IFT-6561: SIMULATION **Automne 2014**

Prof. Pierre L'Ecuyer

DEVOIR 1

Devoir à remettre le *mardi 23 septembre 2014, 9h30, avant le début du cours*. Les numéros indiqués sont ceux des notes de cours. S.v.p. placez les exercices dans le bon ordre sur vos copies.

Les devoirs doivent être faits individuellement: un devoir par étudiant. Pour les expériences numériques, il est très important de bien expliquer tout ce que vous faites et d'expliquer le sens de vos résultats (discussion). Dans la correction, on accordera davantage d'importance à la clarté des explications qu'aux programmes informatiques et aux résultats comme tels. Dans ce sens, remettre uniquement un programme qui donne les bons résultats ne suffit pas et vous donnera peu ou pas de points. Le jour de la remise, envoyez aussi une copie de vos programmes par courriel au correcteur, pour qu'il puisse les essayer au besoin. Attention au plagiat: il n'est pas permis de copier et/ou modifier les programmes ou les solutions d'un(e) autre étudiant(e).

— 1 —

Exercice 1.1 des notes.

— 2 —

Exercise 1.4. Vous pouvez utiliser (*sans la modifier*) la classe `mathematicaSWB.java`, qui implante ce générateur et qui se trouve dans le répertoire `http://www.iro.umontreal.ca/~lecuyer/ift6561/java/examples09/` sur la page web du cours.

— 3 —

Voir l'exercice 1.18 des notes. Il n'y a pas de simulation à implanter pour cette question. L'idée est de comprendre ce qui se passe si on estime le volume d'une sphère de rayon 1 en s dimensions par la méthode Monte Carlo. Il s'agit bien sûr d'un exercice pûrement académique, puisqu'on connait déjà le volume de cette sphère, mais il permet de comprendre un type de difficulté qui survient dans de nombreuses applications pratiques. Pour estimer le volume, on tire n points au hasard dans le cube $(0,1)^s$, on calcule la fraction $\tilde{p}_n$ de ces points qui tombent dans la sphère (pour estimer la fraction p du cube occupé par la sphère), et l'estimateur du volume est $\tilde{\mu}_n = 2^s \tilde{p}_n$.

(a) Prouvez que cet estimateur est sans biais. Donnez aussi (avec preuve) des formules exactes pour la variance et l'erreur relative de cet estimateur, en fonction de s.

(b) Pour avoir une erreur relative constante en fonction de s, disons inférieure à 0.01 pour tout s, à quelle vitesse (ou de quelle manière) doit-on augmenter n en fonction de s, lorsque s est grand? Donnez une formule pour n en fonction de s et expliquez ce que cela implique pour les grandes valeurs de s.

(c) Calculez les valeurs numériques de p, V_s, et de l'erreur relative au carré de $\tilde{\mu}_n$, $\mathrm{RE}^2[\tilde{\mu}_n]$, pour $s = 2, 5, 10, 20$.

Aide: Pour cette question, vous pouvez utiliser le fait que le volume d'une sphère de rayon 1 en s dimensions est $V_s = \pi^{s/2}/\Gamma(1+s/2)$, où $\Gamma(\cdot)$ est la *fonction gamma*, définie par $\Gamma(1/2) = \sqrt{\pi}$, $\Gamma(1) = 1$, et $\Gamma(s+1) = s\Gamma(s)$. Lorsque s est grand, l'approximation de Stirling donne

$$\Gamma(1 + s/2) \approx \lfloor s/2 \rfloor ! \approx (\pi s)^{1/2}(s/(2e))^{s/2}.$$

— 4 —

Le but de cet exercice est de vous familiariser avec la librairie SSJ et son utilisation. On vous demande d'écrire une classe `TandemQueue` permettant de simuler le système de files en tandem de l'exemple 1.7 des notes, avec "production blocking", en utilisant les récurrences données. Pour simplifier, vous pouvez supposer que les durées inter-arrivés A_i suivent une loi exponentielle de taux λ (moyenne $1/\lambda$), que les durés de service à la station j suivent une loi exponentielle de taux μ_j, et que toutes ces variables aléatoires sont indépendantes. (Dans la vraie vie il faudra bien sûr faire une analyse statistique des données pour identifier des lois de probabilité appropriées et estimer leurs paramètres.)

Votre classe devra avoir un constructeur qui prend en entrée les paramètres du modèle, soit m, λ, le vecteur des μ_j, et le vecteur des c_j. Il y aura aussi une méthode `simulateFixedNumber` qui simule un nombre fixe N_c the clients et une méthode `simulateFixedTime` qui simule le système pour un horizon de temps T fixé (après T, il n'y a plus d'arrivées, mais tous les clients déjà arrivés doivent poursuivre leur progression dans le système et comptent dans les statistiques). Dans les deux cas, on suppose que le système est initialement vide. Chacune de ces méthodes prendra en entrée deux `RandomStream`'s, l'un qui servira à générer les A_i et l'autre pour les $S_{i,j}$. Ces méthodes prendront aussi en paramètres des collecteurs statistiqus de type `Tally` pour recueillir le temps total d'attente et le temps total de blocage à chaque station j.

Pour tester cette classe et faire vos expériences, vous invoquerez le constructeur et les méthodes de votre classe `TandemQueue` à partir d'une autre classe (séparée, dans un autre fichier). L'idée est qu'une fois le "simulateur" `TandemQueue` implanté et testé, on ne doit plus jamais modifier son code lors de son utilisation. On n'utilise que les méthodes de son interface.

On vous demande ensuite de faire les expériences suivantes avec votre simulateur. Construisez un modèle avec $\lambda = 1$, $m = 3$, $\mu_1 = 1.5$, $\mu_2 = \mu_3 = 1.2$, $c_1 = \infty$, $c_2 = 4$, et $c_3 = 8$. Simulez ce modèle pour un horizon de temps $T = 1000$ et calculez l'attente totale et le temps total de blocage à chacune des trois stations (sauf à la dernière, où il n'y a pas de blocage). *Notez que ce ne sont pas les mêmes mesures que dans l'algorithme des notes.* Répétez cette simulation $n = 1000$ fois, puis calculez la valeur estimée et un intervalle de confiance à 95% pour l'espérance de chacune des 5 quantités calculés (temps total d'attente et temps de blocage à chaque station). Faites aussi tracer des histogrammes des 1000 valeurs observés pour chacune de ces 5 quantités. Discutez ce que vous observez; par exemple où observe-t-on davantage d'attente ou de blocage?

Important: vous devez bien expliquer et documenter vos programmes. Pour cela, vous pouvez prendre exemple sur la documentation des classes de SSJ.