Lab5,Group15

Naveen Gabriel ,Sridhar Adhikarla

2019-03-08

Contents

1 Genetic Algorithm

1.1 Define the function

Below is the objective function :

$$f(x) := \frac{x^2}{e^x} - 2e^{(-(\frac{9sinx}{x^2+x+1}))}$$

```
#objective function . This is used to check the fitness of the population
funcs <- function(x){
    frst_trm = x^2/exp(x)
    scnd_trm = 2*exp(-(9*sin(x))/((x^2)+x+1))

    y <- frst_trm - scnd_trm
    return(y)
}
```

1.2 Define the function crossover

```
#Produce new kid
crossover <- function(x,y){
    return((x+y)/2)
}
```

1.3 Define function mutate

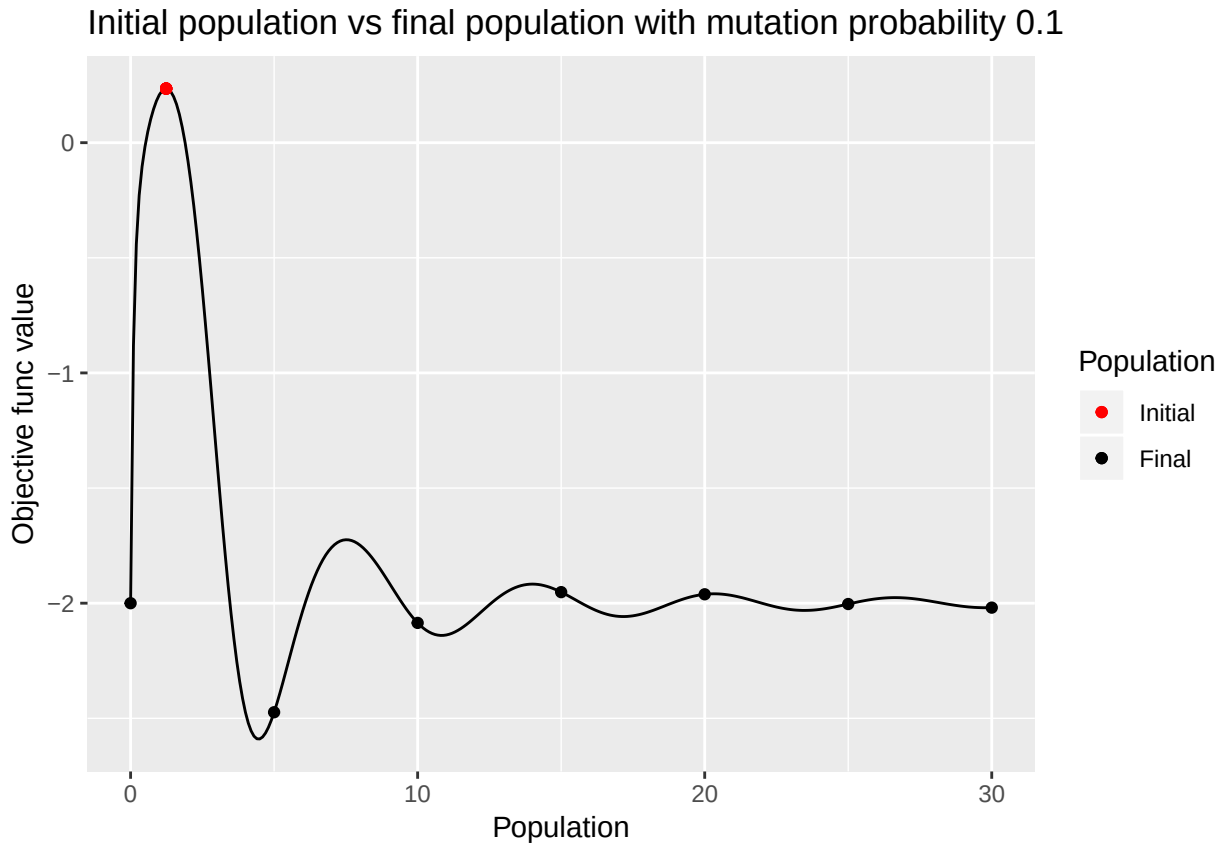
```
#Mutate the kid
mutate <- function(x) {
    return((x^2)%%30)
}
```

1.4 Function for genetic algorithm

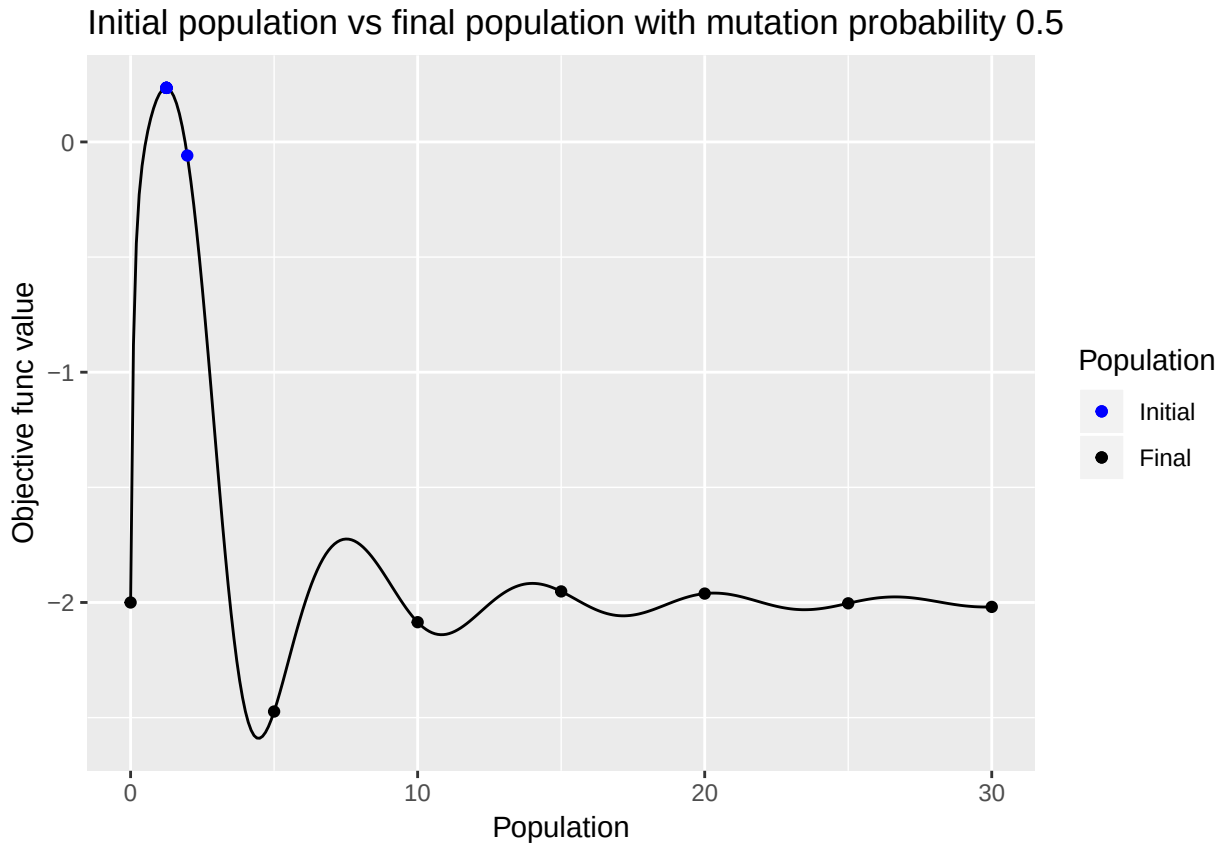
```
genetic_algorithm <- function(maxiter,mutprob) {
    x = seq(0,30,by=5)

    Values <- funcs(x)
    max_obj_func <- 0

    for(i in 1: maxiter) {
        s <- sample(1:7,2)          #Two index is randomly sampled from population

#Find the victim which has the least objectiv func
        victim <- order(Values)[1]
        kid <- crossover(x[s[1]],x[s[2]])  #Send the kids for the crossover

        u <- runif(1,0,1)           #Using random find check if we can mutate the kid
        if(mutprob>u)
            kid <- mutate(kid)
```

```
        x <- replace(x,victim,kid)

        new_val <- funcs(kid)  #calculate objective function with respect to the new kid
        Values <-replace(Values,victim,new_val) #Update the resultant objective function
    }


    return(list(x,Values))
}

gene_algo1 <- genetic_algorithm(10100,0.1)
gene_algo2 <- genetic_algorithm(10100,0.5)
gene_algo3 <- genetic_algorithm(10100,0.9)

 x <- seq(0,30,by=0.1)
 val <- funcs(x)

 data_pt <- cbind(x=x,val=val)
 data_pt <- as.data.frame(data_pt)

 x_ini=seq(0,30,by=5)
 y_ini=funcs(x_ini)
```

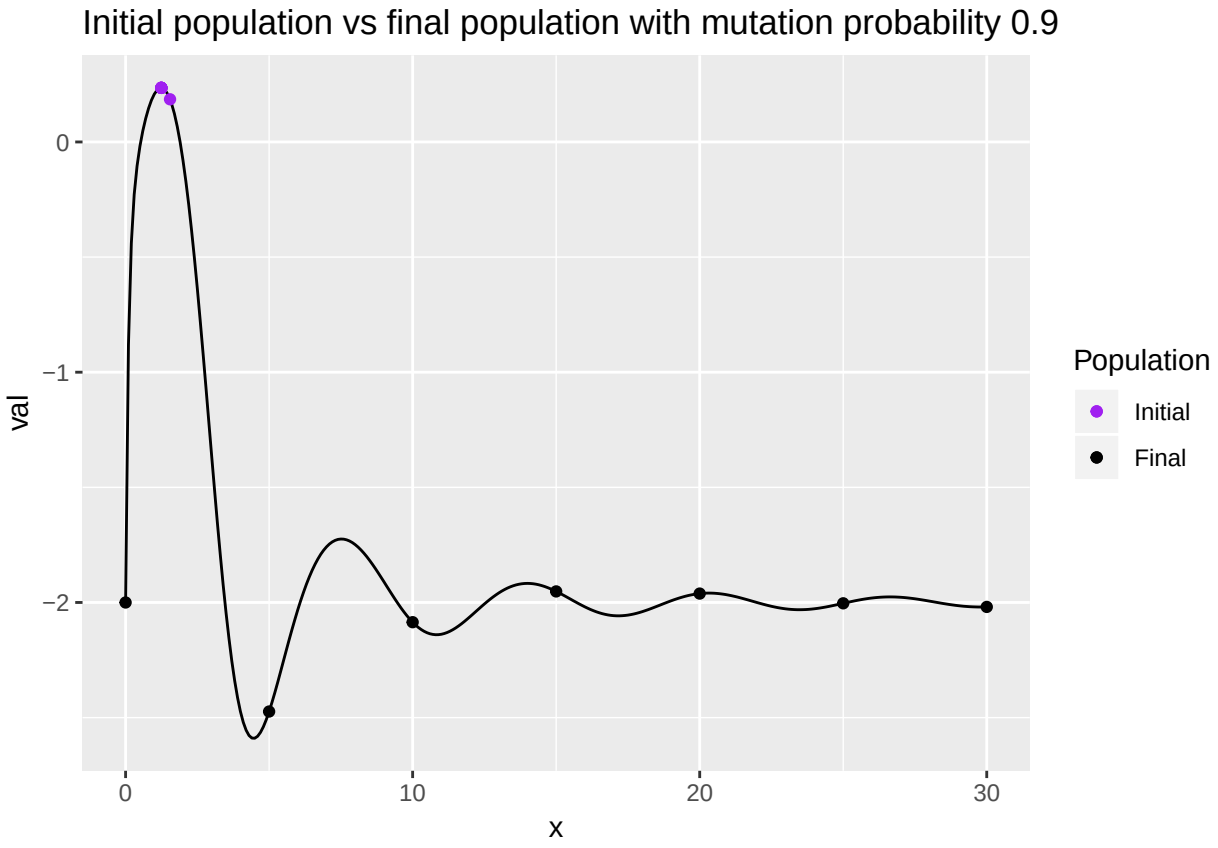
With mutation probability 0.1, not all the 5 points might reach maximum because the probability is very low. But it might happen due to random chance that the points might reach maxima since we are doing many iterations.



With mutation probability 0.5, the probability for the initial population to mutate is more than the previous one. Here we can see all the 5 points has converged to one.



With mutation probability 0.9, the probability to mutate further increases than the previous one. So most of the points lies on the peak.



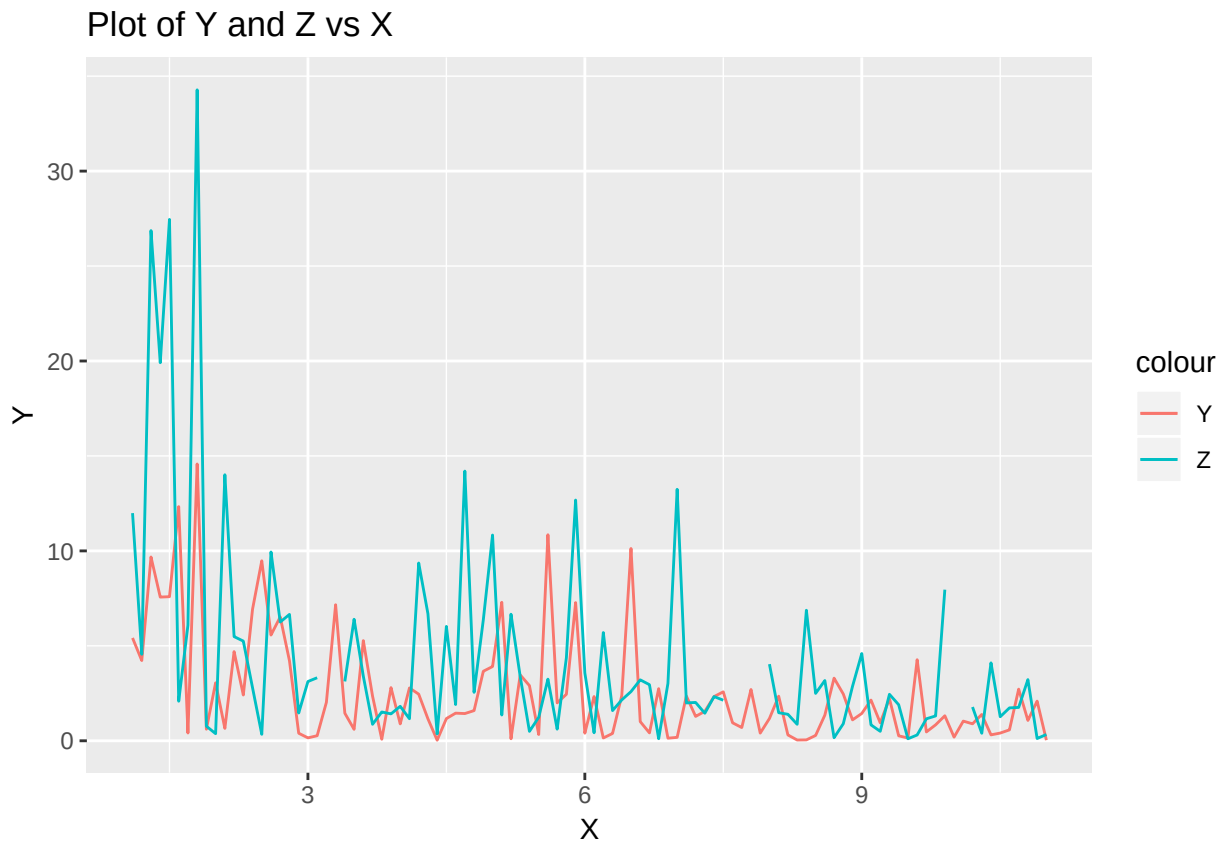
1.5 Analysis

Here we choose victim based on the objective function. Basically we can cosnider the victim as the weakest in the population. So we tend to replace the weakest with the strong one with each iteration. With many iteration, all the the points in the population converges to maximum. With mutation probability 0.1, the tendency to mutate is very less. But with many iterations, the mutation keeps happens although at slow rate. The mutation further increases with probability 0.5 and 0.9

2 EM Algorithm

2.1 Time Series plot describing dependence of Z and Y versus X.

```
#1
library(ggplot2)

physical_data = read.csv("physical1.csv")
#head(physical_data)

ggplot(physical_data) +
  geom_line(aes(X, Y, color='Y')) +
  geom_line(aes(X, Z, color='Z')) +
  ggtitle("Plot of Y and Z vs X")
```



Both Y and Z in this plot have a similar patter, just the scale of the oscillation is different. Z oscilates with an higher amplitude than Y. Both Y and Z decay over time in a similar manner.

2.2 Derive an EM algorithm that estimates λ.

$$Y_i \approx exp(\frac{X_i}{\lambda}), Z_i \approx exp(\frac{X_i}{2 * \lambda})$$

Where λ is an unknown parameters. The goal is to derive the EM algorithm that estimates λ.

$$
\begin{aligned}
L(\lambda|Y,Z) &= \prod_{i=1}^{n} f(Y) \times \prod_{i=1}^{n} f(Z) \\
&= \prod_{i=1}^{n} \frac{X_i}{\lambda} \cdot e^{-\frac{X_i}{\lambda}Y_i} \times \prod_{i=1}^{n} \frac{X_i}{2\lambda} \cdot e^{-\frac{X_i}{\lambda}Z_i} \\
&= \frac{X_1 \cdot \ldots \cdot X_n}{\lambda^n} \times e^{-\frac{1}{\lambda}\sum_1^n X_i Y_i} \times \frac{X_1 \cdot \ldots \cdot X_n}{(2\lambda)^n} \times e^{-\frac{1}{2\lambda}\sum_1^n X_i Z_i} \\
lnL(\lambda|Y,Z) &= \sum_{i=1}^{n} ln(X_i) - nln(\lambda) - \frac{1}{\lambda}\sum_{i=1}^{n} X_i Y_i \ + \sum_{i=1}^{n} ln(X_i) - nln(2\lambda) - \frac{1}{2\lambda}\sum_{i=1}^{n} X_i Z_i
\end{aligned}
$$

2.2.1 E-step : Derive Q function

$$
\begin{aligned}
Q(\theta,\theta^k) &= E[\ loglik(\lambda|Y,Z) \mid \lambda^k,(Y,Z)] \\
&= \sum_{i=1}^{n} ln(X_i) - nln(\lambda) - \frac{1}{\lambda}\sum_{i=1}^{n} X_i Y_i \ + \sum_{i=1}^{n} ln(X_i) - nln(2\lambda) \\
&\quad - \frac{1}{2\lambda}\left[\sum_{i=1}^{n} X_i Z_i \ + m \cdot X_i \cdot \frac{2\lambda_{k-1}}{X_i}\right]
\end{aligned}
$$

Here, we are taking expectation on the missing values in Z, so we need to seperate the Z_{obs} and Z_{miss}. Here we are assuming there are 'm' missing Z values. λ_k is the lambda value from the previous iteration.

2.2.2 M-step

We calculate the maximum likelihood estimate (λ_{MLE}) of the parameters by taking the derivative with respect to λ. Repeat till estimate converges.

$$
\begin{aligned}
-\frac{n}{\lambda} - \frac{n}{\lambda} + \frac{\sum_{i=1}^{n} X_i Y_i}{\lambda^2} \ + \frac{\sum_{i}^{m} X_i Z_i \ + m \cdot 2\lambda_{k-1}}{2\lambda^2} &:= 0 \\
-2\lambda(2n) + 2\sum_{i=1}^{n} X_i Y_i \ + \sum_{i=1}^{n} X_i Z_i + m \cdot 2\lambda_{k-1} &:= 0
\end{aligned}
$$

$$
\lambda = \frac{\sum_{i=1}^{n} X_i Y_i + \frac{1}{2}\sum_{i=1}^{n} X_i Z_i + m \cdot \lambda_{k-1}}{2n}
$$

2.3 Implement EM Algorithm.

```
##3
EM.Exp<-function(data, eps, kmax, lambda_0){
  X <- data$X; Y <- data$Y; Z <- data$Z
  Xobs <- X[!is.na(Z)]
  Zobs <- Z[!is.na(Z)]
  Zmiss <- Z[is.na(Z)]
  n <- length(X)
  m <- length(Zmiss)
  k <<- 0
```

```
  llvalprev <- 0
  llvalcurr <- lambda_0
  cat("Initial lambda value : ", llvalcurr, "\n")

  while ((abs(llvalprev-llvalcurr)>eps) && (k<(kmax+1))){
    llvalprev<-llvalcurr
    llvalprev <- llvalcurr
    llvalcurr <- (sum(X*Y)+sum(Xobs*Zobs)/2+m*llvalprev)/(2*n)
    k <<- k+1
  }
  return(c(llvalprev,llvalcurr,k))
}
lmb = EM.Exp(physical_data,0.001,100,100)
```

```
Initial lambda value :  100
```

```
cat("The final converged lambda value : ", lmb[1], "\n")
```

```
The final converged lambda value :  10.69587
```

```
cat("Number of steps taken to converge : ", lmb[3], "\n")
```

```
Number of steps taken to converge :  5
```

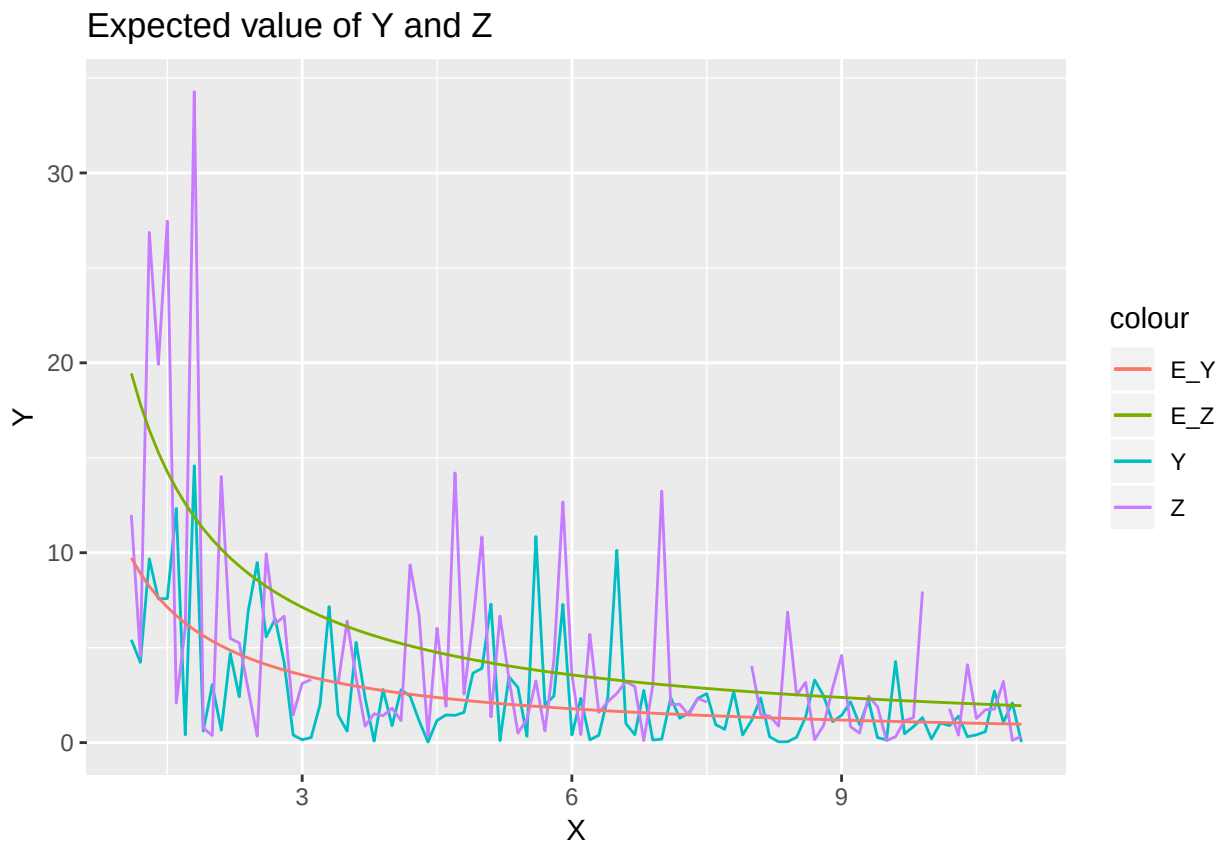
```
lambda <- lmb[2]
new_data <- physical_data
new_data$E_Y <- lambda/physical_data$X
new_data$E_Z <- 2*lambda/physical_data$X
```

We initialized λ_0 to 100 and it got converged at value $\lambda_{Converged}$ to 10.69587

We then went on to calculate the Expected values for Y and Z using this λ. Since Y and Z belong to Exponential distribution, the formula to compute the Expected value for them is :

$$E[Y] = \frac{\lambda}{X_i}\ , \quad E[Z] = \frac{2\lambda}{X_i}$$

```
#4
ggplot(data=new_data) +
  geom_line(aes(x = X, y = Y, colour = "Y")) +
  geom_line(aes(x = X, y = Z, colour = "Z")) +
  geom_line(aes(x = X, y = E_Y, colour = "E_Y")) +
  geom_line(aes(x = X, y = E_Z, colour = "E_Z")) +
  ggtitle("Expected value of Y and Z")
```



From the plot we can see that the E[Y] and E[Z] capture the trend of Y and Z respectively, so we can conclude that the estimated lambda using EM algorithm is a good estimate.

3 Appendix

```
knitr::opts_chunk$set(
    echo = TRUE,
    eval=TRUE,
    message = FALSE,
    warning = FALSE,
    comment = NA
)
library(ggplot2)

#objective function . This is used to check the fitness of the population
funcs <- function(x){
    frst_trm = x^2/exp(x)
    scnd_trm = 2*exp(-(9*sin(x))/((x^2)+x+1))

    y <- frst_trm - scnd_trm
    return(y)
}

#Produce new kid
crossover <- function(x,y){
    return((x+y)/2)
}
#Mutate the kid
mutate <- function(x) {
    return((x^2)%%30)
}
genetic_algorithm <- function(maxiter,mutprob) {
    x = seq(0,30,by=5)

    Values <- funcs(x)
    max_obj_func <- 0

    for(i in 1: maxiter) {
        s <- sample(1:7,2)         #Two index is randomly sampled from population

#Find the victim which has the least objectiv func
        victim <- order(Values)[1]
        kid <- crossover(x[s[1]],x[s[2]])  #Send the kids for the crossover

        u <- runif(1,0,1)           #Using random find check if we can mutate the kid
        if(mutprob>u)
            kid <- mutate(kid)

        x <- replace(x,victim,kid)

        new_val <- funcs(kid)  #calculate objective function with respect to the new kid
        Values <-replace(Values,victim,new_val) #Update the resultant objective function
    }


    return(list(x,Values))
```

```
}

gene_algo1 <- genetic_algorithm(10100,0.1)
gene_algo2 <- genetic_algorithm(10100,0.5)
gene_algo3 <- genetic_algorithm(10100,0.9)

 x <- seq(0,30,by=0.1)
 val <- funcs(x)

 data_pt <- cbind(x=x,val=val)
 data_pt <- as.data.frame(data_pt)

 x_ini=seq(0,30,by=5)
 y_ini=funcs(x_ini)
 ggplot() + geom_line(aes(x=x,y=val))+
     geom_point(aes(x=x_ini,y=y_ini,color="y_ini"))+
     geom_point(aes(x=gene_algo1[[1]],y=gene_algo1[[2]],color="y"))+
     ggtitle("Initial population vs final population with mutation probability 0.1")+
     labs(y = "Objective func value",x="Population") +
     scale_color_manual(name="Population",
                        labels = c("Initial",
                                   "Final"),
                        values = c("y_ini"="black",
                                   "y"="red"))

 ggplot() + geom_line(aes(x=x,y=val))+
     geom_point(aes(x=x_ini,y=y_ini,color="y_ini"))+
     geom_point(aes(x=gene_algo2[[1]],y=gene_algo2[[2]],color="y"))+
     ggtitle("Initial population vs final population with mutation probability 0.5")+
     labs(y = "Objective func value",x="Population") +
     scale_color_manual(name="Population",
                        labels = c("Initial",
                                   "Final"),
                        values = c("y_ini"="black",
                                   "y"="blue"))

 ggplot() + geom_line(aes(x=x,y=val))+
     geom_point(aes(x=x_ini,y=y_ini,color="y_ini"))+
     geom_point(aes(x=gene_algo3[[1]],y=gene_algo3[[2]],color="y"))+
     ggtitle("Initial population vs final population with mutation probability 0.9")+
              scale_color_manual(name="Population",
                        labels = c("Initial",
                                   "Final"),
                        values = c("y_ini"="black",
                                   "y"="purple"))
knitr::opts_chunk$set(
    echo = TRUE,
    eval=TRUE,
    message = FALSE,
    warning = FALSE,
    comment = NA
)
```

```r
#1
library(ggplot2)

physical_data = read.csv("physical1.csv")
#head(physical_data)

ggplot(physical_data) +
  geom_line(aes(X, Y, color='Y')) +
  geom_line(aes(X, Z, color='Z')) +
  ggtitle("Plot of Y and Z vs X")

##3
EM.Exp<-function(data, eps, kmax, lambda_0){
  X <- data$X; Y <- data$Y; Z <- data$Z
  Xobs <- X[!is.na(Z)]
  Zobs <- Z[!is.na(Z)]
  Zmiss <- Z[is.na(Z)]
  n <- length(X)
  m <- length(Zmiss)
  k <<- 0
  llvalprev <- 0
  llvalcurr <- lambda_0
  cat("Initial lambda value : ", llvalcurr, "\n")

  while ((abs(llvalprev-llvalcurr)>eps) && (k<(kmax+1))){
    llvalprev<-llvalcurr
    llvalprev <- llvalcurr
    llvalcurr <- (sum(X*Y)+sum(Xobs*Zobs)/2+m*llvalprev)/(2*n)
    k <<- k+1
  }
  return(c(llvalprev,llvalcurr,k))
}
lmb = EM.Exp(physical_data,0.001,100,100)
cat("The final converged lambda value : ", lmb[1], "\n")
cat("Number of steps taken to converge : ", lmb[3], "\n")
lambda <- lmb[2]
new_data <- physical_data
new_data$E_Y <- lambda/physical_data$X
new_data$E_Z <- 2*lambda/physical_data$X

#4
ggplot(data=new_data) +
  geom_line(aes(x = X, y = Y, colour = "Y")) +
  geom_line(aes(x = X, y = Z, colour = "Z")) +
  geom_line(aes(x = X, y = E_Y, colour = "E_Y")) +
  geom_line(aes(x = X, y = E_Z, colour = "E_Z")) +
  ggtitle("Expected value of Y and Z")
```