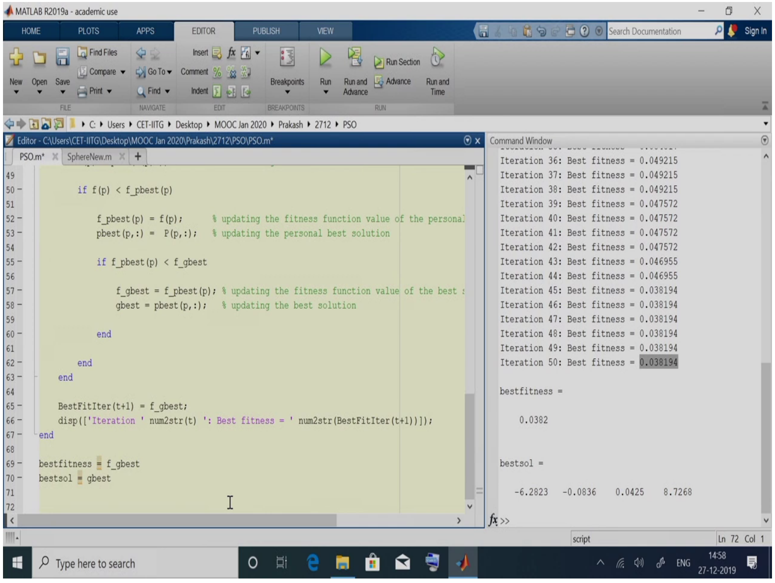
(Refer Slide Time: 32:45)



So, now, if we execute this; this displays the best fitness with respect to every iteration. So, we can see that the value 0.39194 is constant for three iterations; that means that for three iteration, there was no update in g best, right.

So, we cannot comment whether there was an update in p best or not with this information right. And then we obtained one solution which was 0.30439, right. So, that is why the g best would have been updated and that stays on all the way till iteration 25; and then we subsequently obtain the good solution 0.14065 which state for four iterations and we can do similar analysis, right.

(Refer Slide Time: 33:27)

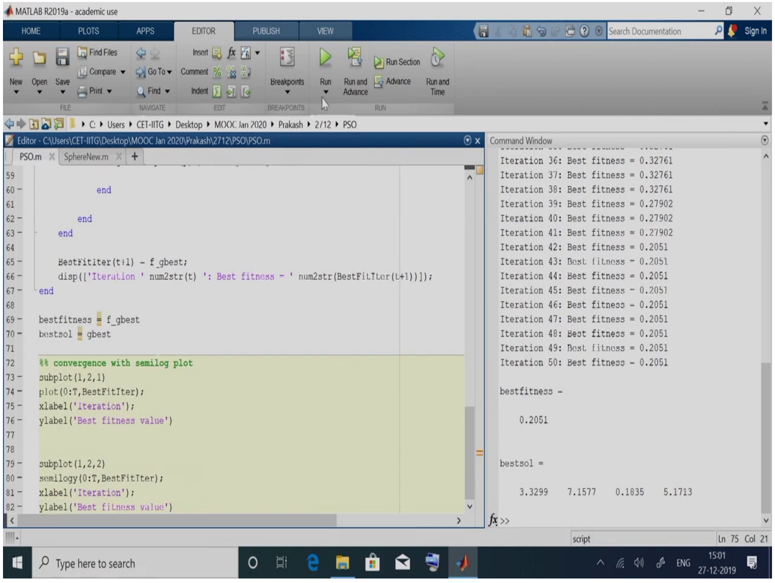


So, at the end of 50 iteration, the value that we obtained is 0.038194, right; so if interested we can also plot this value.

So, similar to TLBO, we can have this small piece of code, right. So, here we are dividing the plot window into two, right. It will have 1 row, 2 columns; in the first position we are plotting in the x axis the iteration right from 0 to T and on the y axis we are taking the best fitness function value obtained in every iteration and then we are adding the x label and the y label right. In the second position, so the second plot. So, that will be the second plot; we are plotting the same thing right, the x values and y values are the same, just that the y axis is now semi log, right.

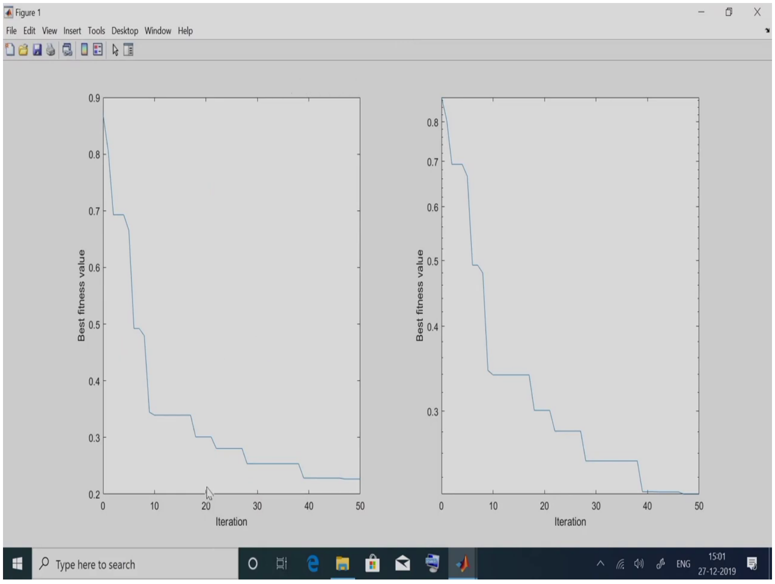
And again we add the x label and y label.

(Refer Slide Time: 34:24)



So, let us look into the plots, right.

(Refer Slide Time: 34:26)



So, we get these two plots. So, as we can see this is the first position, wherein the y axis is not in the logs case; whereas in the second plot, the y axis is in the log scale. So, here we can see that, the objective function value or the fitness function value has been continuously decreasing, right and it is a monotonic convergence because we are plotting the f of g best, right. So, this helps in visualization of the results as to what is happening as the iteration progresses so.

Thank you.