

Prof. Pierre L'Ecuyer

Travail de fin de session

Travail final, à remettre au plus tard **mardi le 20 décembre 2011 à midi**, à Richard Simard au local 3380, ou dans le casier du professeur au secrétariat du DIRO (attention, le secrétariat ferme un peu avant midi).

Le nombre de points accordé à chaque question est indiqué entre parenthèses. Dans la correction, nous prenons en compte non seulement les résultats, mais aussi (et surtout) la clarté et l'exhaustivité des explications.

— 1 —

(30 points)

Dans l'exemple 6.66 des notes de cours, supposons que les C_j suivent la loi $\text{Gamma}(1/2, 1/4)$ et que le taux d'arrivée est $\lambda = 1$. La fonction génératrice des moments pour la loi gamma, dans la notation des notes de cours, est $M(t) = (1 - t/\lambda)^{-\alpha}$.

(a) Quelle est la valeur de θ^* en fonction de c dans ce cas, en supposant que $c > 2$? Aide: Vous allez probablement trouver une équation de troisième degré à résoudre, mais en simplifiant des termes, on peut se ramener à une équation quadratique, qui se résout directement par une formule.

(b) Implantez cette méthode et estimez la probabilité de ruine avec une erreur relative d'au plus 1% pour $R(0) = 200$ et $c = 3$, puis $c = 5$. Pour générer les variables selon la loi gamma, vous pouvez utiliser une méthode de rejet plus rapide (voir `GammaRejectionLoglogisticGen` dans SSJ) plutôt que l'inversion. (Note: Le programme `RuinIS.java` sur la page du cours fait cela pour l'exemple 1.45 des notes).

(c) Donnez une estimation du nombre n de répétitions de la simulation qu'il faudrait faire, avec une méthode de Monte Carlo standard (sans IS) et les paramètres donnés en (b), pour estimer la probabilité de ruine avec une erreur relative d'au plus 1%. Aide: bien sûr, vous ne pouvez pas estimer cette valeur en faisant des simulations directement sans IS.

(d) Pourquoi ne peut-on pas implanter la technique décrite dans cet exemple si les C_j suivent une loi lognormale?

(70 points)

Pour cette question, je vous réfère à l'article de Mike Giles, "Multilevel Monte Carlo Path Simulation," *Operations Research*, 56, 3 (2008) 607–617. Vous pouvez aussi consulter les articles subséquents du même auteur sur son site web: <http://people.maths.ox.ac.uk/gilesm/>, dans la section Multilevel Monte Carlo Research.

As in section 2.15 of the class notes, we consider a stochastic process $\mathbf{X} = \{\mathbf{X}(t), t \geq 0\}$ in \mathbb{R}^d , whose trajectory obeys the stochastic differential equation (SDE)

$$d\mathbf{X}(t) = \boldsymbol{\mu}(\mathbf{X}(t), t)dt + \mathbf{A}(\mathbf{X}(t), t)d\mathbf{B}(t), \quad (1)$$

with $\mathbf{X}(0) = \mathbf{x}_0 \in \mathbb{R}^d$, where \mathbf{B} is a standard Brownian motion in d dimensions, $\boldsymbol{\mu}(\mathbf{X}(t), t)$ is a d -dimensional vector, and $\mathbf{A}(\mathbf{X}(t), t)$ is a $d \times d$ matrix.

For a given time horizon T , we have a (random) payoff

$$P = f(\{\mathbf{X}(t), 0 \leq t \leq T\}),$$

which is a function of the process trajectory over the entire time interval $[0, T]$. Our goal is to estimate the expected payoff, $\mathbb{E}[P]$, by Monte Carlo. For this exercise, we will assume that P cannot be generated from its exact distribution. We will discretize the time and use the Euler method with time step h to generate an approximate skeleton of the process \mathbf{X} over $[0, T]$, and compute a corresponding approximate payoff. To reduce the bias, we would like to take h as small as possible, but the computing cost is usually proportional to the number of time steps, T/h , so a very small h means large CPU times.

In his paper, Giles developed a method based on clever multigrid ideas in numerical analysis. It provides an estimator with the same low bias and almost the same variance as an estimator based on a very fine grid (small h), with a computing effort comparable to that required with a coarse discretization (large h). The idea is to select a decreasing sequence of discretization time steps h , and write the estimator for the finest discretization as equal to the estimator for the coarsest one plus a sum of corrections, where each correction is the difference between the estimators at two successive time steps. We first run the simulation at the coarse discretization for a large sample size, then we estimate each correction term independently of the others by simulating the difference with carefully synchronized common random numbers, using a sample size that decreases when h decreases. Much smaller sample sizes can be used for the corrections at the finer levels because these corrections have much smaller variances.

More specifically, suppose the step sizes are $h_\ell = m^{-\ell}T$, $\ell = 0, \dots, L$, for some integers $m \geq 2$ and $L \geq 0$. Let P_ℓ denote the payoff (a random variable) when the process \mathbf{X} is replaced by its time-discretized version with step size $h = h_\ell$. We can write

$$\mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{\ell=1}^L \mathbb{E}[P_\ell - P_{\ell-1}].$$

The multilevel method estimates $\mathbb{E}[P_0]$ by the average Y_0 of N_0 realizations of P_0 , and estimates $\mathbb{E}[P_\ell - P_{\ell-1}]$ by the average Y_ℓ of N_ℓ realizations of $P_\ell - P_{\ell-1}$, for each ℓ . The overall estimator of $\mathbb{E}[P]$ is

$$Y = \sum_{\ell=0}^L Y_\ell.$$

One important feature is that each realization of $P_\ell - P_{\ell-1}$ is simulated with common random numbers across the two terms. For this, we first simulate the process with time step h_ℓ to obtain P_ℓ . This normally requires dm^ℓ random numbers to simulate the increments of the d -dimensional process over the m^ℓ time steps. Then, to simulate the process with time step $h_{\ell-1}$ and obtain $P_{\ell-1}$, we set the increment of the Brownian process $\mathbf{B}(t)$ over the time interval $[(j-1)h_{\ell-1}, jh_{\ell-1})$ to exactly the same value that it took in the simulation with time step h_ℓ . That is, we add the increments of $\mathbf{B}(t)$ over the m time steps $[(j-1)m+k-1)h_\ell, ((j-1)m+k)h_\ell)$, $k = 1, \dots, m$, to obtain the increment of $\mathbf{B}(t)$ over the interval $[(j-1)h_{\ell-1}, jh_{\ell-1}) = [(j-1)mh_\ell, jmh_\ell)$ for the coarser discretization with $h_{\ell-1}$. Let $V_\ell = \text{Var}[P_\ell - P_{\ell-1}]$. Then, $\text{Var}[Y_\ell] = V_\ell/N_\ell$. The simulations are also independent across the different values of ℓ , and therefore

$$\text{Var}[Y] = \sum_{\ell=0}^L \text{Var}[Y_\ell] = \sum_{\ell=0}^L V_\ell/N_\ell.$$

The simulation time at level ℓ , on the other hand, is roughly proportional to N_ℓ times the number of time steps, which is proportional to $1/h_\ell$. The total simulation time $C(Y)$ is then approximately equal to $c_3 \sum_{\ell=0}^L N_\ell/h_\ell$ for some constant c_3 .

Under mild conditions, which hold for example if we use the Euler discretization and the payoff function is Lipschitz continuous, then $V_\ell = \mathcal{O}(h_\ell)$. We shall assume that $V_\ell = c_2 h_\ell$ for some constant c_2 . Then, the work-normalized variance (the inverse of the efficiency) is

$$\text{WNVar}[Y] = \text{Var}[Y]C(Y) = \left(c_2 \sum_{\ell=0}^L h_\ell/N_\ell \right) \left(c_3 \sum_{\ell=0}^L N_\ell/h_\ell \right). \quad (2)$$

Similar to what we saw for stratification with optimal allocation, with an optimal choice of the N_ℓ 's (that minimizes $\text{WNVar}[Y]$), the derivative of the expression (2) with respect to N_ℓ should be the same for all ℓ (otherwise we could decrease (2) by increasing the N_ℓ for which the derivative is smallest and decreasing the N_ℓ for which the derivative is largest. This can be achieved by taking $N_\ell = K_0 h_\ell$ for a constant K_0 (see below). The choice of K_0 would depend either on our total computing budget, or on the variance that we want to achieve. Here we have neglected the fact that N_ℓ must be an integer. In practice, we can round the value of $K_0 h_\ell$.

(a) Prove that if $N_\ell = K_0 h_\ell$ for some constant K_0 , then the derivative of (2) with respect to N_ℓ is the same for all ℓ .

One particular case of (1) is the stochastic volatility model of Heston, defined by the following two-dimensional SDE:

$$dS(t) = rS(t)dt + V(t)^{1/2}S(t)dB_1(t), \quad (3)$$

$$dV(t) = \lambda(\sigma^2 - V(t))dt + \xi V(t)^{1/2}dB_2(t), \quad (4)$$

for $t \geq 0$, where r , σ , λ , and ξ are positive constants, and $\mathbf{B} = (B_1, B_2)$ is a pair of standard Brownians motion with correlation ρ between them. Note that the volatility process V can never become negative. This process V is in fact a CIR process and its increment over a time step h could be simulated exactly by simulating a non-central chi-square random variable (see section 2.15.3 of the class notes), but we will not use this property here. To simulate this process approximately by Euler's method, the bias is reduced if we make the change of variable $W(t) = e^{\lambda t}(V(t) - \sigma^2)$ and apply Euler's method to the process $\mathbf{X} = (S, W)$ instead of (S, V) . This gives

$$\begin{aligned} dW(t) &= e^{\lambda t}dV(t) + e^{\lambda t}\lambda(V(t) - \sigma^2)dt \\ &= e^{\lambda t}dV(t) + \lambda W(t)dt \\ &= e^{\lambda t}[\lambda(\sigma^2 - V(t))dt + \xi V(t)^{1/2}dB_2(t)] + \lambda W(t)dt \\ &= e^{\lambda t}\xi V(t)^{1/2}dB_2(t). \end{aligned}$$

The Euler scheme with step size $dt = h$ becomes

$$\widetilde{W}((j+1)h) = \widetilde{W}(jh) + e^{\lambda jh}\xi\widetilde{V}(jh)^{1/2}\sqrt{h}Z_{j,2},$$

and then, using the identity $V(t) = \sigma^2 + e^{-\lambda t}W(t)$,

$$\begin{aligned} \widetilde{V}((j+1)h) &= \max\left[0, \sigma^2 + e^{-\lambda(j+1)h}\widetilde{W}((j+1)h)\right] \\ &= \max\left[0, \sigma^2 + e^{-\lambda(j+1)h}\left(\widetilde{W}(jh) + e^{\lambda jh}\xi\widetilde{V}(jh)^{1/2}\sqrt{h}Z_{j,2}\right)\right] \\ &= \max\left[0, \sigma^2 + e^{-\lambda h}\left(e^{-\lambda jh}\widetilde{W}(jh) + \xi\widetilde{V}(jh)^{1/2}\sqrt{h}Z_{j,2}\right)\right] \\ &= \max\left[0, \sigma^2 + e^{-\lambda h}\left(\widetilde{V}(jh) - \sigma^2 + \xi\widetilde{V}(jh)^{1/2}\sqrt{h}Z_{j,2}\right)\right], \end{aligned} \quad (5)$$

$$\widetilde{S}((j+1)h) = \widetilde{S}(jh) + rh\widetilde{S}(jh) + [\widetilde{V}(jh)]^{1/2}\widetilde{S}(jh)\sqrt{h}Z_{j,1}, \quad (6)$$

where the vectors $\mathbf{Z}_j = (Z_{j,1}, Z_{j,2})$ are normal with mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma}$ whose elements are 1 on the diagonal and ρ elsewhere. The “ $\max[0, \dots]$ ” is to make sure that the approximation of V never becomes negative.

(b) Write a complete and detailed algorithm for the multilevel Monte Carlo method for this model. Note that for $L = 0$, your algorithm becomes the ordinary Euler method with step size h_0 . In your algorithm, you can increase the value of ℓ until

$$\max(m^{-1}|Y_{\ell-1}|, |Y_\ell|) < (m-1)\varepsilon,$$

where ε represents a target bound on the absolute bias (see Section 4.2 in Giles's paper).

(c) Write a program that implements this algorithm. In your implementation, the payoff function should be defined separately from the code that simulates the process, and it should be easy to replace the payoff function by a different one without changing the rest of the program. For example, the payoff function can be in a different class (or a Java interface), with one subclass (or implementation) for each choice of payoff.

You will consider and implement two different payoff functions, namely the (discounted) payoff function for (i) an European call option,

$$P = e^{-r} \max(0, S(T) - K),$$

where $r > 0$ is the interest rate and K is the strike price (a constant), and (ii) an Asian call option,

$$P = e^{-r} \max(0, \bar{S} - K),$$

where $\bar{S} = \int_0^T S(t)dt$ is the continuous-time average of the process S over the time interval $[0, T]$. For the latter, you cannot compute \bar{S} exactly, because this would require an infinite number of observation times. At discretization level ℓ , with time step h_ℓ , you will replace \bar{S} by the discretized average

$$\bar{S}_\ell = \frac{1}{m^\ell} \left(\frac{\tilde{S}(0)}{2} + \frac{\tilde{S}(T)}{2} + \sum_{j=1}^{m^\ell-1} \tilde{S}(jh) \right)$$

and the payoff P by

$$P_\ell = e^{-r} \max(0, \bar{S}_\ell - K).$$

(d) Implement these two payoff functions in your program.

(e) Perform experiments with multilevel Monte Carlo methods for these two payoff functions, with the following parameter values: $T = 1$, $K = 100$, $S(0) = 100$, $V(0) = 0.04$, $r = 0.05$, $\sigma = 0.2$, $\lambda = 5$, $\xi = 0.25$, and $\rho = -0.5$. Take $\varepsilon = 10^{-4}$ and $K_0 = 10^7$. Try $m = 2$ and $m = 7$. In each case, plot your estimates of $\ln V_\ell$ and of $\mathbb{E}[P]$ as functions of ℓ , and compute the work-normalized variance defined in (2). Compare the work-normalized variances for $m = 2$ and $m = 7$. According to the development above, we should have $V_\ell \approx c_2 m^{-\ell} T$. Then, what should be the behavior of $\ln V_\ell$ as a function of ℓ ? Is this what you observe empirically? What is your estimate of c_2 for each case?

(f) For the same model and parameters, try ordinary Monte Carlo with a single level, that is $L = 0$, with h_0 equal to the smallest h used in your experiments in (e) and $K_0 = 10^5$. Compare the work-normalized variances.

(g) Repeat the same set of experiments as in (e) with $m = 2$, but this time with RQMC using a Sobol' point set of cardinality N_ℓ with a random digital shift to simulate the N_ℓ realizations of $P_\ell - P_{\ell-1}$ at level ℓ . For this, to make sure that each N_ℓ is a power of 2, take $K_0 = 2^{22}$. Compare the work-normalized variances to those obtained with multilevel Monte Carlo in (e).