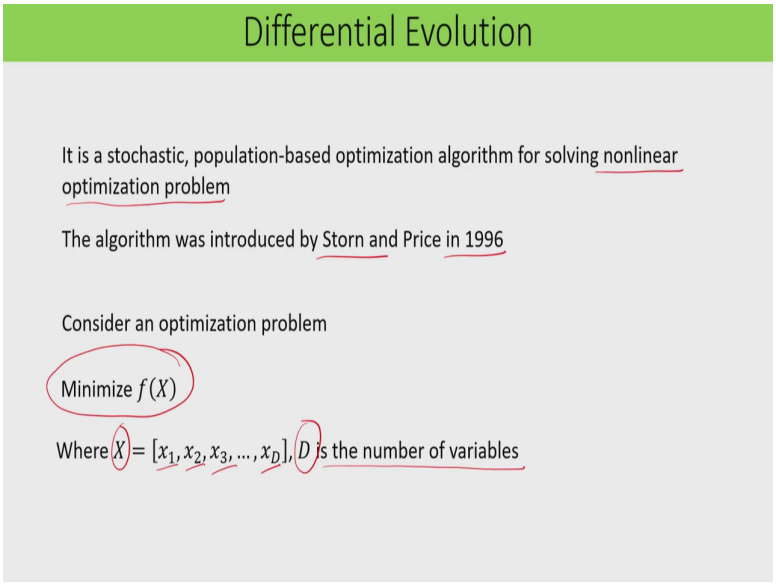
Optimization Methods for Civil Engineering
Dr. Rajib Kumar Bhattacharjya
Department of Civil Engineering
Indian Institute of Technology, Guwahati

Lecture - 30
Introduction to Differential Evolution

Welcome back students. So, today we will discuss another algorithm that is Differential Evolution.

(Refer Slide Time: 00:39)

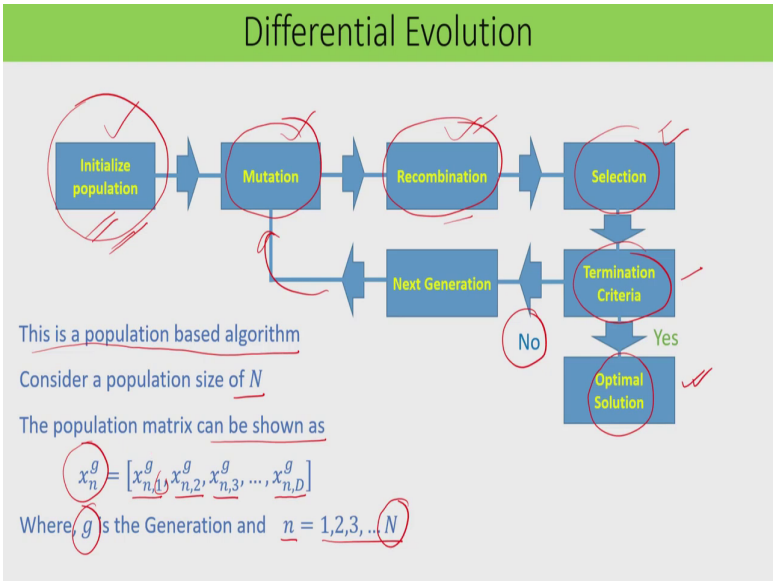


So, differential evolution is a stochastic population based optimization algorithm for solving non-linear optimization problem. Just like genetic algorithm or particle swarm optimization method, so differential evolution is also a population based algorithm. So, we generally used

for solving a non-linear optimization problem. So, this algorithm was introduced by Storn and Price in 1996.

So, let us consider an optimization problem, suppose if we consider a minimization problem here; that is minimize f X and X is a vector and this is a D dimensional problem. So, I have the variable x 1, x 2, x 3 up to x D. So, this is a D dimensional problem, number of variables is D.

(Refer Slide Time: 01:34)



Now, let us see the algorithm. So, here just like genetic algorithm, so we have to initialize the population ok; so in this step, we will initialize the population. So, before that, we have to define the population size; that means the number of solution in the population, so that we have to define, an upper bound and lower bound of its variable you have to define. So, in this case, so I will discuss; so we have to initialize the population in this stage.

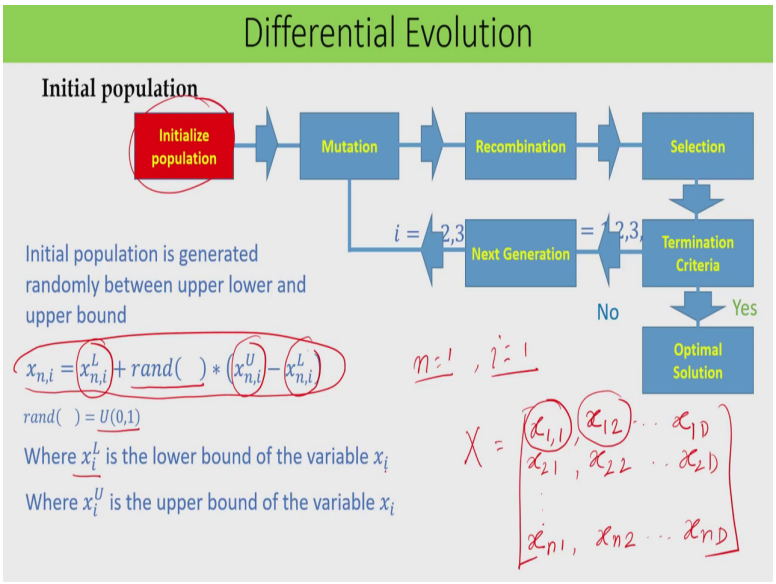
Now, after that we are creating or we are mutating the solution. So, this mutation is different than what we have discussed in case of genetic algorithm. So, that mutation was different, but here mutation process is different; but objective is same that I would like to create a new solution. And then we are using recombination and this is also not similar to what we are using or what we have used in case of genetic algorithm.

So, with recombination, we are creating a new solution and after that we are applying selection operator. So, selection operator is basically used to select the best one and then we will check for termination criteria; if it is satisfied, then we will declare that this is the solution, otherwise this process will continue. So, we will go to the next generation and this process will continue unless and until you are reaching the termination criteria. So, this is the algorithm, so you can see what are the steps.

So, steps are initialized population, then you go for mutation, go for recombination; then you go for selection, then check for termination criteria. If it is satisfied, then you will declare the optimal solution and if it is no, in that case this process will continue. So, as I have said, this is a population base algorithm. So, let us consider the number of solution in a population or population size is N; then the population matrix can be shown as or can be written as like this.

So, x n g, so this is a vector and this vector can be defined is something like that. So, this is the variable 1, this is variable 2, this is variable 3 and this is variable D. So, n is showing the number of population, ok. So, population size is n. So, n is from 1 to N and g is the generation. So, n this 1, 2, 3 up to; so we have D dimension. So, I can say that is a D dimensional problem, small n is showing the population number; that means we have total N population or N solution in the population and g is your generation.

(Refer Slide Time: 04:31)

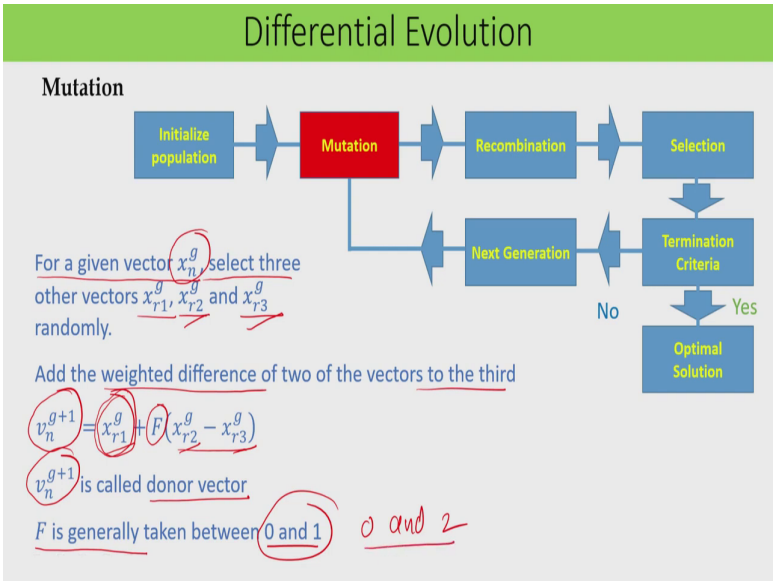


Now, the first step is initialize population. So, I can create the initial solution using this equation suppose x that n equal to 1 and i equal to 1; that means for the first solution and first variable, the lower bound this is x n i L this is the lower bound. Then I am generating a random number between 0 and 1 and then the upper bound of this particular variable and lower bound of this particular variable. So, then I am creating the first solution; suppose I can explain it here. So, this is x 1, 1 and this is x 1, 2; like that this is x 1, D, ok.

So, then this is x 2, 1; x 2, 2 and this is x 2, D, ok. So, similarly I can x n 1, this is x n 2 and then x n D. So, n equal to 1, i equal to 1; means I am generating this particular value. Then similarly n equal to 1, i equal to 2; I will generate this one. So, for 3 like that, so I will generate all these values randomly. So, once I am generating for all n equal to N. So, in that

case I will be getting the total population. So, here as I said that, x i L is the lower bound, lower bound of the variable x i and similarly x i u is the upper bound of the variable x i.

(Refer Slide Time: 06:32)



Now, the next step is the mutation. As I said the mutation here is different than what we have used in genetic algorithm. So, let us see how we are doing the mutation here. In this case for a given vector x n g, select three other vectors; so this is x r 1 g, x r 2 g and x r 3 g randomly. So, what we are doing here from the population, so we are selecting three vectors, that is x r 1, g x r 2 g and x r 3. So, these three vectors are taken from the population randomly in order to get a mutated vector for x n g. Then we are adding the weighted difference of two vectors to the third vector.

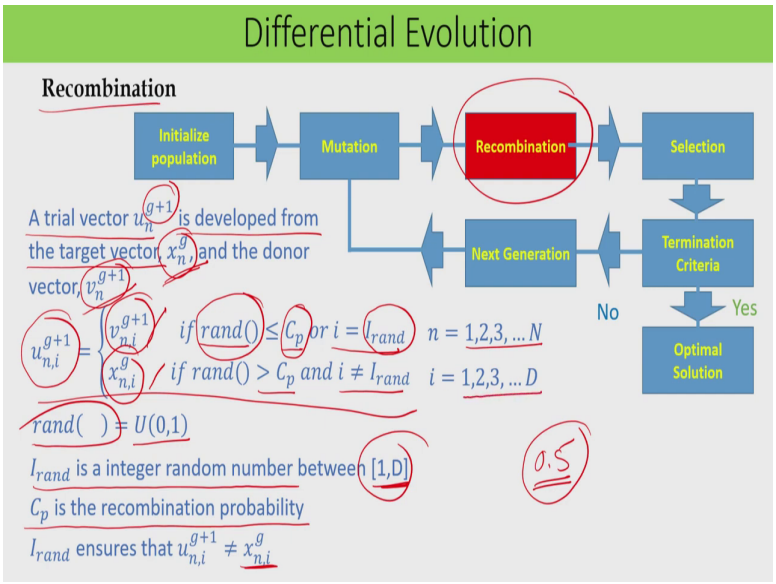
So, what we are doing? So, this is the new vector or we call it donor vector. So, donor vector is created. So, one vector I am taking as a initial solution and then other two vectors are taken

to get the direction and we are multiplying with F. So, F is a number or you can say the parameters of this algorithm. So, and this value is generally between 0 and 1 and some people are also taking between 0 and 2. So, in this case we are considering between 0 and 1 and so, I am getting the donor vector.

So, what I am doing here? So, I am taking three vectors randomly that is r 1, r 2 and r 3; then one vector I am taking as an initial solution and other two vectors are taken to get the direction. And I am getting a another vector in that direction and this vector and the name of this vector is v n g i plus 1. So, this is for the next generation and we call it donor vector. So, as I said, so F is a parameter and it is generally between 0 and 1.

And as I said that some people are considering this value of F between 0 and 2. So, by this I am getting a new vector and I can say that with this mutation, so I am getting a new solution, ok. So, new vector and the name of this vector is donor vector.

(Refer Slide Time: 08:52)



Now, the next step is recombination, so recombination, so this step. So, in this case what we are doing. So, a trial vector u n g i plus 1, so that is for the next generation is developed from the target vector. So, this is the target vector and the donor vector. So, donor vector I have generated for this particular target vector. And so, using these two, so I am trying to get a new vector and we call it trial vector.

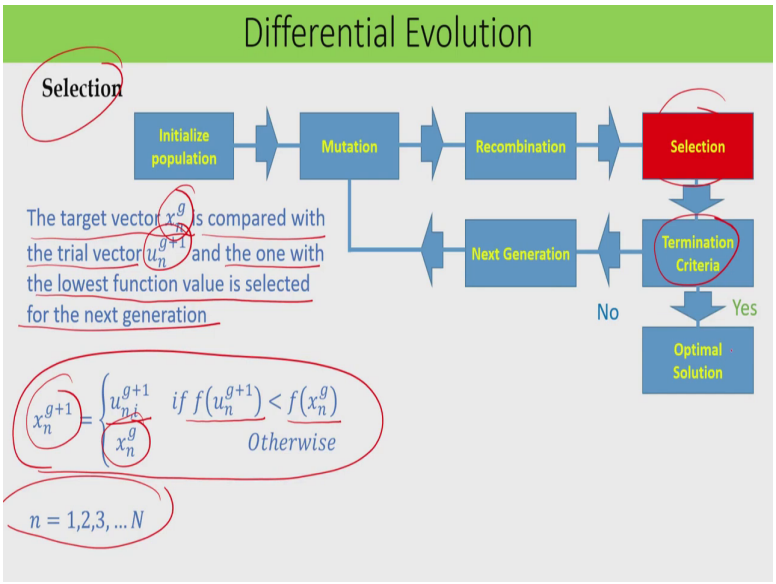
So, you just see now whatever new vector we are getting, so this vector is a combination; that means some variable is from the target vector and some variable from the donor vector. So, we can use this particular process. What is this process? That u n i g plus 1 equal to v n i g plus 1; if a random number, so we are generating a random number and if that random number is less than C p, again C p is the parameter of this algorithm.

So, we will define C p and if it is less than C p, then or i equal to I rand. So, I rand is also generated randomly between 1 and D. So, between 1 and D if rand is less than C p or i equal to I rand; so in that case what I what we are doing, so we are taking that particular variable from the donor vector. Otherwise, if the rand is greater than C p and i is not equal to I rand. So, in that case what we are doing, we are taking from the target vector. So, what is this? This is a combination of the donor vector and target vector.

So, as I said that rand function will give you a random number between 0 and 1 and I rand is an integer random number between 1 and D and C p is the recombination probability. So, suppose if I take C p equal to 0.5; so that means 50 percent we are getting from the donor vector and 50 percent we are getting from the target vector. So, I rand ensures that that u n i g plus 1 is not equal to x n i g, ok. So, this will answer; because once i equal to I rand, so certainly we are getting something from the donor vector.

So, anyway, so depending upon this probability; so you will get your some variable from the donor vector and some variable from the target vector. This process will continue for n equal to 1 to N and i equal to 1 to D. So, you are getting a new trial vector now and this vector is u n i g plus 1. So, what we are doing here? So, we have initially create a donor vector that is v i and after that using v i and the target vector; so using the donor vector and the target vector, so we are creating a new trial vector and that is u n i C plus 1.

(Refer Slide Time: 12:01)



The next step is selection, ok, so this is the step. So, in this step what we are doing; the target vector x n g, so this is the target vector, so for that we have actually created a trial vector, ok. So, this target vector is compared with the trial vector and one with the lowest function value is selected for the next generation. So, what we are doing here? We are now comparing the target vector with the trial vector; if target vector is better than the trial vector, so we will keep the target vector.

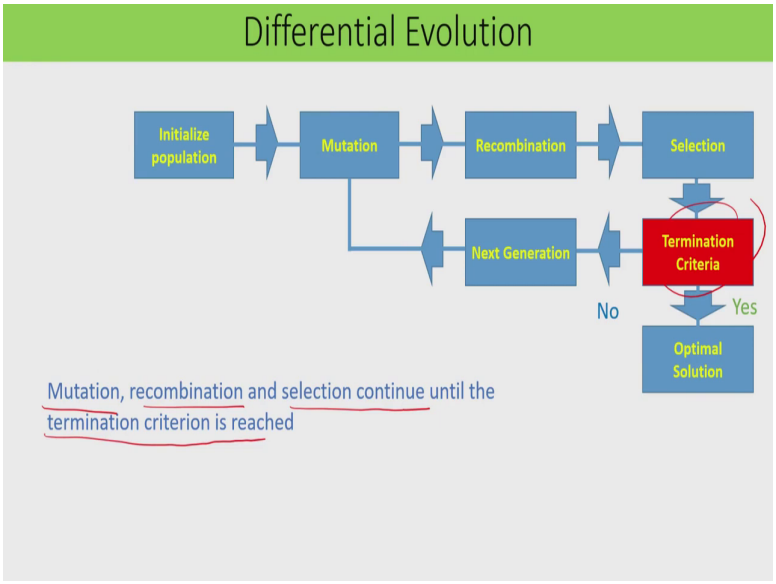
But if the trial vector is better than the target vector, so we will take the trial vector or we will replace the target vector by trial vector. So, this is the replacement we are doing. So, x n g plus 1, so this is the for the next variable. So, this will be equal to this will be equal to the trial vector; if the function value of the trial vector is less than function value of the target

vector. Or otherwise, we will keep the target vector and this process will continue for n equal to 1 to N, ok.

So, we are selecting the better one and then we will check for the termination criteria; if it is satisfied, anyway we will declare the optimal solution or if it is not satisfied, then this process will continue. Again, we will go for mutation, then again we will go for recombination in mutation; we will create a donor vector and using donor vector and the target vector, so we will try to get the trial vector. And then trial vector and the target vector will be compared in this step and the better one will be taken and then again we will check the termination criteria.

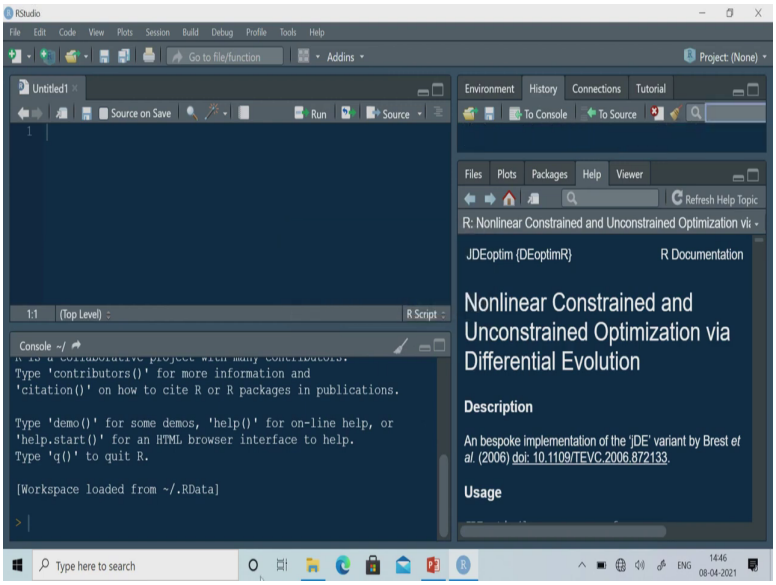
So, this process this iteration process will continue till we are not satisfying or till target termination criteria is not satisfied, ok. And once it is satisfied, then you will declare the optimal solution.

(Refer Slide Time: 14:07)



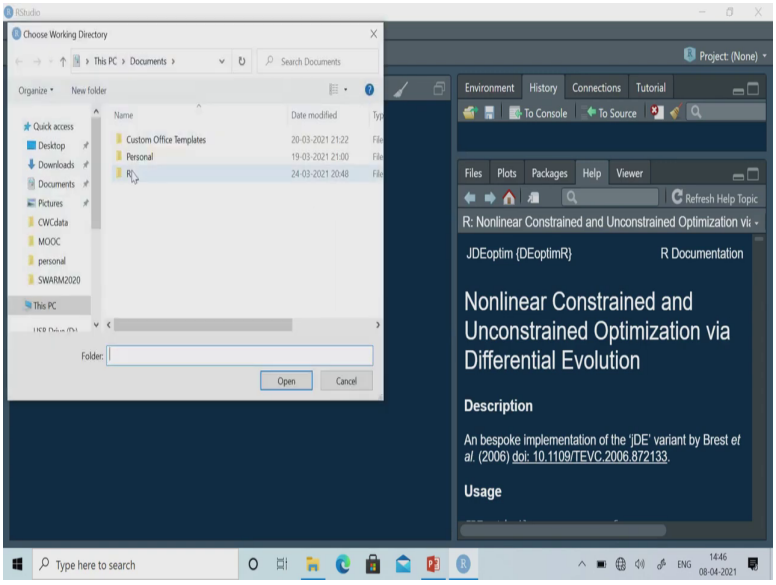
So, mutation, recombination and selection will continue until the termination criteria criterion is not reached. So, this is all about the differential evaluation, so as you have seen. So, this is a very simple algorithm, but again it is a very powerful. So, we can actually solve a very complicated or you can say that. So, we can solve a non-linear problem using this algorithm. Now, I will show you how you can solve a non-linear problem using differential evolution or in R platform, ok. So, we will use the R library for solving the problem.

(Refer Slide Time: 14:54)

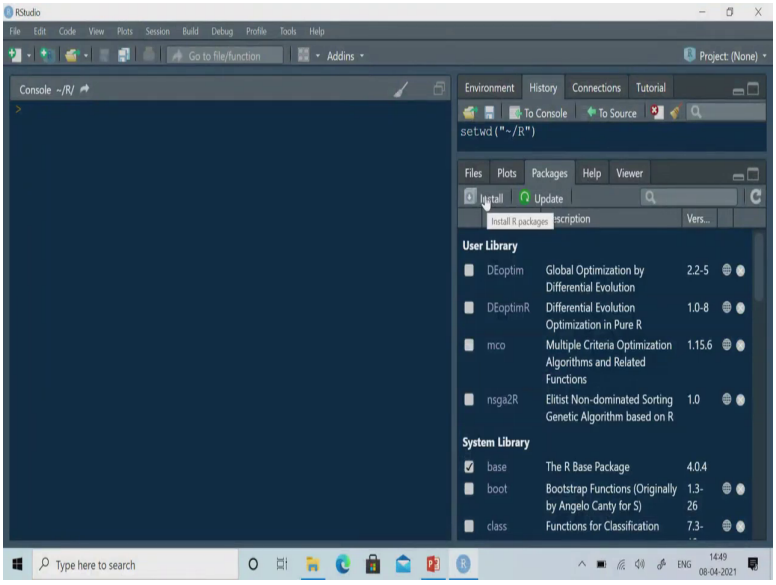


Let us now open the R studio. So, this is the R studio, maybe I would like to clear the environment. Now, let us select an working directory.

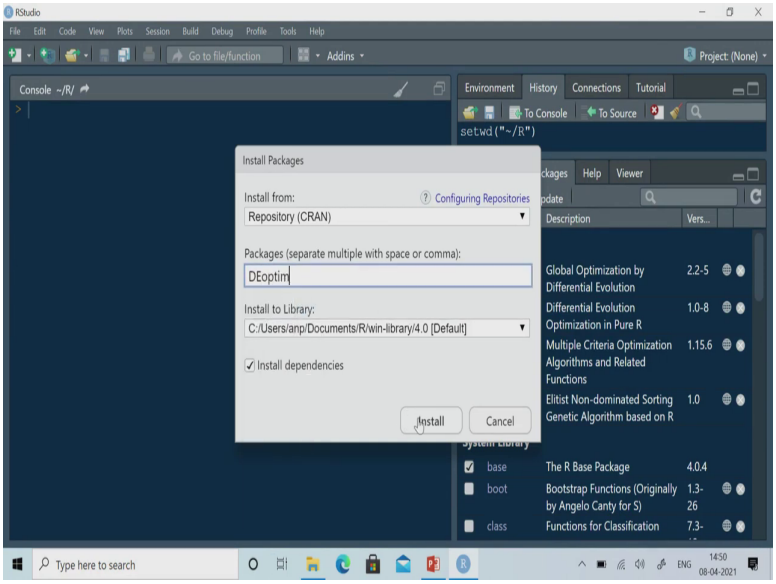
(Refer Slide Time: 15:04)



(Refer Slide Time: 15:14)

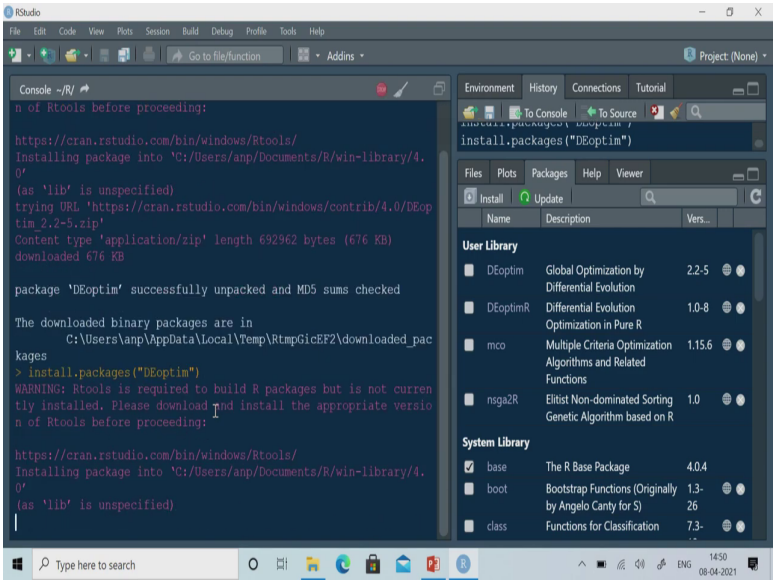


(Refer Slide Time: 15:20)



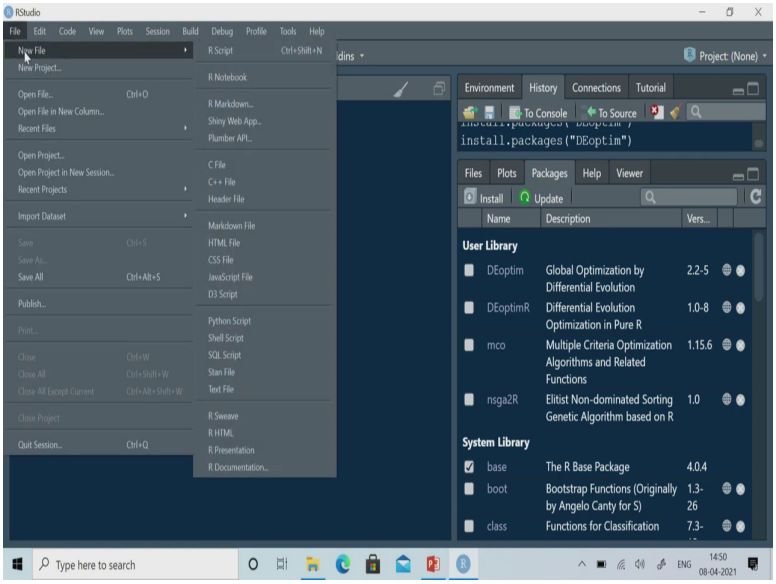
So, I will be working here, so I will be using de optim package here. So, let us install that one. So, I have already installed here, but I would like to show you how you can install. So, you can go to install and you can write here. So, D Eoptim, so this is DEoptim, so you can install.

(Refer Slide Time: 15:30)

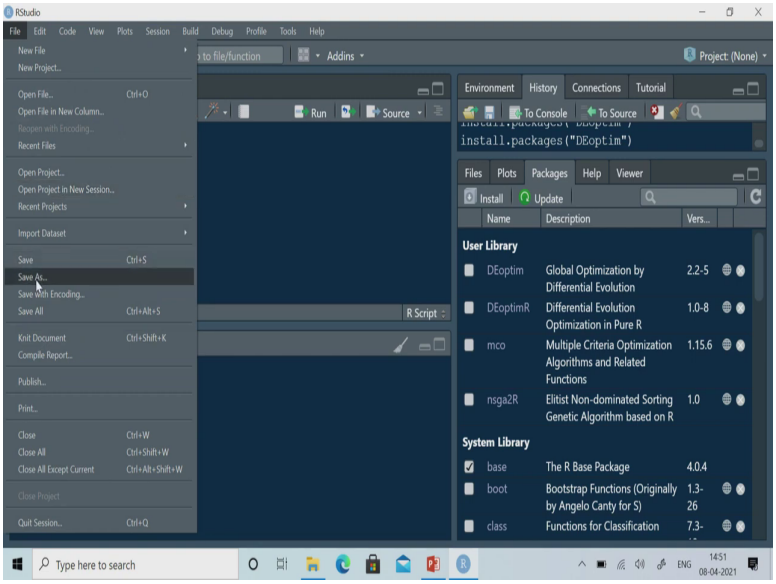


So, now this package will be installed here; this has been installed or otherwise you can also install from here. So, I can write install, install packages DEoptim. So, I can also install from here using the command line, ok, so it has been installed.

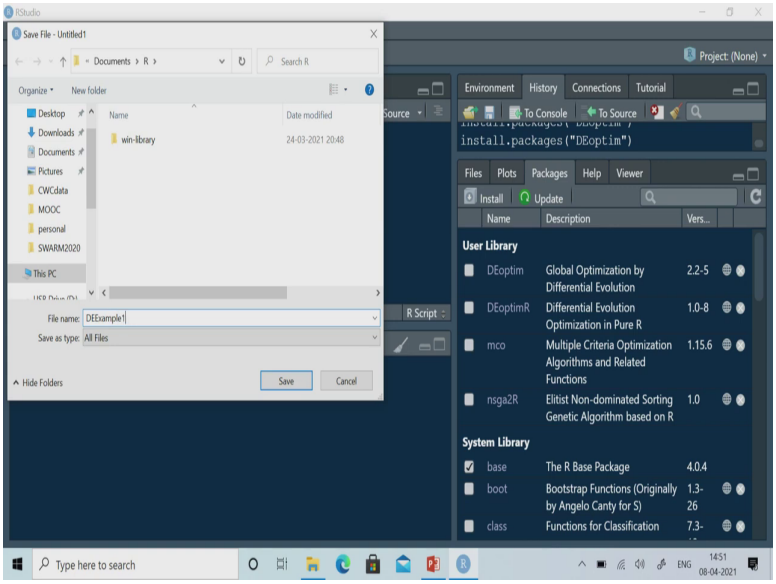
(Refer Slide Time: 15:59)



(Refer Slide Time: 16:05)

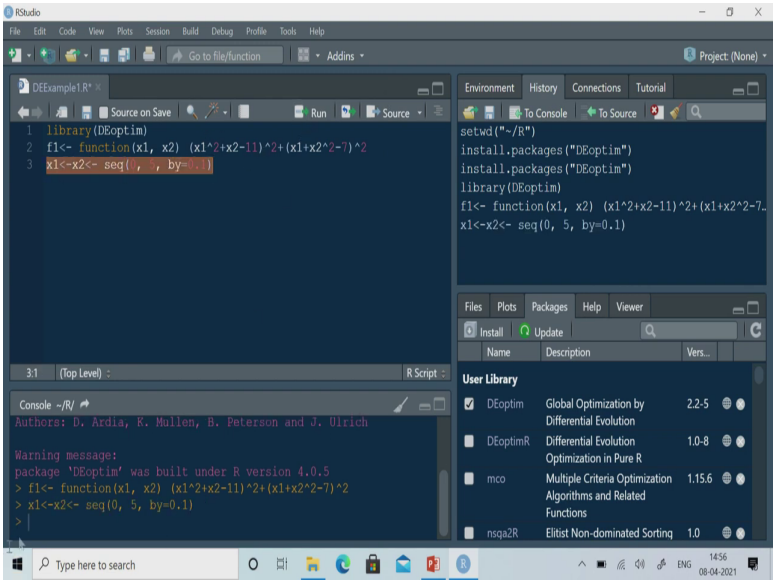


(Refer Slide Time: 16:10)



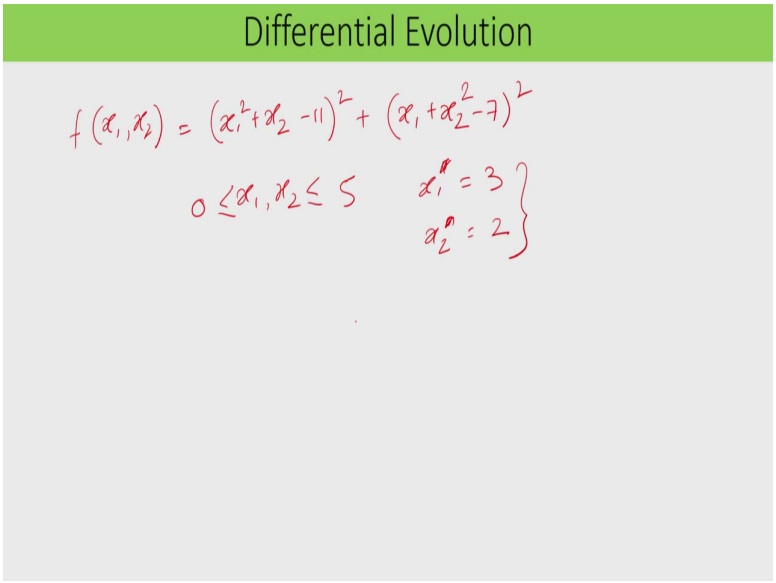
Now, let us open a new file R script. So, I would like to save this file first, so save, so this is my DEexample, example 1.

(Refer Slide Time: 16:27)



So, here first I have to include the library. So, I can write library, this is DEoptim, so I have included the library and then I have to write the function.

(Refer Slide Time: 16:45)



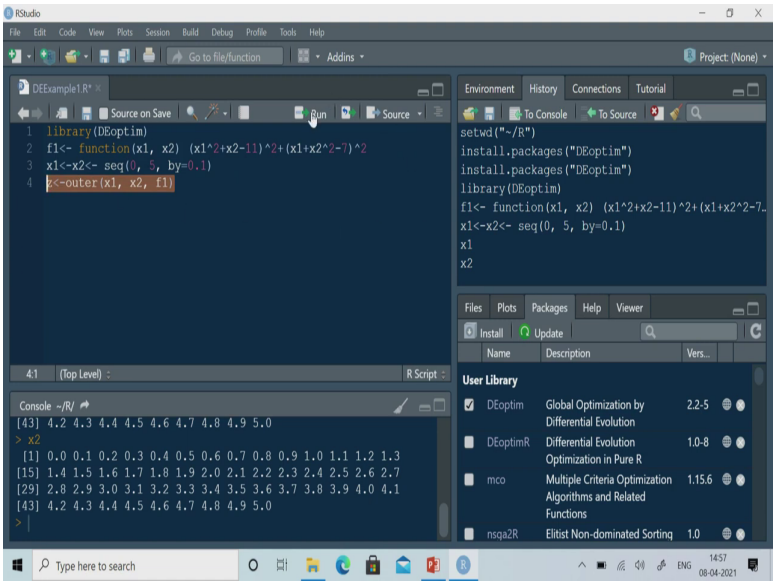
In this case I will be using this function first; that is f of x 1, x 2, which is equal to x 1 square plus x 2 minus 11 whole square plus x 1 plus x 2 square minus 7 whole square. So, if I take the range of x 1, x 2 between 0 and 5; x 1 star equal to 3 and x 2 star equal to 2. So, I should get this particular solution. So, let us see if I apply d, whether I will get the solution or not. So, now, in R I will write the function. So, give the name of the function, suppose I am writing f 1. So, this is the function I can write, ok.

So, this is a function of; first I would like to plot this function. So, for plotting that one, so I am writing x 1 and x 2. So, this is x 1 square plus x 2 minus 11 whole square plus x 1 plus x 2 square minus 7 whole square. So, this is the function I am writing here. So, what I will do? I will let me execute this line. So, I will include the library and then I am executing this particular function, I am executing this particular function.

So, you can see this function should be somewhere here. So, now, I have to define x 1 and x 2. So, x 1 equal to x 2, so both are between 0 and 5. So, I will use the sequence function here.

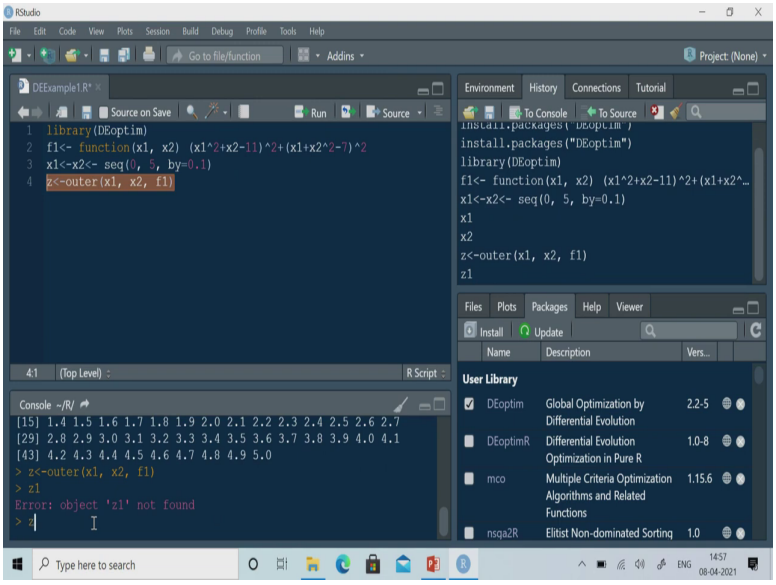
So, this is s e q function to generate x 1 and x 2 value between 0 and 5 and by the difference is 0.1, ok.

(Refer Slide Time: 19:30)

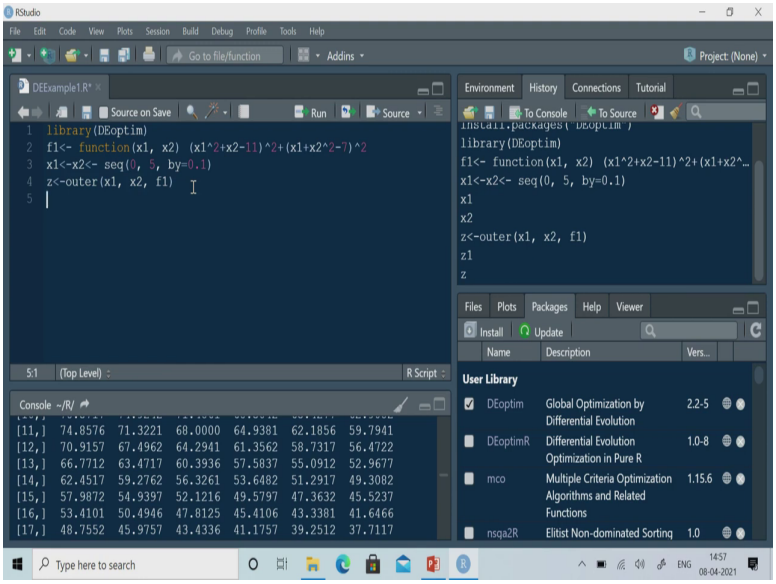


So, let me execute this particular line. So, you can see what is x 1; so x 1 is between 0 and 5 and the interval is 0.1, similarly I am getting x 2 also. Now, what I will do? I will create the function value at its grid point, ok. So, for that I will use the outer function. Now, I am calculating the z value and that is I am calculating the function value at its grid point. So, I will use the outer function, so z equal to outer. So, this is the outer function and this is x 1, x 2 and then I have to write the f 1, ok. So, you can see. So, if I execute this particular line. I will get the function value, sorry this is z.

(Refer Slide Time: 20:28)

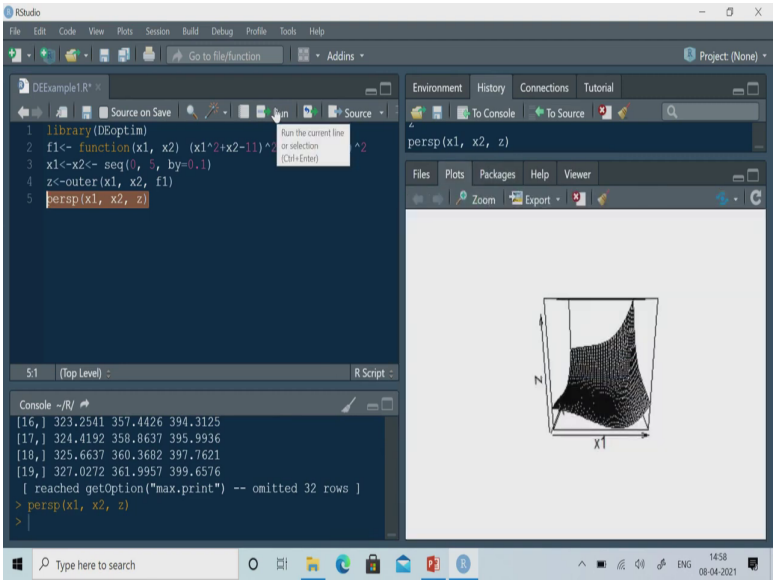


(Refer Slide Time: 20:34)



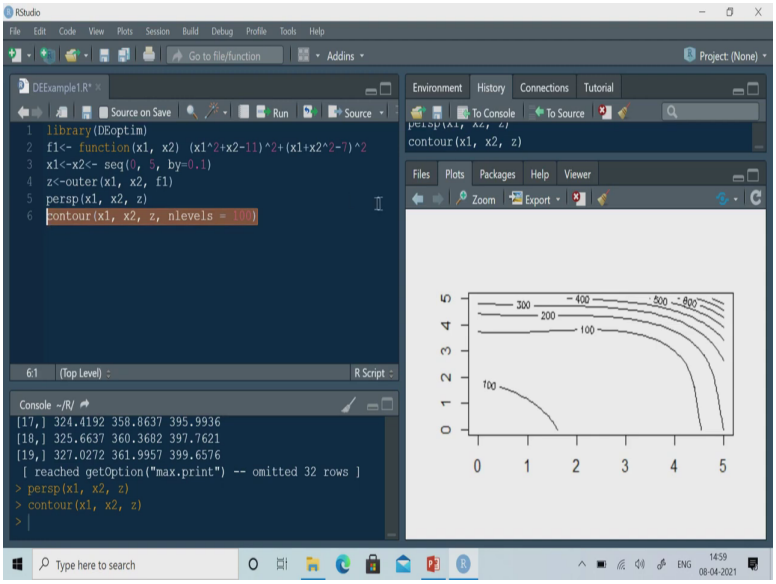
So, I will get the function value at each grid point, ok. So, now, I can plot it. So, I can plot either surface plot or contour plot. So, if I want to plot surface, so in that case I can use p e r s p function, ok.

(Refer Slide Time: 20:51)



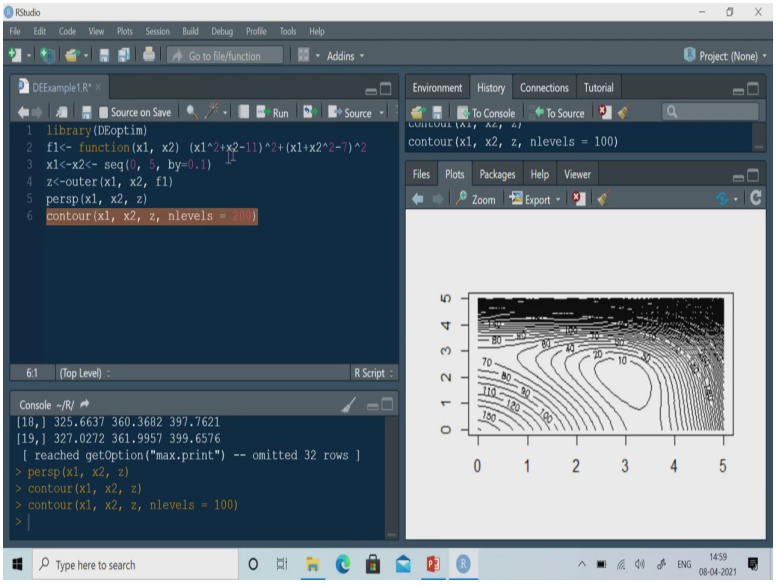
So, this is p e r s p, so this is the function. And so, what I have to pass here? So, this is x 1, this is x 2 and z. So, you can see that if I execute this. So, I will get the surface plot. So, I will get the surface plot of this function or I can plot the contour also c o n t o u r contour and this is x 1, x 2, z. So, you can see if I execute this, I will get the contour plot.

(Refer Slide Time: 21:38)



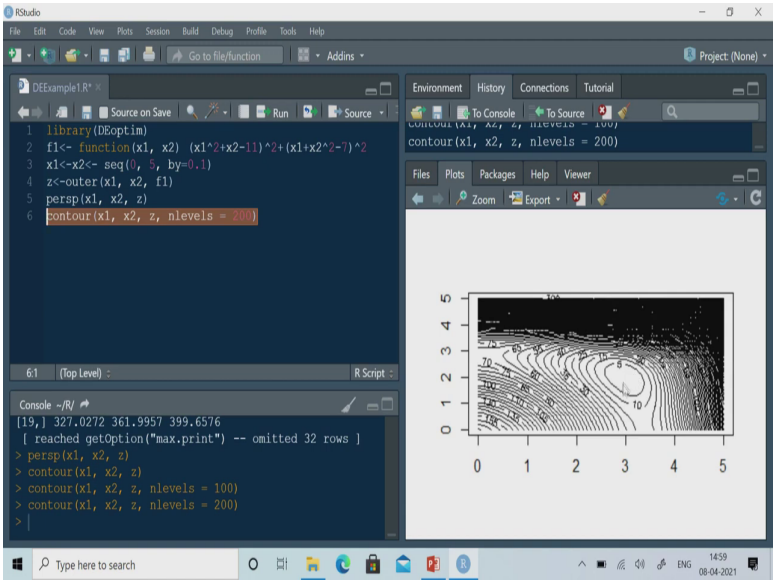
So, I am getting the contour plot here. Now, I am I can increase the number of control, that is n level. So, n level suppose if I put 100, so it will increase the number of contour lines.

(Refer Slide Time: 21:54)



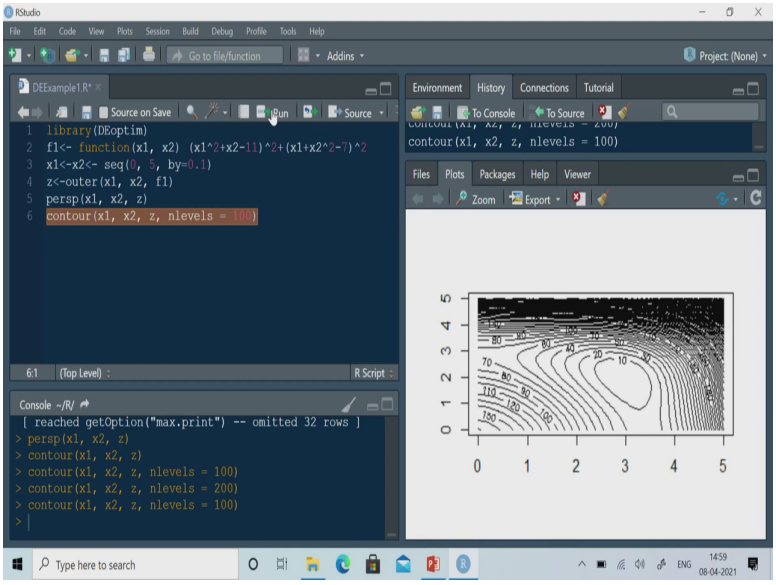
So, you just see the solution is somewhere here that, x 1 equal to 3 and x 2 equal to 2; maybe I can also increase it up to 200, ok.

(Refer Slide Time: 22:06)



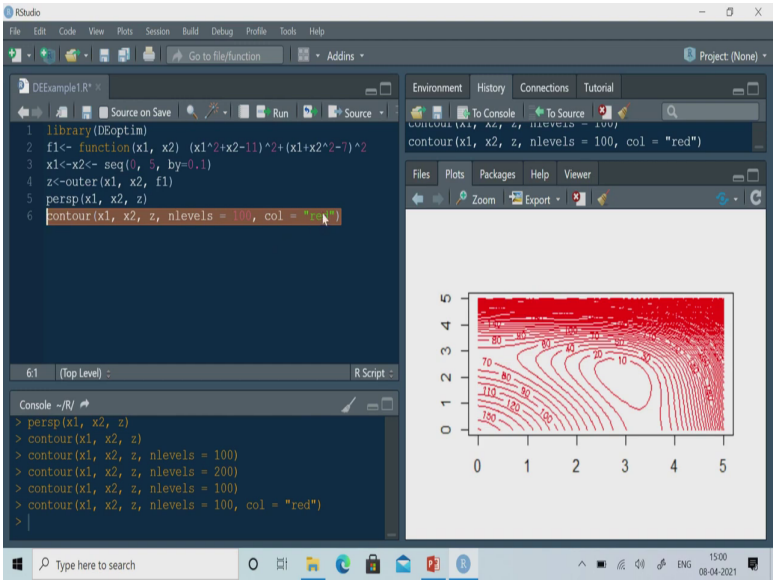
So, somewhere here, solution is somewhere here.

(Refer Slide Time: 22:11)



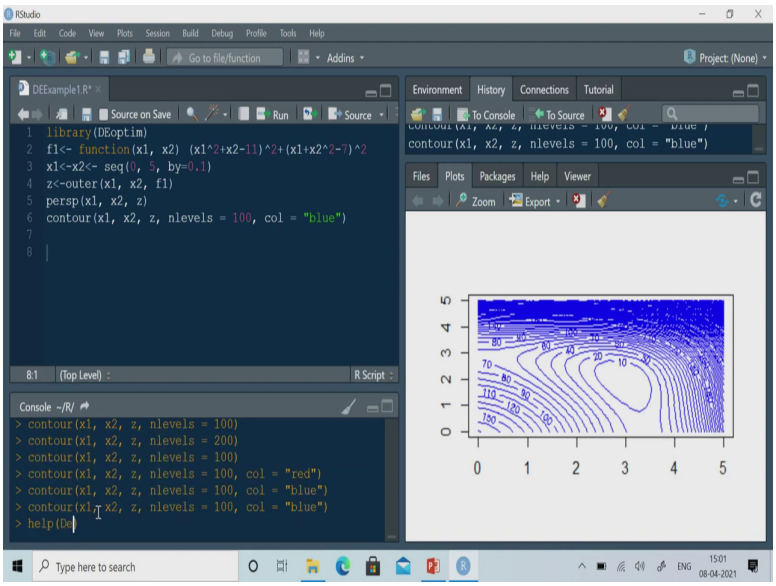
So, I think 100 is sufficient, so let me plot this one yeah.

(Refer Slide Time: 22:18)


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
DEExample1.R*
Source on Save
Run
Source
1 library(DEoptim)
2 f1<- function(x1, x2) (x1^2+x2-11)^2+(x1+x2^2-7)^2
3 x1<-x2<- seq(0, 5, by=0.1)
4 z<-outer(x1, x2, f1)
5 persp(x1, x2, z)
6 contour(x1, x2, z, nlevels = 100, col = "red")
6:1 (Top Level)
R Script
Console ~/R/
> persp(x1, x2, z)
> contour(x1, x2, z)
> contour(x1, x2, z, nlevels = 100)
> contour(x1, x2, z, nlevels = 200)
> contour(x1, x2, z, nlevels = 100)
> contour(x1, x2, z, nlevels = 100, col = "red")
Environment History Connections Tutorial
To Console
To Source
contour(x1, x2, z, nlevels = 100, col = "red")
Files Plots Packages Help Viewer
Zoom
Export
0 1 2 3 4 5
Type here to search
ENG
15:00
08-04-2021

So, I can change the color also. So, color I can change; suppose if I put red or blue I can put, any color I can put, yeah I will get the red contour lines ok or you can also put blue.

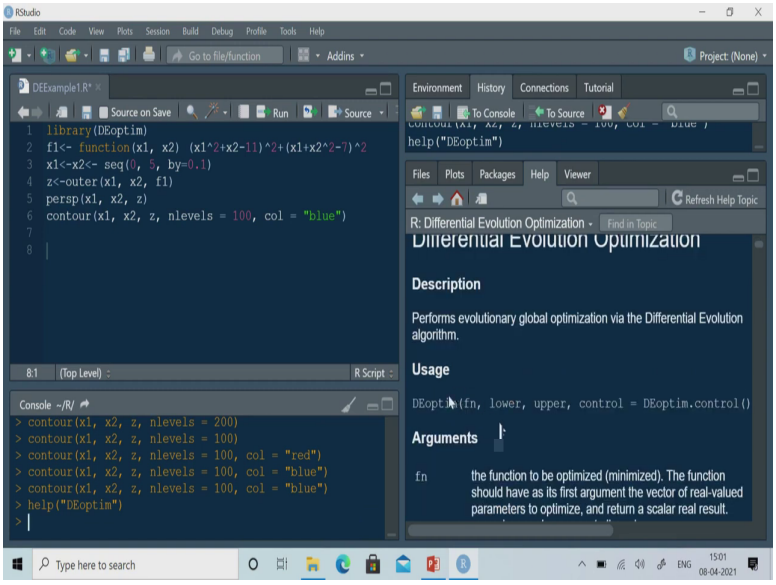
(Refer Slide Time: 22:44)



So, you are getting blue lines, ok. So, this is just to plot the function. So, I would like to get or once we are applying D E. So, I should get this solution that is your 3 and 2.

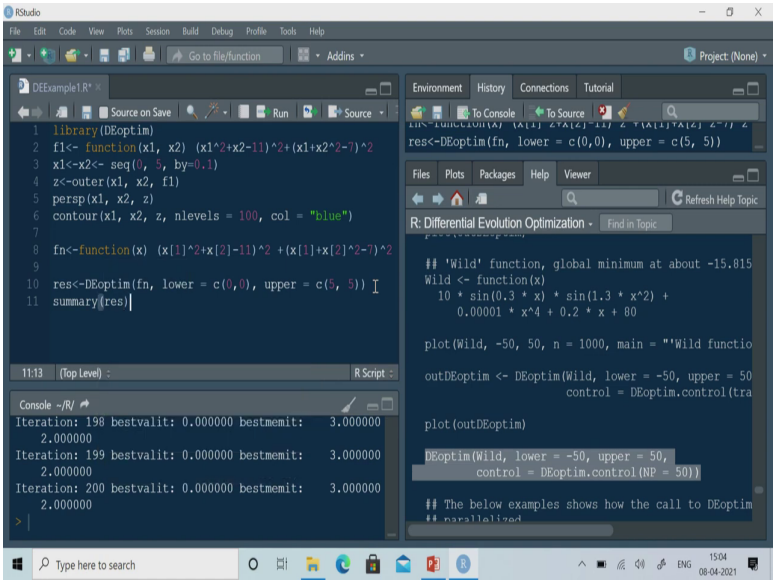
Now, for D E I have already included the library, so that is DEoptim and I have to use the function DEoptim. So, let us see what is this particular function. So, we can go to help, that is D E; let us go to help and we will use DEoptim, ok.

(Refer Slide Time: 23:34)



So, you can see, so this is the D E optim function. So, what are the arguments? So, I have to put this function and then lower bound, upper bound and there are some control parameters, ok. So, control parameters means, that is the parameter of the algorithm you can suppose number of iteration, then c value, f value, so that you can define here, ok. So, you can see this one.

(Refer Slide Time: 24:01)

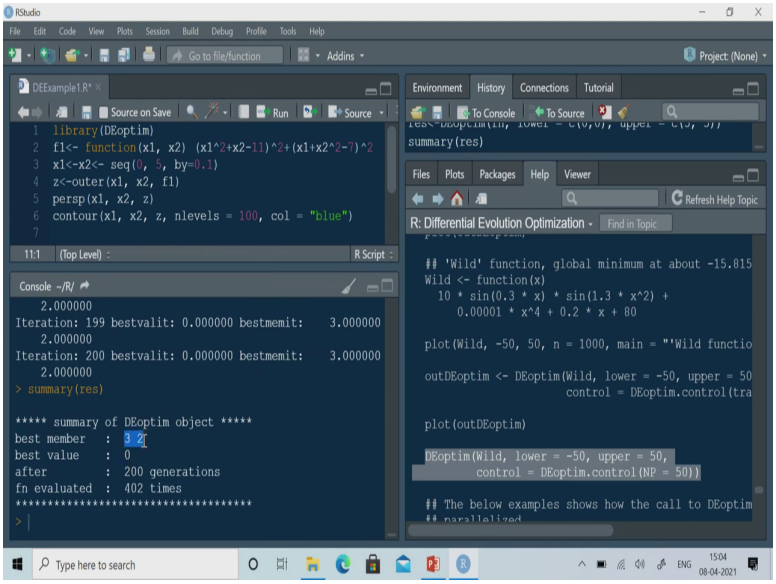


So, this is the function, ok, so this is the function, but I will type it here. So, I will use the function is DEoptim, ok. So, DEoptim and I have to use this function; but before that in that case, I cannot define x 1 and x 2 something like that, so I have to define x as a vector. So, what I will do; so I will create a function that f n. So, in this case I have to define as a vector, ok. So, now, I will do this is x 1 square plus x 2 minus 11, this is whole square plus this is x 1 plus x 2 square minus 7 whole square, ok.

So, I am defining it as a vector, x as a vector, ok, so I hope this is fine yeah. So, now, I have executed this one, so now, this function I have to use. So, this is f n; then I have to use the lower and upper bound, so lower equal to c. So, now, it is between lower is 0 and 0. So, between 0 and 5, so x 1 is between 0 and 5; x 2 is between 0 and 5. So, lower is 0, 0 and upper is 5, 5, so I think it is fine. So, other parameters, so I will use the default value. So, let me store this result somewhere. So, r e s, so I am storing this result.

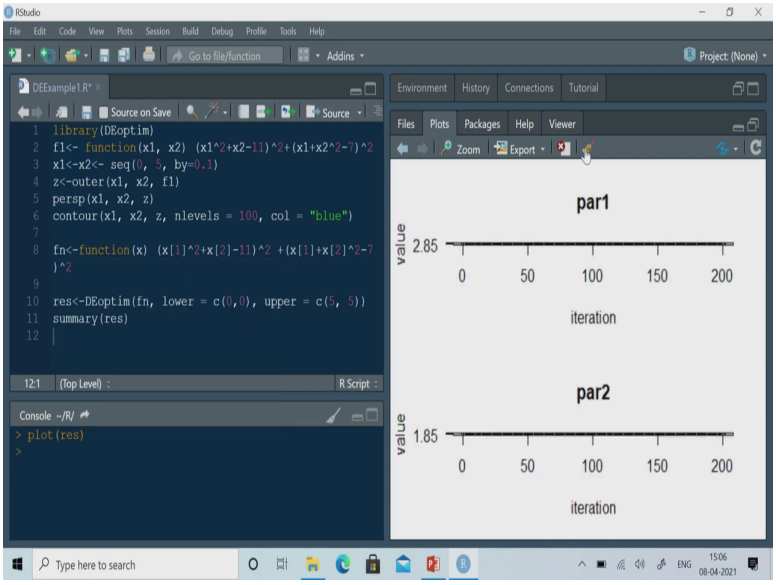
And if I execute this particular line, so this will optimize this function f n between lower bound and upper bound and the result will be stored in r e s, ok. So, let us execute. So, if it is if there is no issue, then I think we should get the solution, ok. So, we have executed up to 200 iteration and I can see whether I am getting the solution or not. So, let us see summary of r e s.

(Refer Slide Time: 26:35)



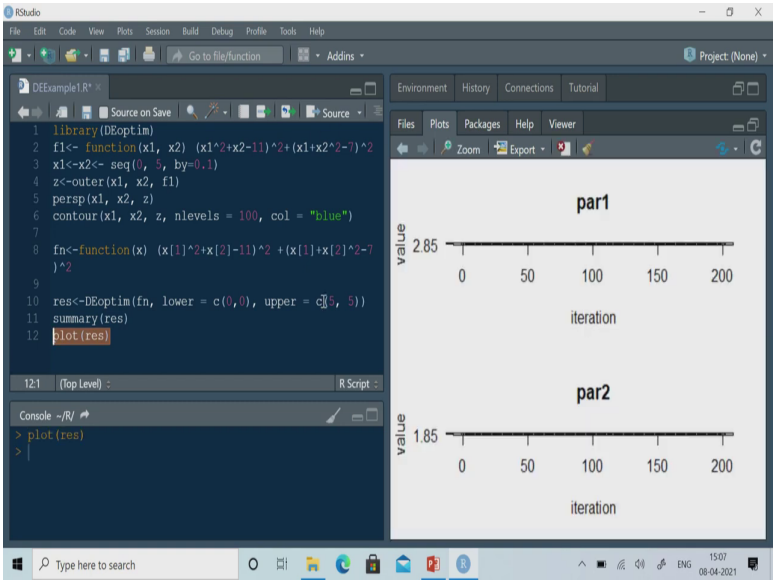
So, what I am getting here summary, I am getting the solution that is 3 2 ok, exact solution I am getting and function value is 0 and iteration is 200. So, that means this is the default value, it that number of iteration that number of generation is 200. And how many function evaluation; that is 402 times function evaluation.

(Refer Slide Time: 27:06)



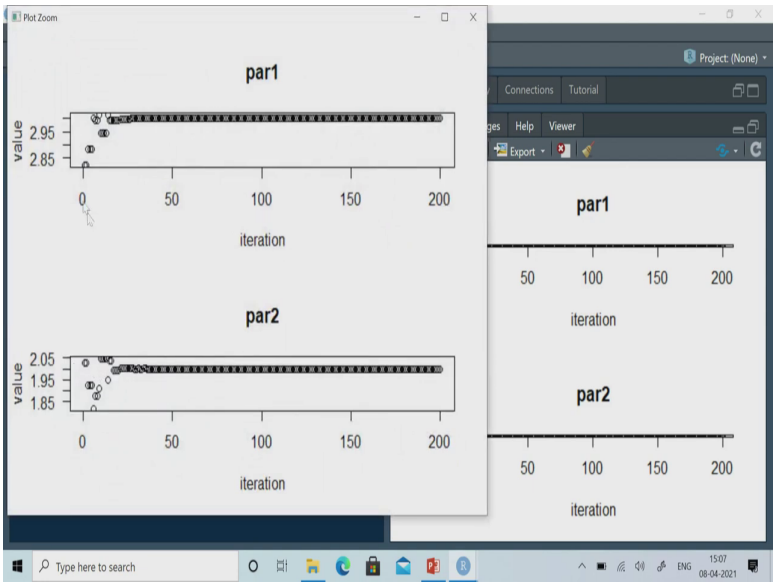
So, I am getting the exact solution. So, I can also see the plot, so if I plot r e s, so just see what I am getting, ok. So, I think there is some problem with this.

(Refer Slide Time: 27:40)



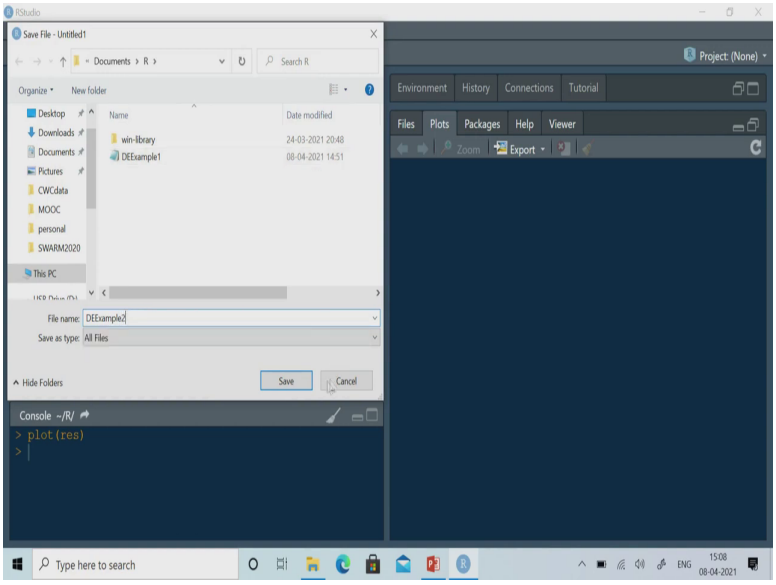
Let me see if I plot r e s, what I will get.

(Refer Slide Time: 27:48)



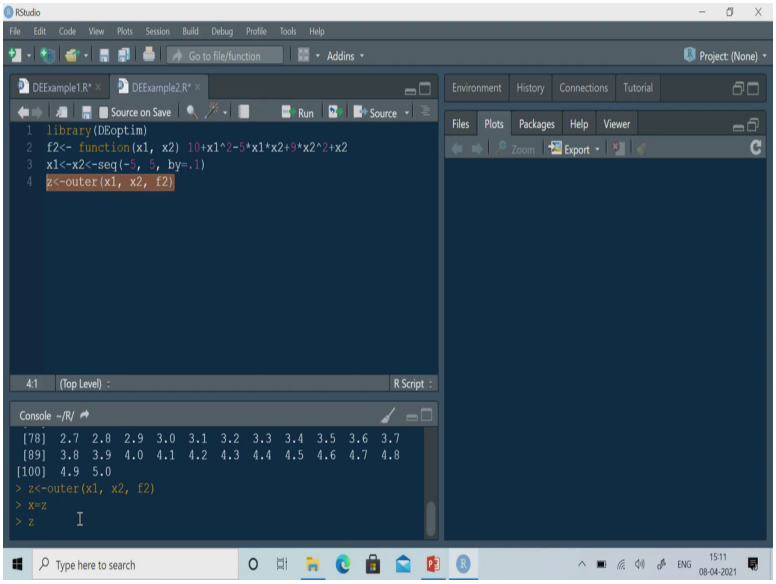
So, you can see that, this is the plot I am getting between iteration and value. And you can see this is for parameter, this is for first variable and this is for second variable. So, I am getting the solution; suppose this solution is first value is x equal to 3, x 1 equal to 3 and x 2 equal to 2. So, this is you can see that after few iteration, the solution is near 3 and 2. So, it is giving the best solution at different iteration. So, I got this solution of this particular problem. So, let us see another example problem. So, I will open another file, so this is a new R script and let me save this one.

(Refer Slide Time: 28:47)



So, this is now example 2.

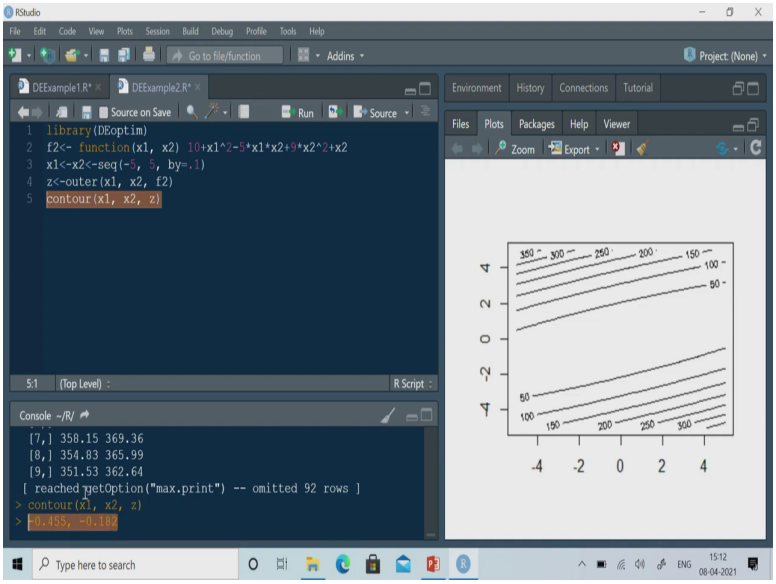
(Refer Slide Time: 28:54)



So, this time also I have to include the library. So, library this is DEoptim, DEoptim, ok. So, this is the library; this time I will use a different function and the function is, so let me write the function first and this is f 2. So, just to plot this one; so I am writing x 1, x 2 and the function is 10 plus x 1 square minus 5 star x 1 star x 2 plus 9 star x 2 square plus x 2. So, the function is 10 plus x 1 square minus 5 x 1 x 2 plus 9 x 2 square plus x 2. So, let me run this particular line.

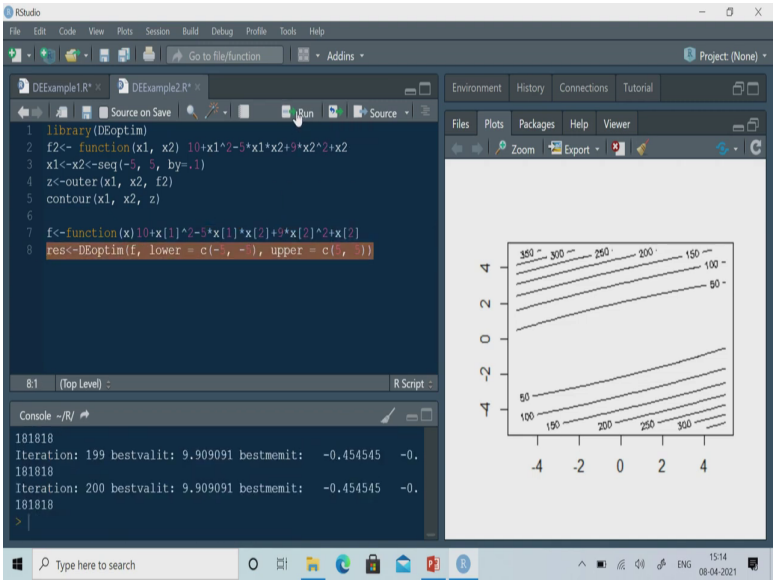
So, now, I will generate the x 1 value and x 2 value. So, using sequence function and that is between minus 5 and 5 by 0, 1 let me see. So, I can see what is x 1 yeah and x 1 you can see, this is from minus 5 to plus 5 and similarly x 2 is also between minus 5 to plus 5. Now, I will use the outer function to calculate the z values. So, outer this is x 1, x 2 and the function is f 2. So, I should get the z values, sorry z values.

(Refer Slide Time: 31:23)



So, now I can plot the contour; contour this is x 1, x 2 and z, let me execute this one. So, this is the contour and in this case the solution is minus 0.455 and. So, x 1 equal to minus 0.455 and x 2 equal to minus 0.182. So, I should get this solution; that is minus 0.455 and I should get minus 0.182.

(Refer Slide Time: 32:14)

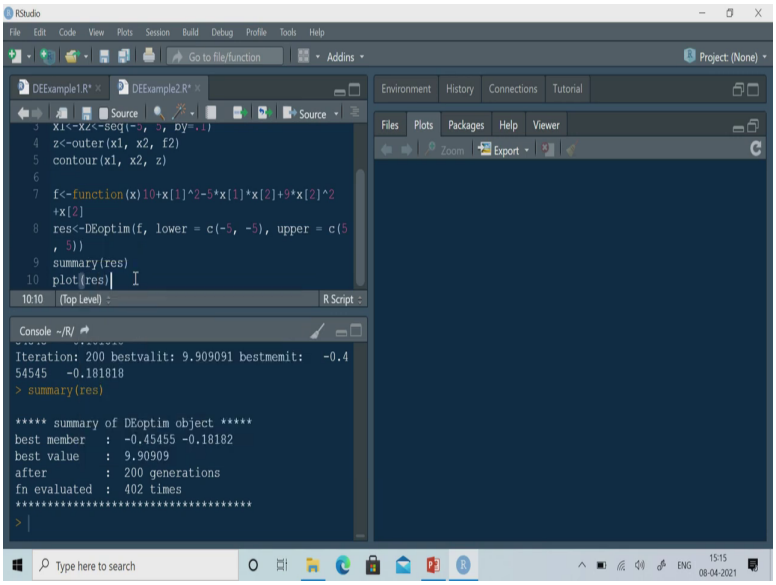


So, let me check that one, so whether I am getting this solution or not. So, for applying DE; so what I have to write, I have to define the function. So, in terms x we have to define as a vector. So, let me write this function. And so, I would like to copy this here, ok. So, now, this is x 1 ok, this is x 1, this is x 2 and this is also x 2, ok. So, this is x 1 square minus 5 x 1 into x 2 plus 9 x 2 square plus x 2, ok, I hope this is fine. So, let me execute this one yeah. So, now, I will use the DEoptim. So, this is DEoptim. So, here what you have to do, you have to define the function.

So, now function name is f. So, I will define lower, so lower is c, that is minus 5 and minus 5; then upper is c, this is 5, 5, ok. So, I am not changing the other parameters of the algorithms. So, I am using the default value only. So, now, if I execute this one, so I should get the

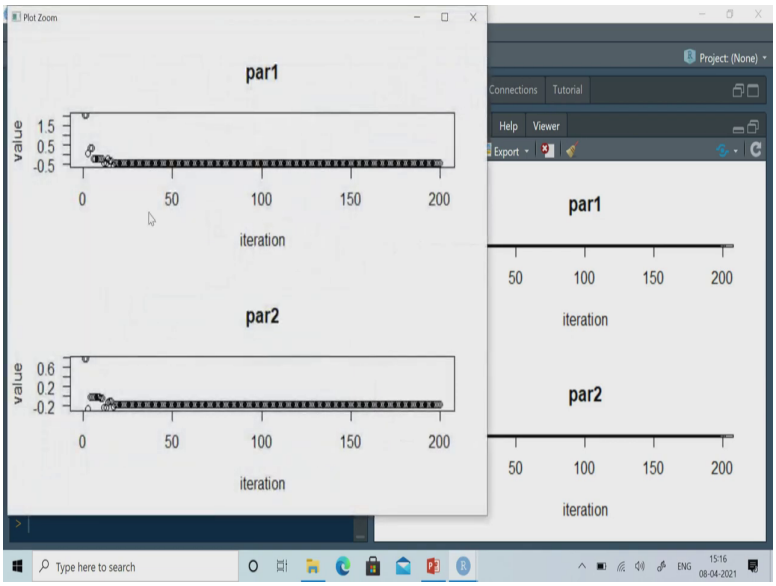
solution; but I would like to store the solution in result, ok. So, let me check if I; if it is correct, then I should get the solution.

(Refer Slide Time: 34:26)



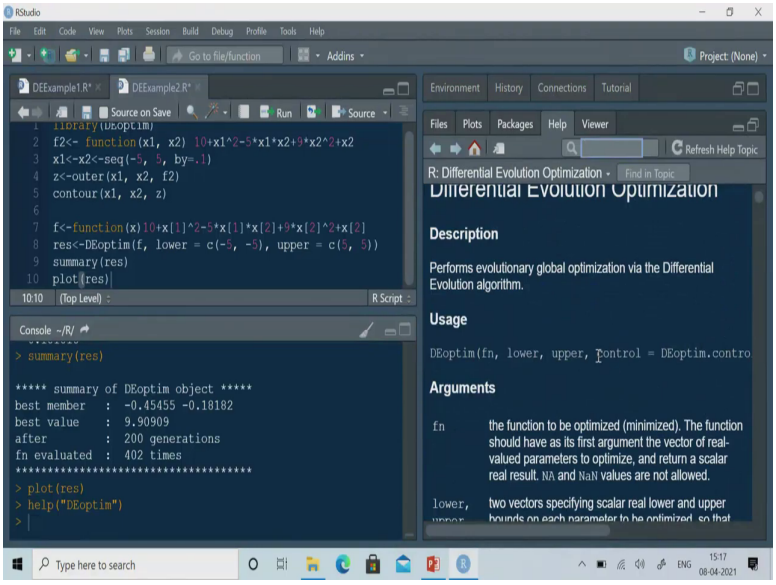
So, let me see the summary of the solution, summary of res, ok. So, I am getting the solution as minus 0.45455 and minus 0.18182. So, as I said the x value solution is minus 0.455; yeah I am also getting the same thing minus 0.45, it is 5 and then minus 182. So, this is I am getting 0.18182, ok. So, I am getting the solution. So, here also the generation is 200 generation and number of function evaluation is 402. So, I am getting this solution. So, if I want I can also see the plot, so let us see. So, if I if I plot it, if I plot res, so just see.

(Refer Slide Time: 35:34)



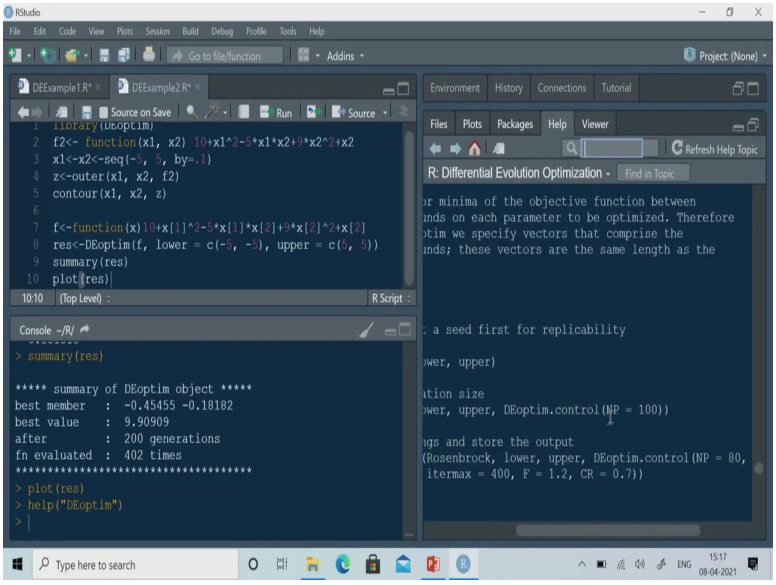
So, this is the plot, you can see the solution of first one is minus 0.455. So, somewhere here minus 4.55; the solution is minus 0.55 for x 1 and x 2 is minus 0.18, 0.182, ok. So, this is the iteration versus the solution of x 1 and x 2. So, I can see and overall I am getting the solution of this particular problem.

(Refer Slide Time: 36:12)



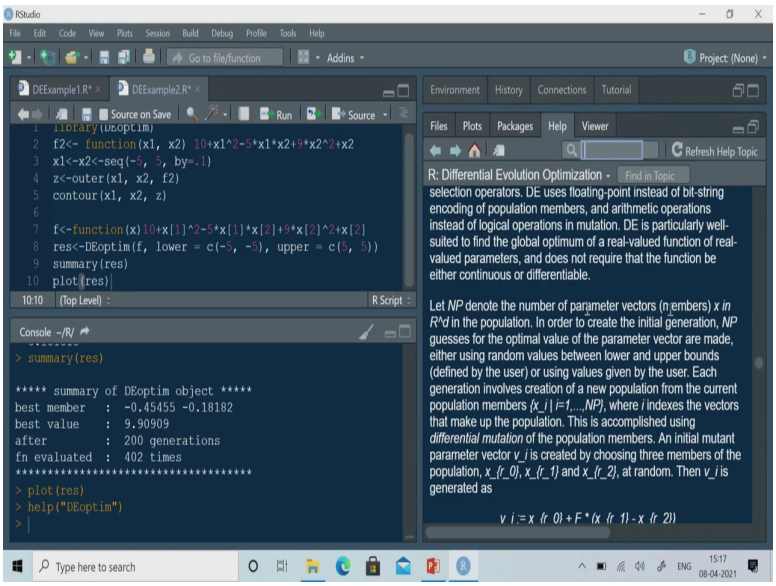
So, here I have not changed the parameters of the algorithm, but if you want you can also change the parameters. So, you can go to help and then this is DEoptim. So, if I want I can change the parameter, ok. So, under control, ok, so within control I can change the parameter. And so, you can see that one. So, there is some example problem also how can, how you can change the parameters. So, I can write that NP.

(Refer Slide Time: 36:44)



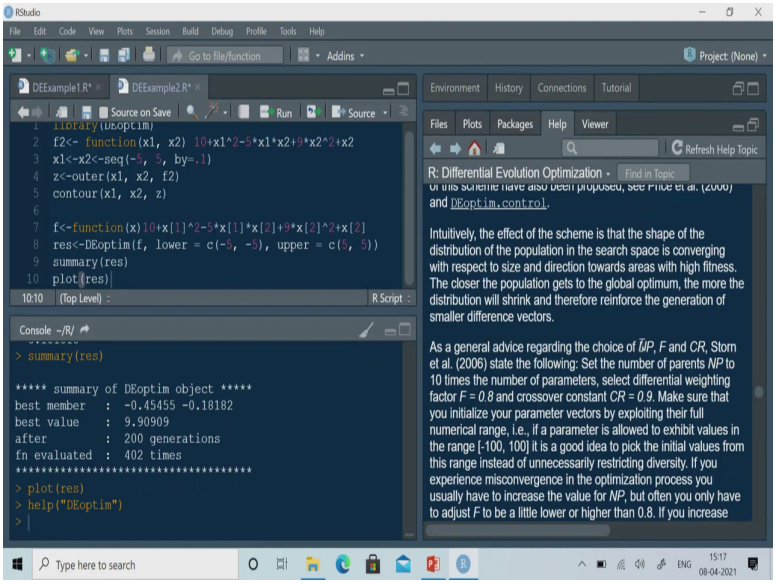
So, what is NP? You can you can see actually, under the control parameter it should be defined.

(Refer Slide Time: 36:56)



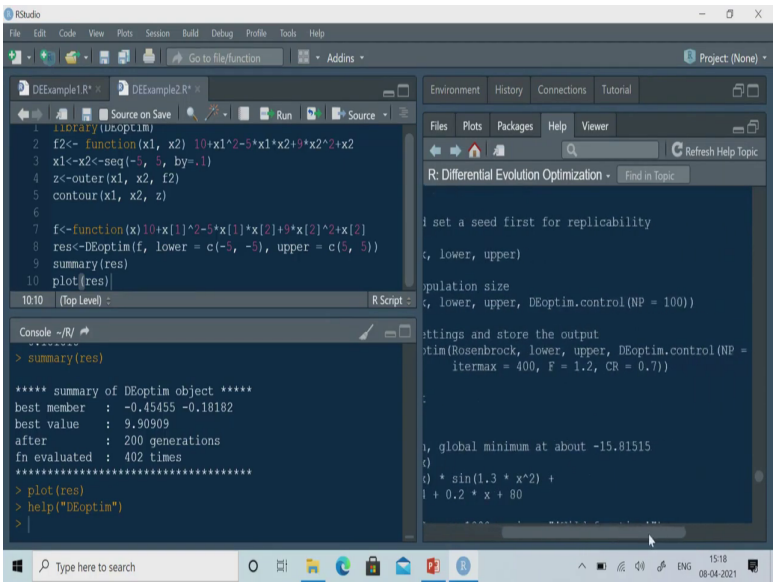
So, number of your parameter vectors, that is the member. So, that is your NP, ok, so number of your parameters.

(Refer Slide Time: 37:06)


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
DEExample1.R*
DEExample2.R*
Source on Save
Run
Source
library(DEoptim)
f2<- function(x1, x2) 10+x1^2-5*x1*x2+9*x2^2+x2
x1<-x2<-seq(-5, 5, by=.1)
z<-outer(x1, x2, f2)
contour(x1, x2, z)
f<-function(x)10+x[1]^2-5*x[1]*x[2]+9*x[2]^2+x[2]
res<-DEoptim(f, lower = c(-5, -5), upper = c(5, 5))
summary(res)
plot(res)
10:10
(Top Level)
R Script
Console ~/R/
> summary(res)
***** summary of DEoptim object *****
best member : -0.45455 -0.18182
best value : 9.90909
after : 200 generations
fn evaluated : 402 times
> plot(res)
> help("DEoptim")
Environment
History
Connections
Tutorial
Files
Plots
Packages
Help
Viewer
Refresh Help Topic
R: Differential Evolution Optimization
Find in Topic
and DEoptim.control.
Intuitively, the effect of the scheme is that the shape of the distribution of the population in the search space is converging with respect to size and direction towards areas with high fitness. The closer the population gets to the global optimum, the more the distribution will shrink and therefore reinforce the generation of smaller difference vectors.
As a general advice regarding the choice of NP, F and CR, Storn et al. (2006) state the following: Set the number of parents NP to 10 times the number of parameters, select differential weighting factor F = 0.8 and crossover constant CR = 0.9. Make sure that you initialize your parameter vectors by exploiting their full numerical range, i.e., if a parameter is allowed to exhibit values in the range [-100, 100] it is a good idea to pick the initial values from this range instead of unnecessarily restricting diversity. If you experience misconvergence in the optimization process you usually have to increase the value for NP, but often you only have to adjust F to be a little lower or higher than 0.8. If you increase
Type here to search
ENG
15:17
08-04-2021

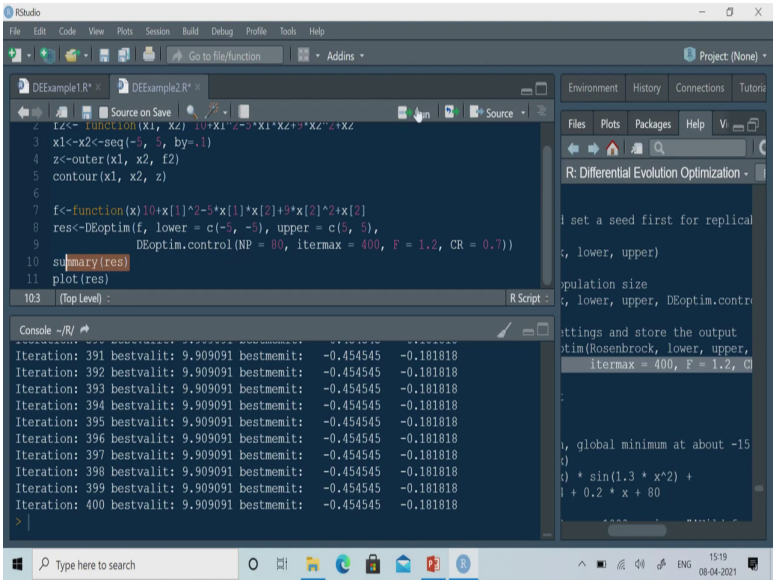
So, NP, so we have we can define NP that is your population, then F and CR, ok. So, I can define using within the control, ok.

(Refer Slide Time: 37:18)



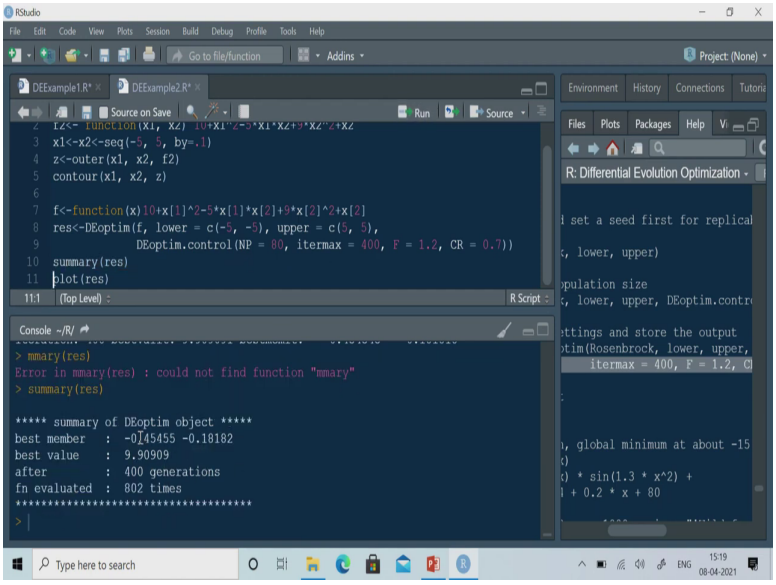
So, maybe I can copy this part. So, I can write it here.

(Refer Slide Time: 37:30)



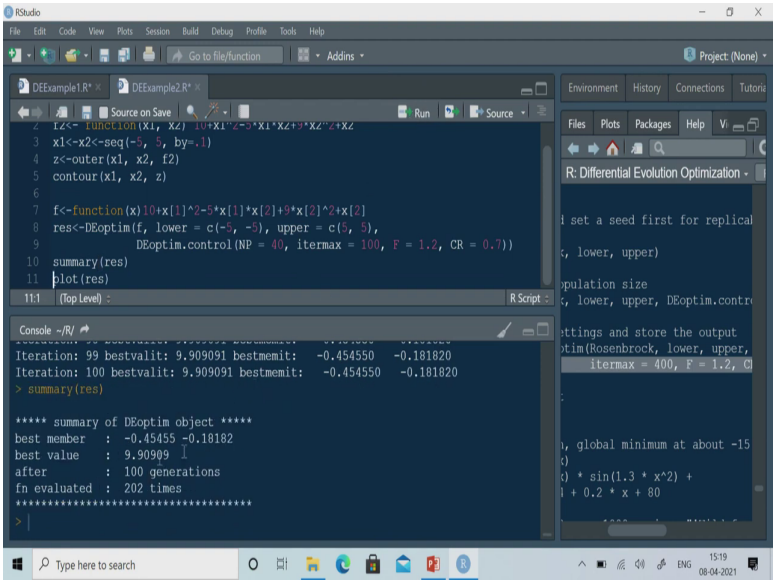
So, now what I am doing here. So, NP that the population size I am putting 80 now, iteration by default it is 200 iteration, so I am going up to 100 iteration. So, F value as I said that some people are taking between 0 and 1, some people are taking between 0 and 2. But here suppose if it is defined 1.2 and CR value, so 0.7, ok. So, you can also define other CR value, but we are considering 0.7 here. So, let us see, let us execute this one, ok, so now it is going up to 400 and the population size is 80, ok. And F value is 1.2 and CR value is 0.7.

(Refer Slide Time: 38:24)



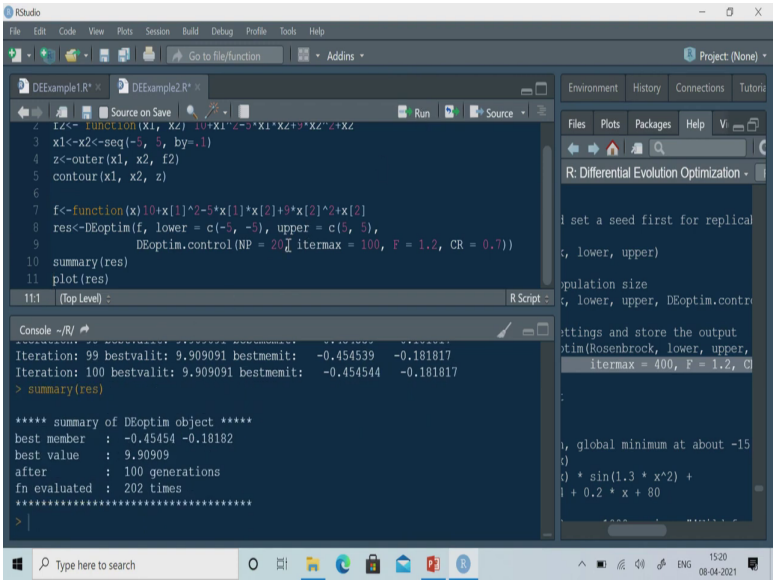
And let us see the summary. So, we you are getting the solution that is 0.455 and this is 0.18182. So, now, in this case this is generation is 400 and function evaluation is 802. So, number of functional value is more; you can see whether in 100 iteration I am getting or if I change the population size to 40, so what will happen.

(Refer Slide Time: 38:45)



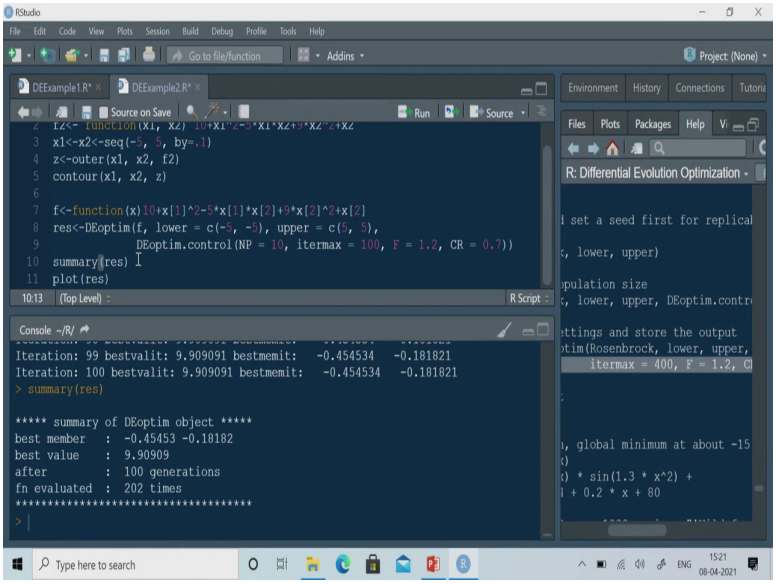
Let me execute this line, let me execute this one; then also I am getting the solution. So, let us see, reduce this one to 20, ok.

(Refer Slide Time: 39:01)



So, let me execute this one; yeah I am getting the solution with 20 population also. So

(Refer Slide Time: 39:11)



, let us see if I population size is 1 10; then what will happen? Then also I am getting the solution. So, as you have seen the differential evaluation, so in case of genetic algorithm; so we need little bit more population or population size is more, but in this case you can see that this for this particular problem, even population size of 10 is also sufficient to get the exact solution of this problem.

So, you have to try with this or you have to change this parameters. So, one is the population size, then iteration, then F value. So, as I said that F value is between 0 and between 0 and 1 or between 0 and 2 basically. So, you can take around 1 or 1.5 or something you can take and CR value you can take around 0.5, 2.7 something like that. So, you can try with this parameter.

So, if you are not getting the solution of this problem, solution of your problem; so you can try with different parameters and you can see whether you are getting any improved solution

or not. So, with this let us stop here. So, today we have discussed differential evaluation and also we have solved some problem using R program, ok.

Thank you.