IFT 6561
Simulation: aspects stochastiques

Fabian Bastin
DIRO
Université de Montréal

Automne 2013

Variables aléatoires uniformes

But: produire des suites de nombres qui ont l'air d'être choisis complètement au hasard:

- suites de bits: 01101010011011010100110110010101000111...
- suite d'entiers de 0 à 100: 31, 83, 02, 72, 54, 26,...
- permutation aléatoire
- suite de nombres réels entre 0 et 1.

Mécanismes physiques

Lancer les dés, bouliers, roulettes...
bruit thermique dans les résistances de circuits electroniques, capteurs de radiations, autres mécanismes basés sur la physique quantique,
microsecondes de l'horloge de l'ordinateur, ou d'un temps d'accès au disque, etc.

Contient de la vraie entropie (incertitude), mais encombrant, pas facilement reproduisible, pas toujours fiable, peu ou pas d'analyse mathématique possible. Certains de ces mécanismes sont brevetés. Plusieurs sont disponibles commercialement. Pour ameliorer les propriétés statistiques: combiner des blocs de bits (XOR).

Générateurs algorithmiques (ou pseudo-aléatoires)

Une fois les paramètres et l'état initial du générateur pseudo-aléatoire (GPA) choisis, la suite produite est complètement déterministe.

Avantages: pas de matériel à installer, un logiciel suffit; souvent plus rapide; on peut facilement répéter la même séquence.

Désavantages: ne peut pas créer de l'entropie. Il y a nécessairement des dépendances entre les nombres en sortie.

GPAs

Qualités requises? Dépend des applications.

Jeux d'ordinateurs personnels: l'apparence suffit.

Simulation stochastique (Monte Carlo): on veut que les propriétés statistiques du modèle mathématique soient bien reproduites par le simulateur.
Générateurs algorithmiques

Loteries, machines de casinos, casinos sur internet,. . .: il ne faut pas que quiconque puisse obtenir un avantage pour inférer les prochains numéros ou encore des combinaisons plus probables. Conditions plus exigeantes que pour la simulation.
Générateurs algorithmiques + mécanismes physiques

Besoins pour la simulation stochastique

Cryptologie: encore plus exigeant. L'observation d'une partie de la sortie ne doit nous aider d'aucune manière à deviner quoi que ce soit dans le reste.
Générateurs algorithmiques non-linéaires avec paramètres aléatoires
Souvent: contraintes sur les ressources disponibles pour les calculs.

On utilise habituellement un GPA qui imite une suite U_0, U_1, U_2,... de variables aléatoires indépendantes de loi uniforme sur l'intervalle $(0, 1)$. Pour générer des v.a. selon d'autres lois, on applique des transformations à ces U_j.

Mythes

Mythe 1

Après au moins 60 ans à étudier les GPA et des milliers d'articles publiés, ce problème est certainement réglé et les GPA disponibles dans les logiciels populaires sont certainement fiables.

Mythe 2

Dans votre logiciel favori, le generateur a une periode supérieure à 21000. Il est donc certainement excellent!

Exemple 1. $u_n = (n/2^{1000}) \mod 1$ pour $n = 0, 1, 2, \ldots$
Exemple 2. Subtract-with-borrow.

Générateur $U(0,1)$: principe de base

Définir une fonction de transition

$$f : \mathcal{S} \to \mathcal{S},$$

où $\mathcal{S}$ est l'espace d'état, de cardinalité finie.

L'état initial (germe, semence): s_0. Récurrence:

$$s_n = f(s_{n-1}).$$

Supposons de plus que f est périodique pour tout $n \geq \tau$ connu (souvent égal à 0), de période $\rho \leq \#\mathcal{S}$: $s_{n+\rho} = s_n$, $\forall n \geq 0$. On supposera $\tau = 0$

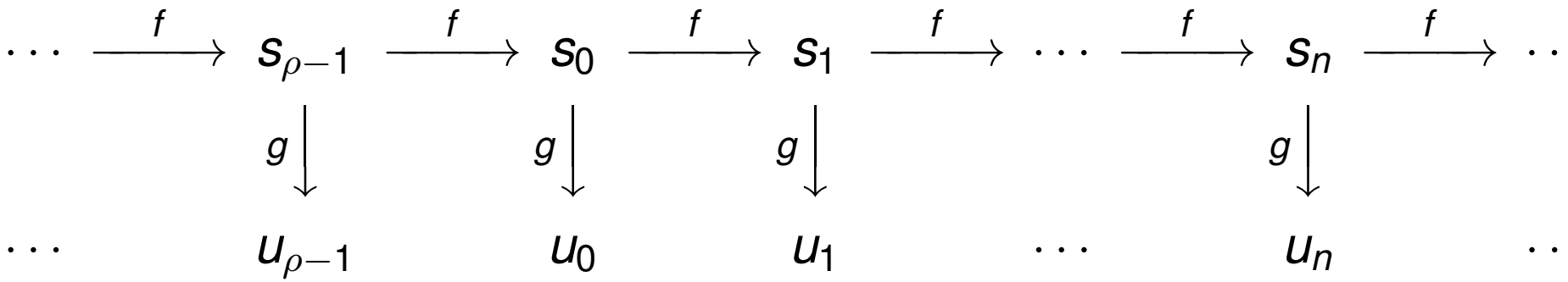
Espace de sortie: $\mathcal{U} = (0,1)$.

Générateur $U(0,1)$: principe de base

Fonction de sortie

$$g : \mathcal{S} \to \mathcal{U}$$

transforme l'état s_n dans la valeur de sortie u_n.



Comment choisir f et g?

Buts: grand ρ, bonne uniformité, comportement "aléatoire".

Objectif

En observant seulement $(u_0, u_1, \ldots)$, il doit être difficile de distinguer cette suite de la réalisation d'une suite de v.a. i.i.d. uniformes sur $\mathcal{U}$.

Utopie: ne pas pouvoir distinguer mieux qu'en tirant à pile ou face. Autrement dit, que la suite passe tous les tests statistiques imaginables. Cela est impossible!.

On veut aussi: vitesse, facilité d'implantation, suites reproduisibles. Compromis entre vitesse / bonnes propriétés statistiques / (im)prévisibilité. Si l'état initial s_0 est choisi au hasard, le GPA est comme une roulette géante: pour générer t nombres aléatoires, on tourne la roulette pour choisir s_0, puis on retient $u = (u_0, \ldots, u_{t-1})$.

Machines de casinos et loteries: on réinitialise s_0 très souvent.

La loi uniforme sur $[0,1]^t$

Choisir s_0 au hasard correspond à choisir un point au hasard dans l'espace échantillonnal

$$\Psi_t = \{u = (u_0, \ldots, u_{t-1}) = (g(s_0), \ldots, g(s_{t-1})), s_0 \in \mathcal{S}\},$$

qui peut être interprété comme une approximation de $[0,1]^t$.

Critère: Ψ_t doit recouvrir $[0,1]^t$ très uniformément pour t jusqu'à (disons) t_0. Il nous faut une mesure d'uniformit e de Ψ_t (ou une mesure de discr epance entre la loi empirique de Ψ_t et la loi uniforme). Plusieurs définitions possibles.

La loi uniforme sur $[0, 1]^t$

Important: doit être facilement calculable sans générer les points.

Pour cela, il faut bien comprendre la structure mathématique de Ψ_t. Pour cette raison, la plupart des GPA utilisés en simulation sont basés sur des récurrences linéaires (simples).
Pourquoi ne pas insister que Ψ_t lui-même ressemble à un ensemble de points choisis au hasard (e.g., ne soit pas trop uniforme)? En fait, on veut cela seulement pour la fraction infime de Ψ_t que l'on utilise.

Généralisation: mesurer l'uniformité de $\Psi_I = \{(u_{i_1}, \ldots, u_{i_t}) \mid s_0 \in \mathcal{S}\}$ pour une classe choisie d'ensembles d'indices (non successifs) de forme $I = \{i_1, i_2, \ldots, i_t\}$. Pour une famille donnée $\mathcal{J}$ de sous-ensembles de $\{0, 1, \ldots, t\}$, on s'assure que Ψ_I est suffisamment uniforme pour tout $I \in \mathcal{J}$.

Générateur congruentiel linéaire (GCL)

Dès 1948 furent introduits des générateurs de la forme

$$ax + c \mod m.$$

En supposant tout d'abord que c vaut 0 (comme dans l'approche proposée par Lehmer), la période maximale est $m - 1$ et est atteinte si et seulement si m est premier et a est une racine primitive de m.

r est une racine primitive de m si les puissances de r (1, r, r^2, r^3,...) génèrent tous les entiers non-nuls modulo m.

Générateur congruentiel linéaire (GCL)

Puisqu'il y a $m-1$ entiers non nuls, ceci signifie que les premières $m-1$ puissances de r doivent être différentes, modulo m.

De manière équivalente, nous pouvons parler de l'ordre de r. L'ordre d'une racine r de m est le plus petit entier (strictement) positif x tel que $r^x = 1 \mod m$.

r est une racine primitive si son ordre est $m-1$. Il est possible de montrer que ceci équivaut à exiger que $a^{(m-1)/p} - 1$ est un multiple de m pour chaque facteur premier p de $m-1$, ou encore le plus petit entier l pour lequel $r^l - 1$ est divisible par m est $l = m-1$.

Générateur congruentiel linéaire (GCL)

Les générateurs congruentiels linéaires qui remplissent ces conditions sont appelés GCL's multiplicatifs à modulus premier.

Notons que la condition m premier suffit pour garantir l'existence d'un générateur de période maximale, en vertu du théorème ci-dessous.

Théorème.
Si m est premier, il existe une racine primitive pour m.

Il n'existe malheureusement pas de méthode simple pour calculer ces racines.

GCL: exemple

Si $m = 7$, alors 3 est une racine primitive de m car les puissances de 3 modulo 7 sont 1, 3, 2, 6, 4, 5, c'est-à-dire chaque entier strictement compris entre 0 et 7. Mais 2 n'est pas une racine primitive de m car les puissances de 2 modulo 7 sont 1, 2, 4, 1, 2, 4, 1, 2, 4,...

Générateur congruentiel linéaire (GCL)

Si $c \neq 0$, il est possible d'obtenir une période égale à m, sous les conditions exposées dans le théoréme ci-dessous. *Le GCL a une période pleine si et seulement si les trois conditions suivantes tiennent:*

1. *le seul entier positif qui divise de manière exacte à la fois m et c est 1;*
2. *si q est un nombre premier qui divise m, alors q divise* $a - 1$*;*
3. *si 4 divise m, alors* 4 *divise* $a - 1$*.*

Générateur standard minimal

Park et Miller ont proposé un générateur standard qu'ils ont appelé le Standard Minimal générateur Standard Minimal, aprés avoir testé divers générateurs connus au moment de leur étude.

Bien qu'il suffise pour les applications simples, les générateurs présentés dans les sections suivantes le surpassent largement, et par conséquent, il est déconseillé de l'utiliser pour des simulations complexes.

Le Standard Minimal est un générateur congruentiel linéaire défini par la récurrence

$$x_{n+1} = 16807 x_n \mod (2^{31} - 1).$$

Implantation de générateurs congruentiels linéaires

Une difficulté principale est de calculer $ax \mod m$ pour de grands m, ce qui entraîne des risques de débordement de registres.

Première approche. Factorisation approximative.

Cette méthode est valide si

$$a^2 < m$$

ou

$$a = \lfloor m/i \rfloor,$$

avec $i^2 < m$, et procède par des calculs sur des entiers.

Factorisation approximative

Précalculons $q := \lfloor m/a \rfloor$ et $r := m \mod a$, puis

$$y := \lfloor x/q \rfloor;$$
$$x := a(x - yq) - yr.$$

Si $x < 0$, nous posons $x := x + m$. Justification:

$$\begin{aligned} ax \mod m &= (ax - \lfloor x/q \rfloor m) \mod m \\ &= (ax - \lfloor x/q \rfloor (aq + r)) \mod m \\ &= (a(x - \lfloor x/q \rfloor q) - \lfloor x/q \rfloor r) \mod m \\ &= (a(x \mod q) - \lfloor x/q \rfloor r) \mod m. \end{aligned}$$

Sous les conditions posées, il est immédiat de noter que toutes les quantités intermédiaires demeurent entre $-m$ et m.

Factorisation approximative: implantation

En C, la procédure peut s'exprimer comme suit:

```
long q, r, y;

q = m/a;
r = m%a;

y = x/q;
x = a*(x-y*q)-y*r;

if (x < 0) x += m;
```

Calculs en point flottant, double précision.

La procédure est valide si tous les entiers à considérer peuvent être représentés de manière exacte en passant en calcul flottant. En particulier, si la double précision fait appel à 64 bits, et suit la norme IEEE, la procédure suivante est correcte si $am < 2^{53}$:

double m, a, x, y; int k;

$y = a * x$; $k = \lfloor y/m \rfloor$; $x = y - k * m$;

Décomposition en puissances de 2.

Supposons que $a = \pm 2^q \pm 2^r$ et $m = 2^e - h$ pour h petit. Dans ce cas,

$$ax \mod m = \pm 2^q x \mod m + \pm 2^r x \mod m.$$

Pour calculer $y = 2^q x \bmod m$ (le calcul de $2^r x$ est similaire), nous décomposons x en $x_0 + 2^{e-q} x_1$.

$x =$

q bits	$(e-q)$ bits
x_1	x_0

Décomposition en puissances de 2.

Pour $h = 1$ (Wu, 1997), on obtient y en permutant x_0 et x_1. En effet,

$$\begin{aligned} 2^q x \mod m &= 2^q(x_0 + 2^{e-q}x_1) \mod (2^e - 1) \\ &= 2^q x_0 + [2^e x_1 \mod (2^e - 1)] \\ &= 2^q x_0 + x_1. \end{aligned}$$

Décomposition en puissances de 2.

Pour $h > 1$ (L'Ecuyer et Simard 1999), nous avons de la même manière

$$y = 2^q(x_0 + 2^{e-q}x_1) \mod (2^e - h) = (2^q x_0 + h x_1) \mod (2^e - h).$$

Si $h < 2^q$ et $h(2^q - (h+1)2^{-e+q}) < m$, comme $x_0 \leq 2^{e-q}$ et $x_1 \leq 2^q$, nous avons

$$2^q x_0 \leq 2^e - 2^q < m.$$

De plus, étant donné que $2^{e-q}x_1 \leq m - 1$, nous avons

$$\begin{aligned} h x_1 \leq h(m-1)/2^{e-q} &= h(2^e - h - 1)/2^{e-q} \\ &= h(2^q - (h+1)2^{-e+q}) < m, \end{aligned}$$

et par conséquent chaque terme est strictement inférieur à m. L'opération modulo revient à soustraire m si la somme est $\geq m$.

Décomposition en puissances de 2: implantation

```
#define m      1073741789     /* 2^30 - 35 */
#define h      35
#define q      15
#define emq    15             /* e - q */
#define mask1  32767          /* 2^(e-q) - 1 */
#define r      13
#define emr    17             /* e - r */
#define mask2  131071         /* 2^(e-r) - 1 */
#define norm   1.0/m

long x;
```

Décomposition en puissances de 2: implantation

```
double axmodm () {
  unsigned long k, x0, x1;

  x0 = x & mask1;
  x1 = x >> emq;
  k = (x0 << q) + h*x1;

  x0 = x & mask2;
  x1 = x >> emr;
  k += (x0 << r) + h*x1;

  if (k < m) x = k;
  else if (k < 2*m) x = k-m;
  else x = k - 2*m;

  return x*norm;
}
```

Décomposition en puissances de 2.

L'Ecuyer et Simard ont toutefois démontré que ces générateurs présentent des faiblesses statistiques s'ils sont utilisés de manière directe.