Simulation “combinatoire” de variables aléatoires continues: une extension de l’algorithme de von Neumann

Philippe Duchon (LaBRI, U. Bordeaux)

Aléa 2015 - 20 mars 2015

Un peu d'histoire

- **J. von Neumann, 1951** "Various techniques used in connection with random digits" (3 pages)

Un peu d'histoire

- **J. von Neumann, 1951** "Various techniques used in connection with random digits" (3 pages)
- (entre autres) description d'un algorithme élémentaire, sans utilisation de fonctions transcendantes, pour simuler une variable *exponentielle* (densité $f(x) = e^{-x}\mathbf{1}_{x>0}$)

Un peu d'histoire

- **J. von Neumann, 1951** "Various techniques used in connection with random digits" (3 pages)
- (entre autres) description d'un algorithme élémentaire, sans utilisation de fonctions transcendantes, pour simuler une variable *exponentielle* (densité $f(x) = e^{-x}\mathbf{1}_{x>0}$)
- "In conclusion, I should like to mention that the above method can be modified to yield a distribution satisfying any first-order differential equation."

Un peu d'histoire

- **J. von Neumann, 1951** "Various techniques used in connection with random digits" (3 pages)
- (entre autres) description d'un algorithme élémentaire, sans utilisation de fonctions transcendantes, pour simuler une variable *exponentielle* (densité $f(x) = e^{-x}\mathbf{1}_{x>0}$)
- "In conclusion, I should like to mention that the above method can be modified to yield a distribution satisfying any first-order differential equation."
- **Objectifs de l'exposé :** présenter l'algorithme, ainsi qu'une extension possible (faisant effectivement intervenir une équation différentielle).

Simuler une exponentielle

- Simuler une exponentielle, c'est simple : si U est uniforme sur $[0,1]$, $X = -\ln(U)$ est exponentielle (de paramètre 1 ; pour paramètre λ quelconque, diviser par λ).

Simuler une exponentielle

- Simuler une exponentielle, c'est simple : si U est uniforme sur $[0,1]$, $X = -\ln(U)$ est exponentielle (de paramètre 1 ; pour paramètre λ quelconque, diviser par λ).
- **von Neumann :** "but... it seems objectionable to compute a transcendental function of a random number"

Simuler une exponentielle

- Simuler une exponentielle, c'est simple : si U est uniforme sur $[0, 1]$, $X = -\ln(U)$ est exponentielle (de paramètre 1 ; pour paramètre λ quelconque, diviser par λ).
- **von Neumann :** "but... it seems objectionable to compute a transcendental function of a random number"
- Autres objections possibles :
  - L'algorithme de von Neumann n'utilise que des comparaisons (et additions d'entiers), les algorithmes pour calculer des fonctions comme ln sont toujours plus complexes.

Simuler une exponentielle

- Simuler une exponentielle, c'est simple : si U est uniforme sur $[0,1]$, $X = -\ln(U)$ est exponentielle (de paramètre 1 ; pour paramètre λ quelconque, diviser par λ).
- **von Neumann :** "but... it seems objectionable to compute a transcendental function of a random number"
- Autres objections possibles :
  - L'algorithme de von Neumann n'utilise que des comparaisons (et additions d'entiers), les algorithmes pour calculer des fonctions comme ln sont toujours plus complexes.
  - Si on peut avoir un algorithme *exact* avec une saveur combinatoire, ce serait dommage de se priver.

L'algorithme de von Neumann : version 1

- On répète jusqu'à réussite :
  - Tirer une suite $X_1, X_2, \ldots X_n$ d'uniformes sur $[0, 1]$ indépendantes, jusqu'à la **première montée** $(X_n > X_{n-1})$
  - Si n est impair : échec (incrémenter le compteur d'échecs K)
  - Si n est pair : succès, retourner $K + X_1$

Déroulement de l'algorithme : un exemple

- Suite de tirages : $0.3, 0.1, 0.4, 0.15, 0.9, 0.2 \ldots$

Déroulement de l'algorithme : un exemple

- Suite de tirages : $0.3, 0.1, 0.4, 0.15, 0.9, 0.2 \ldots$
- Premiere série : $0.3 > 0.1$

Déroulement de l'algorithme : un exemple

- Suite de tirages : $0.3, 0.1, 0.4, 0.15, 0.9, 0.2 \ldots$
- Premiere série : $0.3 > 0.1 < 0.4$: fin de la série (longueur impaire)

Déroulement de l'algorithme : un exemple

- Suite de tirages : $0.3, 0.1, 0.4, 0.15, 0.9, 0.2 \ldots$
- Premiere série : $0.3 > 0.1 < 0.4$: fin de la série (longueur impaire)
- Deuxième série : $0.15 < 0.9$: longueur paire

Déroulement de l'algorithme : un exemple

- Suite de tirages : $0.3, 0.1, 0.4, 0.15, 0.9, 0.2 \ldots$
- Premiere série : $0.3 > 0.1 < 0.4$: fin de la série (longueur impaire)
- Deuxième série : $0.15 < 0.9$: longueur paire
- La valeur retournée est $1 + 0.15 = 1.15$

L'algorithme de von Neumann (reformulé)

- **Frac** : retourne soit `Echec`, soit un $x \in [0,1]$
  - Tirer des variables indépendantes uniformes sur $[0,1]$, $X_1, X_2, \ldots$ jusqu'au premier $n \geq 2$ tel que $X_n > X_{n-1}$
  - Si n est pair, retourner X_1 ; sinon, retourner `Echec`

L'algorithme de von Neumann (reformulé)

- **Frac** : retourne soit `Echec`, soit un $x \in [0,1]$
  - Tirer des variables indépendantes uniformes sur $[0,1]$, $X_1, X_2, \ldots$ jusqu'au premier $n \geq 2$ tel que $X_n > X_{n-1}$
  - Si n est pair, retourner X_1 ; sinon, retourner `Echec`
- **Expo** : retourne un réel positif
  - $k = 0$, $X =$ **Frac**()
  - Tant que $X =$ `Echec` faire $X =$ **Frac**(), $k = k + 1$
  - Retourner $X + k$

L'algorithme de von Neumann (reformulé)

- **Frac** : retourne soit `Echec`, soit un $x \in [0,1]$
  - Tirer des variables indépendantes uniformes sur $[0,1]$, $X_1, X_2, \ldots$ jusqu'au premier $n \geq 2$ tel que $X_n > X_{n-1}$
  - Si n est pair, retourner X_1 ; sinon, retourner `Echec`
- **Expo** : retourne un réel positif
  - $k = 0$, $X =$ **Frac**()
  - Tant que $X =$ `Echec` faire $X =$ **Frac**(), $k = k + 1$
  - Retourner $X + k$
- **Proposition :** la valeur retournée par **Expo()** suit la loi exponentielle de paramètre 1

Analyse de **Frac()**

- Pour tout n, les n premiers tirages $X_1, \ldots, X_n$ sont dans un ordre relatif aléatoire uniforme

Analyse de **Frac()**

- Pour tout n, les n premiers tirages $X_1, \ldots, X_n$ sont dans un ordre relatif aléatoire uniforme
- Donc la probabilité que les n premiers soient décroissants, est $1/n!$

Analyse de **Frac()**

- Pour tout n, les n premiers tirages $X_1, \ldots, X_n$ sont dans un ordre relatif aléatoire uniforme
- Donc la probabilité que les n premiers soient décroissants, est $1/n!$
- Donc la probabilité que, dans une suite infinie, la première montée soit au n-ème tirage, est $1/(n-1)! - 1/n!$

Analyse de **Frac()**

- Pour tout n, les n premiers tirages $X_1, \ldots, X_n$ sont dans un ordre relatif aléatoire uniforme
- Donc la probabilité que les n premiers soient décroissants, est $1/n!$
- Donc la probabilité que, dans une suite infinie, la première montée soit au n-ème tirage, est $1/(n-1)! - 1/n!$
- Donc la probabilité que la première montée soit en position impaire (probabilité de retourner `Echec`) est

$$p = \sum_{k \geq 0} \frac{1}{(2k)!} - \frac{1}{(2k+1)!} = e^{-1}$$

Analyse de **Frac()**

- Pour tout n, les n premiers tirages $X_1, \ldots, X_n$ sont dans un ordre relatif aléatoire uniforme
- Donc la probabilité que les n premiers soient décroissants, est $1/n!$
- Donc la probabilité que, dans une suite infinie, la première montée soit au n-ème tirage, est $1/(n-1)! - 1/n!$
- Donc la probabilité que la première montée soit en position impaire (probabilité de retourner `Echec`) est

$$p = \sum_{k \geq 0} \frac{1}{(2k)!} - \frac{1}{(2k+1)!} = e^{-1}$$

- Pour un $x \in [0,1]$ et un entier $n \geq 2$, la probabilité que $X_1 \leq x$ **et** que la première montée soit au n-ème tirage, est $x^{n-1}/(n-1)! - x^n/n!$

Analyse de **Frac()**

- Pour tout n, les n premiers tirages $X_1, \ldots, X_n$ sont dans un ordre relatif aléatoire uniforme
- Donc la probabilité que les n premiers soient décroissants, est $1/n!$
- Donc la probabilité que, dans une suite infinie, la première montée soit au n-ème tirage, est $1/(n-1)! - 1/n!$
- Donc la probabilité que la première montée soit en position impaire (probabilité de retourner `Echec`) est

$$p = \sum_{k \geq 0} \frac{1}{(2k)!} - \frac{1}{(2k+1)!} = e^{-1}$$

- Pour un $x \in [0,1]$ et un entier $n \geq 2$, la probabilité que $X_1 \leq x$ **et** que la première montée soit au n-ème tirage, est $x^{n-1}/(n-1)! - x^n/n!$
- En sommant : la probabilité de retourner un réel $\leq x$, est $1 - e^{-x}$

Analyse de **Expo()**

- Dans l'algorithme, le X utilisé à la fin est distribué comme la valeur de retour de **Frac()**, conditionnée à ne pas être `Echec` ; soit, comme une exponentielle conditionnée à être dans $[0, 1]$.

Analyse de **Expo()**

- Dans l'algorithme, le X utilisé à la fin est distribué comme la valeur de retour de **Frac()**, conditionnée à ne pas être `Echec` ; soit, comme une exponentielle conditionnée à être dans $[0, 1]$.
- Le nombre de rejets (la valeur de k) suit la loi géométrique de paramètre $1 - e^{-1}$: c'est bien la loi de la partie entière d'une exponentielle.

Analyse de **Expo()**

- Dans l'algorithme, le X utilisé à la fin est distribué comme la valeur de retour de **Frac()**, conditionnée à ne pas être `Echec` ; soit, comme une exponentielle conditionnée à être dans $[0, 1]$.
- Le nombre de rejets (la valeur de k) suit la loi géométrique de paramètre $1 - e^{-1}$: c'est bien la loi de la partie entière d'une exponentielle.
- Dans l'algorithme, k est distribué comme la partie entière d'une exponentielle, X est indépendant de k et distribué comme la partie fractionnaire d'une exponentielle : $k + X$ est bien une exponentielle.

Analyse de **Expo()**

- Dans l'algorithme, le X utilisé à la fin est distribué comme la valeur de retour de **Frac()**, conditionnée à ne pas être `Echec` ; soit, comme une exponentielle conditionnée à être dans $[0, 1]$.
- Le nombre de rejets (la valeur de k) suit la loi géométrique de paramètre $1 - e^{-1}$: c'est bien la loi de la partie entière d'une exponentielle.
- Dans l'algorithme, k est distribué comme la partie entière d'une exponentielle, X est indépendant de k et distribué comme la partie fractionnaire d'une exponentielle : $k + X$ est bien une exponentielle.
- **Flajolet, Saheb 1987 :** analyse précise du nombre de bits aléatoires consommés

Extensions ?

- “the above method can be modified to yield a distribution satisfying any first-order differential equation”

Extensions ?

- “the above method can be modified to yield a distribution satisfying any first-order differential equation”
- Ça veut dire quoi pour une distribution de probabilités, de satisfaire une équation différentielle ? Et est-ce qu'il ne faut pas prévoir une forme particulière d'équation ?

Extensions ?

- "the above method can be modified to yield a distribution satisfying any first-order differential equation"
- Ça veut dire quoi pour une distribution de probabilités, de satisfaire une équation différentielle ? Et est-ce qu'il ne faut pas prévoir une forme particulière d'équation ?
- Une interprétation tentante : distribution dont la *densité* satisfait une équation différentielle *linéaire* du premier ordre : $y'(t) = g(t)y(t)$ (avec g "connue")

Extensions ?

- "the above method can be modified to yield a distribution satisfying any first-order differential equation"
- Ça veut dire quoi pour une distribution de probabilités, de satisfaire une équation différentielle ? Et est-ce qu'il ne faut pas prévoir une forme particulière d'équation ?
- Une interprétation tentante : distribution dont la *densité* satisfait une équation différentielle *linéaire* du premier ordre : $y'(t) = g(t)y(t)$ (avec g "connue")
- En particulier, la densité de la loi normale satisfait une telle équation : $y' = -ty$: $g(t) = -t$ devrait rentrer dans toutes les hypothèses qu'on serait amené à imposer à g (à part éventuellement le caractère borné).

Extensions ?

- "the above method can be modified to yield a distribution satisfying any first-order differential equation"
- Ça veut dire quoi pour une distribution de probabilités, de satisfaire une équation différentielle ? Et est-ce qu'il ne faut pas prévoir une forme particulière d'équation ?
- Une interprétation tentante : distribution dont la *densité* satisfait une équation différentielle *linéaire* du premier ordre : $y'(t) = g(t)y(t)$ (avec g "connue")
- En particulier, la densité de la loi normale satisfait une telle équation : $y' = -ty$: $g(t) = -t$ devrait rentrer dans toutes les hypothèses qu'on serait amené à imposer à g (à part éventuellement le caractère borné).
- **Forsythe, 1972** : fait cette interprétation (équation différentielle sur la densité) **mais** calculs numériques d'intégrales

Une extension de la méthode

- Je propose comme interprétation de la remarque de von Neumann : la méthode peut être modifiée pour donner n'importe quelle distribution sur $\mathbb{R}^+$ dont la fonction complémentaire de la fonction de répartition, $y(t) = \mathbb{P}(X > t)$, satisfait une équation différentielle linéaire du premier ordre $y'(t) = -g(t)y(t)$, pour une fonction g qu'on sait évaluer et majorer.

Une extension de la méthode

- Je propose comme interprétation de la remarque de von Neumann : la méthode peut être modifiée pour donner n'importe quelle distribution sur $\mathbb{R}^+$ dont la fonction complémentaire de la fonction de répartition, $y(t) = \mathbb{P}(X > t)$, satisfait une équation différentielle linéaire du premier ordre $y'(t) = -g(t)y(t)$, pour une fonction g qu'on sait évaluer et majorer.
- Pas besoin d'hypothèse de connaissance analytique précise de la fonction g.

Une extension de la méthode

- Je propose comme interprétation de la remarque de von Neumann : la méthode peut être modifiée pour donner n'importe quelle distribution sur $\mathbb{R}^+$ dont la fonction complémentaire de la fonction de répartition, $y(t) = \mathbb{P}(X > t)$, satisfait une équation différentielle linéaire du premier ordre $y'(t) = -g(t)y(t)$, pour une fonction g qu'on sait évaluer et majorer.
- Pas besoin d'hypothèse de connaissance analytique précise de la fonction g.
- Deux fonctions "boîtes noires" : $g(t)$, et $h(t, u) \geq \sup\{g(x) : t \leq x \leq u\}$

Une extension de la méthode

- Je propose comme interprétation de la remarque de von Neumann : la méthode peut être modifiée pour donner n'importe quelle distribution sur $\mathbb{R}^+$ dont la fonction complémentaire de la fonction de répartition, $y(t) = \mathbb{P}(X > t)$, satisfait une équation différentielle linéaire du premier ordre $y'(t) = -g(t)y(t)$, pour une fonction g qu'on sait évaluer et majorer.
- Pas besoin d'hypothèse de connaissance analytique précise de la fonction g.
- Deux fonctions "boîtes noires" : $g(t)$, et $h(t, u) \geq \sup\{g(x) : t \leq x \leq u\}$
- L'algorithme que je propose est à la fois assez proche de celui de von Neumann, et proche de l'idée des "machines de Buffon" (**Flajolet, Pelletier, Soria 2011**) adaptée aux distributions continues.

L'équation différentielle

- Solution de l'équation : $y(t) = e^{-\int_0^t g(u)du}$ ($y(0) = 1$ par définition) ; pour que le problème définisse une loi de probabilité, il faut que l'on ait $g(x) \geq 0$ et $\int_0^{+\infty} g(u)du = +\infty$.

L'équation différentielle

- Solution de l'équation : $y(t) = e^{-\int_0^t g(u)du}$ ($y(0) = 1$ par définition) ; pour que le problème définisse une loi de probabilité, il faut que l'on ait $g(x) \geq 0$ et $\int_0^{+\infty} g(u)du = +\infty$.
- Conséquence : on a potentiellement un algorithme si on connaît analytiquement g (ou sa primitive) :
  - tirer une exponentielle A
  - retourner X tel que $\int_0^X g(t)dt = A$

L'équation différentielle

- Solution de l'équation : $y(t) = e^{-\int_0^t g(u)du}$ ($y(0) = 1$ par définition) ; pour que le problème définisse une loi de probabilité, il faut que l'on ait $g(x) \geq 0$ et $\int_0^{+\infty} g(u)du = +\infty$.
- Conséquence : on a potentiellement un algorithme si on connaît analytiquement g (ou sa primitive) :
  - tirer une exponentielle A
  - retourner X tel que $\int_0^X g(t)dt = A$
- (Nécessite de savoir inverser la primitive de g ; on ne souhaite pas faire ça de manière numérique)

L'équation différentielle

- Solution de l'équation : $y(t) = e^{-\int_0^t g(u)du}$ ($y(0) = 1$ par définition) ; pour que le problème définisse une loi de probabilité, il faut que l'on ait $g(x) \geq 0$ et $\int_0^{+\infty} g(u)du = +\infty$.
- Conséquence : on a potentiellement un algorithme si on connaît analytiquement g (ou sa primitive) :
  - tirer une exponentielle A
  - retourner X tel que $\int_0^X g(t)dt = A$
- (Nécessite de savoir inverser la primitive de g ; on ne souhaite pas faire ça de manière numérique)
- Le problème est équivalent à celui de trouver *le point de plus petite abscisse* d'un processus ponctuel de Poisson (d'intensité 1) sur le domaine $(x \geq 0, 0 \leq y \leq g(x))$ (mais ce n'est pas non plus ce qu'on va faire)

Intermède : "Générateur de Buffon" pour $x \mapsto e^{-x}$

(**Flajolet, Pelletier, Soria 2011**)

- **Hypothèse :** on "sait" tirer des uniformes, et on a accès à un générateur de Bernoulli de paramètre p ($0 < p < 1$, inconnu), **Bern()**

Intermède : "Générateur de Buffon" pour $x \mapsto e^{-x}$

(**Flajolet, Pelletier, Soria 2011**)

- **Hypothèse :** on "sait" tirer des uniformes, et on a accès à un générateur de Bernoulli de paramètre p ($0 < p < 1$, inconnu), **Bern()**
- **Alors** on sait tirer une Bernoulli de paramètre e^{-p}

Intermède : "Générateur de Buffon" pour $x \mapsto e^{-x}$

(**Flajolet, Pelletier, Soria 2011**)

- **Hypothèse :** on "sait" tirer des uniformes, et on a accès à un générateur de Bernoulli de paramètre p ($0 < p < 1$, inconnu), **Bern()**
- **Alors** on sait tirer une Bernoulli de paramètre e^{-p}
- L'algorithme est une variante de celui de von Neumann !

- Tirer une séquence de **couples** (X_i, B_i) indépendants de variables indépendantes, où X_i est uniforme (sur $[0, 1]$), et B_i est une Bernoulli de paramètre p
- S'arrêter au premier n tel que $B_n = 0$ ou $X_n > X_{n-1}$
- Retourner 1 si n est impair, 0 si n est pair

Un début d'algorithme

- On découpe $\mathbb{R}^+$ en intervalles $I_k = [a_k, a_{k+1}[$, et l'algorithme procéde par étapes
- À l'étape k, on sait déjà que la variable qu'on va engendrer est $\geq a_k$; on va décider si elle va être dans l'intervalle I_k, ou plus grande que a_{k+1}.

Un début d'algorithme

- On découpe $\mathbb{R}^+$ en intervalles $I_k = [a_k, a_{k+1}[$, et l'algorithme procéde par étapes
- À l'étape k, on sait déjà que la variable qu'on va engendrer est $\geq a_k$; on va décider si elle va être dans l'intervalle I_k, ou plus grande que a_{k+1}.
- **Sachant** que le X qu'on veut retourner est $\geq a_k$, la probabilité qu'il soit $\geq a_{k+1}$ est $\exp(-A_k)$, avec

$$A_k = \int_{a_k}^{a_{k+1}} g(t)dt.$$

Un début d'algorithme

- On découpe $\mathbb{R}^+$ en intervalles $I_k = [a_k, a_{k+1}[$, et l'algorithme procéde par étapes
- À l'étape k, on sait déjà que la variable qu'on va engendrer est $\geq a_k$; on va décider si elle va être dans l'intervalle I_k, ou plus grande que a_{k+1}.
- **Sachant** que le X qu'on veut retourner est $\geq a_k$, la probabilité qu'il soit $\geq a_{k+1}$ est $\exp(-A_k)$, avec

$$A_k = \int_{a_k}^{a_{k+1}} g(t)dt.$$

- On utilise le générateur de Buffon : on interpréte l'intégrale comme une aire, donc comme une probabilité en prenant des points aléatoires dans une certaine zone.

Choix des intervalles

- $a_0 = 0$; puis, à partir de a_k, on choisit a_{k+1} de telle sorte que, sur l'intervalle $[a_k, a_{k+1}]$, la fonction g soit majorée par $1/(a_{k+1} - a_k)$
- (par exemple, $a_{k+1} = a_k + \min(1, 1/h(a_k, 1 + a_k))$)

Choix des intervalles

- $a_0 = 0$; puis, à partir de a_k, on choisit a_{k+1} de telle sorte que, sur l'intervalle $[a_k, a_{k+1}]$, la fonction g soit majorée par $1/(a_{k+1} - a_k)$
- (par exemple, $a_{k+1} = a_k + \min(1, 1/h(a_k, 1 + a_k))$)
- À partir de là, $A_k = \int_{a_i}^{a_{i+1}} g(t)dt$ est la *probabilité* qu'un point (x, y) aléatoire uniforme dans le domaine $[a_k, a_{k+1}] \times [0, 1/(a_{k+1} - a_k)]$ satisfasse $y \leq g(x)$: on tient notre générateur de Bernoulli de paramètre A_k, donc le générateur de Buffon nous donne le générateur de paramètre e^{-A_k}.

Choix des intervalles

- $a_0 = 0$; puis, à partir de a_k, on choisit a_{k+1} de telle sorte que, sur l'intervalle $[a_k, a_{k+1}]$, la fonction g soit majorée par $1/(a_{k+1} - a_k)$
- (par exemple, $a_{k+1} = a_k + \min(1, 1/h(a_k, 1 + a_k))$)
- À partir de là, $A_k = \int_{a_i}^{a_{i+1}} g(t)dt$ est la *probabilité* qu'un point (x, y) aléatoire uniforme dans le domaine $[a_k, a_{k+1}] \times [0, 1/(a_{k+1} - a_k)]$ satisfasse $y \leq g(x)$: on tient notre générateur de Bernoulli de paramètre A_k, donc le générateur de Buffon nous donne le générateur de paramètre e^{-A_k}.
- **Yapluka :** trouver un moyen, quand on décide de s'arrêter dans l'intervalle I_k, de simuler exactement la bonne densité...

Choix des intervalles

- $a_0 = 0$; puis, à partir de a_k, on choisit a_{k+1} de telle sorte que, sur l'intervalle $[a_k, a_{k+1}]$, la fonction g soit majorée par $1/(a_{k+1} - a_k)$
- (par exemple, $a_{k+1} = a_k + \min(1, 1/h(a_k, 1 + a_k))$)
- À partir de là, $A_k = \int_{a_i}^{a_{i+1}} g(t)dt$ est la *probabilité* qu'un point (x, y) aléatoire uniforme dans le domaine $[a_k, a_{k+1}] \times [0, 1/(a_{k+1} - a_k)]$ satisfasse $y \leq g(x)$: on tient notre générateur de Bernoulli de paramètre A_k, donc le générateur de Buffon nous donne le générateur de paramètre e^{-A_k}.
- **Yapluka :** trouver un moyen, quand on décide de s'arrêter dans l'intervalle I_k, de simuler exactement la bonne densité...
- **Miracle :** comme dans l'algorithme de von Neumann, c'est donné par le X_1 de la séquence qui provoque l'arrêt.

L'algorithme sur un intervalle $I_k = [a_k, a_{k+1}]$

$(J_k = [0, 1/(a_{k+1} - a_k)]$, $A_k = \int_{I_k} g(t)dt)$

- Tirer une suite de points aléatoires uniformes dans $I_k \times J_k$, $(X_i, Y_i)_{1 \leq i \leq N}$, jusqu'au premier N tel que l'on ait $Y_N \geq g(X_N)$ ou $X_N \geq X_{N-1}$ (y compris $N = 1$, si $Y_1 \geq g(X_1)$)
- Si n est impair, passer à l'intervalle suivant ; sinon, retourner X_1

Correction de l'algorithme

- L'algorithme atteint $N \geq n$ si n points aléatoires dans $I_k \times J_k$ sont d'abscisses décroissantes **et** tous sous la courbe, soit avec probabilité $A_k^n/n!$; donc il s'arrête avec $N = n$ avec probabilité $A_k^{n-1}/(n-1)! - A_k^n/n!$.

Correction de l'algorithme

- L'algorithme atteint $N \geq n$ si n points aléatoires dans $I_k \times J_k$ sont d'abscisses décroissantes **et** tous sous la courbe, soit avec probabilité $A_k^n/n!$; donc il s'arrête avec $N = n$ avec probabilité $A_k^{n-1}/(n-1)! - A_k^n/n!$.
- En sommant sur les n impairs, on obtient e^{-A_k} qui est bien la probabilité correcte que X soit $\geq a_{k+1}$, **sachant** qu'il est $\geq a_k$.

Correction de l'algorithme

- L'algorithme atteint $N \geq n$ si n points aléatoires dans $I_k \times J_k$ sont d'abscisses décroissantes **et** tous sous la courbe, soit avec probabilité $A_k^n/n!$; donc il s'arrête avec $N = n$ avec probabilité $A_k^{n-1}/(n-1)! - A_k^n/n!$.
- En sommant sur les n impairs, on obtient e^{-A_k} qui est bien la probabilité correcte que X soit $\geq a_{k+1}$, **sachant** qu'il est $\geq a_k$.
- La probabilité de s'arrêter sur $N = n$ avec un $X_1 \leq t$ (pour $t \in I_k$) est la probabilité que $n-1$ points, mais pas n, soient d'abscisses décroissantes, et dans le domaine $\{(x, y) : a_k \leq x \leq t, 0 \leq y \leq g(x)\}$, qui est d'aire $A_k(t) = \int_{a_k}^{t} g(u)du$;

Correction de l'algorithme

- L'algorithme atteint $N \geq n$ si n points aléatoires dans $I_k \times J_k$ sont d'abscisses décroissantes **et** tous sous la courbe, soit avec probabilité $A_k^n/n!$; donc il s'arrête avec $N = n$ avec probabilité $A_k^{n-1}/(n-1)! - A_k^n/n!$.
- En sommant sur les n impairs, on obtient e^{-A_k} qui est bien la probabilité correcte que X soit $\geq a_{k+1}$, **sachant** qu'il est $\geq a_k$.
- La probabilité de s'arrêter sur $N = n$ avec un $X_1 \leq t$ (pour $t \in I_k$) est la probabilité que $n-1$ points, mais pas n, soient d'abscisses décroissantes, et dans le domaine $\{(x, y) : a_k \leq x \leq t, 0 \leq y \leq g(x)\}$, qui est d'aire $A_k(t) = \int_{a_k}^{t} g(u)du$;
- donc cette probabilité est $A_k(t)^{n-1}/(n-1)! - A_k(t)^n/n!$;

Correction de l'algorithme

- L'algorithme atteint $N \geq n$ si n points aléatoires dans $I_k \times J_k$ sont d'abscisses décroissantes **et** tous sous la courbe, soit avec probabilité $A_k^n/n!$; donc il s'arrête avec $N = n$ avec probabilité $A_k^{n-1}/(n-1)! - A_k^n/n!$.
- En sommant sur les n impairs, on obtient e^{-A_k} qui est bien la probabilité correcte que X soit $\geq a_{k+1}$, **sachant** qu'il est $\geq a_k$.
- La probabilité de s'arrêter sur $N = n$ avec un $X_1 \leq t$ (pour $t \in I_k$) est la probabilité que $n-1$ points, mais pas n, soient d'abscisses décroissantes, et dans le domaine $\{(x, y) : a_k \leq x \leq t, 0 \leq y \leq g(x)\}$, qui est d'aire $A_k(t) = \int_{a_k}^{t} g(u)du$;
- donc cette probabilité est $A_k(t)^{n-1}/(n-1)! - A_k(t)^n/n!$;
- en sommant sur les n pairs, on obtient comme probabilité de retourner un $X \leq t$: $p(t) = \cdots = 1 - e^{-A_k(t)}$

Correction de l'algorithme

- L'algorithme atteint $N \geq n$ si n points aléatoires dans $I_k \times J_k$ sont d'abscisses décroissantes **et** tous sous la courbe, soit avec probabilité $A_k^n/n!$; donc il s'arrête avec $N = n$ avec probabilité $A_k^{n-1}/(n-1)! - A_k^n/n!$.
- En sommant sur les n impairs, on obtient e^{-A_k} qui est bien la probabilité correcte que X soit $\geq a_{k+1}$, **sachant** qu'il est $\geq a_k$.
- La probabilité de s'arrêter sur $N = n$ avec un $X_1 \leq t$ (pour $t \in I_k$) est la probabilité que $n-1$ points, mais pas n, soient d'abscisses décroissantes, et dans le domaine $\{(x, y) : a_k \leq x \leq t, 0 \leq y \leq g(x)\}$, qui est d'aire $A_k(t) = \int_{a_k}^{t} g(u)du$;
- donc cette probabilité est $A_k(t)^{n-1}/(n-1)! - A_k(t)^n/n!$;
- en sommant sur les n pairs, on obtient comme probabilité de retourner un $X \leq t$: $p(t) = \cdots = 1 - e^{-A_k(t)}$
- C'est bien la probabilité que $X < t$, sachant $X > a_k$!

Au total...

On a un algorithme qui simule la loi cible, en utilisant comme seules opérations :

- l'évaluation des fonctions "boîtes noires" g et h
- des comparaisons
- des tirages d'uniformes dans des intervalles

Au total...

On a un algorithme qui simule la loi cible, en utilisant comme seules opérations :

- l'évaluation des fonctions "boîtes noires" g et h
- des comparaisons
- des tirages d'uniformes dans des intervalles
- Si $g(t) = 1$ et $h(t, u) = 1$, on retombe **exactement** sur l'algorithme de von Neumann (avec, en plus, des tirages inutiles pour les Y)

Petite conclusion

- On arrive à une extension de la méthode, qui en préserve bien l'idée générale

Petite conclusion

- On arrive à une extension de la méthode, qui en préserve bien l'idée générale
- Analyse de l'algorithme : reste à faire, mais dépendra de la qualité de la majoration (fonction h)
  - nombre d'uniformes utilisées : pas très difficile
  - nombre de bits aléatoires (génération bit à bit) : en s'inspirant de Flajolet-Saheb ?

Petite conclusion

- On arrive à une extension de la méthode, qui en préserve bien l'idée générale
- Analyse de l'algorithme : reste à faire, mais dépendra de la qualité de la majoration (fonction h)
  - nombre d'uniformes utilisées : pas très difficile
  - nombre de bits aléatoires (génération bit à bit) : en s'inspirant de Flajolet-Saheb ?
- Équation différentielle sur la **densité :** (travail encore un peu en cours) on arrive à des algorithmes moins élégants ; on retrouve ainsi, quasiment tel quel, un algorithme "à la Buffon" pour la loi normale, décrit par Karney.

Merci de votre attention
Merci aux organisateurs pour une nouvelle édition passionnante d'Aléa