UNIVERSITÉ DE GENÈVE
Sections d'Informatique et de Physique

FACULTÉ DES SCIENCES
Profs. B. Chopard et M. Droz

Outils, modélisation et simulation en calcul numérique – Corrigé série 9

17 mai 2005

Exercice 1.

i) Déterminons l'issue de l'épreuve (position de l'aiguille) par deux nombres : l'abscisse x du centre de l'aiguille par rapport à la droite la plus proche à gauche, et l'angle φ que fait l'aiguille avec la direction des droites (cf. Fig. 1).

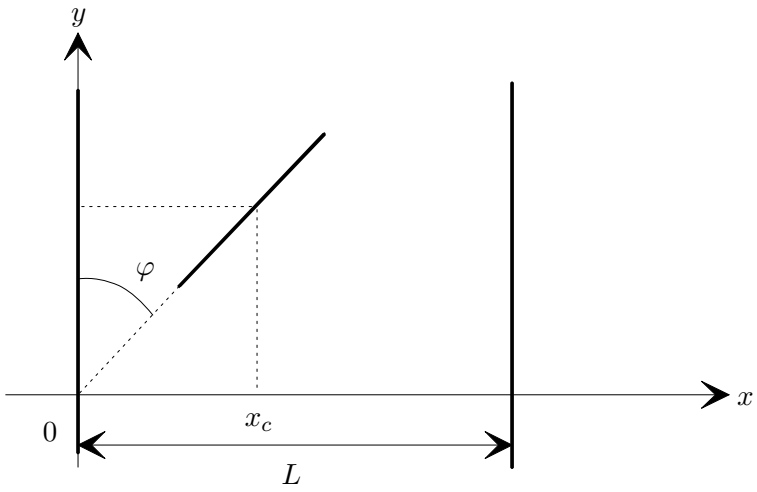

FIG. 1 – Coordonnées de la position de l'aiguille.

Nous entendons par le fait que l'aiguille est jetée au hasard, que toutes les valeurs de x et de ϕ sont équiprobables. Sans restreindre la généralité on peut considérer les valeurs

$$(x, \varphi) \in \Omega = [0, L/2] \times [0, \pi], \tag{1}$$

et envisager le croisement éventuel seulement pour la droite la lus proche à gauche. L'aire de l'espace des événements élémentaires Ω est donc $S_\Omega = L\pi/2$. L'aiguille croisera la droite en $x = 0$ si l'abscisse x_c est inférieure à $(\ell/2)\sin\varphi$ (cf. Fig. 1). L'événement $\mathcal{A}$ qui nous intéresse,

$$\mathcal{A} = \{(x_c, \varphi) \in \Omega | x_c < (\ell/2)\sin\varphi\}, \tag{2}$$

a une aire

$$S_\mathcal{A} = \left| \int_0^\pi \mathrm{d}\varphi \int_0^{(\ell/2)\sin\varphi} \mathrm{d}x \right| = \left| \int_0^\pi \mathrm{d}\varphi \, \frac{\ell}{2} \sin\varphi \right| = 2\frac{\ell}{2}. \tag{3}$$

La probabilité cherchée est donc

$$P(\mathcal{A}) = \frac{S_\mathcal{A}}{S_\Omega} = \frac{2\ell}{\pi L}. \tag{4}$$

Cette formule obtenue par Georges-Louis Leclerc, comte de Buffon (7 septembre 1707 - 16 avril 1788), naturaliste, mathématicien, biologiste, cosmologiste et auteur français (cf. Fig. 2), a été vérifiée expérimentalement à plus d'une reprise, en calculant à cet effet la fréquence des croisements dans une longue série de jets. On s'en est servi pour faire une estimation approchée du nombre π, dont les résultats ont été parfaitement satisfaisants.

FIG. 2 – Portrait de Buffon. Une statue de Buffon se trouve au jardin des plantes de Paris (1908).

ii) Programme Fortran 90 :

```
module fonctions
module fonctions
implicit none
integer, parameter :: N=2*10**9 !number of throws
real, parameter :: length=0.5 !L=1, \ell = length of the nail < 1
real, parameter :: pi = 3.14159265358979323846264338327950288419 7
character*(*), parameter :: filename = 's9-1ii.txt'
contains
!-----------------------------------------------------------------------
FUNCTION ran2(idum) !Initialize idum with negative integer.
INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
REAL :: ran2,AM,EPS,RNMX
PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1,IA1=40014,&
          &IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,IR2=3791,NTAB=32,&
          &NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
INTEGER :: idum2,j,k,iv(NTAB),iy
SAVE iv,iy,idum2
DATA idum2/123456789/, iv/NTAB*0/, iy/0/
if(idum.le.0)then
  idum=max(-idum,1)
  idum2=idum
  do j=NTAB+8,1,-1
    k=idum/IQ1
    idum=IA1*(idum-k*IQ1)-k*IR1
    if(idum.lt.0)idum=idum+IM1
    if(j.le.NTAB)iv(j)=idum
  enddo
  iy=iv(1)
endif
```

```
k=idum/IQ1
idum=IA1*(idum-k*IQ1)-k*IR1
if(idum.lt.0)idum=idum+IM1
k=idum2/IQ2
idum2=IA2*(idum2-k*IQ2)-k*IR2
if(idum2.lt.0)idum2=idum2+IM2
j=1+iy/NDIV
iy=iv(j)-idum2
iv(j)=idum
if(iy.lt.1)iy=iy+IMM1
ran2=min(AM*iy,RNMX)
return
end function ran2
!-----------------------------------------------------------------------
end module fonctions
program s9ex1ii
use fonctions
implicit none
real :: logplot
integer :: seed,cross,i
seed = -14731
logplot=0.
cross=0
open(unit=1,file=filename)
close(1,status='DELETE')
do i=1,N
  if (ran2(seed)/2. < (length/2.)*sin(ran2(seed)*pi)) cross = cross+1
  if ((log(real(i)) - logplot > 0.05) .or. (i==1)) then !linear # in log plot
    logplot=log(real(i))
    open(unit=1,file=filename,position='APPEND')
    if (cross > 0) write(1,*) i,2.d0*dble(length)*(dble(i)/dble(cross))
    close(1)
  end if
end do
end program s9ex1ii
```

Pour certaines simulations Monte Carlo pour lesquelles on peut faire des moyennes avec de l'ordre de 10^9 échantillons, il est vivement conseillé d'utiliser la double précision. La Fig. 3 représente la convergence de l'algorithme vers le nombre π en fonction du nombre de jets N. Soit $\Pi(N)$ la valeur de π obtenue par l'algorithme, alors $\Pi(N) = \pi + b/N^{\beta}$, et donc en prenant le log on obtient $\log[\Pi(N) - \pi] = \log b - \beta \log N$. En représentant $\log[\Pi(N) - \pi]$ en fonction de $\log N$ on peut donc trouver l'exposant de déclin β par interpolation linéaire.

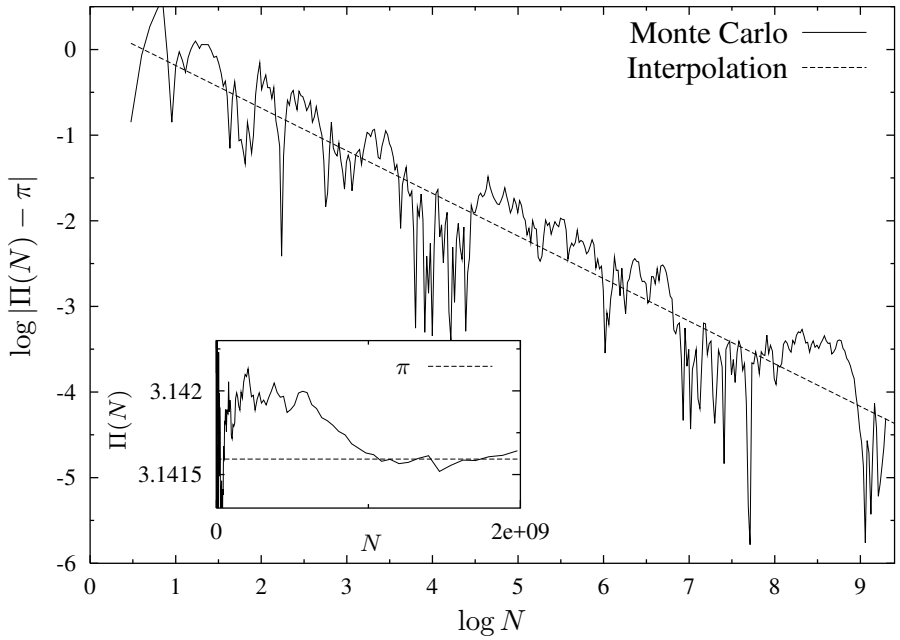

FIG. 3 – Convergence vers π en $N^{-\beta}$. La figure principale est une représentation log-log, tandis que la petite figure est une représentation lin-lin. Les paramètres sont $N = 2 \times 10^9$, $\ell = 0.5$, $L = 1$. Le bruit statistique reste grand, et pour le diminuer il serait nécessaire de faire une moyenne sur des réalisations indépendantes. Néanmoins, ceci ne s'avère pas nécessaire pour extraire une approximation grossière de β. Par interpolation linéaire, on trouve $\beta = 0.498\ldots$, soit une valeur proche de celle attendue $\beta = 1/2$.

Exercice 2.

i) Notons que sur les machines 32 bits le plus grand entier positif est $2^{31} - 1$. La foncion "randu" en Fortran 90 s'écrit :

```
function randu(iseed)
real :: randu,XMAX_INV
integer :: iseed,IMAX
parameter (IMAX=2147483647,XMAX_INV=1./IMAX)
iseed = iseed*65539
if (iseed < 0) iseed = iseed + IMAX + 1
randu = real(iseed)*XMAX_INV
return
end function randu
```

L'opération mod m est réalisée en ajoutant 2^{31} si $iseed < 0$. Comme 2^{31} n'existe pas sur les machines à 32 bits, on ajoute $2^{31} - 1$ (c'est-à-dire $IMAX$) puis ensuite 1.
En Mathematica, la fonction "randu" s'écrit

```
a=65539; m=2^31;
randu[i_] := Mod[a*i,m]
```

ii) La distribution des nombres aléatoires étant uniforme sur $[0,1]$, le résultat à trouver est donné par $\int_0^1 \mathrm{d}x\, x^k = 1/(k+1)$. Programme Fortran 90 :

```
module fonctions
implicit none
integer, parameter :: N=1000000 !number of points for average
integer, parameter :: kmax=50 !kmax values to test
character*(*), parameter :: filename = 's9-2ii.txt'
contains
!-----------------------------------------------------------------------
function randu(iseed)
real :: randu,XMAX_INV
integer :: iseed,IMAX
parameter (IMAX=2147483647,XMAX_INV=1./IMAX)
iseed = iseed*65539
if (iseed < 0) iseed = iseed + IMAX + 1
randu = real(iseed)*XMAX_INV
return
end function randu
!-----------------------------------------------------------------------
end module fonctions
program s9ex2ii
use fonctions
implicit none
double precision, dimension(1:kmax) :: moments
integer :: iseed,i,j
iseed= 123456
moments=0.d0
do i=1,N
  do j=1,kmax
    moments(j)=moments(j)+dble(randu(iseed))**j
  end do
end do
open(unit=1,file=filename)
do i=1,kmax
  write(1,*) i,moments(i)/dble(N),1./(1.+real(i))
end do
close(1)
end program s9ex2ii
```

La Fig. 4 montre que les moments issus du générateur "randu" sont en excellent avec la prédiction théorique.

Ceci n'est néanmoins pas un critère suffisant pour décider de la qualité d'un générateur de nombre aléatoires. Il est nécessaire d'étudier les corrélations entre nombres aléatoires.

iii) Considérons les triplets $y_i = (x_{i1}, x_{i2}, x_{i3})$ de nombres aléatoires, et représentons-les en 3 dimensions. Si ces triplets sont non corrélés, alors ils doivent remplir le cube unité de façon homogène.

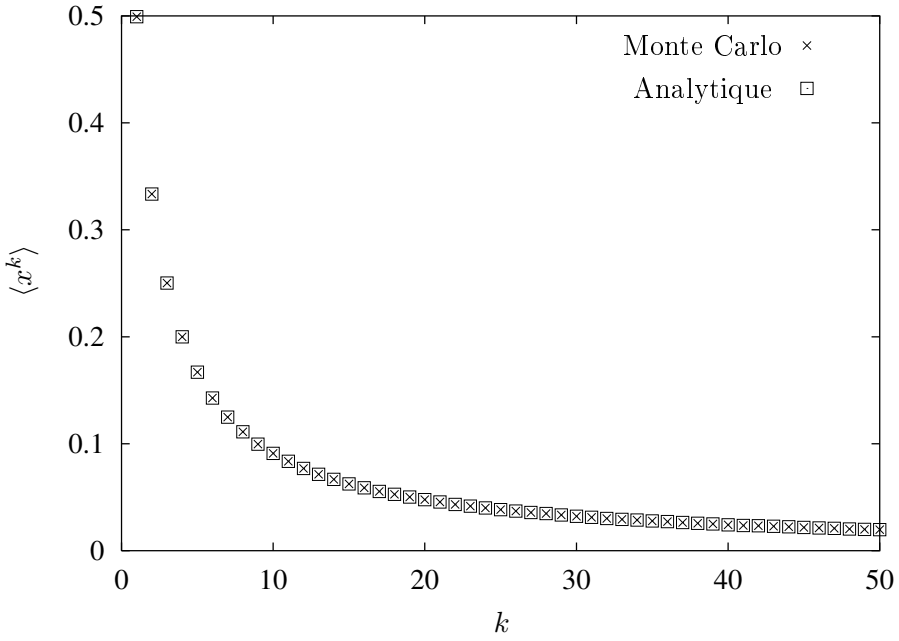

FIG. 4 – Moments du générateur de nombres aléatoires "randu" et résultat exact. Chaque point est issu de 10^6 nombres aléatoires.

Néanmoins, *tous* les triplets des nombres aléatoires générés par "randu" sont disposés sur seulement 15 plans : la combinaison $9x_{i1} - 6x_{i2} + x_{i3}$ est un entier prenant des valeurs entre -5 et 9. Le fait que ces triplets soient répartis sur ces 15 plans ne dépend pas du choix initial de *iseed*. La Fig. 5 représente 5×10^3 tels triplets.

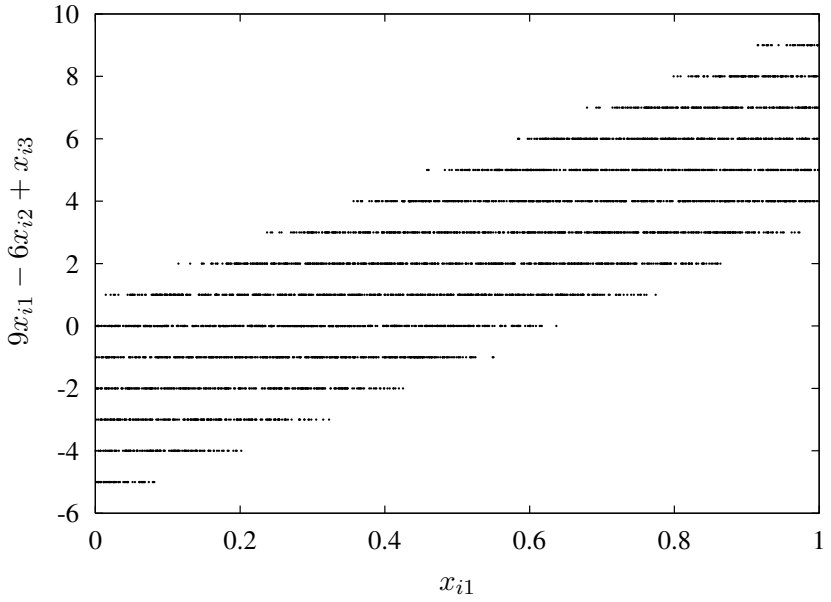

FIG. 5 – Figure montrant que tous les triplets de nombres aléatoires consécutifs générés par "randu" sont situés sur un des 15 plans $9x_{i1} - 6x_{i2} + x_{i3} = m$, $m = -5, -4, \ldots, 8, 9$. Nombre de points : $N = 5'000$.

Le tableau ci-dessous donne les trois premiers triplets pour $iseed = 123456$.

x_{i1}	x_{i2}	x_{i3}	$9x_{i1} - 6x_{i2} + x_{i3}$
0.767751	0.605986	0.726162	4.00000
0.903094	0.883110	0.170816	3.00000
0.076899	0.924058	0.852253	-4.00000

On peut utiliser Mathematica pour produire une image en 3 dimensions des triplets obtenus par "randu" :

```
a = 65539; m = 2^31; i0 = 314159; npoint = 3000; nrand = 3*npoint;
randu[i_] := Mod[a*i, m]
uniform = Drop[NestList[randu, i0, nrand], 1]/N[m];
triple = Table[Take[uniform, {n, n + 2}], {n, 1, nrand - 2, 3}];
Show[Graphics3D[Map[Point, triple]], ViewPoint -> {10, 10, -23}];
```

La Fig. 6 montre clairement que les points sont situés sur 15 plans.

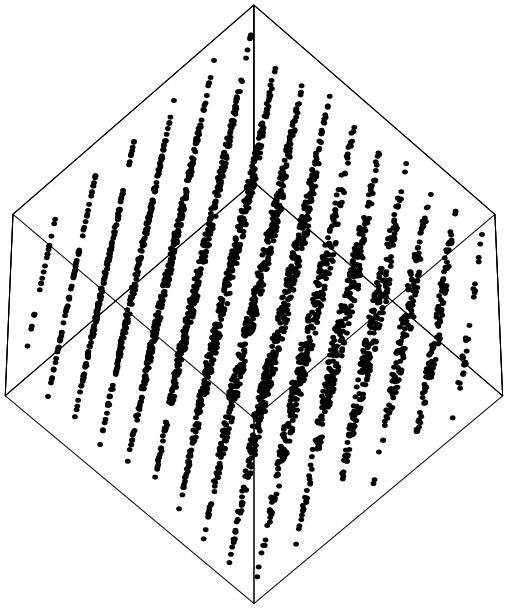

FIG. 6 – Triplets du générateur "randu" répartis dans les plans.

Un générateur de nombres aléatoires adéquat dans la majeure partie des cas est le générateur "ran2" des *Numerical Recipes* (le code Fortran 90 est donné à la fin du corrigé de l'exercice). En utilisant ce générateur, on obtient une répartition homogène des points, comme le montre la Fig. 7.
Le code Fortran 90 permettant de générer les fichiers pour les Figs. 5 et 7 est :

```
module fonctions
implicit none
integer, parameter :: N=5000 !number of points
character*(*), parameter :: filename1 = 's9-2iiia.txt'
```

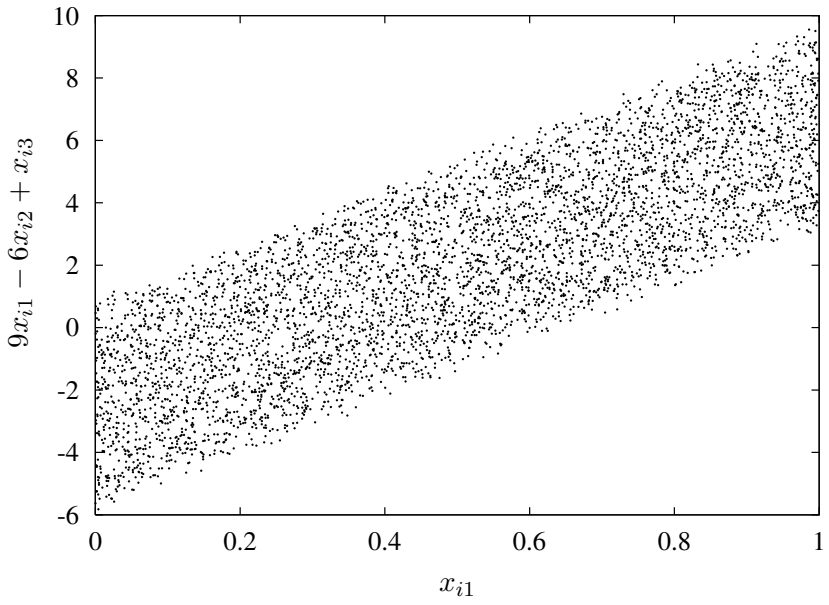

FIG. 7 – Analyse identique à celle de la Fig. 5, mais pour le générateur aléatoire "ran2" des *Numerical Recipes*. Nombre de points : $N = 5'000$.

```
character*(*), parameter :: filename2 = 's9-2iiib.txt'
contains
!-----------------------------------------------------------------------
FUNCTION ran2(idum) !Initialize idum with negative integer.
INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
REAL :: ran2,AM,EPS,RNMX
PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1,IA1=40014,&
          &IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,IR2=3791,NTAB=32,&
          &NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
INTEGER :: idum2,j,k,iv(NTAB),iy
SAVE iv,iy,idum2
DATA idum2/123456789/, iv/NTAB*0/, iy/0/
if(idum.le.0)then
  idum=max(-idum,1)
  idum2=idum
  do j=NTAB+8,1,-1
    k=idum/IQ1
    idum=IA1*(idum-k*IQ1)-k*IR1
    if(idum.lt.0)idum=idum+IM1
    if(j.le.NTAB)iv(j)=idum
  enddo
  iy=iv(1)
endif
k=idum/IQ1
idum=IA1*(idum-k*IQ1)-k*IR1
if(idum.lt.0)idum=idum+IM1
k=idum2/IQ2
```

```
idum2=IA2*(idum2-k*IQ2)-k*IR2
if(idum2.lt.0)idum2=idum2+IM2
j=1+iy/NDIV
iy=iv(j)-idum2
iv(j)=idum
if(iy.lt.1)iy=iy+IMM1
ran2=min(AM*iy,RNMX)
return
end function ran2
!------------------------------------------------------------------------
function randu(iseed)
real :: randu,XMAX_INV
integer :: iseed,IMAX
parameter (IMAX=2147483647,XMAX_INV=1./IMAX)
iseed = iseed*65539
if (iseed < 0) iseed = iseed + IMAX + 1
randu = real(iseed)*XMAX_INV
return
end function randu
!------------------------------------------------------------------------
end module fonctions
program s9ex2iii
use fonctions
implicit none
real :: xi1,xi2,xi3
integer :: seed,iseed,i
seed = -14731
iseed= 123456
open(unit=1,file=filename1)
do i=1,N
  xi1=randu(iseed)
  xi2=randu(iseed)
  xi3=randu(iseed)
  write(1,*) xi1,xi2,xi3,9*xi1-6*xi2+xi3
end do
close(1)
open(unit=1,file=filename2)
do i=1,N
  xi1=ran2(seed)
  xi2=ran2(seed)
  xi3=ran2(seed)
  write(1,*) xi1,xi2,xi3,9*xi1-6*xi2+xi3
end do
close(1)
end program s9ex2iii
```

Exercice 3.

i) Programme Matlab pour I_N :

```
module fonctions
implicit none
integer, parameter :: N=1000000000 !number of points
character*(*), parameter :: filename = 's9-3i.txt'
contains
!---------------------------------------------------------------------
FUNCTION ran2(idum) !Initialize idum with negative integer.
INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
REAL :: ran2,AM,EPS,RNMX
PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1,IA1=40014,&
          &IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,IR2=3791,NTAB=32,&
          &NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
INTEGER :: idum2,j,k,iv(NTAB),iy
SAVE iv,iy,idum2
DATA idum2/123456789/, iv/NTAB*0/, iy/0/
if(idum.le.0)then
  idum=max(-idum,1)
  idum2=idum
  do j=NTAB+8,1,-1
    k=idum/IQ1
    idum=IA1*(idum-k*IQ1)-k*IR1
    if(idum.lt.0)idum=idum+IM1
    if(j.le.NTAB)iv(j)=idum
  enddo
  iy=iv(1)
endif
k=idum/IQ1
idum=IA1*(idum-k*IQ1)-k*IR1
if(idum.lt.0)idum=idum+IM1
k=idum2/IQ2
idum2=IA2*(idum2-k*IQ2)-k*IR2
if(idum2.lt.0)idum2=idum2+IM2
j=1+iy/NDIV
iy=iv(j)-idum2
iv(j)=idum
if(iy.lt.1)iy=iy+IMM1
ran2=min(AM*iy,RNMX)
return
end function ran2
!---------------------------------------------------------------------
function f(x)
double precision :: f
real, intent(in) :: x
f=1.d0/(1.d0+dble(x)**2)
```

```
return
end function f
!---------------------------------------------------------------------
end module fonctions
program s9ex3i
use fonctions
implicit none
integer :: seed,i
real :: logplot
double precision :: result
seed = -14731
result=0.d0
logplot=0.
open(unit=1,file=filename)
close(1,status='DELETE')
do i=1,N
  result = result+f(ran2(seed))
  if ((log(real(i)) - logplot > 0.05) .or. (i==1)) then !linear # in log plot
    logplot=log(real(i))
    open(unit=1,file=filename,position='APPEND')
    write(1,*) i,result/dble(i)
    close(1)
  end if
end do
end program s9ex3i
```

Pour $\widetilde{I}_N$, on a $u(x) = \int_0^x \mathrm{d}y\, \omega(y) = 4x/3 - x^2/3$ et donc $x(u) = 2 - \sqrt{4-3u}$. Ainsi.

$$g(u) = \frac{1}{1 + (2 - \sqrt{4-3u})^2} \left[\frac{4 - 2(2 - \sqrt{4-3u})}{3} \right]^{-1}. \tag{5}$$

Programme Matlab :

```
module fonctions
implicit none
integer, parameter :: N=1000000000 !number of points
character*(*), parameter :: filename = 's9-3ii.txt'
contains
!---------------------------------------------------------------------
FUNCTION ran2(idum) !Initialize idum with negative integer.
INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
REAL :: ran2,AM,EPS,RNMX
PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1,IA1=40014,&
          &IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,IR2=3791,NTAB=32,&
          &NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
INTEGER :: idum2,j,k,iv(NTAB),iy
SAVE iv,iy,idum2
DATA idum2/123456789/, iv/NTAB*0/, iy/0/
if(idum.le.0)then
```

```
  idum=max(-idum,1)
  idum2=idum
  do j=NTAB+8,1,-1
    k=idum/IQ1
    idum=IA1*(idum-k*IQ1)-k*IR1
    if(idum.lt.0)idum=idum+IM1
    if(j.le.NTAB)iv(j)=idum
  enddo
  iy=iv(1)
endif
k=idum/IQ1
idum=IA1*(idum-k*IQ1)-k*IR1
if(idum.lt.0)idum=idum+IM1
k=idum2/IQ2
idum2=IA2*(idum2-k*IQ2)-k*IR2
if(idum2.lt.0)idum2=idum2+IM2
j=1+iy/NDIV
iy=iv(j)-idum2
iv(j)=idum
if(iy.lt.1)iy=iy+IMM1
ran2=min(AM*iy,RNMX)
return
end function ran2
!------------------------------------------------------------------------
function g(u) !g(u)=f[x(u)]/w[x(u)]
double precision :: g,du,tmp
real, intent(in) :: u
du=dble(u)
tmp=dsqrt(4.d0-3.d0*du)
g=3.d0/((18.d0 -8.d0*tmp-6.d0*du)*tmp)
return
end function g
!------------------------------------------------------------------------
end module fonctions
program s9ex3ii
use fonctions
implicit none
integer :: seed,i
real :: logplot
double precision :: result
seed = -14731
result=0.d0
logplot=0.
open(unit=1,file=filename)
close(1,status='DELETE')
do i=1,N
result = result+g(ran2(seed))
if ((log(real(i)) - logplot > 0.05) .or. (i==1)) then !linear # in log plot
```

```
logplot=log(real(i))
open(unit=1,file=filename,position='APPEND')
write(1,*) i,result/dble(i)
close(1)
end if
end do
end program s9ex3ii
```

ii) I_N tendra vers I en $N^{-1/2}$. Si $\omega(x)$ est bien choisie, $\widetilde{I}_N$ tendra plus vite que I_N vers I car la variance $\sigma_{\tilde{I}}$ pourra être beaucoup plus petite que σ_I. En effet, si les différents échantillonnages (des x_i ou u_i) sont statistiquement indépendants, on a

$$\sigma_{\tilde{I}}^2 = \frac{1}{N}\left[\langle g^2\rangle - \langle g\rangle^2\right], \qquad g(x) = f(x)/\omega(x), \tag{6}$$

alors que

$$\sigma_I^2 = \frac{1}{N}\left[\langle f^2\rangle - \langle f\rangle^2\right]. \tag{7}$$

Si $\omega(x)$ est choisie de telle manière à ce que $f(x)/\omega(x)$ varie peu en x [ce qui revient donc à choisir $\omega(x)$ "proche" de $f(x)$], alors $\sigma_{\tilde{I}}^2 \ll \sigma_I^2$.
La Fig. 8 représente $f(x) = 1/(1+x^2)$ ainsi que $\omega(x) = (4-2x)/3$.

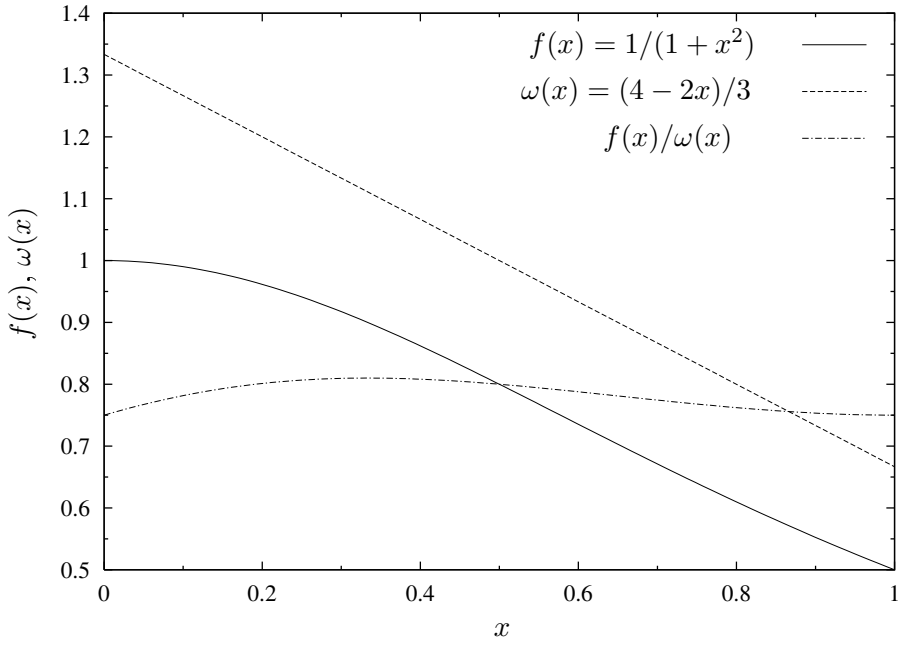

FIG. 8 – Représentation des fonctions $f(x) = 1/(1+x^2)$ ainsi que $\omega(x) = (4-2x)/3$ pour $x \in [0,1]$.

Les Figs. 9 et 10 représentent I_N et $\widetilde{I}_N$ en fonction de N, respectivement.
Même si les pentes obtenues par interpolation ne diffèrent guère entre I_N et $\widetilde{I}_N$, on constate néanmoins en comparant les Figs. 9 et 10 que $\widetilde{I}_N$ converge nettement plus vite vers $\pi/4$ que ne le fait I_N. Cette différence de vitesse de convergence est d'autant plus marquée que la fonction $f(x)$ est piquée.

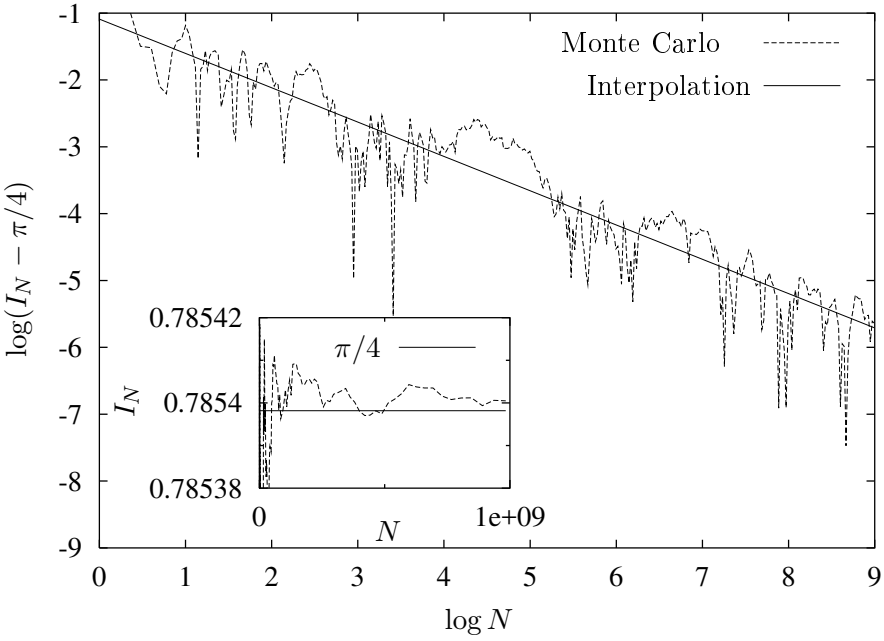

FIG. 9 – Diagramme log-log de $I_N(N)$. La pente de l'interpolation linéaire est de $-0.513\ldots$.

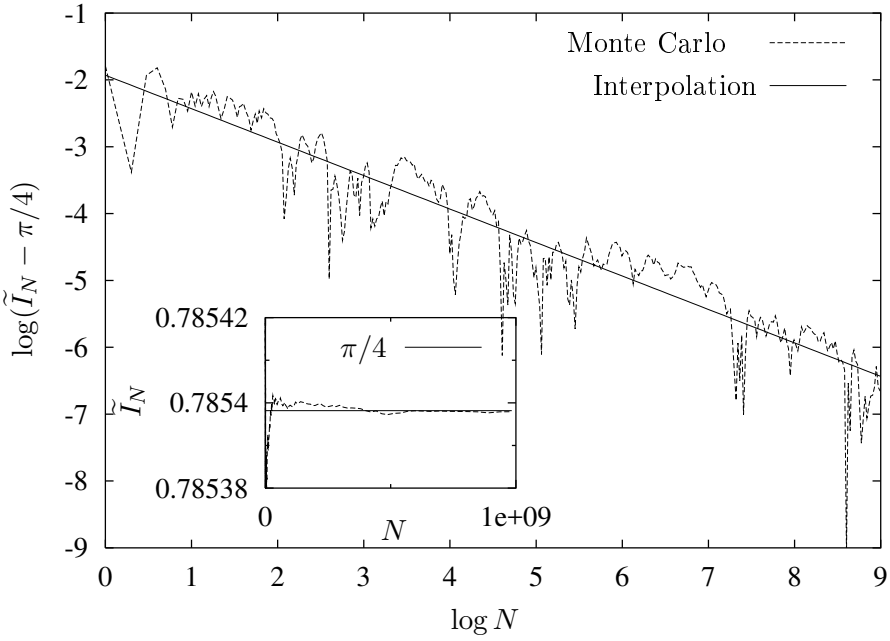

FIG. 10 – Diagramme log-log de $\widetilde{I}_N(N)$. La pente de l'interpolation linéaire est de $-0.501\ldots$.