SIAM REVIEW
Vol. 43, No. 3, pp. 525–546

An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations*

Desmond J. Higham†

Abstract. A practical and accessible introduction to numerical methods for stochastic differential equations is given. The reader is assumed to be familiar with Euler's method for deterministic differential equations and to have at least an intuitive feel for the concept of a random variable; however, no knowledge of advanced probability theory or stochastic processes is assumed. The article is built around 10 MATLAB programs, and the topics covered include stochastic integration, the Euler–Maruyama method, Milstein's method, strong and weak convergence, linear stability, and the stochastic chain rule.

Key words. Euler–Maruyama method, MATLAB, Milstein method, Monte Carlo, stochastic simulation, strong and weak convergence

AMS subject classifications. 65C30, 65C20

PII. S0036144500378302

1. Introduction.

Stochastic differential equation (SDE) models play a prominent role in a range of application areas, including biology, chemistry, epidemiology, mechanics, microelectronics, economics, and finance. A complete understanding of SDE theory requires familiarity with advanced probability and stochastic processes; picking up this material is likely to be daunting for a typical applied mathematics student. However, it is possible to appreciate the basics of how to simulate SDEs numerically with just a background knowledge of Euler's method for deterministic ordinary differential equations and an intuitive understanding of random variables. Furthermore, experience with numerical methods gives a useful first step toward the underlying theory of SDEs. Hence, in this article we explain how to apply simple numerical methods to an SDE and discuss concepts such as convergence and linear stability from a practical viewpoint. Our main target audience comprises advanced undergraduate and beginning postgraduate students.

We have aimed to keep the theory to a minimum. However, we rely on a basic assumption that the reader has at least a superficial feel for random variables, independence, expected values and variances, and, in particular, is familiar with the concept of a normally distributed random variable. Our numerical experiments use

*Received by the editors September 18, 2000; accepted for publication (in revised form) April 3, 2001; published electronically August 1, 2001.
http://www.siam.org/journals/sirev/43-3/37830.html

†Department of Mathematics, University of Strathclyde, Glasgow, G1 1XH, UK (djh@maths.strath.ac.uk). Supported by the Engineering and Physical Sciences Research Council of the UK under grant GR/M42206.

a Monte Carlo approach: random variables are simulated with a random number generator and expected values are approximated by computed averages.

The best way to learn is by example, so we have based this article around 10 MATLAB [3, 13] programs, using a philosophy similar to [14]. The website

`http://www.maths.strath.ac.uk/~aas96106/algfiles.html`

makes the programs downloadable. MATLAB is an ideal environment for this type of treatment, not least because of its high level random number generation and graphics facilities. The programs have been kept as short as reasonably possible and are designed to run quickly (less than 10 minutes on a modern desktop machine). To meet these requirements we found it necessary to "vectorize" the MATLAB code. We hope that the comment lines in the programs and our discussion of key features in the text will make the listings comprehensible to all readers who have some experience with a scientific programming language.

In the next section we introduce the idea of Brownian motion and compute discretized Brownian paths. In section 3 we experiment with the idea of integration with respect to Brownian motion and illustrate the difference between Itô and Stratonovich integrals. We describe in section 4 how the Euler–Maruyama method can be used to simulate an SDE. We introduce the concepts of strong and weak convergence in section 5 and verify numerically that Euler–Maruyama converges with strong order 1/2 and weak order 1. In section 6 we look at Milstein's method, which adds a correction to Euler–Maruyama in order to achieve strong order 1. In section 7 we introduce two distinct types of linear stability for the Euler–Maruyama method. In order to emphasize that stochastic calculus differs fundamentally from deterministic calculus, we quote and numerically confirm the stochastic chain rule in section 8. Section 9 concludes with a brief mention of some other important issues, many of which represent active research areas.

Rather than pepper the text with repeated citations, we will mention some key sources here. For those inspired to learn more about SDEs and their numerical solution we recommend [6] as a comprehensive reference that includes the necessary material on probability and stochastic processes. The review article [11] contains an up-to-date bibliography on numerical methods. Three other accessible references on SDEs are [1], [8], and [9], with the first two giving some discussion of numerical methods. Chapters 2 and 3 of [10] give a self-contained treatment of SDEs and their numerical solution that leads into applications in polymeric fluids. Underlying theory on Brownian motion and stochastic calculus is covered in depth in [5]. The material on linear stability in section 7 is based on [2] and [12].

2. Brownian Motion. A scalar *standard Brownian motion*, or *standard Wiener process*, over $[0, T]$ is a random variable $W(t)$ that depends continuously on $t \in [0, T]$ and satisfies the following three conditions.

1. $W(0) = 0$ (with probability 1).
2. For $0 \le s < t \le T$ the random variable given by the increment $W(t) - W(s)$ is normally distributed with mean zero and variance $t - s$; equivalently, $W(t) - W(s) \sim \sqrt{t-s}\, N(0,1)$, where $N(0,1)$ denotes a normally distributed random variable with zero mean and unit variance.
3. For $0 \le s < t < u < v \le T$ the increments $W(t) - W(s)$ and $W(v) - W(u)$ are independent.

For computational purposes it is useful to consider discretized Brownian motion, where $W(t)$ is specified at discrete t values. We thus set $\delta t = T/N$ for some positive integer N and let W_j denote $W(t_j)$ with $t_j = j\delta t$. Condition 1 says $W_0 = 0$ with

```
%BPATH1  Brownian path simulation

randn('state',100)           % set the state of randn
T = 1; N = 500; dt = T/N;
dW = zeros(1,N);             % preallocate arrays ...
W = zeros(1,N);              % for efficiency

dW(1) = sqrt(dt)*randn;      % first approximation outside the loop ...
W(1) = dW(1);                % since W(0) = 0 is not allowed
for j = 2:N
   dW(j) = sqrt(dt)*randn;   % general increment
   W(j) = W(j-1) + dW(j);
end

plot([0:dt:T],[0,W],'r-')    % plot W against t
xlabel('t','FontSize',16)
ylabel('W(t)','FontSize',16,'Rotation',0)
```

Listing 1 *M-file* `bpath1.m`.

probability 1, and conditions 2 and 3 tell us that

$$W_j = W_{j-1} + \mathrm{d}W_j, \quad j = 1, 2, \ldots, N, \tag{2.1}$$

where each $\mathrm{d}W_j$ is an independent random variable of the form $\sqrt{\delta t}N(0,1)$.

The MATLAB M-file `bpath1.m` in Listing 1 performs one simulation of discretized Brownian motion over $[0,1]$ with $N = 500$. Here, the random number generator `randn` is used—each call to `randn` produces an independent "pseudorandom" number from the $N(0,1)$ distribution. In order to make experiments repeatable, MATLAB allows the initial state of the random number generator to be set. We set the state, arbitrarily, to be `100` with the command `randn('state',100)`. Subsequent runs of `bpath1.m` would then produce the same output. Different simulations can be performed by resetting the state, e.g., to `randn('state',200)`. The numbers from `randn` are scaled by $\sqrt{\delta t}$ and used as increments in the `for` loop that creates the `1`-by-`N` array `W`. There is a minor inconvenience: MATLAB starts arrays from index `1` and not index `0`. Hence, we compute `W` as `W(1),W(2),...,W(N)` and then use `plot([0:dt:T],[0,W])` in order to include the initial value `W(0) = 0` in the picture. Figure 1 shows the result; note that for the purpose of visualization, the discrete data has been joined by straight lines. We will refer to an array `W` created by the algorithm in `bpath1` as a *discretized Brownian path*.

We can perform the same computation more elegantly and efficiently by replacing the `for` loop with higher level "vectorized" commands, as shown in `bpath2.m` in Listing 2. Here, we have supplied two arguments to the random number generator: `randn(1,N)` creates a `1`-by-`N` array of independent $N(0,1)$ samples. The function `cumsum` computes the cumulative sum of its argument, so the `j`th element of the `1`-by-`N` array `W` is `dW(1) + dW(2) + ⋯ + dW(j)`, as required. Avoiding `for` loops and thereby computing directly with arrays rather than individual components is the key to writing efficient MATLAB code [3, Chapter 20]. Some of the M-files in this article would be several orders of magnitude slower if written in nonvectorized form.

The M-file `bpath3.m` in Listing 3 produces Figure 2. Here, we evaluate the function $u(W(t)) = \exp(t + \frac{1}{2}W(t))$ along 1000 discretized Brownian paths. The average of $u(W(t))$ over these paths is plotted with a solid blue line. Five individual paths are also plotted using a dashed red line. The M-file `bpath3.m` is vectorized across paths; `dW` is an `M`-by-`N` array such that `dW(i,j)` gives the increment $\mathrm{d}W_j$ in (2.1) for

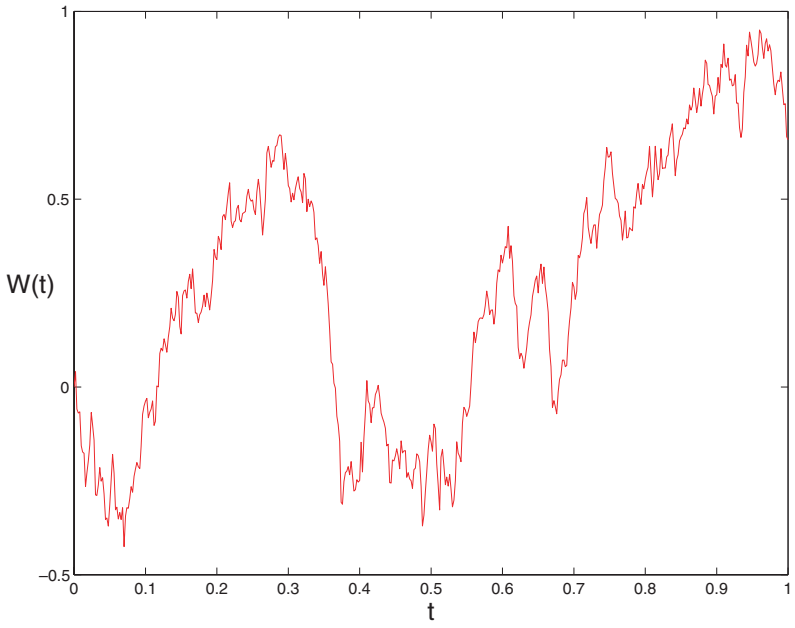

Fig. 1 *Discretized Brownian path from* `bpath1.m` *and* `bpath2.m`.

```
%BPATH2  Brownian path simulation: vectorized

randn('state',100)          % set the state of randn
T = 1; N = 500; dt = T/N;

dW = sqrt(dt)*randn(1,N);   % increments
W = cumsum(dW);             % cumulative sum

plot([0:dt:T],[0,W],'r-')   % plot W against t
xlabel('t','FontSize',16)
ylabel('W(t)','FontSize',16,'Rotation',0)
```

Listing 2 *M-file* `bpath2.m`.

the ith path. We use `cumsum(dW,2)` to form cumulative sums across the second (column) dimension. Hence, `W` is an `M`-by-`N` array whose ith row contains the ith path. We use `repmat(t,[M 1])` to produce an `M`-by-`N` array whose rows are all copies of `t`. The `M`-by-`N` array `U` then has ith row corresponding to $u(W(t))$ along the ith path. Forming `Umean = mean(U)` computes columnwise averages, so `Umean` is a `1`-by-`N` array whose jth entry is the sample average of $u(W(t_j))$.

We see in Figure 2 that although $u(W(t))$ is nonsmooth along individual paths, its sample average appears to be smooth. This can be established rigorously—the expected value of $u(W(t))$ turns out to be $\exp(9t/8)$. In `bpath3.m`, `averr` records the maximum discrepancy between the sample average and the exact expected value over all points t_j. We find that `averr = 0.0504`. Increasing the number of samples to 4000 reduces `averr` to `0.0268`.

```
%BPATH3  Function along a Brownian path

randn('state',100)                                  % set the state of randn
T = 1; N = 500; dt = T/N; t = [dt:dt:1];

M = 1000;                                           % M paths simultaneously
dW = sqrt(dt)*randn(M,N);                           % increments
W = cumsum(dW,2);                                   % cumulative sum
U = exp(repmat(t,[M 1]) + 0.5*W);
Umean = mean(U);
plot([0,t],[1,Umean],'b-'), hold on                 % plot mean over M paths
plot([0,t],[ones(5,1),U(1:5,:)],'r--'), hold off    % plot 5 individual paths
xlabel('t','FontSize',16)
ylabel('U(t)','FontSize',16,'Rotation',0,'HorizontalAlignment','right')
legend('mean of 1000 paths','5 individual paths',2)

averr = norm((Umean - exp(9*t/8)),'inf')            % sample error
```

Listing 3 *M-file* `bpath3.m`.

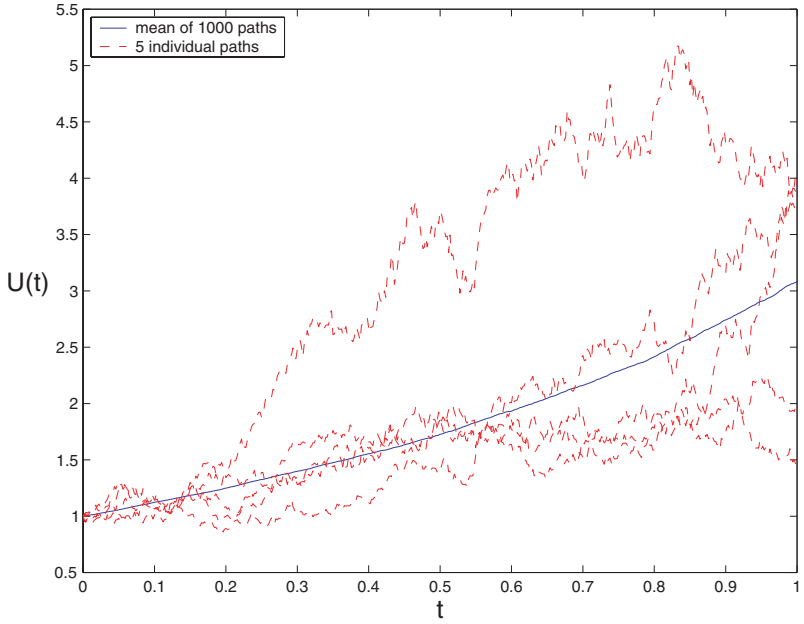

Fig. 2 *The function* $u(W(t))$ *averaged over* 1000 *discretized Brownian paths and along* 5 *individual paths, from* `bpath3.m`.

Note that $u(W(t))$ has the form (4.6) arising in section 4 as the solution to a linear SDE. In some applications the solution is required for a given path—a so-called pathwise or strong solution. As we will see in section 5, the ability of a method to compute strong solutions on average is quantified by the strong order of convergence. In other contexts, only expected value type information about the solution is of interest, which leads to the concept of the weak order of convergence.

```
%STINT  Approximate stochastic integrals
%
% Ito and Stratonovich integrals of W dW

randn('state',100)                      % set the state of randn
T = 1; N = 500; dt = T/N;

dW = sqrt(dt)*randn(1,N);               % increments
W = cumsum(dW);                         % cumulative sum

ito = sum([0,W(1:end-1)].*dW)
strat = sum((0.5*([0,W(1:end-1)]+W) + 0.5*sqrt(dt)*randn(1,N)).*dW)

itoerr = abs(ito - 0.5*(W(end)^2-T))
straterr = abs(strat - 0.5*W(end)^2)
```

Listing 4 *M-file* `stint.m`.

3. Stochastic Integrals. Given a suitable function h, the integral $\int_0^T h(t)dt$ may be approximated by the Riemann sum

$$\sum_{j=0}^{N-1} h(t_j)(t_{j+1} - t_j), \tag{3.1}$$

where the discrete points $t_j = j\delta t$ were introduced in section 2. Indeed, the integral may be *defined* by taking the limit $\delta t \to 0$ in (3.1). In a similar way, we may consider a sum of the form

$$\sum_{j=0}^{N-1} h(t_j)(W(t_{j+1}) - W(t_j)), \tag{3.2}$$

which, by analogy with (3.1), may be regarded as an approximation to a stochastic integral $\int_0^T h(t)dW(t)$. Here, we are integrating h with respect to Brownian motion.

In the M-file `stint.m` in Listing 4, we create a discretized Brownian path over $[0, 1]$ with $\delta t = 1/N = 1/500$ and form the sum (3.2) for the case where $h(t)$ is $W(t)$. The sum is computed as the variable `ito`. Here `.*` represents elementwise multiplication, so `[0,W(1:end-1)].*dW` represents the `1`-by-`N` array whose jth element is `W(j-1)*dW(j)`. The `sum` function is then used to perform the required summation, producing `ito = -0.2674`.

An alternative to (3.1) is given by

$$\sum_{j=0}^{N-1} h\left(\frac{t_j + t_{j+1}}{2}\right)(t_{j+1} - t_j), \tag{3.3}$$

which is also a Riemann sum approximation to $\int_0^T h(t)dt$. The corresponding alternative to (3.2) is

$$\sum_{j=0}^{N-1} h\left(\frac{t_j + t_{j+1}}{2}\right)(W(t_{j+1}) - W(t_j)). \tag{3.4}$$

In the case where $h(t) \equiv W(t)$, the sum (3.4) requires $W(t)$ to be evaluated at $t = (t_j + t_{j+1})/2$. It can be shown that forming $(W(t_j) + W(t_{j+1}))/2$ and adding an

independent $N(0, \Delta t/4)$ increment gives a value for $W((t_j + t_{j+1})/2)$ that maintains the three conditions listed at the start of section 2. Using this method, the sum (3.4) is evaluated in `stint.m` as `strat`, where we find that `strat = 0.2354`. Note that the two "stochastic Riemann sums" (3.2) and (3.4) give markedly different answers. Further experiments with smaller δt reveal that this mismatch does not go away as $\delta t \to 0$. This highlights a significant difference between deterministic and stochastic integration—in defining a stochastic integral as the limiting case of a Riemann sum, we must be precise about how the sum is formed. The "left-hand" sum (3.2) gives rise to what is known as the *Itô* integral, whereas the "midpoint" sum (3.4) produces the *Stratonovich* integral.[1]

It is possible to evaluate exactly the stochastic integrals that are approximated in `stint.m`. The Itô version is the limiting case of

$$\sum_{j=0}^{N-1} W(t_j)(W(t_{j+1}) - W(t_j)) = \tfrac{1}{2} \sum_{j=0}^{N-1} \left(W(t_{j+1})^2 - W(t_j)^2 - (W(t_{j+1}) - W(t_j))^2 \right)$$

$$= \tfrac{1}{2} \left(W(T)^2 - W(0)^2 - \sum_{j=0}^{N-1} (W(t_{j+1}) - W(t_j))^2 \right). \tag{3.5}$$

Now the term $\sum_{j=0}^{N-1} (W(t_{j+1}) - W(t_j))^2$ in (3.5) can be shown to have expected value T and variance of $O(\delta t)$. Hence, for small δt we expect this random variable to be close to the constant T. This argument can be made precise, leading to

$$\int_0^T W(t) dW(t) = \tfrac{1}{2} W(T)^2 - \tfrac{1}{2} T, \tag{3.6}$$

for the Itô integral. The Stratonovich version is the limiting case of

$$\sum_{j=0}^{N-1} \left(\frac{W(t_j) + W(t_{j+1})}{2} + \Delta Z_j \right) (W(t_{j+1}) - W(t_j)),$$

where each ΔZ_j is independent $N(0, \Delta t/4)$. This sum collapses to

$$\tfrac{1}{2} \left(W(T)^2 - W(0)^2 \right) + \sum_{j=0}^{N-1} \Delta Z_j (W(t_{j+1}) - W(t_j)),$$

in which the term $\sum_{j=0}^{N-1} \Delta Z_j (W(t_{j+1}) - W(t_j))$ has expected value 0 and variance $O(\delta t)$. Thus, in place of (3.6) we have

$$\int_0^T W(t) dW(t) = \tfrac{1}{2} W(T)^2. \tag{3.7}$$

The quantities `itoerr` and `strateerr` in the M-file `stint.m` record the amount by which the Riemann sums `ito` and `strat` differ from their respective $\delta t \to 0$ limits (3.6) and (3.7). We find that `itoerr = 0.0158` and `strateerr = 0.0186`.

Itô and Stratonovich integrals both have their uses in mathematical modeling. In subsequent sections we define an SDE using the Itô version (a simple transformation converts from Itô to Stratonovich).

[1] Some authors prefer an almost equivalent definition for the Stratonovich integral based on the sum $\sum_{j=0}^{N-1} \frac{1}{2}(h(t_j) + h(t_{j+1}))(W(t_{j+1}) - W(t_j))$.

4. The Euler–Maruyama Method. A scalar, autonomous SDE can be written in integral form as

$$X(t) = X_0 + \int_0^t f(X(s))\,ds + \int_0^t g(X(s))\,dW(s), \quad 0 \le t \le T. \tag{4.1}$$

Here, f and g are scalar functions and the initial condition X_0 is a random variable. The second integral on the right-hand side of (4.1) is to be taken with respect to Brownian motion, as discussed in the previous section, and we assume that the Itô version is used. The solution $X(t)$ is a random variable for each t. We do not attempt to explain further what it means for $X(t)$ to be a solution to (4.1)—instead we define a numerical method for solving (4.1), and we may then regard the solution $X(t)$ as the random variable that arises when we take the zero stepsize limit in the numerical method.

It is usual to rewrite (4.1) in differential equation form as

$$dX(t) = f(X(t))dt + g(X(t))dW(t), \quad X(0) = X_0, \quad 0 \le t \le T. \tag{4.2}$$

This is nothing more than a compact way of saying that $X(t)$ solves (4.1). To keep with convention, we will emphasize the SDE form (4.2) rather than the integral form (4.1). (Note that we are not allowed to write $dW(t)/dt$, since Brownian motion is nowhere differentiable with probability 1.) If $g \equiv 0$ and X_0 is constant, then the problem becomes deterministic, and (4.2) reduces to the ordinary differential equation $dX(t)/dt = f(X(t))$, with $X(0) = X_0$.

To apply a numerical method to (4.2) over $[0, T]$, we first discretize the interval. Let $\Delta t = T/L$ for some positive integer L, and $\tau_j = j\Delta t$. Our numerical approximation to $X(\tau_j)$ will be denoted X_j. The Euler–Maruyama (EM) method takes the form

$$X_j = X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})\left(W(\tau_j) - W(\tau_{j-1})\right), \quad j = 1, 2, \ldots, L. \tag{4.3}$$

To understand where (4.3) comes from, notice from the integral form (4.1) that

$$X(\tau_j) = X(\tau_{j-1}) + \int_{\tau_{j-1}}^{\tau_j} f(X(s))ds + \int_{\tau_{j-1}}^{\tau_j} g(X(s))dW(s). \tag{4.4}$$

Each of the three terms on the right-hand side of (4.3) approximates the corresponding term on the right-hand side of (4.4). We also note that in the deterministic case ($g \equiv 0$ and X_0 constant), (4.3) reduces to Euler's method.

In this article, we will compute our own discretized Brownian paths and use them to generate the increments $W(\tau_j) - W(\tau_{j-1})$ needed in (4.3). For convenience, we always choose the stepsize Δt for the numerical method to be an integer multiple $R \ge 1$ of the increment δt for the Brownian path. This ensures that the set of points $\{t_j\}$ on which the discretized Brownian path is based contains the points $\{\tau_j\}$ at which the EM solution is computed. In some applications the Brownian path is specified as part of the problem data. If an analytical path is supplied, then arbitrarily small Δt can be used.

We will apply the EM method to the linear SDE

$$dX(t) = \lambda X(t)dt + \mu X(t)dW(t), \quad X(0) = X_0, \tag{4.5}$$

where λ and μ are real constants; so $f(X) = \lambda X$ and $g(X) = \mu X$ in (4.2). This SDE arises, for example, as an asset price model in financial mathematics [4]. (Indeed, the

```
%EM  Euler-Maruyama method on linear SDE
%
% SDE is  dX = lambda*X dt + mu*X dW,   X(0) = Xzero,
%      where lambda = 2, mu = 1 and Xzero = 1.
%
% Discretized Brownian path over [0,1] has dt = 2^(-8).
% Euler-Maruyama uses timestep R*dt.

randn('state',100)
lambda = 2; mu = 1; Xzero = 1;    % problem parameters
T = 1; N = 2^8; dt = 1/N;
dW = sqrt(dt)*randn(1,N);         % Brownian increments
W = cumsum(dW);                   % discretized Brownian path

Xtrue = Xzero*exp((lambda-0.5*mu^2)*([dt:dt:T])+mu*W);
plot([0:dt:T],[Xzero,Xtrue],'m-'), hold on

R = 4; Dt = R*dt; L = N/R;        % L EM steps of size Dt = R*dt
Xem = zeros(1,L);                 % preallocate for efficiency
Xtemp = Xzero;
for j = 1:L
   Winc = sum(dW(R*(j-1)+1:R*j));
   Xtemp = Xtemp + Dt*lambda*Xtemp + mu*Xtemp*Winc;
   Xem(j) = Xtemp;
end

plot([0:Dt:T],[Xzero,Xem],'r--*'), hold off
xlabel('t','FontSize',12)
ylabel('X','FontSize',16,'Rotation',0,'HorizontalAlignment','right')

emerr = abs(Xem(end)-Xtrue(end))
```

Listing 5 *M-file* `em.m`.

well-known Black–Scholes partial differential equation can be derived from (4.5).) It is known (see, for example, [8, p. 105]) that the exact solution to this SDE is

$$X(t) = X(0)\exp\left((\lambda - \tfrac{1}{2}\mu^2)t + \mu W(t)\right). \tag{4.6}$$

In the M-file `em.m` in Listing 5 we consider (4.5) with $\lambda = 2$, $\mu = 1$, and $X_0 = 1$ (constant). We compute a discretized Brownian path over $[0, 1]$ with $\delta t = 2^{-8}$ and evaluate the solution in (4.6) as `Xtrue`. This is plotted with a solid magenta line in Figure 3. We then apply EM using a stepsize $\Delta t = R\delta t$, with $R = 4$. On a general step the EM method (4.3) requires the increment $W(\tau_j) - W(\tau_{j-1})$, which is given by

$$W(\tau_j) - W(\tau_{j-1}) = W(jR\delta t) - W((j-1)R\delta t) = \sum_{k=jR-R+1}^{jR} \mathrm{d}W_k.$$

In `em.m` this quantity appears as `Winc = sum(dW(R*(j-1)+1:R*j))`. The `1`-by-`L` array `Xem` stores the EM solution, which is plotted in Figure 3 as red asterisks connected with dashed lines. The discrepancy between the exact solution and the EM solution at the endpoint $t = T$, computed as `emerr`, was found to be `0.6907`. Taking $\Delta t = R\delta t$ with smaller R values of 2 and 1 produced endpoint errors of `0.1595` and `0.0821`, respectively.

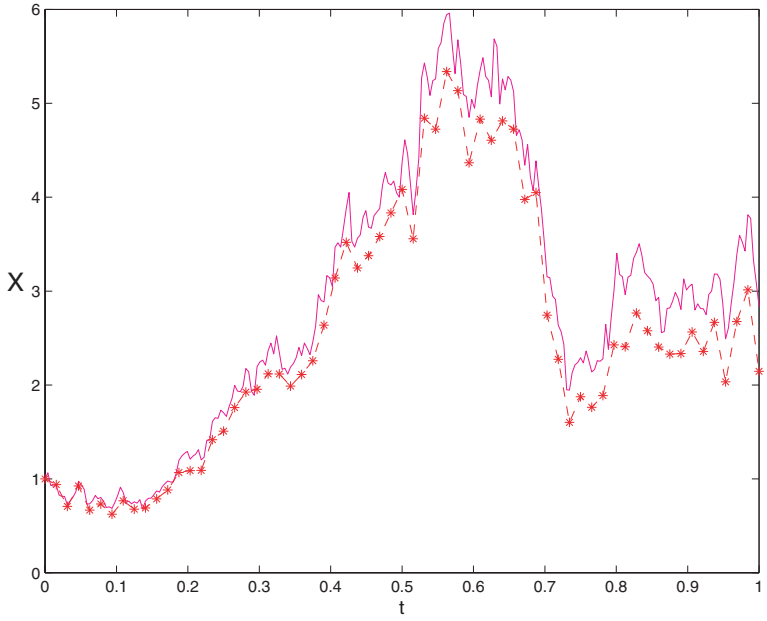

Fig. 3 *True solution and EM approximation, from* `em.m`.

5. Strong and Weak Convergence of the EM Method. In the example above with `em.m` the EM solution matches the true solution more closely as Δt is decreased—convergence seems to take place. Keeping in mind that $X(\tau_n)$ and X_n are random variables, in order to make the notion of convergence precise we must decide how to measure their difference. Using $\mathbb{E}\,|X_n - X(\tau_n)|$, where $\mathbb{E}$ denotes the expected value, leads to the concept of strong convergence. A method is said to have *strong order of convergence* equal to γ if there exists a constant C such that

$$\mathbb{E}\,|X_n - X(\tau)| \leq C\Delta t^{\gamma} \tag{5.1}$$

for any fixed $\tau = n\Delta t \in [0, T]$ and Δt sufficiently small. If f and g satisfy appropriate conditions, it can be shown that EM has strong order of convergence $\gamma = \frac{1}{2}$. Note that this marks a departure from the deterministic setting—if $g \equiv 0$ and X_0 is constant, then the expected value can be deleted from the left-hand side of (5.1) and the inequality is true with $\gamma = 1$.

In our numerical tests, we will focus on the error at the endpoint $t = T$, so we let

$$e_{\Delta t}^{\text{strong}} := \mathbb{E}\,|X_L - X(T)|, \quad \text{where } L\Delta t = T, \tag{5.2}$$

denote the EM endpoint error in this strong sense. If the bound (5.1) holds with $\gamma = \frac{1}{2}$ at any fixed point in $[0, T]$, then it certainly holds at the endpoint, so we have

$$e_{\Delta t}^{\text{strong}} \leq C\Delta t^{\frac{1}{2}} \tag{5.3}$$

for sufficiently small Δt.

The M-file `emstrong.m` in Listing 6 looks at the strong convergence of EM for the SDE (4.5) using the same λ, μ, and X_0 as in `em.m`. We compute 1000 different discretized Brownian paths over $[0, 1]$ with $\delta t = 2^{-9}$. For each path, EM is applied

```
%EMSTRONG  Test strong convergence of Euler-Maruyama
%
% Solves    dX = lambda*X dt + mu*X dW,   X(0) = Xzero,
%         where lambda = 2, mu = 1 and Xzer0 = 1.
%
% Discretized Brownian path over [0,1] has dt = 2^(-9).
% E-M uses 5 different timesteps: 16dt, 8dt, 4dt, 2dt, dt.
% Examine strong convergence at T=1:  E | X_L - X(T) |.

randn('state',100)
lambda = 2; mu = 1; Xzero = 1;    % problem parameters
T = 1; N = 2^9; dt = T/N;         %
M = 1000;                         % number of paths sampled

Xerr = zeros(M,5);                % preallocate array
for s = 1:M,                      % sample over discrete Brownian paths
    dW = sqrt(dt)*randn(1,N);     % Brownian increments
    W = cumsum(dW);               % discrete Brownian path
    Xtrue = Xzero*exp((lambda-0.5*mu^2)+mu*W(end));
    for p = 1:5
        R = 2^(p-1); Dt = R*dt; L = N/R;     % L Euler steps of size Dt = R*dt
        Xtemp = Xzero;
        for j = 1:L
             Winc = sum(dW(R*(j-1)+1:R*j));
             Xtemp = Xtemp + Dt*lambda*Xtemp + mu*Xtemp*Winc;
        end
        Xerr(s,p) = abs(Xtemp - Xtrue);      % store the error at t = 1
    end
end

Dtvals = dt*(2.^([0:4]));
subplot(221)                                  % top LH picture
loglog(Dtvals,mean(Xerr),'b*-'), hold on
loglog(Dtvals,(Dtvals.^(.5)),'r--'), hold off % reference slope of 1/2
axis([1e-3 1e-1 1e-4 1])
xlabel('\Delta t'), ylabel('Sample average of | X(T) - X_L |')
title('emstrong.m','FontSize',10)

%%%% Least squares fit of error = C * Dt^q %%%%
A = [ones(5,1), log(Dtvals)']; rhs = log(mean(Xerr)');
sol = A\rhs; q = sol(2)
resid = norm(A*sol - rhs)
```

Listing 6 *M-file* `emstrong.m`.

with 5 different stepsizes: $\Delta t = 2^{p-1}\delta t$ for $1 \le p \le 5$. The endpoint error in the sth sample path for the pth stepsize is stored in `Xerr(s,p)`; so `Xerr` is a `1000`-by-`5` array. The function `mean` is then used to average over all sample paths: forming `mean(Xerr)` produces a `1`-by-`5` array where each column of `Xerr` is replaced by its mean. Hence, the pth element of `mean(Xerr)` is an approximation to $e^{\text{strong}}_{\Delta t}$ for $\Delta t = 2^{p-1}\delta t$.

If the inequality (5.3) holds with approximate equality, then, taking logs,

$$\log e^{\text{strong}}_{\Delta t} \approx \log C + \tfrac{1}{2}\log \Delta t. \tag{5.4}$$

The command `loglog(Dtvals,mean(Xerr),'b*-')` in `emstrong.m` plots our approximation to $e^{\text{strong}}_{\Delta t}$ against Δt on a log-log scale. This produces the blue asterisks connected with solid lines in the upper left-hand plot of Figure 4. For reference, a dashed red line of slope one-half is added. We see that the slopes of the two curves appear to match well, suggesting that (5.4) is valid. We test this further by assuming

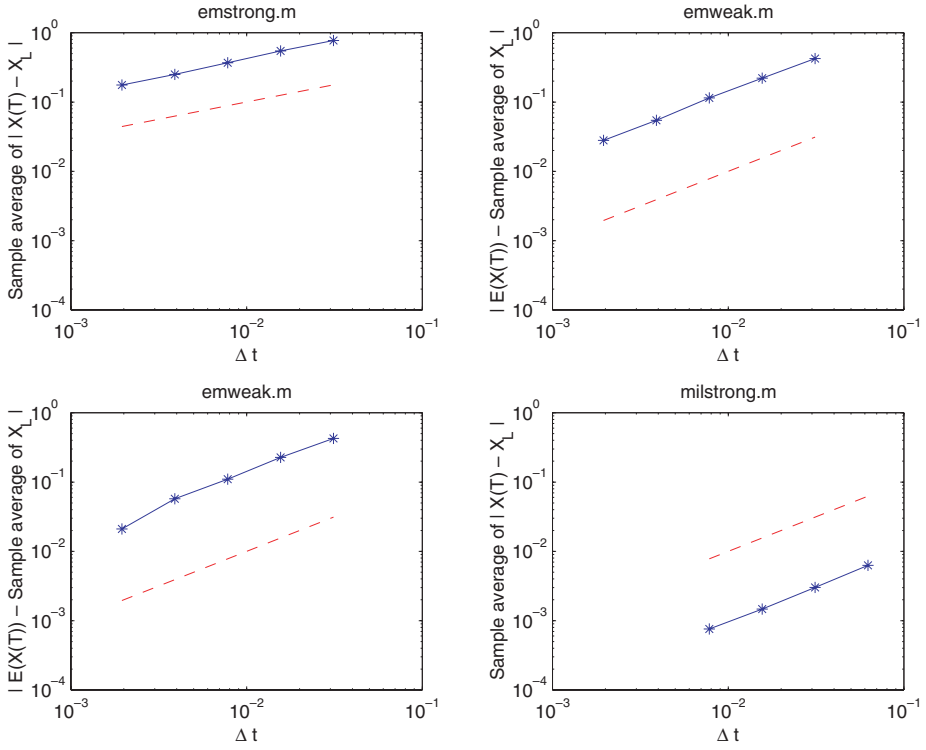

Fig. 4 *Strong and weak error plots: dashed red line is the appropriate reference slope in each case. Top left and right are for EM, bottom left is for weak EM, and bottom right is for Milstein.*

that a power law relation $e_{\Delta t}^{\text{strong}} = C\Delta t^q$ exists for some constants C and q, so that $\log e_{\Delta t}^{\text{strong}} = \log C + q \log \Delta t$. A least squares fit for $\log C$ and q is computed at the end of `emstrong`, producing the value `0.5384` for `q` with a least squares residual of `0.0266`. Hence, our results are consistent with a strong order of convergence equal to one-half.

While monitoring the error $e_{\Delta t}^{\text{strong}}$ in `emstrong.m`, we are implicitly assuming that a number of other sources of error are negligible, including the following.

Sampling error: the error arising from approximating an expected value by a sampled mean.

Random number bias: inherent errors in the random number generator.

Rounding error: floating point roundoff errors.

For a typical computation the sampling error is likely to be the most significant of these three. In preparing the programs in this article we found that some experimentation was required to make the number of samples sufficiently large and the timestep sufficiently small for the predicted orders of convergence to be observable. (The sampling error decays like $1/\sqrt{M}$, where M is the number of sample paths used.) A study in [7] indicates that as Δt decreases, lack of independence in the samples from a random number generator typically degrades the computation before rounding errors become significant.

Although the definition of strong convergence (5.1) involves an expected value, it has implications for individual simulations. The Markov inequality says that if a

random variable X has finite expected value, then for any $a > 0$ the probability that $|X| \geq a$ is bounded above by $(\mathbb{E}\,|X|)/a$, that is,

$$\mathbb{P}(|X| \geq a) \leq \frac{\mathbb{E}\,|X|}{a}.$$

Hence, taking $a = \Delta t^{1/4}$, we see that a consequence of EM's strong convergence of order $\gamma = \frac{1}{2}$ is

$$\mathbb{P}(|X_n - X(\tau)| \geq \Delta t^{1/4}) \leq C\Delta t^{1/4},$$

or, equivalently,

$$\mathbb{P}(|X_n - X(\tau)| < \Delta t^{1/4}) \geq 1 - C\Delta t^{1/4}.$$

This shows that the error at a fixed point in $[0, T]$ is small with probability close to 1.

The strong order of convergence (5.1) measures the rate at which the "mean of the error" decays as $\Delta t \to 0$. A less demanding alternative is to measure the rate of decay of the "error of the means." This leads to the concept of weak convergence. A method is said to have *weak order of convergence* equal to γ if there exists a constant C such that for all functions p in some class

$$|\mathbb{E}\,p(X_n) - \mathbb{E}\,p(X(\tau))| \leq C\Delta t^{\gamma} \tag{5.5}$$

at any fixed $\tau = n\Delta t \in [0, T]$ and Δt sufficiently small. Typically, the functions p allowed in (5.5) must satisfy smoothness and polynomial growth conditions. We will focus on the case where p is the identity function. For appropriate f and g it can be shown that EM has weak order of convergence $\gamma = 1$.

Mimicking our strong convergence tests, we let

$$e_{\Delta t}^{\text{weak}} := |\mathbb{E}\,X_L - \mathbb{E}\,X(T)|, \quad \text{where } L\Delta t = T, \tag{5.6}$$

denote the weak endpoint error in EM. So (5.5) for $p(X) \equiv X$ with $\gamma = 1$ immediately implies that

$$e_{\Delta t}^{\text{weak}} \leq C\Delta t \tag{5.7}$$

for sufficiently small Δt.

We examine the weak convergence of EM in the M-file `emweak.m` in Listing 7. Here we solve (4.5) over $[0, 1]$ for $\lambda = 2$, $\mu = 0.1$, and $X_0 = 1$. We sample over 50000 discretized Brownian paths and use five stepsizes $\Delta t = 2^{p-10}$ for $1 \leq p \leq 5$ in EM. This code has one extra level of vectorization compared to `emstrong`—we compute simultaneously with all 50000 paths. This improves the execution time at the expense of extra storage requirements. To compensate, we have used different paths for each Δt so that only the current increments, rather than the complete paths, need to be stored. Further, we choose the path increment $\delta t = \Delta t$ for extra efficiency. The sample average approximations to $\mathbb{E}\,X_L$ are stored in `Xem`. It follows from (4.6) that $\mathbb{E}\,X(T) = e^{\lambda T}$ for the true solution and `Xerr` stores the corresponding weak endpoint error for each Δt. The upper right-hand plot of Figure 4 shows how the weak error varies with Δt on a log-log scale. A dashed red reference line of slope one is added. It seems that (5.7) holds with approximate equality. As in `emstrong.m` we do a least squares power law fit that gives `q = 0.9858` and `resid = 0.0508`, confirming this behavior.

```
%EMWEAK  Test weak convergence of Euler-Maruyama
%
% Solves    dX = lambda*X dt + mu*X dW,   X(0) = Xzero,
%        where lambda = 2, mu = 1 and Xzer0 = 1.
%
% E-M uses 5 different timesteps: 2^(p-10),  p = 1,2,3,4,5.
% Examine weak convergence at T=1:   | E (X_L) - E (X(T)) |.
%
% Different paths are used for each E-M timestep.
% Code is vectorized over paths.
%
% Uncommenting the line indicated below gives the weak E-M method.

randn('state',100);
lambda = 2; mu = 0.1; Xzero = 1; T = 1;  % problem parameters
M = 50000;                               % number of paths sampled

Xem = zeros(5,1);                        % preallocate arrays
for p = 1:5                              % take various Euler timesteps
       Dt = 2^(p-10); L = T/Dt;          % L Euler steps of size Dt
       Xtemp = Xzero*ones(M,1);
       for j = 1:L
           Winc = sqrt(Dt)*randn(M,1);
           % Winc = sqrt(Dt)*sign(randn(M,1)); %% use for weak E-M %%
           Xtemp = Xtemp + Dt*lambda*Xtemp + mu*Xtemp.*Winc;
       end
       Xem(p) = mean(Xtemp);
end
Xerr = abs(Xem - exp(lambda));

Dtvals = 2.^([1:5]-10);
subplot(222)                             % top RH picture
loglog(Dtvals,Xerr,'b*-'), hold on
loglog(Dtvals,Dtvals,'r--'), hold off    % reference slope of 1
axis([1e-3 1e-1 1e-4 1])
xlabel('\Delta t'), ylabel('| E(X(T)) - Sample average of X_L |')
title('emweak.m','FontSize',10)

%%%% Least squares fit of error = C * dt^q %%%%
A = [ones(p,1), log(Dtvals)']; rhs = log(Xerr);
sol = A\rhs; q = sol(2)
resid = norm(A*sol - rhs)
```

Listing 7 *M-file* `emweak.m`.

It is worth emphasizing that for the computations in `emweak.m`, we used different paths for each stepsize Δt. This is perfectly reasonable. Weak convergence concerns only the mean of the solution, and so we are free to use any $\sqrt{\Delta t}\mathrm{N}(0,1)$ sample for the increment $W(\tau_j) - W(\tau_{j-1})$ in (4.3) on any step. In fact, the order of weak convergence is maintained if the increment is replaced by an independent two-point random variable $\sqrt{\Delta t}V_j$, where V_j takes the values $+1$ and -1 with equal probability. (Note that $\sqrt{\Delta t}V_j$ has the same mean and variance as $\sqrt{\Delta t}\mathrm{N}(0,1)$.) Replacing the Brownian increment by $\sqrt{\Delta t}V_j$ in this way leads to the weak Euler–Maruyama (WEM) method, which has weak order of convergence $\gamma = 1$, but, since it uses no pathwise information, offers no strong convergence. The motivation behind WEM is that random number generators that sample from V_j can be made more efficient than those that sample from N(0,1). In the M-file `emweak.m` we have included the comment line

```
% Winc = sqrt(Dt)*sign(randn(M,1)); %% use for weak E-M %%
```

Deleting the leading `%` character, and hence uncommenting the line, implements WEM, since `sign(randn(M,1))` is equally likely to be `+1` or `-1`. (Clearly, because we are using the built-in normal random number generator, there is no efficiency gain in this case.) The resulting error graph is displayed as the lower-left hand picture in Figure 4. A least squares power law fit gives `q = 1.0671` and `resid = 0.2096`.

6. Milstein's Higher Order Method. We saw in the previous section that EM has strong order of convergence $\gamma = \frac{1}{2}$ in (5.1), whereas the underlying deterministic Euler method converges with classical order 1. It is possible to raise the strong order of EM to 1 by adding a correction to the stochastic increment, giving Milstein's method. The correction arises because the traditional Taylor expansion must be modified in the case of Itô calculus. A so-called Itô–Taylor expansion can be formed by applying Itô's result, which is a fundamental tool of stochastic calculus. Truncating the Itô–Taylor expansion at an appropriate point produces Milstein's method for the SDE (4.2):

$$\begin{aligned} X_j = X_{j-1} &+ \Delta t f(X_{j-1}) + g(X_{j-1})\left(W(\tau_j) - W(\tau_{j-1})\right) \\ &+ \tfrac{1}{2} g(X_{j-1}) g'(X_{j-1}) \left((W(\tau_j) - W(\tau_{j-1}))^2 - \Delta t \right), \quad j = 1, 2, \ldots, L. \end{aligned} \tag{6.1}$$

The M-file `milstrong.m` in Listing 8 applies Milstein's method to the SDE

$$dX(t) = rX(t)(K - X(t))dt + \beta X(t) dW(t), \quad X(0) = X_0, \tag{6.2}$$

which arises in population dynamics [9]. Here, r, K, and β are constants. We take $r = 2$, $K = 1$, $\beta = 0.25$, and $X_0 = 0.5$ (constant) and use discretized Brownian paths over $[0, 1]$ with $\delta t = 2^{-11}$. The solution to (6.2) can be written as a closed-form expression involving a stochastic integral. For simplicity, we take the Milstein solution with $\Delta t = \delta t$ to be a good approximation of the exact solution and compare this with the Milstein approximation using $\Delta t = 128\delta t$, $\Delta t = 64\delta t$, $\Delta t = 32\delta t$, and $\Delta t = 16\delta t$ over 500 sample paths. We have added one more level of vectorization compared with the `emstrong.m` file—rather than using a `for` loop to range over sample paths, we compute with all paths simultaneously. We set up `dW` as an `M`-by-`N` array in which `dW(s,j)` is the jth increment for the sth path. The required increments for Milstein with timestep `R(p)*dt` are

```
Winc = sum(dW(:,R(p)*(j-1)+1:R(p)*j),2);
```

This takes the sub-array consisting of all rows of `dW` and columns `R(p)*(j-1)+1` to `R(p)*j` and sums over the second (column) dimension. The result is an `M`-by-`1` array whose jth entry is the sum of the entries in row i of `dW` between columns `R(p)*(j-1)+1` and `R(p)*j`. The `M`-by-`5` array `Xmil` stores all numerical solutions for the `M` paths and 5 stepsizes. The resulting log-log error plot is shown as the lower right-hand picture in Figure 4 along with a reference line of slope 1. The least-squares power law fit gives `q = 1.0184` and `resid = 0.0350`.

7. Linear Stability. The concepts of strong and weak convergence concern the accuracy of a numerical method over a finite interval $[0, T]$ for small stepsizes Δt. However, in many applications the long-term, $t \to \infty$, behavior of an SDE is of interest. Convergence bounds of the form (5.1) or (5.5) are not relevant in this context since, generally, the constant C grows unboundedly with T. For deterministic ODE methods, a large body of *stability theory* has been developed that gives insight into the behavior of numerical methods in the Δt fixed, $t_j \to \infty$ limit. Typically, a numerical method is applied to a class of problems with some qualitative feature,

```
%MILSTRONG  Test strong convergence of Milstein: vectorized
%
% Solves   dX = r*X*(K-X) dt + beta*X dW,  X(0) = Xzero,
%        where r = 2, K= 1, beta = 1 and Xzero = 0.5.
%
% Discretized Brownian path over [0,1] has dt = 2^(-11).
% Milstein uses timesteps 128*dt, 64*dt, 32*dt, 16*dt (also dt for reference).
%
% Examines strong convergence at T=1:  E | X_L - X(T) |.
% Code is vectorized: all paths computed simultaneously.

rand('state',100)
r = 2; K = 1; beta = 0.25; Xzero = 0.5;   % problem parameters
T = 1; N = 2^(11); dt = T/N;              %
M = 500;                                  % number of paths sampled
R = [1; 16; 32; 64; 128];                 % Milstein stepsizes are R*dt

dW = sqrt(dt)*randn(M,N);                 % Brownian increments
Xmil = zeros(M,5);                        % preallocate array
for p = 1:5
     Dt = R(p)*dt; L = N/R(p);            % L timesteps of size Dt = R dt
     Xtemp = Xzero*ones(M,1);
     for j = 1:L
          Winc = sum(dW(:,R(p)*(j-1)+1:R(p)*j),2);
          Xtemp = Xtemp + Dt*r*Xtemp.*(K-Xtemp) + beta*Xtemp.*Winc ...
              + 0.5*beta^2*Xtemp.*(Winc.^2 - Dt);
     end
     Xmil(:,p) = Xtemp;  % store Milstein solution at t =1
end

Xref = Xmil(:,1);                              % Reference solution
Xerr = abs(Xmil(:,2:5) - repmat(Xref,1,4));    % Error in each path
mean(Xerr);                                    % Mean pathwise erorrs
Dtvals = dt*R(2:5);                            % Milstein timesteps used

subplot(224)                                   % lower RH picture
loglog(Dtvals,mean(Xerr),'b*-'), hold on
loglog(Dtvals,Dtvals,'r--'), hold off          % reference slope of 1
axis([1e-3 1e-1 1e-4 1])
xlabel('\Delta t')
ylabel('Sample average of | X(T) - X_L |')
title('milstrong.m','FontSize',10)

%%%% Least squares fit of error = C * Dt^q %%%%
A = [ones(4,1), log(Dtvals)]; rhs = log(mean(Xerr)');
sol = A\rhs; q = sol(2)
resid = norm(A*sol - rhs)
```

Listing 8 *M-file* `milstrong.m`.

and the ability of the method to reproduce this feature is analyzed. Although a wide variety of problem classes have been analyzed, the simplest, and perhaps the most revealing, is the linear test equation $dX/dt = \lambda X$, where $\lambda \in \mathbb{C}$ is a constant parameter. For SDEs, it is possible to develop an analogous linear stability theory, as we now indicate.

We return to the linear SDE (4.5), with the parameters λ and μ allowed to be complex. In the case where $\mu = 0$ and X_0 is constant, (4.5) reduces to the deterministic linear test equation, which has solutions of the form $X_0 \exp(\lambda t)$. If we use the term *stable* to mean that $\lim_{t\to\infty} X(t) = 0$ for any X_0, then we see

that stability is characterized by $\Re\{\lambda\} < 0$. In order to generalize this idea to the SDE case, we must be more precise about what we mean by "$\lim_{t\to\infty} X(t) = 0$"—random variables are infinite-dimensional objects and hence norms are not equivalent in general. We will consider the two most common measures of stability: mean-square and asymptotic. Assuming that $X_0 \neq 0$ with probability 1, solutions of (4.5) satisfy

$$\lim_{t\to\infty} \mathbb{E}\, X(t)^2 = 0 \Leftrightarrow \Re\{\lambda\} + \tfrac{1}{2}|\mu|^2 < 0, \tag{7.1}$$

$$\lim_{t\to\infty} |X(t)| = 0, \text{ with probability } 1 \Leftrightarrow \Re\{\lambda - \tfrac{1}{2}\mu^2\} < 0. \tag{7.2}$$

The left-hand side of (7.1) defines what is meant by *mean-square stability*. The right-hand side of (7.1) completely characterizes this property in terms of the parameters λ and μ. Similarly, (7.2) defines and characterizes *asymptotic stability*. Setting $\mu = 0$, the characterizations collapse to the same condition, $\Re\{\lambda\} < 0$, which, of course, arose for deterministic stability. It follows immediately from (7.1) and (7.2) that if (4.5) is mean-square stable, then it is automatically asymptotic stable, but not vice versa. Hence, on this test problem, mean-square stability is a more stringent requirement than asymptotic stability. Both stability definitions are useful in practice.

Now suppose that the parameters λ and μ are chosen so that the SDE (4.5) is stable in the mean-square or asymptotic sense. A natural question is then, "For what range of Δt is the EM solution stable in an analogous sense?" The mean-square version of this question is easy to analyze. Simple properties of the expected value show that

$$\lim_{j\to\infty} \mathbb{E}\, X_j^2 = 0 \Leftrightarrow |1 + \Delta t\lambda|^2 + \Delta t|\mu|^2 < 1 \tag{7.3}$$

for EM applied to (4.5). The asymptotic version of the question can be studied via the strong law of large numbers and the law of the iterated logarithm, leading to

$$\lim_{j\to\infty} |X_j| = 0, \text{ with probability } 1 \Leftrightarrow \mathbb{E}\, \log\left|1 + \Delta t\lambda + \sqrt{\Delta t}\mu N(0,1)\right| < 0. \tag{7.4}$$

These results are illustrated by the M-file `stab.m` in Listing 9. To test mean-square stability, we solve (4.5) with $X_0 = 1$ (constant) over $[0, 20]$ for two parameter sets. The first set has $\lambda = -3$ and $\mu = \sqrt{3}$. These values satisfy (7.1) and hence the problem is mean-square stable. We apply EM over 50000 discrete Brownian paths for three different stepsizes: $\Delta t = 1, 1/2, 1/4$. Only the third of these, $\Delta t = 1/4$, satisfies the right-hand side of (7.3). The upper picture in Figure 5 plots the sample average of X_j^2 against t_j. Note that the vertical axis is logarithmically scaled. In this picture the $\Delta t = 1$ and $\Delta t = 1/2$ curves increase with t, while the $\Delta t = 1/4$ curve decays toward to zero. Hence, this test correctly implies that for $\Delta t = 1, 1/2$ and $\Delta t = 1/4$, EM is unstable and stable, respectively, in the mean-square sense. However, the number of samples used (50000) is not sufficient to resolve the behavior fully; the three curves should be straight lines. This highlights the fact that simplistic sampling without further checks may lead to misleading conclusions.

To examine asymptotic stability, we use the parameter set $\lambda = 1/2$ and $\mu = \sqrt{6}$. It follows from (7.2) that the SDE is asymptotically stable (although, from (7.1), it is not mean-square stable). Since asymptotic stability concerns a probability 1 event, we apply EM over a single discrete Brownian path for $\Delta t = 1, 1/2, 1/4$, and because computing with a single path is cheap, we integrate over $[0, 500]$. It can be shown that only the smallest of these timesteps, $\Delta t = 1/4$, satisfies the condition (7.4)—this is

```
%STAB  Mean-square and asymptotic stability test for E-M
%
% SDE is  dX = lambda*X dt + mu*X dW,   X(0) = Xzero,
%      where lambda and mu are constants and Xzero = 1.

randn('state',100)
T = 20; M = 50000; Xzero = 1;
ltype = {'b-','r--','m-.'};             % linetypes for plot

subplot(211)  %%%%%%%%%%%% Mean Square %%%%%%%%%%%%
lambda = -3; mu = sqrt(3);              % problem parameters
for k = 1:3
    Dt = 2^(1-k);
    N = T/Dt;
    Xms = zeros(1,N); Xtemp = Xzero*ones(M,1);
    for j = 1:N
           Winc = sqrt(Dt)*randn(M,1);
           Xtemp = Xtemp + Dt*lambda*Xtemp + mu*Xtemp.*Winc;
           Xms(j) = mean(Xtemp.^2);     % mean-square estimate
    end
    semilogy([0:Dt:T],[Xzero,Xms],ltype{k},'Linewidth',2), hold on
end
legend('\Delta t = 1','\Delta t = 1/2','\Delta t = 1/4')
title('Mean-Square: \lambda = -3, \mu = \surd 3','FontSize',16)
ylabel('E[X^2]','FontSize',12), axis([0,T,1e-20,1e+20]), hold off

subplot(212)  %%%%% Asymptotic: a single path %%%%%%%
T = 500;
lambda = 0.5; mu = sqrt(6);             % problem parameters
for k = 1:3
    Dt = 2^(1-k);
    N = T/Dt;
    Xemabs = zeros(1,N); Xtemp = Xzero;
    for j = 1:N
           Winc = sqrt(Dt)*randn;
           Xtemp = Xtemp + Dt*lambda*Xtemp + mu*Xtemp*Winc;
           Xemabs(j) = abs(Xtemp);
    end
    semilogy([0:Dt:T],[Xzero,Xemabs],ltype{k},'Linewidth',2), hold on
end
legend('\Delta t = 1','\Delta t = 1/2','\Delta t = 1/4')
title('Single Path: \lambda = 1/2, \mu = \surd 6','FontSize',16)
ylabel('|X|','FontSize',12), axis([0,T,1e-50,1e+100]), hold off
```

Listing 9 *M-file* `stab.m`.

illustrated in Figure 6, which is discussed below. The lower picture in Figure 5 plots $|X_j|$ against t_j along the path. We see that only the $\Delta t = 1/4$ solution appears to decay to zero, in agreement with the theory.

Figure 6 offers further insight into these computations. Here we have plotted regions of stability for $\lambda, \mu \in \mathbb{R}$. The x-axis and y-axis represent $\Delta t\lambda$ and $\Delta t\mu^2$, respectively. In this notation, it follows from (7.1) that the SDE is mean-square stable for $y < -2x$ (horizontal magenta shading, marked SDE:ms) and asymptotically stable for $y > 2x$ (vertical green shading, marked SDE:as). The condition (7.3) for mean-square stability of EM requires y to be positive and lie beneath the parabola $-x(2+x)$. The parabola is shown as a solid red curve in the figure and the corresponding mean-square stability region for EM is marked EM:ms. The condition (7.4) that determines asymptotic stability of EM leads to the flattened-egg-shaped boundary in solid blue.

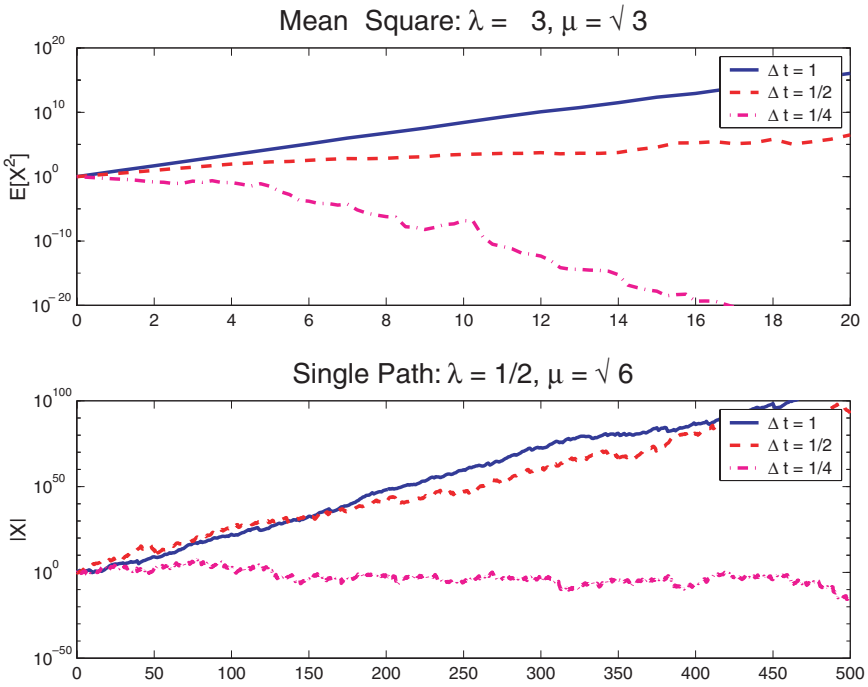

Fig. 5 *Mean-square and asymptotic stability tests, from* `stab.m`.

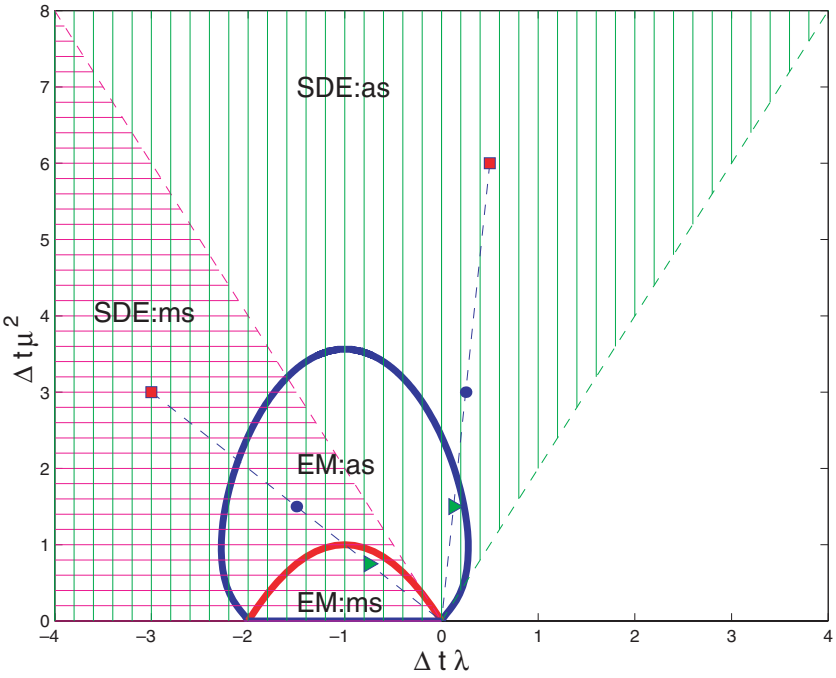

Fig. 6 *Mean-square and asymptotic stability regions.*

The resulting region is marked EM:as. To interpret the picture, notice that given values for λ and μ, the point $(x, y) = (\lambda, \mu^2)$ corresponds to the timestep $\Delta t = 1$, and then varying the stepsize Δt corresponds to moving along the ray that connects (λ, μ^2) with the origin. The parameter sets $(\lambda, \mu^2) = (-3, 3)$ and $(\lambda, \mu^2) = (1/2, 6)$ used by `stab.m` are marked as red squares. We see that the first set lies in SDE:ms and the second in SDE:as, but neither are stable for EM with $\Delta t = 1$. Reducing Δt from 1 to 1/2 takes us to the points marked with blue circles in the figure. Now the first set is in EM:as (but not EM:ms) and the second set remains outside EM:as. Reducing Δt further to the value 1/4 takes us to the points marked by green triangles. We see that the first parameter set is now in EM:ms and the second set in EM:as, so the stability property of the underlying SDE is recovered in both cases.

8. Stochastic Chain Rule. We saw in section 3 that there is more than one way to extend the concept of integration to a stochastic setting. In this section we briefly mention another fundamental difference between stochastic and deterministic calculus.

In the deterministic case, if $dX/dt = f(X)$ then, for any smooth function V, the chain rule says that

$$\frac{dV(X(t))}{dt} = \frac{dV(X(t))}{dX}\frac{d(X(t))}{dt} = \frac{dV(X(t))}{dX} f(X(t)). \tag{8.1}$$

Now, suppose that X satisfies the Itô SDE (4.2). What is the SDE analogue of (8.1) for $V(X)$? A reasonable guess is $dV = (dV/dX)dX$, so that, using (4.2),

$$dV(X(t)) = \frac{dV(X(t))}{dX}\left(f(X(t))dt + g(X(t))dW(t)\right). \tag{8.2}$$

However, a rigorous analysis using Itô's result reveals that an extra term arises[2] and the correct formulation is

$$dV(X(t)) = \frac{dV(X(t))}{dX}dX + \tfrac{1}{2}g(X(t))^2\frac{d^2V(X(t))}{dX^2}dt,$$

which, using (4.2), becomes

$$dV(X(t)) = \left(f(X(t))\frac{dV(X(t))}{dX} + \tfrac{1}{2}g(X(t))^2\frac{d^2V(X(t))}{dX^2}\right)dt + g(X(t))\frac{dV(X(t))}{dX}dW(t). \tag{8.3}$$

We will not attempt to prove, or even justify, (8.3). Instead we will perform a numerical experiment.

We consider the SDE

$$dX(t) = (\alpha - X(t))\,dt + \beta\sqrt{X(t)}dW(t), \quad X(0) = X_0, \tag{8.4}$$

where α and β are constant, positive parameters. This SDE is a mean-reverting square root process that models asset prices [8, Chapter 9]. It can be shown that if $X(0) > 0$ with probability 1, then this positivity is retained for all $t > 0$. Taking $V(X) = \sqrt{X}$, an application of (8.3) gives

$$dV(t) = \left(\frac{4\alpha - \beta^2}{8V(t)} - \tfrac{1}{2}V(t)\right)dt + \tfrac{1}{2}\beta dW(t). \tag{8.5}$$

[2] In fact, (8.2) turns out to be valid in the Stratonovich framework, but here we are using Itô calculus.

```
%CHAIN  Test stochastic Chain Rule
%
% Solve SDE for V(X) = sqrt(X) where X solves
%         dX = (alpha - X) dt + beta sqrt(X) dW,   X(0) = Xzero,
%      with alpha = 2, beta = 1 and Xzero = 1.
% Xem1 is Euler-Maruyama solution for X.
% Xem2 is Euler-Maruyama solution of SDE for V from Chain Rule.
% Hence, we compare sqrt(Xem1) and Xem2.
% Note: abs is used for safety inside sqrt, but has no effect in this case.

randn('state',100)
alpha = 2; beta = 1; T = 1; N = 200; dt = T/N;  % Problem parameters
Xzero = 1; Xzero2 = 1/sqrt(Xzero);              %

Dt = dt;                                        % EM steps of size Dt = dt
Xem1 = zeros(1,N); Xem2 = zeros(1,N);           % preallocate for efficiency
Xtemp1 = Xzero;  Xtemp2 = Xzero2;
for j = 1:N
   Winc = sqrt(dt)*randn;
   f1 = (alpha-Xtemp1);
   g1 = beta*sqrt(abs(Xtemp1));
   Xtemp1 = Xtemp1 + Dt*f1 + Winc*g1;
   Xem1(j) = Xtemp1;
   f2 = (4*alpha-beta^2)/(8*Xtemp2) - Xtemp2/2;
   g2 = beta/2;
   Xtemp2 = Xtemp2 + Dt*f2 + Winc*g2;
   Xem2(j) = Xtemp2;
end

plot([0:Dt:T],[sqrt([Xzero,abs(Xem1)])],'b-',[0:Dt:T],[Xzero,Xem2],'ro')
legend('Direct Solution','Solution via Chain Rule',2)
xlabel('t','FontSize',12)
ylabel('V(X)','FontSize',16,'Rotation',0,'HorizontalAlignment','right')

Xdiff = norm(sqrt(Xem1) - Xem2,'inf')
```

Listing 10 *M-file* `chain.m`.

In the M-file `chain.m` in Listing 10 we apply EM to the SDE (8.4) for $\alpha = 2$, $\beta = 1$, and $X_0 = 1$ (constant). We compute a discrete Brownian path over $[0, 1]$ with $\delta t = 1/200$ and use $\Delta t = \delta t$. The square root of the numerical solution is plotted with a solid blue line in Figure 7. We also solve the SDE (8.5) for V and plot the solution with a red "o" symbol. We observe that there is good agreement between the two solutions. This is confirmed by a check on the maximum discrepancy: `Xdiff = 0.0151`.

9. Further Issues. This article is designed to give readers a brief and practical introduction to the numerical simulation of SDEs with a minimum of technical detail. As a consequence, we have glossed over a vast amount of theory and left out many important topics. Consulting the references cited at the end of section 1 is a good way to fill in some of the gaps. In this final section, we hint at some of the topics omitted.

First, we have not discussed what conditions on f and g in (4.2) can be imposed in order to guarantee the existence of a unique solution to the SDE. Typical existence and uniqueness theorems (and convergence theorems for numerical methods) impose much more stringent constraints than their deterministic counterparts. Second, we have not mentioned the connection between SDEs and certain time-dependent partial differential equations. Third, we restricted attention to scalar problems—the EM

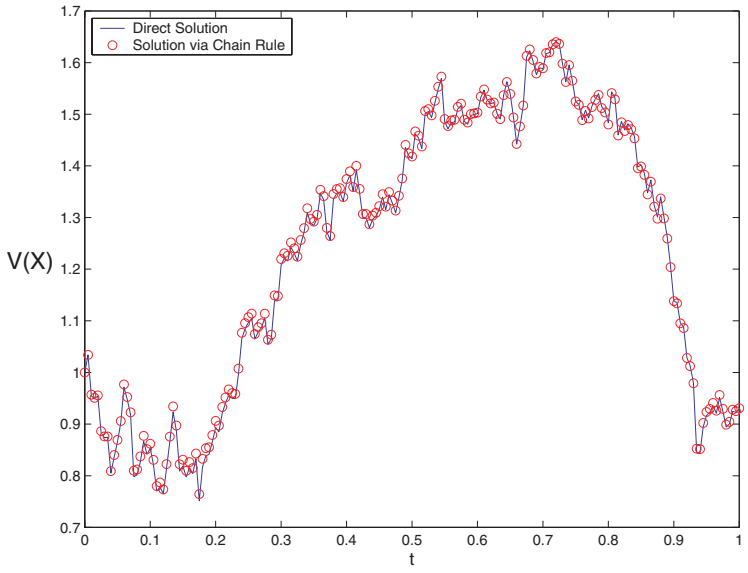

Fig. 7 *EM approximations of* $V(X(t)) = \sqrt{X(t)}$ *using* (8.4) *directly and the chain rule version* (8.5)*, from* `chain.m`.

method looks much the same when applied to an SDE system, but Milstein's method becomes more complicated. Research into numerical methods for SDEs is being actively pursued in a number of directions, including the construction of methods with high order of strong or weak convergence or improved stability, the design of variable timestep algorithms, and the analysis of long-term properties such as ergodicity for nonlinear problems. Pointers to the recent literature can be found in [11].

Acknowledgment. I thank Nick Higham, Andrew Stuart, and Thomas Gorm Theting for their valuable comments.

REFERENCES

[1] T. C. Gard, *Introduction to Stochastic Differential Equations*, Marcel Dekker, New York, 1988.
[2] D. J. Higham, *Mean-square and asymptotic stability of the stochastic theta method*, SIAM J. Numer. Anal., 38 (2000), pp. 753–769.
[3] D. J. Higham and N. J. Higham, *MATLAB Guide*, SIAM, Philadelphia, 2000.
[4] J. C. Hull, *Options, Futures, and Other Derivatives*, 4th ed., Prentice–Hall, Upper Saddle River, NJ, 2000.
[5] I. Karatzas and S. E. Shreve, *Brownian Motion and Stochastic Calculus*, 2nd ed., Springer-Verlag, Berlin, 1991.
[6] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, Berlin, 1999.
[7] Y. Komori, Y. Saito, and T. Mitsui, *Some issues in discrete approximate solution for stochastic differential equations*, Comput. Math. Appl., 28 (1994), pp. 269–278.
[8] X. Mao, *Stochastic Differential Equations and Applications*, Horwood, Chichester, 1997.
[9] B. Øksendal, *Stochastic Differential Equations*, 5th ed., Springer-Verlag, Berlin, 1998.
[10] H. C. Öttinger, *Stochastic Processes in Polymeric Fluids*, Springer-Verlag, Berlin, 1996.
[11] E. Platen, *An introduction to numerical methods for stochastic differential equations*, Acta Numer., 8 (1999), pp. 197–246.
[12] Y. Saito and T. Mitsui, *Stability analysis of numerical schemes for stochastic differential equations*, SIAM J. Numer. Anal., 33 (1996), pp. 2254–2267.
[13] The MathWorks, Inc., *MATLAB User's Guide*, Natick, Massachusetts, 1992.
[14] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.