Computation Stats Lab 6 Report - Group6

Jaskirat S Marar (jasma356), Filip Berndtsson (filbe354),
Dinuke Jayaweera (dinja628), Raja Uzair Saeed (rajsa233)

12/7/2021

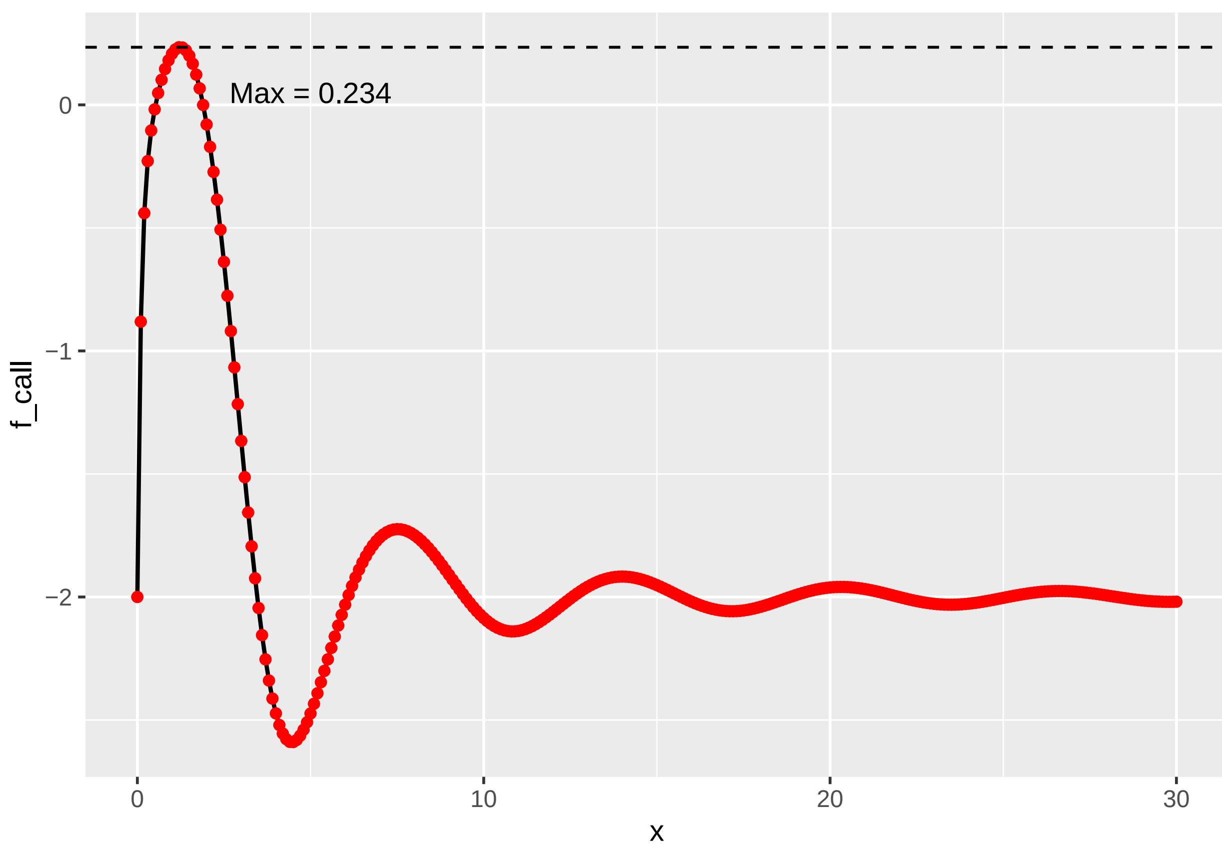
Statement of Contribution

This lab work was divided among group members as follows:

1. Assignment1: Filip Berndtsson, Jaskirat Marar, Uzair Saeed
2. Assignment2: Dinuke Jayaweera, Jaskirat Marar

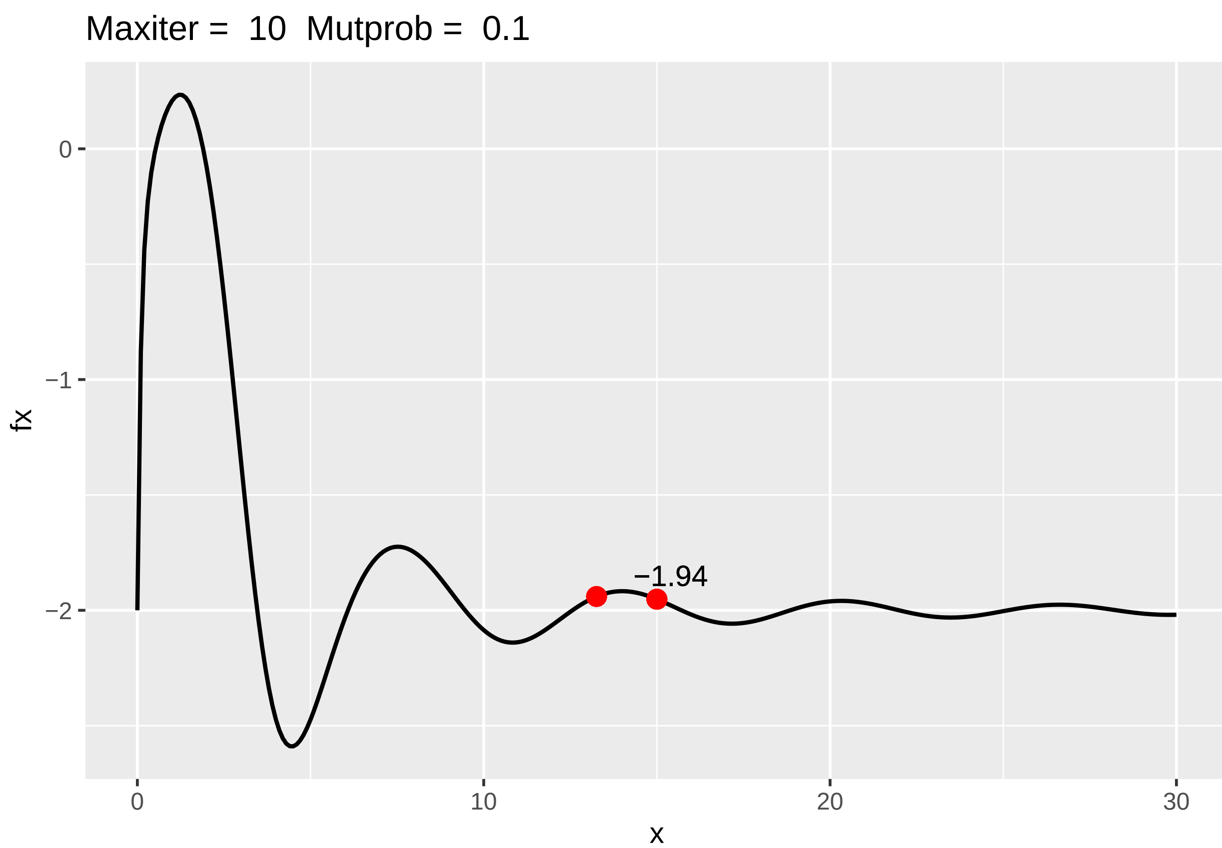
Assignment 1 - Genetic Algorithm

```r
# writing the objective function
fx <- function(x) (x^2) / (exp(x)) - 2 * exp(-(9*sin(x))/(x^2+x+1))

# Crossover function to 'birth' kid from 'parents'
crossover <- function(x, y) (x+y)/2

# mutate function to mutate 'kid' based on probability
mutate <- function(x) x^2 %% 30

# Initial plot to identify max of objective function
x <- seq(0,30, by = 0.1)
f_call <- fx(x)
f_max = round(max(f_call),3)
x_max = x[which.max(f_call)]
plot_df <- data.frame(x = x, fx = f_call)
ggplot(plot_df, aes(x, f_call)) +
  geom_line(size = 0.75, col = 'black') +
  geom_point(col = 'red') +
  geom_hline(yintercept = max(f_call), linetype = 2) +
  annotate(geom = "text", x = 5, y = 0.05, label = "Max = 0.234")
```

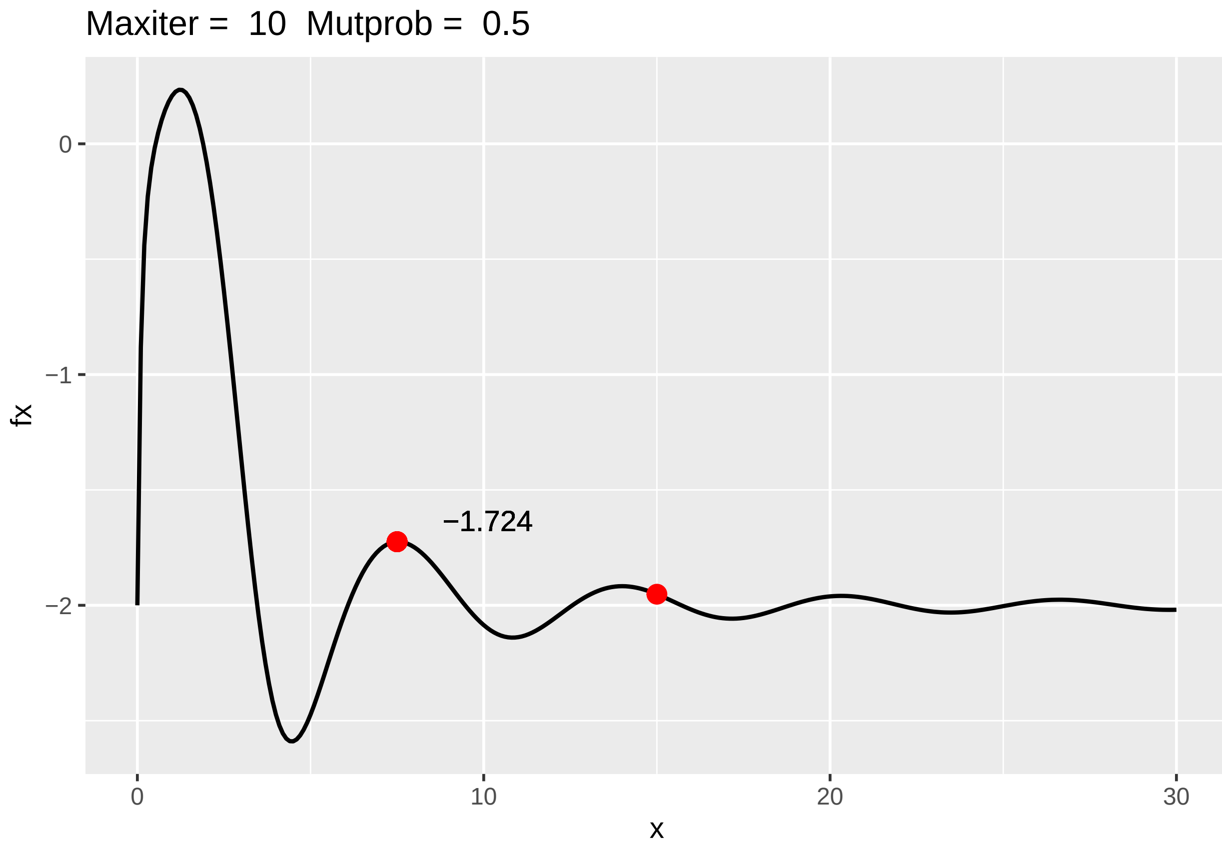



```r
# Genetic Algorithm function
GA_fn <- function(maxiter, mutprob) {
```

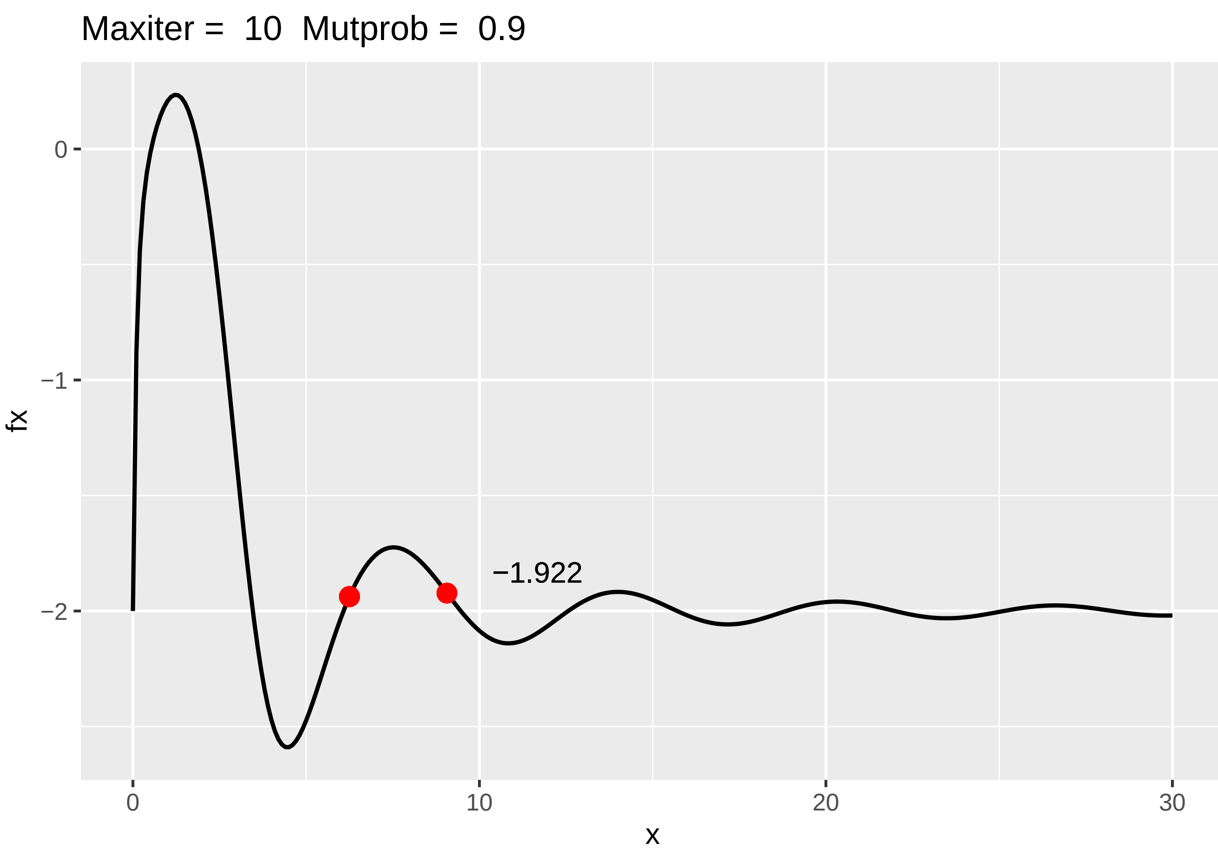
```r
  #Initial population
  X <- seq(0, 30, by = 5)

  #function values for initial population
  Values <- fx(X)

  #counter for iterations
  i = 0

  #initialize vector to store max values
  fx_max <- matrix(NA_real_, nrow = maxiter, ncol = 2)

  #iterative loop for finding maxima

  for (i in seq(maxiter)) {
    parents <- sample(X, 2) #sample parents from initial population
    victim <- X[which.min(Values)] #find lowest function value as victim in population
    kid <- crossover(parents[1], parents[2]) #create kid from parents
    if (runif(1) < mutprob) kid <- mutate(kid) #mutate with mutation probability
    X[X == victim] <- kid #replace victim with kid

    Values <- fx(X) #update Values vector

    fx_max[i, 1] <- X[which.max(Values)]
    fx_max[i, 2] <- max(Values)
  }

  #setup plot data
  colnames(fx_max) <- c("x", "fx")
  fx_max <- as.data.frame(fx_max)
  fx_max$status <- "new"
  plot_df$status <- "original"
  plot_df <- rbind.data.frame(plot_df, fx_max)

  #printing plot
  print(ggplot(plot_df, aes(x, fx)) +
    geom_line(size = 0.75, col = 'black') +
    geom_point(data = fx_max, aes(x = x, y = fx), color = 'red', size = 3) +
    labs(title = paste("Maxiter = ", maxiter, " Mutprob = ", mutprob)) +
    geom_text(data = fx_max, aes(label = ifelse(fx == max(fx),as.character(round(fx,3)),'')), hjust = -0


}

#check outputs
GA_fn(10, 0.1)
```



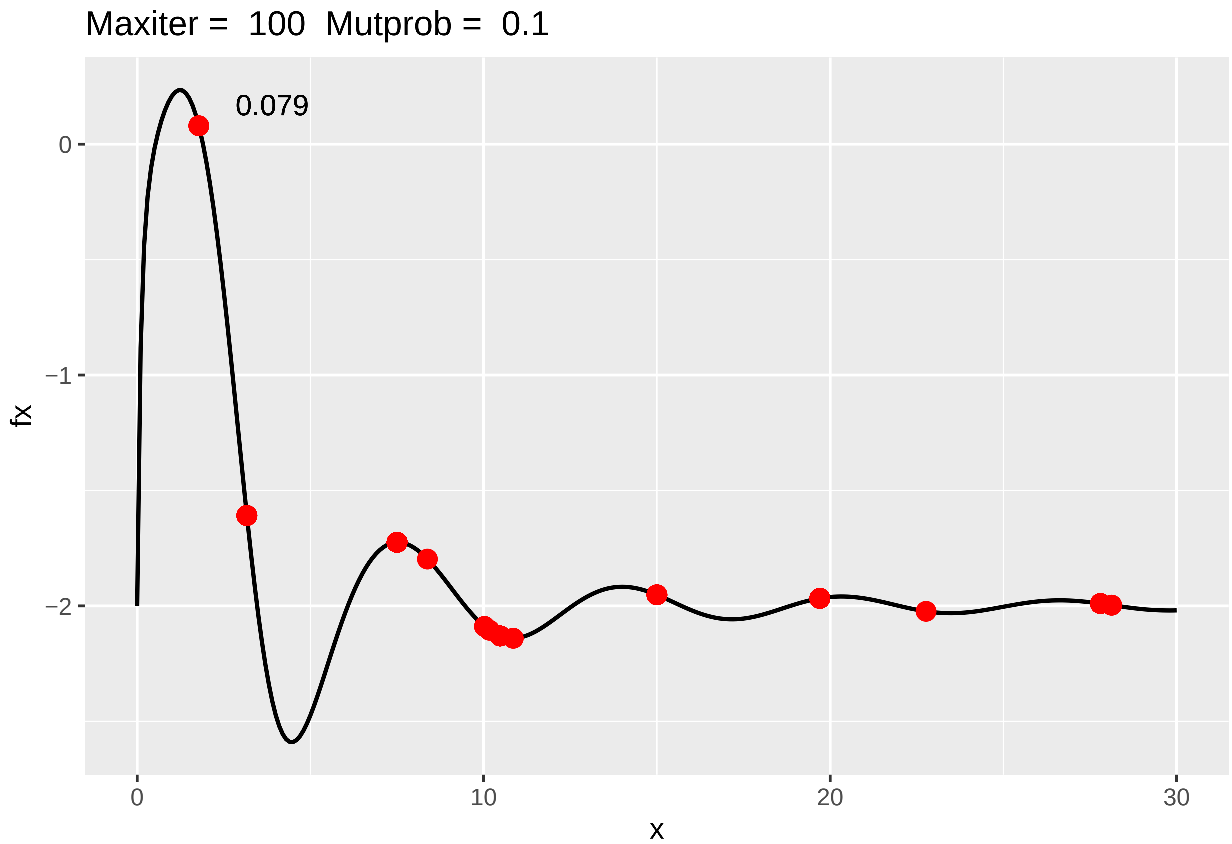

```
GA_fn(10, 0.5)
```



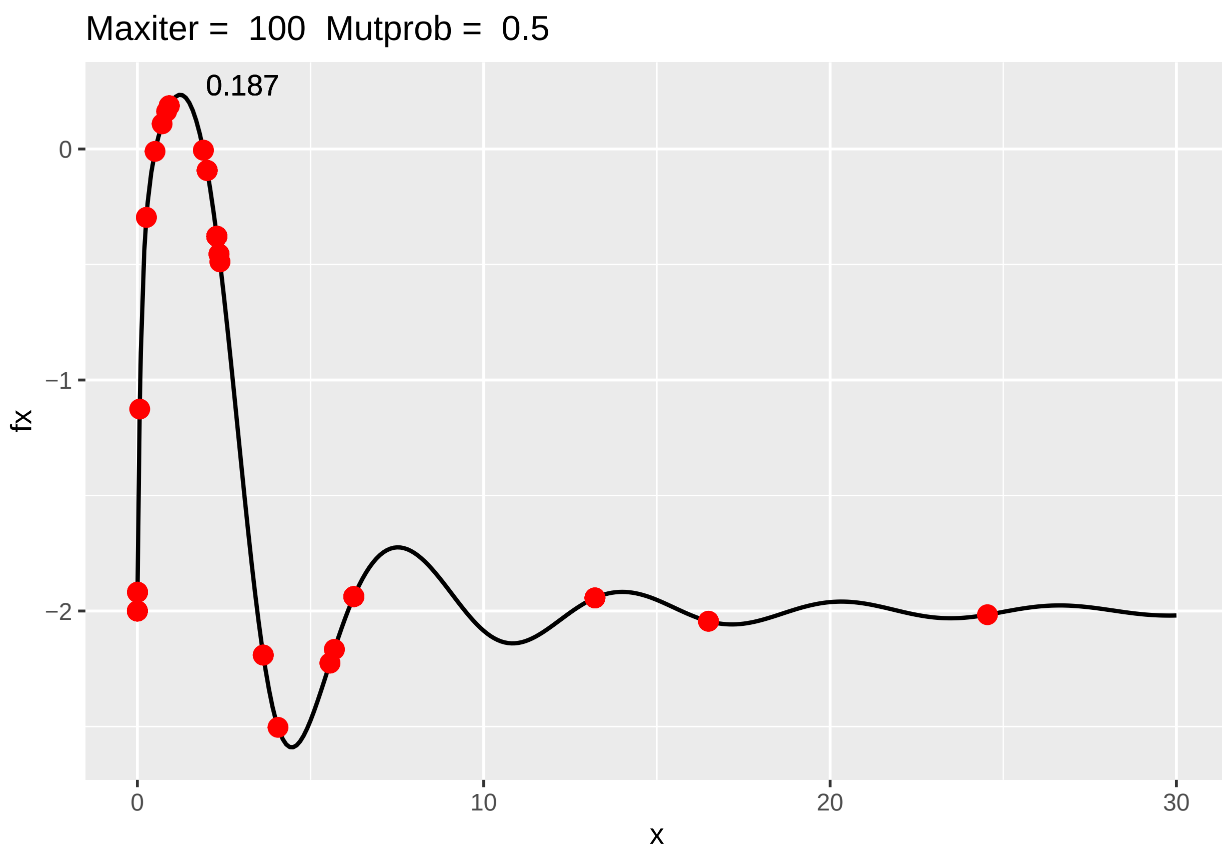

```
GA_fn(10, 0.9)
```



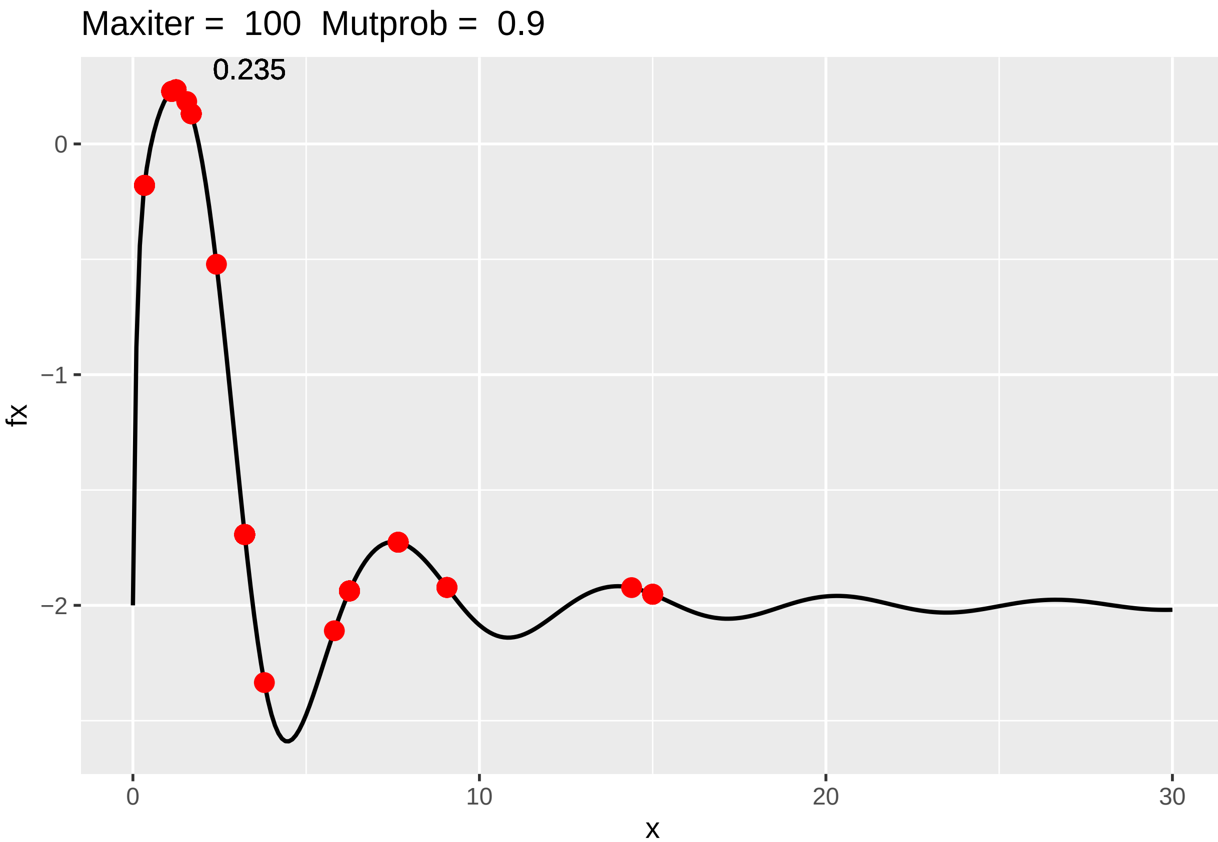

```r
GA_fn(100, 0.1)
```




```
GA_fn(100, 0.5)
```




```
GA_fn(100, 0.9)
```

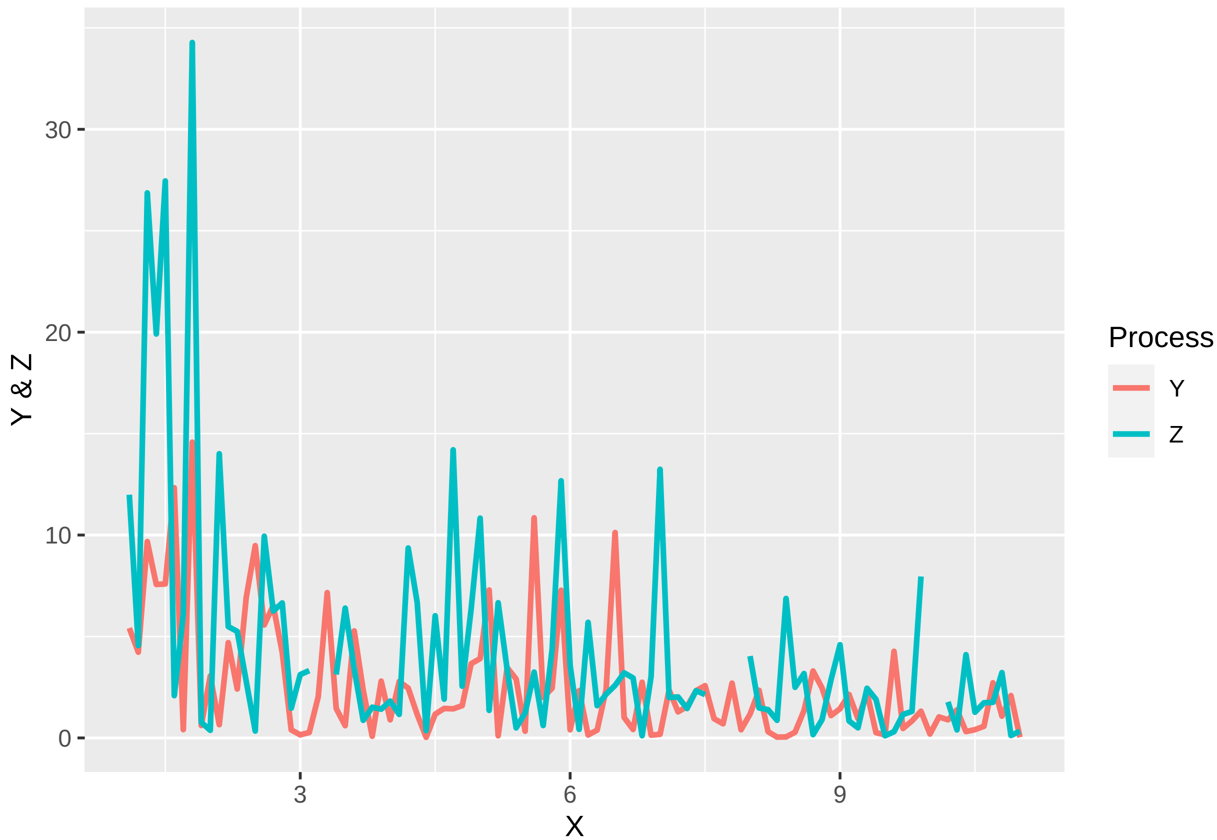


From the initial plot of the objective function we were able to note that the objective function was a decaying function with its global maxima $f(x) \approx 0.234$.

We ran the algorthm as specified in the assignment and were able to generate the plots for different configurations of the max iterations and mutation probabilities. The inferences were as follows:

1. Larger number iterations has a better chance of estimating the maxima, as more experiments are available for convergence
2. Higher mutation probability results in introduction of new population and hence discovery of new objective points, this increasing the chance of discovering the maxima

```r
setwd("D:/Documents/LiU Final/Autumn 21/732A90 - Computational Stats/Labs/Lab6")
data <- read.csv("physical1.csv")
data_plot <- melt(data, variable.name = 'Process', id.vars = 'X')
ggplot(data_plot, aes(x= X, y = value, col = Process)) + geom_line(size = 1) + labs(y = "Y & Z")
```



The 2 processes Y & Z seem to have some correlation between them as the amplitude of the spikes in the responses Y & Z seems to resemble in relative magnitude to each other. In other words, both series are peaking and dropping in neighboring ranges of X.

Both Y & Z seem to show a dampening in their amplitudes wrt X as X increases. This is slightly harder to claim for Z as it has a lot of missing values.

```r
X <- data$X
Y <- data$Y
Z <- data$Z

# function for the EM algorithm
lambda_EM <- function(X, Y, Z, lambda0, tolerance) {

  n <- length(X) #number of observations
  u <- which(is.na(Z)) #indexes of the na observations in Z

  X_z <- X[-u] #those X for which Z is not na
  Z_notna <- Z[-u] #Z without na values
  lambda_k <- lambda0

  #calculate first values of lambda for input to while loop
```

```r
  lambda_kplus1 <- 0.5/n*(sum(X*Y) + 0.5*sum(X_z*Z_notna) + length(u)*lambda_k)  # known from the assig
  tol_check <- lambda_kplus1 - lambda_k #tolerance

  while (abs(tol_check) > tolerance) {

    lambda_kplus1 <- 0.5/n*(sum(X*Y) + 0.5*sum(X_z*Z_notna) + length(u)*lambda_k)

    tol_check <- lambda_kplus1 - lambda_k

    lambda_k <- lambda_kplus1

    count <<- count + 1 #to count number of iterations to reach lambda within tolerance
  }
return(lambda_kplus1)
}

count <- 1
EM_result <-  lambda_EM(X, Y, Z, 100, 0.001)

#result of EM
cat("Lambda from EM = ",EM_result)
```

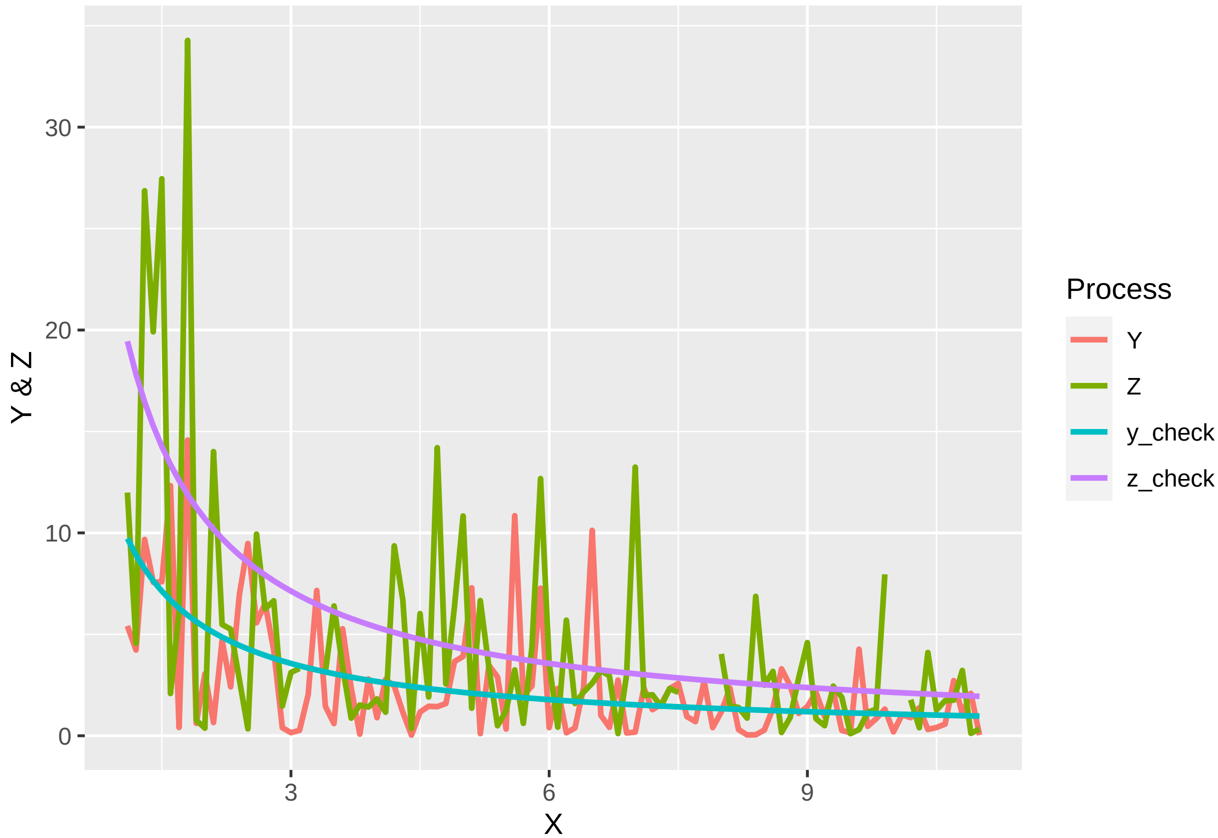
```
## Lambda from EM =  10.69566
```

```r
cat("\n Number of iterations needed = ",count)
```

```
##
##  Number of iterations needed =  6
```

We can see from the result that the algorithm converged to a value of $\lambda_{EM} = 10.696$ and 6 iterations were required to reach this solution.

```r
y_check <- EM_result/X
z_check <- 2*EM_result/X

data_check <- data.frame(X, Y, Z, y_check, z_check)
data_plot_check <- melt(data_check, variable.name = 'Process', id.vars = 'X')
ggplot(data_plot_check, aes(x= X, y = value, col = Process)) + geom_line(size = 1) + labs(y = "Y & Z")
```



Y & Z
30
20
10
0
3
6
9
X
Process
Y
Z
y_check
z_check