Zhao Ping HW2

Ping Zhao

2/10/2018

Problem 1

(a) Gradient Function

```
#x=(x1,x2,x3,x4), and t os the parameter (theta)
grad <- function (x, t) {
  dl = x[1]/(2+t)-(x[2]+x[3])/(1-t)+x[4]/t
}
```

(b)Stop your algorithm when either of the following criteria is satisfied:

```
secant <- function(maxit, x, t0, t1, tolerr, tolgrad){
  tstar = -1657/7680+sqrt(3728689)/7680
  digits = -log10(abs(t1 - tstar))
  it=0
  stop=0

  theta=c()
  converage.rate=c()
  digit=c()
  iteration=c()
  modified.relerr=c()
  Gradients=c()
  while (it < maxit & stop==0){
      it = it+1
      dl.1 = grad(x,t1)
      dl.0 = grad(x,t0)

      tnew = t1-dl.1*(t1-t0)/(dl.1-dl.0)


      rate = abs(tnew - tstar)/abs(t1 - tstar) #check convergent rate
      digits = -log10(abs(tnew - tstar)/abs(tstar))
      mod.relerr = abs(t1 - tnew)/max(1,abs(tnew))
      dl.new <-grad(x,tnew)

      if (mod.relerr < tolerr & abs(dl.new) < tolgrad) stop=1 # Stop iteration condition
      #print(c(it, t, rate,digits))
      #print(sprintf('it = %2.0f    teta = %12.12f     Rate = %4.2f      digits = %2.1f',it,tnew,rate,digi
      #print(sprintf('          relerr = %4.1e,        grad = %4.1e',relerr,dl))
      t0 = t1 # Update t0
      t1 = tnew # Update and return

      theta[it]<- tnew
      converage.rate[it]<- rate
```

```
      digit[it]<-digits
      iteration[it]<- it
      modified.relerr[it] <-mod.relerr
      Gradients[it]<-dl.new
    }
  conver.info<-as.data.frame(cbind(iteration, theta, converage.rate, digit, modified.relerr, Gradients))
  return(conver.info)
}
```

(c) &(d)

$\theta^{(0)} = .02$ and $\theta^{(1)} = .01$, and use tolerr = 1e-6 and tolgrad=1e-9

The iteration information is in matrix m. A larger matrix m.index contains information of starting points.

```
d <- c(1997,907,904,32)

#secant(20,x=d,0.02,0.01,1e-6,1e-9 )



m<-secant(20,x=d,0.02,0.01,1e-6,1e-9)

# Now, I will modify the out put digit format
m.dig1<-format(m[,3:6],digit=1)
m.t<-format(m$theta, digits = 12)
m.digit<-cbind.data.frame(m$iteration,m.t,m.dig1)

names(m.digit)[1:2] <- c("iteration","theta")

m.digit
```

```
##   iteration          theta converage.rate digit modified.relerr Gradients
## 1         1 0.0245618480110          4e-01   0.5           1e-02     4e+02
## 2         2 0.0278232129178          7e-01   0.7           3e-03     3e+02
## 3         3 0.0333510470016          3e-01   1.2           6e-03     7e+01
## 4         4 0.0351975582670          2e-01   1.9           2e-03     1e+01
## 5         5 0.0356461851804          6e-02   3.1           4e-04     8e-01
## 6         6 0.0356743090371          1e-02   5.0           3e-05     1e-02
## 7         7 0.0356746555712          7e-04   8.2           3e-07     7e-06
## 8         8 0.0356746558229          9e-06  13.2           3e-10     6e-11
```

The matrix above gives iteration information about the secant method.

However, I need to add information of the starting point θ_0 and θ_1 to the convergence matrix. As we are applying secant method, the θ_2 is from the first iteration. (I don't think this index system is good.) I added an column of θ index to the matrix.

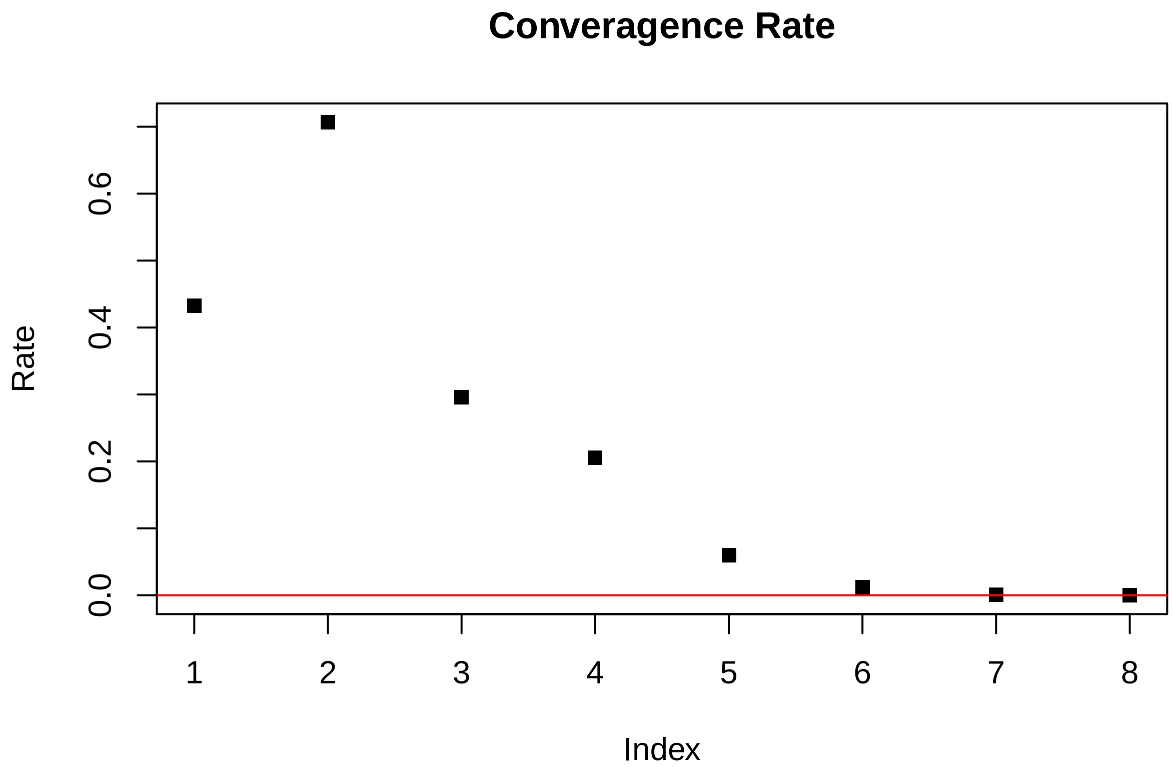
```
Index.Theta<-seq(0,nrow(m)+1)
tstar = -1657/7680+sqrt(3728689)/7680

#mod.relerr.0 = abs(0.02 - tnew)/max(1,abs(tnew))
mod.relerr.1 = abs(0.01 - 0.02)/max(1,abs(0.01))

rate.t1 = abs(0.01 - tstar)/abs(0.02 - tstar)
```

```
digits.1 = -log10(abs(0.01 - tstar)/abs(tstar))
digits.0 = -log10(abs(0.02 - tstar)/abs(tstar))

t0.info<-c(NA,0.02,NA,digits.0, NA, grad(d,0.02))
t1.info<-c(NA,0.01,rate.t1,digits.1,mod.relerr.1,grad(d,0.01))

m.starting<-rbind(t0.info,t1.info,m)
m.index<-cbind(Index.Theta,m.starting)

#Now, I will modify the output digits
m.in.dig<-format(m.index[,4:7],digit=1)
m.index.digit<-cbind(m.index[,1:3],m.in.dig)


m.index.digit
```

```
##    Index.Theta iteration      theta converage.rate digit modified.relerr
## 1            0        NA 0.02000000            NA   0.4              NA
## 2            1        NA 0.01000000         2e+00   0.1           1e-02
## 3            2         1 0.02456185         4e-01   0.5           1e-02
## 4            3         2 0.02782321         7e-01   0.7           3e-03
## 5            4         3 0.03335105         3e-01   1.2           6e-03
## 6            5         4 0.03519756         2e-01   1.9           2e-03
## 7            6         5 0.03564619         6e-02   3.1           4e-04
## 8            7         6 0.03567431         1e-02   5.0           3e-05
## 9            8         7 0.03567466         7e-04   8.2           3e-07
## 10           9         8 0.03567466         9e-06  13.2           3e-10
##    Gradients
## 1      7e+02
## 2      2e+03
## 3      4e+02
## 4      3e+02
## 5      7e+01
## 6      1e+01
## 7      8e-01
## 8      1e-02
## 9      7e-06
## 10     6e-11
```

(e)

```
rate.sq<-m$converage.rate^2

plot(m$converage.rate, ylab = "Rate", main = "Converagence Rate", pch=15)

abline(h=0, col=2)
```

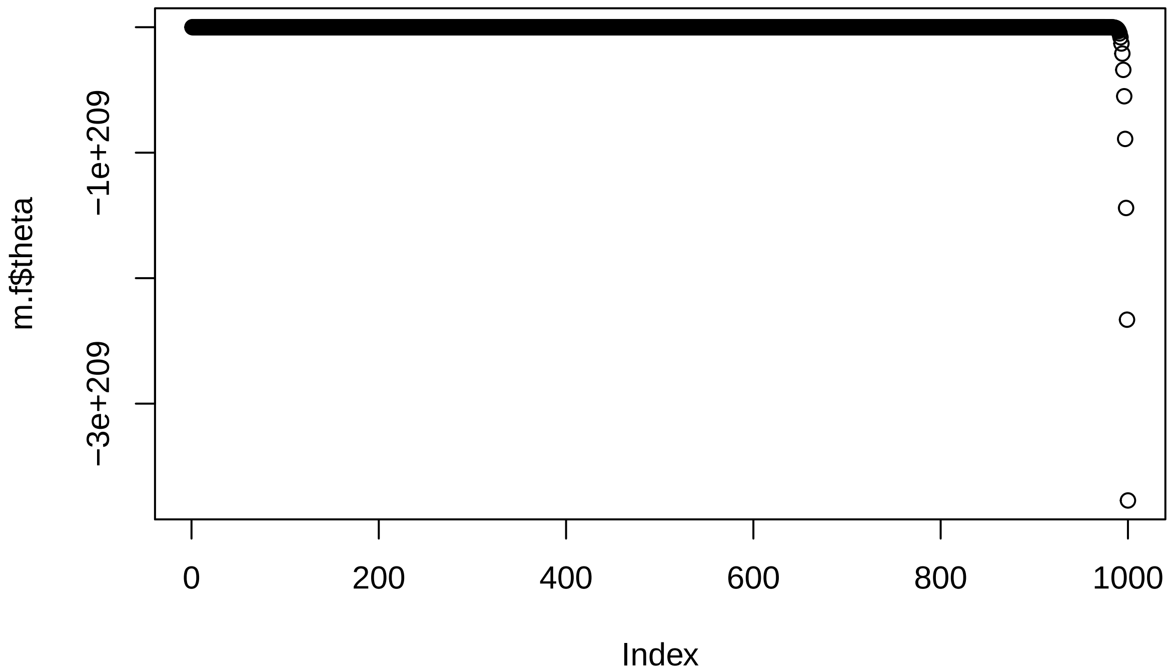


As we can see from the rate plot above, the rate is less than 1 and going to 0. Thus, it is supper-linear converagence.

(f)

$\theta^{(0)} = 0.5$ and $\theta^{(1)} = 0.01$

```
m.f<-secant(1000,x=d,0.5,0.01,1e-6,1e-6)
plot(m.f$theta)
```



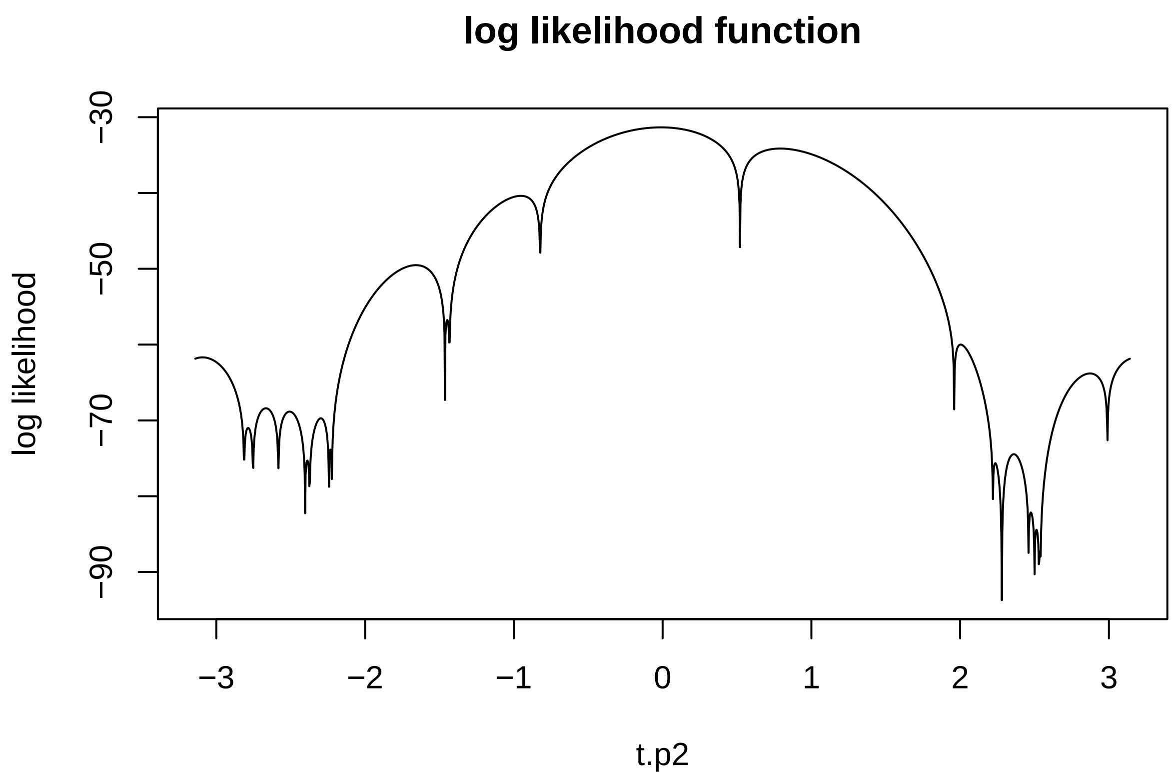
From the result above, we can see that the secant method doesn't converage in this case.

Problem 2

(a) log likelihood function

$$l = -20 * log 2\pi + \Sigma_1^{20} log(1 - cos(x_i - \theta))$$

```
d.2<-c(3.91, 4.85, 2.28, 4.06, 3.70, 4.04, 5.46, 3.53, 2.28, 1.96, 2.53,
       3.88, 2.22, 3.47, 4.82, 2.46, 2.99, 2.54, 0.52, 2.50)

# Defind l as the log likelihood function
l<-function(t){sum(log(1-cos(d.2-t)))-20*log(2*pi)}

t.p2<-seq(-pi,pi,length.out = 2000)
y=c()

for(i in 1:2000){y[i]<-l(t.p2[i])}


plot(t.p2,y, type="l", main = "log likelihood function", ylab = "log likelihood")
```



(b) MOM Estimator

The first moment is

$$E(X) = \int_0^{2\pi} x(1 - cos(x - \theta))/2\pi dx = sin\theta + \pi = \bar{x}$$

Solve for MOM estimatior

```
mom.est<-asin(mean(d.2)-pi)
```

(c) Newton–Raphson method

The Newton-Raphson Method is updating θ via:

$$\theta^{(n+1)} = \theta^{(n)} - \frac{l'(\theta^{(n)})}{l''(\theta^{(n)})}$$

```
grad.l<-function(x,t){
  sum(sin(t-x)/(1-cos(t-x)))
}

hess.2<-function(x,t){
  sum(cos(t-x)/(1-cos(t-x))-sin(t-x)^2/(1-cos(t-x))^2)
}


Newton <- function (maxit, x, t, tolerr, tolgrad) {
  stop=0
  it=0
```

```r
  iter=c()
  Theta=c()
  Gradi=c()
  Error=c()
  while (stop==0 & it < maxit) {
    it <- it +1


    dl = grad.1(x,t)
    ddl = hess.2(x,t)
    tnew = t - dl/ddl
    dl.new<-grad.1(x,tnew)
    iter[it]<-it
    Theta[it]<-tnew
    mod.err = abs(tnew - t)/max(1,abs(tnew))



    iter[it]<-it
    Theta[it]<-tnew
    Gradi[it]<- dl.new
    Error[it]<-mod.err

    #rate = abs(tnew - tstar)/abs(t - tstar) # Not knowing tstar here

    t = tnew



    if(mod.err<tolerr & abs(dl.new)<tolgrad) stop=1
  }

  Gradien<-format(Gradi,digits = 2)
  Errors<-format(Error,digits = 2)
  theta.12<-format(Theta, digits = 12)
  it.info<-cbind.data.frame(iter,Theta,theta.12, Gradien, Errors)
}
```

```r
start.mle<-Newton(20, d.2, mom.est, 1e-6, 1e-9)
start.left<-Newton(20, d.2, -2.7, 1e-6, 1e-9)
start.right<-Newton(20, d.2, 2.7, 1e-6, 1e-9)



start.mle # starting point is the MOM estimator
```

```
##   iter        Theta          theta.12  Gradien  Errors
## 1    1 -0.009098574 -0.00909857374527 -6.3e-02 6.8e-02
## 2    2 -0.011968738 -0.01196873791323 -7.2e-05 2.9e-03
## 3    3 -0.011972002 -0.01197200228331 -9.1e-11 3.3e-06
## 4    4 -0.011972002 -0.01197200228744 -8.3e-16 4.1e-12
```

```r
start.left # starting point is -2.7
```

```
##   iter     Theta       theta.12 Gradien  Errors
## 1    1 -2.674114 -2.67411365583 5.5e+00 9.7e-03
## 2    2 -2.666794 -2.66679392707 7.0e-02 2.7e-03
## 3    3 -2.666700 -2.66669992713 7.6e-07 3.5e-05
## 4    4 -2.666700 -2.66669992610 4.5e-13 3.9e-10
```

```r
start.right # starting point is 2.7
```

```
##   iter    Theta      theta.12  Gradien  Errors
## 1    1 2.825724 2.82572448457  1.0e+01 4.4e-02
## 2    2 2.877549 2.87754910830 -1.1e+00 1.8e-02
## 3    3 2.873184 2.87318445612 -2.3e-02 1.5e-03
## 4    4 2.873095 2.87309454904 -8.7e-06 3.1e-05
## 5    5 2.873095 2.87309451425 -1.3e-12 1.2e-08
```

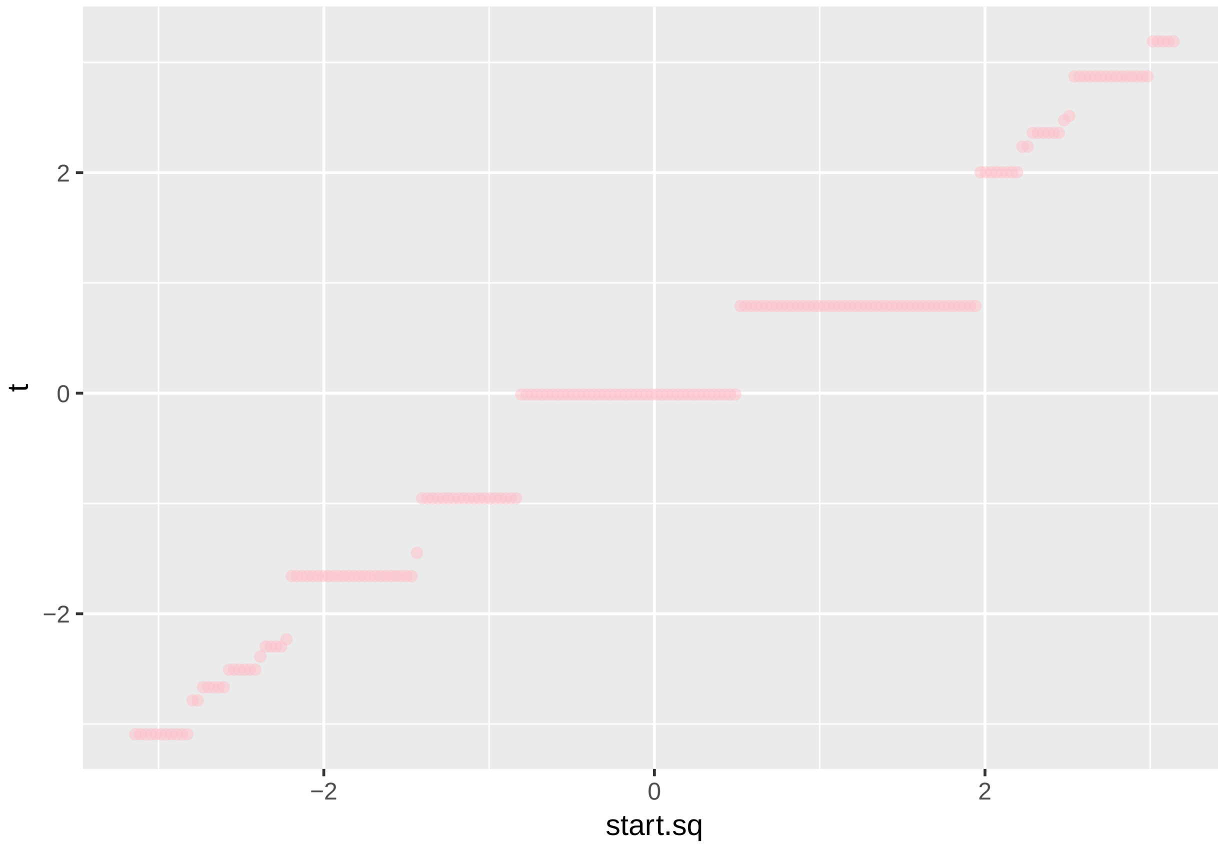
From the outcome above, we can see that with different starting points, the $\hat{\theta}$ converage to different value(local optimization).

(d)

```r
start.sq<-seq(-pi,pi,length.out = 200)

t=c()
Gr<-c()
for(i in 1:200){

  t[i]<-tail(Newton(20, d.2, start.sq[i], 1e-6, 1e-9)$Theta, n=1)
  Gr[i]<-tail(Newton(20, d.2, start.sq[i], 1e-6, 1e-9)$Gradi, n=1)
}

plot.data<-cbind.data.frame(start.sq,t,Gr)



require(ggplot2)
```

```
## Loading required package: ggplot2
```

```r
p <- ggplot(data = plot.data, mapping = aes(x = start.sq, y = t))

p+geom_point(shape=19, alpha=0.5, color="pink")
```



From the plot above, we can see that with different starting points, the approximations given by Newton Method converge to local optimization. Also, we can see that there is a 45 degree line trend between starting points and approximation. Therefore, starting points have impact on the final convergence for Newton-Method.

```r
# The following is to check different level with significant digits at 4

g <- factor(round(t, digits = 4))

xg <- split(t, g)

print(xg)
```

```
## $`-3.0931`
##  [1] -3.093092 -3.093092 -3.093092 -3.093092 -3.093092 -3.093092 -3.093092
##  [8] -3.093092 -3.093092 -3.093092 -3.093092
##
## $`-2.7862`
## [1] -2.786167 -2.786167
##
## $`-2.6667`
## [1] -2.6667 -2.6667 -2.6667 -2.6667 -2.6667
##
## $`-2.5076`
## [1] -2.507613 -2.507613 -2.507613 -2.507613 -2.507613 -2.507613
##
## $`-2.3882`
## [1] -2.3882
```

```
##
## $`-2.2973`
## [1] -2.297256 -2.297256 -2.297256 -2.297256
##
## $`-2.2322`
## [1] -2.232167
##
## $`-1.6583`
##  [1] -1.658283 -1.658283 -1.658283 -1.658283 -1.658283 -1.658283 -1.658283
##  [8] -1.658283 -1.658283 -1.658283 -1.658283 -1.658283 -1.658283 -1.658283
## [15] -1.658283 -1.658283 -1.658283 -1.658283 -1.658283 -1.658283 -1.658283
## [22] -1.658283 -1.658283 -1.658283
##
## $`-1.4475`
## [1] -1.447479
##
## $`-0.9533`
##  [1] -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363
##  [7] -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363
## [13] -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363
## [19] -0.9533363
##
## $`-0.012`
##  [1] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
##  [8] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
## [15] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
## [22] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
## [29] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
## [36] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
##
## $`0.7906`
##  [1] 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013
##  [8] 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013
## [15] 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013
## [22] 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013
## [29] 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013
## [36] 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013 0.7906013
## [43] 0.7906013 0.7906013 0.7906013 0.7906013
##
## $`2.0036`
## [1] 2.003645 2.003645 2.003645 2.003645 2.003645 2.003645 2.003645 2.003645
##
## $`2.2362`
## [1] 2.236219 2.236219
##
## $`2.3607`
## [1] 2.360718 2.360718 2.360718 2.360718 2.360718 2.360718
##
## $`2.4754`
## [1] 2.475374
##
## $`2.5136`
## [1] 2.513593
##
```

```
## $`2.8731`
##  [1] 2.873095 2.873095 2.873095 2.873095 2.873095 2.873095 2.873095
##  [8] 2.873095 2.873095 2.873095 2.873095 2.873095 2.873095 2.873095
## [15] 2.873095
##
## $`3.1901`
## [1] 3.190094 3.190094 3.190094 3.190094 3.190094
```

From the list output, we can see that there are 19 different approximation values according to different starting points (round to 4 digit)

(e) There are several pair of close starting points that go to different converagence. I take the 10th and 11th groups as example.

```
# Check the approximation of the 10th and 11th group

xg[[10]]
```

```
##  [1] -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363
##  [7] -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363
## [13] -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363 -0.9533363
## [19] -0.9533363
```

```
xg[[11]]
```

```
##  [1] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
##  [8] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
## [15] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
## [22] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
## [29] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
## [36] -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972 -0.011972
```

Above are the 10th (value=-0.9533363, length of 19) the 11th group (value=-0.011972, length of 42) group for level of approximation.

```
# Find the index of the flipping starting point from the 10th group to the 11th group



id<-sum(plot.data$t<= -0.9533363)
id
```

```
## [1] 74
```

```
start.sq[id]
```

```
## [1] -0.8367056
```

```
t[id]
```

```
## [1] -0.9533363
```

```
start.sq[id+1]
```

```
## [1] -0.8051318
```

```
t[id+1]
```

```
## [1] -0.011972
```

At the 74th, the starting point is -0.8367056, with the Newton approximation -0.9533363. While for the next starting point, the 75th, the starting point value is -0.8051318, but the Newton approximation coverages to

-0.011972.