Assignment2b

Xin Li

2/22/2020

Assignment2b

Define Secant Method

```
library(shape)
Secant<-function(func,x0, x1,Tolerance,MaxNumberOfIterations,DrawLines){
#initialize  Deviation to record |f(x)| that is large enough.
  Deviation <- 1000
  #Set up a counter, i, to record how many iterations you have performed. Set it equal t
o 0
  i <- 0
  # Initialize the values of X0 and X1
  x0 <- 1
  x1 <- 2

  #Set up a while loop until we hit the required target accuracy or the max number of st
eps
  while ((i<MaxNumberOfIterations)&&(Deviation>Tolerance))
  {
    y0 <- func(x0)
    y1 <- func(x1)

    if ((y0=="NaN")||(y1=="NaN")){
      cat("\nFunction or derivative not defined error.\n")
      break
    }
    if (DrawLines){
      Arrows(x0,0,x0,y0,col="blue",lty=2,arr.length=0.01, arr.type = "T")
      Arrows(x1,0,x1,y1,col="blue",lty=2,arr.length=0.01, arr.type = "T")
      }

    #Find the next X-value using Secant formula.
    x2 <- x1 - y1*(x0-x1)/(y0-y1)
    x0 <- x1
    x1 <- x2
    y2 <- func(x2)
    # calculate Deviation<- |f(x)-0|
    Deviation <- abs(y2)
    # increase the value of your iteration counter
    i <- i+1

    # if you like, have the program write out how it is getting on
    cat(paste("\nIteration ",i,":   X=",x1,"  Y=",y2))

    # If you are feeling fancy, add some line segments to the screen to show where it ju
st went
    # See the 'fixed points' code for a reminder of how to do that.
    # output the result
    if (Deviation<Tolerance){
      cat(paste("\nFound the root point: ",x1, " after ", i, "iterations"))
    }else{
      cat(paste("\nConvergence failure. Deviation: ",Deviation, "after ", i,    "iterati
ons"))}
  }


  # have the function return the answer
```

```
  return(x1)
}

F2 <- function(z){
  return(log(z)-exp(-z))
}
```

Define Newton-Raphson function

```
library(shape)
NewtonRaphson<-function(func,StartingValue,Tolerance,MaxNumberOfIterations,DrawLines){
  #initialize a variable, Deviation (say), to record |f(x)| so that you know how far awa
y you are from 0.
  #(So initialize it to some arbitrary large number)
  Deviation <- 1000
  #Set up a counter, i, to record how many iterations you have performed. Set it equal t
o 0
  i <- 0
  # Initialize the values of x and f(x)
  X  <- StartingValue

  #Set up a while loop until we hit the required target accuracy or the max. number of s
teps
  Z <- c()
  while ((i<MaxNumberOfIterations)&&(Deviation>Tolerance))
  {
    # Record the value of f(x) and f'(x), for the current x value.
    Xprime <- func(X)
    Z[1] <- Xprime[1]
    Z[2] <- Xprime[2]
    X_1 <- X - Z[1]/Z[2] #To draw line segment for Xn+1
    # I put them in a variable Z. Z[1]<-f(x); Z[2]<-f'(x)
    # To be safe, check that the function and it's derivative are defined at X (either c
ould be NaN if you are unlucky)
    if ((Z[1]=="NaN")||(Z[2]=="NaN")){
      cat("\nFunction or derivative not defined error.\n")
      break
    }
    if (DrawLines){
      Arrows(X,0,X,Z[1],col="blue",lty=2,arr.length=0.01, arr.type = "T")
      Arrows(X,Z[1],X_1,0,col="blue",lty=2,arr.length=0.01, arr.type = "T")
      }

    #Find the next X-value using Newton-Raphson's formula. Let's call that value X
    X <- X - Z[1]/Z[2]
    Y <- func(X)[1]
    # calculate Deviation<- |f(x)-0|
    Deviation <- abs(Z[1]-0)
    # increase the value of your iteration counter
    i <- i+1

    # if you like, have the program write out how it is getting on
    cat(paste("\nIteration ",i,":   X=",X,"  Y=",Y))

    # If you are feeling fancy, add some line segments to the screen to show where it ju
st went
    # See the 'fixed points' code for a reminder of how to do that.
    # output the result
    if (Deviation<Tolerance){
      cat(paste("\nFound the root point: ",X, " after ", i, "iterations"))
    }else{
      cat(paste("\nConvergence failure. Deviation: ",Deviation, "after ", i,    "iterati
```

```
ons"))}
  }


  # have the function return the answer
  return(X)
}
```

Function cos(x)-x

```
par(mfrow = c(1,2),oma = c(0,0,3,0))

#Secant Method
F1 <- function(z){
  return(cos(z)-z)
}
curve(cos(x)-x,-1,2,main="Secant:y=cos(x)-x")
Secant(F1,1,2,1e-3,40,1)
```

```
##
## Iteration  1 :   X= 0.765034682391819   Y= -0.0436763442286054
## Convergence failure. Deviation:  0.0436763442286054 after  1 iterations
## Iteration  2 :   X= 0.742299406864944   Y= -0.00538326126319721
## Convergence failure. Deviation:  0.00538326126319721 after  2 iterations
## Iteration  3 :   X= 0.739103270158936   Y= -3.03543288344699e-05
## Found the root point:  0.739103270158936  after  3 iterations
```

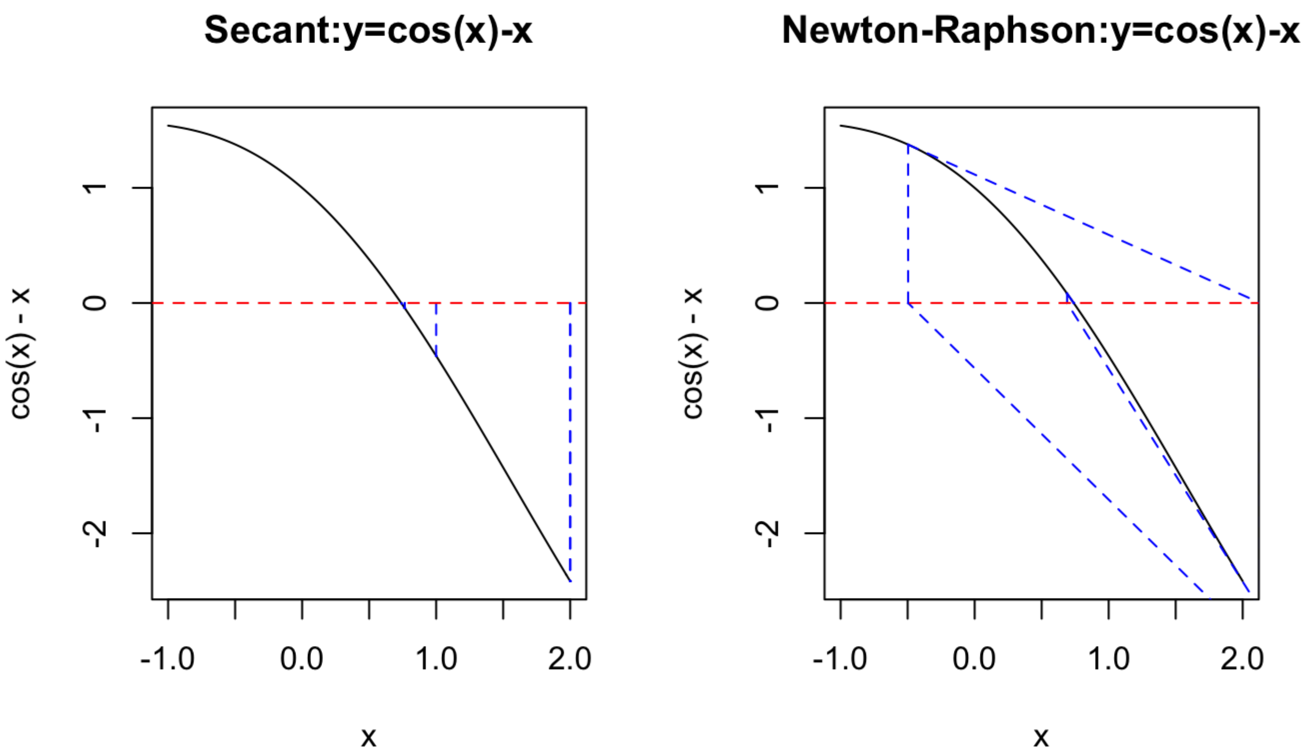
```
## [1] 0.7391033
```

```
abline(h=0, col="red", lty=2)

#Newton-Raphson function
F1_prime <- function(z){
  return(c(cos(z)-z, -sin(z)-1))
}
curve(cos(x)-x,-1,2,main="Newton-Raphson:y=cos(x)-x")
NewtonRaphson(F1_prime,3,1e-3,40,1)
```

```
##
## Iteration  1 :   X= -0.496558178297331   Y= 1.37578563617707
## Convergence failure. Deviation:  3.98999249660045 after  1 iterations
## Iteration  2 :   X= 2.131003844481   Y= -2.6623658513834
## Convergence failure. Deviation:  1.37578563617707 after  2 iterations
## Iteration  3 :   X= 0.689662720778373   Y= 0.0817979411125979
## Convergence failure. Deviation:  2.6623658513834 after  3 iterations
## Iteration  4 :   X= 0.739652997531334   Y= -0.000950503696277361
## Convergence failure. Deviation:  0.0817979411125979 after  4 iterations
## Iteration  5 :   X= 0.739085204375836   Y= -1.19095364348176e-07
## Found the root point:  0.739085204375836  after  5 iterations
```

```
## [1] 0.7390852
```

```
abline(h=0, col="red", lty=2)
```



Function log(x)-exp(-x)

```
par(mfrow = c(1,2),oma = c(0,0,3,0))
#Secant Method
F2 <- function(z){
  return(log(z)-exp(-z))
}
curve(log(x)-exp(-x),-1,2,main="Secant:y=log(x)-exp(-x)")
Secant(F2,1,2,1e-3,40,1)
```

```
##
## Iteration  1 :   X= 1.39741048216961   Y= 0.0873845096214802
## Convergence failure. Deviation:  0.0873845096214802 after  1 iterations
## Iteration  2 :   X= 1.28547612015065   Y= -0.0253897248274014
## Convergence failure. Deviation:  0.0253897248274014 after  2 iterations
## Iteration  3 :   X= 1.31067675808254   Y= 0.000906097784013626
## Found the root point:  1.31067675808254  after  3 iterations
```

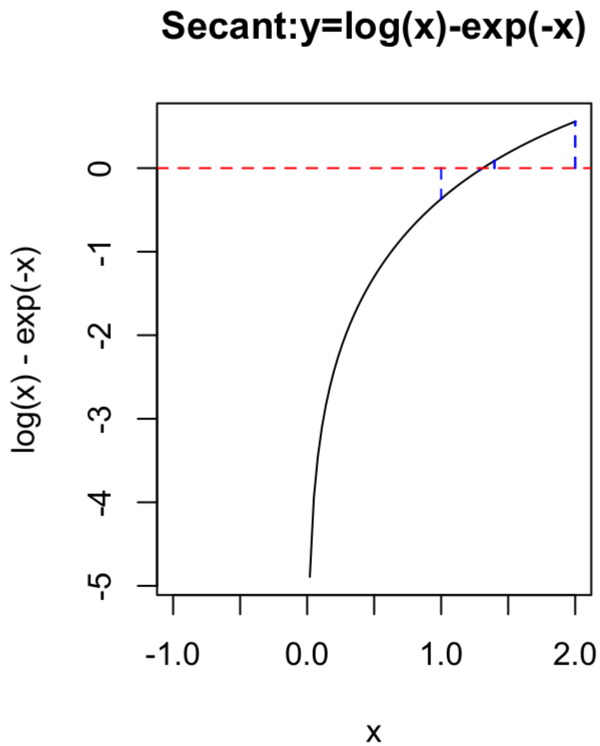
```
## [1] 1.310677
```

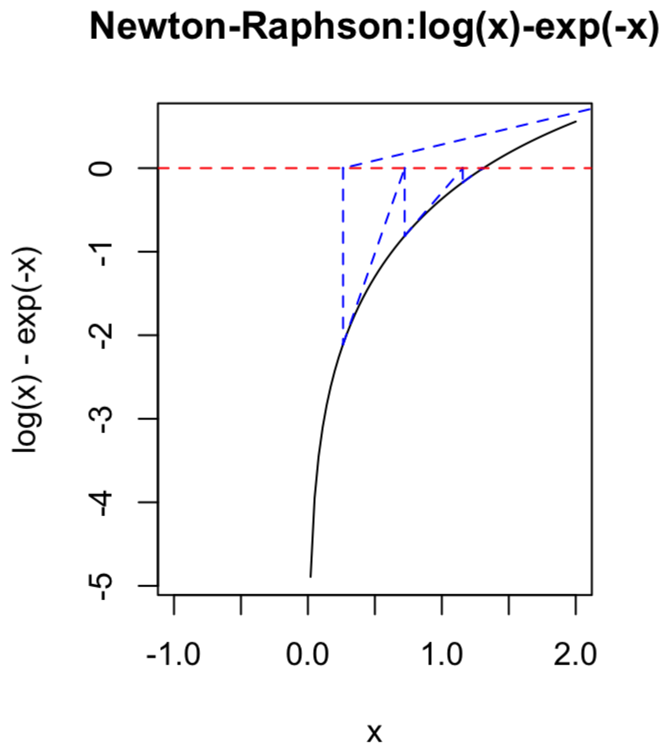
```
abline(h=0, col="red", lty=2)

#Newton-Raphson function
F2_prime <- function(z){
  return(c(log(z)-exp(-z), (1/z)+exp(-z)))
}
curve(log(x)-exp(-x),-1,2,main="Newton-Raphson:log(x)-exp(-x)")
NewtonRaphson(F2_prime,3,1e-3,40,1)
```

```
##
## Iteration  1 :   X= 0.262413550301494   Y= -2.10702644136201
## Convergence failure. Deviation:  1.04882522030025 after  1 iterations
## Iteration  2 :   X= 0.72246583637481   Y= -0.810638628545847
## Convergence failure. Deviation:  2.10702644136201 after  2 iterations
## Iteration  3 :   X= 1.1560315265697   Y= -0.16973967224609
## Convergence failure. Deviation:  0.810638628545847 after  3 iterations
## Iteration  4 :   X= 1.29990784429521   Y= -0.0102635365997112
## Convergence failure. Deviation:  0.16973967224609 after  4 iterations
## Iteration  5 :   X= 1.30975917924314   Y= -4.17548046200422e-05
## Convergence failure. Deviation:  0.0102635365997112 after  5 iterations
## Iteration  6 :   X= 1.30979958513046   Y= -6.96156243762402e-10
## Found the root point:  1.30979958513046  after  6 iterations
```

```
## [1] 1.3098
```

```
abline(h=0, col="red", lty=2)
```


Secant:y=log(x)-exp(-x)
log(x) - exp(-x)
0
-1
-2
-3
-4
-5
-1.0
0.0
1.0
2.0
x


Newton-Raphson:log(x)-exp(-x)
log(x) - exp(-x)
0
-1
-2
-3
-4
-5
-1.0
0.0
1.0
2.0
x