École Polytechnique Fédérale de Lausanne

Semester project

Adaptive time-stepping methods for numerical integration of stochastic differential equations using S-ROCK methods

---

Student:

Aleksa Stankovic

Mentors:

Prof. Assyr Abdulle

Giacomo Rosilho de Souza

Lausanne, November 20, 2017

Abstract

In this paper we study adaptive time-stepping scheme for S-ROCK2 method. S-ROCK2 is explicit stabilized method especially suitable for integration of stiff stochastic differential equations. Other explicit stabilized methods for solving ordinary and stochastic stiff differential equations are briefly discussed as well. New local error estimator based on embedding is developed for S-ROCK2 method. Using this estimator, a variable time-stepping scheme for S-ROCK2 method is introduced. Several numerical tests are performed, and the properties of new scheme is discussed. In addition to this, another approach for dealing with non-commutative stochastic differential equations based on Brownian trees is presented.

Acknowledgements

I would like to thank professor Abdulle for finding very interesting topic which kept me captivated for months, and supporting me throughout the project duration.

I am also very thankful to Giacomo Rosilho de Souza, who invested a lot of time in supervising my work and helping whenever it was necessary. His support and ideas were of invaluable significance for completing this project.

Contents

1 Introduction

$\mathcal{C}$ Stochastic modeling is gaining more and more interest in recent years. Day by day, new, more sophisticated models are being developed. Some areas in which stochastic differential equations occur are molecular biology, epidemiology, hydrology, finance, chemistry and physics [30, 15, 10].

Research in stochastic modeling is followed by fundamental research in numerical methods for evaluating these models. Popular choices for tackling these problems lie in the class of Runge-Kutta based methods for stochastic differential equations, with the most popular being Euler-Maruyama [24] and Milstein-Talay methods [29]. For a special class of problems which are commonly refereed to as *stiff* problems these methods are unfortunately experiencing strict step-size restrictions due to their non-favorable stability properties. In this case we need to turn either to implicit or to stabilized explicit methods [4, 23]. In this paper we concentrate on S-ROCK2 method, because of its good stability and performance characteristics in this setting.

Treating stochastic differential equation turns to be computationally way more expensive than treating the deterministic problems. High computational cost shows the importance of finding new ways to improve performance of existing strategies. One of the ways to improve existing algorithms is to use time-steps of variable size for iterations of Runge-Kutta solver. For example, in case we are in 'smooth' part of an equation, we can use large time-steps, while in the parts that are tricky for approximation we use smaller time-steps in order retain the precision.

Adaptive time-stepping strategies for strong approximation of stochastic differential equations were already studied by Lambda [19], Mauthner [25], Hofmann, Müller-Gronbach & Ritter [13]. Some of the research is oriented towards Stratonovich stochastic differential equations [7, 8]. The specific difficulty we face when dealing with adaptive algorithms for stochastic differential equations is that, when the step-size is rejected, we need to make sure to sample the random variables in such way to stay on the same Brownian path. Otherwise, a bias is introduced and we get incorrect solutions.

In this paper we focus our attention to S-ROCK2 method, for which we design adaptive time-stepping scheme. The paper is organized as follows. In section 2 we introduce reader to

stiff problems, stability concept and show how these problems can be tackled with explicit stabilized methods. Then, in section 3, we expand our discussion to stiff stochastic problems, and show how methods from section 1 can be adjusted to the new setting. In section 4, we discuss common techniques for local error estimation (extrapolation, embedding), and derive a local error estimator for S-ROCK2 method. In this section we also show how we can sample Brownian motion and approximate integrals 'back-in-time', and stay on the same path. Finally, in this section we briefly give an idea of how Brownian trees can be used when stochastic methods are applied to non-commutative stochastic differential equations. Then, in section 5, we perform numerical tests of designed scheme, and show the results. In the final section 6, we comment on the results and propose further directions of research.

2 Stabilized explicit methods for deterministic problems

In this subsection we discuss solving deterministic differential equations, which have the following form:

$$dX = f(X(t))dt, \qquad X(0) = X_0, \tag{1}$$

with $X_0 \in \mathbb{R}^n$.

For these problems efficient explicit methods with high convergence rate have been developed. The most common ones are Runge-Kutta and multistep methods [12]. But for certain types of problems application of these methods requires using a very small step-size h in order to retain satisfactory stability of a method. These problems are known as *stiff* problems in the literature. There is no widely accepted definition of *stiffnes*, but we usually define a problem to be stiff if there is at least one very large eigenvalue of $\frac{\partial f}{\partial X}$. Method of choice in this context will not only need to have a good order of convergence, but robust stability properties as well.

One of the most common ways to study stability of a method is to apply it to the Dahlquist equation

$$dX = \lambda X dt, \quad X(0) = X_0, \tag{2}$$

where $Re(\lambda) < 0$. With this assumption, the exact solution $X(t) = X_0 e^{\lambda t}$ approaches 0 as

$t \to \infty$. We want our approximation to mimic the behavior of the exact solution, so in this case we want it to converge to 0 as well. Application of a Runge-Kutta method to (2) gives us a numerical approximation $X_1 = R(\lambda h)X_0$, where R is a rational function which depends on the method. By introducing $z = \lambda h$ we see that after n iterations the n-th iterate will be $X_n = R(z)^n X_0$, and thus

$$X_n \to 0 \text{ as } n \to 0 \;\Leftrightarrow\; |R(z)| < 1.$$

In the framework of our discussion we see that of special importance is the set[1] $S = \{z \in \mathbb{C}^- \;;\; |R(z)| < 1\}$, which we call the stability domain.

Let us shortly discuss the stability of a very simple Runge-Kutta method known as explicit Euler method. This method is given by

$$X_1 = X_0 + hf(X_0). \tag{3}$$

Applying this method to the test equation (2) we have that $X_1 = X_0 + \lambda h X_0 = (1 + \lambda h)X_0$. Hence, we have $R(z) = 1 + z$ and the stability region for explicit Euler method is $S = \{z \in \mathbb{C}^{-1} \;;\; |1 + z| < 1\}$. Graphical representation of this stability region is given in figure 1.

Classical explicit Runge-Kutta methods have very small stability regions. The straightforward way to deal with this problem is to switch to implicit Runge-Kutta methods which are A-stable[2]. Unfortunately, this tends to be computationally very expensive since implicit Runge-Kutta methods require solving system of equations at each step. Moreover, extending implicit methods to stochastic equations can present a problem, and thus these methods are not suitable for our discussion.

Another idea is to construct a class of explicit Runge-Kutta methods which have extended stability region at the expense of using more stages at each time-step. For this we first fix an order of a method p and a number of stages $s \geq p$. Then we search for a method of order p that will have $|R(z)| \leq 1$ with $z \in [-l_s, 0]$ for l_s as big as possible.

Methods need to have $R(0) = 1$ and $R'(0) = 1$ to be of order 1. In [14] it was shown that the optimal choice of such R for s-stage method of order 1 is $R(z) = T_s(1 + \frac{z}{s^2})$ where T_s is s-th Chebyshev polynomial of the first kind. For this choice of polynomial R we have

[1] $\mathbb{C}^- = \{z \in \mathbb{C} \;;\; Re(z) < 0\}$

[2] A-stable methods have stability region which covers $\mathbb{C}^-$

that $l_s = 2s^2$. Using the recursive definition of Chebyshev polynomials we can realize this method with the following algorithm:

$$\begin{aligned} K_0 &:= X_0, \\ K_1 &:= K_0 + \frac{h}{s^2} f(K_0), \\ K_j &:= \frac{2h}{s^2} f(K_{j-1}) + 2K_{j-1} - K_{j-2}, \qquad 2 \leq j \leq s, \\ X_1 &= K_s. \end{aligned} \tag{4}$$

From now on we will call this method RKC1. In figure 1 we can see improvement of stability over the classical explicit Euler method.

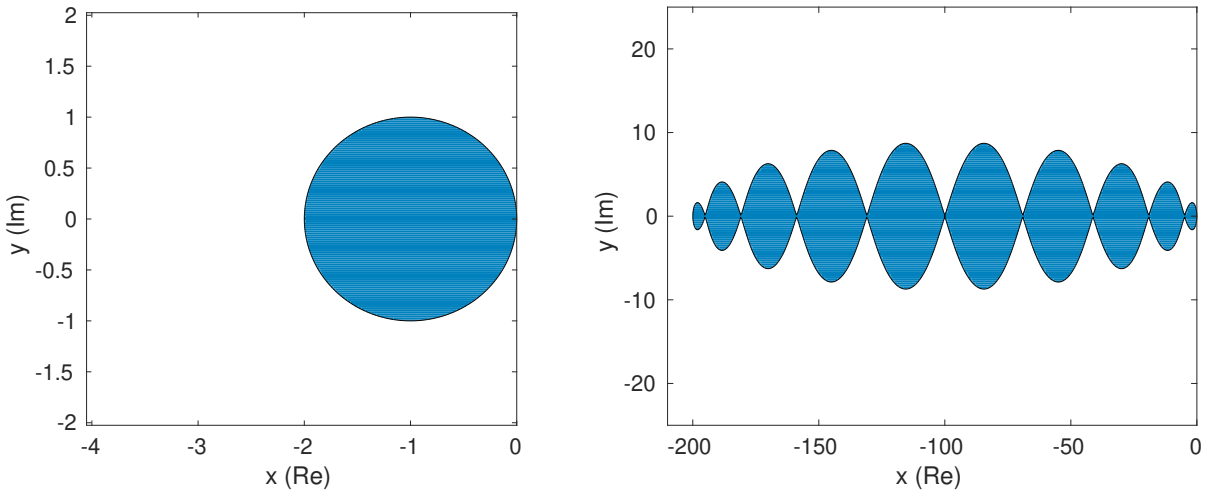

Figure 1: Stability regions for explicit Euler method (left figure) and for RKC1 method with s=8 (right figure). Notice the difference in scales.

For higher order methods there are no explicit analytic expressions for a polynomial R with optimal stability properties. Only analytic expressions in form of elliptic integrals are found [20]. Different approaches in literature are proposed for dealing with this problem. Some of them aim to approximate elliptic integral [26, 21, 22]. Other search for suboptimal polynomial R which computation can be expressed in explicit analytic form using recurrence relations. These are ROCK [1, 3] and RKC methods [14].

Here we restrict ourselves to the second-order ROCK2 method [1, 2] because it is expressed with three-term recurrence formula and has larger stability region than the corre-

sponding second order RKC2 method. The method is given by

$$\begin{aligned}
&K_0 := X_0,\\
&K_1 := K_0 + \mu_1 h f(K_0),\\
&K_j := \mu_j h f(K_{j-1}) - \nu_j K_{j-1} - \kappa_j K_{j-2}, \qquad 2 \le j \le s-2\\
&K_{s-1} := K_{s-2} + 2\tau h f(K_{s-2}),\\
&X_1 := K_{s-2} + (2\sigma - \frac{1}{2}) h f(K_{s-2}) + \frac{1}{2} h f(K_{s-1}),
\end{aligned} \tag{5}$$

where $\mu_j, \nu_j, \kappa_j, \sigma, \tau$ depend on s and are computed numerically. More details about construction of these coefficients, convergence analysis and stability properties of ROCK2 method can be found in [1].

2.1 Damping of a method

Even though the growth of stability region in real-axis with the respect to the number of stages is very good, there are zones in the graph where stability region is very thin in the direction of imaginary axis, which we can see from figure 1. This issue is resolved by introducing a damping parameter $0 < \nu < 1$ and requiring from our method to satisfy $|R(z)| < \nu$ on $[-l_s^\nu, -\epsilon]$, for l_s^ν as big as possible.

For RKC1 method this is achieved by the following scheme:

$$\begin{aligned}
&\mu = 1 - \nu, \quad w_0 = 1 + \mu/s^2, \quad w_1 = T_s(w_0)/T_s'(w_0),\\
&K_0 := X_0,\\
&K_1 := K_0 + h\frac{w_1}{w_0} f(K_0),\\
&K_j := 2hw_1 \frac{T_{j-1}(w_0)}{T_j(w_0)} f(K_{j-1}) + w_0 \frac{T_{j-1}(w_0)}{T_j(w_0)} K_{j-1} - \frac{T_{j-2}(w_0)}{T_j(w_0)} K_{j-2}, \quad 2 \le j \le s,\\
&X_1 = K_s.
\end{aligned} \tag{6}$$

We can observe that for $\nu = 0$ we get the same method as before. But for other values of ν we get improved stability of the method, as can be seen from the figure 2.

Stability properties of ROCK2 method in its form (5) depend on the coefficients $\mu_j, \nu_j, \kappa_j, \sigma, \tau$. If we construct these coefficients as proposed in [1], ROCK2 method will be already damped for some value ν.

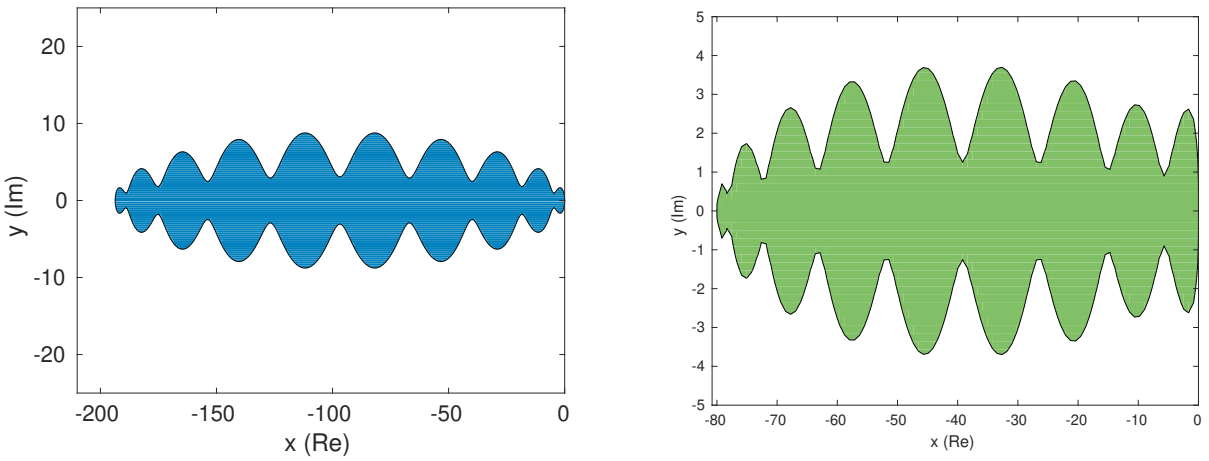

Figure 2: Stability regions of damped RKC1 method (left figure) and ROCK2 method (right figure) with damping parameter $\nu = 0.95$ and number of stages $s = 10$.

In practice we want to avoid recomputing all the coefficients $\mu_j, \nu_j, \kappa_j, \tau, \sigma$ each time we change the damping of the method. We usually have them precomputed in a table just for some specific value of ν (for example $\nu = 0.95$). In case ROCK2 method with different damping is needed, an additional parameter $\alpha > 0$ can be introduced and then we can use following adjusted scheme:

$$\begin{aligned} &K_0 := X_0, \qquad K_1 := K_0 + \alpha\mu_1 hf(K_0), \\ &K_j := \alpha\mu_j hf(K_{j-1}) - \nu_j K_{j-1} - \kappa_j K_{j-2}, \qquad 2 \le j \le s-2, \\ &K_{s-1} := K_{s-2} + 2\tau_\alpha hf(K_{s-2}), \\ &X_1 := K_{s-2} + (2\sigma_\alpha - \frac{1}{2})hf(K_{s-2}) + \frac{1}{2}hf(K_{s-1}), \end{aligned} \tag{7}$$

where σ_α and τ_α satisfy

$$\sigma_\alpha = \frac{1-\alpha}{2} + \alpha\sigma, \qquad \tau_\alpha = \frac{(\alpha-1)^2}{2} + 2\alpha(1-\alpha)\sigma + \alpha^2\tau.$$

Obviously, for $\alpha = 1$ the original ROCK2 method is recovered.

3 Stochastic methods

In this section we will present different stochastic methods for solving stochastic differential equations, discuss their properties and scope of application. We will also introduce the reader

to a special class of stochastic methods that are designed to tackle stiff problems, for which we analyze time-stepping scheme in later chapters.

3.1 General definitions and properties

We are interested in numerical integration of the following equation:

$$dX = f(X(t))dt + \sum_{r=1}^{m} g^r(X(t))dW_r(t), \qquad X(0) = X_0, \tag{8}$$

where $X(t)$ is a random variable with values in $\mathbb{R}^n$, $f, g^r : \mathbb{R}^n \to \mathbb{R}^n, r = 1, ..., m$, $W^r(t)$ are independent one-dimensional standard Brownian motions and X_0 is given random variable. The term $f(X(t))dt$ is also known as the *drift*, while $\sum_{r=1}^{m} g^r(X(t))dW_r(t)$ is called the *stochastic term* or *diffusion*. The equation (8) can be also expressed in integral form

$$X(t_1) = X(t_0) + \int_{t_0}^{t_1} f(X(t))dt + \sum_{r=1}^{m} \int_{t_0}^{t_1} g^r(X(t)dW_r(t). \tag{9}$$

Terms $\int_{t_0}^{t_1} g^r(X(t)dW_r(t), r = 1, ..., m$ depend on the stochastic calculus chosen for solving the problem, with the most common ones being Itô and Stratonovich calculus. For the sake of brevity we will concentrate on Itô calculus in this paper. The whole discussion can be easily extended to the Stratonovich calculus as well.

The fact that equation (8) is given in an autonomous form does not reduce the generality of our framework. In order to proceed with integration of (8), we need to make sure that solution exists and that it is unique on an interval of interest $[0, T]$. In case f and g^r satisfy following conditions:

$$|f(x) - f(y)| + \sum_{r=1}^{m} |g^r(x) - g^r(y)| \leq K|x - y|,$$

$$|f(x)|^2 + \sum_{r=1}^{m} |g^r(x)|^2 \leq K^2(1 + |x|^2), \quad \text{with } K \in \mathbb{R}^+,$$

with $x, y \in [0, T]$, then the equation indeed has an unique solution. Thus, from now on we will assume that functions f, g^r satisfy these conditions.

The class of numerical methods for solving SDE can be put in an abstract context by giving a map:

$$X_{n+1} = \Psi(X_n, h, \zeta_n), \tag{10}$$

where $X_n \in \mathbb{R}^n$ is our approximation at n-th step, h is a step size and ζ_n is a random vector. Then we proceed by iterating this function N times in order to get X_n at time $T = Nh$. Two most usual ways to describe precision of our methods is by stating its strong or weak order. For a numerical method we say that it has strong order τ if

$$\mathbb{E}|X_n - X(t_n)| \leq Ch^\tau,$$

and to have weak order of τ if and only if for all functions[3] $\phi \in C_p^{2(\tau+1)}(\mathbb{R}^n, \mathbb{R})$ holds

$$|\mathbb{E}(\phi(X_n) - \phi(X(t_n)))| \leq Ch^\tau$$

for any $t_n = nh \in [0, T]$, all h small enough, and for a constant C independent of h.

Theorem of Milstein [27] gives us a neat way to determine global convergence order by looking only at the local error (error after one step).

Theorem 1. *Consider a stochastic differential equation (8) and a numerical method (10). Assume that $f, g^r \in C_P^{2(\tau+1)}(\mathbb{R}^n, \mathbb{R}^n)$ and that they are Lipschitz continuous. Assume also that the moments $\mathbb{E}(|X_n|^{2r})$ are bounded for $r \in \mathbb{N}, 0 \leq nh \leq T$ uniformly with respect to all sufficiently small h. Finally, assume that the local error bound for any $\phi \in C_p^{2(\tau+1)}(\mathbb{R}^n, \mathbb{R})$ and all initial values X_0 satisfies*

$$|\mathbb{E}(\phi(X_1) - \phi(X(t_1))| \leq Ch^{\tau+1}.$$

Then the method has a global weak order τ.

For the method to have strong order of convergence τ it is sufficient that f, g^r are sufficiently smooth, Lipschitz continuous and that they satisfy

$$\mathbb{E}|X_1 - X(t_1)| \leq Ch^{\tau+1/2} \text{ and } |\mathbb{E}(X_1 - X(t_1))| \leq Ch^{\tau+1},$$

for any initial value X_0.

Proof. Proof can be found in [28, Chapter 2.2]. □

It is also useful to show the global order of convergence in practice, by running Monte-Carlo simulations for a certain problem and plotting the errors against time-step size in log-log graph. For example, in figure 3 we can deduce that S-ROCK and S-ROCK2 have

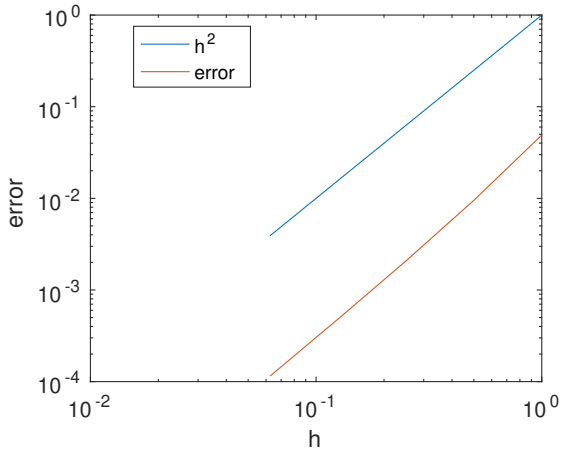

Figure 3: Graphical study of convergence for S-ROCK(left) and S-ROCK2(right) methods. Errors are calculated by running Monte-Carlo simulations for 10^8 different Brownian paths on linear SDE (11) . Graph is plotted in log-log scales.

weak order of 1 and 2, respectively. As it was the case for deterministic problems, stability of a method plays very important role for some types of problems. Two most common stability notions for a stochastic methods are mean-square and asymptotic stability.

To study mean-square and asymptotic stability stability we use a test equation

$$dX = \lambda X dt + \mu X dW, \quad X(0) = X_0, \tag{11}$$

on which we apply method of choice to get approximations X_n of a solution at time t_n. If for any given X_0 that satisfies $\mathbb{E}(|X_0|^2) < \infty$, and for any $t_n \to \infty$ we have

$$\lim_{n\to\infty} \mathbb{E}(|X_n|^2) = 0,$$

then we say that the method is mean-square stable.

If for any X_0 and $t_n \to \infty$ we have

$$\lim_{n\to\infty} |X_n| = 0, \text{ with probability 1,}$$

then the method is said to be asymptotically stable.

[3] $C_P^l(\mathbb{R}^n, \mathbb{R}) = \{f : \mathbb{R}^n \to \mathbb{R} \mid f \text{ is } l \text{ times continuously differentiable and all partial derivatives of } f \text{ up to order } l \text{ have polynomial growth}\}$

3.2 Explicit methods

The simplest method for approximating (8) is just an extension of Euler's explicit method (3) with an additional term dealing with stochastic part of SDE. The method is known as Euler-Maruyama method and its iterations are given by the following formula

$$X_1 = X_0 + f(X_0)h_n + \sum_{r=1}^{m} g^r(X_0)\Delta W_r, \tag{12}$$

where $\Delta W_r = W_r(t_1) - W_r(t_0) \sim \mathcal{N}(0, t_1 - t_0)$ are Wiener increments.

For construction of higher-order methods asymptotic analysis of stochastic integrals $\int_{t_0}^{t_1} g^r(X(t)dW_r(t)$ is needed. The analysis can be performed by using Itô expansions. This gives rise to well known, weak second-order, Milstein-Talay[29] method:

$$\begin{aligned} X_1 = X_0 &+ f(X_0)h + \sum_{r=1}^{m} g^r(X_0)I_r + \sum_{q,r=1}^{m} (g^r)'(X_0)g^q(X_0)I_{q,r} \\ &+ \left(f'(X_n)f(X_n) + \frac{1}{2}\sum_{r=1}^{m} f''(X_0)(g^r(X_0), g^r(X_0)) \right) \frac{h^2}{2} \\ &+ \sum_{r=1}^{m} f'(X_0)g^r(X_0)I_{r,0} \\ &+ \sum_{r=1}^{m} \left((g^r)'(X_0)f(X_0) + \frac{1}{2}\sum_{q=1}^{m} (g^r)''(X_0)(g^q(X_0), g^q(X_0)) \right) I_{0,r}, \end{aligned} \tag{13}$$

where $I_r, I_{r,0}, I_{0,r}$ and $I_{q,r}$ represent following stochastic integrals:

$$I_r = \int_{t_0}^{t_1} dW_r(t) = \Delta W_r, I_{r,0} = \int_{t_0}^{t_1}\int_{t_0}^{t} dW_r(s)dt, I_{0,r} = \int_{t_0}^{t_1}\int_{t_0}^{t} dsdW_r(t), I_{q,r} = \int_{t_0}^{t_1}\int_{t_0}^{t} dW_q(s)dW_r(t).$$

Unfortunately, this scheme is not very useful in practice because its formula involves derivatives of f and g^r for which computation is expensive in most situations, as well as stochastic integrals $I_{q,r}$ which are expensive for numerical integration. In [4, eq. (3.7)] a modification of this method is presented which solves these problems. Firstly, it was observed that replacing stochastic integrals with discrete random variables will not change the second order of the scheme. By introducing independent discrete random variables χ_r, ξ_r, $r = 1, ..., m$ with:

$$\mathbb{P}(\chi_r = \pm 1) = 1/2, \quad \mathbb{P}(\xi_r = \pm\sqrt{3h}) = 1/6, \quad \mathbb{P}(\xi_r = 0) = 2/3, \tag{14}$$

we can approximate stochastic integrals $I_{r,0}$ and $I_{0,r}$ with $h\xi_r$, while integrals $I_{q,r}$ can be approximated by $J_{q,r}$ which are defined as:

$$J_{q,r} = \begin{cases} (\xi_r\xi_r - h)/2 & \text{if } q = r, \\ (\xi_q\xi_r - h\chi_q)/2 & \text{if } r < q, \\ (\xi_q\xi_r + h\chi_r)/2 & \text{if } r > q. \end{cases} \tag{15}$$

Secondly, thanks to the clever use of χ_r and the fact that χ_r, ξ_r are independent, first and second derivatives of f and g^r were approximated with finite differences. The derived scheme reads as:

$$\begin{aligned} K_1 &= X_0 + hf(X_0), \qquad K_2 = K_1 + \sum_{r=1}^{m} g^r(X_0)\xi_r \\ X_1 &= X_0 + \frac{h}{2}\left(f(X_0) + f(K_2)\right) + \frac{1}{2}\sum_{r=1}^{m}\left(g^r\left(X_0 + \sum_{r=1}^{m} g^q(X_0)J_{q,r}\right) - g^r\left(X_0 - \sum_{r=1}^{m} g^q(X_0)J_{q,r}\right)\right) \\ &+ \frac{1}{2}\sum_{r=1}^{m}\left(g^r\left(\frac{X_0 + K_1}{2} + \sqrt{\frac{1}{2h}}\sum_{q=1}^{m} g^q(X_0)\chi_q\right) + g^r\left(\frac{X_0 + K_1}{2} - \sqrt{\frac{1}{2h}}\sum_{q=1}^{m} g^q(X_0)\chi_q\right)\right)\xi_r \end{aligned} \tag{16}$$

We also remark here that discrete random variables ξ_r are defined in slightly different manner in comparison to those defined in [4]. They are scaled by $\sqrt{h}$ to make their connection to Brownian increments more intuitive, which will be useful latter when we introduce adaptive scheme. Finally, we make remark that by using discrete variables also methods higher weak order of convergence can be designed. More details can be found in [16].

3.3 Stabilized explicit methods

Methods presented in the previous chapter were developed just by using asymptotic analysis of given SDE (8), and therefore as in the case of ODEs can require stringent conditions on the step-size h when applied to stiff problems. For these problems we need to use methods with suitable stability properties. Research in stochastic methods has shown that the stability of methods is largely influenced by treatment of the drift term.

Therefore, it is natural to use the same ideas employed in stabilized explicit methods for ODEs here as well. By combining RKC1 method to deal with the drift term and the same

procedure as in Euler-Maruyama for stochastic part we get the following method [5]:

$$
\begin{aligned}
&\mu = 1-\nu, \quad w_0 = 1+\mu/s^2, \quad w_1 = T_s(w_0)/T_s'(w_0),\\
&K_0 := X_0,\\
&K_1 := K_0 + h\frac{w_1}{w_0} f(K_0),\\
&K_j := 2hw_1\frac{T_{j-1}(w_0)}{T_j(w_0)} f(K_{j-1}) + w_0\frac{T_{j-1}(w_0)}{T_j(w_0)}K_{j-1} - \frac{T_{j-2}(w_0)}{T_j(w_0)}K_{j-2}, \quad 2\le j\le s,\\
&X_1 = K_s + \sum_{r=1}^{m} g^r(K_{s-1})I_r.
\end{aligned}
\tag{17}
$$

which has strong order 1/2 and weak order 1. This method is known as S-ROCK(1,1/2).

For commutative noise[4] or one-dimensional Wiener process it is also possible to improve the strong convergence of the method by considering finishing procedure:

$$
\begin{aligned}
K^*_{s-1} &= K_{m-1} + \sum_{r=1}^{m} g^r(K_{s-1})I_r,\\
K^{**,r}_{s-1} &= K_{m-1} + \sqrt{h} g^r(K_{s-1}),\\
K_s &= 2\sqrt{h}w_1\frac{T_{s-1}(w_0)}{T_s w_0} f(K_{s-1}) + 2w_0\frac{T_{s-1}(w_0)}{T_s(w_0)}K_{s-1} - \frac{T_{s-2}(w_0)}{T_s(w_0)}K_{s-2}\\
&\quad + \sum_{r=1}^{m} I_r g^r(K_{s-1}) + \frac{1}{2}\sum_{r=1}^{m} I_r\left(g^r(K^*_{s-1}) - g^r(K_{s-1})\right)\\
&\quad - \frac{1}{2}\sum_{r=1}^{m}\sqrt{h}\left(g^r(K^{**,r}_{s-1}) - g^r(K_{s-1})\right),\\
X_1 &= K_s.
\end{aligned}
\tag{18}
$$

We call this method S-ROCK(1,1). It is of weak and strong order 1. For a convergence of an adaptive scheme, method needs to have at least strong order 1 [9]. Thus, this is the simplest S-ROCK method that can be considered for adaptive time-stepping schemes.

Remark. *For one-dimensional Wiener noise ($m = 1$), the S-ROCK(1,1) can be further*

[4]Consider $L^l = \sum_{k=1}^{n} g^l_k \frac{\partial}{\partial y^k}, l = 1, 2, \ldots, m$. Commutative noise means that the condition $L^l g^r_k = L^r g^l_k k$ is satisfied for every $l, r = 1, \ldots, m; k = 1, \ldots, n$.

simplified and written as:

$$\begin{aligned} K_{s-1}^* &= K_{s-1} + \sqrt{h} g(K_{s-1}), \\ K_s &= 2hw_1 \frac{T_{s-1}(w_0)}{T_s(w_0)} f(K_{s-1}) + 2w_0 \frac{T_{s-1}(w_0)}{T_s(w_0)} K_{s-1} - \frac{T_{s-2}(w_0)}{T_s(w_0)} K_{s-2}, \\ &\quad + I g(K_{s-1}) + \frac{I^2 - h}{2\sqrt{h}} \left(g(K_{s-1}^*) - g(K_{s-1}) \right), \\ X_1 &= K_s. \end{aligned} \tag{19}$$

It is also possible to construct a weak second-order method based on ROCK2 method for deterministic term known as S-ROCK2 [4]. The treatment of the stochastic term in this method is taken from the derivative-free Milstein-Talay scheme (16). The method is given by:

$$\begin{aligned} &K_0 := X_0, \quad K_1 := K_0 + \alpha\mu_1 h f(K_0), \\ &K_j := \mu_j \alpha h f(K_{j-1}) - \nu_j K_{j-1} - \kappa_j K_{j-2}, \\ &K_{s-1}^* := K_{s-2} + 2\tau_\alpha h f(K_{s-2}) + \sum_{r=1}^{m} g^r(K_s)\xi_r, \\ &X_1 := K_{s-2} + (2\sigma_\alpha - \frac{1}{2}) h f(K_{s-2}) + \frac{1}{2} h f(K_{s-1}^*) \\ &+ \frac{1}{2}\sum_{r=1}^{m} \left(g^r \left(K_s + \sum_{q=1}^{m} g^q(K_s) J_{q,r} \right) - g^r \left(K_s - \sum_{q=1}^{m} g^q(K_s) J_{q,r} \right) \right) \\ &+ \frac{1}{2}\sum_{r=1}^{m} \left(g^r \left(K_{s-1} + \sqrt{\frac{h}{2}} \sum_{q=1}^{m} g^q(K_s)\chi_q \right) + g^r \left(K_{s-1} - \sqrt{\frac{h}{2}} \sum_{q=1}^{m} g^q(K_s)\chi_q \right) \right) \xi_r, \end{aligned} \tag{20}$$

and α, σ_α and τ_α satisfy:

$$\begin{aligned} \alpha &:= 1/(2P'_{s-1}(0)), \qquad \sigma_\alpha := \frac{1-\alpha}{2} + 2\alpha\sigma \\ \tau_\alpha &:= \frac{(\alpha-1)^2}{2} + 2\alpha(1-\alpha)\sigma + \alpha^2\tau \end{aligned} \tag{21}$$

in which coefficients $\mu_j, \nu_j, \kappa_j, \sigma, \tau$ have the same meaning as in ROCK2 method, while random variables $J_{q,r}, \xi_r, \chi_r$ are the same as in (14),(15). This method has strong order of 1.

4 Adaptive timestepping approaches

In the previous section we described a method that has good stability properties and therefore is very suitable for solving stiff problems. We aim to improve this performance even further

by investigating different strategies for finding adequate time-step h at each iteration of our algorithm. For methods dealing with ordinary differential equations this problem is very well studied and there exist strategies that give drastic improvements over the fixed time-stepping approach. Unfortunately, in the case of SDE the problem of choosing good strategy proves to be more challenging, partially due to the increased complexity of formulas occurring in stochastic calculus, and partially due to the inevitable problem of sampling error showing up.

For the time-stepping strategy user usually needs to provide a parameter τ which we call tolerance, a function σ, as well as a suggestion for the first time-step h_0. Let's suppose that the algorithm has finished n steps $h_1, ...h_n$. Then, by studying previous values we make another step with h_{n+1} in such way that the error of the step is smaller[5] than $\sigma(\tau, h, X_0, ..., X_n)$. But we also want the time-step h_{n+1} to be as big as possible, so that we don't lose too much on the performance. In particular, the best performance is achieved when errors made at each time-step have values close to each other.

One of the first challenges when designing an adaptive strategy for SDE methods is measuring an error of a specific time-step. Currently, following two approaches are most common in the literature:

- Error measuring based on extrapolation,
- Error measuring based on asymptotic analysis of the method.

In the following two sections we analyze differences between the two approaches, after which we pick more suitable one.

4.1 Local error estimation based on extrapolation

The idea of error estimation by extrapolation consists in finding an asymptotic expression for the error of a method. This is done by using Taylor-Itô expansion of an exact solution in neighborhood of a point, and then comparing it to the method. Error estimators of this type have already been applied to Euler-Maruyama and Milstein-Talay [17, 19, 30] with success.

[5]Usually in literature σ is taken to depend only on the last approximation X_n. Here we want to use more general, so called control systems with memory, for getting more control on choosing new time-step.

Unfortunately, this type of estimators are not suitable for stabilized methods. This is because stabilized methods typically use larger time-steps, and therefore asymptotic analysis is not sharp.

4.2 Local error estimator based on embedded method

The idea of this strategy is to have two methods of different orders, and to consider the solution of the more precise method to be exact. Then, for the error estimation we can use the difference between these two values. Furthermore, we want to isolate error that are comes from integration of deterministic term from the error that comes from treatment of stochastic term, which can be useful for later analysis. Let us just denote with E_d deterministic error, and with E_s stochastic error of a time-step.

The inspiration for this treatment of the errors comes from the well known fact that for the most of Milstein-Talay based methods with global convergence order of τ, the local stochastic error needs to be $O(h^{\tau+\frac{1}{2}})$, while the deterministic error is usually in $O(h^{\tau+1})$. Thus, the two errors have different importance for convergence, with stochastic one having slightly more relaxed constraints. This gives rise to an idea of treating these errors differently in time-stepping scheme. Apart from this, we can also think about reducing number of Monte-Carlo simulations in case stochastic error turns out to be not very big, since then the sampling error will not be big as well.

Let us now proceed and explain how we can define stochastic and deterministic error and show that they have different asymptotic orders. For purpose of minimizing additional computations we will use existing stages of S-ROCK2 method to derive first order method. This is very well known idea of embedding, which is taken from the ODE error estimation theory. Finally, let us make a remark that for convergence of adaptive time-stepping scheme our method needs to have strong order of convergence 1, as noted in [9]. Since S-ROCK2 is indeed of first strong order of convergence, this condition is satisfied.

4.2.1 ROCK2 local error estimator

First we show how local error estimator can be introduced for deterministic problem, so we assume for now that $g = 0$, or equivalently that we are using ROCK2 method. Embedded

approach was already studied in original paper [1, eq. 27]. Unfortunately, method proposed there is not very suitable here because the final stages of S-ROCK2 are slightly adjusted to make handling stochastic terms easier, and thus we need a bit more calculations to reproduce embedded method from aforementioned paper then it is necessary. Thus, we propose a different scheme:

$$\begin{aligned}
&K_0 := X_0, \qquad K_1 := K_0 + \alpha\mu_1 hf(K_0),\\
&K_j := \alpha\mu_j hf(K_{j-1}) - \nu_j K_{j-1} - \kappa_j K_{j-2}, \qquad 2 \le j \le s-2,\\
&K^{**}_{s-1} := K_{s-2} + 2\tau_\alpha hf(K_{s-2}),\\
&X^d_1 := K_{s-2} + \left(2 - \frac{1}{2}\frac{\sigma_\alpha}{\tau_\alpha}\right)\sigma_\alpha hf(K_{s-2}) + \frac{1}{2}\frac{\sigma^2_\alpha}{\tau_\alpha} hf(K^{**}_{s-1}),
\end{aligned} \tag{22}$$

where $\alpha, \tau_\alpha, \sigma_\alpha$ is defined in the same way as for the S-ROCK2 scheme. Then, deterministic error estimator is

$$E_d = ||X^d_1 - X_1|| = \frac{1}{2}h\left(1 - \frac{\sigma^2_\alpha}{\tau_\alpha}\right)||f(K_{s-2}) - f(K^{**}_{s-2})|| = O(h^2). \tag{23}$$

4.2.2 Stochastic term local error estimator

Now we turn to estimation of local error that comes from integrating stochastic term. For this we need to use a method of strong order 1, deterministic order 2 and weak order 1. We also need to use method with same or better stability properties as S-ROCK2. Let us look at the scheme defined with:

$$\begin{aligned}
&K_0 := X_0, \qquad K_1 := K_0 + \alpha\mu_1 hf(K_0),\\
&K_j := \alpha\mu_j hf(K_{j-1}) - \nu_j K_{j-1} - \kappa_j K_{j-2}, \qquad 2 \le j \le s-2,\\
&K^{**}_{s-1} := K_{s-2} + 2\tau_\alpha hf(K_{s-2}),\\
&X^s_1 := K_{s-2} + \left(2\sigma_\alpha - \frac{1}{2}\right) hf(K_{s-2}) + \frac{1}{2}\frac{\sigma^2_\alpha}{\tau_\alpha} hf(K^{**}_{s-1}),\\
&+ \frac{1}{2}\sum_{r=1}^{m}\left(g^r\left(K_s + \sum_{q=1}^{m} g^q(K_s)J_{q,r}\right) - g^r\left(K_s - \sum_{q=1}^{m} g^q(K_s)J_{q,r}\right)\right),
\end{aligned} \tag{24}$$

and let us define a stochastic error estimator as $E_s = ||X^s_1 - X_1||$. Now, we analyze asymptotic behavior of E_s:

$$
\begin{aligned}
E_s \leq & \frac{1}{2}h||f(K_{s-1}^{**}) - f(K_{s-1}^{*})|| \\
& + \left|\left| \sum_{r=1}^{m} g^r(K_s)\xi_r - \frac{1}{2}\sum_{r=1}^{m}\left(g^r(K_{s-1} + \sqrt{\frac{h}{2}}\sum_{q=1}^{m} g^q(K_s)\chi_q) + g^r(K_{s-1} - \sqrt{\frac{h}{2}}\sum_{q=1}^{m} g^q(K_s)\chi_q)\right)\xi_r \right|\right| \\
\leq & \frac{1}{2}h\left|\left| \sum_{r=1}^{m} f'(X_0)g^r(X_0)\xi_r \right|\right| + \left|\left| \sum_{r=1}^{m}(g^r(K_s) - g^r(K_{s-1}))\xi_r \right|\right| \\
& + \left|\left| \sum_{r=1}^{m}(g^r)''(K_{s-1})\left(\sqrt{\frac{h}{2}}\sum_{p=1}^{m} g^p(K_s)\chi_p, \sqrt{\frac{h}{2}}\sum_{q=1}^{m} g^q(K_s)\chi_q \right)\xi_r \right|\right| + O(h^2) \\
\leq & \frac{1}{2}h\left|\left| \sum_{r=1}^{m} f'(X_0)g^r(X_0)\xi_r \right|\right| + \left|\left| \sum_{r=1}^{m}(g^r)'(K_s - K_{s-1})\xi_r \right|\right| \\
& + \frac{h}{2}\left|\left| \sum_{r=1}^{m}(g^r)''(K_{s-1})\left(\sum_{p=1}^{m} g^p(K_s)\chi_p, \sum_{q=1}^{m} g^q(K_s)\chi_q \right)\xi_r \right|\right| + O(h^2) \\
\leq & \frac{1}{2}h\left|\left| \sum_{r=1}^{m} f'(X_0)g^r(X_0)\xi_r \right|\right| + h\left|\left| \sum_{r=1}^{m}(g^r)'(\alpha(P_s'(0) - P_{s-1}'(0))f(X_0)\xi_r \right|\right| \\
& + \frac{h}{2}\left|\left| \sum_{r=1}^{m}(g^r)''(K_{s-1})\left(\sum_{p=1}^{m} g^p(K_s)\chi_p, \sum_{q=1}^{m} g^q(K_s)\chi_q \right)\xi_r \right|\right| + O(h^2) \\
\leq & \frac{1}{2}h\left|\left| \sum_{r=1}^{m} f'(X_0)g^r(X_0)\xi_r \right|\right| + h(\alpha P_s'(0) - \frac{1}{2})\left|\left| \sum_{r=1}^{m}(g^r)'f(X_0)\xi_r \right|\right| \\
& + \frac{h}{2}\left|\left| \sum_{r=1}^{m}(g^r)''(K_{s-1})\left(\sum_{p=1}^{m} g^p(K_s)\chi_p, \sum_{q=1}^{m} g^q(K_s)\chi_q \right)\xi_r \right|\right| + O(h^2).
\end{aligned}
\tag{25}
$$

Finally, noting that $\xi_r = O(\sqrt{h}), \chi_r = O(1), r = 1, \dots m$, we have that

$$
E_s = O(h^{3/2}). \tag{26}
$$

As mentioned earlier, it is important that method X_1^s has at least as good stability properties as X_1. Thus, for analyzing mean-square-stability, we apply it to the method (11) with $X(0) = 1$. We denote with $p = \lambda h, q = \sqrt{h}\mu$, and calculate :

$$
\begin{aligned}
K_0 = 1, \quad & K_1 = P_1(\alpha p), \\
K_j = P_j(\alpha p), \quad & j = 2, \dots s \\
K_{s-1}^{**} = (1 + 2\tau_\alpha p)P_{s-2}(\alpha p) & \\
X_1^s = w_\alpha(p)P_{s-2}(\alpha p) + q\xi P_s(\alpha p) + q^2\frac{\xi^2 - 1}{2}P_s(\alpha p) &
\end{aligned}
\tag{27}
$$

where $w_\alpha(p) = 1 + 2\sigma_\alpha p + \tau_\alpha p^2$ and . The embedded method is mean-square stable if $\mathbb{E}(|X_1^s|^2) \leq 1$, which is equivalent to

$$\mathbb{E}(|X_1^s|^2) = |w_\alpha(p)P_{s-2}(\alpha p)|^2 + |qP_s(\alpha p)|^2 + \frac{1}{2}|q^2 P_s(\alpha p)|^2 \leq 1. \tag{28}$$

Now, because for S-ROCK2 we have that

$$\mathbb{E}(|X_1|^2) = |w_\alpha(p)P_{s-2}(\alpha p)|^2 + |q(P_{s-1}(\alpha p) + \frac{p}{2}P_s(\alpha p))|^2 + \frac{1}{2}|q^2 P_s(\alpha p)|^2,$$

we are very confident that X_1^s is as stable as S-ROCK2. Thus, to conclude, we can use

$$\begin{aligned} E_d &= \frac{1}{2}h\left(1 - \frac{\sigma_\alpha^2}{\tau_\alpha}\right)||f(K_{s-2}) - f(K_{s-2}^{**})||, \\ E_s &= \frac{1}{2}h\Bigg|\Bigg|f(K_{s-1}^{**}) - f(K_{s-1}^{*}) + \sum_{r=1}^{m} g^r(K_s)\xi_r \\ &\quad - \frac{1}{2}\sum_{r=1}^{m}\left(g^r(K_{s-1} + \frac{1}{2}\sum_{q=1}^{m} g^q(K_s)\chi_q) + g^r(K_{s-1} - \frac{1}{2}\sum_{q=1}^{m} g^q(K_s)\chi_q)\right)\xi_r\Bigg|\Bigg|. \end{aligned} \tag{29}$$

For error estimation we need we need only 1 more evaluation of drift function f, assuming that all phases of S-ROCK2 method are evaluated. Thus, for a time-step with error estimation we need $s + 3$ evaluations of f, and $5m$ evaluations of g.

4.3 Problem of back-in-time sampling

Each time we sample a random variable during the execution of our algorithm, the distribution of the random variable will be conditioned on all past events. In the case of fixed time-stepping algorithm we don't need to care about this because the Wiener increment ΔW_r was independent from all the previous increments. But for the time-stepping algorithm in case of time-step rejection we will be interested in value $W_r(t_n) - W_r(t_{n-1})$ which is conditioned on the values of W_r that are drawn at $t > t_{n-1}$. These values must not be ignored, otherwise the simulation will be biased. The following theorem is crucial for finding a remedy to this problem:

Theorem 2. *Let $\{W(t); t \geq 0\}$ be a Wiener process. Fix $0 < t_0 < t_1 < \infty$ and $t \in]t_0, t_1[$. Then the random variable $W(t)$, conditioned on $W(t_0) = \alpha$ and $W(t_1) = \gamma$ is normally distributed with mean*

$$\alpha + (t - t_0)(\gamma - \alpha)/(t_1 - t_0)$$

and variance

$$(t-t_0)(t_1-t)/(t_1-t_0).$$

Proof. If we denote with $X = W(t) - W(t_0)$ and with $Y = W(t_1) - W(t_0)$, we have that the probability density function for X can be calculated using

$$f_X(X = x|Y = y) = \frac{f_{X,Y}(X = x, Y = y)}{f_Y(Y = y)}, \tag{30}$$

where f_X, f_Y are probability density functions for random variable X, Y, while $f_{X,Y}$ is a joint density of X and Y. Now, let us assume that no Brownian motion W was sampled after t_1, so we know that the increment X does not depend on history, while the increment Y depends only on sampled value of X. Thus, the joint density $f_{X,Y}$ and probability density function f_Y can be calculated as :

$$\begin{aligned} f_{X,Y}(X = x, Y = y) =& \frac{1}{\sqrt{2\pi(t-t_0)}} e^{-\frac{x^2}{2(t-t_0)}} \frac{1}{\sqrt{2\pi(t_1-t)}} e^{-\frac{(y-x)^2}{2(t_1-t)}}, \\ f_Y(Y = y) =& \frac{1}{\sqrt{2\pi(t_1-t_0)}} e^{-\frac{y^2}{2(t_1-t_0)}}. \end{aligned} \tag{31}$$

Now, putting this values into equation (30) gives us that

$$f_X(X = x|Y = y) = \frac{1}{\sqrt{2\pi\frac{(t_1-t)(t_1-t)}{(t_1-t_0)}}} e^{\frac{\left(x-\frac{(t-t_0)y}{t_1-t_0}\right)^2}{2\frac{(t_1-t)(t_1-t)}{(t_1-t_0)}}}, \tag{32}$$

so we have that X conditioned on $Y = y$ is normally distributed with mean $\frac{(t-t_0)y}{t_1-t_0}$ and variance $\frac{(t_1-t)(t_1-t)}{(t_1-t_0)}$. Now, noting that $Y = W(t_1) - W(t_0) = \gamma - \alpha$ and $X = W(t) - W(t_0) = W(t) - \alpha$ we see that we have exactly showed the result from the statement.

This proof was done in the case no values of W were sampled after t_1. In case we have values sampled after t_1, we can prove the theorem using induction on the number of sampled values after t_1 and using the same idea, with just the fact that mean of Y and X will not be zero anymore. □

This theorem can not be directly applied to the proposed derivative-free S-ROCK2 method (20) since there we have not used Brownian motion but discrete random variables ξ_r, χ_r. To see how this can be achieved, let us look at the second order Milstein-Talay method (13) in which all stochastic terms are expressed in function of Wiener stochastic

process. The distribution of these Wiener stochastic processes are known from the previous theorem. Then, for integration of (double) Itô integrals we can use Karhunen-Loève or Fourier expansion. But both approaches seem to be very expensive.

Since we want to study effect of time-stepping strategy on precision of a method, we can concentrate here just on problems with one-dimensional noise. For them, double Itô integrals $I_{q,p}$ are not required, but only integrals $I_p = W(t_1) - W(t_0)$ as well as $I_{p,p} = \frac{(W(t_1)-W(t_0))^2-h}{2} = \frac{(I_p)^2-h}{2}$. And for these integrals theorem 2 gives us enough information to do the time-stepping scheme. In particular, our random variable ξ_r is actually approximation of I_r, so we can can always switch between Wiener increment $\Delta W_r \sim \mathcal{N}(\mu, \sigma^2)$ and discrete random variable ξ_r, where $\mathbb{P}(\xi_r = \mu \pm \sqrt{3}\sigma) = 1/6, \mathbb{P}(\xi_r = \mu) = 2/3$.

When calculating Brownian increments $W(t_1) - W(t_0)$ we need to know the history of sampling stochastic process W, in order to apply the theorem 2. Actually, we will already know the value of $W(t_0)$ during simulation since it will be a sampled during previous accepted iteration or it will be a starting point. For calculation of $W(t_1)$ we need information about all possible samples taken in the interval $[t_0, t_1]$, as well as information about first sample taken after t_1(if it exists). Actually, we need only to know the value of $W(t_0)$, and values of first sample before t_1(at some time $t_1^- < t_1$) and after point t_1(at the moment $t_1^+ > t_1$). For quick searching the values $W(t_0)$, as well as values $W(t_1^-)$, $W(t_1^+)$, it is natural to use binary (balanced) tree. The tree contains all sampled values $W(t)$ sorted by t.

Each time we accept time-step on interval $[t_0, t_1]$, we can erase all history before t_1, since from now on we will be sampling only values of Brownian motion for $t > t_1$, which will not depend on any sample from before t_1. Using this approach will significantly decrease the size of the binary tree during the simulation. Also, since the number of rejections is rather small compared to the number of accepted steps, average size of the three in this case we be very small(usually just containing one or two nodes). Thus, the use of binary tree is not real necessity for this algorithm.

4.3.1 Using Brownian trees for integrating non-commutative problems

Finally, let us briefly explain an approach that can be used for problems with non-commutative noise. This approach uses so called Brownian trees, that were introduced in [9]. The ap-

proach originally presented is relying on a relationship of Lévy

$$\begin{aligned}\Delta W_{k,j} &= W\left(\frac{(k+1)T}{2^j}\right) - W\left(\frac{kT}{2^j}\right), \quad j \in \mathbb{N}, 0 \leq k < 2^j,\\ \Delta W_{2k-1,j} &= \frac{1}{2}\Delta W_{k,j} - y_{k,j},\\ \Delta W_{2k,j} &= \frac{1}{2}\Delta W_{k,j} + y_{k,j},\end{aligned} \tag{33}$$

to calculate Brownian increments on $[0,T]$. The obvious drawback of this method is the fact that it only allows for doubling or halving the time-step during the simulation. But it is not hard to extend this method to allow for more choices on time-step update.

Here we want to propose another approach that is inspired by the Brownian trees, and that will allow for implementation of adaptive time-stepping methods even for multidimensional problems with general(non-commutative) noise. Let us first fix the depth of a Brownian three, so that j mentioned in (33) can be only as big as some $b \in \mathbb{N}$. This b will be very small, typically $b < 22$. Also, for simplicity let us assume as before that we want to integrate equation over time interval $[0,T]$, and let us denote with $t_{k,j} = \frac{kT}{2^j}$, and also:

$$\begin{aligned}I_r^{k,j} &= W^r(t_{k+1,j}) - W^r(t_{k,j}), \qquad I_{r,0}^{k,j} = \int_{t_{k,j}}^{t_{k+1,j}} \int_{t_{k,j}}^{t} dW_r(s)dt,\\ I_{0,r}^{k,j} &= \int_{t_{k,j}}^{t_{k+1,j}} \int_{t_{k,j}}^{t} ds\, dW_r(t), \qquad I_{q,r}^{k,j} = \int_{t_{k,j}}^{t_{k+1,j}} \int_{t_{k,j}}^{t} dW_q(s)dW_r(t).\end{aligned} \tag{34}$$

Assuming that there is no history of sampling, random variables $I_r^{k,j}, I_{r,0}^{k,j}, I_{r,0}^{k,j}, I_{q,r}^{k,j}$ can be approximated by $J_r^{k,j}, J_{r,0}^{k,j}, J_{r,0}^{k,j}, J_{q,r}^{k,j}$ that are defined in similar way as in (14,15)

$$\begin{aligned}h_j = \frac{T}{2^j}, \quad \mathbb{P}(\chi_r^{k,j} = \pm 1) = 1/2, \quad \mathbb{P}(\xi_r^{k,j} = \pm\sqrt{3h_j}) = 1/6, \quad \mathbb{P}(\xi_r^{k,j} = 0) = 2/3,\\ J_r^{k,j} = \xi_r^{k,j}, \quad J_{0,r}^{k,j} = J_{r,0}^{k,j} = h_j \xi_r^{k,j}\\ J_{q,r}^{k,j} = \begin{cases} (\xi_r^{k,j}\xi_r^{k,j} - h_j)/2 & \text{if } q = r,\\ (\xi_q^{k,j}\xi_r^{k,j} - h_j\chi_q^{k,j})/2 & \text{if } r < q,\\ (\xi_q^{k,j}\xi_r^{k,j} + h_j\chi_r^{k,j})/2 & \text{if } r > q.\end{cases}\end{aligned} \tag{35}$$

Before the start of simulation, we can evaluate all variables in (35) for $j = b$, because they are all defined at disjoint intervals, and are thus independent. Once these values are calculated, we can start simulation, and calculate values of $J_r^{k,j}, J_{r,0}^{k,j}, J_{r,0}^{k,j}, J_{q,r}^{k,j}$ as necessary from the

values at the bottom of the tree. We can construct values of J only by "going up" in the tree, by noting that following formulas hold for Itô integrals

$$\begin{aligned} I_r^{k,j} &= I_r^{2k-1,j+1} + I_r^{2k,j+1}, \quad I_{r,0}^{k,j} = I_r^{2k-1,j+1} + I_r^{2k,j+1} + h_{j+1} I_r^{2k-1,j+1} \\ I_{0,r}^{k,j} &= I_{0,r}^{2k-1,j+1} + I_{0,r}^{2k,j+1} + h_{j+1} I_r^{2k,j+1}, I_{q,r}^{k,j} = I_{q,r}^{2k-1,j+1} I_{q,r}^{2k,j+1} + I_q^{2k-1,j+1} I_r^{2k,j+1}, \end{aligned} \tag{36}$$

and using them for approximations of these integrals $J_r^{k,j}, J_{r,0}^{k,j}, J_{r,0}^{k,j}, J_{q,r}^{k,j}$. This way any necessary Itô integral can be constructed from leafs of the tree.

The drawback of this strategy is that it constrains the adjustment of time-step to halving and doubling, and it also introduces constraint on smallest possible step-size. Another flaw of this strategy is that it requires computation of a lot random variables before the start of simulation, and it is expected that most of these variables will never be used. On the other side, generation of the leafs of the tree is completely parallelizable, and it can be generated well before the start of simulation. Brownian tree proposed in this section is also reusable, in case we are evaluating different problems. Finally, this approach seems promising compared to very expensive approach recommended in [9] which suggests solving double Itô integrals by introducing fine discretization over domain of integration and using Brownian increments to approximate integral similar to classical integral approximation methods like trapezoid rule etc. It would be also interesting to see how well this approach compares to using Fourier and Karhunen-Loève expansions for solving conditional double Itô integrals.

4.4 Step-size control algorithm

Finally, the idea of treating deterministic and stochastic error differently is following: in the case of embedded method error estimation the local asymptotic bound of deterministic error is $O(h^2)$, while the local stochastic error is $O(h^{\frac{3}{2}})$. For very small h, the deterministic error will be negligible to the stochastic one, and vice-versa for bigger values of h. Thus, we might want to define error as

$$E = E_d + E_s^{4/3}, \tag{37}$$

so that $E = O(h^2)$. Next, for the tolerance τ we define function σ from beginning of this chapter as:

$$\sigma(\tau, h_n, X_{n-1}, X_n) = max(|X_n|, |X_{n-1}|) \cdot \tau + \tau. \tag{38}$$

In case $E < \sigma(\tau, h, X_{n-1}, X_n)$, we accept the time-step and chose new time-step in the same manner as suggested in [1]:

$$h_{n+1} = fac \cdot h_n \left(\frac{1}{err_n}\right)^{1/2} \frac{h_n}{h_{n-1}} \left(\frac{err_{n-1}}{err_n}\right)^{1/2}, \tag{39}$$

where $err_n = E_n/\sigma(\tau, h_n, X_{n-1}, X_n)$. The idea of using memory to control step-sizes is originally presented by Watts [31] and Gustafsson[11]. Here $fac = 0.8$ is introduced to slightly decrease next time-step so that we can hopefully avoid time-step rejection at the next step.

In case we reject the iteration, we reintegrate equation for new time-step of size:

$$\tilde{h}_n = h_n \cdot \sqrt{\frac{1}{err_n}}.$$

We also make sure never to pick time-step h_{n+1} bigger then some predefined constant h_{max}.

Finally, we also try another step-size control algorithm, that was proposed in [18]. For this, we introduce $k = \gamma + 1/2$, where $\gamma = 2$ is global order of convergence for the method of choice. Then, the time-steps are updated using

$$h_{n+1} = h_n \left(\frac{fac}{err_n}\right)^{k_I + k_P} \left(\frac{err_{n-1}}{fac}\right)^{k_P}, \tag{40}$$

where $(k_I, k_P) = \frac{1}{k}(0.101, 0.009)$. Both approaches are tested and analyzed in the numerical results section.

5 Numerical results

In this section we test the developed algorithm. We are specifically interested in studying the benefits of introducing time-stepping scheme in comparison to the S-ROCK2 method with fixed time-steps.

For measuring the performance of the method we assume that the most expensive operations are evaluations of functions f and g, and thus it is enough to count evaluations of these to measure the performance. Other strategy to evaluate performance would be to measure the execution time, but this statistic can be inaccurate since it depends on the computational power assigned to the program doing the simulation.

The stiff problem on which we do our tests is given by equation

$$dX(t) = \left(\frac{1}{4}X(t) + \frac{1}{2}\sqrt{X(t)^2+1}\right)dt + \sqrt{\frac{X(t)^2+1}{2}}dW(t), \quad X(0) = 0. \tag{41}$$

The exact solution to this problem is $X(t) = \sinh(\frac{t}{2} + W(t)/\sqrt{2})$. We test errors against $\phi(x) = \text{arcsinh(x)}^2$. For such chosen ϕ, we have

$$\mathbb{E}\left((\text{arcsinh}(X(t))^2\right) = \frac{t^2}{4} + \frac{t}{2}.$$

In the following tests we assume that the stopping time of the equation is $T = 1$.

First, we want to show numerically that error due to bias really occurs if we don't adjust mean and variance of Brownian motion in case of time-step rejection in a way proposed in theorem 2. Thus, we run adaptive time-stepping scheme with and without bias. Results are shown in figure 4.

Figure 4: Analysis of biased(red) vs unbiased(blue) time-stepping scheme. Precision is plotted against number of f function evaluations. Simulations are run over 10^8 realizations of Brownian path.

Next, we want to check which of the step-size control algorithms suggested in section 4.4 behaves better. We run again numerical tests for the same problem and using same set of tolerances. The figure showing performance of two algorithms is given in figure 5. Once again, we plot error of a method against the number evaluations of function f.

Figure 5: Performance of two different step-size control algorithms. Algorithm (39) is presented in blue, while (40) is drawn in red color. Algorithms are tested for problem (41) by using 10^8 Monte-Carlo simulations.

Now, we want to test if adaptive time-stepping scheme brings improvement of performance over the fixed time-stepping approach. For this, we run fixed time-stepping scheme for different time-step sizes h, with $h \in 1/\{1, 2, 4, 8, 16\}$. For adaptive scheme, we run algorithms for tolerance $\tau = \{1, 2, 4, 6, 8\}/1000$. As before, we average the results over 10^8 Monte-Carlo simulations. Performance comparison between two schemes can be observed from figure 6. From this graph we conclude that adaptive time-stepping algorithm doesn't introduce any performance improvement over fixed time-stepping method.

Figure 6: Performance graph for fixed and adaptive time-stepping schemes. Fixed time-stepping algorithm is presented in blue, while adaptive method (40) is drawn in red color. Algorithms are tested for problem (41) by using 10^8 Monte-Carlo simulations.

6 Conclusion

In this paper first adaptive scheme for S-ROCK family of algorithms was developed. As could have been seen, first results show that time-stepping scheme doesn't seem to introduce any improvement. The reasons for this problem can be various. First of all, step-size control needs to be studied

References

[1] A. Abdulle & A. Medovikov, *Second order Chebyshev methods based on orthogonal polynomials*, Numer. Math., 90(2001), pp 1-18

[2] A. Abdulle, *On roots and error constants of optimal stability polynomials*, BIT, 40 (2000), pp. 177-182.

[3] A. Abdulle, *Fourth order Chebyshev methods with recurrence relation,* SIAM J. Sci. Comput., 23 (2002), pp. 2041-2054.

[4] A. Abdulle, G.Vilmart, K.Zygalakis, *Weak second order explicit stabilized methods for stiff stochastic differential equations.* SIAM J. Sci. Comput. 35(4), 1792–1814 (2013)

[5] A. Abdulle, T. Li, *S-ROCK methods for stiff Ito SDEs,* Comm. Math. Sci., 6 (2008)

[6] L. Arnold, *Stochastic Differential Equations: Theory and Applications*, John Wiley & Sons, New York, 1974

[7] K. Burrage, P.M. Burrage, *A variable stepsize implementation for stochastic differential equations*, SIAM J. Sci. Comput. 24(3) (2002) 848 – 864.

[8] K. Burrage, P.M. Burrage, T. Tian, *Numerical methods for strong solutions of stochastic differential equations: an overview*, Proc. R. Soc. Lond. A 460 (2041) (2004) 373 – 402.

[9] J.G. Gaines, and T.J. Lyons, *Variable Step Size Control in the Numerical Solution of Stochastic Differential Equations*,SIAM J. Appl. Math., 57(5), 1455–1484.

[10] P. Glasserman, *Monte Carlo methods in financial engineering*, Springer, New York, 2010.

[11] K. Gustafsson, *Control-theoretic techniques for stepsize selection in implicit Runge-Kutta methods*, ACM Trans. Math. Soft. 20, 496-517(1994)

[12] E. Hairer, C. Lubich, G. Wanner, *Geometric numerical integration. Structure-preserving algorithms for ordinary differential equations. 2nd ed.* Vol. 31. Springer Series in Computational Mathematics. Berlin: Springer-Verlag, 2006, pp. xviii+644.

[13] N. Hofmann, T. Müller-Gronbach, K. Ritter, *The optimal discretization of stochastic differential equations*, J. Complexity 17 (2001) 117 – 153.

[14] P. J. Van der Houwen, B. P. Sommeijer , *On the internal stability of explicit, m-stage Runge–Kutta methods for large m-values.* Z. Angew. Math. Mech. 60, 479–485 (1980)

[15] N.G. van Kampen, *Stochastic processes in physics and chemistry*, Amsterdam, North-Holland, 2007.

[16] P. Kloeden, E. Platen, *Numerical Solution of Stochastic Differential Euations,* Springer-Verlag, Berlin, New York, 1992.

[17] D. Küpper, Jürgen Lehn, Andreas Rößler, *A step size control algorithm for the weak approximation of stochastic differential equations*, Numer. Algor. (2007) 44:335-346

[18] S. Ilie, K.R. Jackson, W.H. Enright, *Adaptive time-stepping for the strong numerical solution of stochastic differential equations*, Numer Algor (2015) 68: 791.

[19] H. Lamba, *An adaptive timestepping algorithm for stochastic differential equations*, Journal of Computational and Applied Mathematics, 161 (2003) 417-430

[20] V.I. Lebedev, *Zolotarev polynomials and extremum problems.* Russian J. Numer. Anal. Math. Modelling, 9 (1994), pp. 231–263.

[21] V.I. Lebedev, *How to solve stiff systems of differential equations by explicit methods*,Numerical Methods and Applications, pp. 45–80. Boca Raton :CRC Press, 1994

[22] V.I. Lebedev, *A new method for determining the zeros of polynomials of least deviation on a segment with weight and subject to additional conditions.* Part I, Part II. Russian J. Numer. Anal. Math. Modelling. V. 8, N 3 (1993), pp. 195–222; V. 8, N 5 (1993), pp.397–426

[23] T. Li, A. Abdulle and E. Weinan, *Effectiveness of implicit methods for stiff stochastic differential equations*, Commun. Comput. Phys., 3(2008), no. 2, 295-307.

[24] G. Maruyama, *Continuous markov processes and stochastic equations*, Rend. Circ. Mat. Palermo, 4(0):48–90, 1955.

[25] S. Mauthner, *Step size control in the numerical solution of stochastic differential equations*, J. Comput. Appl. Math. 100 (1998) 93 – 109.

[26] A.A. Medovikov, *High order explicit methods for parabolic equations.* BIT 38, 372–390 (1998)

[27] G. N. Milstein, *Weak approximation of solutions of systems of stochastic differential equations.* Theory Probab. Appl., 30 (1986), pp. 750-766

[28] G. N. Milstein and M. V. Tretyakov, *Stochastic Numerics for Mathematical Physics*, Scientific Computing, Springer-Verlag, Berlin, New York, 2004

[29] D.Talay, *Efficient numerical schemes for the approximation of expectations of functionals of the solutions of a SDE and applications,* in Filtering and Control of Random Processes, Lecture Notes in Control and Inform. Sci. 61, Springer, Berlin, pp. 294-313.

[30] V. Sotiropoulos, Y. N. Kaznessis, *An adaptive time step scheme for a system of stochastic differential equations with multiple multiplicative noise: Chemical Langevin equation, a proof of concept*, The Journal of Chemical Physics 128 (2008) 014103

[31] H.A. Watts, *Step size control in ordinary differential equation solvers*, Trans. Soc. Computer Simulation 1, 15-25 (1984)