Optimization Methods for Civil Engineering

Prof. Rajib Kumar Bhattacharjya
Department of Civil Engineering
Indian Institute of Technology, Guwahati

Lecture - 33
A Tutorial on Differential Evolution

Hello student. So, in this class we will solve some problem using differential evolution. In the last class, I have shown you how you can solve a unconstrained problem so unconstrained problem using differential evolution. In this class, we will solve some constrained optimization problem using differential evolution.

(Refer Slide Time: 00:57)

DE Tutorial

Problem No 1

Minimize $f(x) = (x_1 - 2)^2 + (x_2 - 1)^2$

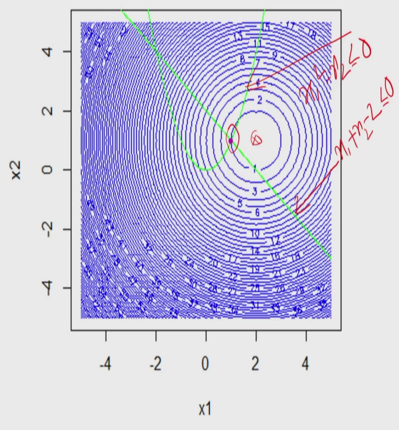
Subject to $x_1 + x_2 - 2 \le 0$

$x_1^2 - x_2 \le 0$

Range $-5 \le x_1 \le 5$ and $-5 \le x_2 \le 5$

The minimum point of the problem is $x^* = (1,1)^T$ and $f(x^*) = 1$

DE

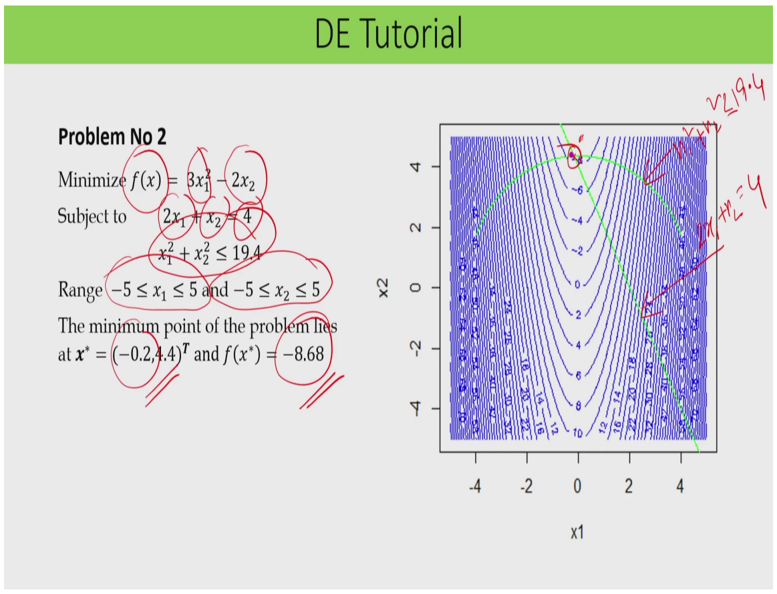


So, we will mainly consider three problems here. The first problem is a minimization problem. It is a minimization of f x. So, it is a function of two variable that is your x 1 and x 2 and x 1 minus 2 whole square plus x 2 minus 1 whole square. The unconstrained optimal solution of this problem is 2 and 1, but we have 2 and 1 so, somewhere here. So, this is the unconstrained solution of this particular problem and, but we have to constrain here that is x 1 plus x 2 minus 2 is less than equal to 0 and x 1 x 1 square minus x 2 is less than equal to 0, ok.

So, this is the first constrain that is x 1 plus x 2 minus 2 less than equal to 0 and this is your second constrain that is your x 1 square minus x 2 less than equal to 0. So, therefore the solution has to satisfy these two constrains. So, therefore the constrained optimal solution of this problem is somewhere here and the solution is 1 1 and the function value is 1 f x.

So, we will apply the differential evolution to solve this problem and we should get the solution that is x star equal to 1 1 and function value at optimal solution is equal to 1. So, this is the first problem we will solve using differential evolution.

(Refer Slide Time: 02:26)



DE Tutorial

Problem No 2

Minimize $f(x) = 3x_1^2 - 2x_2$

Subject to $2x_1 + x_2 \le 4$

$x_1^2 + x_2^2 \le 19.4$

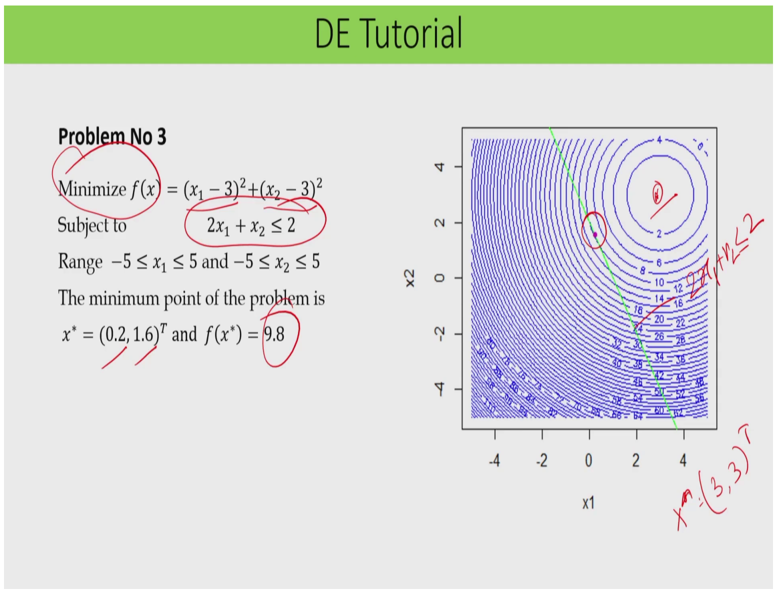
Range $-5 \le x_1 \le 5$ and $-5 \le x_2 \le 5$

The minimum point of the problem lies at $x^* = (-0.2, 4.4)^T$ and $f(x^*) = -8.68$

The second problem is also a minimization problem and the function is 3 x 1 square minus twice x 2 that is the that is your objective function. So, and if you minimize it, so then you should get solution somewhere here. So, for this trends that is x 1 is varying between minus 5 and plus 5 and x 2 is varying between minus 5 and plus 5, but there are two constrains one is a linear constrain that is your twice x 1 plus x 2 equal to 4.

So, this is the linear constrain. So, that is twice x 1 plus x 2 equal to 4 and we have one non-linear constrain that is x 1 square plus x 2 square less than equal to 19.04. So, this is the constrain that is x 1 square plus x 2 square less than equal to 19.4, ok. So, if we solve it the constrain solution is minus 0.2 ok. So, this is somewhere here this is the constrain solution of this problem minus 0.2 and 4.4. So, this is the optimal solution and function value is minus 8.67, ok. So, this is the function value. So, we will apply DE and try to get this constrained optimal solution of this problem.

(Refer Slide Time: 04:04)



DE Tutorial

Problem No 3

Minimize $f(x) = (x_1 - 3)^2 + (x_2 - 3)^2$

Subject to $2x_1 + x_2 \leq 2$

Range $-5 \leq x_1 \leq 5$ and $-5 \leq x_2 \leq 5$

The minimum point of the problem is

$x^* = (0.2, 1.6)^T$ and $f(x^*) = 9.8$

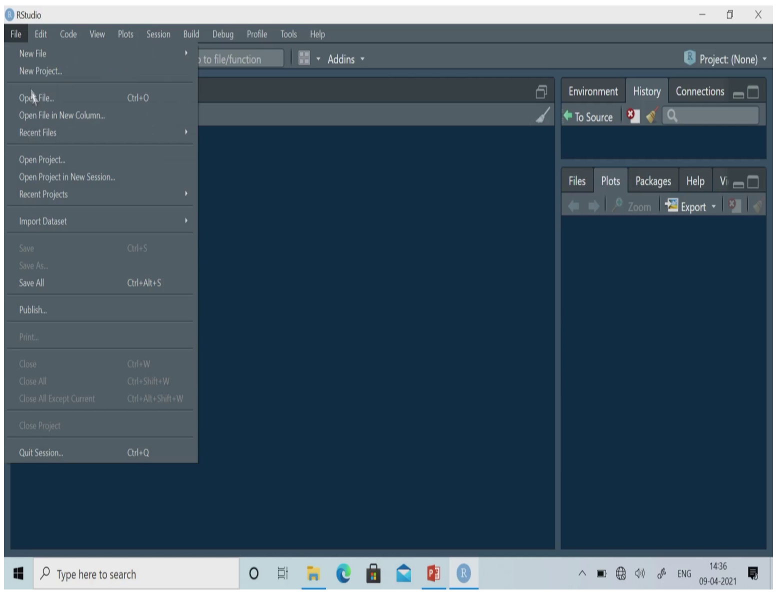
x2
x1
$2x_1+x_2 \leq 2$
$x^* = (3,3)^T$

The third problem is also a minimization problem that is x 1 minus 3 whole square plus x 2 minus 3 whole square. So, therefore the unconstrained solution is somewhere here and the solution is 3 and 3. So, this is the unconstrained solution that is x star equal to 3 comma 3 ok. So, you should get the solution, but we have a linear constrain here.

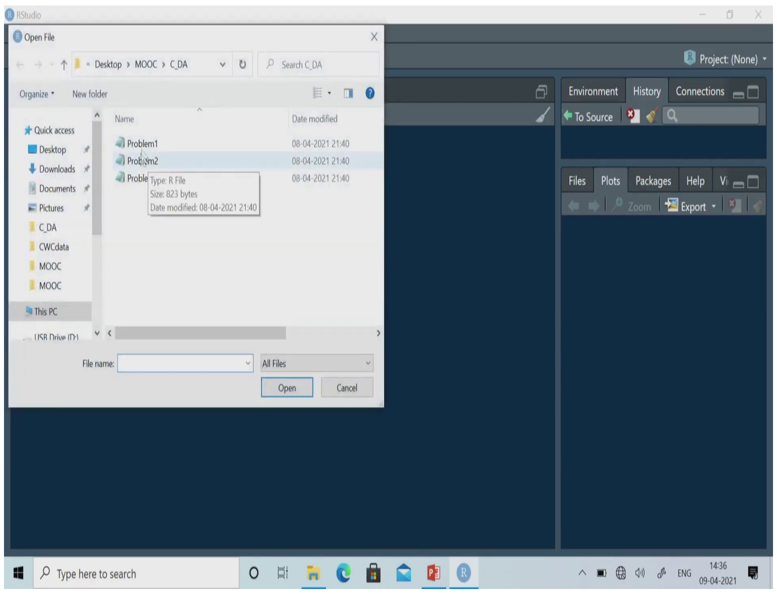
So, this is the constrain and this constrain is twice x 1 plus x 2 less than equal to 2 and therefore, the earlier solution is infeasible solution ok. Once you are putting this constrain and the constrain solution is somewhere here and this solution is minus 2 and 1.6 and function value is 9.8.

So, we will apply differential evolution on these three particular problems. So, we will try to solve these three problems using differential evolution and we will use R platform, ok. So, let us see we will try to solve this problem using R.

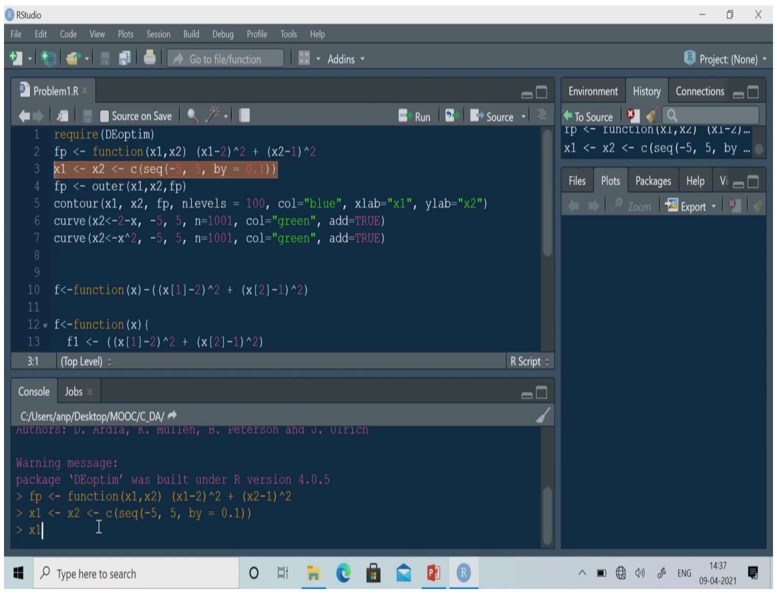
(Refer Slide Time: 05:12)



(Refer Slide Time: 05:18)



(Refer Slide Time: 05:23)



```
require(DEoptim)
fp <- function(x1,x2) (x1-2)^2 + (x2-1)^2
x1 <- x2 <- c(seq(-5, 5, by = 0.1))
fp <- outer(x1,x2,fp)
contour(x1, x2, fp, nlevels = 100, col="blue", xlab="x1", ylab="x2")
curve(x2<-2-x, -5, 5, n=1001, col="green", add=TRUE)
curve(x2<-x^2, -5, 5, n=1001, col="green", add=TRUE)


f<-function(x)-((x[1]-2)^2 + (x[2]-1)^2)

f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
```

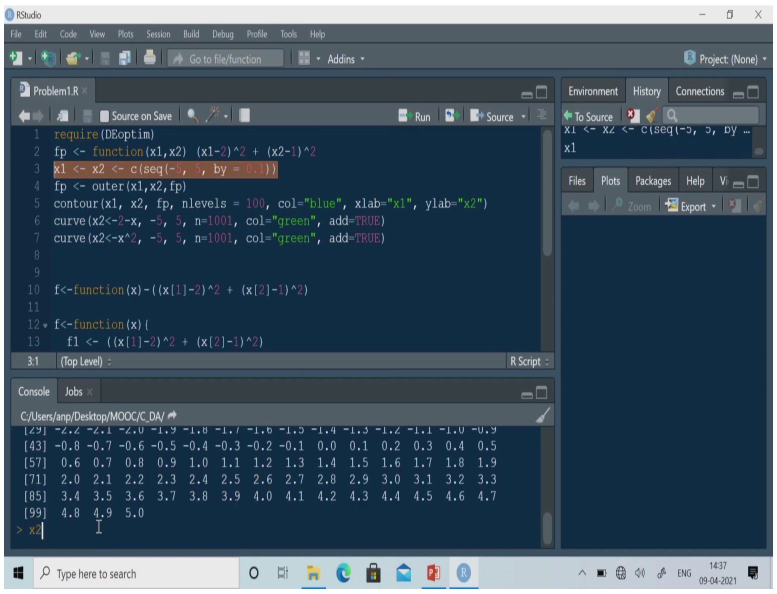
```
fp <- function(x1,x2) (x1-2)…
x1 <- x2 <- c(seq(-5, 5, by …
```

```
C:/Users/anp/Desktop/MOOC/C_DA/
Authors: D. Ardia, K. Mullen, B. Peterson and J. Ulrich

Warning message:
package 'DEoptim' was built under R version 4.0.5
> fp <- function(x1,x2) (x1-2)^2 + (x2-1)^2
> x1 <- x2 <- c(seq(-5, 5, by = 0.1))
> x1
```

So, I have opened my R studio. So, let us open the file problem 1. So, this is your problem 1 file and. So, what I need basically I need DEoptim. So, I am putting require DEoptim. So, this is the library I need this is the software package I need basically. So, I am putting that one. So, these lines I have written to plot the function, ok.

The function is that is x 1 minus 2 whole square plus x 2 minus 1 whole square. So, this is the function, this is the function of x 1 and x 2. So, let me execute this lines ok. So, I need DE and then this is the function ok and now I have to create the value for x 1 and x 2. So, x 1 is varying between minus 5 and plus 5 and I have taken a grid size of 0.1. That means, interval of 0.1. So, and I have used sequence function ok. So, you can see what is x 1 here.

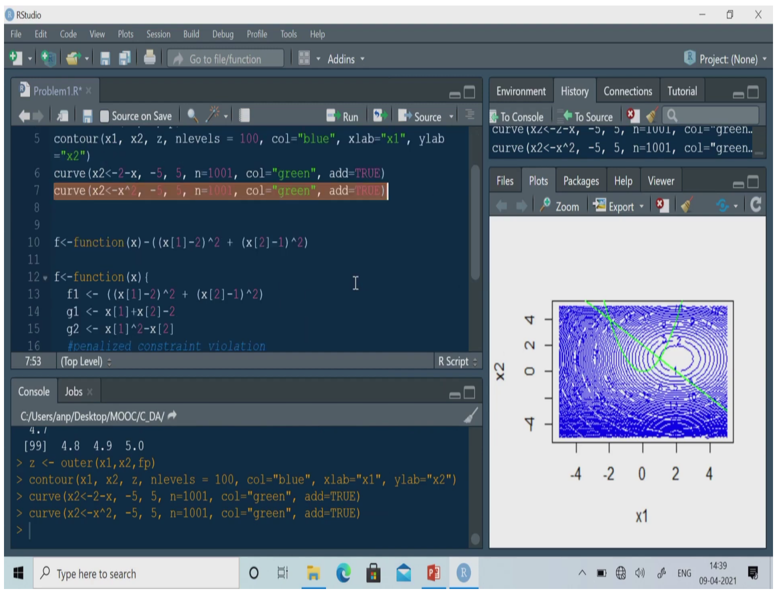
(Refer Slide Time: 06:24)



```
require(DEoptim)
fp <- function(x1,x2) (x1-2)^2 + (x2-1)^2
x1 <- x2 <- c(seq(-5, 5, by = 0.1))
fp <- outer(x1,x2,fp)
contour(x1, x2, fp, nlevels = 100, col="blue", xlab="x1", ylab="x2")
curve(x2<-2-x, -5, 5, n=1001, col="green", add=TRUE)
curve(x2<-x^2, -5, 5, n=1001, col="green", add=TRUE)


f<-function(x)-((x[1]-2)^2 + (x[2]-1)^2)

f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
```

```
[29] -2.2 -2.1 -2.0 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9
[43] -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1  0.0  0.1  0.2  0.3  0.4  0.5
[57]  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9
[71]  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3
[85]  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3  4.4  4.5  4.6  4.7
[99]  4.8  4.9  5.0
> x2
```

(Refer Slide Time: 06:36)



```
contour(x1, x2, z, nlevels = 100, col="blue", xlab="x1", ylab
="x2")
curve(x2<-2-x, -5, 5, n=1001, col="green", add=TRUE)
curve(x2<-x^2, -5, 5, n=1001, col="green", add=TRUE)


f<-function(x)-((x[1]-2)^2 + (x[2]-1)^2)

f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
  g1 <- x[1]+x[2]-2
  g2 <- x[1]^2-x[2]
  #penalized constraint violation
```

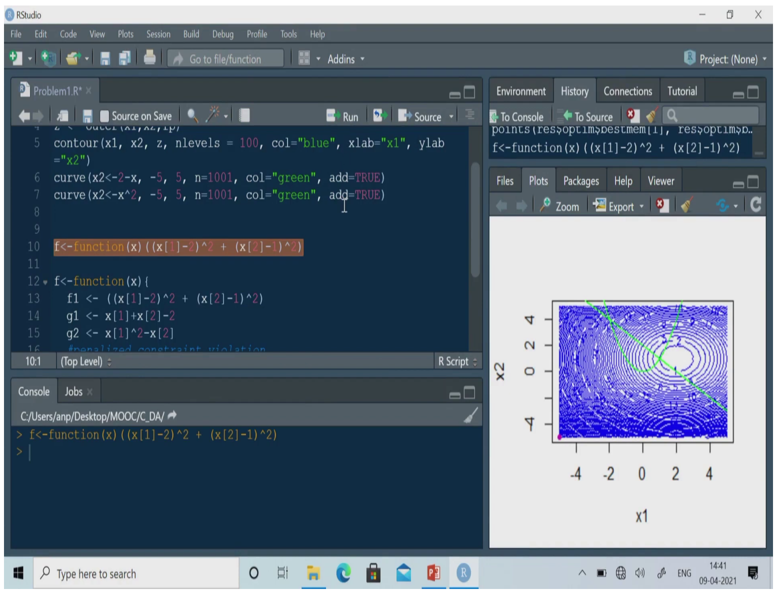
```
curve(x2<-2-x, -5, 5, n=1001, col="green…
curve(x2<-x^2, -5, 5, n=1001, col="green…
```

```
 4.7
[99]  4.8  4.9  5.0
> z <- outer(x1,x2,fp)
> contour(x1, x2, z, nlevels = 100, col="blue", xlab="x1", ylab="x2")
> curve(x2<-2-x, -5, 5, n=1001, col="green", add=TRUE)
> curve(x2<-x^2, -5, 5, n=1001, col="green", add=TRUE)
> 
```

So, this is your x 1 and this is your x 2. So, both are from minus 5 to plus 5 and then I have calculated the z value. So, this is the z value. So, I have used a outer function here and so, outer x 1 x 2 and fp. So, this is the z value I am calculating and it is grid point so, this is the z and then let us plot the contour.

So, contour you can plot. So, this is now z. So, contour I have written x 1 x 2 z, then number of contours are 100. I put color blue, then level I am putting x 1 and x 2. So, if I executed this particular line I should get the contour, ok. So, this is the contour I am getting and then let me plot the constrain. So, by this line this is constrain 1, ok. So, you are getting the constrain 1 and similarly you are getting the constrain 2, ok. So, optimal solution is somewhere here. So, I have plotted this particular function.

(Refer Slide Time: 07:42)



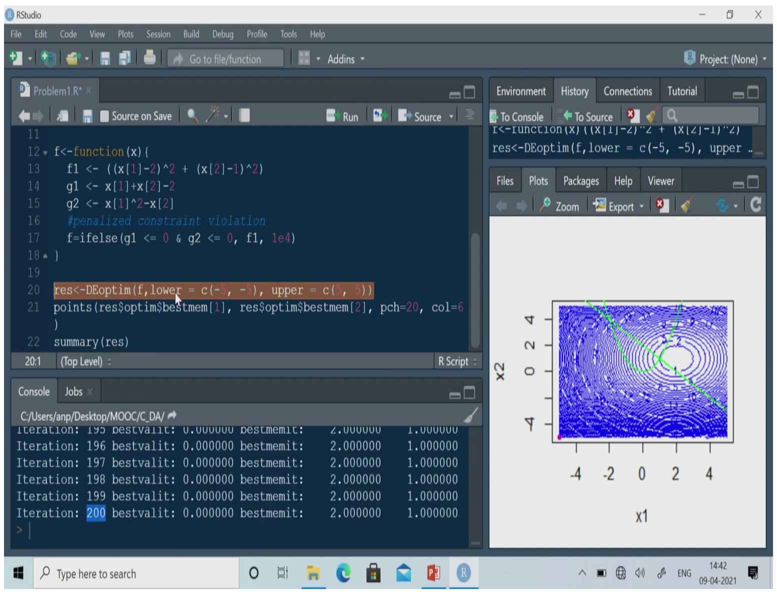
```
contour(x1, x2, z, nlevels = 100, col="blue", xlab="x1", ylab
="x2")
curve(x2<-2-x, -5, 5, n=1001, col="green", add=TRUE)
curve(x2<-x^2, -5, 5, n=1001, col="green", add=TRUE)


f<-function(x) ((x[1]-2)^2 + (x[2]-1)^2)

f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
  g1 <- x[1]+x[2]-2
  g2 <- x[1]^2-x[2]
```

```
C:/Users/anp/Desktop/MOOC/C_DA/
> f<-function(x)((x[1]-2)^2 + (x[2]-1)^2)
>
```

```
points(res$optim$bestmem[1], res$optim$b..
f<-function(x)((x[1]-2)^2 + (x[2]-1)^2)
```

So, let me get the unconstrained solution of this particular problem. So, this is the unconstrained function. So, let me execute this one and so, this is the constrained function. So, I will discuss this later on, but let us solve the unconstrained problem. So, this is the function for solving the problem.

(Refer Slide Time: 08:04)



```r
f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
  g1 <- x[1]+x[2]-2
  g2 <- x[1]^2-x[2]
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

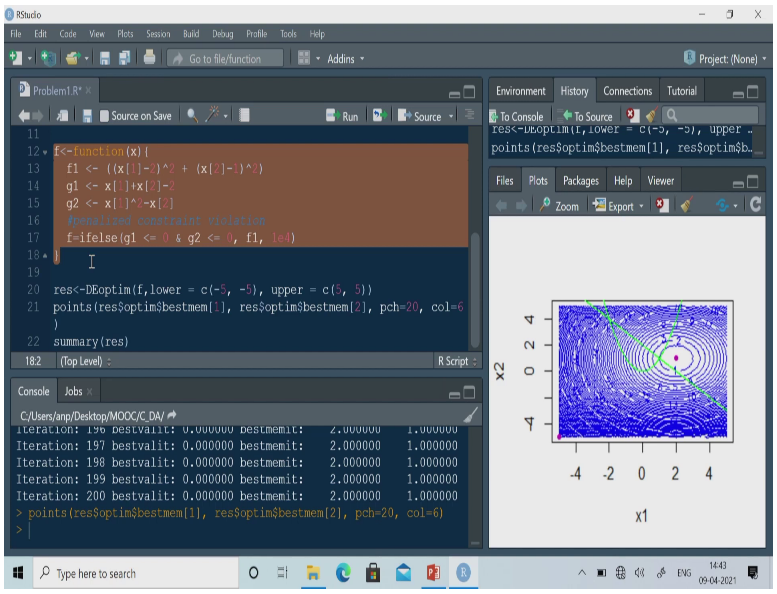
```
f<-function(x)((x[1]-2)^2 + (x[2]-1)^2)
res<-DEoptim(f,lower = c(-5, -5), upper ..
```

```
C:/Users/anp/Desktop/MOOC/C_DA/
Iteration: 195 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 196 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 197 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 198 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 199 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 200 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
>
```

So, here you have to; you have to define what is function. So, already I have defined. So, f is this f, I have executed. So, therefore this is the unconstrained function. Lower bound is minus 5 and minus 5 upper bound is 5 and 5 and I have not defined the other parameters. That means I have used the default parameters, ok.

So, let me execute this particular line then ok. So, I have executed up to 200 iteration that is the default value of iteration. And now let us plot the optimal point. So, optimal solution is somewhere here. So, this is the unconstrained solution of this particular problem.

(Refer Slide Time: 08:44)



```r
f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
  g1 <- x[1]+x[2]-2
  g2 <- x[1]^2-x[2]
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

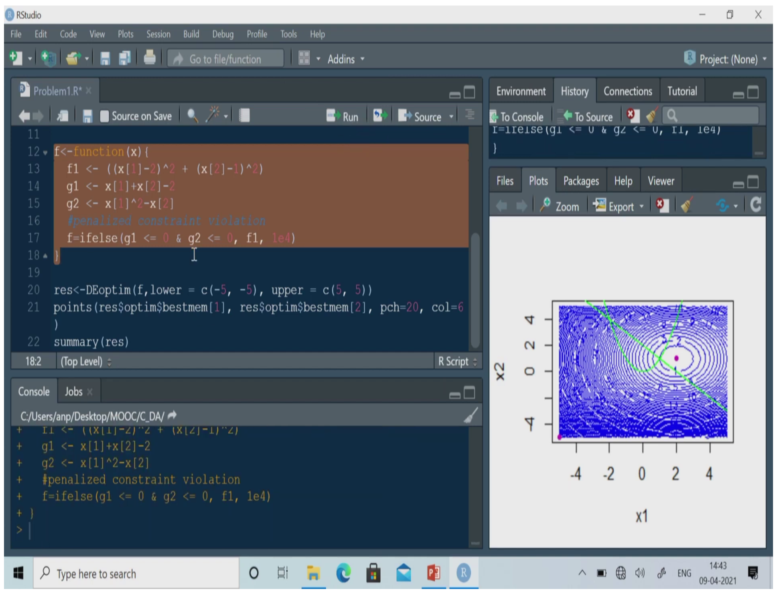
```
Iteration: 196 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 197 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 198 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 199 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
Iteration: 200 bestvalit: 0.000000 bestmemit:    2.000000    1.000000
> points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6)
>
```

```
res<-DEoptim(f,lower = c(-5, -5), upper ..
points(res$optim$bestmem[1], res$optim$b..
```

Now, let us define the constrain here. So, this is the constrain function. So, what I am doing here, this is the unconstrained function f 1 and this is the g 1. So, g 1 is x 1 plus x 2 minus 2 and g 2 is x 1 square minus x 2. Now, I have used ifelse function. So, if ifelse that g 1 is less than 0 and g 2 is less than 0, that means constraints are satisfied. Then f will be your f 1. So, else I am putting a very large number. So, that means I am minimizing this function. So, therefore I am putting a very large number ok.

So, this is my constrain function. So, what it will, what this function will give? If there is no violation so, it will give you f 1 or if there is a violation, so it will f value will be a very large number. So, therefore that solution will be avoided. So, let us execute this particular function ok.

(Refer Slide Time: 09:54)

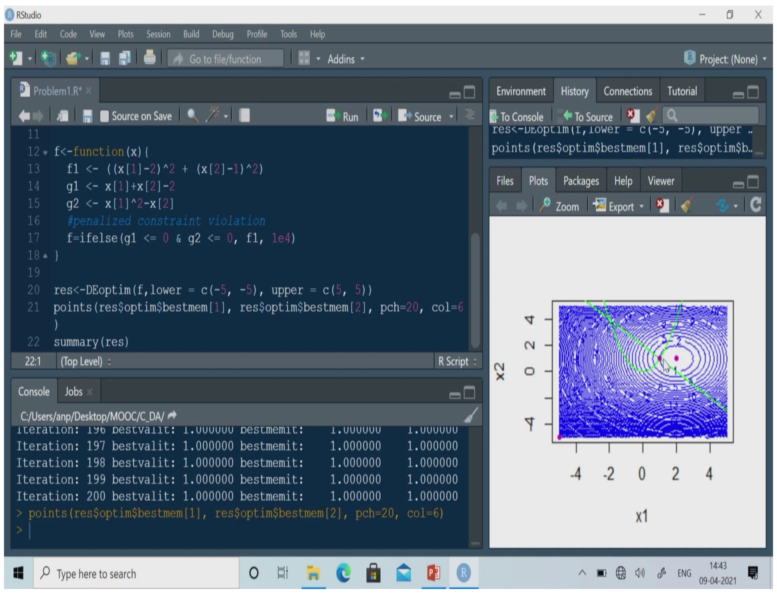


```r
f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
  g1 <- x[1]+x[2]-2
  g2 <- x[1]^2-x[2]
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
+   f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
+   g1 <- x[1]+x[2]-2
+   g2 <- x[1]^2-x[2]
+   #penalized constraint violation
+   f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
+ }
> 
```

And now I would like to rerun my differential evolution with this constrain function, ok.

(Refer Slide Time: 10:06)

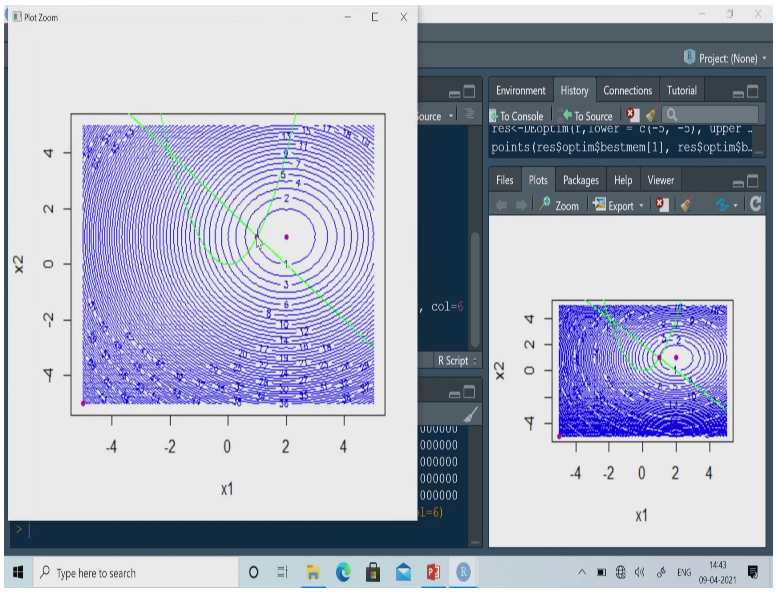


```r
f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
  g1 <- x[1]+x[2]-2
  g2 <- x[1]^2-x[2]
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
Iteration: 196 bestvalit: 1.000000 bestmemit:    1.000000    1.000000
Iteration: 197 bestvalit: 1.000000 bestmemit:    1.000000    1.000000
Iteration: 198 bestvalit: 1.000000 bestmemit:    1.000000    1.000000
Iteration: 199 bestvalit: 1.000000 bestmemit:    1.000000    1.000000
Iteration: 200 bestvalit: 1.000000 bestmemit:    1.000000    1.000000
> points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6)
>
```

So, I got the solution. So, let me plot this. So, you just see now I am getting the constrain solution.

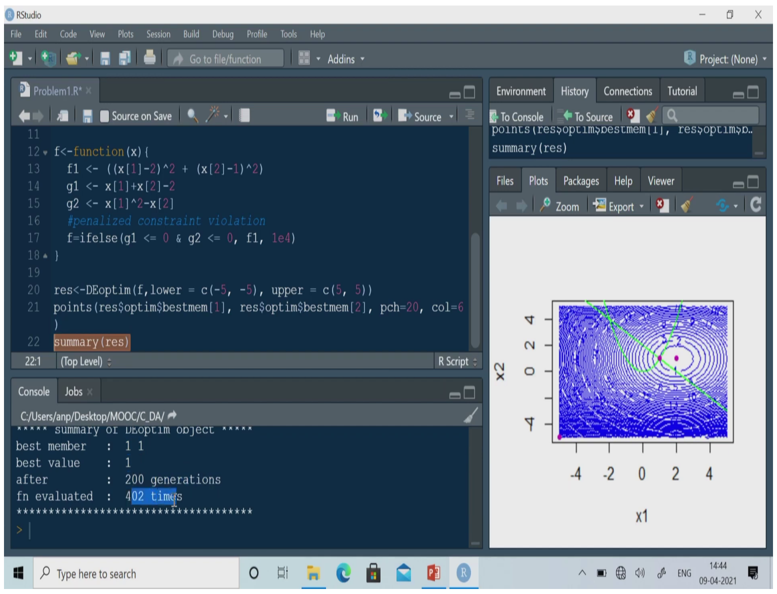
(Refer Slide Time: 10:14)



```
res<-DEoptim(f,lower = c(-5, -5), upper …
points(res$optim$bestmem[1], res$optim$b…
```

So, you can see that. So, I am getting the constrain solution of this particular problem. So, earlier solution was this that is that was the unconstrained your solution of this problem and constrained solution is here.

(Refer Slide Time: 10:30)



```r
f<-function(x){
  f1 <- ((x[1]-2)^2 + (x[2]-1)^2)
  g1 <- x[1]+x[2]-2
  g2 <- x[1]^2-x[2]
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

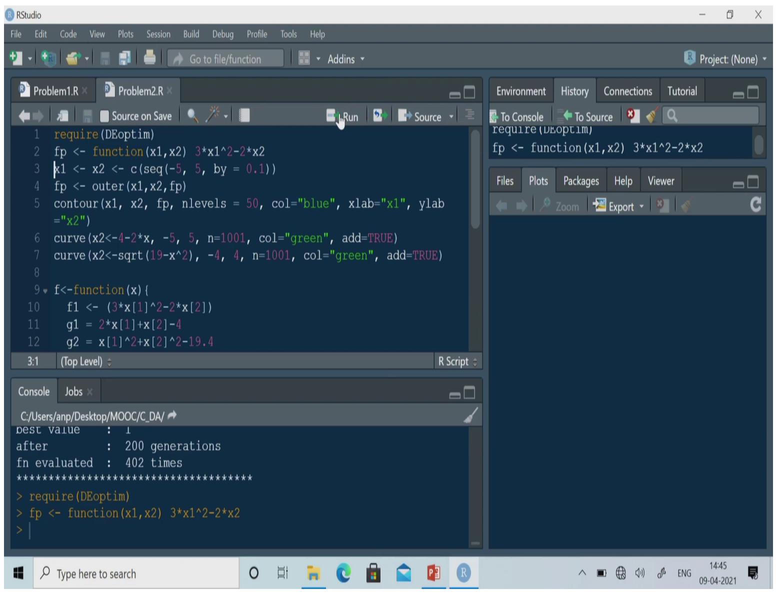
```
points(res$optim$bestmem[1], res$optim$b..
summary(res)
```

```
***** summary of DEoptim object *****
best member   :  1 1
best value    :  1
after         :  200 generations
fn evaluated  :  402 times
*************************************
>
```

So, I am getting that solution and if I would like to see, so I can see the summary of res. So, you can see that whatever solution I got this is 1 1 and function value is 1. So, anyway so generation is 200 generation and function evaluation is 402.

So, I am getting the solution of this particular problem ok. So, anyway so I can handle the constrain within this particular function. Now, let us solve the next problem. So, let me open the next file. So, that is your problem 2.

(Refer Slide Time: 11:05)



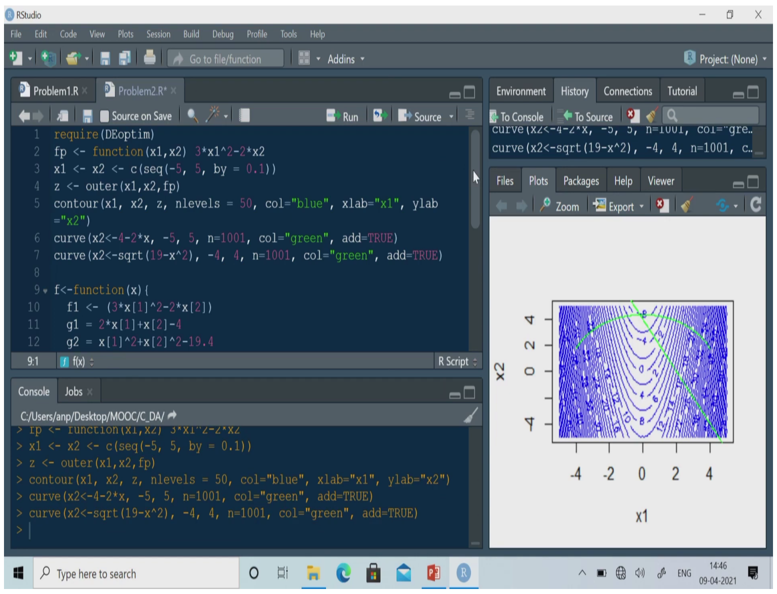
```
require(DEoptim)
fp <- function(x1,x2) 3*x1^2-2*x2
x1 <- x2 <- c(seq(-5, 5, by = 0.1))
fp <- outer(x1,x2,fp)
contour(x1, x2, fp, nlevels = 50, col="blue", xlab="x1", ylab
="x2")
curve(x2<-4-2*x, -5, 5, n=1001, col="green", add=TRUE)
curve(x2<-sqrt(19-x^2), -4, 4, n=1001, col="green", add=TRUE)

f<-function(x){
  f1 <- (3*x[1]^2-2*x[2])
  g1 = 2*x[1]+x[2]-4
  g2 = x[1]^2+x[2]^2-19.4
```

```
require(DEoptim)
fp <- function(x1,x2) 3*x1^2-2*x2
```

```
best value    :  1
after         :  200 generations
fn evaluated  :  402 times
*************************************
> require(DEoptim)
> fp <- function(x1,x2) 3*x1^2-2*x2
> 
```

So, here the objective function is 3 x 1 square minus twice x 2. So, this is the objective function. So, I required DEoptim. So, then let me run these objective function, then x 1 and x 2. So, x 1 and x 2, both are varying between minus 5 and plus 5.

(Refer Slide Time: 11:35)



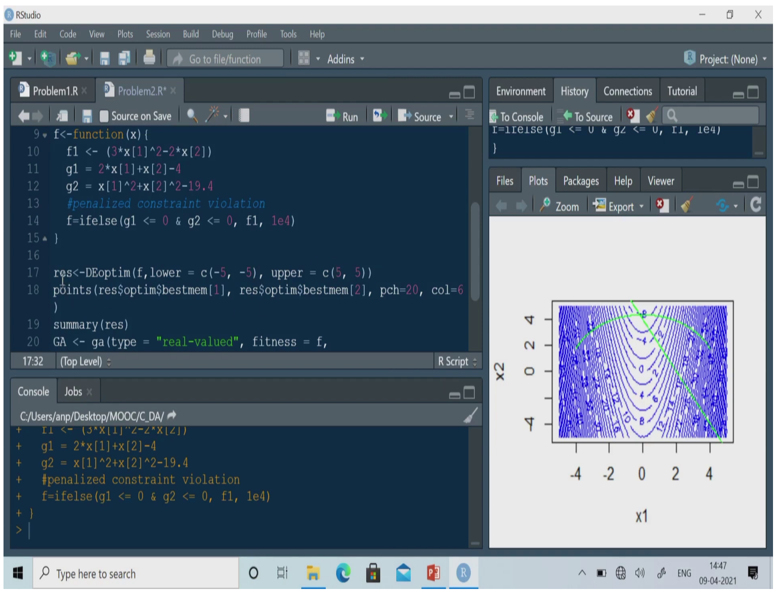
```r
require(DEoptim)
fp <- function(x1,x2) 3*x1^2-2*x2
x1 <- x2 <- c(seq(-5, 5, by = 0.1))
z <- outer(x1,x2,fp)
contour(x1, x2, z, nlevels = 50, col="blue", xlab="x1", ylab
="x2")
curve(x2<-4-2*x, -5, 5, n=1001, col="green", add=TRUE)
curve(x2<-sqrt(19-x^2), -4, 4, n=1001, col="green", add=TRUE)

f<-function(x){
  f1 <- (3*x[1]^2-2*x[2])
  g1 = 2*x[1]+x[2]-4
  g2 = x[1]^2+x[2]^2-19.4
```

```
C:/Users/anp/Desktop/MOOC/C_DA/
> fp <- function(x1,x2) 3*x1^2-2*x2
> x1 <- x2 <- c(seq(-5, 5, by = 0.1))
> z <- outer(x1,x2,fp)
> contour(x1, x2, z, nlevels = 50, col="blue", xlab="x1", ylab="x2")
> curve(x2<-4-2*x, -5, 5, n=1001, col="green", add=TRUE)
> curve(x2<-sqrt(19-x^2), -4, 4, n=1001, col="green", add=TRUE)
>
```

```
curve(x2<-4-2*x, -5, 5, n=1001, col="gre..
curve(x2<-sqrt(19-x^2), -4, 4, n=1001, c..
```

So, let me generate the value of x 1 and x 2 and then generate the value of z, ok. So, this is I am generating the value of z and then let me plot the contour map, plot the contour ok. So, if I execute this particular line, so I am getting this contour and as I said that minima of this particular function is somewhere here. So, I can plot the constrain now. So, this is the constrain, first constrain that is the linear constrain and this is the second constrain and optimal solution is somewhere here, ok.

(Refer Slide Time: 12:18)



```
f<-function(x){
  f1 <- (3*x[1]^2-2*x[2])
  g1 = 2*x[1]+x[2]-4
  g2 = x[1]^2+x[2]^2-19.4
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
GA <- ga(type = "real-valued", fitness = f,
```

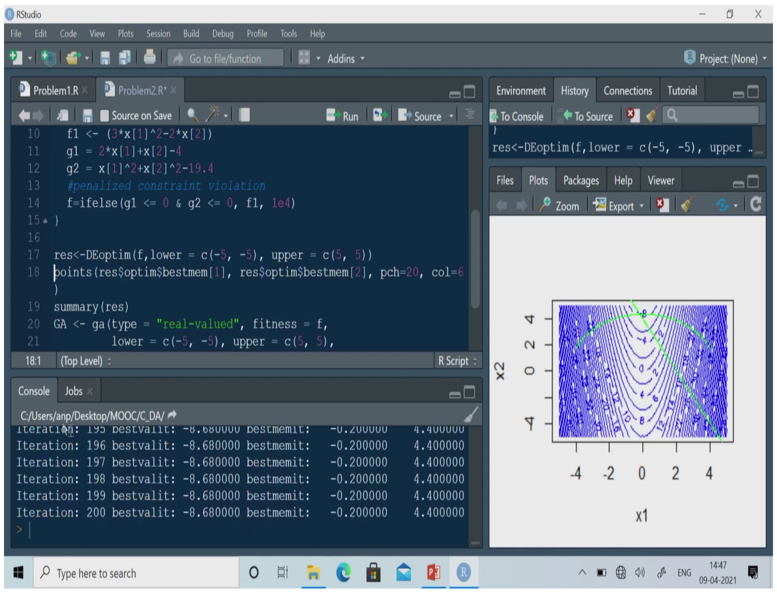
```
C:/Users/anp/Desktop/MOOC/C_DA/
+   f1 <- (3*x[1]^2-2*x[2])
+   g1 = 2*x[1]+x[2]-4
+   g2 = x[1]^2+x[2]^2-19.4
+   #penalized constraint violation
+   f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
+ }
> 
```

Now, let us define the objective function. So, objective function here is f 1. So, for solving for using DE. So, I have to define objective function in this order. That means, x has to be a vector. So, f 1 is 3 x 1 square minus twice x 2, then g 1 is twice x 1 plus x 2 minus 4. So, this is the linear constrain and z 2 is x 1 square plus x 2 square minus 19.4, ok.

Now, so here also I have used ifelse function ifelse the g 1 is less than 0 and g 2 is less than 0. So, in that case it will give f 1. So, it will take f 1. That means if constraints are satisfied, so f value will be equal to f 1. That means objective function value will be f 1 and if it is not satisfied so I am putting a very large number that is 1 e to the power 4 in that case.

So, that solution will not be selected. So, when you are going for a selection operator, so this solution will not be selected ok. So, let me execute this particular function, then here I am using the differential evolution and I use the default value only. I have defined the lower bound and upper bond. So, lower bound is minus 5, upper bound is plus 5 and plus 5.

(Refer Slide Time: 13:50)

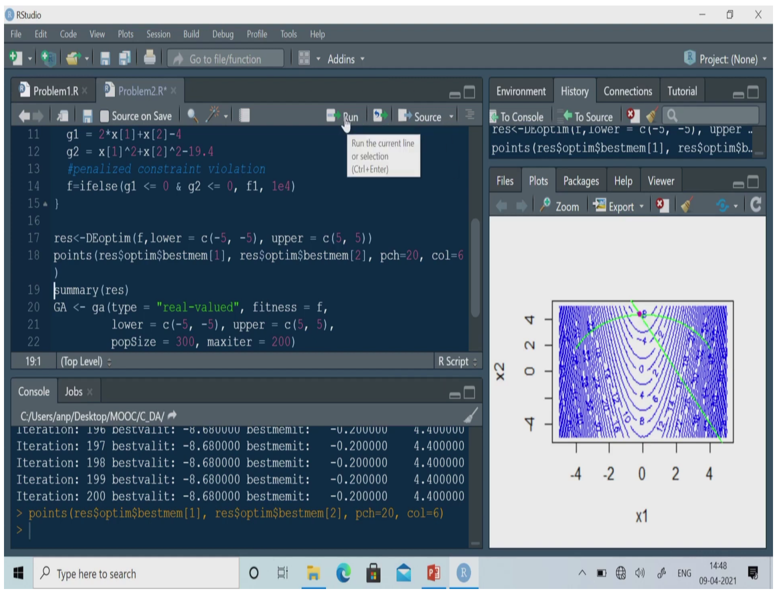


```
  f1 <- (3*x[1]^2-2*x[2])
  g1 = 2*x[1]+x[2]-4
  g2 = x[1]^2+x[2]^2-19.4
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
GA <- ga(type = "real-valued", fitness = f,
         lower = c(-5, -5), upper = c(5, 5),
```

```
Iteration: 195 bestvalit: -8.680000 bestmemit:   -0.200000   4.400000
Iteration: 196 bestvalit: -8.680000 bestmemit:   -0.200000   4.400000
Iteration: 197 bestvalit: -8.680000 bestmemit:   -0.200000   4.400000
Iteration: 198 bestvalit: -8.680000 bestmemit:   -0.200000   4.400000
Iteration: 199 bestvalit: -8.680000 bestmemit:   -0.200000   4.400000
Iteration: 200 bestvalit: -8.680000 bestmemit:   -0.200000   4.400000
>
```

```
res<-DEoptim(f,lower = c(-5, -5), upper ...
```

(Refer Slide Time: 13:58)



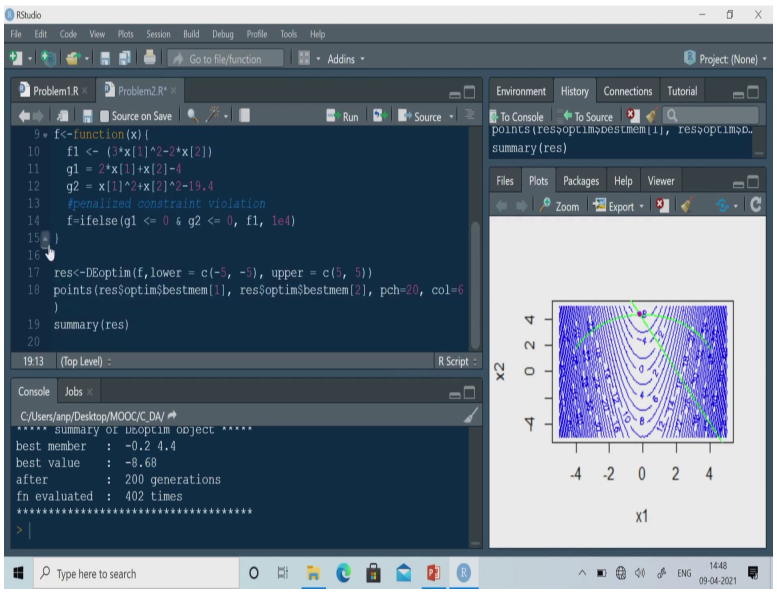
```r
  g1 = 2*x[1]+x[2]-4
  g2 = x[1]^2+x[2]^2-19.4
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
GA <- ga(type = "real-valued", fitness = f,
         lower = c(-5, -5), upper = c(5, 5),
         popSize = 300, maxiter = 200)
```

```
Iteration: 196 bestvalit: -8.680000 bestmemit:   -0.200000    4.400000
Iteration: 197 bestvalit: -8.680000 bestmemit:   -0.200000    4.400000
Iteration: 198 bestvalit: -8.680000 bestmemit:   -0.200000    4.400000
Iteration: 199 bestvalit: -8.680000 bestmemit:   -0.200000    4.400000
Iteration: 200 bestvalit: -8.680000 bestmemit:   -0.200000    4.400000
> points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6)
>
```

RStudio
Problem1.R
Problem2.R*
Run the current line or selection (Ctrl+Enter)
Console
C:/Users/anp/Desktop/MOOC/C_DA/
History
x2
x1

So, let me execute this one ok, then I am getting the solution. So, let me plot this solution ok. So, I am getting this particular solution.

(Refer Slide Time: 14:03)



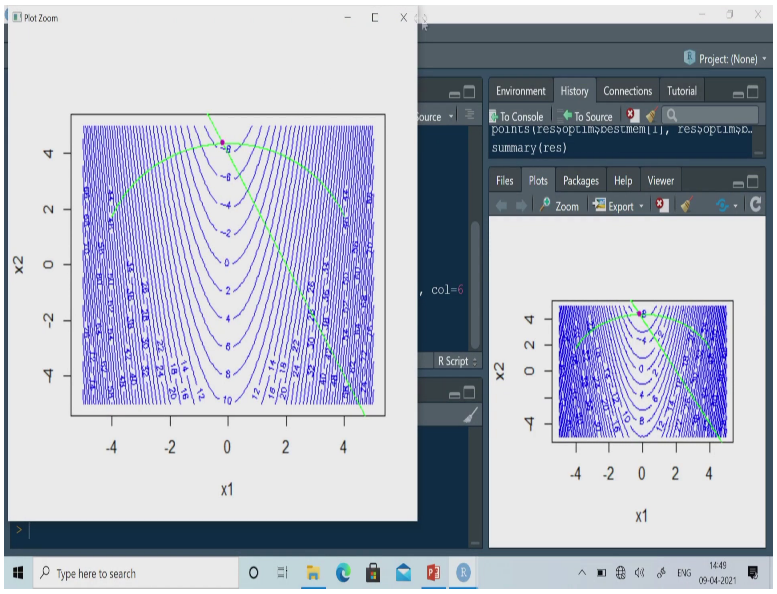
```r
f<-function(x){
  f1 <- (3*x[1]^2-2*x[2])
  g1 = 2*x[1]+x[2]-4
  g2 = x[1]^2+x[2]^2-19.4
  #penalized constraint violation
  f=ifelse(g1 <= 0 & g2 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
***** summary of DEoptim object *****
best member   :  -0.2 4.4
best value    :  -8.68
after         :  200 generations
fn evaluated  :  402 times
*************************************
>
```

And I can see the summary the best solution is minus 2 and 4.4. So, this is the solution and objective function value is minus 8.67. So, I got the exact solution of this particular problem and you can see that DE is quite effective in solving constrain optimization problem also, ok.

So, you will get the constrain optimal solution of the problem and it is quite efficient. So, I can also solve using GA. So, if you want whether you are getting the in that case you have to include the GA library ok. So, I am not doing that one ok.

(Refer Slide Time: 14:48)

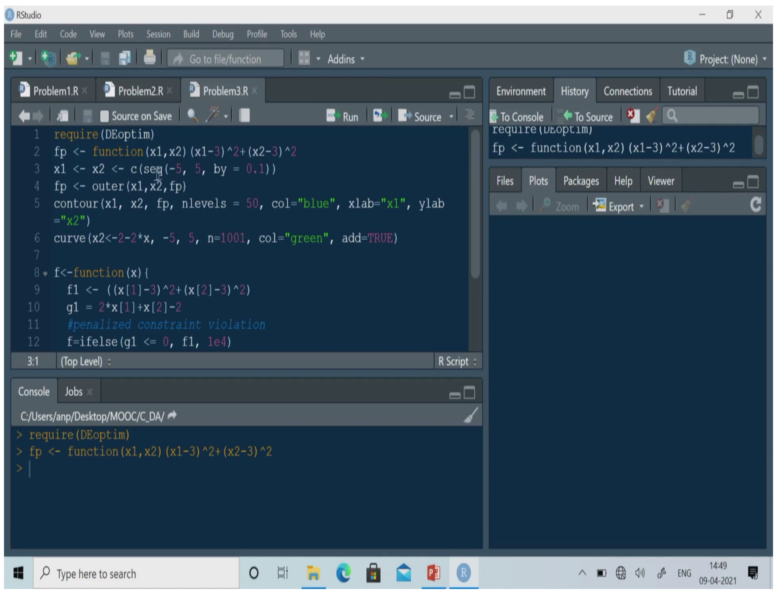


```
points(res$optim$bestmem[1], res$optim$b…
summary(res)
```

```
, col=6
```

So, I am getting the solution of this second problem. So, you can see that I got this solution ok. Solution is on the constrain boundary. Now let me solve the third problem.

(Refer Slide Time: 15:01)



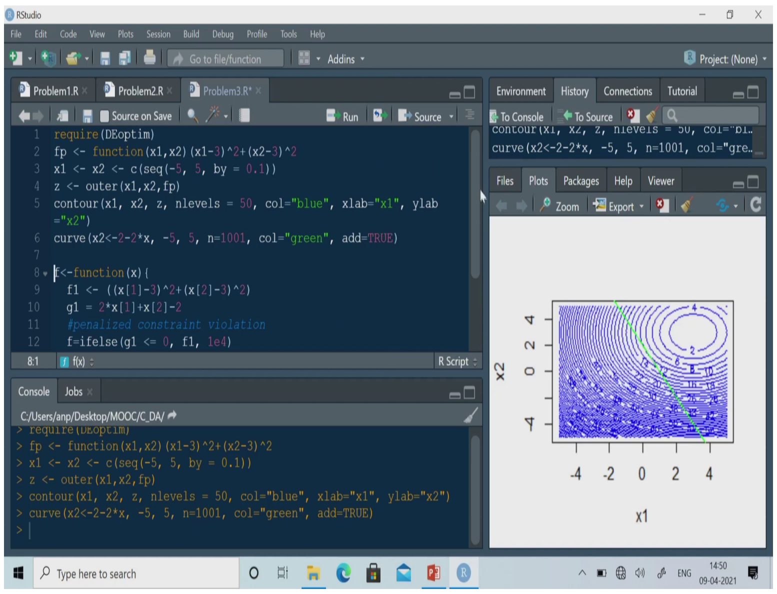
```
require(DEoptim)
fp <- function(x1,x2) (x1-3)^2+(x2-3)^2
x1 <- x2 <- c(seq(-5, 5, by = 0.1))
fp <- outer(x1,x2,fp)
contour(x1, x2, fp, nlevels = 50, col="blue", xlab="x1", ylab
="x2")
curve(x2<-2-2*x, -5, 5, n=1001, col="green", add=TRUE)

f<-function(x){
  f1 <- ((x[1]-3)^2+(x[2]-3)^2)
  g1 = 2*x[1]+x[2]-2
  #penalized constraint violation
  f=ifelse(g1 <= 0, f1, 1e4)
```

```
C:/Users/anp/Desktop/MOOC/C_DA/
> require(DEoptim)
> fp <- function(x1,x2) (x1-3)^2+(x2-3)^2
>
```

```
require(DEoptim)
fp <- function(x1,x2) (x1-3)^2+(x2-3)^2
```

So, this is the third problem. So, here again I have plot this particular functions. So, let me delete these ok let me clear all the space. So, now this is the objective function. Objective function is x 1 minus 3 whole square plus x 2 minus 3 whole square. So, this is the objective function and I am defining x 1 and x 2 and x 1 and x 2 are varying between minus 5 and plus 5. So, let me generate x 1 and x 2 values ok.

(Refer Slide Time: 15:41)

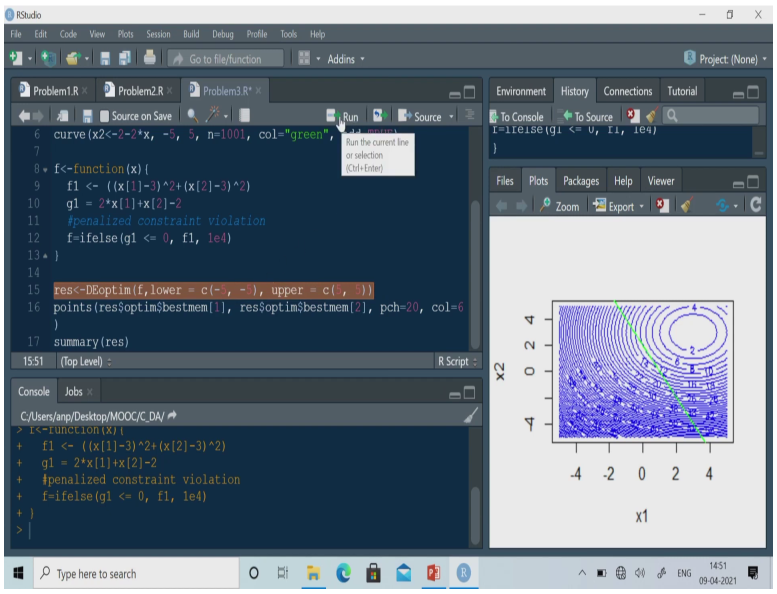


```r
require(DEoptim)
fp <- function(x1,x2) (x1-3)^2+(x2-3)^2
x1 <- x2 <- c(seq(-5, 5, by = 0.1))
z <- outer(x1,x2,fp)
contour(x1, x2, z, nlevels = 50, col="blue", xlab="x1", ylab
="x2")
curve(x2<-2-2*x, -5, 5, n=1001, col="green", add=TRUE)

f<-function(x){
  f1 <- ((x[1]-3)^2+(x[2]-3)^2)
  g1 = 2*x[1]+x[2]-2
  #penalized constraint violation
  f=ifelse(g1 <= 0, f1, 1e4)
```

```
> require(DEoptim)
> fp <- function(x1,x2) (x1-3)^2+(x2-3)^2
> x1 <- x2 <- c(seq(-5, 5, by = 0.1))
> z <- outer(x1,x2,fp)
> contour(x1, x2, z, nlevels = 50, col="blue", xlab="x1", ylab="x2")
> curve(x2<-2-2*x, -5, 5, n=1001, col="green", add=TRUE)
>
```

Then calculate the z values. So, I have to use outer function, ok. Now plot the contour. So, this is the function and optimal solution unconstrained solution is somewhere here. Let me plot the constrain function. So, there is only one constrain function and that is a linear constrain. So, I have plotted this one. So, optimal solution is somewhere here at this particular point.

(Refer Slide Time: 16:18)



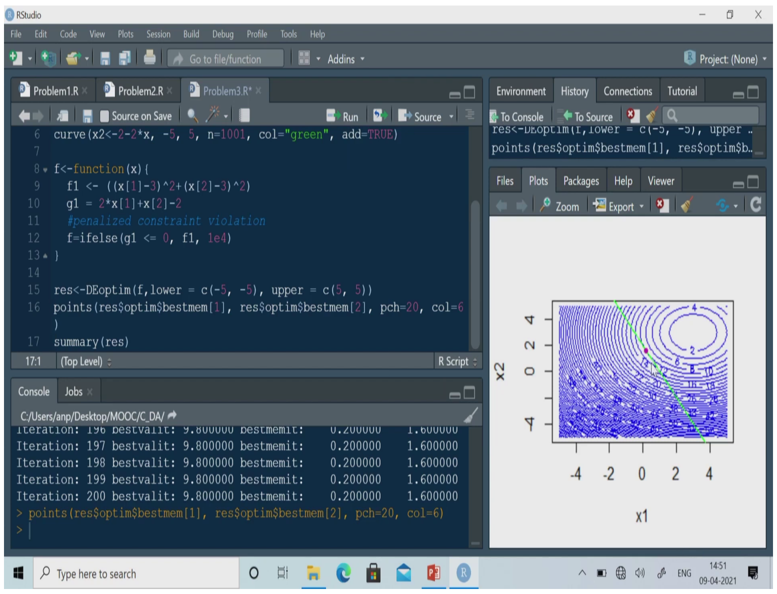
```
curve(x2<-2-2*x, -5, 5, n=1001, col="green",

f<-function(x){
  f1 <- ((x[1]-3)^2+(x[2]-3)^2)
  g1 = 2*x[1]+x[2]-2
  #penalized constraint violation
  f=ifelse(g1 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
> f<-function(x){
+   f1 <- ((x[1]-3)^2+(x[2]-3)^2)
+   g1 = 2*x[1]+x[2]-2
+   #penalized constraint violation
+   f=ifelse(g1 <= 0, f1, 1e4)
+ }
>
```

RStudio
Problem1.R
Problem2.R
Problem3.R*
Run
Source
Run the current line or selection (Ctrl+Enter)
Console
Jobs
C:/Users/anp/Desktop/MOOC/C_DA/
x2
x1
-4
-2
0
2
4

Now, let me define the constrain function. So, this is the objective function and the constrain is g 1 equal to twice x 1 plus x 2 minus 2. So, this is the constrain and here also I have used ifelse function. So, if else that means if there is no violation say then this function will return f 1. That means, I will take f 1 and if there is a violation, so in that case I will take a big number just to avoid this particular solution. So, you execute this particular function. So, I have executed that one. Now, I am using DEoptim, ok. So, again I have used the default values ok. It is done.

(Refer Slide Time: 17:12)

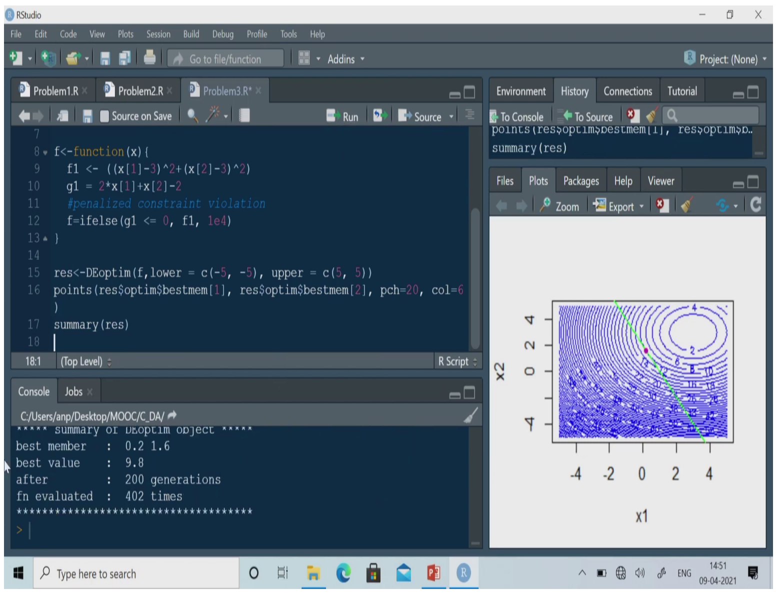


```r
curve(x2<-2-2*x, -5, 5, n=1001, col="green", add=TRUE)

f<-function(x){
  f1 <- ((x[1]-3)^2+(x[2]-3)^2)
  g1 = 2*x[1]+x[2]-2
  #penalized constraint violation
  f=ifelse(g1 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
Iteration: 196 bestvalit: 9.800000 bestmemit:    0.200000    1.600000
Iteration: 197 bestvalit: 9.800000 bestmemit:    0.200000    1.600000
Iteration: 198 bestvalit: 9.800000 bestmemit:    0.200000    1.600000
Iteration: 199 bestvalit: 9.800000 bestmemit:    0.200000    1.600000
Iteration: 200 bestvalit: 9.800000 bestmemit:    0.200000    1.600000
> points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6)
> 
```

```
res<-DEoptim(f,lower = c(-5, -5), upper …
points(res$optim$bestmem[1], res$optim$b…
```

(Refer Slide Time: 17:20)

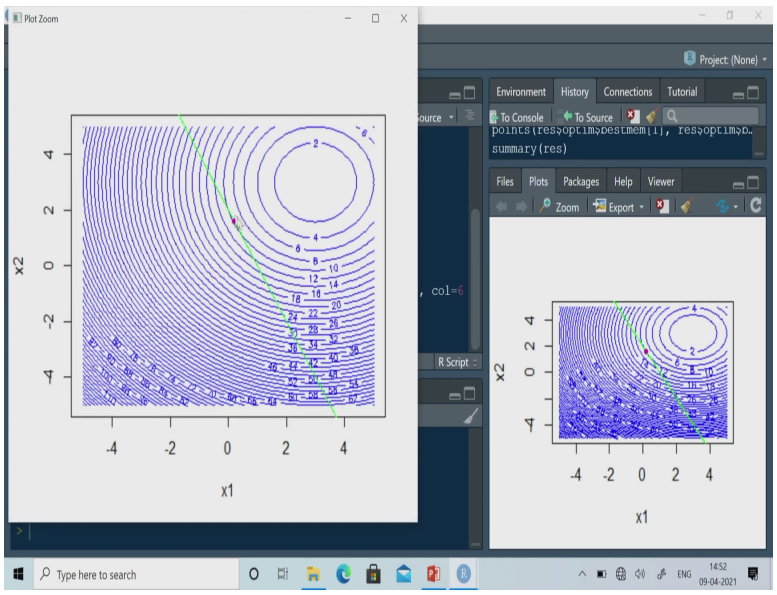


```
f<-function(x){
  f1 <- ((x[1]-3)^2+(x[2]-3)^2)
  g1 = 2*x[1]+x[2]-2
  #penalized constraint violation
  f=ifelse(g1 <= 0, f1, 1e4)
}

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
***** summary of DEoptim object *****
best member   :  0.2 1.6
best value    :  9.8
after         :  200 generations
fn evaluated  :  402 times
*************************************
>
```

So, let me plot the points optimal point ok. So, this is the solution of this particular problem. So, I can see the summary of the solution. So, best solution is 0.2 and 1.6. That means, this solution is x equal to 0.2 and x 1 equal to 0.2 and x 2 equal to 1.6 and the function value is 9.8 ok. The function value is 9.8 and generation was 200 generation and function evaluation was done for 402 times, ok.

(Refer Slide Time: 17:50)

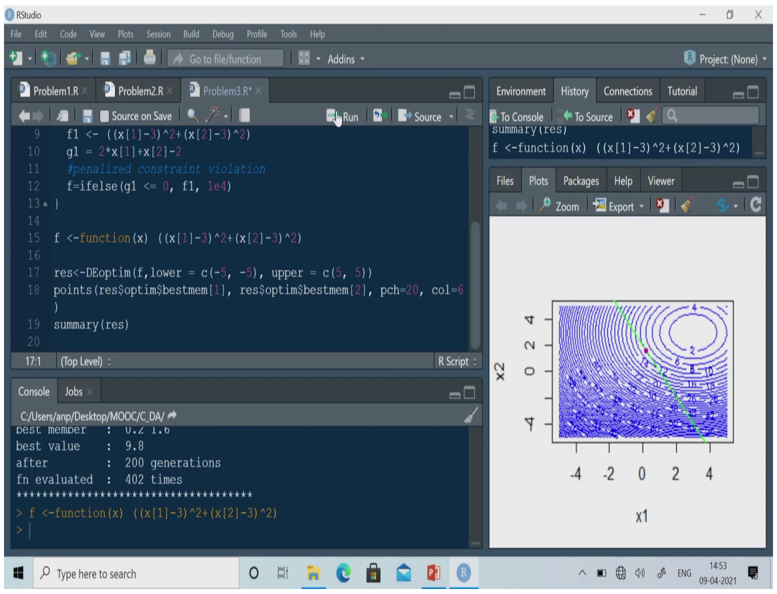


```
points(res$optim$bestmem[1], res$optim$b…
summary(res)
```

```
, col=6
```

So, this is the solution of this particular problem. So, you can see this is the constrain solution of this problem, but if you are solving unconstrained so, but if you are not considering this constrain, so you should get this particular solution that also you can check. So, I will only use this particular function ok.

(Refer Slide Time: 18:09)

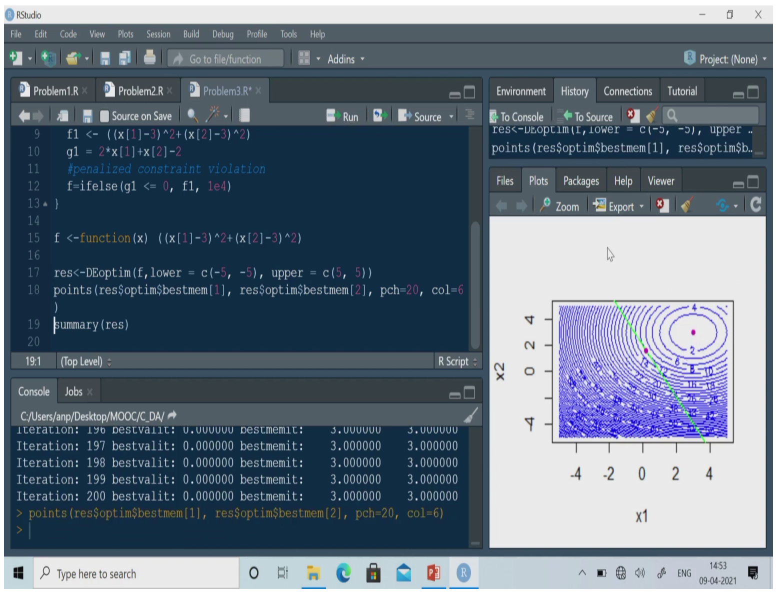


```
  f1 <- ((x[1]-3)^2+(x[2]-3)^2)
  g1 = 2*x[1]+x[2]-2
  #penalized constraint violation
  f=ifelse(g1 <= 0, f1, 1e4)
}

f <-function(x) ((x[1]-3)^2+(x[2]-3)^2)

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
best member   :  0.2 1.6
best value    :  9.8
after         :  200 generations
fn evaluated  :  402 times
*************************************
> f <-function(x) ((x[1]-3)^2+(x[2]-3)^2)
>
```

```
summary(res)
f <-function(x) ((x[1]-3)^2+(x[2]-3)^2)
```

(Refer Slide Time: 18:33)



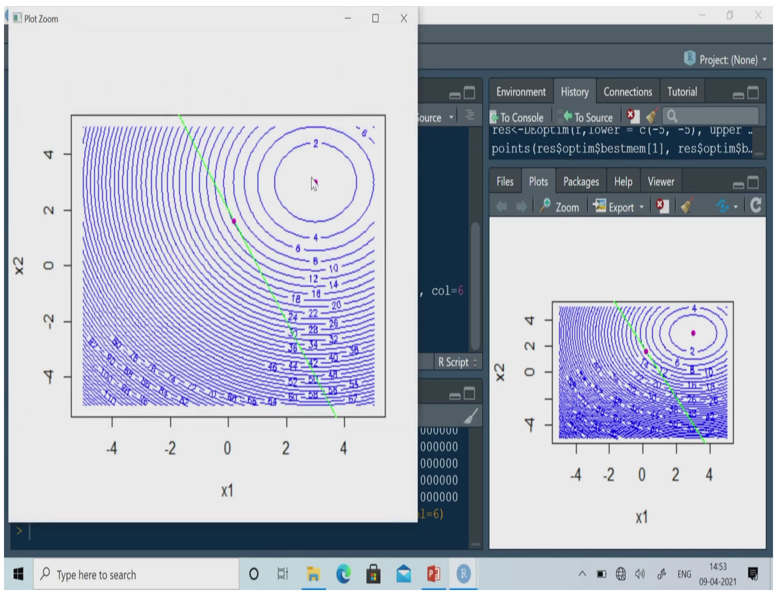
```
  f1 <- ((x[1]-3)^2+(x[2]-3)^2)
  g1 = 2*x[1]+x[2]-2
  #penalized constraint violation
  f=ifelse(g1 <= 0, f1, 1e4)
}

f <-function(x) ((x[1]-3)^2+(x[2]-3)^2)

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
Iteration: 196 bestvalit: 0.000000 bestmemit:    3.000000    3.000000
Iteration: 197 bestvalit: 0.000000 bestmemit:    3.000000    3.000000
Iteration: 198 bestvalit: 0.000000 bestmemit:    3.000000    3.000000
Iteration: 199 bestvalit: 0.000000 bestmemit:    3.000000    3.000000
Iteration: 200 bestvalit: 0.000000 bestmemit:    3.000000    3.000000
> points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6)
>
```

```
res<-DEoptim(f,lower = c(-5, -5), upper ..
points(res$optim$bestmem[1], res$optim$b..
```

So, let me check this one. So, this is function x ok. So, let me set this one. So, if I execute this one, then I will execute the DEoptim and then let us plot this one. So, you just see I am getting now I am getting this particular solution, ok.

(Refer Slide Time: 18:42)



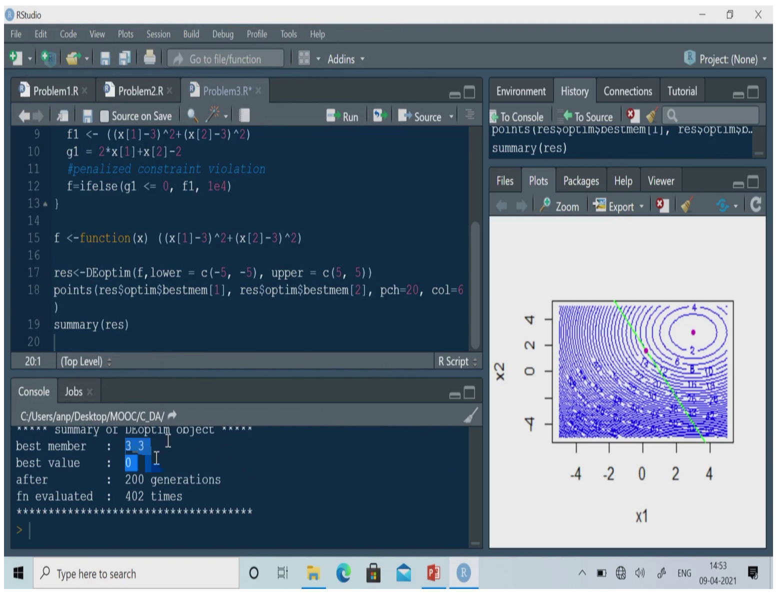
```
res<-DEoptim(f,lower = c(-5, -5), upper …
points(res$optim$bestmem[1], res$optim$b…
```

```
, col=6
```

```
000000
000000
000000
000000
000000
l=6)
```

So, this is the unconstrained solution of this particular problem and if you are putting the constrain, you should get this particular solution. You can see the summary.

(Refer Slide Time: 18:55)



```
  f1 <- ((x[1]-3)^2+(x[2]-3)^2)
  g1 = 2*x[1]+x[2]-2
  #penalized constraint violation
  f=ifelse(g1 <= 0, f1, 1e4)
}

f <-function(x) ((x[1]-3)^2+(x[2]-3)^2)

res<-DEoptim(f,lower = c(-5, -5), upper = c(5, 5))
points(res$optim$bestmem[1], res$optim$bestmem[2], pch=20, col=6
)
summary(res)
```

```
***** summary of DEoptim object *****
best member   :  3 3
best value    :  0
after         :  200 generations
fn evaluated  :  402 times
*************************************
>
```

```
points(res$optim$bestmem[1], res$optim$b..
summary(res)
```

So, now solution is 3 3 and function value is 0. So, that is this is 3 3 and function value is 0. So, in this class we have solve three problems and basically we have solve the constrained optimization problem. So, you can apply the same DE. So, DE usually you cannot define constrain separately, but in other cases in some other algorithms. So, you can define the constrain separately, but in this case you cannot define it.

So, this particular package can handle the unconstrained function. So, therefore you have to make your function and convert and you have to make your function and convert the constrained optimization problem to an unconstrained optimization problem.

So, in this class we have converted it something like that if there is a violation, we are putting a very large number because the problem is a minimization type problem and I would like to avoid that solution. So, if there is a violation, so I am putting a very large number in order to avoid that solution. If there is no violation, so in that case it will consider that particular function value ok. So, let us stop here. So, in the next class we will discuss some other problems.

Thank you.