2022년 2학기

Computational Statistics

HW#1

222STG10

김희숙

Problem1. Bisection / Newton's / Secant Method 를 구현하시오.

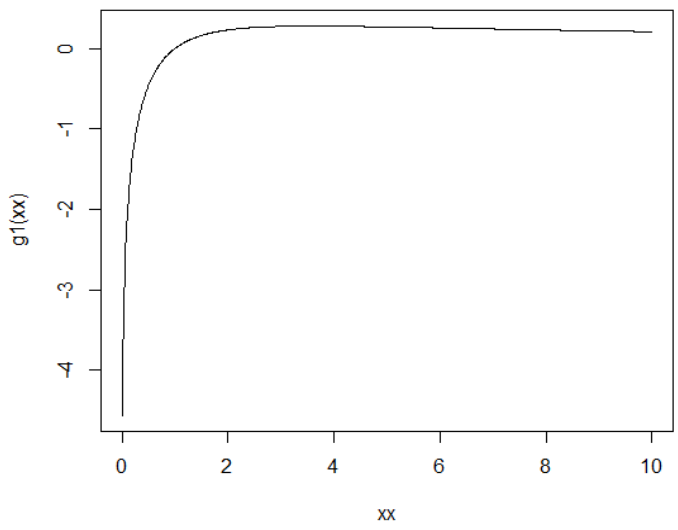
Problem2. 실제 3개 정도의 target function에 대해서 계산하시오.

1. $\frac{\log x}{1+x}$ 2. $-(x-1)^2-4$ 3. $e^x * sinx + \frac{\log x}{x^2}$

Result

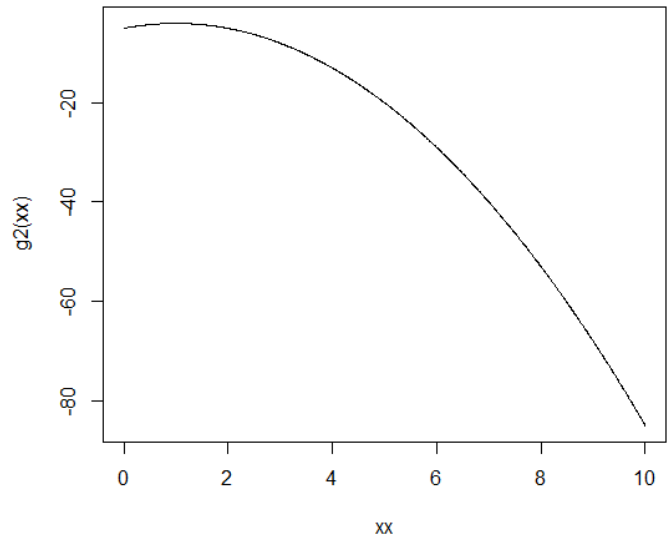
대체로 주어진 target function에 대해 세가지 방법 모두 argmax g(x) (= x^*)와 max g(x)를 잘 찾았으나, $e^x * sinx + \frac{\log x}{x^2}$ 함수의 경우 Bisection를 제외한 Method는 x^*를 제대로 찾지 못하는 것을 확인 할 수 있다. Newton's 과 Secant method의 경우 초기값에 예민하며, number of iter가 Bisection보다 작다. 두 method는 초기값의 설정이 중요하다고 생각된다.

1)

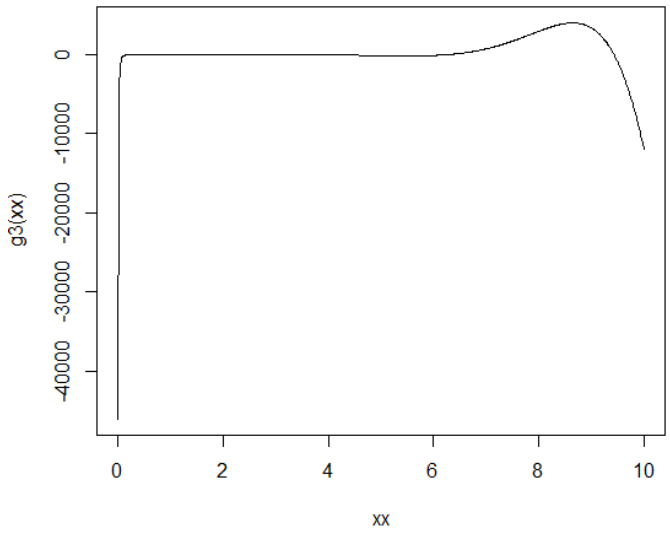
g(x)		input	x^*	max	# of iter	comp. time
$\frac{\log x}{1+x}$	Bisection	a=0.1, b=10	3.5911215	0.2784645	36	0.00535
	Newton's	x0=0.1	3.5911215	0.2784645	13	0.00206
	Secant	x0=2, x1=4	3.5911215	0.2784645	9	0.002
Graph						

2)

g(x)		input	x^*	max	# of iter	comp. time
$-(x-1)^2-4$	Bisection	a=0.1, b=10	1	-4	36	0.00525
	Newton's	x0=0.1	1	-4	4	0.00058

	Secant	x0=2, x1=4	1	-4	3	0.00064
Graph						

3)

g(x)		input	x*	max	# of iter	comp. time
$e^x * sinx + \frac{\log x}{x^2}$	Bisection	a=6, b=10	8.639379	3995.058249	35	0.00534
	Newton's	x0=8	5.497829	-172.584485	8	0.00124
	Secant	x0=6, x1=10	5.497829	-172.584485	11	0.00253
Graph						

** Newton's 은 x0≥8.1, Secant은 x0≥7.8, x1=10으로 했을 때 제대로 찾는다.

Code appendix

```
#the simplest function
# 임의의 target function g(x)에 대해서,
#install.packages('numDeriv')
library("numDeriv")


# Bisection
bisect<-function(g,a,b){
  maxiter<-1000
  threshold<-10^(-10)
  err<-1
  niter<-0
  x0<-(a+b)/2

  while ( niter<=maxiter && err >= threshold){
    #update interval
    if (genD(g,a)$D[1]*genD(g,x0)$D[1] <=0) {b<-x0}
    else    {a<-x0}

    #update x
    oldx0<-x0
    x0<-(a+b)/2 #새로 업데이트

    #update error and niter
    err<-abs(oldx0-x0)
    niter<-niter+1
  }
  return(c(x0, g(x0), niter))
}


# Newton's
newton<-function(g,x0){

  maxiter<-1000
  threshold<-10^(-10)
  err<-1
  niter<-0

  while ( niter<=maxiter && err >= threshold){
    #update
    oldx0<-x0
    x0 <- x0 - genD(g,x0)$D[1]/genD(g,x0)$D[2]

    #update error and niter
    err<-abs(oldx0-x0)
    niter<-niter+1
  }
  return(c(x0, g(x0), niter))
}


# Secant
secant<-function(g,x0,x1){
  maxiter<-1000
```

```
  threshold<-10^(-10)
  err<-1
  niter<-0

  while ( niter<=maxiter && err >= threshold){
    #update x
    newx0<-x1
    x1<- x1 - genD(g,x1)$D[1]*((x1-x0)/(genD(g,x1)$D[1]-genD(g,x0)$D[1])) # new

    #update error and niter
    err<-abs(newx0-x0) #x1 - x0
    niter<-niter+1

    x0 <- newx0
  }
  return(c(x1, g(x1), niter))
}



g1 <- function(x){
  log(x)/(1+x) #함수
}


g2 <- function(x){
  -(x-1)^2-4 #함수
}

g3 <- function(x){
  exp(x)*sin(x) + log(x)/(x^2)#함수
}

xx<-seq(0.01,10,0.01)
plot(xx,g1(xx),type="l")

plot(xx,g2(xx),type="l")

plot(xx,g3(xx),type="l")



bisect(g1,0.1,10)

## [1]  3.5911215  0.2784645 36.0000000

#system.time(for(i in 1:1000){bisect(g1,0.1,10)})

newton(g1,0.1)

## [1]  3.5911215  0.2784645 13.0000000

#system.time(for(i in 1:1000){newton(g1,0.1)})

secant(g1,2,4)

## [1] 3.5911215 0.2784645 9.0000000
```

```
#system.time(for(i in 1:1000){secant(g1,2,4)})


bisect(g2,0.1,10)
## [1]  1 -4 36
#system.time(for(i in 1:1000){bisect(g2,0.1,10)})
newton(g2,0.1)
## [1]  1 -4  4
#system.time(for(i in 1:1000){newton(g2,0.1)})
secant(g2,2,4)
## [1]  1 -4  3
#system.time(for(i in 1:1000){secant(g2,2,4)})


bisect(g3,6,10)
## [1]    8.639379 3995.058249   35.000000
#system.time(for(i in 1:1000){bisect(g3,6,10)})
newton(g3,8)
## [1]    5.497829 -172.584485    8.000000
#system.time(for(i in 1:1000){newton(g3,8)})
secant(g3,6,10)
## [1]    5.497829 -172.584485    11.000000
#system.time(for(i in 1:1000){secant(g3,6,8)})
```