Fiche 2 – Génération de variables pseudo-aléatoires

«Nul n'est censé ignorer la loi.» Principe de droit français.

1 Introduction

L'objet de la fiche précédente était d'expliquer comment on pouvait simuler, à l'aide d'un ordinateur, des suites de nombres réels pouvant être considérées en pratique comme des suites de variables aléatoires uniformément distribuées entre 0 et 1. Nous nous intéressons dans cette fiche à l'étape de transformation, qui consiste à simuler des variables aléatoires de loi donnée, non-uniforme en général, à partir des nombres produits par un générateur pseudo-aléatoire du type décrit dans la fiche précédente.

1.1 Méthodes génériques et méthodes spécifiques

Pour la plupart des lois classiques telles que les lois gaussiennes, binomiales, exponentielle, géométrique, de Poisson, Gamma, Beta, etc... de nombreux algorithmes performants, souvent très astucieux, optimisés dans leur implémentation, et reposant sur les propriétés particulières de ces lois et sur les relations qui existent entre elles, ont été développés. Ces algorithmes spécifiques sont implémentés dans divers logiciels (tels que R) et bibliothèques (tels que la bibliothèque GSL (GNU Scientific Library) [G], en C et C++). Nous en donnerons quelques exemples, non pas à titre de référence, mais plutôt d'illustration du type de propriétés sur lesquelles ces méthodes sont basées.

En revanche, nous nous attacherons dans cette fiche à décrire précisément les méthodes génériques de transformation, qui permettent de simuler des variables aléatoires de loi quelconque, et ne reposent pas comme les précédentes sur les propriétés particulières des lois classiques.

Pour plus de détails, nous renvoyons (encore et toujours) à l'ouvrage de D. Knuth [K], ainsi qu'à celui de L. Devroye [D]

(voir également la page de L. Devroye `http://cgm.cs.mcgill.ca/~luc/` pour des références récentes.)

1.2 Validité des méthodes de transformation

Les méthodes que nous présentons dans la suite supposent que l'on dispose de variables aléatoires indépendantes $U_1, U_2, \ldots$ de loi uniforme sur $[0, 1]$, et expliquent comment transformer ces variables afin de produire une variable aléatoire, disons X, possédant la loi souhaitée. La validité de ces méthodes est à chaque fois établie en prouvant mathématiquement que, si l'on dispose en entrée de véritables variables aléatoires i.i.d. uniformes sur $[0, 1]$, la variable aléatoire X produite par la méthode considérée possède effectivement la loi de probabilité souhaitée. Pour produire plusieurs exemplaires indépendants de même loi que X, on répète le procédé en employant en entrée des tronçons disjoints de la suite de nombres pseudo-aléatoires pseudo-uniformes produite par le générateur utilisé[1].

On voit donc que la validité d'une telle méthode, c'est-à-dire le fait que l'on puisse en pratique manipuler les nombres qu'elle produit comme des réalisations indépendantes de variables aléatoires possédant la loi voulue, repose fondamentalement sur la validité du procédé de génération de nombres pseudo-aléatoires employé, celle-ci étant censée avoir été établie avant son utilisation. Toutes les remarques concernant les précautions et les difficultés associées à la génération et à l'utilisation de nombres pseudo-aléatoires s'appliquent ici. Nous distinguerons plusieurs critères de validité des méthodes présentées :

- le premier critère consiste simplement à prouver mathématiquement que la méthode fonctionne si on lui fournit en entrée de véritables variables aléatoires i.i.d. uniformes sur $[0, 1]$; il s'agit c'est un critère minimal de validité, sans lequel l'emploi d'une méthode de transformation paraît difficilement envisageable ;
- le deuxième critère consiste à prouver que la méthode demeure correcte lorsque l'on prend en compte le caractère discret des nombres utilisés en entrée (et éventuellement les autres problèmes liés à l'utilisation de calculs en précision

[1] il s'agit d'une remarque importante : dans ce qui suit, nous décrirons souvent des procédés algorithmiques permettant de générer **un** nombre pseudo-aléatoire de loi donnée en partant d'un ou plusieurs (le nombre sera parfois lui-même aléatoire) nombres pseudo-uniformes entre 0 et 1 produits par le générateur, et, si l'on désire simuler des suites indépendantes de nombres de loi donnée, il est indispensable de vérifier que les procédés algorithmiques sont appliqués à des blocs **disjoints** de nombres issus du générateur, ou, autrement dit, que les nombres pseudo-aléatoires issus du générateur ne sont utilisés qu'une seule fois chacun.

finie) : au mieux, on peut en réalité espérer disposer de variables aléatoires i.i.d. uniformes, non pas sur $[0, 1]$, mais sur un ensemble de la forme

$$\left\{\frac{0}{K}, \frac{1}{K}, \ldots, \frac{K}{K}\right\};$$

- le troisième critère consiste à prendre en compte explicitement le générateur utilisé, et à prouver dans ce contexte que la méthode se comporte de manière satisfaisante ; on doit alors étudier comment les propriétés connues du générateur peuvent influencer les propriétés des nombres obtenus par transformation ; à ce sujet, on peut au moins retenir que les garanties de qualité des générateurs de nombres pseudo-aléatoires portent principalement sur la qualité de la répartition des bits de poids fort, – les bits de poids faible présentant souvent un assez mauvais comportement ;
- le quatrième critère est un critère (indispensable) de validité empirique : on vérifie à l'aide de tests statistiques que les nombres fournis par la méthode (lors d'un grand nombre d'utilisations supposées indépendantes) sont compatibles avec l'hypothèse selon laquelle ceux-ci forment une suite de variables aléatoires indépendantes possédant la loi souhaitée.

En général, le premier et le quatrième critère, et éventuellement le deuxième, sont étudiés. Soulignons la difficulté d'étude du troisième, la structure des appels au générateur effectués par l'utilisation répétée de méthodes de transformation au cours d'une simulation pouvant varier de manière fort complexe et arbitraire. Le minimum vis-à-vis du quatrième critère est d'examiner graphiquement la répartition empirique (par exemple, au moyen d'un histogramme) des valeurs obtenues en appliquant la méthode de manière répétée, et de pratiquer des tests standards d'adéquation. Bien entendu, on ne teste (quasiment) pas ainsi l'indépendance des valeurs successives obtenues, qui est également une propriété très importante. De plus, il n'est bien entendu pas clair que des tests empiriques réussis avec une certaine implémentation (c'est-à-dire avec une sous-suite donnée de la suite produite par le générateur) garantissent que la méthode fonctionne en général, même si notre confiance en la méthode en est renforcée.

Au passage, notons que la vérification empirique de l'adéquation des résultats obtenus à la loi recherchée permet non seulement de mettre à l'épreuve la validité pratique de la méthode décrite et de son association au procédé de génération utilisé, mais également de tester le fait que l'implémentation effectuée correspond bien à la

méthode proposée et ne comporte pas d'erreurs.

1.3 Validité et efficacité

La validité d'une méthode de transformation n'est pas le seul élément à prendre en compte pour décider de son utilisation. Son efficacité, mesurée la plupart du temps en termes de temps d'exécution et d'espace mémoire, est également importante, en particulier lorsque l'on mène des simulations massives au cours desquelles un très grand nombre de variables aléatoires doit être simulé.

2 Un exemple très simple

Supposons que l'on cherche à produire des variables aléatoires de loi uniforme sur l'ensemble des entiers de 0 à H, où H est un entier positif fixé.

Partant d'un nombre aléatoire U uniforme sur $[0, 1]$, on obtient un entier V distribué uniformément entre 0 et H en posant :

$$V \leftarrow \lfloor (H+1) \times U \rfloor ,$$

$\lfloor x \rfloor$ désignant la partie entière de x, c'est-à-dire le plus grand entier inférieur à x.

On vérifie le premier critère de validité de la méthode en calculant, pour tout entier k tel que $0 \leq k \leq H$, la probabilité $P(V = k)$

$$P(V = k) = P(l \leq (H+1)U < k+1)$$

$$= P(l/(H+1) \leq U < (k+1)/(H+1)) = 1/(H+1).$$

Comme nous l'avons souligné précédemment, U ne peut pas suivre exactement la loi uniforme sur l'intervalle $[0, 1]$.

Quant au deuxième critère, en admettant que U suive exactement la loi uniforme sur l'ensemble discret $\left\{ \frac{0}{K}, \frac{1}{K}, \ldots, \frac{K}{K} \right\}$, on constate que la probabilité pour que pour que $\lfloor (H+1) \times U \rfloor$ soit égal à i peut ne pas être exactement égale à $1/H$, mais présenter un écart par rapport à cette valeur de l'ordre de $1/K$. Le nombre K étant en général supposé extrêmement grand, ceci ne pose pas nécessairement de problème sérieux.

On pourrait envisager une autre méthode en utilisant le générateur pour produire un entier pseudo-aléatoire pseudo-uniforme entre 0 et un grand entier M, par

exemple, en se rappelant que U peut être vu comme un entier pseudo-aléatoire uniforme (il s'agit simplement d'utiliser la même séquence de bits, mais en l'interprétant de manière différente) entre 0 et un grand entier RAND_MAX, et choisir V en réduisant X modulo $H+1$. Cependant, cette stratégie n'est pas à recommander car elle fait intervenir les bits de poids faible de X, alors que les (relatives) garanties de qualité des générateurs de nombres pseudo-aléatoires portent sur l'équirépartition des bits de poids fort,– les bits de poids faible présentant souvent un comportement moins bon. Le troisième critère nous permet ici à disqualifier une telle méthode.

3 Variables aléatoires de loi discrète

On suppose donc que l'on dispose d'un ensemble S fini ou dénombrable, que l'on met sous la forme $\{x_i,\ i \in I\}$, où I est un ensemble d'indices de la forme $I = \{1, \ldots, k\}$ ou $I = \{1, 2, \ldots\}$, ainsi que d'une liste de nombres $(p(x_i))_{i\in I}$, positifs ou nuls et vérifiant $\sum_{i\in I} p(x_i) = 1$.

Notre objectif est de générer un élément aléatoire V de S vérifiant $PV = x_i = p(x_i)$ pour tout $i \in I$. Pour alléger les notations, nous poserons dans la suite $p_i = p(x_i)$.

3.1 Méthode de découpage d'intervalles

En partant d'un nombre aléatoire U de loi uniforme sur $[0, 1[$, il suffit de poser : $V = x_i$, où i est l'unique élément de I vérifiant $\sum_{j=1}^{i-1} p_j \leq U < \sum_{j=1}^{i} p_j$.

L'intervalle $[0, 1]$ est ainsi découpé en intervalles de longueurs p_i, et l'indice de l'intervalle dans lequel tombe la variable U fournit l'indice de l'élément de S que l'on renvoie.

La principale difficulté de cette méthode est liée à la recherche de i à partir de U. La méthode la plus naïve consiste à tester d'abord si $U < p_1$, puis, si ce n'est pas le cas, si $U < p_1 + p_2$, et ainsi de suite jusqu'à obtenir i. Les probabilités cumulées $p_1 + \cdots + p_k$ peuvent éventuellement être précalculées, afin de gagner du temps. D'autre part, en plaçant d'abord les intervalles les plus larges, (ce qui nécessite également un prétraitement de la loi à simuler), on peut réduire le nombre moyen de comparaisons à effectuer. D'autres méthodes de recherche, telles que par exemple la dichotomie, peuvent également être employées. En outre, si les p_i vérifient des propriétés particulières, la recherche de i peut être grandement accélérée : c'est

exactement le cas de la méthode décrite dans le paragraphe précédent pour générer des entiers uniformes entre 0 et H.

De manière générale, on retient que l'efficacité de la méthode est conditionnée par l'efficacité avec laquelle il est possible de déterminer l'intervalle contenant U. De plus, quitte à effectuer un prétraitement de la loi à simuler, il peut être possible de diminuer le temps moyen nécessaire à la simulation, ce qui peut être intéressant lorsque l'on cherche à simuler un grand nombre de variables aléatoires possédant la loi recherchée (bien entendu, le prétraitement n'est en général pas rentable si l'on ne simule qu'une seule variable aléatoire). D'autre part, des propriétés particulières des p_i peuvent s'avérer très utiles pour accélérer le procédé.

3.2 Méthode du rejet

La méthode du rejet repose sur deux observations, dont la première est la suivante : il est équivalent de pouvoir générer une variable aléatoire V vérifiant $P(V = x_i) = p(x_i)$ pour tout $i \in I$, ou une variable aléatoire (V, W) de loi uniforme sur l'ensemble $A = \bigcup_{i=1}^{k} \{x_i\} \times [0, d \times p(x_i)]$, où d est une constante positive arbitraire. La preuve (immédiate) est la suivante : si (V, W) est uniformément distribuée sur un ensemble tel que A, la probabilité pour que $V = x_i$ est proportionnelle à $dp(x_i)$, par définition de la loi uniforme, et donc égale à $p(x_i)$ (car nous avons affaire à des probabilités, dont la somme est obligatoirement égale à 1). Inversement, si V vérifie $P(V = x_i) = p(x_i)$ pour tout i, on vérifie que la variable aléatoire $(V, U \times dp(V))$, où U est uniforme sur $[0, 1]$ et indépendante de V, suit la loi uniforme sur A.

Le seconde observation est que, si l'on est en mesure de simuler des variables aléatoires uniformes à valeurs dans un ensemble B contenant A, il est possible de simuler par rejet des variables aléatoires uniformes sur A. Précisément : si (V, W) est une variable aléatoire de loi uniforme sur un ensemble B qui contient l'ensemble A, la loi de (V, W) conditionnelle au fait que $(V, W) \in A$ est la loi uniforme sur A. Ceci n'est pas lié à la forme spécifique de l'ensemble A, il s'agit d'une propriété tout-à-fait générale qui a lieu pour tout (ou presque) couple d'ensembles A et B vérifiant $A \subset B$ et sur lesquels on peut définir la loi uniforme.

Ainsi, il est possible de simuler des variables aléatoires uniformes sur A en simulant des variables aléatoires uniformes indépendantes à valeurs dans B jusqu'à obtenir une variable se trouvant dans A, dont la loi est, d'après ce qui précède, uni-

forme sur A. L'idée est qu'il peut être beaucoup plus facile de générer des variables uniformément distribuées sur B plutôt que directement sur A.

En d'autres termes, l'algorithme suivant renvoie une variable aléatoire (V, W) uniformément distribuée sur A :

1) Générer (V, W) uniformément distribuée sur B

2) si $(V, W) \in A$, renvoyer$V$

3) sinon retourner à l'étape 1)

On notera que les générations successives de (V, W) sont supposées indépendantes.

Le nombre de tentatives nécessaires jusqu'à obtention d'un résultat suit une loi géométrique de paramètre $L(A)/L(B)$, où $L(\cdot)$ désigne la mesure de Lebesgue dans la dimension appropriée (la longueur pour des objets de dimension 1, la surface pour des objets de dimension 2, etc...).

Un exemple simple, que l'on peut mettre en œuvre dans le cas où l'ensemble S est fini, est celui où l'on prend pour B l'ensemble $S \times [0, M]$, où $M = \max_i p_i$, dans lequel il suffit de simuler indépendamment deux coordonnées indépendantes et uniformes.

Il est alors facile de générer des variables aléatoires uniformément dans B, puisqu'il suffit de poser $V = x_{\lceil U_1 \times k \rceil}$ et $W = U_2 \times M$, où U_1 et U_2 sont deux variables aléatoires indépendantes et de loi uniforme sur $[0, 1[$.

De manière synthétique, la méthode du rejet s'écrit donc, dans ce cas particulier, de la manière suivante :

1) Générer+ V distribuée uniformément sur S

2) Générer W uniformément distribuée sur $[0, M]$

3) si $W < p(V)$, renvoyer V

4) sinon retourner à l'étape 1)

On notera que le nombre d'étapes de la méthode, comme d'ailleurs dans le cas du découpage d'intervalles, est lui-même une variable aléatoire, susceptible de varier d'un appel à l'autre. Le nombre de comparaisons effectuées lors d'un passage est limité à 1 (on compare T et $p(x_L)$), mais il faut le multiplier par le nombre de passages effectués jusqu'à obtenir une acceptation. Naturellement, plus la taille de l'ensemble A, au sens de la mesure de Lebesgue, est petite par rapport à celle de l'ensemble B dans lequel on l'inscrit, plus le nombre de rejets est important, et plus l'exécution de l'algorithme est coûteuse en temps. Il est donc souhaitable que

l'ensemble B «colle» au plus près l'ensemble A, même si cette exigence est en général contradictoire avec le fait qu'il doit être facile de tirer uniformément des points de B. Dans certains cas, il est possible de trouver mieux qu'un ensemble de barres de hauteurs toute égales comme ensemble B. Par exemple, lorsque l'on sait facilement générer des variables aléatoires dont la loi, sans être identique à celle souhaitée pour V, s'en rapproche. Plus précisément, supposons que l'on sache facilement générer des variables aléatoires W à valeurs dans l'ensemble S, et dont la loi $q(x_i) = P(W = x_i)$ est telle que, pour une certaine constante $c > 0$, on ait pour tout $i \in I$, l'inégalité $p(x_i) \leq c \times q(x_i)$. On peut alors inscrire l'ensemble A dans l'ensemble

$$B = \bigcup_{i=1}^{k} \{x_i\} \times [0, c \times q(x_i)],$$

et, conformément à l'observation précédente, il est facile de produire des éléments de (V, W) de B uniformément distribués à l'aide de la méthode suivante

1) Générer V uniformément dans S

2) Générer+ W uniformément distribuée sur+ $[0, c \times q(V)]$

(Exercice : vérifier que le couple (W, T) défini ci-dessus possède bien la loi uniforme sur l'ensemble B.)

On obtient ainsi une nouvelle méthode du rejet pour générer des variables aléatoires de loi $P(V = x_i) = p(x_i)$:

1) Générer V uniformément dans S

2) Générer W uniformément distribuée sur $[0, c \times q(V)]$

3) si+ $W < p(V)$, renvoyer V

4) sinon retourner à l'étape 1)

On constate que le nombre de rejets sera d'autant plus faible que l'approximation $p(x) \sim cz(x)$ est précise.

Une remarque qui a son importance est que l'on peut, en fait, se contenter de manipuler les probabilités p et q à une constante multiplicative près, ce qui est utile lorsque le calcul des constantes de normalisations de p ou de q est difficile ou coûteux, et présente en particulier un intérêt lorsque l'on manipule des lois conditionnelles. Supposons donc que $p(z) = \beta p_1(z)$, et $q(z) = \gamma q_1(z)$, où p_1 et q_1 sont des fonctions plus simples à calculer que p et q, et que l'on ait une relation de la forme $cq_1 \geq p_1$, où c est connue explicitement. Alors, en générant V selon la loi q et W selon la loi uniforme sur $[0, cq_1(V)]$, on obtient un point (V, W) uniformément distribué sur

l'ensemble $\bigcup_{i=1}^{k}\{x_i\} \times [0, c \times q_1(x_i)]$. Conditionnellement au fait que $W \leq p_1(V)$, V est alors distribué selon la loi p, ce que l'on voit grâce aux mêmes arguments que précédemment. L'algorithme correspondant ne nécessite pour fonctionner que la connaissance de p_1 et q_1, et son efficacité repose sur le fait que p_1 et cq_1 soient relativement proches.

Méthode de Walker

Une méthode plus élaborée pemettant de limiter considérablement le temps d'exécution est la méthode de Walker, aussi connue sous le nom de «méthode des alias», qui suppose un prétraitement de la loi à simuler (voir l'article original [W]), et permet un gain de temps considérable lorsque l'on cherche à produire un grand nombre de variables pseudo-aléatoires distribués suivant une loi discrète fixée. Intuitivement, l'idée est de tronçonner puis recoller les barres qui constituent l'ensemble A défini précédemment, de manière à obtenir un ensemble de barres de hauteur toutes égales, en limitant à deux le nombre de tronçons de provenance différente dans chaque barre ainsi formée, le point important étant que le tronçonnage/recollement conserve la surface des morceaux déplacés. Plus formellement, la méthode de Walker repose sur l'utilisation d'une table de la forme suivante (pour alléger les notations, rappelons que l'on pose $p_i = p(x_i)$).

état	1	2	...	$k-1$	k
seuil	q_1	q_2	...	q_{k-1}	q_k
alias	l_1	l_2	...	l_{k-1}	l_k

vérifiant, pour tout $1 \leq i \leq k$, les hypothèses :

- $0 \leq q_i \leq 1$
- $l_i \in \{1, \ldots, k\}$
- $q_i + \sum_{j\ :\ l_j = i}(1 - q_j) = kp_i$

Pour simuler la loi en question, on procède de la façon suivante :

1) Générer un entier L uniformément entre 1 et k

2) Générer un réel U uniformément dans l'intervalle $[0, 1]$

3) si $U < q_L$, renvoyer $V \leftarrow x_L$

4) sinon, renvoyer la valeur «alias» $V \leftarrow x_{l_L}$

Interprétation géométrique de la méthode : dessin au tableau.

On note que la méthode de Walker constitue une amélioration considérable de la méthode du rejet (moyennant le calcul préliminaire de la table) : au lieu de rejeter

le résultat obtenu dans le cas où $U \geq p_L$ et de recommencer un nouveau tirage, on renvoie dans ce cas la valeur alias figurant dans la table.

Le nombre d'opérations requis par l'emploi de cette méthode ne dépend donc pas, une fois qu'une telle table est construite, de la valeur de k. Lorsque k est grand, et que l'on est amené à répéter un grand nombre de fois la simulation de la loi considérée, il est donc très avantageux d'utiliser cette méthode.

Bien entendu, une question importante est de savoir comment fabriquer la table de telle façon que la loi obtenue soit bien celle que l'on cherche à simuler.

L'algorithme suivant permet, à partir de la donnée des probabilités p_i, de fabriquer une telle table. L'algorithme renvoie deux tableaux R et A, contenant respectivement les valeurs des seuils de rejet $R(i) = q_i$ et les alias $A(i) = l_i$. Ce qui suit concernant la méthode de Walker est librement reproduit du rapport de DESS de F. Morata.

Construction des tables de Walker.

Pour i de 1 à k **faire** :

$R(i) \leftarrow k * p_i$; {*Initialisations des tables de seuils et d'alias*}

$A(i) \leftarrow i$;

Si $(R(i) > 1)$ **alors** :

insérer i dans H ;

sinon :

insérer i dans L ;

finsi ;

finpour ;

Tant que $(H \neq \emptyset)$ **faire** :

choisir $j \in L$; {*choix d'indices*}

choisir $s \in H$;

$A(j) \leftarrow s$; {*valeur alias de* j}

$R(s) \leftarrow R(j) + R(s) - 1$; {*modification du seuil de rejet*}

Si $(R(s) <= 1)$ **alors** :

supprimer s dans H ;

insérer s dans L ;

finsi ;

supprimer j dans L ;

fintantque ;

Remarque : Il est facile de remarquer que :

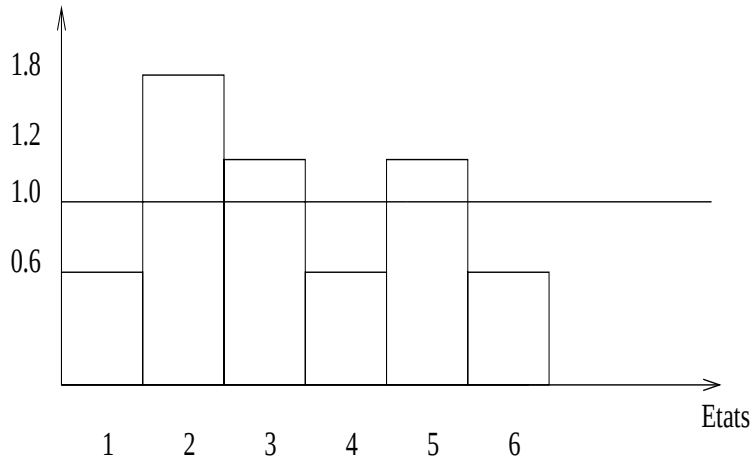
– Si $k * p_i \leq 1$, on a : $\{j;\ A(j) = i\} = \emptyset$.

– Si $k * p_i > 1$, on a : $R(i) + \sum_{\{j;\ A(j)=i\}} (1 - R(j)) = k * p_i$.

Exemple :[Tables de Walker] Soit X une variable aléatoire discrète définie sur $\{1..6\}$ distribuée comme suit :

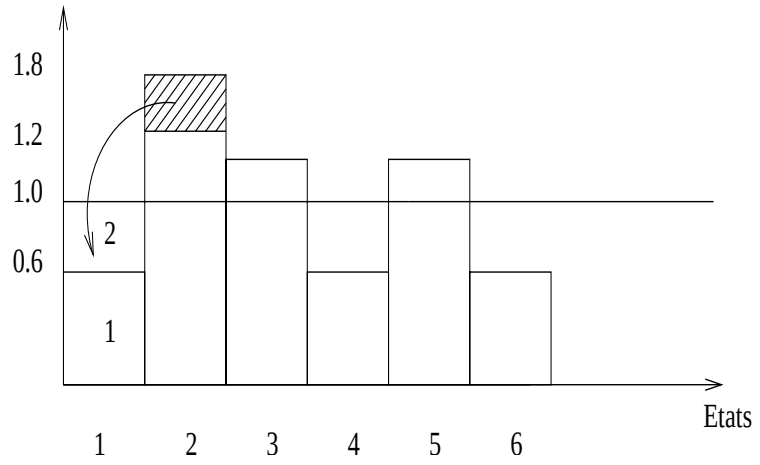
$$\begin{cases} p_1 = 0.1 \\ p_2 = 0.3 \\ p_3 = 0.2 \\ p_4 = 0.1 \\ p_5 = 0.2 \\ p_6 = 0.1 \end{cases}$$

Méthode d'Aliasing : phase initiale



Initialement, on a : $L = \{1, 4, 6\}$ et $H = \{2, 3, 5\}$.

Méthode d'Aliasing : première étape

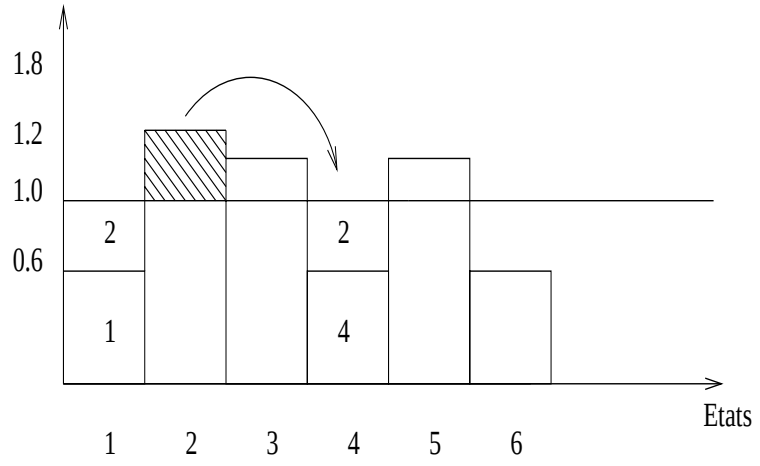


$$R[2] = R[2] + R[1] - 1 = 1.8 + 0.6 - 1 = 1.4$$

$$A[1] = 2$$

$L = \{4, 6\}$ et $H = \{2, 3, 5\}$.

Méthode d'Aliasing : deuxième étape

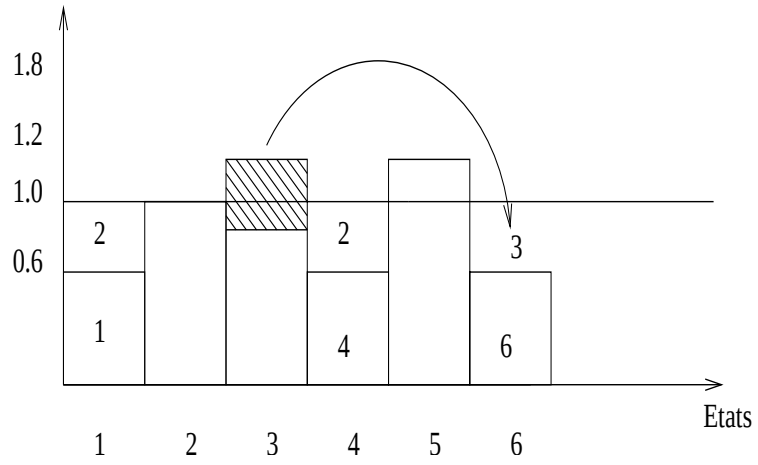


$$R[2] = R[2] + R[4] - 1 = 1$$

$$A[4] = 2$$

$L = \{2, 6\}$ et $H = \{3, 5\}$.

Méthode d'Aliasing : troisième étape

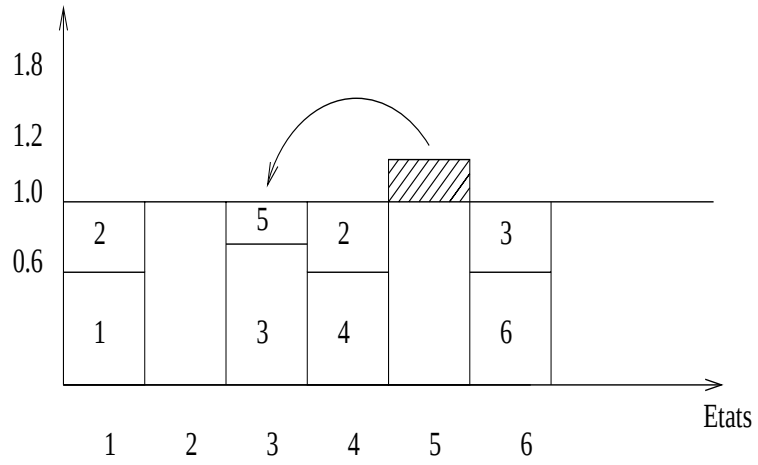


$$R[3] = R[3] + R[6] - 1 = 0.8$$

$$A[6] = 3$$

$L = \{2, 3\}$ et $H = \{5\}$.

Méthode d'Aliasing : étape finale

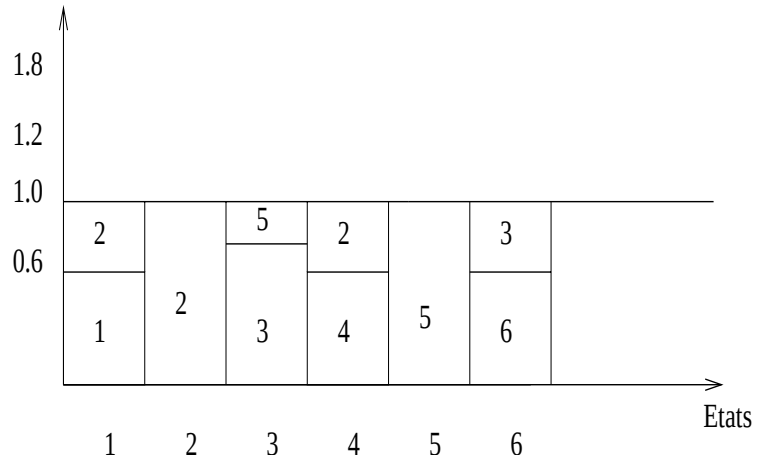


$$R[5] = R[5] + R[3] - 1$$

$$A[3] = 5$$

$L = \{2\}$ et $H = \emptyset$

Ce qui produit la construction finale :



On obtient via cette construction les tables de Walker suivantes :

Etats	1	2	3	4	5	6
Seuils	0.6	1.0	0.8	0.6	1.0	0.6
Alias	2	/	5	2	/	3

On notera que l'algorithme de fabrication des tables de Walker est $O(k)$ si les listes L et H sont par exemple manipulées comme des piles.

3.3 Remarque

Lorsqu'une méthode de génération de lois suppose un prétraitement qui peut être réemployé lors d'appels ultérieurs à la méthode, il est très fortement recommandé de n'effectuer qu'une seule fois la phase de prétraitement, et de la réutiliser systématiquement. Ceci est évident lorsque vous implémentez vous-même la méthode dans tous ses détails (ce qui est rare), mais peut l'être moins lorsque vous faites appel à des procédures déjà écrites (ou à un logiciel évolué tel que R ou MATLAB), dans lesquelles le prétraitement n'apparaît pas de manière explicite. D'autre part, regrouper en un seul plusieurs appels à une procédure rapide peut permettre d'économiser des étapes d'interfaçage plus lentes, par exemple en utilisant des fonctions vectorielles de haut niveau plutôt que des boucles. Ainsi, dans R, il est beaucoup plus rapide, pour

générer un échantillon de taille 1000000 d'une loi de Poisson de paramètre 2 (stocké dans le tableau `h`) d'effectuer la commande :

```
h<-rpois(1000000,2)
```

plutôt que :

```
for(i in (1:1000000)) h[i]<-rpois(1,2)
```

4 Lois à densité

Il est utile de se représenter les variables aléatoires continues comme des limites de variables aléatoires discrètes associées à une discrétisation de l'espace dont le pas tend vers zéro. Les deux méthodes qui suivent ne sont que des adaptations au cas continu des méthodes du découpage d'intervalles et de rejet vues précédemment dans le cas discret.

On souhaite donc simuler une variable aléatoire V dont la loi est donnée par une densité f, c'est-à-dire une fonction intégrable $f : \mathbb{R} \to \mathbb{R}_+$ telle que

$$P(a < V < b) = \int_a^b f(x)dx.$$

4.1 Méthode d'inversion

Pour simplifier, nous supposerons que f est strictement positive sur un intervalle ouvert I, et nulle hors de I (autrement dit, la variable aléatoire prend toutes ses valeurs dans l'intervalle I), et continue sur I.

La fonction de répartition :

$$F(s) = P(X \leq s) = \int_{-\infty}^{s} f(u)du$$

définit alors une bijection strictement croissante de I sur $]0,1[$, et l'on peut donc définir

$$F^{-1} :]0,1[\to I.$$

Pour simuler une variable aléatoire dont la loi possède la densité f, la méthode d'inversion est la suivante : partant d'un nombre pseudo-aléatoire U uniforme sur $[0,1]$, on pose :

$$V \leftarrow F^{-1}(U).$$

La validité de cette méthode est prouvée grâce aux égalités :

$$P(F^{-1}(U) < x) = P(U < F(x)) = F(x) = P(V < x),$$

la première égalité reposant sur la stricte croissance de F.

En théorie, il est donc possible de simuler grâce à la méthode d'inversion n'importe quelle loi à densité. En pratique, cette méthode pose le problème du calcul numérique (souvent approché) de F^{-1}, qui peut se révéler difficile, ou tout au moins coûteux en temps de calcul. De plus, si les opérations arithmétiques usuelles (somme, produit, quotient) sont en général rapides à effectuer, l'appel à des opérations plus complexes (logarithme, fonctions trigonométriques,...) peut également ralentir significativement la méthode, même si l'on dispose d'une formule d'inversion explicite. Notez que cette méthode est étroitement analogue à la méthode de découpage d'intervalles dans le cas discret, la recherche de $F^{-1}(U)$ correspondant à la recherche de l'intervalle dans lequel U se trouve. Nous laissons à titre d'exercice le fait de décrire complètement l'analogie, dans le cas où l'on approche la loi continue de densité f par une loi discrète associée à une discrétisation fine de l'intervalle I.

4.2 Méthode du rejet

La méthode est exactement analogue au cas discret. Elle repose sur le fait que l'on peut simuler par rejet des variables aléatoires uniformes dans l'ensemble $A = \bigcup_{x\in\mathbb{R}}\{x\} \times [0, f(x)]$, en inscrivant A dans un ensemble B sur lequel la génération directe de variables aléatoires uniformes est possible.

Par exemple, dans le cas où la densité f est nulle hors d'un intervalle $[a, b]$, et majorée par une constante M, on peut inscrire A dans le rectangle $B = [a, b] \times [0, M]$ et tirer des points uniformément dans B en tirant indépendamment et uniformément chacune des deux coordonnées dans leurs domaines respectifs.

Dans ce cas particulier, la méthode s'écrit donc ainsi :

1) Générer+ $U1$ uniformément distribuée sur $[a, b]$

2) Générer U_2 uniformément distribuée sur $[0, M]$

3) si $U_2 < f(U_1)$, $V \leftarrow U_1$

4) sinon retourner à l'étape 1)

En ramenant $\mathbb{R}$ sur l'intervalle $[0, 1]$ par une transformation adéquate, on peut utiliser cette méthode pour générer des variables aléatoires continues dont le support est $\mathbb{R}$ tout entier.

Comme dans le cas discret, on peut utiliser la méthode du rejet dans le cas où l'on est peut simuler des variables aléatoires possédant une densité g, la densité f vérifiant, pour tout x :

$$f(x) \leq cg(x),$$

où c est une constante positive :

1) Générer V distribuée selon la densité g

2) Générer W uniformément distribuée sur $[0, c \times g(X)]$

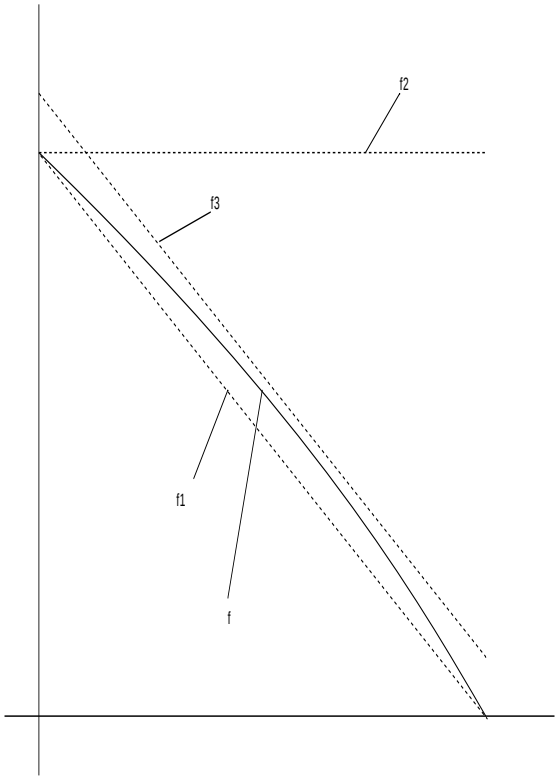
3) si+ $W < f(V)$, renvoyer V

4) sinon retourner à l'étape 1)

Comme dans le cas discret, on peut se contenter de ne manipuler les densités f et g qu'à une constante multiplicative près, c'est-à-dire si f s'écrit αf_1 et g βg_1, et que l'on a $f_1 \leq cg_1$.

La méthode du rejet a l'avantage de ne pas nécessiter le calcul de F^{-1}, puisqu'elle suppose seulement la connaissance de f, voire de f à une constante multiplicative près. En outre, si f n'est pas connue explicitement mais peut être calculée au moyen d'approximations successives, cette méthode peut se révéler particulièrement efficace puisqu'il s'agit seulement d'effectuer des comparaisons entre les valeurs prises par f et d'autres valeurs, et non pas de calculer f avec la plus grande précision possible. Comme dans le cas discret, ce qui conditionne l'efficacité de la méthode est le fait que l'ensemble B ne soit pas trop gros par rapport à l'ensemble A, de façon à ce que le nombre de rejets reste raisonnable, au moins en moyenne ou avec forte probabilité.

Par exemple, si la densité f à simuler est proche d'une fonction affine g (et à support borné), simuler d'abord la densité correspondant à cette fonction affine conduira a rejeter beaucoup moins souvent qu'en appliquant la méthode du rejet à partir d'une loi de densité constante (ce à quoi revient la version la plus simple présentée ci-dessus).



Dans le cas où la densité f est «coincée» entre deux fonctions affines g_1 et g_2, on peut encore réduire le nombre de fois où le calcul de f est nécessaire. (Pour plus de détails, voir [K]).

Insistons une fois encore sur le fait que l'indépendance (supposée) des appels successifs au générateur de nombres pseudo-aléatoires est fondamentale pour la validité de cette méthode.

La méthode de Walker ne peut pas être adaptée directement au cas continu.

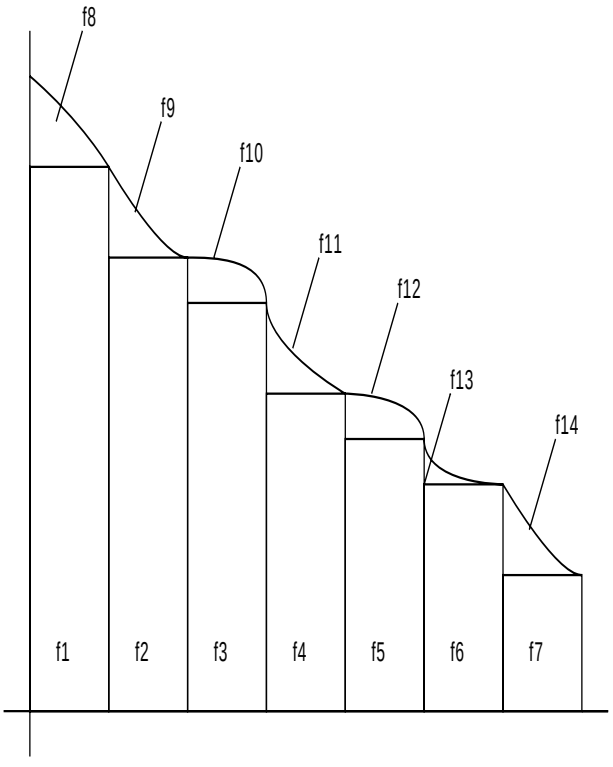
5 Décompositions de la loi à simuler

Divers types de décomposition de la loi que l'on cherche à simuler en lois plus simples – en tout cas plus faciles à simuler directement – sont employés pour accélérer les procédés de simulation. Certains sont très spécifiques de la forme des lois à simuler, d'autres sont de portée plus générale.

Souvent, on utilise des décompositions permettant de réaliser la loi recherchée à partir de combinaisons de variables aléatoires de loi plus simples, en effectuant des opérations telles que somme, produit, maximum, etc... L'utilisation de variables aléatoires conditionnées à vérifier certaines conditions, et obtenues par la méthode du rejet, est également très répandue.

Plusieurs exemples intéressants de ces idées sont présentés dans la partie suivante, et nous y renvoyons pour des exemples explicites.

Une autre idée, de portée assez générale, et permettant de réduire le temps nécessaire à la simulation de certaines lois est de décomposer celles-ci en parties «faciles» à simuler et en parties «difficiles», l'idée sous-jacente étant que, si la loi est essentiellement composée de parties faciles à simuler, et de quelques parties difficiles à traiter, la simulation de la loi consistera la plupart du temps en une simulation facile, et rarement (mais de temps en temps quand même) en une étape plus coûteuse, d'où un temps **moyen** d'exécution raisonnable. Illustrons ceci par un exemple :



Pour simuler la loi dont la densité f est représentée ci-dessus, on l'écrit, conformément au découpage représenté, sous la forme :

$$f(x) = f_1(x) + f_2(x) + \cdots + f_{14}(x),$$

que l'on réécrit :

$$f(x) = p_1 g_1(x) + p_2 g_2(x) + \cdots + p_{14} g_{14}(x),$$

chaque p_i étant égal à l'aire sous le graphe de f_i, de telle sorte que les g_i sont des densités de probabilité. On obtient alors un moyen de simuler la loi en procédant de la manière suivante :

1) Simuler $L \in \{1, \ldots, 14\}$ selon la loi $P(L = i) = p_i$

2) Renvoyer V simulé selon la densité f_L

Les densités f_i pour $i = 1, \ldots, 7$ sont tout simplement constantes, et par conséquent très faciles à simuler. L'aire $p_1 + \ldots + p_7$ recouvrant l'essentiel de la surface sous la courbe f, la méthode consistera donc la plupart du temps à effectuer une simulation très rapide. Dans le cas (très peu fréquent) où $L \geq 8$, on doit bien entendu effectuer une simulation plus complexe.

6 Exemples de méthodes spécifiques

Nous avons mentionné le fait que les méthodes spécifiques – parfois très ingénieuses à la fois dans leur principe théorique et dans leur implémentation – reposent sur des propriétés particulières des lois qu'elles visent à simuler. Bien entendu, cela ne les empêche nullement d'incorporer et d'exploiter les idées provenant des méthodes génériques décrites plus haut (notamment la méthode du rejet). Voici quelques exemples de telles méthodes, qui sont donnés, rappelons-le, à titre d'illustration, et non pas de référence. Nous vous invitons à consulter la bibliographie pour plus de détails.

6.1 Simulation de variables aléatoires gaussiennes

Quitte à centrer et réduire en dimension 1, ou à diagonaliser la matrice de covariance pour un vecteur gaussien de dimension $d \geq 2$, on peut ramener la simulation

de variables gaussiennes à la simulation de variables gaussiennes standards $\mathcal{N}(0,1)$ (unidimensionnelle, $m = 0, \sigma^2 = 1$.)

Une méthode possible est la méthode dite «rectangle-wedge-tail» (voir [K]), développée par R. Marsaglia, qui constitue une application à la loi normale de l'écriture de la loi à simuler sous la forme d'un mélange de lois plus faciles à traiter. Le graphe de la densité gaussienne est découpé en 31 portions, de trois types : rectangulaires, correspondant à un petit tronçon de gaussienne au-dessus d'un rectangle d'appui, et queue de la distribution. Les rectangles sont aisément simulés. Les tronçons également, grâce à un coincement entre deux portions de droite, en utilisant la méthode du rejet, et en ne calculant la valeur de la densité que lorsque la droite minorante ne suffit pas à déterminer si l'on doit accepter ou rejeter. La queue de la distribution est traitée par la méthode du rejet basée sur la racine carrée d'une variable de loi exponentielle. La méthode ne requiert lors de la plupart des appels, que des opérations arithmétiques élémentaires extrêmement rapides.

Un autre exemple est la méthode polaire (ou encore de Box-Muller), qui renvoie deux variables aléatoires V_1, V_2 indépendantes distribuées selon la loi $\mathcal{N}(0,1)$, et repose sur des propriétés particulières de la loi gaussienne :

1) Générer deux variables aléatoires uniformes $U_1, U_2 \in [-1,1]$

2) $S \leftarrow U_1^2 + U_2^2$

3) Si $S \geq 1$ retourner en 1)

4) Si $S = 0$ renvoyer $V_1 \leftarrow V_2 \leftarrow 0$

5) Sinon, renvoyer $V_1 \leftarrow U_1(-2\log(S)/S)^{1/2}$, $V_2 \leftarrow U_2(-2\log(S)/S)^{1/2}$

Pour prouver la validité de cette méthode, on remarque tout d'abord que l étape de rejet 3) permet d'obtenir un point de coordonnées (U_1, U_2) de loi uniforme sur le disque unité, et l'on passe en coordonnées polaires pour décrire plus commodément la loi de (U_1, U_2) conditionnellement au non-rejet, et en déduire celle du couple (V_1, V_2). On constate que l'on ne rejette qu'avec une probabilité égale à $1 - \pi/4 \approx 0,21$. Une alternative est de fabriquer directement (sans rejet) $U1$ et U_2 sous la forme $U_1 = \sqrt{U_4}\cos(2\pi U_3)$ et $U_2 = \sqrt{U_4}\sin(2\pi U_3)$, où U_3 et U_4 sont deux variables aléatoires indépendantes de loi uniforme sur $[-1,1]$, ce qui nécessite le calcul de fonctions trigonométriques.

Il existe encore bien d'autres méthodes...

6.2 Lois de Cauchy

Rappel : il s'agit de la famille de lois sur $\mathbb{R}$ dont les densités s'écrivent $f(x) = (a\pi)^{-1}(1+(x/a)^2)$, où $a > 0$. On se ramène par multiplication à une variable aléatoire de loi de Cauchy de paramètre $a = 1$.

On peut par exemple l'obtenir en renvoyant $\tan(\pi U_1)$, ou encore en générant par rejet (U_1, U_2) de loi uniforme sur le disque unité comme dans la méthode polaire ci-dessus, et en renvoyant le rapport U_1/U_2.

6.3 Lois Gamma et Beta

Rappelons qu'il s'agit des lois sur $\mathbb{R}_+$ dont les densités s'écrivent $f(x) = \Gamma(a)^{-1}x^{a-1}e^{-x}$.

Pour a entier, on peut utiliser le fait qu'une somme de a variables aléatoires indépendantes de loi exponentielle possède une loi Gamma de paramètre a.

Lorsque a est grand, cette méthode n'est plus efficace. On peut appliquer la méthode du rejet avec pour densité g celle de $X = \sqrt{2a-1}Y + a - 1$ conditionnée à prendre des valeurs positives (que l'on simulera également par rejet), où Y suit une loi de Cauchy de paramètre 1. On peut mettre g sous la forme $g(x) = \beta g_1(x)$, où $g_1(x) = (1 + (x-(a-1))/(\sqrt{2a-1}))^2)$ pour $x > 0$ (on a donc $g_1(X) = 1/(1+Y^2)$), et f sous la forme $f(x) = \gamma f_1(x)$, avec $f_1(x) = x^{a-1}e^{-x}$. On vérifie que, pour $a > 1$, $f_1 \leq cg_1$, où $c = (\frac{e}{a-1})^{a-1}$. Lorsque $a \geq 3$, la probabilité de rejet est inférieure à $1/2$.

En utilisant le fait que, si X_1 et X_2 sont deux variables aléatoires indépendantes de loi Gamma de paramètres a et b respectivement, l'expression $X_1/(X_1 + X_2)$ suit une loi Beta de paramètre (a, b) on peut effectuer des simulations à l'aide de la méthode ci-dessus.

6.4 Loi binomiale

En utilisant le fait qu'une somme de variables aléatoires indépendantes de lois de Bernoulli de paramètre p suit la loi binomiale de paramètres n et p, on peut effectuer des simulations pour des valeurs de n raisonnables. Lorsque n est grand, d'autres approches doivent être utilisées. Par exemple la méthode récursive suivante.

On écrit n sous la forme $n = a+b-1$, où a et b sont du même ordre de grandeur. On génère ensuite une variable aléatoire X de loi Beta de paramètre (a, b). Si $X \geq p$, on génère en appelant récursivement la méthode une variable aléatoire N_1 de loi binomiale de paramètres $a - 1$ et p/X, et l'on renvoie $V = N_1$. Si $X < p$, on génère

par un appel récursif une variable aléatoire N_2 de loi binomiale de paramètres $b-1$ et $(p-X)/(1-X)$, et l'on renvoie la valeur $V = a+N_1$. Clairement, après de l'ordre de $\log(n)$ appels récursifs, on est ramené à la génération de variables aléatoires de loi binomiale faisant intervenir des valeurs raisonnables de n.

La preuve de la validité de la méthode se fait en observant les faits suivants. Tout d'abord, étant données n variables aléatoires indépendantes de loi uniforme sur $[0,1]$, le nombre de ces variables qui sont $\leq p$ suit une loi binomiale de paramètres n et p. La loi de la $b-$ème plus grande de ces variables suit, quant à elle, une loi Beta de paramètre (a,b). De plus, conditionnellement à la valeur X de ce $b-$ème maximum, les $b-1$ variables supérieures à X sont indépendantes et de loi uniforme sur l'intervalle $[X,1]$, tandis que les $a-1$ variables aléatoires inférieures à X sont indépendantes et de loi uniforme sur l'intervalle $[0,X]$.

Bibliographie

[D] L. Devroye. Non-Uniform Random Variate Generation, Springer-Verlag, New York, 1986.

[G] GSL – The GNU Scientific Library http ://sources.redhat.com/gsl/

[K] D. Knuth. The Art of Computer Programming. Vol. 2. Seminumerical Algorithms. Third edition. Addison-Wesley, 1998.

[W] A. Walker. An efficient method for generating discrete random variables with general distributions, ACM Transactions on mathematical software 3 : 253–256. 1977.