Simulation de loi Gamma par méthode de rejet sous CUDA

On veut simuler des variables aléatoires de loi Gamma(a) pour $a \in (0,1)$, dont une densité est donnée par

$$f_a(t) = \frac{1}{\Gamma(a)} t^{a-1} e^{-t} 1_{[0,\infty)}(t).$$

1 La méthode de rejet.

Considérons I un intervalle réel et $f : I \to \mathbb{R}$, $g : I \to \mathbb{R}$ deux densités de probabilités définies sur cet intervalle. On suppose qu'il existe $M \geq 1$ telle que

$$\forall x \in I, \quad \frac{f(x)}{g(x)} \leq M.$$

On suppose que l'on sait simuler une suite i.i.d. de variables aléatoires $(Y_i)_i$ de loi de densité donnée par g, ainsi qu'une suite de variables de loi uniforme sur $[0,1]$. Alors la *méthode de rejet* permet de simuler une variable X de loi donnée par la densité f.

L'algorithme est le suivant :

- Etape 1 : On simule Y v.a. de densité g et U v.a. de loi uniforme sur $[0,1]$ avec $Y \perp\!\!\!\perp U$
- Etape 2 :
  - Si $U \leq \frac{f(Y)}{Mg(Y)}$, on pose $X = Y$ et on arrête.
  - Si $U > \frac{f(Y)}{Mg(Y)}$, on recommence à l'étape 1.

La méthode de rejet peut être assez lente si l'on a besoin de répéter de nombreuses fois l'itération, c'est à dire si on tombe souvent sur la condition $U > \frac{f(Y)}{Mg(Y)}$.

La probabilité de s'arrêter à chaque itération est $P(U \leq \frac{f(Y)}{Mg(Y)}) = 1/M$. On a donc intérêt à prendre $M \geq 1$ le plus petit possible.

2 Application à la loi Gamma.

Pour $a \in (0,1)$ considérons la densité de la loi de Weibull (a), égale à

$$g_a(t) = at^{a-1} \exp(-t^a) 1_{[0,\infty)}(t),$$

et dont la fonction de répartition associée G_a est donnée par

$$G_a(t) = \int_0^t a s^{a-1} \exp(-s^a) \mathrm{d}s = 1 - e^{-t^a}, \text{ pour } t \geq 0,$$

et $G_a(t) = 0$ pour $t < 0$.

On sait que si V est une v.a. de loi uniforme sur $[0, 1]$ alors $Y = G_a^{-1}(V)$ est une variable de loi de Weibull (a).

Exercice : Calculer G_a^{-1}.

Comme nous savons simuler des v.a. de loi uniforme sur ordinateur, nous sommes donc capable de simuler des v.a. de loi de Weibull.

Pour simuler des loi Gamma nous allons utiliser la méthode de rejet en nous basant sur les tirages de la loi de Weibull. Pour pouvoir l'appliquer, il est nécessaire de majorer f_a/g_a. Or, nous pouvons écrire, pour $t > 0$,

$$\frac{f_a(t)}{g_a(t)} = \frac{1}{a\Gamma(a)} e^{(t^a - t)} \leq \frac{e}{a\Gamma(a)}, \tag{1}$$

en ayant utilisé que si $t \geq 1$ alors $t^a - t \leq 0$, et si $t \in [0, 1]$ alors $t^a - t \leq t^a \leq 1$. L'equation (1) permet donc de voir que la méthode de rejet peut s'appliquer avec $M = \frac{e}{a\Gamma(a)}$.

3 Implémentation CUDA/PyCUDA.

Rappelons l'objectif :

- On veut créer un kernel qui génère un grand nombre de variables de loi Gamma(a) et les stocke dans un tableau.
- La méthode pour générer une v.a. de loi Gamma est la méthode de rejet.
- Chaque thread générera une seule variable aléatoire de loi Gamma par méthode de rejet et la stockera dans le tableau.

Exercice : Ecrire un kernel qui prend en entrée : `n` la longueur du tableau, `res` l'adresse du tableau pour stocker les resultats, `a` le paramètre de la loi Gamma et `M` le majorant utilisé dans la méthode. Ce kernel rempli le tableau `res` avec des simulations $X_1, \ldots, X_n$ de loi Gamma(a). Chaque thread simule une seule variable de loi Gamma et la stocke une case du tableau `res`.

Rq : Pour importer la fonction Gamma sous Python :
`from scipy.special import gamma`
et la fonction qui calcule $x \mapsto log(Gamma(x))$ est de base présente dans CUDA : `lgamma()`.

Exercice : Ecrire un kernel, modification du précédent, qui permet de simuler les valeurs de $Z_1, \ldots, Z_n$ où $Z_i = 1_{\{X_i \leq x\}}$ et x est un paramètre réel en plus que l'on passera à ce kernel.

Exercice : Ecrire sous Python une fonction qui permet de calculer, par Monte Carlo, la valeur de $P(X \leq x)$ lorsque X suit une loi Gamma(a).