Techniques de la Physique Expérimentale

2eme partie :

Simulation de Données par la Méthode de Monte-Carlo

G.Wilquet

16/10/2006

1. Introduction

La méthode de Monte-Carlo est utilisée pour analyser les conséquences d'un processus aléatoire trop complexe pour être résolu analytiquement. La méthode est applicable lorsque le processus global peut être décomposé en séquences de processus élémentaires, corrélés ou non, et décrits chacun par une ou plusieurs variables de fonction de densité de probabilité connue. Elle consiste à simuler le processus par la génération successive de valeurs pour les variables distribuées aléatoirement suivant leurs fonctions de densité de probabilités respectives. Un ensemble de valeurs numériques spécifiant chacune des variables décrit complètement une configuration possible du processus étudié. Le nombre de configurations à simuler dépendra de la précision statistique souhaitée, et, pour une précision donnée, de la nature du processus au travers des variances des fonctions de densité de probabilité.

Les champs d'application habituels sont :

i) L'obtention de distributions de variables observables, de la valeur de paramètres mesurables, au départ de modèles théoriques complexes,

ii) La simulation des phénomènes aléatoires qui vont stimuler les signaux délivrés par l'appareillage expérimental,

iii) La simulation de la réponse de l'appareillage à ces stimulations,

iv) La conjonction de i), ii) et iii) permet de concevoir les caractéristiques d'un dispositif susceptible de réaliser une mesure expérimentale avec la résolution souhaitée,

v) La conjonction de i) ii) et iii) permet d'appliquer aux prédictions d'un modèle les distorsions tenant compte des résolutions expérimentales, des biais d'acceptance, des contaminations, pour comparaison avec les résultats expérimentaux; elle permet aussi de mesurer l'implication, sur les résultats, des imprécisions sur la connaissance des modèles.

vi) La méthode de Monte-Carlo est également une technique d'intégration numérique singulièrement efficace quand la dimensionnalité de l'intégrale est grande mais très inefficace quand elle est petite. Cet aspect ne sera abordé qu'incidemment dans la suite.

2. Nombres pseudo-aléatoires

Les nombres pseudo-aléatoires sont générés suivant un algorithme mathématique défini. Ils sont donc parfaitement reproductibles et pas du tout aléatoires au sens strict. L'algorithme doit les rendre indiscernables de nombres aléatoires réels. Il doit être impossible de déterminer a posteriori s'ils résultent d'un phénomène naturel ou d'un algorithme mathématique ayant même FDP.

Les générateurs de nombres aléatoires génèrent habituellement des nombres suivant une distribution uniforme entre 0 et 1. Pour générer un nombre aléatoire x uniformément distribué sur $[a,b]$, il suffit de générer ξ sur $[0, 1]$ et de faire $x = a + (b - a)\,\xi$..

L'obtention de nombres aléatoires suivant une fonction de probabilité non uniforme à partir de nombres aléatoires générés suivant une distribution uniforme sera discutée plus loin.

Les Qualités d'un générateur de nombres aléatoires uniformes :

Uniformité et stochasticité :

Pour être correcte, la distribution doit être uniforme et aléatoire. Les tests d'uniformité et du caractère aléatoire de la génération seront discutés plus loin. On notera que le caractère aléatoire de la génération est essentiel. Il assure l'absence de corrélations entre les processus élémentaires simulés successivement. Seules les corrélations introduites explicitement apparaissent.

Longueur de la période :

Tous les générateurs finissent par reproduire les mêmes séries de nombres. Typiquement, la période est limitée au nombre de configurations possibles d'un nombre entier dans un mot d'ordinateur. Ce nombre, la "semence", divisé par sa valeur maximum, constitue le nombre généré sur [0, 1] et, sert de base de calcul de la prochaine valeur prise par la semence. Si n est le nombre de bits dans un mot d'ordinateur, la longueur maximum de la séquence est de l'ordre de 2^{n-2}, soit le

nombre de nombres entiers impairs positifs pouvant être représentés par un mot. Si n=32 bits, la période sera de l'ordre de 10^9, ce qui est relativement court pour beaucoup d'applications complexes; on aura recours à deux ou plusieurs semences (voir plus loin) pour allonger la longueur de la période.

Reproductibilité :

Il suffit de partir des mêmes valeurs initiales des semences pour reproduire les mêmes séquences de nombres pseudo-aléatoires. Ceci est en contradiction totale avec la notion de processus aléatoire, mais est essentiel pour :

- tester un programme en cours de développement, par comparaison entre résultats successifs;
  on s'assure que les différences résultent uniquement des modifications apportées à l'algorithme

- pouvoir générer à nouveau la configuration en cause, et elle uniquement, en cas d'erreur.

Longueurs de sous séquences disjointes :

Les générateurs permettent d'accéder à la valeur actuelle de leurs semences et donc de couper en travaux successifs un travail de simulation trop lourd en temps de calcul pour être effectué en une seule fois. Cependant, chaque travail ne peut être commencé qu'une fois le précédent terminé et connue la nouvelle valeur de départ à donner aux semences. Certains générateurs à plusieurs semences permettent de démarrer les travaux de simulation en parallèle. La séquence produite par une première semence est assez longue pour un travail, alors que la valeur donnée à une autre semence garantit la génération de séquences disjointes,

Portabilité:

La portabilité est essentielle pour le travail en collaboration. Elle permet de reproduire les mêmes séquences sur des ordinateurs d'architecture ou de système différent. Pour cela, il faut que le code soit portable et débarrassé des conséquences de la précision, par exemple, des unités de calcul en virgule flottante. Il faut, bien sûr, que le reste du code utilisateur n'entraîne pas de divergence à l'exécution, due, par exemple, à la précision.

Rapidité :

Les générateurs installés sur les ordinateurs actuels sont très rapides et le temps de calcul est très souvent négligeable par rapport à l'exécution du reste du code. En fait, le temps de transfert entre programme appelant et sous-programme appelé est comparable au temps de calcul nécessaire pour générer un nombre. C'est pourquoi ces sous-programmes génèrent habituellement un vecteur de nombres plutôt qu'un seul nombre.

3. Générateurs de nombres aléatoires uniformes sur [0, 1]

Historique : le générateur de Von Neuman

- soit un nombre de départ X_0 à r chiffres,
- Y_1, constitué des $r/2$ chiffres centraux de X_0, est le premier nombre aléatoire,
- soit $X_1 = Y_1 \times Y_1$ un nouveau nombre de r chiffres,
- Y_2, constitué des $r/2$ chiffres centraux de X_1, est le second nombre aléatoire,
- ...

Un problème est que certains nombres se reproduisent et, dans ce cas, la longueur de la série devient 1.

Générateurs simples

Soit la nouvelle valeur de la semence à la génération $n+1$ calculée à partir des k valeurs précédentes:

$$X_i = (a_{i-1} X_{i-1} \bullet a_{i-2} X_{i-2} \ldots\ldots \bullet a_{i-k} X_{i-k} + c) \text{ , modulo}(m)$$

où $\bullet$ est une opération peu coûteuse en temps d'exécution comme **+, -** ou OR. L'opérateur OR logique présente l'inconvénient d'introduire une corrélation systématique entre un bit donné des semences précédentes et celui de la nouvelle semence, alors que **+** et **-** "mélangent" les bits.

Les générateurs n'utilisant que la dernière valeur de la semence ne peuvent produire que des séquences dont la longueur maximum est de ~ 10^9 sur un ordinateur à 32 bits. Deux classes de générateurs de ce type sont décrites ci-dessous:

Méthode multiplicative linéaire congruente (D.H. Lehmer) **:**

La nouvelle valeur de la semence est donnée par :

$$x_i = (a\, x_{i-1} + c) \quad \text{modulo } m$$

La valeur de c a peu d'importance. Elle ne peut être nulle si l'on veut pouvoir générer la valeur 0; sinon la période deviendrait 1. La valeur donnée à m est habituellement très proche du plus grand nombre entier représentable et la valeur du nombre réel généré est donnée par x/m. La faiblesse de la méthode (démontrée par Marsaglia) réside dans la présence de fortes corrélations: les nombres pris par

groupes successifs de *d* pour former les coordonnées d'un point dans l'espace à *d* dimensions conduisent à la concentration de points sur un nombre d'hyperplans parallèles très réduit par rapport à celui attendu pour des nombres aléatoires. On montre que le nombre d'hyperplans dans l'espace à d dimensions pour des nombres à n bits est au maximum égal à : $N = \left(d!\ 2^n\right)^{1/d}$

Par exemple, pour $n = 32$:

d =	3	4	6	10
N =	2953	566	120	41

Le problème est donc de trouver un multiplicateur *a*, pour *m* très grand, donnant une période longue et un nombre d'hyperplans aussi proche que possible du nombre maximum.

Quelques multiplicateurs, à titre d'exemple :

- Knuth $a = 1664525, \quad m = 2^{32}$
- L'Ecuyer : $a = 40014, \quad m = 2147483563$
  $a = 40692, \quad m = 2147483399$

Série de Fibonacci différée:

Il s'agit en fait d'une série de Fibonacci différée de *p* et *q* itérations :

$x_i = \left(x_{i-p} \bullet x_{i-q}\right)$ modulo *m,* avec $q < p$.

La série est non différée si $p = 2$ et $q = 1$. On montre que, pour certaines valeurs de *p* et *q*, la période de la série est de $\left(2^{(p-1)}\right) \times \left(2^{(n-1)}\right)$. Cette méthode nécessite la génération préalable et la rétention en mémoire de *p* nombres aléatoires auxquels il faut avoir accès si l'on veut interrompre puis continuer la série.

<u>Amélioration des générateurs simples</u>

But : doubler l'ordre de grandeur de la longueur de la série; passer de ~ 10^9 à 10^{18}. Les deux méthodes décrites introduisent l'utilisation d'un second générateur éventuellement plus simple et rapide:

Algorithme de Bays-Durham:

Le second générateur n'est utilisé que pour générer un ordre aléatoire dans le choix des nombres générés par le premier.

Exemple :

- génération d'un vecteur *V* de *n* nombres aléatoires par le premier générateur,
- génération d'un indice *1* □ *j* □ *n* par le second,
- choix de *V(j)* comme nombre aléatoire,
- remplacement de *V(j)* par un nouveau nombre généré par le premier.

Mélange de bits :

Deux nombres *x* et *y* sont générés par deux générateurs différents et mélangés par une opération simple comme $z = (x \bullet y) \ mod(1)$

Un générateur très performant, **RANMAR**, est basé sur la méthode du mélange des bits. (G. Marsaglia, A. Zaman and W.-W Tsang, *Stat. Prob. Lett* 9 (1990) 35)

- $x_i = (x_{i-p} - x_{i-q} + 1.) \ mod(1.)$, $q = 33, p = 97$
- soit $0 < c, d < 1$
  $y_i = y_{i-1} - c$
  si $y_i < 0.$ alors $y_i = y_i + d$
- $z_i = (x_i - y_i + 1.) \ mod(1.)$

Les principaux avantages de RANMAR sont:

- code portable en langage de haut niveau,
- période ~ 10×10^{43}
- grand nombre (30000 × 30000) de séquences parallèles indépendantes de périodicité moyenne ~ 10^{31} définies par les valeurs initiales de deux semences de 5 chiffres.

Un générateur encore plus performant, **RANLUX,** a une période de ~ 10×10^{165} (M. Lüsher, *Computer Phys. Comm.* 79 (1994) 100).

<u>Tests d'uniformité et du caractère aléatoire</u>

Uniformité : calcul des moments centraux

$$\mu_n = \frac{1}{(n+1)2^n} \text{ si } n \text{ est pair} \rightarrow \mu_2 = \sigma^2 = \frac{1}{12}$$
$$\mu_n = 0 \qquad \text{si } n \text{ est impair}$$

Caractère aléatoire :

Les tests du caractère aléatoire d'une série de nombre consistent à calculer une fonction des nombres aléatoires générés dont l'espérance mathématique peut être calculée : la distribution des niveaux de confiance de cet ensemble de tests doit être uniforme entre 0 et 1. Un test simple repose sur le théorème central limite :

La distribution des N moyennes $\overline{x}_i$, $i = 1, N$ de N ensembles disjoints de n nombres consécutifs doit être compatible avec une normale de moyenne 0.5 et de variance $1/12n$.

4. Génération de nombres aléatoires suivant une FDP non uniforme

Ces méthodes présupposent l'existence d'un générateur correct de nombres aléatoires uniformément distribués. Sauf indication contraire, on supposera dorénavant que ξ est un nombre aléatoire distribué uniformément sur [0, 1]

4.1.Variable discrète

4.1.1.Nombre fini de valeurs

Soit une variable discrète x pouvant prendre N valeurs possibles $(x_1,...,x_N)$
dont la fonction de fréquence est définie par les N valeurs des probabilites $(p_1,...,p_N)$
et la fonction de distribution par $P_n = \sum_{i=1}^{n} p_i \ , n = 1,N$
Soit $P_0 = 0$; on a $P_N = 1$

La valeur x_n attribuée à x satisfait à la condition $P_{n-1} < \xi \leq P_n$

Cette méthode s'applique directement aux distributions binomiales et multinomiales. Elle s'applique aussi à une variable continue si sa FDP est spécifiée par les contenus de classes comme, par exemple, les contenus des boîtes d'un histogramme. Il convient de d'abord normaliser les contenus des classes en les transformant en fréquences.

4.1.2.Nombre infini de valeurs

Si le nombre de valeurs possibles de x est infini, l'ensemble des valeurs du vecteur P ne peuvent plus être calculées avant d'entreprendre le processus de génération. On peut cependant calculer un ensemble fini N de valeurs $P_1, P_2, \cdots P_N$ telles que $P_N <\cong 1$. Pour chaque valeur prise par ξ inférieure ou égale à P_N, le problème est ramené à un nombre fini de valeurs. Pour les valeurs supérieures, il faut construire successivement $P_{N+1,}$ …, P_{N+k-1}, PN_{+k} jusqu'à ce que la condition $P_{N+k-1} < \xi \leq P_{N+k}$ soit satisfaite. La valeur x_{N+k} est attribuée à la variable. La distribution de Poisson est simulée suivant cette méthode.

4.2.Variable continue

4.2.1.Méthode cumulative

Soit $f(x)$ la fdp de x sur l'intervalle $[a, b]$

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

et $F(x)$ la fonction de distribution:

$$F(x) = \int_a^x f(x')\,dx' \quad ; \; 0 \le F(x) \le 1$$

$$dF(x) = f(x)\,dx$$

$\boxed{x = F^{-1}(\xi) \text{ est distribué suivant } f(x)}$

Démonstration :

$$fdp(x) = fdp(\xi)\,\frac{d\xi}{dx} = 1 \times \frac{dF(x)}{dx} = \frac{f(x)dx}{dx} = f(x)$$

La méthode ne s'applique malheureusement qu'à un nombre limité de FDP f telles que F et F^{-1} puissent être calculées analytiquement comme les distributions exponentielle, (co)sinusidale , polynomiale de degré peu élevé, triangulaire, trapézoïdale. Des exemples sont donnés dans le chapitre suivant.

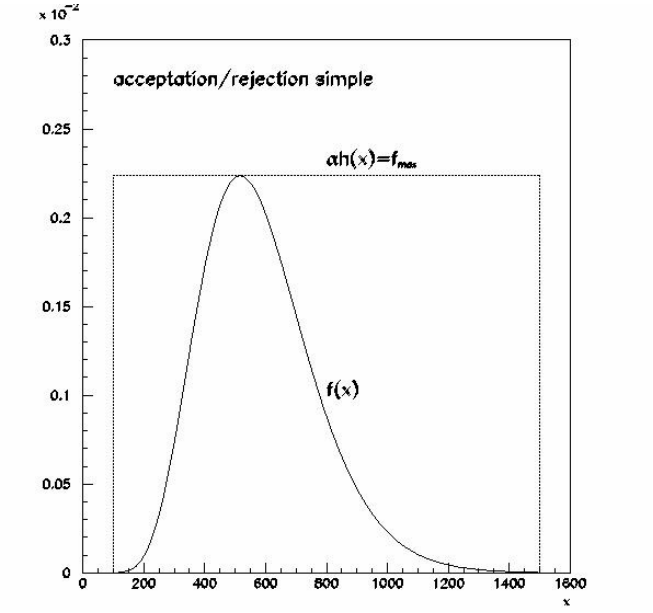
4.2.2.Méthodes d'acceptation /réjection

Les méthodes d'acceptation/réjection consistent à approximer la FDP $f(x)$ selon laquelle x doit être distribuée par une FDP $h(x)$ aussi proche que possible de $f(x)$ et à laquelle peut s'appliquer la méthode cumulative. Dans un premier temps, x est simulé suivant $h(x)$. On corrige ensuite pour les différences entre les deux fonctions $f(x)$ et $h(x)$ en rejetant certaines des valeurs générées. La méthode de réjection fera usage d'une constante α définie comme la plus petite valeur telle que $\alpha h(x) \ge f(x)$. On a $\alpha > 1$.

Notation : On appellera

- ξ un nombre aléatoire uniformément distribué sur [0,1].
- $[a, b]$ le domaine sur lequel la FDP $f(x)$ selon laquelle x doit être distribuée est non nulle. Le cas échéant, a et/ou b pourront être infinis.
- $h(x)$ toute FDP à laquelle peut s'appliquer la méthode cumulative.
- α la plus petite valeur de la constante telle que $\alpha h(x) \ge f(x$

Méthode d'acceptation /réjection simple



On borne $f(x)$ par la fdp uniforme $h(x) \equiv 1/(b-a)$
multipliée par $\alpha = f_{max} \times (b-a)$ où $f_{max} = Max(f(x))$
de sorte que $\alpha h(x) = f_{max} \geq f(x)$ sur $[a,b]$

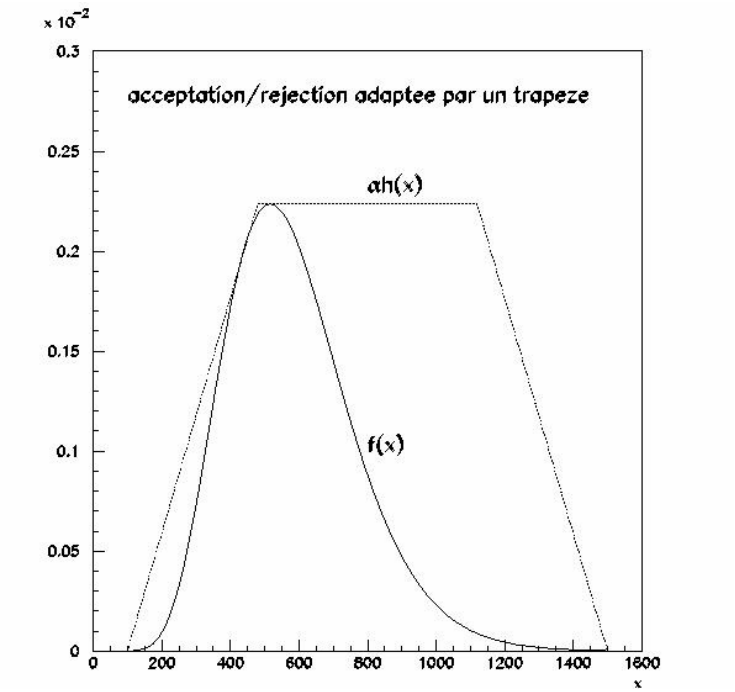
Algorithme:

- générer ξ_1 et ξ_2 sur [0, 1]
- $x = a + \xi_1 \times (b-a)$
- accepter x si $\xi_2 < \dfrac{f(x)}{\alpha h(x)} = \dfrac{f(x)}{f_{max}}$

Nombre moyen de nombres aléatoires uniformes à générer
pour générer 1 nombre aléatoire suivant $f(x)$:

$$2\frac{\int_a^b \alpha h(x)\,dx}{\int_a^b f(x)\,dx} = 2\alpha$$

Méthode d'acceptation / réjection adaptée à l'échantillon

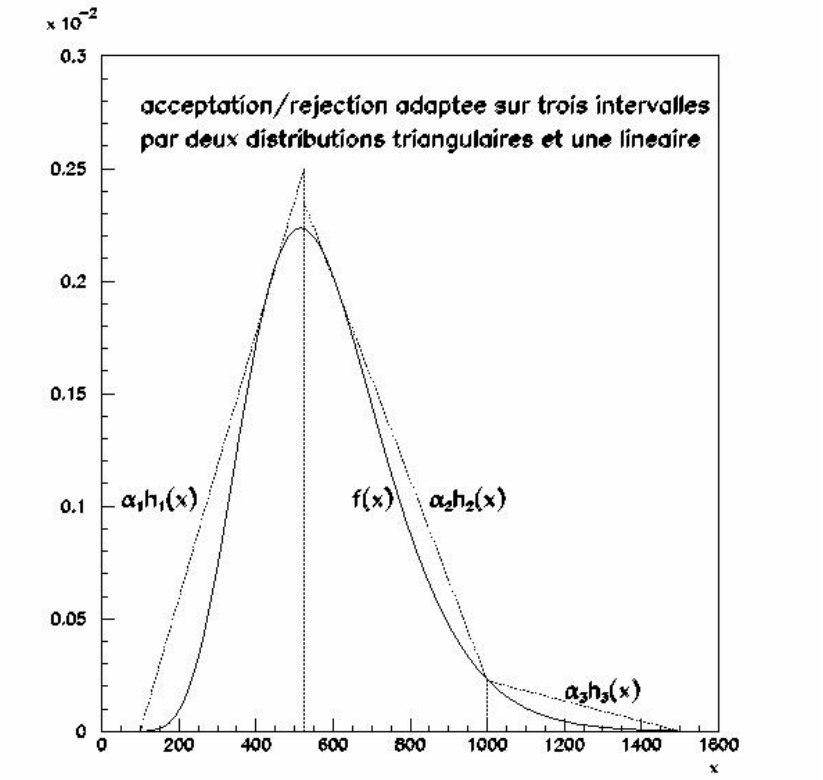


On borne $f(x)$ par la fdp $h(x)$ de forme aussi voisine que possible de $f(x)$ multipliée par $\alpha > 1$ de sorte que $\alpha h(x) \geq f(x)$ sur $[a,b]$

Algorithme:
- générer x sur $[a,b]$ suivant $h(x)$
- générer ξ sur [0, 1]
- accepter x si $\xi < f(x) / \alpha h(x)$

Nombre moyen de nombres aléatoires uniformes à générer
pour générer 1 nombre aléatoire suivant $f(x) = 2\alpha$.

On augmente de l'efficacité de la méthode en divisant [a, b] en n intervalles de bornes $a_0 = a, a_1, a_2, ..., a_n = b$
Sur chaque intervalle $k = 1,n$, $f(x)$ est bornée par $\alpha_k h_k(x)$



Algorithme:

- $p_j = \dfrac{\alpha_j}{\sum_{i=1}^{n} \alpha_i}$, $P_j = \dfrac{\sum_{i=1}^{j} \alpha_i}{\sum_{i=1}^{n} \alpha_i}$ $\quad \forall j = 1,n$

- générer ξ sur [0,1]

-sélectionner l' intervalle k tel que $P_{k-1} \le \xi < P_k$

- procéder comme auparavant sur l'intervalle $[a_{k-1}, a_k]$ avec la fonction $\alpha_k h_k(x)$

Méthode de Composition / Réjection [J.C. Butcher]

Cette méthode est très semblable à la méthode d'acceptation / réjection adaptée à l'échantillon sur plusieurs intervalles.

$$0 \le g_k(x) = \frac{f(x)}{\alpha_k h_k(x)} \le 1 \text{ sur } [a_{k-1}, a_k]$$

Puisque α_k>1 est le plus petit nombre tel que $\alpha_k h_k(x) \ge f(x)$: $Max(g_k(x)) = 1$

On a donc

$$f(x) = \alpha_k h_k(x)\, g_k(x)$$

sur l'intervalle *k*, plutôt que

$$f(x) \le \alpha_k h_k(x)$$

comme dans la méthode adaptée à l'échantillon.

Algorithme :
-sélectionner aléatoirement un intervalle *k* comme dans la méthode adaptée à l'échantillon
- générer *x* suivant $h_k(x)$
- accepter *x* si $\xi \le g_k(x)$

Un exemple de simulation suivant une FDP normale et basée sur cette méthode est donné plus loin.

5. Exemple de simulation de FDP spécifiques

Distribution Multinomiale

C'est l'application directe de la simulation d'une variable discrète à nombre fini de valeurs possibles.

Distribution de Poisson

C'est l'application directe de la simulation d'une variable discrète à nombre infini de valeurs possibles.

Le poids attribué à la valeur n se calcule par la relation itérative :

$$p_0 = e^{-\mu} \quad , \quad p_n = p_{n-1}\frac{\mu}{n}$$

Distributions trapézoïdale et triangulaire

Soit $a < b$ les demi-bases du trapèze et c le minimum de x: $x \in [c, c+2b]$

$x = c + (b+a)\,\xi_1 + (b-a)\,\xi_2$

Si $a = 0$ la distribution est triangulaire:

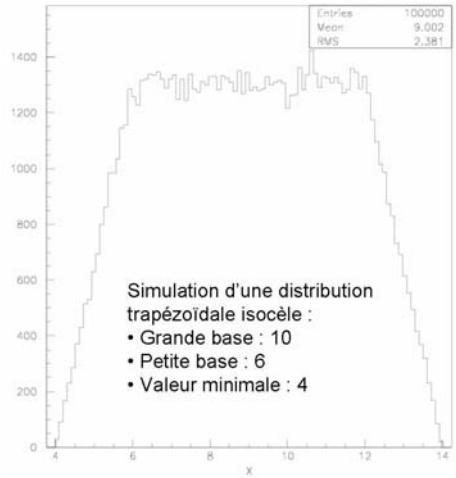
$x = c + b(\xi_1 + \xi_2)$

On obtient un demi-trapèze ou triangle gauche par la transformation:

si $x > c+b : x = (c+b) - (x-(c+b))$

et un demi-trapèze ou triangle droit par la transformation:

si $x < c+b : x = (c+b) + ((c+b) - x)$



Distribution linéaire

Soit $[a,b]$ le domaine des valeurs de *x*.

Par la transformation $y = \dfrac{x-a}{b-a}$, on est ramené à la simulation de *y* suivant $f(y) = \alpha y + \beta$ sur $[0,1]$

Définition de la droite: $r = \dfrac{f(y=1)}{f(y=0)} = \dfrac{\alpha+\beta}{\beta}$

Normalisation: $\int_0^1 \alpha y + \beta \, dy = \dfrac{\alpha}{2} + \beta = 1$

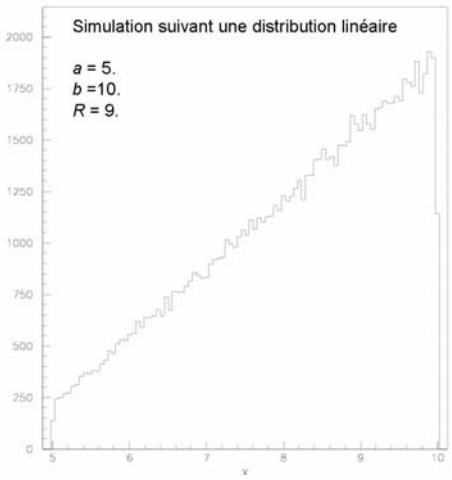
Coéficents de la droite: $\beta = \dfrac{2}{r+1} \qquad \alpha = \beta(r-1)$

Simulation par la méthode cumulative

$$F(y) = \xi = \frac{\alpha}{2} y^2 + \beta y$$

$$y = \frac{-\beta + \sqrt{\beta^2 + 2\alpha\xi}}{\alpha}$$

$$x = a + (b-a) \times y$$



Exemple

$a = 5. \quad b = 10.$

$y = \dfrac{x - 5.}{5.}$

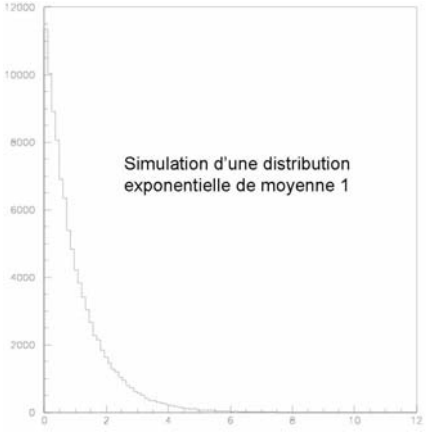
$r = f(x=b)/f(x=a) = f(y=1)/f(y=0) = 9.$

$\int_0^1 (\alpha y + \beta) dy = 1$

$\Rightarrow \alpha = 1.6 \quad \beta = 0.2$

● Distributions exponentielle, hyperexponentielle, gamma

Toutes ces FDP peuvent être simulées par la méthode cumulative



Exponentielle

$$f(x) = \frac{1}{\beta} e^{-\frac{x}{\beta}}$$

$$F(x) = 1 - e^{-\frac{x}{\beta}} = \xi$$

$x = -\beta \log(1-\xi)$ ou $-\beta \log(\xi)$

Hyperexponentielle

C'est la la somme pondérée de *n* fdp exponentielles de moyennes différentes:

$$f(x) = \sum_{i=1}^{n} \frac{p_i}{\beta_i} e^{-\frac{x}{\beta_i}} \quad \text{avec} \quad \sum_{i=1}^{n} p_i = 1$$

- sélectionner aléatoirement l'exponentielle de moyenne β_k suivant la méthode adaptée à un nombre fini de valeurs discrètes en utilisant le poids p_k
- simuler *x* suivant cette exponentielle

Gamma

C'est la distribution de la somme de *n* variables aléatoires indépendantes distribuées suivant une même exponentielle:

$$f(x) = \frac{1}{\beta \Gamma(n)} \left(\frac{x}{\beta}\right)^{n-1} e^{-\frac{x}{\beta}}$$

$$x = -\beta \log\left(\xi_1 \xi_2 \xi_3 \ldots \xi_n\right)$$

Distribution Normale

La simulation se fait sur la variable réduite suivant une FDP de moyenne 0 et variance 1.

$$y = \frac{x - \mu}{\sigma}$$

$$f(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2}$$

Méthode basé sur le théorème central limite

$x = \dfrac{\sum_{i=1}^{n} \xi_i - n/2}{\sqrt{n/12}}$ est distribué suivant une normale réduite si *n* grand

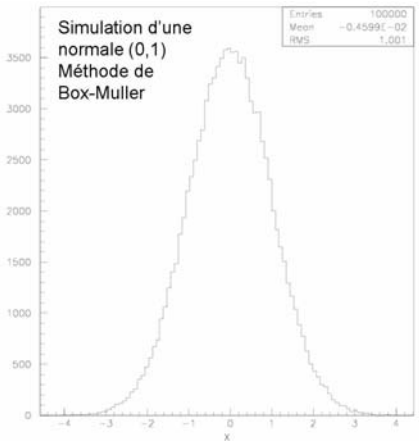
si $n = 12 : x = \sum_{i=1}^{6} \xi_i - 6$

Cette méthode est donnée a titre anecdotique parcequ'elle présente deux inconvénients majeurs :

inefficacité: tirage de *n* nombres aléatoires uniformes

approximation: distribution limitée à $\pm n/2\,\sigma$

Méthode exacte de Box et Muller



Soient

$$x_1 = \sqrt{-2\log \xi_1}\cos 2\pi\xi_2$$

$$x_2 = \sqrt{-2\log \xi_1}\sin 2\pi\xi_2$$

d'où

$$\xi_1 = e^{-\frac{\left(x_1^2+x_2^2\right)}{2}}$$

$$\xi_2 = \frac{arg\ \mathrm{tg}\left(x_2 / x_1\right)}{2\pi}$$

$$f(x_1,x_2) = \begin{vmatrix} \frac{\partial\xi_1}{\partial x_1} & \frac{\partial\xi_1}{\partial x_2} \\ \frac{\partial\xi_2}{\partial x_1} & \frac{\partial\xi_2}{\partial x_2} \end{vmatrix} f(\xi_1,\xi_2) \quad \text{avec} \quad f(\xi_1,\xi_2) = 1$$

$$f(x_1,x_2) = \frac{1}{2\pi}e^{-\frac{1}{2}\left(x_1^2+x_2^2\right)} = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x_1^2}\ \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x_2^2} = f\left(x_1\right)f\left(x_2\right)$$

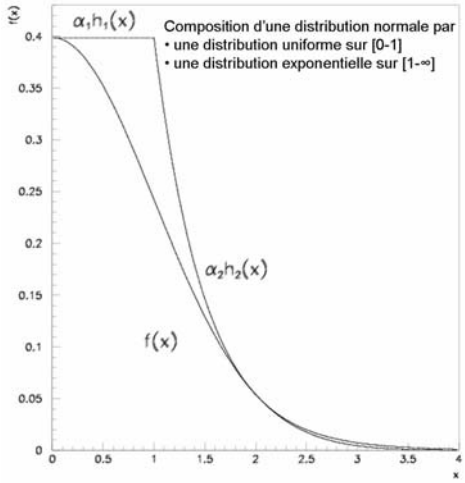
x_1 et x_2 sont distribuées indépendamment comme des normales réduites.

Méthode de composition / réjection de Burcher

Simulation de

$f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} = \alpha_i\, g_i(x)h_i(x)$ sur deux domaines $i = 1,2$

$$\begin{bmatrix} i & [x] & \alpha_i & g_i(x) & h_i(x) \\ 1 & [0-1] & \frac{1}{\sqrt{2\pi}} & e^{-\frac{x^2}{2}} & 1 \\ 2 & [1-\infty] & \frac{1}{2\sqrt{2\pi}} & e^{-\frac{(x-2)^2}{2}} & 2e^{-2(x-1)} \end{bmatrix}$$



On observe que :

les $g_i(x)$ satisfont à la condition $0 \le g_i(x) \le 1$ sur les domaines respectifs,

$h_1(x)$ est une fdp uniforme,

$h_2(x)$ est une fdp exponentielle décroissante de moyenne $\frac{1}{2}$, décallée de 1.

les $h_i(x)$ sont correctement normalisées: $\int_{domaine\ i} h_i(x)\,dx = 1$

$$\frac{\alpha_1}{\alpha_1+\alpha_2} = \frac{2}{3} \quad \text{et} \quad \frac{\alpha_2}{\alpha_1+\alpha_2} = \frac{1}{3}$$

Algorithme:

- générer ξ_1, ξ_2 et ξ_3
- si $\xi_1 \geq 2/3$: le domaine 2 $(x > 1)$ est choisi

  $\xi = 3\xi_1 - 2$ est distibué uniformément sur $[0,1]$

  $x = 1 - \frac{1}{2} log(\xi)$ est distibué suivant $h_2(x)$

  si $\xi_2 \leq g_2(x) = e^{-\frac{(x-2)^2}{2}}$: la valeur de x est acceptée
- si $\xi_1 < 2/3$: le domaine 1 $(x < 1)$ est choisi

  $\xi = 3\xi_1/2$ est distibué uniformément sur $[0,1]$

  $x = \xi$ est distibué suivant $h_1(x)$

  si $\xi_2 \leq g_1(x) = e^{-\frac{x^2}{2}}$: la valeur de x est acceptée
- attribution du signe :

  si $\xi_3 > 0.5$: $x = -x$

La méthode a une efficacité de 0.84, mais est pratiquemet aussi rapide que la méthode exacte (facteur 1.19) parcequ'elle ne demande pas le calcul de sinus et de cosinus.

Distribution en χ^2

La méthode exacte repose sur la définition de la variable distribuée suivant la distribution χ^2_ν à ν degrés de libertés qui est la somme des carrés de ν variables normales réduites. Elle utilise la transformation Box-Muller introduite dans la méthode de simulation exacte de la distibution normale:

Si $\nu = 2n$ pair: $x = -2\, log(\xi_1 \xi_2 \ldots \xi_n)$

Si $\nu = 2n+1$ impair: $x = -2\, log(\xi_1 \xi_2 \ldots \xi_n) - 2\, log\, \xi_{n+1}\, cos^2(2\pi\xi_{n+2})$

Distribution de Student

La méthode repose sur la définition de la variable distribuée suivant la distribution de Student à ν degrés de libertés:

$$x = \frac{y}{\sqrt{z/\nu}}$$

où y est distribué comme une normale réduite, z comme une χ^2_ν, et y et z sont indépendantes.

Distribution F de Fisher

La méthode repose sur la définition de la variable distribuée suivant la distribution de Fisher à μ,ν degrés de libertés:

$$x = \frac{y/\mu}{z/\nu}$$

où y et z sont indépendantes et distribuées comme des χ^2_μ et χ^2_ν.