Monte Carlo Methods Appl. **18** (2012), 161 – 179
DOI 10.1515/mcma-2012-0005

Parallel random number generators in Monte Carlo derivative pricing: An application-based test

Michael Mascagni and Lin-Yee Hin

Abstract. Parallel pseudorandom number generators (PPRNG) that satisfy classical statistical tests may still demonstrate intra-stream and inter-stream correlations in real life applications. In order to investigate the suitability of a PPRNG for use in Monte Carlo pricing of financial derivatives, an application-based test is proposed to evaluate the bias and the standard error of the mean (SE) associated with the PPRNG as a gauge of intra-stream and inter-stream correlations respectively. This test involves estimating the price of a vanilla European call option via Monte Carlo simulation, where the asset price at maturity is estimated by propagating the Black–Scholes stochastic differential equation via the Euler–Maruyama discretization scheme. The mean and SE profiles of the numerical results based on three PPRNG libraries (RNGSTREAM, TRNG and SPRNG) that implement parallel random numbers via sequence splitting strategies (RNGSTREAM and TRNG) and parameterization strategy (SPRNG) are compared. In terms of the bias and SE profiles, the best performing PPRNG constructed using the sequence splitting strategy is comparable to that constructed using parameterization, both use multiple recursive generators in their kernel.

Keywords. Parallel random number generators, testing random numbers, financial applications.

2010 Mathematics Subject Classification. 65C05, 65C10, 65C30, 91G20, 91G60.

1 Introduction

The objective of Monte Carlo (MC) derivative pricing is to produce an accurate and precise stochastic estimate of the present value and the sensitivities of a financial instrument. The former estimate is intended for pricing and the latter for hedging and risk management. Inaccurate estimates of these quantities creates risk and may result in financial losses, at times staggering ones. The increased sophistication of stochastic dynamics used to describe the evolution of asset prices and the advent of complex path-dependent payoff functions render the availability of closed-form solutions a rarity, and MC simulation a powerful and virtually un-

avoidable tool. However, the slow convergence rate of MC simulation limits the speed at which pricing of derivatives can be done. While variance reduction techniques and the use of quasi-Monte Carlo methods have been developed to address this issue by improving the convergence rate for a given number of simulations [4, 10], parallelization of MC simulation is a natural direction to achieve higher speed by sharing the required number of simulations among multiple processors.

In MC simulation, the quality of the pseudorandom number generator (PRNG) is pivotal to its success. The random variates generated should be independent and identically distributed (IID), implying an absence of correlation among the random variates. This requirement translates into an absence of intra-stream correlation for a single processor MC, and absence of both intra-stream and inter-stream correlation for parallel MC.

For sequential MC, as long as there are no short-range correlations among random variates, the PRNG will work reasonably well because the number of random variates consumed in the stochastic propagation of an asset price path in each simulation is generally not large enough to uncover the effect of long-range correlation among the random variates for that path. That said, this may not be true for Brownian Bridge computations. However, the story is different in parallel MC. A large number of processors can be committed to simulation concurrently and are thus capable of consuming a much larger number of random variates compared to a single processor simulation. The fact that there can be correlation between different parts of the stream of random variates [5] implies that long-range correlation may present itself as either inter-stream or intra-stream correlations in parallel MC, the former compromising the standard error of mean (SE) for the estimate while the latter causing bias in the estimate for the same period of computing time. The impact of inter-stream and intra-stream correlations on MC derivative pricing is considered in greater detail in Section 2.

Two common strategies have been proposed to provide independent streams of random variates from a PRNG for parallel MC simulation. The *parameterization* strategy provides independent streams of random variates by assigning, in different processors, different values to parameters in the generation algorithm of the parallel PRNG (PPRNG) (see [15]). The *sequence splitting* strategies include block-splitting and leapfrog. In block-splitting, widely separated seeds are chosen deterministically for the processors to divide a sequential stream of random variates into sufficiently large blocks so that each processor consumes one of the substreams and the blocks of random variates do not overlap. However, if the user happen to consume more random variates than anticipated, and the blocks do overlap, this may lead to correlation among segments of random variates in different streams (inter-stream correlation). In leapfrog, if there are n processors, then each processor is assigned a stream of random variates that are n positions apart in the

original sequence. Depending on the number of processors involved and the number of random variates making up the segments of correlated random variates, the segments of correlated random variates far apart in the original stream may now find themselves close to each other as some of the interposing random variates have been distributed to other streams, leading to short-range intra-stream correlation. In the worst case scenario, most, if not all, of the parallel streams may exhibit short-range intra-stream correlation if the segments of random variates exhibiting long range correlation consist of sizable blocks of random variates [16]. In addition, if correlated blocks of random variates are distributed to different streams, this can result in intra-stream as well (Figure 1).

Short of conclusive theoretical and empirical evidence favoring one strategy over another, application-based tests are necessary to uncover strengths and shortfalls of each PRNGs when parallelized using these different strategies. In the context of parallel MC for derivative pricing, accurate point estimates and small standard error (SE) of the estimates are of practical importance. In MC derivative pricing, both accuracy and precision are of paramount importance. The former can be assessed by bias while the latter by variance. The choice from among a set of competing PRNGs can be made based on their bias-variance trade-off profile.

Using a parallel MC simulation framework that is designed to estimate the present value of a derivative instrument using different PRNGs, the SEs of the present value estimates of a derivative instrument can be compared based on simulation results estimated with respect to a chosen set of model parameters. When the derivative instrument used in the test is analytically tractable, the bias associated with different PRNGs can be evaluated based on the present value of the derivative instrument evaluated analytically. Such tests are described in detail in Section 3

The outline of the paper is as follows. In Section 2, we discuss the impact of inter-stream and intra-stream correlations on the bias and variance of the estimated present value for a derivative product priced using the parallel MC framework. In Section 3, an application-based test is proposed to investigate the bias and standard error of the mean (SE) associated with the PRNG that is being tested by comparing the result of estimation against the analytic solution. In Section 4, we subject three PRNG libraries to the application-based test proposed in Section 3 and report the test results. The three PRNG libraries being tested are the

(i) Scalable Parallel Random Number Generators (SPRNG) library that implement the parameterization strategy [17],

(ii) TINA's Random Number Generator library (TRNG) that implements both block-splitting and leapfrog sequence splitting strategy [18],

(iii) RNGSTREAM library that implements block-splitting strategy [14].

There have been no publicly available reports on a comparison of their performance in terms of bias and SE in derivative pricing under the parallel MC framework as reported in this paper.

2 Impact of correlation among random variate in derivative pricing

The use of MC is extensive in a large number of application areas in mathematics, physics, chemistry, biology, and the various engineering disciplines. This is due to its intrinsic nature and often the similarity of the MC approach to an underlying mechanism. However, one of the areas where MC has grown tremendously in its use and its effectiveness is in finance. In the past, methods of testing the quality of PRNGs that were based on applications came from the sciences and engineering ([6, 8, 13]); however, it seems appropriate to do the same with calculations that are somehow characteristic of important financial computations. Thus, we have chosen to fashion an application-based test for PRNGs from the calculation of the price of a financial derivative. This is a very basic financial computation, yet if it is done from the point-of-view of solving a stochastic differential equation, it is a challenging computation and provides a unique application-based test of randomness.

Bias and SE are metrics that can be used to measure the accuracy and precision of the point estimates obtained via MC simulation. They are related to the mean squared error (MSE) by the expression $\text{MSE} = \text{bias}^2 + \text{SE}^2$. In Sections 2.1 and 2.2, we demonstrate that the presence of inter-stream correlation increases the variance while the presence of intra-stream correlation increases the bias.

2.1 Inter-stream correlation

Let there be n processors indexed by $i = 1, \ldots, n$ each carrying out m_i MC simulations, so that the total number of MC simulations carried out across all n processors is $N = \sum_{i=1}^{n} m_i$. Let $\xi_{i,j}$ be the MC estimate for sample j on processor i where $j = 1, \ldots, m_i$. In the context of derivative pricing, $\xi_{i,j}$ typically denotes the present value of a financial instrument or one of its sensitivities evaluated using pathwise differentiation [7]. In addition, let $x_i = \sum_{j=1}^{m_i} \xi_{i,j}$ and $\hat{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ where $\hat{x}$ is the global point estimate across all n processors. In general,

$$\text{Var}[\hat{x}] = \text{Var}\left[\frac{1}{n}\sum_{i=1}^{n} x_i\right] = \frac{1}{n^2}\left\{\sum_{i=1}^{n} \text{Var}[x_i] + \sum_{i \neq j} \text{Cov}(x_i, x_j)\right\} \tag{2.1}$$

where $\text{Var}[x_i]$ is the variance of x_i and $\text{Cov}(x_i, x_j)$ is the covariance between x_i and x_j when the x_i are identically distributed. When there is no inter-stream cor-

relation, $\sum_{i\neq j}\mathrm{Cov}(x_i,x_j)=0$ and we have $\mathrm{SE}[\hat{x}]=\mathrm{SE}[x_i]/\sqrt{n}$, where $\mathrm{SE}[\hat{x}]$ is the standard error of the mean and $\mathrm{SE}[x_i]$ is the standard error of the sample because the x_i are IID. On the contrary, when inter-stream correlation is non-zero, i.e., $\rho_{j,k}\neq 0$, then $\sum_{i\neq j}\mathrm{Cov}(x_i,x_j)\neq 0$, leading to $\mathrm{Var}[\hat{x}]\neq\frac{1}{n}\mathrm{Var}[x_i]$. Positive inter-stream correlation leads to

$$\mathrm{Var}[\hat{x}]>\frac{1}{n}\mathrm{Var}[x_i],$$

resulting in a larger standard error compared to the case of no inter-stream correlation. The larger the number of processors, the greater the deviation of $\mathrm{Var}[x_i]$ away from $\mathrm{Var}[x_i]/n$ because the number of covariance terms increases at the rate of $O(\binom{n}{2})=O(n^2)$. Therefore, a larger number of processors tend to reveal inter-stream correlation more readily.

2.2 Intra-stream correlation

Let $C(t,X(t))$ be the price of a derivative instrument at time t, the value of which depends on the time, t, and the price of an underlying asset, $X(t)$, observed at time t. Let the price evolution of the underlying asset follow a stochastic differential equation (SDE) defined as $dX(t)=a(X(t))dt+b(X(t))dW$. The price evolution of this derivative instrument from the current time t_1 to a future time t_K where $t_1<t_K$ can be estimated using Itô's lemma [11]

$$\begin{aligned}C(t_K,X(t_K))=C(t_1,X(t_1))&+\left\{\frac{\partial C(t,X(t))}{\partial t}dt\right\}_{t=t_1,X(t)=X(t_1)}\\&+\left\{\frac{\partial C(t,X(t))}{\partial X(t)}dX(t)\right\}_{t=t_1,X(t)=X(t_1)}\\&+\left\{\frac{1}{2}\frac{\partial^2 C(t,X(t))}{\partial X(t)^2}\{b(X(t))\}^2dW(\tau)dW(\tau)\right\}_{t=t_1,X(t)=X(t_1)}\end{aligned}$$

where $\tau=t_K-t_1$ and $dW(\tau)$ is the Wiener process spanning the time interval $[t_1,t_K]$. By definition, $E[dW(\tau)]=0$ and $E[dW(\tau)dW(\tau)]=\tau$ so that the quadratic term in Itô's lemma is approximated by

$$\begin{aligned}E[dW(\tau)dW(\tau)]\times&\left\{\frac{1}{2}\frac{\partial^2 C(t)}{\partial X^2}\{b(X(t))\}^2\right\}_{t=t_1,X(t)=X(t_1)}\\&=\tau\left\{\frac{1}{2}\frac{\partial^2 C(t)}{\partial X^2}\{b(X(t))\}^2\right\}_{t=t_1,X(t)=X(t_1)}.\end{aligned}\tag{2.2}$$

Let $dW(t_2-t_1),dW(t_2-t_1),\ldots,dW(t_K-t_{K-1})$ be $K-1$ successive Wiener processes spanning $[t_1,t_K]$ representing the decomposition of $dW(\tau)$. In solving

the partial differential equation for $C(t_K)$ by MC simulation using stochastic numeric strategies such as the Euler or the Milstein scheme [12], the discretization expression $dW(t_k - t_{k-1}) \approx \sqrt{t_k - t_{k-1}}Z(k)$, $k = 2, \ldots, K$, is used to approximate the Wiener process spanning the interval $[t_k, t_{k-1}]$. Since

$$dW(\tau) = \sum_{k=2}^{K} dW(t_k - t_{k-1}),$$

we have $dW(\tau) \approx \sum_{k=2}^{K} \sqrt{t_k - t_{k-1}}Z(k)$ where $Z(2), \ldots, Z(K)$ is a stream of $K-1$ standard Gaussian random variates with mean 0 and variance 1 so that

$$\begin{aligned} E[dW(\tau)dW(\tau)] &\approx E\left[\left(\sum_{k=2}^{K} \sqrt{t_k - t_{k-1}}Z(k)\right)\left(\sum_{s=2}^{K} \sqrt{t_s - t_{s-1}}Z(s)\right)\right] \\ &= \tau + \left\{\sum_{\substack{k,s=2 \\ k\neq s}}^{K} \sqrt{(t_k - t_{k-1})(t_s - t_{s-1})}\rho_{Z(k),Z(s)}\right\}, \end{aligned} \tag{2.3}$$

where $\sum_{k=2}^{K}(t_k - t_{k-1}) = \tau$ and $\rho_{Z(k),Z(s)}$ is the correlation between $Z(k)$ and $Z(s)$ since $\mathrm{Cov}(Z(k), Z(s)) = \rho_{Z(k),Z(s)}$ because $Z(k)$ and $Z(s)$ are standard Gaussian random variates. If the stream of standard Gaussian random variates is IID, $\mathrm{Cov}(Z(k), Z(s)) = 0$, $k, s = 2, \ldots, K$, $k \neq s$, the definition of Wiener process is observed in the discretization scheme and the estimation of the quadratic term in the Itô lemma is consistent with (2.2). On the contrary, if intra-stream correlation is present among the random variates, $\mathrm{Cov}(Z(k), Z(s)) \neq 0$, the definition of Wiener process is violated, leading to biased estimation of the quadratic term in the Itô lemma demonstrated by substituting (2.3) into the left hand side of (2.2) as

$$\begin{aligned} E\left[dW(\tau)dW(\tau)\right] \times &\left\{\frac{1}{2}\frac{\partial^2 C(t)}{\partial X^2}\{b(X(t))\}^2\right\}_{t=t_1, X(t)=X(t_1)} \\ \approx &\left\{\frac{1}{2}\frac{\partial^2 C(t)}{\partial X^2}\{b(X(t))\}^2\tau\right\}_{t=t_1, X(t)=X(t_1)} \\ &+ \left\{\frac{1}{2}\frac{\partial^2 C(t)}{\partial X^2}\{b(X(t))\}^2\right\}_{t=t_1, X(t)=X(t_1)} \\ &\times \left\{\sum_{\substack{k,s=2 \\ k\neq s}}^{K} \sqrt{(t_k - t_{k-1})(t_s - t_{s-1})}\rho(Z(k), Z(s))\right\}. \end{aligned} \tag{2.4}$$

Here the second term on the right hand side is the magnitude of bias due to intra-stream correlation. The magnitude of the bias term depends on the overall intra-stream correlation and is stream specific.

3 Test description

In the proposed test, a vanilla European call option is used as the test financial instrument. This is an option that is exercised only at maturity, and the price depends solely on the price of the underlying asset at maturity, $X(t_K)$. Its present value is readily calculated using a closed-form formula, facilitating assessment of the option price estimated via parallel MC simulation powered by different PPRNGs. The present value at time t_1 of a European vanilla call option maturing at time t_K with strike price Y and underlying asset price at maturity $X(t_K)$ is

$$V_0 = e^{-r\tau} E\big[(X(t_K) - Y)^+\big], \tag{3.1}$$

where r is the risk-free interest rate, suppressing the notational dependence of V_0 on t and $X(t)$. If we apply Itô's lemma to (3.1), and since the option's price depends on the underlying asset price at maturity only, the potential bias incurred by intra-stream correlation is reflected in the bias term defined in (2.4) with respect to V_0 is

$$\begin{aligned}\text{Bias}(V_0) = &\left\{\frac{1}{2}\frac{\partial^2 V_0}{\partial X^2}\{b(X(t))\}^2\right\}_{t=t_1, X(t)=X(t_1)} \\ &\times \left\{\sum_{\substack{k,s=2 \\ k\neq s}}^{K} \sqrt{(t_k - t_{k-1})(t_s - t_{s-1})}\rho(Z(k), Z(s))\right\}.\end{aligned} \tag{3.2}$$

Positive intra-stream correlation leads to upward bias while negative intra-stream correlation leads to downward bias.

Under the risk-neutral assumption, the Black–Scholes model [3] is

$$dS(t)/S(t) = r dt + \sigma dW(t) \tag{3.3}$$

where $S(t)$ is the asset price at time t, r the risk-free interest rate, σ the implied volatility, and $dW(t)$ the stochastic diffusion component following a Wiener process. Within the Black–Scholes framework, (3.1) has a closed-form solution

$$V_0 = e^{-r\tau} E[(S(t_K) - Y)^+] = S_0 \mathcal{N}(d_+) - e^{-r\tau} Y \mathcal{N}(d_-). \tag{3.4}$$

Here V_0 is the present value of the vanilla call option, t_1 is the present time, t_K is the time at maturity, $\tau = t_K - t_1$ is the time interval between now and the option

maturity date, $S(t_K)$ is the asset price at maturity, $S_0 = S(t_1)$ is the spot asset price, Y is the strike price,

$$\mathcal{N}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-y^2/2} dy$$

is the cumulative distribution function of the standard normal distribution,

$$d_+ = \left\{ \log(S_0/Y) + \left(r - \frac{1}{2}\sigma^2 \right) \tau \right\} / (\sigma \sqrt{\tau}), \quad \text{and} \quad d_+ = d_- + \sigma \sqrt{\tau},$$

assuming the dividend rate is zero, without loss of generality.

Monte Carlo estimation of V_0 is performed under three different market scenarios:

(i) at the money (ATM, $S_0 = Y$): $S_0 = Y = 1$,

(ii) out of the money (OTM, $S_0 < Y$): $S_0 = 1$, $Y = 1.05$,

(iii) in the money (ITM, $S_0 > Y$): $S_0 = 1$, $Y = 0.95$,

where $\sigma = 0.19$, $T = 1$, $r = 0.01$. The SDE for the asset path is numerically integrated via the log-Euler discretization scheme [9]

$$\log S(t_{k_1}) = \log S(t_k) + (r - \sigma^2/2)h + \sigma \sqrt{h} Z(t_{k+1})$$

where $S(t_k)$ is the asset price at time t_k, h is the equally spaced discretization time-step, and $Z(t_{k+1})$ is the standard Gaussian random variable spanning $[t_k, t_{k+1}]$. The use of the log-Euler scheme is to ensure positivity of the simulated asset price. In this test, a discretization time-step of size $h = 10^{-4}$ is used and 10^6 simulation runs were performed for each PRNG setting to estimate V_0 and the MSE of V_0. In this setting, simulating one stochastic path for the underlying asset consumes 10^4 random variates and the entire simulation consumes 10^{10} random variates. The choice of testing whether the PRNG passes the proposed application-based test at 10^{10} random number is inspired by the observation that block-splitting of the 48-bit Cray linear congruential generator fails the blocking test with around 10^{10} random variates without inter-stream sequence overlap [16], indicating that 10^{10} may be a reasonable size of random variates to uncover defect due to long-range correlations.

Let $\text{Bias}(V_0) = \hat{V}_{0,\text{PRNG}} - V_{0,\text{T}}$ where $\text{Bias}(V_0)$ is the estimated bias of V_0 for the MC simulation framework powered by a PRNG, $\hat{V}_{0,\text{PRNG}}$ is the value of V_0 estimated using MC simulation powered by the PRNG considered, and $V_{0,\text{T}}$ is the value of V_0 evaluated using (3.4). Comparison against the closed form is intended to measure the accuracy of each PRNG being tested by quantifying the magnitude of the bias that can potentially be incurred by the presence of intra-stream correlation among the random variates.

Let R-SE($\hat{V}_0$) = SE($\hat{V}_{0,\text{PRNG}}$)/SE($\hat{V}_{0,\text{PRNG,SERIAL}}$) be the relative SE of the V_0 estimate, where SE($\hat{V}_{0,\text{PRNG}}$) is the SE of V_0 estimate obtained using the parallel MC framework, SE($\hat{V}_{0,\text{PRNG,SERIAL}}$) using the serial MC framework, and both parallel and serial MC simulations are powered by the same PRNG. Since the parallel and serial MC frameworks share identical discretization time-steps, they share similar discretization error. The ratio defined by R-SE($\hat{V}_0$) is intended to measure the precision of each PRNG being tested by quantifying the relative magnitude of the MSE that can potentially be incurred by the presence of inter-stream correlation among the random variates.

4 PRNGs tested

The SPRNG, TRNG and RNGSTREAM libraries are subjected to the application-based test detailed in Section 3.

The PRNGs from version 2.0 of the SPRNG library being tested are tabulated in Table 1. For this library, parallelization of PRNG streams is implemented by appropriate parametrization of the coefficients in the generators and the magnitude of moduli (prime or power of 2) that ensures long period and passing of statistical tests [16].

The PRNGs from version 4.10 the TRNG library being tested are tabulated in Table 2. Each of the PRNGs in TRNG listed are tested under both the block-splitting and leapfrog methods in the parallel MC framework.

The RNGSTREAM PRNG deploys the block-splitting strategy to provide parallel streams of random variates based on a direct combination of two shift-registered generators both of order 2 with prime moduli. This generator is defined by

$$z_n = \left(x_{1,n} - x_{2,n}\right) \pmod{4294967087},$$

where

$$x_{1,n} = 1403580 \times x_{1,n-2} - 810728 \times x_{1,n-3} \pmod{2^{32} - 209},$$
$$x_{2,n} = 527612 \times x_{2,n-2} - 810728 \times x_{2,n-3} \pmod{2^{32} - 22853}.$$

The period is approximately 3.1×10^{57}.

5 Test results

A PRNG suitable for parallelization should be free of intra-stream correlation in both sequential and parallel implementations, the latter include the block-splitting implementation, leapfrog implementation, and parameterization implementation.

The PRNG of choice should demonstrate comparable magnitudes of price estimation bias for all implementations.

For a PRNG to be considered safe for use under a particular implementation in pricing derivative instruments, it should demonstrate the same bias profile across all moneyness scenarios consistently.[1] For example, a PRNG demonstrating the smallest bias for the block-splitting implementation under all moneyness scenarios can be considered safe for use in block-splitting. On the contrary, a PRNG with an inconsistent bias profile cannot be considered safe for use in a particular implementation because it is unclear the precise point on the moneyness axis at which the bias profile changes from one to another. It would be impractical to evaluate bias profiles for all implementations on a fine grid along the moneyness axis due to the computational burden. In addition, the point along the moneyness axis that marks where change in the bias profile takes place may differ for different derivative instruments, different discretization sizes and schemes, and different scenario parameters including moneyness. Therefore, it would be safer to use a PRNG that has a consistent bias profile across all moneyness scenarios.

In addition, a PRNG suitable for parallelization should be free of inter-stream correlation in a parallel implementation, demonstrating comparable magnitudes of price estimate SE for sequential and parallel implementations. This is reflected by R-SE($\hat{V}_0$), the ratio of the standard error of the mean for the parallel implementation to the standard error of the mean of the sequential implementation, that is near unity.

The sequential and parallel MC simulations are carried out on a Linux Ubuntu platform with an Intel Core 2 processor that supports two threads. Thus, one can have two streams of random variates generated from each PRNG tested in each parallel MC simulation. This mimics the situation when a MC simulation pricing algorithm is executed on an average desktop or notebook computer currently available, since most of these machine are equipped with multicore processors. Therefore, the test results reported in this section are specific to a dual-core setting. Before these results can be generalized to a larger number of processors, the tests detailed in Section 3 should be performed for the PRNGs within the parallel framework in question because, for sequence splitting strategies, the partition of segments of random variates that exhibit long range correlation to different streams and the resultant intra-stream proximity varies as the number of streams differ. In addition, a larger number of parallelized streams that exhibit inter-stream correlations tend to cause an increase in SE to a greater extent because there are more covariance terms in the variance expression, as per (2.1).

[1] The moneyness of a derivative refers to the degree to which the derivative will pay off as "in the money" (ITM), "at the money" (ATM), and "out of the money" (OTM).

5.1 Tests on TRNG generators

The test results for PRNGs in the TRNG library under ITM, ATM and OTM moneyness scenarios are tabulated in Table 3.

The PRNGs `lcg64`, `mrg4`, `mrg5`, `mrg3s` and `yarn5s` demonstrate a consistent bias profile across all moneyness scenarios with the smallest bias in block-splitting implementation compared to leapfrog implementation and sequential implementation. Therefore, it appears that these PRNGs can be safely used in block-splitting parallel implementation.

However, the PRNGs `lcg64_shift`, `mrg2`, `mrg3`, `yarn4` demonstrate their lowest bias in sequential implementation under the ATM and OTM scenarios. However, under the ITM scenario, these PRNGs demonstrate their lowest bias in either block-splitting implementation or leapfrog implementation. The other PRNGs considered, `mrg5s`, `yarn2`, `yarn3`, `yarn5` and `yarn3s`, demonstrate inconsistent bias profiles as well. The extreme case is `mrg5s` where different implementations demonstrate the smallest bias profile under different moneyness scenarios. Therefore, these PRNGs cannot be safely recommended for use in parallel implementation because of inconsistency in their bias profile.

5.2 Test on RNGSTREAM generator

The test results for PRNGs in the RNGSTREAM library under the ITM, ATM and OTM scenarios are tabulated in Table 4. The bias profile for the RNGSTREAM PRNG is larger in the block splitting parallel MC framework compared to that in the sequential MC framework. This suggests that some degree of intra-stream correlation may exist in the parallel implementation that leads to estimation bias. Nonetheless, RNGSTREAM will serve as a good sequential PRNG as the bias profile is the smallest for the sequential implementation for all three scenarios tested.

5.3 Tests on SPRNG generators

The test results for PRNGs in the SPRNG library under the ITM, ATM and OTM scenarios are tabulated in Table 5. For `sprng::lcg`, the bias profile for sequential and parallel implementations are comparable. The bias profiles for `sprng::cmrg` and `sprng::lfg` is higher in parallel implementation than that in sequential implementation. For `sprng::pmlcg`, the bias profile is consistently smaller for sequential implementation under the ITM, ATM and OTM scenarios. The bias profiles for `sprng::lcg64` and `sprng::cmrg` are consistently smaller for parallel implementation compared to sequential implementation under all three scenarios, making them suitable for parallel implementation of Monte Carlo pricing of financial instruments that are similar in nature to the proposed application based test.

5.4 Bias and SE profile comparison among TRNG, RNGSTREAM and SPRNG

Among the PRNGs tested, `sprng::cmrg` and `trng::mrg`n where $n = 4$ demonstrate the smallest bias profile in parallel implementation, where the bias profile associated with `sprng::cmrg` is smaller than that of `trng::mrg`n where $n = 4$ in the ITM and OTM scenarios, while their bias profiles are comparable in the ATM scenario. In addition, `sprng::pmlcg` has the smallest bias profile across all moneyness scenarios in sequential implementation among all the PRNGs tested, making it a PRNG of choice among the tested PRNGs for sequential implementation. The SE for sequential and parallel MC implementations for all PRNGs in TRNG RNGSTREAM and SPRNG are comparable. One reason for this may be that only two processors are used in this test. Whether the same standard error of mean profiles hold for a larger number of parallel threads will require a larger number of parallel threads.

6 Discussion

The PRNG kernels used in sequence splitting strategies are not entirely the same as that used in the parametrization strategy. In order to achieve rapid calculations to facilitate jumping ahead in sequence in sequence splitting, multiple recursion generators and additive lagged-Fibonacci generators are commonly used due to efficiency consideration, such as their application in the TRNG and RNGSTREAM PRNGs.

Therefore, the comparison between PRNGs implementing sequence splitting strategies and those implementing the parameterization strategy is not merely a comparison between parallelization strategies, but also a comparison of the quality of the generators used to implement these strategies.

It is difficult to draw a general conclusion on whether the PRNGs produced using the sequence splitting strategy or the parameterization strategy are in general associated with a more favorable bias and SE profiles. However, the PRNG with the smallest bias profile among PRNGs produced by sequence splitting parallelization has a bias profile comparable to the PRNG with the smallest bias profile among PRNGs produced by parameterization parallelization.

It is interesting to note that both `trng::mrg`n where $n = 4$ and `sprng::cmrg` involve the use of multiple recursive generator in their kernel. Whether combined PRNGs that involve the use of multiple recursive generators in composition or direct combination will produce new PRNGs with bias and SE profiles that are at least comparable to those of `trng::mrg`n where $n = 4$ and `sprng::cmrg` while satisfying all the necessary statistical tests is an open research question.

PRNG name	Definition	Approximate period
`sprng::lcg`	$x_n = ax_{n-1} + p \pmod{2^{48}}$	2^{48}
`sprng::lcg64`	$x_n = ax_{n-1} + p \pmod{2^{64}}$	2^{64}
`sprng::pmlcg`	$x_n = ax_{n-1} \pmod{2^{61} - 1}$	$2^{64} - 2$
`sprng::cmrg`	$z_n = x_n + y_n \times 2^{32} \pmod{2^{64}}$	2^{219}
`sprng::mlfg`	$x_n = x_{n-j} \times x_{n-k} \pmod{2^{64}}$	$2^{61}(2^k - 1)$
`sprng::lfg`	$z_n = \tilde{x}_n \oplus \tilde{x}_n$	$2^{31}(2^k - 1)$

Table 1. Random number generators from SPRNG. Here, p is a prime number, a is a multiplier, x_n is the sequence generated by the 64-bit LCG, y_n is the sequence generated by $y_n = 107374182y_{n-1} + 104480y_{n-5} \pmod{2147483647}$, and $\oplus$ denotes exclusive-or operator, $\tilde{x}_n$ and $\tilde{y}_n$ are sequences obtained from lagged-Fibonacci generator (LFG) sequences X and Y defined by $X_n = X_{n-j} + X_{n-k} \pmod{2^{32}}$, and $Y_n = Y_{n-j} + Y_{n-k} \pmod{2^{32}}$, where x_n is obtained by setting the least significant bit of X_n to zero and y_n is obtained by right-shifting Y_n by one bit. By default, $k = 1279$.

PRNG name	Definition	Approximate period
`trng::lcg64`	$x_n = ax_{n-1} + p \pmod{2^{64}}$	2^{64}
`trng::lcg64_shift`	$\overline{x}_n = ax_{n-1} + p \pmod{2^{64}}$	2^{64}
`trng::mrg`n	$x_n = \sum_{i=1}^{j} a_i x_{n-i} \pmod{m}$	2^{155} for $j = 5$
`trng::mrg`n`s`	$x_n = \sum_{i=1}^{j} a_i x_{n-i} \pmod{\tilde{m}}$	2^{155} for $j = 5$
`trng::yar`n	$r_n = \begin{cases} g^{\tilde{x}_n} \pmod{m} & \text{if } q_i > 0, \\ 0 & \text{if } q_i = 0, \end{cases}$	2^{155} for $j = 5$
`trng::yar`n`s`	$r_n = \begin{cases} g^{\tilde{x}_n} \pmod{m} & \text{if } q_i > 0, \\ 0 & \text{if } q_i = 0, \end{cases}$	2^{155} for $j = 5$

Table 2. Random number generators from TRNG. Here, $x \gg n$ is bit-shift of x to the right of size n and $x \ll n$ is bit-shift of x to the left of size n, $t_{n,0} = \overline{x}_n$, $t_{n,1} = t_{n,0} \oplus (t_{n,0} \gg 17)$, $t_{n,2} = t_{n,1} \oplus (t_{n,1} \ll 31)$, $t_{n,3} = t_{n,2} \oplus (t_{n,2} \gg 8)$, m is the Mersenne prime, a_i denotes a multiplier, $\tilde{m}$ is a Sophie-Germain prime, $\tilde{x}_n$ is defined as $\tilde{x}_n = \sum_{i=1}^{j} a_i x_{n-i} \pmod{m}$, and for `trng::yar`$n$ and `trng::yar`n`s`, $n \in \{1, 2, 3, 4, 5\}$.

PRNG name	Implementation	Bias(V_0)($\times 10^{-4}$)			R-SE($\hat{V}_0$)		
		ITM	ATM	OTM	ITM	ATM	OTM
lcg64	leapfrog	−0.324	−0.647	−0.598	0.9987	0.9991	0.9991
lcg64	block-split	−0.131	−0.236	−0.188	1.0051	1.0015	1.0015
lcg64	sequential	−0.236	−0.375	−0.316	1.0000	1.0000	1.0000
lcg64_shift	leapfrog	−0.044	0.140	0.178	0.9967	0.9995	0.9994
lcg64_shift	block-split	−0.189	−0.350	−0.265	0.9961	0.9981	0.9981
lcg64_shift	sequential	−0.105	−0.132	−0.082	1.0000	1.0000	1.0000
mrg2	leapfrog	−0.191	−0.379	−0.321	0.9979	0.9995	0.9995
mrg2	block-split	0.093	0.379	0.422	1.0037	1.0025	1.0025
mrg2	sequential	−0.098	−0.113	−0.064	1.0000	1.0000	1.0000
mrg3	leapfrog	−0.140	−0.205	−0.161	1.0015	0.9996	0.9996
mrg3	block-split	0.054	0.304	0.344	1.0045	1.0018	1.0019
mrg3	sequential	−0.071	0.022	0.062	1.0000	1.0000	1.0000
mrg4	leapfrog	−0.083	0.065	0.124	0.9974	1.0002	1.0001
mrg4	block-split	−0.083	−0.020	0.021	0.9991	1.0004	1.0004
mrg4	sequential	−0.108	−0.090	−0.043	1.0000	1.0000	1.0000
mrg5	leapfrog	−0.198	−0.275	−0.218	0.9988	0.9985	0.9985
mrg5	block-split	−0.131	−0.131	−0.082	0.9996	0.9997	0.9997
mrg5	sequential	−0.155	−0.251	−0.196	1.0000	1.0000	1.0000
mrg3s	leapfrog	−0.281	−0.474	−0.410	0.9998	0.9988	0.9988
mrg3s	block-split	−0.182	−0.260	−0.205	1.0012	1.0006	1.0006
mrg3s	sequential	−0.251	−0.414	−0.349	1.0000	1.0000	1.0000
mrg5s	leapfrog	−0.019	0.160	0.211	1.0028	1.0004	1.0004
mrg5s	block-split	−0.094	−0.086	−0.048	0.9991	0.9994	0.9994
mrg5s	sequential	−0.070	0.044	0.097	1.0000	1.0000	1.0000

to be continued

PRNG name	Implementation	Bias(V_0)($\times 10^{-4}$)			R-SE($\hat{V}_0$)		
		ITM	ATM	OTM	ITM	ATM	OTM
yarn2	leapfrog	−0.125	−0.039	0.009	0.9963	0.9996	0.9996
yarn2	block-split	−0.061	−0.020	0.018	1.0006	1.0014	1.0015
yarn2	sequential	−0.208	−0.477	−0.426	1.0000	1.0000	1.0000
yarn3	leapfrog	−0.117	−0.196	−0.125	1.0020	1.0002	1.0002
yarn3	block-split	0.093	0.472	0.532	1.0033	1.0020	1.0020
yarn3	sequential	0.020	0.281	0.321	1.0000	1.0000	1.0000
yarn4	leapfrog	0.015	0.311	0.345	1.0021	1.0000	1.0000
yarn4	block-split	−0.168	−0.220	−0.175	0.9979	0.9986	0.9986
yarn4	sequential	−0.093	−0.042	0.005	1.0000	1.0000	1.0000
yarn5	leapfrog	−0.163	−0.320	−0.272	0.9998	0.9997	0.9998
yarn5	block-split	−0.170	−0.167	−0.114	0.9966	0.9988	0.9988
yarn5	sequential	−0.063	0.140	0.204	1.0000	1.0000	1.0000
yarn3s	leapfrog	0.048	0.250	0.280	1.0024	1.0021	1.0022
yarn3s	block-split	−0.070	−0.008	0.039	1.0007	1.0002	1.0001
yarn3s	sequential	−0.051	0.060	0.102	1.0000	1.0000	1.0000
yarn5s	leapfrog	−0.194	−0.457	−0.415	1.0054	1.0019	1.0019
yarn5s	block-split	−0.148	−0.292	−0.243	1.0064	1.0020	1.0021
yarn5s	sequential	−0.266	−0.459	−0.411	1.0000	1.0000	1.0000

Table 3. TRNG results: Bias(V_0): bias of $\hat{V}_0$ with respect to V_0, R-SE($\hat{V}_0$): ratio between SE($\hat{V}_0$) of parallel schemes and that of sequential MC simulations, leapfrog: results estimated using parallel MC with leapfrog PRNG, block-split: results estimated using parallel MC with block splitting PRNG, sequential: results estimated using sequential MC, ITM: into the money, ATM: at the money, OTM: out of the money.

Implementation	Bias(V_0)($\times 10^{-4}$)			R-SE($\hat{V}_0$)		
	ITM	ATM	OTM	ITM	ATM	OTM
block-split	0.245	0.297	0.056	1.002543	1.002555	1.006525
sequential	−0.149	−0.110	−0.136	1.000000	1.000000	1.000000

Table 4. RNGSTREAM results: Bias(V_0): bias of $\hat{V}_0$ with respect to V_0, R-SE($\hat{V}_0$): ratio between SE($\hat{V}_0$) of parallel schemes and that of sequential MC simulations, block-split: results estimated using parallel MC with block splitting PRNG, sequential: results estimated using sequential MC, ITM: into the money, ATM: at the money, OTM: out of the money.

PRNG name	Implementation	Bias(V_0)($\times 10^{-4}$)			R-SE($\hat{V}_0$)		
		ITM	ATM	OTM	ITM	ATM	OTM
lfg	parameterization	−0.1650	−0.4310	−0.3910	1.0006	0.9993	0.9994
lfg	sequential	−0.1230	−0.2110	−0.1600	1.0000	1.0000	1.0000
lcg	parameterization	−0.1980	−0.4140	−0.3610	0.9994	0.9997	0.9997
lcg	sequential	−0.2150	−0.4490	−0.3830	1.0000	1.0000	1.0000
lcg64	parameterization	−0.0110	0.1080	0.1480	1.0002	1.0010	1.0010
lcg64	sequential	−0.1530	−0.3340	−0.2770	1.0000	1.0000	1.0000
cmrg	parameterization	−0.0440	−0.0330	0.0100	1.0003	0.9991	0.9991
cmrg	sequential	0.0650	0.4100	0.4560	1.0000	1.0000	1.0000
mlfg	parameterization	−0.0370	0.1640	0.1990	0.9997	0.9996	0.9996
mlfg	sequential	0.0110	0.3260	0.3620	1.0000	1.0000	1.0000
pmlcg	parameterization	−0.0810	−0.1190	−0.0850	0.9972	0.9991	0.9991
pmlcg	sequential	−0.0270	−0.0020	0.0310	1.0000	1.0000	1.0000

Table 5. SPRNG results: Bias(V_0): bias of $\hat{V}_0$ with respect to V_0, R-SE($\hat{V}_0$): ratio between SE($\hat{V}_0$) of parallel schemes and that of sequential MC simulations. parameterization: results estimated using parallel MC, sequential: results estimated using sequential MC, ITM: into the money, ATM: at the money, OTM: out of the money.

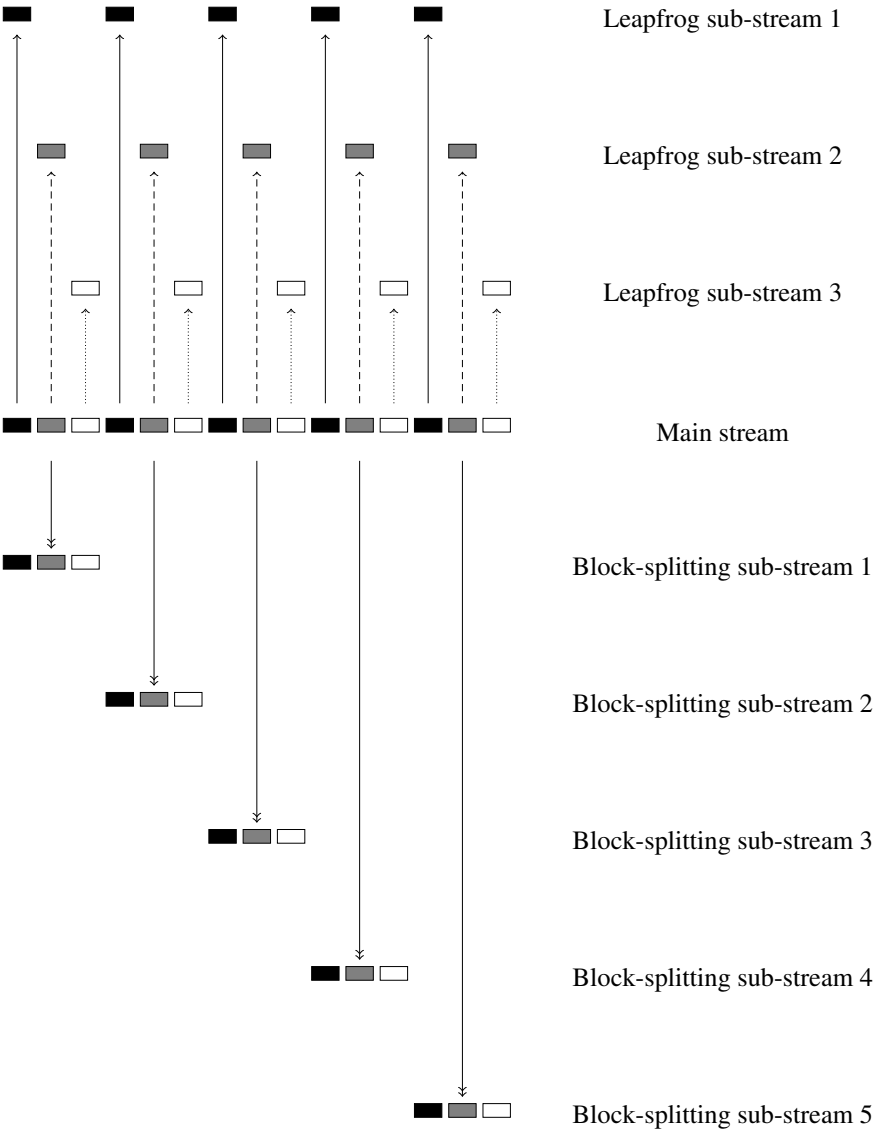

Figure 1. Parallelization by leapfrogging or block splitting from a stream of random numbers.

Bibliography

[1] P. Coddington, Random number generators for parallel computers, working paper (1997), available at `http://surface.syr.edu/npac/13/`.

[2] H. Bauke and S. Mertens, Random numbers for large-scale distributed Monte Carlo simulations, *Physical Review E* **75** (2007), 066701.

[3] F. Black and M. Scholes, The pricing of options and corporate liabilities, *Journal of Political Economy* **81** (1973), 637–659.

[4] P. Boyle, M. Broadie and P. Glasserman, Monte Carlo methods for security pricing, *Journal of Economic Dynamics and Control* **21** (1997), 1267–1321.

[5] A. De'Matteis and S. Pagnutti, Long-range correlations in linear and non-linear random number generators, *Parallel Computing* **14** (1990), 207–210.

[6] A. M. Ferrenberg, D. P. Landau and Y. J. Wong, Monte Carlo simulations: Hidden errors from 'good' random number generators, Physical Review Letters **69** (1992), 3382–3384.

[7] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Springer-Verlag, New York, 2004.

[8] P. Hellekalek, Good random number generators are (not so) easy to find, *Mathematics and Computers in Simulation* **46** (1998), 485–505.

[9] P. Jäckel, *Monte Carlo Methods in Finance*, Wiley, Chichester, 2002.

[10] C. Joy, P. P. Boyle and K. S. Tan, Quasi-Monte Carlo methods in numerical finance *Management Science* **42** (1996), 926–938.

[11] I. Karatzas and S. E. Shreve, *Brownian Motion and Stochastic Calculus*, Springer-Verlag, New York, 1991.

[12] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, Berlin, 1992.

[13] P. L'Ecuyer and R. Simard, TestU01: A C library for empirical testing of random number generators, *ACM Transactions on Mathematical Software* **33** (2007), Article 22.

[14] P. L'Ecuyer, R. Simard, J. Chen and W. D. Kelton, An object-oriented random-number package with many long streams and substreams, Technical report, University of Montreal, available at `http://www.iro.umontreal.ca/~lecuyer/myftp/streams00/c++/`.

[15] M. Mascagni and A. Srinivasan, Algorithm 806. SPRNG: A scalable library for pseudorandom number generation, *ACM Transaction on Mathematical Software* **26** (2000), 436–461.

[16] A. Srinivasan, M. Mascagni, and D. Ceperley, Testing parallel random number generators, *Parallel Computing* **29** (2003), 69–94.

[17] SPRNG – Scalable Parallel Random Number Generators library, SPRNG 1.0 available at http://www.ncsa.uiuc.edu/Apps/SPRNG, SPRNG 2.0 available at http://sprng.cs.fsu.edu, SPRNG through 4.0 available at http://www.sprng.org.

[18] Tina's Random Number Generator Library, http://trng.berlios.de/.

Received September 29, 2011; accepted February 20, 2012.

Author information

Michael Mascagni, Departments of Computer Science, Mathematics & Scientific Computing, and Graduate Program in Molecular Biophysics, Florida State University, Tallahassee, FL 32308-4530, USA.
E-mail: mascagni@fsu.edu

Lin-Yee Hin, Emphron Informatics, Level 3, 88 Jephson St.,
Toowong, Queensland 4066, Australia.
E-mail: linyeehin@gmail.com