Partie II - Simulation Stochastique
Chapitre 3 : Génération et Simulation de Variables Aléatoires

Dr. Nawel Arrar Remita
3 ème année ENSIA

Novembre 2023

Introduction

Lorsque les systèmes sont modélisés comme des processus stochastiques ou des systèmes, ils deviennent complexes et dynamiques, les solutions analytiques ou numériques peuvent devenir intraitables. Dans de tels cas, un programme informatique qui imite le comportement du système peut être utilisé, dont le but principal est d'étudier les performances d'un système complexe.
La programmation informatique (ou simulation) est exécuté avec plusieurs valeurs aléatoires et les comportements ainsi modélisés sont enregistrés pour analyse.
On cherche donc à produire une suite de nombres qui auront les apparences du hasard.

Génération des nombres aléatoires uniformes

- La clé d'une bonne simulation est la qualité des générateurs de nombres aléatoires qui y sont utilisés.
- Toute simulation de phénomènes aléatoires, quelle que soit leur nature, se fait par ce type de nombres.
- Il est important de disposer de méthodes fiables et efficaces pour les générer.
- On s'intéresse donc à la source capable de produire une suite $\{u_i\}$ de nombres aléatoires indépendants et uniformément distribués sur $[0, 1]$. Ces derniers sont **pseudo aléatoires.**
- On peut à partir d'une telle suite générer toute autre suite de variables aléatoires de loi arbitraire.

Génération des nombres aléatoires uniformes

Un générateur algorithmique doit posséder une bonne forme mathématique, simple et facillement programmable. L'algorithme doit être récursif,

$$u_{i+1} = f(u_i, u_{i-1}, ..., u_{i-k}) \text{ avec } k < i \text{ et } i \geq 1,$$

ou

$$u_{i+1} = f(u_i) \text{ avec } i \geq 1.$$

Ici f est une fonction donnée à valeurs dans $[0, 1]$. Un bon générateur de nombres aléatoires devrait avoir une longue période avant que les nombres aléatoires ne se recyclent.

Méthodes congruentielles

La méthode linéaire congruentielle (LC) est une technique largement utilisée pour générer des nombres aléatoires. Dans cette méthode, le nombre aléatoire suivant y_i est généré en utilisant le nombre aléatoire actuel y_{i-1} en utilisant l'équation suivante :

$$y_i = (a \times y_{i-1} + b) \bmod m,$$

où a et b sont des constantes non négatives. Pour produire m

nombres différents, les conditions suivantes doivent être satidfaites

- *Les réels m et b sont premiers entre eux.*
- *Tout diviseur premier de m divise $(a-1)$.*
- *Les nombres $\{u_i\}$ uniformes sur $[0,1]$ seront obtenus par normalisation $u_i = \frac{y_i}{m}$, $i \geq 0$.*

Pour démarrer le générateur, on donne la racine y_0. En général m est une puissancede 2 ou 100.

Méthodes congruentielles

Pour augmenter la gamme des nombres générés et réduire la corrélation entre les nombres successifs, plusieurs variantes de la méthode LC ont été proposées.

- Méthode congruentielle multiplicative

$$y_{i+1} = a y_i \bmod m,$$

la période maximale est m-1.

- Méthode congruentielle additive

$$y_{i+1} = (y_i + y_{i-1}) \bmod m.$$

- Méthode congruentielle mixte

$$y_{i+1} = (a y_i + b) \bmod m,$$

où $0 < a < m$ et $0 < b < m$. La période maximale est m.

On nomme a le multiplicateur, b l'incrément et m le modulo.

Méthode des carrés moyens de Von Neumann

Il s'agit d'un des premiers générateurs de nombres au hasard.

Algorithme

1. Choisir la racine u_0;
2. Calculer $a = u_0^2$;
3. Poser u_1 suite de chiffres situés au milieu de la partie décimale de a;
4. Poser $a = u_1^2$;
5. Retour en 3.

Remark

Le nombre devra, à chaque itération, comporter autant de chiffres significatifs que le nombre u_0.

Inconvénient majeur : si à une étape donnée, l'un des nombres est nul, c'est toute la suite qui sera constituée de termes nuls.

Validation des générateurs

Pour vérifier que la suite $\{u_i\}$ est bien aléatoire et issue d'une loi uniforme, on doit s'assurer que lorsque $n \longrightarrow \infty$ la moyenne empirique $\overline{U} = \frac{1}{n}\sum_{i=1}^{n} u_i$ doit être voisine de $\frac{1}{2}$. En plus d'un des critères suivants :

Critère 1 : Soit la variance $Var\,(U) = \frac{1}{n}\sum_{i=1}^{n}\left(u_i - \overline{U}\right)^2$. Alors

$$Var\,(U) \longrightarrow \frac{1}{12} \text{ (convergence en probabilité).}$$

Validation des générateurs

Critère 2 : On partage $[0,1]$ en m intervalles $I_j = \left[\frac{j-1}{m}, \frac{j}{m}\right]$, tel que $1 \leq j \leq m$ et $m << n$.
Soit V_j le nombre de points u_i qui tombent dans l'intervalle I_j et $n = \sum_{j=1}^{m} V_j$ le nombre total d'observations. Alors,

$$\frac{V_j}{n} = \frac{1}{m}, \forall 1 \leq j \leq m.$$

Validation des générateurs

Critère 3 : (Test d'uniformité Khi-deux)
Soit la suite $(I_j)_{1\leq j\leq m}$ une partition de l'ensemble de définition de la variable à tester.
Soit l'hypothèse $\mathcal{H}_0$: la suite $\{u_i\}$ i.i.d $\in \mathcal{U}[0,1]$.

On considère $p_j = \mathbf{P}(U \in I_j) = \frac{longueur\ I_j}{1-0}$, $\sum_{j=1}^{m} p_j = 1$.

On calcule $W = \sum_{j=1}^{m} \frac{(V_j - np_j)^2}{np_j} = \chi^2_r$ qui suit une loi Khi-deux à $r = m-1$ degrés de liberté. Si $\chi^2_{m-1} > \chi^2_{\alpha,m-1}$, où $P\left(\chi^2_{m-1} > \chi^2_{\alpha,m-1}\right) = \alpha$ et $\alpha - 1$ est le niveau de confiance ($\alpha = 0.01$ ou 0.05), on rejette $\mathcal{H}_0$.

Validation des générateurs

Critère 4 : Si $U \in \mathcal{U}[0,1]$, alors

$$E[U] \pm \sigma_U = \frac{1}{2} \pm \frac{1}{2\sqrt{3}}$$

et la proportion de points dans l'intervalle $[0.21132; 0.78868]$ doit être voisine de $2\sigma_U = 0.57736$.

Validation des générateurs

Critère 5 : (Test de Kolmogorov-Smirnov)
On l'utilise lorsque la loi à tester possède une fonction de répartition $F(x)$ continue. La statistique du test est $D_n = \max |F_n(x) - F(x)|$, où $F_n(x)$ est la fonction de répartition empirique.
Soit $\{u_i\}$ et posons l'hypothèse $\mathcal{H}_0 : \{u_i\}$ i.i.d $\in \mathcal{U}[0,1]$.

1. ordonner les observations dans l'ordre croissant $u_1 \leq u_2 \leq ... \leq u_n$
2. calculer $D^+ = \max \left| \frac{i}{n} - u_i \right|$; $D^- = \max \left| u_i - \frac{i-1}{n} \right|$;
3. calculer $D = \max \{D^+, D^-\}$;
4. La valeur critique D_α est lue de la table pour α spécifié et taille de l'échantillon n donnée;
5. Si $D_\alpha > D$, $\mathcal{H}_0$ est acceptée au seuil α.

Procédés généraux de simulation des variables aléatoires

Soit à produire une suite $\{x_i\}$ issue de la loi de probabilité d'une variable aléatoire X de fonction de répartition $F(x)$. On cherche à obtenir la suite $\{x_i\}$ à partir d'une suite préalable de nombres au hasard $\{u_i\}$ issus de la loi uniforme sur $[0, 1]$.
Il y a des méthodes qui se basent sur la distribution que l'on cherche à générer, cependant, il existe quelques approches basées sur les propriétés spécifiques de la loi.

Méthode de l'inverse

Cas continu

Theorem

Soit F_X une fonction de répartition définie sur un intervalle $[a, b]$ on applle fonction quantile définie par :

$$F_X^{-1}(u) = \inf\{x \in [a, b] : F_X(x) \geq u\} \text{ où } 0 \leq u \leq 1.$$

La variable aléatoire $X = F^{-1}(U)$ possède la fonction de répartition F_X lorsque U est de loi uniforme sur $(0, 1)$.

L'algorithme pour générer une observation d'une variable aléatoire X par l'inverse généralisé:

Algorithme

1. Generate $u \in \mathcal{U}(0, 1)$.
2. Return $X = F_X^{-1}(u)$.

Méthode de l'inverse

Cas discret

Dans le cas où on n'impose plus la contrainte de la continuité sur la fonction F, on utilise F^* définie par

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

au lieu de F^{-1}.
Soit X une variable aléatoire de loi discrète:

$$p_1\delta_{x_1} + p_2\delta_{x_2} + ... + p_n\delta_{x_n};\ p_i > 0\ i = 1, 2, .., n\ ;\ \sum_{i=1}^{n} p_i = 1.$$

On simule X à partir de $u \in U[0, 1]$ en posant

$$X = x_1 1_{u<p_1} + x_2 1_{p_1 \leq u < p_1+p_2} + ... + x_n 1_{p_1+p_2+...+p_{n-1} \leq u < 1}. \quad (4)$$

En d'autre termes, on suppose que
$x_1 < x_2 < ... < x_n; F(x_k) = \sum_{i=1}^{k} p_i, k \in \{1, 2, ..., n\}$.

Méthode de l'inverse

Cas discret

Dans ce cas si $u \in U[0,1]$, alors
$F(x_{k-1}) < u \leq F(x_k) \Rightarrow X = x_k, \forall k \in \{1, 2, ..., n\}$.

Algorithme

1. Arrange $x_1, x_2, ..., x_n$ s.t. $x_1 < x_2 < ... < x_n$.

Méthode de l'inverse

Cas discret

Dans ce cas si $u \in U[0,1]$, alors
$F(x_{k-1}) < u \leq F(x_k) \Rightarrow X = x_k, \forall k \in \{1, 2, ..., n\}$.

Algorithme

1. Arrange $x_1, x_2, ..., x_n$ s.t. $x_1 < x_2 < ... < x_n$.
2. Generate $u \in U[0,1]$.

Méthode de l'inverse

Cas discret

Dans ce cas si $u \in U[0,1]$, alors
$F(x_{k-1}) < u \leq F(x_k) \Rightarrow X = x_k, \forall k \in \{1, 2, ..., n\}$.

Algorithme

1. Arrange $x_1, x_2, ..., x_n$ s.t. $x_1 < x_2 < ... < x_n$.
2. Generate $u \in U[0,1]$.
3. If $0 < u \leq p_1$, then output x_1.
   If $p_1 < u \leq p_1 + p_2$, then output x_2.
   If $p_1 + p_2 < u \leq p_1 + p_2 + p_3$, then output x_3.
   If $\sum_{i=1}^{k-1} p_i < u \leq \sum_{i=1}^{k} p_i$, then output x_k, where $0 \leq k \leq n$.

Méthode de l'inverse

Simulation d'un évènement

Soit à simuler un évènement A de probabilité $p = P(A)$. Si $u \in \mathcal{U}[0,1]$, alors $P(u \leq p) = p$, c.à.d : la probabilité que u appartienne à l'intervalle $[0,p]$ est égale à la longueur de l'intervalle. Ainsi on connait la réalisation de u, on peut vérifier que A s'est réalisé en vérifiant l'inégalité $u \leq p$; d'où l'algorithme suivant :

Algorithme

1. Tirer $u \in \mathcal{U}[0,1]$;
2. si $u \in \mathcal{U}[0,p]$ alors A s'est réalisé
3. si $u \in \mathcal{U}]p,1]$ alors l'évènement contraire $\overline{A}$ s'est réalisé.

Méthode d'acceptation-rejet

Principe de la méthode

Cette méthode peut être untilsée si la densité de la v.a à générer $f_X(x)$ est bornée et x appartient à un dommaine borné c'est à dire $a \leq x \leq b$. Soit f_X, elle se résume en quatre étapes :

1. Normaliser le dommaine de f à l'aide d'une échelle C, comme suit :
$$0 \leq g(x) = \frac{1}{C} f(x) \leq 1, \forall x \in [a, b];$$
2. définir x comme fonction linéaire au point $u \in \mathcal{U}[0, 1]$ alors $x = a + (b - a) u$;
3. engendrer une paire de nombre $u_1, u_2 \in \mathcal{U}[0, 1]$;
4. si $u_2 \leq \frac{1}{C} f(a + (b - a) u_1)$ on accepte $x = a + (b - a) u_1$, sinon, on rejette et on choisit une autre paire.

Méthode d'acceptation-rejet

Principe de la méthode

Algorithme

1. Engendrer $u_1 \in \mathcal{U}[0,1]$;
2. Calculer $g(u_1) = \frac{1}{C} f(a + (b-a) u_1)$;
3. Engendrer $u_2 \in \mathcal{U}[0,1]$;
4. Comparer u_2 à $g(u_1)$; si $u_2 \leq g(u_1)$, on accepte $x = a + (b-a) u_1$,
   sinon, on rejette (u_1, u_2) ;
5. On retourne à l'étape 1.

Méthode d'acceptation-rejet

Méthode généralisée (Von Neumann généralisé)

On cherche à majorer la densité $f_X(x) > 0$ à l'aide de $g(x) > 0$ une densité instrumentale d'une v.a. Y et de l'échelle C de sorte que $f_X(x) \leq Cg_Y(x), \forall x \in [a, b]$. Il est important que $C \geq 1$.

Algorithme

1. Find continuous r.v. Y s.t. $g(x) > 0$, and let C be a constant s.t. $\frac{f_X(x)}{g(x)} \leq C, \forall x$ s.t. $f_X(x) > 0$.
2. Generate $u_1 \in \mathcal{U}[0, 1]$.
3. Generate an instance x of Y.
4. Generate $u_2 \in \mathcal{U}[0, 1]$.
5. Compare u_2 to $\frac{f_X(x)}{Cg(x)}$; if $u_2 \leq \frac{f_X(x)}{Cg(x)}$, accepte $X = x$ and stop. Else reject u_1 and return to step 2.

Méthode de composition

Cette méthode suppose qu'une densité de probabilité $f(x)$ peut être exprimée comme un mélange de probabilité et de fonction de densité $g_i(x)$ telle que

$$f(x) = \sum_{i=1}^{n} p_i g_i(x) \tag{9}$$

où $\{p_i\}$ sont des probabilités pour une v. a. discrète N à valeurs dans $\{1, 2, ..., n\}$ et telles que $p_i = P(N = i)$, $0 \leq p_i \leq 1$, $\sum_{i=1}^{n} p_i = 1$; $g_i(x)$ est la densité de probabilité d'une v. a. continue X_i , $i \in \{1, 2, ..., n\}$

Méthode de composition

Algorithme

1. Générer $u_1 \in \mathcal{U}[0,1]$;
2. Générer la valeur de N issue de la distribution $\{p_i\}$ poser $N = j$ pour $\sum_{i=1}^{j-1} p_i < u_1 \leq \sum_{i=1}^{j} p_i$;
3. Générer $u_2 \in \mathcal{U}[0,1]$;
4. Générer une valeur de X_j (si $N = j$) à partir de la loi ayant la densité $g_N(x) = g_j(x)$ à l'aide d'un algorithme approprié (inversion ou autre)

$$X_N = X_j = F_j^{-1}(u_2), \text{ où } F_j(x) = \int_{-\infty}^{x} g_j(y)dy$$

5. $X = X_j$;

Méthodes spécifiques

Loi binomiale

Soit $X \sim \mathcal{B}(n, p)$, alors

$$X = \sum_{i=1}^{n} X_i, \text{ telle que } X_i = \begin{cases} 1, & \text{avec la probabilité } p \\ 0, & \text{avec la probabilité } 1-p \end{cases}$$

c'est-à-dire $X_i \sim \mathcal{B}(p)$.
Pour générer X, il suffit de simuler n v.a. i.i.d $\in \mathcal{B}(p)$.

Algorithme

1. Poser $B_0 = 0$;
2. Pour $i = 1, n$ faire générer $u_i \in \mathcal{U}[0, 1]$.

$$B_i = \begin{cases} B_{i-1} + 1 \text{ si } u_i < p; \\ B_{i-1} \text{ si } p \leq u_i < 1. \end{cases}$$

3. Poser $X = B_n$;

Méthodes spécifiques

Loi de Poisson

Soit X à générer tq : $X \sim \mathcal{P}(\lambda)$

$$p_k = \mathbb{P}(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}, \forall k \in \mathbb{N}.$$

les valeurs p_k satisfont la relation de récurence suivante

$$p_{k+1} = e^{-\lambda}\frac{\lambda^{k+1}}{(k+1)!} = \frac{\lambda}{k+1}p_k, \forall k \geq 0, \text{ avec } p_0 = e^{-\lambda}.$$

Algorithme

1. Poser $k = 0, p = e^{-\lambda}, F = p$;
2. Générer $u \in \mathcal{U}[0,1]$.
3. Tant que $u > F$ poser $p = \frac{\lambda}{k+1}p, F = F + p$ et $k = k + 1$.
4. Poser $X = k$.

Méthodes spécifiques

Loi Normale

Soit $X \sim \mathcal{N}(m_X; \sigma_X)$ de densité de probabilité

$f(x) = \frac{1}{\sigma_X \sqrt{2\pi}} e^{-\frac{(x-m_X)}{2\sigma_X^2}}$, $\forall x \in \mathbb{R}$, $\mathbb{E}[X] = m_X$ et $Var(X) = \sigma_X^2$.

Alors $Z = \frac{X - m_X}{\sigma_X} \sim \mathcal{N}(0; 1)$.

1ère méthode

D'après le théorème de la centrale limite, si $u_1, u_2, \cdots, u_n$ sont des v.a.i.i.d. de même loi de probabilité avec $\mathbb{E}[u_i] = m$ et $Var(u_i) = \sigma^2, \forall i = 1, \cdots, n.$ Alors

$$\lim_{n \to \infty} \mathbb{P}\left(a \leq \frac{\sum_{i=1}^{n} u_i - nm}{\sigma \sqrt{n}} \leq b \right) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-\frac{z^2}{2}} dz$$

où $\mathbb{E}\left[\sum_{i=1}^{n} u_i\right] = nm$, $Var\left(\sum_{i=1}^{n} u_i\right) = n\sigma^2$ et $Z = \frac{\sum_{i=1}^{n} u_i - nm}{\sigma \sqrt{n}} \sim \mathcal{N}(0; 1)$.

Méthodes spécifiques

Loi Normale

C'est-à-dire une v.a. normale peut être simulée en utilisant la somme de n v.a. $\{u_i\}_{i=1}^{n} \in \mathcal{U}[0,1]$.
Comme : $\mathbb{E}[u_i] = 0,5$ et $Var(u_i) = \frac{1}{12}, \forall i = 1, \cdots, n.$ Alors

$$Z = \frac{\sum_{i=1}^{n} u_i - \frac{n}{2}}{\sqrt{\frac{n}{12}}} = \frac{X - m_X}{\sigma_X},$$

d'où

$$X = \left(\sum_{i=1}^{n} u_i - \frac{n}{2}\right) \sigma_X \sqrt{\frac{12}{n}} + m_X, \forall i = 1, n.$$

Remark

De préférence il faut que n soit grand (en général $n \geq 50$); mais pour des résultats satisfaisants on prend $n = 12$.

Méthodes spécifiques

L'approche de Box et Muller pour la loi normale

Il s'agit d'une méthode populaire pour générer des variables gaussiennes. Son avantage réside dans sa simplicité mais la qualité des réalisations dépend de celles des variables uniformes. Elle permet d'engendrer une paire indépendante de variables aléatoires Z_1, Z_2 obéissant chacune à une loi normale centrée réduite. Puis, nous pouvons avoir deux réalisations de la variable X réparties selon $\mathcal{N}(m_X, \sigma_X)$ à l'aide de $X_1 = \sigma_X Z_1 + m_X$ et $X_2 = \sigma_X Z_2 + m_X$.

Soient U_1 et U_2 deux variables aléatoires indépendantes générées selon une loi uniforme sur $(0, 1)$. Alors,

$$\begin{aligned} Z_1 &= (-2\ln(U_1))^{\frac{1}{2}} \cos(2\pi U_2) \text{ et} \\ Z_2 &= (-2\ln(U_1))^{\frac{1}{2}} \sin(2\pi U_2). \end{aligned}$$

Génération d'une loi particulière

L'approche de Box et Muller pour la loi normale

Algorithme

1. Générer u_1 et u_2 selon des lois uniformes indépendantes sur $(0,1)$.
2. Calculer z_1 et z_2 en utilisant les formules suivantes :

$$\begin{aligned} Z_1 &= (-2\ln(u_1))^{\frac{1}{2}}\cos(2\pi u_2), \\ Z_2 &= (-2\ln(u_1))^{\frac{1}{2}}\sin(2\pi u_2). \end{aligned}$$

Génération de vecteurs aléatoires

Supposons que nous devions générer un vecteur aléatoire $X = (X_1, X_2, ..., X_n)$ à partir de n distributions données de densité de probabilité $f(x)$ et de fonction de répartition $F(x)$.
Lorsque les composantes $X_1, X_2, ..., X_n$ sont indépendantes, la situation est simple : il suffit d'appliquer la méthode de l'inverse transformation ou une autre méthode de génération de notre choix à chaque composante individuellement.

Génération de vecteurs aléatoires

Algorithme du conditionnement

Si $X_1, ..., X_n$ sont dépendantes. En vertu de la formule des probabilités conditionnelles, on a

$$f(x_1, x_2, ..., x_n) = f_1(x_1) \times f_2(x_2/x_1) \times ... \times f_n(x_n/x_1, ..., x_{n-1})$$

où $f_1(x_1)$ est la densité marginale de X_1 et $f_i(x_i/x_1, ..., x_{i-1})$ sont les densités conditionnelles de X_i sanchant $X_1 = x_1, ..., X_{i-1} = x_{i-1}$. D'où l'algorithme suivant

Algorithme

1. Tirer $u_1 \in \mathcal{U}[0, 1]$;
2. Générer X_i de densité $f_i(x_i/x_1, ..., x_{i-1})$,
3. Si $i < n$, aller en 1;
4. Ecrire $X_1, X_2, ..., X_n$.

Génération de vecteurs gaussiens multivariés

Si $X = (X_1, X_2, ..., X_n) \sim \mathcal{N}(m, C)$;

$$f(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |C|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-m)^T C^{-1}(x-m)\right) \qquad (14)$$

$$C \text{ est } \begin{cases} \text{Symétrique} \\ \text{Définie positive} \end{cases} \qquad (15)$$

L'idée pour générer un vecteur aléatoire normal multivarié (ou simplement multinormal) est de l'écrire sous la forme $Z = m + AZ$, où A est une matrice telle que $AA^T = C$, et Z est un vecteur de variables aléatoires i.i.d. $\mathcal{N}(0, I)$.

Génération de vecteurs gaussiens multivariés

Décomposition de Cholesky

Algorithme

Entrée : Vecteur moyen m et matrice de covariance C.
sortie : Vecteur aléatoire $X \sim N(m, C)$.
Donnée : la décomposition de Cholesky AA^T de C.

1. Générer $Z = (Z_1, Z_2, ..., Z_n)$ de loi $\mathcal{N}(0, I)$;
2. Poser $X = m + AZ$.

Génération de points dans la sphère

Un vécteur aléatoire uniformément distribuée sur la sphère $S_n = \{x \in \mathbb{R}^n : \|x\| = r\}$ de rayon r, centrée à l'origine, possède la fonction densité

$$f(x/r) = \begin{cases} \frac{\Gamma\left(\frac{n}{2}\right)}{(2\pi)^{\frac{n}{2}} r^{n-1}}, x \in S_n \\ 0, \text{ ailleurs} \end{cases}.$$

Cette densité ne dépend que de r et de n et est indépendante de la position exacte du point x dans $S_n(r)$.

Génération de points dans la sphère

Par passage aux coordonées sphériques, cela nous permet de générer un point aléatoire uniforme sur $S_n(r)$ en normant un vecteur $Z \sim \mathcal{N}(0, I)$.

Algorithme

1. Générer $Z = (Z_1, Z_2, ..., Z_n)$ de loi $\mathcal{N}(0, I)$;
2. Calculer $\|Z\| = \sqrt{Z_1^2 + ... + Z_n^2}$;
3. Poser $X = r\left(\frac{Z}{\|Z\|}\right)$.

Génération de points dans la Boule

Un vécteur aléatoire uniformément distribuée sur la boule $B_n = \{x \in R^n : \|x\| \leq r\}$ de rayon r, centrée à l'origine, possède une norme euclidienne dont la fonction de densité est $\varphi(y) = \frac{ny^{n-1}}{r^n}, 0 \leq y \leq r$.
Ainsi en utilisant la densité $f(x/r)$ définie pour une sphère, la densité conjointe d'un point x, de norme y, uniforme dans la boule $B_n(r)$ est

$$f(x,y) = f(x/r)\,\varphi(y) = \begin{cases} \frac{\Gamma\left(1+\frac{n}{2}\right)}{(2\pi)^{\frac{n}{2}} r^n}, x \in B_n(y) \text{ et } 0 \leq y \leq r \\ 0, \text{ sinon} \end{cases}.$$

Génération de points dans la Boule

L'algorithme suivant permet d'obtenir un point en générant dans un premier temps une norme aléatoire Y selon la densité $\varphi(y)$ puis en générant un vecteur aléatoire X uniforme sur $S_n(r)$:

Algorithme

1. Générer $Z \in \mathcal{N}(0, I)$ et poser $Y = rZ^{\frac{1}{n}}$;
2. Générer X uniforme sur $S_n(Y)$, avec l'algorithme précédent.

Simulation de chaîne de Markov à temps discret

Soit une chaîne de Markov $\{X_n, n \geq 0\}$ à espace d'états fini $S = \{1, 2, ..., n\}$ (ou dénombrable) de matrice de transition $P = (p_{ij})_{n \times n}$, où

$$p_{ij} = \mathbb{P}\left(X_n = j / X_{n-1} = i\right), i = 1, ..., n$$

et de distribution initiale $\pi_i^0 = \mathbb{P}\left(X_0 = i\right)$ avec $\sum_{i=1}^{n} \pi_i^0 = 1$.

Simulation de chaîne de Markov à temps discret

Algorithme

1. Partager l'intervalle $[0,1]$ de deux manière :

   a) Pour générer la distribution de l'état initial :

   $[0,1] = \bigcup_{i=1}^{n} \Delta_i = \sum_{i=1}^{n} \Delta_i$; la mesure de l'intervalle

   $\left|\Delta_i^0\right| = \pi_i^0,\ i = 1,..,n; \Delta_i = \sum_{k=1}^{i} \Delta_k^0$ (c'est la fonction de répartition).

   b) Pour générer l'état courant j sachant l'état précédent i

   $[0,1] = \bigcup_{i=1}^{n} \Delta_{ij} = \sum_{i=1}^{n} \Delta_{ij}$; la mesure de l'intervalle

   $|\Delta_{ij}| = p_{ij},\ i = 1,..,n; \Delta_{ij} = \sum_{k=1}^{j} \Delta_{ik}.$

Simulation de chaîne de Markov à temps discret

Algorithme

2. On détermine les réalisations successives de la chaîne de Markov $\{X_n, n \geq 0\}$ à l'aide des formules récurrentes :
   $X_0 = i$, si $u_0 \in \Delta_i$;
   $X_n = j$, si $u_n \in \Delta_{X_{n-1},j}$; $n \geq 1$.

Simulation de chaîne de Markov à temps discret

Example

Simuler la DTMC dont la matrice de transition est

$$P = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{pmatrix}$$

et la distribution initiale $\pi^0 = (0.819, 0.177, 0.008)$, $S = \{1, 2, 3\}$
on donne la suite $\{u_i\} = \{0.048, 0.841, 0.953, 0920\}$

Simulation de chaîne de Markov à temps continu

Soit un processus homogène de Markov $\{X(t), t \geq 0\}$ à espace d'état discret ou dénombrable S. Les prbabilités de transition $p_{ij}(t) = \mathbb{P}(X(s+t) = j / X(s) = i)$, et la distribution initiale $p_i^0 = P(X(0) = i)$ avec $\sum_i p_i^0 = 1$. Si les dérivées existent, on définit les taux de transition

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

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 est la suivante :

Simulation de chaîne de Markov à temps continu

Soit i_0 l'état initial déterminé par la distribution $\left\{p_i^0\right\}$; la durée de séjour à cet état est une v.a. τ_0 de loi exponentielle $\exp\left(\lambda_{i_0}\right)$:

$$\mathbb{P}\left(\tau_0 < x\right) = 1 - e^{-\lambda_{i_0} x}, x \geq 0.$$

La probabilité pour que l'état suivant soit i_1 est

$$\mathbb{P}\left(X\left(\tau_0\right) = i_1 / X\left(0\right) = i_0\right) = \frac{\lambda_{i_0,i_1}}{\lambda_{i_0}} = p_{i_0,i_1}.$$

Le processus séjourne à cet état un temps τ_1 de loi exponentielle $\exp\left(\lambda_{i_1}\right)$:

$$\mathbb{P}\left(\tau_1 < x\right) = 1 - e^{-\lambda_{i_1} x}, x \geq 0.$$

Simulation de chaîne de Markov à temps continu

De nouveau,

$$\mathbb{P}\left(X\left(\tau_0+\tau_1\right)=i_2/X\left(\tau_0\right)=i_1\right)=\frac{\lambda_{i_1,i_2}}{\lambda_{i_1}}=p_{i_1,i_2}$$

et

$$\mathbb{P}\left(\tau_2<x\right)=1-e^{-\lambda_{i_2}x},x\geq 0,\ \text{etc...}$$

En d'autres termes, les états pris par le processus $\left\{X\left(t\right),t\geq 0\right\}$ au cours des transitions successives forment une chaîne de Markov $\left\{X_n\right\}$, définie par $X_n=X\left(\tau_0+\tau_1+...+\tau_{n-1}\right)$ de matrice de transtion $P=\left(p_{ij}\right)$, où $p_{ij}=\frac{\lambda_{ij}}{\lambda_i}$ et de distribution initiale $\left\{p_i^0\right\}$.
La varaiable τ_i désigne la durée de séjour de la chaîne à l'état i

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

Simulation de chaîne de Markov à temps continu

La simulation du processus $\{X(t), t \geq 0\}$, consiste donc à simuler la chaîne de Markov induite

$$\{X_n, n \geq 1\},$$

à temps discret, où $X_n = X(t_n)$.

Simulation de chaîne de Markov à temps continu

Algorithme

1. Engendrer $\{u_n\} \in \mathcal{U}[0,1]$.
2. Générer l'état initial i_0 selon la distribution $\{p_i^0\} : X_0 = i$ si $u_0 \in \Delta_i$;
3. Engendrer la durée de séjour à cet état : $\tau_0 = -\frac{1}{\lambda_{i_0}} \ln u_0$.
4. Les états suivants sont ainsi générés par les termes pairs de la suite :
   $X_n = j$ si $u_{2k} \in \Delta_{n-1,j}, n \geq 1$;
   $X_n = X(\tau_0 + \tau_1 + ... + \tau_{n-1})$.
5. Engendrer les durées de séjour successives selon les lois exponentielles correspondantes pou tous les termes impairs :
   $\tau_n = -\frac{1}{\lambda_{X_n}} \ln u_{2k+1}, n \geq 1$.

Simulation de chaîne de Markov à temps continu

Le processus considéré est alors défini par la relation

$$X(t) = X_n,$$

pour

$$\tau_0 + \tau_1 + ... + \tau_{n-1} \leq t \leq \tau_0 + \tau_1 + ... + \tau_n.$$

Cette Méthode est souvent appelée : "simulation événement par événement" ou "simulation à événement discrets".

Simulation de chaîne de Markov à temps continu

Example

Considérons un système où les clients arrivent selon un taux λ clients/unité de temps (les durées entre deux arrivées successives suivent une loi $\exp(\lambda)$). Soit le taux de service μ (les durées de service suivent une loi $\exp(\mu)$). Soit $\{X(t), t \geq 0\}$, où $X(t)$ est le nombre de clients présents dans le système à l'instant t. Il s'agit d'une CTMC. On distingue deux types de sauts :

"$+1$ correspond à l'arrivée d'un client, avec une probabilité $\frac{\lambda}{\lambda+\mu}$" : événement A

"-1 correspond à l'arrivée d'un client, avec une probabilité $\frac{\mu}{\lambda+\mu}$" : événement B

Présenter l'algorithme de simulation. Faire les applications avec $\lambda = 0.3$ et $\mu = 0.5$.