Optimization-HW2-FangZheng

FangZheng

2018/2/6

Q1

(a)

Since for every \$ x_i, i = 1, 2,..., n \$, there are

$$p(x_i, \Theta) = \frac{1}{\pi[1 + (x_i - \Theta)]}$$

So

$$l(\theta) = \ln(\prod_{i=1}^{n} p(x_i, \Theta)) = \sum_{i=1}^{n} \ln(p(x_i, \Theta)) = -n\ln(\pi) - \sum_{i=1}^{n} \ln(1 + (\Theta - x)^2)$$

Take derivatives on both sides

$$l'(\theta) = -2\sum_{i=1}^{n} \frac{\theta - x_i}{1 + (\Theta - x)^2}$$

Take second derivatives on both sides

$$l''(\theta) = -2\sum_{i=1}^{n} \frac{1 - (\theta - x_i)^2}{(1 + (\Theta - x)^2)^2}$$

We alredy kown

$$P(x) = \frac{1}{\pi(1 + x^2)}; P'(x) = -\frac{1}{\pi}\ \frac{2x}{(1 + x^2)^2}; P'^2(x) = \frac{4}{\pi^2}\ \frac{x^2}{(1 + x^2)^4}$$

So

$$I(\theta) = n\int \frac{p'(x)^2}{p(x)} dx = \frac{4n}{\pi}\int_{-\infty}^{\infty} \frac{x^2 dx}{(1 + x^2)^3}$$

Let $x = \tan\beta$, then we have $1 + \tan^2\beta = \arcsin^2\beta = \frac{1}{\cos^2\beta}$ and $dx = \frac{d\beta}{\cos^2 beat}$ Then

$$I(\theta) = \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\tan^2\beta}{(1 + tan^2\beta)^3}\frac{1}{cos^2\beta} d\beta = \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin^2\beta\cos^2\beta\ d\beta = \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin^2\beta(1 - \sin^2\beta)\ d\beta = \frac{4n}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin^2\beta - \sin^4\beta\ d\beta$$

As $1 - \cos 2\beta = 2\sin^2\beta$, then we have

$$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin^2\beta\ d\beta = \frac{1}{2}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} (1 - \cos 2\beta) d\beta = \frac{1}{2}(\beta - \frac{1}{2}\sin 2\beta)|_{-\frac{\pi}{2}}^{\frac{\pi}{2}} = \frac{1}{2}$$

$$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin^4\beta\ d\beta = \frac{1}{4}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} (1 - \cos 2\beta)^2 d\beta = \frac{1}{4}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^2 2\beta - 2\cos 2\beta + 1\ d\beta =$$

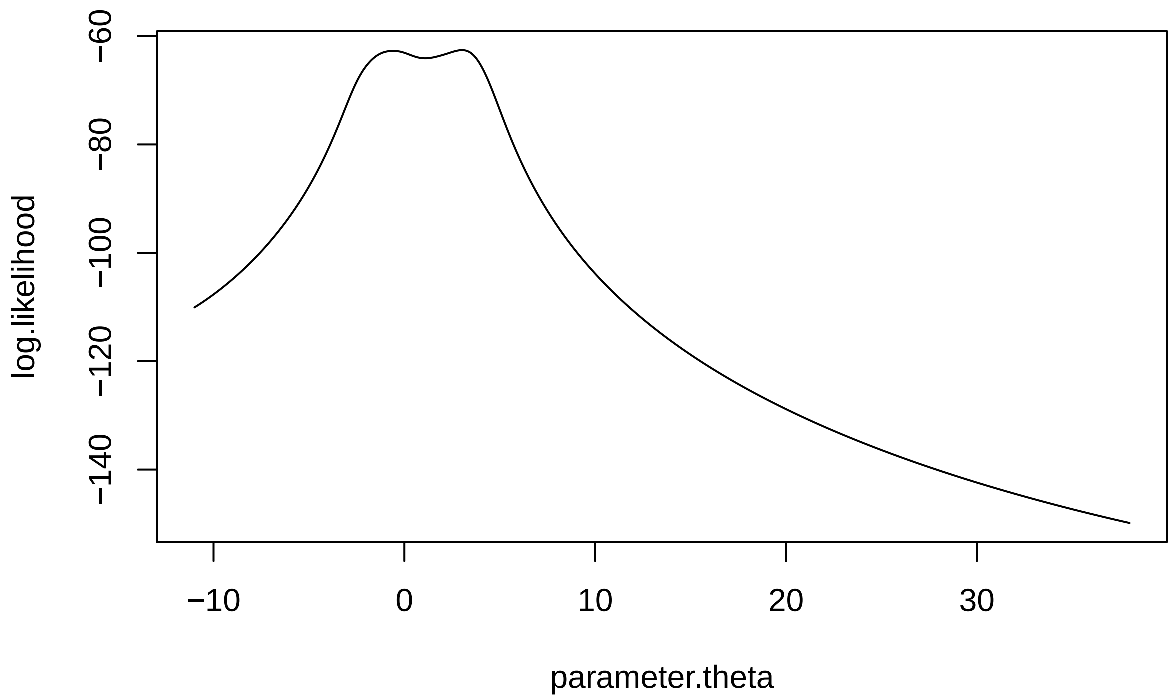
$$= \frac{1}{4}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\cos 4\beta}{2} - 2\cos 2\beta + \frac{3}{2}\ d\beta = \frac{1}{4}(\frac{1}{8}\sin 4\beta - \sin 2\beta + \frac{3}{2}\beta)|_{-\frac{\pi}{2}}^{\frac{\pi}{2}} = \frac{3\pi}{8}$$

Finally we have:

$$I(\theta) = \frac{4n}{\pi}(\frac{\pi}{2} - \frac{3\pi}{8}) = \frac{n}{2}$$

(b)

```
x1=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,-1.7

n = length(x1)

l <- function(theta) {
  return(n*log(pi) + sum(log(1+(theta-x1)^2))) #-log
}

l.grid <- function(theta) {
  return(2*sum((theta-x1)/(1+(theta-x1)^2))) #-l.grid
}

l.hess <- function(theta) {
  return(matrix(2*sum((1-(theta-x1)^2)/(1+(theta-x1)^2)^2),nrow=1)) #-l.hess
}

theta.est <- function(starting) {
  ## MLE
  theta <- nlminb(start=starting,l,l.grid,l.hess)$par
  return(theta)
}



# plot

theta.value = seq(-11, 38, by=0.05)

log.likelihood = c()

for (i in 1:length(theta.value)){
  log.likelihood[i] = -l(theta.value[i])
}

plot(theta.value, log.likelihood, xlab="parameter.theta",ylab="log.likelihood", type="l")
```




```
theta.est(-11)
```

```
## [1] 3.021345
```

```
theta.est(-1)
```

```
## [1] -0.5914735
```

```
theta.est(0)
```

```
## [1] -0.5914735
```

```
theta.est(1.5)
```

```
## [1] 3.021345
```

```
theta.est(4)
```

```
## [1] 3.021345
```

```
theta.est(4.7)
```

```
## [1] 3.021345
```

```
theta.est(7)
```

```
## [1] 3.021345
```

```
theta.est(8)
```

```
## [1] 3.021345
```

```
theta.est(38)
```

```
## [1] 3.021345
```

(c)

```
theta.est.fix <- function(starting) {
  alpha <- c(1,0.64,0.25)
  ## MLE
  theta <- c()
  for (j in 1:3){
    l.hess.fix <- function(theta) {
      return(matrix(1/alpha[j],nrow=1))
    }
    theta[j] <- nlminb(start=starting,l,l.grid,l.hess.fix)$par
  }
  return(theta)
}
theta.est.fix(-11)
```

```
## [1] -0.5914827  3.0213332 -0.5915249
```

```
theta.est.fix(-1)
```

```
## [1] -0.5914791 -0.5914824 -0.5915051
```

```
theta.est.fix(0)
```

```
## [1] -0.5914717 -0.5914659 -0.5914218
```

```
theta.est.fix(1.5)
```

```
## [1] 3.021345 3.021345 3.021344
```

```
theta.est.fix(4)
```

```
## [1] 3.021345 3.021338 3.021345
```

```
theta.est.fix(4.7)
```

```
## [1] 3.021345 3.021328 3.021344
```

```
theta.est.fix(7)
```

```
## [1] 3.021335 3.021345 3.021343
```

```
theta.est.fix(8)
```

```
## [1] -0.5914408 -0.5914885  3.0213434
```

```
theta.est.fix(38)
```

```
## [1] -0.5914796  3.0213451  3.0213435
```

(d)

```
theta.est.fisher <- function(starting){
  ## MLE
  theta <- starting
  delta <- 1
  while(abs(delta) >= 0.00001){
    theta1 <- theta
    dl <- l.grid(theta)
```

```r
    theta <- theta - dl/(n/2)
    delta <- theta - theta1
  }
  return(theta)
}

theta.est.fisher(-11)
```

```
## [1] -0.5915031
```

```r
theta.est.fisher(-1)
```

```
## [1] -0.5915022
```

```r
theta.est.fisher(0)
```

```
## [1] -0.5914394
```

```r
theta.est.fisher(1.5)
```

```
## [1] 3.021336
```

```r
theta.est.fisher(4)
```

```
## [1] 3.021354
```

```r
theta.est.fisher(4.7)
```

```
## [1] 3.021359
```

```r
theta.est.fisher(7)
```

```
## [1] 3.021358
```

```r
theta.est.fisher(8)
```

```
## [1] 3.021356
```

```r
theta.est.fisher(38)
```

```
## [1] 3.021357
```

(e)

Based on the results above, blow is the table that shows the rank of some feature of different method.

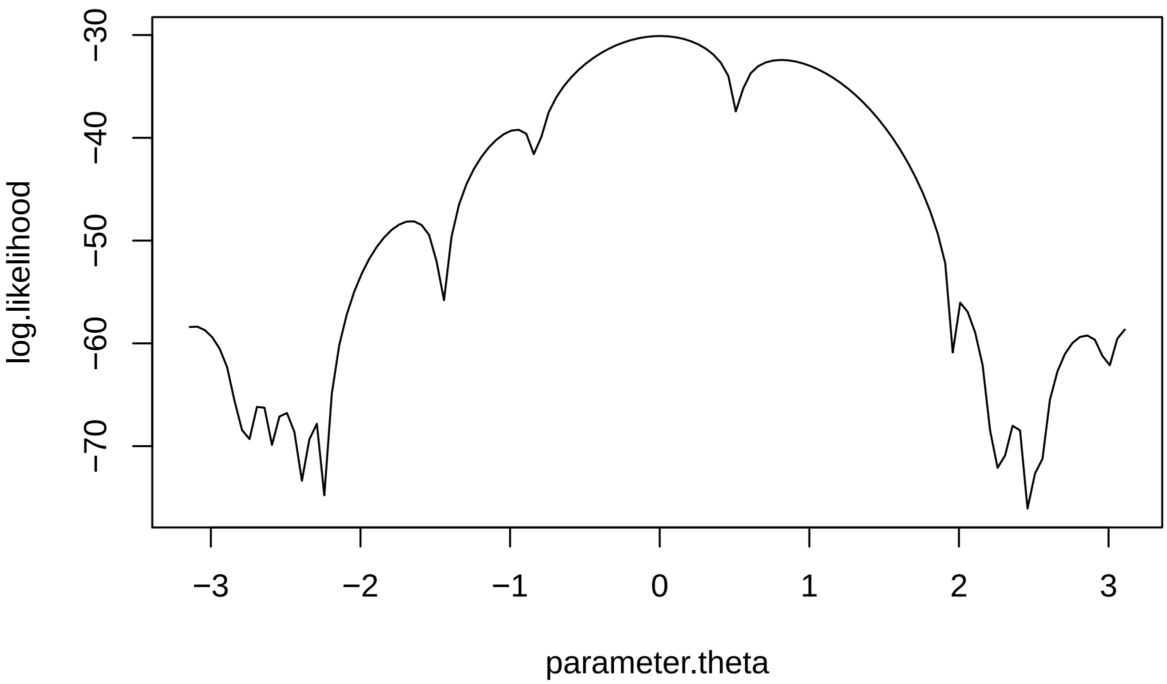
```r
knitr::kable(
  matrix (c (1, 3, 2, 1, 3, 1, 3, 1, 2), nrow = 3, ncol = 3, byrow = TRUE, dimnames = list(c("speed",
)
```

	method_b	method_c	method_d
speed	1	3	2
stability	1	3	1
accuracy	3	1	2

Q2

(a)

```
x2 = 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)
a.value = seq(-pi, pi, by=0.05)
n = length(x2)
log.like = c()
for (i in 1:length(a.value)){
  log.like[i] = -n*log(2*pi)+sum(log(1-cos(x2-a.value[i])))
}
plot(a.value, log.like, xlab="parameter.theta",
     ylab="log.likelihood", type="l")
```



(b)

```
a.est <- function(sample,starting) {
  n = length(sample)

  l <- function(a) {
    return(-n*log(2*pi) + sum(log(1-cos(sample-a)))) #log
  }

  l.grid <- function(a) {
    return(sum(sin(a-sample)/(1-cos(a-sample)))) #l.grid
  }

  l.hess <- function(a) {
    return(sum(1/(cos(a-sample)-1))) #l.hess
  }
```

```r
  ## MLE
  theta <- starting
  delta <- 1
  while(abs(delta) >= 1e-10){
    theta1 <- theta
    dl <- l.grid(theta)
    ddl <- l.hess(theta)
    if (ddl==0){
      return(NULL)
    }
    theta <- theta - dl/ddl
    delta <- theta - theta1
  }

  return(c(theta,l(theta)))
}

m = mean(x2)
theta0 = asin(m-pi)
theta0
```

```
## [1] 0.09539407
```

(c)

```r
a.est1 = a.est(x2,theta0)
a.est1[1]
```

```
## [1] 0.003118157
```

(d)

```r
a.est2 = a.est(x2,-2.7)
a.est3 = a.est(x2,2.7)
a.est2[1]
```

```
## [1] -2.668857
```

```r
a.est3[1]
```

```
## [1] 2.848415
```

(e)

```r
a.value = seq(-pi,pi,by = 2*pi/199); # set 200 equally spaced starting point
a.par = c()
a.mle = c()
for (i in 1:length(a.value)){
  a.par[i] = a.est(x2,a.value[i])[1]
  a.mle[i] = a.est(x2,a.value[i])[2]
}
```

```r
ascend = order(a.mle)
a.mle.ascend = a.mle[ascend]
a.par.ascend = a.par[ascend]

# find the unique outcome
j = 1
p = 2
i = 1
e = c()
e[1] = a.mle.ascend[1]
while(j<=200){
  if(a.mle.ascend[i]!=a.mle.ascend[j]){
    e[p] = a.mle.ascend[j]
    p = p+1
    i = j
    j = j+1
  }else{j = j+1}
}

# classify the starting point according to the unique outcome
group = matrix(nrow = length(e),ncol = 46)
for (y in 1:length(e)){
  t = 1
  for (i in 1:200){
    if(a.mle.ascend[i]== e[y]){
      group[y,t] = round(a.value[ascend[i]],5)
      t = t+1
    }
  }
}
group[is.na(group)]=""
knitr::kable(group)
```

-2.38382									
-2.22595									
2.47854									
2.51012									
2.22595									
2.25753									
-2.79428									
-2.76271									
2.32067									
2.44697									
2.2891	2.35225	2.38382	2.4154						
-2.35225	-2.2891	-2.25753							
-2.32067									
-2.57326	-2.54169	-2.47854	-2.4154						
-2.51012	-2.44697								
-2.73113	-2.69956	-2.66799							
-2.63641	-2.60484								
2.54169	2.57326	2.63641	2.66799	2.73113	2.76271	2.85743	2.889	2.92058	2.98372
2.60484	2.69956	2.79428	2.82585	2.95215					
-3.14159	-3.11002	-3.07845	-3.04687	-3.0153	-2.95215	-2.92058	-2.889	-2.85743	-2.82585

-2.98372												
3.0153	3.04687	3.07845	3.11002	3.14159								
2.1628												
1.97336	2.13123											
2.00494	2.06808	2.19438										
2.09966												
2.03651												
-1.43661												
-2.1628	-2.13123	-2.09966	-1.91021	-1.87864	-1.84707	-1.81549	-1.72077	-1.5629	-1.49976			
-2.19438	-2.06808	-2.03651	-2.00494	-1.97336	-1.94179	-1.78392	-1.75235	-1.6892	-1.65762	-1.62605	-1.5	
-1.18402												
-1.40503	-1.37346	-1.34189	-1.31031	-1.27874	-1.24716	-1.21559	-1.15244	-1.12087	-1.0893	-1.05772	-1.0	
0.52097	0.55254	0.58412	0.61569	0.64726	0.67884	0.71041	0.74198	0.77356	0.80513	0.83671	0.8	
-0.80513	-0.77356	-0.74198	-0.71041	-0.67884	-0.64726	-0.61569	-0.58412	-0.55254	-0.52097	-0.48939	-0.4	

Q3

(a)

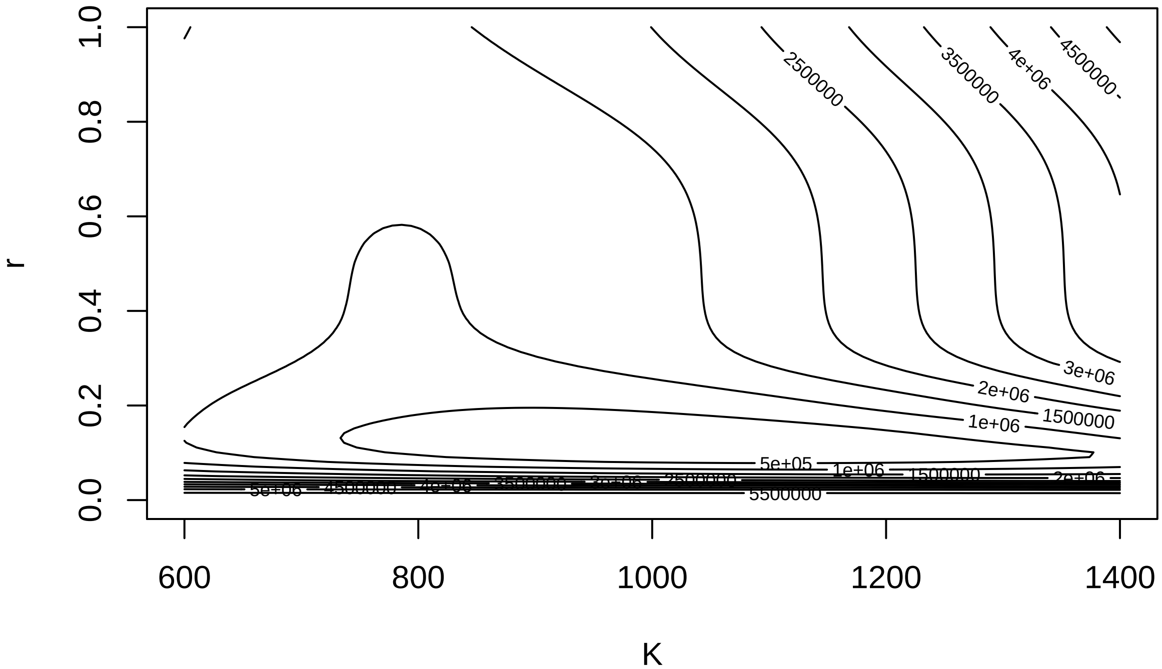
```r
t <- c(0,8,28,41,63,69,97,117,135,154)
x <- c(2,47,192,256,768,896,1120,896,1184,1024)
f <- expression(2*K/(2+(K-2)*exp(-r*t)))

df <- function(K,r,t){
  dfk <- D(f,"K")
  dfr <- D(f,"r")
  K <- K
  r <- r
  t <- t
  a <- eval(dfk)
  b <- eval(dfr)
  c <- array(c(a,b),c(1,2))
  return(c)
}

Df <- function(K,r){
  a <- K
  b <- r
  m <- df(a,b,t[1])
  for(i in 2:10){
    c <- df(a,b,t[i])
    m <- rbind(m,c)
  }
  return(m)
}

Z <- function(K,r){
  a <- c()
  for(i in 1:10){
    a[i] <- x[i] - 2*K/(2+(K-2)*exp(-r*t[i]))
  }
  m <- array(a,c(10,1))
```

```r
  return(m)
}

theta <- matrix(c(1200,0.2),nrow=2)
delta <- matrix(c(1,1),nrow=2)

while(crossprod(delta,delta)>=0.001){
  theta1 <- theta
  a <- Df(theta[1,1],theta[2,1])
  z <- Z(theta[1,1],theta[2,1])
  theta <- theta + solve(t(a)%*%a)%*%t(a)%*%z
  delta <- theta - theta1
}

a.est <- theta
print(a.est)
```

```
##              [,1]
## [1,] 1049.4038970
## [2,]    0.1182693
```

(b)

```r
f <- function(K,r){
  return(sum((x-2*K/(2+(K-2)*exp(-r*t)))^2))
}

z <- matrix(0,100,100,byrow=T)
for (i in 1:100){
  for (j in 1:100){
    K <- 600 + 8*j
    r <- 0 + 0.01*i
    z[j,i] <- f(K,r)
  }
}
K <- seq(600,1400,length.out = 100)
r <- seq(0,1,length.out = 100)
contour(K,r,z,xlab="K",ylab="r")
```



(c)

```
l <- expression(log(1/(sqrt(2*pi)*sigema))-
                  (log((2*2+2*(K-2)*exp(-r*0))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*47+47*(K-2)*exp(-r*8))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*192+192*(K-2)*exp(-r*28))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*256+256*(K-2)*exp(-r*41))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*768+768*(K-2)*exp(-r*63))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*896+896*(K-2)*exp(-r*69))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*1120+1120*(K-2)*exp(-r*97))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*896+896*(K-2)*exp(-r*117))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*1185+1184*(K-2)*exp(-r*135))/(2*K)))^2/(2*sigema^2)+
                  log(1/(sqrt(2*pi)*sigema))-
                  (log((2*1024+1024*(K-2)*exp(-r*154))/(2*K)))^2/(2*sigema^2))

dl <- function(beita){
  dlk <- D(l,"K")
  dlr <- D(l,"r")
  dlsigema <- D(l,"sigema")
  K <- beita[1]
  r <- beita[2]
  sigema <- beita[3]
```

```r
  a <- eval(dlk)
  b <- eval(dlr)
  c <- eval(dlsigema)
  return(c(a,b,c))
}

ddl <- function(beita){
  dlkk <- D(D(l,"K"),"K")
  dlkr <- D(D(l,"K"),"r")
  dlksigema <- D(D(l,"K"),"sigema")
  dlrr <- D(D(l,"r"),"r")
  dlrsigema <- D(D(l,"r"),"sigema")
  dlsigema2 <- D(D(l,"sigema"),"sigema")
  K <- beita[1]
  r <- beita[2]
  sigema <- beita[3]
  a <- c(eval(dlkk),eval(dlkr),eval(dlksigema),eval(dlkr),eval(dlrr),
         eval(dlrsigema),eval(dlksigema),eval(dlrsigema),eval(dlsigema2))
  m <- matrix(a,byrow=TRUE,nrow=3)
  return(m)
}

a <- matrix(c(1200,0.2,0.5),nrow=3)
delta <- matrix(c(1,1,1),nrow=3)
while(crossprod(delta,delta)>=0.001){
  b <- a
  c <- matrix(dl(a),nrow=3)
  d <- solve(ddl(a))
  a <- a - d%*%c
  delta <- a - b
}
a
```

```
##             [,1]
## [1,] 820.5349872
## [2,]   0.1926176
## [3,]   0.6441323
```

```r
solve(-ddl(a))
```

```
##               [,1]          [,2]          [,3]
## [1,]  6.248530e+04 -9.054089e+00  1.051866e-07
## [2,] -9.054089e+00  3.974068e-03 -5.353628e-11
## [3,]  1.051866e-07 -5.353628e-11  2.074532e-02
```