IFT 3245
Simulation et modèles

Fabian Bastin
DIRO
Université de Montréal

Automne 2012

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: s_0. Récurrence:

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

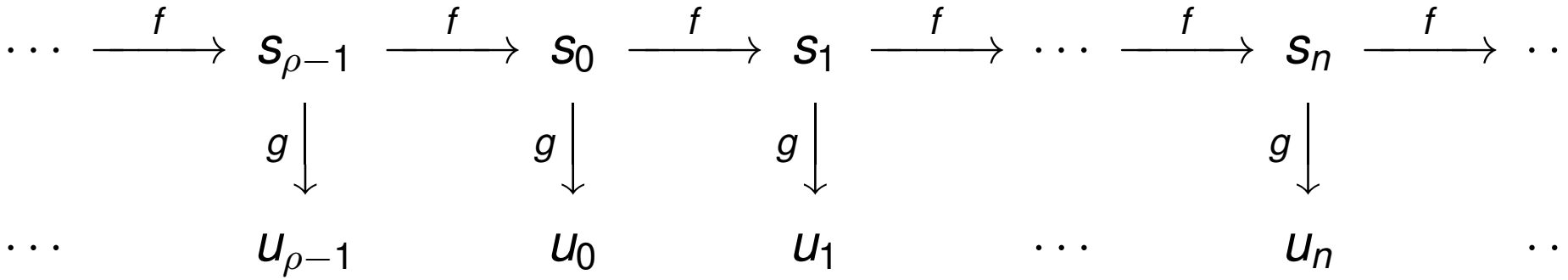
Supposons de plus que f est périodique pour tout $n \geq \tau$ connu (souvent égal à 0), de période $\rho \leq \#\mathcal{S}$.

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

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”.

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

Exemple.
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

$$hx_1 \leq h(m-1)/2^{e-q} = h(2^e - h - 1)/2^{e-q} = h(2^q - (h+1)2^{-e+q}) < m,$$

et par conséquent chaque terme est strictement inférieur à m. L'opération modulo revient dès lors 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.