

Université Abdelmalek Essaadi

Faculté des sciences Tétouan



**DEPARTEMENT DE SCIENCE DE LA MATIERE PHYSIQUE
Master PMR**

Le compte rendu de simulations de lois rectangulaires

Pr. Tarik EL BARDOUNI

➤ **Réalisé par : EL KABIL Soufiane
ISSAAD Anasss
CHOURAK Aissam**

2015/2016

Sommaire

- I. Introduction
- II. Générer des variables aléatoires entre $[0,1]$.
- III. Générer des variables aléatoires entre $[-1,1]$.
- IV. Calcul de π par la méthode de rejet.
- V. Calcul de π par la méthode d'intégration.
- VI. Calcul de π par la méthode de l'aiguille de Buffon.
- VII. Calcul de l'angle solide par la méthode de rejet.
 - A. Par disque.
 - B. Par rectangle.
 - C. Par carrée.
- VIII. Conclusion

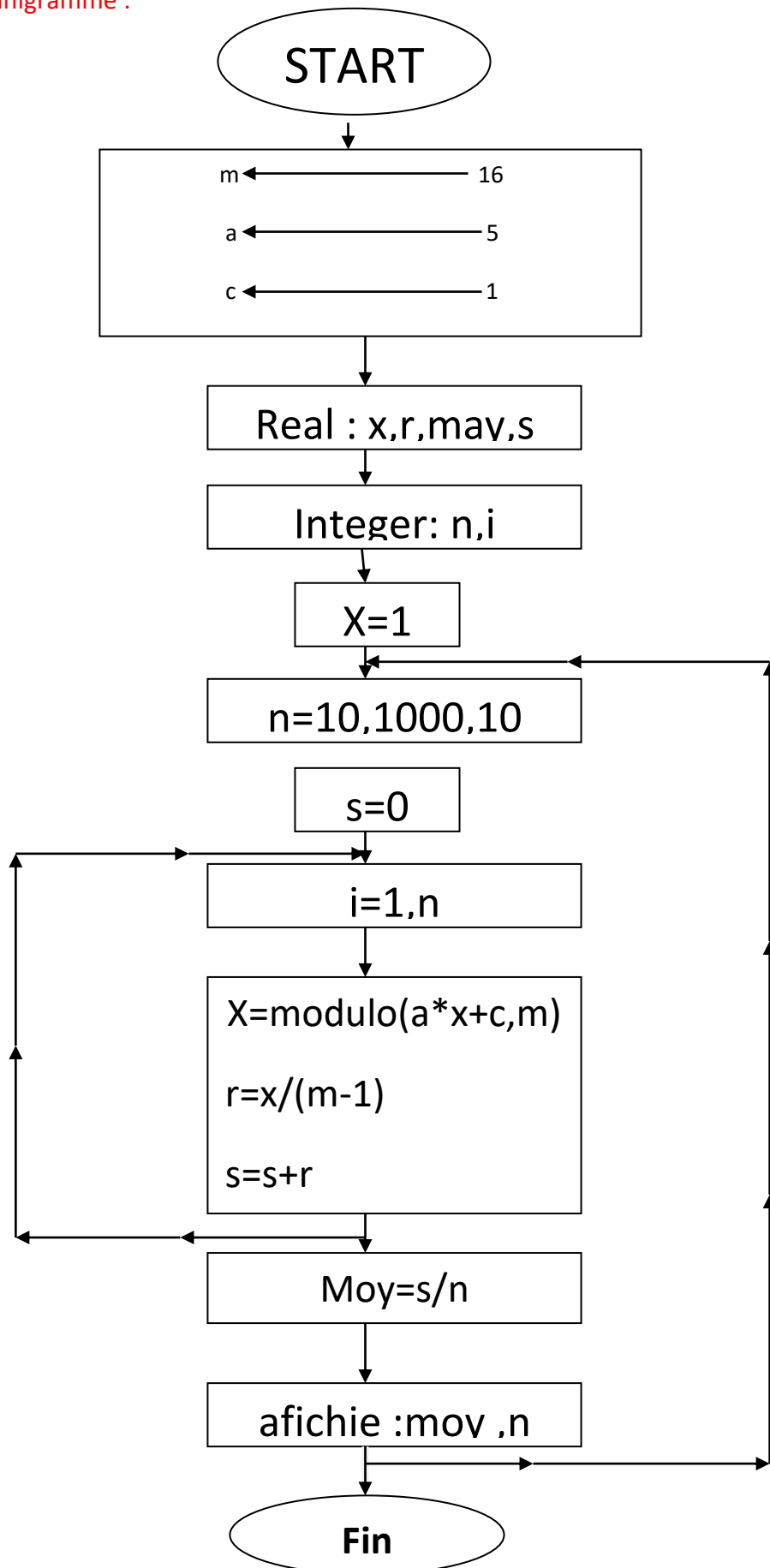
I. Introduction :

Le terme **méthode de Monte-Carlo**, désigne une famille de méthodes algorithmiques visant à calculer une valeur numérique approchée en utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes.

Dans ce travail on va traiter plusieurs exercices par différentes méthodes pour calculer la valeur de π , Tel que : Méthode de rejet, Méthode d'intégration MC , Méthode de l'aiguille de Buffon , finalement on calcul l'angle solide après de connaitre comment générer des variables aléatoires .

II. Générer des variables aléatoires entre [0,1] :

1. Organigramme :



2. Algorithme :

Variable n,i : integer

X, r, moy , s ,moy1 , var : real

m ← 16 : Real

a ← 5 : Real

c ← 1 : Real

Debut

X ← 1

Pour n ← 10 a 1000 par 10 faire

S ← 0

Pour i ← 1 a n faire

x ← modulo(a*x+c, m)

r ← x/(m-1)

s ← s+r

fin pour

Moy ← s/n

Moy1 ← (s*s)/n

var ← (moy1-(moy*moy))/n

Ecrire (' la valeur de n : ', n)

Ecrire (' la valeur de moy : ', moy)

Ecrire (' la valeur de var: ', var)

Fin pour

Fin

3. Programme :

```
program exercice1
implicit none
real,parameter :: m=16., a=5., c=1.
real :: x,r,moy,s,moy1,var
integer :: n ,i

open(unit=5,file='ex1.xls')
x=1

do n=10,1000,10
  s=0.

  do i=1,n

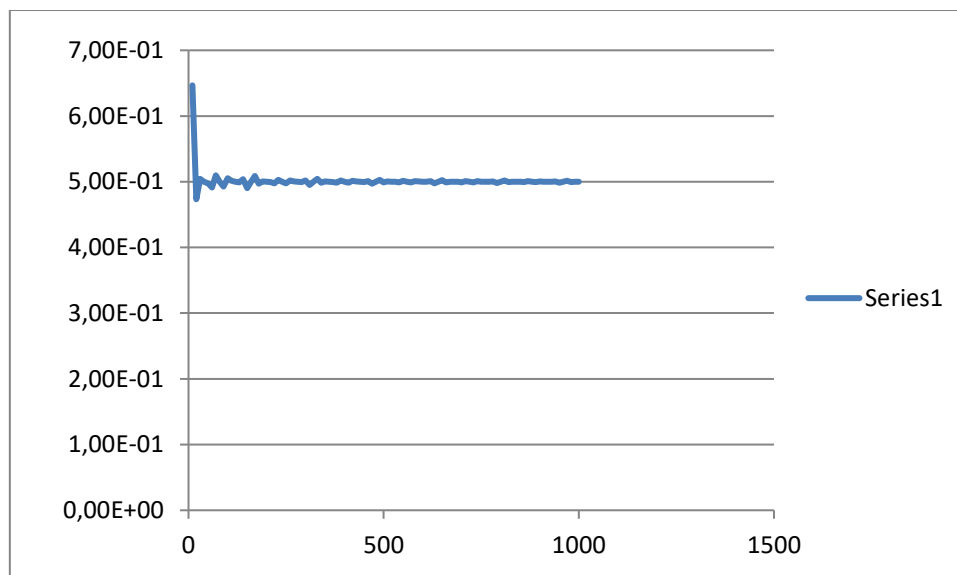
    x=modulo(a*x+c,m)
    r=x/(m-1)
    s=s+r

  enddo
  moy=s/n
  moy1=(s*s)/n
  var=(moy1-(moy*moy) )/n
  write(5,*)n,moy,var

enddo
end program
```

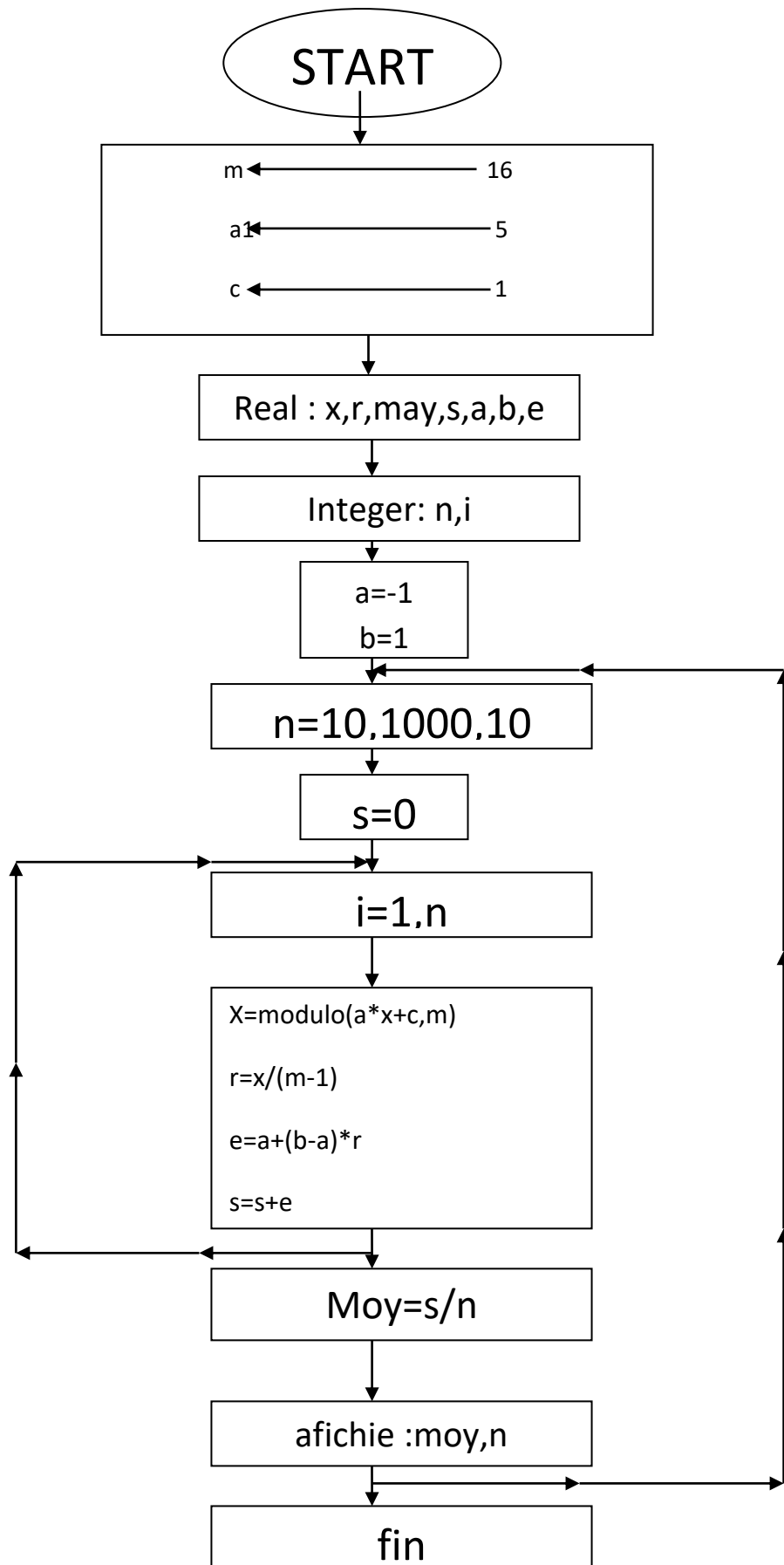
4. Traçage :

Traçage de l'évolution de la valeur moyenne en fonction de la taille de l'échantillon :



III. Générer des variables aléatoires entre $[-1,1]$:

1-Organigramme :



2- Algorithme :

Algorithme exercice2

Variable n,i : integer

X, r, moy , s,a,b ,moy1 , var : real

m ← 16 : Real

a1 ← 5 : Real

c ← 1 : Real

Debut

b ← 1

a ← -1

Pour n ← 10 a 1000 par 10 faire

S ← 0

Pour i ← 1 a n faire

x ← modulo(a1*x+c, m)

r ← x/(m-1)

e ← a+(b-a)*r

s ← s+e

fin pour

Moy ← s/n

Moy1 ← (s*s)/n

var ← (moy1-(moy*moy))/n

Ecrire (' la valeur de n : ', n)

Ecrire (' la valeur de moy : ', moy)

Ecrire (' la valeur de var: ', var)

Fin pour

Fin

3-programme :

```
program exercice2
implicit none
real,parameter :: m=16., a1=5., c=1.
real :: x,a,b,r,moy,s,e,var,moy1
integer :: n,i
open(unit=5,file='ex2.xls')
  b=1.
  a=-1

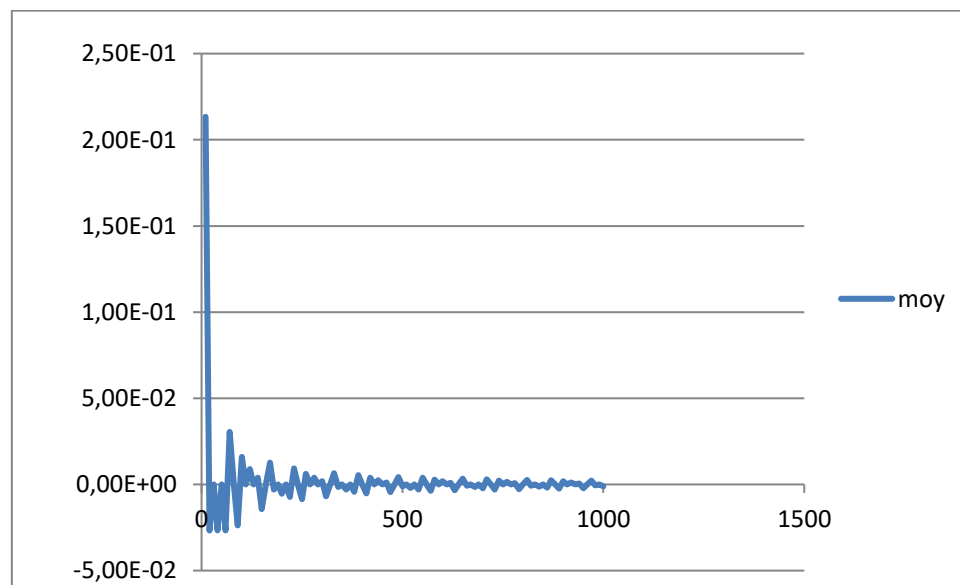
  do n=10,1000,10
    s=0.

    do i=1,n
      x=modulo(a1*x+c,m)
      r=x/(m-1)
      e=a+(b-a)*r
      s=s+e
    enddo
    moy=s/n
    moy1=(s*s)/n
    var=(moy1-(moy*moy))/n
    write(5,*)n,moy,var

  enddo
end program
```

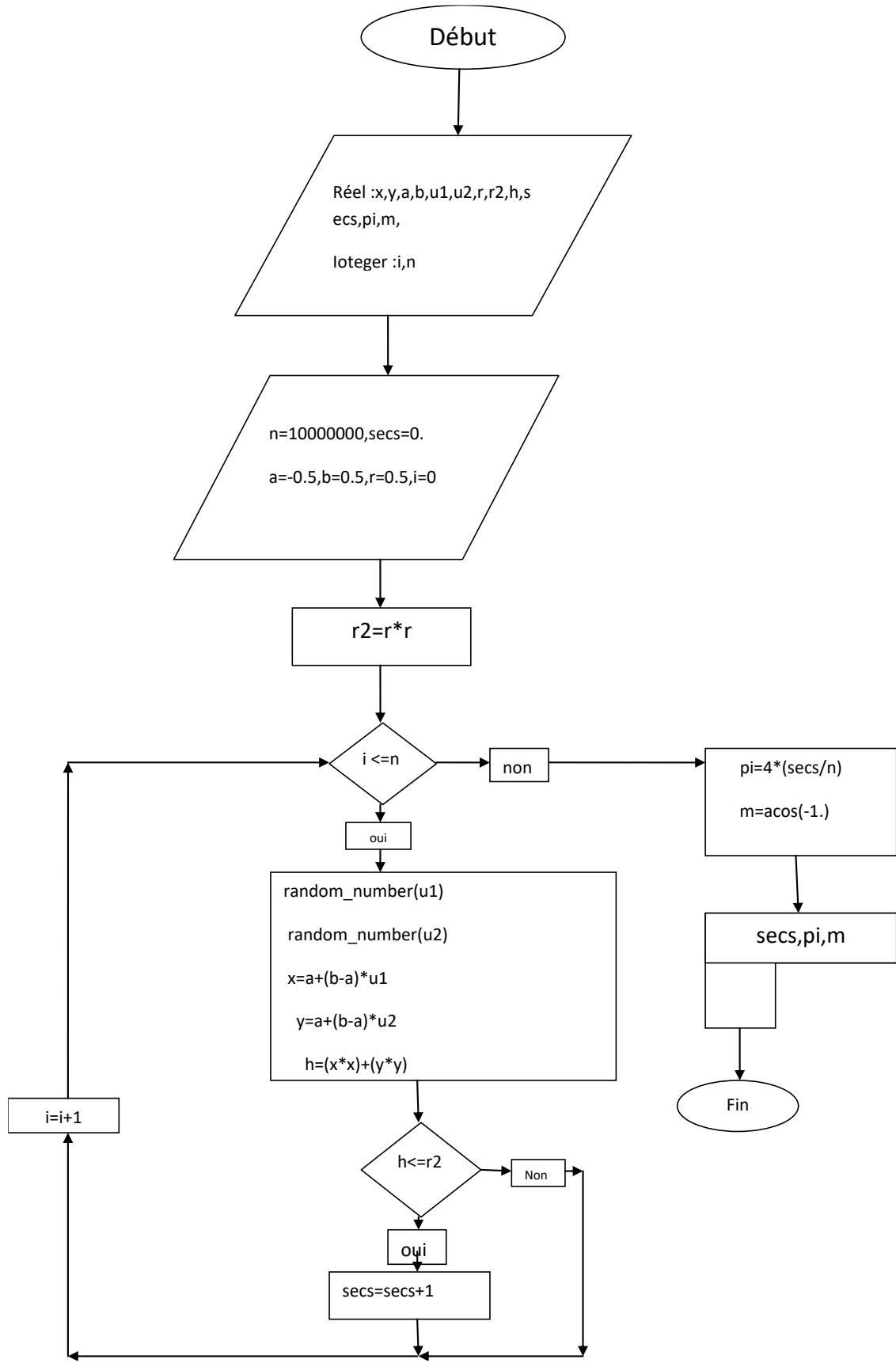
4-traçage :

Traçage de l'évolution de la valeur moyenne en fonction de la taille de l'échantillon :



IV. Calcul de π par la méthode de rejet :

1. L'organigramme :



2. Algorithme :

algorithme calcul_pi_exercice2

variables n,i : integer

m,x,y,a,b,u1,u2,r,r2,h,secs,pi

n ← 10000

secs ← 0

a ← 0,5

b ← 0,5

r ← 0,5

debut

r2 ← r*r

pour i ← 10 a n faire

call random_number(u1)

call random_number(u2)

x ← a+(b-a)*u1

y ← a+(b-a)*u2

h ← x²+y²

si h ≤ r2 alors

secs=secs+1

fin si

fin pour

pi ← 4*(secs/n)

m ← acos(-1)

ecrire(' la valeur de pi est :',pi,secs,m)

Fin

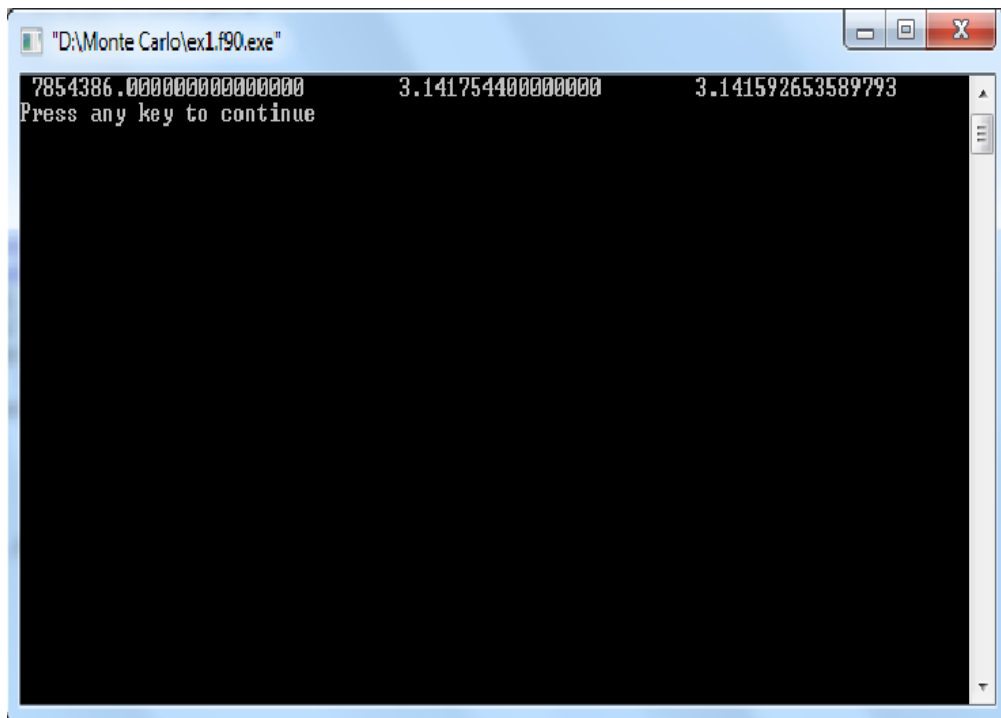
3. Programme :

```

program exericel
implicit none
real*8 :: x,y,a,b,u1,u2,r,r2,h,secs,pi,m
integer:: i,n
n=10000000
secs=0.
a=-0.5
b=0.5
r=0.5
r2=r*r
do i=1,n
    call random_number(u1)
    call random_number(u2)
    x=a+(b-a)*u1
    y=a+(b-a)*u2
    h=(x*x)+(y*y)
    if(h<=r2) then
        secs=secs+1
    endif
enddo
pi=4*(secs/n)
m=acos(-1.)
print*,secs,pi,m
end

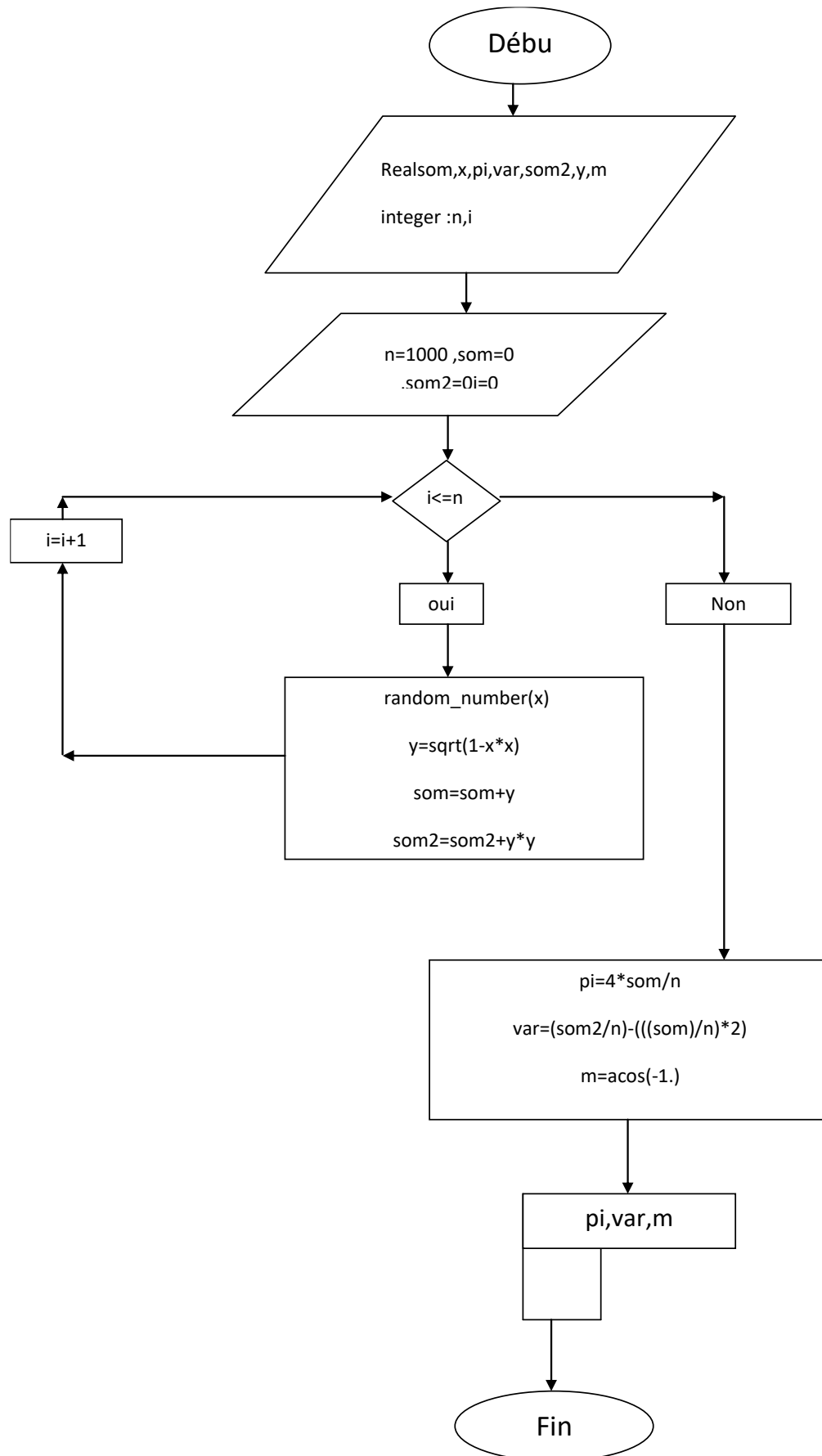
```

4. Résultat :



V. Calcul de π par la méthode d'intégration :

1. L'organigramme :



2. Algorithme :

```
algorithme calcul_pi-ex2
variables n,i : integer
som ,x,pi,var,som2,y,m
n ← 1000
som ← 0
som2 ← 0

debut
pour i ← 0 a n faire
    call random_number(x)
    y ← sqrt(1-x^2)
    som ← som+y
    som2 ← som2+y*y
fin pour

pi <----- 4*(som/n)
var <----- (som2/n)*(som/n)^2
m <----- acos(-1)

ecrire(' la valeur de pi est :',pi,var,m)

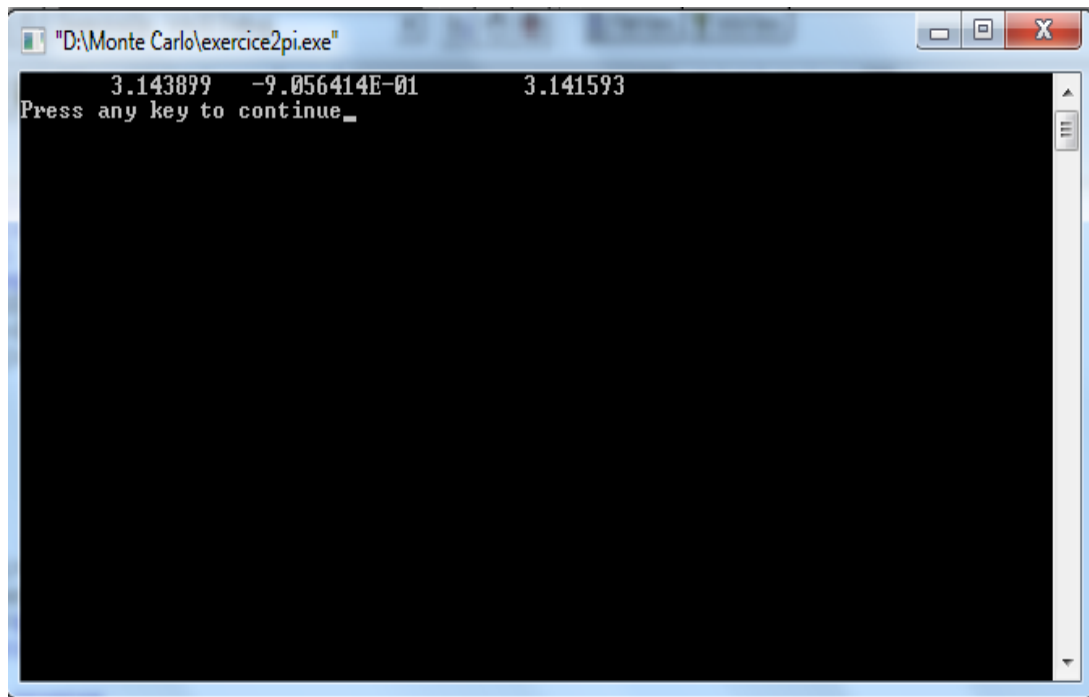
Fin
```

3. Programme:

```
program calcul_pi
implicit none
real :: som,x,pi,var,som2,y,m
integer :: n,i
n=1000
som=0.
som2=0.
do i=0,n
    call random_number(x)
    y=sqrt(1-x*x)
    som=som+y
    som2=som2+y*y
end do
pi=4*som/n
var=(som2/n)-(((som)/n)*2)
m=acos(-1.)
print*,',',pi,var,m

end program
```

4. Résultat :



```
"D:\Monte Carlo\exercice2pi.exe"
3.143899 -9.056414E-01 3.141593
Press any key to continue_
```


2. Algorithme :

Algorithme calcul-pi_ex3

Variable real : a,y,q,pi,b1,u1,u2,b,moy,x,moy1,var

Variable entie : n,i,jinb,nbr,y1,sec

b1 ← 3.14

a1 ← 2

b1 ← 4

n ← 1000

a ← 0

b ← d1/2

x ← 0

y1 ← 0

nbr ← 0

débu

pour j ← 0 à n pas 10 faire

i ← f*n

nbr ← nbr+1

nb ← 0

sec ← 0

tant que nb < i

nb ← nb+1

random (u1)

random(u2)

y ← a+(b-a)*u1

q ← a+(b-a)*u2

si y ≤ a1*sin(q)/2 alors

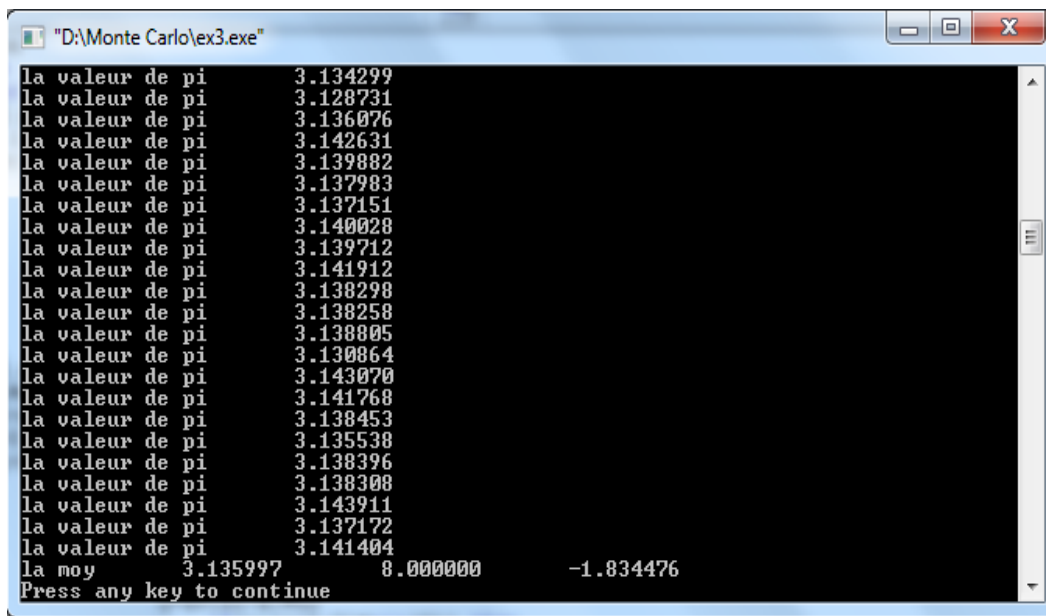
sec ← sec+1

fin si

3. Programme :

```
implicit none
real,parameter :: a1=2.,d1=4.
real :: a,y,Q,pi,b1,u1,u2,b,moy,x,moy1,var
integer :: n,i,j,nb,nbr,y1
integer :: sec
b1=3.14
n=1000
a=0.,b=(d1/2),x=0.,y1=0.,nbr=0
do j=1,n,10
  i=j*n
  nbr=nbr+1
  nb=0
  sec=0.
  do while (nb<i)
    nb=nb+1
    call random_number(u1)
    call random_number(u2)
    y=a+(b-a)*u1
    Q=a+(b1-a)*u2
    if(y<=(a1/2)*sin(Q)) then
      sec=sec+1
    endif
  enddo
  pi=((2*a1*i)/(sec*d1))
  x=x+pi
  y1=y1+(pi*pi)
  print*, 'la valeur de pi',pi
enddo
moy=(x/nbr)
moy1=(y1/nbr)
var=moy1-(moy*moy)
print*, 'la moy',moy,moy1,var
end program
```

4. Résultat :

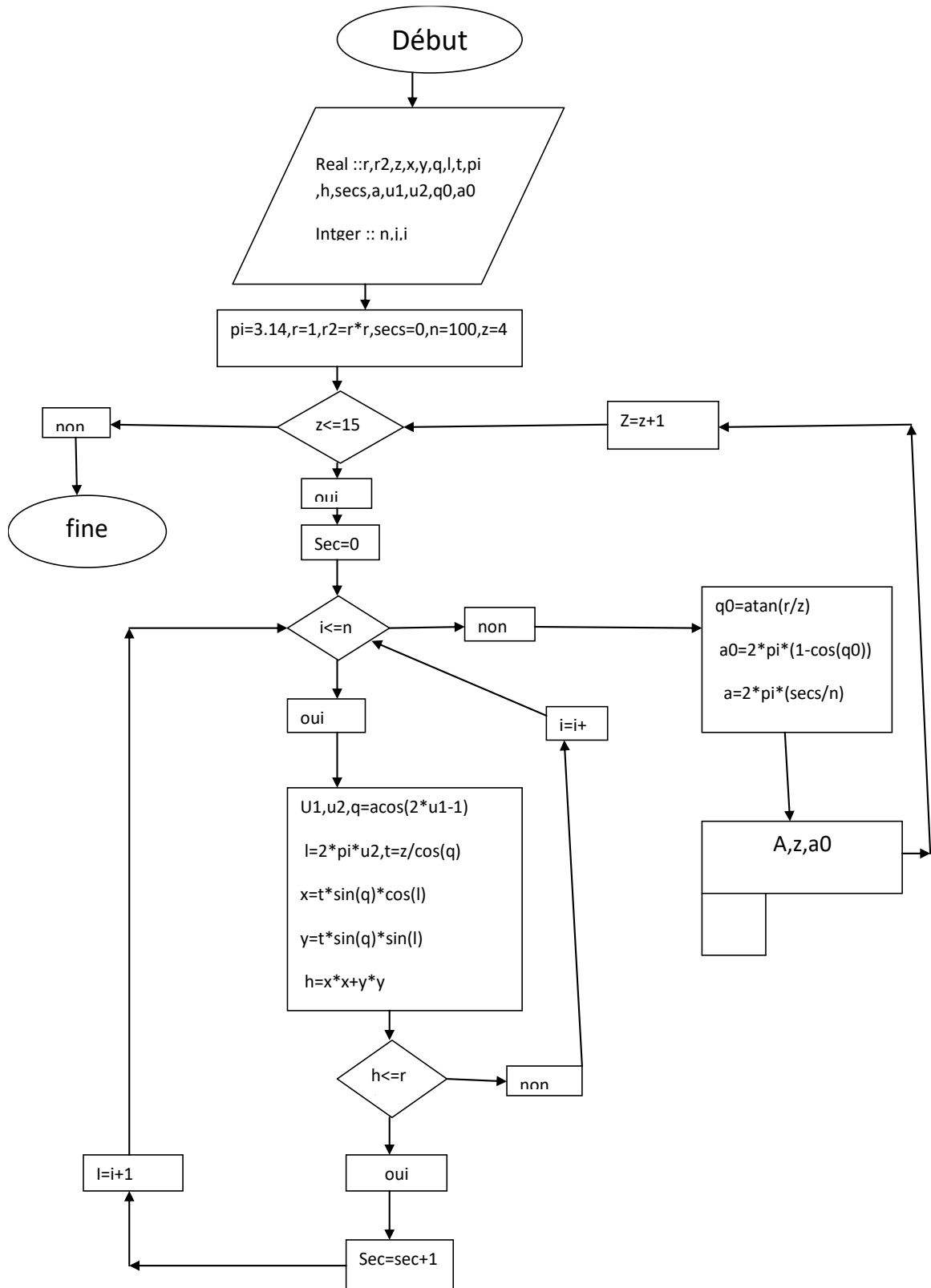


```
"D:\Monte Carlo\ex3.exe"
la valeur de pi      3.134299
la valeur de pi      3.128731
la valeur de pi      3.136076
la valeur de pi      3.142631
la valeur de pi      3.139882
la valeur de pi      3.137983
la valeur de pi      3.137151
la valeur de pi      3.140028
la valeur de pi      3.139712
la valeur de pi      3.141912
la valeur de pi      3.138298
la valeur de pi      3.138258
la valeur de pi      3.138805
la valeur de pi      3.130864
la valeur de pi      3.143070
la valeur de pi      3.141768
la valeur de pi      3.138453
la valeur de pi      3.135538
la valeur de pi      3.138396
la valeur de pi      3.138308
la valeur de pi      3.143911
la valeur de pi      3.137172
la valeur de pi      3.141404
la moy      3.135997      8.000000      -1.834476
Press any key to continue
```

VII. Calcul de l'angle solide par la méthode de rejet :

A. D'un disque :

1. L'organigramme :



2. Algorithme :

```
algorithmcalcul_angle_solide  
variables n,i : integer  
r,r2,x,y,z,q,l,t,pi,h,secs,q,u1,u2,q0,a0
```

```
n ← 1000
```

```
secs ← 0
```

```
pi ← 3.14
```

```
r ← 1
```

```
debut
```

```
    r2 ← r*r
```

```
    pour i ← 10 a n pas 100 faire
```

```
        pour z ← 1 a 20 faire
```

```
            secs ← 0
```

```
            call random_number(u1)
```

```
            call random_number(u2)
```

```
                q ←  $\arccos(2*u1-1)$ 
```

```
                l ←  $z/(\cos(q))$ 
```

```
                x ←  $t*\sin(q)*\cos(l)$ 
```

```
                y ←  $t*\sin(q)*\sin(l)$ 
```

```
                h ←  $x*x+y*y$ 
```

```
            si  $h \leq r2$  alors
```

```
                secs ← secs+1
```

```
            fin si
```

```
        q0 ←  $\arctan(r/z)$ 
```

```
        a0 ←  $2*\pi*(1-\cos(q0))$ 
```

```
        a ←  $2*\pi*(secs/n)$ 
```

```
    ecrire (' la valeur est : ',a,z,a0)
```

```
    Fin pour
```

```
    Fin pour
```

```
Fin
```

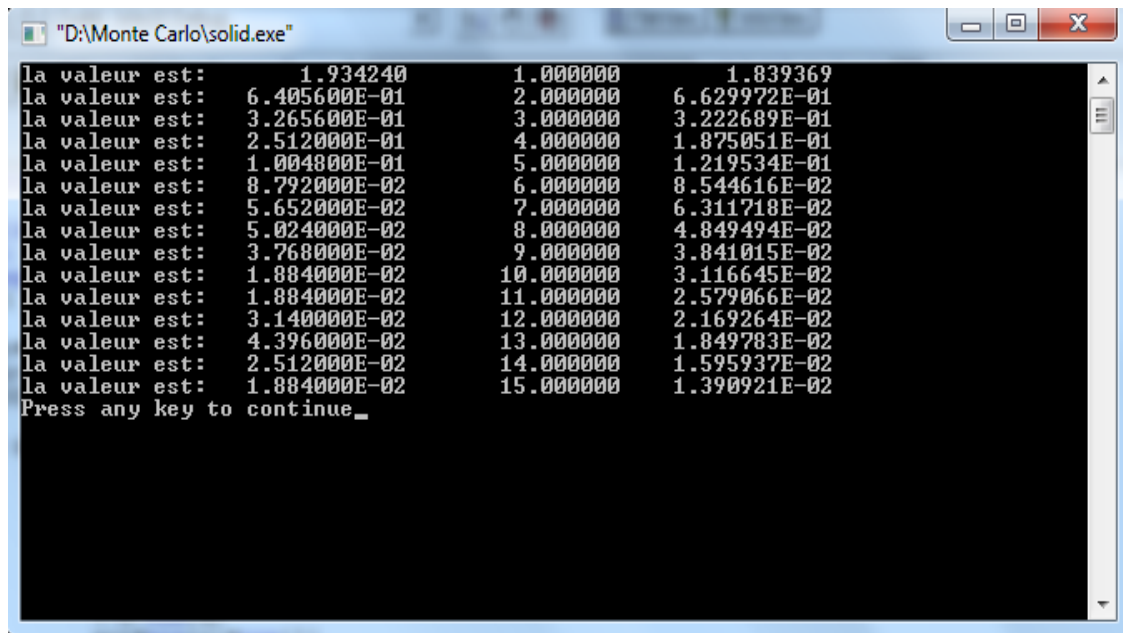
3. Programme :

```
program solid

implicit none
real::r,r2,x,y,z,q,l,t,pi,h,secs,a,u1,u2,q0,a0
integer::n,i
pi=3.14
r=1
r2=r*r
secs=0
n=1000

do z=1,15
  secs=0
  do i=1,n
    call random_number(u1)
    call random_number(u2)
    q=acos(2*u1-1)
    l=2*pi*u2
    t=z/cos(q)
    x=t*sin(q)*cos(l)
    y=t*sin(q)*sin(l)
    h=x*x+y*y
    if (h<=r2) then
      secs=secs+1
    end if
  end do
  q0=atan(r/z)
  a0=2*pi*(1-cos(q0))
  a=2*pi*(secs/n)
  print*, 'la valeur est:',a,z,a0
end do
```

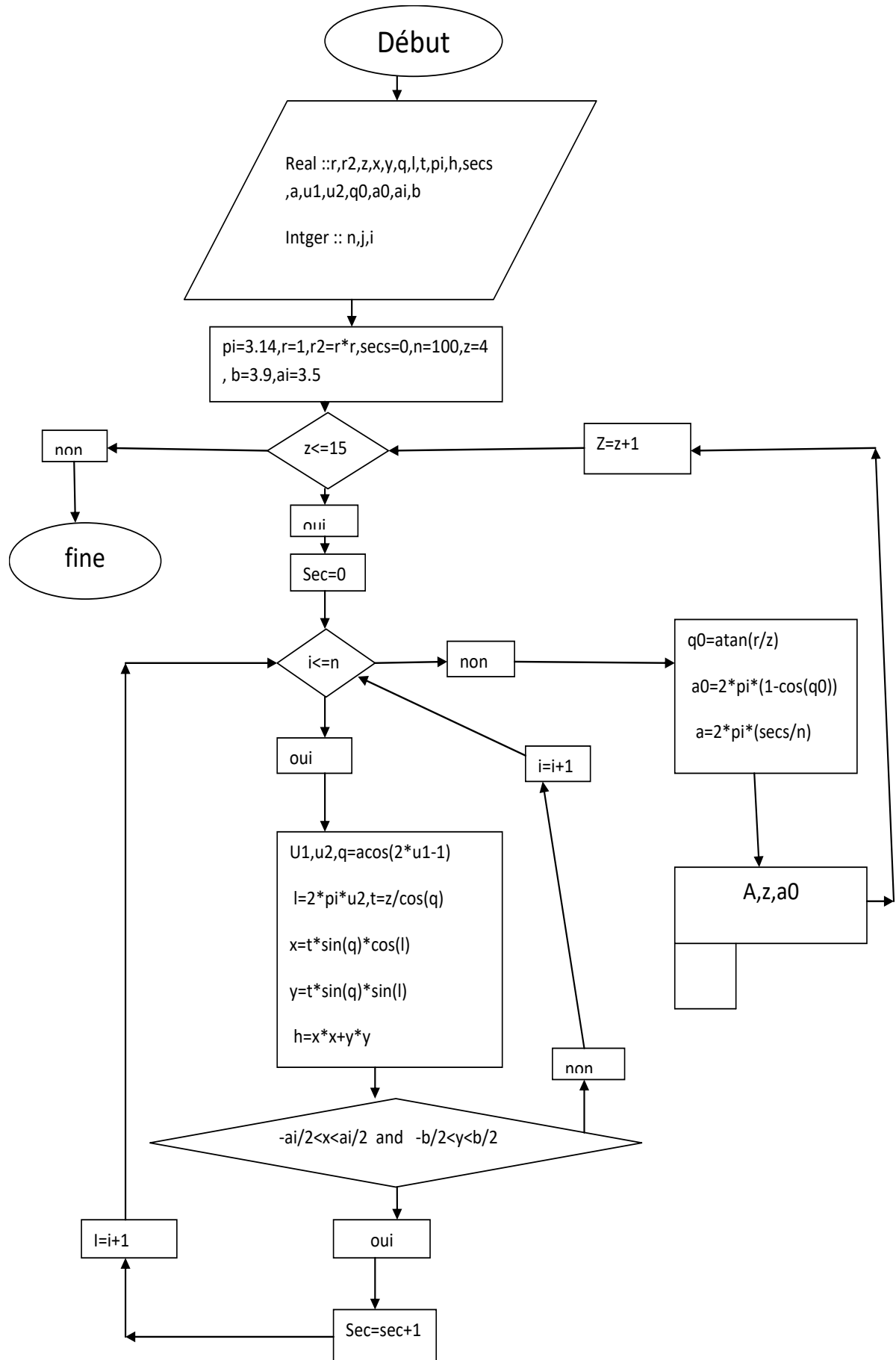
4. Résultat :



```
"D:\Monte Carlo\solid.exe"
la valeur est:      1.934240      1.000000      1.839369
la valeur est:  6.405600E-01      2.000000      6.629972E-01
la valeur est:  3.265600E-01      3.000000      3.222689E-01
la valeur est:  2.512000E-01      4.000000      1.875051E-01
la valeur est:  1.004800E-01      5.000000      1.219534E-01
la valeur est:  8.792000E-02      6.000000      8.544616E-02
la valeur est:  5.652000E-02      7.000000      6.311718E-02
la valeur est:  5.024000E-02      8.000000      4.849494E-02
la valeur est:  3.768000E-02      9.000000      3.841015E-02
la valeur est:  1.884000E-02     10.000000      3.116645E-02
la valeur est:  1.884000E-02     11.000000      2.579066E-02
la valeur est:  3.140000E-02     12.000000      2.169264E-02
la valeur est:  4.396000E-02     13.000000      1.849783E-02
la valeur est:  2.512000E-02     14.000000      1.595937E-02
la valeur est:  1.884000E-02     15.000000      1.390921E-02
Press any key to continue_
```

B. Par un rectangle :

1. L'organigramme :



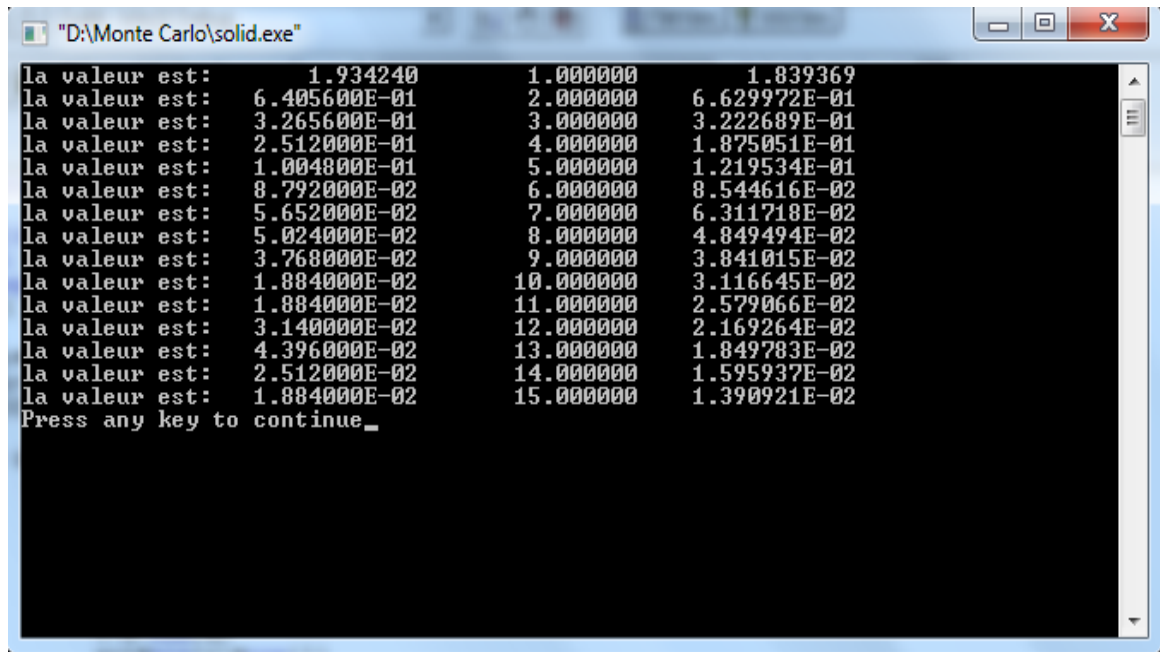
2. Algorithmme :

```
algorithmcalcul_angle_solide 2
variables n,i : integer
r,r2,x,y,z,q,l,t,pi,h,secs,q,u1,u2,q0,a0,ai,b
n ← 1000
secs ← 0
pi ← 3.14
r ← 1
ai ← 3.5
b ← 3.9
debut
    r2 ← r*r
    pour i ← 10 a n pas 100 faire
        pour z ← 1 a 20 faire
            secs ← 0
            call random_number(u1)
            call random_number(u2)
            q ← acos(2*u1-1)
            l ← z/(cos(q))
            x ← t*sin(q)*cos(l)
            y ← t*sin(q)*sin(l)
            h ← x*x+y*y
            si -ai/2<x<ai/2 et -b/2<y<b/2 alors
                secs ← secs+1
            fin si
            q0 ← atang(r/z)
            a0 ← 2*pi*(1-cos(q0))
            a ← 2*pi*(secs/n)
            ecrire (' la valeur est : ',a,z,a0)
        fin pour
    fin pour
Fin
```

3. Programme :

```
program solid_car
implicit none
real::r,r2,x,y,z,q,l,t,pi,h,secs,a,u1,u2,q0,a0,ai,b
integer::n,j,i
ai=3.5
b=3.9
pi=3.14
r=1
r2=r*r
secs=0
n=1000
do i=10,n,100
  do z=1,20
    secs=0
    do j=1,i
      call random_number(u1)
      call random_number(u2)
      q=acos(2*u1-1)
      l=2*pi*u2
      t=z/cos(q)
      x=t*sin(q)*cos(l)
      y=t*sin(q)*sin(l)
      h=x*x+y*y
      if ((-ai/2)< x.and. x>(ai/2).and.(-b/2)<y .and. y>(b/2)) then
        secs=secs+1
      end if
    end do
    q0=atan(r/z)
    a0=2*pi*(1-cos(q0))
    a=2*pi*(secs/n)
    print*, 'la valeur est:',a,z,a0
  end do
enddo
end
```

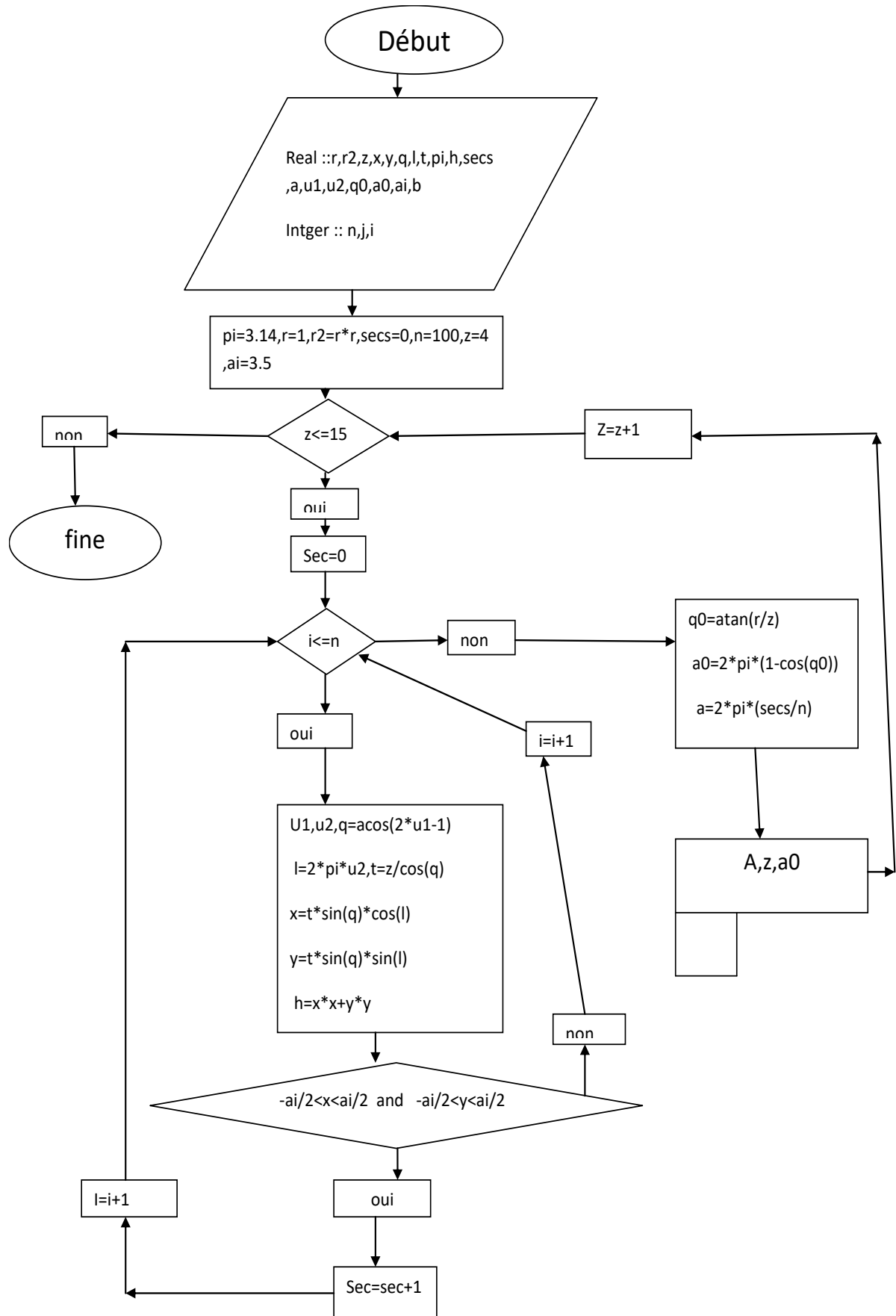
4. Résultat :



```
"D:\Monte Carlo\solid.exe"
la valeur est:      1.934240      1.000000      1.839369
la valeur est:      6.405600E-01    2.000000    6.629972E-01
la valeur est:      3.265600E-01    3.000000    3.222689E-01
la valeur est:      2.512000E-01    4.000000    1.875051E-01
la valeur est:      1.004800E-01    5.000000    1.219534E-01
la valeur est:      8.792000E-02    6.000000    8.544616E-02
la valeur est:      5.652000E-02    7.000000    6.311718E-02
la valeur est:      5.024000E-02    8.000000    4.849494E-02
la valeur est:      3.768000E-02    9.000000    3.841015E-02
la valeur est:      1.884000E-02   10.000000    3.116645E-02
la valeur est:      1.884000E-02   11.000000    2.579066E-02
la valeur est:      3.140000E-02   12.000000    2.169264E-02
la valeur est:      4.396000E-02   13.000000    1.849783E-02
la valeur est:      2.512000E-02   14.000000    1.595937E-02
la valeur est:      1.884000E-02   15.000000    1.390921E-02
Press any key to continue_
```


C. Par carrée :

1. L'organigramme :



2. Algorithme :

algorithmecalcul_angle_solide 3

variables n, i : integer

$r, r2, x, y, z, q, l, t, \pi, h, \text{secs}, q, u1, u2, q0, a0, ai, b$

$n \leftarrow 1000$

$\text{secs} \leftarrow 0$

$\pi \leftarrow 3.14$

$r \leftarrow 1$

$ai \leftarrow 3.5$

debut

$r2 \leftarrow r * r$

pour $i \leftarrow 10$ a n pas 100 faire

pour $z \leftarrow 1$ a 20 faire

$\text{secs} \leftarrow 0$

call random_number($u1$)

call random_number($u2$)

$q \leftarrow \arccos(2 * u1 - 1)$

$l \leftarrow z / (\cos(q))$

$x \leftarrow t * \sin(q) * \cos(l)$

$y \leftarrow t * \sin(q) * \sin(l)$

$h \leftarrow x * x + y * y$

si $-ai/2 < x < ai/2$ et $-ai/2 < y < ai/2$ alors

$\text{secs} \leftarrow \text{secs} + 1$

fin si

$q0 \leftarrow \arctan(r/z)$

$a0 \leftarrow 2 * \pi * (1 - \cos(q0))$

$a \leftarrow 2 * \pi * (\text{secs}/n)$

ecrire (' la valeur est : ', $a, z, a0$)

Fin pour

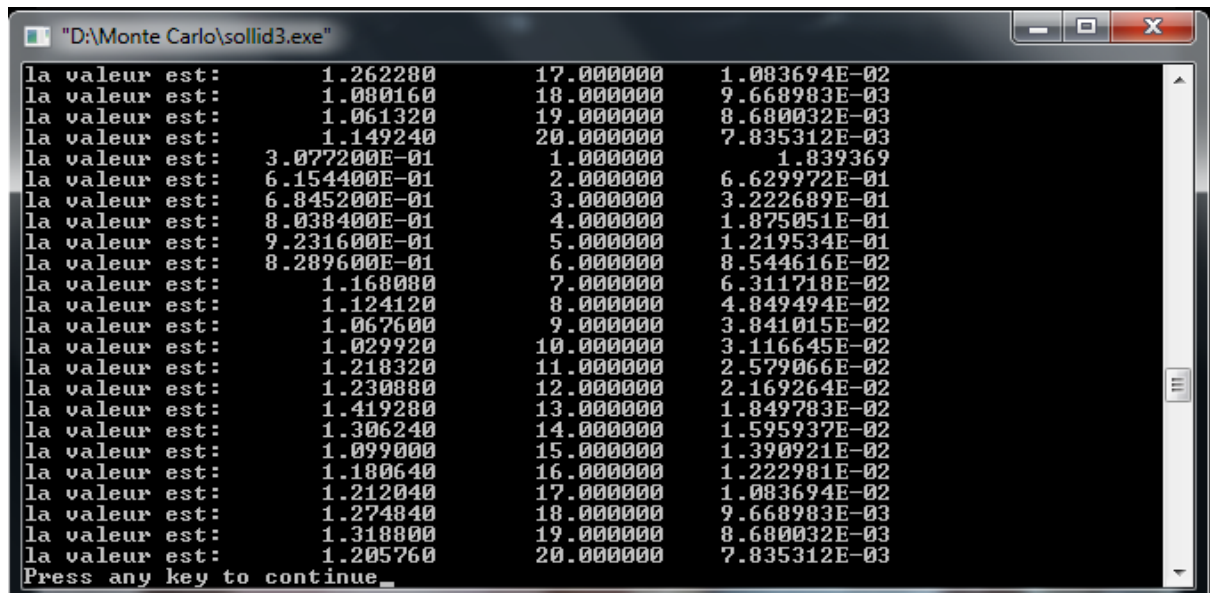
Fin pour

Fin

3. Programme :

```
program solid_car
implicit none
real::r,r2,x,y,z,q,l,t,pi,h,secs,a,u1,u2,q0,a0,ai,b
integer::n,j,i
ai=3.5
b=3.9
pi=3.14
r=1
r2=r*r
secs=0
n=1000
do i=10,n,100
  do z=1,20
    secs=0
    do j=1,i
      call random_number(u1)
      call random_number(u2)
      q=acos(2*u1-1)
      l=2*pi*u2
      t=z/cos(q)
      x=t*sin(q)*cos(l)
      y=t*sin(q)*sin(l)
      h=x*x+y*y
      if ((-ai/2)< x.and. x>(ai/2).and.(-b/2)<y .and. y>(b/2)) then
        secs=secs+1
      end if
    end do
    q0=atan(r/z)
    a0=2*pi*(1-cos(q0))
    a=2*pi*(secs/n)
    print*, 'la valeur est:',a,z,a0
  end do
enddo
end
```

4. Résultat :



```
"D:\Monte Carlo\sollid3.exe"
la valeur est:      1.262280      17.000000      1.083694E-02
la valeur est:      1.080160      18.000000      9.668983E-03
la valeur est:      1.061320      19.000000      8.680032E-03
la valeur est:      1.149240      20.000000      7.835312E-03
la valeur est:      3.077200E-01      1.000000      1.839369
la valeur est:      6.154400E-01      2.000000      6.629972E-01
la valeur est:      6.845200E-01      3.000000      3.222689E-01
la valeur est:      8.038400E-01      4.000000      1.875051E-01
la valeur est:      9.231600E-01      5.000000      1.219534E-01
la valeur est:      8.289600E-01      6.000000      8.544616E-02
la valeur est:      1.168080      7.000000      6.311718E-02
la valeur est:      1.124120      8.000000      4.849494E-02
la valeur est:      1.067600      9.000000      3.841015E-02
la valeur est:      1.029920      10.000000      3.116645E-02
la valeur est:      1.218320      11.000000      2.579066E-02
la valeur est:      1.230880      12.000000      2.169264E-02
la valeur est:      1.419280      13.000000      1.849783E-02
la valeur est:      1.306240      14.000000      1.595937E-02
la valeur est:      1.099000      15.000000      1.390921E-02
la valeur est:      1.180640      16.000000      1.222981E-02
la valeur est:      1.212040      17.000000      1.083694E-02
la valeur est:      1.274840      18.000000      9.668983E-03
la valeur est:      1.318800      19.000000      8.680032E-03
la valeur est:      1.205760      20.000000      7.835312E-03
Press any key to continue
```

VIII. Conclusion

Finalelement on conclu que ces méthodes donne des bons résultats par rapport aux résultats exacte. ce qui donne a la méthode de monte Carlo un avantage que les autres méthodes.