# On the Efficiency of Simplified Weak Taylor Schemes for Monte Carlo Simulation in Finance

Nicola Bruti Liberati and Eckhard Platen

University of Technology Sydney, School of Finance & Economics and Department of Mathematical Sciences, PO Box 123, Broadway, NSW, 2007, Australia

**Abstract.** The purpose of this paper is to study the efficiency of simplified weak schemes for stochastic differential equations. We present a numerical comparison between weak Taylor schemes and their simplified versions. In the simplified schemes discrete random variables, instead of Gaussian ones, are generated to approximate multiple stochastic integrals. We show that an implementation of simplified schemes based on random bits generators significantly increases the computational speed. The efficiency of the proposed schemes is demonstrated.

## 1 Introduction

As described, for instance, in [7] to price an option via simulation, one does not require a pathwise approximation of the solution of the underlying stochastic differential equation (SDE). Only an approximation of its probability distribution has to be considered. Thus the appropriate notion of convergence for such a Monte Carlo simulation should be the weak one, instead of the strong convergence, as described in [6]. It is well known that in order to achieve a certain order of weak convergence one can approximate the random variables in a weak Taylor scheme by appropriate discrete random variables. For instance, instead of a Gaussian increment we can employ in an Euler scheme a much simpler two point distributed random variable. In general, the simplified random variables have to coincide only for certain lower order moments with those of the random variables appearing in the Taylor schemes. In the case of a weak Taylor scheme of second order, to construct a second order simplified method we can use a three point distributed random variable. The aim of this paper is to show that an implementation of such simplified schemes based on random bits generators significantly increases the computational efficiency.
It should be noticed that the simplified Euler method is equivalent to some random walk, which again is approximately equivalent to a binomial tree. The possible states of the tree and of the simplified Euler scheme are approximately the same. Small differences arise only for the level of these states depending on the chosen parametrization of the binomial tree. However, while the tree is a deterministic backward algorithm, the simplified method is a forward method which generates paths. As we will report in Section 4, the numerical behaviour of simplified methods is similar to that of trees. For instance, we will obtain

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 771–778, 2004.

an oscillatory convergence in the case of a European call payoff. This is a well-known effect of tree methods, but as will be shown, not limited to this class of methods, see, for instance, [2].
The widespread application of the tree methodology in finance motivates the application of the simplified schemes that will be presented in this note. The similarity between simplified schemes and tree methods is important for the understanding of the numerical properties for both types of methods.
Simplified schemes, being forward algorithms, are not easily suitable to price American options, even that corresponding algorithms have been developed, see for instance [8]. Moreover, with the simplified methods we always have to face the typical statistical error from Monte Carlo simulations. Major advantages of simplified schemes over tree methods are that of flexibility and general applicability in high dimensions.
The implementation of random bits generators will be proposed in this note. It makes simplified methods highly efficient. As shown in [5], implicit simplified methods can overcome certain numerical instability. Most importantly, random bits generators can be efficiently applied to implicit schemes, while tree methods cannot be made implicit. Note that simplified implicit schemes can be understood as being equivalent to implicit finite difference partial differential equation(PDE) methods. However, PDE methods cannot be easily implemented for higher dimensions.
The order of convergence of simplified schemes is independent of the dimension of the problem. As shown in [1], around dimension three or four simulation methods typically become more efficient than tree or PDE methods. It will be shown that simplified methods with random bits generators outperform significantly Taylor schemes, which are based on Gaussian and other random variables. This makes simplified methods with random bits generators efficient tools for high dimensional problems.

## 2 Weak Taylor Schemes and Simplified Methods

For the dynamics of the underlying security let us consider the following SDE:

$$dX_t = a(t, X_t)dt + b(t, X_t)dW_t \tag{1}$$

for $t \in [0, T]$, with $X_0 \in R$. A derivative pricing problem consists in computing an expectation of a payoff function $g(X_T)$ of the solution of the SDE (1). For the numerical approximation of such an expectation we require only an approximation of the probability distribution $X_T$. Therefore, the appropriate notion of convergence is that of weak convergence, see [6].
Let us assume an equidistant time dicretisation with $n$th discretisation time $t_n = n\Delta$ for $n \in \{0, 1, \ldots, N\}$ where $\Delta = \frac{T}{N}$ and $N \in \{1, 2, \ldots\}$. As a set of test functions we use the space $\mathcal{C}_P^l$ of the $l$ times continuously differentiable functions $g$ which, together with their partial derivatives of orders up to and including $l$, have polynomial growth.
We say that a time discrete approximation $Y^\Delta = \{Y_t^\Delta,\, t \in [0, T]\}$ converges weakly to $X = \{X_t,\, t \in [0, T]\}$ at time T with order $\gamma$ if for each $g \in \mathcal{C}_P^{2(\gamma+1)}$

there exists a positive constant $K$, which does not depend on $\Delta$, and a $\Delta_0 > 0$ such that $\varepsilon(\Delta) = |E(g(X_T) - E(g(Y_N^{\Delta}))| \leq K\Delta^{\gamma}$ for each $\Delta \in (0, \Delta_0)$.
As explained in [6], based on the Wagner-Platen expansion one can construct the, so called, weak Taylor schemes of any given weak order $\gamma \in \{1, 2, \ldots\}$. The simplest weak Taylor scheme is the *Euler method*, which has the weak order of convergence $\gamma = 1.0$. It is given by the scheme

$$Y_{n+1} = Y_n + a(t_n, Y_n)\Delta + b(t_n, Y_n)\Delta W_n, \tag{2}$$

where $\Delta W_n = W_{t_{n+1}} - W_{t_n}$ is the Gaussian increment of the Wiener process $W$ for $n \in \{0, 1, 2 \ldots, N-1\}$ and $Y_0 = X_0$.
If one uses in the above Euler scheme instead of Gaussian random variables simpler multi-point distributed random variables, then one can still obtain the same weak order of convergence $\gamma = 1.0$, see Theorem 14.5.2 p. 474 in [6]. For the Euler method these simpler random variables have to coincide in their first three moments with those of the Gaussian Wiener process increments. This permits to replace the Gaussian increment $\Delta W_n$ in (2), by a two point distributed random variable $\Delta\widehat{W}_n$, where $P(\Delta\widehat{W}_n = \pm\sqrt{\Delta}) = \frac{1}{2}$. We then obtain the *simplified Euler scheme*. Here the first three moments of the Wiener process increments $\Delta W_n$ match those of $\Delta\widehat{W}_n$.
The same applies to the *order* 2.0 *weak Taylor scheme*

$$\begin{aligned} Y_{n+1} = Y_n + a\Delta + b\Delta W_n + \frac{1}{2}b'b\left\{\left(\Delta W_n^2\right) - \Delta\right\} + \frac{1}{2}\left(aa' + \frac{1}{2}a''b^2\right)\Delta^2 \\ + a'b\,\Delta Z_n + \left(ab' + \frac{1}{2}b''b^2\right)\{\Delta W_n\,\Delta - \Delta Z_n\}, \end{aligned} \tag{3}$$

where $\Delta Z_n$ represents the double Itô integral $\Delta Z_n = \int_{t_n}^{t_{n+1}} \int_{t_n}^{s_2} dW_{s_1} ds_2$. Here we replace the Gaussian random variables $\Delta W_n$ and $\Delta Z_n$ by expressions that use a three point distributed random variable $\Delta\widehat{W}_n$ with $P(\Delta\widehat{W}_n = \pm\sqrt{3\Delta}) = \frac{1}{6}$ and $P(\Delta\widehat{W}_n = 0) = \frac{2}{3}$.
Then we obtain the *second order simplified method*

$$\begin{aligned} Y_{n+1} = Y_n + a\Delta + b\Delta\widehat{W}_n + \frac{1}{2}bb'\left\{\left(\Delta\widehat{W}_n\right)^2 - \Delta\right\} + \frac{1}{2}\left(aa' + \frac{1}{2}a''b^2\right)\Delta^2 \\ + \frac{1}{2}\left(a'b + ab' + \frac{1}{2}b''b^2\right)\Delta\widehat{W}_n\Delta. \end{aligned} \tag{4}$$

Since the three point distributed random variable $\Delta\widehat{W}_n$ is such that the first five moments of the increments of the schemes (3) and (4) are matched, the second order simplified scheme (4) can be shown to achieve the weak order $\gamma = 2.0$.
By using four or even five point distributed random variables for approximating the random variables needed, we can obtain simplified weak Taylor schemes of weak order $\gamma = 3$ or 4, respectively, as shown in [6] and in [4].
An important issue for simulation methods for SDEs is their numerical stability. As noticed in [5], when considering test equations with multiplicative noise, the weak schemes described above show narrow regions of numerical stability. In

order to improve the numerical stability one needs to introduce implicitness in the diffusion terms. This leads, for instance, to the *fully implicit Euler scheme*

$$Y_{n+1} = Y_n + \left\{ a\left(t_{n+1}, Y_{n+1}\right) - b\left(t_{n+1}, Y_{n+1}\right) \frac{\partial}{\partial y} b\left(t_{n+1}, Y_{n+1}\right) \right\} \Delta + b\left(t_{n+1}, Y_{n+1}\right) \Delta W_n. \tag{5}$$

Also in this case one can employ the two point distributed random variable $\Delta \widehat{W}_n$ instead of $\Delta W_n$ in (5) to obtain the *simplified fully implicit Euler scheme* that still achieves an order $\gamma = 1.0$ of weak convergence.

## 3 Random Bits Generators

We now demonstrate, for simplified schemes, how to implement highly efficient random bits generators, that exploit the architecture of a digital computer. The crucial part of the resulting simplified schemes, are the random bits generators. These substitute the Gaussian random number generators needed for weak Taylor schemes.

A well known and efficient method to generate a pair of independent standard Gaussian random variables is the polar Marsaglia-Bray method coupled with a linear congruential random number generator, as described in [9]. In our comparative study we use, as our Gaussian random number generator, the routine *gasdev*, see p. 293 of [9].

For the simplified Euler scheme (2) and simplified fully implicit Euler scheme (5) we use a two point distributed random variable in each time step, which is obtained from a random bits generator. This generator is an algorithm that generates a single bit 0 or 1 with probability 0.5. The method implemented is based on the theory of primitive polynomials modulo 2. These are polynomials satisfying particular conditions whose coefficients are zero or one. The important property is that every primitive polynomial modulo 2 of order $n$ defines a recurrence relation for obtaining a new bit from the n preceding ones with maximal length. This means that the period lenght of the recurrence relation is equal to $2^n - 1$. For a study on random number generators based on primitive polynomials modulo 2 we refer to [11].

Since the random number generator for the polar Marsaglia-Bray method has a period of $2^{31}$ we use a random bits generator based on the following primitive polynomial modulo 2 of order 31: $y(x) = x^{31} + x^3 + 1$. The C++ implementation of this generator is reported in Figure 1, see also [9]. This method is extremely fast and suitable for direct hardware implementation.

On the test computer the CPU time needed to generate 10 million random numbers with the polar Marsaglia-Bray method amounts to 4.7 seconds. The two point random bits generator, described above, is almost 30 times faster using only 0.16 seconds.

For simplified methods of higher order similar multi-point random bits generators can be constructed. For the second order simplified method (4) it is sufficient to use a three point random bits generator. A corresponding code is presented in Figure 2. It produces three bits coupled with an acceptance-rejection method.

```
int irbit1per31(unsigned long & iseed)
{   unsigned long newbit;
    newbit = ((iseed >> 31) & 1)
    ^  ((iseed >> 2) & 1);
    iseed = (iseed << 1) | newbit;
    return int(newbit); }
```

**Fig. 1.** C++ code of the two point random bits generator.

On the test computer the CPU time needed to generate 10 million random numbers with this generator amounts to 0.8 seconds, which is still 5 times less than the polar Marsaglia-Bray method.

```
int ranbit3per31(unsigned long & iseed)
{   int x1 = 1, x2 = 1, x3 = 0;
    while ( (x1 == 1 && x2 == 1 && x3 == 0)
    || (x1 == 0 && x2 == 1 && x3 == 1))
    {  x1 = irbit1per31(iseed);
       x2 = irbit1per31(iseed);
       x3 = irbit1per31(iseed); }
       return x1 - x3; }
```

**Fig. 2.** C++ code of the three point random bits generator.

## 4 Numerical Results

Now, we present some numerical results for the Euler, fully implicit Euler and order 2.0 weak Taylor schemes as well as their simplified versions. As test dynamics we choose an SDE with multiplicative noise of the Black-Scholes type, where

$$dX_t = \mu X_t dt + \sigma X_t dW_t \tag{6}$$

for $t \in [0, T]$. The SDE admits the closed form solution $X_T = X_0 \exp\{(\mu - \frac{\sigma^2}{2})T + \sigma W_T\}$. The CPU times needed to compute 4 million approximate paths with 64 time steps with the Euler, fully implicit Euler and order 2.0 weak Taylor scheme amount to 107, 114 and 110 seconds, respectively. The corresponding approximate simplified versions only require 3.8, 6.2 and 25.6 seconds, respectively. Thus, for the Euler method the simplified version is roughly 28 times faster than the Gaussian one. The simplified fully implicit Euler method is about 18 times

faster than its Gaussian counterpart. For the second order simplified method we found that it is roughly four times more efficient than the order 2.0 weak Taylor scheme.
We analyse now the weak convergence of Monte Carlo simulations when using a smooth payoff function, where we choose the first moment for illustration and consider later on also a non smooth payoff which will be that of a European call option.

### 4.1 A Smooth Payoff Function

At first, we study the weak error for a fixed number of simulations and time steps. We also compare the CPU time needed to reach a given accuracy. In order to analyse the weak error $\varepsilon(\Delta)$, we run sufficiently many simulations such that the statistical error can be neglected. We use the following parameters: $X_0 = 1$, $\mu = 1.5$, $\sigma = 0.01$, $T = 1$.
An important application of Monte Carlo simulation is the calculation of Value at Risk via the simulation of moments, as applied in Edgeworth expansions and saddle point methods, see [10]. Therefore, as test function we use the first moment $E(X_T)$ of $X_T$ at time $T$. Other moments give similar numerical results due to the lognormal structure of the Black-Scholes dynamics. We then estimate the weak error of the first moment by comparing the simulated Monte Carlo estimate with the exact expectation $E(X_T) = X_0\, e^{\mu T}$.
In the first plot of Figure 3 we show the logarithm $log_2(\varepsilon(\Delta))$ of the weak error for the Euler, fully implicit Euler, and order 2.0 weak Taylor method versus the logarithm $log_2(\Delta)$ of the time step size. The errors for the corresponding simplified versions are almost identical and therefore omitted. The number of simulated paths amounted to 16 million, which resulted in extremely small confidence intervals that practically do not show up in Figure 3.
We emphasize the important observation that the simplified methods achieve

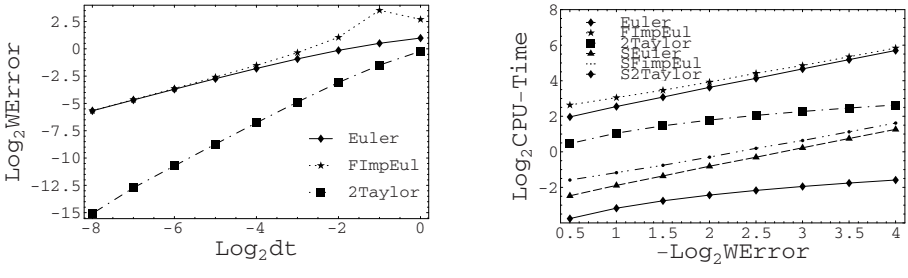


**Fig. 3.** Log-log plots of weak error versus time step size and CPU time versus the weak error for the Euler, fully implicit Euler and order 2.0 weak Taylor schemes.

almost exactly the same accuracy of their Taylor counterparts. Note in Figure 3 that the Euler and the fully implicit Euler scheme reproduce in the log-log plot the theoretically predicted weak order $\gamma = 1.0$. Furthermore, the order 2.0 weak

Taylor scheme achieves a weak order of about $\gamma = 2.0$, as expected. Moreover, we note in Figure 3 that the fully implicit Euler scheme shows poor results for very large step sizes. However, as shown in [5], the fully implicit method has better stability properties than the explicit schemes once the time step size becomes sufficiently small.
What really matters in practice is the time needed to reach a given level of accuracy. In the second plot of Figure 3 we show the logarithm of the CPU time versus the negative of the logarithm of the weak error observed for the three methods described above and their simplified versions. Since the accuracy for a given time step size is almost identical for the schemes of the same order, the increase in efficiency simply reflects the fact that the simplified schemes are computationally less intensive than their Gaussian counterparts. We recall that, for instance, the simplified Euler scheme is 28 times faster than the Euler scheme.
By comparing all six methods, we conclude that the second order simplified scheme is significantly more efficient for the given example than any other of the considered schemes. This result is rather important in simulations of Black-Scholes dynamics since it points out efficient Monte Carlo simulation algorithms for smooth payoffs.

### 4.2 An Option Payoff

In option pricing we are confronted with the computation of expectations of non smooth payoffs. To give a simple example, let us compute the price of a European call option. Here we have a continuous but only piecewise differentiable payoff $(X_T - K)^+ = \max(X_T - K, 0)$ with strike price $K$ and the well known Black-Scholes formula as closed form solution for the option price at time $t = 0$. For this non smooth payoff we study the weak error for the Euler and the simplified Euler method, assuming the volatility $\sigma = 0.2$ and the short rate $\mu = 0.1$. We observed no major gain by using higher order methods, which is likely to be due to the non smooth option payoff. Since the second order simplified method (4) is approximately equivalent to a trinomial tree, as discussed in Section 1, this is consistent with an observation in [3]. In [3] it was observed that in option pricing the order of convergence of trinomial trees is not superior to that of binomial trees.
In the first plot of Figure 4 we show the log-log weak error plot for an at the money-forward option, with strike $K = X_0\, e^{\mu T}$. The Euler method generates a weak order $\gamma = 1.0$ with the log error forming a perfect line in dependence on the log time step size. As mentioned earlier, the simplified Euler method is approximately equivalent to a binomial tree. This method still achieves a weak order $\gamma = 1.0$. However, its log-log error plot does not exhibit a perfect line, which is due to the discrete nature of the random variables used. This appears to be the same effect as noticed for tree methods, see [2]. We observed for in the money and out of the money options a similar order of convergence with similar log error patterns.
In the second plot of Figure 4 we show the logarithm of the CPU time versus the negative logarithm of the weak error. For the considered non smooth payoff the increase in computational speed is still about 28 times. The simplified Euler

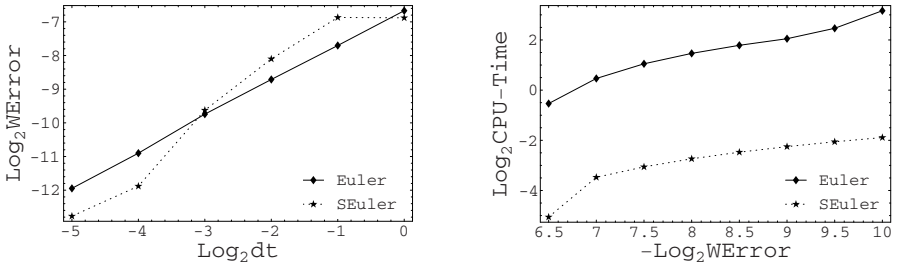


**Fig. 4.** Log-log plots of weak error versus time step size and CPU time versus the weak error for call option with Euler and simplified Euler scheme.

method is significantly more efficient than the Euler scheme, for every level of accuracy. We observed similar results also for in the money and out of the money options. In summary, one can say that the proposed rather simple random bits generators when combined with simplified schemes can significantly enhance the efficiency of typical Monte Carlo simulations in finance.

## References

1. Boyle, P., M. Broadie, & P. Glasserman (1997). Monte Carlo methods for security pricing. *J. Econom. Dynam. Control* **21**(8-9), 1267–1321.
2. Boyle, P. P. & S. H. Lau (1994). Bumping up against the barrier with the binomial method. *J. Derivatives*, 6–14.
3. Heston, S. L. & G. Zhou (2000). On the rate of convergence of discrete-time contingent claims. *Math. Finance* **10**(1), 53–75.
4. Hofmann, N. (1994). *Beiträge zur schwachen Approximation stochastischer Differentialgleichungen*. Ph. D. thesis, Dissertation A, Humboldt Universität Berlin.
5. Hofmann, N. & E. Platen (1996). Stability of superimplicit numerical methods for stochastic differential equations. *Fields Inst. Commun.* **9**, 93–104.
6. Kloeden, P. E. & E. Platen (1999). *Numerical Solution of Stochastic Differential Equations*, Volume 23 of *Appl. Math.* Springer. Third corrected printing.
7. Kloeden, P. E., E. Platen, & H. Schurz (2003). *Numerical Solution of SDE's Through Computer Experiments*. Universitext. Springer. Third corrected printing.
8. Longstaff, F. A. & E. S. Schwartz (2001). Valuing American options by simulations: A simple least-squares approach. *Rev. Financial Studies* **14**(1), 113–147.
9. Press, W. H., S. A. Teukolsky, W. T. Vetterling, & B. P. Flannery (2002). *Numerical Recipes in C++. The art of Scientific Computing* (2nd ed.). Cambridge University Press.
10. Studer, M. (2001). *Stochastic Taylor Expansions and Saddlepoint Approximations for Risk Management*. Ph. D. thesis, Swiss Federal Institute of Technology Zurich.
11. Tausworthe, R. C. (1965). Random numbers generated by linear recurrence modulo two. *Mathematics of Computation* **19**, 201–209.