5.3. Génération de variables aléatoires

5.3.1. Introduction. Soit à produire une suite $\{x_i\}$ de nombres aléatoires issus de la loi de probabilité d'une variable aléatoire X de fonction de répartition $F(x)=P(X\le x)$. On cherche à obtenir la suite $\{x_i\}$ à partir d'une suite préalable $\{U_i\}$ de nombres au hasard issus de la loi uniforme $U[0,1]$, moyennant une certaine transformation à définir.

Il existe une multitude de méthodes qui se basent généralement sur certaines propriétés propres à la distribution qu'on cherche à générer. Cependant, on peut tenter de mettre en évidence quelques approches générales, susceptibles de pouvoir être utilisées pour la construction de générateurs de certaines classes de distributions. Il n'y a pas de recommandations particulières lorsqu'on est en présence de plusieurs générateurs possibles pour une seule et même distribution. On se basera peut être sur la simplicité de programmation de l'algorithme générateur, la rapidité d'exécution. On veillera également à ce que le générateur construit n'accentue pas les insuffisances de la séquence initiale $\{U_i\}$ du point de vue de l'uniformité, de l'indépendance, de la périodicité...

5.3.2. Méthodes générales.

Chacune des méthodes décrites dans ce paragraphe peut s'appliquer à une large classe de distributions $F(x)$. Dans les paragraphes suivants, nous étudierons les applications de ces méthodes à divers types de lois usuelles.

5.3.2.1. Méthode d'inversion.

Le principe de cette méthode a déjà été présenté précédemment, dans le cas d'une distribution F(x) continue. En effet, $F(.)$F(.) étant monotone croissante, et par conséquent inversible, on peut générer la suite $\{x_i\}$ à partir de l'algorithme suivant.

Algorithme 1.

1.Tirer U issu de la loi uniforme $U[0,1]$.
2. Poser $X=F^{-1}(U)$.

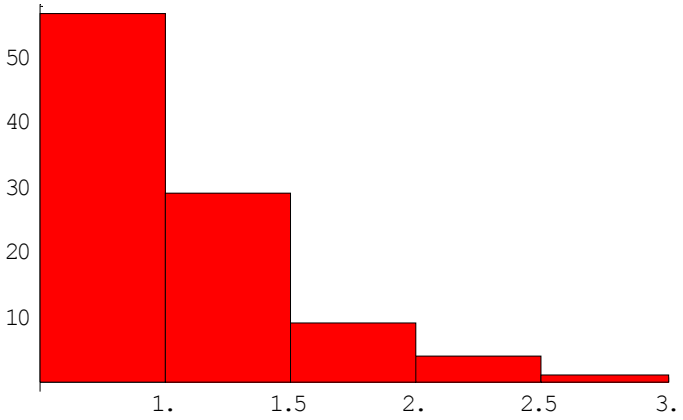
Exemple 8 : Soit la loi exponentielle $f(x)=\lambda e^{-\lambda x}$, $x\ge 0$. L'algorithme d'inversion donne

$$X=-\{Log(1-U)/\lambda\}$$

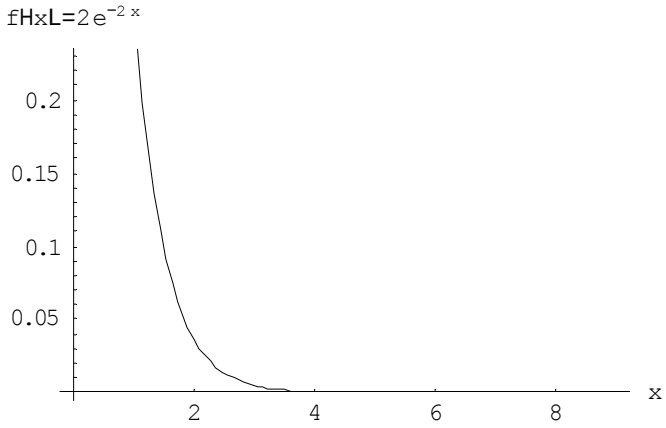
On montre que si $X\in U[0,1]$, alors $X'=1-X\in U[0,1]$, et l'algorithme ci-dessus peut alors s'écrire : $X=-\{Log(U)\}/\lambda$. Cela permet d'économiser en temps de calcul dans les applications de simulation qui nécessite un grand nombre d'appels à ce générateur.

Exemple 9: Le programme suivant génère d'abord 100 observations de la loi exponentielle de paramètre $\lambda=2$ et construit l'histogramme de cette série statistique artificielle.

```
a = Table@Random@D, 8100<D;
Expdata = N@Table@-0.5 Log@a@@iDDD, 8i, 100<DD;
<< Statistics`DataManipulation`
f = RangeCounts@Expdata, 80.5, 1., 1.5, 2.0<D;
<< Graphics`Graphics`
Histogram@f, FrequencyData → True,
 HistogramCategories -> 80.5, 1., 1.5, 2.0, 2.5, 3.0<D
```



On peut comparer l'allure de cette distribution expérimentale avec la vraie densité exponentielle de paramètre λ=2.



5.3.2.2. Cas discret.

Dans le cas où on n'impose plus la contrainte de continuité sur la fonction F, l'algorithme générateur ci-dessus reste valable en remplaçant la fonction inverse F^{-1} par la fonction F^* définie par

$$F^*(u)=\min\{x: F(x)\geq u\} \qquad (5.8)$$

En effet, le minimum dans (5.8) est atteint en raison de la continuité à droite de la fonction F. Par conséquent, $F(F^{-1}(u))\geq u$, et $F^*(F(x))$ = min $\{y:F(y)\geq F(x)\}\leq x$. Il en résulte que les ensembles suivants coïncident: $\{(u,x):F^*(u)\leq x\}$= { (u,x) : $u\leq F(x)$ et la probabilité de l'événement $\{X\leq x\}$ est bien égale à F(x)

$$\mathbf{P}\{X\leq x\}=\mathbf{P}\{F^*(U)\leq x\}=\mathbf{P}\{U\leq F(x)\}=F(x)$$

pour toute variable $U\in U[0,1]$.

Exemple 10. Simulation d'un événement). Soit à simuler un événement A de probabilité p=P(A). Si U est une variable aléatoire de loi uniforme sur U[0,1], alors $\mathbf{P}\{U\leq p\}=p$, c'est-à-dire que la probabilité que U tombe dans l'intervalle (0,p) est égale à la longueur de l'intervalle, soit p, la probabilité de l'événement A. Ainsi, si on connaît la réalisation de U, on peut vérifier que A s'est réalisé simplement en vérifiant l'inégalité $U\leq p$. Cette règle est schématisée par l'algorithme suivant.

Algorithme 2.

1. Tirer $U\in U[0,1]$
2. Si $U\in[0,p]$, alors A s'est réalisé
   Si $U\in(p,1]$, l'évènement contraire $\overline{A}$ s'est réalisé.

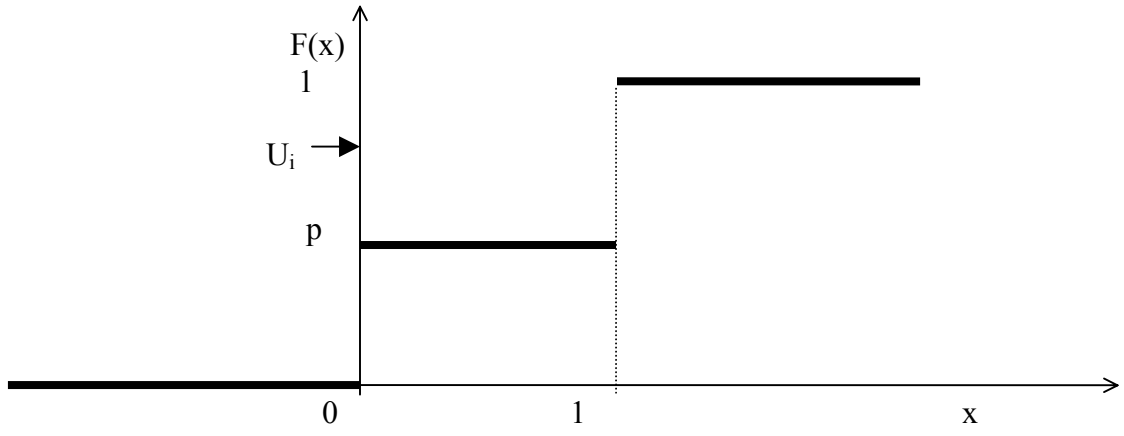
Exemple 11. (Loi de Bernouilli).A l'expérience aléatoire d'issue A ou, on peut associer une variable binaire X de Bernouilli à valeurs dans E={0,1}:

$$X=\begin{cases}0, si\ A\ se réalise\\ 1, si\ \overline{A}\ se réalise\end{cases}$$

Dans ce cas, $p=\mathbf{P}(X=0)=\mathbf{P}(A)$, $q=1-p=\mathbf{P}(X=1)=\mathbf{P}(\overline{A})$, $0<p<1$.

$\mathbf{E}(X)=0.\mathbf{P}(X=0)+1.\mathbf{P}(X=1)=p$; $\mathbf{E}(X^k)=p$; $\mathbf{Var}(X)=\mathbf{E}(X^2)-[\mathbf{E}(X)]^2=p(1-p)$

La fonction de répartition de X est représentée sur la figure suivante.



Elle s'écrit $F(x)=p+(1-p)\chi(x\geq 1)$, où $\chi(A)$ est la fonction caractéristique de l'ensemble A. Par conséquent, $F^*(u)=\chi(u\geq p)$, et la variable X peut être générée à l'aide de l'algorithme 1 qui est équivalent dans ce cas à l'algorithme 2.

Algorithme 2'.

1. Tirer $U \in U[0,1]$.
2. Si $U \in [0,p]$, alors X=0,
3. Si $U \in (p,1]$, alors X=1.

Exemple 12. Jeu de « Pile » ou « Face ».

Simulons le jeu d'un joueur A qui gagne 10 D.A. si la pièce tombe côté « Pile », et en perd 10 D.A. si c'est le côté « Face » qui sort. La simulation portera sur un jeu de 10 lancers, en supposant que p = **P**(A) = **P**(Pile) = 3/4, **P**(Face) = 1/4. La suite $\{U_i\}$ est tirée de la table de l'exemple 5, et a été générée à l'aide de l'instruction Random() de MATEMATIKA, basée sur la méthode congruentielle. La simulation du jeu fournit les résultats suivants.

N° du lancer	Test	Résultat simulé	Gain cumulé
1	U_1=0,40419<3/4	Pile	10 D.A.
2	U_2=0,735378<3/4	Pile	20
3	U_3=0,369251<3/4	Pile	30
4	U_4=0,7051<3/4	Pile	40
5	U_5=0,833575>3/4	Face	30
6	U_6=0,397475<3/4	Pile	40
7	U_7=0,151053<3/4	Pile	50
8	U_8=0,925697>3/4	Face	40
9	U_9=0,928252>3/4	Face	30
10	U_{10}=0,481475<3/4	Pile	40

Ainsi, à l'issue du jeu le joueur a gagné 40 D.A.
Supposons maintenant que la pièce soit parfaite, c'est-à-dire que p=**P**(Pile)=**P**(Face)=1/2

N° du lancer	Test	Résultat simulé	Gain cumulé
1	U_1 =0.40419<0.5	*Pile*	10 *D.A.*
2	U_2=0,735378>1/2	Face	0
3	U_3=0,369251<1/2	Pile	10
4	U_4=0,7051>1/2	Face	0
5	U_5=0,833575>1/2	Face	-10
6	U_6=0,397475<1/2	Pile	0
7	U_7=0,151053<1/2	Pile	10
8	U_8=0,925697>1/2	Face	0
9	U_9=0,928252>1/2	Face	-10
10	U_{10}=0,481475<1/2	Pile	0

On s'aperçoit que dans un jeu transparent, le match serait nul. Mais la conclusion n'est pas représentative car ne se référant qu'à une seule expérience de simulation.

Exemple 13. (Génération d'une variable aléatoire discrète).

Soit X une variable aléatoire discrète de densité de probabilité

$$p_K=\mathbf{P}(X=k) \quad (k=1,2,...), \quad \sum_k p_k = 1$$

et de fonction de répartition: $F(x)=P(X\le x)=\sum_{k\le x} p_k \quad ,x\ge 1; \qquad F(x)=0, \quad x<1$

Toute loi discrète peut être ramenée sous cette forme moyennant une numérotation appropriée de ses valeurs possibles. En utilisant la méthode d'inversion pour le cas non continu:

$$F^*(y)=\min\{x:F(x)\ge u)=j$$

ou bien $F(j-1)<u\le F(j), \qquad$ et $\quad F(j)=\sum_{k=1}^{j} p_k$

Dans ce cas, si $U\in U[0,1]$, alors

$$\mathbf{P}\{F(j-1)<U\le F(j)\}=\mathbf{P}(X=j)=p_j,\ j\ge 0$$

Algorithme 3.

1. Initialiser j=0
2. Tirer un nombre au hasard $U\in U[0,1]$,
3. Si $F(j)\le U$, alors poser j=j+1 et aller en 2,
4. Si F(j)>U, X=j.

Exemple 14. Etudions le cas d'une variable aléatoire X discrète à 6 valeurs possibles représentant les issues du lancer d'un dé que nous supposerons parfait. La variable aléatoire X représentant l'issue du dé a pour fonction de répartition F et densité f:

x	1	2	3	4	5	6
f(x)	1/6	1/6	1/6	1/6	1/6	1/6
F(x)	1/6	1/3	1/2	2/3	5/6	1

La règle d'inversion de l'algorithme 3 peut être réalisée automatiquement de la manière suivante:

Si $0\le U\le 1/6 \Rightarrow X=1$; Si $1/6\le U\le 1/3 \Rightarrow X=2$
Si $1/3\le U\le 1/2 \Rightarrow X=3$; Si $1/2\le U\le 2/3 \Rightarrow X=4$
Si $2/3\le U\le 5/6 \Rightarrow X=5$; Si $5/6\le U\le 1 \Rightarrow X=6$

Si on utilise la suite $\{U_i;\ i=\overline{1,10}\}$ des exemples précédents, alors on a la simulation suivante du jeu:

N° du lancer	1	2	3	4	5	6	7	8	9	10
Résultat	3	5	3	5	6	3	1	6	6	3

5.3.2.3. Méthode de composition (mélange de distributions).

Soit à générer une variable aléatoire X de densité : $f(x)=\sum_{i=1}^{n} p_i f_i(x)$ où $\{p_i\}$ est la densité de probabilité d'une variable discrète N à valeurs dans {1,2,...,n}, et telle que $p_i=P(N=i)$, $0\le p_i\le 1$ (i=1,2,...,n), $\sum_{i=1}^{n} p_i = 1$; $f_i(x)$ est la densité de probabilité d'une variable aléatoire X_i ($1\le i\le n$). La densité f(x) est appelée **mélang**e (ou composition) des densités $f_1(x),...,f_n(x)$, d'où le nom de la méthode.

Algorithme 5.

1. Tirer un nombre au hasard $U_1\in U[0,1]$.
2. Générer la valeur de N issue de la distribution $\{p_i\}$ à l'aide de l'algorithme 3, ou tout autre algorithme spécifique à cette loi.
   Poser N=j pour $\sum_{i=1}^{j-1} p_i < U_1 < \sum_{i=1}^{j} p_i$,
3. Tirer un autre nombre au hasard $U_2\in U[0,1]$.
4. Générer un nombre aléatoire X_j (si N=j) à partir de la distribution de densité $f_N(x)=f_j(x)$ à l'aide d'un algorithme approprié (inversion ou autre).
   $$X_N=X_j=F_j^{-1}(U_2)\text{ , où }F_j(x)=\int_{-\infty}^{x} f_j(y)dy$$
5. $X=X_j$.

Exemple 15. La loi exponentielle $f(x)=e^{-x}$ de paramètre $\lambda=1$ s'écrit sous la forme (3.3), avec $p_i=e^{-(i-1)}(1-e^{-1})$ (loi géométrique de paramètre $q=e^{-1}$).

$f_i(x)=e^{-[x-(i-1)]}/(1-e^{-1})$, si $x\in[i-1,i]$.

L'algorithme 5 s'écrit alors:

1. Tirer $U\in U[0,1]$,
2. Si $\sum_{i=1}^{j-1} e^{-(i-1)}(1-e^{-1}) < U \le \sum_{i=1}^{j} e^{-(i-1)}(1-e^{-1})$, N=j,
3. Tirer $U_2\in U[0,1]$,
4. $X=(j-1)-\text{Log}U_2 - \text{Log}(1-e^{-1})$ (en utilisant l'inversion).

On peut considérer de la même manière un mélange « continu » de densités. Soit à générer la variable aléatoire X de densité de mélange

$$f(x)=\int_{\Re} f_z(x)dG(z) \qquad (5.9)$$

où $G(z)=\mathbf{P}(Z\le z)$ est la fonction de répartition d'une certaine variable aléatoire Z, et $\{f_z(x)\}_{z\in\Re}$ une famille de densités de probabilité indexée par un paramètre réel z.

Algorithme 6 .

1. Tirer $U_1\in U[0,1]$.
2. Former $Z=G^{-1}(U_1)$.

3. Tirer $U_2 \in U[0,1]$.
4. Former $X=F_z^{-1}(U_2)$, $(F_z(x)=\int_{-\infty}^{x} f_z(y)dy$

Dans ce cas, X est bien issu de la densité (5.9).

5.3.2.4. Méthode du rejet (ou d'acceptation).

Soit à générer une variable aléatoire X de densité f(x) bornée, $f(x) \leq B$, $\forall x$ $(B<\infty)$. Si $f(x) \neq 0$ pour $x \in [a,b]$, on peut toujours faire le changement de variables:

$$0 \leq y=\frac{x-a}{b-a} \leq 1,$$

$$0 \leq f^*(y)=\frac{1}{B} f(a+(b-a)y] \leq 1$$

Générer (x,f(x)) revient donc à générer $(y,f^*(x))$.

Algorithme 7 [Von Neuman (1951)].

1. Tirer indépendamment deux nombres au hasard U_1, $U_2 \in U[0,1]$.
2. Si $U_2 \leq f^*(U_1)$, accepter le couple (U_1,U_2), aller en 4.
3. Si $U_2 > f^*(U_1)$, rejeter le couple (U_1,U_2), aller en 1.
4. $Y=U_1$.
5. $X=a+U_1(b-a)$.

Le taux de couples acceptés est proportionnel à $f^*(x)$, c'est-à-dire f(x) ici.

Exemple 16. Soit X une variable aléatoire de densité

$$f(x)=\frac{c}{b-a}\left(\frac{x-a}{b-a}\right)^2\left(1-\frac{x-a}{b-a}\right) \ , \ a \leq x \leq b$$

En utilisant le changement de variables (3.5), (3.6), on se ramène à une variable aléatoire Y de densité $f^*(y)=Cy^2(1-y)$ $(C=12)$.
Le maximum de $f^*(y)$ est atteint au point y=2/3, et sa valeur

$$f^*(\frac{2}{3})=C\left(\frac{2}{3}\right)^2 \frac{1}{3}=\frac{4C}{27}$$

L'algorithme 7 devient:

1. Tirer $U_1 \in U[0,1]$, $U_2 \in U[0,1]$.
2. Si $U_2 \leq \frac{27}{4C}[CU_1^2(1-U_1)]$, aller en 3.
   sinon rejeter le couple (U_1,U_2) et aller en 1.
3. $Y=U_1$.
4. $X=a+(b-a)Y$.

Une méthode plus générale consiste à considérer une variable aléatoire Z de densité g, plus simple que f. On commence par former Z de densité g. Les valeurs de Z sont acceptées (X=Z) avec une probabilité h(Z) dépendant de Z. En effet, dans ce cas,

$$\mathbf{P}(X\le x) = \mathbf{P}(X\le x/Z \text{ accepté}) = \frac{P(Z \le x \text{ et } Z \text{ accepté})}{P(Z \text{ accepté})} = \frac{\int_{-\infty}^{x} h(z)g(z)dz}{\int_{-\infty}^{\infty} h(z)g(z)dz}$$

En d'autres termes, le taux de valeurs acceptées est $gh/\int gh$ et X a une densité proportionnelle à g(x)h(x).

Pour générer X, on procède comme suit. Supposons qu'il existe B tel que $f/g \le B < \infty$. On pose $h=f/gB$, et d'après ce qui précéder, X a pour densité $f/B \int gh = f$. De plus,
$\mathbf{P}(Z \text{ accepté}) = \int gh = 1/B$.

Algorithme 8 [Von Neumann généralisé].

1. Générer Z de densité g.
2. Tirer un nombre au hasard $U \in U[0,1]$.
3. Si $U \le f(Z)/g(Z)B$, aller en 4.
   sinon aller en 2.
4. X=Z.

Notons que le test à l'étape 2 accepte la valeur de Z avec la probabilité requise. D'autre part, le nombre d'essais avant que Z ne soit accepté suit une loi géométrique de moyenne B, et l'algorithme est d'autant plus rapide que B est petit.

On peut simplifier les calculs en appliquant la méthode à une fonction f^* proportionnelle à F, si seulement on connaît une borne de f^*/g.

5.3.2.5. Opérations et simplifications.

Les algorithmes de générations de variables aléatoires utilisent souvent les opérations usuelles sur ces dernières. La densité de la somme X=Y+Z de deux variables aléatoires indépendantes Y et Z est le produit de convolution des densités marginales du couple (Y,Z):

$$f_X(x) = \int_{\Re} f_{(Y,Z)}(y; x-y)dy = \int_{\Re} f_{(Y,Z)}(x-z, z)dz \qquad (5.10)$$

où $f_{(Y,Z)}(.,.)$ est la densité conjointe de probabilité du couple (Y,Z). En vertu de l'indépendance de Y et Z, (5.10) s'écrit:

$$f_X(x) = \int_{\Re} f_Y(y) f_Z(x-y)dy = \int_{\Re} f_{(Y,Z)}(x-z) f_Z(z)dz$$

On peut donc générer Y et Z indépendamment de densités f_Y et f_Z respectivement, puis former X=Y+Z. Cette propriété est utilisée par exemple pour générer une loi d'Erlang. On peut de la même manière générer les variables aléatoires

$$X_{min}=\min(X_1, X_2, ..., X_n)$$
$$X_{max}=\min(X_1, X_2, ..., X_n)$$

de densités respectives:

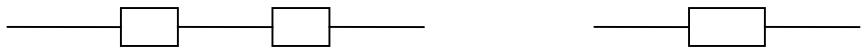
$$f_{\min}(x)=\sum_{i=1}^{n}\frac{fi(x)}{1-Fi(x)}\prod_{j=1}^{n}[1-Fi(x)]$$

$$f_{\max}(x)=\sum_{i=1}^{n}\frac{fi(x)}{Fi(x)}\prod_{j=1}^{n}Fi(x)$$

où $f_1(x),...,f_n(x)$ sont les densités de $X_1,...,X_n$ supposées indépendantes.

Exemple 17: (Fiabilité de systèmes série et parallèle).

1.Un système S a une configuration série si son fonctionnement nécessite le fonctionnement de tous ses éléments. Le diagramme fiabiliste d'un tel système est représenté sur le schéma ci-dessous:



La variable X_{min} de fonction de répartition $F_{min}(x)=1-\prod_{i=1}^{n}[1-F_i(x)]$, représente en Théorie de fiabilité, la durée de vie du système S.

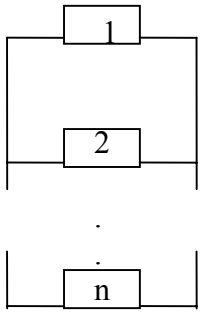
Algorithme 9.

1. Tirer indépendamment $U_1, U_2, ...,U_n \in U[0,1]$,
2. Calculer $X_1=F_1^{-1}(U_1)$, $X_2 = F_2^{-1}(U_2)$, …, $X_n = F_n^{-1}(U_n)$.
3. Poser $X=\min(X_1, X_2,...,X_n)$.

Par exemple, pour générer X de densité $f(x)=nx^{n-1}$,

1. Tirer indépendamment $U_1, U_2, ...,U_n \in U[0,1]$,
2. Poser $X=\min(U_1, U_2,...,U_n)$.

Un système P a une configuration parallèle si son fonctionnement nécessite le fonctionnement d'au moins un élément.



La variable X_{max} de fonction de répartition $F_{max}(x)=\prod_{i=1}^{n} F_i(x)$ représente la durée de vie du système P.

Algorithme 10.

1. Tirer indépendamment $U_1, U_2, ...,U_n \in U[0,1]$,
2. Calculer $X_1=F_1^{-1}(U_1)$, $X_2 = F_2^{-1}(U_2)$, …,$X_n = F_n^{-1}(U_n)$.
3. Poser $X=\max(X_1, X_2,...,X_n)$.

Pour générer $f(x)=n(1-x)^{n-1}$, $0\le x\le 1$, on pose $X=\max(U_1,...,U_n)$.

On utilise parfois les propriétés asymptotiques de la densité f(x). La loi normale par exemple est la loi limite d'une somme de variables aléatoires indépendantes. On peut donc générer une variable Z de loi normale N(0,1), en tirant d'abord n variables aléatoires X_1, X_2,...,X_n indépendantes et de même loi, pour n suffisamment grand. On forme ensuite la somme $X_1+X_2+...+X_n$ de loi $N(n\mu;n\sigma^2)$ ($\mu=E(X_i)$; $\sigma^2=Var(X_i)$).
et on pose enfin la somme

$$Z=\frac{X_1 + X_2 +\ldots+X_n - n\mu}{\sigma},$$

la variable centrée réduite qui sera asymptotiquement de loi N(0,1).

5.4.Génération de Vecteurs aléatoires.

Soit à générer un vecteur aléatoire $(X_1,X_2\ldots,X_n)$ à n dimensions, de densité $f(x_1,x_2,\ldots,x_n)= f_{(X_1,X_2,...,X_n)}(x_1,x_2,...,x_n)$.

5.4.1. Algorithme du conditionnement. En vertu du théorème des probabilités conditionnelles, on a :

$$f(x_1,x_2,\ldots,x_n)= f_1(x_1)\,.\, f_2(x_2/x_1)\,.\, f_3(x_3/x_1,x_2)\ldots. f_n(x_n/x_{11},...,x_{n-1}) \qquad (1)$$

où $f_i(x_i/x_{11},...,x_{i-1})$ est la densité (marginale) conditionnelle de la variable X_i sachant $X_1=x_1,\ldots,X_{i-1}=x_{i-1}$. La représentation (1) conduit à l'algorithme :

Algorithme 10. (du conditionnement).

1. Tirer $U_i\in U[0,1]$.
2. Générer X_i de densité $f_i(x_i/x_1,...,x_{i-1})$
3. Si $i<n$, aller en 1.
4. Ecrire $X_1,X_2\ldots,X_n$.

La méthode est similaire pour $n\ge 2$, mais les calculs se compliquent lorsque n croît.

5.4.2. Méthode du rejet. Elle évite l'inversion et ne nécessite que l'évaluation des valeurs de la densité.

Algorithme 11. (du rejet).

On suppose que f( ) est bornée, M=max f, $x_i \in [a_i,b_i]$, i=1,2,...,n.

1. Tirer $U_1, U_2, ...,U_n \in U[0,1]$.
2. Former $\xi_i = a_i+(b_i-a_i)U_i \in U[a_i,b_i]$, i=1,n.
3. Tirer $U_{n+1} \in U[0,1]$.
4. Former $\xi_{n+1} = MU_{n+1} \in U[0,M]$.
5. Si $f(\xi_1, ...,\xi_n) \le \xi_{n+1}$, on accepte les valeurs $\xi_1, \xi_2, ...,\xi_n$ comme composants du vecteur aléatoire $(X_1,..,X_n)$ de densité $f(x_1,...,x_n)$.

   Si $f(\xi_1,...,\xi_n) > \xi_{n+1}$, on rejette l'essai, aller en 1.

On peut utiliser la variante suivante parfois plus commode.

Soit $0 \le g(x_1,\ldots,x_n) \le g_1(x,\ldots,x_n)$, et

$$G=\int ...\int g(x_1,...,x_n)dx_1...dx_n \le G_1=\int ...\int g_1(x_1,...,x_n)dx_1...dx_n \le +\infty$$

Soit à générer le vecteur X de densité $f(x,\ldots,x_n)=g(x,\ldots,x_n)/G$.

Algorithme 11'. (du rejet).

1. Générer le vecteur X^0 de densité $g_1(x_1,\ldots,x_n)/G_1$
2. Tirer $U \in U[0,1]$.
3. Former $\zeta=U.g_1(X_1^0,\ldots,X^0_n)$.
4. Si $\zeta > .g(X_1^0,\ldots,X^0_n)$, aller en 1.
5. $X=X^0$.

Ce procédé reste malgré tout très lourd lorsque n augmente.

5.4.3. Méthodes spécifiques. Comme pour le cas n=1, on peut utiliser les propriétés statistiques de la distribution à générer.

5.4.3.1.Loi normale multidimensionnelle.

La densité de la loi normale s'écrit

$$f(x)=(2\pi)^{-n/2} |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(x-m)^T \Sigma^{-1}(x-m)\right\}$$

où m est le vecteur espérance mathématique et Σ la matrice des covariances.

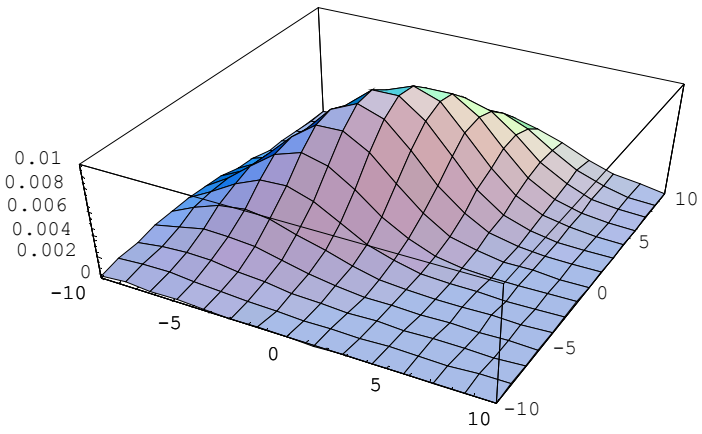


Figure . Densité de la loi Normale bidimensionnelle de moyenne et de matrice des covariances

$$K=\begin{pmatrix}16 & 12\\ 12 & 25\end{pmatrix}$$

On peut ramener le problème à la génération de n variables gaussiennes indépendantes N(0,1) à l'aide de transformations spécifiques (souvent linéaires).

Supposons qu'il existe une matrice S de dimension nxn telle que $\Sigma=SS^T$. Si Z_1, Z_2,...,$Z_n \in N(0,1)$, où $Z^T=(Z_1,...,Z_n)$ et les variables Z_1,...,Z_n sont indépendantes, $m \in \Re$, alors $E\{(X-m)(X-m)^T\}=SE\{ZZ^T\}S^T=SS^T=\Sigma$ et $X=m+SZ \in N(m,\Sigma)$. Une telle matrice S existe toujours. Elle peut être obtenue par exemple à l'aide de la méthode de Cholesky qui consiste à trouver une matrice L unique, triangulaire inférieure vérifiant $\Sigma=LL^T$. Le vecteur $X=m+LZ$ peut alors être formé plus rapidement. La décomposition de Cholesky est particulièrement intéressante si Σ^{-1} est telle que $LL^T=\Sigma^{-1}$. On pose alors $S=\left(L^{-1}\right)^T$, et on forme $X=m+Y$, où $L^TY=Z$. Ce système triangulaire se résout aisément.

La méthode du conditionnement consiste à générer un vecteur $Y \in N(0,\Sigma)$ en générant d'abord $_1$, puis Y_2 sachant Y_1 etc. Soit Σ_j la matrice bloc supérieur de Σ et $\sigma=(\Sigma_{ij})_{1\le i\le j-1}$. Alors, la distribution conditionnelle de Y_j sachant $W_j=(Y_1,...,Y_{j-1})^T$ suit une loi normale de moyenne $\sigma^T\Sigma_{j-1}^{-1}W_j$ et de matrice des covariances $\Sigma_{jj}-\sigma^T\Sigma_{j-1}^{-1}\sigma$ qui peuvent être calculées au préalable; Σ_j^{-1} peut être calculée à partir de Σ_{j-1}^{-1} sans passer par une inversion directe. Notons que les deux méthodes ci-dessus sont similaires en nombre d'opérations et donc en temps de calcul. Cependant, on préfère en général la décomposition de Cholesky.

5.4.3.2.Lois multidimensionnelles discrètes. On peut théoriquement utiliser les méthodes de simulation de lois discrètes puisque l'ensemble des valeurs est un ensemble discret. En pratique, on peut utiliser d'autres méthodes plus efficaces. Par exemple, pour la loi de (X,Y) sur {1,2,...,n}x{1,2,...,m}, on peut utiliser x comme

indice. On cherche d'abord x tel que $P(X\le x-1)\le U\le P(X\le x)=P_{xm}$. Ensuite, à partir de P_{x0}, on cherche y tel que $P_{x,y-1}\le U\le P_{xy}$.

Exercice : Donner un algorithme de simulation d'une loi normale de moyenne et de matrice des covariances $\Sigma=\begin{pmatrix}1 & 1 & 3\\ 1 & 2 & 4\\ 3 & 4 & 11\end{pmatrix}$; générer ensuite quelques observations à l'aide de l'algorithme obtenu.

Solution : Etape préalable : Diagonaliser Σ (trouver S telle que $\Sigma=SS^t$).

($s_{11}=\sqrt{\Sigma_{11}}=\sqrt{1}=1$; $s_{12}=0$;

5.5. Génération de processus aléatoires.

La génération directe de tels processus à partir des fonctions de répartition finies s'avèrerait trop complexe. Les propriétés des processus particuliers étudiés permettent de fournir des algorithmes intéressants et qui recouvrent un grand nombre de situations pratiques.

5.5.1. Chaîne de Markov à temps discret.

Soit une chaîne de Markov $\{X_n\}$ à espace des états finis $E=\{1,2,...,n\}$ (ou dénombrable), de matrice des transitions $P=||p_{ij}||_{mxm}$,

$$p_{ij}=P\{X_n=j/X_{n-1}=i\}\ge 0\ ,\ i=1,2,...,m;\qquad \sum_{j=1}^{m} p_i^0 = 1$$

La matrice est stochastique et chaque ligne représente une probabilité discrète pour i fixé qu'on peut générer comme au chapitre précédent.

Algorithme 12.

Partageons l'intervalle [0,1] de deux manières. L'une pour générer la distribution de l'état initial:

1) $[0,1]=\sum_{i=1}^{m}\Delta_i$, $\quad |\Delta_i^0|=p_i^0$, $i=1,2,...,m.$

et l'autre pour générer l'état courant disons j sachant i à l'étape précédente:

2) $[0,1]=\sum_{j=1}^{m}\Delta_{ij}$, $\quad |\Delta_{ij}|=p_{ij}$, pour chaque $i=1,2,...,m.$

(Σ désigne la réunion d'intervalles disjoints et $|\Delta|$ la mesure de l'intervalle Δ).

On détermine alors les réalisations successives de la chaîne de Markov $\{X_n\}$ à l'aide des formules récurrentes:

$$X_0=i \quad si \quad U_0\in\Delta_i\ ,$$
$$X_n=j \quad si \quad U_n\in\Delta_{X_{n-1,j}}\ ,\ n\ge 1$$

Le cas d'une chaîne dénombrable se traite de manière analogue.

Exemple. Soit une chaîne de Markov de matrice des transitions:

$$P=\begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/3 & 1/3 & 1/3 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \qquad p^{0^T} = (0{,}3;0{,}5;0{,}2)$$

Si l'état initial à l'étape n-1 est $X_{n-1}=1$, alors

Si $0<U\leq 1/2 \Rightarrow X_n=1$; Si $½<U\leq 3/4 \Rightarrow X_n=2$; Si $¾<U\leq 1 \Rightarrow X_n=3$

Si $X_{n-1}=2$, alors

Si $0<U\leq 1/3 \Rightarrow X_n=1$; Si $1/3<U\leq 2/3 \Rightarrow X_n=2$; Si $2/3<U\leq 1 \Rightarrow X_n=3$

Si $X_{n-1}=3$, alors

Si $0<U\leq 1/4 \Rightarrow X_n=1$; Si $¼<U\leq 12 \Rightarrow X_n=2$; Si $½<U\leq 1 \Rightarrow X_n=3$

Cette méthode présente des inconvénients dans la recherche pour chaque état. On peut utiliser les méthodes de recherche traitées au chapitre précédent. Une autre approche utilise la propriété suivante des chaînes de Markov à temps discret. Supposons que la chaîne se trouve à l'instant n à l'état $X_n=i$. Soit T_n le temps passé à cet état avant le prochain changement vers un état $j\neq i$. Alors T_n suit une loi géométrique de paramètre p_{ii} et X_{n+T_n} obéit à une loi discrète de distribution $\left\{P(i \to j) = P(X_{n+T_n} = j / X_n = i) = \frac{p_{ij}}{1-p_{ii}}, j \in E\right\}$. Cette méthode est un plus rapide.

5.5.2.Chaîne de Markov à temps continu.

Soit $\{X(t),t\geq 0\}$ un processus homogène de Markov à espace des états discret ou dénombrable $\{i\}$ et de matrice des probabilités des transitions

$$p_{ij}(t)=P\{ X(s+T)=j/X(s)=i\}$$

et de distribution initiale: $p_i^0= P\{X(0)=i, \qquad \sum_i p_i^0 =1.$

Si les dérivées existent, on définit les taux de transition:

$$\lambda_{ij}= \lim_{dt\to 0} \frac{p_{ij}(dt)}{dt}, \quad i\neq j, \qquad \lambda_i=\sum_{j\neq i}\lambda_{ij}$$

avec la condition:

$$p_{ij}(0)= \lim_{dt\to 0} p_{ij}(dt) = \begin{cases} 1 & , \text{ si } j = i \\ 0 & , \text{ si } j \neq i \end{cases}$$

La trajectoire d'un tel processus peut-être décrite de la manière suivante. Soit i_0 l'état initial déterminé par la distribution initiale $\{p_i^0\}$; la durée de séjour à cet état est une variable aléatoire τ_0 de loi exponentielle de paramètre λ_{i_0}.

$$P(\tau_0<x)= 1- e^{-\lambda i 0 x}, x\geq 0.$$

La probabilité pour que l'état suivant soit i_1 est $P\{X(\tau_0)=i_1/X(0)=i_0\}=\frac{\lambda i0 i1}{\lambda i0}=p_{i_0 i_1}$

Le processus séjournera à cet état un temps τ_1 exponentiellement distribué $P(\tau_1<x)=1-e^{-\lambda_{i_1} x}$ De nouveau, $P\{X(\tau_0+\tau_1)=i_2/X(\tau_0)=i_1\}=\frac{\lambda_{i_1 i_2}}{\lambda_{i_1}}=p_{i_1 i_2}$

et $P(\tau_2<x)=1-e^{-\lambda_{i_2} x}$, etc...

En d'autres termes, les états pris par le processus au cours des transitions successives forment une chaîne de Markov $\{X_n\}$, définie par $X_n=X(\tau_0+\tau_1+...+\tau_{n-1})$ de matrice des transitions $||p_{ij}||$, où $p_{ij}=\frac{\lambda_{ij}}{\lambda_i}$, et de distribution initiale $\{p_i^0\}$.

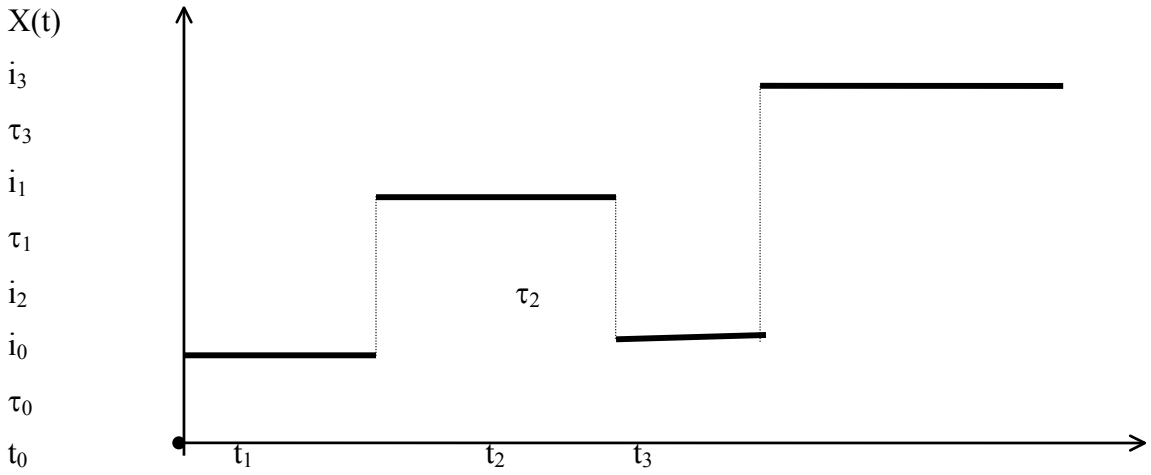


Figure . Evolution d'une chaîne de Markov à temps continu et espace des états discret.

La variable τ_i désigne la durée de séjour de la chaîne $\{X(t),t\geq 0\}$ à l'état i

$$F_i(x)=P(\tau_i<x)=1-e^{-\lambda_i x},\quad x\geq 0$$

La simulation du processus $\{X(t);t\geq 0\}$ consistera donc à simuler la chaîne de Markov incluse $\{X_n, n\geq 1\}$ à temps discret, où $X_n=X(t_n)$, $n=0,1,2,\ldots$.(c'est une suite d'instants où les observations forment une chaîne de Markov).

Algorithme 13.

1. Tirer une suite aléatoire $\{U_n\}$ de loi uniforme U[0,1].
2. Générer l'état initial i_0 selon la distribution $\{p_i^0\}$: $X_0=i$ si $U_0\in\Delta_i$
3. Tirer ensuite la durée de séjour à cet état:
   $\tau_0=-\frac{1}{\lambda}\mathrm{Log}\,U_1$
4. Les états suivants sont ainsi générés pour les termes pairs de la suite:
   $X_n=j$ si $U_{2n}\in\Delta_{X_{n-1},j}$ $n\geq 1$.

   $X_n=X(\tau_0+\tau_1+...+\tau_{n-1})$

5. Tirer les durées de séjour successives selon les lois exponentielles correspondantes pour les termes impairs:

$$\tau_n=-\frac{1}{\lambda_{X_n}}\mathrm{Log}\,U_{2n+1}\ ,\ n\geq 1$$

Le processus considéré est alors défini par la relation:

$X(t)=X_n$ pour $\tau_0+\tau_1+....+\tau_{n-1}\leq t<\tau_0+\tau_1+...+\tau_n$.

Cette méthode est souvent appelée par les praticiens, simulation événement par événement ou simulation à événements discrets. Dans le cas d'une chaîne de Markov à temps continu, il est possible d'obtenir toute l'information nécessaire à partir de la suite discrète $\{X_n\}$ sans avoir à connaître la suite $\{\tau_n\}$. En particulier, τ_n peut être supposée discrète.

5.5.3. Simulation de files d'attente:

Les questions de modélisation à l'aide de la théorie des files d'attente (Queueing Theory) ont été évoquées dans le chapitre 4, où l'on a pu constater leurs limites, les nôtres en fait. L'illustration de la simulation sur les problèmes d'attente est ainsi plus évocateur.

On distingue deux approches principales correspondant à la gestion du temps :

-Approche synchrone : qui correspond à une évolution fixe du temps.

-Approche asynchrone : évolution par incréments variables.

5.5.3.1. Méthode synchrone. Elle consiste à discrétiser le temps en intervalles de durées identiques h, [0,h] , [h,2h], [2h,3h],...., [(n-1)h, nh]. Par exemple, pour un processus de naissance et de mort de paramètre λ et μ (système M/M/1)

$$N_n=N_{n-1}+\begin{cases}+1 & \textit{avec une probabilité } \lambda h\\ 0 & \textit{avec une probabilité } 1-(\lambda+\mu)h\\ -1 & \textit{avec une probabilité } \mu h\end{cases}$$

En effet, si h est petit, la probabilité d'une naissance (saut +1) durant h.

$$P\{N(t+h)=1/N(t)=0\}=\lambda h+o(h)$$

De la même manière,

$$P\{N(t+h)=0/N(t)=0\}=1-\lambda h+o(h)$$
$$P\{N(t+h)=0/N(t)=1\}=\mu h+o(h)$$
$$P\{N(t+h)=0/N(t)=0\}=1-\mu h+o(h)$$

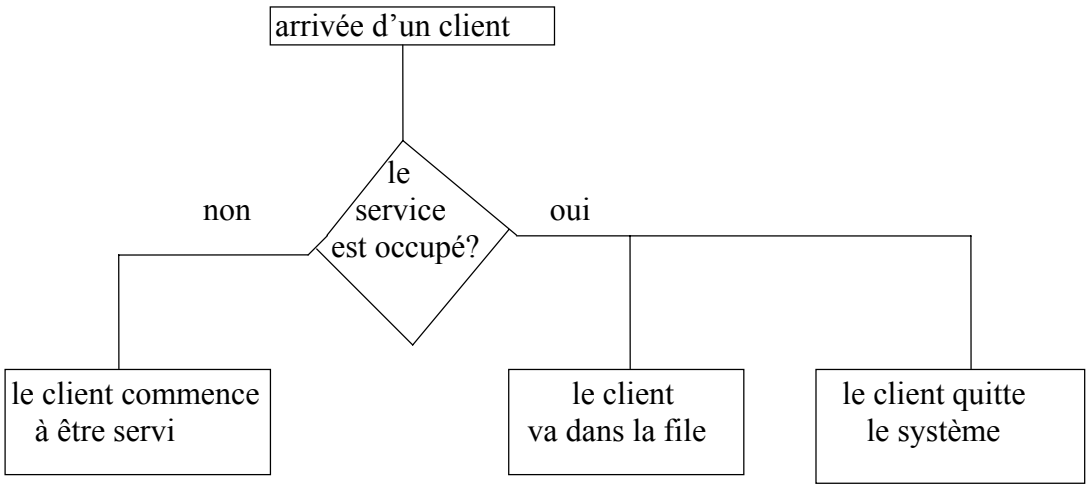
L'avantage de cette méthode est la simplicité de la réalisation du processus de simulation. L'inconvénient est que si h est trop grand, les résultats du processus de simulation ne reflètent pas de manière adéquate la trajectoire de l'état du système (un vecteur en général). Pour h trop petit, il faut un temps CPU important car pour chaque changement d'état, il faut un grand nombre de simulations.

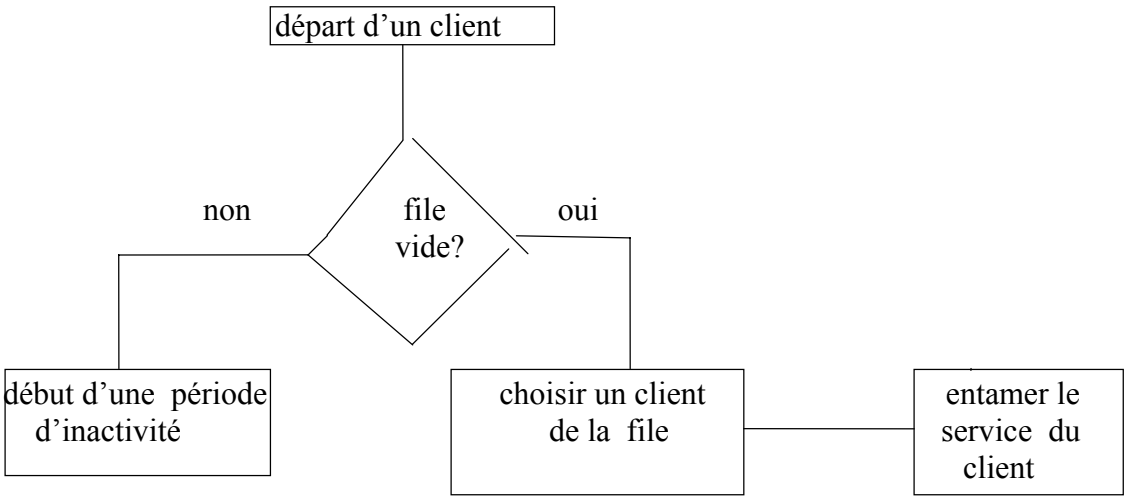
5.5.3.2. Méthode asynchrone (ou évènement par événement). Plusieurs visions existent :

-vision par évènements, que nous présentons plus loin ;

-visions par processus (un exemple sera donné dans l'exercice du chapitre 8).

C'est un exemple typique de la simulation à évènements discret. Elle consiste à suivre l'évolution continue du système évènement par évènement. Par exemple, dans le cas d'un système à un serveur, on peut avoir deux évènements possibles : arrivée d'un nouveau client ou départ d'un client qui été refusé ou dont le service s'est achevé





Ces événements peuvent être générés à partir des lois paramétriques du système:

-le flux des arrivées $t_1=\xi_1$, $t_2=\xi_1+\xi_2+....$, où t_n=date d'arrivée du nième client.

-la suite $\{\tau_n\}$ des durées de service des clients.

-éventuellement d'autres paramètres qu'il est nécessaire de prendre en compte (pannes, priorités, rappels,...

On effectue ensuite un traitement statistique des échantillons pour en déduire les principales mesures de performance: nombre de clients dans le système, durée d'attente (ou de séjour), taille de la file, probabilité de refus,....

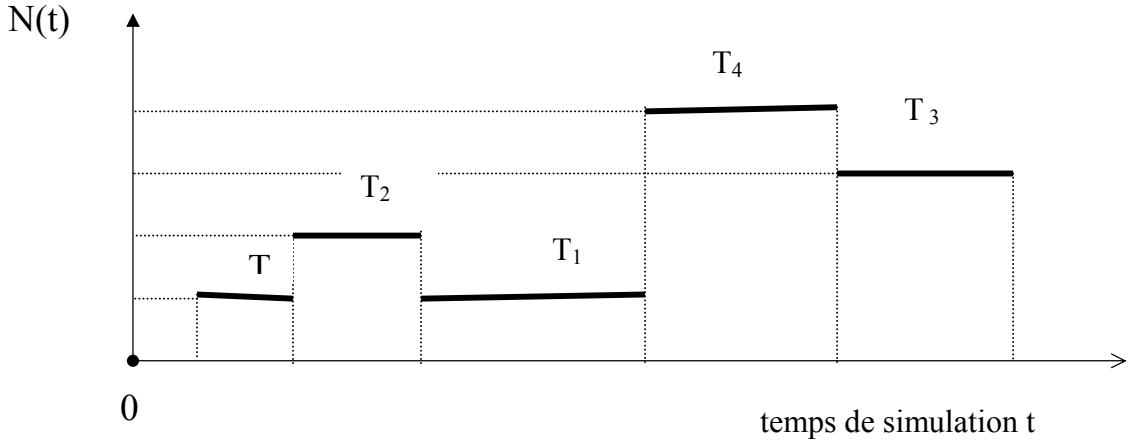


Figure Nombre de clients N(t) dans le système à l'instant t.

Soit T_i le temps cumulé dans l'intervalle [0,T] où le système contient exactement i clients. En général, on arrête les observations à un instant de changement d'état, aussi $\sum_{i=0}^{\infty} T_i = T$. Dans ce cas, le nombre moyen de clients N(t) dans le système peut être estimé à l'aide de la quantité

$$\hat{N}(T)=\hat{N}=\frac{1}{T}\sum_{i=0}^{\infty} iTi=\frac{1}{T}\int_0^T N(t)dt \qquad (\ )$$

Notons que $\int_0^T N(t)dt=\sum_{i=0}^{\infty} iT_i$ représente la surface totale comprise entre la courbe de N(t) et l'axe des abscisses. Sur la figure ? , la variable T_i est égale à la somme de tous les intervalles notés T_i. Si le régime stationnaire existe, (dans le cas où $r<1$), alors le nombre stationnaire moyen de clients dans le système $E(N)=\lim_{t\to\infty} N(t)$ existe et

$$\hat{N}=\frac{1}{T}\sum_{i=0}^{\infty} iTi=\frac{1}{T}\int_0^T N(t)dt \xrightarrow[T\to\infty]{} E(N)$$

La période de simulation T doit donc être suffisamment grande pour que l'estimateur observée $\hat{N}$ soit convergent (voir annexe B) et par conséquent proche de sa vraie valeur (stationnaire) E(N) . Pour les petites valeurs de T, l'estimation devient moins bonne car dépendant des conditions initiales.

Ces considérations restent valables pour les autres mesures de performance (temps d'attente, taille de la file,…). Une technique alternative (pour le régime stationnaire au moins) consiste

à utiliser les lois de conservation. Par exemple, le temps moyen d'attente est donné par la formule de Little $E(W)=\lambda/E(N)$.

Illustrons la méthode sur un exemple simple.

Exemple 1. Systèmes Markoviens M/M/1. .

On note λ le taux d'arrivées des clients et μ le taux de service. Soit X(t) le nombre de clients présents dans le système à l'instant t qui forme un processus homogène de Markov. On distingue deux types de sauts:

a) +1 correspond à l'arrivée d'un client, avec une probabilité égale à $\lambda/(\lambda+\mu)$;
b) -1 , correspond à une fin de service, avec une probabilité $\mu/(\lambda+\mu)$.

Algorithme 14.

0.Initialisation: $X_0=0$.

1. Tirer la date d'arrivée du premier client $t_0=\tau_0=-\frac{1}{\lambda}\mathrm{Log}\,U_0$, $X_1=1$.

2.Tirer $U_1\in U[0,1]$.

Si $U_1\in(0,\frac{\lambda}{\lambda+\mu})$, alors le prochain événement A est l'arrivée d'un client et $X_2=X_1+1$

Si, le prochain événement est un service (B) et $X_2=X_1-1$.

3. Tirer $U_2\in U[0,1]$ et tirer le temps de séjour à cet état X_1 avant de passer à X_2.

Si A se réalise $\tau_1=-\frac{1}{\lambda}\mathrm{Log}\,U_2$

Si B se réalise $\tau_2=-\frac{1}{\mu}\mathrm{Log}\,U_2$.

Cette méthode n'est malheureusement valable que lorsque le nombre de clients est markovien. Elle peut être utilisée pour simuler la chaîne de Markov incluse. Ici aussi, la difficulté réside dans notre aptitude à détecter des instants de régénérations.

5.6. Simulation à évènements discrets : Méthodologie générale.

Ainsi qu'il apparaît des exemples précédents, la simulation par évènements discrets désigne ainsi la modélisation d'un système réel tel qu'il évolue dans le temps. C'est une représentation où les descripteurs des grandeurs (paramètres, variables ou vecteurs d'état) caractérisant le système ne changent qu'en un nombre fini ou dénombrable de points isolés dans le temps. Le principe de la simulation consiste, une fois le modèle décrit à faire évoluer l'état du système , en mettant à jour la description d'état liée à chaque événement. Un événement pour ce type de simulation est un bloc de données : une date d'occurrence, une identité, etc…c'est par le biais de ces données que l'on peut décrire de la manière la plus détaillée possible toute modification du vecteur d'état. Le **simulateur** passe ainsi d'un événement à un autre, en négligeant ce qui se passe entre ces évènements. Par contre, il faut tenir compte des évènements engendrés par chaque évènements et mettre à jour la description du système. Dans l'exemple de files d'attente, ces évènements « potentiellement futurs » sont l'arrivée du

prochain client ou le départ du client en cours de service. En termes de RdP, ce sont les transitions franchissables.

Lors de la simulation de systèmes (d'attente ou autre) plus complexes, le nombre d'évènements futurs peut être très important. Une partie du travail consiste donc à « optimiser » la gestion des données correspondantes. C'est en partie le rôle de l'**échéancier** (souvent considéré comme le cœur de la simulation). L'échéancier est une liste de données (event list), souvent des évènements ordonnés chronologiquement selon l'heure (l'horloge) de la simulation. Son rôle est de stocker les données, les classer et les extraire aux instants voulus.

Ainsi, le temps est géré par l'échéancier et par l'horloge centrale. L'**horloge de simulation** est une variable qui enregistre le temps de simulation actuel qui est avancé dès qu'un événement se produit. La tâche de l'échéancier va donc consister dans un premier temps à insérer dans la table de la liste des évènements un événement futur nouvellement créé. Il procèdera ensuite à l'extraction de l'événement le plus imminent i.e. de date d'occurrence (durée de temporisation) minimale et qui se trouve en tête de liste. Notons qu'au cours de ces opérations, on n'intervient en aucune façon sur les autres évènements futurs de cette liste.

Mise en œuvre de la simulation.

Initialement, l'horloge est mise à zéro et les évènements futurs qui peuvent se produire sont insérés dans la liste selon l'ordre chronologique. L'événement suivant le plus proche est extrait de la liste pour exécution, et l'horloge de simulation est avancée de la durée de temporisation de cet événement. L'état du système est mis à jour à la fin de cette durée. Les évènements nouvellement créés sont insérés dans la liste et certains en sont extraits. Ce processus est répété jusqu'à ce que l'une des deux conditions suivantes soit vérifiée : (a) il n' y a plus d'évènements qui puisse se produire (plus de transition franchissable) ; (b) le temps de simulation a expiré.

Structures de données. Il apparaît que l'échéancier assure la gestion d'une table ordonnée en perpétuelle évolution. La structure dynamique de cette table va donc déterminer en partie les performances du simulateur. Par analogie aux algorithmes de tri, différents modes de classement peuvent être proposés. En particulier,

(a) structure linéaire : c'est une table totalement ordonnée. On insère un nouvel élément en queue (ou au début) et on réarrange la table. L'extraction se fait en tête. On peut alternativement utiliser un pointeur pour l'insertion au milieu.

(b) Structure linéaire indexée : utilise plusieurs pointeurs intermédiaires.

(c) Arbre binaire : c'est une structure pyramidale à Log(n) niveaux, où n est le nombre d'éléments de la table. Les évènements du niveaux k sont postérieurs à ceux du niveau k-1.

Génération des évènements : Ils sont générés à l'aide de nombre aléatoires selon les techniques de génération décrites au début de ce chapitre. Les performances du simulateur vont dépendre également des techniques utilisées et discutées préalablement. Mais la programmation est facilitée par l'existence de générateurs standards à la plupart des langages de simulation (voir chapitre 8).

Résultats de la simulation. La dernière mise à jour des variables d'états fournit en général une estimation statistique des grandeurs mesurées. Dans les exemples de files d'attente, on

s'intéressait au temps **moyen** d'attente, à la taille **moyenne** de la file ou à la période **moyenne** d'inactivité.

L'estimateur de la moyenne de n observations $X_1, X_2, ..., X_n$ (par exemple du nombre de clients dans le système ou du temps d'attente) est (voir annexe B) : $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$.

Précision de la simulation. Diverses questions se posent à ce niveau. D'abord, comme pour toute expérience, il s'agit d'estimer la précision de ces estimations.

(i) La précision peut d'abord être estimée en fournissant un intervalle de confiance de l'estimation. On montre en statistique (voir annexe B) qu'un intervalle de confiance de niveau 1-α (α petit, par exemple 5%) pour la grandeur moyenne E(X) est :

$$\overline{X} - u_{\alpha/2}\sigma/\sqrt{n} \prec X \prec \overline{X} + u_{\alpha/2}\ \sigma/\sqrt{n}$$

où $\Phi(u_\alpha)=1-\alpha$, $\Phi(.)$ étant la fonction de répartition de la loi normale standard N(0,1) de moyenne nulle et de variance unité. Lorsque la variance s^2 est inconnue, on peut l'estimer à l'aide de la variance empirique $S^{*2} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2$. La précision est d'autant plus grande que n (la taille de l'échantillon des observations) est grande. La question qui se pose naturellement est de savoir à quel moment il faut s'arrêter de générer des observations. La procédure suivante permet de déterminer une valeur de n fournissant une précision fixée à l'avance.

(a) choisir une valeur satisfaisante de la précision δ.

(b) Générer un nombre k≥30 valeurs (c'est la taille minimale d'échantillon admise dans les applications), où k est tel que $S^*/\sqrt{k} < \delta$.

(ii) On peut conduire les expériences de simulation en améliorant cette précision. En vertu de l'intervalle de confiance ci-dessus, l'erreur quadratique moyenne entre la vraie valeur $E(X)$ et son estimation $\overline{X}$ s'écrit :

$$E(\overline{X} - E(X))^2 = Var(\overline{X}) = \frac{Var(X)}{n} = \frac{\sigma^2}{n}$$

Cette erreur peut être diminuée en augmentant n (ce que nous savons déjà), mais aussi en diminuant la variance s^2. Les techniques de réduction de la variance sont discutées au chapitre suivant.

Le graphisme. La plupart des logiciels et langages de programmation sont dotés actuellement d'éditeur graphique, ce qui permet d'assurer une meilleure convivialité dans les programmes de simulation. Il est difficile de donner des recettes car cela dépend des bibliothèques utilisées, du système d'exploitation …. Il faut se référer au menu d'aide de l'éditeur utilisé. Si on utilise C++, on peut par exemple

(i) création d'un projet (sous Window dans cet exemple).

(ii) commencer par écrire une classe **Fenêtre** qui est la fenêtre graphique et qui dispose de méthodes réservées permettant de gérer le clavier , la souris, etc…Elle permettra de tracer des courbes. Utiliser pour cela la classe CframeWind de la bibliothèque MFC (Microsoft Fundation Classe).

(iii) écrire ensuite une classe **« Application** » permettant la gestion de la fenêtre ;

(iv) enfin écrire un pseudo-programme principal.

5.7.Travaux dirigés.

Exercice 1: Soient deux variables aléatoires U et V indépendantes. La variable U suit une loi exponentielle de paramètre $\lambda=1$; la variable V suit une loi normale de moyenne N(0,1), de moyenne nulle et variance égale à1

1. Ecrire la densité unidimensionnelle du processus aléatoire $X(t)=U+Vt$ pour $t>0$.
2. Calculer les fonctions moyenne, variance, d'auto-covariance et d'auto-corrélation de ce processus.
3. En déduire si le processus est stationnaire : au sens large ? au sens strict ?
4. Indiquer une manière de générer les valeurs du processus aux instants $t_1=1$, $t_2=2$ en utilisant la suite de nombres aléatoires suivante :

$U_1=0.8134$, $U_2=0.9754$, $U_3=0.5423$, $U_4=0.8643$, $U_5=0.8214$, $U_6=0.6238$, $U_2=0.4748$, $U_3=0.8522$, $U_4=0.642$, $U_5=0.8214$.

Exercice 2: Soient deux variables aléatoires U et V de densités $f_U(x)=\lambda e^{-\lambda X}$, et $f_V(x)=\mu e^{-\mu x}$ ($x\geq 0$), indépendantes.

5. Ecrire la densité unidimensionnelle du processus aléatoire $X(t)=U+Vt$ pour $t>0$.
6. Calculer les fonctions moyenne, variance, d'autocovariance et d'autocorrélation de ce processus.
7. En déduire si le processus est stationnaire : au sens large ? au sens strict ?
8. Indiquer une manière de générer les valeurs du processus aux instants $t_1=1$, $t_2=2$ en utilisant la suite de nombres aléatoires suivante :

$U_1=0.9234$, $U_2=0.6547$, $U_3=0.4532$, $U_4=0.6754$, $U_5=0.3214$, $U_6=0.8238$, $U_2=0.9748$, $U_3=0.6562$, $U_4=0.8782$, $U_5=0.6214$.

Exercice 3: Générer 3 observations de la loi exponentielle de paramètre 1 (avec une probabilité 0.25) et de paramètre 2 avec la probabilité complémentaire.

5.3.2.6.Travaux dirigés.

Exercice 10 : (suite de l'exercice 9) : nombres aléatoires java).

L'instruction « nextGaussian » permet de générer un nombre aléatoire issu de la loi Normale standard N(0,1) de moyenne nulle et de variance unité.

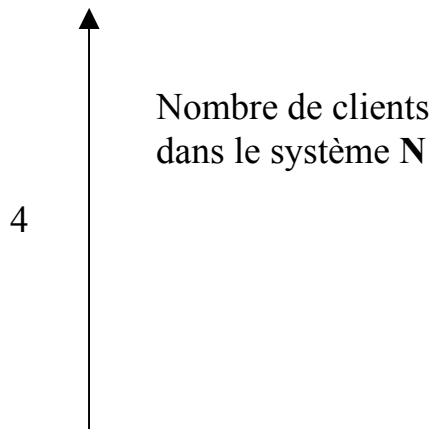
```
double r=generator.nextGaussian( ) ;
```

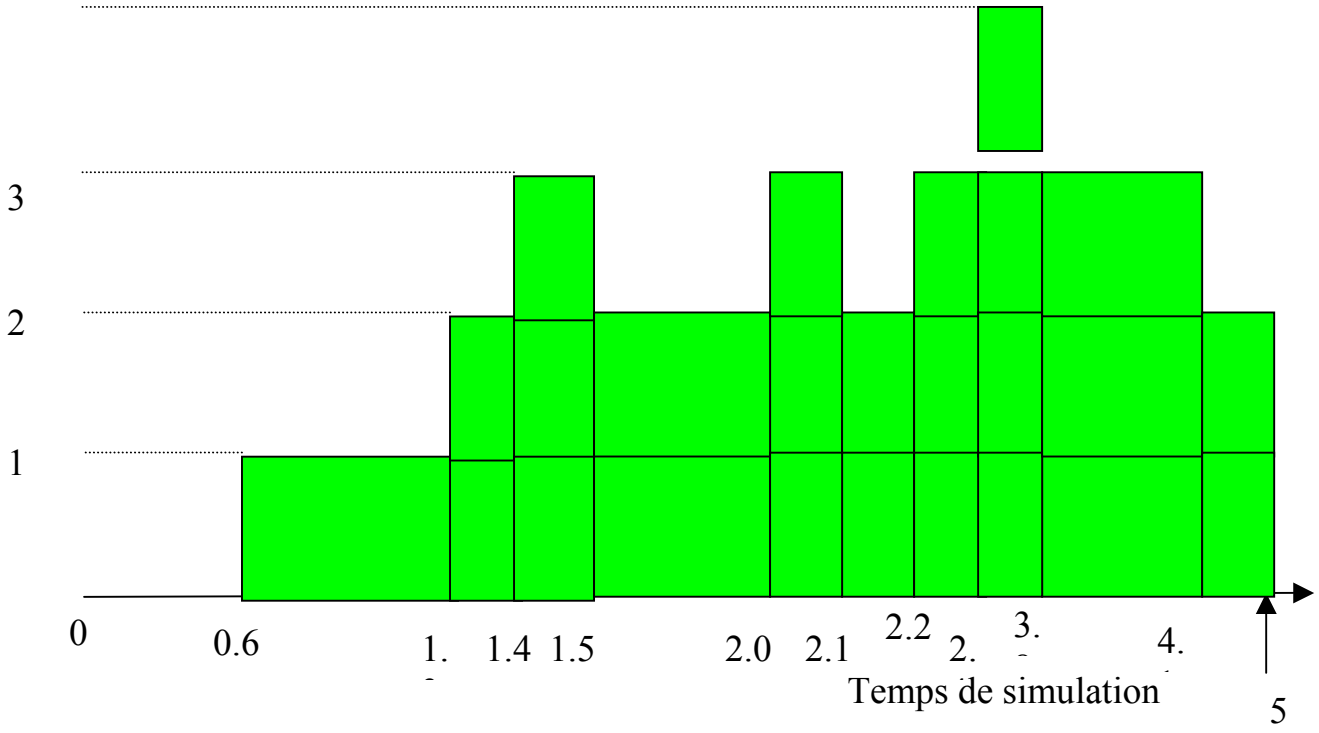
Concevez une Applet java permettant de tester l'uniformité et l'indépendance des suites aléatoires ainsi générées. Voir dans un premier temps l'Applet java : Uniform Applet sur le site http://ubmail.ubalt.edu/~harsham/simulation/sim.htm

Exercice 4. Le processus X(t) est un processus de Poisson représentant le nombre de pannes d'un ordinateur, de paramètre 0,002 pannes par heure. Montrer comment évaluer par simulation la probabilité qu'au cours d'une période de 200 heures il y ait un nombre pair de pannes et la probabilité pour qu'il n'y ait pas de pannes durant 100 heures.

Exercice 5.Simuler le comportement des 5 premiers clients au niveau d'une file d'attente FIFO : le processus des arrivées est Poissonien de paramètre 1 arrivée/seconde et le temps de service obéit à une loi d'Exponentielle de paramètre μ=1.1 services/seconde.

Solution :





t=horloge de simulation.
A l'aide de Random[], on génère la suite suivante :

U_1=0.526599 ; U_1=0.502976 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 ; U_1=0.526599 .....

- A l'instant initial l'horloge est mise à zéro
- t=0
- Le compteur du nombre de clients dans le système est aussi initialisé
- N=0
- Générer l'instant d'arrivée du prochain client N°1. Pour cela, il suffit de générer le temps d'inter arrivée ξ_1=-Log U_1=-Log(0.526599)=0.641316 à l'aide du premier nombre aléatoire.
- Actualiser l'horloge t=0+0.641316=0.641316 et le compteur N=0+1=1.
- Générer l'instant de la prochaine arrivée N°2
  t :=t+ξ_2=t-Log(U_2)=0.641316-Log(0.502976)=0.641316+0.687213=1.328529
  et l'instant de la prochaine fin de service (du client N°1):
  t :=t+S_1=t-Log(U_3)=0.641316-Log(0.6611167)=0.641316+0.858731=1.500647
- Le prochain événement est une arrivée N°2 à l'instant t=1.328529 et N :=N+1=2
- Générer la prochaine arrivée N°3
  t :=t+ξ_3=t-Log(U_4)=1.328529-Log(0.89275)=1.328529+0.113449=1.441978
- Le prochain événement est encore une arrivée à l'instant t=1.441978 et N=N+1=3
- L'arrivée suivante N°4:
  t :=t+ξ_4=t-Log(U_5)=1.441978-Log(0.566786)=1.441978+0.567773=2.0097510
- Par conséquent le prochain événement est une fin de service (N°1) à l'instant t=1.500647 et N :=N-1=2
- La prochaine fin de service (du client N°2 qui vient d'accéder au serveur) est à
  t :=t+S_2=t-Log(U_6)=1.500647-Log(0.528431)=1.500647+0.683355=2.184002

- Le prochain événement est donc une arrivée N°4 à t=2.0097510
  Le compteur est mis à jour et N :=N+1=3
- Arrivée N°5 :
  t :=t+ξ_5=t-Log(U_7)=2.0097510-Log(0.814406)=2.0097510+0.205296=2.215046
- Le prochain événement est une fin de service à t=2.184002 ; N :=N-1=2
- Prochaine fin de service
  t :=t+S_3=t-Log(U_8)=2.184002-Log(0.837307)=2.184002+1.65081=3.834812
- Le prochain événement est une arrivée à t=2.215046
  mise à jour du compteur N :=N+1=3
- Prochaine arrivée N°6
  t :=t+ξ_6=t-Log(U_9)=2.215046-Log(0.788154)=2.215046+0.238062=2.453108
- Le prochain événement est une arrivée à t=2.453108 ; N :=N+1=4

Etc……

L'exécution de cette simulation peut être représentée dans le tableau suivant.

Les deux premières colonnes représentent la mise à jour de l'horloge et l'état du système (nombre de clients dans le système).
La troisième colonne est l'échéancier (event List) qui contient les évènements futurs à l'instant marqué par l'horloge t. En observation, on peut mentionner dans cet exemple, l'événement le plus proche, et enfin la dernière colonne comporte les mises à jour des statistiques des mesures de performance que nous souhaitons évaluer.

Les évènements futurs sont notés sous la forme (événement, date d'occurrence) avec les codes suivants : A pour arrivée, D pour départ et F pour fin de simulation. Pour des raisons de commodité, nous reportons les dates à deux chiffres après la virgule. Par exemple,

(A ; 0.64) : événement arrivée à l'instant t=0.64.
(F ; 3) : fin de simulation programmée au temps t=5.0

Nous mesurons deux quantités : O : période moyenne d'occupation du serveur
N : nombre moyen de clients
En observation, nous indiquons le plus proche événement

				Statistiques cumulées	
t	N	Echéancier	Observation	O	N
0	0	(A ;0.64) ; (F,5)	(A ;0.64)	0	0
0.64	1	(A ;1.32) ;(D;1.50) ; (F,5)	(A ;1.32)	0.64	1
1.32	2	(A ;1.44) ;(D;1.50) ; (F,5)	(A ;1.44)	1.32	2
1.44	3	(A ;2.00) ;(D;1.50) ; (F,5)	(D;1.50)	1.50	3
1.50	2	(A ;2.00) ;(D;2.18) ; (F,5)	(A;2.00)	2.00	2
2.00	3	(A ;2.21) ;(D;2.18) ; (F,5)	(D;2.18)	2.18	3
2.18	2	(A ;2.21) ;(D;3.83) ; (F,5)	(A ;2.21)	2.21	2
2.21	3	(A ;2.45) ;(D;3.83) ; (F,5)	(A ;2.45)	2.45	3
2.45	4	(A ;5.04) ;(D;3.83) ; (F,5)	(D;3.83)	3.83	4
3.83	3	(A ;5.04) ;(D;4.14) ; (F,5)	(D;4.14)	4.14	3
4.14	2	(A ;5.04) ;(D;6.24) ; (F,5)	(F;5)	5	2

Estimation du nombre moyen de clients dans le système:

Rappelons que $\hat{N}=\frac{1}{T}\sum_{i=0}^{\infty} iTi=\frac{1}{T}\int_0^T L(t)dt$. Ici ,

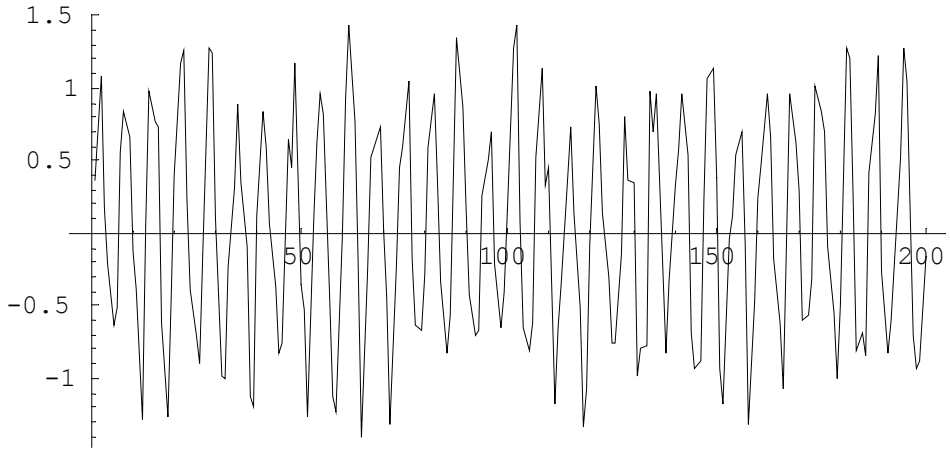
$$\hat{N}=\frac{1}{5}\times\{0\times(0.64-0)+1\times(1.3-0.64)+2\times(1.4-1.3)+2\times(2.0-1.5)+2\times(2.21-2.18)+2\times(5-4.14)$$
$$+3\times(1.5-1.4)+3\times(2.18-2.0)+3\times(2.4-2.2)+3\times(4.1-3.8)+4\times(3.8-2.4)=\frac{11.58}{5}\approx 3.044\}\ \text{clients.}$$

Exercice 6. Simuler le comportement des 3 premiers clients au niveau d'une file d'attente FIFO : le processus des arrivées est de paramètre 1 arrivée/seconde et le temps de service obéit à une loi d'Erlang (Gamma) d'ordre 2 de paramètre $\lambda=2$.

Exercice 7: Un signal périodique $X(t)=\frac{302\pi t}{200}+\varepsilon(t)$, où $\varepsilon(t)=U-1/2$ est un bruit aléatoire avec U uniforme sur [0,1]. Le petit programme suivant calcule 200 valeurs du processus et les représente dans le plan.

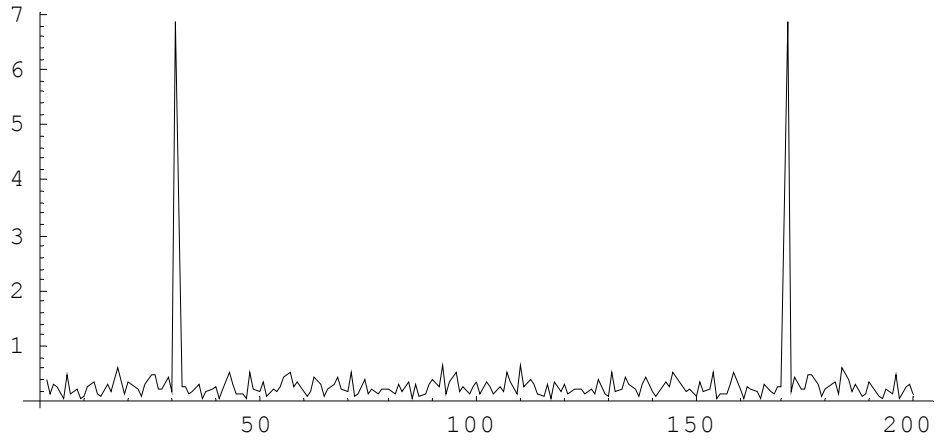
```
data = Table[ N[Sin[30 2 Pi n/200] + (Random[ ] - 1/2)],
        {n, 200} ] ;

ListPlot[ data, PlotJoined -> True ]
```



La transformée de Fourier montre un pic à 30+1 et un pic symétrique 201-30, ce qui suggère une composante de fréquence du signal original au voisinage de 30/200.

```
ListPlot[ Abs[Fourier[data]], PlotJoined -> True,
        PlotRange -> All ]
```



Exercice 7 : On cherche à simuler des observations d'une variable aléatoire prenant les valeurs 1,2,3,4 avec les probabilités
p_1=P(X=1)=0.4 ; p_2=P(X=2)=0.4 ; p_3=P(X=3)=0.1 ; p_4=P(X=4)=0.1

1.Ecrire l'algorithme de génération d'une telle variable par la méthode d'inversion.
2.Ecrire l'algorithme d'une telle simulation en utilisant la méthode du rejet.

3.Générer trois observations avec chacune des deux méthodes.

4.Comparer les deux algorithmes (du point de vue du nombre moyen d'itérations).

Exercice. Ecrire un programme de simulation d'une file d'attente M/M/1/1 avec refus.

Indication : Le flux des arrivées est poissonnien de taux λ; la durée de service exponentielle de paramètre μ; le client qui trouve le serveur occupé essuie un refus de service. L'instant initial t_0=0 sera assimilé à l'arrivée du premier client. On suppose qu'à cet instant, le serveur est libre.

Soit T la période de simulation. On notera par t la date de libération du canal. Le premier client qui arrive dans le système commence immédiatement à être pris en charge par le serveur et $t=t_0+\tau$ où τ est une variable de loi exponentielle de paramètre μ. Le compteur m des clients servis augmente à cet instant d'une unité, et on s'intéresse au client suivant.

Les dates d'occurrence des clients $t_{k+1}=t_k+\xi_{k+1}$, où ξ_{k+1} est la durée entre deux arrivées successives de loi exponentielle de paramètre λ.

A la k+1ème étape, k clients ont été examinés et on s'intéresse au k+1ème client. On calcule t_{k+1} à l'aide de la formule ci-dessus. Si à cet instant $t \le t_{k+1}$, on vérifie la condition: $t_{k+1}<T$. Si cette condition est remplie, le compteur k des arrivées augmente d'une unité. Si par contre la condition n'est pas remplie, on arrête la simulation. Si $t>t_{k+1}$, on vérifie de nouveau notre condition. Si elle est vérifiée, on modifie t_k et le compteur k augmente d'une unité. On effectue ainsi N simulations de durée T chacune. L'estimation du nombre de clients servis est

$$\mathbf{E}(m)=\frac{1}{N}\sum_{j=1}^{N} m_j$$

où m_j est le nombre de clients servis au cours de la jème simulation. Le nombre moyen de clients perdus est **E**(k)-**E**(m).

Exercice : Ecrire un programme de simulation d'une file d'attente M/M/1avec rappels. Les arrivées sont poissoniennes de taux λ et la loi de service exponentielle de paramètre μ. Le client qui trouve le serveur libre est immédiatement pris en charge. Dans le cas contraire, il rappelle à des instants aléatoires (on dit que le client entre en orbite). Le taux de rappel de chaque client est constant de taux ν .Comparer les résultats de vos simulations avec les formules exactes. (On donne au chapitre 8 une résolution à l'aide du langage de simulation MOSEL).

Indications : Donnons à des fins de comparaisons les résultats exacts connus concernant le système M/M/1 avec rappels. Notons S(t)=0 si le serveur est libre, S(t)=1 si le serveur est occupé. R(t)= nombre de clients en orbite (en instance de rappel) à l'instant t. Le régime stationnaire existe si r=λ/μ<1, dans ce cas les probabilités stationnaires d'état

$p_{in}=\lim_{t\to\infty} P(S(t)=i,R(t)=n)$ existent et sont données par les formules :

$$p_{\grave{a}n}=\frac{\rho^n}{n!\nu^n}\prod_{i=\grave{a}}^{n-1}(\lambda+i\nu).(1-\rho)^{\frac{\lambda}{\nu}+1}$$

$$p_{1n}=\frac{\rho^{n+1}}{n!\nu^n}\prod_{i=1}^{n}(\lambda+i\nu).(1-\rho)^{\frac{\lambda}{\nu}+1}$$

De plus, $E(R)=\frac{\rho(\lambda+\rho\nu)}{(1-\rho)\nu}$; et $E(N)=\rho+E(R)=\frac{\rho(\lambda+\nu)}{(1-\rho)\nu}$.

Exercice 8: Evaluer par la méthode de MonteCarlo le volume de la sphère unité en 3D

$$x^2+y^2+z^2<1\ ,\ (x,y,z)\in[-1,+1]\times[-1,+1]\times[-1,+1]$$

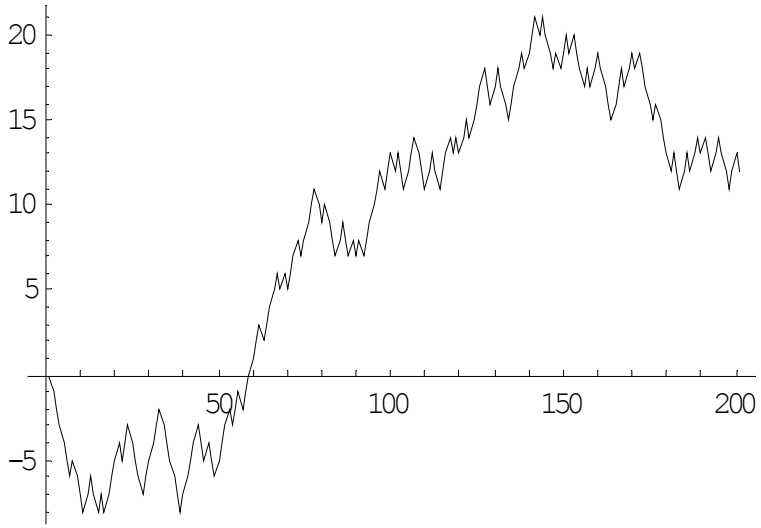
Solution :

```
NIntegrate[If[x^2 + y^2 + z^2 < 1, 1, 0], {x, -1, 1},
   {y, -1, 1}, {z, -1, 1}, MaxPoints->10000]
```

V=4.18106 et la valeur exacte est $V_{EX}=\frac{4\pi}{3}\approx 4{,}18879$

Exercice 9: Simuler une marche aléatoire de pas +1 et –1 ;S=S+$(-1)^U$

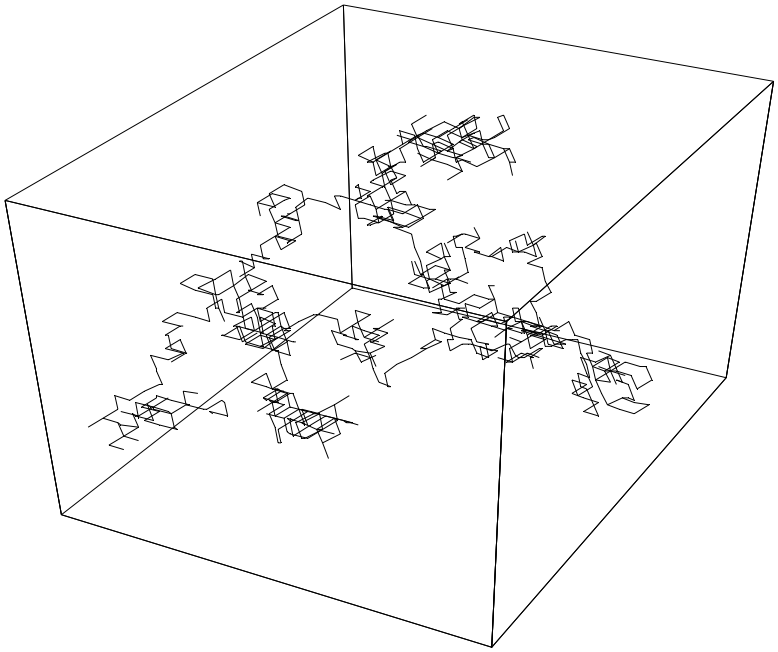
```
RandomWalk@n_D := NestList@H#+H-1L^Random@IntegerDL &,
  0, nD
ListPlot@RandomWalk@200D, PlotJoined→ TrueD
```



Marche aléatoire en 3D

```
RandomWalk@n_, d_D :=
 NestList@H# + H-1L^Table@Random@IntegerD, 8d<DL &,
  Table@0, 8d<D, nD

Show@Graphics3D@Line@RandomWalk@1000, 3DDDD
```



5.5.5.Sites utiles.

1. Simulation et modélisations discrètes ;
   http://ina.eivd.ch/ina/Collaborateurs/cez/Mod im/modsim.htmhttp

2. http://rfv.insa-lyon.fr/~jolion/STAT/node45.html