An adaptive timestepping algorithm for stochastic differential equations

H. Lamba
Department of Mathematical Sciences,
George Mason University, MS 3F2,
4400 University Drive,
Fairfax, VA 22030, USA.

Phone: +1 (703) 993 1489
Fax: +1 (703) 993 1491
E-mail: hlamba@gmu.edu

Abstract

We introduce a variable timestepping procedure using local error control for the pathwise (strong) numerical integration of a system of stochastic differential equations forced by a single Wiener process. The Milstein method is used to advance the numerical solution and the stepsizes are determined via two local error estimates that roughly correspond to leading order deterministic and stochastic local error components. One advantage of using two error controls is an increased flexibility that allows for the treatment of both drift and diffusion dominated regimes in a consistent manner. Numerical results are presented and the generalization of this approach to wider classes of problems and methods is discussed.

Keywords Error control, numerical integration, stochastic differential equations

1 Introduction

In this paper we are concerned with the development and analysis of variable timestepping methods for the strong (pathwise) solution to stochastic differential equations (SDEs), written in Itô form as

$$dX = f(X)dt + g(X)\,dW, \qquad X(0) = X_0, \qquad t \in [0,T] \tag{1.1}$$

or, as an integral equation,

$$X(t) = X(0) + \int_0^t f(X(s))\,ds + \int_0^t g(X(s))\,dW \tag{1.2}$$

where $X(t) \in \mathbb{R}^m$ and $f, g : \mathbb{R}^m \to \mathbb{R}^m$. $W(t)$ is a scalar Wiener process (Brownian motion) and without loss of generality (1.1) is in autonomous form. We assume that a unique solution exists for all time which has been proved under a variety of Lipschitz conditions, moment bounds and growth conditions on f and g. In addition we require that f and g be twice differentiable.

In recent years there has been enormous interest in SDEs and they are now standard mathematical models in many of the same scientific disciplines that have benefited from the study of deterministic differential equations in the past. However, at almost every stage of the modeling process there are additional complications introduced by the stochastic component and this is also true for the numerical solution of equations such as (1.1). While the restriction to SDEs forced by a single Wiener process is a severe one, such systems are still of great practical relevance and form an easier subclass of problems on which to test and develop numerical strategies.

For ODEs there is a vast and highly successful body of work consisting of basic numerical schemes (most notably Taylor, linear multistep and Runge-Kutta methods) together with extremely robust and highly efficient implementation strategies often involving variable timestepping. For SDEs there are currently only relatively few schemes available. This is primarily due to the greatly increased complexity of the Taylor series expansions of the exact and numerical solutions, even when the equations are defined in Stratonovich rather than Itô form, and the resulting increase in the number and complexity of the order conditions that must be satisfied. The reader is referred to standard texts such as [13, 19] for any necessary background material. To date the literature on variable timestepping algorithms for SDEs is rather sparse (see for example [1, 3, 4, 7, 12, 18, 20, 15]) but it is to be anticipated that the development of such algorithms will play an important role in the numerical solution of at least some classes of SDEs.

The most common strategy for variable timestepping using a local error control in the deterministic case can be broken down into three basic components:

- The underlying numerical scheme used to advance the integration.
- An *estimate* of the local (one-step) error which must be bounded from above at every step by some quantity based upon the user-defined tolerance τ. If this criterion is not met then the step is rejected and a smaller timestep is chosen.
- A timestep selection mechanism that attempts to choose candidate timesteps as large as possible consistent with the local error criterion.

We shall take a similar approach. The stochastic method used to advance the numerical solution is the Milstein scheme which has strong order 1 and this paper will focus upon the choice of possible error estimates and timestep selection strategies. Much of the analysis that follows is either specific to the Milstein method or, at the very least, to methods of strong order 1. This is the maximal order, for a scalar stochastic forcing, that can be attained if only values of $W(t)$ are available. Higher-order methods or SDEs with multi-dimensional forcing require the independent generation of additional stochastic integrals of $W(t)$. Indeed a fundamental, as yet unanswered, issue concerning SDE solvers is to determine the point at which the very significant additional complexity of higher-order methods outweighs the benefits for a 'typical' user with 'typical' accuracy requirements. This increase in complexity also carries over to the problem of efficiently adapting timesteps. For ODEs the efficiency gains of adaptive timestepping are very impressive and almost entirely independent of the order of the method. For SDEs the efficiency gains are not as striking and carry increasing computational overheads as the complexity of the underlying method increases. It therefore seems entirely possible that an efficient adaptive scheme based upon a low-order numerical method may be the algorithm of choice for many applications.

The approach outlined in this paper is based upon the fact that for the Milstein method applied to (1.1) it is possible to (very cheaply) obtain two local error estimates, one based upon the drift component and the other based on the diffusion. This allows for an algorithm that can behave differently for drift or diffusion-dominated regimes, in particular with regard to how candidate timesteps are chosen. For example, in a drift-dominated (low diffusion) regime the choice of candidate timesteps could be chosen by a procedure close to that employed in adaptive ODE solvers, with little regard to the stochastic error. Another feature is that, when the diffusion is important, timesteps can be rejected with a high degree of certainty if the jump in the Wiener process on that timestep is deemed to be outside a suitable range, but before the Milstein approximation is calculated. Yet another advantage of introducing these two separate local error estimates is a demonstrable and dramatic improvement in the mean-square stability properties of the algorithm over the fixed-timestep Milstein method. These stability results will be reported elsewhere.

In Section 2 we define the Milstein method for a problem of the form (1.1) and, based upon leading-order terms of the local error expansion, define the local error estimates that will be controlled. Then in Section 3 we describe in detail how to generate, as and when needed, values of the Wiener process allowing for arbitrary stepsize changes and rejections. While the underlying mathematical result is elementary and fairly well-known, a precise description of the algorithmic procedure appears to be lacking in the adaptive timestepping literature and so this is included for completeness. Section 4 describes a complete algorithm by combining the analyses of Sections 2 and 3 with a timestep selection strategy. This strategy is far from definitive, rather it is designed to highlight the degree of flexibility inherent in the dual error control approach. Numerical results on standard test problems are also presented. Finally, in Section 5 we comment upon the development of adaptive algorithms for higher-order schemes and/or for SDEs with multi-dimensional forcing. We also demonstrate that the class of analytically solvable test problems, now becoming standard in the literature, may not be sufficiently general to adequately test adaptive algorithms.

2 The Milstein method and local error estimates

We start by briefly describing an archetypal adaptive algorithm for ODEs based upon an embedded Runge-Kutta pair of orders $p-1$ and p. Let us suppose that the numerical integration has reached the n^{th} step with value X_n. Then for a candidate timestep h the local error estimate $E(X_n, h)$ is defined as the norm of the difference between the two approximations. The user-defined tolerance is denoted τ and a timestep will only be accepted, and the solution advanced, if

$$E(X_n, h) \leq \sigma(X_n, \tau) h^{\rho} \tag{2.1}$$

for some quantity σ closely related to τ. This allows for absolute, relative or mixed error control (by choosing for example $\sigma = \tau$, $\sigma = \|X_n\|\tau$ or $\sigma = \tau \max(1, \|X_n\|)$ respectively). The choice $\rho = 0$ corresponds to an error-per-step (EPS) strategy and $\rho = 1$ to an error-per-unit-step (EPUS) strategy. Note that $E(X_n, h)$ is, strictly speaking, an error estimate for the lower-order method but it is common to advance the solution using the higher-order method since it has already been calculated, an idea known as *extrapolation.*

Whether or not the error criterion (2.1) is satisfied using a timestep h, the next candidate timestep h' (for the same step if h was rejected and for the next step if h was accepted) is given by

$$h' = \min\left(h_{\max}, \theta h \left(\frac{\sigma(X_n,\tau)}{E(X_n,h)}\right)^{\frac{1}{p-\rho}}\right) \tag{2.2}$$

where X_n is the latest computed numerical value. The quantity $h_{\max}$ is a maximum timestep that is often a function of the integration time T. The exponent $\frac{1}{p-\rho}$ appears since the leading terms in the local error estimate $E(X_n, h)$ are elementary differentials premultiplied by constants of order $\mathcal{O}(h^p)$ (see for example [5]) and so, for a given X_n, $\|E(X_n, h)\| \approx Kh^p$ for some constant K in the limit as $h \to 0$. Thus, ignoring the factor θ in (2.2), h' should be close to optimal while still satisfying the local error criterion. The 'safety factor' $\theta < 1$ reduces the number of stepsize rejections. A typical value for θ is 0.8 and it is expected that the ratio of rejected to accepted steps tends to zero as $\tau \to 0$ for any $\theta < 1$. Even in the deterministic case there are many possible modifications and improvements that can be applied to the above basic strategy, see for example [8, 9, 21]. A very general convergence proof for a class of adaptive ODE algorithms can be found in [14].

We now return to SDEs and define the Milstein scheme used to advance the numerical solution of (1.1). This is

$$X_{n+1} = X_n + hf(X_n) + \Delta W g(X_n) + \frac{1}{2}g'(X_n)g(X_n)(\Delta W^2 - h) \tag{2.3}$$

where h is the stepsize used, X_n and X_{n+1} are the numerical approximations at times t_n and $t_n + h$ respectively and $\Delta W = W(t_n + h) - W(t_n)$. A numerical scheme for (1.1) is defined to converge with strong order p if, for fixed timesteps h, there exists $C > 0$ (independent of h), $h^* > 0$ such that

$$E(\max_{t_n<T} \|X(t_n) - X_n\|) \leq Ch^p \quad \forall h \leq h^*$$

where $X(t)$ is the exact solution at time t and $p > 0$ is as large as possible. The Milstein scheme has strong order 1.

The convergence of a general adaptive algorithm to the exact solution as the user-defined tolerance $\tau \to 0$, is proved in [7][Corollary 4.4] under the assumptions that the underlying method has strong order ≥ 1 and the maximum accepted stepsize also tends to zero. It should be noted however that the maximum stepsize need not necessarily tend to zero in the neighbourhood of fixed points and when the error control locally fails to adequately approximate the true truncation error. This is a subtle point, for a discussion of the ODE case see for example [10, 11] and for the SDE case [15].

In our analysis of the leading terms of the local error it will be convenient to work with Stratonovich, rather than Itô, Taylor expansions. Thus we rewrite (1.1) as

$$dX = \underline{f}(X)dt + g(X) \circ dW, \qquad X(t_0) = X_0 \tag{2.4}$$

where $\underline{f}(X) = f(X) - \frac{1}{2}g'(X)g(X)$ and rewrite (2.3) as

$$X_{n+1} = X_n + h\underline{f}(X_n) + \Delta W g(X_n) + \frac{1}{2}g'(X_n)g(X_n)\Delta W^2. \tag{2.5}$$

The Stratonovich-Taylor expansions of both the exact and numerical solutions of (2.4) are linear combinations of elementary differentials that consist of both functions f and g and various combinations of their derivatives. The reader is directed towards [3, 13] for further details. The local error of the Milstein method has the following expansion

$$J_{10}\underline{f}'g + J_{01}g'\underline{f} + \frac{1}{6}J_1^3 g''gg + \frac{1}{6}J_1^3(g')^2 g + \mathcal{O}(h^2) \tag{2.6}$$

where J_1, J_{10} and J_{01} are the multiple Stratonovich integrals defined by

$$J_1 = \int_{t_n}^{t_n+h} \circ\, dW = \Delta W, \quad J_{10} = \int_{t_n}^{t_n+h} \int_{t_n}^{s_1} \circ\, dW\; ds_1, \quad J_{01} = \int_{t_n}^{t_n+h} \int_{t_n}^{s_1} ds \;\circ\; dW(s_1).$$

From (2.6) we see that the leading order local error consists of 4 terms of $\mathcal{O}(h^{\frac{3}{2}})$, with the 3 different multiple Stratonovich integrals J_{10}, J_{01} and J_1^3 appearing as factors. This is to be contrasted with the deterministic case where, for a method of order p, all the leading order elementary differentials in the local error are simply multiplied by h^{p+1}, meaning that we only have to control h^{p+1} irrespective of which elementary differential(s) are dominating the local error. Note also that the quantities J_{10} and J_{01} cannot be derived from the values $W(t_n)$ and $W(t_n+h)$ which is why the maximal strong order of a method that only uses values of $W(t)$ is 1. However, the final term depends only upon g and g' which have already been calculated. So we proceed by assuming that controlling this final term will result in an acceptable control of the actual local error. This assumption is similar in nature to that underlying extrapolation, namely that controlling a quantity closely related to the actual truncation error effectively controls this error.

Therefore we define our first local error estimate to be

$$E(X_n, h) := \frac{1}{6}|\Delta W^3|\, \|g'\|_\infty\, \|g'g\|_\infty \tag{2.7}$$

which is an upper bound for the ∞-norm of the last term in (2.6) that only requires the additional calculation of $\|g'\|_\infty$ and thus avoids a matrix-vector multiplication. Our EPS error control is therefore

$$E(X_n, h) \leq \sigma(X_n, \tau) \tag{2.8}$$

for some choice of the function σ.

However, $E(X_n, h)$ is independent of the drift term $\underline{f}$ in (2.4). Therefore the algorithm cannot reasonably be expected to operate efficiently in the weak-diffusion limit where a more appropriate mode of operation would be one closer to the deterministic paradigm. And of course it may not be known a priori whether or when the numerical solution enters a weak-diffusion regime so it is important that a robust algorithm be able to deal with this case. Furthermore, for an SDE with additive noise the Milstein method, while retaining its strong order of 1, simply reduces to the Euler-Maruyama method and $E(X_n, h) \equiv 0$ causing the error control to fail completely. For both these reasons we introduce a second local error estimate based upon the drift term.

If we consider the $\mathcal{O}(h^2)$ terms in (2.6), then the leading order term that is defined solely by the drift is $\frac{1}{2}h^2\underline{f}'\underline{f}$ since the Milstein reduces to the deterministic Euler method when applied to ODEs. So, in the weak-diffusion limit, this is a better quantity to control than $E(X_n, h)$. Alternatively, we could also use $\frac{1}{2}h^2 f'f$ depending upon whether $f'f$ or $\underline{f}'\underline{f}$ is easier to compute. This will often depend upon whether the SDE being solved was originally derived in Itô or Stratonovich form.

We therefore introduce an additional EPS error control by approximating either $\frac{1}{2}h^2\underline{f}'\underline{f}$ or $\frac{1}{2}h^2 f'f$ as the difference between the Euler method and the improved Euler (Heun) method applied to either of the ODEs

$$\frac{dX}{dt} = f(X) \quad \text{or} \quad \frac{dX}{dt} = \underline{f}(X).$$

This results in a second local error estimate

$$E_d(X_n, h) := \left\| \frac{h}{2}(f^*(X_n + hf^*(X_n)) - f^*(X_n)) \right\|_\infty \tag{2.9}$$

for the cost of one extra function evaluation of either $f^* = f$ or $f^* = \underline{f}'$ and in addition to (2.8) we also enforce

$$E_d \leq \sigma(X_n, \tau). \tag{2.10}$$

We also note that $E = \mathcal{O}(h^{\frac{3}{2}})$ and $E_d = \mathcal{O}(h^2)$ so that unless τ and therefore h are sufficiently small, these error estimates may still be of comparable size far from the weak-diffusion limit. Indeed numerical experiments suggest this is the case and provide a further justification for the inclusion of a second error control that includes the drift.

3 Adaptive generation of the Wiener Process

In any variable timestepping algorithm that allows for the possibility of stepsize rejections, it is necessary to be able to generate new intermediate values of certain stochastic integrals (in our case just values of the Wiener process itself) at some time t conditional upon known values at times t_- and t_+ with $t_- < t < t_+$. In the case of ODEs the simplest, and indeed historically one of the first, timestep selection mechanisms is simply to halve the candidate timestep immediately after a stepsize rejection or to double it if the local error criterion is satisfied by a sufficient amount. However it is now generally accepted that timestep selection mechanisms based upon asymptotic considerations and without the halving-or-doubling restriction result in significantly more efficient algorithms. Most of the existing variable timestepping algorithms for SDEs based upon local error control also employ a halving-or-doubling strategy, using the known conditional distribution for the value at the midpoint of a Wiener process whose end values are known. These algorithms also have restrictions upon when doubling can occur (see [7] for details) but even with this very limited choice of stepsizes, the benefits of being able to vary the timesteps are apparent. A possible advantage of the halving-or-doubling restriction is that the Wiener process can be stored in a binary tree structure. But it would appear that in many practical applications this is unnecessary, especially if no other simulations are required using the same realization of the Wiener process (another reason for the halving-or-doubling restriction is due to the problem of approximating Lévy areas in the case of multi-dimensional forcing for non-commutative problems [6, 7, 17] but that is not relevant here).

The following elementary lemma (see for example [16]) defines the probability distribution of $W(t)$ given the values $W(t_-)$ and $W(t_+)$ for all $t_- < t < t_+$ and will allow us to generate intermediate values of the Wiener process without any restrictions.

Lemma 3.1 *Let* $\{W_t; t \geq 0\}$ *be a Wiener process and fix* $0 < t_- < t_+ < \infty$ *and* $t \in (t_-, t_+)$ *Then the random variable* $W(t)$*, conditioned on* $W(t_-) = \alpha$ *and* $W(t_+) = \gamma$*, is normally distributed with mean*

$$\alpha + (t - t_-)(\gamma - \alpha)/(t_+ - t_-)$$

and variance

$$(t - t_-)(t_+ - t)/(t_+ - t_-).$$

Lemma 3.1 will be used to generate the scalar Wiener process $W(t)$ at the endpoints of each candidate integration interval $[t_n, t_n + h]$ via the following procedure, which is based upon the Markovian properties of $W(t)$. For completeness we include a detailed description of this procedure.

Let us suppose that the integration has proceeded successfully as far as some time t_n and a timestep h has been chosen by the timestep selection mechanism. There are two cases to consider if $W(t_n + h)$ has not already been generated. First suppose that there are no currently generated values of $W(t)$ for $t > t_n + h$ and that the last known value of $W(t)$ is at time $t_+ \geq t_n$. Then $W(t_n + h) - W(t_+)$ will be normally distributed with mean 0 and variance $(t_n + h - t_+)$ and can be computed using a pseudo-random number algorithm. For the remaining case we define the times t_- and t_+ to be the closest times to $t_n + h$ with $t_- < t_n + h < t_+$ and then use Lemma 3.1 to generate $W(t_n + h)$ from the correct distribution.

An obvious but crucial observation is that although stepsizes can be rejected if the local error criterion is not satisfied, generated values of $W(t)$ must never be rejected (or ignored at a later integration step) since this would certainly generate a biased stochastic forcing. Therefore once the value of $W(t)$ has been generated it must be stored and utilized until no longer required, i.e. until the numerical integration has passed time t.

There are some important points to note about the above procedure. Firstly, the complete algorithm will not necessarily produce a solution output at all the times for which the Wiener process is generated. However the values at any of these times may be used by the timestep selection mechanism to help determine the choice of new timestep. Secondly, the algorithm can either generate a brand new Wiener process as the integration proceeds (no preprocessing or prior generation of the forcing is required as in [7, 18]) or it can take a Wiener process from a previous run and use this sequence of times t and values $W(t)$ as an additional input. The augmented Wiener process consisting of this inputted data together with all the new values generated during the run can then be considered as an additional output. In this way a sequence of runs could be performed using the same underlying Wiener process (for example, decreasing the tolerance with each run). We also note that in the case of multi-dimensional stochastic forcing for commutative problems (which is not considered here) the above procedure can be used exactly as above, with a Wiener process being generated for each dimension.

4 A prototype adaptive algorithm and numerical results

We now describe a straightforward error-per-step timestep selection strategy intended to provide a basic yet robust algorithm that displays the inherent flexibility of the adaptive approach. Throughout, any values of $W(t)$ required must be generated from the correct conditional probability distributions described in Section 3. Perhaps the most interesting feature of the algorithm is that timesteps can be rejected after generating ΔW but before calculating the Milstein approximation if the magnitude of the Brownian increment $|\Delta W|$ is deemed to be either too large or too small. Thus a 'near-optimal' value of ΔW can be sought before an updated solution is calculated.

This can result in a significant computational saving but also introduces new issues. In particular, the relative cost of performing one step of the Milstein method versus the cost of generating a new Gaussian random variable becomes significant. For example, if the functions f, g and g' are very expensive to compute, then the algorithm should make more of an effort to find near-optimal values of h and the corresponding ΔW (that will be accepted with a high probability) before calculating a Milstein approximation and testing it against the error controls. This will result in a possibly large number of evaluations of the Brownian at times where the numerical solution is not calculated, but hopefully very few rejections of the numerical approximations themselves. On the other hand, if f, g and g' are very cheap to compute, then the algorithm should relax the requirements on whether particular values of h and ΔW are acceptable for the Milstein approximation to be computed, and be more willing to tolerate rejected solution updates.

The first case is the more interesting for expository purposes as it justifies a more sophisticated timestep selection process. It is also relevant to situations where even if f, g and g' are not especially complicated functions, Gaussian random variables have either already been calculated and stored, or perhaps where parallel computation is available. With this very firmly in mind we now define a prototype algorithm designed to minimize the number of stepsize rejections (after the Milstein approximation has been calculated) while attempting to satisfy the error controls as closely as possible. As the numerical results of the next section will show, this can be achieved relatively easily.

In the description of the algorithm, the quantities $h, \Delta W, E$ and E_d always refer to the values of those variables from the most recent calculation of the Milstein method and the error estimates, whether or not that timestep resulted in an advancement of the numerical solution. We introduce the variables $\kappa, \kappa_d, \Delta W_{\text{opt}}$ and α that will appear in the algorithm. These are defined as

$$\kappa = \frac{E}{\sigma}, \quad \kappa_d = \frac{E_d}{\sigma}, \quad \Delta W_{\text{opt}} = 0.9\kappa^{-\frac{1}{3}}|\Delta W|, \quad \alpha = \frac{|\Delta W|}{\sqrt{h}}.$$

The quantity ΔW_{opt} is an estimate for a near-optimal value of ΔW consistent with meeting the error criterion (2.8) (with a safety factor of 0.9 included). When we are attempting to control E it will be our aim to find a choice of timestep with an absolute Brownian increment close to but less than ΔW_{opt}. The quantity α is the number of standard deviations from the mean of the Brownian increment ΔW.

In order to start the algorithm an initial candidate timestep h_{init} must be chosen. Since asymptotically as $\tau \to 0$ we expect $h \approx \mathcal{O}(\tau^{\frac{2}{3}})$ we simply choose $h_{\text{init}} = \tau^{\frac{2}{3}}$. Now let us suppose that the Milstein approximation and E, E_d have just been calculated at some point X_n using a timestep h and corresponding Brownian increment ΔW. Then the solution is updated using the Milstein approximation if and only if $\max(E, E_d) \leq \sigma$. Now a new candidate timestep must be selected and we consider two cases, depending upon the relative magnitudes of E and E_d.

Case 1 Let us first suppose that $E_d \geq E$. If the numerical solution is in a drift-dominated regime this would suggest a timestep selection strategy closer to that for the deterministic paradigm (2.2) for the Euler-Heun pair and so we define the quantity

$$h' = \min(h_{\max}, 1.5h, 0.8h\kappa_d^{-\frac{1}{2}}).$$

Of course it may be that $E_d \geq E$ due to an unusually small value of $|\Delta W|$ and hence E (since $E \propto |\Delta W^3|$ it is highly variable from one timestep to the next). For this reason the maximum timestep ratio is kept relatively low at 1.5 and we choose the next candidate timestep $h_{\text{new}} = kh'/3$ for $k = 1, 2$ or 3 where k is chosen as follows. Define ΔW_j as the Brownian increment for the timestep $jh'/3$ and $k' = \max\{l : |\Delta W_j| \leq \Delta W_{\text{opt}}\ \forall j = 0, \ldots, l\}$. Then $k = \min(\max(k', 1), 3)$. Note that this additional testing of the Brownian increments for timesteps $h'/3$ and $2h'/3$ is equivalent to rejecting potential timesteps (either because $|\Delta W|$ is too large or too small) before calculating the Milstein approximation, but at the expense of generating extra values of the Brownian motion.

Case 2 Now suppose instead that $E_d < E$. We choose h_{new} by comparing the Brownian increments for $h/3, 2h/3, \ldots, 2h$. Define ΔW_j as the Brownian increment for the timestep $jh/3$ and $k' = \max\{l : |\Delta W_j| \leq \Delta W_{\text{opt}}\ \forall j = 0, \ldots, l\}$. We define $h_{\text{new}} = kh/3$ where k is defined as follows. If the latest timestep h was rejected (i.e. if $E > \sigma$) we ensure that $h_{\text{new}} < h$ by defining $k = \min(2, \max(k', 1))$. If $E \leq \sigma$ and $\alpha < 2$ then $k = \min(4, \max(k', 1))$ otherwise $k = \min(6, \max(k', 1))$. Thus we have an effective maximum timestep ratio of $4/3$ unless the last Brownian increment ΔW was an outlier (more than 2 standard deviations from the mean) in which case we increase the maximum stepsize ratio to 2.

Finally, we define $h = h_{\text{new}}$ using the corresponding ΔW to calculate the Milstein approximation and the error estimates and the process then repeats.

That completes the description of our prototype algorithm. It is not intended to be definitive, either in its algorithmic structure or for the optimality of any of the parameters used (the algorithm appears to be robust to moderate and sensible changes in the parameter values). More elaborate strategies that, for example, automatically calculate and optimize the trade off between the number of evaluations of $W(t)$ and the coefficients of the SDE are certainly possible. Indeed the scope for modifications is very large.

Finally we remark that if the above algorithm is applied to an SDE with additive noise then it simply reduces to a standard deterministic error control and should complete the integration successfully although its unmodified use is not necessarily recommended on such problems.

4.1 Numerical results

We now introduce numerical results for four test problems with exact analytic solutions and compare the numerical solutions generated by the algorithm described above to an equivalent fixed timestepping Milstein method. The algorithm was run using the absolute error criterion $\sigma = \tau$ and the maximum timestep was chosen to be $T/16$ in each case.

Our first two test problems are, in Itô form,

$$dX = -(1 + \beta^2 X)(1 - X^2)\, dt + \beta(1 - X^2) dW, \qquad X(0) = 0 \tag{4.1}$$

with $\beta = 0.1$ and 1.5 over the integration range $[0, 10]$. Even though the functions f, g and g' are all very cheap to evaluate, we shall proceed on the basis of the comments at

Table 1: Numerical results for $\beta = 0.1$.

τ	Method	# Acc. Steps	# Rej. steps	Error
10^{-2}	Variable	33	1	0.028
	Fixed			0.063
10^{-3}	Variable	63	1	0.0094
	Fixed			0.032
10^{-4}	Variable	160	2	0.0031
	Fixed			0.013

the start of the section. Note that $\beta = 0.1$ corresponds to a very weak stochastic forcing and is included to demonstrate the efficacy of the program in the weak-diffusion limit. Similar test problems also appear in [4, 18, 20] and the exact solutions are

$$X(t) = \frac{e^{-2t+2\beta W(t)} - 1}{e^{-2t+2\beta W(t)} + 1}.$$

Since $\underline{f}(X) = X^2 - 1$ is cheaper to evaluate than $f(X)$ we choose that as the basis of our deterministic error control.

Table 1 shows numerical results for $\beta = 0.1$ with $\tau = 10^{-2}, 10^{-3}, 10^{-4}$. For each value of τ, 100 simulations were run and the average number of accepted and rejected stepsizes is recorded. By a rejected stepsize, we mean a stepsize that was used to generate a Milstein approximation and error estimates but was not accepted. Since the exact solution tends to ± 1 as $t \to \infty$ with probability 1, we record the average of the maximum absolute error $E(\max_{t_n<T} |X(t_n) - X_n|)$ over the entire integration interval $[0, 10]$ rather than just the errors at $t = T$. The Milstein method was then run for 100 Wiener processes with a fixed stepsize corresponding to the integration time divided by the average total (accepted plus rejected) number of steps. The maximum error committed by the variable timestepping algorithm is about 3 times smaller than that committed by the fixed timestepping equivalent for each value of τ. There also appears to be approximate tolerance proportionality occurring. As expected, for each tolerance, $E_d > E$ for almost every timestep so the algorithm performs in a very similar manner to an ODE adaptive solver.

Table 2 was produced in exactly the same way for $\beta = 1.5$. The adaptive algorithm now outperforms its fixed timestepping counterpart by approximately a factor of 10. The last column displays the percentage of accepted timesteps on which $E_d > E$. As expected, reducing the tolerance results in a decrease in the proportion of steps where the drift error estimate dominates. This is because $E = \mathcal{O}(h^{\frac{3}{2}})$ while $E_d = \mathcal{O}(h^2)$ and so E will increasingly dominate as the timesteps decrease in size. However, the most striking feature is the very low number of stepsize rejections, approximately 2–3%, which was one of the main aims of the timestep selection procedure. This should be compared with the much higher rejection rates reported for other algorithms [4, 18], typically 25% or more.

Table 2: Numerical results for $\beta = 1.5$.

τ	Method	# Acc. Steps	# Rej. steps	Error	$E_d > E$
10^{-2}	Variable	127	3	0.081	42%
	Fixed			1.02	
10^{-3}	Variable	350	12	0.032	32%
	Fixed			0.28	
10^{-4}	Variable	1585	39	0.0079	23%
	Fixed			0.081	

Table 3: Numerical results for Test Problem 3.

τ	Method	# Acc. Steps	# Rej. steps	Error	$E_d > E$
10^{-2}	Variable	43	1	0.052	50%
	Fixed			0.16	
10^{-3}	Variable	172	6	0.011	39%
	Fixed			0.0309	
10^{-4}	Variable	761	26	0.0029	24%
	Fixed			0.014	

The third test problem (see [13] and [20][4.4.27]) in Itô form is

$$dX = -\sin(X)\cos^3(X)\,dt + \cos^2(X)dW, \qquad X(0) = 0 \tag{4.2}$$

with exact solution

$$X(t) = \tan^{-1}(W(t))$$

over the interval $[0, 5]$. However, when written in Stratonovich form we see that $\underline{f}(x) \equiv 0$ and so is not suitable for use in the deterministic error control. This point is explored further in the next section and is due to the very special nature of SDEs with exact analytic solutions. The algorithm was in fact tested twice, once on (4.2) and once on (4.2) but with the diffusion term modified to $\cos^2(2X)$. This modified SDE does not have an exact analytic solution but has similar characteristics to (4.2). In each case the deterministic error control was based upon the Itô drift and the simulation results were recorded exactly as for the first two test problems (for the modified version, the numerical solution was compared against an extremely accurate fixed-timestep Milstein approximation). The results were very similar and so only those for (4.2) are given in Table 3.

The final test problem, also in Itô form is

$$dX = \frac{1}{3}X^{\frac{1}{3}}\,dt + X^{\frac{2}{3}}dW, \qquad X(0) = 1 \tag{4.3}$$

with exact solution

$$X(t) = (1 + \frac{1}{3}W(t))^3$$

Table 4: Numerical results for Test Problem 4.

τ	Method	# Acc. Steps	# Rej. steps	Error	$E_d > E$
10^{-2}	Variable	19	0	0.0027	22%
	Fixed			0.0074	
10^{-3}	Variable	55	1	0.00086	19%
	Fixed			0.0032	
10^{-4}	Variable	235	3	0.00011	17%
	Fixed			0.00086	

and $T = 1$. Once again $\underline{f} \equiv 0$ but, as for the third test problem, this characteristic does not appear to affect the performance of the algorithm significantly one way or the other. The results are shown in Table 4. Once again there is a very low percentage of rejected Milstein updates and a favorable comparison with the fixed Milstein scheme in terms of required function evaluations to obtain a given accuracy.

5 Conclusions

The development and analysis of adaptive SDE solvers is still at a very early stage. Even for problems with a single stochastic forcing being integrated using a low-order method, the complicated nature of the local error expansion makes adaptivity a more difficult and less rewarding task than in the ODE case. This complexity gets significantly worse for multi-dimensional stochastic forcing (and also when using higher-order methods) and the potential usefulness of adaptive methods in such cases has not yet clearly been demonstrated.

The algorithm described above introduces two novel features, the use of a second (deterministic) error estimate and the possibility of early rejection of a candidate timestep by simply examining the Brownian increment ΔW. Both these features are easily incorporated into the above algorithm because of the simple form of the Milstein method, but variations of these ideas should also be applicable to algorithms based upon other underlying schemes. The control of two error estimates, one using only the drift term and the other using only the diffusion, also imparts desirable stability properties that will be reported elsewhere (a detailed analysis of the stabilizing effects of adaptive error control for ODE solvers can be found in [22]).

We conclude by demonstrating the potential inadequacy of certain model problems such as those in Section 4, for testing sophisticated adaptive algorithms. The vast majority of test problems that have been used in the literature to date have analytic, computable solutions where at time t, $X(t)$ depends only upon $W(t)$ (see [13][Chapter 4] for an extensive list). Furthermore, they are derived either by setting $\underline{f} \equiv 0$ (as in (4.2) and

(4.3)), or $g \equiv K\underline{f}$ for some constant $K \neq 0$ (as in (4.1)) and then transforming to a linear SDE. Both cases result in

$$\underline{f}'g = g'\underline{f} \tag{5.1}$$

and this induces considerable simplifications in the Taylor series of both the exact and numerical solutions. For example, substitution of (5.1) into (2.6), together with the relation $J_{10} + J_{01} = hJ_1$ results in a leading order local error for the Milstein method of

$$hJ_1\underline{f}'g + \frac{1}{6}J_1^3 g''gg + \frac{1}{6}J_1^3(g')^2 g + \mathcal{O}(h^2).$$

The additional numerical data, gathered from test problems similar to (4.1)–(4.3) but not satisfying (5.1), suggests that the numerical results in Section 4 are not due to the special nature of the test problems.

However, for certain numerical methods, the effect of (5.1) upon the local error expansion may be even more significant. As an example we consider the explicit stochastic Runge-Kutta method

$$\begin{aligned}
\eta_1 &= X_n \\
\eta_2 &= X_n + \frac{2}{3}hf(\eta_1) + \frac{2}{3}J_1 g(\eta_1) \\
X_{n+1} &= X_n + \frac{1}{4}hf(\eta_1) + \frac{3}{4}hf(\eta_2) + \frac{1}{4}J_1 g(\eta_1) + \frac{3}{4}J_1 g(\eta_2).
\end{aligned}$$

This method was introduced in [2] and has strong order 1 with minimized leading order error constants. However, when applied to problems satisfying (5.1), this method results in a greatly simplified truncation error

$$\frac{1}{6}J_1^3(g')^2 g + \mathcal{O}(h^2).$$

Thus test problems satisfying (5.1) may not always be suitable for evaluating the performance of algorithms on more general problems.

References

[1] S.S. Artemiev and T.A. Averina. *Numerical Analysis of systems of ordinary and Stochastic Differential Equations.* VSP, Utrecht, 1997.

[2] K. Burrage and P. M. Burrage. High strong order explicit Runge-Kutta methods for stochastic ordinary differential equations. *Appl. Numer. Math.*, 22(1-3):81–101, 1996.

[3] P. M. Burrage. *Runge-Kutta methods for stochastic differential equations.* PhD thesis, University of Queensland, 1998.

[4] P.M. Burrage, K. Burrage and T. Mitsui. Numerical solutions of stochastic differential equations – implementation and stability issues. *J. Comp. Appl. Math.*, 125:171–182, 2000.

[5] J.C. Butcher. *The numerical analysis of ordinary differential equations.* Wiley, New York, 1992.

[6] J.G. Gaines and T.J. Lyons. Random generation of stochastic area integrals. *SIAM J. Appl. Math.*, 54:1132–1146, 1994.

[7] J.G. Gaines and T.J. Lyons. Variable stepsize control in the numerical solution of stochastic differential equations. *SIAM J. Appl. Math.*, 57:1455–1484, 1997.

[8] K. Gustafsson, M. Lundh, and G Soderlind. A PI-stepsize control for the numerical solution of ordinary differential equations. *BIT*, 28:270–287, 1988.

[9] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving ordinary differential equations I. Nonstiff problems.* Springer-Verlag, Berlin, second edition, 1993.

[10] G. Hall. Equilibrium states of Runge-Kutta schemes. *ACM Trans. Math. Software*, 11:289–301, 1985.

[11] G. Hall and D.J. Higham. Analysis of stepsize selection schemes for Runge-Kutta codes. *IMA J. Numer. Anal.*, 8:305–310, 1988.

[12] N. Hoffman, T. Muller-Gronbach, and K. Ritter. Optimal approximation of stochastic differential equations by adaptive step-size control. *Math. Comp.*, 69:1017–1034, 2000.

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

[14] H. Lamba. Dynamical systems and adaptive time-stepping in ODE solvers. *BIT*, 40:314–335, 2000.

[15] H. Lamba, J. Mattingly, and A.M. Stuart. Strong convergence of an adaptive Euler-Maruyama scheme for stochastic differential equations. Technical Report. University of Warwick 2003.

[16] P. Lévy. *Processus Stochastiques et Mouvement Brownien, Monographies des Probabilites*. Gauthier-Villars, Paris, 1948.

[17] C.W. Li and X.Q. Liu. Approximation of multiple stochastic integrals and its application to stochastic differential equations. *Nonlinear Anal.*, 30:697–708, 1997.

[18] S. Mauthner. Stepsize control in the numerical solution of stochastic differential equations. *J. Comp. Appl. Math.*, 100:93–109, 1998.

[19] B Oksendal. *Stochastic Differential Equations – an introduction with applications.* Springer-Berlin, 1998.

[20] A. Rößler, J. Lehn and O. Schein. Adaptive schemes for the numerical solution of SDEs – a comparison. *J. Comp. Appl. Math.*, 138:297–308, 2002.

[21] L.F. Shampine. *Numerical Solution of Ordinary Differential Equations.* Chapman and Hall, New York, 1994.

[22] A.M. Stuart and A.R. Humphries. The essential stability of local error control for dynamical systems. *SIAM J. Num. Anal.*, 32:1940–1971, 1995.