Ecole de Recherche CIMPA
Analyse et Probabilités
Université FHB, Abidjan
17-28 mars 2014

Correction TP cours 7, 1ère séance

Exercice 1.

```
> test1 <- function(n,m) { n*m }
> test1(3,6)
[1] 18
```

Exercice 2. *Simulation de réalisations de $(B_1, \ldots, B_n)$ avec $(B_t)_{t\in\mathbb{R}}$ un mouvement brownien standard.*

1. *Méthode de Choleski*

```
n=1000
#On initialise avec une matrice de zeros
R=matrix(0,n,n)
#On remplit la matrice de covariance
for (i in 1:n){
  for (j in 1:n){
    R[i,j]=min(i,j)}
}
#On obtient une racine de R par la méthode de Choleski
A=chol(R)
#On simule une réalisation d'un vecteur gaussien centré de matrice de covariance R
B=t(A)%*%rnorm(n)
plot(B,type="l")
```

On obtient le temps de calcul avec

```
ptm<-proc.time()
n=1000
R=matrix(0,n,n)
for (i in 1:n){
  for (j in 1:n){
    R[i,j]=min(i,j)}
}
A=chol(R)
B=t(A)%*%rnorm(n)
proc.time() -ptm
plot(B,type="l")
```

```
n=1000
 user  system elapsed
  17.33    0.03   17.43
n=100
    user  system elapsed
   0.29    0.02    0.31
```

Si on souhaite simuler $(B_{t_1}, \ldots, B_{t_n})$ *pour* $0 < t_1 < \ldots < t_n$ *il suffit de remplir la matrice* R *en remplaçant* $min(i, j)$ *par* $min(t_i, t_j)$*. Lorsque l'on souhaite faire partir le brownien de la valeur* 0, $B_0 = 0$ *p.s., on peut remarquer que* $(B_{t_1} - B_{t_1}, \ldots, B_{t_n} - B_{t_1}) \overset{d}{=} (B_0, B_{t_2 - t_1}, \ldots, B_{t_n - t_1})$ *par accroissements stationnaires.*

2. *On peut également remarquer que* $A^t = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 \\ 1 & \ldots & \ddots & 1 \end{pmatrix}$ *et*

$$(B_1, B_2, \ldots, \ldots B_n) \overset{d}{=} (\varepsilon_1, \varepsilon_1 + \varepsilon_2, \ldots, \sum_{k=1}^{n} \varepsilon_k).$$

Ainsi on a un algorithme beaucoup plus rapide :

```
> ptm<-proc.time()
> n=1000
> B=cumsum(rnorm(n))/sqrt(n)
> proc.time() -ptm
   user  system elapsed
   0.00    0.02    0.01
```

Exercice 3.

```
CircEmbCov <-function(r)
{n=length(r)-1
rr=rev(r)
s=c(r,rr[2:n])
z=fft(s)
print(min(Re(z)))}
```

1. ```
   > n=2^8
   > r=exp(-(0:n)/n)
   > CircEmb(r)
   [1] 0.001234609
   > n=2^9
   > r=exp(-(0:n)/n)
   > CircEmb(r)
   [1] 0.000617305
   > n=2^(10)
   > r=exp(-(0:n)/n)
   > CircEmb(r)
   [1] 0.0003086526
   ```
   ```

2.
```
> n=2^8
> r=exp(-((0:n)/n)^2)
> CircEmb(r)
[1] -9.905638
> n=2^9
> r=exp(-((0:n)/n)^2)
> CircEmb(r)
[1] -19.81056
> n=2^(10)
> r=exp(-((0:n)/n)^2)
> CircEmb(r)
[1] -39.62075
```

3.
```
> n=2^8
> r=exp(-((0:n)/n)^(1/2))
> CircEmb(r)
[1] 0.04618694
> n=2^9
> r=exp(-((0:n)/n)^(1/2))
> CircEmb(r)
[1] 0.03293234
> n=2^(10)
> r=exp(-((0:n)/n)^(1/2))
> CircEmb(r)
[1] 0.0234238
```

Exercice 4.

```
CircEmbSim <-function(r)
{
n=length(r)-1

#Matrice circulante
rr=rev(r)
s=c(r,rr[2:n])

#Valeurs propres et racines
shat=fft(s)
sq=shat^(1/2)
print(min(Re(shat)))

#Vecteur gaussien
e1=rnorm(c)
e2=rnorm(c)
e=e1+1i*e2
z=sq*e
zf=(2*n)^(-(1/2)*fft(z,inverse=TRUE)
x=Re(zf)
# pour une deuxième réalisation indépendante on choisit y=Im(zf)

#Vecteur gaussien de covariance r
```

```
g=x[1:(n+1)]
plot(g,type='l')}

ChoToepSim <-function(r)
{n=length(r)-1
R=toeplitz(r)
A=chol(R)
B=t(A)%*%rnorm(n+1)
plot(B,type="l")
}
```

1.
```
> n=2^(10)
> r=exp(-((0:n)/n))
> system.time(CircEmbSim(r))
[1] 0.0003086526
   user  system elapsed
   0.08    0.04    0.11
> system.time(ChoToepSim(r))
   user  system elapsed
   1.23    0.11    1.34

> n=2^(11)
> r=exp(-((0:n)/n))
> system.time(CircEmbSim(r))
[1] 0.0001543263
   user  system elapsed
   0.08    0.05    0.13
> system.time(ChoToepSim(r))
   user  system elapsed
   9.47    0.31    9.84
```

2.
```
> system.time(CircEmbSim(r))
[1] 0.0234238
   user  system elapsed
   0.03    0.08    0.11
> system.time(ChoToepSim(r))
   user  system elapsed
   1.33    0.16    1.48
> n=2^(11)
> r=exp(-((0:n)/n)^(1/2))
> system.time(CircEmbSim(r))
[1] 0.01663187
   user  system elapsed
   0.08    0.08    0.15
> system.time(ChoToepSim(r))
   user  system elapsed
   9.39    0.39    9.79
```

Exercice 5.

1.
```
CovMBF <- function(H,n){
k=0:n
```

```
  H2=2*H
  k1=abs(k-rep(1,length(k)))
  k2=abs(k+rep(1,length(k)))
  r=0.5*(k2^(H2)+k1^(H2)-2*abs(k)^(H2))
  r
  }
```

2.
```
n=2^(10)
for (H in seq(0.1,0.9,0.1)){
  r=CovMBF(H,n)
  CircEmb(r)
}
[1] 0.0007812502
[1] 0.006250001
[1] 0.0375
[1] 0.2
[1] 1
[1] 0.7878123
[1] 0.5777895
[1] 0.3736131
[1] 0.1794617
n=2^(11)
[1] 0.0004487103
[1] 0.004123462
[1] 0.02841969
[1] 0.1741101
[1] 1
[1] 0.7878125
[1] 0.5777903
[1] 0.3736168
[1] 0.1794716
```

Exercice 6.

```
MBF <- function(H,n){
  r=CovMBF(H,n)

  #Matrice circulante
  rr=rev(r)
s=c(r,rr[2:n])

  #Valeurs propres et racines
  shat=fft(s)
  sq=shat^(1/2)
  print(min(Re(shat)))

  #Bruit gaussien fractionnaire
  e1=rnorm(c)
  e2=rnorm(c)
  e=e1+1i*e2
z=sq*e
```

```
zf=(2*n)^(-(1/2)*fft(z,inverse=TRUE)
x=Re(zf)
  #Mouvement brownien fractionnaire
  B=n^(-H)*cumsum(g)
  plot(B,type='l')}
```