

10. Générateur aléatoire uniforme sur $[0,1]$, applications

1 Préliminaire arithmétique

(Peut être sauté en première lecture)

1.1 Division euclidienne

On note \mathbb{N} l'ensemble des entiers naturels. La division euclidienne est l'opération qui, au couple d'entiers $a \in \mathbb{N}$, $b \in \mathbb{N}^*$ fait correspondre le couple d'entiers unique $k \in \mathbb{N}$, $r \in \mathbb{N}$, tel que $a = kb + r$, avec $0 \leq r \leq b - 1$. a est le dividende, b le diviseur, k le quotient et r le reste.

1.2 Définition d'une suite u_n

On choisit un entier positif q , premier, et un entier p tel que $0 < p < q$.

On définit une suite d'entiers u_n par un terme de départ $u_0 \in [1, q - 1]$ (u_0 est appellé le « germe ») et la relation de récurrence : pour $n > 0$, u_n est le reste de la division euclidienne de pu_{n-1} par q . On a donc $pu_{n-1} = kq + u_n$, $k \in \mathbb{N}$. Cette définition peut aussi s'écrire $u_n = pu_{n-1} \pmod{q}$ ou, autre notation, $u_n \equiv pu_{n-1} \pmod{q}$.

Dans tous les exemples qui suivent on choisit $u_0 = 1$, on verra que cela n'a pas beaucoup d'importance.

La figure ci-dessous donne une représentation graphique d'une telle suite dans le cas $q = 11$, $p = 3$, $u_0 = 1$:

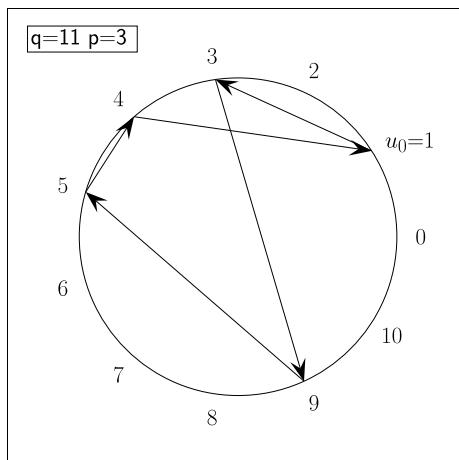


FIGURE 1 –

1.3 Trois propriétés de la suite u_n

1. u_n n'est jamais nul, $\forall n \in \mathbb{N}$.

A priori les termes de la suite appartiennent à l'intervalle $[0, q - 1]$ puisqu'ils sont le reste d'une division par q . Supposons qu'un des termes de la suite, noté u_n , soit nul :

$$pu_{n-1} = kq \implies pu_{n-1} \text{ est divisible par } q$$

Or si un nombre premier divise un produit il divise l'un des facteurs : q divise donc p ou u_{n-1} . Mais on a choisi p strictement inférieur à q et non nul donc q ne peut le diviser. D'autre part, par définition, u_{n-1} vaut au maximum $q - 1$, la seule solution pourqu'il soit divisible par q est donc qu'il soit nul. On a donc $u_n = 0 \implies u_{n-1} = 0$, et, par récurrence décroissante, $u_0 = 0$, ce qui est contraire au choix $u_0 \in [1, q - 1]$.

Remarque

Si q n'était pas premier la suite u_n pourrait être nulle à partir d'un certain rang, comme dans l'exemple suivant : $q = 8$, $p = 6$, $u_0 = 1$, $u_1 = 6$, $u_2 = 4$, $u_3 = 0$, $u_4 = 0$, etc.

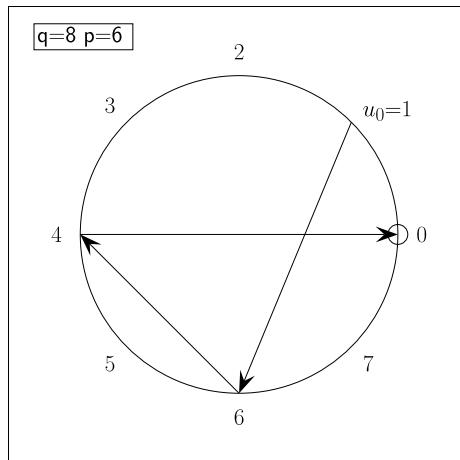


FIGURE 2 –

L'ensemble des valeurs possibles de u_n est donc compris dans l'intervalle $[1, q - 1]$.

2. u_n est périodique et la valeur u_0 est la première à être prise une seconde fois.

Puisque u_n ne peut prendre au plus que $q - 1$ valeurs différentes et qu'il est déterminé de façon unique par u_{n-1} la suite u_n est nécessairement périodique à partir d'un certain rang, avec une période inférieure ou égale à $q - 1$.

D'autre part il est impossible que deux valeurs différentes de u_{n-1} , notées u' et u'' , donnent le même u_n . En effet, il existerait alors k' et k'' tels que :

$$pu' = k'q + u_n \text{ et } pu'' = k''q + u_n$$

ce qui implique :

$$p(u'' - u') = (k'' - k')q$$

donc que q divise $p(u'' - u')$ et donc que q divise $u'' - u'$ (puisque on a vu qu'il ne peut diviser p). Or :

$$1 \leq u' \leq q - 1 \text{ et } 1 \leq u'' \leq q - 1 \implies |u'' - u'| \leq q - 2$$

car le maximum de $|u'' - u'|$ est obtenu pour $u' = 1$ et $u'' = q - 1$ (ou l'inverse). La seule possibilité pour que q divise $u'' - u'$ est donc :

$$u'' = u'$$

Soit u_n une valeur prise une seconde fois. D'après ce qui vient d'être démontré précédemment elle a nécessairement le même prédécesseur u_{n-1} que la première fois, donc u_{n-1} a été prise une seconde fois avant u_n . En poursuivant ce raisonnement par récurrence décroissante on aboutit à u_0 qui est donc la première valeur à être prise une seconde fois. La suite u_n n'est donc pas seulement périodique à partir d'un certain rang mais périodique dès le terme de départ u_0 .

Ce ne serait pas toujours le cas si q n'était pas premier comme le montre l'exemple $q = 20$, $p = 18$, $u_0 = 1$.

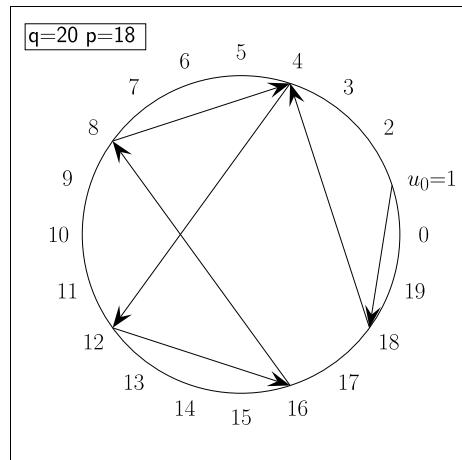


FIGURE 3 –

3. Il existe toujours des valeurs de p telles que u_n prend toutes les valeurs de $[1, q - 1]$

On peut montrer¹ qu'il existe toujours des valeurs particulières de p , appelées « racines primitives » de q , telles que u_n prend toutes les valeurs de l'intervalle $[1, q - 1]$, la période ayant alors sa valeur maximum $q - 1$. La figure suivante montre que le cas $p = 2$ est une racine primitive pour $q = 11$ puisque toutes les valeurs de l'intervalle $[1, 10]$ sont atteintes :

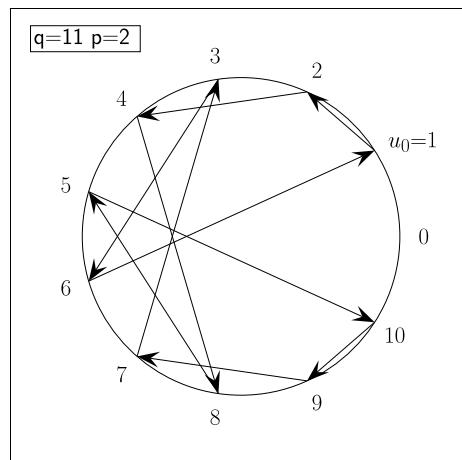


FIGURE 4 –

Si p est une racine primitive, on voit que changer la valeur de u_0 fait décrire la même séquence cyclique mais à partir d'un point de départ différent².

La séquence des valeurs de u_n ayant un aspect désordonné, on considère qu'elle simule une suite d'entiers tirés au hasard avec une probabilité uniforme entre 1 et $q - 1$ ³.

Pour calculer les racines primitives on procède par essais successifs : pour p et q donnés on calcule les u_n jusqu'à retrouver u_0 . Si le nombre de termes différents obtenus est égal à $q - 1$, p est une racine primitive.

Exemple

Pour $q = 11$ on calcule la suite des u_n pour toutes les valeurs de p de 1 à $q - 1$ (en prenant $u_0 = 1$). On obtient :

$q=11$

1. ce qui n'est pas fait ici
 2. donc changer de germe ne produit qu'une permutation circulaire des u_n
 3. cependant une suite réellement aléatoire de $q - 1$ entiers compris entre 1 et $q - 1$ aurait une probabilité très faible de ne pas comporter deux valeurs égales

```

p=1 u(n)=1
p=2 u(n)=1 2 4 8 5 10 9 7 3 6
p=3 u(n)=1 3 9 5 4
p=4 u(n)=1 4 5 9 3
p=5 u(n)=1 5 3 4 9
p=6 u(n)=1 6 3 7 9 10 5 8 4 2
p=7 u(n)=1 7 5 2 3 10 4 6 9 8
p=8 u(n)=1 8 9 6 4 10 3 2 5 7
p=9 u(n)=1 9 4 3 5
p=10 u(n)=1 10

```

ce qui montre que les racines primitives de 11 sont 2, 6, 7 et 8.

Les cas $p=2$ à $p=10$ sont représentés graphiquement sur la figure suivante :

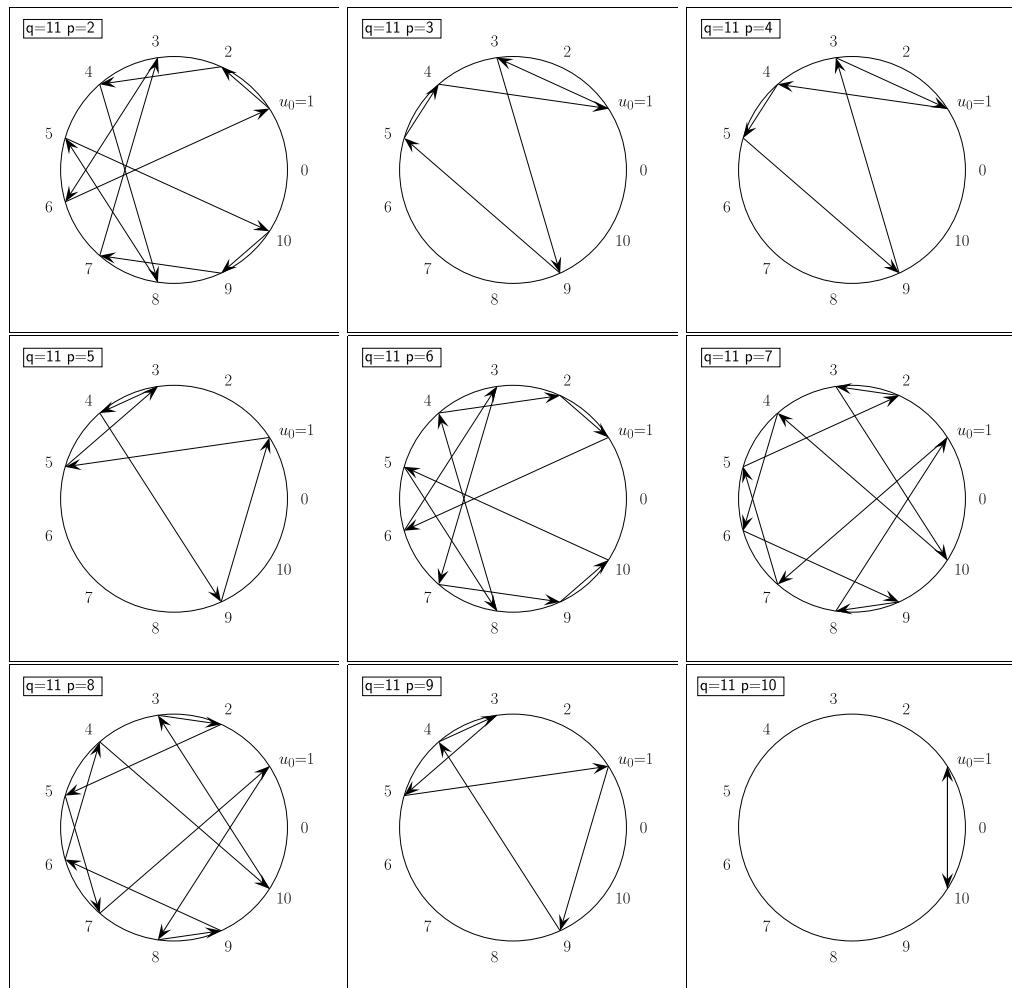


FIGURE 5 –

Pour l'usage pratique qui nous intéresse on veut une séquence d' u_n la plus longue possible. On peut choisir par exemple le nombre premier $q = 1664713$ et la racine primitive $p = 1288$. Il y a un très grand nombre de choix possibles mais q doit être suffisamment grand pour que la période soit grande et on est limité par le fait que le produit $p(q - 1)$ ne doit pas dépasser la valeur maximum possible pour un int c'est à dire $2^{31} - 1$. On a constitué ainsi l'équivalent approché d'un dé à $q - 1 = 1664712$ faces équiprobables. Mais ce dé n'en est pas vraiment un parce que si on le lance 1664712 fois on n'obtiendra jamais la même valeur et, au delà de 1664712, on retrouvera la même séquence.

Remarque

La méthode pour simuler le dé à $q - 1$ faces utilisée ici est très sommaire. Il en existe de plus raffinées permettant d'avoir des suites « ressemblant mieux » à des suites aléatoires.

2 Simulation d'une fonction à valeurs pseudo-aléatoires réelles à distribution uniforme sur $[0, 1]$

On souhaite écrire une fonction qui, à chaque appel, renvoie une valeur aléatoire réelle x telle que la probabilité dp pour qu'elle soit comprise entre x et $x + dx$ est :

$$\begin{aligned} dp &= dx \quad \text{pour } x \in [0, 1] \\ dp &= 0 \quad \text{sinon} \end{aligned}$$

Notation :

On désigne par $\text{rect}_{[a,b]}(x)$ la fonction valant 1 pour $a \leq x \leq b$ et 0 ailleurs.

La densité de probabilité de la variable aléatoire x est donc $\text{rect}_{[0,1]}(x)$.

Pour cela, on utilise le dé à $q - 1$ faces équiprobables numérotées de 1 à $q - 1$ réalisé à la section précédente et, pour un lancer du dé donnant la valeur n , on prend pour x la valeur réelle $\frac{n-1}{q-2}$.

Cette fonction peut être programmée de la façon suivante :

```
double alea() {
    const int q = 1664713, p = 1288;
    static int n = 1;
    n = n * p % q;
    return (double)(n-1)/(q-2);
}
```

(on a choisi arbitrairement le terme initial égal à 1). Il est indispensable que `n` soit statique sinon elle serait réinitialisée à 1 à chaque appel de `alea()` qui fournirait toujours la même valeur.

Un inconvénient de cette fonction est qu'à chaque nouvelle exécution du programme qui la contient, la séquence sera la même puisque la valeur initiale est toujours 1. On rajoute donc une seconde fonction permettant de faire varier la valeur initiale et nommée pour cela `germe`, ce qui conduit à :

```
const int q = 1664713, p = 1288;
int n = 1;
void germe(int g) {
    n = g % q; /* on met g%q et non g pour que n*p ne risque pas de dépasser
                   la valeur max permise pour un int : 2^31-1 */
    if(n == 0) {cout << "Prendre une valeur du germe >= 1 et <= " << q-1 << endl; exit 0;}
}
double alea() {
    n = n * p % q;
    return (double)(n-1)/(q-2);
}
```

la variable globale `n` étant commune aux deux fonctions `germe` et `alea`, on modifie sa valeur en appelant `germe` avec une valeur quelconque de l'argument.

Remarque

Les deux instructions :

```
n = g % q;
if(n == 0) ...
```

peuvent être condensées en une seule :

```
if((n = g%q) == 0) ...
```

ou même :

```
if(!(n = g%q)) ...
```

Exemple d'utilisation de `germe` et `alea` :

```
#include<iostream>
using namespace std;
#include<bibli_fonctions.h>
int main() {
    int i;
    germe(469880);
    for(i = 1; i <= 10; i++)
        cout << alea() << endl;
    return 0;
}
```

Résultat :

```
0.549417
0.648615
0.415484
0.142915
0.0749385
0.521488
0.676658
0.535218
0.361277
0.325148
```

Il n'est pas nécessaire de ré-écrire les fonctions `germe` et `alea` pour les utiliser, car elles sont placées dans une bibliothèque écrite au fur et à mesure pour les besoins du magistère. Il suffit de placer un `#include<bibli_fonctions.h>` au début du fichier contenant le programme, et `germe` et `alea` sont alors disponibles, exactement comme les fonctions de la bibliothèque mathématique du C telles que `sin` ou `pow` le sont, à condition d'avoir mis un `#include<math.h>`. On verra dans un chapitre ultérieur comment un utilisateur peut constituer ainsi sa propre bibliothèque.

La fonction `alea` a des applications nombreuses et puissantes en physique sous le nom de « Méthode de Monte-Carlo » dont on étudie quelques exemples dans la suite.

Remarque

Le C fournit une fonction analogue à `alea` nommée `drand48` qui s'emploie exactement de la même façon. Pour pouvoir l'utiliser il faut mettre `#include<stdlib.h>` (en C) ou `#include<cstdlib>` (en C++). L'équivalent de `germe` est `srand48`. Le programme précédent ré-écrit avec `drand48` et `srand48` se présente ainsi :

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int main() {
    int i;
    srand48(469880);
    for(i = 1; i <= 10; i++)
        cout << drand48() << endl;
    return 0;
}
```

Résultat :

```
0.759624
0.717241
0.8973
0.785434
0.366478
0.66685
0.0656149
0.639337
0.41923
0.117331
```

L'intérêt de `drand48` est qu'il utilise des entiers de 6 octets au lieu de 4 et fournit donc plus de nombres pseudo-aléatoires différents que `alea`. L'intérêt de `alea` est qu'il est particulièrement simple ce qui permet de voir facilement le mécanisme de génération des nombres pseudo-aléatoires. Les générateurs pseudo-aléatoires ayant tous des défauts, il est prudent de faire le même calcul avec plusieurs d'entre eux, aussi différents que possible les uns des autres.

3 Test du générateur uniforme

3.1 Distribution statistique

On a vu dans les sections précédentes comment générer des nombres pseudo-aléatoires sur l'intervalle $[0, 1]$ avec une densité uniforme, à l'aide de la fonction `alea()`. On peut tester la fonction `alea()` en constituant la distribution statistique d'un grand nombre de tirages⁴. Pour cela on divise l'intervalle $[0, 1]$ en N sous-intervalles consécutifs de longueurs égales :

$[0, 1/N[, [1/N, 2/N[, [2/N, 3/N[\dots [(N - 1)/N, 1]$

et, pour une suite d'un grand nombre de tirages, on compte combien sont tombés dans le premier intervalle, combien dans le second, etc. jusqu'au $N^{\text{ième}}$ intervalle. On note y_i le nombre de tirages tombés dans l'intervalle numéro i . Les y_i représentent une fonction à valeurs entières y de la variable entière i qui est la distribution statistique sur N intervalles des valeurs successives de `alea()`. Ceci peut, schématiquement, être fait par les instructions suivantes :

```
int i, k, kmax = 10000, n = 100 // kmax nombre de tirages, n nombre d'intervalles
int* y = (int*)malloc(n*sizeof(int)); // (ou int* y = I_1(n); si on utilise
... // les fonctions du Magistère)
for(k = 1; k <= kmax; k++) {
    i = floor(alea()*n);
    if(i < 0 || i > n-1) continue; // on teste i pour être sûr de ne
    y[i] = y[i]+1; // pas sortir des limites du tableau
}
...

```

Il ne reste plus ensuite qu'à faire tracer la fonction y_i sous forme d'histogramme par exemple, et on obtient :

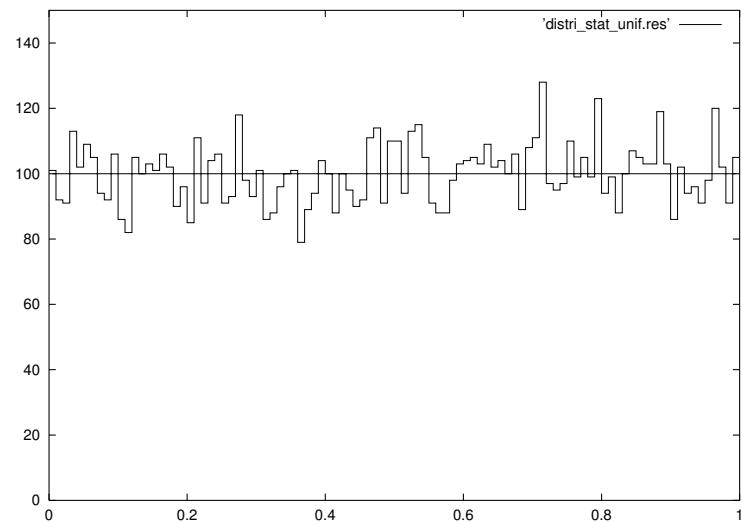


FIGURE 6 –

3.2 Corrélations

On peut également tester s'il y a corrélation entre deux tirages consécutifs x_n et x_{n+1} en portant dans un plan les points de coordonnées $x = x_n$ et $y = x_{n+1}$. On obtient :

4. ici : tirage = valeur obtenue par un appel de `alea()`

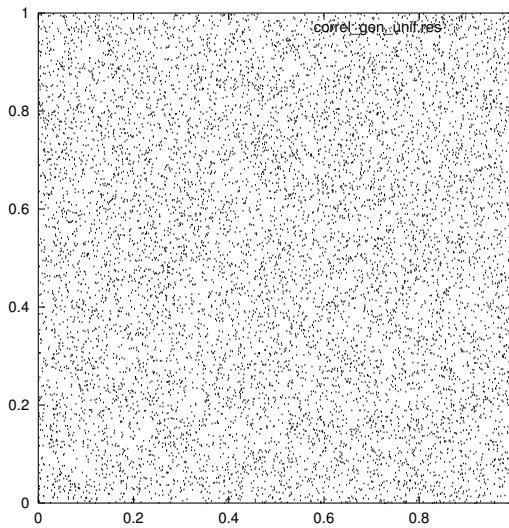


FIGURE 7 –

4 Générateur d'une variable aléatoire suivant une loi donnée

On remarque que si x est une variable aléatoire de densité $\text{rect}_{[0,1]}(x)$, $px + q$ est une v. a. de densité $\frac{1}{p} \text{rect}_{[q,p+q]}(x)$ (le facteur $\frac{1}{p}$ assurant la normalisation à 1). De façon plus générale, comment, à partir d'une v. a. de densité uniforme, construire une v. a. de densité quelconque ?

4.1 Méthode de l'inversion de la primitive

4.1.1 Principe

1) Soit x une v. a. de densité $f(x)$. Par définition de la densité, la probabilité dp pour que x soit dans l'intervalle $[x, x + dx]$ est :

$$dp = f(x) dx$$

Soit une fonction $y(x)$: quelle est la densité $g(y)$ de la v. a. y ? On a :

$$x \in [x, x + dx] \implies y \in [y, y + dy]$$

donc, par définition de la densité de y :

$$dp = g(y) dy = f(x) dx \implies g(y) = f(x) \frac{dx}{dy}$$

2) On envisage la réciproque du problème précédent : quelle doit être la fonction $y(x)$ pour que, x ayant la densité $f(x)$, y ait la densité $g(y)$? On a vu que :

$$g(y) \frac{dy}{dx} = f(x)$$

Soit G une primitive de g , F une primitive de f :

$$\frac{d}{dx} G(y) = g(y) \frac{dy}{dx} = f(x) = \frac{dF}{dx} \implies G(y) = F(x) + \text{cte}$$

On se place dans le cas où G^{-1} existe et est calculable⁵. On peut alors écrire :

$$y = G^{-1} [F(x) + \text{cte}]$$

5. Sinon il faudra employer la méthode de Von Neuman exposée ci-dessous

Il est normal de trouver une constante arbitraire puisqu'on a résolu une équation différentielle du premier ordre. Le choix de la valeur y_0 que l'on veut attribuer à y pour une valeur particulière de x notée x_0 détermine la constante qui vaut : $G(y_0) - F(x_0)$. Finalement :

$$y = G^{-1} [F(x) - F(x_0) + G(y_0)]$$

Cas particulier :

si $f(x) = \text{rect}_{[0,1]}(x)$ une primitive de $f(x)$ est $F(x) = x$ et :

$$y = G^{-1} [x - x_0 + G(y_0)]$$

4.1.2 Exemple

Prenons :

$$f(x) = \text{rect}_{[0,1]}(x)$$

$$g(y) = \frac{1}{\alpha} \exp\left(-\frac{y}{\alpha}\right) \text{ pour } y \geq 0 \text{ et } 0 \text{ ailleurs } (\alpha > 0)$$

$g(y)$ est la loi exponentielle que l'on rencontre fréquemment en physique.

α est un paramètre qui représente la valeur moyenne de y :

$$\langle y \rangle = \frac{1}{\alpha} \int_0^{+\infty} y \exp\left(-\frac{y}{\alpha}\right) dy = \alpha$$

Une primitive de $g(y)$ est $G(y) = -\exp\left(-\frac{y}{\alpha}\right)$ dont la fonction réciproque est $G^{-1}(z) = -\alpha \ln(-z)$. Donc :

$$y = G^{-1} [x - x_0 + G(y_0)] = -\alpha \ln \left[-x + x_0 + \exp\left(-\frac{y_0}{\alpha}\right) \right]$$

Choisissons par exemple $y(0) = 0$, on a alors $x_0 = 0$ et $y_0 = 0$ et :

$$y = -\alpha \ln(-x + 1)$$

$1 - x$ et x ayant la même distribution $\text{rect}_{[0,1]}(x)$ on peut écrire plus simplement :

$$y = -\alpha \ln(x)$$

On pourra donc, dans un programme, tirer des valeurs aléatoires dont la densité suit une loi exponentielle de paramètre α par l'instruction :

```
...
y = -alpha*log(alea());
...
```

On peut, exactement comme dans le cas uniforme, faire une distribution statistique des valeurs obtenues avec, par exemple, 10000 tirages et comparer avec la courbe théorique $1/\alpha \exp(-y/\alpha)$, ce qui donne le résultat suivant :

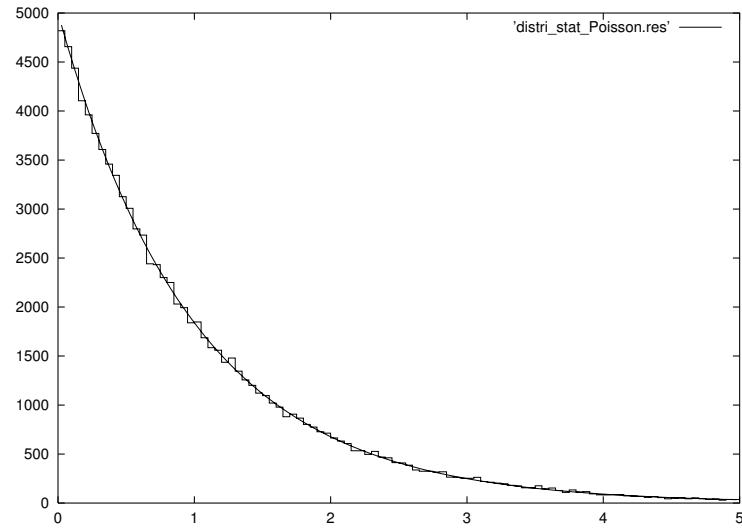


FIGURE 8 –

4.2 Méthode du rejet de Von Neuman

Dans le cas où la primitive de la densité recherchée n'est pas inversible et calculable, on applique une méthode beaucoup plus générale.

Soit x une v. a. de loi $f(x)$ sur $[a, b]$ et soit M la valeur supérieure⁶ de $f(x)$ sur $[a, b]$.

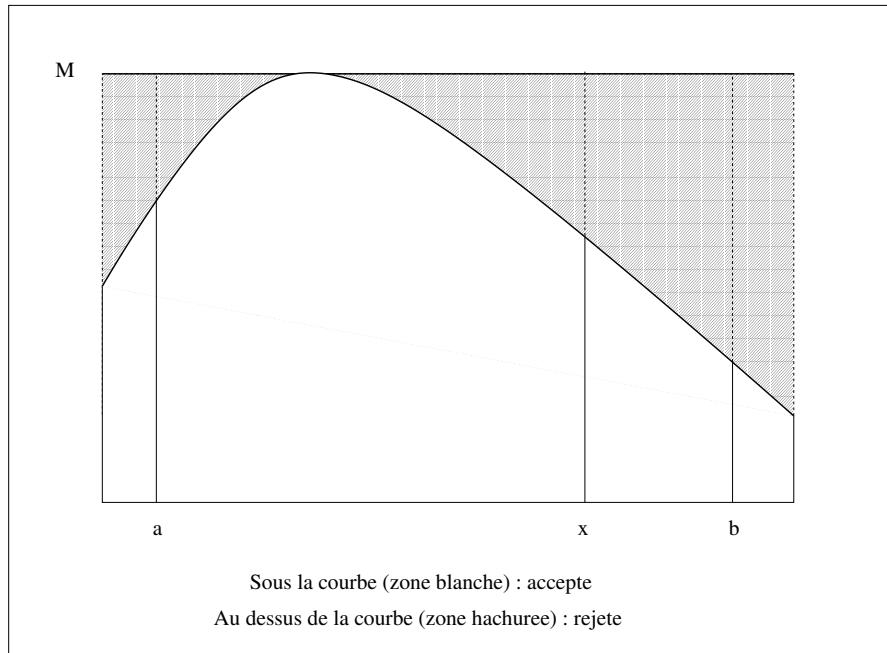


FIGURE 9 –

On tire une valeur aléatoire x avec une densité uniforme sur $[a, b]$, puis une valeur aléatoire y avec une densité uniforme sur $[0, M]$. Si le point de coordonnées (x, y) est au-dessus de la courbe de $f(x)$ on le rejette, sinon on l'accepte. Les valeurs de x ainsi sélectionnées suivent la loi de densité $f(x)$. En effet, supposons qu'on ait effectué N_0

6. supposée exister

doubles tirages (x et y). Le nombre de valeurs de x situées dans l'intervalle $[x, x + dx]$ est :

$$dN = N_0 \underbrace{\frac{dx}{b-a}}_{\text{1er tirage}} \underbrace{\frac{f(x)}{M}}_{\text{2ème tirage}}$$

Pour déduire de dN la densité de probabilité de x , il faut diviser dN par le nombre N de valeurs de x effectivement obtenues, qui est différent de N_0 puisqu'il y a des valeurs rejetées. Le nombre de valeurs de x retenues est :

$$N = \int_a^b dN = \frac{N_0}{(b-a)M} \int_a^b f(x) dx$$

$f(x)$ étant une densité de probabilité on a :

$$\int_a^b f(x) dx = 1$$

soit :

$$N = \frac{N_0}{(b-a)M}$$

Donc :

$$dp = \frac{dN}{N} = f(x) dx$$

La densité de probabilité de x est bien $f(x)$.

Exemple

On veut générer des valeurs aléatoires avec la densité :

$$\frac{3}{2} \sin x \cos^2 x \quad \text{sur} \quad [0, \pi]$$

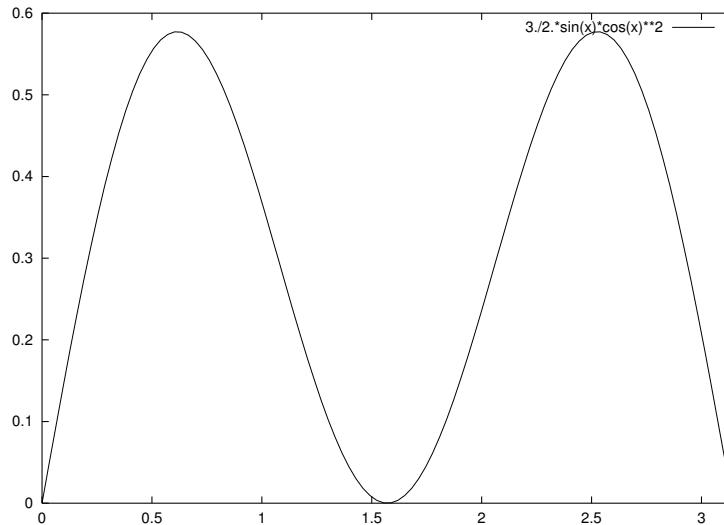


FIGURE 10 –

Le maximum de cette fonction est $1/\sqrt{3}$. Donc :

on tire avec une densité uniforme sur $[0, \pi]$ ce qui donne une valeur x

on tire avec une densité uniforme sur $[0, 1/\sqrt{3}]$ ce qui donne une valeur y

si $y \leq \frac{3}{2} \sin x \cos^2 x$ alors on retient x sinon on le rejette et on recommence.

Remarque

En pratique si, au lieu d'utiliser $f(x)$ qui est normée à 1, on utilise $f'(x) = \lambda f(x)$ (λ quelconque $\in \mathbb{R}$), le résultat est le même puisque le maximum M' de $f'(x)$ est multiplié dans le même rapport λ et on a : $f(x)/M = f'(x)/M'$. Il n'est donc pas nécessaire de travailler avec une fonction normée à 1.

Complément :

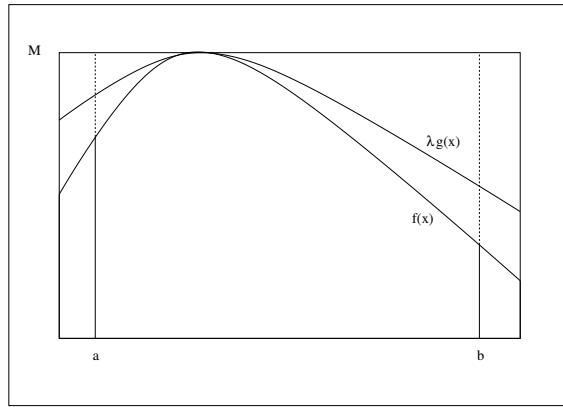


FIGURE 11 –

L'inconvénient de cette méthode est qu'il peut y avoir beaucoup de tirages inutiles. On peut y remédier si on connaît une densité $g(x)$ sur $[a, b]$ pour laquelle on sait générer des valeurs aléatoires et que cette densité satisfait à la propriété :

$$\exists \lambda \in \mathbf{R} \text{ tel que } \lambda g(x) \geq f(x) \forall x \in [a, b]$$

On procède encore à un double tirage mais :

au lieu de tirer x avec une densité uniforme sur $[a, b]$, on le tire avec la densité $g(x)$
on tire y avec une densité uniforme sur $[0, \lambda g(x)]$

puis, comme précédemment, on rejette le point s'il est au-dessus de la courbe de $f(x)$.

Pour N_0 doubles tirages le nombre de valeurs de x situées dans l'intervalle $[x, x + dx]$ est donc :

$$dN = N_0 \underbrace{g(x)dx}_{\text{1er tirage}} \underbrace{\frac{f(x)}{\lambda g(x)}}_{\text{2ème tirage}}$$

Le nombre de valeurs de x retenues est :

$$N = \int_a^b dN = \frac{N_0}{\lambda} \int_a^b f(x) dx = \frac{N_0}{\lambda}$$

Donc :

$$dp = \frac{dN}{N} = f(x) dx$$

La densité de probabilité de x est bien $f(x)$.

Remarque

On est obligé d'introduire le facteur λ car on ne pourrait avoir simultanément : $\int_a^b f(x) dx = 1$, $\int_a^b g(x) dx = 1$
et $g \geq f$ sauf si $f = g$.

5 Applications

5.1 Mouvement brownien

On étudie le mouvement brownien d'une particule dans un plan. Entre deux chocs contre les molécules du milieu elle parcourt en ligne droite une distance l dont la densité de probabilité est une loi exponentielle dont le paramètre est son libre parcours moyen \bar{l} . Chaque choc est considéré comme le tirage au hasard :

- d'un angle entre 0 et 2π avec une densité uniforme
- d'une longueur entre 0 et l'infini avec une densité exponentielle

On a donc déjà tous les éléments pour faire le calcul et on obtient :

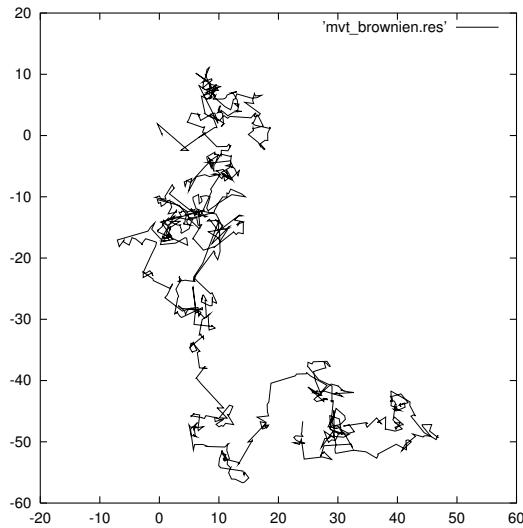


FIGURE 12 –

À partir de là, en étudiant les positions d'un grand nombre de particules en fonction du nombre de chocs subis, on reproduit facilement la diffusion d'un nuage de particules dans un gaz par exemple (diffusion de la fumée de cigarette en l'absence de courant d'air).

5.2 Rayonnement d'une sphère radioactive

On considère une sphère homogène, de rayon R , constituée d'un matériau radioactif émettant des photons γ . Un photon peut être émis depuis n'importe quel point de la sphère et dans n'importe quelle direction avec la même probabilité. On peut considérer schématiquement que dans la matière, chaque photon est absorbé après un trajet rectiligne dont la longueur l varie aléatoirement avec la densité :

$$\frac{1}{\alpha} \exp\left(-\frac{l}{\alpha}\right) \quad (1)$$

Les parcours des photons peuvent être représentés de la façon suivante :

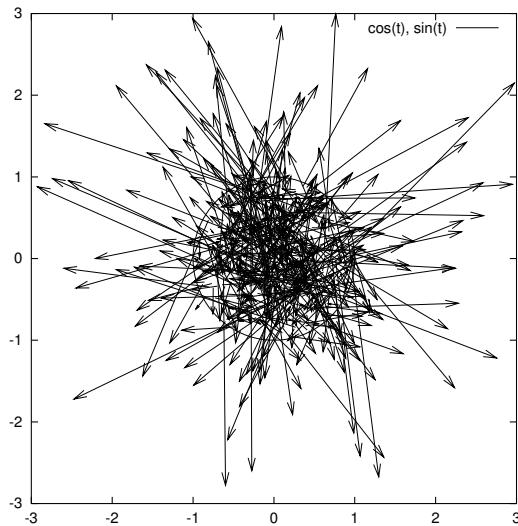


FIGURE 13 –

(contrairement à ce qu'indique la figure, les photons qui sortent de la sphère ne sont plus absorbés ensuite)

On sait que α représente le libre parcours moyen (=paramètre du problème).

On veut calculer le taux moyen τ de photons qui sortent de la sphère :

$$\tau = \frac{\text{nombre de photons sortant}}{\text{nombre de photons émis depuis l'intérieur de la sphère}}$$

Il est évident que τ dépend de la valeur relative de $\langle l \rangle$ et R et on peut dire à priori :

$$\langle l \rangle \gg R \implies \tau \simeq 1$$

$$\langle l \rangle \ll R \implies \tau \simeq 0$$

on cherche donc à calculer τ en fonction du rapport $\langle l \rangle / R$.

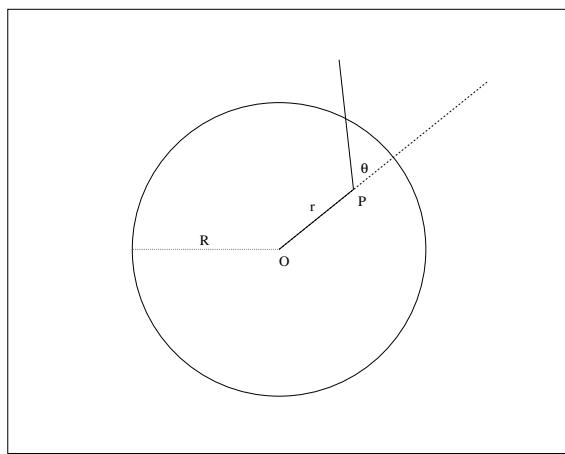


FIGURE 14 –

Pour cela on simule l'expérience :

on tire au hasard un noyau émetteur P dans la sphère et une direction d'émission.

on en déduit la longueur d à parcourir pour, à partir de ce point et dans cette direction, parvenir à la surface de la sphère

on tire au hasard une longueur de parcours l du photon selon la loi (1)
si $l \geq d$ le photon sort

On renouvelle cette simulation un grand nombre de fois et on en déduit τ .

La sphère étant homogène et l'émission isotrope le problème se simplifie. Tous les noyaux émetteurs situés à la même distance du centre de la sphère donnent la même contribution. Pour tirer au hasard la position d'un noyau il suffit de tirer une valeur de r entre 0 et R avec un densité proportionnelle à l'élément de volume $4\pi r^2 dr$.

On repère la direction d'émission par ses angles (θ, φ) dans un repère sphérique de centre P et d'axe $Oz \parallel OP$. Le système constitué par la sphère et le noyau ayant la symétrie de révolution autour de l'axe OP , la valeur de φ n'intervient pas. Il suffit de tirer un angle θ entre 0 et π avec une densité proportionnelle à l'élément d'angle solide $2\pi \sin \theta d\theta$.

Enfin un calcul géométrique indique que la longueur à parcourir pour sortir de la sphère depuis le point P dans la direction déterminée par θ est :

$$d = -r \cos \theta + \sqrt{R^2 - r^2 \sin^2 \theta}$$

Les densités de r et θ sont assez simples pour qu'on puisse générer les valeurs aléatoires par la méthode de l'inversion de la primitive.

Génération des valeurs de r

La densité de r est :

$$\frac{4\pi r^2 dr}{\frac{4}{3}\pi R^3} = \frac{3r^2 dr}{R^3}$$

x étant la v. a. à densité uniforme sur $[0, 1]$ il s'agit de trouver la fonction $r(x)$ telle que r ait la distribution précédente. On a vu que :

$$r = G^{-1} [x - x_0 + G(r_0)]$$

G étant ici une primitive de $\frac{3r^2}{R^3}$ c'est à dire $\frac{r^3}{R^3}$. G est donc inversible et on a :

$$r = R \left(x - x_0 + \frac{r_0^3}{R^3} \right)^{1/3}$$

On choisit que $r = 0$ pour $x = 0$, on a donc finalement :

$$r = Rx^{1/3}$$

Génération des valeurs de θ

La densité de θ est :

$$\frac{2\pi \sin \theta}{4\pi} = \frac{\sin \theta}{2}$$

donc ici :

$$G = -\frac{\cos \theta}{2}$$

Cette fonction est inversible sur $[0, \pi]$:

$$\theta = \arccos \left[-2(x - x_0 - \frac{1}{2} \cos \theta_0) \right]$$

On choisit que $\theta = 0$ pour $x = 0$, donc :

$$\theta = \arccos(-2x + 1)$$

Génération des valeurs de l

On a vu ci-dessus que, la densité de l étant une loi exponentielle de paramètre $\alpha = \langle l \rangle$:

$$l = -\alpha \ln x$$

Tous les éléments étant réunis, la simulation d'une émission peut s'écrire :

```
...
double a = 1./3.;
for(s = 0, i = 1; i <= n; i++) {
    r = pow(alea(),a)*R;                      /* position du noyau émetteur */
    th = acos(2.*alea()-1.);                   /* direction d'émission du photon */
    rs = r*sin(th);
    d = -r*cos(th) + sqrt(R2-rs*rs);          /* distance à parcourir pour sortir */
    do
        x = alea();
        while(x == 0.);
        l = -alpha*log(x);                      /* longueur parcourue par le photon */
        if(l > d)                                /* on compte les photons qui sortent */
            s++;
    }
...

```

Le programme entier est donné en annexe. Il fait ce calcul pour des valeurs de α échelonnées entre 0 et R , ce qui donne la courbe suivante pour τ en fonction de α/R :

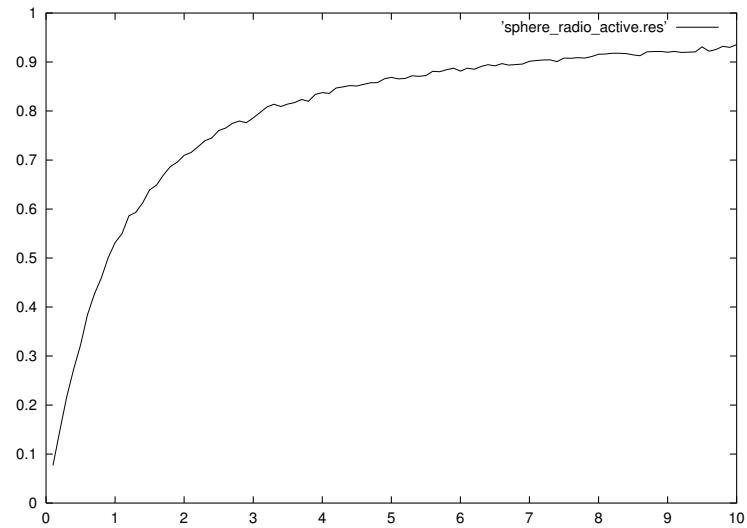


FIGURE 15 –

Remarque

Pour faire le calcul analytiquement il faudrait calculer l'intégrale :

$$8\pi^2 \int_0^R \int_0^\pi r^2 \sin \theta_P \left[1 - \exp\left(-\frac{d}{\alpha}\right) \right] dr d\theta_P$$

avec :

$$d = -r \cos \theta + \sqrt{R^2 - r^2 \sin^2 \theta}$$

5.3 Miroir hémisphérique

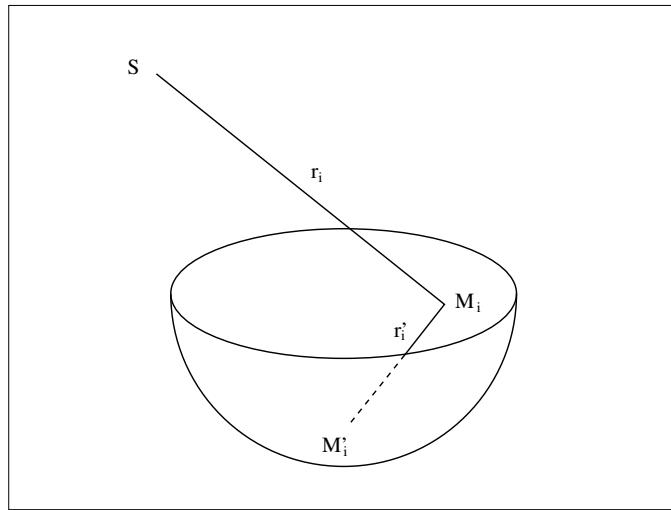


FIGURE 16 –

On considère un miroir hémisphérique concave et une source ponctuelle S , située n'importe où, mais de telle façon qu'au moins une partie de ses rayons puissent se réfléchir sur la surface concave du miroir⁷. On tire une séquence de rayons lumineux r_i issus de S , répartis au hasard de façon isotrope dans toutes les directions de l'espace. Si un rayon

7. S doit donc être du côté opposé à la demi-sphère par rapport au plan Π contenant le bord du miroir

r_i atteint la surface concave du miroir on appelle M_i le point d'incidence et on calcule la position de l'intersection M'_i du premier rayon réfléchi r'_i avec le miroir. Puis on représente graphiquement en perspective le miroir et l'ensemble des points M'_i . On obtient la caustique suivante :

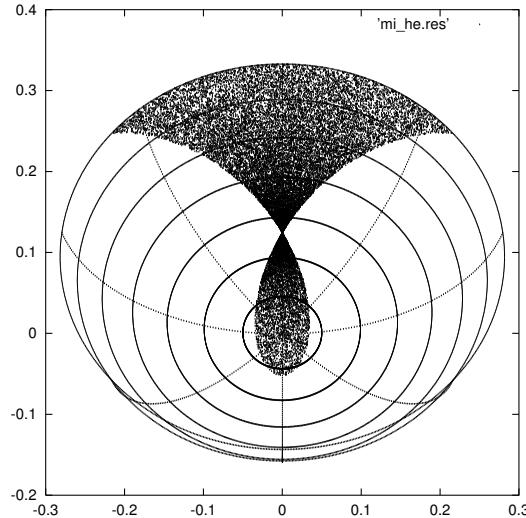


FIGURE 17 –

La méthode consistant à utiliser des rayons lumineux distribués au hasard de façon isotrope est la plus simple car il n'est pas possible de répartir de façon régulière un grand nombre de rayons dans toutes les directions de l'espace.

5.4 Représentation d'une orbitale atomique

5.4.1 Densité de probabilité de présence

Considérons l'état $n = 2$, $l = 1$, $m = 0$ de l'électron de l'atome d'hydrogène. La fonction d'onde s'écrit :

$$\psi(\mathbf{r}) = \frac{1}{4\sqrt{2\pi a_0^3}} \frac{r}{a_0} \exp\left(-\frac{r}{2a_0}\right) \cos\theta$$

(les vecteurs sont représentés en caractères gras)

a_0 = rayon de Bohr = $\frac{\hbar^2}{\mu e^2} = 0.52 \text{ \AA}$, μ = masse réduite du système électron proton, e = charge de l'électron.

La densité de probabilité de présence de l'électron dans l'élément de volume $d^3\mathbf{r} = r^2 \sin\theta dr d\theta d\varphi$ est :

$$|\psi(\mathbf{r})|^2 = \alpha \left(\frac{r}{a_0}\right)^2 \exp\left(-\frac{r}{a_0}\right) \cos^2\theta$$

α est une constante qu'il est inutile d'expliciter. On choisit a_0 comme unité de longueur. La probabilité de présence dp dans l'élément de volume $d^3\mathbf{r}$ est donc :

$$dp = \alpha r^4 \exp(-r) \sin\theta \cos^2\theta dr d\theta d\varphi$$

et, puisque cette densité ne dépend pas de φ , il suffit de la représenter dans un plan $\varphi = \text{constante}$ ⁸ en fonction de r et θ . D'autre part cette densité est un produit d'une fonction de r par une fonction de θ donc l'étude, à priori à deux dimensions, se réduit à deux fois une dimension. Il suffit de générer des couples r, θ avec la loi :

$$\begin{aligned} r^4 \exp(-r) &\text{ pour } r \in [0, +\infty[\\ \sin\theta \cos^2\theta &\text{ pour } \theta \in [0, \pi] \end{aligned}$$

On peut de plus utiliser le fait que la fonction $\sin\theta \cos^2\theta$ est symétrique par rapport à $\pi/2$ et se contenter de tirer θ dans l'intervalle $[0, \pi/2]$.

8. c'est à dire un plan passant par Oz

5.4.2 Loi de r

La fonction $f(r) = r^4 \exp(-r)$ peut être considérée comme négligeable à partir d'une certaine valeur r_{\max} de r .

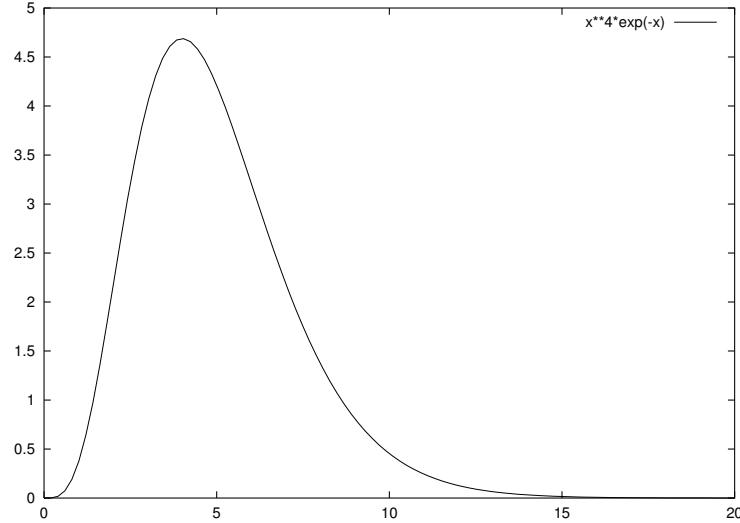


FIGURE 18 –

En prenant par exemple $r_{\max} = 15$ on a $f(r_{\max}) = 3.3 \cdot 10^{-4}$.

Par ailleurs la valeur maximum atteinte par $f(r)$ est $M = (4/e)^4$. Finalement :

on tire avec une densité uniforme sur $[0, r_{\max}]$ ce qui donne une valeur r
 on tire avec une densité uniforme sur $[0, M]$ ce qui donne une valeur x
 si $x \leq f(r)$ on garde r sinon on le rejette.

5.4.3 Loi de θ

A la section **Méthode du rejet de Von Neuman** on a déjà vu, en exemple, comment générer des valeurs de θ suivant la loi $\sin \theta \cos^2 \theta$.

5.4.4 Mise en œuvre

Le cœur du programme s'écrit :

```

...
int imax = 30000; double rmax = 15.;
const double c1 = pow(4./exp(1.),4.), c2 = 2./3./sqrt(3.);
...
for(i = 1; i <= imax; i++) {
  do {
    r = rmax*alea();
    x = c1*alea();
  } while(x > pow(r,4.)*exp(-r));
  do {
    tet = 2.*M_PI*alea();
    x = c2*alea();
  } while(x > fabs(sin(tet)*cos(tet)*cos(tet))); /* on doit prendre la valeur absolue
  parce qu'on choisit de tirer tet sur
  [0,2pi] au lieu de [0,pi/2] */
  fich << r*cos(tet) << " " << r*sin(tet) << endl;
}
...

```

ce qui donne la représentation :

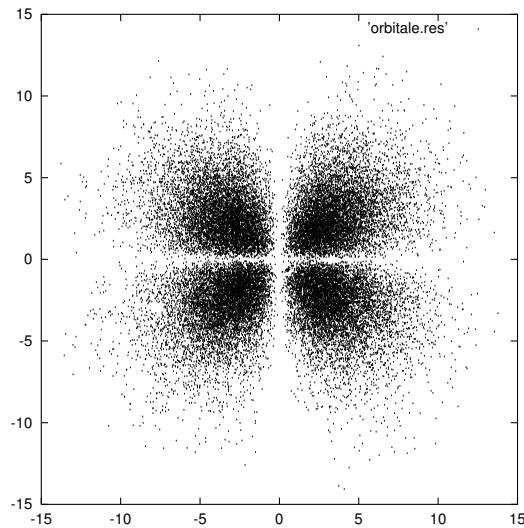


FIGURE 19 –

Remarque

On a choisi de tirer θ sur $[0, 2\pi]$ au lieu de $[0, \pi/2]$ uniquement pour obtenir une représentation sur les quatre quadrants, donc plus concrète.

A titre de comparaison, le tracé de la surface $z = r^4 \exp(-r) \sin \theta \cos^2 \theta$ donne :

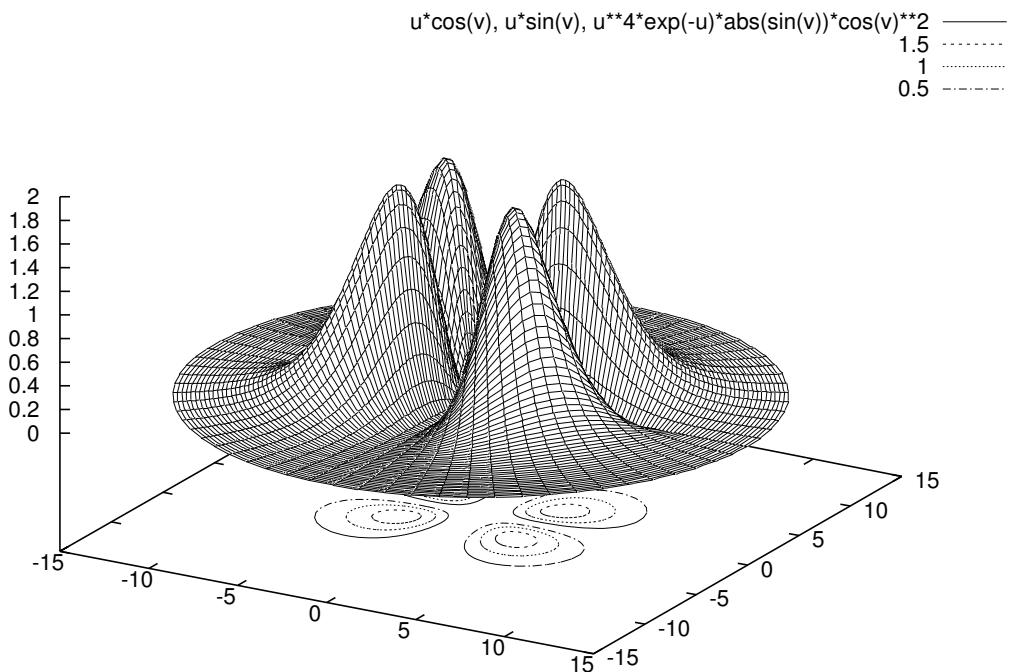


FIGURE 20 –

Annexe : rayonnement d'une sphère radioactive

```

#include<iostream>
#include<fstream>
#include<math.h>
#include<stdlib.h>
#include<time.h>
using namespace std;
int main() {
    /* CALCUL DU TAUX DE SORTIE DE PHOTONS GAMMA D'UNE SPHERE RADIOACTIVE
    EN FONCTION DU RAPPORT: PARCOURS MOYEN DES PHOTONS/RAYON DE LA SPHERE
    Notations:
    R rayon de la sphere
    alpha parcours moyen des photons dans la sphere
    r distance du noyau émetteur P au centre O de la sphere
    th angle entre OP et la direction d'émission du photon
    d distance à parcourir pour sortir
    l longueur parcourue par le photon
    s compteur de sorties
    n nombre d'émissions simulées
    k indice de la boucle sur les différentes valeurs de alpha/r
    kmax nombre de valeurs de alpha/r
    i indice de la boucle sur les n émissions simulées
    x variable aléatoire uniforme sur [0,1], de probabilité nulle en dehors
    ****
    -----Declarations---- */
    fstream fich;
    double l, R = 1., R2 = R*R, rs, alpha, d, x, r, th;
    int i, n = 10000, s, k, kmax = 100;
    /* -----
    srand48(time(NULL)); /* Initialise les nombres aléatoires par l'heure actuelle en secondes */
    fich.open("sphere_radioactive.res", ios::out);
    for(k = 1; k <= kmax; k++) {
        alpha = k*R/10.;
        for(s = 0, i = 1; i <= n; i++) {
            r = pow(drand48(), 1./3.) * R; /* position du noyau émetteur */
            th = acos(2.*drand48()-1.); /* direction d'émission du photon */
            rs = r*sin(th);
            d = -r*cos(th) + sqrt(R2-rs*rs); /* distance à parcourir pour sortir */
            do
                x = drand48();
                while(x == 0.0);
                l = -alpha*log(x); /* longueur parcourue par le photon */
                if(l > d)
                    s++;
        }
        cout << "k = " << k << endl;
        cout << "nb de des. = " << n << " nb de photons sortis = " << s
        << " taux de sortie = " << (double)s/n << endl;
        fich << alpha << " " << (double)s/n << endl;
    }
    fich.close();
    return 0;
}

```

Annexe : représentation d'une orbitale atomique

```

#include<fstream>
#include<math.h>
#include<stdlib.h>
#include<time.h>
using namespace std;
int main() {
    double r, tet, x, rmax;
    const double c1 = pow(4./exp(1.),4.), c2 = 2./3./sqrt(3.);
    int i, imax;
    fstream fich;
    fich.open("orbitale.res", ios::out);
    imax = 30000; rmax = 15.;
    srand48(time(NULL));
    for(i = 1; i <= imax; i++) {
        do {
            r = rmax*drand48();
            x = c1*drand48();
        } while(x > pow(r,4.)*exp(-r));
        do {
            tet = 2.*M_PI*drand48();
            x = c2*drand48();
        } while(x > fabs(sin(tet)*cos(tet)*cos(tet)));
        fich << r*cos(tet) << " " << r*sin(tet) << endl;
    }
    return 0;
}

```