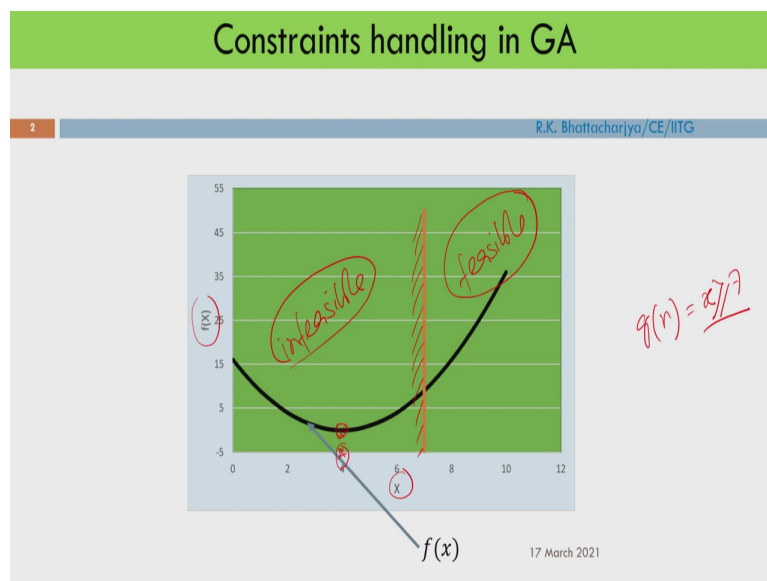


Optimization Methods for Civil Engineering
Dr. Rajib Kumar Bhattacharjya
Department of Civil Engineering
Indian Institute of Technology, Guwahati

Lecture - 25
Constraints Handling in GA

Welcome back to the course on Optimization Methods for Civil Engineering. So, in this class, we will discuss Constraint Handling in GA. So, already we have discussed how we can handle constraint using penalty parameter approach, in case of classical optimization technique. But in case of GA, so we may use little bit different constraint handling technique. So, that we will discuss today.

(Refer Slide Time: 00:58)



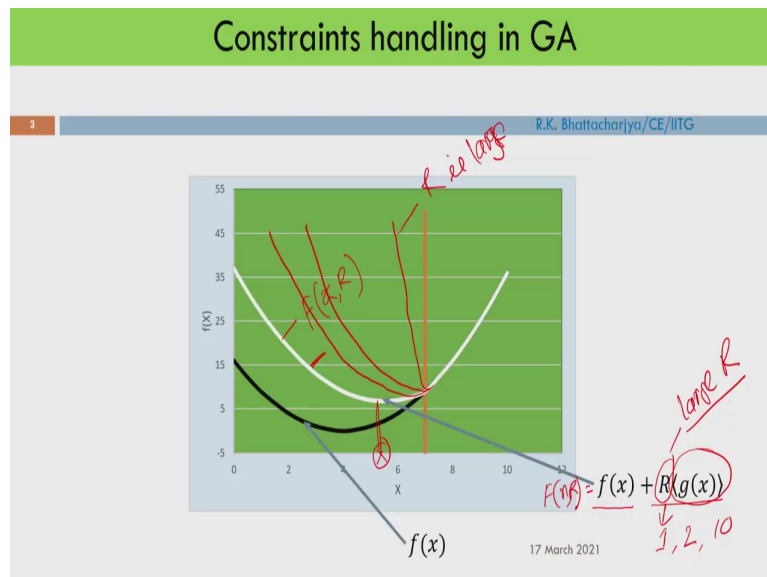
Suppose we have taken a function $f(x)$ and this is the function I have plotted here. So, this is the function $f(x)$. And if you look at this particular function, and this is a single variable function and we have x in the x axis and in the y axis I have plotted $f(x)$. So, if you look at this

particular function, that minimize somewhere here and probably the x^* is somewhere here. So, this is the optimal; or you can say this is the stationary point of this particular function.

Now, we have a constraint, and this is the constraint, ok. So, as per this constraint, so any solution on the on this side of, so it is infeasible. So, if I say the infeasible, so any solution in this part will be infeasible and this is the feasible part, ok. So that means, the solution should be greater than 7, ok.

So, suppose the $g(x)$ which is equal to x greater than equal to 7, ok. So therefore, any solution if it is less than 7, then it is an infeasible solution, and on the right-hand side of this particular line x equal to 7, so this solution is your feasible solution. So, therefore, in this particular case, the earlier optimal solution which is somewhere here and this solution is now an infeasible solution.

(Refer Slide Time: 02:43)



So, we have already discussed the penalty parameter approach. So, what I can do? I can apply the penalty parameter approach. So, in this case, what we have done? That I am writing a penalty function that is your capital $F(x)$, ok. So, capital $F(x)$ and it is a function of R , which is small $f(x)$ plus R and we have use suppose bracket operator here. So, we have used bracket operator here. Now, if I write, so a particular value of R , this is the capital $F(x)$. So, penalty your function, ok. So, this is the capital $F(x)$.

Now, if you look at this particular function that for smaller value of R , suppose if I take R equal to 1 or R equal to 2 R equal to 10, something like that; in that case, what will happen? So, this is the function I am getting. So, in that case, also whatever solution you are getting. So, this is the solution of this $F(x)$ function. So, that solution is also in the infeasible region.

So therefore, what you have to do in this case? So, I have to use a large value of R , ok large R I have to use. So, suppose if I increase the R value, then my function will be something like this, ok something like this. So, I have shown; and for large value of R your function will be something like that. So, this is for R is large, ok R is large.

So, now in case of classical method. So, what we have done? So, we have started our iteration with smaller value of R and then we are increasing the value of R . So, initially we cannot consider large value of R because the gradient based method will not work. So therefore, we have started our iteration with smaller value of R maybe 1; 1, 2 within 1 to 10, and then we are increasing the value of R gradually.

And finally, we are getting the constraint optimal solution of this particular problem for larger value of R , ok. So, this is the exterior penalty method. So, in this case, what we have done? We have penalized the solution in the infeasible region, ok. So, we are getting the constraint optimal solution for large value of R .

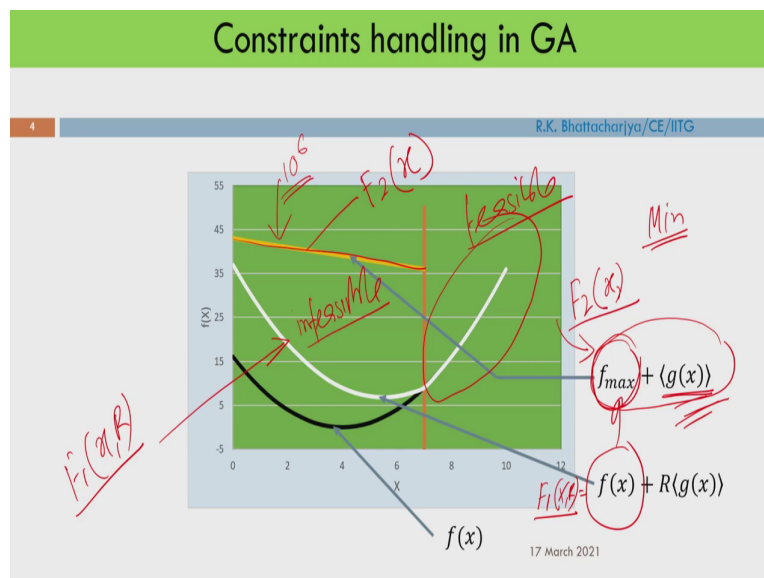
Now, I can say that this is one of the limitation of this particular approach, because you have to select a suitable value of R initially and then you have to increase this increase the value of R , and you have to do multiple iteration in between. Now, question is that how to select the

value of R . So, that again you have to in on trial and error basis, so you are selecting initial R , then you are increasing, and finally, once you are getting the solution that will be the constraint optimal solution of the problem.

So therefore, I can say that this is one of the limitation of this particular method. In case of classical method, so I do not have other options. So, I have to start with smaller value of R and then I am increasing; and again whatever I am telling this is in case of exterior penalty methods, ok. So, anyway.

So, in case of interior penalty method, so we are starting with a larger value of R , and then we are decreasing the value of R . So therefore, I can say that this is one of the limitation of this particular approach. But in case of GA, as we are not using the gradient information, so our penalty parameter function may be different. So, I can use actually a different penalty parameter function, ok.

(Refer Slide Time: 06:24)



So, let us see what I can do in case of GA. So, in case of GA, what I can do, that I can write this function something like this. So, if I write that f makes plus $g \cdot x$. So, I am using the bracket operator here and I am actually in place of $f \cdot x$, so I am using f_{\max} . So, what is f_{\max} ?

That is the maximum function value of the solution in a feasible region, ok. So, that is f_{\max} is coming from this particular suppose region. So, what is the maximum function value? That I am adding with the violation. So, this $g \cdot x$ is will give you the violation. So, I am adding the maximum function value.

Again, please note that I am discussing a minimization problem. So, this problem is a minimization problem. So, in case of minimization problem, so what I am doing? I am adding f_{\max} with the violation, ok. So, this is the function now. This is the; if I write that this is F_1

$x \in R$. So, this is the penalty parameter approach, and this is your there is no R now, this is your suppose I am writing $f^2(x)$.

So, in this case, there is no penalty parameter. In case of classical method, so we have used penalty parameter, but here there is no penalty parameter and we are just using $f_{\max} + g(x)$. So, I am getting this function. So, I can say that this is my $f^2(x)$, ok and this particular function is F_1 , this is $x \in R$, ok. So, this is your we have penalty parameter in this case, but in this case, we do not have the penalty parameter.

And again, in this case also, what we have done? We have penalized the solution in the infeasible region, ok. So, this is your infeasible region infeasible, and this is your feasible region. So, we have penalized the solution in the infeasible region. How we have penalized? So, we have actually added the violation with f_{\max} . So, I am getting this particular function, ok.

Suppose, you are applying the tournament selection. So, what will happen? That if any solution, any infeasible solution is playing tournament with a feasible solution. So, in that case, feasible solution will be selected because their fitness value will be more, ok. And then any two solution, any two infeasible solution if they are going for tournament, the solution with lesser constraint violation will be selected, is not it.

So, therefore, so this is an a very efficient method. There is no penalty parameter. So, you need not define any parameter here. And this is actually selecting. If there is a competition between two infeasible solution, the solution having lesser constraint violation will be selected, and if there is a competition between two feasible solution the solution with minimum function value will be selected, ok.

So therefore, this is a very efficient method and we are not actually using any penalty parameter. So, you need not define any penalty parameter. Only you have to define the constraint violation that actually you can calculate. Suppose in place of these, so in the infeasible region, so I can define the function value, suppose a very large value I can define

suppose 10^6 or a very large value I can define, but I am defining this value for all the solution in a infeasible region.

So, therefore, in that case also GA will work, but the problem will be if there is a competition between two infeasible solution, both the solution will be equal now. So, you have to select one of them. But in this method the solution having lesser constraint value will be selected.

Suppose some solution which is near the your constraint boundary, so and one solution is away from the constraint boundary, then solution near the constraint boundary will be selected in this case.

(Refer Slide Time: 10:44)

Constraints handling in GA

5
R.K. Bhattacharjya/CE/IITG

Minimize $f(X)$

Subject to

$g_j(x) \leq 0 \quad j = 1, 2, 3, \dots, J$

$h_k(x) = 0 \quad k = 1, 2, 3, \dots, K$

Deb's approach

$$F = f(X)$$

$$= f_{max} + \sum_{j=1}^J \langle g_j(x) \rangle + \sum_{k=1}^K |h_k(x)|$$

If X is feasible

Otherwise

17 March 2021

So therefore, finally, what I can define? So, this approach was given by Professor Kalyanmoy Deb, ok. So, we can say this is a Deb approach. And suppose this is the problem, it is a

minimization problem again. As I said that we are discussing minimization problem. So we have two constraint; one is inequality type constraint, one is equality type constraint that is $g \leq 0$ and $h(x) = 0$.

And for that, so I can write the penalty function, I can write the constraint handling function something like that that $F = f(X)$, if X is feasible. So, if it is in the feasible, so in that case, you are not adding any penalty. But if it is in the infeasible region in the otherwise, so what you are doing? You are calculating f_{max} .

So, what is the f_{max} ? So, f_{max} is the maximum function value of the feasible solution. So, you have a set of feasible solution and f_{max} is the maximum function value of that particular set and then you are adding the violation. So, this is the violation of less than equality type constraint, and this is the violation of equality type constraints. And this way I can implement constraint in case of genetic algorithm.

Now, let us discuss a different algorithm, and we call it evolutionary strategy. So, this is similar to genetic algorithm, but exactly not genetic algorithm or you can say this is the real coded genetic algorithm with only mutation. So, there is no crossover, ok. So, let us see what is this algorithm.

(Refer Slide Time: 12:26)

The slide is titled "Evolutionary Strategies" in a green header. Below the header, there is a blue bar with the text "R.K. Bhattachariya/CE/IITG" on the right and a small orange square with the number "2" on the left. The main content area is light gray and contains a bulleted list of three points. At the bottom right of the slide, the date "17 March 2021" is displayed.

Evolutionary Strategies

- ✓ ES use real parameter value
- ✓ ES does not use crossover operator
- ✓ It is just like a real coded genetic algorithms with selection and mutation operators only

17 March 2021

So, evolution strategies or we can say a ES, so ES use real parameter value. So, we are not converting to binary bit here. So, binary conversion is not required. So, we are directly using the real parameter value. So, ES does not use crossover operators. There is no crossover operator in case of ES, and we are just using the mutation operator.

So, it is just like a real coded genetic algorithm with selection and mutation operators only. So therefore, this is a very simple algorithm, but very powerful in solving non-linear, non-convex problem, but as I said it is a; it is a very simple algorithm.

(Refer Slide Time: 13:13)

The slide is titled "Evolutionary Strategies" in a green header. Below the header, there is a blue bar with the text "R.K. Bhattacharjya/CE/IITG" on the right and a small orange square with the number "3" on the left. The main content area is light gray and contains the text "Two members ES: (1+1) ES" in bold. Below this, it says "In each iteration one parent is used to create one offspring by using Gaussian mutation operator". At the bottom right, the date "17 March 2021" is displayed.

Now, let us discuss two member ES, so we call it 1 plus 1 ES. So, here only you have two members and this is one of the simplest algorithm optimization algorithm, so which can be implemented using only few lines maybe within 5 to 10 lines you can implement this particular algorithm. So, let us see.

So, what you, what we do here? In each iteration; one parent is used to create one offspring by using Gaussian mutation operator. So, here it is an 1 plus 1; that means, one parent I am using, and from one parent I am actually creating one offspring using Gaussian mutation operator, ok. So, let us see.

(Refer Slide Time: 13:56)

Evolutionary Strategies

4R.K. Bhattacharjya/CE/IITG

Two members ES: (1+1) ES

- ✓ Step 1: Choose a initial solution x and a mutation strength σ
- ✓ Step 2: Create a mutate solution
$$y = x + N(0, \sigma)$$

$\text{Min } f(n)$
- ✓ Step 3: If $f(y) < f(x)$, replace x with y
- ✓ Step 4: If termination criteria is satisfied, stop, else go to step 2

17 March 2021

So, it has only four steps. The step one, you have to choose an initial solution x and a mutation strength σ . So, in this case, what we are doing? We are starting with a initial solution. Just like your classical method, so you have to give a initial; just like the classical method you have to give an initial point and let us see that initial solution is x and you have to define a mutation strength σ , ok. So, I have to define this particular value. So, I have to define an initial point, initial solution and also I have to define the mutation strength σ .

Then, in step 2, what will do basically that create a muted solution, ok. So, how we are creating a; creating this solution? That y equal to x plus, this is the random number I am creating with mean 0 and your and σ . σ is already defined. So, I can create a random number with mean 0 and σ equal to σ .

Now, you are creating an offspring. Now, what you are doing? You are calculating the function value that is f of y and if f of y is less than f of x ; that means, in case of minimization problem. So, in case of minimization problem, minimization of f x I am doing. So, in that case if f of y , so whatever new solution you are creating f of y is less than f of x , then what you are doing? You are getting a better solution.

So, in that case, you are replacing x with y . So, whatever x value, so you have actually, you will you are replacing x with y . And then, I will continue this iteration if termination criteria is satisfied stop, otherwise you go to step 2; that means, step 2 means you create another new solution, ok.

So, this is a very simple algorithm. So I, as I said that I can implement this particular algorithm with few lines of code. I will also show you R code, how to implement this particular algorithm.

(Refer Slide Time: 16:13)

Evolutionary Strategies

5 R.K. Bhattacharjya/CE/IITG

Two members ES: (1+1) ES

- ✓ Strength of the algorithm is the proper value of σ
- ✓ Rechenberg postulate
 - ✓ The ratio of successful mutations to all the mutations should be $1/5$. If this ratio is greater than $1/5$, increase mutation strength. If it is less than $1/5$, decrease the mutation strength.

17 March 2021

So, in this case, the strength of the algorithm is the proper value of sigma. So, we have to use a proper value of sigma. What will happen? Suppose, if I am using a smaller value of sigma, then it may take lot of time to reach the optimal solution, if you are away from the optimal solution. Suppose, you are far away from the optimal solution and if you are using a smaller value of sigma, then it may take lot of iteration to reach the optimal solution.

On the other hand, if your sigma is very large, so in that case, you may not be able to converse at the optimal solution. So, therefore, what you have to do? If you are away from the optima then you have to use large sigma value, so that very quickly you are reaching the optimal solution. But once you are reaching the optima then you should reduce your sigma value. In that case, you are getting the optimal solution, otherwise you may not get the optimal solution. So, therefore, the strength of the algorithm is the proper value of sigma.

Rechenberg, what he said? That the ratio of successful mutation to all mutations should be 1 by 5, ok. So, it should be one-fifth. Ratio of successful mutation to all mutation should be 1 by 5, ok. So, and if the ratio, if the ratio is greater than 1 by 5; that means, you are more successful. If the ratio is greater than 1 by 5; that means, you are more successful. So, in that case, you are going towards the optimal solution and you increase your mutation strength.

And if it is less than 1 by 5, that means, you are not successful; that means, you are not getting an optimal solution. If the ratio is less than 1 by 5 decrease the mutation strength. So, what does it mean? That if the ratio between successful mutation to all mutation is less than 1 by 5; that means, you are not successful. You are, you in that case, what will happen? That you are not actually you are near the optimal solution, but actually you are looking at the other region

So therefore, in that case what you have to do you have to reduce your sigma value. So, that you can actually search nearby. So, this is the; if you can use Rechenberg postulate. So, in that case, the ratio between successful mutation to all mutation is greater than 5, then increase the mutation strength and if it is less than 1 by 5 decrease the mutation strength.

(Refer Slide Time: 18:55)

Evolutionary Strategies

6R.K. Bhattacharjya/CE/IITG

Two members ES: (1+1) ES

- ✓ A mutation is defined as successful if the mutated offspring is better than the parent solution.
- ✓ If P_s is the ratio of successful mutation over n trial, Schwefel (1981) suggested a factor $C_d = 0.817$ in the following σ update rule

$$\sigma^{t+1} = \begin{cases} C_d \sigma^t & \text{if } P_s < \frac{1}{5} \\ \frac{1}{C_d} \sigma^t & \text{if } P_s > \frac{1}{5} \\ \sigma^t & \text{if } P_s = \frac{1}{5} \end{cases}$$

17 March 2021

A mutation is defined as successful if the mutated offspring is better than the parent solution. So, what is successful mutation? If the mutation is creating a better solution than parents, then we are saying that that is a successful mutation. But if it is not creating a better individual better solution, in that case, what we will say? That is an unsuccessful mutation.

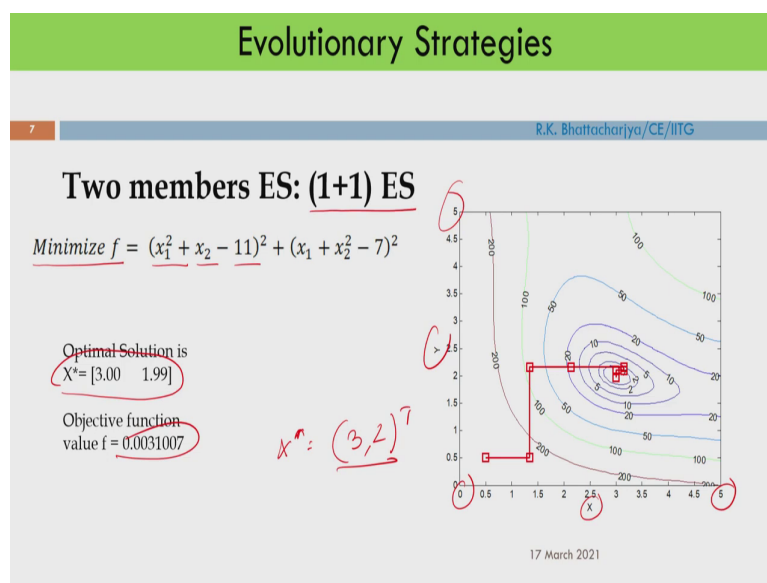
Then Schwefel he define if P_s is the ratio of successful mutation over n trial, then Schwefel suggested a factor C_d which is 0.817 in the following sigma update rule. So, he suggested this one, that means, how you are updating sigma in the next generation this that if the P_s , P_s is the ratio between successful mutation to all mutation. If it is less than 1 by 5, then you are using C_d into sigma t .

That means, what you are doing in that case, that is you are not successful, the ratio is less than 1 by 5, so you are in you are decreasing the mutation, ok. If it is more than 1, 1 by 5; that

means, you have you are more successful. So, in that case you are increasing the sigma. So, that is why you are dividing sigma t by C d, ok. And if the ratio is 1 by 5, in that case, you are not doing anything, you are not changing the sigma value

So, what we can do? So, we can use this sigma update rule and can sense the sigma value with the iteration, ok.

(Refer Slide Time: 20:46)



So, this is the problem I would like to show you. So, I think you have seen this particular function. So, this is a minimization problem that x_1 square plus x_2 minus 11 whole square plus x_1 plus x_2 square minus 7 whole square. And we I have plotted this function between 0 and 5; that means, x_1 is 0 and 5. So, I have written x and y .

So, x is between 0 and 5 and y is also between 0 and 5; that means, it is in the first quadrant and first quadrant solution is 3 and 2, so this is the solution somewhere here. So, x star; x star is 3 and 2, ok. So, this is the solution. And I have applied 1 plus 1 ES, and I got this solution that is 3 and 1.19. So, this is the solution of this particular problem, anyway. Function value is something like that.

So, I will also show you the R code for solving this particular problem.

(Refer Slide Time: 21:45)

Evolutionary Strategies

8
R.K. Bhattacharjya/CE/IITG

Multimember ES

$(\mu + \lambda)$ ES

Step 1: Choose an initial population of μ solutions and mutation strength σ

Step 2: Create λ mutated solution $y^i = x^i + N(0, \sigma)$

Step 3: Combine μ and λ , and choose the best solutions μ

Step 4: Terminate; Else go to step 2

17 March 2021

Now, let us discuss the multimember ES, ok. So, we call it μ plus λ ES. In the earlier one, so we have 1 plus 1; that means, from one parent I am creating another offspring. But in this case, what we are doing? We are calling; we call it μ plus λ ES. So, let us see the steps of this particular algorithm. So, multimember ES.

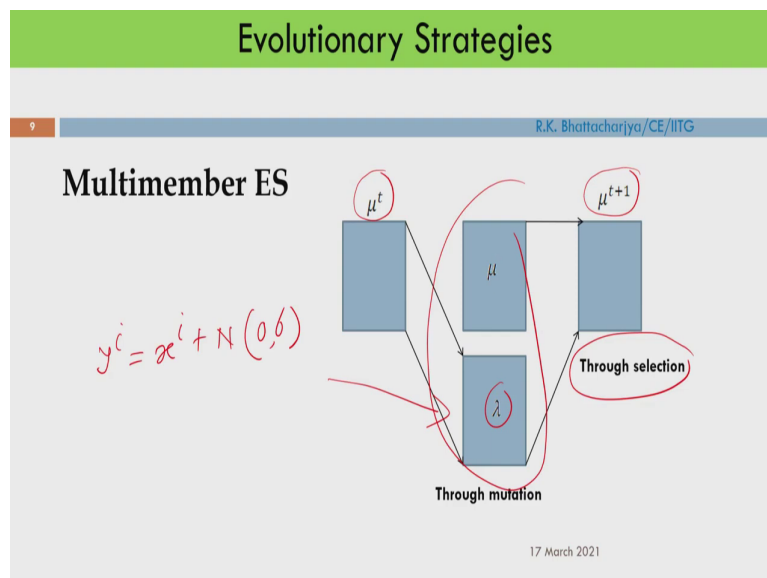
The step 1, shows an initial population of μ solutions and mutation strength of σ . So, what we are doing? We are creating a population, μ population, so that means, there are μ numbers of solution in that population. So, here this is not 1 to 1, but it is many to many. So, we have created μ solutions. And I can say that this is the initial population, ok or initial solution.

Now, from this initial μ solution create λ mutated solution using this equation; that means, y_i equal to x_i plus $N(0, \sigma)$. So, anyway σ we have already defined and mean 0. So, we have used Gaussian distribution here and with mean 0, so this is μ equal to 0, and σ equal to σ , so I am creating a new population, ok. So, λ population from the μ population.

Now, in step 3, what we are doing? Combine μ and λ and choose the best μ solutions, ok. So, what I am doing? I am combining μ and λ . So, in that way, I am also implementing the elitism here. So, if there is any better solution and which was there in the μ actually, so in the whole population, so that will also be preserved. So, I am using that; we are combining μ plus λ and then we are selecting best μ solution.

And step 4, terminate, else go to step 2, ok. So, you create another λ solution and that iteration will continue. So, we have only 4 line algorithm that is 4 step algorithm. And we will continue this iteration until and unless we are not reaching the termination criteria.

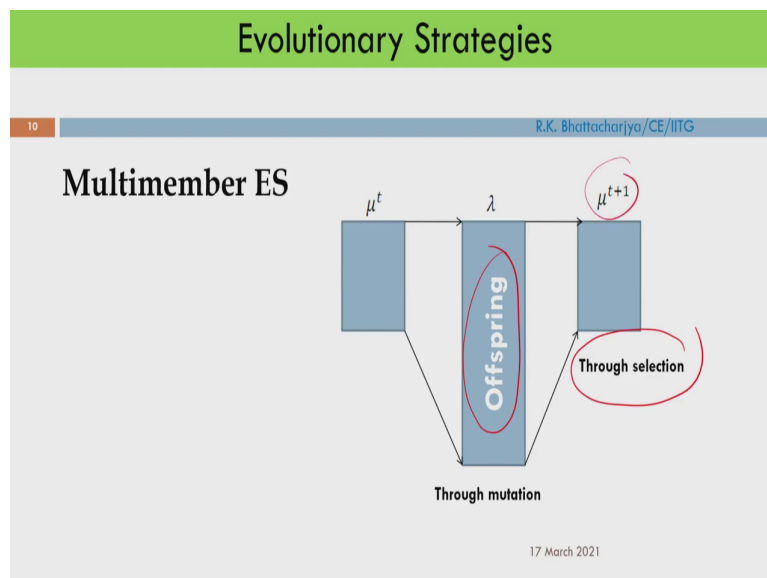
(Refer Slide Time: 24:14)



So, let us see this. So, what we are doing here. So, in this case, suppose this is the μ solution and you can say that whole population, and from that μ solution I am creating λ . So, how I am creating λ ? So, this is $y^i = x^i + N(0, \sigma)$. So, I am creating this λ population, and that I am combining this thing and from this combined solution I am creating again μ solution, I am getting the solution in the next iteration.

And how I am doing that one? Through selection. So, you can apply selection operator. Suppose, we have discussed several selection operator, that tournament selection, rolled wheel selection, proportionate selection. So, you can apply this selection operators and out of μ plus λ , so you can select μ solution, ok.

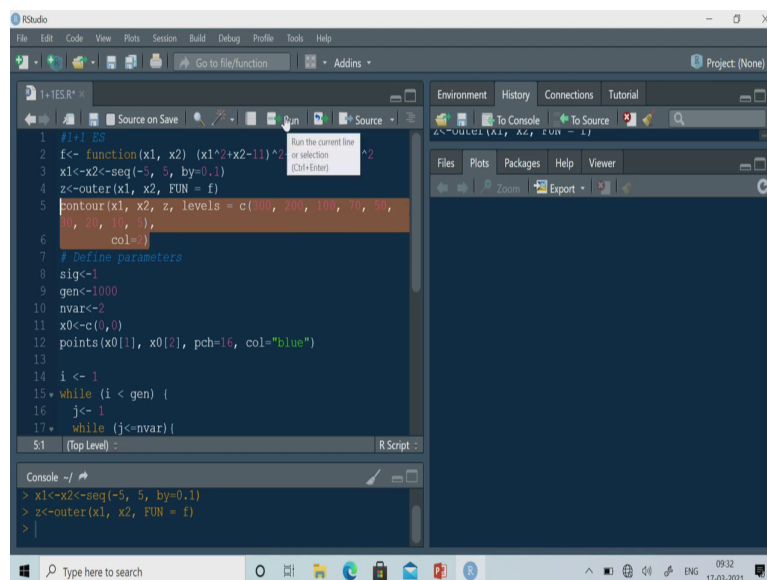
(Refer Slide Time: 25:09)



So, you are combining this thing and from here you are creating, your through selection you are creating mu solution. So, this is also a very simple algorithm. As I said, so probably you can implement this hardly it will have 15 to 20 lines code, and you can implement this particular algorithm using any programming language. You can use MATLAB, you can use C programming, you can use R, you can use python, any platform you can use, and hardly it will be 15 to 20 lines code.

Now, I will show you the R code for implementing 1 plus 1 ES. I have implemented 1 plus 1 ES in using R programming. So, let us see the R code for implementing 1 plus 1 ES.

(Refer Slide Time: 26:03)



The screenshot shows the RStudio interface. The main editor window contains an R script with the following code:

```
1 #1+1 ES
2 f<- function(x1, x2) (x1^2+x2-11)^2
3 x1<-x2<-seq(-5, 5, by=0.1)
4 z<-outer(x1, x2, FUN = f)
5 contour(x1, x2, z, levels = c(100, 200, 300, 40, 50,
6       60, 70, 80, 90),
7       col="blue")
8 # Define parameters
9 sig<-1
10 gen<-1000
11 nvar<-2
12 x0<-c(0,0)
13 points(x0[1], x0[2], pch=16, col="blue")
14
15 i <- 1
16 while (i < gen) {
17   j<- 1
18   while (j<-nvar){
```

The console window at the bottom shows the execution of the first three lines of the script:

```
> x1<-x2<-seq(-5, 5, by=0.1)
> z<-outer(x1, x2, FUN = f)
>
```

So, here first line is the function, so anyway. So, I am implementing 1 plus 1 ES. So, I have written this comment. And first line is the function, I am defining this function. What is this function that? It is a function of x_1 and x_2 , and this is $x_1^2 + x_2 - 11$ whole square then plus x_1 plus x_2 minus 7 whole square. So, this line if I execute, then I am executing this particular function, ok.

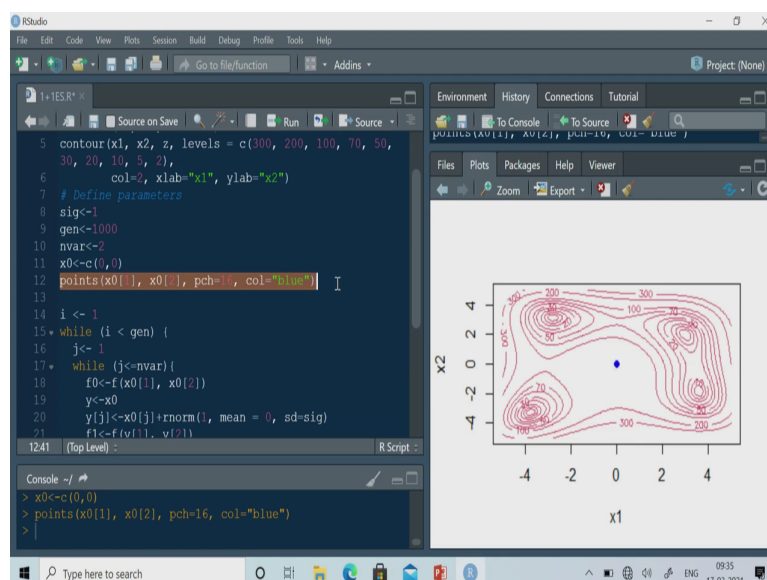
And then I am defining the value of x_1 and x_2 , and because I would like to plot this contour, so I am defining x_1 and x_2 . So, by using the sequence function, and this is between minus 5 and plus 5, so I have defined between minus 5 and plus 5. And then the interval between two points is 0.1.

So, let us execute this one. So, I can execute. So, I am getting the x_1 and x_2 . And then I am calculating the z value. What is z value? This is the function value at its grid point. So, I am

using outer function here and then I am defining, what is x_1 , then x_2 and then function equal to f , ok. So, I am calculating the z value using this particular line.

So, now I can plot the contour, you using the contour function. And what is contour function? That, it is the arguments are x_1 . So, I have to pass x_1 , then x_2 , then I have to put the z values and I am defining here the contour level. So, I would like to draw the contour line with 300 value, 200, 100, 70, 50, 30, 20, 10, 5. So, if you want to put any other, so you can put it; and colour I am putting 2. So, if I execute this particular your lines, then I should get the contour plot.

(Refer Slide Time: 28:04)



So, you can see that I am getting the contour plots here. So, I can also increase suppose if you need another contour level, suppose I would like to know this level. And I can put x lab and y lab, suppose x lab equal to x_1 here and y lab equal to x_2 , ok. So, let me execute this

particular lines again. So, in that case, now I am getting x level and y levels, that is x 1 and x 2, ok. So now, my contour, yeah, plot, I am getting the contour plot of this particular function.

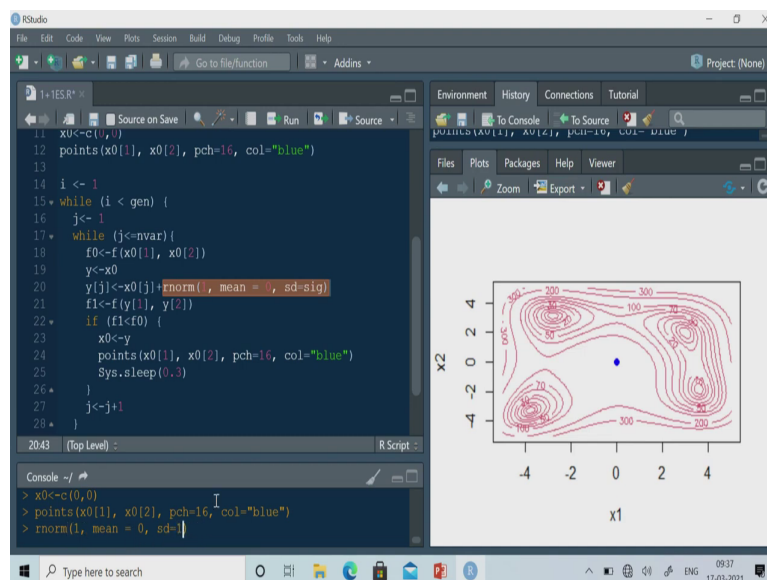
Now, let us define the parameters required for 1 plus 1 algorithm. So, I have to define a sigma value, so I am defining I am using 1 here. So, I am not changing the sigma value with the iteration, constant sigma value I am using. So, this is the sigma value. So, let us define sigma.

Then maximum generation that means, iteration I would like to do 1000. So, I am defining generation equal to 1000. And in my, this in this case I have two variables, so I am defining number of variable that is 2. And then, I am defining an initial solution, so I would like to start this iteration from 0, 0. So, you can also sense this thing, but I would like to start from 0, 0. So, therefore, x naught I am putting 0, 0. So, I am defining an initial solution.

So, I would like to plot this initial solution over this contour plot. So, in that case, I can use point function. So, this point you are actually putting the point x 0, 1 and x 0, 2. So, this is x, y you have to define, and then I am using the circle here, and colour I am using blue; pch 16 means it will give circle and blue means blue colour circle.

So, let us execute this particular then I should get 0, 0. So, you can see that this is the initial solution that is 0, 0, ok.

(Refer Slide Time: 30:27)



Now, let us write the 1 plus 1 ES. So, first iteration, so iteration equal to 1. So, I am using while function here. So, while i less than generation, then I would like to execute all these particular lines, ok. So, this iteration will go up to 1000 generation, ok.

So, within that and I have two variables. So, I have also used variable counter here; that means, I have started with j 1 and then I am also using while function here. That means, I would like to go from j 1 to j 2, while j is less than equal to number of variable. So, in that case, it will execute this particular your lines, ok.

So, therefore; so I am using this one and now within this what I am doing, that first I am calculating what is the function value of the initial solution; what is the function value of the initial solution. So, I am creating, so already I have defined this function. So, I can say that f naught, so f naught I am passing two variable in f. So, this is your x naught 1 and x naught 2.

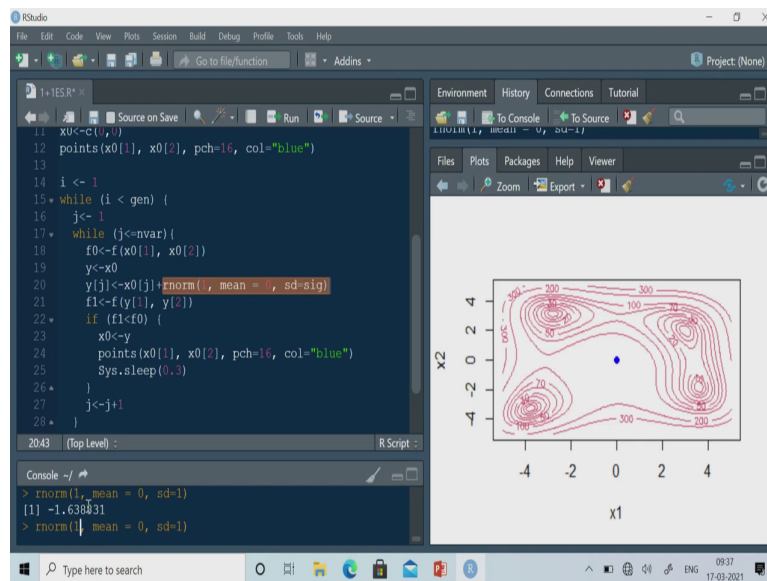
So, these are the two variables, you have to pass to this because I have written this function x_1 and x_2 .

So, I am putting x_1 and x_2 here. So, I am getting the function value at x_1 at x_{naught} function value x_{naught} . So, now you are defining a y variable. So, y equal to x_{naught} ; that means that is the next point I would like to create. So, next point is y . So, now, I have mutated this using the Gaussian distribution here.

So, what I am doing? So, I have mutated 1 by 1, that means, the first value that is x_1 , I would like to muted. So, I have mutated this is x_1 . Then, here in R, you can use R norm function to create a random number using Gaussian distribution that is mean 0 and sigma equal to whatever sigma I have defined. So, this is as the equal to sigma I have defined.

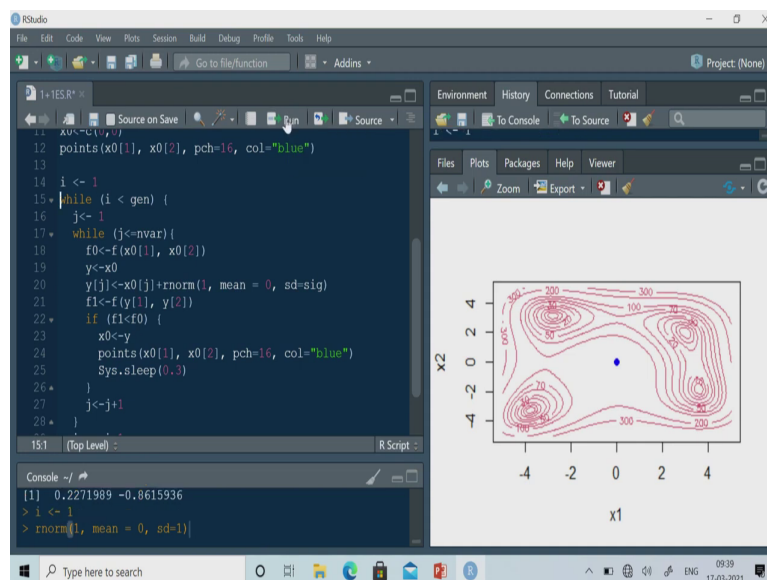
So, this part will create a random number. So, I can show you this one, suppose in this case I have used this is 1, ok sigma equal to 1.

(Refer Slide Time: 32:44)



So, you can see that it will create a random number, ok for with mean 0 and standard deviation 1. And this 1, I have written 1 here that means, it will create only 1 random number.

(Refer Slide Time: 33:00)



Suppose if I write it 2. So, it will create 2 random number, ok. But in this case, I would like to create only 1 random number, so therefore I have written 1, ok. So now, I am getting a new solution that is y. So now, I am calculating the function value at y; that is I am writing f 1 and f 1 equal to the f. Now, I am passing the y's, that is y 1 and y 2. So, this is the new function value.

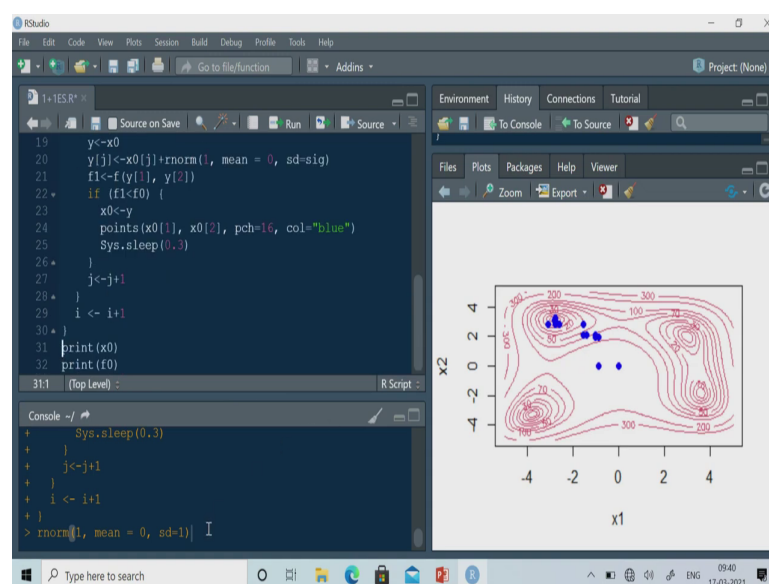
Now, what is the; what is next step? Next step is to compare if f of 1; that means, f 1 that means, the new solution is better than the older one, then what I will do? Then I will replace x naught by y, ok. So, this is the condition I have to use; that means, if f 1 is less than f naught, then I am replacing x naught by y, ok.

And then, I am just plotting that one; if it is successful then I am plotting that one. So, I am using the point function here again. So, I am plotting, I am going to the new your points. And just to see the iteration, so system will slip for 0.3, ok and then this iteration will continue.

So, once we are completing. So, this is the for the first variable it will be done, and for the second variable also it will be done, and then this iteration will continue unless and until you are not reaching the generation number; that is in this case this is 1000 generation. So, after that I can print what is the value of x and what is the function value optimal function value I can plot, ok.

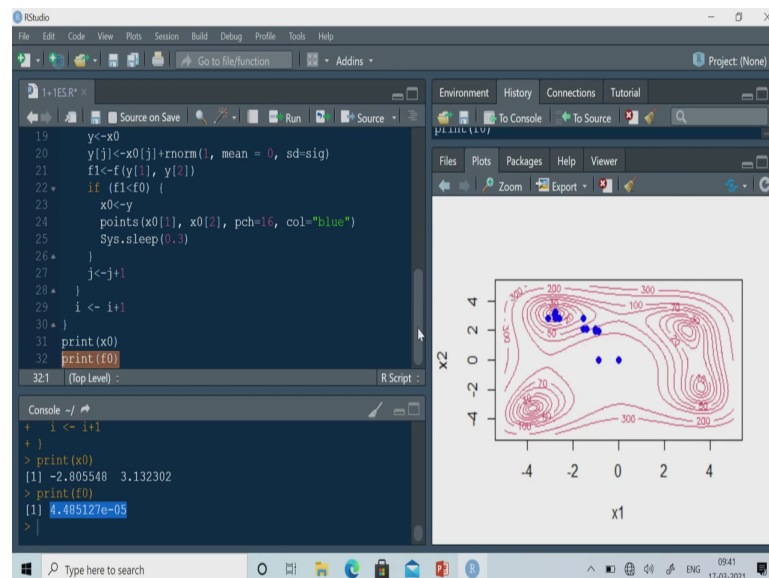
So, let us see. So, I have executed up to this thing. So now, I am defining x that i, I am defining i and then I am executing this iteration. So, let us see.

(Refer Slide Time: 34:54)



So, you just see that this is moving something like that and it has actually reached the solution at the second quadrant. So, you can see actually, if I execute this particular line, then I can see that one; just see.

(Refer Slide Time: 35:15)



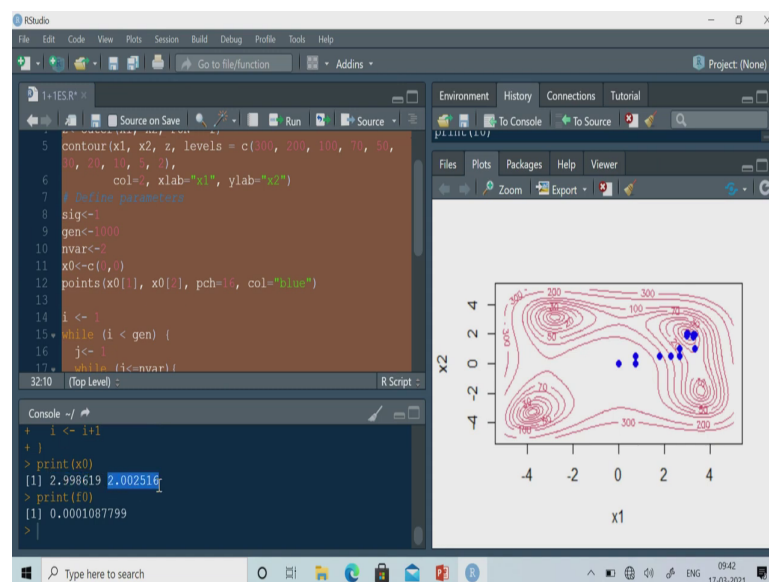
So, if I execute this particular line. So, what is the solution? The solution is minus 2.805548 and 3.132302. So, that is the solution I am getting. So, it is in the second quadrant. So, this is near the optimal solution. It may not be exact optimal solution, but it is near the optimal solution.

So therefore, what is the function value? It should be 0, but it will not be 0. But it is a very small numbers because this is just near the optimal solution. So, it is 4.48 into 10 to the power minus 5, ok. So, you just see with only few lines. So, the algorithm is from here only

from 15 to, so you can say from 14, line 14 to line 30; that means, it is only your 15 line code, ok.

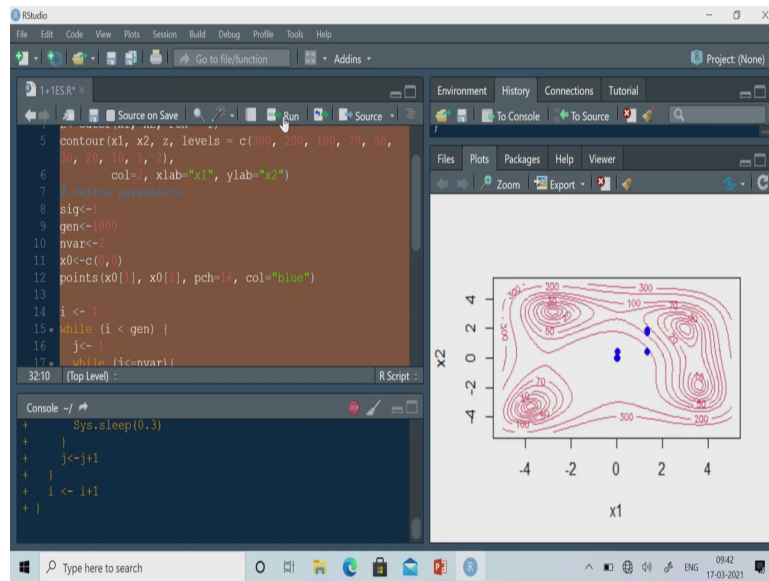
So, this is a very simple one. And as I said that you can implement it, you can implement this code with only 10 to 15 line your program, so and you are getting this optimal solution. So, I can also try, next time certainly I may get this solution, but I may also get the other solution. So, let us try I would like to execute this entire solution now. So, I am just clearing all those things.

(Refer Slide Time: 36:40)



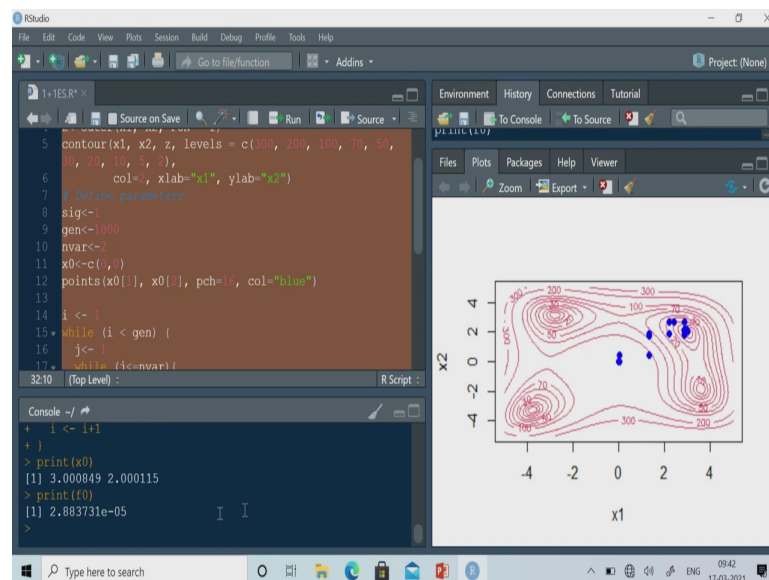
So, now I would like to execute all this line. Just see. So, this time I am getting the solution on the first quadrant, ok. So, first solution I am getting. What is the solution? That is 3 and 2. But I am getting 2.99 and 2, I am getting and function value is this, ok. So, I am getting the solution at the first quadrant.

(Refer Slide Time: 37:07)



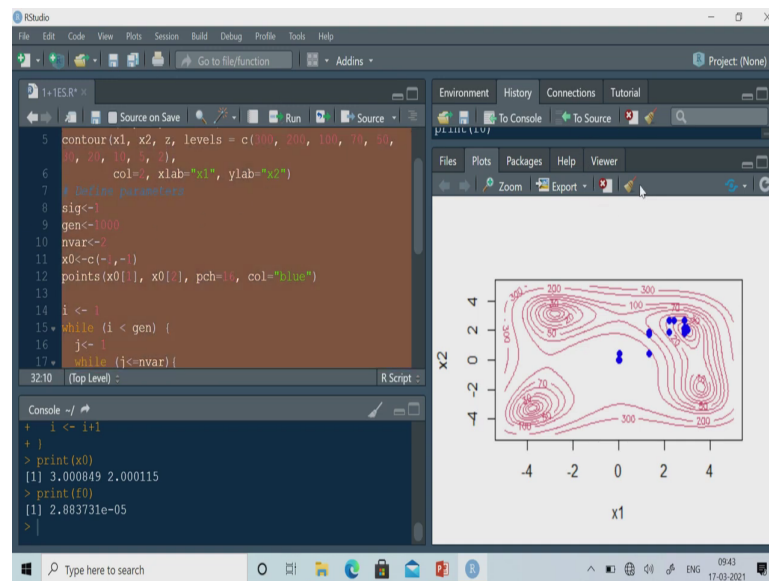
So, let me try again. So, now, let us execute it again.

(Refer Slide Time: 37:17)



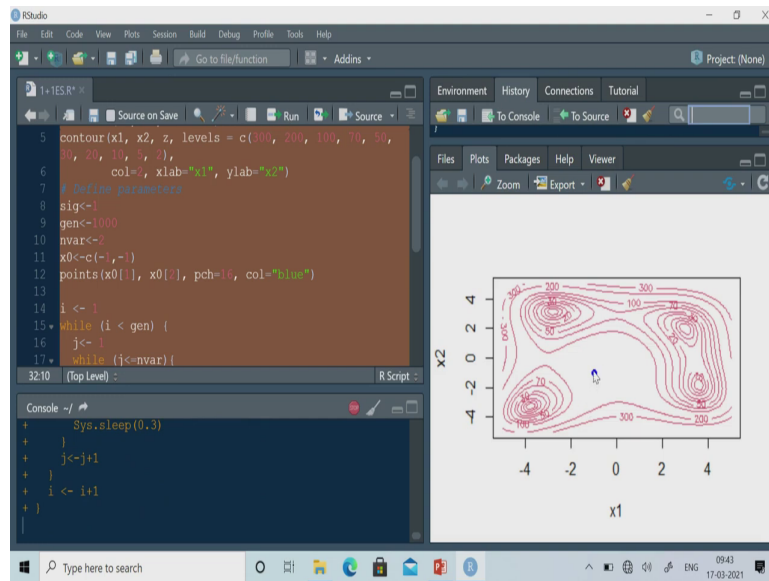
So, in this case, also I am getting the solution at the first quadrant. Just see. So, yeah it is moving, yeah, to the optimal solution at first quadrant. And what is the solution? That is 3 and 2. You just see I am getting the exact almost exact optimal solution that is 3 and 2, and the function value is 10 to the power 2 in 2.88 into 10 to the power minus 5.

(Refer Slide Time: 37:47)



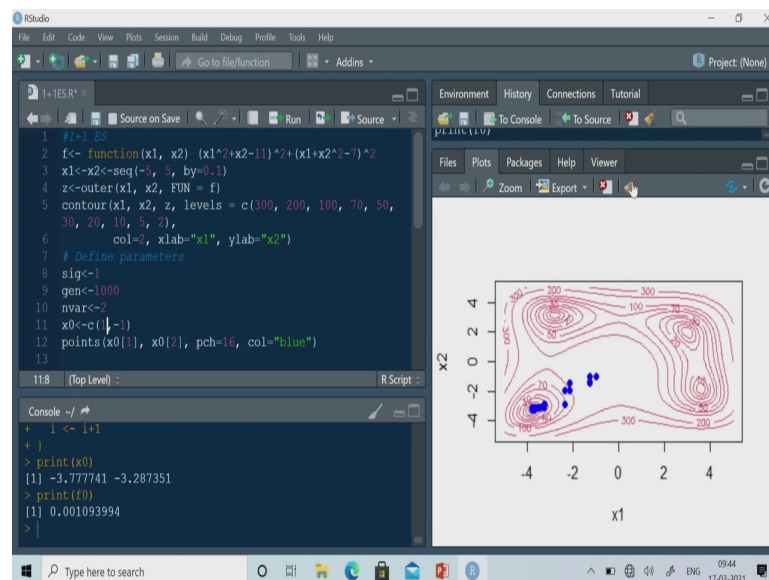
Now, if I sense my initial solution, suppose if I take initial solution, right now I have taken 0, 0, here, let us see. I have taken 0, 0, but if I take minus 1 and minus 1. Then, what will happen? Just see this 1 minus 1, and minus 1, and now if I execute this one. Before that I would like to clean, ok.

(Refer Slide Time: 38:02)



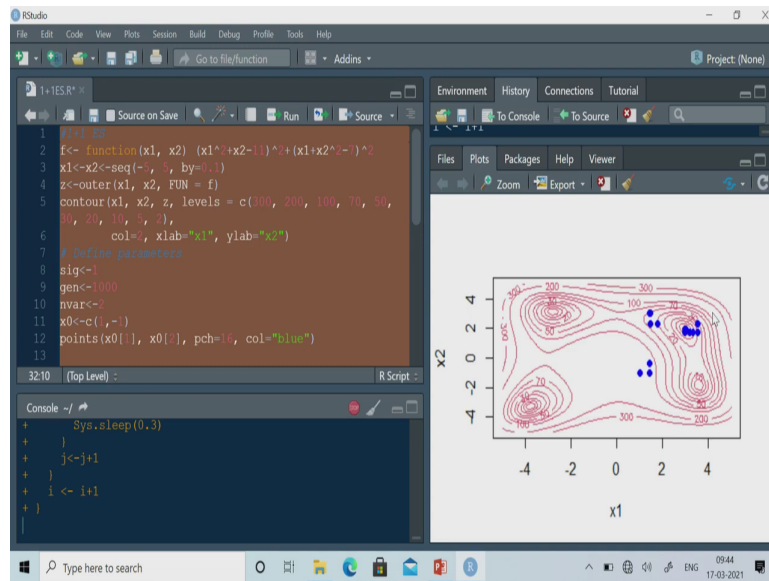
So, now execute this one. So, now; so, this is the initial solution minus 1 and minus 1.

(Refer Slide Time: 38:08)



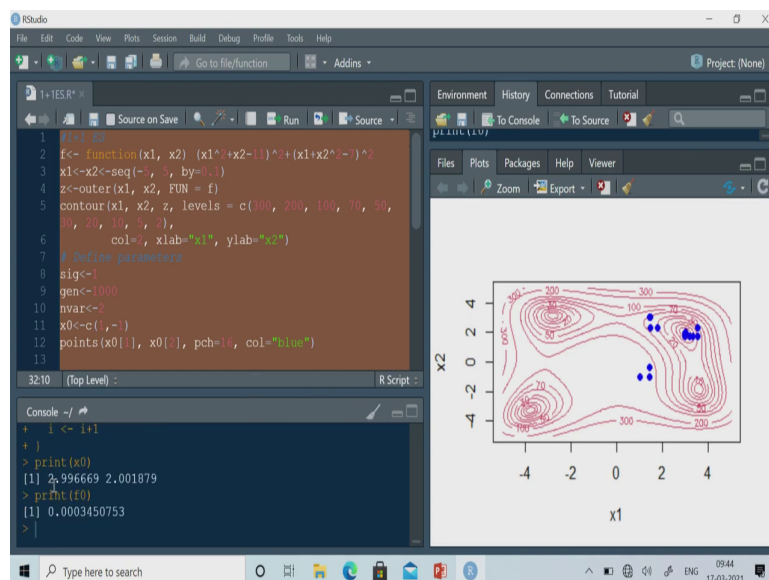
So, you are getting the solution at the third quadrant, ok. So, third quadrant and this is the solution 3.77 minus 3.77 and minus 3.28. So, this is the solution you are getting at the third quadrant. So, maybe I can change it to 1 minus 1, just see, ok. So, this is 1 and minus 1, so somewhere here. So, let us let us try with 1 minus 1.

(Refer Slide Time: 38:41)



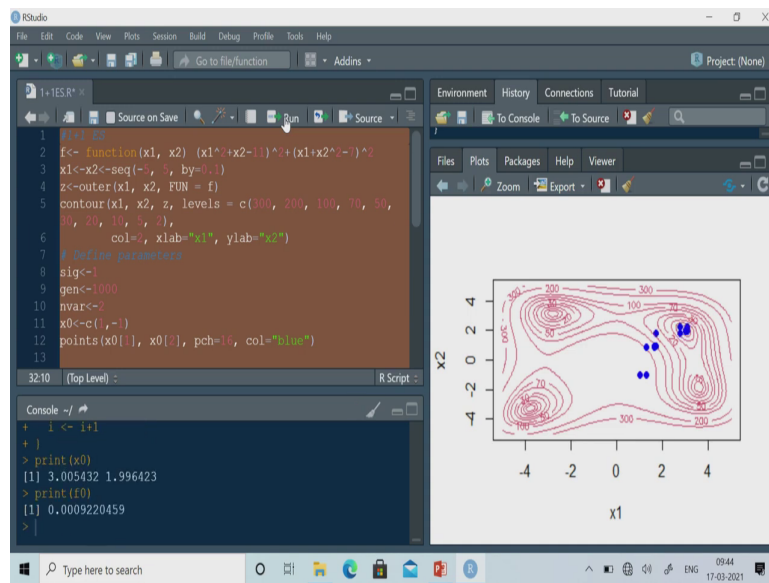
Now, this is the initial solution and probably I will get this the solution at the first quadrant, ok.

(Refer Slide Time: 38:53)

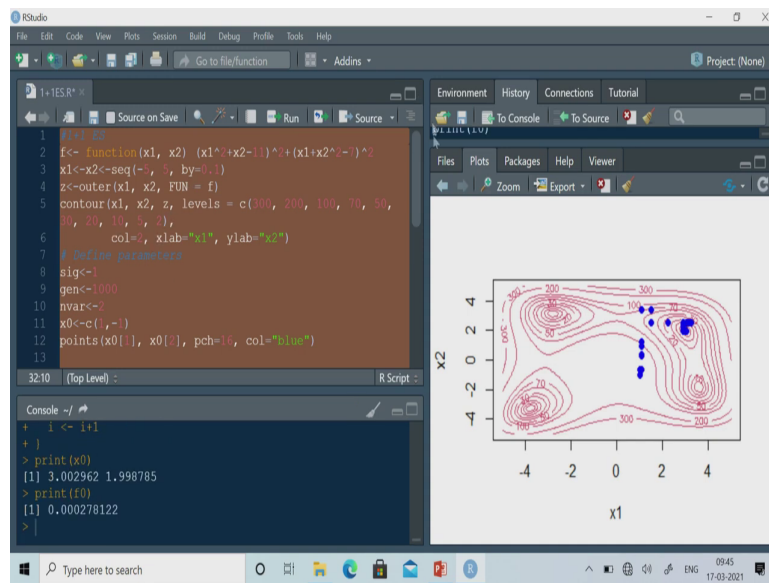


So, this is the solution I am getting, that solution is again 2.99, and 2. Let me try again. Again, I am getting the solution at the first quadrant, ok.

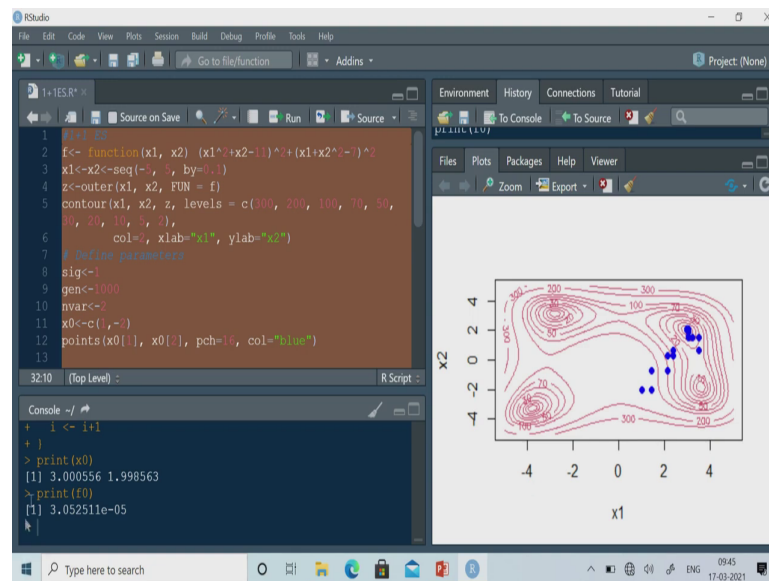
(Refer Slide Time: 39:02)



(Refer Slide Time: 39:11)

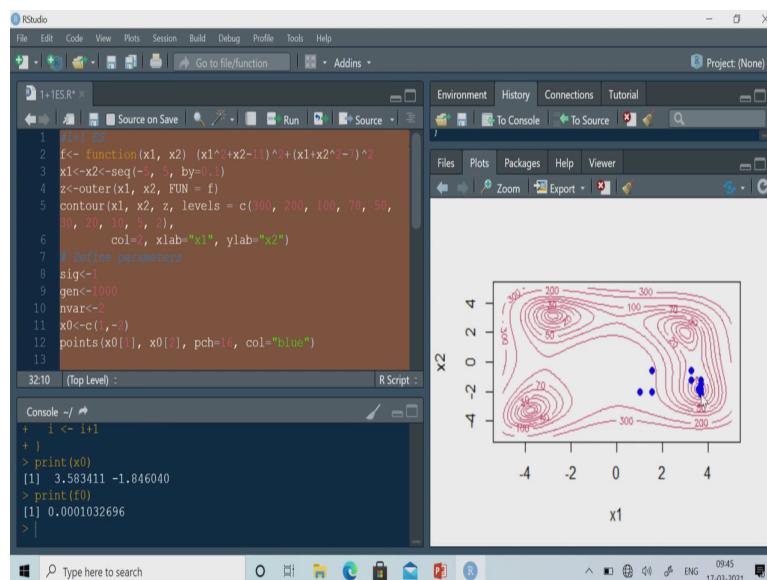


(Refer Slide Time: 39:16)



So, let me try again. So, this time also I am getting the solution at the first quadrant. So, maybe you can try with some other values. This is I am putting minus 2. So, let me try. So, this is the initial solution that is 1 n minus 2. Again, I am getting the solution at the first quadrant, yeah.

(Refer Slide Time: 39:30)



So, this time I am getting the solution at the fourth quadrant. So therefore, because I am using random number, so therefore, every time I may get different solution. If you compare this solution because in this particular approach 1 plus 1 ES, so I am not using the population.

So, what we are doing? From one solution, I am creating another solution. So, this is not a population based algorithm. So therefore, if you are applying this particular algorithm, so you may get the local optimal solution near the this thing. But any you may get the local optimal solution.

So in this case, you are actually doing the local source, not the global source here. So, but as I said we can also use mu plus lambda ES. So, in that case, this is a population base, so initial

population can be distributed over the entire source space. And then in that case, you may be able to find out the global optimal solution of the problem.

So, in the next class or in one of the tutorial, so I will show you $\mu + \lambda$ ES. So, that coding will be little bit more, more than 15 lines, so it may be a 30 line coding and you will be able to implement $\mu + \lambda$ ES.

Thank you.