LichenDu_HW2b

LichenDu

2/17/2020

2b Secant method

Secant function

```
#StartingValue needs to be two numbers
Secant=function(func,StartingValue,Tolerance,MaxNumberOfIterations){
  i=0 #number of iterations
  X1=StartingValue[1]
  X2=StartingValue[2]
  Y1=func(X1)[1]
  Y2=func(X2)[1]
  Deviation=1000
  allx=c()
  ally=c()
  allslope=c()
  allx[1]=X1
  allx[2]=X2
  ally[1]=Y1
  ally[2]=Y2
  allslope[1]=func(X1)[2]
  allslope[2]=func(X2)[2]

    while ((i<MaxNumberOfIterations)&&(Deviation>Tolerance))
  {

      if ((Y2=="NaN")||(Y1=="NaN")){
      cat("Function not defined error")
      cat("\n",Y2,Y1)
      break
          }

    # Find the next (X1,X2)-value using Newton-Raphson's formula
    if(Y1==Y2){
    NewX=X2-Y2*(X2-X1)/(Y2-Y1+Tolerance)
    NewY=func(NewX)[1]
    Deviation=abs(NewY)}
      else{
        NewX=X2-Y2*(X2-X1)/(Y2-Y1)
      NewY=func(NewX)[1]
      Deviation=abs(NewY)
      }

    #update value of x1 and x2
    X1=X2
    Y1=func(X1)[1]
    X2=NewX
    Y2=func(X2)[1]

    # increase the value of your iteration counter
    i=i+1
    cat(paste("\nIteration ",i,":   X=",NewX,"  Y=",NewY))
    allx[i+2]=NewX
    ally[i+2]=NewY
    allslope[i+2]=func(NewX)[2]

    }
  # output the result
  if (Deviation<Tolerance){
    cat(paste("\nFound the root point: ",NewX, " after ", i, "iterations"))
  }else{
    cat(paste("\nConvergence failure. Deviation: ",Deviation, "after ", i,  "iterations"
))}

  # have the function return the answer
  df=cbind(allx,ally,allslope)
  return(df)
}
```

Secant Plot function

```
Secant_plot=function(func){
curve(func,lwd=5,xlim=c(min(df[,1]),max(df[,1])),ylim=c(min(df[,2]),max(df[,2])))
abline(h=0)
n=length(df[,1])
for (i in 1:(n-2)){
segments(df[i,1],0,df[i,1],df[i,2],lty=2,col="orange",lwd=2) #(x1,0) to (x1,y1)
segments(df[i,1],df[i,2],df[i+1,1],df[i+1,2],lty=2,col="red",lwd=2) #(x1,y1) to (x2,y2)
segments(df[i+1,1],df[i+1,2],df[i+2,1],0,lty=2,col="red",lwd=2) #(x2,y2) to (x3,0)
}
}
```

Newton-Raphson function

```
# Define your Newton-Raphson function
NewtonRaphson<-function(func,StartingValue,Tolerance,MaxNumberOfIterations){
  i=0
  X=StartingValue
  Y=func(X)[1]
  Deviation=abs(Y)
  allx=c()
  ally=c()
  allslope=c()
  allx[1]=X
  ally[1]=Y
  allslope[1]=func(X)[2]
  while ((i<MaxNumberOfIterations)&&(Deviation>Tolerance))
  { Z=c()
    Z[1]=func(X)[1]
    Z[2]=func(X)[2]
    if ((Z[1]=="NaN")||(Z[2]=="NaN")){
      cat("\nFunction or derivative not defined error.\n")
      break
    }
   #update X and Y
   X=X-Z[1]/Z[2]
   Y=func(X)[1]
   Deviation<-abs(Y)
   i<-i+1
   allx[i+1]=X
   ally[i+1]=Y
   allslope[i+1]=func(X)[2]
   cat(paste("\nIteration ",i,":   X=",X,"  Y=",Y))
  }
  if (Deviation<Tolerance){
    cat(paste("\nFound the root point: ",X, " after ", i, "iterations"))
  }else{
    cat(paste("\nConvergence failure. Deviation: ",Deviation, "after ", i,  "iterations"
))}
  df=cbind(allx,ally,allslope)
  return(df)
}
```

Newton plot function

```
Newton_plot=function(func){
curve(func,xlim=c(min(df[,1]),max(df[,1])),ylim=c(min(df[,2]),max(df[,2])),lwd=5)
abline(h=0)
n=length(df[,1])
for (i in 1:(n-1)){
segments(df[i,1],0,df[i,1],df[i,2],lty=2,col="orange",lwd=2) #(x1,0) to (x1,y1)
segments(df[i,1],df[i,2],df[i+1,1],0,lty=2,col="red",lwd=2) #(x1,y1) to (x2,0)
}
}
```

2 Functions

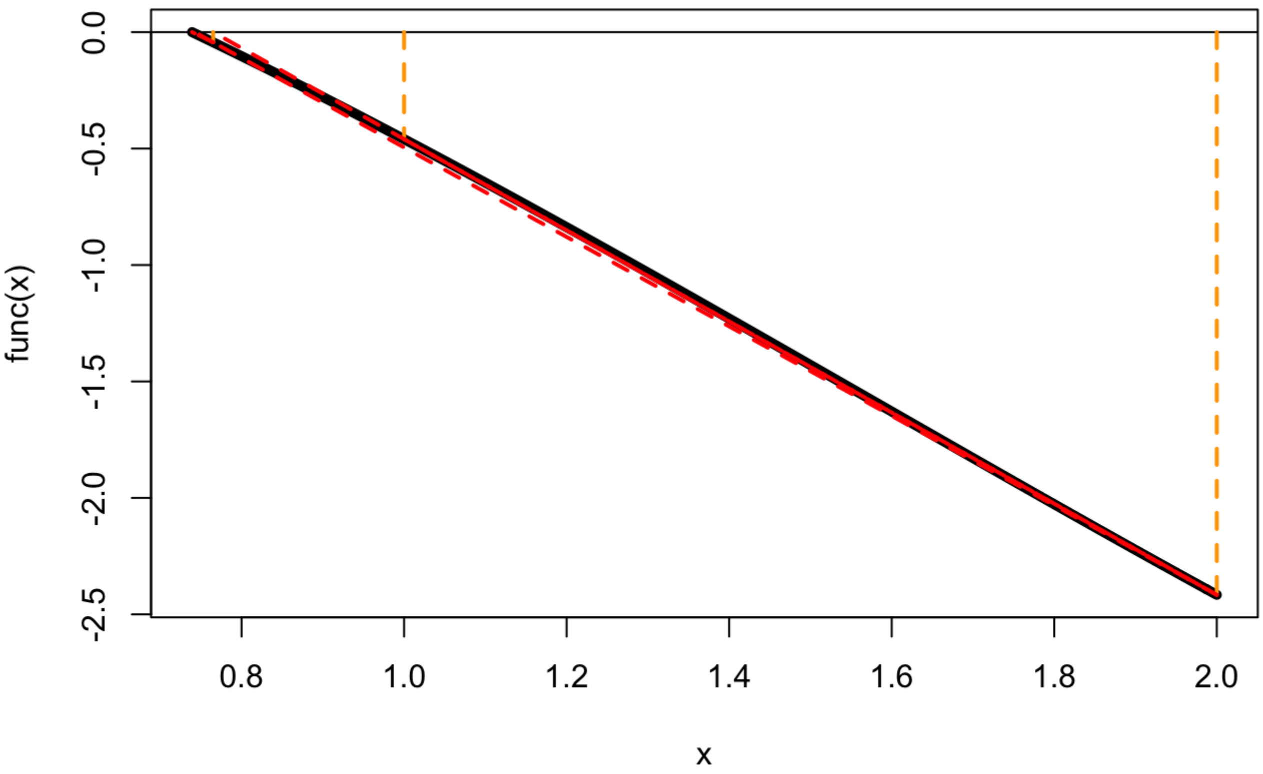
```
#Functions
F1=function(x){
  return(c(cos(x)-x,-sin(x)-1))}
F11=function(x){
  return(cos(x)-x)}

F2=function(x){
  return(c(log(x)-exp(-x),1/x+exp(-x)))}
F22=function(x){
  return(log(x)-exp(-x))}
```

Function1

```
df=Secant(F1,c(1,2),0.0005,200)
```

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

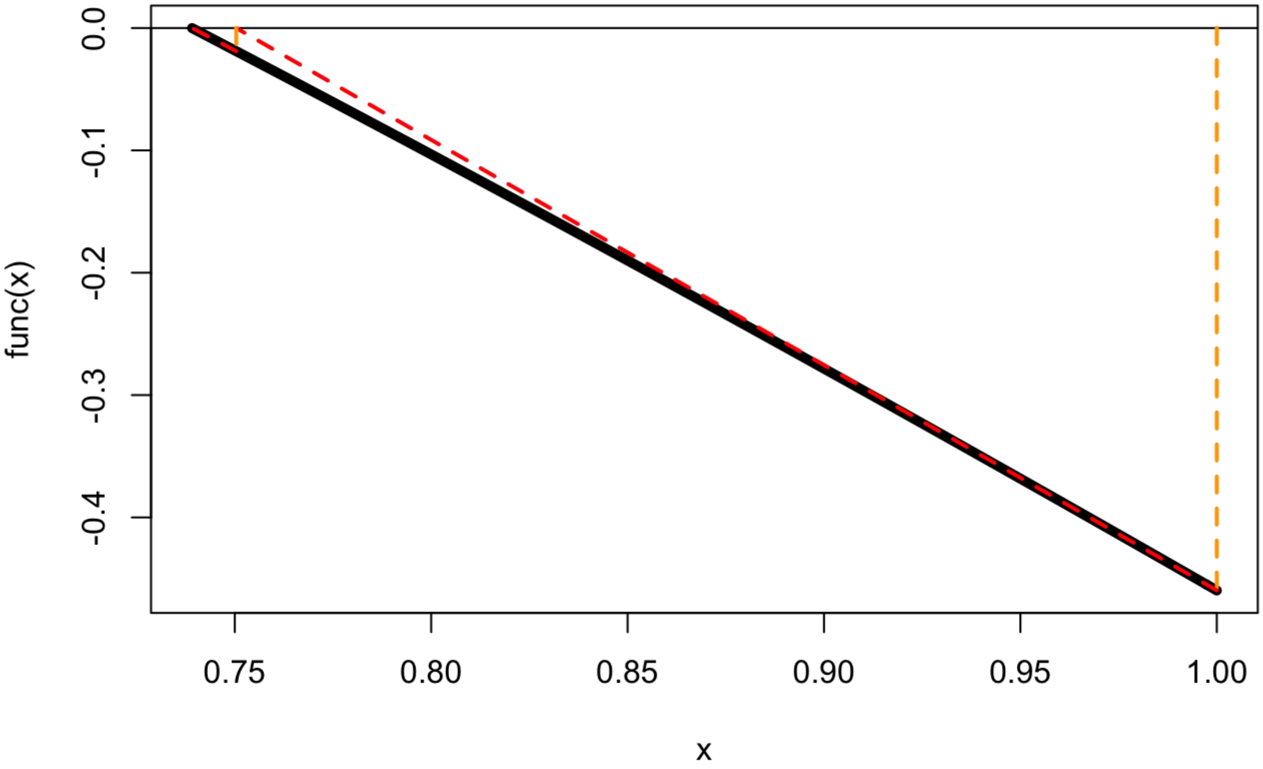
```
Secant_plot(F11)
```



```
df=NewtonRaphson(F1,1,0.0005,200)
```

```
##
## Iteration  1 :   X= 0.750363867840244   Y= -0.0189230738221174
## Iteration  2 :   X= 0.739112890911362   Y= -4.64558989908825e-05
## Found the root point:  0.739112890911362  after  2 iterations
```

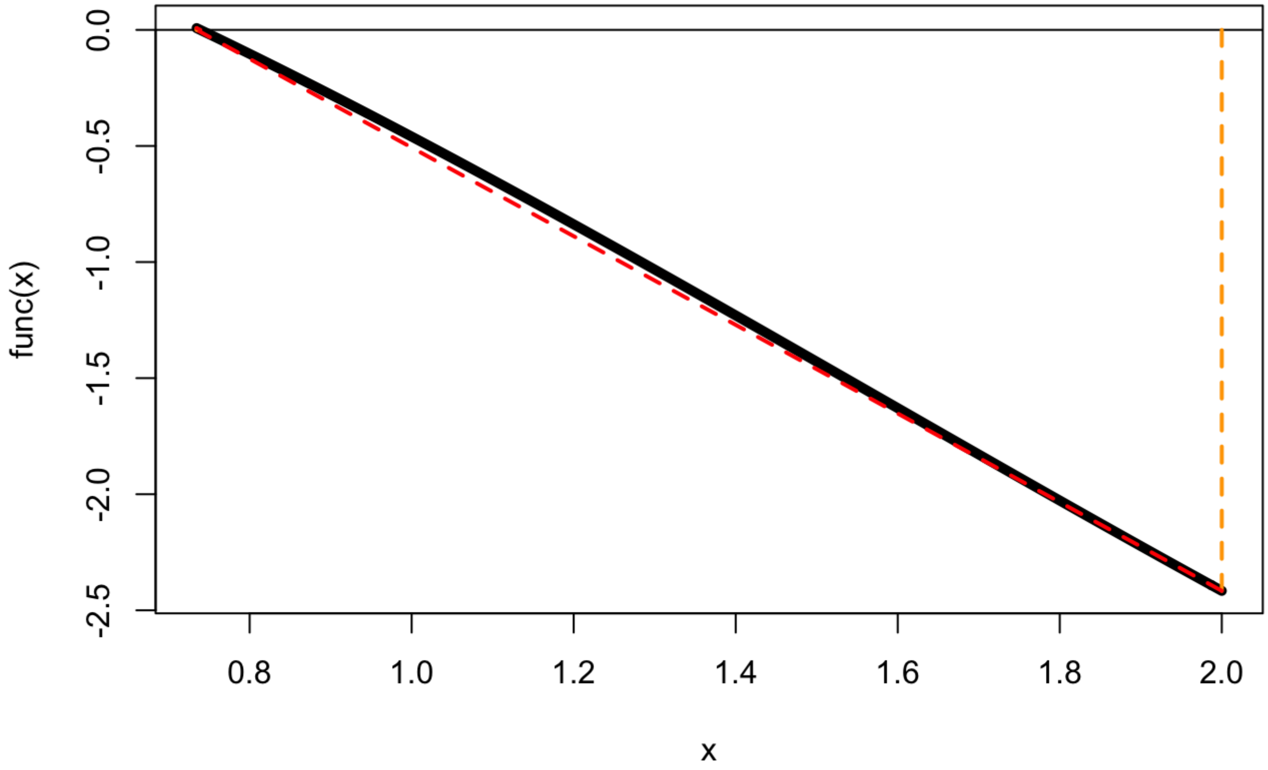
```
Newton_plot(F11)
```



```
df=NewtonRaphson(F1,2,0.0005,200)
```

```
##
## Iteration  1 :   X= 0.734536168854463   Y= 0.00760554394680923
## Iteration  2 :   X= 0.739089724205369   Y= -7.68354422797657e-06
## Found the root point:  0.739089724205369  after  2 iterations
```

```
Newton_plot(F11)
```

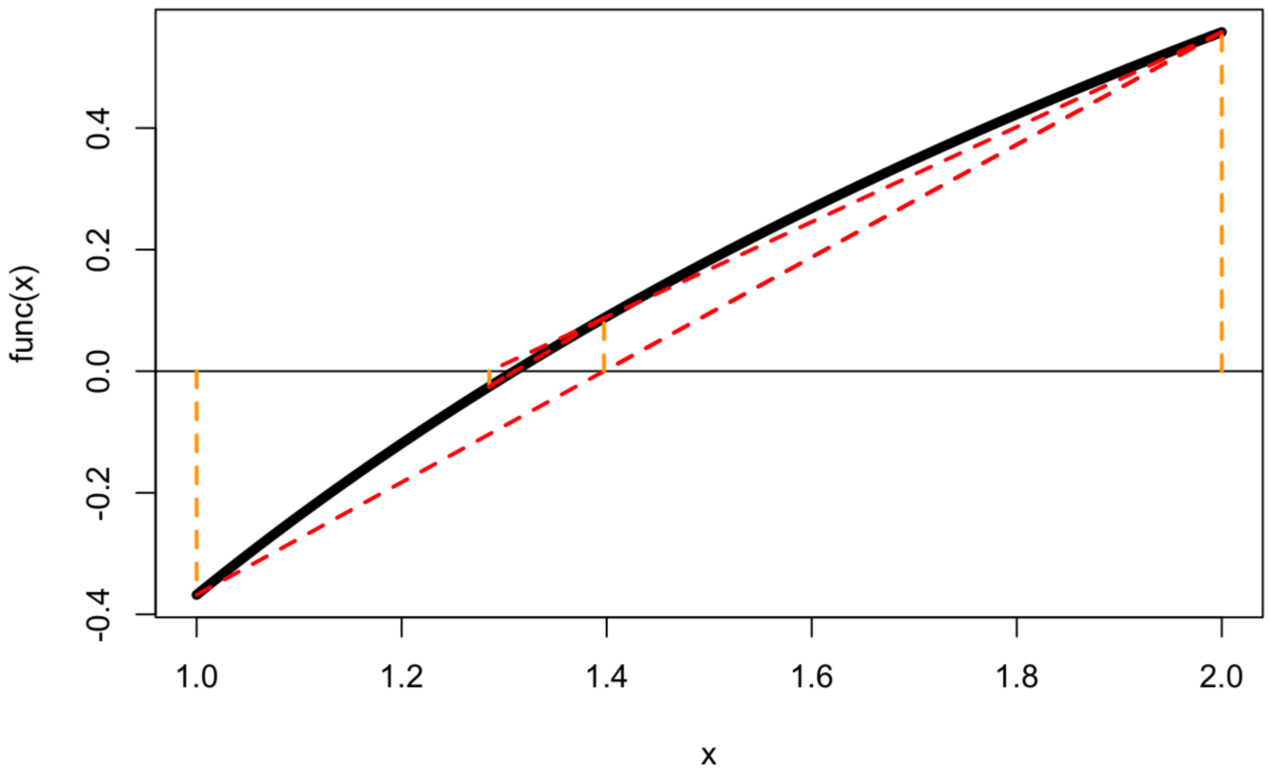


Function2

```
df=Secant(F2,c(1,2),0.0005,200)
```

```
##
## Iteration  1 :   X= 1.39741048216961   Y= 0.0873845096214802
## Iteration  2 :   X= 1.28547612015065   Y= -0.0253897248274014
## Iteration  3 :   X= 1.31067675808254   Y= 0.000906097784013626
## Iteration  4 :   X= 1.3098083980193   Y= 9.10606693577121e-06
## Found the root point:  1.3098083980193  after  4 iterations
```

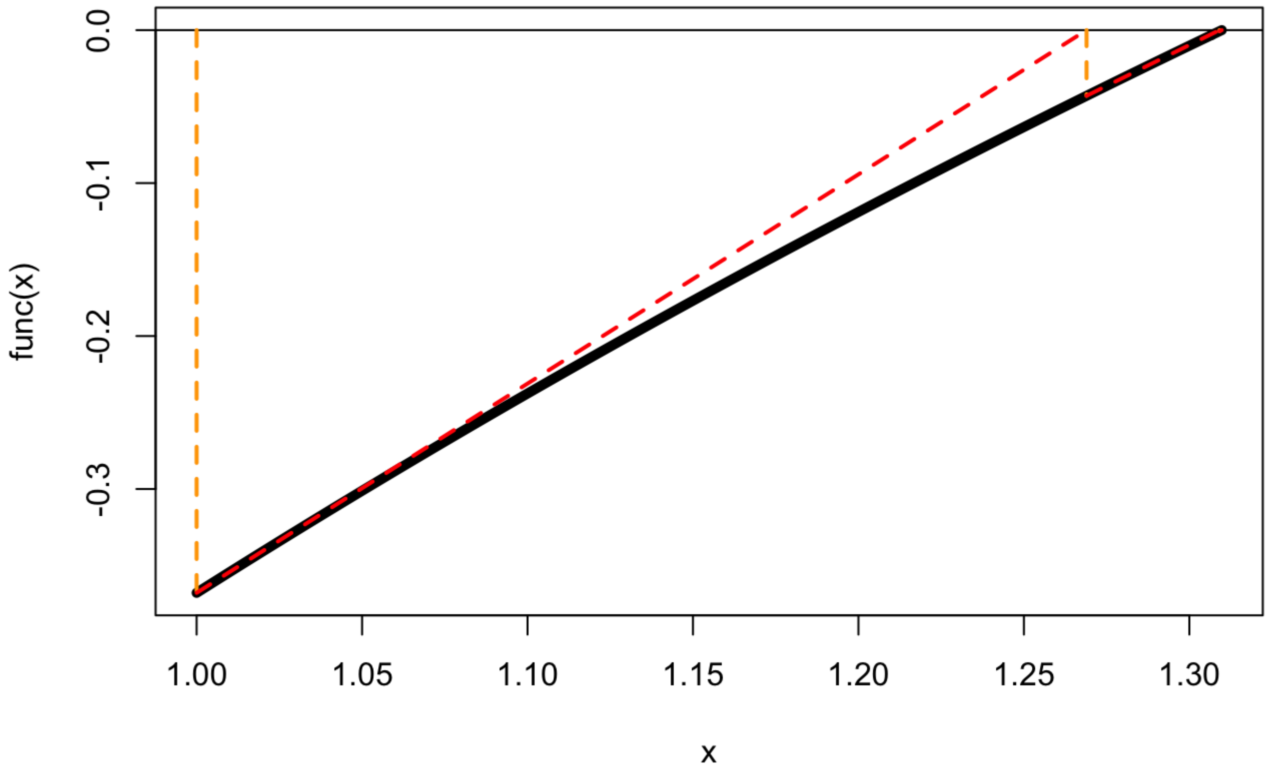
```
Secant_plot(F22)
```



```
df=NewtonRaphson(F2,1,0.0005,200)
```

```
##
## Iteration  1 :   X= 1.26894142137   Y= -0.0429460351219054
## Iteration  2 :   X= 1.30910840327402   Y= -0.000714437035072013
## Iteration  3 :   X= 1.30979938866897   Y= -2.03709596136026e-07
## Found the root point:  1.30979938866897  after  3 iterations
```

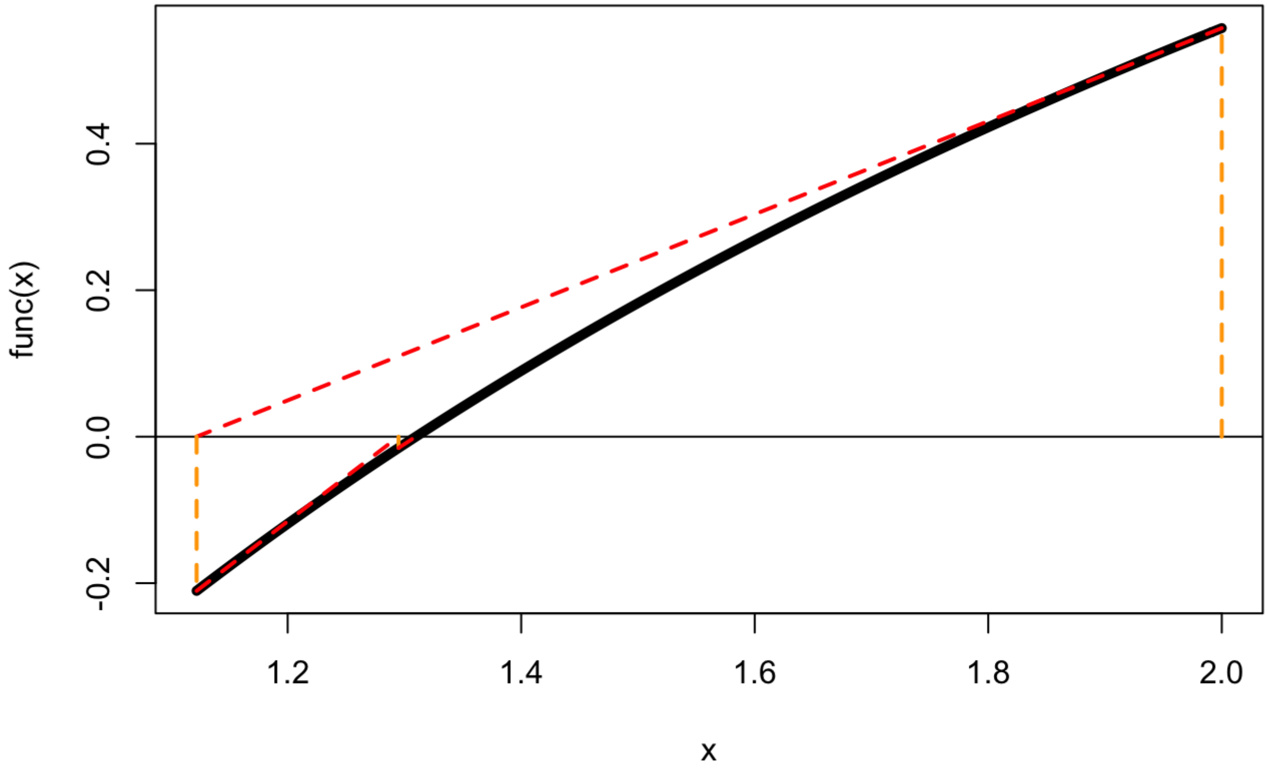
```
Newton_plot(F22)
```



```
df=NewtonRaphson(F2,2,0.0005,200)
```

```
##
## Iteration  1 :   X= 1.12201964530972   Y= -0.210491174024784
## Iteration  2 :   X= 1.29499697043904   Y= -0.0153903384035281
## Iteration  3 :   X= 1.30970906266486   Y= -9.35455546514641e-05
## Found the root point:  1.30970906266486  after  3  iterations
```

```
Newton_plot(F22)
```



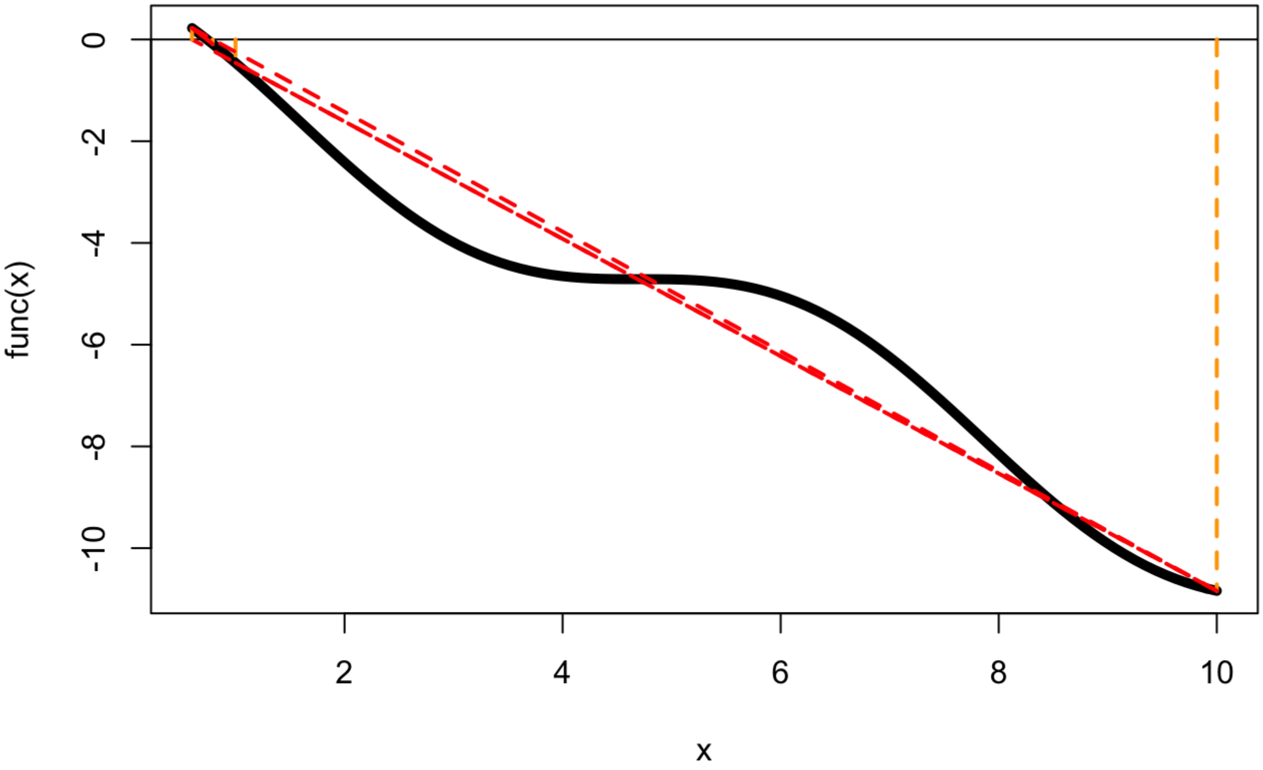
Comparison: On those two functions, NewtonRaphson method has less iterations and performs better than Secant method. But when I try function1 in another startvalue, the situation changes.

Function1 start from (1,10)

```
df=Secant(F1,c(1,10),0.0005,200)
```

```
##
## Iteration  1 :   X= 0.601394138704435   Y= 0.223153484463515
## Iteration  2 :   X= 0.790988152971929   Y= -0.0878451185362659
## Iteration  3 :   X= 0.737435151458587   Y= 0.00276042275426391
## Iteration  4 :   X= 0.73906671749562   Y= 3.08206444229464e-05
## Found the root point:  0.73906671749562  after  4 iterations
```

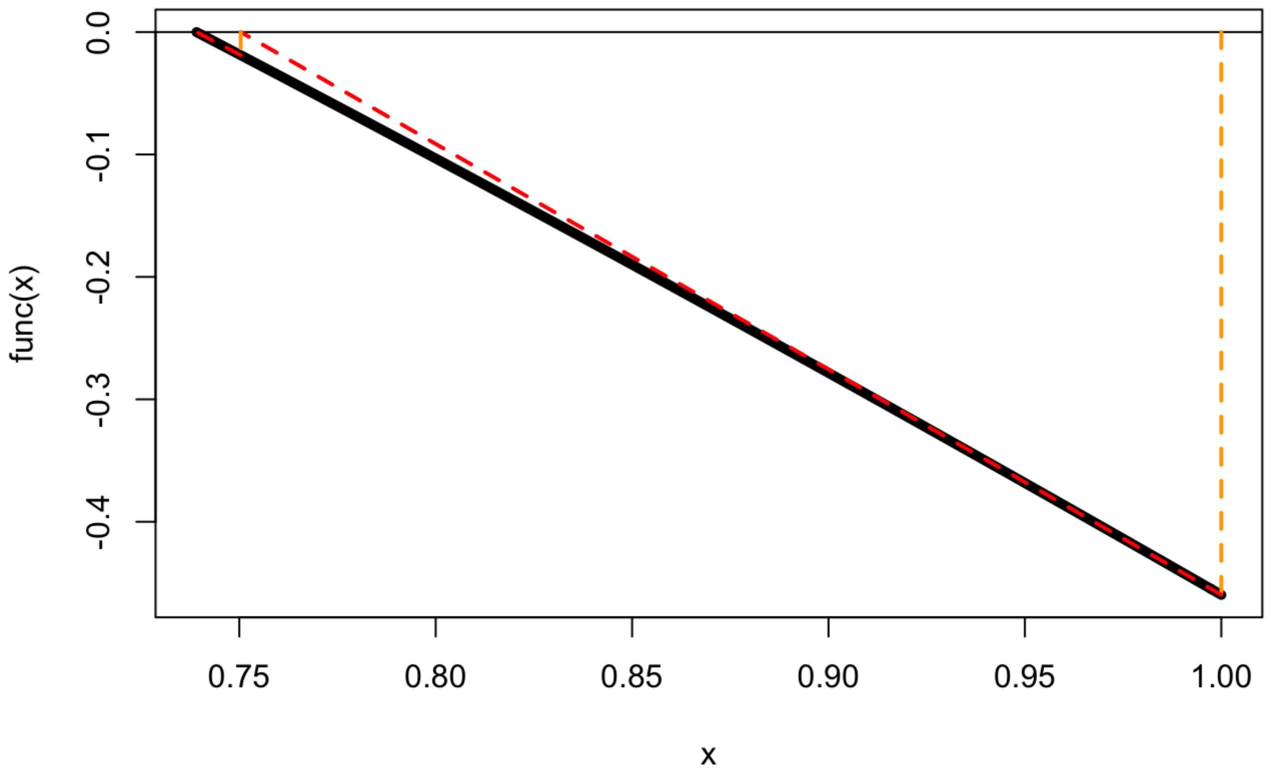
```
Secant_plot(F11)
```



```
df=NewtonRaphson(F1,1,0.0005,200)
```

```
##
## Iteration  1 :   X= 0.750363867840244   Y= -0.0189230738221174
## Iteration  2 :   X= 0.739112890911362   Y= -4.64558989908825e-05
## Found the root point:  0.739112890911362  after  2 iterations
```

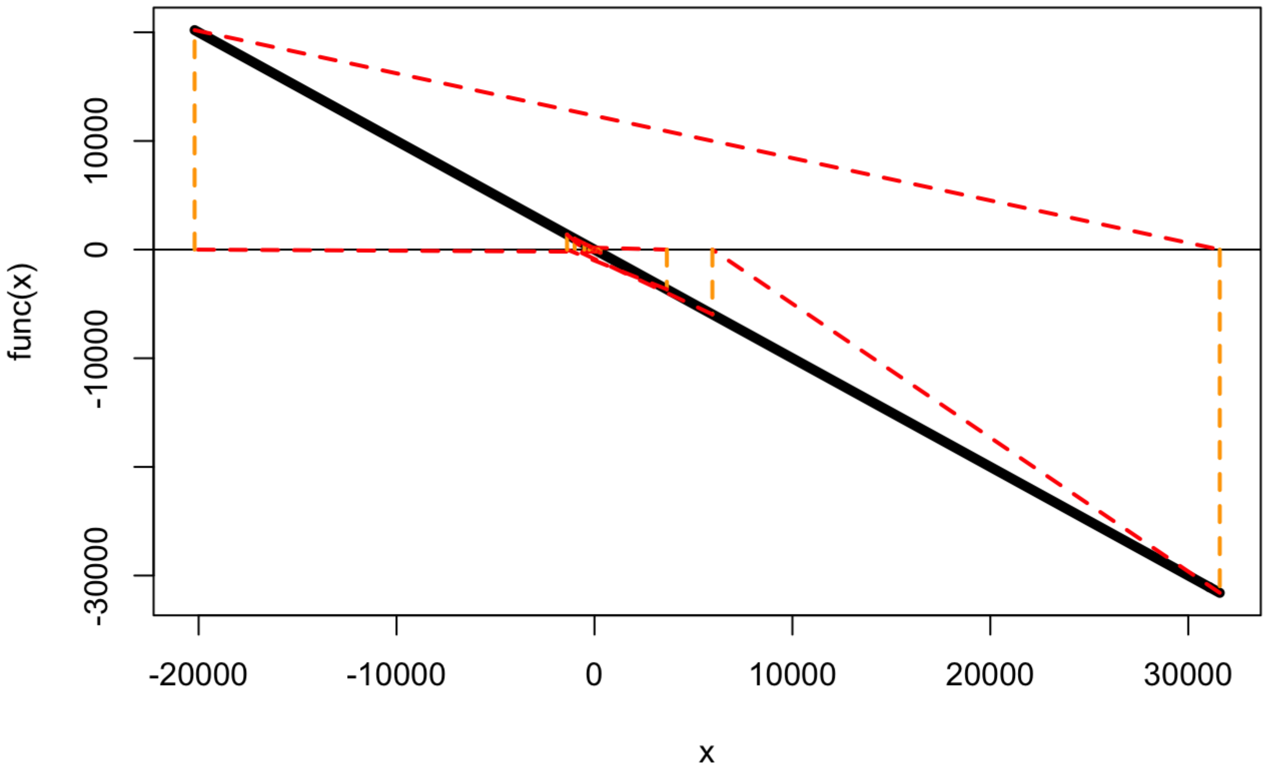
```
Newton_plot(F11)
```



```
df=NewtonRaphson(F1,10,0.0005,200)
```

```
##
## Iteration  1 :   X= -13.7709942015466   Y= 14.129038734246
## Iteration  2 :   X= 199.351177515312   Y= -199.490677048078
## Iteration  3 :   X= -20202.9263452617   Y= 20202.1339129731
## Iteration  4 :   X= 31592.0901722927   Y= -31591.1175295223
## Iteration  5 :   X= 5956.31501541367   Y= -5955.32545983105
## Iteration  6 :   X= -1002.07508213696   Y= 1001.07940114005
## Iteration  7 :   X= 101.456728081534   Y= -100.855503145095
## Iteration  8 :   X= 45.3972350979282   Y= -45.242006967144
## Iteration  9 :   X= 22.6382970893967   Y= -23.4361033295433
## Iteration  10 :   X= -36.3819126684131   Y= 36.6328003448928
## Iteration  11 :   X= -17.7678381269576   Y= 18.2376507741329
## Iteration  12 :   X= -8.08121253682765   Y= 7.85593206073693
## Iteration  13 :   X= 297.525172543404   Y= -298.126104671857
## Iteration  14 :   X= 131.835128829026   Y= -130.841367772618
## Iteration  15 :   X= -15.4308252371192   Y= 14.4689828144781
## Iteration  16 :   X= 4.48803858298975   Y= -4.71051166930108
## Iteration  17 :   X= -183.472540223593   Y= 183.778198780471
## Iteration  18 :   X= 3656.5351954684   Y= -3655.57376873632
## Iteration  19 :   X= -1386.06086979039   Y= 1385.24612623841
## Iteration  20 :   X= -509.223741720818   Y= 510.183197354437
## Iteration  21 :   X= 201.198989198996   Y= -200.208367139949
## Iteration  22 :   X= 25.0570154616502   Y= -24.0598812876757
## Iteration  23 :   X= -0.972053785069505   Y= 1.53565798723309
## Iteration  24 :   X= 7.85584872032523   Y= -7.8577158055912
## Iteration  25 :   X= 3.92698739351674   Y= -4.63409659545835
## Iteration  26 :   X= -11.8946772859372   Y= 12.6774462953494
## Iteration  27 :   X= -4.08024735264901   Y= 3.48937345298272
## Iteration  28 :   X= -2.14896380204296   Y= 1.60247362307541
## Iteration  29 :   X= 7.71031988446863   Y= -7.5671517905171
## Iteration  30 :   X= 3.90715462050577   Y= -4.62814768117285
## Iteration  31 :   X= -11.1654112620078   Y= 11.3344329333222
## Iteration  32 :   X= -5.45713037840447   Y= 6.13491207100543
## Iteration  33 :   X= -1.92169433525707   Y= 1.5779531015294
## Iteration  34 :   X= 23.9737318914406   Y= -23.5734843154778
## Iteration  35 :   X= -258.029520768668   Y= 258.943048266441
## Iteration  36 :   X= 178.472501556767   Y= -179.298807305592
## Iteration  37 :   X= 63.7742508448148   Y= -63.1864008543905
## Iteration  38 :   X= 28.8447592491274   Y= -29.686430606563
## Iteration  39 :   X= -35.6895729046694   Y= 35.2647714002484
## Iteration  40 :   X= -17.1806656904975   Y= 17.082729027274
## Iteration  41 :   X= -8.61872104323264   Y= 7.92637205374503
## Iteration  42 :   X= 19.8486392854786   Y= -19.3075658840571
## Iteration  43 :   X= 9.36095640258396   Y= -10.3589204981325
## Iteration  44 :   X= -0.376900700612104   Y= 1.30671046771872
## Iteration  45 :   X= 1.69081160087546   Y= -1.81053897238949
## Iteration  46 :   X= 0.782274473338603   Y= -0.0729623630941729
## Iteration  47 :   X= 0.739478651753245   Y= -0.000658654578396312
## Iteration  48 :   X= 0.739085167394271   Y= -5.72025703471368e-08
## Found the root point:  0.739085167394271  after  48 iterations
```

```
Newton_plot(F11)
```



Conclusion: I cannot draw a conclustion regarding on the performance of these two root-finding methods cause their performance depends. Both starting values and function curve will affect the number of iterations.