Modélisation et simulation

Simulation Monte Carlo

Gianluca Bontempi
Département d'Informatique
Boulevard de Triomphe - CP 212
http://www.ulb.ac.be/di

December 9, 2021

As far as the laws of mathematics refer to reality, they are not certain, as far as they are certain, they do not refer to reality. (Albert Einstein, 1921)

Simulation statistique

- Jusqu'ici, nos avons considéré des modèles déterministes.
- Le manque de connaissance, la procédure d'abstraction accomplie pendant la modélisation, les erreurs de mesures rendent souvent une représentation déterministe de la réalité trop limitée.
- Il est important donc représenter à l'intérieur de notre modèle l'incertitude et être capable de la tenir en considération pendant la simulation.
- Le formalisme probabiliste a été souvent utilisé dans les sciences pour représenter et manipuler l'incertitude.

Simulation statistique

- Trouver une solution analytique d'un modèle probabiliste est souvent impossible. Dans ces cas, la seule manière d'étudier le système est donc la simulation.
- Simuler un système avec des paramètres ou des conditions initiales probabilistes demande la capacité de générer des nombres selon une distribution de probabilité.
- On définit *simulation statistique* toute méthode qui utilise des séquences des nombres aléatoires (ou *random*).

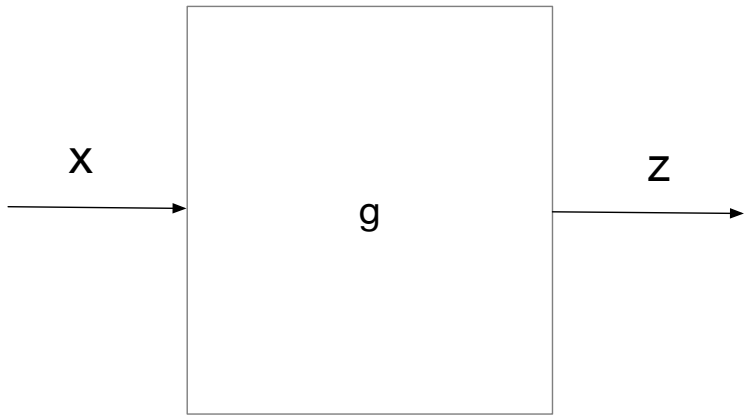
Les méthodes Monte-Carlo

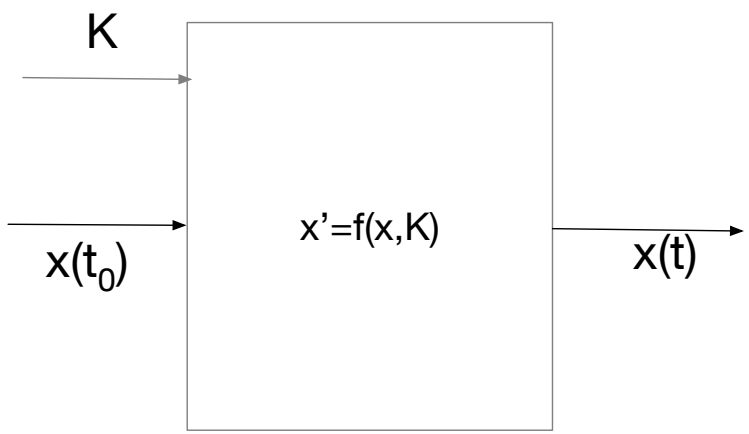
- Les techniques de simulation Monte-Carlo sont utilisées pour simuler des systèmes déterministes avec des paramètres ou des entrées stochastiques.
- Le nom a été proposé par les scientifiques du projet Manhattan lors de la deuxième guerre mondiale et fait allusion aux jeux de hasard pratiqués à Monaco.
- Parmi les pionniers des méthodes MC nous retrouvons E. Fermi, J. Neumann, S. Ulam, N. Metropolis.
- Les méthodes MC sont aujourd'hui utilisées pour simuler des phénomènes physiques complexes dans plusieurs domaines scientifiques et appliqués: radioactivité, physique des hautes énergies, réseaux, économétrie, logistique.

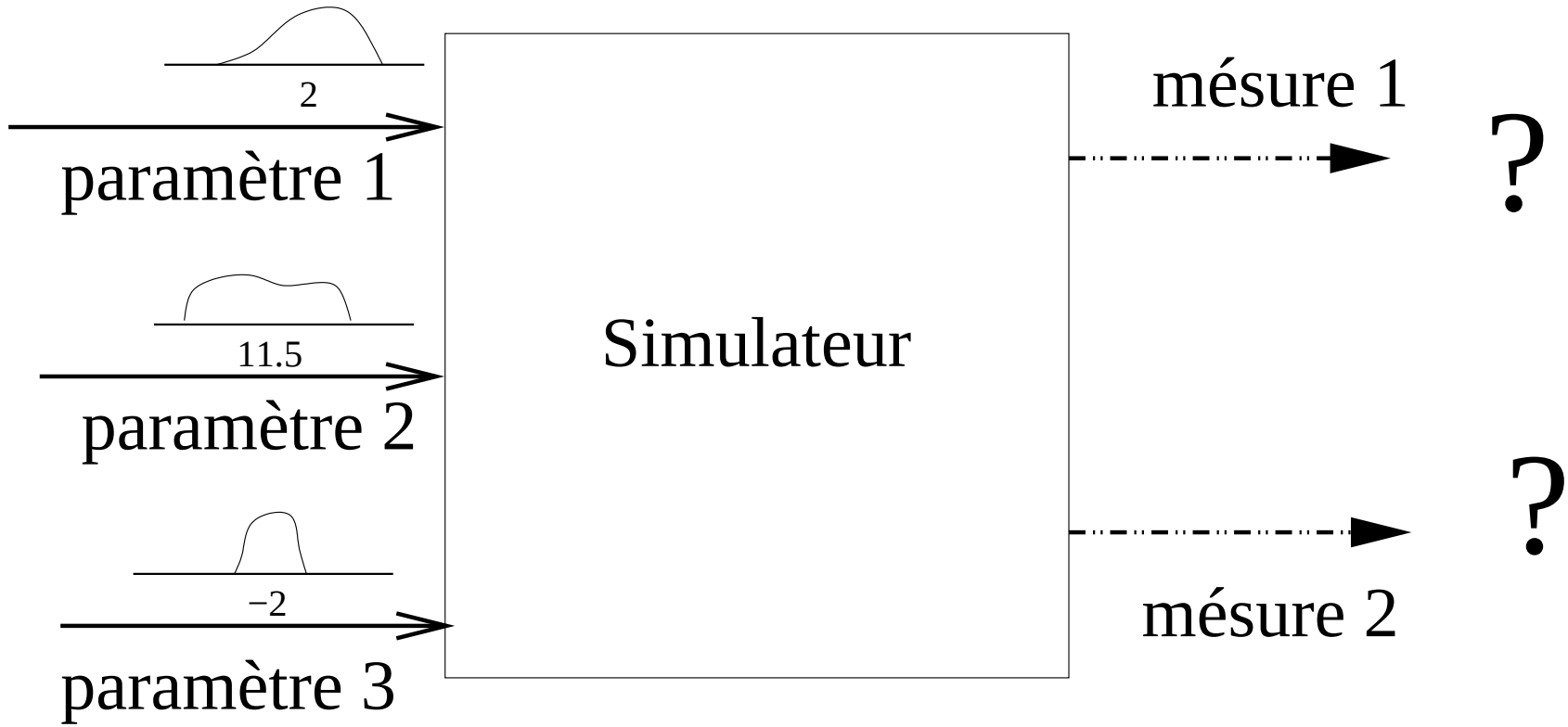
Les méthodes Monte-Carlo

- La technique de simulation Monte-Carlo s'appuie sur l'échantillonnage des distributions des quantités incertaines.
- Il peut être visualisé comme une boite noire où entre un flux de nombres pseudo-aléatoires (c.-à-d. générés par l'ordinateur) et un flux de nombres sort.
- L'estimation de la quantité d'intérêt est obtenue en analysant l'output.

Simulation comme une fonction boite-noire






2
paramètre 1
11.5
paramètre 2
−2
paramètre 3
Simulateur
mésure 1
?
?
mésure 2

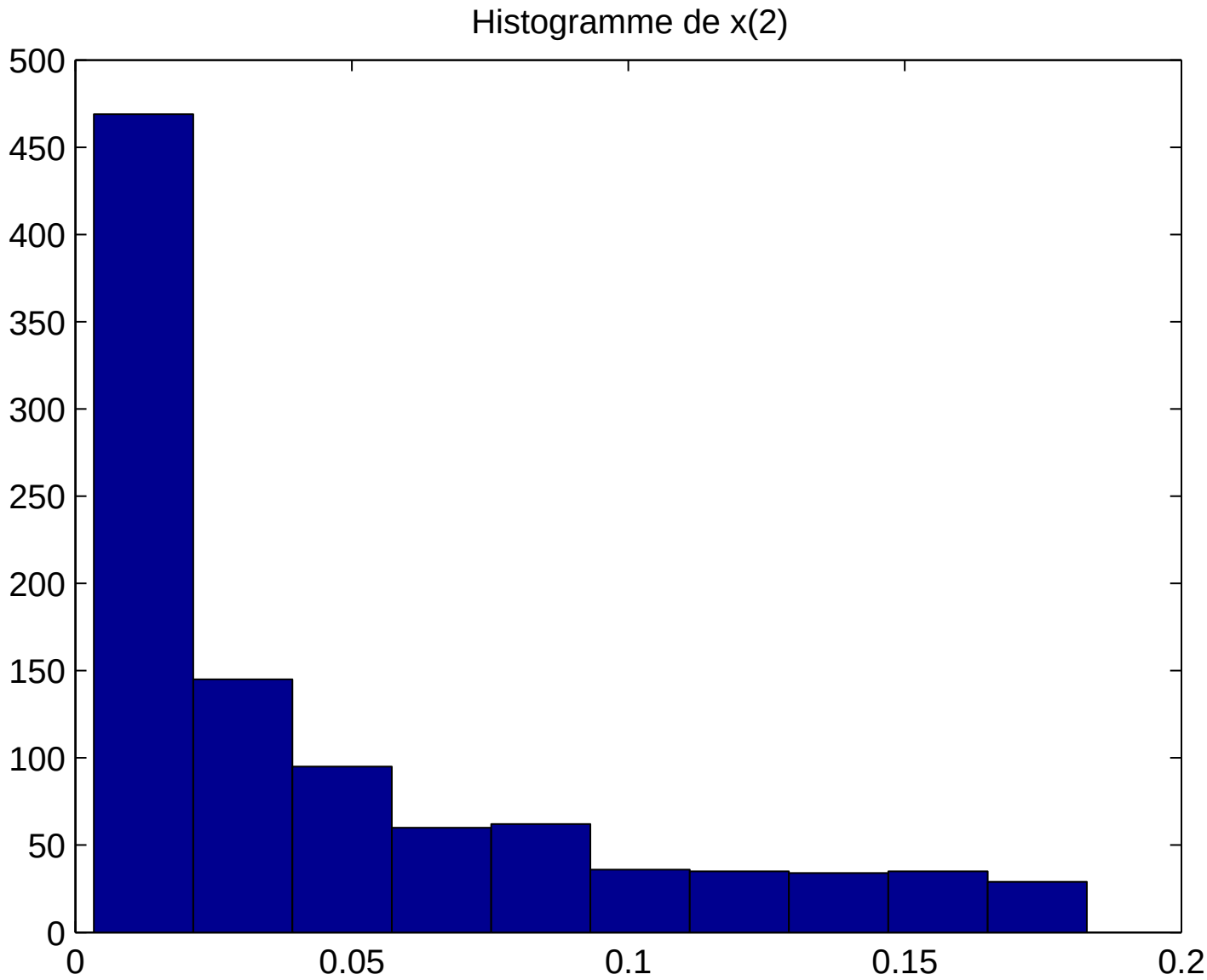
Exemple

- Supposons d'avoir une population dont la décroissance est décrite par l'équation différentielle

$$\dot{x} = Kx, \quad x(0) = 10$$

où la quantité $K < 0$ n'est pas connue de manière exacte, mais qui peut être décrite par une distribution de probabilité uniforme $\mathbf{K} \leftarrow \mathcal{U}(-4, -2)$.

- Puisque $\mathbf{K}$ est aléatoire, la quantité $\mathbf{x}(t)$ le sera aussi.
- Supposons de vouloir répondre aux questions suivantes: Aura quelle forme la distribution de la valeur $\mathbf{x}(t)$ pour $t = 2$? Sera-t-elle encore une distribution uniforme? Quelles moyenne et/ou variance a-t-elle? Quelle probabilité nous avons que $\mathbf{x}(t)$ est entre .1 et .2?
- Script OCTAVE `mc.m`.



1000 réalisations de la variable K.

Les méthodes Monte-Carlo

- La simulation Monte-Carlo peut être utilisée toutes les fois que le comportement d'un système peut être décrit par l'évolution d'une densité de probabilité (par exemple densité de $\mathbf{x}(t)$).
- Il est intéressant de remarquer que parfois même des problèmes **non random** peuvent être résolus par une approche stochastique. Un exemple est le calcul de la surface d'un lac.
- La méthode est basée sur (i) l'échantillonnage des quantités aléatoires, (ii) de la répétition d'une simulation déterministe (*trial*) pour chaque quantité échantillonnée et (iii) sur l'agrégation des résultats.
- MC est souvent considérée comme la méthode en *dernier ressort* puisqu'elle demande des ressources computationnelles assez consistantes.
- Ils existent des problèmes qui ne peuvent être résolus que par la méthode de MC et des problèmes qui sont plus faciles à résoudre par la méthode de MC.

Calcul surface d'un lac

- Considérons une zone rectangulaire ou carrée dont la surface $\mathrm{A}_{\mathrm{terrain}}$ est connue.
- Au sein de cette aire se trouve un lac dont la superficie $\mathrm{A}_{\mathrm{lac}}$ est inconnue.
- Pour trouver l'aire du lac, on demande à une armée de tirer N coups de canon de manière aléatoire et uniforme sur cette zone.
- On compte ensuite le nombre N_0 de boulets qui sont restés sur le terrain et ainsi le nombre $N - N_0$ de boulets qui sont tombés dans le lac.

- Puisque la probabilité qu'un boulet tombe dans une région d'aire A est proportionnelle à l'aire même, il suffit de calculer:

$$\frac{\mathrm{A}_{\text{terrain}}}{\mathrm{A}_{\text{lac}}} = \frac{N}{N - N_0} \Longrightarrow \quad \mathrm{A}_{\text{lac}} = \frac{(N - N_0)}{N} \cdot \mathrm{A}_{\text{terrain}}$$

  Si $\mathrm{A}_{\text{terrain}} = 1000$ m2 et l'armée tire $N = 500$ boulets et que $N - N_0 = 100$ projectiles sont tombés dans le lac alors l'estimation est: $\mathrm{A}_{\text{lac}} \approx 100/500 * 1000 = 200$ m2.

- D'après la loi des grands nombres, la qualité de l'estimation s'améliore en augmentant le nombre de tirs et *en s'assurant que les artilleurs ne visent pas toujours le même endroit, mais couvrent bien la zone*.

Estimation de π

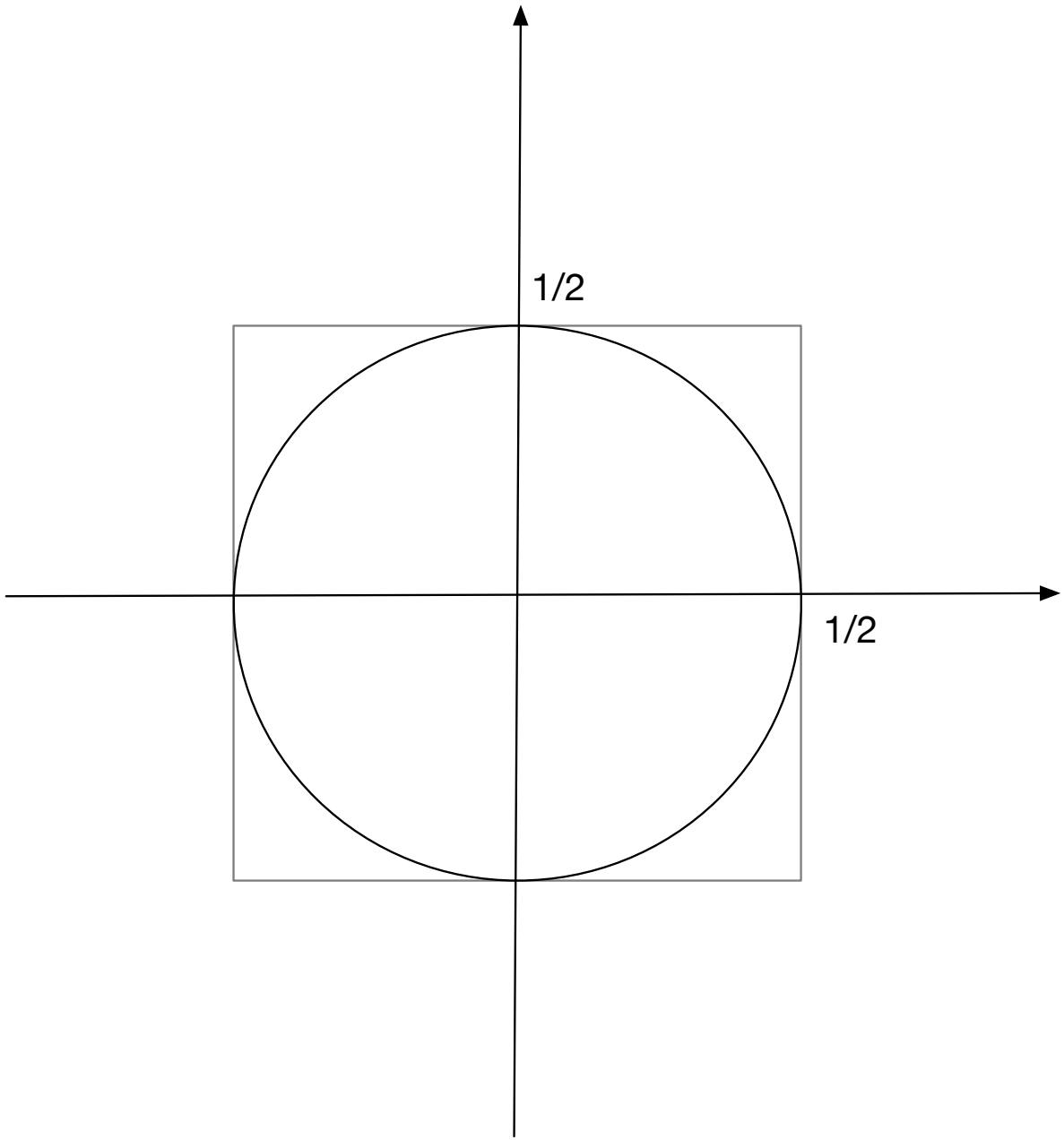
- Supposons que la zone est carrée de côté 1
- Soit le lac un cercle de rayon $1/2$
- Il s'ensuit que

$$\frac{\mathrm{A}_{\mathrm{terrain}}}{\mathrm{A}_{\mathrm{lac}}} = \frac{1}{\pi \left(\frac{1}{2}\right)^2} = \frac{4}{\pi}$$

- Donc

$$\frac{N}{N - N_0} \approx \frac{4}{\pi} \Rightarrow \pi \approx \frac{4(N - N_0)}{N}$$

- Script Octave `mcPi.m`


1/2
1/2

Composantes d'un algorithme MC

- **Description probabiliste:** un modèle stochastique du problème.
- **Générateur uniforme de nombres aléatoires:** un générateur de nombres aléatoires uniformément distribués sur $[0, 1]$.
- **Loi d'échantillonnage:** une technique pour échantillonner une distribution de probabilité générique.
- **Simulateur:** un simulateur déterministe qui renvoie l'output quand tous les paramètres sont connus.
- **Collecteur des outputs:** structure des données pour stocker tous les outputs de la simulation.
- **Analyseur de l'output:** ensemble de techniques statistiques qui permettent de tirer des conclusions à partir des données générées par le simulateur.
- **Estimateur d'erreur:** il associe à chaque quantité estimée à partir de l'output une indication sur l'erreur (par exemple en fonction du nombre de répétitions).

Variables aléatoires continues

- Considérons une variable aléatoire $\mathbf{z}$ réelle et continue.

Definition

La *fonction de répartition* de $\mathbf{z}$ est la fonction

$$F_{\mathbf{z}}(z) = \text{Prob}\{\mathbf{z} \leq z\} \tag{1}$$

Definition

La *fonction de densité* de $\mathbf{z}$ est la dérivée de la fonction de répartition:

$$p_{\mathbf{z}}(z) = \frac{dF_{\mathbf{z}}(z)}{dz} \tag{2}$$

- La probabilité d'une v.a. continue n'est pas définie pour des valeurs z ponctuelles, mais pour des intervalles

$$\text{Prob}\{a < z < b\} = \int_a^b p_{\mathbf{z}}(v)dv, \qquad \int_{\mathcal{Z}} p_{\mathbf{z}}(v)dv = 1$$

Moyenne et variance d'une v.a. continue

Espérance:

$$\mu = E[\mathbf{z}] = \int_l^h v p(v) dv$$

Variance:

$$\sigma^2 = \int_l^h (v - \mu)^2 p(v) dv$$

Moments d'ordre r :

$$\mu_r = E[\mathbf{z}^r] = \int_l^h v^r p(v) dv$$

Integral et MC

Supposons de vouloir calculer la quantité suivante

$$\theta = \int_a^b g(v)dv$$

pour laquelle une solution analytique n'est pas disponible. Nous pouvons résoudre ce problème déterministe de manière aléatoire en introduisant une variable aléatoire uniforme $x \sim \mathcal{U}(a, b)$ et en calculant

$$E[g(\mathbf{x})] = \int_a^b \frac{1}{b-a} g(v)dv$$

puisque

$$\theta = (b-a)E[g(\mathbf{x})]$$

Le problème Monte-Carlo

Considérons une variable aléatoire $\mathbf{x} \in \mathbb{R}^n$ de dimension n. Les méthodes Monte-Carlo essaient de résoudre les deux problèmes suivants

- **Echantillonnage:** comment générer une série de valeurs indépendants $\{x_i\}$, $i = 1, \ldots, N$, qui soient tous tirés à partir de la même densité de probabilité $p_{\mathbf{x}}(x)$.
- **Estimation:** étant donnée la fonction $g(x)$ et la variable aléatoire $\mathbf{x} \in \mathbb{R}^n$, estimer l'espérance de la variable aléatoire $g(\mathbf{x})$

$$\theta = E[g(\mathbf{x})] = \int g(v)p(v)dv$$

θ est la moyenne de la variable aléatoire $\mathbf{z} = g(\mathbf{x})$ qui représente la sortie du système.

La solution du premier problème rend possible la solution du deuxième.

L'estimation de $\theta = E[g(\mathbf{x})]$ est obtenue à partir des N valeurs x_i par

$$\hat{\theta} = \frac{1}{N} \sum_{i=1}^{N} g(x_i)$$

La loi des grands nombres nous garantit que

$$\lim_{N \to \infty} \hat{\theta} = \theta$$

Il est possible aussi de montrer que cet estimateur a des propriétés intéressantes d'un point de vue statistique. Il est non polarisé (en anglais unbiased), c.-à-d.

$$E[\hat{\boldsymbol{\theta}}] = \theta$$

et sa variance est

$$\text{Var}\left[\hat{\boldsymbol{\theta}}\right] = \frac{\sigma^2}{N}$$

où σ^2 est la variance de $g(\mathbf{x})$.

- La racine de la variance est une mesure de l'erreur moyenne commise une fois que nous estimons θ avec $\hat{\theta}$.
- Nous remarquons que la variance de $\hat{\boldsymbol{\theta}}$ est indépendante de la dimension n.
- La propriété la plus importante des méthodes MC est donc que **la précision de l'estimation renvoyée par Monte-Carlo est indépendante de la dimension n de la variable x en entrée**.
- Ceci signifie qu'indépendamment de la dimension de **x** quelques dizaines de tirages de **x** peuvent être suffisantes pour estimer θ d'une manière satisfaisante.
- Limitation: l'erreur moyenne est inversement proportionnelle à $\sqrt{N}$. Si on veut réduire l'erreur d'une taille 10 il faut multiplier N par 100.
- script Octave `estim.m`

Qu'est-ce que un nombre aléatoire?

- La simulation Monte-Carlo s'appuie sur la capacité de générer des nombres aléatoires. Un autre domaine qui est également intéressé par la capacité de générer des nombres aléatoires est celui de la cryptographie. Mais
  - Qu'est-ce que un nombre aléatoire?
  - Peut un nombre aléatoire être généré par une machine déterministe, comme l'ordinateur?
  - Est-il possible de générer des séquences *vraiment* aléatoires?
- Il n'existe pas une réponse universellement partagée à ce genre de questions.
- La raison est que plusieurs définitions d'aléatoire (en anglais *randomness*) existent.
- Selon le livre de Papoulis, deux définitions des nombres aléatoires peuvent être formulées.

Les définitions de nombre aléatoire

- **Définition conceptuelle:** une séquence de nombres x_i est appelée aléatoire (*random*) si elle est égale à l'échantillon d'une variable aléatoire **x**.
- **Définition empirique:** une séquence de nombres x_i est appelée aléatoire (*random*) si ses propriétés statistiques sont les mêmes que les propriétés d'une séquence de nombres obtenues dans une expérience aléatoire.

La définition empirique est une définition opérationnelle qui suggère une manière pratique pour répondre à la question si une séquence de nombres (par exemple générés par un ordinateur) est aléatoire. L'atout de cette définition est que la *randomness* d'une séquence peut être déterminée par des outils statistiques conventionnels (tests d'hypothèse,...).

Les propriétés des nombres aléatoires

Considérons une séquence de N nombres x_i tirés à partir d'une distribution uniforme de la variable aléatoire **x** sur $[0, 1]$. Cette séquence de nombres doit satisfaire deux propriétés:

1. **uniformité**, c.-à-d. ils sont équirépartis sur $[0, 1]$.
2. **indépendance**, c.-à-d. la valeur x_i ne doit avoir aucune relation ou dépendance par rapport à la valeur x_{i-1} ou plus en général avec les autres valeurs $x_j, j = 1, \ldots, i-1, i+1, \ldots, N$.

Il s'ensuit que

1. Si l'intervalle $[0, 1]$ est reparti en n sous-intervalles de la même taille, alors le nombre d'observations attendu dans chaque intervalle est N/n. Notons que N doit être suffisamment grand pour remarquer cette propriété.
2. La probabilité d'observer une valeur dans un certain intervalle est indépendante des valeurs tirées à l'avance.

Génération des nombres aléatoires

Les nombres aléatoires peuvent être générés par les manières suivantes

- **observation d'expériences de nature aléatoire:**
  - mesurer un dispositif physique qui présente certaines propriétés stochastiques (bruit d'une résistance, diode, compteur Geiger).
  - observer les lancements d'une pièce équilibrée afin de générer une séquence aléatoire de 0 (pile) et 1 (face);
  - extraire des billes d'une urne (loterie);
  - utiliser les chiffres décimaux de π
- **par l'ordinateur:** ceux-ci sont appelés nombres *pseudo-aléatoires*

L'approche pseudo-aléatoire est souvent préférable puisqu'elle est plus rapide, plus facilement gérable et permet la contrôlabilité et la répétitivité de l'expérience si nécessaire (par exemple pour des raisons de débogage)

Génération des nombres aléatoires

- *The generation of random numbers is too important to be left to chance.* (Robert R. Coveyou (1969))
- *Anyone who considers arithmetical methods of producing random digits, is, of course, in a state of sin. ... there is no such thing as a random number, there are only methods of producing random numbers, and a strict arithmetic procedure of course is not such a method.* (J. von Neuman (1951))

- Le but est de générer à l'ordinateur des échantillons d'une distribution de probabilité générique.
- Bien que les nombres pseudo-random ne soient pas *vraiment* aléatoires puisqu'ils ont été générés de manière déterministe, le but est de *créer une séquence qui puisse apparaitre comme non déterministe à quelqu'un qui ne connait pas l'algorithme sous-jacent*.
- D'un point de vue statistique, apparaître comme aléatoire revient à passer des tests statistiques (comme le test de Kolmogorov Smirnov).
- Toutefois, à l'état actuel, ceci n'est pas fait de manière directe pour une distribution $p_{\mathbf{z}}(\cdot)$ quelconque.
- On génère une séquence de nombres aléatoires uniformes et on applique une transformation pour obtenir une séquence qui appartient à la distribution désirée $p_{\mathbf{z}}(\cdot)$.

Nous verrons dans la suite comment générer des nombres aléatoires uniformes et les méthodes pour les transformer.

Propriétés d'un générateur pseudo-aléatoire

Un bon générateur de nombres aléatoires devrait satisfaire les critères suivants:

- **Distribution correcte.** Les points devraient être distribués de manière conforme à la distribution échantillonnée. Aussi, aucune corrélation (ni dépendance) ne devrait être observable entre les nombres générés séquentiellement.
- **Longue période.** Comme un générateur de nombres aléatoires est exécuté sur un ordinateur déterministe, il devient de facto un *algorithme déterministe*. Ses sorties sont inévitablement entachées d'une caractéristique absente d'une vraie suite aléatoire : la périodicité. Avec des ressources limitées (mémoire, nombre de registres, etc.), le générateur retrouvera le même état interne au moins deux fois. Afin d'éviter des corrélations non désirées, la quantité de nombres générée devrait toujours être inférieure à cette période.

- **Reproductibilité.** Afin de faciliter l'utilisation et le testing de code qui s'appuie sur la génération des nombres aléatoires, il est nécessaire de pouvoir reproduire une séquence déjà obtenue. Aussi il serait bien de pouvoir générer de nouveau une partie de la séquence sans recommencer du début. A cette fin il est nécessaire d'associer un état (*graine* ou *seed* en anglais) au générateur afin de pouvoir rétablir une configuration déjà vue.
- **Longues séquences non corrélées.** Pour des simulations de longues durées, il est important de pouvoir exécuter plusieurs sous-simulations indépendantes et de pouvoir les recombiner dans la suite en s'assurant que la propriété d'indépendance soit satisfaite.

- **Portabilité.** ceci ne signifie pas seulement que le code devrait être portable (p.ex. implémenté en un langage haut-niveau comme Fortran ou C), mais aussi qu'il devrait générer exactement la même séquence sur des machines différentes.
- **Efficacité.** La génération d'une séquence de nombre pseudo-aléatoires ne devrait pas demander des ressources de calcul excessives. La plupart des langages de programmation contiennent aujourd'hui un générateur de nombres pseudo-aléatoires.

La méthode de Von Neumann

En 1946, John von Neumann propose un générateur pseudo-aléatoire connu sous le nom de la méthode *middle-square* (carré médian). Très simple, elle consiste à prendre un nombre, à l'élever au carré et à prendre les chiffres au milieu comme sortie. Celle-ci est utilisée comme graine pour l'itération suivante.
Exemple: prenons le nombre 1111. Les étapes de l'algorithme sont:

1. $1111^2 = 1234321$
2. on récupère les chiffres du milieu: 3432. C'est la sortie du générateur.
3. $3432^2 = 11778624$
4. on récupère les chiffres du milieu: 7786, et ainsi de suite.

La méthode a surtout une importance historique. Elle est faible et la qualité des sorties dépend de la graine. Par exemple l'état 0000 produit toujours la même séquence et constitue un *état absorbant* de l'algorithme.

Générateur congruentiel linéaire

- Le générateur congruentiel linéaire (en anglais *linear congruential generator (LCG)*) a été proposé en 1951 par Lehmer pour générer des nombres pseudo-aléatoires selon une distribution uniforme $\mathcal{U}(0,1)$.
- Un générateur LCG génère une séquence z_i de nombres entiers non négatifs grâce à la formule de récurrence (aussi connue comme l' **algorithme de Lehmer**)

$$z_i = (az_{i-1} + c) \mod m \qquad i \geq 1$$

où a, c sont des entiers, m est un nombre premier très grand et $\mod$ dénote le reste de la division entière (par exemple $15 \mod 4 = 3$).

- La valeur initiale z_0 est appelée la graine (ou **seed**) de la séquence.
- La suite $u_i = \frac{z_i}{m} \in [0, 1]$ est la suite de nombres pseudo-aléatoires qui sont renvoyés par le générateur LCG.
- Si $c = 0$ le générateur est appelé générateur multiplicatif.
- La séquence de nombres entiers z_i prenne valeur entre 0 et $m - 1$; il s'ensuit que si nous exécutons l'itération pour m étapes, au moins deux valeurs z_i (et donc deux valeurs u_i) seront identiques.
- La séquence est donc périodique pour $N > m$ avec une période $m_0 \leq m$. Si la période est $m_0 = m$ alors on dit que le générateur est à *période maximale*.

- Une séquence périodique ne peut évidemment pas être une séquence aléatoire. Toutefois, pour des applications où le nombre de u_i généré est inférieur à m_0, la périodicité n'a aucun effet.
- Il est possible de montrer que seulement quelque ensemble de paramètres permet la génération di une séquence avec des bonnes propriétés statistiques.
- Des choix communs sont: $a = 7^5$, $c = 0$ et $m = 2^{31} - 1$.

Tests d'aleatoireté

Supposons d'avoir généré N nombres pseudo-aléatoires z_i par un générateur uniforme.

- Test des moments: si $\mathbf{z} \sim \mathcal{U}(0,1)$ alors les trois premiers moments devraient être $1/2$, $1/3$ et $1/4$, respectivement.
- Test chi-carré: supposons de diviser l'intervalle $[0,1]$ en s sous-intervalles I_j de taille égale. Soit

$$c_j = \{\#z_i \in I_j\}, \qquad j = 1, \ldots, s$$

la quantité de valeurs observée à l'intérieur du $j$ème intervalle. La quantité

$$C = \frac{s}{N} \sum_{j=1}^{s} \left(c_j - \frac{N}{s} \right)^2$$

est une mesure de concordance entre les valeurs observées et les valeurs attendues. Si les nombres avaient été générés par une variable aléatoire uniforme alors la quantité C devrait être distribuée comme une variable χ^2 avec $s-1$ degrés de liberté.

Tests d'aleatoireté (II)

- *Goodness of fit*: ceci calcule l'écart

$$K_N = \sup_z |\hat{F}_{\mathbf{z}}(z) - F_{\mathbf{z}}(z)|$$

  entre la fonction de répartition empirique

$$\hat{F}_{\mathbf{z}}(z) = \frac{\#z_i \leq z}{N}$$

  et la fonction de répartition attendue $F_{\mathbf{z}}$
- Test d'autocorrélation. Script Octave `test_ind.m`
- Test du maximum.
- Test des séquences binaires.

Transformation des variables aléatoires

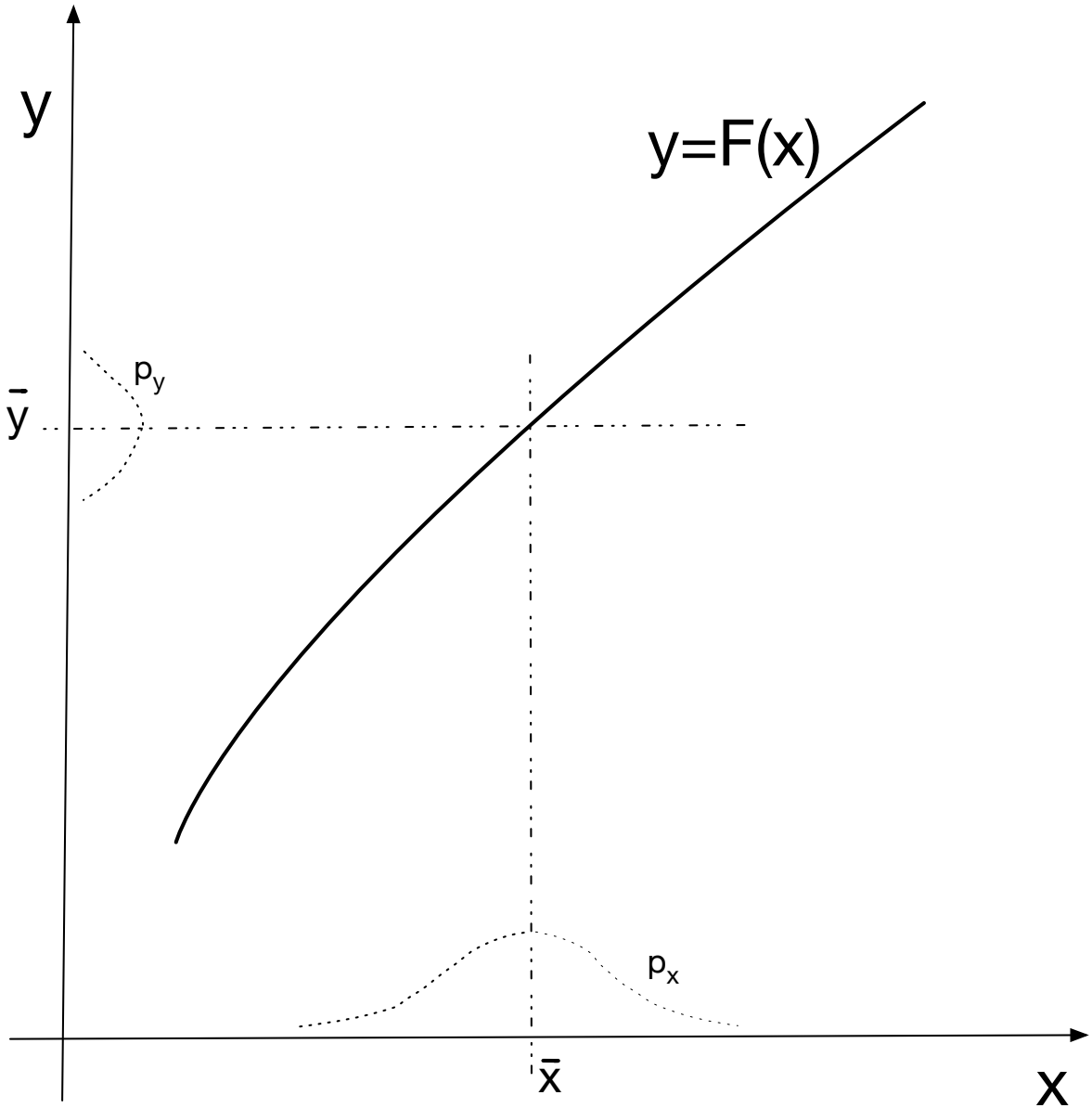
- Considérons une variable continue $\mathbf{x} \in X$ de densité $p_{\mathbf{x}}(x)$.
- Définissons une nouvelle variable aléatoire $\mathbf{y} \in Y$ en sort que $y = F(x)$ est une fonction monotone non décroissante.
- Que pouvons-nous dire sur la densité $p_{\mathbf{y}}(y)$?
- Selon la définition de densité de probabilité nous avons

$$p_{\mathbf{x}}(x)dx = p_{\mathbf{y}}(y)dy$$

et donc

$$p_{\mathbf{y}}(y) = \frac{p_{\mathbf{x}}(x)}{\frac{dF}{dx}}$$

si $F(x)$ est une fonction monotone non décroissante.


y
y=F(x)
p_y
$\bar{y}$
p_x
$\bar{x}$
x

Fonction de répartition

- Prenons comme fonction $F(x)$ la fonction suivante

$$F(x) = F_{\mathbf{x}}(x)$$

où $F_{\mathbf{x}} : \mathbb{R} \to [0, 1]$ est la fonction de répartition de $\mathbf{x}$.

- Puisque $F_{\mathbf{x}}(x)$ est une fonction monotone non décroissante et

$$\frac{dy}{dx} = \frac{dF_{\mathbf{x}}(x)}{dx} = p_{\mathbf{x}}(x)$$

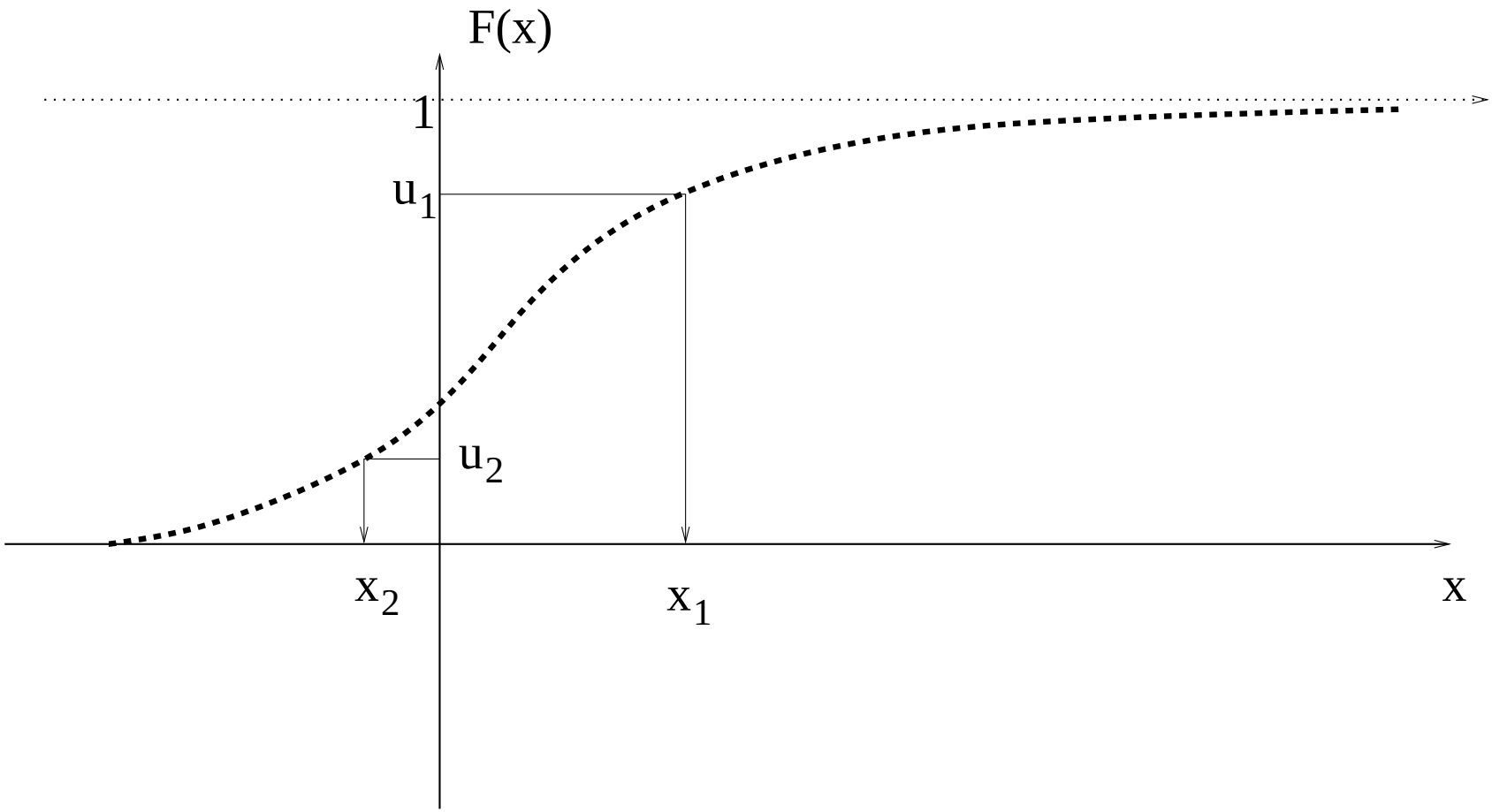
- Nous obtenons

$$p_{\mathbf{y}}(y) = \frac{p_{\mathbf{x}}(x)}{\frac{dF_{\mathbf{x}}}{dx}} = \frac{p_{\mathbf{x}}(x)}{p_{\mathbf{x}}(x)} = 1$$

- En d'autres termes la variable $\mathbf{y} = F_{\mathbf{x}}(\mathbf{x})$ est toujours distribuée de manière uniforme sur $[0, 1]$ pour quelconque densité de probabilité $p_{\mathbf{x}}(x)$.
- Ce résultat a un rôle clé pour la génération d'échantillons à partir de distributions autres que l'uniforme.

La méthode de la transformation inverse

- La méthode de la transformation inverse s'appuie sur la propriété de monotonie non décroissante de la fonction de répartition $F_{\mathbf{x}}(\cdot)$ et sur la propriété $\mathbf{y} = F_{\mathbf{x}}(\mathbf{x}) \sim \mathcal{U}(0,1)$.
- Soit x une variable aléatoire avec fonction de répartition $F_{\mathbf{x}}(x)$. La méthode de la transformation inverse d'échantillonnage de la variable $\mathbf{x}$ effectue les deux opérations suivantes
  1. un nombre u est tiré à partir de la distribution $\mathcal{U}(0,1)$
  2. la valeur $x = F^{-1}(u)$ est calculée.
- Cette règle connue comme la "Golden Rule for Sampling" a été proposée par von Neumann en 1947.


F(x)
1
u_1
u_2
x_2
x_1
x

Exemples de transformation inverse

- **Distribution uniforme** Soit $\mathbf{x} \sim \mathcal{U}(a, b)$ une variable aléatoire distribuée de manière uniforme entre a et $b \geq a$

$$F_{\mathbf{x}}(x) = \frac{x - a}{b - a}$$

Si u est un échantillon de $\mathcal{U}(0, 1)$ alors une réalisation de $\mathcal{U}(a, b)$ obtenue, selon la méthode inverse par

$$\frac{x - a}{b - a} = u \Leftrightarrow x = a + (b - a)u$$

Exemples de transformation inverse (II)

- **Distribution exponentielle**. Soit $\mathbf{x} \sim \mathcal{E}(\lambda)$ et

$$F_{\mathbf{x}}(x) = 1 - e^{-\lambda x}$$

Si u est une réalisation de la variable uniforme $\mathcal{U}(0,1)$ alors une réalisation de $\mathcal{E}(\lambda)$ est obtenue par

$$1 - e^{-\lambda x} = u \Leftrightarrow x = -\frac{1}{\lambda} \ln(1-u)$$

Il est important, toutefois, remarquer que pour plusieurs distributions (par exemple Gaussiennes) il n'existe pas une forme analytique de l'inverse de la fonction de répartition. Méthodes alternatives ou méthodes numériques (par exemple basées sur l'interpolation) doivent être appliquées dans ces cas.

Considérations méthodes MC

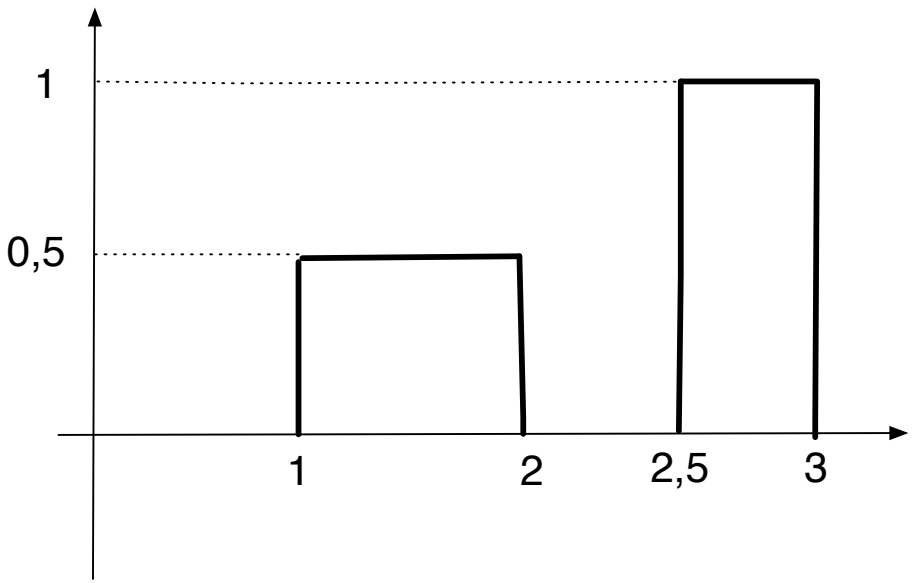
- La solution d'un problème par méthode MC est autant plus bonne que
  1. le nombre de répétitions est grand (calcul distribué est recommandé)
  2. la représentation probabiliste est correcte (par exemple en termes de densité et dépendance entre variables)
- Méthodes MC sont moins fiables quand nous essayons d'estimer la probabilité d'évènements rares, souvent dénotés *black swans* (p.ex. krach boursier, inondation, tremblements de terre).
- MC est un outil utile, mais il reste toujours un outil.

Question orale

Considérons le système dynamique à temps continu d'ordre 1

$$\dot{x} = 0.5x, \quad x \in \mathbb{R}$$

dont la condition $\mathbf{x}(0)$ à l'instant $t = 0$ est distribuée selon la densité de probabilité



Soit $U = [0.15, 0.6, 0.53, 0.9, 0.34, 0.17, 0.82, 0.08, 0.37, 0.03]$ une séquence de 10 nombres aléatoires tirés à partir d'une distribution uniforme entre 0 et 1.
L'étudiant devra estimer en utilisant la séquence U par Monte-Carlo

- la moyenne de la solution $\mathbf{x}(1)$,
- la probabilité que $\mathbf{x}(1) < 1$.