TP d'algorithmes Stochastiques 2

Méthodes de Monte Carlo : mouvements Browniens, marches sur les cercles ou les sphères, équation différentielle stochastique

janvier 2021

CROGUENNEC Guillaume

DUPONT Ronan

Enseignant : M. Maire

Table des matières

Introduction

Les méthodes de Monte-Carlo désignent des méthodes numériques utilisant le hasard via des nombres aléatoires. La méthode de Monte-Carlo la plus classique est le calcul d'une quantité que l'on a écrit sous forme d'une espérance et que l'on approche par sa moyenne empirique. Ces méthodes s'utilisent pour une très grande diversité de problèmes comme le calcul d'intégrales en grande dimension ou historiquement le calcul de criticité pour les réacteurs nucléaires. Elles remplacent les méthodes classiques d'analyse numérique quand ces problèmes deviennent trop complexes pour différentes raisons (grande dimension, géométrie complexe ou irrégularité de la solution).

1 Problème de la plaque

Le premier problème que l'on aborde avec les méthodes de Monte Carlo est est problème de thermique. On veut simplement connaître la température à chaque endroit d'une plaque dont le bord supérieur est maintenu à 100 degrés celsius et les autres à 0. Dans ce problème, on considère que l'énergie thermique se propage de manière uniforme.
Cela revient donc à résoudre l'équation $\triangle u = 0$ sur le carré $D = [0,1]^2$ avec comme conditions aux limites de Dirichlet 100 sur le bord supérieur et 0 ailleurs. C'est un problème que l'on a déjà traité plusieurs fois par différentes méthodes comme les différences finies, le éléments finis ou les chaînes de Markov.
On sait donc que la solution de ce problème en tout points est égale à

$$u(x,y) = \frac{400}{\pi} \sum_{k=0}^{+\infty} \frac{\sin[(2k+1)\pi x]\sinh[(2k+1)\pi y]}{(2k+1)\sinh[(2k+1)\pi]} \tag{1}$$

Mais pour obtenir une solution approchée qui va demander beaucoup moins de calculs on va utiliser la méthode de Monte Carlo du mouvement Brownien avec deux schémas d'Euler différents.

Pour faire cela, nous devons définir le mouvement Brownien à l'aide de [1] comme un mouvement aléatoire dans l'espace (ici en 2D). Contrairement au marcheur ivres qui lui va dans 4 directions (car il doit atterrir sur un point de discrétisation), le Brownien lui va dans toutes les directions, car on prend simplement ses nouvelles coordonnées d'arrivée.

On peut simuler ce mouvement en $2D$ pour les schémas numériques suivant :

$$\begin{cases} X_0 = x \\ X_{n+1} = X_n + \sqrt{\Delta t}\ Z_n^1 \end{cases} \quad \text{et} \quad \begin{cases} Y_0 = x \\ Y_{n+1} = Y_n + \sqrt{\Delta t}\ Z_n^2 \end{cases} \tag{2}$$

avec Δt le pas de temps que l'on choisiras et (Z_n^1, Z_n^2) deux gaussiennes indépendantes simulées selon la formule de Box-Muller :

$$\left(\sqrt{(-2\ln(x))}\cos(2\pi y), \sqrt{(-2\ln(x))}\sin(2\pi y)\right) \tag{3}$$

où x, $y \sim U[0,1]$: des lois uniforme sur [0,1].

En essayant de simuler le mouvement brownien figure 1sur $D = [0,1]^2$ avec pour point de départ $X_0 = 0.5$ et $Y_0 = 0.5$ on obtient :

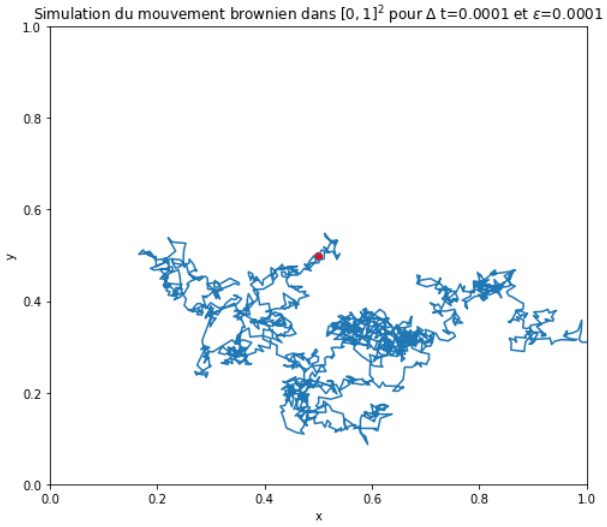

FIGURE 1 – Simulation du mouvement brownien dans $[0,1]^2$

On peut voir que le mouvement est très aléatoire. On va donc se heurter à un problème : celui de la condition de sortie de D. En effet, le mouvement n'étant plus limité à 4 directions, on va devoir imposer des conditions pour savoir de quel côté du domaine le brownien est sorti.
Pour cela, nous allons utiliser les deux schémas d'Euler figure 2 suivants :

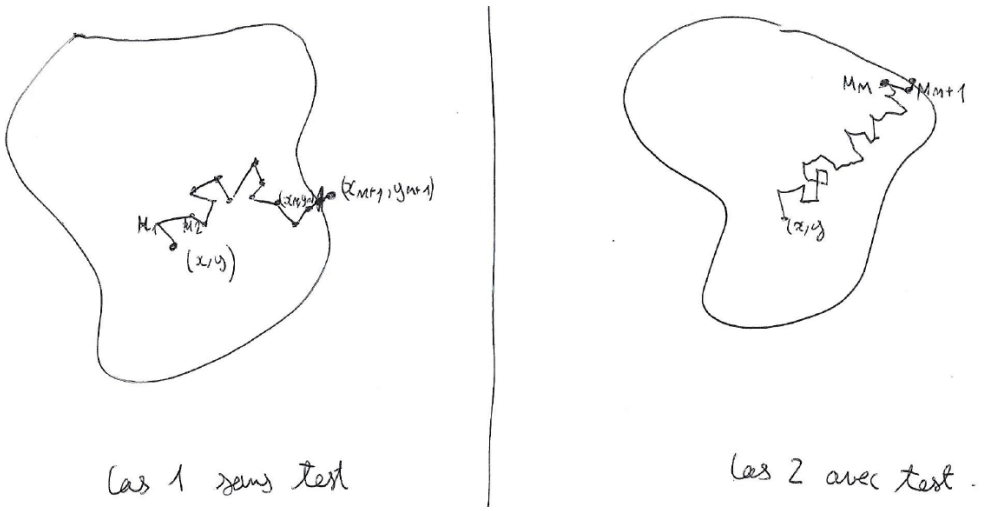

FIGURE 2 – Différentes méthodes de sortie de D

La première méthode Euler consiste à prendre l'intersection entre $[M_n,M_{n+1}]$ et la frontière du domaine D où M_n est dans le domaine D avec pour coordonnées (X_n, Y_n) et M_{n+1} est hors du domaine D. Cette méthode a une erreur globale en $O(\sqrt{\Delta t} + \frac{\sigma}{\sqrt{n}})$ avec σ l'ecart type. Une variante de cette méthode est le fait de faire une projection orthogonale du point M_{n+1} sur la frontière par rapport au centre de D.

La seconde méthode Euler consiste à faire un test de sortie où on prend aussi en compte la probabilité que le Brownien soit sorti pendant l'intervalle Δt. Si on considère M_n et M_{n+1} dans D et $d_n = d(M_n, \partial D)$ la distance de M_n à la frontière la plus proche du domaine.
Alors la condition nécessaire et suffisante pour que le marcheur soit sorti entre les deux points est :

$$e^{-\frac{d_n d_{n+1}}{2\Delta t}} > U_n$$

où $U_n \sim U[0,1]$. Cette méthode a une erreur globale en $O(\Delta t + \frac{\sigma}{\sqrt{n}})$ avec σ l'ecart type.

On programme donc ces deux méthodes sur fortran 5.1.2 puis on trace les erreurs en fonction des méthodes utilisées pour résoudre le problème figure 3 4 et 5 à des points données : (0.5,0.5), (0.05,0.05) ainsi que pour calculer la moyenne de température sur D.
Afin d'avoir une comparaison pertinente des deux simulations, on fixera pour l'un $\delta t = \frac{1}{n}$ et l'autre $\delta t = \frac{1}{\sqrt{n}}$.

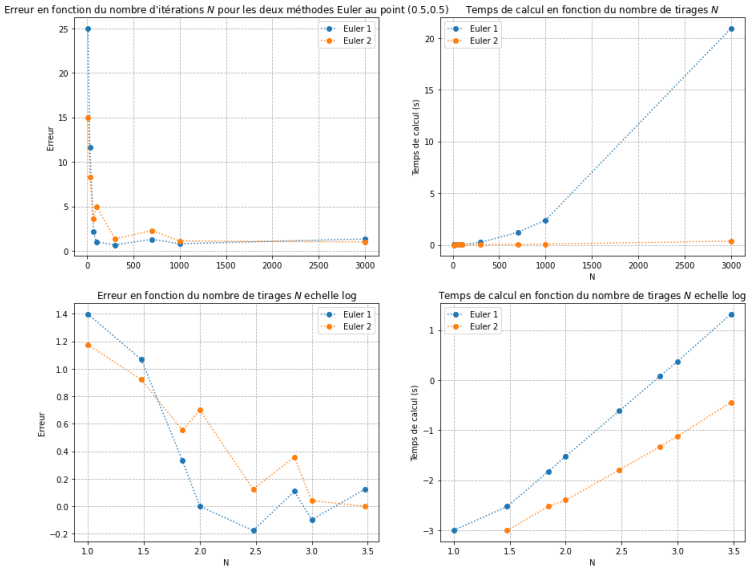

FIGURE 3 – Erreur pour les deux méthodes Euler pour le point (0.5,0.5)

Sans surprise, la température moyenne au centre de D est de 25°C. On remarque que la méthode Euler 2 est nettement plus performante que Euler 1.

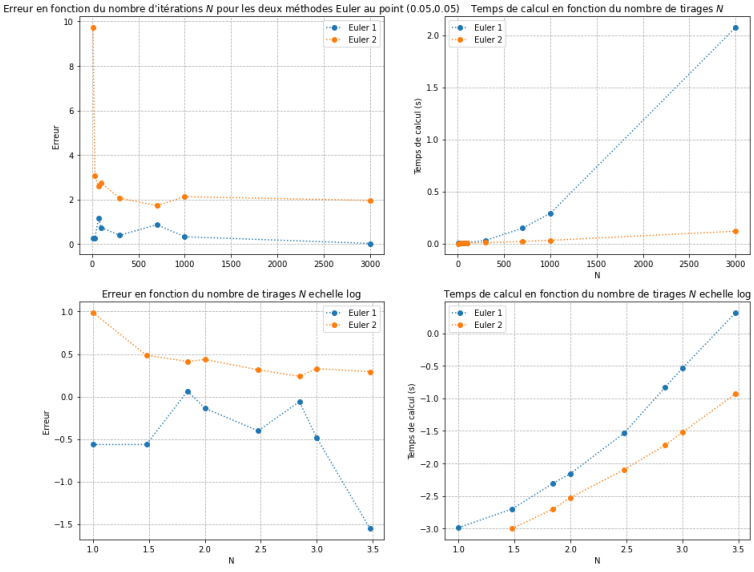

FIGURE 4 – Erreur pour les deux méthodes Euler pour le point (0.05,0.05)

Pour ce point, on calcule la température exacte avec la méthode de marche sur des cercles avec un très grand nombre d'itérations et on trouve la température de 0.2721 °C sur ce point [0.05,0.05].

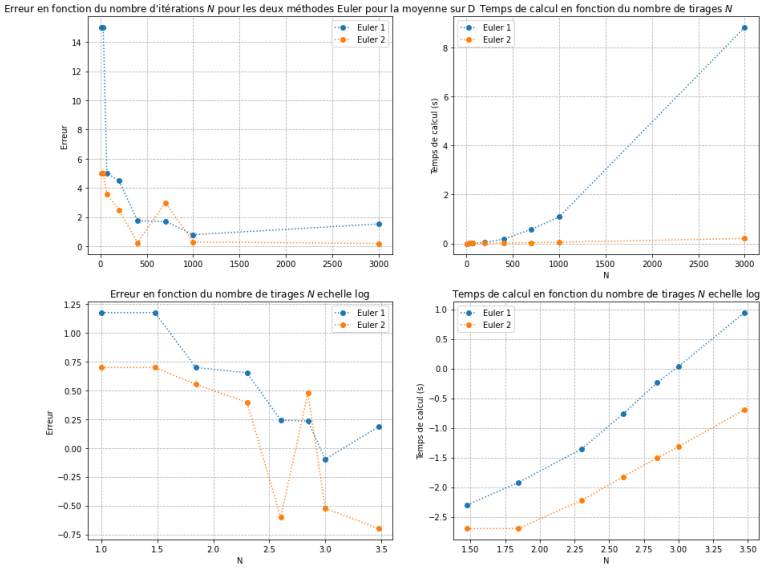

FIGURE 5 – Erreur pour les deux méthodes Euler pour la moyenne sur D

Ici, encore une fois sans surprise, on obtient une température moyenne à 25 °C.

Ensuite, on s'intéresse à une nouvelle méthode nommée **marche sur les cercles** [2]. Le principe de la méthode est illustré figure 6.

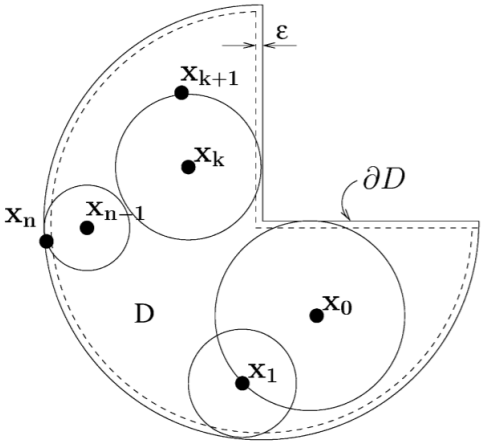

FIGURE 6 – Méthode de marche sur les cercles

On part d'un point $M_0 = (x, y)$. On calcule $r_{i+1} = d(M_i, \partial D)$ et on définit :

$$\begin{cases} x_{i+1} = x_i + r_{i+1}\cos(\theta_{i+1}) \\ y_{i+1} = y_i + r_{i+1}\sin(\theta_{i+1}) \end{cases} \tag{4}$$

où $\theta_{i+1} \sim U[0, 2\pi]$ tant que $r_i > \epsilon$.Une fois que $r_i < \epsilon$, on calcule la valeur de g au projeté de M_i sur le bord ∂D (voir figure 6). Il ne reste plus qu'à répéter ce procédé n fois puis faire la moyenne des valeurs de g au bord pour obtenir l'approximation de u. Le nombre moyen de cercles pour obtenir une précision ϵ est un $O(\ln(\epsilon))$. Cette méthode est donc bien plus efficace que les deux précédentes.
Quant à la condition de sortie, on estime que dès que le marcheur est suffisamment proche d'un

bord (comme à moins de 10^{-5}) c'est comme s'il était sorti.

Comme précédemment, on regarde les erreurs pour différentes simulations mais avec différentes conditions de bords. On peut voir les erreurs 7 et 8.

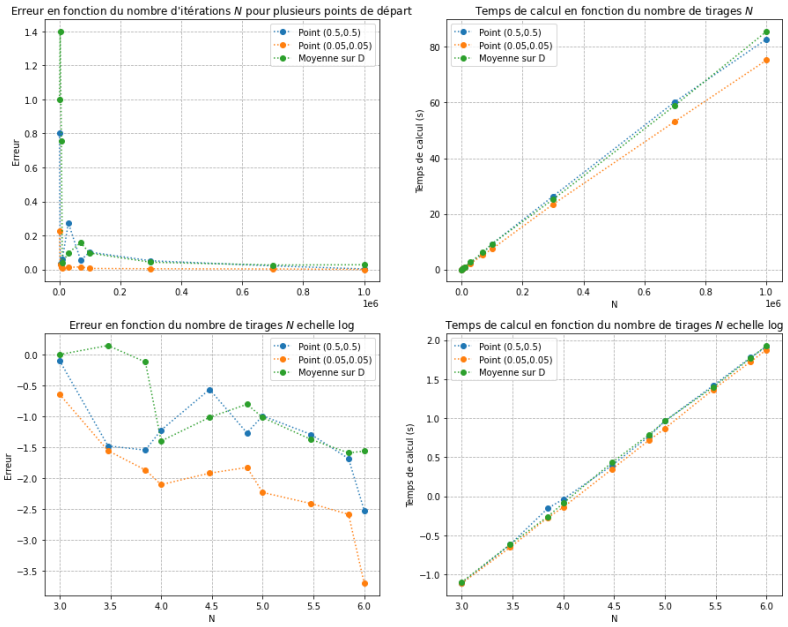

FIGURE 7 – Erreur pour la méthode marche sur les cercles

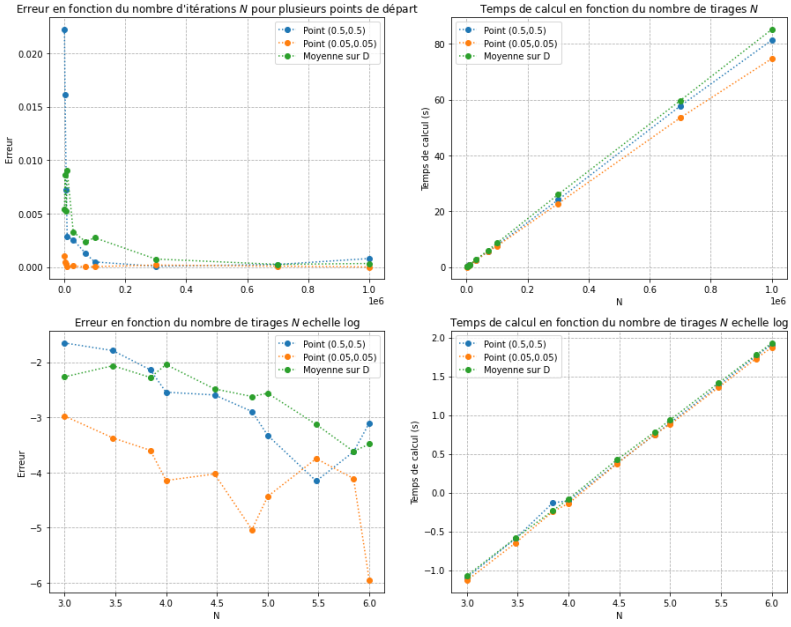

FIGURE 8 – Erreur pour la méthode marche sur les cercles avec BC : $x^2 - y^2$

Si l'on pose comme conditions aux limites de Dirichlet $g(x, y) = x^2 - y^2$, on a une moyenne de température de 0. En effet, la solution de cette équation vaut $u(x, y) = x^2 - y^2$. Or,

$$\int_{x,y\in[0,1]^2} (x^2 - y^2)dxdy = \int_{x\in[0,1]} x^2 dx - \int_{y\in[0,1]} y^2 dy = 0$$

2 Problème du studio

On s'intéresse ensuite à un nouveau problème de diffusion thermique, mais cette fois en trois dimensions spatiales. Ce problème est la diffusion thermique dans un studio qui a la configuration suivante figure 9 avec les différentes températures fixées sur les bords :

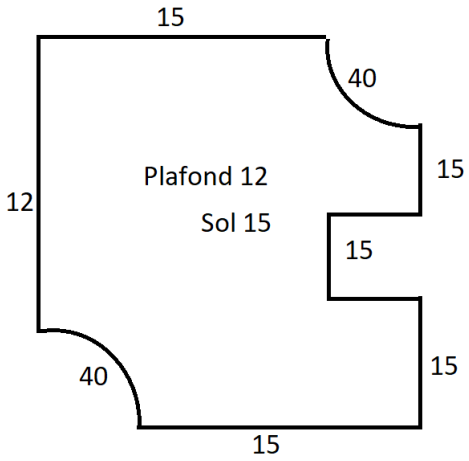

FIGURE 9 – Configuration du studio

Les quarts d'arc de cercle sont des chauffages. On pourra voir plus de détail au niveau des dimensions de la pièces (domaine $[0,5]^2$) sur la figure 10. L'idée dans ce problème est d'appliquer les méthodes précédentes du mouvement Brownien et la marche sur les cercles en 3D soit marche sur les sphères.
Pour la méthode de marche sur les sphères, on simule un point de la sphère unité par :

$$\begin{cases} X = \frac{G_1}{\sqrt{G_1+G_2+G_3}} \\ Y = \frac{G_2}{\sqrt{G_1+G_2+G_3}} \\ Z = \frac{G_3}{\sqrt{G_1+G_2+G_3}} \end{cases} \tag{5}$$

où G_1, G_2, G_3 sont des gaussiennes indépendantes simulés selon la formume de Box-Muller (3). On définit cette fois les schémas suivants :

$$\begin{cases} x = x_0 + rX \\ y = y_0 + rY \\ z = z_0 + rZ \end{cases} \tag{6}$$

en gardant exactement le même principe qu'en 2D.

Et pour ce qui est de la condition de sortie, elle reste la même que celle pour la marche sur les cercles sauf que cette fois la frontière du domaine est plus complexe, donc on a créé une matrice contenant des points de discrétisation de cette frontière.
Grâce à cette méthode, on obtient la température au milieu de la pièce en (2.5,2.5,1.25) à 14.41 °C. Si l'on augmente la température des radiateurs à 80°C, on obtient cette fois-ci une température à 15.58°C. On a pu calculer la température moyenne de la pièce à 18.35°C. Avec la méthode du mouvement Brownien on obtient des résultats moins précis, mais néanmoins cohérents. Comme 14.14 °C au centre de la pièce avec les radiateurs à 40 °C.

Ensuite, on s'est intéressé à afficher la diffusion thermique sur la pièce en 3D. C'est pourquoi nous avons calculé la température de la pièce sur 40 000 points en utilisant un faible nombre de simulations. On a pu obtenir les figures 10, 11 et 12.

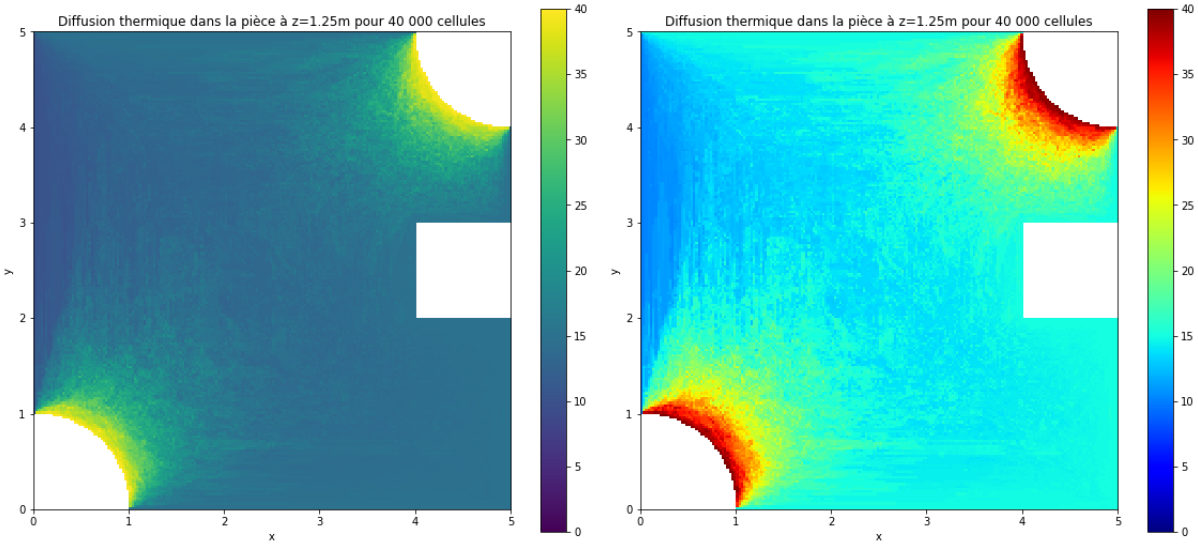

FIGURE 10 – Diffusion à $z = 1.25$m

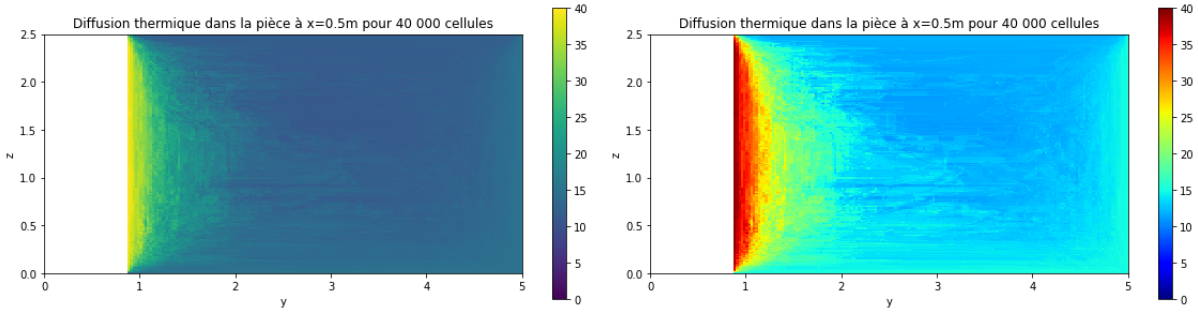

FIGURE 11 – Diffusion à $x = 0.5$m

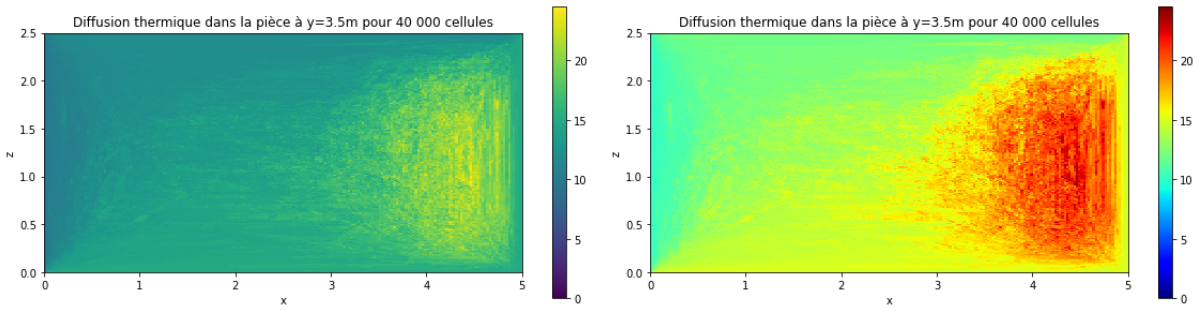

FIGURE 12 – Diffusion à $y = 3.5$m

On remarque bien la diffusion thermique partant des radiateurs vers le mur proche de l'extérieur qui est plus froid que les autres figure 10. Sur la figure 11 on peut voir la différence de température entre le sol et le plafond.

3 Problème du naufragé

Le dernier problème que l'on choisi est de simuler la probabilité qu'un naufragé survive. Pour cela, on modélise son mouvement par une équation différentielle d'advection/diffusion. C'est une équation stochastique que l'on discrétise par un schéma d'Euler. Ceci conduit à la dynamique suivante

$$X_{n+1} = X_n + \alpha\Delta t + \sigma\sqrt{\Delta t}Z_n, Y_{n+1} = Y_n + \beta\Delta t + \sigma\sqrt{\Delta t}W_n$$

où Z_n et W_n sont des Gaussiennes indépendantes. Les paramètres α et β modélisent respectivement l'advection due au vent et au courant et σ les aléas du mouvement.

On se place dans la configuration suivante figure 13 :

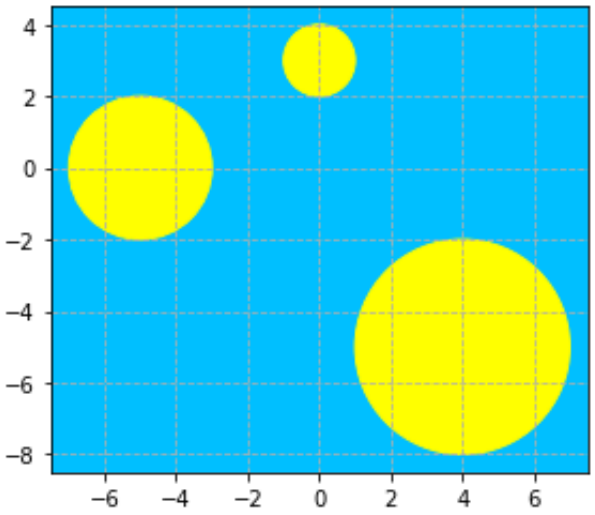

FIGURE 13 – Configuration des iles

où le naufragé commence son périple en (0,0).

Par exemple, on peut observer figure 14 un cas où le naufragé arrive à rallier une île :

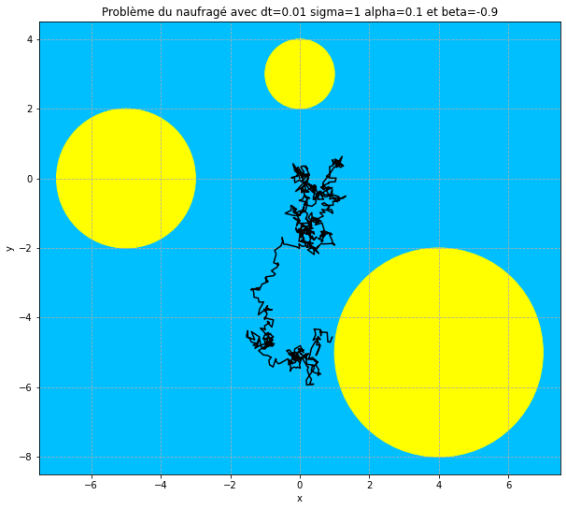

FIGURE 14 – La naufragé arrive à rejoindre une des îles

On peut de même montrer une configuration où le naufragé n'arrive pas à rallier les îles et meurt comme on peut voir figure 15.

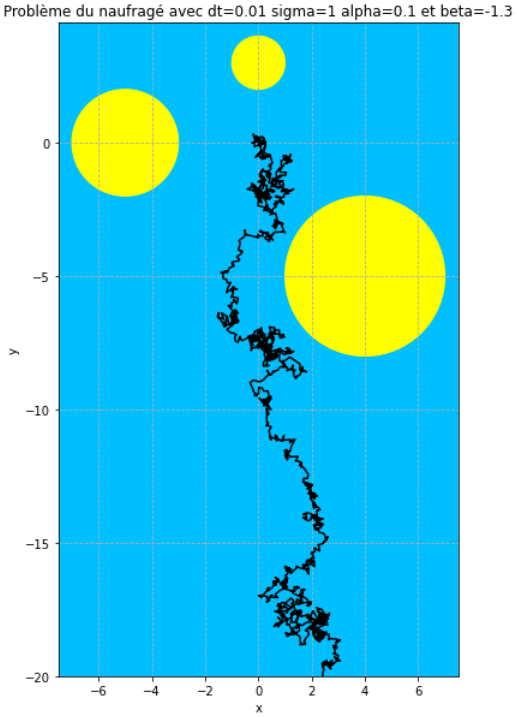

FIGURE 15 – Configuration des iles

En effet, il n'a aucune chance de rejoindre une île à partir de ce stade.
Pour aller plus loin, on étudiera la probabilité de survie du naufragé en fonction des différents paramètres. On considère qu'il ne survit pas s'il n'atteint pas une île avant une date $T = 3$.
Pour élaborer nos probabilités, nous regarderons l'angle vers lequel le naufragé est déporté. Celui ci est égal à $\theta = \arctan(\frac{\beta}{\alpha})$.
Par exemple, si l'on prend un angle $\theta = \frac{\pi}{3}$, avec un grand nombre de simulations, on obtient la figure 16 :

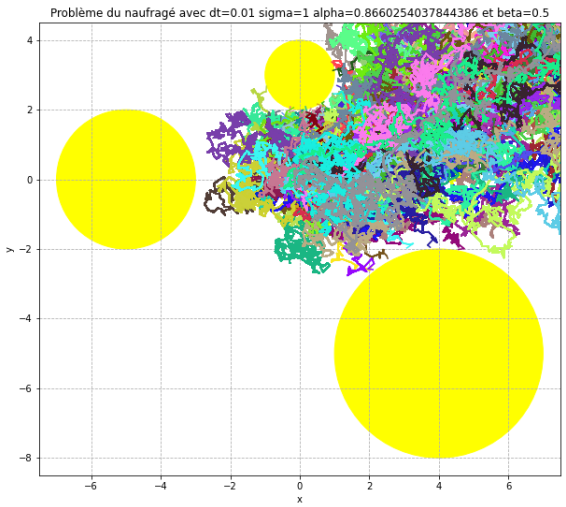

FIGURE 16 – Simulation de N naufragés pour un angle $\theta = \frac{\pi}{3}$

Ici, la probabilité de survie est critique, elle vaut p=0.265 avec les coefficients ci-dessus.

On peut faire le bilan dans le tableau ci-dessous pour $\Delta t = 0.01$ et $\sigma = 1$ avec $N = 1000$ simulations :

θ_{rad}	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	$\frac{2\pi}{3}$	$\frac{3\pi}{4}$	$\frac{5\pi}{6}$
θ_{deg}	0	30	45	60	90	120	135	150
$P(\theta)$	0.232	0.286	0.369	0.479	0.627	0.512	0.555	0.584

θ_{rad}	π	$\frac{7\pi}{6}$	$\frac{5\pi}{4}$	$\frac{4\pi}{3}$	$\frac{3\pi}{2}$	$\frac{5\pi}{3}$	$\frac{7\pi}{4}$	$\frac{11\pi}{6}$
θ_{deg}	180	210	225	240	270	300	315	330
$P(\theta)$	0.568	0.38	0.267	0.214	0.258	0.389	0.439	0.453

Si l'on prend $\sigma = 0.5$, on obtient les probabilités suivantes :

θ_{rad}	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	$\frac{2\pi}{3}$	$\frac{3\pi}{4}$	$\frac{5\pi}{6}$
θ_{deg}	0	30	45	60	90	120	135	150
$P(\theta)$	0.011	0.047	0.142	0.414	0.807	0.163	0.225	0.2

θ_{rad}	π	$\frac{7\pi}{6}$	$\frac{5\pi}{4}$	$\frac{4\pi}{3}$	$\frac{3\pi}{2}$	$\frac{5\pi}{3}$	$\frac{7\pi}{4}$	$\frac{11\pi}{6}$
θ_{deg}	180	210	225	240	270	300	315	330
$P(\theta)$	0.51	0.212	0.06	0.005	0.069	0.217	0.328	0.334

Si l'on prend $\sigma = 1.5$, on obtient les probabilités suivantes :

θ_{rad}	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	$\frac{2\pi}{3}$	$\frac{3\pi}{4}$	$\frac{5\pi}{6}$
θ_{deg}	0	30	45	60	90	120	135	150
$P(\theta)$	0.232	0.286	0.369	0.479	0.627	0.512	0.555	0.584

θ_{rad}	π	$\frac{7\pi}{6}$	$\frac{5\pi}{4}$	$\frac{4\pi}{3}$	$\frac{3\pi}{2}$	$\frac{5\pi}{3}$	$\frac{7\pi}{4}$	$\frac{11\pi}{6}$
θ_{deg}	180	210	225	240	270	300	315	330
$P(\theta)$	0.501	0.468	0.546	0.572	0.697	0.658	0.713	0.754

On remarque que le naufragé a la plus faible probabilité de survivre pour un angle θ à 240 degrés. De plus, on peut voir l'influence de σ qui modélise la dissipation du mouvement brownien. Par exemple avec $\sigma = 0.5$, le mouvement brownien suit l'angle sans trop dévier alors qu'avec $\sigma = 1.5$ il dévie très fortement et donc ralie avec une plus forte probabilité les autres îles.
Il n'y a aucune île pour un angle θ à 240 degrés. Avec ces tableaux, on peut donc trouver la probabilité d'un naufragé de s'être échoué sur une île avec les coefficients de vent et de courant. Si celui-ci ne s'est pas échoué sur une île, on pourra trouver sa position approximative en prenant l'angle θ et en calculant sa position approximative en fonction du temps.

4 Conclusion

Ainsi, au travers des trois problèmes abordés on a pu voir que les méthodes de Monte Carlo demandent un coup de calcul bien moindre que les méthodes d'éléments finis par exemple, et que qu'elle restent néanmoins extrêmement précises. Bien sûr, la précision n'est pas la même selon la méthode de Monte Carlo, tout comme l'est le coût en calcul ou le nombre de lignes nécessaires au programme, mais dans tous les cas cela reste un coût calculatoire très réduit.
On en conclue donc que, même si la méthode des éléments finis est extrêmement précis, pour des problèmes avec des géométries complexes les méthodes de Monte Carlo sont largement préférables.

Références

[1] Maire Sylvain. *Cours sur les méthodes Monte Carlo*, 2020.

[2] Chi-Ok Hwang, Michael Mascagni, James A. Given. *A Feynman–Kac path-integral implementation for Poisson's equation using an h-conditioned Green's function*, 2003. http://websrv.cs.fsu.edu/~mascagni/papers/RIJP2003_1.pdf

5 Annexe

5.1 Problème de la plaque

5.1.1 Euler 1

```
program 1.1
        implicit none

integer                     :: N,i, j, k
real ( kind = 8)            :: ix,na,nmax,pi,c,dt,u1,u2,Z1,Z2,m
real ( kind = 8)            :: Xold,X,Yold,Y
real(kind=kind(0.d0)), dimension(:),allocatable :: tM
allocate(tM(3))

ix=51477.d0
na=16807.d0
nmax=2147483647.d0 ! nombre premier tres grand
pi=3.14159

N=100000
c=0.d0
Xold=0.5
Yold=0.5
dt=0.00001

!================ Corps du programme =================

DO i=1,N
  X=0.5
  Y=0.5
  DO WHILE((0<X).AND.(X<1).AND.(0<Y).AND.(Y<1)) !Dans D ?
    Xold=X
    Yold=Y
    CALL rd(U1)
        CALL rd(U2)
        CALL gauss(U1,U2,Z1,Z2)
        X=Xold+dsqrt(dt)*Z1
        Y=Yold+dsqrt(dt)*Z2
  ENDDO

  IF (Y>=1) THEN
    tM(1)=Xold**2
        tM(2)=(1-Xold)**2
        tM(3)=(1-Yold)**2
        m=minval(tM)
        IF (m==tM(3)) THEN !On projette sur le côté du haut
          c=c+100
        END IF
  END IF
ENDDO
```

```
print*,c/N

!============================
CONTAINS
SUBROUTINE rd(u)
real ( kind = 8)            :: u
ix=abs(ix*na)
ix=mod(ix,nmax)
u=dble(ix)/dble(nmax)
  RETURN
END SUBROUTINE rd

SUBROUTINE gauss(U1,U2,Z1,Z2)
real ( kind = 8)            :: U1,U2,Z1,Z2
  Z1=dsqrt(-2*dlog(U1))*dcos(pi*2*U2)
  Z2=dsqrt(-2*dlog(U1))*dsin(pi*2*U2)
  RETURN
END SUBROUTINE gauss
! ========== FIN DU PROGRAMME =============
 end program
```

5.1.2 Euler 2

```
!=============== Corps du programme =================

DO i=1,N
  X=0.5
  Y=0.5
  CALL rd(U)
  test=0
  DO WHILE(test<U) !Test exp
    Xold=X
    Yold=Y
        CALL gauss(Z1,Z2)
        X=Xold+dsqrt(dt)*Z1
        Y=Yold+dsqrt(dt)*Z2
        tM(1)=dexp(-(1-Y)*(1-Yold)/2/dt)  !test haut
        tM(2)=dexp(-(Y)*(Yold)/2/dt)        !test bas
        tM(3)=dexp(-(1-X)*(1-Xold)/2/dt) !test droite
        tM(4)=dexp(-(X)*(Xold)/2/dt)       !test gauche
        test=maxval(tM)
  ENDDO

  !Projeté
  CALL proj(test,tM,X,Y)

  ! Boundary condition
  IF (Y==1) THEN
    c=c+100
  ENDIF
ENDDO
```

```
print*,c/N

!============================
CONTAINS

SUBROUTINE proj(test,tM,X,Y)
real ( kind = 8)             :: test,X,Y
real(kind=kind(0.d0)), dimension(:),allocatable :: tM
  IF (test==tM(1)) THEN !haut
    Y=1.d0
  END IF
   IF (test==tM(2)) THEN !bas
    Y=0.d0
  END IF
   IF (test==tM(3)) THEN !droite
    X=1.d0
  END IF
   IF (test==tM(4)) THEN !gauche
    X=0.d0
  END IF
  RETURN
END SUBROUTINE proj
! ============ FIN DU PROGRAMME =============
 end program
```

5.1.3 Euler 2 avec BC

```
!================ Corps du programme =================

DO i=1,N
  X=0.5
  Y=0.5
  ex=X**2-Y**2
  CALL rd(U)
  test=0
  DO WHILE(test<U) !Test exp
    Xold=X
    Yold=Y
        CALL gauss(Z1,Z2)
        X=Xold+dsqrt(dt)*Z1
        Y=Yold+dsqrt(dt)*Z2
        tM(1)=dexp(-(1-Y)*(1-Yold)/2/dt)  !test haut
        tM(2)=dexp(-(Y)*(Yold)/2/dt)        !test bas
        tM(3)=dexp(-(1-X)*(1-Xold)/2/dt) !test droite
        tM(4)=dexp(-(X)*(Xold)/2/dt)       !test gauche
        test=maxval(tM)
  ENDDO
  !Projeté
  CALL proj(test,tM,X,Y)
  ! Boundary condition
  c=c+X**2-Y**2
```

```
ENDDO

print*,c/N,ex
! ========== FIN DU PROGRAMME =============
 end program
```

5.1.4 Marche sur les cercles

```
!================ Corps du programme =================
DO i=1,N
  X=0.5
  Y=0.5
  R=0.5
  DO WHILE(R>eps) !Test
        CALL rd(theta)
        theta=theta*2*pi
        tM=(/1-Y,Y,1-X,X/)
        R=minval(tM)
        X=X+R*dcos(theta)
        Y=Y+R*dsin(theta)
  ENDDO

  !Projeté
  CALL proj(R,tM,X,Y)

  ! Boundary condition
  IF (Y==1) THEN
    c=c+100
  ENDIF
ENDDO

print*,c/N
```

5.2 Problème du studio

5.2.1 Studio avec Marche sur les sphères

```
do i=1,400
        Un(i,1)=1.d0+i/100.d0
        Un(i,2)=0.d0
        Un(400+i,1)=0.d0
        Un(400+i,2)=1.d0+i/100.d0
        Un(800+i,1)=i/100.d0
        Un(800+i,2)=5.d0
enddo
do i=1,200
        Un(1200+i,1)=5.d0
        Un(1200+i,2)=i/100.d0
enddo
do i=1,100
        Un(1400+i,1)=4.d0+i/100.d0
        Un(1400+i,2)=2.d0
        Un(1500+i,1)=4.d0
```

```
        Un(1500+i,2)=2.d0+i/100.d0
        Un(1600+i,1)=4.d0+i/100.d0
        Un(1600+i,2)=3.d0
        Un(1700+i,1)=5.d0
        Un(1700+i,2)=3.d0+i/100.d0
enddo
do i=1,157
        Un(1800+i,1)=dcos(1.57-i/100.d0)
        Un(1800+i,2)=dcos(i/100.d0)
        Un(1957+i,1)=5.d0-dcos(i/100.d0)
        Un(1957+i,2)=4.d0+dcos(i/100.d0)
enddo

ix=51477.d0
na=16807.d0
nmax=2147483647.d0

N=10000
dt=1.d0/100.d0
v=0.d0
r=0.d0


DO i=1,N
        x=2.5
        y=2.5
        z=1.25
        d=5
        do while (d>0.1)

                do j=1,2114
                        if (((x-Un(j,1))**2+(y-Un(j,2))**2)**(1/2.d0)<d) then
                                d=((x-Un(j,1))**2+(y-Un(j,2))**2)**(1/2.d0)
                                u=j
                        endif
                enddo
                if (abs(z-2.5)<d) then
                        d=abs(z-2.5)
                        u=2115
                endif
                if (abs(z)<d) then
                        d=abs(z)
                        u=2116
                endif

                if (d>0.1) then

                        ix=abs(ix*na)
                        ix=mod(ix,nmax)
                        c=(dble(ix)/dble(nmax))
```

```
                        ix=abs(ix*na)
                        ix=mod(ix,nmax)
                        t=(dble(ix)/dble(nmax))

                        ix=abs(ix*na)
                        ix=mod(ix,nmax)
                        q=(dble(ix)/dble(nmax))

                        ix=abs(ix*na)
                        ix=mod(ix,nmax)
                        s=(dble(ix)/dble(nmax))

                        G1=(-2*dlog(t))**(1/2.d0)*dcos(2*3.1415*c)
                        G2=(-2*dlog(t))**(1/2.d0)*dsin(2*3.1415*c)
                        G3=(-2*dlog(q))**(1/2.d0)*dcos(2*3.1415*s)

                        x=x+d*G1/((G1**2+G2**2+G3**2)**(1/2.d0))
                        y=y+d*G2/((G1**2+G2**2+G3**2)**(1/2.d0))
                        z=z+d*G3/((G1**2+G2**2+G3**2)**(1/2.d0))
                endif
        enddo
        if (u==2115) then
                r=r+12
        elseif (u<=400) then
                r=r+10
        elseif (u>1800 .and. u<=2114) then
                r=r+40
        else
                r=r+15
        endif
ENDDO

print*,r/N
```

5.2.2 Studio avec mouvements Browniens

```
do i=1,400
        Un(i,1)=1.d0+i/100.d0
        Un(i,2)=0.d0
        Un(400+i,1)=0.d0
        Un(400+i,2)=1.d0+i/100.d0
        Un(800+i,1)=i/100.d0
        Un(800+i,2)=5.d0
enddo
do i=1,200
        Un(1200+i,1)=5.d0
        Un(1200+i,2)=i/100.d0
enddo
do i=1,100
        Un(1400+i,1)=4.d0+i/100.d0
        Un(1400+i,2)=2.d0
        Un(1500+i,1)=4.d0
```

```
        Un(1500+i,2)=2.d0+i/100.d0
        Un(1600+i,1)=4.d0+i/100.d0
        Un(1600+i,2)=3.d0
        Un(1700+i,1)=5.d0
        Un(1700+i,2)=3.d0+i/100.d0
enddo
do i=1,157
        Un(1800+i,1)=dcos(1.57-i/100.d0)
        Un(1800+i,2)=dcos(i/100.d0)
        Un(1957+i,1)=5.d0-dcos(i/100.d0)
        Un(1957+i,2)=4.d0+dcos(i/100.d0)
enddo

ix=51477.d0
na=16807.d0
nmax=2147483647.d0

N=10000
dt=1.d0/100.d0
v=0.d0
r=0.d0


DO i=1,N
        x=2.5
        y=2.5
        z=1.25
        xold=2.5
        yold=2.5
        zold=1.25
        d=5
        dold=5
        do while (((x-xold)**2+(y-yold)**2+(z-zold)**2)**(1/2.d0)<d .and.&
        &((x-xold)**2+(y-yold)**2+(z-zold)**2)**(1/2.d0)<dold)

                dold=d
                do j=1,2114
                        if (((x-Un(j,1))**2+(y-Un(j,2))**2)**(1/2.d0)<d) then
                                d=((x-Un(j,1))**2+(y-Un(j,2))**2)**(1/2.d0)
                                u=j
                        endif
                enddo
                if (abs(z-2.5)<d) then
                        d=abs(z-2.5)
                        u=2115
                endif
                if (abs(z)<d) then
                        d=abs(z)
                        u=2116
                endif
```

```
                xold=x
                yold=y
                zold=z

                ix=abs(ix*na)
                ix=mod(ix,nmax)
                c=(dble(ix)/dble(nmax))

                ix=abs(ix*na)
                ix=mod(ix,nmax)
                t=(dble(ix)/dble(nmax))

                ix=abs(ix*na)
                ix=mod(ix,nmax)
                q=(dble(ix)/dble(nmax))

                ix=abs(ix*na)
                ix=mod(ix,nmax)
                s=(dble(ix)/dble(nmax))

                x=x+dt**(1/2.d0)*(-2*dlog(t))**(1/2.d0)*dcos(2*3.1415*c)
                y=y+dt**(1/2.d0)*(-2*dlog(t))**(1/2.d0)*dsin(2*3.1415*c)
                z=z+dt**(1/2.d0)*(-2*dlog(q))**(1/2.d0)*dcos(2*3.1415*s)

        enddo

        if (u==2115) then
                r=r+12
        elseif (u<=400) then
                r=r+10
        elseif (u>1800 .and. u<=2114) then
                r=r+40
        else
                r=r+15
        endif
ENDDO

print*,r/N
```

5.3 Problème du Naufragé

```
DO i=1,N
        x=0.d0
        y=0.d0
        Tf=0.d0

        do while (((x+5.d0)**2+(y)**2)**(1/2.d0)>2.and.((x)**2+(y-3.d0)**2)**(1/2.d0)>1&
        &.and.((x-4.d0)**2+(y+5.d0)**2)**(1/2.d0)>3.and.&
        &Tf<10.d0)

                Tf=Tf+dt
```

```
                ix=abs(ix*na)
                ix=mod(ix,nmax)
                c=(dble(ix)/dble(nmax))

                ix=abs(ix*na)
                ix=mod(ix,nmax)
                t=(dble(ix)/dble(nmax))

                x=x+alpha*dt+sigma*dt**(1/2.d0)*(-2*dlog(t))**(1/2.d0)*dcos(2*3.1415*c)
                y=y+beta*dt+sigma*dt**(1/2.d0)*(-2*dlog(t))**(1/2.d0)*dsin(2*3.1415*c)

        enddo

        if (Tf<10.d0) then
                r=r+1.d0
        endif
ENDDO

print*,r/N
```