Assignment 2 - Optimization and Simulating random variables

Name: Amine Natik
Date Date: February 15, 2018

1. (a) We define the following function :

```
SampleBeta=function(a,b){
  if ((a==round(a))&(b==round(b))){
    X=sum(-log(runif(a)))
    Y=sum(-log(runif(b)))
  }
  else{
    stop("please enter integer values for a and b.")
  }
  return(X/(X+Y))
}
```

(b) See Figure 1 for the graphs,

```
> numsteps=1000
> parameters=list(c(3,5),c(2,7),c(5,5),c(4,6))
> par(mfrow=c(2  ,2))
> for(i  in  1:length(parameters)){
+   a=parameters[[i]][1]
+   b=parameters[[i]][2]
+   betavector=replicate(numsteps,SampleBeta(a,b))
+   plot(density(betavector),main=paste('Beta(',a,',',b,')',
+   sep=""),xlab="", ylab="")
+   curve(dbeta(x,shape1=a,shape2=b),from=0,to=1,col="red",add=T)
+ }
```

(c) We are going to use the following fact, if $X \sim \text{Beta}\left(\frac{d_1}{2}, \frac{d_2}{2}\right)$ then $\frac{d_2 X}{d_1(1-X)} \sim F_{d_1,d_2}$ for all $d_1, d_2 > 0$.

```
SampleF=function(m,n){
    X=SampleBeta(m/2,n/2)
    return(n*X/m*(1-X))
}
```

Since our function SampleBeta defined in the first question takes integers as inputs, m and n should both be even numbers.

2. (a) We first denote, $R = \sqrt{-2\log U_2}$ and $\Theta = 2\pi U_2$, we then have :

$$\begin{cases} X_1 = R\cos(\Theta) \\ X_2 = R\sin(\Theta) \end{cases}$$

It is clear that $\Theta \sim \mathcal{U}[0, 2\pi]$. And using the change variable theorem one could find the distribution of R, for all $r \geq 0$ we have,

$$\begin{aligned} f_R(r) &= f_{U_1}\left(e^{-\frac{1}{2}r^2}\right)\left|\frac{\mathrm{d}}{\mathrm{d}r}e^{-\frac{1}{2}r^2}\right| \\ &= re^{-\frac{1}{2}r^2} \end{aligned}$$

now define the transformation,

$$\begin{aligned} \varphi : \quad & \mathbb{R}^+ \times \mathbb{R} && \longrightarrow \mathbb{R}^2 \\ & (r, \theta) && \longmapsto (r\cos(\theta), r\sin(\theta)) \end{aligned}$$

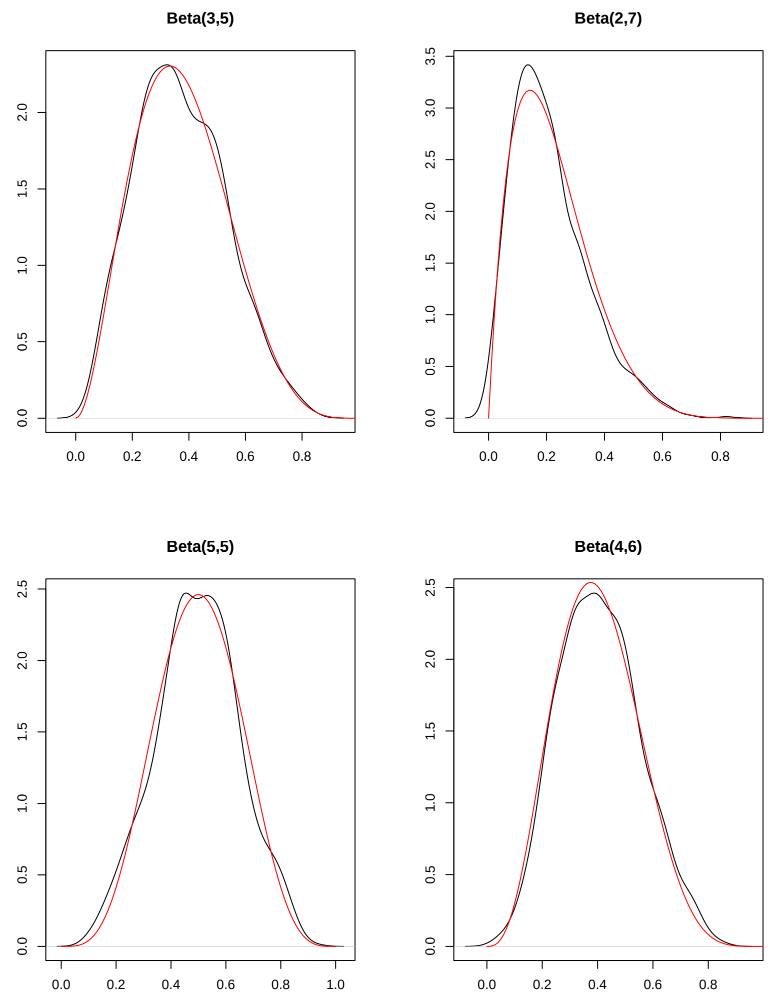


Figure 1: Q1 (b) – Beta Simulations

then obviously: $(X_1, X_2) = \varphi(R, \Theta)$. Therefore, using again the change of variable theorem:

$$f_{X_1,X_2}(x_1,x_2) = f_{R,\Theta}\left(\sqrt{x_1^2+x_2^2}, \arctan\left(\frac{x_2}{x_1}\right)\right) \cdot \left|J^{-1}(x_1,x_2)\right|$$

where J is the Jacobian matrix, clearly $J(x_1,x_2) = \sqrt{x_1^2+x_2^2}$, therefore

$$\begin{aligned} f_{X_1,X_2}(x_1,x_2) &= \frac{1}{\sqrt{x_1^2+x_2^2}} f_R\left(\sqrt{x_1^2+x_2^2}\right) f_\Theta\left(\arctan\left(\frac{x_2}{x_1}\right)\right) \\ &= \frac{1}{\sqrt{x_1^2+x_2^2}} \sqrt{x_1^2+x_2^2} e^{-\frac{1}{2}\left(x_1^2+x_2^2\right)} \frac{1}{2\pi} \\ &= \underbrace{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x_1^2}}_{f_{X_1}(x_1)} \cdot \underbrace{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x_2^2}}_{f_{X_2}(x_2)} \end{aligned}$$

Which means that X_1 and X_2 are *i.i.d* $\mathcal{N}(0,1)$

(b) Define the following function to sample from a $\mathcal{N}(\mu, \sigma^2)$

```
NormUnif=function(n,mean=0,sd=1){
  vec=c()
  for(i in 1:n){
    u1=runif(1)
    u2=runif(1)
    vec=c(vec, sqrt(-2*log(u1))*cos(2*pi*u2))
  }
  return(sd*vec+mean)
}
```

(c) Let $X \sim \mathcal{N}(0,1)$, we wish to sample from double exponential distribution with density :

$$g(x|\theta) = \frac{\theta}{2}e^{-\theta|x|}$$

let,

$$M = \inf_{\theta} \sup_{x} \frac{f(x)}{g(x|\theta)} = \inf_{\theta} \sup_{x} \left(\sqrt{\frac{2}{\pi}}\theta^{-1}e^{-\frac{x^2}{2}+\theta|x|} \right) = \sqrt{\frac{2e}{\pi}}, \tag{1}$$

with equality if $\theta = 1$ and $x = \theta$. Thus $(\forall x \in \mathbb{R}) : f(x) \leq Mg(x|1)$. In order to use the accept-reject method, we should be able to sample from the density $g(x|1)$. Note that the cumulative distribution function is as follows:

$$F(x) = \frac{1}{2} + \frac{1}{2}\text{sign}(x)\left(1 - e^{-|x|}\right)$$

therefore the inverse cumulative distribution function is given by

$$F^{-1}(x) = -\text{sign}\left(x - \frac{1}{2}\right)\ln\left(1 - 2\left|x - \frac{1}{2}\right|\right) \tag{2}$$

Moreover, if $U \sim \mathcal{U}[0,1]$ then $F^{-1}(U)$ has a double exponential distribution with density $g(x|1)$. We can use the following function to sample from it:

```
doubexp=function(n){
  vec=c()
  for(i in 1:n){
    u=runif(1)
    vec=c(vec,-sign(u-0.5)*log(1-2*abs(u-0.5))) # according to equation (2)
  }
  return(vec)
}
```

Now, to use the accept-reject sampling, we have to simulate a random variable $U \sim \mathcal{U}[0,1]$ and X from a double exponential, and accept it if

$$u \leq \frac{f(x)}{Mg(x|1)} = \exp\left(-\frac{x^2}{2} + |x| - \frac{1}{2}\right). \tag{3}$$

Therefore the function is defined as follows:

```
rejectS_normal=function(n,mean=0,sd=1){
  vec=c()
  M=sqrt(2*exp(1)/pi) # M is defined in equation (1)
  for(i in 1:n){
    accept=F
    while(accept==F){
      u=runif(1)
      x=doubexp(1) # We call the function defined previously
      if(u<=exp(-x^2/2+abs(x)-1/2)){# The test in equation (3)
        accept=T
        vec=c(vec,x)
      }
```

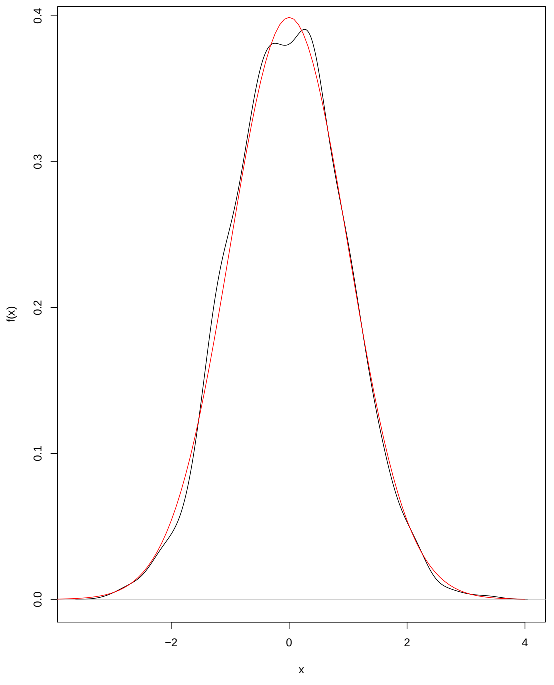
```
        }
    }
    return (sd*vec+mean)
}
```

(d) We compare the run times of the functions,

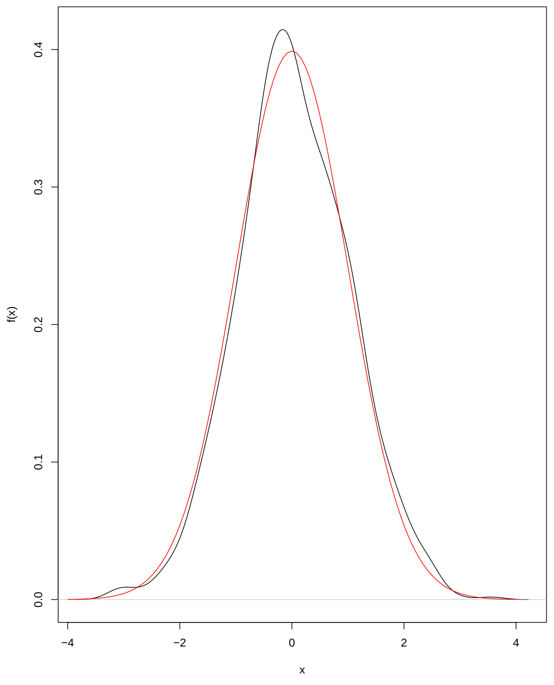
```
> system.time(NormUnif(n=1000))
   user  system elapsed
  0.007   0.000   0.009
> system.time(rejectS_normal(n=1000))
   user  system elapsed
  0.011   0.000   0.012
```

We can see that simulation from the method in question (b) is faster than the accept-reject method. See Figure 2 for the plots of the random sample.

```
> par(mfrow=c(1,2))
> vector1=NormUnif(1000)
> vector2=rejectS_normal(1000)
> plot(density(vector1),main="",xlab="x",ylab="f(x)")
> curve(dnorm(x),from=-4,to=4,col="red",add=T)
> plot(density(vector2),main="",xlab="x",ylab="f(x)")
> curve(dnorm(x),from=-4,to=4,col="red",add=T)
```



(a) Uniform simulation



(b) Accept-reject Method

Figure 2: Q2 (d): Normal distribution plots

3. (a) Suppose that $(X_1, X_2) \sim \mathcal{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the observed data (with missing data), where

$$\boldsymbol{\mu} = (\mu_1, \mu_2) \text{ and } \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{1,2} \\ \sigma_{1,2} & \sigma_2^2 \end{pmatrix}$$

therefore, $X_1 \sim \mathcal{N}(\mu_1, \sigma_1)$, $X_2 \sim \mathcal{N}(\mu_2, \sigma_2)$. Suppose that we make the following n-observations:

$$\begin{aligned}(x_{i,1}, x_{i,2}) &\text{ for } 1 \le i \le n_1. \ (n_1 \text{ observations without missing data})\\ x_{i,2} &\text{ for } n_1+1 \le i \le n_1+n_2. \ (n_2 \text{ missing values in } X_1)\\ x_{i,1} &\text{ for } n_1+n_2+1 \le i \le n_1+n_2+n_3. \ (n_3 \text{ missing values in } X_2)\end{aligned}$$

which can be presented in the following way,

$$\begin{array}{lcccccccc} X_1: & x_{1,1} & \dots & x_{n_1,1} & NA & \dots & NA & x_{n_1+n_2+1,1} & \dots & x_{n,1}\\ X_2: & x_{1,2} & \dots & x_{n_1,2} & x_{n_1+1,2} & \dots & x_{n_1+n_2,2} & NA & \dots & NA\end{array}$$

where $n = n_1+n_2+n_3$. Denote by $Z = (x_{n_1+1,1}, \dots, x_{n_1+n_2,1}, x_{n_1+n_2+1,2}, \dots, x_{n,2})$ the missing data vector, and Y the complete data. Finally let Θ be the parameters vector that we are interested to estimate $\Theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The likelihood function is given by:

$$\mathcal{L}_c(\Theta|Y=y) = \prod_{i=1}^{n} \frac{1}{2\pi |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu})}$$

Hence the log-likelihood is:

$$\log(\mathcal{L}_c(\Theta|Y=y)) = -n\log(2\pi) - \frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu})$$

This is a quadratic polynomial, therefore the calculation of the expected value step in the EM algorithm, which is,

$$Q\left(\Theta|\Theta^{(k)}\right) = \mathbb{E}_{Z|X,\Theta^{(k)}}\left[\log(\mathcal{L}_c(\Theta|Y=y))\right]$$

requires only $\mathbb{E}_{Z|X,\Theta^{(k)}}[X_{i,h}|Y=y]$ and $\mathbb{E}_{Z|X,\Theta^{(k)}}\left[X_{i,h}^2|Y=y\right]$, where $h=0,1$ and $1 \le i \le n$. Note that the conditional distributions of the bivariate normal distribution still normal, in fact

$$X_{i,1}|(X_{i,2}=x_{i,2}) \sim \mathcal{N}\left(\mu_1 + \frac{\sigma_{1,2}}{\sigma_2^2}(x_{i,2}-\mu_2), \left(1 - \frac{\sigma_{1,2}^2}{\sigma_1^2\sigma_2^2}\right)\sigma_1^2\right)$$

for all $n_1+1 \le i \le n_1+n_2$, hence:

$$x_{i,1}^{(k)} \overset{\text{def}}{=} \mathbb{E}_{Z|X,\Theta^{(k)}}[X_{i,1}|Y=y] = \mu_1^{(k)} + \frac{\sigma_{1,2}^{(k)}}{\left(\sigma_2^2\right)^{(k)}}\left(x_{i,2}^{(k)} - \mu_2^{(k)}\right) \tag{4}$$

$$\left(x_{i,1}^2\right)^{(k)} \overset{\text{def}}{=} \mathbb{E}_{Z|X,\Theta^{(k)}}\left[X_{i,1}^2|Y=y\right] = \left(x_{i,1}^{(k)}\right)^2 + \left(1 - \left(\frac{\sigma_{1,2}^{(k)}}{\sigma_1^{(k)}\sigma_2^{(k)}}\right)^2\right)\left(\sigma_1^{(k)}\right)^2 \tag{5}$$

similarly for all $n_1+n_2+1 \le i \le n$,

$$x_{i,2}^{(k)} \overset{\text{def}}{=} \mathbb{E}_{Z|X,\Theta^{(k)}}[X_{i,2}|Y=y] = \mu_2^{(k)} + \frac{\sigma_{1,2}^{(k)}}{\left(\sigma_1^2\right)^{(k)}}\left(x_{i,1}^{(k)} - \mu_1^{(k)}\right) \tag{6}$$

$$\left(x_{i,2}^2\right)^{(k)} \overset{\text{def}}{=} \mathbb{E}_{Z|X,\Theta^{(k)}}\left[X_{i,2}^2|Y=y\right] = \left(x_{i,2}^{(k)}\right)^2 + \left(1 - \left(\frac{\sigma_{1,2}^{(k)}}{\sigma_1^{(k)}\sigma_2^{(k)}}\right)^2\right)\left(\sigma_2^{(k)}\right)^2. \tag{7}$$

Finally for $1 \le i \le n_1$, (if there is no missing data we do not change the data)

$$\begin{aligned} x_{i,1}^{(k)} &\overset{\text{def}}{=} x_{i,1}\\ x_{i,2}^{(k)} &\overset{\text{def}}{=} x_{i,2}\\ \left(x_{i,1}^2\right)^{(k)} &\overset{\text{def}}{=} x_{i,1}^2\\ \left(x_{i,2}^2\right)^{(k)} &\overset{\text{def}}{=} x_{i,2}^2\end{aligned}$$

Consider the maximization step vector,

$$\Theta^{(k+1)} = \arg\max_{\Theta} Q\left(\Theta|\Theta^{(k)}\right)$$

Then $\Theta^{(k+1)}$ represent the MLEs vector of a bivariate normal distribution $\mathcal{N}_2(\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$, we have:

$$\mu_1^{(k+1)} = \frac{1}{n}\sum_{i=1}^{n} x_{i,1}^{(k)} \tag{8}$$

$$\mu_2^{(k+1)} = \frac{1}{n}\sum_{i=1}^{n} x_{i,2}^{(k)} \tag{9}$$

$$\left(\sigma_1^2\right)^{(k+1)} = \frac{1}{n}\left(\sum_{i=1}^{n}\left(x_{i,1}^2\right)^{(k)} - n\left(\mu_1^{(k)}\right)^2\right) \tag{10}$$

$$\left(\sigma_2^2\right)^{(k+1)} = \frac{1}{n}\left(\sum_{i=1}^{n}\left(x_{i,2}^2\right)^{(k)} - n\left(\mu_2^{(k)}\right)^2\right) \tag{11}$$

$$\sigma_{1,2}^{(k+1)} = \frac{1}{n}\left(\sum_{i=1}^{n} x_{i,1}^{(k)} x_{i,2}^{(k)} - n\left(\mu_1\right)^{(k)}\left(\mu_2\right)^{(k)}\right) \tag{12}$$

The equations (8-12) below gives the EM Algorithm from a suitable starting point $\Theta^{(0)}$.
Implementation in R:
We first read the given data in R,

```
biv=read.table("bivariatenormal.txt",sep="",fill=FALSE,strip.white=TRUE)
X1=biv[,1]
X2=biv[,2]
n=length(X1)
```

Then we define the the euclidean distance function (this function will be useful for the stopping test)

```
norm_vec=function(x){
  return(sqrt(sum(x^2)))
}
```

Finally, we define our EM Algorithm function,

```
em_bivnormal=function(mu0,sigma0,maxit=1000,epsilon=10^{-6}){
  #initialisation
  i=0
  stop=FALSE
  mu=mu0
  sigma=sigma0
  #Create a copies of X1 and X2, and their squares.
  cX1=X1
  cX2=X2
  sqX1=X1^2
  sqX2=X2^2
  while((i<maxit)&&(stop==FALSE)){
    for (k in 1:n){
      if(is.na(X1[k])){#looking for missing data in X1
        #according to equations (4-5) :
        cX1[k]=mu[1]+sigma[1,2]*(X2[k]-mu[2])/sigma[2,2]
        sqX1[k]=(cX1[k])^2
        +(1-((sigma[1,2])^2/(sigma[1,1]*sigma[2,2])))*sigma[1,1]
      }
```

```
        if(is.na(X2[k])){#looking for missing data in X2
        #according to equations (6-7) :
          cX2[k]=mu[2]+sigma[1,2]*(X1[k]-mu[1])/sigma[1,1]
          sqX2[k]=(cX2[k])^2
          +(1-((sigma[1,2])^2/(sigma[1,1]*sigma[2,2])))*sigma[2,2]
        }
      }
      # new values of the parameters according to equations (8-12) :
      new_mu=c()
      new_mu[1]=mean(cX1)
      new_mu[2]=mean(cX2)
      new_sigma=matrix(NA,nrow=2,ncol=2)
      new_sigma[1,1]=(sum(sqX1)-n*(new_mu[1])^2)/n
      new_sigma[2,2]=(sum(sqX2)-n*(new_mu[2])^2)/n
      new_sigma[1,2]=(sum(cX1*cX2)-n*(new_mu[1])*(new_mu[2]))/n
      new_sigma[2,1]=new_sigma[1,2]
      if(norm_vec(c(new_mu-mu,new_sigma-sigma))<epsilon){#Stopping condition
        stop=TRUE
      }
      mu=new_mu
      sigma=new_sigma
      i=i+1
    }
    return(list(mean=mu, covariance=sigma, numit=i))
  }
```

(b) We choose as starting points $\mu^{(0)} = (0,0)$ and $\mathbf{\Sigma}^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and try the function,

```
>#Starting point
>mu0=c(0,0)
>sigma0=matrix(c(1,0,0,1),nrow=2,ncol=2)
> em_bivnormal(mu0,sigma0)
$mean
[1] 0.8603353 2.8363979

$covariance
          [,1]      [,2]
[1,] 1.3528566 0.9877001
[2,] 0.9877001 0.7929804

$numit
[1] 15
```

Therefore the MLEs are :

$$\hat{\mu} \approx (0.8603353, 2.8363979)$$
$$\hat{\mathbf{\Sigma}} \approx \begin{pmatrix} 1.3528566 & 0.9877001 \\ 0.9877001 & 0.7929804 \end{pmatrix}$$

4. Let $N = (N_1, \ldots, N_{26})$ be the number of oil spills vector during the 1974-1999 period, and let $n = (n_1, \ldots, n_{26})$ the corresponding observations and put $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$. For each i we have, $N_i \sim \mathcal{P}oisson\,(\lambda_i)$. The likelihood function is given by,

$$\mathcal{L}\left(\boldsymbol{\alpha}|N=n\right) = \prod_{i=1}^{26} \frac{e^{-(\alpha_1 x_{i,1}+\alpha_2 x_{i,2})}\left(\alpha_1 x_{i,1}+\alpha_2 x_{i,2}\right)^{n_i}}{n_i!}$$

thus the log-likelihood is,

$$\ell(\boldsymbol{\alpha}|N=n) = -\sum_{i=1}^{26}(\alpha_1 x_{i,1}+\alpha_2 x_{i,2}) + \sum_{i=1}^{26} n_i \log(\alpha_1 x_{i,1}+\alpha_2 x_{i,2}) - \sum_{i=1}^{26}\log(n_i!)$$

The first derivatives are given by,

$$\frac{\partial}{\partial\alpha_1}\ell(\boldsymbol{\alpha}|N=n) = -\sum_{i=1}^{26} x_{i,1} + \sum_{i=1}^{26} n_i \frac{x_{i,1}}{\alpha_1 x_{i,1}+\alpha_2 x_{i,2}}, \tag{13}$$

$$\frac{\partial}{\partial\alpha_2}\ell(\boldsymbol{\alpha}|N=n) = -\sum_{i=1}^{26} x_{i,2} + \sum_{i=1}^{26} n_i \frac{x_{i,2}}{\alpha_1 x_{i,1}+\alpha_2 x_{i,2}}. \tag{14}$$

Similarly, the second derivatives are,

$$\frac{\partial^2}{\partial\alpha_1^2}\ell(\boldsymbol{\alpha}|N=n) = -\sum_{i=1}^{26} \frac{n_i x_{i,1}^2}{(\alpha_1 x_{i,1}+\alpha_2 x_{i,2})^2} \tag{15}$$

$$\frac{\partial^2}{\partial\alpha_2^2}\ell(\boldsymbol{\alpha}|N=n) = -\sum_{i=1}^{26} \frac{n_i x_{i,2}^2}{(\alpha_1 x_{i,1}+\alpha_2 x_{i,2})^2} \tag{16}$$

$$\frac{\partial^2}{\partial\alpha_1\partial\alpha_2}\ell(\boldsymbol{\alpha}|N=n) = -\sum_{i=1}^{26} \frac{n_i x_{i,1} x_{i,2}}{(\alpha_1 x_{i,1}+\alpha_2 x_{i,2})^2} \tag{17}$$

before giving the algorithms, we read the given data in R using the following commands:

```
> data=read.table("oilspills.txt",sep="", fill=FALSE, strip.white=TRUE)
> N=data[2]
> X1=data[3]
> X2=data[4]
```

(a) Define,

$$V(\boldsymbol{\alpha}|N=n) = \begin{pmatrix} \frac{\partial}{\partial\alpha_1}\ell(\boldsymbol{\alpha}|N=n) \\ \frac{\partial}{\partial\alpha_2}\ell(\boldsymbol{\alpha}|N=n) \end{pmatrix},$$

$$J(\boldsymbol{\alpha}|N=n) = \begin{pmatrix} \frac{\partial^2}{\partial\alpha_1^2}\ell(\boldsymbol{\alpha}|N=n) & \frac{\partial^2}{\partial\alpha_1\partial\alpha_2}\ell(\boldsymbol{\alpha}|N=n) \\ \frac{\partial^2}{\partial\alpha_1\partial\alpha_2}\ell(\boldsymbol{\alpha}|N=n) & \frac{\partial^2}{\partial\alpha_2^2}\ell(\boldsymbol{\alpha}|N=n) \end{pmatrix}.$$

$V(\boldsymbol{\alpha}|N=n)$ and $J(\boldsymbol{\alpha}|N=n)$ are called score function, Hessian matrix respectively, their values are given in the computation below. The NR scheme is given as follows:

$$\begin{cases} \boldsymbol{\alpha}^{(0)} = (\alpha_1^{(0)}, \alpha_2^{(0)}) \text{ starting point} \\ \boldsymbol{\alpha}^{(k+1)} = \boldsymbol{\alpha}^{(k)} - \left(J(\boldsymbol{\alpha}^{(k)}|N=n)\right)^{-1} V(\boldsymbol{\alpha}^{(k)}|N=n) \end{cases} \tag{18}$$

Implementation in R:

```
score_vector=function(alpha){
  V=matrix(,nrow=2,ncol=1)
  #According to equations (13-14):
  V[1]=sum(N*X1/(alpha[1]*X1+alpha[2]*X2)-X1)
  V[2]=sum(N*X2/(alpha[1]*X1+alpha[2]*X2)-X2)
  return(V)
}

hessian_matrix=function(alpha){
  H=matrix( ,nrow=2,ncol=2)
  #According to equations (15-17):
  H[1,1]=-sum(N*(X1)^2/(alpha[1]*X1+alpha[2]*X2)^2)
  H[1,2]=-sum(N*X1*X2/(alpha[1]*X1+alpha[2]*X2)^2)
  H[2,1]=-sum(N*X1*X2/(alpha[1]*X1+alpha[2]*X2)^2)
  H[2,2]=-sum(N*(X2)^2/(alpha[1]*X1+alpha[2]*X2)^2)
  return(H)
}

NR=function(alpha0, epsilon=10^{-6},maxit=1000){
  i=0
  stop=FALSE
  alpha=alpha0
  while((i<maxit)&&(stop==FALSE)){
    V=score_vector(alpha)
    H=hessian_matrix(alpha)
    #According to the algorithm in (18):
    new_alpha=alpha-solve(H)%*%V
    #Stopping condition :
    if((norm_vec(new_alpha-alpha)<epsilon)){#norm_vec is defined in Q3.
      stop=TRUE
    }
    alpha=new_alpha
    i=i+1
  }
  return(list(alphahat=alpha, numit=i))
}
```

(b) The Fisher-Information matrix $\mathcal{I}(\boldsymbol{\alpha})$ is given by,

$$[\mathcal{I}(\boldsymbol{\alpha})]_{h,k} = -E\left[\frac{\partial^2}{\partial\alpha_h\partial\alpha_k}\ell(\boldsymbol{\alpha}|N)\right] \tag{19}$$

$$= \sum_{i=1}^{26} \frac{x_{i,h}x_{i,k}}{\alpha_1 x_{i,1} + \alpha_2 x_{i,2}}, \qquad \text{for all } h,k \in \{0,1\}. \tag{20}$$

Therefore the Fisher-Schoring Scheme is given by,

$$\begin{cases} \boldsymbol{\alpha}^{(0)} = (\alpha_1^{(0)}, \alpha_2^{(0)}) \text{ starting point} \\ \boldsymbol{\alpha}^{(k+1)} = \boldsymbol{\alpha}^{(k)} + \big(\mathcal{I}(\boldsymbol{\alpha}^{(k)})\big)^{-1} V(\boldsymbol{\alpha}^{(k)}|N=n) \end{cases} \tag{21}$$

Implementation in R:

```
Fisher_info=function(alpha){
  I=matrix( ,nrow=2,ncol=2)
  #according to equation (20):
  I[1,1]=sum(X1^2/(alpha[1]*X1+alpha[2]*X2))
  I[1,2]=sum(X1*X2/(alpha[1]*X1+alpha[2]*X2))
  I[2,1]=sum(X1*X2/(alpha[1]*X1+alpha[2]*X2))
  I[2,2]=sum(X2^2/(alpha[1]*X1+alpha[2]*X2))
  return(I)
}

FS=function(alpha0, epsilon=10^{-6},maxit=1000){
  i=0
  stop=FALSE
  alpha=alpha0
  while((i<maxit)&&(stop==FALSE)){
    V=score_vector(alpha)
    I=Fisher_info(alpha)
    #according to the algorithm in (21)
    new_alpha=alpha+solve(I)%*%V
    #Stopping condition :
    if((norm_vec(new_alpha-alpha)<epsilon)){
      stop=TRUE
    }
    alpha=new_alpha
    i=i+1
  }
  return(list(alphahat=alpha,numit=i))
}
```

(c) Now let us call the two functions from the same starting point $\boldsymbol{\alpha}^{(0)} = (0.1, 0.5)$

```
> alpha0=c(0.1,0.5) #starting point
> NR(alpha0)
$alphahat
             [,1]
[1,]  1.0971525
[2,]  0.9375546

$numit
[1]  8

> FS(alpha0)
$alphahat
             [,1]
[1,]  1.0971524
[2,]  0.9375547

$numit
[1]  14
```

Moreover, the running times of the two methods are:

```
> system.time(NR(alpha0))
   user  system elapsed
  0.074   0.000   0.075
> system.time(FS(alpha0))
   user  system elapsed
  0.118   0.001   0.120
```

Conclusion: The Newton-Raphson method is better since it requires less iterations and less time to run than the Fisher-Scoring method.

(d) Let $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \hat{\alpha}_2)$ be the MLE of $\boldsymbol{\alpha}$. We know using the asymptotic distribution of the MLE that :

$$\hat{\alpha}_1 \overset{n\to\infty}{\sim} \mathcal{N}\left(\alpha_1, \frac{1}{n}\left[\mathcal{I}(\boldsymbol{\alpha})\right]_{1,1}\right)$$
$$\hat{\alpha}_2 \overset{n\to\infty}{\sim} \mathcal{N}\left(\alpha_2, \frac{1}{n}\left[\mathcal{I}(\boldsymbol{\alpha})\right]_{2,2}\right)$$

Hence the standard error of the MLEs of $\hat{\boldsymbol{\alpha}}$ are :

$$\mathbf{SE}(\hat{\alpha}_1) \approx \sqrt{\frac{1}{n}\left[\mathcal{I}(\boldsymbol{\alpha})\right]_{1,1}}$$
$$\mathbf{SE}(\hat{\alpha}_2) \approx \sqrt{\frac{1}{n}\left[\mathcal{I}(\boldsymbol{\alpha})\right]_{2,2}}$$

```
> alphahat=NR(alpha0)[[1]] #We get the MLEs by Newton-Raphson
> I=Fisher_info(alphahat) #the corresponding Fisher Information matrix
> SE1=sqrt(I[1,1]/26) #Standard error for alpha1
> SE1
[1] 0.7945017825
> SE2=sqrt(I[2,2]/26) #standard error for alpha2
> SE2
[1] 0.5505245082
```

Hence,

$$\mathbf{SE}(\hat{\alpha}_1) \approx 0.7945017825$$
$$\mathbf{SE}(\hat{\alpha}_2) \approx 0.5505245082$$