

Applied Stochastic Processes and Control for Jump-Diffusions: Modeling, Analysis and Computation

Floyd B. Hanson
University of Illinois
Chicago, Illinois, USA

Chapter 10 Stochastic Simulations

Copyright © 2006 by the Society for Industrial and Applied Mathematics.

May 24, 2006

DRAFT

Chapter 10

Stochastic Simulations

Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin.

—John von Neumann (1903-1957),
apparently meant just as a caution,
http://en.wikiquote.org/wiki/John_von_Neumann.

*Fast cars, fast women, fast algorithms...
what more could a man want?*

—Joe Mattis at http://www.xs4all.nl/~jcdverha/scijokes/1_5.html#subindex.

Methods are considered that treat stochastic dynamics, such as direct simulations of SDEs [162, 161] with many numerical techniques offering improvements over the elementary integration methods beyond stochastic versions of Euler's method.

Monte Carlo methods simulate solutions to higher level applications, which include many improvements to increase the probable accuracy in order to reduce the need of large scale sample sizes, many of the techniques involve variance reduction and generation sample variates for nonuniform distributions [96, 147, 285].

10.1 SDE Simulation Methods

Simulation methods for the dynamics of stochastic differential equations are discussed. Basic simulation procedures have been introduced in Chapters 2-5, but here the simulation of diffusion and jump-diffusion simulation is discussed and explored much further. Primary references are by Kloeden et al. [162], Cyganowski et al. [65, 64, 66], the compact review by D. Higham [136] and D. Higham and Kloeden [140, 141]. Many of these references deal almost entirely with diffusions and the most comprehensive, theoretically and numerically, on diffusions is the mono-

graph of Kloeden and Platen [161]. The references of Maghsoodi [187], Cyganowski et al. [65, 64] and D. Higham and Kloeden [141] treat jump-diffusions in a serious way. However, random simulations to solve stochastic optimal control problems are not too useful due to the additional complexity involved in the optimization step, while optimal control problems can be reduced to deterministic ODE or PDE formulations which can be solved more systematically.

10.1.1 Convergence and Stability for Stochastic Problems and Simulations

Consider the jump-diffusion stochastic differential equation.

$$dX(t) = f(X(t), t)dt + g(X(t), t)dW(t) + h(X(t), t)dP(t), \quad (10.1)$$

$X(0) = x_0$ with probability one and $0 \leq t \leq t_f$, where the coefficient functions $f(X(t), t)$, $g(X(t), t)$ and $h(X(t), t)$ are continuously differentiable (see [161] for tighter conditions; $h(X(t), t)$ could also depend on random marks Q).

In Section 4.3.3, the main concern was formal SDE simulations, but here there will be more attention on convergence of the simulations. Let t_k denote a discrete time such that $t_{k+1} = t_k + \Delta t$ for $k = 0 : N_t - 1$, so $t_{N_t} = t_f$ and $\Delta t = t_f / N_t$. For the state, let X_k denote the discrete approximation at time t_k to the exact value $X(t_k)$, i.e., $X_k \simeq X(t_k)$.

Definition 10.1. *The approximation X_k is said to **converge** to the exact value $X(t_k)$,*

- in the **strong mean absolute error sense** if the conditional expectation,

$$\mathbb{E}[|X_k - X(t_k)| \mid X(0) = x_0] \rightarrow 0^+ \text{ as } \Delta t \rightarrow 0^+, \quad (10.2)$$

for fixed time $t_k = k\Delta t$, e.g., $t_f = t_{N_t}$;

the strong convergence in the mean absolute error is said to be **order** or with **log-rate** $\gamma_s > 0$ in mean absolute error if

$$\mathbb{E}[|X_k - X(t_k)| \mid X_0 = x_0] \leq C_s \cdot (\Delta t)^{\gamma_s}, \quad (10.3)$$

for sufficiently small Δt , for fixed time $t_k = k\Delta t$, e.g., $t_f = t_{N_t}$ and constant $C_s > 0$.

- in the **weak sense** if the difference in conditional expectations,

$$|\mathbb{E}[X_k \mid X_0 = x_0] - \mathbb{E}[X(t_k) \mid X(0) = x_0]| \rightarrow 0^+ \text{ as } \Delta t \rightarrow 0^+, \quad (10.4)$$

for fixed time $t_k = k\Delta t$, e.g., $t_f = t_{N_t}$;

the weak convergence is said to be **order** or with **log-rate** $\gamma_w > 0$ in mean error if

$$|\mathbb{E}[X_k \mid X_0 = x_0] - \mathbb{E}[X(t_k) \mid X(0) = x_0]| \leq C_w \cdot (\Delta t)^{\gamma_w}, \quad (10.5)$$

for sufficiently small Δt , for fixed time $t_k = k\Delta t$, e.g., $t_f = t_{N_t}$ and constant $C_w > 0$.

- Alternately, **strong convergence in mean square error (mse)**, instead of mean error, can be defined (Maghsoodi [187]),

$$\sup_k (\mathbb{E} [(X_k - X(t_k))^2 | X_0 = x_0]) \leq C_s^{(\text{mse})} \cdot (\Delta t)^{\gamma_s^{(\text{mse})}}, \quad (10.6)$$

for sufficiently small Δt and constant $C_s^{(\text{mse})} > 0$; thus the maximal root mean square error rate is

$$O\left((\Delta t)^{\gamma_s^{(\text{mse})}/2}\right),$$

so it is fair to compare the mean absolute error rate γ_s with the root mean square error rate $\gamma_s^{(\text{mse})}/2$.

For ordinary differential equations, a solution $X(t)$ is **asymptotic stable** as $t \rightarrow +\infty$ if

$$\lim_{t \rightarrow +\infty} |X(t)| = 0,$$

in the continuous time case and in the discrete time case the approximation X_k is **asymptotic stable** as $k \rightarrow +\infty$ if

$$\lim_{k \rightarrow +\infty} |X_k| = 0.$$

However, such a definition is not applicable even if the coefficient functions are bounded and otherwise nicely behaved, since for diffusions the range of the random process $W(t)$ is infinite. Thus, the notion of stochastic asymptotic stability has to be modified for stochastic processes.

Definition 10.2.

- For continuous time, the real stochastic solution $X(t)$ is said to be **asymptotically mean square stable** if

$$\lim_{t \rightarrow +\infty} \mathbb{E} [X^2(t) | X(0) = x_0] = 0. \quad (10.7)$$

Alternately, $X(t)$ is **asymptotically stable in probability** if

$$\text{Prob} \left[\lim_{t \rightarrow +\infty} |X(t)| = 0 \mid X(0) = x_0 \right]. \quad (10.8)$$

- For discrete time, the real stochastic approximation X_k is said to be **asymptotically mean square stable**

$$\lim_{k \rightarrow +\infty} \mathbb{E} [X_k^2 | X_0 = x_0] = 0. \quad (10.9)$$

Alternately, X_k is **asymptotically stable in probability** if

$$\text{Prob} \left[\lim_{k \rightarrow +\infty} |X_k| = 0 \mid X_0 = x_0 \right]. \quad (10.10)$$

As a continuous-time example, consider the linear, constant coefficient SDE, letting $(f(x, t), g(x, t), h(x, t)) = (\mu_0, \sigma_0, \nu_0)$ in (10.1),

$$dX(t) = X(t)(\mu_0 dt + \sigma_0 dW(t) + \nu_0 dP(t)),$$

where μ_0, σ_0, ν_0 and λ_0 are constants and where $E[dP(t)] = \lambda_0 dt$. From (4.80), the exact solution is

$$X(t) = x_0 \exp((\mu_0 - \sigma_0^2/2)t + \sigma_0 W(t))(1 + \nu_0)^{P(t)}. \quad (10.11)$$

Using the independent increment techniques for the expectation in (4.81), the mean square is

$$E[X^2(t_f) | X(0) = x_0] = x_0^2 e^{(2(\mu_0 + \lambda_0 \nu_0) + \sigma_0^2 + \lambda_0 \nu_0^2)t_f}.$$

Thus, $X(t_f)$ is asymptotically mean square stable if the exponential is decaying as $t_f \rightarrow +\infty$, so

$$2(\mu_0 + \lambda_0 \nu_0) + \sigma_0^2 + \lambda_0 \nu_0^2 < 0, \quad (10.12)$$

which, in qualitative terms of the relative conditional infinitesimal moments, can be put in the form:

$$E[dX(t)/X(t) | X(t)] < -0.5 \text{Var}[dX(t)/X(t) | X(t)],$$

assuming $x_0 > 0$ so $X(t) > 0$. Hence, the combined jump-diffusion relative infinitesimal mean has to be less than minus one-half of the relative infinitesimal variance.

10.1.2 Stochastic Diffusion Euler Simulations

The simplest simulation model using Euler's Method for SDEs is more properly called the **Euler-Maruyama (EM) method** to distinguish it from the deterministic Euler method for DEs and this was used in Section 4.3.3 in this text and has the stochastic difference form

$$X_{k+1} = X_k + F_k \Delta t + G_k \Delta W_k, \quad (10.13)$$

for $k = 0 : N_t - 1$, where $F_k \equiv f(X_k, t_k)$, $G_k \equiv g(X_k, t_k)$ and $\Delta W_k \equiv W(t_{k+1}) - W(t_k)$. For instance, in MATLAB™, a fragment of the code for the discrete diffusion approximation for a linear would be like that given in Fig. 10.1. Recall that MATLAB™ is unit based, i.e., array subscripts start at one. In this example, the drift coefficient rate is time-dependent with $f(x, t) = \mu(t)x$ where $\mu(t) = 1/(1 + 0.5t)^2$, but the $dW(t)$ -coefficient is time-independent with $g(x, t) = \sigma(t)x$ where $\sigma(t) = \sigma_0$ where σ_0 is a constant, i.e.,

$$dX(t) = X(t)(\mu(t)dt + \sigma(t)dW(t)). \quad (10.14)$$

In this case the log-transformation $Y(t) = \ln(X(t))$ by the Itô stochastic chain rule leads to a state-independent SDE, $Y(t) = (\mu(t) - \sigma^2(t)/2)dt + \sigma(t)dW(t)$ and a

```

function sdeulersim
% Euler-Maruyama Simulation Test: Linear SDE:
% dX(t) = X(t)(mu(t)dt+sigma(t)dW(t)),
% Given Initial data: x0, t0, tf, Nt; functions: f, g, xexact
clc
%
randn('state',8); % Set random state or seed;
x0 = 1; t0 = 0; tf = 5; Nt = 2^14; DT = tf/Nt; sqrtt = sqrt(DT);
X(1) = x0; Xexact(1) = x0; t = [t0:DT:tf];
DW = randn(1,Nt)*sqrtt; % Simulate DW as sqrt(DT)*randn;
W = cumsum(DW); % Omits initial zero value;
for k = 1:Nt % Exact formula to fine precision}
    Xexact(k+1) = xexact(x0,t(k+1),W(k)); % Calls subfunction;
end
L = 2^3; NL = Nt/L; KL = [0:L:Nt]; DTL = L*DT; tL = [t0:DTL:tf];
for k = 1:NL % Euler formula to lumped, coarse precision:
    DWL = sum(DW(1,KL(k)+1:KL(k+1)));
    Xeul(k+1)=Xeul(k)+f(Xeul(k),tL(k))*DTL+g(Xeul(k),tL(k))*DWL;
    Xdiff(k+1) = Xeul(k+1) - Xexact(KL(k+1));
end
plot(tL,Xeul,'k--','linewidth',3); hold on
plot(t,Xexact,'k-','linewidth',3); hold off
title('SDE Euler-Maruyama and Exact Linear SDE Simulations');
xlabel('t, Time'); ylabel('X(t), State');
legend('X(t): Euler','Xexact: Exact','Location','Best');
%
function y = f(x,t)
    mu = 1/(1+0.5*t)^2; % Change with application;
    y = mu*x;
%
function y = g(x,t)
    sig = 0.5; % Change with application;
    y = sig*x;
%
function y = xexact(x0,t,w)
% exact solution if available for general linear SDE:
    mubar = 2-2/(1+0.5*t); sig = 0.5; sig2bar = sig^2*t/2;
    y = x0*exp(mubar-sig2bar + sig*w);
%End Code

```

Figure 10.1. Code: Euler SDE simulations.

simple integration followed by a transformation inversion leads to the general exact stochastic solution

$$X^{(\text{exact})}(t) = x_0 \exp(\overline{\mu}(t) - \overline{\sigma^2}(t)/2 + \overline{(\sigma * W)}(t)), \quad (10.15)$$

where $\overline{\mu}(t) = \int_0^t \mu(s)ds$, $\overline{\sigma^2}(t) = \int_0^t \sigma^2(s)ds$ and $\overline{(\sigma * W)}(t) = \int_0^t \sigma(s)dW(s)$, which in the simpler case here reduces the integral to $\overline{(\sigma * W)}(t) = \sigma_0 W(t)$, so that an

approximation of this diffusion integral is not necessary. Equation (10.15) is an exact formula, but comparison of the Euler-Maruyama approximation to that of the exact requires an approximate simulation of $W(t)$ in $(\sigma * W)(t)$. Following D. Higham's [136] lead, a fine grid of N_t sample points is used for the exact formula and a lumped, coarse grid with $N_t/8$ points is taken from the set for the exact case. This makes for a more accurate comparison. The comparison between the coarse Euler-Maruyama approximation and the fine exact approximation $X^{(\text{exact})}(t)$ in (10.15), is illustrated in Fig. 10.2. The error between the Euler-Maruyama approximate path

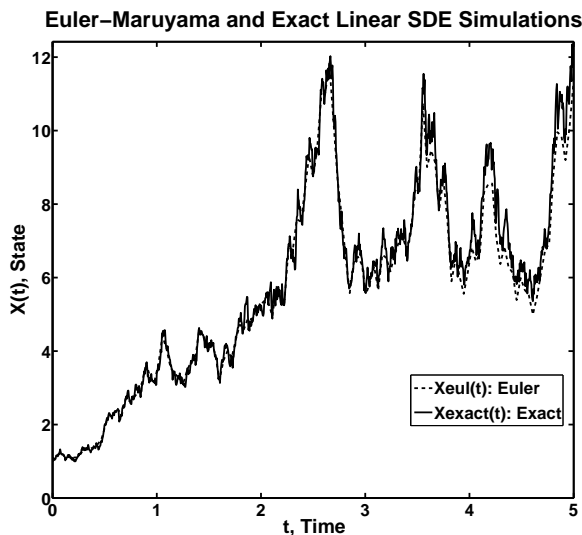


Figure 10.2. Comparison of coarse Euler-Maruyama and fine exact paths, simulated using MATLAB with $N_t = 1024$ fine sample points for the exact path (10.15) and $N_t/8 = 128$ coarse points for the Euler path (10.13), initial time $t_0 = 0$, final time $t_f = 5$ and initial state $x_0 = 1.0$. Time-dependent parameter values are $\mu(t) = 0.5/(1 + 0.5t)^2$ and $\sigma(t) = 0.5$.

and the exact path at the coarse time points is presented in Fig. 10.3. For further computer experiments verifying convergence using paths averages, see D. Higham [136]. For the complete sample code used to generate these Euler-Maruyama figures, see Sect. A.16 of Appendix A.

Kloeden and Platen [161, Section 10.2] show for the Euler-Maruyama simulation method, using a level of analysis beyond the scope of this text, that the log-rate of convergence in the strong sense is $\gamma_s = 0.5$, while in the weak sense the rate is $\gamma_w = 1$. Thus, the log-rate for convergence in the weak sense is the same as that for the traditional Euler's method applied to deterministic DEs in the strong or weak sense, i.e., $\gamma = 1$ for the deterministic case, since the expectation operator plays no role.

For convergence in the weak sense, the Euler-Maruyama method and the lin-

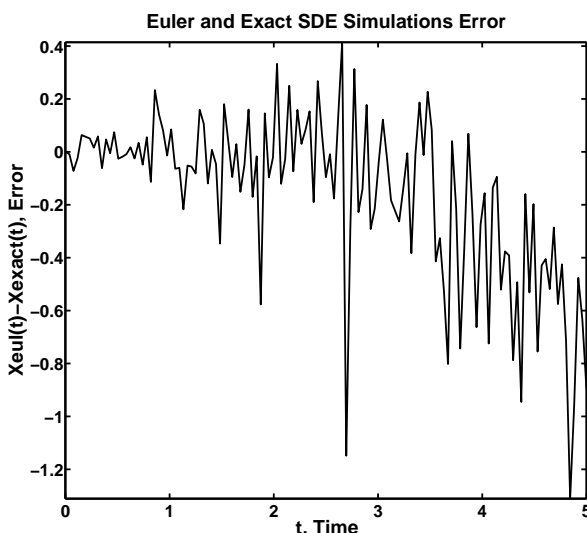


Figure 10.3. Error in coarse Euler-Maruyama and fine exact paths using the coarse discrete time points. The simulations use MATLAB with the same values and time-dependent coefficients as in Fig. 10.2. The Euler maximal-absolute error for this example is $1.3 \approx 34\Delta t/8$, while for $N_t = 4096$ the maximal error is better at $0.28 \approx 29\Delta t/8$.

ear, constant rate SDE,

$$dX(t) = \mu_0 X(t)dt + \sigma_0 X(t)dW(t),$$

where μ_0 and σ_0 are constants, the log-rate result can be shown with a reasonable effort. From (10.15) or (10.11) with $\nu_0 = 0$, the exact solution is

$$X^{(\text{exact})}(t) = x_0 \exp((\mu_0 - \sigma_0^2/2)t + \sigma_0 W(t)).$$

In this case, the EM approximation from (10.13) has the form of a stochastic difference equation (SΔE),

$$X_k = X_{k-1} \cdot (1 + \mu_0 \Delta t + \sigma_0 \Delta W_{k-1}), \quad (10.16)$$

for $k = 1:N_t$, and the expectation of X_k conditioned on the past value X_{k-1} is

$$\mathbb{E}[X_k | X_{k-1}] = X_{k-1} \cdot (1 + \mu_0 \Delta t),$$

so by iterated expectations

$$\mathbb{E}[X_k | X(0) = x_0] = (1 + \mu_0 \Delta t) \mathbb{E}[X_{k-1} | X_j, j = 0:k-2] = (1 + \mu_0 \Delta t)^k x_0$$

and finally $E[X_{N_t} | X(0) = x_0] = x_0(1 + \mu_0\Delta t)^{N_t}$ at $t_{N_t} = t_f$. From (4.81), for jump-diffusions but ignoring the jumps, the expectation of the exact solution at the final fixed time is

$$E \left[X^{(\text{exact})}(t_f) \mid X(0) = x_0 \right] = x_0 e^{\mu_0 t_f}.$$

The asymptotic evaluation, for sufficiently small Δt , of weak convergence criteria is then

$$\begin{aligned} |E[X_{N_t} | X_0 = x_0] - E[X^{(\text{exact})}(t_f) | X(0) = x_0]| &= |x_0| \cdot |(1 + \mu_0\Delta t)^{N_t} - e^{\mu_0 t_f}| \\ &= |x_0| \cdot |e^{N_t \ln(1 + \mu_0\Delta t)} - e^{\mu_0 N_t \Delta t}| \\ &\sim |x_0| e^{\mu_0 t_f} \cdot |e^{-0.5\mu_0^2 t_f \Delta t} - 1| \\ &\sim |x_0| e^{\mu_0 t_f} \cdot 0.5\mu_0^2 t_f \Delta t = \tilde{C}_w \Delta t, \end{aligned}$$

so $\gamma_w = 1$, as advertised, with $\tilde{C}_w = 0.5\mu_0^2|x_0|\exp(\mu_0 t_f)$, for both linear deterministic and stochastic Euler's method, although only in the weak sense in the linear stochastic case.

Finally, consider the mean square stability of the EM approximation X_k . Recasting the EM SΔE (10.16) to the recursion form $X_k = A_{k-1} \cdot X_{k-1}$, where $A_k \equiv (1 + \mu_0\Delta t + \sigma_0\Delta W_k)$, so that the solution can be written

$$X_k = x_0 \prod_{\ell=0}^{k-1} A_\ell.$$

Next, considering the mean square,

$$\begin{aligned} E[X_k^2 | X_0 = x_0] &= x_0^2 E \left[\left(\prod_{\ell=0}^{k-1} A_\ell \right)^2 \right] = x_0^2 E \left[\prod_{\ell=0}^{k-1} A_\ell^2 \right] = x_0^2 \prod_{\ell=0}^{k-1} E[A_\ell^2] \\ &= x_0^2 \prod_{\ell=0}^{k-1} ((1 + \mu_0\Delta t)^2 + \sigma_0^2 \Delta t) = x_0^2 ((1 + \mu_0\Delta t)^2 + \sigma_0^2 \Delta t)^k \quad (10.17) \\ &= x_0^2 (1 + 2\mu_0\Delta t + (\mu_0\Delta t)^2 + \sigma_0^2 \Delta t)^k, \end{aligned}$$

by interchanging the power and product operators, interchanging the product and expectation operators due to the independent increments property of the ΔW_k , using $E[\Delta W_\ell] = 0$ and $E[\Delta W_\ell^2] = \Delta t$, and the final fact that $\prod_{\ell=0}^{k-1} \theta = \theta^k$. Since as $k \rightarrow \infty$, $\theta^k \rightarrow 0$ if and only if $\theta < 1$ and in this case obviously $\theta > 0$, so asymptotic mean square stability of the X_k requires that

$$2\mu_0 + \sigma_0^2 + \mu_0^2 \Delta t < 0. \quad (10.18)$$

Note from (10.12) with $\nu_0 = 0$, the corresponding critical stability condition for the exact solution is $2\mu_0 + \sigma_0^2 < 0$ or that $\mu_0 < -0.5\sigma_0^2$ and that μ_0 must be sufficiently negative, but (10.18) for EM is much more restrictive requiring

$$\mu_0 < -0.5(\sigma_0^2 + \mu_0^2 \Delta t),$$

since the discrete term $\mu_0^2 \Delta t$ has been retained because Δt may not be so small to be negligible, although $\mu_0^2 dt$ would be negligible compared to one in the dt -precision

used in the exact, continuous time case. For numerical consideration, (10.18) could be interpreted as a constraint on the discrete time-step, i.e.,

$$\Delta t < 2 |\mu_0 + 0.5\sigma_0^2| / \mu_0^2,$$

valid only if μ_0 is selected to be in the asymptotically mean square stable range, $\mu_0 < -0.5\sigma_0^2$, of the exact solution. For more elaborate discussion of asymptotic stability, see D. Higham [136] for diffusions or D. Higham and Kloeden [142] for jump-diffusions.

10.1.3 Milstein's Higher Order Stochastic Diffusion Simulations

It is difficult to see how to improve on the Euler-Maruyama method (10.13) since it is perfectly consistent with Itô's formulation of forward integration of the diffusion stochastic integral equation

$$X(t) = X(0) + \int_0^t (f(X(s), s)ds + g(X(s), s)dW(s)), \quad (10.19)$$

corresponding to the diffusion SDE (10.1). Here, only a formal applied mathematical derivation is given, since comprehensive details fill the large volume of Kloeden and Platen [161]. Clues about where to start are the fact that Euler's method has a theoretical log-rate of $\gamma_s = 0.5$ for strong convergence ([161]) and that the same power obtained for just the expectation of absolute value of the standard diffusion process, $E[|\Delta W_k|] = O(\sqrt{E[\Delta W_k^2]}) = O(\sqrt{\Delta t})$ as given in Table 1.1 on page 85 of Chapter 1. The main idea of expanding the simulation approximation is to expand the coefficient $g(x, t)$ of the term whose expected absolute value would give rise to the $O(\sqrt{\Delta t})$ convergence. A way to do this is to apply iterations with Itô's stochastic chain rule in integral of $g(X(t), t)$ on $[t_k, t]$, $t \geq t_k$,

$$g(X(t), t) = g(X_k, t_k) + \int_{t_k}^t ((g_t + fg_x + 0.5g^2g_{xx})(X(s), s)ds + (gg_x)(X(s), s)dW(s)), \quad (10.20)$$

loosely upgrading the $g(x, t)$ requirements needed to twice continuously differential and where **wholesale arguments** have been used, e.g., $(gg_x)(x, t) = g(x, t)g_x(x, t)$.

This stochastic Taylor technique is also called an **Itô-Taylor expansion**. It can be used recursively to obtain very high order approximations, but here just (10.20) is substituted into a version of (10.19) rewritten for $[t_k, t_{k+1}]$,

$$\begin{aligned} X_{k+1} &= X_k + \int_{t_k}^{t_{k+1}} (f(X(t), t)dt + g(X(t), t)dW(t)) \\ &= X_k + \int_{t_k}^{t_{k+1}} (f(X(t), t)dt + (g(X_k, t_k) \\ &\quad + \int_{t_k}^t ((g_t + fg_x + 0.5gg_{xx})(X(s), s)ds \\ &\quad + (gg_x)(X(s), s)dW(s))) dW(t)) \\ &\simeq X_k + F_k\Delta t + G_k\Delta W_k + G_kGX_k \int_{t_k}^{t_{k+1}} \int_{t_k}^t dW(s)dW(t), \end{aligned} \quad (10.21)$$

where $GX_k \equiv g_x(X_k, t_k)$. Next, using the Itô forward integration approximation on coefficient terms and the negligibility of the residual double integral,

$$\int_{t_k}^{t_{k+1}} \int_{t_k}^t ds dW(t) = \int_{t_k}^{t_{k+1}} (t - t_k) dW(t) = \int_0^{\Delta t} t dW(t) \stackrel{\text{dt}}{=} 0,$$

by Itô mean square rules in dt -precision, which justifies dropping the corresponding terms. The retained double integral is just another form of Itô's fundamental Theorem 2.30 on page 117,

$$\begin{aligned} \int_{t_k}^{t_{k+1}} \int_{t_k}^t dW(s) dW(t) &= \int_{t_k}^{t_{k+1}} (W(t) - W_k) dW(t) \\ &= \left(\int_0^{\Delta t} W(t) dW(t) \right)_k \stackrel{\text{dt}}{=} 0.5 \cdot (\Delta W_k^2 - \Delta t). \end{aligned} \quad (10.22)$$

Thus, **Milstein's approximate method** is the stochastic difference equation (SΔE),

$$X_{k+1} = X_k + F_k \Delta t + G_k \Delta W_k + 0.5 G_k G X_k \cdot (\Delta W_k^2 - \Delta t), \quad (10.23)$$

for the SDE (10.1) and $k = 0 : N_t - 1$, where $F_k \equiv f(X_k, t_k)$, $G_k \equiv g(X_k, t_k)$, $G X_k \equiv g_x(X_k, t_k)$ and $\Delta W_k \equiv W(t_{k+1}) - W(t_k)$. Using the linear, time-dependent SDE model (10.14) as in Fig. 10.2 and the same fine-coarse grid numerical procedure, the Milstein and exact simulations are displayed in Fig. 10.4. The difference is very slight and hardly noticeable and the error between the Milstein approximate path and the exact path at the coarse time points is presented in Fig. 10.5. Finally, Fig. 10.6 illustrates the direct difference between the Milstein and Euler-Maruyama approximations. (For the sample code used to generate these Milstein figures, see Sect. A.17 of the Appendix.)

The Milstein algorithm converges strongly with log-rate $\gamma_s = 1$, but for the proof and computational justification see Kloeden and Platen [161, Sections 10.3 and 10.6]. Also see D. Higham's very accessible tutorial review [136] for computational justification and a nice Milstein-strong MATLAB™ code. Maple™ and MATLAB™ codes for diffusion SDEs for finance can be given in D. Higham and Kloeden [140] along with higher order approximations. Other diffusion Maple™ codes are found in Cyganowski, Kloeden and Ombach [66]. *Mathematica*™ diffusion SDE codes are presented in Stojanovic [254].

However, note that the diffusion factor $0.5(\Delta W_k^2 - \Delta t)$ in the Milstein approximation has the mean $E[0.5(\Delta W_k^2 - \Delta t)] = 0$ and variance

$$\text{Var}[0.5(\Delta W_k^2 - \Delta t)] = 0.25(E[\Delta W_k^4] - (\Delta t)^2) = 0.5(\Delta t)^2,$$

which normally would be negligible in dt -precision. Using Table 1.1 on page 85, indicates limited correction possibilities.

10.1.4 Convergence and Stability of Jump-Diffusion Euler Simulations

The stochastic Euler's method for jump-diffusions governed by the SDE (10.1) with discrete Poisson jumps at mark-independent amplitudes $h(x, t)$, i.e.,

$$dX(t) = f(X(t), t)dt + g(X(t), t)dW(t) + h(X(t), t)dP(t),$$

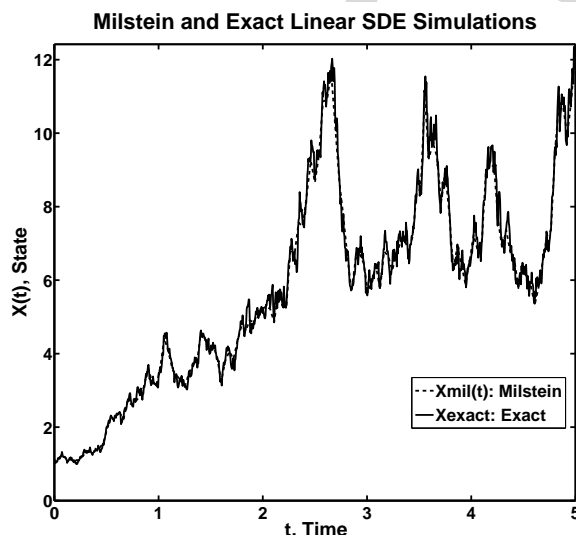


Figure 10.4. Comparison of coarse Milstein and fine exact paths, simulated using MATLAB with $N_t = 1024$ fine sample points for the exact path (10.15) and $N_t/8 = 128$ coarse points for the Milstein path (10.23), initial time $t_0 = 0$, final time $t_f = 5$ and initial state $x_0 = 1.0$ as in Fig. 10.2. Time-dependent parameter values are $\mu(t) = 0.5/(1 + 0.5t)^2$ and $\sigma(t) = 0.5$.

in its simplest form using the forward integral approximation of Itô for fixed Δt is

$$X_{k+1} = X_k + F_k \Delta t + G_k \Delta t + H_k \Delta P_k, \quad (10.24)$$

where $(F_k, G_k, H_k) = (f(X_k, t_k), g(X_k, t_k), h(X_k, t_k))$, $\Delta t = t_{k+1} - t_k$, $\Delta W_k = W_{k+1} - W_k$, and $\Delta P_k = P_{k+1} - P_k$, for $k = 0 : N_t$. Maghsoodi [187] and also Maghsoodi and Harris [188] derived most of the theory behind this method and derived numerous Milstein-like higher order approximations, so sometimes (10.24) is called the **Euler-Maghsoodi method**.

Linear Jump-Diffusion Euler Method Convergence

Following the stochastic diffusion Euler analysis for the linear, constant coefficient case,

$$dX(t) = X(t)(\mu_0 dt + \sigma_0 dW(t) + \nu_0 dP(t)),$$

the discrete Euler is written

$$X_k = B_{k-1} \cdot X_{k-1}; \quad B_k \equiv (1 + (\mu_0 + \lambda_0 \nu_0) \Delta t + \sigma_0 \Delta W_k + \nu_0 (\Delta P_k - \lambda_0 \Delta t)),$$

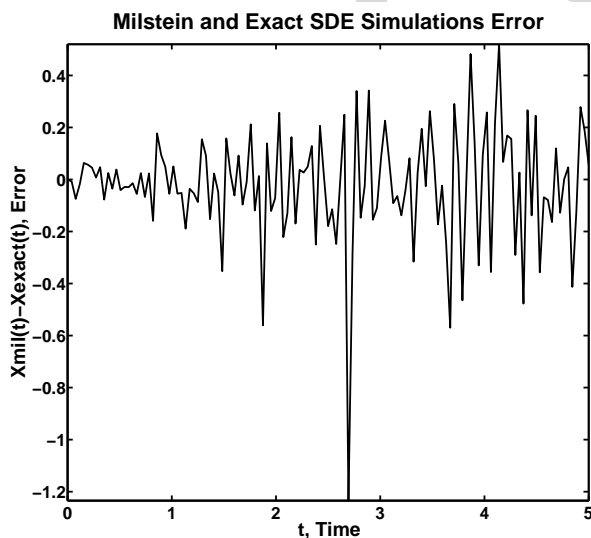


Figure 10.5. Error in coarse Milstein and fine exact paths using the coarse discrete time points. The simulations use MATLAB with the same values and time-dependent coefficients as in Fig. 10.2. The Milstein maximal-absolute error for this example is 1.2, while for $N_t = 4096$ the maximal error is better at 0.95.

where the discrete Poisson process is written in mean-zero (i.e., martingale) independent increment form for convenience, so that

$$X_k = x_0 \prod_{\ell=0}^{k-1} B_\ell,$$

and by independent increments as well as independent jump-diffusion processes,

$$E[X_k | X_0 = x_0] = x_0 \prod_{\ell=0}^{k-1} E[B_\ell] = x_0 \prod_{\ell=0}^{k-1} (1 + (\mu_0 + \lambda_0 \nu_0) \Delta t) = x_0 (1 + (\mu_0 + \lambda_0 \nu_0) \Delta t)^k.$$

From the exact solution (10.11) using the expectation in (4.81), the final mean square at $t_f = N_t \Delta t$ is

$$E[X(t_f) | X(0) = x_0] = x_0 e^{(\mu_0 + \lambda_0 \nu_0) t_f}.$$

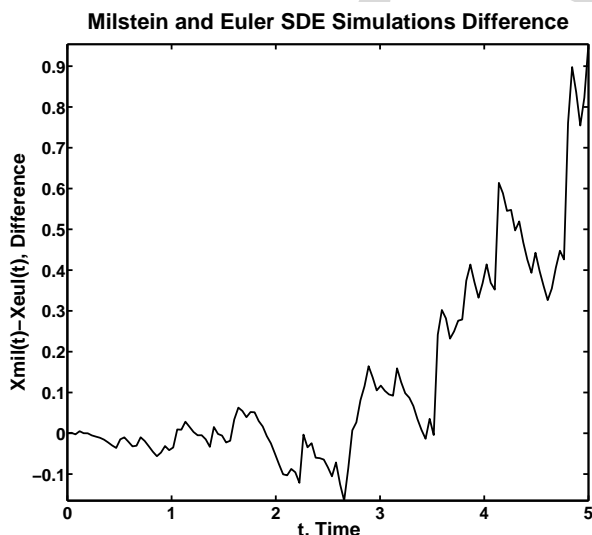


Figure 10.6. *Difference in coarse Milstein and Euler paths using the coarse discrete time points. The simulations use MATLAB with the same values and time-dependent coefficients as in Fig. 10.2. The Milstein-Euler maximal-absolute difference for this example is 0.19, while for $N_t = 4096$ the maximal difference is comparable at 0.24.*

Next, computing the convergence criteria in the weak sense asymptotically,

$$\begin{aligned}
 & |E[X_{N_t} | X_0 = x_0] - E[X(t_f) | X_0 = x_0]| \\
 &= |x_0| \left| (1 + (\mu_0 + \lambda_0 \nu_0) \Delta t)^{N_t} - e^{(\mu_0 + \lambda_0 \nu_0) N_t \Delta t} \right| \\
 &= |x_0| e^{(\mu_0 + \lambda_0 \nu_0) t_f} \left| e^{N_t \ln(\mu_0 + \lambda_0 \nu_0) \Delta t} - e^{(\mu_0 + \lambda_0 \nu_0) N_t \Delta t} - 1 \right| \\
 &\sim |x_0| e^{(\mu_0 + \lambda_0 \nu_0) t_f} \left| e^{-0.5 N_t (\mu_0 + \lambda_0 \nu_0)^2 \Delta t^2} - 1 \right| \\
 &\sim C_w \Delta t,
 \end{aligned}$$

where $C_w = |x_0| (\mu_0 + \lambda_0 \nu_0)^2 t_f \exp(\mu_0 + \lambda_0 \nu_0) t_f$ and the convergence in the weak sense is order one in Δt with $\gamma_w = 1$.

The distributed jump case is somewhat similar, except the marks introduce much more complications. Let the linear distributed jump-diffusion SDE have constant coefficients except that the relative jump amplitude depends on the random mark Q and the symbolic product $\nu(Q)dP(t)$ is replaced by the proper jump sum.

So

$$dX(t) = X(t) \left(\mu_0 dt + \sigma_0 dW(t) + \sum_{\ell=1}^{dP(t)} \nu(Q_\ell) \right),$$

and the discrete Euler processes are written in zero mean form,

$$\begin{aligned} X_k &= \beta_{k-1} \cdot X_{k-1}; \\ \beta_k &\equiv 1 + (\mu_0 + \lambda_0 \mathbb{E}[\nu(Q)])\Delta t + \sigma_0 \Delta W_k + \mathbb{E}[\nu(Q)](\Delta P_k - \lambda_0 \Delta t) \\ &\quad + \sum_{\ell=1}^{\Delta P_k} (\nu(Q_\ell) - \mathbb{E}[\nu(Q)]) \end{aligned}$$

The exact solution at node t_k upon using the stochastic chain rule and integrating is

$$X(t_k) = x_0 \exp \left((\mu_0 - \sigma_0^2/2)t_k + \sigma_0 W_k + \sum_{\ell=1}^{P_k} Q_\ell \right),$$

where we have again set $Q \equiv \ln(1 + \nu(Q))$ or $\nu(Q) = \exp(Q) - 1$ for convenience of setting the mark distribution appropriate for the log-process. Using the iterated expectations technique to nest the Poisson and jump mark expectations, the expectations are

$$\begin{aligned} \mathbb{E}[X_k | X_0 = x_0] &= x_0 \mathbb{E} \left[\prod_{j=0}^{k-1} \beta_j \right] = x_0 \prod_{j=0}^{k-1} \mathbb{E}[\beta_j] \\ &= x_0 \prod_{j=0}^{k-1} (1 + (\mu_0 + \lambda_0 (\mathbb{E}[\exp(Q)] - 1))\Delta t) \\ &= x_0 (1 + (\mu_0 + \lambda_0 (\mathbb{E}[\exp(Q)] - 1))\Delta t)^k \end{aligned}$$

for the approximation and

$$\begin{aligned} \mathbb{E}[X(t_k) | X(0) = x_0] &= x_0 \exp((\mu_0 - \sigma_0^2/2)t_k) \mathbb{E}_{W_k}[\exp(\sigma_0 W_k)] \\ &\quad \cdot \mathbb{E}_{P_k} \left[\mathbb{E}_Q \left[\exp \left(\sum_{\ell=1}^{P_k} Q_\ell \right) \middle| P_k \right] \right] \\ &= x_0 \exp((\mu_0 + \lambda_0 (\mathbb{E}[\exp(Q)] - 1))t_k), \end{aligned}$$

for the exact. Again, asymptotic results are derived for weak absolute mean error as $\Delta t \rightarrow 0^+$ for fixed t_k ,

$$\begin{aligned} &|\mathbb{E}[X_k | X_0 = x_0] - \mathbb{E}[X(t_k) | X(0) = x_0]| \\ &= |x_0| \cdot \left| e^{-k \ln(1 + (\mu_0 + \lambda_0 (\mathbb{E}[\exp(Q)] - 1))\Delta t)} - e^{(\mu_0 + \lambda_0 (\mathbb{E}[\exp(Q)] - 1))t_k} \right| \\ &\sim |x_0| e^{(\mu_0 + \lambda_0 (\mathbb{E}[\exp(Q)] - 1))t_k} \left| e^{-0.5k(\mu_0 + \lambda_0 (\mathbb{E}[\exp(Q)] - 1))^2 \Delta t^2} - 1 \right| \sim C_w \Delta t, \end{aligned}$$

where

$$C_w = 0.5|x_0|t_k(\mu_0 + \lambda_0(\mathbb{E}[\exp(Q)] - 1))^2 e^{(\mu_0 + \lambda_0(\mathbb{E}[\exp(Q)] - 1))t_k}.$$

Again the weak convergence rate is linear with $\gamma_w = 1$.

Maghsoodi [187] shows that the strong mean square error convergence rate (10.6) is $O(\Delta t)$ for the jump-diffusion Euler method for nonlinear coefficients subject to linear Lipschitz bounds, which translates into a strong root mean square rate of

$$O(\sqrt{\Delta t}).$$

A similar result was shown by D. Higham and Kloeden [142] for the implicit jump-diffusion or **stochastic theta method (STM)** with the mean square error based upon piece-wise-constant interpolation functions rather than the discrete approximate and exact values themselves ($\theta = 0$ is the explicit, stochastic Euler method, while the theta method is implicit for $0 < \theta \leq 1$). For the jump-diffusion problem, the theta method only applies to the drift term in (10.24),

$$X_{k+1} = X_k + ((1 - \theta)F_k + \theta F_{k+1}) + G_k \Delta W_k + H_k \Delta P_k, \quad (10.25)$$

in order to preserve stochastic consistency with jump-diffusion with the jump-diffusion conditional infinitesimal moments (9.46,9.65), by avoiding implicit, backward steps in the diffusion and jump terms. The technical details of STM are beyond the scope of this chapter.

Euler Mean Square Linear Asymptotic Stability for Jump-Diffusions

For the mean square asymptotic stability of the jump-diffusion Euler method, the procedure leading up to the corresponding diffusion critical condition (10.18) is used. Starting with the jump-diffusion linear system recursive form,

$$X_k = B_{k-1} \cdot X_{k-1},$$

then the mean square is

$$\begin{aligned} E[X_k^2 | X_0 = x_0] &= x_0^2 E \left[\left(\prod_{\ell=0}^{k-1} B_\ell \right)^2 \right] = x_0^2 \prod_{\ell=0}^{k-1} E [B_\ell^2] \\ &= x_0^2 \prod_{\ell=0}^{k-1} ((1 + (\mu_0 + \lambda_0 \nu_0) \Delta t)^2 + (\sigma_0^2 + \lambda_0 \nu_0^2) \Delta t) \\ &= x_0^2 ((1 + (\mu_0 + \lambda_0 \nu_0) \Delta t)^2 + (\sigma_0^2 + \lambda_0 \nu_0^2) \Delta t)^k. \end{aligned}$$

Again, as $k \rightarrow \infty$, the base of the power k must be less than one since the base is non-negative, so the mean square asymptotic stability criterion for the linear, constant coefficient, jump-diffusion Euler approximation is

$$2(\mu_0 + \lambda_0 \nu_0) + \sigma_0^2 + \lambda_0 \nu_0^2 + (\mu_0 + \lambda_0 \nu_0)^2 \Delta t < 0, \quad (10.26)$$

which means that $\mu_0 + \lambda_0 \nu_0$ needs to be sufficiently negative (note that $\lambda_0 > 0$ if the jump process is to be genuine),

$$\mu_0 + \lambda_0 \nu_0 < -0.5(\sigma_0^2 + \lambda_0 \nu_0^2 + (\mu_0 + \lambda_0 \nu_0)^2 \Delta t)$$

and when interpreted in terms of the first and second relative conditional infinitesimal moments is

$$E[\Delta X_k / X_k | X_k \neq 0] < -0.5 E[(\Delta X_k / X_k)^2 | X_k \neq 0].$$

If we restrict our attention to when the exact solution is mean square stable, i.e., $2(\mu_0 + \lambda_0\nu_0) + \sigma_0^2 + \lambda_0\nu_0^2 < 0$ from (10.12), then (10.26) can be used to construct a constraint on the discrete time step,

$$\Delta t < 2 \left| \mu_0 + \lambda_0\nu_0 + 0.5(\sigma_0^2 + \lambda_0\nu_0^2) \right| / (\mu_0 + \lambda_0\nu_0)^2 .$$

10.1.5 Jump-Diffusion Euler Simulation Procedures

A simple numerical procedure is given in Subsection 4.3.3 on page 189 for the linear system with discrete jump of size ν_0 ,

$$dX(t) = X(t)(\mu_0 dt + \sigma_0 dW(t) + \nu_0 dP(t)),$$

using MATLAB™'s normal random number generator `randn` and a small time-step zero-one Poisson-Bernoulli jump law using the acceptance-rejection method. Since this zero-one jump law uses the Δt -order asymptotic precision definition of the Poisson process there is a restriction that $\lambda\Delta t < 1$ so that the one-jump probability is positive. See Program A.14 in the Appendix A for the MATLAB code used.

However, this $\lambda\Delta t < 1$ condition can be easily rectified by just renormalizing Poisson distribution, $p_k(\lambda\Delta t) = \exp(-\lambda\Delta t)(\lambda\Delta t)^k/k!$, for a finite number of jumps $k \leq j$ without expanding the $\exp(-\lambda\Delta t)$ factor in the numerator, so

$$p_k^{(j)}(\lambda\Delta t) \equiv \frac{(\lambda\Delta t)^k/k!}{\sum_{\ell=0}^j (\lambda\Delta t)^\ell/\ell!} \tag{10.27}$$

is valid as long as $\lambda\Delta t > 0$ and conserves probability. This is the same as if the original normalization $\exp(+\lambda\Delta t)$ were expanded by $\lambda\Delta t$ in the denominator to $j + 1$ terms and the result called a Padé approximation or rational function. Form (10.27) also exactly preserves the ordering of the Poisson jump probabilities, i.e.,

$$\frac{p_{k+1}^{(j)}(\lambda\Delta t)}{p_k^{(j)}(\lambda\Delta t)} = \frac{\lambda\Delta t}{k+1}$$

as long as $k = 0:j - 1$. This form can be used with the acceptance-rejection method as long as the unit interval $[0, 1]$ is partitioned into $j + 1$ subintervals of length $p_k^{(j)}(\lambda\Delta t)$ for $k = 0:j$ such that a random number generator like MATLAB™'s `rand` is used and if the number generated lands in the subinterval corresponding to $p_k^{(j)}(\lambda\Delta t)$, then the realized number of jumps is k . Computer experiment experience indicates that it is best not to put the small subintervals adjacent to the endpoints of $[0, 1]$ due to the open interval $(0, 1)$ bias of computer random generators.

Distributed Jump Linear Jump-Diffusion Euler Method

In Fig. 5.1 on page 233, the simulations for uniformly distributed marks Q on $(a, b) = (-2, +1)$ and time-dependent linear or geometric jump-diffusion SDE,

$$dX(t) = X(t)(\mu_d(t)dt + \sigma(t)dW(t) + \nu(Q)dP(t)).$$

However, it is more convenient to work with the exponent of the exact solution derived by the stochastic chain rule to obtain the SDE,

$$dY(t) = d\ln(X(t)) = (\mu(t) - \sigma^2(t)/2)dt + \sigma(t)dW(t) + QdP(t),$$

where the mark has been selected as $Q \equiv \ln(1 + \nu(Q))$ for convenience (this would seem to preclude time-dependence in the jump amplitude $\nu(Q)$, but time can be included in the mark range $[a, b]$ or the mark density $\phi_Q(q)$). The MATLAB code A.15 is a modification of the linear jump-diffusion SDE simulator code A.14 illustrated in Fig. 4.3 for constant coefficients and discrete mark-independent jumps. The state exponent $Y(t)$ is simulated as

$$YS(k+1) = YS(k) + (\mu(k) - \sigma^2(k)/2) * DT + \sigma(k) * DW(k) + Q(k) * DP(k),$$

with $t(k+1) = t_0 + k * DT$ for $k = 0 : NI - 1$ with $NI = 1,000$, $t_0 = 0$ and $0 \leq t(k) \leq 2$. The incremental Poisson jump term $DP(k) = P(t(k) + DT) - P(t(k))$ is simulated by the MATLAB™ uniform random number generator `rand` on $(0, 1)$ using the acceptance-rejection technique [226, 96] (see also Subsect. 10.2.3 on p. 388) to implement the zero-one jump law to obtain the probability of $\lambda(i)Dt$ that a jump is accepted, else a jump is rejected. The same random state (seed), but a different set of generated random samples, is used to obtain the simulations of the uniformly distributed Q on (a, b) . i.e., $Q = a + (b - a) * \text{rand}(1, NI)$, that is used only if there is a jump event. Finally, the state itself is computed by a simple exponential inversion of the log-process as

$$X(k+1) = x_0 * \exp(Y(k+1)),$$

and should be highly accurate for sufficiently small DT since this procedure based upon the exact exponent is the same procedure that is used for producing the exact simulation, say by Maghsoodi [187]. Clearly, if one has a linear SDE with constant parameter coefficients for an application, then the best strategy is to simulate the exact solution since it is available. However, if the object is just to use the linear SDE for testing a method on more general nonlinear SDEs, related perhaps by similar Lipschitz linear bounds, then simulation of the original linear SDE for $X(t)$ is recommended.

Many deterministic numerical methods are difficult to translate directly into numerical methods of diffusions or jump-diffusions due to the non-smooth or discontinuous nature of the diffusion process $W(t)$ or the jump process $P(t)$, respectively. Hence, implicit methods or multistep methods (many of these are designed to reduce or eliminate the implicitness of implicit methods) have to be modified to separate the treatment of the deterministic term ($f(x, t)\Delta t$) from that of the diffusion term ($g(x, t)\Delta W(t)$) or that of the jump term ($h(x, t)\Delta P(t)$ or $h(x, t, q)\Delta P(t)$). It is necessary to preserve stochastic approximation consistency with respect to the jump-diffusion conditional infinitesimal moments (9.46, 9.65).

Stochastic Split-Step Backward Euler Method

One such method is a stochastic modification of the **deterministic backward Euler (DBE)** method ($X_{k+1} = X_k + f(X_{k+1}, t_{k+1})\Delta t$) which for the jump-diffusion

problem is split into two stages by Cyganowski and Kloeden [65] and more recently by D. Higham and Kloeden [141], the first stage is just a backward Euler step, $X_{k+1}^{(\text{dbe})}$, only improved by the deterministic drift and a second stage that adds diffusion and jump term improvement,

$$\begin{aligned} X_{k+1}^{(\text{dbe})} &= X_k + f\left(X_{k+1}^{(\text{dbe})}, t_{k+1}\right) \Delta t, \\ X_{k+1}^{(\text{ssbe})} &= X_k + g\left(X_{k+1}^{(\text{dbe})}, t_{k+1}\right) \Delta W_k + h\left(X_{k+1}^{(\text{dbe})}, t_{k+1}\right) \Delta P_k, \end{aligned} \tag{10.28}$$

which they call a **split-step backward Euler (SSBE)**. The first stage is implicit in $X_{k+1}^{(\text{dbe})}$, so enhances the stability and convergence, for which some results are given in [65, 141], but no rates of convergence. The coefficients in [141] are autonomous, but time-dependence is added here for generality. An improved refinement is also included in [65, 141] and that is using the compensated or zero-mean Poisson $\Delta P_k - \lambda_k \Delta t$, a martingale, to obtain the **compensated split-step backward Euler (CSSBE)**,

$$\begin{aligned} X_{k+1}^{(\text{dbe})} &= X_k + \left(f\left(X_{k+1}^{(\text{dbe})}, t_{k+1}\right) + \lambda_k h\left(X_{k+1}^{(\text{dbe})}, t_{k+1}\right) \right) \Delta t, \\ X_{k+1}^{(\text{cssbe})} &= X_k + g\left(X_{k+1}^{(\text{dbe})}, t_{k+1}\right) \Delta W_k + h\left(X_{k+1}^{(\text{dbe})}, t_{k+1}\right) (\Delta P_k - \lambda_k \Delta t), \end{aligned} \tag{10.29}$$

which provides better improvement in the first, deterministic backward Euler, stage. No computational validation is given in [65, 141]. In [143], D. Higham, Mao and Stuart show $O(\Delta t)$ mean square error convergence rates for SSBE on nonlinear diffusion SDEs with coefficient functions satisfying linear Lipschitz conditions.

Maghsoodi [187] also extended the Milstein algorithm for diffusions to jump-diffusions by expanding the jump coefficient $h(x, t)$ like the diffusion coefficient $g(x, t)$ stochastic Taylor expansion. However, the new and numerous jump terms are much more complicated than the diffusion version and Cyganowski, Kloeden and Ombach [66] demonstrate by computer experiment that this method works well for discrete jump problems but not for distributed (mark-dependent) jump problems, so the extension will not be discussed here.

Related convergence and stability results for discrete jump-diffusions are given by D. Higham and Kloeden in [141] for the stochastic theta method as previously mentioned in association with the STM algorithm (10.25).

Jump-Adapted Euler Method

Thus far, methods using constant time-steps $\Delta t = t_{k+1} - t_k$ or a fixed set of variable time-steps $\Delta t_{k-1} t_{k+1} - t_k$ have been discussed, such that the number of jumps of ΔP_k in $[t_k, t_{k+1}]$ have been enumerated and corresponding jump marks, if present, simulated. An alternate numerical approach, suggested by Maghsoodi [187], is to interlace the set of Poisson random jump times, T_j for $j = 1 : N_J$ such that $T_{N_J} \leq t_f$, with a fixed set t_ℓ for $\ell = 0 : N_t$ to define a **jump-adaptive (JA) method** grid augmented by initial and final times, such that $\tau_0 \equiv 0 < \tau_k < \tau_{k+1} = \tau_k + \Delta \tau_k < \tau_{N^{(\text{ja})}} = \tau_f$ with subintervals of length $\Delta \tau_k = \tau_{k+1} - \tau_k$ for $k = 0 : N^{(\text{ja})} - 1$. One

restriction is that the mesh measure satisfies $\max_{0 \leq k \leq N^{(ja)}-1} (\Delta\tau_k) \leq \overline{\Delta\tau}$ where $\overline{\Delta\tau} \simeq \Delta t$ plus some leeway.

It is well known that it is the Poisson subintervals or the time to the next jump $\Delta T_j = T_{j+1} - T_j$ are independent and identically, exponentially distributed (1.24) with rate λ (unfortunately, the literature on jump-adapted method confuses the IID properties of the inter-jump times and the interdependence of the jump-times themselves). The exponentially distributed Poisson jump-time generation is given on page 92 using the logarithmic transformation of a uniform random number generator and a vector version is

```
% log-uniform exponential density:
DT=-log(rand(1,NJ))/lambda;
T=cumsum(DT);
```

(10.30)

where `rand(1,NJ)` is MATLAB™'s $1 \times NJ$ vector random generator and `cumsum` is the cumulative sum function, assuming that the total number of jumps is known.

Let the discrete state be denoted as $X_k^{(ja)} \simeq X(\tau_k)$ corresponding to adapted-jump-time τ_k , so the jump-diffusion Euler method for discrete jumps is

$$X_{k+1}^{(ja)} = X_k^{(ja)} + F_k^{(ja)} \Delta\tau_k + G_k^{(ja)} \Delta W_k^{(ja)} + H_k^{(ja)} \Delta P_k^{(ja)}, \quad (10.31)$$

where $\Delta W_k^{(ja)} = W(\tau_{k+1}) - W(\tau_k)$, $\Delta P_k^{(ja)} = P(\tau_{k+1}) - P(\tau_k)$, $F_k^{(ja)} = f(X_k^{(ja)}, \tau_k)$; similarly for $G_k^{(ja)}$ and $H_k^{(ja)}$. Note that if τ_{k+1} coincides with a jump time T_j for some j then $\Delta P_k^{(ja)} = 1$, otherwise $\Delta P_k^{(ja)} = 0$. However, as Maghsoodi [187] warns, when analyzing something like convergence in the mean then it must be recognized that if $\tau_{k+1} = T_j$ then $\Delta W_k^{(ja)} = W(T_j) - W(\tau_k)$ is not statistically independent of $\Delta P_k^{(ja)} = P(T_j) - P(\tau_k)$, if expectations are to be calculated. A sample fragment of the code to compute $\Delta\tau_k$, $\Delta W_k^{(ja)}$ and $\Delta P_k^{(ja)}$ could be as given in Fig. 10.7. This code fragment can be patched together with the given application SDE and chosen base numerical algorithm such as the jump-diffusion Euler or split-step backward Euler, for instance.

10.2 Monte Carlo Methods

The Monte Carlo method started as a statistical sampling procedure at Los Alamos National Laboratory in 1946 from an idea of Ulam in analogy considering the probability of winning the card game of solitaire, from the idea of von Neumann for the programming neutron transport on a newly emerging electronic computer and Metropolis for computer implementation [78, 202, 204]. Without the emergence of electronic computers very few people would attempt to use large scale statistical sampling to solve large problems. One exception was the famous physicist Fermi who could calculate very fast using a mechanical calculator and had time to do big calculations because he often could not sleep, so in fact he was using a smaller scale version of the Monte Carlo method fifteen years before it had a name (for other earlier examples see Hammersley and Handscomb [104] for instance). The method

```
function jumpadapt
% Jump adaptive (JA) code fragment:
% merged regular and jump times
Nt=10; lambda=9; t0=0; tf=1; Dt = (tf-t0)/Nt;
t = Dt*[0:Nt]; % Regular grid
DT = -log(rand)/lambda; S=DT; j=0;
while S < tf % Get jump time grid, T(NJ)<tf
    j=j+1;
    NJ=j;
    T(j)=S; DTJ(j)=DT;
    DT = -log(rand)/lambda; % Exponential density
    S=S+DT;
end
[tau,ktau]=sort([t T]); % Concatenate and sort times
Dtau=tau(2:Nja)-tau(1:Nja-1); % Concatenate and sort times
randn('state',10);
RN=randn(1,Nja-1); % Std. normal density
DP=zeros(1,Nja-1);
for k=2:Nja
    DW(k-1)=sqrt(Dtau(k-1))*RN(k-1); % Get DW
    if ktau(k)>Nt+1
        DP(k-1)=1; % Get final DP
    end
end
end
```

Figure 10.7. Code: Jump-adapted code fragment.

was named for an uncle of Ulam's who had a obsession about going to gamble at Monaco, the gambling capital of Europe. In a 1949 paper of Metropolis and Ulam [204] entitled **The Monte Carlo Method**, they spelled out the basic ideas in a more or less essay form: the potential applications, the statistical approach, the independent random sampling, the frequency distributions, the law of large numbers for convergence and the asymptotic theorems for probable errors. Although von Neumann is not an author on this paper, it contains his ideas on techniques of random number generation and a hint of his acceptance and rejection method to handle general shaped domains by rejecting those samples which land outside of the domain.

A more major idea of von Neumann was the logical structure of most modern programmable computers, the **von Neumann computer**. The newly emerging electronic computer mentioned was the ENIAC, a very primitive, nonprogrammable and decidedly non-von computer as non-von Neumann computers are called. Not too long afterward, there was a parallel effort at both Princeton with von Neumann and at Los Alamos with Metropolis to build a von Neumann computer, but Metropolis was able to get the Los Alamos computer named MANIAC working first.

As it is with most computer advances, faster computers do not save the user time because the user will bring a bigger problem that will take about the same amount of time as the previous problem. The user who thought of the larger Monte Carlo problem to bring to the MANIAC was the physicist Teller and the problem was calculating the equation of state of an ideal rigid sphere gas. However, the major contribution of the resulting 1953 paper by Metropolis, the Rosenbluths and the Tellers [203] was the use of **weighted sampling**, now called **importance sampling**, by using the exponential distribution of the energy change as the weight. This version of the Monte Carlo method is called the **Metropolis algorithm** [70] and was selected a one of **ten top algorithms of the century** [68, 23]. This may be confusing, because the basic Monte Carlo algorithm is sometimes called the **Metropolis algorithm** too. The 1953 paper [203] contains significantly more detail than the 1949 paper [204], in both cases Metropolis is the lead author and some would say the lead Monte Carlo computation teacher. The title of the 1953 paper is **Equation of State Calculation by Fast Computing Machines** and the quoted cycle time of the MANIAC translated to 5.6 mHz, i.e., 5.6×10^{-3} cycles per second, which would be extremely slow compared to today's 2GHz to 4GHz PCs or 2.0×10^9 to 4.0×10^9 cycles per second, not fast at all.

For general references on the Monte Carlo method, see the classic monograph of Hammersley and Handscomb [104] or the more recent book of Kalos and Whitlock, 1986 [154]. Much of the more recent advances have come from applications of the Monte Carlo method to finance, so for general references on Monte Carlo with application to finance see Glasserman [96] and Jäckel [147]. For the pioneering and award winning paper on application of the Monte Carlo method to financial options see Boyle [38] or for a two decade update see Boyle, Broadie and Glasserman [39].

10.2.1 Basic Monte Carlo Simulations

The benefits of Monte Carlo are only realized in high dimensions and for functionals of stochastic processes with simulation complexities beyond direct simulations of SDEs as covered in the previous section or for deterministic problems such as physical diffusions whose solutions can be simulated by Monte Carlo. Many problems can be transformed into an integral form or integral functional such as

$$I[F] = \int_{\mathcal{V}} F(\mathbf{x}) d\mathbf{x}, \quad (10.32)$$

where $\mathbf{x} = [x_i]_{n_x \times 1}$ is a n_x -dimensional vector on volume \mathcal{V} and $F(\mathbf{x})$ is a bounded, integrable scalar-valued function on \mathcal{V} . For instance, if \mathcal{V} is finite then $I[F]$ could be interpreted in terms of the expectation

$$I[F] = V \cdot E_{\mathbf{X}}[F(\mathbf{X})]$$

of F with respect to uniform variates \mathbf{X} such that $V \equiv \int_{\mathcal{V}} d\mathbf{x} < \infty$ with uniform density $\phi_{\mathbf{X}}(\mathbf{x}) = 1/V$ on domain \mathcal{V} .

In general (10.32) can be interpreted to include nonuniform distributions by scaling F by a suitable density $\phi_{\mathbf{X}}(\mathbf{x})$ for variates \mathbf{X} on \mathcal{V} so that

$$F(\mathbf{x}) = f(\mathbf{x})\phi_{\mathbf{X}}(\mathbf{x}),$$

$$\int_{\mathcal{V}} \phi_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = 1$$

and

$$I[F] = E_{\mathbf{X}}[f(\mathbf{X})] = \int_{\mathcal{V}} f(\mathbf{x}) \phi_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}. \tag{10.33}$$

The general rule for the selection of the density $\phi_{\mathbf{X}}(\mathbf{x})$ is that it capture important characteristics, such as variability, of the integrand $F(\mathbf{x})$ on domain \mathcal{V} such that the function $f(\mathbf{X})$ is bounded and not very variable. The density $\phi_{\mathbf{X}}(\mathbf{x})$ should be known and the generation of its variates should be computable with reasonable effort. In the uniform case, $\phi_{\mathbf{X}}(\mathbf{x}) = 1/V$ and $f(\mathbf{X}) = V \cdot F(\mathbf{X})$.

Example 10.3. Risk-Neutral European Call Option Pricing:

An example of a complex functional is the risk-neutral European call option pricing model of Zhu and Hanson [285] using a jump-diffusion SDE with log-uniformly distributed jump-amplitude marks,

$$C(S_0, t_f) = E_{\tilde{P}(t_f)} \left[C^{(BS)} \left(S_0 e^{\tilde{P}(t_f) - \lambda \bar{J} t_f}, t_f \right) \right] \tag{10.34}$$

where

$$\tilde{P}(t_f) = \sum_{i=1}^{P(t_f)} Q_i \tag{10.35}$$

is the compound Poisson jump process cumulative sum at the strike time t_f with uniformly distributed IID random marks Q_i on $[a, b]$, mean jump-amplitude

$$\bar{J} \equiv E_Q[J(Q)] \equiv E[\exp(Q) - 1] = (\exp(b) - \exp(a)) / (b - a) - 1 \tag{10.36}$$

and Black-Scholes call option price

$$C^{(BS)}(s, t_f) \equiv s\Phi(d_1(s)) - Ke^{-rt_f}\Phi(d_2(s)), \tag{10.37}$$

with strike price K , interest rate r , diffusive volatility σ , standardized normal distribution function $\Phi(x)$ and Black-Scholes argument functions $d_1(s) \equiv (\ln(s/K) + (r + \sigma^2/2)t_f) / (\sigma\sqrt{t_f})$ and $d_2(s) \equiv d_1(s) - \sigma\sqrt{t_f}$. Refer to [285] for the transformations used to achieve this form, which one would not attempt to evaluate directly but would try to estimate the call option price.

Returning to the general integral functional problem (10.33), an estimate \hat{I}_n of the value of the integral $I[F] = E_{\mathbf{X}}[f(\mathbf{X})]$ is the **sample mean** s_n of n independent, identically distributed sample points \mathbf{X} distributed on \mathcal{V} corresponding to the density $\phi_{\mathbf{X}}(\mathbf{x})$,

$$\hat{I}_n = s_n, \tag{10.38}$$

where the sample mean s_n or **Monte Carlo Estimator** $\hat{\mu}_n = s_n$ is

$$\hat{\mu}_n \equiv s_n = \frac{1}{n} \sum_{i=1}^n f(\mathbf{X}_i) \equiv \frac{1}{n} \sum_{i=1}^n f_i, \quad (10.39)$$

the estimate of the mean of f with respect to $\phi_{\mathbf{X}}(\mathbf{x})$. Obviously, the function $f(\mathbf{x})$ must be bounded for the sample mean to exist. The true mean of f is

$$\mu_f = E_{\mathbf{X}}[f(\mathbf{X})] = \int_{\mathcal{V}} f(\mathbf{x})\phi_{\mathbf{X}}(\mathbf{x})d\mathbf{x}.$$

Then, the estimate $\hat{\mu}_n$ is an **unbiased estimate**, since the **bias** of the estimator from the true mean is zero, i.e.,

$$\beta_{\hat{\mu}_n} \equiv E_{\mathbf{X}}[\hat{\mu}_n - \mu_f] = \frac{1}{n} \sum_{i=1}^n E_{\mathbf{X}}[f(\mathbf{X}_i)] - \mu_f = E_{\mathbf{X}}[f(\mathbf{X})] - \mu_f = 0, \quad (10.40)$$

using the IID property of the sample points. Further, by the **strong law of large numbers (SLLN)** (0.115),

$$\hat{\mu}_n \longrightarrow \mu_f \text{ with probability one as } n \rightarrow +\infty.$$

The true variance of f is

$$\sigma_f^2 = \text{Var}_{\mathbf{X}}[f(\mathbf{X})] = \int_{\mathcal{V}} (f(\mathbf{x}) - \mu_f)^2 \phi_{\mathbf{X}}(\mathbf{x})d\mathbf{x}$$

and so the unbiased estimate of the sample variance from (0.109) is

$$\hat{\sigma}_n^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \hat{\mu}_n)^2. \quad (10.41)$$

Example 10.4. Choice of Monte Carlo Sampling Distribution:

A rule of thumb is that, while many other distributions may work in generating Monte Carlo estimations, the better density captures more variability of $F(\mathbf{x})$ along with the domain \mathcal{V} and leaves a less variable $f(\mathbf{x})$ to simulate. Thus, the better choice will be the the better Monte Carlo results.

It is general numerical practice to choose an integrand weight function that captures most of the variability and can easily be integrated exactly so that the remaining integrand factor can be discretely and well approximated. For example the truncated normal distribution,

$$I = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-x^2/2} dx, \quad (10.42)$$

can be Monte Carlo estimated using a uniform (u) density $\phi^{(u)}(x) = 1/(b-a)$ on $[a, b]$ with sampled function

$$f^{(u)}(x) = (b-a) \exp(-x^2/2) / \sqrt{2\pi}$$

or a normal (n) density $\phi^{(n)}(x) = \exp(-x^2/2)/\sqrt{2\pi}$ on $[-\infty, +\infty]$ and

$$f^{(n)}(x) = \mathbf{1}_{x \in [a,b]} = \{1, x \in [a, b]; 0, x \notin [a, b]\},$$

is an indicator function. The exact mean is invariant with respect to the density,

$$\mu_f^{(n)} = I = \Phi_n(a, b; 0, 1) = \mu_f^{(u)},$$

where $\Phi_n(x, y; 0, 1)$ is the usual standard normal distribution in this book on $[x, y]$. However, it is obvious that the exact variance assuming a normal density factor will be much smaller than the exact variance assuming a uniform density factor and a highly variable $f(x)$, if a and b are not small. In fact, $(\sigma_f^{(n)})^2 = I - I^2$ for the normal case since $\mathbf{1}_{x \in [a,b]}^2 = \mathbf{1}_{x \in [a,b]}$ and $(\sigma_f^{(u)})^2 = (b-a)\Phi_n(\sqrt{2}a, \sqrt{2}b; 0, 1)/(2\sqrt{\pi}) - I^2$ for the uniform by transformation $E^{(u)}[(f^{(u)})^2(x)]$ to the standard normal distribution. As $a \rightarrow -\infty$ and $b \rightarrow +\infty$, the standard normal distributions $\Phi_n \rightarrow 1$ in uniform as well as in normal cases and the difference has the unbounded asymptotic approximation,

$$(\sigma_f^{(u)})^2 - (\sigma_f^{(n)})^2 \sim \frac{b-a}{2\sqrt{\pi}} - 1,$$

demonstrating in this extreme case that the choice of the sampling density $\phi_{\mathbf{X}}(\mathbf{x})$ can make a big difference in the variance σ_f^2 . A companion computational demonstration code A.18 for this problem when $[a, b] = [-R, R]$ is given on p. 552 of Appendix A. Of course, one would not use the uniform distribution on an infinite domain.

Convergence of Scaled Monte Carlo Estimate Distribution to a Normal Distribution

By the central limit theorem (0.116) the sample mean converges in distribution to a normal distribution,

$$\text{Prob} \left[\frac{\hat{\mu}_n - \mu_f}{\sigma_f/\sqrt{n}} \leq \xi \right] \rightarrow \Phi_n(\xi; 0, 1) \text{ as } n \rightarrow +\infty, \quad (10.43)$$

or alternately we say $(\hat{\mu}_n - \mu_f)/(\sigma_f/\sqrt{n}) \xrightarrow{\text{dist}} \xi$, distributed according to $\Phi_n(\xi; 0, 1)$, where $\Phi_n(\xi; 0, 1)$ is the standard normal distribution defined in (0.1.4) and σ_f/\sqrt{n} is called the **standard error** or **probable error**. However, this form of the standard error is not too useful since neither σ_f or μ_f are known, else a Monte Carlo approximation would not be needed, but $\hat{\sigma}_n^2$ is an unbiased estimator of σ_f^2 and therefore $\hat{\sigma}_n^2$ must converge to σ_f^2 in distribution too and thus σ_f will be replaced by $\hat{\sigma}_n$ relying on continuous extensions of the central limit theorem [147]. However, in general $\hat{\sigma}_n$ is not necessarily an unbiased estimate of σ_f , since a function of an unbiased estimator of a parameter is not the unbiased estimate of the function of the parameter, as pointed out by Hammersley and Handscomb [104].

Monte Carlo Estimate Confidence Intervals

Following Glasserman’s [96] arguments for confidence intervals with variations, the convergence in distribution (10.43) implies as $n \rightarrow +\infty$,

$$\text{Prob} \left[\hat{\mu}_n - \mu_f \leq \frac{\hat{\sigma}_n}{\sqrt{n}} \xi \right] \sim \Phi_n(\xi; 0, 1),$$

so replacing ξ by $-\xi$,

$$\text{Prob} \left[\hat{\mu}_n - \mu_f \leq -\frac{\hat{\sigma}_n}{\sqrt{n}} \xi \right] \sim \Phi_n(-\xi; 0, 1)$$

and consequently we have an asymptotic formula for confidence intervals about the true mean μ_f ,

$$\text{Prob} \left[-\frac{\hat{\sigma}_n}{\sqrt{n}} \xi \leq \hat{\mu}_n - \mu_f \leq \frac{\hat{\sigma}_n}{\sqrt{n}} \xi \right] \sim \Phi_n(\xi; 0, 1) - \Phi_n(-\xi; 0, 1) = 2\Phi_n(\xi; 0, 1) - 1.$$

Putting this in a more useful form, let $\delta > 0$ and $\xi = \xi(\delta)$ such that $2\Phi_n(\xi(\delta); 0, 1) - 1 = 1 - \delta$ or

$$\Phi_n(\xi(\delta); 0, 1) = 1 - \delta/2 \tag{10.44}$$

to simplify the inversion. Thus, a practical, asymptotic **confidence level** $1 - \delta$ or $100(1 - \delta)\%$ is given by the probability

$$\text{Prob} \left[\hat{\mu}_n - \frac{\hat{\sigma}_n}{\sqrt{n}} \xi(\delta) \leq \mu_f \leq \hat{\mu}_n + \frac{\hat{\sigma}_n}{\sqrt{n}} \xi(\delta) \right] \sim 1 - \delta, \tag{10.45}$$

that the true mean μ_f is in the **confidence interval**

$$\left(\hat{\mu}_n - \frac{\hat{\sigma}_n \xi(\delta)}{\sqrt{n}}, \hat{\mu}_n + \frac{\hat{\sigma}_n \xi(\delta)}{\sqrt{n}} \right).$$

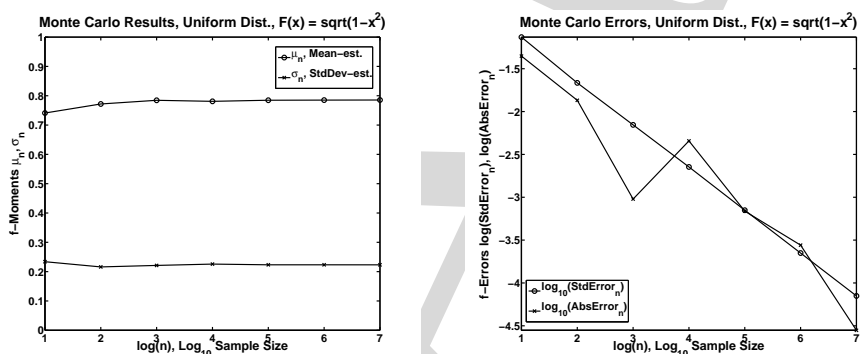
If $\xi(\delta) = 1$, the difference between the true value and the estimate is just \pm standard error with a confidence level of 68.27% that the simulation will be in the confidence interval, but 32.63% chance that it will be out of it. If the difference is ± 2 -standard error then the level is 95.45%, but only a 4.55% “**lack of confidence**” level. Anyway, it will be assumed that the probable error of the Monte Carlo estimator

$$\hat{e}_n = |\hat{\mu}_n - \mu_f| \propto \hat{\sigma}_n / \sqrt{n}.$$

An important observation is that this probable or standard error is independent of the dimension of the volume n_x , as long as the volume is known. However, if it is necessary to approximate the volume due to its complexity, then this approximation will influence the real error.

Example 10.5. Convergence and Errors in Monte Carlo Estimators:
Monte Carlo simulations are illustrated in Fig. 10.8 using the uniform density

$\phi_X(x) = 1/(b - a)$ on $[a, b]$ for the one-dimensional integral of $F(x) = \sqrt{1 - x^2}$ on $[a, b]$, $-1 \leq a < b \leq +1$, so $f(x) = (b - a) \cdot F(x)$. The computational convergence of the mean $\hat{\mu}_n$ and standard deviation $\hat{\sigma}_n$ estimations of $f(x)$ versus the logarithm of sample size $\log_{10}(n)$ are exhibited in Subfig. 10.8(a), while the logarithm of the standard error $\log_{10}(\hat{\sigma}_n/\sqrt{n})$ is shown versus the logarithm of the actual absolute error $\log_{10}(|\hat{\mu}_n - \mu_f|)$, in Subfig. 10.8(a). The computational convergence is somewhat smooth from $n = 10$ to $n = 10,000,000$, but differences in the errors are more dramatic reflecting the slight variability of $\hat{\sigma}_n/\sqrt{n}$ and the greater variability of $\hat{\mu}_n$ compared to the constant exact value μ_f on a log-log plot.



(a) Moments of $f(x)$, $\hat{\mu}_n$ and $\hat{\sigma}_n/\sqrt{n}$.

(b) Logarithm of errors, $\log_{10}(\hat{\sigma}_n/\sqrt{n})$ and $\log_{10}(|\hat{\mu}_n - \mu_f|)$.

Figure 10.8. Monte Carlo simulations for testing use of the uniform distribution to approximate the integral of the integrand $F(x) = \sqrt{1 - x^2}$ on $(a, b) = (0, 1)$ using MATLAB code A.19 on p. 554 for $n = 10^k$, $k = 1:7$.

Finite Difference Comparison

Three important characteristics of Monte Carlo estimators, from Glasserman [96], are bias, variance and computational effort or time. For computational effort, a primary comparison is with the traditional finite difference methods.

Let the Monte Carlo target integral of (10.32) be over a unit n_x -dimensional hypercube for simplicity, i.e.,

$$\mathcal{V} \equiv [0, 1]^{n_x} = [0, 1] \times [0, 1] \times \cdots \times [0, 1]; \quad V = (1 - 0)^{n_x} = 1,$$

decomposed into a regular grid of m fixed steps $\Delta X = 1/m$ in each dimension. so that the grid points in the i th dimension are

$$X_{i,j_i} = j_i/m, \text{ for } j_i = 0:m \text{ and } i = 1:n_x.$$

The finite difference approximation will be an expansion of the form,

$$I[F] \simeq I_m^{(\text{fd})} = \sum_{j_1=1}^m \cdots \sum_{j_{n_x}=1}^m \omega_{j_1} \cdots \omega_{j_{n_x}} \cdot F(j_1/m, \dots, j_{n_x}/m),$$

where the finite difference method weights are denoted by ω_{j_i} for $j_i = 0 : m$ and $i = 1 : n_x$, but must at least satisfy the volume conservation consistency condition that

$$\prod_{i=1}^{n_x} \sum_{j_i=1}^m \omega_{j_i} \cdot 1 = V = 1,$$

and the higher order the method will have even more conditions to be satisfied. There are $m + 1$ grid points per dimension, so the total number of grid points will be $n_{\text{fd}} = (m + 1)^{n_x}$ or $m = n_{\text{fd}}^{1/n_x} - 1$. An **r th order finite difference (fd) method** will have the following error estimate

$$e_{\text{fd}} = I_m^{(\text{fd})} - I[F] = O((\Delta X)^r) = O(m^{-r}) = O((n_{\text{fd}})^{-r/n_x}), \quad (10.46)$$

so for n_{fd} and r fixed,

$$e_{\text{fd}} \longrightarrow O((n_{\text{fd}})^{-0}) = O(1), \text{ as } n_x \rightarrow \infty,$$

i.e., in the limit of high problem dimensions, finite difference methods with fixed step sizes become useless, independent of the order r of the method.

A rough theoretical comparison between the computational effort of the Monte Carlo method and fixed spaced finite difference methods (Newton-Cotes rules) can be made by assuming that the gross computational effort will be the order of the total number of points and they will be the same for both types of methods, i.e., $n_{\text{fd}} = n$. Also, for a fair comparison, assume that these methods have comparable global errors, i.e., $e_{\text{fd}} = O(\hat{\epsilon}_n)$ or that the orders of the errors are the same,

$$n^{-r/n_x} = 1/\sqrt[n]{n},$$

which implies that the dimension of \mathcal{V} is related to the order of the finite difference method r ,

$$n_x = 2r.$$

Since the Monte Carlo method is a global method, r must be taken to be the global order of the finite difference method. For the simplest integration rule, the left or right rectangular rules (Itô's forward integration is the left rectangular rule), the global order is $r = 1$, so Monte Carlo and finite differences are comparable in computational effort and error when $n_x = 2$. For the trapezoidal or midpoint rule, $r = 2$ and $n_x = 4$ when comparable. For Simpson's (1/3) rule, $r = 4$ and $n_x = 8$ when comparable for even spacing, but for uneven grid spacing $r = 3$ since the cubic bonus due to even spacing symmetry is lost and $n_x = 6$ instead (similarly the midpoint rule order is reduced to that of the other rectangular rules). See the comments corresponding to Fig. 10.10 for comparing results from the trapezoidal and Simpson's rules with the Monte Carlo method using the rejection technique.

Monte Carlo Advantages*

- Error is theoretically independent of problem dimension, $n_x = \dim[\mathcal{V}]$.
- So, no curse of dimensionality, but best if $n_x \geq 5$ or so and several random samples are used, i.e.,

$$\left\{ X_{i,j}^{(k)} \mid i = 1:n_x, j = 1:n \text{ sample points}, k = 1:K \text{ samples} \right\}.$$

- Works for complex integrands and domains.
- Not too sensitive to reasonable sample random number generator.

Monte Carlo Disadvantages*

- Probabilistic error bounds, not strict errors bounds that can not be exceeded, e.g., 32% of samples can exceed standard error, $\sigma_f/\sqrt{n} \simeq \hat{\sigma}_n/\sqrt{n}$.
- Irregularity of $F(\mathbf{x})$ or $f(\mathbf{x})$ is not considered, so missed spikes or outliers possible.
- Generating many large random sample sets for high accuracy can be costly in computer and user time.
- Interplay of functions and volumes can be very complex.

Monte Carlo Ratios and Efficiencies

Any advantages* and disadvantages* are subject to testing and performance evaluation in each case. When comparing two different Monte Carlo methods, say one with the **basic Monte Carlo method** of Subsect. 10.2.1 with variance σ_1^2 and another with variance reduced to variance σ_2^2 , both likely to be estimated valued, then the user should compare the methods with the **variance reduction ratio**, or simply the **variance ratio**, defined [104] as the improvement ratio from method 1 relative to method 2,

$$\text{VRR}_{1,2} = \frac{\sigma_1^2}{\sigma_2^2}, \tag{10.47}$$

that is method 2 is the better variance reducer if $\text{VRR}_{1,2} > 1$ and significantly larger.

However, checking for variance reduction alone is not sufficient since the computational costs of the variance reduction should not be excessive, so the **computational cost ratio**

$$\text{CCR}_{1,2} = \frac{\tau_1}{\tau_2}, \tag{10.48}$$

should also be checked, where τ_1 is the computational cost (e.g., CPU time) of the first method (usually the basic Monte Carlo method) and τ_2 is the computational cost of the second method.

Hammersley and Handscomb [104] combine both the variance and computational cost ratios into the **efficiency** of method 2 relative to method 1 as

$$\text{Eff}_{1,2} = \text{VRR}_{1,2} \cdot \text{CCR}_{1,2} = \frac{\sigma_1^2 \tau_1}{\sigma_2^2 \tau_2}. \quad (10.49)$$

See also Glasserman [96, pp. 9-12] or Glynn and Whitt [97] for a more thorough description of Monte Carlo efficiency. In addition, Glasserman [96, p. 185] has observed that

The greatest gains in efficiency from variance reduction techniques result from exploiting specific features of a problem, rather than from generic potential variance reduction.

In fact, two primary methods of variance reduction rely on the Monte Carlo user choosing a known factor that represents a significant proportion of the variability of the target function $f(\mathbf{x})$ or the associated density $\phi_{\mathbf{X}}(\mathbf{x})$. **Importance sampling** techniques rely on finding a multiplicative factor that is a better density than the one originally proposed. **Control variate** techniques rely on finding a known additive factor so that when the factor is subtracted from the target function the variance is significantly reduced. In any case, the selection usually depends on good user knowledge of the problem or related model problems.

10.2.2 Inverse Method for Generating Non-Uniform Variates

When there is an explicit formula for a distribution of a non-uniform variate in terms of elementary functions, then since the distribution function must lie in $[0, 1]$, an inverse of the distribution function in terms of a uniform variate can transform the non-uniform random variate so that it can be generated by a uniform random variate.

Example 10.6. Inversion of Exponential to Uniform Distribution:

This was illustrated very early in Subsection 0.1.7 for the exponential distribution. From (0.40), the exponential distribution for variable $x \geq 0$ and mean $\mu \geq 0$ is

$$\Phi_e(x; \mu) = 1 - \exp(-x/\mu),$$

so equating this to the uniform distribution on $[0, 1]$,

$$\Phi_u(u) = \text{Prob}[0 \leq U \leq u] = u = 1 - \exp(-x/\mu)$$

and inverting yields the inverse relation,

$$x = -\mu \cdot \ln(1 - u).$$

However, some computing effort can be saved by eliminating the floating point subtraction in the log-argument by using the complementary property of $\Phi_u(u)$ that $1 - \Phi_u(u) = 1 - u = \text{Prob}[0 \leq U \leq 1 - u]$ is also a uniform distribution for $(1 - u)$

on $[0, 1]$ (this may seem overly simple, but many students in the sciences without strong statistics background have difficulty accepting this unless it is spelled out). Thus, matching the uniform to the exponential distribution can also be formatted as,

$$\text{Prob}[0 \leq U \leq 1 - u] = 1 - u = 1 - \exp(-x/\mu),$$

leading to a more efficient form for simulations,

$$x = -\mu \cdot \ln(u), \tag{10.50}$$

especially when there are a large number of simulations, $X_i = -\mu \cdot \ln(U_i)$ for $i = 1:n$, e.g., $n = 1.e+6$.

In general, if it is necessary to generate random variates from a non-uniform random variate X_i with a known distribution function $\Phi_X(x)$ but without an existing random number generator, then if $\Phi_X(x)$ is strictly increasing, $\Phi'_X(x) > 0$, and so an inverse exists,

$$U_i = \Phi_X(X_i) \iff X_i = \Phi_X^{-1}(U_i). \tag{10.51}$$

Validation that (10.51) is correct follows from the chain of equations,

$$\begin{aligned} \Phi_X(x) &\equiv \text{Prob}[X \leq x] = \text{Prob}[\Phi_X^{-1}(U) \leq x] \\ &= \text{Prob}[U \leq \Phi_X(x)] = \text{Prob}[U \leq u] \equiv \Phi_U(u), \end{aligned}$$

using the definition of a probability distribution, (10.51) for pairs (X, U) and (u, x) and the definition of the inverse. For practical purposes, this would mean that $\Phi_X(x)$ is in the form of elementary functions.

Example 10.7. Use of Built-in Inverses:

*In some special cases, efficient numerical inverses are available, such as the inverse for the error function or complementary error function, `erfinv` or `erfcinv`, in MATLAB™, which can be used for inverting the normal distribution (if access to the Statistics Toolbox of MATLAB™ is available, then `norminv` builtin function can be used, but the definition `norminv(x) = -sqrt(2)*erfcinv(2*x)` is trivial, so the toolbox is not necessary). In Maple™, the general procedure using the `stats[random]` statistics subpackage is based upon its uniform random generator function with the specification of 'inverse' option for non-uniform distributions by the inverse cumulative distribution function ('`icdf`') method, unless a builtin function is called by name, e.g., `normald` for normal distribution, or the automatic ('`auto`') builtin option is specified.*

For more general cases, when either (1) the distribution $\Phi_X(X_i)$ has a flat subinterval on the interior of its range, say (c, d) , i.e., there is a least one subinterval $c < x_i \leq x \leq x_{i+1} < d$ where $\Phi'_X(x) = 0$, or (2) the distribution has a jump in the interior of its range, i.e., there is an x_j such that $\Phi_X(x_j^+) > \Phi_X(x_j^-)$. The book of Glasserman [96, Section 2.2.1] is a good reference for these irregular cases and also a good source for many **inverse transform method** examples.

One important example for this book on jump-diffusions is the inversion of the cumulative discrete Poisson distribution with mean Λ to the continuous uniform distribution. The Poisson distribution (0.50) is written as the n th order cumulative distribution is written with a distribution recursion as

$$P(N) = \sum_{k=0}^N p_k(\Lambda); \quad p_0(\Lambda) = 1; \quad p_{k+1}(\Lambda) = \Lambda \cdot p_k(\Lambda)/(k+1).$$

Glasserman's [96] pseudo-code is translated to MATLAB™ code in Fig. 10.9 below. A facsimile of the code in Fig 10.9 has been used successfully by Zhu and

```
function N = cumpois(Lambda)
% cumpois function turns uniform point into Poisson count;
U = rand; % generate 1 uniform random point;
% code can be changed to use vector U if needed;
pk = exp(-Lambda); % initialize Poisson distribution;
P = 0; % initialize cumulative distribution;
N = 0; % initialize cumulative jump counter;
while U > P, % generate 1 uniform random point;
    N = N + 1; \% step jump counter if U too small;
    pk = Lambda*pk/N; \% update Poisson distribution;
    P = P + pk; \% update cumulative distribution;
end
% End function cumpois; returns count N at mean rate Lambda;
```

Figure 10.9. Code: Inverse of Poisson to Uniform Distribution [96, Fig. 3.9].

Hanson [285] in their Monte Carlo simulation of risk-neutral European call option pricing, cited in Example 10.3.

If the components of a vector variate are an independent set of random variates, then it is fairly easy to invert the distribution in favor of a set of independent uniform variates since the joint distribution of independent variates is the product of component marginal distributions (Defn. 0.35, p. 25), .i.e., if

$$\Phi_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{n_x} \Phi_{X_i}(x_i) = \prod_{i=1}^{n_x} u_i, \quad (10.52)$$

then

$$\mathbf{x} = [x_i]_{n_x \times 1} = [\Phi_{X_i}^{-1}(u_i)]_{n_x \times 1}, \quad (10.53)$$

using the inversion transform method component by component.

For instance, if \mathbf{X}_j are IID exponentially distributed random vectors with vector mean $\boldsymbol{\mu}$ and \mathbf{U}_j are generated IID uniformly distributed random vectors for each sample point $j = 1:n$, then the \mathbf{X}_j can be generated by

$$\mathbf{X}_j = [X_{i,j}]_{n_x \times 1} = [-\boldsymbol{\mu}_i \ln(U_{i,j})]_{n_x \times 1} = -\boldsymbol{\mu} \cdot \log(\mathbf{U}_j),$$

for all j , where $\cdot*$ is the elements-wise multiplication symbol and $\log(\mathbf{U})$ is the vectorized natural logarithm function of vector \mathbf{U} as in MATLABTM (`log10` is the corresponding MATLABTM base 10 logarithm).

10.2.3 Acceptance and Rejection Method of von Neumann

The method of acceptance and rejection is due to von Neumann [268], one of the earliest techniques introduced into the Monte Carlo method. It can be applied to produce samples for unusual probability distributions as well as for unusual domains since the method uses simpler problems that are easier to draw variates in simpler domains. In two dimensions, it is just a matter to find the proportion of points from the simpler, bounding area which lie in the more complicated, interior area.

Note that knowing the formula for a density function, $\phi_{\mathbf{X}}^{(1)}(\mathbf{x})$ on domain \mathcal{V} , does not mean we know how to generate random variates \mathbf{X}_i for it. Let $\phi_{\mathbf{X}}^{(2)}(\mathbf{x})$ be another density function, such as a uniform, normal or exponential density function, which is simpler (else not useful), for which there is a known method for generating the corresponding random variates, $\mathbf{X}_i^{(2)}$ and suppose there is a positive constant c for the relative bound

$$\phi_{\mathbf{X}}^{(1)}(\mathbf{x}) \leq c \cdot \phi_{\mathbf{X}}^{(2)}(\mathbf{x}), \quad (10.54)$$

for \mathbf{x} in \mathcal{V} . For consistency, the target density $\phi_{\mathbf{X}}^{(1)}(\mathbf{x})$ should have a zero when the known comparison, generating density $\phi_{\mathbf{X}}^{(2)}(\mathbf{x})$ does, so the relative bound can be written

$$\frac{\phi_{\mathbf{X}}^{(1)}(\mathbf{x})}{(c\phi_{\mathbf{X}}^{(2)}(\mathbf{x}))} \leq 1,$$

assuming that $0/0 \leq 1$ has been defined. The unit bound indicating that a scalar uniform density will be useful. Since both are densities, the relative bound means that $1 \leq c \cdot 1$ upon integrating both sides of (10.54), so $c \geq 1$ is required. The procedure for the acceptance-rejection method or technique on the i th step is

1. Generate a random variate $\mathbf{X}_i^{(2)}$ for the comparison density $\phi_{\mathbf{X}}^{(2)}(\mathbf{x})$ (e.g., this comparison density could also be a uniform density for one-dimension, in which case, `X(i) = rand`, in MATLABTM).
2. Compute the relative ordinate

$$Y_i = \frac{\phi_{\mathbf{X}}^{(1)}(\mathbf{X}_i^{(2)})}{(c\phi_{\mathbf{X}}^{(2)}(\mathbf{X}_i^{(2)}))}, \quad (10.55)$$

with generated $\mathbf{X}_i^{(2)}$, assuming the relative bound constant c has already been calculated.

3. Generate a scalar uniform random variate U_i and use it to accept or reject the relative ordinate Y_i , such that

- If $U_i \leq Y_i$, then **accept** $\mathbf{X}_i^{(1)} = \mathbf{X}_i^{(2)}$ as a variate for the target density $\phi_{\mathbf{X}}^{(1)}$ and get another point \mathbf{X}_{i+1} , stepping i , unless $i + 1 > n$.
- Else, if $U_i > Y_i$, then **reject** the current $\mathbf{X}_i^{(2)}$ and try another i th generated variate from comparison density $\phi_{\mathbf{X}}^{(2)}$.

Note that

$$\text{Prob}[\mathbf{X}^{(2)} \text{ Rejected}] = \frac{\text{TotalArea} [c\phi_{\mathbf{X}}^{(2)}(\mathbf{x}) - \phi_{\mathbf{X}}^{(1)}(\mathbf{x})]}{\text{TotalArea} [c\phi_{\mathbf{X}}^{(2)}(\mathbf{x})]} = \frac{c - 1}{c} \leq 1,$$

so, in addition, the user wants $(c - 1)$ to be small and positive, i.e., c should be a tight bound constant, to reduce the amount of computation to avoid too many rejected attempts and thus increase efficiency. Also, the target distribution $\Phi_{\mathbf{X}}^{(1)}(\mathbf{x})$ for $\mathbf{X} = \mathbf{X}^{(1)}$ (vector inequalities are shorthand notation for a set of component equalities) is

$$\begin{aligned} \text{Prob}[\mathbf{X} \leq \mathbf{x}] &= \frac{\text{TotalArea} [\phi_{\mathbf{X}}^{(1)}(\mathbf{y}) \mid \mathbf{y} \leq \mathbf{x}]}{\text{TotalArea} [c\phi_{\mathbf{X}}^{(2)}(\mathbf{x})]} + \text{Prob}[\mathbf{X}^{(2)} \text{ Rejected}] \cdot \text{Prob}[\mathbf{X} \leq \mathbf{x}] \\ &= \frac{1}{c}\Phi_{\mathbf{X}}^{(1)}(\mathbf{x}) + \frac{c - 1}{c}\Phi_{\mathbf{X}}^{(1)}(\mathbf{x}) = \Phi_{\mathbf{X}}^{(1)}(\mathbf{x}), \end{aligned}$$

consistent with the definition of a distribution.

Example 10.8. Application of Acceptance-Rejection with Normal Distribution:

Figure 10.10, a computational application of the acceptance-rejection technique is illustrated for the truncated normal distribution $\Phi_n(a, b; 01)$ defined for a previous uniform-normal comparison in (10.42) of Example 10.4. The computation converges nicely, with standard errors of $2.1e-4$ when $n = 10^6$ sample points and $6.59e-5$ when $n = 10^7$. However, when these one-dimensional results are compared to standard finite difference methods the results are not so impressive, e.g., the trapezoidal rule has an absolute error of $2.88e-05$ using 101 points and Simpson's (1/3) rule has an absolute error of $3.09e-9$ using the same 101 points, although, as we have said, the finite difference methods are better for low dimensions.

Example 10.9. Multidimensional Application of Acceptance-Rejection Technique:

Figure 10.11 illustrates the application of Monte Carlo multidimensional simulations with the von Neumann acceptance-rejection technique similar the former $n_x = 1$ truncated normal distribution problem (10.42) in Example 10.4, but here for dimensions $n_x = 2 : 5$. Subfig. 10.11(a) exhibits the Monte Carlo mean estimates, $\hat{\mu}_n$, which roughly settle down by sample size $n = 10^4$, but definitely by $n = 10^5$ for this problem and sample sets. In Subfig. 10.11(b), the Monte Carlo standard error

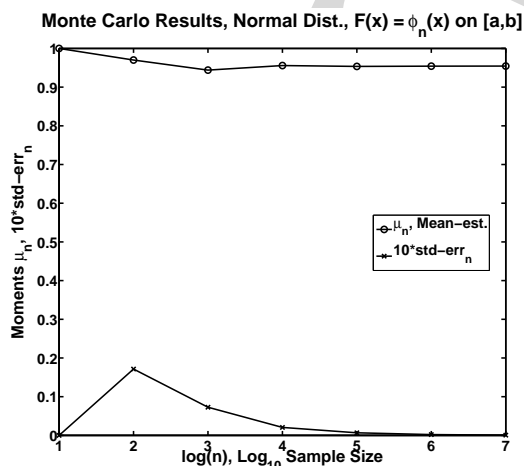


Figure 10.10. Monte Carlo simulations shown apply the acceptance and rejection technique and the normal distribution to compute the estimates for the mean $\hat{\mu}_n$ and the magnified standard error $10 \cdot \hat{\sigma}_n/\sqrt{n}$ for the integral of the truncated normal distribution with $F(x) = \phi_n(x)$ on $[a, b] = [-2, 2]$ using MATLAB code A.20 on p. 556 for $n = 10^k$, $k = 1:7$.

estimates, $\hat{\sigma}_n/\sqrt{n}$ are displayed, showing a remarkable similar decay in sample size beyond sample size $n = 10^3$. Note that since the integrand $F(\mathbf{x}) = \phi_n(\mathbf{x})$ is the normal density restricted to the vector interval $[\mathbf{a}, \mathbf{b}]$, the normal density scaled integrand is $f(\mathbf{x}) = \mathbf{1}_{\mathbf{x} \in [\mathbf{a}, \mathbf{b}]}$, an indicator function for the set $[\mathbf{a}, \mathbf{b}]$, so $f^2(\mathbf{x}) = f(\mathbf{x})$ and the estimate of the standard error,

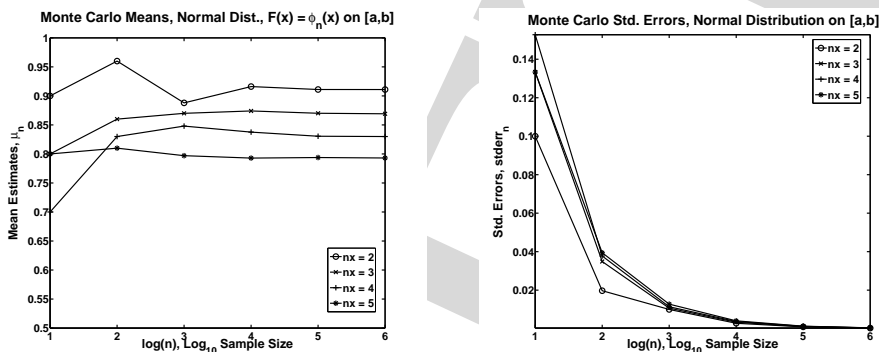
$$\hat{\sigma}_n/\sqrt{n} = \sqrt{\hat{\mu}_n(1 - \hat{\mu}_n)/(n - 1)},$$

satisfies the same formula regardless of dimension n_x as suggested by the Monte Carlo theory.

Box-Muller Algorithm for Normal Random Variates

Many of the normal random number generators, if not all, use the algorithm of Box and Muller [37] or updates of it [191] (see also [226, 96]). Since the normal distribution is a special function that cannot be put in terms of elementary functions, it is not exactly invertible by the inverse transform method, except numerically or artificially by defining another special function for the inverse. Box and Muller use pairs of uniform variates and polar coordinate to construct their algorithm to compute a pair of normal variates.

Let U_1 and U_2 be two independent uniform variates on $(0, 1)$, use them to construct a pair of polar coordinates (R, T) and then use those to construct two



(a) Mean estimates, $\hat{\mu}_n$, for $f(\mathbf{x})$.

(b) Logarithm of standard errors, $\log_{10}(\hat{\sigma}_n/\sqrt{n})$.

Figure 10.11. Monte Carlo simulations for estimating multi-dimensional integrals for the n_x -dimension normal integrand $F(\mathbf{x}) = \phi_n(\mathbf{x})$ on $[\mathbf{a}, \mathbf{b}] = [-2, 2]^{n_x}$ using MATLAB code A.21 on p. 558 for $n = 10^k$, $k = 1:6$. The acceptance-rejection technique is used to handle the finite domain.

independent normal variates (X_1, X_2) ,

$$R = \sqrt{-2 \ln(U_1)} \quad \text{and} \quad T = 2\pi U_2,$$

$$X_1 = R \cos(T) \quad \text{and} \quad X_2 = R \sin(T),$$

where $0 < R < \infty$ and $0 < T < 2\pi$ since $0 < U_i < 1$ for $i = 1:2$. The inverse transformation is then $\tan(2\pi U_2) = X_2/X_1$ and $-2 \ln(U_1) = X_1^2 + X_2^2$ or

$$U_1 = \exp(-(X_1^2 + X_2^2)/2) \quad \text{and} \quad U_2 = \tan^{-1}(X_2/X_1)/(2\pi).$$

The Jacobian of the transformation, $(X_1, X_2) \rightarrow (U_1, U_2)$, is

$$J = \frac{\partial(U_1, U_2)}{\partial(X_1, X_2)} = \text{Det} \begin{bmatrix} \partial U_1 / \partial X_1 & \partial U_1 / \partial X_2 \\ \partial U_2 / \partial X_1 & \partial U_2 / \partial X_2 \end{bmatrix} \quad (10.56)$$

$$= -\exp(-(X_1^2 + X_2^2)/2) / (2\pi) = -\Phi_n(X_1, X_2; (0, 0), (1, 1)),$$

i.e., the negative of a standard 2-dimensional normal distribution for two independent, standard normal variates (X_1, X_2) , so the objective and only $|J|$ is needed. Conservation of probability consistency is easily verified, since in theory,

$$1 = \int_0^1 \int_0^1 du_1 du_2 = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} |J| dx_1 dx_2 = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} e^{-x_1^2/2} e^{-x_2^2/2} dx_1 dx_2 = 1.$$

Marsaglia and Bray [191] modified the Box-Muller algorithm save computing costs by using the acceptance-rejection technique between a square enclosing the

unit circle so that the sine and cosine functions would not be needed. They begin by generating two independent uniform variates on the square $(-1, +1) \times (-1, +1)$ rather than on $(0, 1) \times (0, 1)$, i.e., keeping U_1 and U_2 as the initial $(0, 1)$ uniform variates, $U_3 = 2 \cdot U_1 - 1$ and $U_4 = 2 \cdot U_2 - 1$. Next let the squared radius be $R_2 = U_3^2 + U_4^2$ and while $R_2 > 1$, i.e., out of the unit circle, then reject it and try again, but if $R_2 \leq 1$ then compute the normalized Box-Muller radius $R_3 = \sqrt{-2 \ln(R_2)}/R_2$ and finally output the independent, standard normal variate pair,

$$X_3 = R_3 \cdot U_3 \quad \text{and} \quad X_4 = R_3 \cdot U_4.$$

10.2.4 Importance Sampling

There are two principal ways to reduce the standard error and thus improve the likely accuracy of Monte Carlo simulation relative to the basic Monte Carlo simulation (Subsection 10.2.1; Hammersley and Handscomb call the method in their 1964 compact little book [104, Section 5.2] the **crude Monte Carlo method**). One way is to increase the sample size n , but the computational cost is high, e.g., increasing the sample size 100 times is necessary to reduce the standard error by 1/10th of its magnitude due to the weak reciprocal square root order. The other way is to reduce the variance and a way to do that is to pick a better density to draw samples from that more closely matches the integrand $F(\mathbf{x})$. **Importance sampling** methods strive to find the better or practical best distribution. As previously mentioned, importance sampling was introduced into the Monte Carlo method in one of the earliest papers [203] on the subject, sometimes called the Metropolis algorithm, in which the desirable sampling distribution was the exponential distribution of energy changes.

Suppose there is an initial density $\phi_{\mathbf{X}}(\mathbf{x})$ with mean of $f(x)$ integral

$$\mu_f = E_{\mathbf{X}}[f(\mathbf{X})] = \int_{\mathcal{V}} f(\mathbf{x}) \phi_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = \int_{\mathcal{V}} F(\mathbf{x}) d\mathbf{x}, \quad (10.57)$$

but we seek a better density $\tilde{\phi}_{\mathbf{X}}(\mathbf{x})$ that more closely characterizes the original integrand $F(\mathbf{x})$ and leads to the equivalent formulation,

$$\tilde{\mu}_{\tilde{f}} = E_{\tilde{\phi}} \left[\left(f \phi / \tilde{\phi} \right) (\mathbf{X}) \right] = \int_{\mathcal{V}} \left(f \phi / \tilde{\phi} \right) (\mathbf{x}) \tilde{\phi}_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = \mu_f, \quad (10.58)$$

where $\tilde{f}(\mathbf{x}) \equiv \left(f \phi / \tilde{\phi} \right) (\mathbf{x})$ is a potentially less variable sample target function and $E_{\tilde{\phi}}$ denotes an expectation with respect to the new density $\tilde{\phi}_{\mathbf{X}}(\mathbf{x})$, subject to minimal likelihood properties that

$$\tilde{\phi}_{\mathbf{X}}(\mathbf{x}) \geq 0 \iff \phi_{\mathbf{X}}(\mathbf{x}) \geq 0,$$

mainly so that any indeterminate 0/0 form can be defined as 1. The corresponding variance is given by

$$\tilde{\sigma}_{\tilde{f}}^2 = \int_{\mathcal{V}} \left(\tilde{f}(\mathbf{x}) - \tilde{\mu}_{\tilde{f}} \right)^2 \tilde{\phi}_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = \int_{\mathcal{V}} \tilde{f}^2(\mathbf{x}) \tilde{\phi}_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} - \tilde{\mu}_{\tilde{f}}^2. \quad (10.59)$$

Since the means are the same, $\tilde{\mu}_f^2 = \mu_f^2$, reduction of the variance is equivalent to reduction of the new second moment, i.e., making

$$\int_{\mathcal{V}} \tilde{f}^2(\mathbf{x}) \tilde{\phi}_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} < \int_{\mathcal{V}} f^2(\mathbf{x}) \phi_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}.$$

In importance sampling the goal is to sample at important points of $\tilde{f}(\mathbf{x})$ such as points of maximum likelihood (see Glasserman [96]).

The Monte Carlo unbiased estimates are the means for n -point samples,

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n f(\mathbf{X}_i) \quad \text{and} \quad \hat{\tilde{\mu}}_n = \frac{1}{n} \sum_{i=1}^n \tilde{f}(\tilde{\mathbf{X}}_i), \quad (10.60)$$

where the points $\tilde{\mathbf{X}}_i$ are sampled from the distribution of the $\tilde{\phi}_{\mathbf{X}}(\mathbf{x})$ density, while the unbiased sample variances are

$$\hat{\sigma}_n^2 = \frac{1}{n-1} \sum_{i=1}^n (f(\mathbf{X}_i) - \hat{\mu}_n)^2 \quad \text{and} \quad \hat{\tilde{\sigma}}_n^2 = \frac{1}{n-1} \sum_{i=1}^n (\tilde{f}(\tilde{\mathbf{X}}_i) - \hat{\tilde{\mu}}_n)^2. \quad (10.61)$$

As with the exact variances or second moments, it is expected that the new second sampled moment is reduced, i.e.,

$$\frac{1}{n} \sum_{i=1}^n \tilde{f}^2(\tilde{\mathbf{X}}_i) < \frac{1}{n} \sum_{i=1}^n f^2(\mathbf{X}_i),$$

since this is equivalent to $\hat{\tilde{\sigma}}_n^2 < \hat{\sigma}_n^2$ for sample variances or $\hat{\tilde{\sigma}}_n/\sqrt{n} < \hat{\sigma}_n/\sqrt{n}$ for the Monte Carlo standard error estimates.

The best choice of a new density would obviously be the normalized absolute value the full problem,

$$\tilde{\phi}_{\mathbf{X}}(\mathbf{x}) = |F(\mathbf{x})| / \int_{\mathcal{V}} |F(\mathbf{y})| d\mathbf{y},$$

but that would be an absurd circular argument since the normalization factor in the denominator would be the integral we are seeking to estimate if it were the case that $F(\mathbf{x}) > 0$. As Glasserman [96, Fig. 4.16] states, importance sampling is the most complex of Monte Carlo techniques for reducing variance, but has the potential whose effectiveness ranges from the best to the worst. See Glasserman's [96] book for a more advanced treatment.

Analogous concepts arose long ago in the statistically related Gaussian quadrature rules [226], i.e., Gauss statistics quadrature [270], of numerical analysis. For instance, the Gauss-Legendre rules correspond to integrals weighted in proportion to a uniform density on $[-1, +1]$, Gauss-Laguerre rules to the exponential or gamma densities on $(0, \infty)$ and Gauss-Hermite to the normal distribution $(-\infty, +\infty)$. The criteria for the numerical weights w_i and nodes x_i is that the Gaussian rules give the **best polynomial precision** for the polynomial approximation to the weighted

function corresponding to the importance sampled $f(x)$. Practical criteria concern matching the rule with the domain, whether finite, semi-infinite or full-infinite, but also matching integrand singularities in the case of certain Gaussian rules not mentioned here.

There is a more advanced code like the adaptive Monte Carlo code called **VEGAS** [179] of Lepage that primarily uses importance sampling, but also uses stratified sampling discussed in the next subsection. The VEGAS algorithm and code is discussed in Numerical Recipes [226].

10.2.5 Stratified Sampling

If the integrands are very variable, then partitioning the domain into disjoint subdomains, computing Monte Carlo estimates on each subdomain and reassembling the estimates to form a global estimate can usually reduce the global estimated variance, sometimes significantly [104, 96, 147, 226].

Consider a partition of the domain \mathcal{V} into np disjoint parts, called **strata**, such that the union

$$\bigcup_{k=1}^{np} \Delta\mathcal{V}_k = \mathcal{V}$$

and the Monte Carlo integral of interest (10.33)

$$\mu_f = E_{\mathbf{X}}[f(\mathbf{X})] = \sum_{k=1}^{np} \int_{\Delta\mathcal{V}_k} f(\mathbf{x})\phi_{\mathbf{X}}(\mathbf{x})d\mathbf{x} = \sum_{k=1}^{np} p_k\mu_f^{(k)}, \quad (10.62)$$

where the k th stratum probability is

$$p_k = \int_{\Delta\mathcal{V}_k} \phi_{\mathbf{X}}(\mathbf{x})d\mathbf{x} > 0 \quad \ni \quad \sum_{k=1}^{np} p_k = 1,$$

assumed known, and the strata mean

$$\mu_f^{(k)} = \int_{\Delta\mathcal{V}_k} f(\mathbf{x})\phi_{\mathbf{X}}(\mathbf{x})d\mathbf{x}/p_k.$$

Let $\mathbf{X}_i^{(k)}$ be a sample point drawn from the density $\phi_{\mathbf{X}}(\mathbf{x})$ on the k th strata $\Delta\mathcal{V}_k$ for $i = 1 : n_k$ where $n_k > 0$ and $\sum_{k=1}^{np} n_k = n$, the sample size. Also let $f_i^{(k)} \equiv f(\mathbf{X}_i^{(k)})$, so that the k th strata Monte Carlo estimate of the mean is

$$\hat{\mu}_{n_k}^{(k)} = \frac{1}{n_k} \sum_{i=1}^{n_k} f_i^{(k)} \quad (10.63)$$

and since $\mu_f = \sum_{k=1}^{np} p_k\mu_f^{(k)}$ the total mean estimate is

$$\hat{\mu}_{n,np} = \sum_{k=1}^{np} p_k \hat{\mu}_{n_k}^{(k)} = \sum_{k=1}^{np} \frac{p_k}{n_k} \sum_{i=1}^{n_k} f_i^{(k)}. \quad (10.64)$$

This strata sampled estimate is an unbiased estimate since

$$\mu_f^{(k)} = E^{(k)} [f(\mathbf{X}_i^{(k)})] \equiv E [f(\mathbf{X}_i^{(k)}) | \mathbf{X}_i^{(k)} \in \Delta\mathcal{V}] .$$

Then,

$$E[\hat{\mu}_{n,np}] = \sum_{k=1}^{np} \frac{p_k}{n_k} \sum_{i=1}^{n_k} E^{(k)} [f_i^{(k)}] = \sum_{k=1}^{np} \frac{p_k}{n_k} \sum_{i=1}^{n_k} \mu_f^{(k)} = \sum_{k=1}^{np} p_k \mu_f^{(k)} = \mu_f ,$$

independent of the sample distribution n_k . Note that the order of Monte Carlo estimation and stratification are not generally interchangeable if the unbiased property is to be preserved. For instance, if the original simple Monte Carlo estimate $\hat{\mu}_n$ (10.39) is directly converted to a stratified sum,

$$\hat{\mu}_n = \frac{1}{n} \sum_{k=1}^{np} \sum_{i=1}^{n_k} f_i^{(k)}$$

and the expectation is calculated as

$$E[\hat{\mu}_n] = \frac{1}{n} \sum_{k=1}^{np} \sum_{i=1}^{n_k} E^{(k)} [f_i^{(k)}] = \frac{1}{n} \sum_{k=1}^{np} n_k \mu_f^{(k)} ,$$

which for general strata means $\mu_f^{(k)}$ this sum will not be μ_f . However, in the special case of **proportional strata sampling** in which $n_k = p_k \cdot n$, then

$$E[\hat{\mu}_n] = \sum_{k=1}^{np} p_k \mu_f^{(k)} = \mu_f .$$

Recall that the exact variance of f is

$$\sigma_f^2 = \text{Var}_{\mathbf{X}} [f(\mathbf{X})] = \sum_{k=1}^{np} \int_{\Delta\mathcal{V}_k} (f(\mathbf{x}) - \mu_f)^2 \phi_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} ,$$

but due to the total mean μ_f the total variance does not easily decompose into the strata variances,

$$\begin{aligned} (\sigma_f^{(k)})^2 &= \text{Var}_{\mathbf{X}}^{(k)} [f(\mathbf{X})] = E^{(k)} \left[(f(\mathbf{X}) - \mu_f^{(k)})^2 \right] \\ &= \int_{\Delta\mathcal{V}_k} (f(\mathbf{x}) - \mu_f^{(k)})^2 \phi_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} / p_k . \end{aligned} \tag{10.65}$$

Following Glasserman [96], the variance is written in with the usual second and first moment technique,

$$\begin{aligned} \sigma_f^2 &= E [f^2(\mathbf{X})] - \mu_f^2 = \sum_{k=1}^{np} p_k E^{(k)} [f^2(\mathbf{X})] - \mu_f^2 \\ &= \sum_{k=1}^{np} p_k \left((\sigma_f^{(k)})^2 + (\mu_f^{(k)})^2 \right) - \left(\sum_{k=1}^{np} p_k \mu_f^{(k)} \right)^2 . \end{aligned} \tag{10.66}$$

In contrast, the strata Monte Carlo estimate, the variance, using prior definitions and the IID property of the $X_i^{(k)}$, is

$$\begin{aligned} \sigma_{\hat{\mu}_{n,np}}^2 &= \text{Var} [\hat{\mu}_{n,np}] = \text{E} \left[\sum_{k=1}^{np} p_k \left(\frac{1}{n_k} \sum_{i=1}^{n_k} f_i^{(k)} - \mu_f^{(k)} \right)^2 \right] \\ &= \sum_{k=1}^{np} \sum_{\ell=1}^{np} \frac{p_k p_\ell}{n_k n_\ell} \sum_{i=1}^{n_k} \sum_{j=1}^{n_\ell} \text{E}^{(k)} \left[\left(f_i^{(k)} - \mu_f^{(k)} \right) \left(f_j^{(\ell)} - \mu_f^{(\ell)} \right) \right] \quad (10.67) \\ &= \sum_{k=1}^{np} \frac{p_k^2}{n_k} \sum_{i=1}^{n_k} \left(\sigma_f^{(k)} \right)^2 = \sum_{k=1}^{np} \frac{p_k^2}{n_k} \left(\sigma_f^{(k)} \right)^2. \end{aligned}$$

Thus, the strata reduction of variance will be

$$\begin{aligned} \sigma_f^2 - \sigma_{\hat{\mu}_{n,np}}^2 &= \sum_{k=1}^{np} p_k \left(1 - \frac{p_k}{n_k} \right) \left(\sigma_f^{(k)} \right)^2 \\ &\quad + \sum_{k=1}^{np} p_k \left(\mu_f^{(k)} \right)^2 - \left(\sum_{k=1}^{np} p_k \mu_f^{(k)} \right)^2 \quad (10.68) \\ &\geq \sum_{k=1}^{np} p_k \left(1 - \frac{p_k}{n_k} \right) \left(\sigma_f^{(k)} \right)^2, \end{aligned}$$

since the second moment majorizes the square of the first moment, here for $\mu_f^{(k)}$ with probability $p_k = 1$. For strata proportional sampling, $n_k = p_k \cdot n$, then

$$\sigma_f^2 - \sigma_{\hat{\mu}_{n,np}}^2 \geq \frac{n-1}{n} \sum_{k=1}^{np} p_k \left(\sigma_f^{(k)} \right)^2, \quad (10.69)$$

so proportional sampling stratification always reduces the variance.

Another form of strata sampling, $n_k = q_k \cdot n$ so $q_k > 0$ and $\sum_{k=1}^{np} q_k = 1$, but q_k is otherwise arbitrary. This form is called **fractional sampling**. The arbitrariness of the fractions q_k can be used to determine the **optimal sampling allocation** of the stratification with the objective to achieve maximum variance reduction for stratification. Since when $n_k = q_k \cdot n$,

$$\sigma_f^2 - \sigma_{\hat{\mu}_{n,np}}^2 \geq \sum_{k=1}^{np} p_k \left(1 - \frac{p_k}{n q_k} \right) \left(\sigma_f^{(k)} \right)^2, \quad (10.70)$$

but instead of maximizing the full right-hand-side of (10.68) for $\sigma_f^2 - \sigma_{\hat{\mu}_{n,np}}^2$, it is only necessary to minimize the bound in (10.70) containing the variable parameter q_k . This can be done using the **Lagrange multiplier** technique to handle the $\sum_{k=1}^{np} q_k = 1$ constraint with λ as the multiplier by letting

$$S(\mathbf{q}, \lambda) = \sum_{k=1}^{np} p_k \left(1 - \frac{p_k}{n q_k} \right) \left(\sigma_f^{(k)} \right)^2 + \lambda \left(\sum_{k=1}^{np} q_k - 1 \right).$$

The reader can easily verify that the optimal allocation solution for the vector of probabilities \mathbf{q} is

$$\mathbf{q}^* = \frac{\mathbf{p} \cdot \sigma}{(\mathbf{b}^\top \sigma)} = \frac{[p_i \sigma_i]_{np \times 1}}{\sum_{k=1}^{np} p_k \sigma_k}, \quad (10.71)$$

by taking the gradient of the objective $S(\mathbf{q}, \lambda)$ with respect to \mathbf{q} and elimination of the multiplier λ . Hence, the optimal bound on the variance reduction is

$$\sigma_f^2 - \left(\sigma_{\hat{\mu}_{n,np}}^*\right)^2 \geq \sum_{k=1}^{np} p_k \left(\sigma_f^{(k)} - \frac{1}{n} \sum_{\ell=1}^{np} p_\ell \sigma_f^{(\ell)} \right) \sigma_f^{(k)}, \quad (10.72)$$

See Glasserman [96] for a more advanced treatment of stratified sampling and see Numerical Recipes [225, 226] for a discussion and the advanced **recursive stratified sampling** code called **MISER**.

10.2.6 Antithetic Variates

The **antithetic variate technique** of variance reduction reuses a prior draw, called the **thetic (or thesis) variate**, to construct an opposing random variable, called the antithetic variate and is usually a mirror image of the thetic variate with the same mean, that has a negative correlation with the thetic variate. The most common antithetic examples are $U_i^{(a)} = 1 - U_i$ for the standard uniform distribution on $[0, 1]$ and $Z_i^{(a)} = -Z_i$ for the standard normal distribution. Hence, $E[U_i^{(a)}] = 0.5 = E[U_i]$, $\text{Var}[U_i^{(a)}] = 1/12 = \text{Var}[U_i]$ and

$$\text{Cov}[U_i, U_i^{(a)}] = -1/12 < 0$$

for the uniform, while $E[Z_i^{(a)}] = 0 = E[Z_i]$, $\text{Var}[Z_i^{(a)}] = 1 = \text{Var}[Z_i]$ and

$$\text{Cov}[Z_i, Z_i^{(a)}] = -1 < 0.$$

The analogous properties hold when the uniform and normal distributions are not standard, i.e. $X_i^{(a)} = b + a - X_i$ on $[a, b]$ for the uniform and $X_i^{(a)} = 2\mu - X_i$ for the normal with mean μ and variance σ . For most other continuous distributions, the samples are drawn from these to standard distributions and are converted by transformation to the target distribution. For example, $X_i = -\mu \ln(U_i)$ and

$$X_i^{(a)} = -\mu \ln(1 - U_i) = -\mu \ln(1 - \exp(-X_i/\mu))$$

for the exponential distribution with mean μ , using Example 10.6.

In order to keep this section from being too complicated, it will be assumed that the distribution from which the Monte Carlo random variates will be drawn will be from the general uniform in one-dimension ($n_x = 1$) with density $\phi(x) = 1/(b-a)$ on (a, b) ,

$$\mu_f = \frac{1}{(b-a)} \int_a^b f(x) dx.$$

Note that the antithetic mean will be the same as the thetic mean,

$$\mu_f^{(a)} = \frac{1}{(b-a)} \int_a^b f(b+a-x) dx = \frac{1}{(b-a)} \int_a^b f(y) dy = \mu_f.$$

For the Monte Carlo estimates,

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n f(X_i); \text{ and } \hat{\mu}_n^{(a)} = \frac{1}{n} \sum_{i=1}^n f(X_i^{(a)}),$$

both converging to μ_f by the strong law of large numbers. For the antithetic variate (*av*) technique, define the thetic-antithetic average mean estimate with limit μ_f as

$$\hat{\mu}_n^{(av)} = \frac{1}{2} (\hat{\mu}_n + \hat{\mu}_n^{(a)}) \tag{10.73}$$

and note that the Monte Carlo sample size has been doubled to $2n$ using only the original IID sample $\{X_i\}$ sample of n points, but at the computational cost of double the number of function evaluations of $f(x)$. However, if the variance can be reduced substantially then the original sample size of n could be reduced to compensate for the additional function evaluations.

The new variance is then

$$\begin{aligned} \text{Var} [\hat{\mu}_n^{(av)}] &= \frac{1}{4} \text{Var} [\hat{\mu}_n + \hat{\mu}_n^{(a)}] \\ &= \frac{1}{4} \text{E} \left[(\hat{\mu}_n - \mu_f)^2 + (\hat{\mu}_n^{(a)} - \mu_f)^2 + 2(\hat{\mu}_n - \mu_f)(\hat{\mu}_n^{(a)} - \mu_f) \right] \tag{10.74} \\ &= \frac{1}{4} \text{Var}[\hat{\mu}_n] + \frac{1}{4} \text{Var} [\hat{\mu}_n^{(a)}] + \frac{1}{2} \text{Cov} [\hat{\mu}_n, \hat{\mu}_n^{(a)}]. \end{aligned}$$

If the covariance $\text{Cov}[\hat{\mu}_n, \hat{\mu}_n^{(a)}]$ is negative, then a variance reduction ratio of no more than one half would be guaranteed, thus paying for the doubled function evaluations in terms of efficiency (10.49). By a result quoted in Boyle et al. [39], if the target function of f is monotonic, then

$$\text{Cov}[\hat{\mu}_n, \hat{\mu}_n^{(a)}] < 0,$$

which is likely true in many applications, e.g., positive payoffs, but perhaps difficult to verify. In multidimensions, negativity conditions will likely have to be replaced by negative semi-definite conditions for practical purposes due to independence across dimensions.

Example 10.10. Antithetic Variates for Compound Poisson Process:

In the jump-diffusion European call option pricing problem of Zhu and Hanson [285], it was necessary to draw a sample from the compound Poisson process with rate λ ,

$$S_i = \sum_{j=1}^{N_i} Q_{i,j}, \text{ for } i = 1:n,$$

estimating the Poisson cumulative sum $\tilde{\mathcal{P}}(t_f)$ in Eq. (10.35) of Example 10.3, where the jump-amplitude marks $Q_{i,j}$ are uniformly distributed on $[a, b]$. First the jump count N_i for $i = 1:n$ sample points is computed by the inverse transform method in Example 10.9, then a set of standard uniform variates $U_{i,j}$ for $j = 1:N_i$ jumps and

$i = 1:n$ points (i.e., $Q_{i,j} = a + (b - a)U_{i,j}$ and $Q_{i,j}^{(a)} = a + (b - a)(1 - U_{i,j})$). Next the partial sums are computed,

$$S_i = aN_i + (b - a) \sum_{j=1}^{N_i} U_{i,j} \quad \text{and} \quad S_i^{(a)} = (a + b)N_i - S_i, \quad (10.75)$$

which are then used to compute thetic-antithetic averages of jump-shifted Black-Scholes formulas and associated jump-exponentials.

10.2.7 Control Variates

As in importance sampling (Subsection 10.2.4) with its multiplicative factoring of the density by seeking a better density, the **control variate** technique [104] seeks an additive factor, but a known one, that is representative of the variability in the target integrand. This technique was introduced in general by Hammersley-Handscorn [104] in their little book and later introduced to finance along with the antithetic techniques to finance by Boyle [38] in 1977 with a substantial update by Boyle, Broadie and Glasserman [39] in 1997. See also Glasserman's book [96, Sect. 4.1] for more recent advances in finance.

Again, consider the target integral, returning back to n_x -dimensional space \mathcal{V} with density $\phi_{\mathbf{X}}(\mathbf{x})$,

$$\mu_f = \int_{\mathcal{V}} f(\mathbf{x})\phi_{\mathbf{X}}(\mathbf{x})d\mathbf{x}$$

and basic Monte Carlo estimate

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n f(\mathbf{X}_i),$$

converging to μ_f as $n \rightarrow \infty$ by the strong law of large numbers, where the set $\{\mathbf{X}_i\}$ of n IID sample points drawn are from the density $\phi_{\mathbf{X}}(\mathbf{x})$.

Next, through knowledge of the problem a simpler function $f^{(c)}(\mathbf{x})$ is found which significantly represents the variability of the target function $f(\mathbf{x})$ and can be used as a control (c) enabler, such that

$$\mu_f^{(c)} = \int_{\mathcal{V}} f^{(c)}(\mathbf{x})\phi_{\mathbf{X}}(\mathbf{x})d\mathbf{x}$$

is known or those value can be accurately approximated. Using the same IID set $\{\mathbf{X}_i\}$ of sample points, the basic Monte Carlo estimate is

$$\hat{\mu}_n^{(c)} = \frac{1}{n} \sum_{i=1}^n f^{(c)}(\mathbf{X}_i),$$

which is convergent to and is an unbiased estimate of $\mu_f^{(c)}$. The error $(\hat{\mu}_n^{(c)} - \mu_f^{(c)})$ will be used as a control variable to control the variance reduction of the basic

unbiased estimate $\hat{\mu}_n$ of μ_f by constructing a potentially improved control variate (*cv*) estimate,

$$\hat{\mu}_n^{(cv)}(\alpha) \equiv \hat{\mu}_n - \alpha \left(\hat{\mu}_n^{(c)} - \mu_f^{(c)} \right), \quad (10.76)$$

where α is a control parameter that will be optimized given the knowledge of the control function $f^{(c)}$. In particular, the partly known error $\left(\hat{\mu}_n^{(c)} - \mu_f^{(c)} \right)$ will be used to control the control variate estimate error $\left(\hat{\mu}_n^{(cv)}(\alpha) - \mu_f \right)$, noting from (10.76),

$$\mathbb{E} \left[\hat{\mu}_n^{(cv)}(\alpha) \right] = \mu_f - \alpha \left(\mu_f^{(c)} - \mu_f^{(c)} \right) = \mu_f,$$

that the unbiased estimation of μ_f is unchanged.

Upon examining the variance of the control variate estimate in terms of α following [96],

$$\begin{aligned} \text{Var} \left[\hat{\mu}_n^{(cv)}(\alpha) \right] &= \text{Var} \left[\hat{\mu}_n - \alpha \left(\hat{\mu}_n^{(c)} - \mu_f^{(c)} \right) \right] \\ &= \mathbb{E} \left[\left(\left(\hat{\mu}_n - \mu_f \right) - \alpha \left(\hat{\mu}_n^{(c)} - \mu_f^{(c)} \right) \right)^2 \right] \\ &= \text{Var}[\hat{\mu}_n] - 2\alpha \text{Cov} \left[\hat{\mu}_n, \hat{\mu}_n^{(c)} \right] + \alpha^2 \text{Var} \left[\hat{\mu}_n^{(c)} \right], \end{aligned} \quad (10.77)$$

a simple quadratic optimization in α produces an optimal control parameter,

$$\alpha^* = \frac{\text{Cov} \left[\hat{\mu}_n, \hat{\mu}_n^{(c)} \right]}{\text{Var} \left[\hat{\mu}_n^{(c)} \right]} = \frac{\rho_{\hat{\mu}_n, \hat{\mu}_n^{(c)}} \sqrt{\text{Var}[\hat{\mu}_n]}}{\sqrt{\text{Var} \left[\hat{\mu}_n^{(c)} \right]}}, \quad (10.78)$$

where the correlation function is

$$\rho_{X,Y} = \frac{\text{Cov}[X, Y]}{\sqrt{\text{Var}[X] \text{Var}[Y]}}.$$

Thus, the optimal control variate variance is

$$\text{Var} \left[\hat{\mu}_n^{(cv)}(\alpha^*) \right] = \text{Var}[\hat{\mu}_n] - \frac{\left(\text{Cov} \left[\hat{\mu}_n, \hat{\mu}_n^{(c)} \right] \right)^2}{\text{Var} \left[\hat{\mu}_n^{(c)} \right]} = \left(1 - \left(\rho_{\hat{\mu}_n, \hat{\mu}_n^{(c)}} \right)^2 \right) \text{Var}[\hat{\mu}_n], \quad (10.79)$$

so the absolute value of the correlation $|\rho_{\hat{\mu}_n, \hat{\mu}_n^{(c)}}|$ must be less than one for variance reduction. Note that Hammersley and Handscomb in their 1964 book [104, Sect. 5.5] do not use a control parameter (i.e., $\alpha \equiv 1$) and so require from (10.77) with $\alpha = 1$ that

$$2\text{Cov} \left[\hat{\mu}_n, \hat{\mu}_n^{(c)} \right] > \text{Var} \left[\hat{\mu}_n^{(c)} \right],$$

i.e., the covariance must be sufficiently positive, unlike (10.79). In fact, the optimal variance reduction ratio, from (10.79) and from the definition of VRR (10.47), is

$$\text{VRR}_{\hat{\mu}_n, \hat{\mu}_n^{(c)}}^* \equiv \frac{\text{Var}[\hat{\mu}_n]}{\text{Var} \left[\hat{\mu}_n^{(cv)}(\alpha^*) \right]} = \frac{1}{\left(1 - \left(\rho_{\hat{\mu}_n, \hat{\mu}_n^{(c)}} \right)^2 \right)}, \quad (10.80)$$

so the absolute value of the correlation $|\rho_{\hat{\mu}_n, \hat{\mu}_n^{(c)}}|$ should not only be less than one, but should be sufficiently close to one for significant variance reduction, in theory.

However, the exact statistics represented in the optimal parameter α^* in (10.78) and particularly the related optimal correlation $\rho_{\hat{\mu}_n, \hat{\mu}_n^{(c)}}$ are unknown. Hence, in practice, an estimate of α^* is needed, leading to the sample control parameter estimate of α^* ,

$$\begin{aligned} \hat{\alpha}_n &= \frac{\hat{c}_n^{(c)}}{(\hat{\sigma}_n^{(c)})^2} \equiv \frac{\frac{1}{n-1} \sum_{i=1}^n (f_i - \mu_f) (f_i^{(c)} - \mu_f^{(c)})}{\frac{1}{n-1} \sum_{j=1}^n (f_j^{(c)} - \mu_f^{(c)})^2} \\ &= \frac{\sum_{i=1}^n (f_i - \mu_f) (f_i^{(c)} - \mu_f^{(c)})}{\sum_{j=1}^n (f_j^{(c)} - \mu_f^{(c)})^2} \end{aligned} \tag{10.81}$$

and the corresponding estimated control variate Monte Carlo estimate

$$\hat{\mu}_n^{(cv)}(\hat{\alpha}_n) = \hat{\mu}_n - \hat{\alpha}_n (\hat{\mu}_n^{(c)} - \mu_f^{(c)}), \tag{10.82}$$

but introducing some bias particularly due to the approximate covariance $\hat{c}_n^{(c)}$ in (10.81). The bias (10.40) is given by

$$\beta_{\hat{\mu}_n^{(cv)}} = \mathbb{E} [\hat{\mu}_n^{(cv)}(\hat{\alpha}_n) - \mu_f] = -\mathbb{E} [\hat{\alpha}_n (\hat{\mu}_n^{(c)} - \mu_f^{(c)})], \tag{10.83}$$

which in general will be nonzero due to the nonlinear dependence of $\hat{\alpha}_n$.

Example 10.11. Control Variate Adjusted Jump-Diffusion Payoff:

Zhu and Hanson [285] further reduced the variance of the thetic-antithetic adjusted jump-factor Black-Scholes mentioned in Example 10.10 using the error in the thetic-antithetic adjusted jump-factor,

$$\Delta Y_i = 0.5 (e^{S_i} + e^{S_i^{(a)}}) - e^{\lambda t_f \bar{J}},$$

where the partial sums S_i and $S_i^{(a)}$ are given in (10.75), $\bar{J} \equiv \mathbb{E}[J(Q)]$ is the asset mean jump amplitude given in (10.36) of Example 10.3 and t_f is the option exercise time. The complex corrections to the bias $\beta_{\hat{\mu}_n^{(cv)}}$ in (10.83) are given and proven in [285, 284] along with other results. The combination of antithetic and control variate variance reduction techniques were easy to implement and were efficient in spite of the theoretical complexity and the combination was better than either one separately.

For more formation in depth on the control variate technique, see Boyle, Broadie and Glasserman [39] and Glasserman [96].

Another topic that is important but beyond the scope of this book is the quasi-Monte Carlo method which uses quasi-random or low-discrepancy number sequences which are more genuine deterministic sequences than the pseudo-random number sequences commonly used. Their generation is more complex generally than the pseudo-random sequences, but their big benefit is that convergence is between genuine order $\text{ord}(1/\sqrt{n})$ and $\text{ord}(1/n)$, so can outperform the variance reduction techniques just discussed. See Niederreiter [214] for the basic theoretical background to the quasi-Monte Carlo method. For more general information, see Glasserman [96, Chapt. 5] and Jäckel [147, Chapt. 8]. The Sobol' [248] quasi-random numbers seem to be the best overall performers in various measures as demonstrated in [96, Figs. 5.14-5.16] and [147, Figs. 8.2-8.9]. Also, see *Numerical Recipes* [226, Sect. 7.7] of Press et al. for the Sobol' sequence code `sobseq`.

Suggested References for Further Reading

- Applebaum, 2004 [12].
- Beichl and Sullivan, 2000 [23].
- Boyle, 1977 [38].
- Boyle, Broadie and Glasserman, 1997 [39].
- Cyganowski and Kloeden, 2000 [65].
- Cyganowski, Grüne and Kloeden, 2002 [64].
- Cyganowski, Kloeden and Ombach, 2002 [66].
- Glasserman, 2003 [96].
- Glynn and Whitt, 1992 [97].
- Hammersley and Handscomb, 1964 [104].
- D. Higham, 2001 [136] and 2004 [137].
- D. Higham and Kloeden, 2002 [140] and 2005 [141].
- D. Higham, Mao and Stuart, 2002 [143].
- Jäckel, 2002 [147].
- Kalos and Whitlock, 1986 [154].
- Kloeden and Platen, 1992 [161].

- Kloeden, Platen and Schurz, 1994 [162].
- Lepage, 1978 [179].
- Maghsoodi , 1996 [187].
- Maghsoodi and Harris, 1987 [188].
- Metropolis, A. Rosenbluth, M Rosenbluth, A. Teller, and E. Teller, 1953 [203].
- Metropolis and Ulam, 1949 [204].
- Niederreiter, 1992 [214].
- Press and Farrar, 1990 [225].
- Press, Teukolsky, Vetterling and Flannery, 2002 [226].
- Zhu, 2005 [284].
- Zhu and Hanson, 2005 [285].