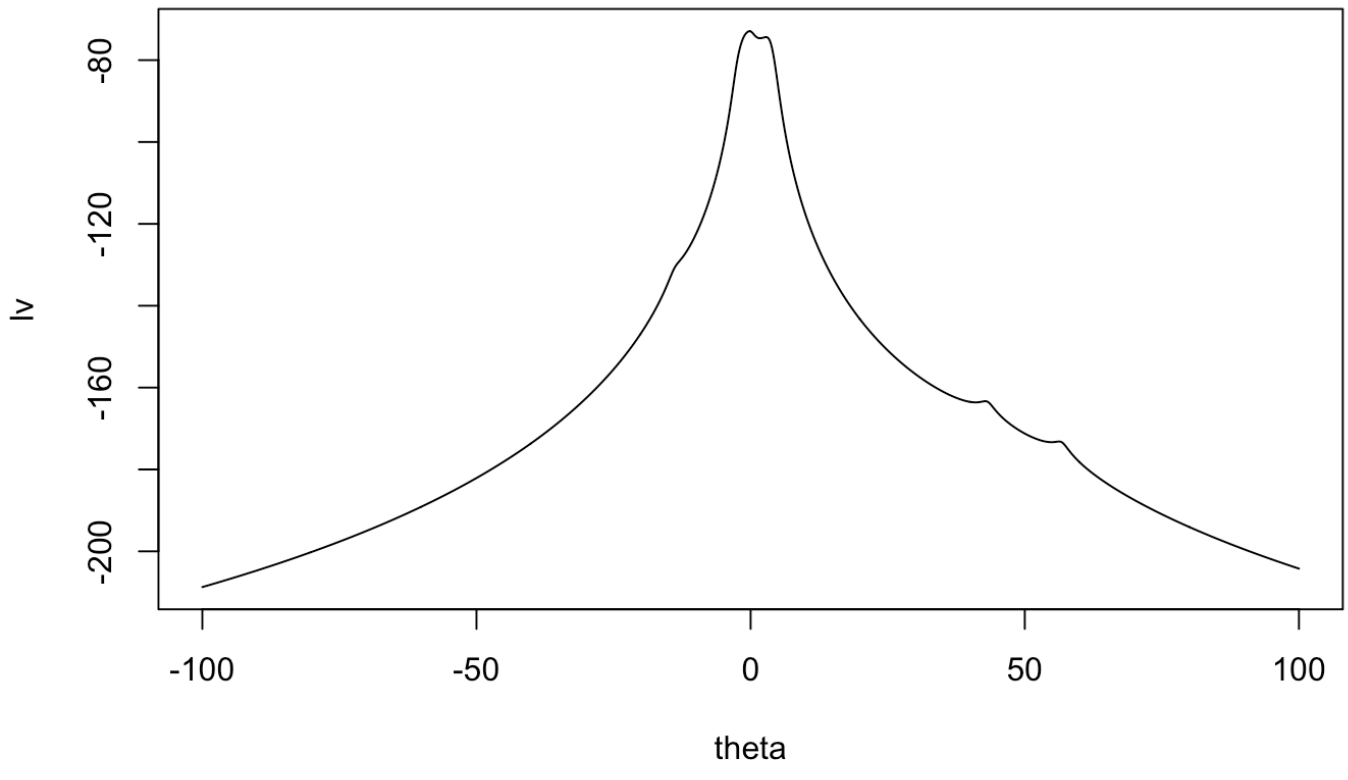
Computing_HW1

Code

Question 1-a

Hide

```
library("Rcpp")

# Compile C++ code and import function into R
current_path = rstudioapi::getActiveDocumentContext()$path
setwd(dirname(current_path))
sourceCpp("Computing_HW1_2020324011_민동은.cpp")

# 1-a: Graph log likelihood function
input = c(1.77, -0.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2
.40,
          4.53, -0.07, -1.05, -13.87,-2.53, -1.75, 0.27, 43.21)
theta = seq(-100,100,length=1000)
lv = theta
for(i in 1:length(theta)){
  lv[i] = g(input, theta[i])
}
plot(theta, lv, type="l")
```

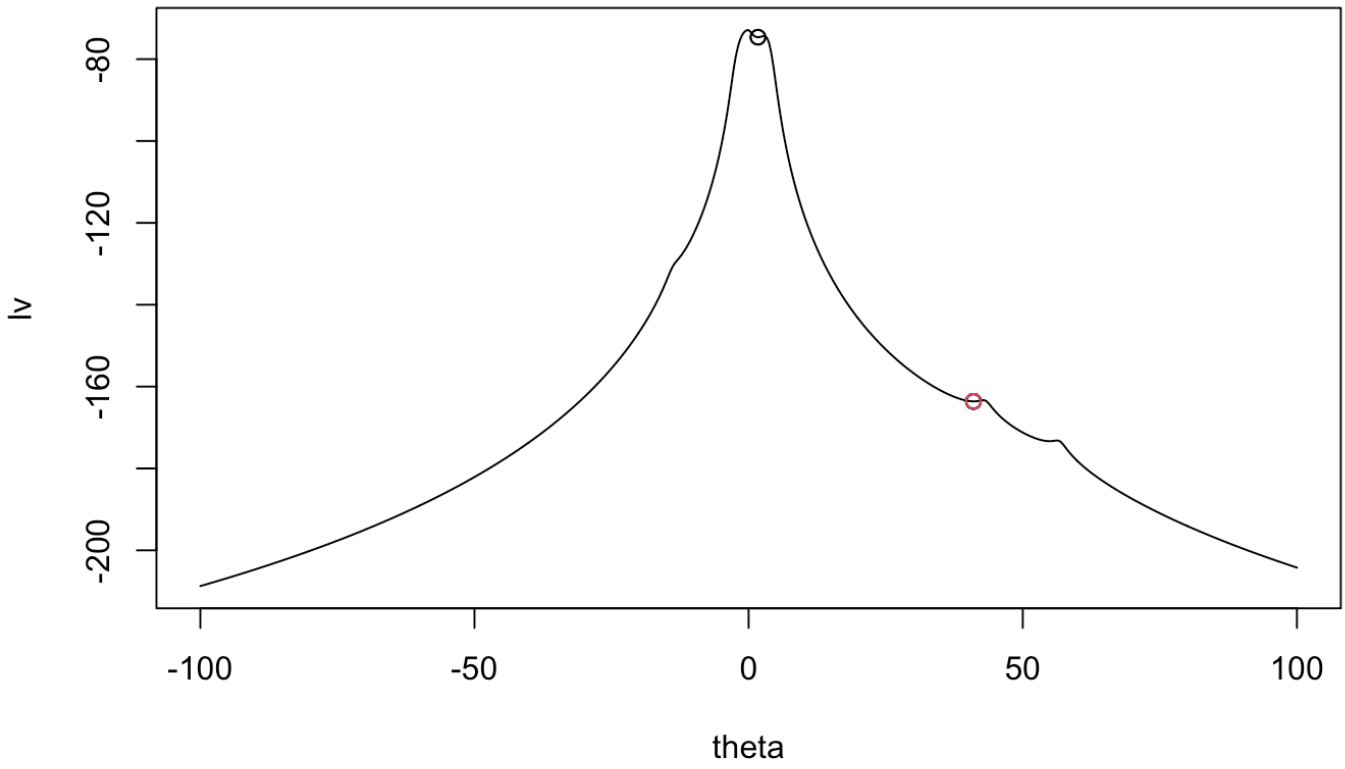


Hide

```
# the following plotting process is repeated in purpose of knitting the result
input = c(1.77, -0.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2
.40,
          4.53, -0.07, -1.05, -13.87,-2.53, -1.75, 0.27, 43.21)
theta = seq(-100,100,length=1000)
lv = theta
for(i in 1:length(theta)){
  lv[i] = g(input, theta[i])
}
plot(theta, lv, type="l")

#Newton-Raphson
starting = c(-11, -1, 0, 1.5, 4, 4.7, 7, 8, 38)
theta.sol = rep(NA,length(starting))
lval.sol = rep(NA,length(starting))
for(i in 1:length(starting)){
  theta = newton(input, starting[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = g(input, theta)
}
points(theta.sol,lval.sol)
```

Hide

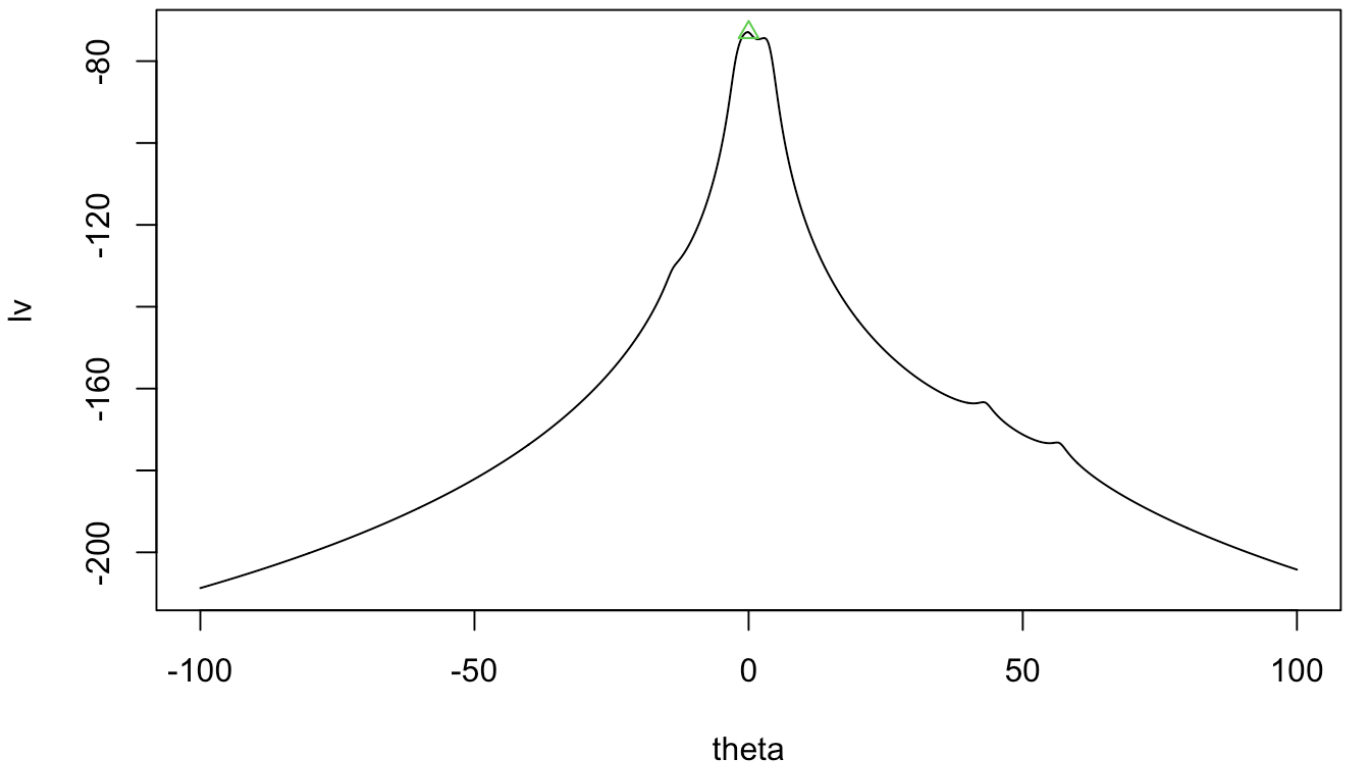
```
theta_m = newton(input, mean(input), 1e-2)
points(theta_m, g(input, theta_m), col=2)  # mean of the data isn't a good startin
g point. it doesn't find the global maximum
```



Question 1-b

Hide

```
# the following plotting process is repeated in purpose of knitting the result
input = c(1.77, -0.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2
.40,
          4.53, -0.07, -1.05, -13.87,-2.53, -1.75, 0.27, 43.21)
theta = seq(-100,100,length=1000)
lv = theta
for(i in 1:length(theta)){
  lv[i] = g(input, theta[i])
}
plot(theta, lv, type="l")

# 1-b: Bisection
start1 = -1
start2 = 1
theta = bisection(input, start1, start2)
points(theta, g(input, theta), pch=2, col=3)  # finds the global maximum
```



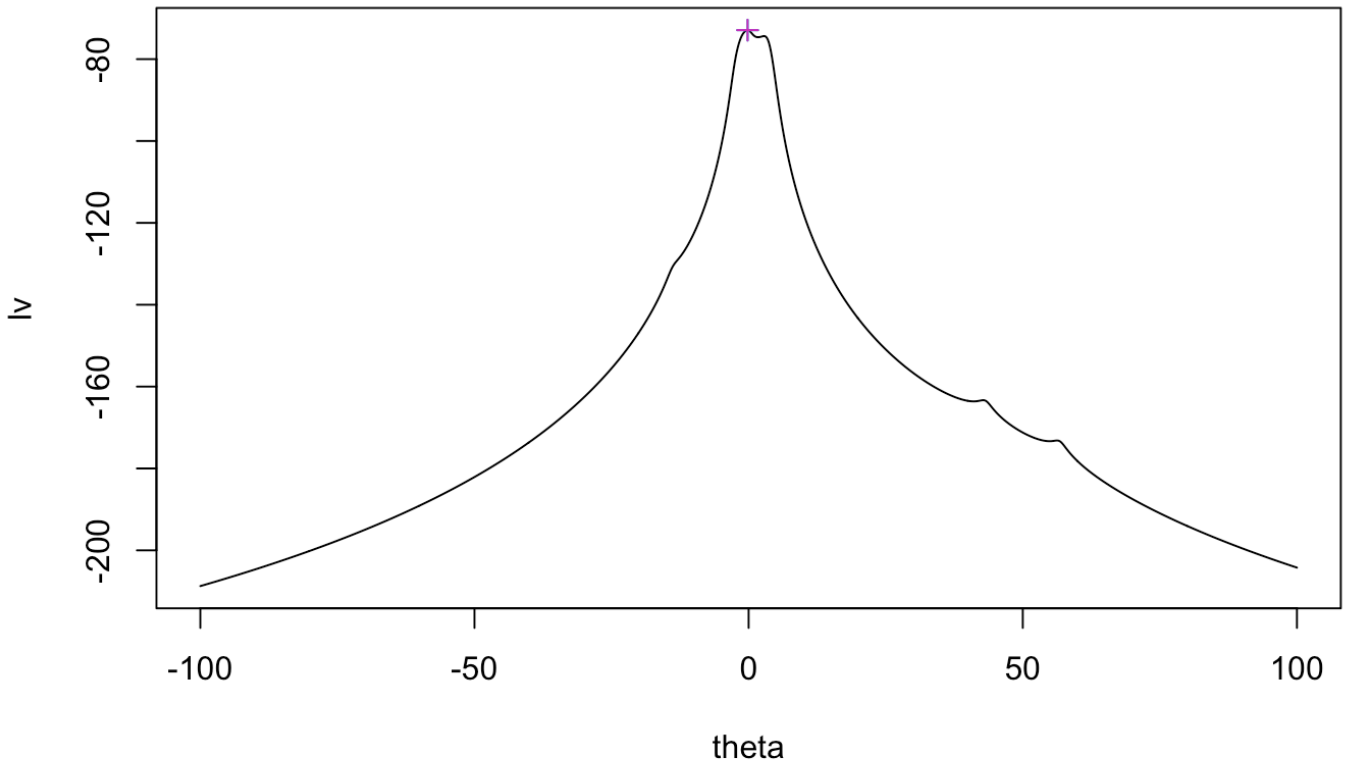
Question 1-c

Hide

```
# the following plotting process is repeated in purpose of knitting the result
input = c(1.77, -0.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2
.40,
          4.53, -0.07, -1.05, -13.87,-2.53, -1.75, 0.27, 43.21)
theta = seq(-100,100,length=1000)
lv = theta
for(i in 1:length(theta)){
  lv[i] = g(input, theta[i])
}
plot(theta, lv, type="l")

# 1-c: fixed-point iteration
start1 = -1
start2 = 0
start3 = 1
alpha = c(1, 0.64, 0.4, 0.3, 0.25)
theta.sol = rep(NA,length(alpha))
lval.sol = rep(NA,length(alpha))
for(i in 1:length(alpha)){
  theta = fixedpoint(input, start1, alpha[i])
  theta.sol[i] = theta
  lval.sol[i] = g(input, theta)
}
points(theta, g(input, theta),pch=3,col=4)  # finds the global maximum
```

Hide

```
for(i in 1:length(alpha)){
  theta = fixedpoint(input, start2, alpha[i])
  theta.sol[i] = theta
  lval.sol[i] = g(input, theta)
}
points(theta, g(input, theta),pch=3,col=5)  # finds the global maximum
for(i in 1:length(alpha)){
  theta = fixedpoint(input, start3, alpha[i])
  theta.sol[i] = theta
  lval.sol[i] = g(input, theta)
}
points(theta, g(input, theta),pch=3,col=6)  # finds the global maximum
```



Question 1-d

Hide

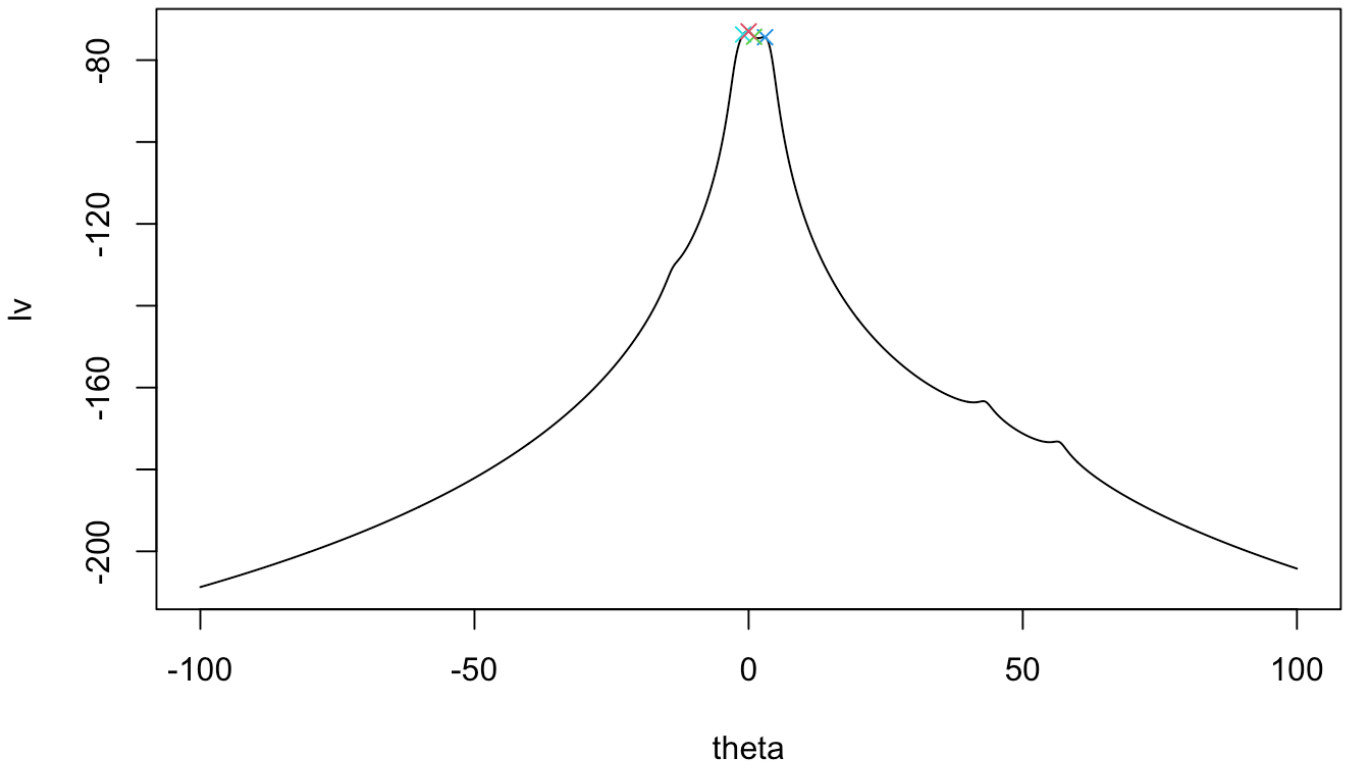
```
# the following plotting process is repeated in purpose of knitting the result
input = c(1.77, -0.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2
.40,
          4.53, -0.07, -1.05, -13.87,-2.53, -1.75, 0.27, 43.21)
theta = seq(-100,100,length=1000)
lv = theta
for(i in 1:length(theta)){
  lv[i] = g(input, theta[i])
}
plot(theta, lv, type="l")

# 1-d: secant method
start1 = c(-2, -1)
start2 = c(-3, 3)
start3 = c(-1, 1)
start4 = c(-2, 0)
theta = secant(input, start1[1], start1[2])
points(theta, g(input, theta),pch=4,col=5)
```

Hide

```
theta = secant(input, start2[1], start2[2])
points(theta, g(input, theta),pch=4,col=4)
theta = secant(input, start3[1], start3[2])
points(theta, g(input, theta),pch=4,col=3)
```

Hide

```
theta = secant(input, start4[1], start4[2])
points(theta, g(input, theta),pch=4,col=2)  # start4 finds the global maximum
```



Question 1-e

Hide

```
# the following plotting process is repeated in purpose of knitting the result
input = c(1.77, -0.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2
.40,
          4.53, -0.07, -1.05, -13.87,-2.53, -1.75, 0.27, 43.21)
theta = seq(-100,100,length=1000)
lv = theta
for(i in 1:length(theta)){
  lv[i] = g(input, theta[i])
}
plot(theta, lv, type="l")

# 1-e: compare methods - when finding the right global maximum
start.time <- Sys.time()
theta = newton(input, starting[5], 1e-2)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken  #for starting[5], 0.001935005 secs
```

```
Time difference of 0.01210403 secs
```

Hide

```
points(theta, g(input, theta), col=1)
```

Hide

```

start.time <- Sys.time()
theta = bisection(input, -1, 1)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken  #0.01088405 secs
```

```
Time difference of 0.01000881 secs
```

Hide

```
points(theta, g(input, theta), col=2)

start.time <- Sys.time()
theta = fixedpoint(input, 0, alpha[2])
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken  #0.002526999 secs
```

```
Time difference of 0.01562595 secs
```

Hide

Hide

```
points(theta, g(input, theta), col=3)
```

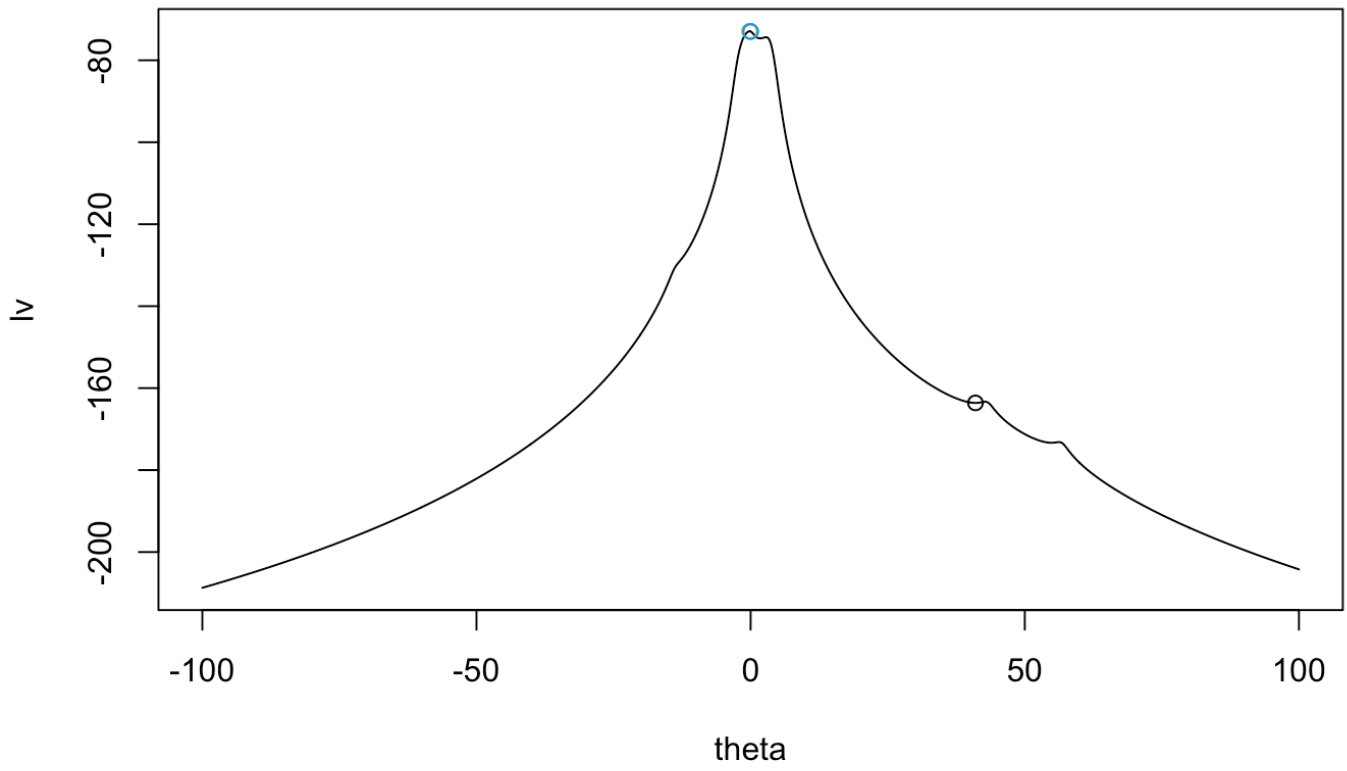
Hide

```
start.time <- Sys.time()
theta = secant(input, start4[1], start4[2])
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken  #0.0008690357 secs
```

```
Time difference of 0.006687164 secs
```

Hide

```
points(theta, g(input, theta), col=4)
```



[Conclusion]

The speed of each method varies by which starting point is used. The fixed point iteration method seems to be stabler, compared to other methods.

When given a right starting point, the speed relationship is: bisection < fixed point iteration < newton < secant, although theoretically, the newton method's converging speed is the fastest. The fact that the

example above is a relatively easy example, and the convergence speed between the newton method, fixed point iteration method, and secant method rarely differs must be considered.
Also, the secant method's result varies a lot depending on which starting point is used.

Hide

```
library("Rcpp")

# Compile C++ code and import function into R
sourceCpp("/Users/michelle/Computing_HW1_2020324011_민동은.cpp")

# (likelihood) plotting process
samples = rnorm(20)
theta = seq(-10,10,length=100)
lv = theta
for(i in 1:length(theta)){
  lv[i] = n(samples, theta[i])
}
plot(theta, lv, type="l")

start.time <- Sys.time()
theta = nnewton(samples, 0, 1e-2)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```

```
Time difference of 0.01424813 secs
```

Hide

```
points(theta, n(samples, theta), col=1)
```

Hide

```


start.time <- Sys.time()
theta = nbisection(samples, -1, 1)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```

```
Time difference of 0.01091909 secs
```

Hide

```
points(theta, n(samples, theta), col=2)

start.time <- Sys.time()
theta = nfixedpoint(samples, 0, alpha[2])
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```

```
Time difference of 0.006536007 secs
```

Hide

```
points(theta, n(samples, theta), col=3)
```

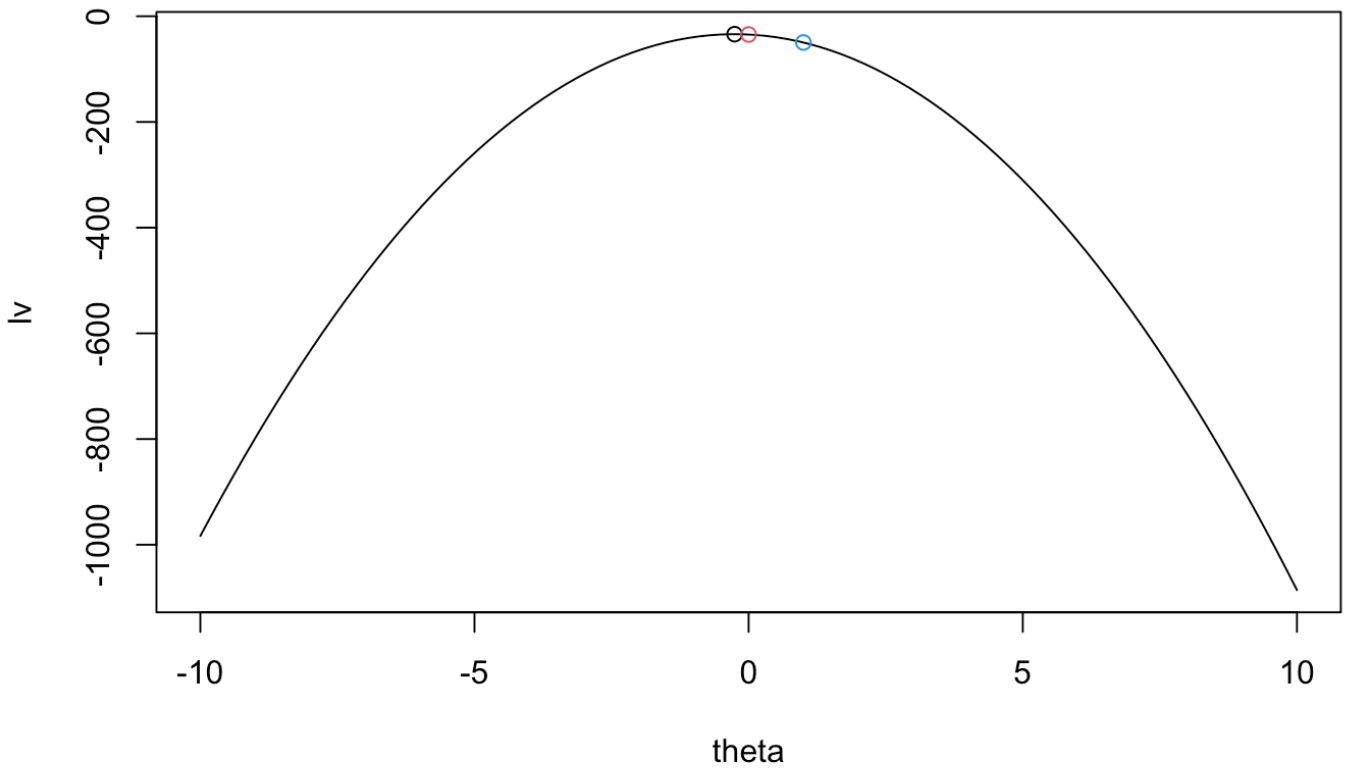
Hide

```


start.time <- Sys.time()
theta = nsecant(samples, -1, 1)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
```

```
Time difference of 0.00675106 secs
```

Hide

```
points(theta, n(samples, theta), col=4)
```



Since the normal distribution is uni-modal, it would be natural that it is easier and faster to find the global optimum with any method, including the slower or unstable methods such as the bisection method.

Question 2

Hide

```
## Graph the log likelihood function between -pi and pi
input = 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)
theta = seq(-pi, pi, length=200)
lv = theta
for(i in 1:length(theta)){
  lv[i] = f(input, theta[i])
}
plot(theta, lv, type="l")

## Find the mle for theta using the Newton-Raphson method
theta1 = fnewton(input, -0.0584, 1e-2)
print(theta1)
```

```
[1] -0.01197187
```

Hide

```
points(theta1, f(input, theta1))
```

Hide

```
theta2 = fnewton(input, -2.7, 1e-2)
print(theta2)
```

```
[1] -2.6667
```

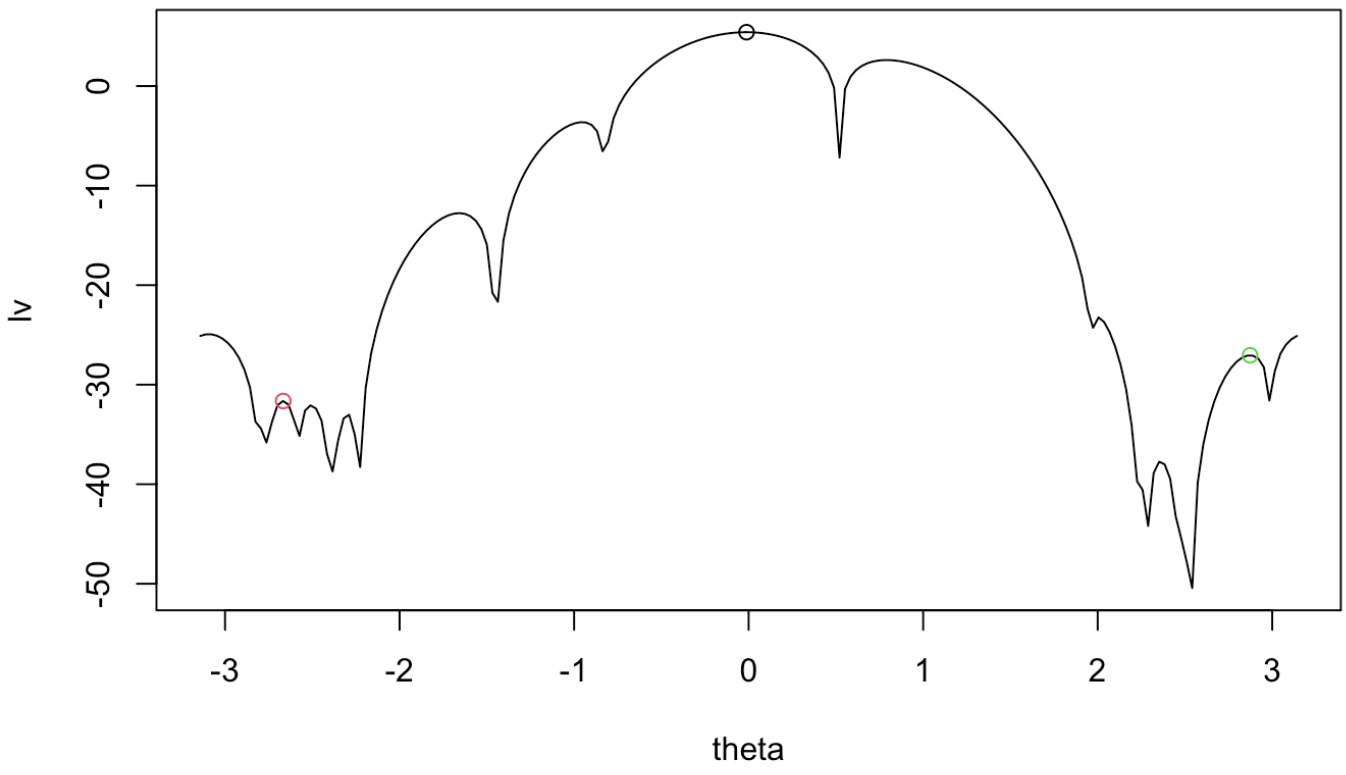
Hide

```
points(theta2, f(input, theta2), col=2)
theta3 = fnewton(input, 2.7, 1e-2)
print(theta3)
```

```
[1] 2.873095
```

Hide

```
points(theta3, f(input, theta3), col=3)
```



Hide

```
## the following plotting process is repeated in purpose of knitting the result
input = 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)
theta = seq(-pi, pi, length=200)
lv = theta
for(i in 1:length(theta)){
  lv[i] = f(input, theta[i])
}
plot(theta, lv, type="l")

## Repeat using 200 equally spaced starting values between -pi and pi
starting = seq(from = -pi, to = pi, length.out = 200)
theta.sol = rep(NA,length(starting))
lval.sol = rep(NA,length(starting))
for(i in 1:length(starting)){
  theta = fnewton(input, starting[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=1)
```

Hide

```
start_1 = starting[1:11]
theta.sol = rep(NA,length(start_1))
lval.sol = rep(NA,length(start_1))
for(i in 1:length(start_1)){
  theta = fnewton(input, start_1[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=2)

start_2 = starting[12:13]
theta.sol = rep(NA,length(start_2))
lval.sol = rep(NA,length(start_2))
for(i in 1:length(start_2)){
  theta = fnewton(input, start_2[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=3)
```

Hide

```
start_3 = starting[14]
theta = fnewton(input, start_3, 1e-2)
points(theta, f(input, theta), col=4)

start_4 = starting[15:19]
theta.sol = rep(NA,length(start_4))
lval.sol = rep(NA,length(start_4))
for(i in 1:length(start_4)){
  theta = fnewton(input, start_4[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=5)
```

Hide

```
start_5 = starting[20:25]
theta.sol = rep(NA,length(start_5))
lval.sol = rep(NA,length(start_5))
for(i in 1:length(start_5)){
  theta = fnewton(input, start_5[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=6)

start_6 = starting[26]
theta = fnewton(input, start_6, 1e-2)
points(theta, f(input, theta), col=7)
```

Hide

```
start_7 = starting[27:30]
theta.sol = rep(NA,length(start_7))
lval.sol = rep(NA,length(start_7))
for(i in 1:length(start_7)){
  theta = fnewton(input, start_7[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=8)

start_8 = starting[31]
theta = fnewton(input, start_8, 1e-2)
points(theta, f(input, theta), col=9)
```

Hide

```
start_9 = starting[32:55]
theta.sol = rep(NA,length(start_9))
lval.sol = rep(NA,length(start_9))
for(i in 1:length(start_9)){
  theta = fnewton(input, start_9[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=10)

start_10 = starting[56]
theta = fnewton(input, start_10, 1e-2)
points(theta, f(input, theta), col=11)
```

Hide

```
start_11 = starting[57:75]
theta.sol = rep(NA,length(start_11))
lval.sol = rep(NA,length(start_11))
for(i in 1:length(start_11)){
  theta = fnewton(input, start_11[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=12)

start_12 = starting[76:117]
theta.sol = rep(NA,length(start_12))
lval.sol = rep(NA,length(start_12))
for(i in 1:length(start_12)){
  theta = fnewton(input, start_12[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=13)
```

Hide

```
start_13 = starting[118:163]
theta.sol = rep(NA,length(start_13))
lval.sol = rep(NA,length(start_13))
for(i in 1:length(start_13)){
  theta = fnewton(input, start_13[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=14)

start_14 = starting[164:171]
theta.sol = rep(NA,length(start_14))
lval.sol = rep(NA,length(start_14))
for(i in 1:length(start_14)){
  theta = fnewton(input, start_14[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=15)
```

Hide

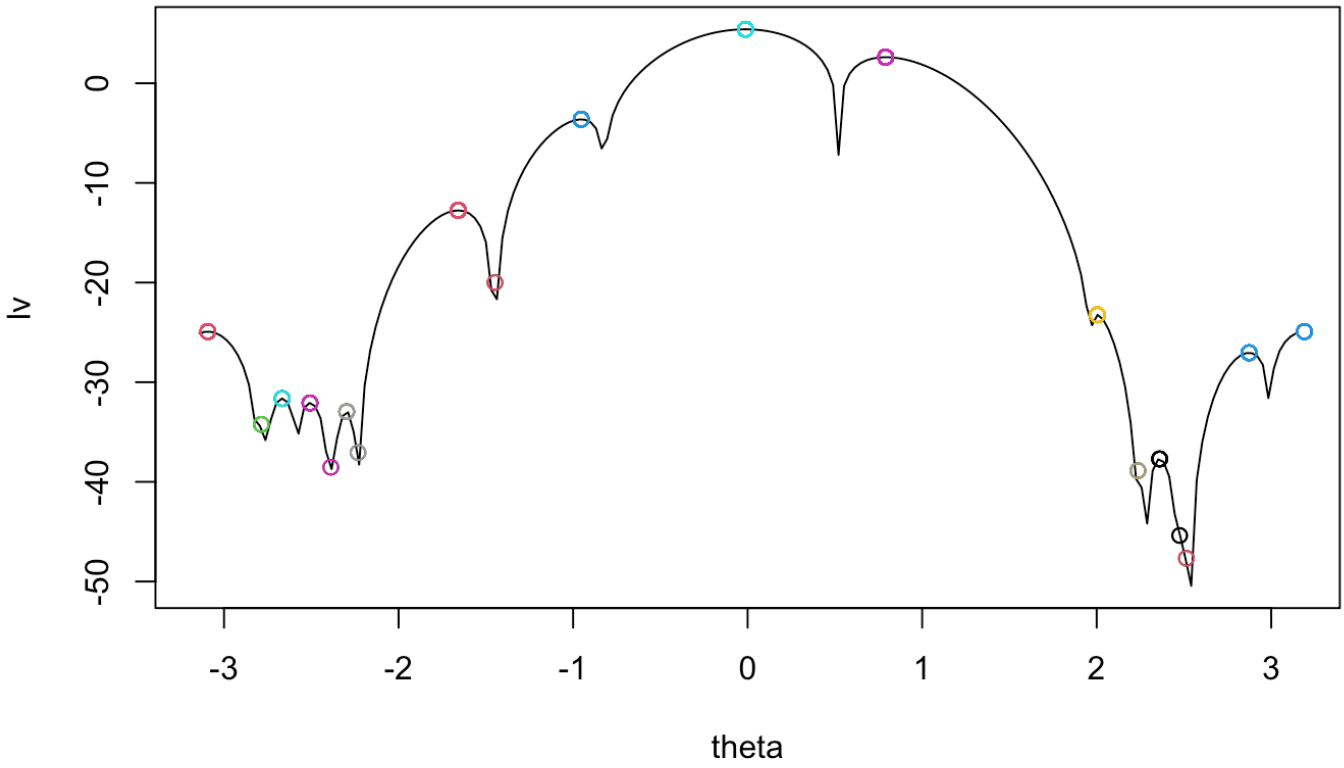
```
start_15 = starting[172:173]
theta.sol = rep(NA,length(start_15))
lval.sol = rep(NA,length(start_15))
for(i in 1:length(start_15)){
  theta = fnewton(input, start_15[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=16)

start_16 = starting[174:179]
theta.sol = rep(NA,length(start_16))
lval.sol = rep(NA,length(start_16))
for(i in 1:length(start_16)){
  theta = fnewton(input, start_16[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=17)
```

Hide

```
start_17 = starting[180]
theta = fnewton(input, start_17, 1e-2)
points(theta, f(input, theta), col=18)

start_18 = starting[181]
theta = fnewton(input, start_18, 1e-2)
points(theta, f(input, theta), col=19)
```

Hide

```
start_19 = starting[182:200]
theta.sol = rep(NA,length(start_19))
lval.sol = rep(NA,length(start_19))
for(i in 1:length(start_19)){
  theta = fnewton(input, start_19[i], 1e-2)
  theta.sol[i] = theta
  lval.sol[i] = f(input, theta)
}
points(theta.sol, lval.sol, col=20)
```

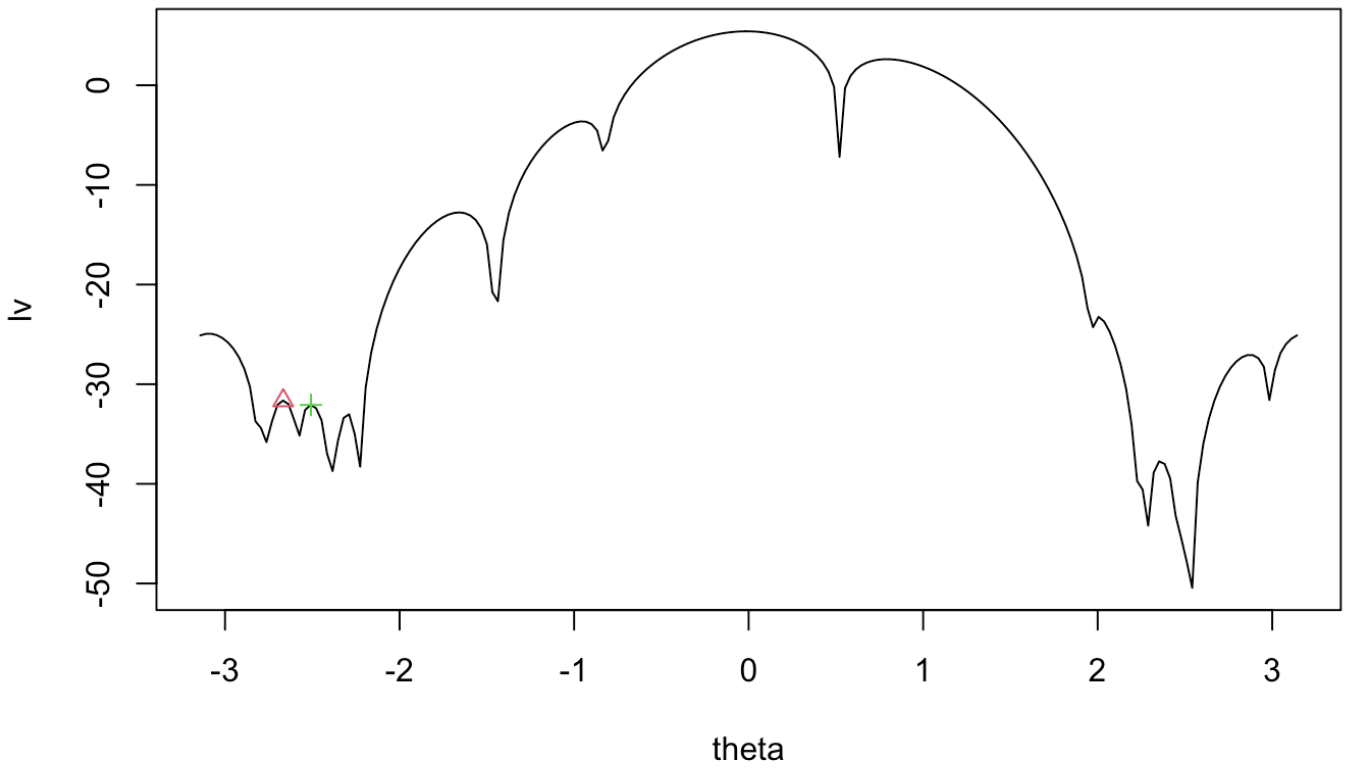


Hide

```
## the following plotting process is repeated in purpose of knitting the result
input = 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)
theta = seq(-pi, pi, length=200)
lv = theta
for(i in 1:length(theta)){
  lv[i] = f(input, theta[i])
}
plot(theta, lv, type="l")

## Find 2 starting values (nearly equal) that the Newton method converges to 2 dif
ferent solutions
theta = fnewton(input, starting[18], 1e-2)
points(theta, f(input, theta), pch=2, col=2)
```

Hide

```
theta = fnewton(input, starting[19], 1e-2)
points(theta, f(input, theta), pch=3, col=3)
```



Assignment 01 other questions

Hide

```
# Find max of h(x)=log(x)/(1+x) (1<x<5), using (1) Bisection, (2) Newton, (3) Seca
nt, and (4) Fixed point method.
theta = seq(1, 5, length=500)
lv = theta
for(i in 1:length(theta)){
  lv[i] = h(theta[i])
}
plot(theta, lv, type="l")

x_hat = hbisection(1, 5)
points(x_hat, h(x_hat), col=1)
```

Hide

```
x_hat = hnewton(3, 1e-2)
points(x_hat, h(x_hat), col=2)
x_hat = hsecant(3, 4)
points(x_hat, h(x_hat), col=3)
```

Hide

```
x_hat = hfixedpoint(4, 0.25)
points(x_hat, h(x_hat), col=4)
```