Computer Lab 6

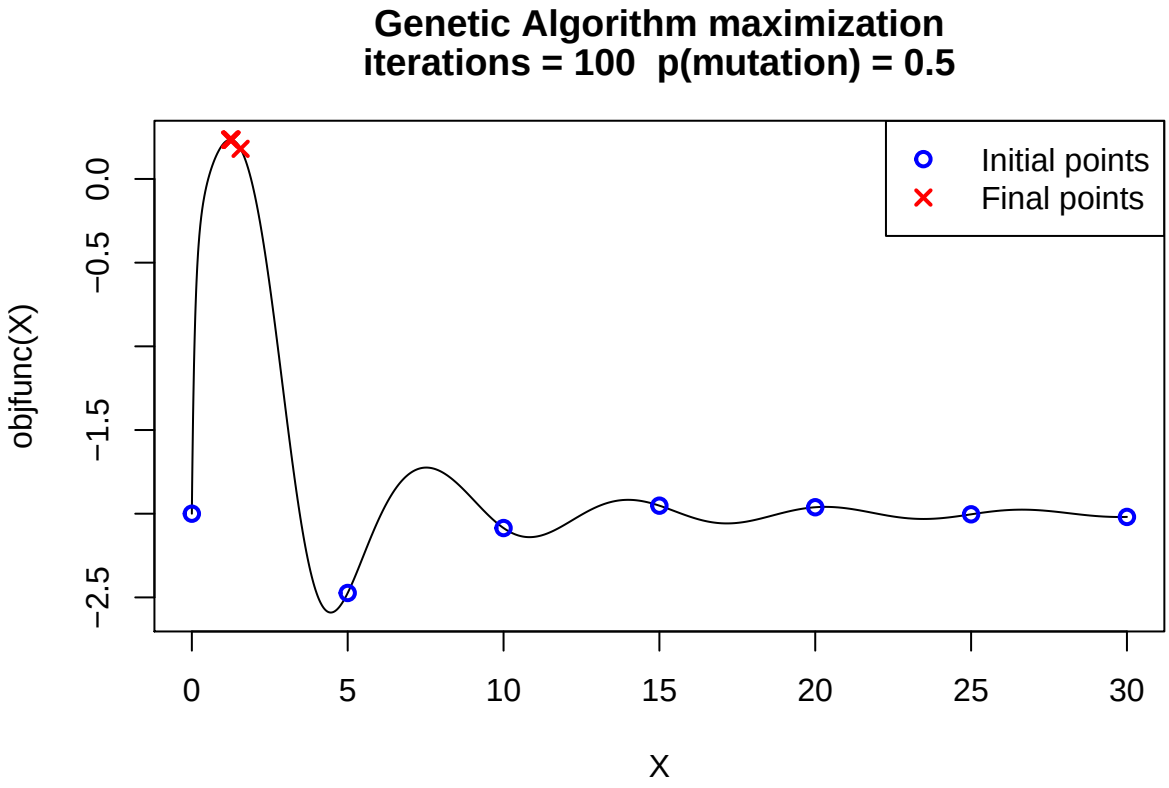
Thomas Zhang

2016 M03 10

Assignment 1

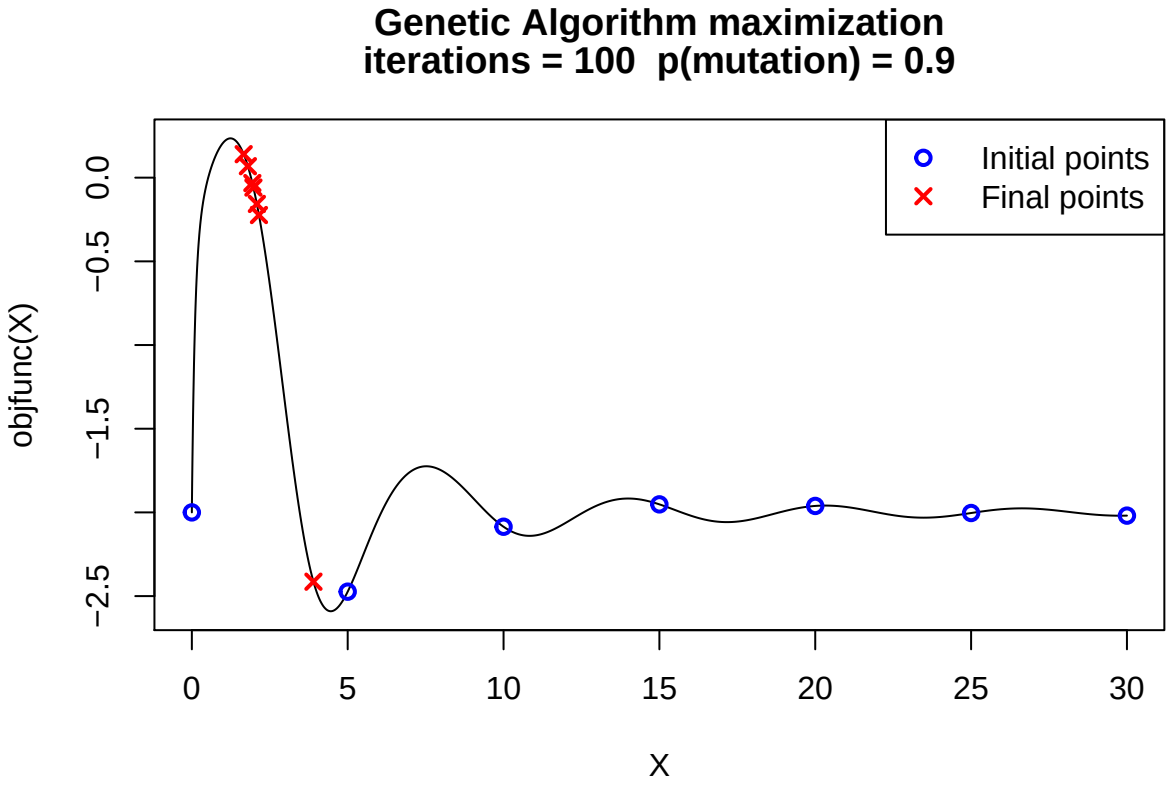
In this assignment we try to perform one-dimensional maximization, using the genetic algorithm specified in the instructions, on the objective function:

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



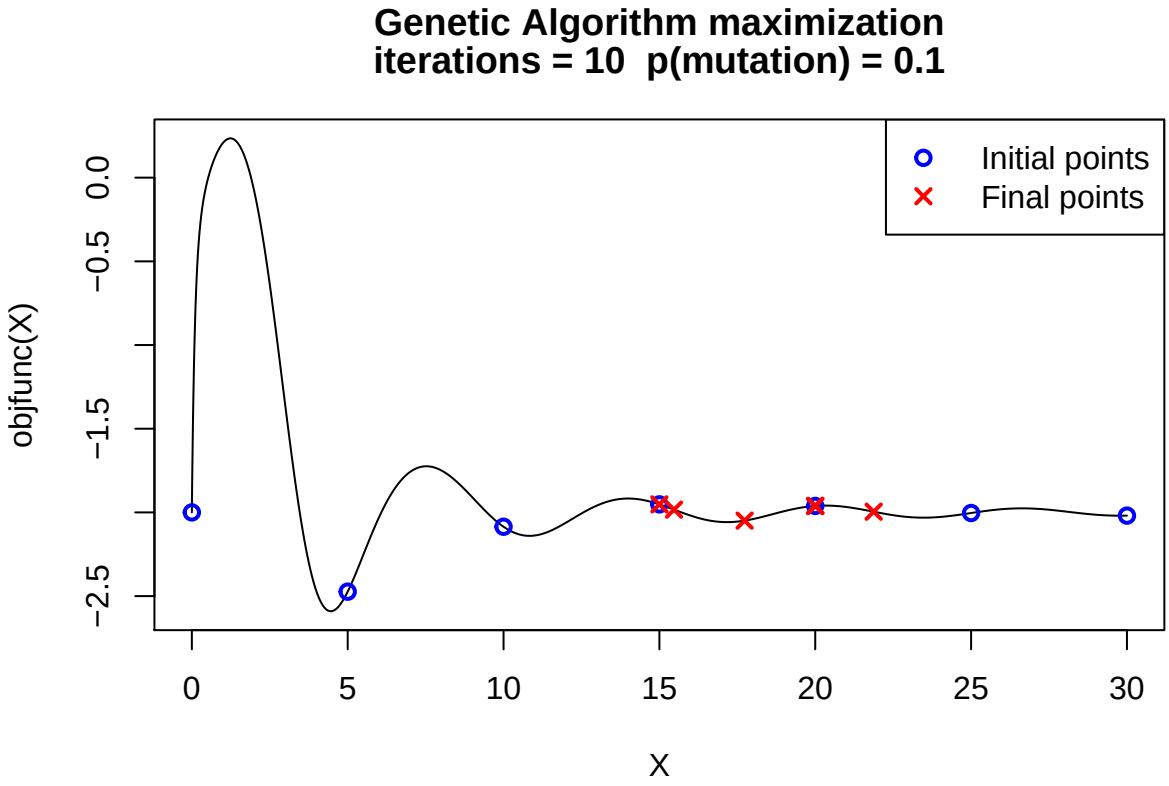

```
## [1] "Final maximum value achieved: 0.2348"
```

As we can see, setting `maxiter = 100` and `p(mutation) = 0.5` gives rather good results, in the sense that the set of final points all are located close to the true maximum. The setting `maxiter = 100` and `p(mutation) = 0.9` produces similar results but the final points are more spread out along the function curve, due to the higher mutation rate.




```
## [1] "Final maximum value achieved: 0.1403"
```

To contrast, we can try setting `maxiter = 10` and `p(mutation) = 0.1`, which often results in the final points being located not far from where they started.

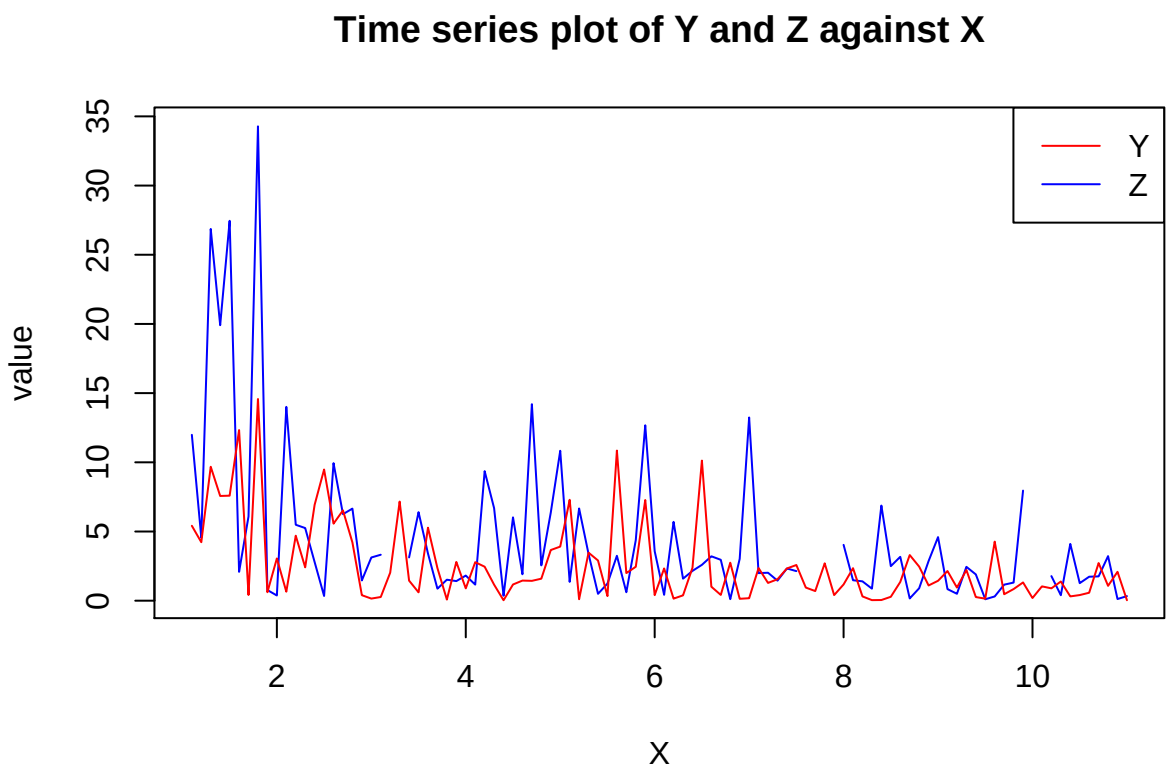



```
## [1] "Final maximum value achieved: -1.9519"
```

We can infer that the genetic algorithm used requires more than 10 iterations to be effective and that the mutation rate should be moderately high for best results.

Assignment 2

We plot Z and Y as time series vs X.



We see that both are Z and Y stochastic processes, with a high degree of correlation. There are some Z values missing at $X = 3, 8, 10$ and Z values tend to be larger than Y values for a given X. The values appear to be decreasing with increasing X.

We are going to estimate Z and Y using the EM-algorithm, existing data and the models

$$Y_i \sim Exp(\frac{X_i}{\lambda}) \text{ and } Z_i \sim Exp(\frac{X_i}{2\lambda})$$

Where λ is a unknown parameter to be estimated.

We start by writing down the likelihood functions for Z and Y.

$$L(Y|\lambda) = \frac{\prod_{i=1}^{n} X_i}{\lambda^n} \exp\left(-\frac{\sum_{i=1}^{n} Y_i X_i}{\lambda}\right)$$

$$L(Z|\lambda) = \frac{\prod_{i=1}^{n} X_i}{2^n \lambda^n} \exp\left(-\frac{\sum_{i=1}^{n} Z_i X_i}{2\lambda}\right) = \frac{\prod_{i=1}^{n} X_i}{2^n \lambda^n} \exp\left(-\frac{\sum_{O} Z_i X_i}{2\lambda} - \frac{\sum_{M} Z_i X_i}{2\lambda}\right)$$

where O is the set of indices for the observed values of Z and M is the set of indices for the missing values of Z.

Then, we multiply the likelihoods together and take the logarithm.

$$l(Y, Z|\lambda) = \log \frac{(\prod_{i=1}^{n} X_i)^2}{2^n \lambda^{2n}} - \frac{\sum_{i=1}^{n} Y_i X_i}{\lambda} - \frac{\sum_{O} Z_i X_i}{2\lambda} - \frac{\sum_{M} Z_i X_i}{2\lambda}$$

At this point the E-step can be done. We are going to set every missing Z_i to its Expected value given X_i and the last lambda value, λ_t.

$$E[Z_i|X_i, \lambda_t] = \frac{2\lambda_t}{X_i}$$

Due to the exponential distribution of Z_i.

$$E(l(Y, Z|\lambda)) = \log \frac{(\prod_{i=1}^{n} X_i)^2}{2^n \lambda^{2n}} - \frac{\sum_{i=1}^{n} Y_i X_i}{\lambda} - \frac{\sum_O Z_i X_i}{2\lambda} - \frac{|M|\lambda_t}{\lambda}$$

Where $|M|$ is the number of missing Z values. For the M-step, we have to compute the derivative with respect to λ and put it equal to zero.

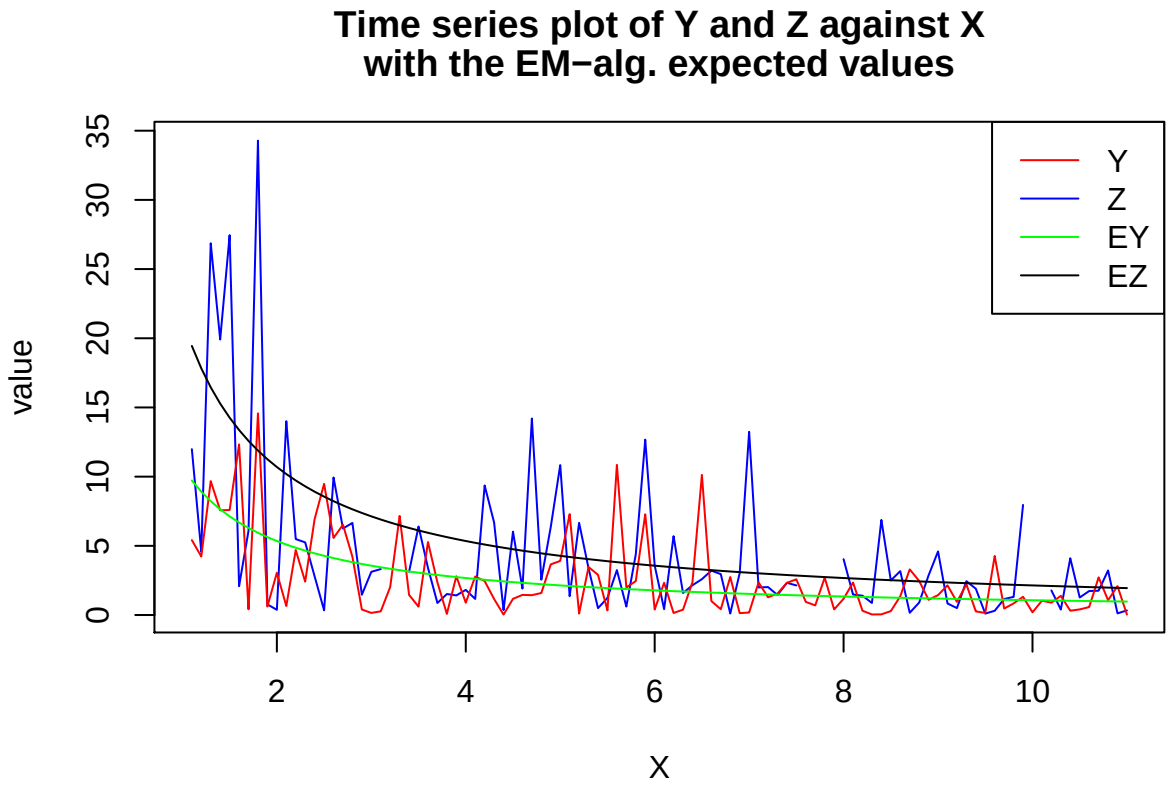
Doing this we get:

$$\lambda_{t+1} = \frac{\sum_{i=1}^{n} Y_i X_i}{2n} + \frac{\sum_O Z_i X_i}{4n} + \frac{|M|\lambda_t}{2n}$$

We implement this algorithm in R where we start with $\lambda_0 = 100$ and stop when one step yields less than 0.001 change in λ_t. Then we plot the expected values of Z and Y given our model and this λ value.

```
## [1] "Optimal Lambda value:  10.6956189726956"
```

```
## [1] "No of EM alg. iterations run:  4"
```



We see that the expected values fit the data well.

Appendix

R code

```
data <- read.csv("physical.csv")
n <- dim(data)[1]
#head(data)
objfunc <- function(x){
  res <- x^2/exp(x) - 2 * exp(-9 * sin(x) / (x^2 + x + 1))
  return(res)
}
crossover <- function(x,y){
  return((x + y) / 2)
}
mutate <- function(x){
  res <- x^2 %% 30
  return(res)
}
genalgfunc <- function(maxiter = 100 , mutprob = 0.5){
  X <- seq(0,30,0.01)
  plot(X,objfunc(X), main = c("Genetic Algorithm maximization",
                              paste("iterations =",maxiter,
                                    " p(mutation) =",mutprob)),
       type="l")
  X <- seq(0,30,5)
  Values <- objfunc(X)
  points(X,Values,col = "blue",cex = 1, pch = 1,lwd = 2)
  count <- 0
  maxvals <- c()
  repeat{
    if(count == maxiter){
      break
    }
    parents <- sample(X,2)
    victim <- order(Values)[1]
    kid <- crossover(parents[1],parents[2])
    if(runif(1) < mutprob){
      kid <- mutate(kid)
    }
    X[victim] <- kid
    Values <- objfunc(X)
    maxvals <- c(maxvals,max(Values))
    count <- count + 1
  }
  points(X,Values,col = "red",cex = 1, pch = 4,lwd = 2)
  legend("topright",legend = c("Initial points","Final points"),
         lty = c(0,0), col= c("blue","red"), pch = c(1,4), lwd = c(2,2))
  paste("Final maximum value achieved:", round(maxvals[maxiter],4))
}

#par(mfrow = c(1,1))
#maxiter <- c(10,20,30)
#mutprob <- c(0.1,0.2,0.3)
```

```
#for(i in 1:2){
#  for(j in 1:2){
#    genalgfunc(maxiter[i],mutprob[j])
#  }
#}

set.seed(-3456)
genalgfunc( maxiter = 100, mutprob = 0.5)
set.seed(-3456)
genalgfunc( maxiter = 100, mutprob = 0.9)
set.seed(-3456)
genalgfunc( maxiter = 10, mutprob = 0.1)
plot(data$X,data$Z, type = "l", col = "blue",
     main=c("Time series plot of Y and Z against X"),xlab = "X",ylab = "value")
lines(data$X,data$Y,col = "red")
legend("topright",legend = c("Y","Z"),
       lty = c(1,1), col= c("red","blue"))
plot(data$X,data$Z, type = "l", col = "blue",
     main=c("Time series plot of Y and Z against X",
            "with the EM-alg. expected values"),xlab = "X",ylab = "value")
lines(data$X,data$Y,col = "red")
legend("topright",legend = c("Y","Z","EY","EZ"),
       lty = c(1,1,1,1), col= c("red","blue","green","black"))

Zmiss <- which(is.na(data$Z))
numbofmiss <- length(Zmiss)
Zobs <- which(!is.na(data$Z))
Zobsvals <- data$Z[Zobs]

helper <- function(lambdaold){
  res <- 1/ (2 * n) * ( sum(data$Y * data$X) + sum(data$Z[Zobs] * data$X[Zobs]) / 2 +
                          numbofmiss * lambdaold )
  return(res)
}

runner <- function(startlambda){
  oldlambda <- startlambda
  newlambda <- 0
  iterashuns <- 0
  repeat{
    if(abs(newlambda - oldlambda) < 0.001){
      break
    }
    oldlambda <- newlambda
    newlambda <- helper(oldlambda)
    iterashuns <- iterashuns + 1
  }

  return(newlambda)
}
paste("Optimal Lambda value: ",runner(100))
paste("No of EM alg. iterations run: ",4)
lambdaconverge <- runner(100)
```

```r
lines(data$X,lambdaconverge / data$X ,col = "green")
lines(data$X,2 * lambdaconverge / data$X ,col = "black")
## NA
```