Stability-Optimized High Order Methods and Stiffness Detection for Pathwise Stiff Stochastic Differential Equations

Chris Rackauckas and Qing Nie

January 2, 2018

1 Introduction

The unique features of stochastic models in many cases are pathwise-dependent. An ODE model of a chemical reaction network may stay at a constant steady state, but in the presence of randomness this may be switching between various steady states. These pathwise properties are of interest because they capture effects which cannot be found in deterministic models. However, these same effects cause numerical difficulties. A minimal example of these properties is demonstrated in the equation

$$dX_t = \left[-1000X_t\left(1-X_t\right)\left(2-X_t\right)\right]dt + g(t,X_t)dW_t,\;\; X_0 = 2,\;\; t\in[0,5]\,. \tag{1}$$

with additive noise $g(t,X_t) = 10$ where a sample trajectory is shown in Figure 1. This equation has two stable steady states, one at $X = 0$ and another at $X = 2$, which the solution switches between when the noise is sufficiently large. While near a steady state the derivative is near zero making the problem non-stiff, during these transitions the derivative of the deterministic portion reaches a maximum of 1000. This means in order to be stable, explicit Stochastic Runge-Kutta (SRK) must have a small Δt. This display of large, transient and random switching behavior in a given trajectory causes stochastic bursts of numerical stiffness, a phenomena which we will denote pathwise stiffness. The fixed time step Euler-Maruyama method would require $dt < 10^{-4}$ to be stable for most trajectories, thus requiring greater than 5×10^4 steps to solve this seemingly simple equation. In many cases the switching behavior can be rare (due to smaller amounts of noise) or can happen finitely many times like in the multiplicative noise version with $g(t,X_t) = 10X_t$, yet these switches dominate the computational cost of the problem. While implicit methods can be used to increase the stability range to handle the equation, this can immensely increase the overall computational cost of each step, especially in the case large systems of SDEs like discretizations of stochastic reaction-diffusion equations. In addition, implicit solvers have in practice a smaller stability region than the theory due to requiring convergence of the quasi-Newton solvers for the implicit steps. This problem is mitigated in ODE software by high-quality stage predictors given by extrapolation algorithms which give good initial conditions for the Newton steps [12]. However, there are no known algorithms for stage predictors when the large bursts of noise occur and thus implicit solvers have an additional form of instability. Thus (we will show) both fixed time step explicit and implicit solvers are inadequate for handling this common class of SDEs.

Since these features exist in the single trajectories of the random processes, methods which attempt to account for the presence of such bursts must do so on each individual trajectory in order to be efficient. In previous work, the authors have shown that by using adaptive time-stepping a stochastic reaction network of 19 reactants is able to be solve with an average time step 100,000x larger than the value that was found

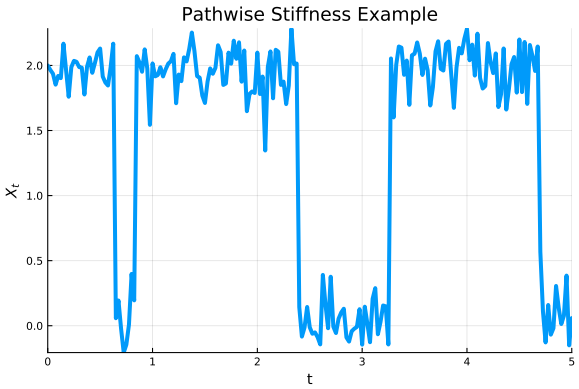

Figure 1: Example of a pathwise stiff solution. Depicted is a sample trajectory of Equation 1 solved using the SOSRI methods developed in this manuscript. 1

necessary for stability during the random stiff events for a high order SRK method [28]. This demonstrated that the key to solving these equations efficiently required controlling the time steps in a pathwise manner. However, the methods were still largely "stability-bound", that is the tolerance was set to solve the model was determined by what was necessary for stability but was far below the error necessary for the application. The purpose of this investigation is to develop numerical methods with the ability to properly handle pathwise stiffness and allow for efficient solving of large Monte Carlo experiments.

We approach this through four means. On one end we develop adaptive stability-optimized SRK methods with enlarged stability regions. This builds off of similar work for ODE integrators which optimize the coefficients of a Butcher tableau to give enhanced stability [23, 2, 38]. Similar to the Runge-Kutta Chebyschev methods [12] (and the S-ROCK extension to the stochastic case [21, 1, 20]), these methods are designed to be efficient for equations which display stiffness without fully committing to implicit solvers. Given the complexity of the stochastic stability equations and order conditions, we develop a novel scalable mechanism for the derivation of "optimal" Runge-Kutta methods. We use this method to design stability-optimized methods for additive noise and diagonal noise SDEs. We show through computational experiments that these adaptive stability-optimized SRK methods can adequately solve transiently stiff equations without losing efficiency in non-stiff problems and thus serve as a good default for software in problem solving environments (PSEs). On the other hand, to handle extreme stiffness we develop implicit RK methods for additive noise SDEs. We extend the definition of L-stability to additive noise SDEs and develop two strong order 1.5 methods: a fully implicit 2-stage L-stable method and an extension of the a well-known L-stable explicit first stage singly diagonally implicit RK (ESDIRK) method due to Kennedy and Carpenter which is commonly used for convection-diffusion-reaction equations [18]. To the author's knowledge, these are the first high order adaptive L-stable methods for SDEs. In addition, to extend the utility of these additive noise methods, we derive an extension of the methods for additive SDEs to affine SDEs (mixed multiplicative and additive noise terms) through a Lamperti transformation [25]. Lastly, in addition to the new methods, in order to handle extreme transient stiffness, for each of these types of methods we derive computationally cheap methods for detecting stiffness and switching between implicit and explicit integrators in the presence of stiffness. We show that these methods can robustly detect pathwise stiff transients and thus can serve as the basis for automatic switching methods for SDEs. Together we test on non-stiff, semi-stiff, and stiff equations with 2 to $6 \times 20 \times 100$ SDEs from biological literature and show speedups between

6x-60x over the previous adaptive SRIW1 algorithm, and demonstrate the infeasibility of common explicit and implicit methods (Euler-Maruyama, Runge-Kutta Milstein, Drift-Implicit Euler-Maruyama, and Drift-Implicit Runge-Kutta Milstein) found as the basis of many SDE solver packages [32, 10, 16].

2 Adaptive Strong Order 1.0/1.5 SRK Methods for Additive and Diagonal Noise SDEs

Take stochastic differential equations of the general form:

$$dX_t = f(t, X_t)dt + g(t, X_t)dW_t$$

where X_t is a d-dimensional vector, where $f : \mathbb{R}^d \to \mathbb{R}^d$ is the drift coefficient and $g : \mathbb{R}^d \to \mathbb{R}^{d\times m}$ is a matrix equation known as the diffusion coefficient which describes the amount and mixtures of the noise process W_t which is a m-dimensional Brownian motion. In many models, noise is added to deterministic equations phenomenologically. These models are numerically simulated for their qualitative behavior: scientists are interested not in their high accuracy numerical predictions but to understand the general phenomena which only occurs in the presence of stochasticity, like random switching between cell types. In these cases, the noise process can be modeled as exogenous to the system and thus not dependent the system itself, leading to the assumption that $g(t, X_t) \equiv g(t)$ which is known as additive noise. Another common case is multiplicative noise, where to each deterministic equation a noise term $\sigma_i X_t^i dW_t$ is added to give proportional amounts of randomness added to the dynamics. This results in $g(t, X_t)$ being the diagonal matrix $\left(\sigma_i X_t^i\right)$ and thus falls into the more general category of diagonal noise.

The class of methods we wish to study are the adaptive strong order 1.5 SRK methods [30, 28]. The diagonal noise methods utilize the same general form and order conditions as the methods for scalar noise so we use their notation for simplicity. The strong order 1.5 methods for scalar noise are of the form

$$X_{n+1} = X_n + \sum_{i=1}^{s} \alpha_i f\left(t_n + c_i^{(0)}h, H_i^{(0)}\right) + \sum_{i=1}^{s}\left(\beta_i^{(1)} I_{(1)} + \beta_i^{(2)}\frac{I_{(1,1)}}{\sqrt{h}} + \beta_i^{(3)}\frac{I_{(1,0)}}{h} + \beta_i^{(4)}\frac{I_{(1,1,1)}}{h}\right) g\left(t_n + c_i^{(1)}h\right) \tag{2}$$

with stages

$$H_i^{(0)} = X_n + \sum_{j=1}^{s} A_{ij}^{(0)} f\left(t_n + c_j^{(0)}h, H_j^{(0)}\right) h + \sum_{j=1}^{s} B_{ij}^{(0)} g\left(t_n + c_j^{(1)}h, H_j^{(1)}\right)\frac{I_{(1,0)}}{h} \tag{3}$$

$$H_i^{(1)} = X_n + \sum_{j=1}^{s} A_{ij}^{(1)} f\left(t_n + c_j^{(0)}h, H_j^{(0)}\right) h + \sum_{j=1}^{s} B_{ij}^{(1)} g\left(t_n + c_j^{(1)}h, H_j^{(1)}\right)\sqrt{h}$$

where the I_j are the Wiktorsson approximations to the iterated stochastic integrals [39]. In the case of additive noise, this reduces to the form

$$X_{n+1} = X_n + \sum_{i=1}^{s} \alpha_i f\left(t_n + c_i^{(0)}h, H_i^{(0)}\right) + \sum_{i=1}^{s}\left(\beta_i^{(1)} I_{(1)} + \beta_i^{(2)}\frac{I_{(1,0)}}{h}\right) g\left(t_n + c_i^{(1)}h\right) \tag{4}$$

with stages

$$H_i^{(0)} = X_n + \sum_{j=1}^{s} A_{ij}^{(0)} f\left(t_n + c_j^{(0)}h, H_j^{(0)}\right) h + \sum_{j=1}^{s} B_{ij}^{(0)} g\left(t_n + c_j^{(1)}h\right)\frac{I_{(1,0)}}{h}. \tag{5}$$

The tuple of coefficients $\left(A^{(j)}, B^{(j)}, \beta^{(j)}, \alpha\right)$ thus fully determines the SRK method. These coefficients must satisfy the constraint equations described in Appendix 10.1 in order to receive strong order 1.5. These methods are appended with error estimates

$$\begin{aligned} E_D &= \left|\Delta t \sum_{i\in I_1}(-1)^{\sigma(i)} f\left(t_n + c_i^{(0)}\Delta t, H_i^{(0)}\right)\right| \text{ or } E_D = \Delta t \sum_{i\in I_1}\left|f\left(t_n + c_i^{(0)}\Delta t, H_i^{(0)}\right)\right| \\ E_N &= \left|\sum_{i\in I_2}\left(\beta_i^{(3)}\frac{I_{(1,0)}}{\Delta t} + \beta_i^{(4)}\frac{I_{(1,1,1)}}{\Delta t}\right) g\left(t_n + c_i^{(1)}\Delta t, H_i^{(1)}\right)\right| \end{aligned}$$

and the rejection sampling with memory (RSwM) algorithm to give it fully adaptive time-stepping [28]. Thus unlike in the theory of ordinary differential equations [22, 8, 6, 37, 36], the choice of coefficients for SRK methods does not require explicitly finding an embedded method when developing an adaptive SRK method and we will therefore take for granted that each of the derived methods is adaptive.

3 Optimized-Stability High Order SRK Methods with Additive Noise

We use a previous definition of a discrete approximation as numerically stable if for any finite time interval $[t_0, T]$, there exists a positive constant Δ_0 such that for each $\epsilon > 0$ and each $\delta \in (0, \Delta_0)$

$$\lim_{\left|X_0^\delta - \bar{X}_0^\delta\right|\to 0} \sup_{t_0\le t\le T} P\left(\left|X_t^\delta - \bar{X}_t^\delta\right| \ge \epsilon\right) = 0 \tag{6}$$

where X_n^δ is a discrete time approximation with maximum step size $\delta > 0$ starting at X_0^δ and $\bar{X}_n^\delta$ respectively starting at $\bar{X}_n^\delta$ [19]. For additive noise, we consider the complex-valued linear test equations

$$dX_t = \mu X_t dt + dW_t \tag{7}$$

where μ is a complex number. In this framework, a scheme which can be written in the form

$$X_{n+1}^h = X_n^h G\left(\mu h\right) + Z_n^\delta \tag{8}$$

with a constant step size $\delta \equiv h$ and Z_n^δ are random variables which do not depend on the Y_n^δ, then the region of absolute stability is the set where for $z = \mu h$, $|G(z)| < 1$.

The additive SRK method can be written as

$$X_{n+1}^h = X_n^h + z\left(\alpha \cdot H^{(0)}\right) + \beta^{(1)}\sigma I_{(1)} + \sigma\beta^{(2)}\frac{I_{(1,0)}}{h} \tag{9}$$

where

$$H^{(0)} = \left(I - zA^{(0)}\right)^{-1}\left(\hat{X_n^h} + B^{(0)}e\sigma\frac{I_{(1,0)}}{h}\right) \tag{10}$$

where $\hat{X_n^h}$ is the size s constant vector of elements X_n^h and $e = (1,1,1,1)^T$. By substitution we receive

$$X_{n+1}^h = X_n^h\left(1 + z\left(\alpha \cdot \left(I - zA^{(0)}\right)^{-1}\right)\right) + \left(I - zA^{(0)}\right)^{-1} B^{(0)}e\sigma\frac{I_{(1,0)}}{h} + \beta^{(1)}\sigma I_{(1)} + \sigma\beta^{(2)}\frac{I_{(1,0)}}{h} \tag{11}$$

This set of equations decouples since the iterated stochastic integral approximation I_j are random numbers and are independent of the X_n^h. Thus the stability condition is determined by the equation

$$G(z) = 1 + z\alpha \cdot \left(I - zA^{(0)}\right)^{-1} \tag{12}$$

which one may notice is the stability equation of the drift tableau applied to a deterministic ODE [5]. Thus the stability properties of the deterministic Runge-Kutta methods carry over to the additive noise SRA methods on this test equation. However, most two-stage tableaus from ODE research were developed to satisfy higher order ODE order constraints which do not apply. Thus we will instead look to maximize stability while satisfying the stochastic order constraints.

3.1 Explicit Methods for Non-Stiff SDEs with Additive Noise

3.1.1 Stability-Optimal 2-Stage Explicit SRA Methods

For explicit methods, $A^{(0)}$ and $B^{(0)}$ are lower diagonal and we receive the simplified stability function

$$G(z) = 1 + A_{21}z^2\alpha_2 + z\left(\alpha_1 + \alpha_2\right) \tag{13}$$

for a two-stage additive noise SRK method. For this method we will find the method which optimizes the stability in the real part of z. Thus we wish to find $A^{(0)}$ and α s.t. the negative real roots of $|G(z)| = 1$ are minimized. By the quadratic equation we see that there exists only a single negative root: $z = \frac{1-\sqrt{1+8\alpha_2}}{2\alpha_2}$. Using Mathematica's minimum function, we determine that the minimum value for this root subject to the order constraints is $z = \frac{3}{4}\left(1 - \sqrt{\frac{19}{3}}\right) \approx -1.13746$. This is achieved when $\alpha = \frac{2}{3}$, meaning that the SRA1 method due to Rossler achieves the maximum stability criteria. However, given extra degrees of freedom, we attempted to impose that $c_1^{(0)} = c_1^{(1)} = 0$ and $c_2^{(0)} = c_2^{(1)} = 1$ so that the error estimator spans the whole interval. This can lead to improved robustness of the adaptive error estimator. In fact, when trying to optimize the error estimator's span we find that there is no error estimator which satisfies $c_2^{(0)} > \frac{3}{4}$ which is the span of the SRA1 method [30]. Thus SRA1 is the stability-optimized 2-stage explicit method which achieves the most robust error estimator.

$$\begin{aligned} A^{(0)} &= \begin{pmatrix} 0 & 0 \\ \frac{3}{4} & 0 \end{pmatrix}, \quad B^{(0)} = \begin{pmatrix} 0 & 0 \\ \frac{3}{2} & 0 \end{pmatrix}, \quad \alpha = \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \end{pmatrix} \\ \beta^{(1)} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \beta^{(2)} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad c^{(0)} = \begin{pmatrix} 0 \\ \frac{3}{4} \end{pmatrix}, \quad c^{(1)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{aligned} \tag{14}$$

3.1.2 Stability-Optimal 3-Stage Explicit SRA Methods

For the 3-stage SRA method, we receive the simplified stability function

$$G(z) = A_{21}A_{31}\alpha_3 z^3 + A_{21}\alpha_2 z^2 + A_{31}\alpha_3 z^2 + A_{32}\alpha_3 z^2 + \alpha_1 z + \alpha_2 z + \alpha_3 z + 1 \tag{15}$$

To optimize this method, we attempted to use the same techniques as before and optimize the real values of the negative roots. However, in this case we have a cubic polynomial and the root equations are more difficult. Instead, we turn to a more general technique to handle the stability optimization which will be employed in later sections as well. To do so, we generate an optimization problem which we can numerically solve for the coefficients. To simplify the problem, we let $z \in \mathbb{R}$ and define the function:

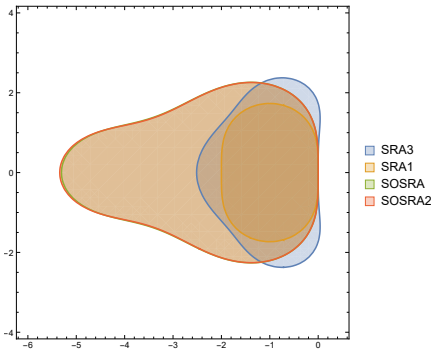

Figure 2: SOSRA Stability Regions. The stability regions ($|G(z)| < 1$) are plotted in the (x, y)-plane for $z = x + iy$. Shown are the SRA methods SRA1 and SRA3 in addition to the SOSRA and SOSRA2 methods. Note that SOSRA and SOSRA2 have nearly identical stability regions, making it hard to discern the two in the plot.

$$f(z, w; N, M) = \int_D \chi_{G(z)\leq 1}(z) dz \tag{16}$$

Notice that f is the area of the stability region when D is sufficiently large. Thus we define the stability-optimized SRK method for additive noise SDEs as the set of coefficients which achieves

$$\max_{A^{(i)}, B^{(i)}, \beta^{(i)}, \alpha} f(z) \tag{17}$$
$$\text{subject to: Order Constraints}$$

In all cases we impose $0 < c_i^{(0)}, c_i^{(1)} < 1$. We use the order constraints to simplify the problem to a nonlinear optimization problem on 14 variables with 3 equality constraints and 4 inequality constraints (with bound constraints on the 10 variables). However, we found that simplifying the problem even more to require $c_1^{(0)} = c_1^{(1)} = 0$ and $c_3^{(0)} = c_3^{(1)} = 1$ did not significantly impact the stability regions but helps the error estimator and thus we reduced the problem to 10 variables, 3 equality constraints, and 2 inequality constraints. This was optimized using the COBYLA local optimization algorithm [17, 26] with randomized initial conditions 100 times and all gave similar results. In the Mathematica notebook we show the effect of changing the numerical integration region D on the results, but conclude that a D which does not bias the result for better/worse real/complex handling does not improve the result. The resulting algorithm, SOSRA, we given by the coefficients in table in Section 9.1. Lastly, we used the condition that $c_2^{(0)} = c_3^{(0)} = c_2^{(1)} = c_3^{(1)} = 1$ to allow for free stability detection (discussed in Section 5.4). The method generated with this extra constraint is SOSRA2 whose coefficients are in the table in Section 9.2. These methods have their stability regions compared to SRA1 and SRA3 in Figure 2 where it is shown that the SOSRA methods more than doubles the allowed time steps when the eigenvalues of the Jacobian are dominated by the real part.

3.2 Drift Implicit Methods for Stiff SDEs with Additive Noise

3.2.1 An L-Stable 2-Stage (Semi-)Implicit SRA Method

It's clear that, as in the case for deterministic equations, the explicit methods cannot be made A-stable. However, the implicit two-stage additive noise SRK method is determined by

$$G(z) = \frac{z(A_{11}(A_{22}z - \alpha_2 z - 1) + A_{12}z(\alpha_1 - A_{21}) + A_{21}\alpha_2 z - A_{22}(\alpha_1 z + 1) + \alpha_1 + \alpha_2) + 1}{A_{11}z(A_{22}z - 1) - z(A_{12}A_{21}z + A_{22}) + 1} \tag{18}$$

which is A-stable if

$$A_{11}z(A_{22}z-1)-z(A_{12}A_{21}z+A_{22})+1 > z(A_{11}(A_{22}z-\alpha_2 z-1)+A_{12}z(\alpha_1-A_{21})+A_{21}\alpha_2 z-A_{22}(\alpha_1 z+1)+\alpha_1+\alpha_2)+1. \tag{19}$$

Notice that the numerator equals the denominator if and only if $z = 0$ or

$$z = \frac{\alpha_1 + \alpha_2}{(A_{22} - A_{12})\,\alpha_1 + (A_{11} - A_{21})\,\alpha_2}. \tag{20}$$

From the order conditions we know that $\alpha_1 + \alpha_2 = 1$ which means that no root exists with $Re(z) < 0$ if $(A_{22} - A_{12})\,\alpha_1 + (A_{11} - A_{21})\,\alpha_2 > 0$. Thus under these no roots conditions, we can determine A-stability by checking the inequality at $z = 1$, which gives $1 > (A_{22} - A_{12})\,\alpha_1 + (A_{11} - A_{21})\,\alpha_2$. Using the order condition, we have a total of four constraints on the $A^{(0)}$ and α:

$$\begin{aligned} (A_{11} + A_{12})\,\alpha_1 + (A_{21} + A_{22})\,\alpha_2 &= \frac{1}{2} \\ \alpha_1 + \alpha_2 &= 1 \\ 0 < (A_{22} - A_{12})\,\alpha_1 + (A_{11} - A_{21})\,\alpha_2 &< 1 \end{aligned} \tag{21}$$

However, A-stability is not sufficient for most ODE integrators to properly handle stiff equations and thus extra properties generally imposed [12]. One important property we wish to extend to stochastic integrators is L-stability. The straightforward extension of L-stability is the condition

$$\lim_{z\to\infty} G(z) = 0. \tag{22}$$

This implies that

$$\frac{-A_{11}A_{22} + A_{11}\alpha_2 + A_{12}A_{21} - A_{12}\alpha_1 - A_{21}\alpha_2 + A_{22}\alpha_2\alpha_1}{A_{12}A_{21} - A_{11}A_{22}} = 0 \tag{23}$$

The denominator is $-\det(A^{(0)})$ which implies $A^{(0)}$ must be non-singular. Next, we attempt to impose B-stability on the drift portion of the method. We use the condition due to Burrage and Butcher that for $B = \mathrm{diag}\,(\alpha_1, \alpha_2)$ $M = BA^{(0)} + A^{(0)}B - \alpha\alpha^T$ (for ODEs) [4], we require both B and M to be non-negative definite. However, in the supplemental Mathematica notebooks we show computationally that there is no 2-stage SRK method of this form which satisfies all three of these stability conditions. Thus we settle for A-stability and L-stability.

Recalling that $c^{(0)}$ and $c^{(1)}$ are the locations in time where f and g are approximated respectively, we

wish to impose

$$\begin{aligned} c_1^{(0)} &= 0 \\ c_2^{(0)} &= 1 \\ c_1^{(1)} &= 0 \\ c_2^{(1)} &= 1 \end{aligned} \tag{24}$$

so that the error estimator covers the entire interval of integration. Since $c^{(0)} = A^{(0)}e$, this leads to the condition $A_{21} + A_{22} = 1$. Using the constraint-satisfaction algorithm FindInstance in Mathematica, we look for tableaus which satisfy the previous conditions with the added constraint of semi-implicitness, i.e. $B^{(0)}$ is lower triangular. This assumption is added because the inverse of the normal distribution has unbounded moments, and thus in many cases it mathematically simpler to consider the diffusion term as explicit (though there are recent methods which drop this requirement via truncation or extra assumptions on the solution [24]). However, we find that there is no coefficient set which meets all of these requirements. However, if we relax the interval estimate condition to allow $0 \leq c_2^{(0)} \leq 1$, we find an A-L stable method:

$$\begin{aligned} A^{(0)} &= \begin{pmatrix} -1 & 1 \\ 0 & \frac{3}{4} \end{pmatrix}, \quad B^{(0)} = \begin{pmatrix} 0 & 0 \\ \frac{3}{2} & 0 \end{pmatrix}, \quad \alpha = \begin{pmatrix} \frac{1}{3} \\ \frac{2}{3} \end{pmatrix} \\ \beta^{(1)} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \beta^{(2)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad c^{(0)} = \begin{pmatrix} 0 \\ \frac{3}{4} \end{pmatrix}, \quad c^{(1)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned} \tag{25}$$

which we denote LSRA. If we attempt to look for a 2-stage SDIRK-like method to reduce the complexity of the implicit equation, i.e. $A_{12}^{(0)} = 0$, using FindInstance we find the constraints unsatisfiable. Note that if we drop the semi-implicit assumption we find that the full constraints cannot be satisfied there (we still cannot satisfy $c_1^{(0)} = 0$ and $c_2^{(0)} = 1$), and there does not exist a 2-stage A-L stable SDIRK method in that case.

3.2.2 Extensions of ODE Implicit Runge-Kutta Methods to Implicit SRA Methods

Since the stability region of the SRA methods is completely determined by the deterministic portion $A^{(0)}$, in some cases there may exist a sensible extension of implicit Runge-Kutta methods for ordinary differential equations to high order adaptive methods stochastic differential equations with additive noise which keep the same stability properties. Since the order constraints which only involve the deterministic portions $A^{(0)}$, $c^{(0)}$, and α match the conditions required for ODE integrators, existence is dependent on finding $\beta^{(1)}$, $\beta^{(2)}$, $c^{(1)}$, and $B^{(0)}$ that satisfy the full order constraints. In this case, an adaptive error estimator can be added by using the same estimator as the ODE method (which we call E_D) but adding the absolute size of the stochastic portions

$$E_N = \left| \sum_{i=1}^{s} \left(\beta_i^{(1)} I_{(1)} + \beta_i^{(2)} \frac{I_{(1,0)}}{h} \right) \right| \tag{26}$$

leading to the error estimator

$$E = \delta E_D + E_N. \tag{27}$$

This can be shown similarly to the construction in [28]. Given the large literature on implicit RK methods for ODEs, this presents a large pool of possibly good methods and heuristically one may believe that these would do very well in the case of small noise.

However, we note that there does not always exist such an extension. Using the constraint-satisfaction algorithm FindInstance in Mathematica, we looked for extensions of the explicit first stage singly-diagonally implicit RK (ESDIRK) method TRBDF2 [15] and could not find values satisfying the constraints. In addition, we could not find values for an extension of the 5th order Radau IIA method [13, 12] which satisfies the constraints. In fact, our computational search could not find any extension of a 3-stage L-stable implicit RK method which satisfies the constraints.

But, the 4-stage 3rd order ODE method due to Kennedy and Carpenter [18] can be extended to the following:

$$A^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{1767732205903}{4055673282236} & \frac{1767732205903}{4055673282236} & 0 & 0 \\ \frac{2746238789719}{10658868560708} & -\frac{640167445237}{6845629431997} & \frac{1767732205903}{4055673282236} & 0 \\ \frac{1471266399579}{7840856788654} & -\frac{4482444167858}{7529755066697} & \frac{11266239266428}{11593286722821} & \frac{1767732205903}{4055673282236} \end{pmatrix}, \tag{28}$$

$$\alpha = \begin{pmatrix} \frac{1471266399579}{7840856788654} \\ -\frac{4482444167858}{7529755066697} \\ \frac{11266239266428}{11593286722821} \\ \frac{1767732205903}{4055673282236} \end{pmatrix}$$

$$\beta^{(1)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \beta^{(2)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix}, \quad c^{(0)} = \begin{pmatrix} 0 \\ \frac{1767732205903}{4055673282236} \\ \frac{3}{5} \\ 1 \end{pmatrix}, \quad c^{(1)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$K = \frac{87294609440832483406992237\left(-5398340639937138772271239371353578627 6 - 26826820\sqrt{6853072660943221216270384658311461343029149665543510113394397}\right)}{486873851673469189145809 7}$$

$$B^{(0)}_{2,1} = \frac{K - 3540384151924107906194832136662001932}{2107581741132311678779814352587817066 48},$$

$$B^{(0)}_{4,3} = \frac{-5398340639937138772271239371353578627 6 - 26826820\sqrt{6853072660943221216270384658311461343029149665543510113394397}}{8606625878152317177894269252900546591},$$

$$B^{(0)}_{i,j} = 0 \; o.w.,$$

$$B^{(0)}_{2,1} \approx -12.246764387585055918338744103409192607986567514699471403397969732723452087723101$$

$$B^{(0)}_{4,3} \approx -14.432096958608752822047165680776748797565142459789556194474191884258734697161106$$

(E)SDIRK methods are particularly interesting because these methods can be solved using a single factorization of the function of the Jacobian $I - \gamma dtJ$ where J is the Jacobian. Additionally, explicit handling of the noise term is similar to the Implicit-Explicit (IMEX) form for additive Runge-Kutta methods in that it occurs by adding a single constant term to the Newton iterations in each stage, meaning it does not significantly increase the computational cost. The chosen ESDIRK method has a complimentary explicit tableau to form an IMEX additive Runge-Kutta method, and the chosen values for the stochastic portions are simultaneously compatible with the order conditions for this tableau. In Section 6.1 we numerically investigate the order of the IMEX extension of the method and show that it matches the convergence of the other SRA methods on the test equation. One thing to note is that since the problem is additive noise the method is never implicit in the dependent variables in the noise part, so in theory this can also be extended with $B^{(1)}$ implicit as well (with convergence concerns due to the non-finite inverse moments of the Normal distribution [19]).

4 Optimized-Stability Methods for Affine Noise via Transformation

Given the efficiency of the methods for additive noise, one method for developing efficient methods for more general noise processes is to use a transform of diagonal noise processes to additive noise. This transform is due to Lamperti [25], which states that the SDE of the form

$$dX_t = f(t, X_t)dt + \sigma(t, X_t)R(t)dW_t \tag{29}$$

where $\sigma > 0$ is a diagonal matrix with diagonal elements $\sigma_i(t, X_{i,t})$ has the transformation

$$Z_{i,t} = \psi_i(t, X_{i,t}) = \int \frac{1}{\sigma_i(x,t)} dx \mid_{x=X_{i,t}} \tag{30}$$

which will result in an Ito process with the ith element given by

$$dZ_{i,t} = \left(\frac{\partial}{\partial t}\psi_i(t,x) \mid_{x=\psi^{-1}(t,Z_{i,t})} + \frac{f_i(t, \psi^{-1}(t, Z_t))}{\sigma_i\left(t, \psi_i^{-1}(t, Z_{i,t})\right)} - \frac{1}{2}\frac{\partial}{\partial x}\sigma_i\left(t, \psi_i^{-1}(t, Z_{i,t})\right) \right) dt + \sum_{j=1}^{n} r_{ij}(t) dw_{j,t} \tag{31}$$

with

$$X_t = \psi^{-1}(t, Z_t). \tag{32}$$

This is easily verified using Ito's Lemma. In the case of mixed multiplicative and additive noise (affine noise), the vector equation:

$$dX_t = f(t, X_t)dt + (\sigma_M X_t + \sigma_A) dW_t \tag{33}$$

with $\sigma_M > 0$ and $\sigma_A > 0$, the transform becomes element-wise in the system. Thus we can consider the one-dimensional case. Since $\psi(t, X_t) = \int \left(\frac{1}{\sigma_M X_t + \sigma_A}\right) dx \mid_{x=X_t} = \frac{\log(\sigma_M X_t + \sigma_A)}{\sigma_M}$, then $X_t = \frac{\exp(\sigma_M Z_t) - \sigma_A}{\sigma_M}$ and

$$\begin{aligned} dZ_t &= \tilde{f}(t, X_t)dt + dW_t \\ \tilde{f}(t, X_t) &= \frac{f(t, X_t)}{\sigma_M X_t + \sigma_A} - \frac{1}{2}\sigma_M \end{aligned} \tag{34}$$

provided $\sigma_M X_t$ is guaranteed to be sufficiently different from σ_A to not cause definitional issues. It is common in biological models like chemical reaction networks that $X_t \geq 0$, in which case this is well-defined for any $\sigma_A > 0$ when $\sigma_M > 0$.

For numerical problem solving environments (PSEs), one can make use of this transformation in two ways. Source transformations could transform affine noise SDEs element-wise to solve for the vector Z_t which is the same as X_t if $\sigma_M \neq 0$ and is the transformed X_t otherwise (assuming parameters must be positive). When doing so, references of $X_{i,t}$ must be changed into $\frac{\exp(\sigma_M Z_{i,t}) - \sigma_A}{\sigma_M}$. For example, the affine noise Lotka-Volterra SDE:

$$\begin{aligned} dx &= (ax - bxy)\, dt + (\sigma_M x + \sigma_A)\, dW_t^1 \\ dy &= (-cy + dxy)\, dt + \sigma_{\tilde{A}} dW_t^2 \end{aligned}$$

only has noise on the first term, so this transforms to

$$\begin{aligned} x &= \frac{\exp(\sigma_M z) - \sigma_A}{\sigma_M} \\ dz &= \left(\frac{ax - bxy}{\sigma_M x + \sigma_A} - \frac{1}{2}\sigma_M\right) dt + dW_t^1 \\ dy &= (-cy + dxy)\, dt + \sigma_{\tilde{A}} dW_t^2 \end{aligned}$$

along with the change to the initial condition and can thus be solved with the SRA methods. We note a word of caution that the above transformation only holds when $\sigma_A > 0$ and when $\sigma_A = 0$, the transformation is different, with $X_t = \frac{\exp(Z_t)}{\sigma_M}$ (instead of $\frac{\exp(\sigma_M Z_t)}{\sigma_M}$ which one would get by taking $\sigma_A = 0$).

Instead of performing the transformations directly on the functions themselves, we can modify the SRA algorithm to handle this case as:

$$X_{n+1} = \psi^{-1}\left(\psi\left(X_n\right) + \sum_{i=1}^{s} \alpha_i \tilde{f}\left(t_n + c_i^{(0)} h, H_i^{(0)}\right) + \sum_{i=1}^{s} \left(\beta_i^{(1)} I_{(1)} + \beta_i^{(2)} \frac{I_{(1,0)}}{h}\right) \tilde{g}\left(t_n + c_i^{(1)} h\right)\right) \tag{35}$$

with stages

$$H_i^{(0)} = \psi\left(X_n\right) + \sum_{j=1}^{s} A_{ij}^{(0)} \tilde{f}\left(t_n + c_j^{(0)} h, H_j^{(0)}\right) h + \sum_{j=1}^{s} B_{ij}^{(0)} \tilde{g}\left(t_n + c_j^{(1)} h\right) \frac{I_{(1,0)}}{h} \tag{36}$$

where ψ is the element-wise function:

$$\psi_i(x) = \begin{cases} \frac{\log(\sigma_{i,M} x + \sigma_{i,A})}{\sigma_{i,M}} & \sigma_{i,M} > 0, \sigma_{i,A} > 0 \\ \frac{\log(x)}{\sigma_{i,M}} & \sigma_{i,M} > 0, \sigma_{i,A} = 0 \\ x & o.w. \end{cases}, \quad \psi_i^{-1}(z) = \begin{cases} \frac{\exp(\sigma_{i,M} z)}{\sigma_{i,M}} & \sigma_{i,M} > 0, \sigma_{i,A} > 0 \\ \frac{\exp(z)}{\sigma_{i,M}} & \sigma_{i,M} > 0, \sigma_{i,A} = 0 \\ x & o.w. \end{cases}$$

and

$$\tilde{g}_i(t) = \begin{cases} 1 & \sigma_{i,M} > 0 \\ \sigma_{i,A} & o.w. \end{cases}$$

This can be summarized as performing all internal operations in Z-space (where the equation is additive) but saving each step in X-space.

5 Optimized-Stability Order 1.5 SRK Methods with Diagonal Noise

5.1 The Stability Equation for Order 1.5 SRK Methods with Diagonal Noise

For diagonal noise, we use the mean-square definition of stability [19]. A method is mean-square stable if $\lim_{n\to\infty} \mathbb{E}\left(|X_n|^2\right) = 0$ on the test equation

$$dX_t = \mu X_t dt + \sigma X_t dW_t. \tag{37}$$

In matrix form we can re-write our method as given by

$$X_{n+1} = X_n + \mu h \left(\alpha \cdot H^{(0)}\right) + \sigma I_{(1)} \left(\beta^{(1)} \cdot H^{(1)}\right) + \sigma \frac{I_{(1,1)}}{\sqrt{h}} \left(\beta^{(2)} \cdot H^{(1)}\right) + \sigma \frac{I_{(1,0)}}{h} \left(\beta^{(3)} \cdot H^{(1)}\right) + \sigma \frac{I_{(1,1,1)}}{h} \left(\beta^{(4)} \cdot H^{(1)}\right) \tag{38}$$

with stages

$$\begin{aligned} H^{(0)} &= X_n + \mu \Delta t A^{(0)} H^{(0)} + \sigma \frac{I_{(1,0)}}{h} B^{(0)} H^{(1)}, \\ H^{(1)} &= X_n + \mu \Delta t A^{(1)} H^{(0)} + \sigma \sqrt{\Delta t} B^{(1)} H^{(1)} \end{aligned} \tag{39}$$

where $\hat{X_n}$ is the size s constant vector of X_n.

$$\begin{aligned} H^{(0)} &= \left(I - hA^{(0)}\right)^{-1} \left(\hat{X_n} + \sigma \frac{I_{(1,0)}}{h} B^{(0)} H^{(1)}\right), \\ H^{(1)} &= \left(I - \sigma\sqrt{h} B^{(1)}\right)^{-1} \left(\hat{X_n} + \mu h A^{(1)} H^{(0)}\right) \end{aligned} \tag{40}$$

By the derivation in the appendix, we receive the equation

$$\begin{aligned} S = E\left[\frac{U_{n+1}^2}{U_n^2}\right] &= \{1 + \mu h t \left(\alpha \cdot \left[\left(I - \mu \Delta t A^{(0)} - \mu \sigma I_{(1,0)} A^{(1)} B^{(0)} \left(I - \sigma\sqrt{h} B^{(1)}\right)^{-1}\right)^{-1} \left(I + \sigma \frac{I_{(1,0)}}{h} B^{(0)} \left(I - \sigma\sqrt{h} B^{(1)}\right)^{-1}\right)\right]\right) \\ &+ \sigma I_{(1)} \left(\beta^{(1)} \cdot \left[\left(I - \sigma\sqrt{h} B^{(1)} - \mu h A^{(1)} \left(I - \mu h A^{(0)}\right)^{-1} \sigma \frac{I_{(1,0)}}{h} B^{(0)}\right)^{-1} \left(I + \mu h A^{(1)} \left(I - \mu h A^{(0)}\right)^{-1}\right)\right]\right) \\ &+ \sigma \frac{I_{(1,1)}}{\sqrt{h}} \left(\beta^{(2)} \cdot \left[\left(I - \sigma\sqrt{h} B^{(1)} - \mu h A^{(1)} \left(I - \mu h A^{(0)}\right)^{-1} \sigma \frac{I_{(1,0)}}{h} B^{(0)}\right)^{-1} \left(I + \mu h A^{(1)} \left(I - \mu h A^{(0)}\right)^{-1}\right)\right]\right) \\ &+ \sigma \frac{I_{(1,0)}}{h} \left(\beta^{(3)} \cdot \left[\left(I - \sigma\sqrt{h} B^{(1)} - \mu h A^{(1)} \left(I - \mu h A^{(0)}\right)^{-1} \sigma \frac{I_{(1,0)}}{h} B^{(0)}\right)^{-1} \left(I + \mu h A^{(1)} \left(I - \mu h A^{(0)}\right)^{-1}\right)\right]\right) \\ &+ \sigma \frac{I_{(1,1,1)}}{h} \left(\beta^{(4)} \cdot \left[\left(I - \sigma\sqrt{h} B^{(1)} - \mu h A^{(1)} \left(I - \mu h A^{(0)}\right)^{-1} \sigma \frac{I_{(1,0)}}{h} B^{(0)}\right)^{-1} \left(I + \mu h A^{(1)} \left(I - \mu h A^{(0)}\right)^{-1}\right)\right]\right)\}^2 \end{aligned} \tag{41}$$

We apply the substitutions from the Appendix and let

$$\begin{aligned} z &= \mu h, \\ w &= \sigma\sqrt{h}. \end{aligned} \tag{42}$$

In this space, z is the stability variable for the drift term and w is the stability in the diffusion term. Under this scaling $\left(h, \sqrt{h}\right)$, the equation becomes independent of h and thus becomes a function $S(z, w)$ on the coefficients of the SRK method where mean-square stability is achieved when $|S(z, w)| < 1$. The equation $S(z, w)$ in terms of its coefficients for explicit methods ($A^{(i)}$ and $B^{(i)}$ lower diagonal) has millions of terms and is shown in the supplemental Mathematica notebook. Determination of the stability equation for the implicit methods was found to be computationally intractable and is an avenue for further research.

5.2 An Optimization Problem for Determination of Coefficients

We wish to determine the coefficients for the diagonal SRK methods which optimize the stability. To do so, we generate an optimization problem which we can numerically solve for the coefficients. To simplify the problem, we let $z, w \in \mathbb{R}$. Define the function

$$f(z,w;N,M) = \int_{-M}^{M}\int_{-N}^{1} \chi_{S(z,w)\leq 1}(z,w)dzdw. \tag{43}$$

Notice that for $N, M \to \infty$, f is the area of the stability region. Thus we define the stability-optimized diagonal SRK method as the set of coefficients which achieves

$$\max_{A^{(i)},B^{(i)},\beta^{(i)},\alpha} f(z,w) \tag{44}$$
$$\text{subject to: Order Constraints}$$

However, like with the SRK methods for additive noise, we impose a few extra constraints to add robustness to the error estimator. In all cases we impose $0 < c_i^{(0)}, c_i^{(1)} < 1$. Additionally we can prescribe $c_4^{(0)} = c_4^{(1)} = 1$ which we call the End-C Constraint. Lastly, we can prescribe the ordering constraint $c_1^{(j)} < c_2^{(j)} < c_3^{(j)} < c_4^{(j)}$ which we denote as the Inequality-C Constraint.

The resulting problem is a nonlinear programming problem with 44 variables and 42-48 constraint equations. The objective function is the two-dimensional integral of a discontinuous function which is determined by a polynomial of in z and w with approximately 3 million coefficients. To numerically approximate this function, we calculated the characteristic function on a grid with even spacing dx using a CUDA kernel and found numerical solutions to the optimization problem using the JuMP framework [9] with the NLopt backend [17]. A mixed approach using many solutions of the semi-local optimizer LN_AUGLAG_EQ [7, 3] and fewer solutions from the global optimizer GN_ISRES [31] were used to approximate the optimality of solutions. The optimization was run many times in parallel until many results produced methods with similar optimality, indicating that we likely obtained values near the true minimum.

The parameters N and M are the bounds on the stability region and also represent a trade-off between the stability in the drift and the stability in the diffusion. A method which is optimized when M is small would be highly stable in the case of small noise, but would not be guaranteed to have good stability properties in the presence of large noise. Thus these parameters are knobs for tuning the algorithms for specific situations, and thus we solved the problem for different combinations of N and M to determine different algorithms for the different cases.

5.3 Resulting Approximately-Optimal Methods

The coefficients generated for approximately-optimal methods fall into three categories. In one category we have the drift-dominated stability methods where large N and small M was optimized. On the other end we have the diffusion-dominated stability methods where large M and small N was optimized. Then we have the mixed stability methods which used some mixed size choices for N and M. As a baseline, we optimized the objective without constraints on the c_i to see what the "best possible method" would be. When this was done with large N and M, the resulting method, which we name SOSRI, has almost every value of c satisfy the constraints, but with $c_2^{(0)} \approx -0.04$ and $c_4^{(0)} \approx 3.75$. To see if we could produce methods which were more diffusion-stable, we decreased N to optimize more in w but failed to produce methods with substantially enlarged diffusion-stability over SOSRI.

Adding only the inequality constraints on the c_i and looking for methods for drift-dominated stability, we failed to produce methods whose c_i estimators adequately covered the interval. Some of the results did produce stability regions similar to SOSRI but with $c_i^{(0)} < 0.5$ which indicates the method could have problems with error estimation. When placing the equality constraints on the edge c_i, one method, which we

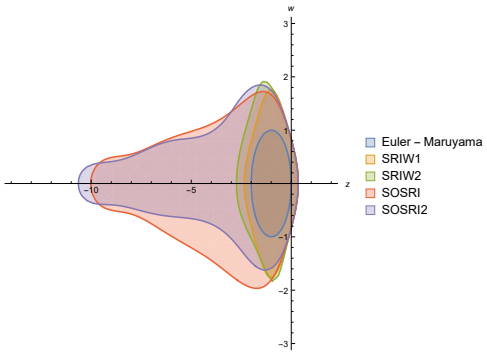

Figure 3: SOSRI Stability Regions. The stability regions for the SOSRI methods are plotted in the (z, w)-plane. In addition, the regions for Euler-Maruyama, SRIW1, and SRIW2 are shown for comparison.

label SOSRI2, resulted in similar stability to SOSRI but satisfy the c_i constraints. In addition, this method satisfies $c_3^{(0)} = c_4^{(0)} = 1$ and $c_3^{(1)} = c_4^{(1)} = 1$, a property whose use will be explained in Section 5.4. The stability regions for these methods is shown in Figure 3.

To look for more diffusion-stable methods, we dropped to $N = 6$ to encourage the methods to expand the stability in the w-plane. However, we could not find a method whose stability region went substantially beyond $[-2, 2]$ in w. This was further decreased to $N = 1$ where methods still could not go substantially beyond $|2|$. Thus we were not able to obtain methods optimized for the diffusion-dominated case. This hard barrier was hit under many different constraint and objective setups and under thousands of optimization runs, indicating there might be a diffusion-stability barrier for explicit methods.

5.4 Approximately-Optimal Methods with Stability Detection and Switching Behaviors

In many real-world cases, one may not be able to clearly identify a model as drift-stability bound or diffusion-stability bound, or if the equation is stiff or non-stiff. In fact, many models may switch between such extremes. An example is a model with stochastic switching between different steady states. In this case, we have that the diffusion term $f(t, X_{ss}) \approx 0$ in the area of many stochastic steady states, meaning that while straddling a steady state the integration is heavily diffusion-stability dominated and usually non-stiff. However, when switching between steady states, f can be very large and stiff, causing the integration to be heavily drift-stability dominated. Since these switches are random, the ability to adapt between these two behaviors could be key to achieving optimal performance. Given the trade-off, we investigated how our methods allow for switching between methods which optimize for the different situations.

The basis for our method is an extension of a method proposed for deterministic differential equations [34, 35, 12]. The idea is to create a cheap approximation to the dominant eigenvalues of the Jacobians for the drift and diffusion terms. If v is the eigenvector of the respective Jacobian, then for $\|v\|$ sufficiently small,

$$|\lambda_D| \approx \frac{\|f(t, x+v) - f(t, x)\|}{\|v\|}, \quad |\lambda_N| \approx \frac{\|g(t, x+v) - g(t, x)\|}{\|v\|} \tag{45}$$

where $|\lambda_D|$ and $|\lambda_N|$ are the estimates of the dominant eigenvalues for the deterministic and noise functions respectively. We have in approximation that $H_i^{(k)}$ is an approximation for $X_{t+c_i^{(k)}h}$ and thus the difference between two successive approximations at the same time-point, $c_i^{(k)} = c_j^{(k)}$, then the following serves as a local Jacobian estimate:

$$|\lambda_D| \approx \frac{\|f(t+c_i^{(0)}h, H_i^{(0)}) - f(t+c_j^{(0)}h, H_j^{(0)})\|}{\|H_i^{(0)} - H_j^{(0)}\|}, \quad |\lambda_N| \approx \frac{\|f(t+c_i^{(1)}h, H_i^{(1)}) - f(t+c_j^{(1)}h, H_j^{(1)})\|}{\|H_i^{(1)} - H_j^{(1)}\|} \tag{46}$$

If we had already computed a successful step, we would like to know if in the next calculation we should switch methods due to stability. Thus it makes sense to approximate the Jacobian at the end of the interval, meaning $i = s$ and $j = s-1$ where s is the number of stages. Then if z_{min} is the minimum $z \in \mathbb{R}$ such that z is in the stability region for the method, $\frac{h|\lambda_D|}{z_{min}} > 1$ when the steps are outside the stability region. Because the drift and mixed stability methods do not track the noise axis directly, we instead modify w_{min} to be $\frac{2}{3}$ of the maximum of the stability region in the noise axis.

Hairer noted that, for ODEs, if a RK method has $c_i = c_j = 1$, then it follows that

$$\rho = \frac{\|k_i - k_j\|}{\|g_i - g_j\|} \tag{47}$$

where $k_i = f(t + c_i h, g_i)$ is an estimate of the eigenvalues for the Jacobian of f. Given the construction of SOSRI2, a natural extension is

$$|\lambda_D| \approx \frac{\|f\left(t_n + c_4^{(0)}h, H_4^{(0)}\right) - f\left(t_n + c_3^{(0)}h, H_3^{(0)}\right)\|}{\|H_4^{(0)} - H_3^{(0)}\|}, \quad |\lambda_N| \approx \frac{\|g\left(t_n + c_4^{(0)}h, H_4^{(1)}\right) - g\left(t_n + c_3^{(0)}h, H_3^{(1)}\right)\|}{\|H_4^{(1)} - H_3^{(1)}\|} \tag{48}$$

Given that these values are all part of the actual step calculations, this stiffness estimate essentially is free. By comparing these values to the stability plot in Figure 2, we use the following heuristic to decide if SOSRI2 is stability-bound in its steps:

1. If $10 > |\lambda_D| > 2.5$, then we check if $h\,|\lambda_N| > \omega$.
2. If $|\lambda_D| < 2.5$, then we check if $h\,|\lambda_N|\,/2 > \omega$.

The denominator is chosen as a reasonable box approximation to the edge of the stability region. ω is a safety factor: in theory ω is 1 since we divided by the edge of the stability region, but in practice this is only an eigenvalue estimate and thus ω allows for a trade-off between the false positive and false negative rates. If either of those conditions are satisfied, then h is constrained by the stability region. The solver can thus alert the user that the problem is stiff or use this estimate to switch to a method more suitable for stiff equations. In addition, the error estimator gives separate error estimates in the drift and diffusion terms. A scheme could combine these two facts to develop a more robust stiffness detection method, and label the stiffness as either drift or diffusion dominated.

We end by noting that SOSRA2 has the same property, allowing stiffness detection via

$$|\lambda_D| \approx \frac{\|f\left(t_n + c_3^{(0)}h, H_3^{(0)}\right) - f\left(t_n + c_2^{(0)}h, H_2^{(0)}\right)\|}{\|H_3^{(0)} - H_2^{(0)}\|} \tag{49}$$

and, employing a similar method as the deterministic case, check for stiffness via the estimate $h\left|\lambda_D\right|/5 > \omega$.

In addition, stiff solvers can measure the maximal eigenvalues directly from the Jacobian. Here we suggest the measure from Shampine [34, 35, 12] of using $\|J\|_\infty$ as a cheap upper bound. For semi-implicit methods like LSRA we only get a stability bound on the drift term, but this should be sufficient since for additive noise diffusive noise instability is not an issue.

6 Numerical Results

6.1 Convergence Tests

In order to test the efficiency and correctness of the SRA algorithms, we chose to use the additive noise test equation

$$dX_t = \left(\frac{\beta}{\sqrt{1+t}} - \frac{1}{2\left(1+t\right)}X_t\right)dt + \frac{\alpha\beta}{\sqrt{1+t}}dW_t,\ \ X_0 = \frac{1}{2}, \tag{50}$$

where $\alpha = \frac{1}{10}$ and $\beta = \frac{1}{20}$ with true solution

$$X_t = \frac{1}{\sqrt{1+t}}X_0 + \frac{\beta}{\sqrt{1+t}}\left(t + \alpha W_t\right). \tag{51}$$

Figure 4A demonstrates that the SOSRA and SKenCarp methods achieve the strong order 2.0 on Equation 50. To test the convergence of the SRI algorithms, we used the linear test equation

$$dX_t = \alpha X_t dt + \beta X_t dW_t\ \ X_0 = \frac{1}{2}, \tag{52}$$

where $\alpha = \frac{1}{10}$ and $\beta = \frac{1}{20}$ with true solution

$$X_t = X_0 e^{\left(\beta - \frac{\alpha^2}{2}\right)t + \alpha W_t}. \tag{53}$$

Figure 4B demonstrates that the SOSRI methods achieve the strong order 1.5 on Equation 52. Lastly, we tested the convergence of the IMEX version of the SKenCarp integrator. We defined the split SDE

$$dX_t = \left(f_1(t, X_t) + f_2(t, X_t)\right)dt + \frac{\alpha\beta}{\sqrt{1+t}}dW_t,\ \ X_0 = \frac{1}{2}, \tag{54}$$

where

$$f_1(t, X_t) = \frac{\beta}{\sqrt{1+t}}$$
$$f_2(t, X_t) = -\frac{1}{2\left(1+t\right)}X_t$$

with true solution Equation 51. Figure 4C demonstrates that the IMEX SKenCarp method achieves strong order 2.0 on Equation 50 when the f_1 part is solved implicitly and the f_2 part is solved explicitly. Note that this does not demonstrate that the method always achieves strong order 1.5 since sufficient conditions for the IMEX pairing are unknown, but it gives numerical evidence that the method can be high order.

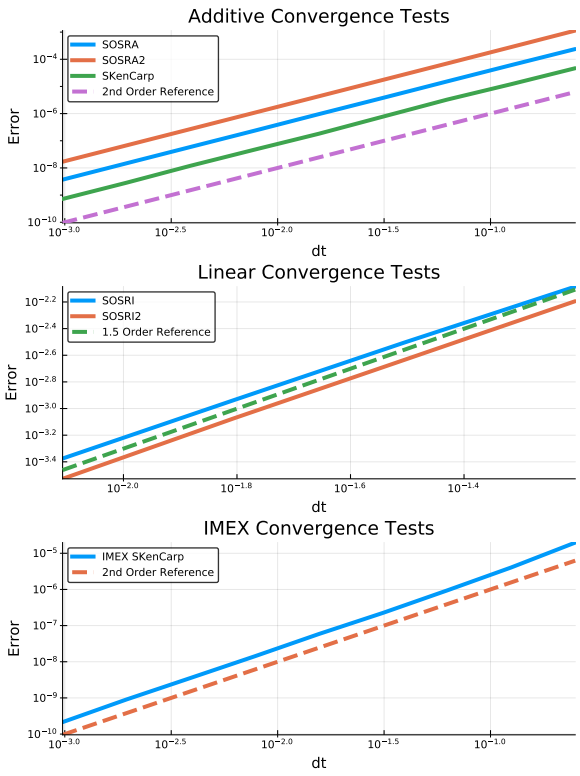

Figure 4: Convergence tests. The error is averaged over 1000 trajectories. Shown are the strong l_2 error along the time series of the solution. **(A)** Convergence results on Equatino 50. The test used a fixed time step $h = 1/2^{-2}$ to $h = 1/2^{-10}$. **(B)** Convergence results on Equation 52. The test used a fixed time step $h = 1/2^{-4}$ to $h = 1/2^{-7}$. **(C)** Convergence results on the IMEX Equation 54. The test used a fixed time step $h = 1/2^{-2}$ to $h = 1/2^{-10}$.

6.2 SOSRA Numerical Efficiency Experiments

To test the efficiency we first plotted work-precision [11, 33, 12] diagrams for the SOSRA, SOSRA2, and SKenCarp methods against the SRA1, SRA2, SRA3 [30] methods, and fixed time step Euler-Maruyama method (Milstein is equivalent to Euler-Maruyama in this case [19]). We tested the error and timing on Equation 50. In addition, we tested using the Lotka-Volterra equation with additive noise:

$$\begin{aligned} dx &= (ax - bxy)\,dt + \sigma_A dW_t^1 \\ dy &= (-cy + dxy)\,dt + \sigma_A dW_t^2 \end{aligned} \tag{55}$$

where $a = 1.5$, $b = 1$, $c = 3.0$, $d = 1.0$, $\sigma_A = 0.01$. Since 55 does not have an analytical solution, a reference solution was computed using a low tolerance solution via SOSRA for each Brownian trajectory. The plots show that there is a minimal difference in efficiency between the SRA algorithms for errors in the interval $\left[10^{-6}, 10^{-2}\right]$, while these algorithms are all significantly more efficient than the Euler-Maruyama method when the required error is $< 10^{-4}$ (Figure 5). The weak error work-precision diagrams show that when using between 100 to 10,000 trajectories, the weak error is less than the sample error in the regime where there is no discernible efficiency difference between the SRA methods. These results show that in the regime of mild accuracy on non-stiff equations, the SOSRA, SOSRA2, and SKenCarp methods are much more efficient than low order methods yet achieve the same efficiency as the non-stability optimized SRA variants. Note that these results also show that the error estimator for adaptivity is highly conservative, generating solutions with around 2 orders of magnitude less error than the tolerance suggests.

To test how efficiently the algorithms could achieve solve semi-stiff equations, we chose to analyze the qualitative results of the driven Van der Pol equation. The driven Van der Pol equation is given by the system:

$$\begin{aligned} dy &= \mu((1 - x^2)y - x)dt \\ dx &= ydt \end{aligned} \tag{56}$$

where μ is the driving factor. As μ increases the equation becomes more stiff. $\mu = 10^6$ is a common test for stiff ODE solvers [13], with lower values used to test the semi-stiff regime for ODEs. For our purposes, we chose $\mu = 10^5$ as a semi-stiff test case. The ODE case, solved using the Tsit5 explicit Runge-Kutta algorithm [37, 29], and demonstrates the mild stiffness which is still well-handled by explicit methods (Figure 6A). We extend this model to the driven Van der Pol model with additive noise:

$$\begin{aligned} dy &= \mu((1 - x^2)y - x)dt + \rho dW_t^{(1)} \\ dx &= y + \rho dW_t^{(2)} \end{aligned} \tag{57}$$

where $\rho = 3.0$ is the noise gain and $dW^{(1)}$ and $dW^{(2)}$ are independent Brownian motions. The solution to this model is interesting because it gives the same qualitative behavior, large bursts when $x(t)$ crosses zero, but in this case the zero crossings are stochastic. Even at high tolerances, ($abstol = 10$,$reltol = 1/2^1$), SOSRA is able to reproduce this qualitative behavior (Figure 6B), and SOSRA2 producing similar results at the same tolerances a factor of two lower. Given the conservativeness of the error estimators shown in previous (and other tests), this case corresponds to roughly two decimal places of accuracy, which is more than sufficient for many phenomenological models. However, even at tolerances of $abstol = 1/2^3$,$reltol = 1/2^3$ SRA3 was unable to reproduce the correct qualitative behavior (Figure 6C). Thus we decreased the tolerances by factors

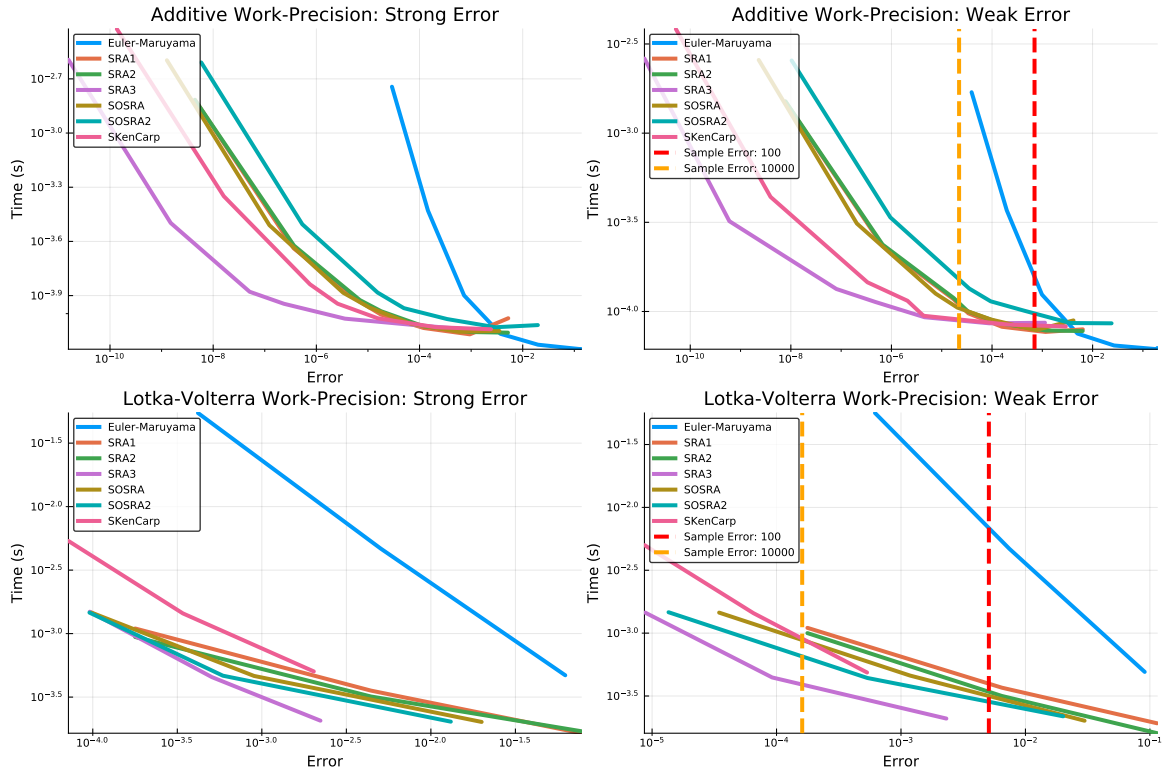

Figure 5: SOSRA efficiency tests. The error was taken as the average of 10,000 trajectories for Equation 50 and 100 trajectories for the Lokta-Volterra Equation 55. The sample error was determined for the weak error as the normal 95% confidence interval for the mean using the variance of the true solution Equation 51 or the variance of the estimated true solutions via low tolerance solutions. The time is the average time to compute a trajectory and is averaged over 1000 runs at the same tolerance or step size. **(A)** Shown are the work-precision plots for the methods on Equation 50. Each of the adaptive time-stepping methods solved the problem on the interval using changing values of tolerances, with $tol = abstol = reltol$ starting at 10^2 and ending at 10^{-4} going in increments of 10. The fixed time-stepping methods used time steps of size $h = 1/5^{-1}$ to $h = 1/5^4$, changing the value by factors of 5. The error is the strong l_2 error computed over the time series. **(B)** Same setup as the previous plot but using the weak error at the final time-point. **(C)** Shown are the work-precision plots for the methods on the Equation 55. Each of the adaptive time-stepping methods solved the problem on the interval using changing values of tolerances, with $tol = abstol = reltol$ starting at 4^{-2} and ending at 4^{-4} going in increments of 4. The fixed time-stepping methods used time steps of size $h = 1/12^{-2.5}$ to $h = 1/12^{-6.5}$, changing the value by factors of 12. The error is the strong l_2 error computed over the time series. **(D)** Same setup as the previous plot but using the weak error at the final time-point.

Algorithm	Run-time (seconds)	Relative Time (vs SKenCarp)
SKenCarp	116.238533	1x
SOSRA	220.318093	1.895x
SOSRA2	229.158667	1.971x
SRA3	2222.266499	19.12x
SRA1	7381.638625	63.50x

Table 1: SRA Run times on Van der Pol with additive noise. The additive noise Van der Pol equation was solved 100 times using the respective algorithms.

of 2 until it was able to reproduce the correct qualitative results (Figure 6D). This shows that the SOSRA are more reliable on models with transient stiffness. To test the impact on the run time of the algorithms, each of the algorithms were run 100 times with the tolerance setup that allows them to most efficiently generate correct qualitative results. The run times are shown in Table 1, which show that SRA1 takes more than 60 times and SRA3 nearly 20 times as long as the SOSRA methods. In this case the implicit method SKenCarp is the fastest by besting the SOSRA methods by 2x while achieving similar qualitative results. This shows that as stiffness comes into play, the SOSRA methods along with the implicit SKenCarp method are more robust and efficient. We note that we attempted to solve the equations using a fixed time step implicit method as well (implicit Euler-Maruyama), though during the zero-crossing events the quasi-Newton steps would diverge even when $dt = 10^{-6}$ (using an efficient and numerically robust implementation found in common stiff methods for ODEs [12]).

In addition to testing efficiency, we used this to test the stiffness detection in SOSRA2. Using a safety factor of $\omega = 5$, we added only two lines of code to make the algorithm print out the timings for which the algorithm predicts stiffness. The results on two trajectories were computed and are shown in Figure 7. The authors note that the stiffness detection algorithms are surprisingly robust without any tweaking being done and are shown to not give almost any false positives nor false negatives on this test problem. While this safety factor is set somewhat high in comparison to traditional ODE stiffness detection, we note that these algorithms were designed to efficiently handle mild stiffness and thus we see it as a benefit that they only declare stiffness when it appears to be in the regime which is more suitable for implicit methods.

6.3 SOSRI Numerical Efficiency Experiments

To test the efficiency we plotted a work-precision diagram with SRIW1, SOSRI, SOSRI2, and the fixed time step Euler-Maruyama and a Runge-Kutta Milstein schemes for Equation 52 and the multiplicative noise Lotka-Volterra equation

$$\begin{aligned} dx &= (ax - bxy)\,dt + \sigma_A dW_t^1 \\ dy &= (-cy + dxy)\,dt + \sigma_A dW_t^2 \end{aligned} \tag{58}$$

where $a = 1.5$, $b = 1$, $c = 3.0$, $d = 1.0$, $\sigma_A = 0.01$. As with Equation 55, Equation 58 does not have an analytical solution so a reference solution was computed using a low tolerance solution via SOSRI for

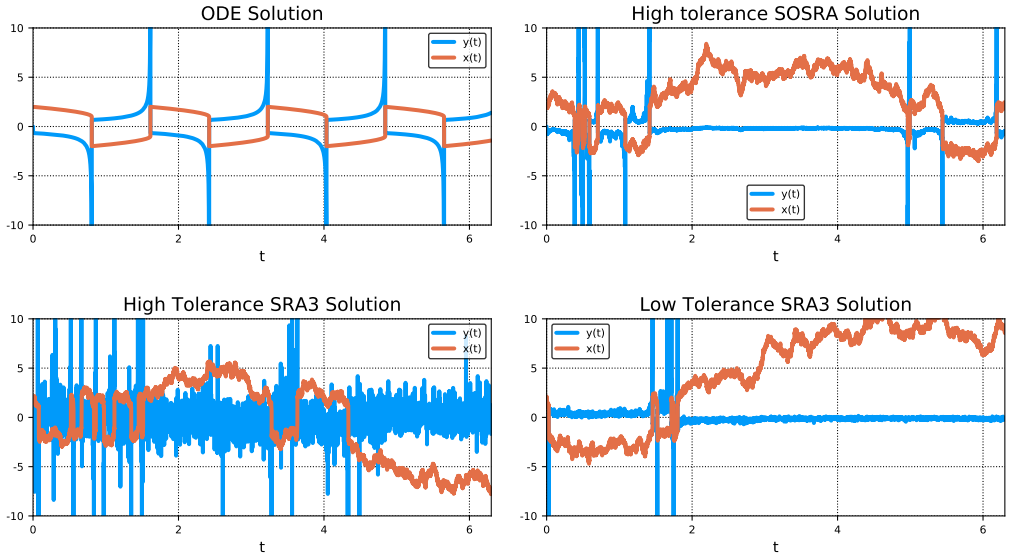

Figure 6: Representative trajectories for solutions to the Van der Pol equations with additive noise. **(A)** The solution to the ODE with the explicit Runge-Kutta method Tsit5. **(B)** The solution to the SDE with tolerance $abstol = 1, reltol = 1/2^1$ from SOSRA. **(C)** Solution to the SDE with tolerances $abstol = 2^{-3}, reltol = 2^{-3}$ with SRA3. **(D)** Solution to the SDE with tolerances $abstol = 2^{-6}, reltol = 2^{-4}$ with SRA3.

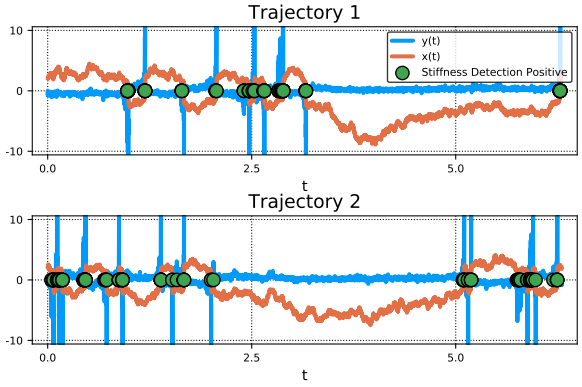

Figure 7: Stiffness detection in the Van der Pol equations with additive noise Equation 57. Two representative trajectories to the Van der Pol equations are plotted. The green dots indicate time-points where the stiffness detection algorithm detected stiffness.

Algorithm	Abstol	Reltol	Run-time (seconds)	Relative Time (vs SOSRI)
SOSRI	2^{-7}	2^{-4}	2.624539158	1x
SOSRI	2^{-7}	2^{-6}	2.750385673	1.048x
SOSRI	2^{-12}	2^{-15}	8.785915946	3.348x
SOSRI	2^{-13}	2^{-7}	3.052668315	1.163x
SOSRI2	2^{-12}	2^{-15}	8.694172692	3.313x
SOSRI2	2^{-13}	2^{-11}	5.564607661	2.210x
SRIW1	2^{-13}	2^{-7}	15.168839222	5.780x
Euler-Maruyama			169.96851466	64.76x
Runge-Kutta Milstein			182.597809791	69.57x
Fixed Time-step SRIW1			424.301395463	161.7x

Table 2: SRI times for the the EMT model on $t \in [0, 1]$. The equations were solve 10,000 times with the given tolerances to completion and the elapsed time was recorded. The fixed time step methods had their Δt determined as the largest Δt in increments of powers of 2 that produced no unstable trajectories, as shown in [28].

each Brownian trajectory. The results show that there is a minimal difference in efficiency between the SRI algorithms for errors over the interval $\left[10^{-6}, 10^{-2}\right]$, while these algorithms are all significantly more efficient than the lower order algorithms when the required error is $< 10^{-2}$ (Figure 8A-D). The weak error work-precision diagrams show that when using between 100 to 10,000 trajectories, the weak error is less than the sample error in the regime where there is no discernible efficiency difference between the SRI methods.These results show that in the regime of mild accuracy on non-stiff equations, these methods are much more efficient than low order methods yet achieve the same efficiency as the non-stability optimized SRI variants. Note that these results also show the conservativeness of the error estimators.

To test the real consequences of the enhanced stability, we use the Epithelial-Mesenchymal Transition (EMT) model of 19 pathwise stiff reaction equations introduced in [14], studied as a numerical test in [28], and written in Section 8.1. In the previous work it was noted that $t \in [0, 1]$ was a less stiff version of this model. Thus we first tested the speed that the methods could solve for 10,000 trajectories with no failures due to numerical instabilities. The tolerances were tuned for each method by factors of 2 and finding the largest values that were stable. Since SOSRI demonstrated that its stability is much higher than even SOSRI2, we show the effect of tolerance changes on SOSRI as well. The results show that at similar tolerances the SOSRI method takes nearly the same amount of time as SRIW1 (Table 2). However, there is an upper bound on the tolerances before the adaptivity is no longer able to help keep the method stable. For SRIW1, this bound is much lower, causing it to run more than 15x slower than the fastest SOSRI setup. Interestingly SOSRI2 required a higher tolerance than SRIW1 but was 3x faster than SRIW1's fastest setup. We note that SOSRI's highest relative tolerance $2^{-7} \approx 7 \times 10^{-3}$ is essentially requiring 4 digits of accuracy (in strong error) when considering the conservativeness of the error estimator, which is far beyond the accuracy necessary in many cases. Lastly, we note that the SOSRI method is able to solve for 10,000 stable trajectories more than 60x faster than any of the tested fixed time step methods.

We then timed the run time to solve 10 trajectories in the $t \in [0, 500]$ case (Table 3). This time we found the optimal tolerance in terms of powers of 10. Once again, SRIW1 needed a lower tolerance than

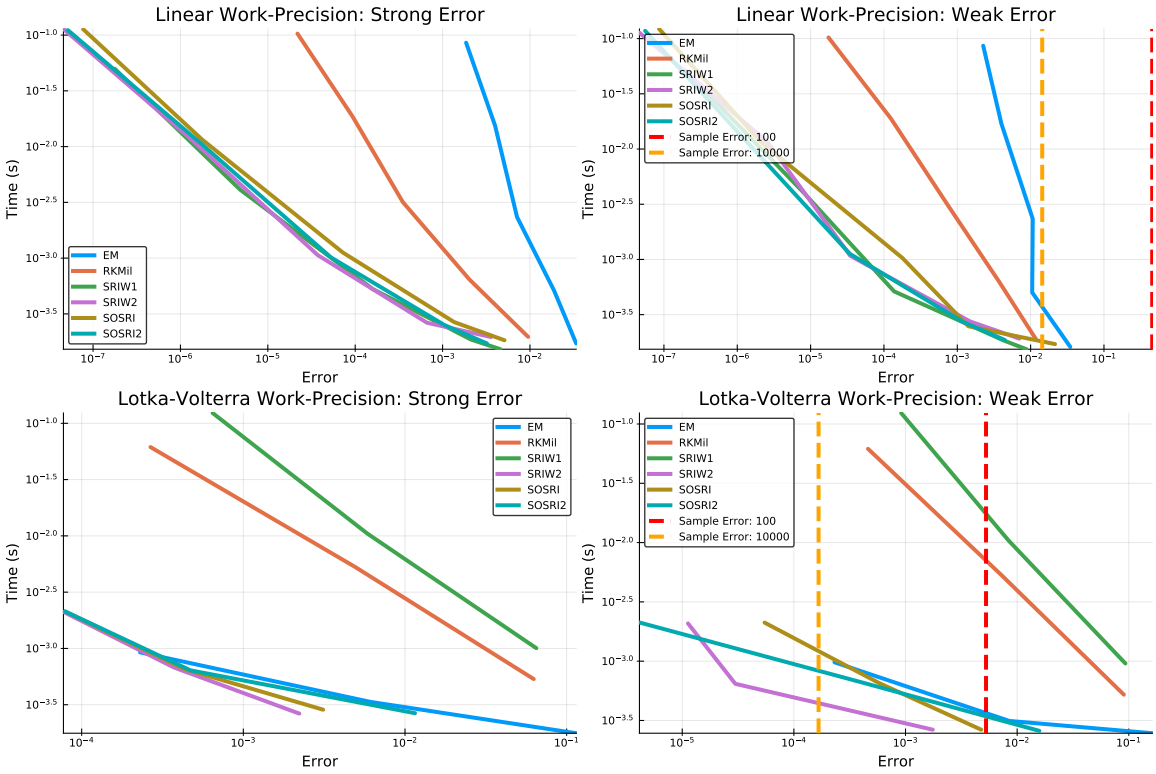

Figure 8: SOSRI efficiency on non-stiff test equations. The error was taken as the average of 10,000 trajectories for the Equation 52 and 100 trajectories for the Lokta-Volterra Equation 58. The sample error was determined for the weak error as the normal 95% confidence interval for the mean using the variance of the true solution Equation 53 or the variance of the estimated true solutions via low tolerance solutions. The time is the average time to compute a trajectory and is averaged over 1000 runs at the same tolerance or step size. **(A)** Shown are the work-precision plots for the methods on Equation 52. Each of the adaptive time-stepping methods solved the problem on the interval using changing values of tolerances, with $tol = abstol = reltol$ starting at 10^{-1} and ending at 10^{-5} going in increments of 10. The fixed time-stepping methods used time steps of size $h = 5^{-2}$ to $h = 5^{-7}$, changing the value by factors of 5. The error is the strong l_2 error computed over the time series. **(B)** Same setup as the previous plot but using the weak error at the final time-point. **(C)** Shown are the work-precision plots for the methods on the multiplicative noise Lotka-Volterra Equation 58. Each of the adaptive time-stepping methods solved the problem on the interval using changing values of tolerances, with $tol = abstol = reltol$ starting at 4^{-2} and ending at 4^{-4} going in increments of 4. The fixed time-stepping methods used time steps of size $h = 1/12^{-2.5}$ to $h = 1/12^{-6.5}$, changing the value by factors of 12. The error is the strong l_2 error computed over the time series. **(D)** Same setup as the previous plot but using the weak error at the final time-point.

Algorithm	Abstol	Reltol	Run-time (seconds)	Relative Time (vs SOSRI)
SOSRI	10^{-2}	10^{-2}	22.472149	1x
SOSRI	10^{-4}	10^{-4}	73.624740	3.276x
SOSRI	10^{-5}	10^{-3}	89.198343	3.969x
SOSRI2	10^{-4}	10^{-4}	76.128561	3.388x
SOSRI2	10^{-5}	10^{-3}	121.756225	5.418x
SRIW1	10^{-5}	10^{-3}	147.898779	6.581x
Drift-Implicit Euler-Maruyama			8796.472603	391.4x
Drift-Implicit Runge-Kutta Milstein			7378.555228	328.3x

Table 3: SRI times for the the EMT model on $t \in [0, 500]$. The equations were solved 10 times with the given tolerances to completion and the elapsed time was recorded. The fixed timestep methods had their Δt chosen by incrementing by 10^{-5} until 10 consecutive trajectories were stable. Drift-Implicit Euler Maruyama had $\Delta t = \frac{1}{60000}$ and Drift-Implicit Runge-Kutta Milstein had $\Delta t = \frac{1}{50000}$.

is necessary in order to stay stable. SOSRI is able to solve the problem only asking for around 2 digits of accuracy, while the others require more (especially in absolute tolerance as there is a stiff reactant whose values travel close to zero). One interesting point to note is that at similar tolerances both SOSRI and SOSRI2 receive similar timings and both over 6 times faster than the fastest SRIW1 tolerance setup. Both are nearly twice as fast as SRIW1 when matching tolerances as well. Given the conservativeness of the error estimators generally being around 2 orders of magnitude more precise than the local error estimate, the low tolerance solutions are accurate enough for many phenomenological experiments and thus present a good speedup over previous methods. The timings for Euler-Maruyama and Runge-Kutta Milstein schemes are omitted since the tests were unable to finish. From the results of [28] we note that the average dt for SRIW1 on the edge of its stability had that the smallest dt was approximately 10^{-11}. The stability region for fixed step-size Euler-Maruyama is strictly smaller than SRIW1 (Figure 3) which suggests that it would require around 5×10^{12} time steps (with Runge-Kutta Milstein being similar) to solve to $t = 500$. Thus, given it takes on our setup extrapolating the time given 170 seconds for 2^{20} steps, this projects to around 1.6×10^8 seconds, or approximately 5 years.

As another test we applied the methods to a method of lines discretization of a stochastic partial differential equation (SPDE) describing the spatial regulation of the zebrafish hindbrain via retinoic acid signaling [27]. The model is described in Section 8.2. The discretization results in a system of $6 \times 20 \times 100$ SDEs. The model was solved nearly to steady state on the PDE described by the drift term. Starting from the ODE steady state, noise was added to the gradient by solving the extension to an SPDE with multiplicative noise on $t \in [0, 50]$. Each of the methods solved the problem at the highest tolerance possible and the timings (Table 4). Since this problem starts from the only positive steady state, it is only semi-stiff. Yet we receive similar results as the other problems with the SOSRI methods able to compute steady state stochastic gradients nearly 6x faster than SRIW1, signaling that the expanded stability domain can lead to speedups against other high order SRK methods even to equations which only exhibit slight amounts of stiffness. A stable Δt for the Euler-Maruyama and Runge-Kutta Milstein methods could not be found (must be $< 10^{-6}$ and too long to a true estimate).

Algorithm	Abstol	Reltol	Run-time (seconds)	Relative Time (vs SOSRI)
SOSRI	10^{-1}	10^{-1}	1845.5754615	1x
SOSRI2	10^{-1}	10^{-1}	2420.1778155	1.311x
SRIW1	10^{-5}	10^{-3}	11174.834712	6.055x

Table 4: SRI times for the the retinoic acid SPDE model on $t \in [0, 50]$. The equations were solved twice with the given tolerances to completion and the elapsed time was recorded. The tolerances were chosen as the highest pair of tolerances which did not diverge (going up by powers of 10). Note that none of the cases did the two timings vary by more than 1% of the total run time.

7 Discussion

In this work we derived stability-optimized SRK methods for additive and diagonal noise equations, and used a transformation to allow the additive noise methods to solve affine noise problems. Many other equations can be reduced to the additive noise case as well using the same means. Importantly, our derivation methods utilized heavy computational tools in order to approximately optimize otherwise intractable equations. This same method of derivation can easily be scaled up to higher orders, and by incorporating the coefficients for higher conditions, efficiency can be optimized as well by adding the norm of the principle error coefficients to the optimization function. We note that the majority of the search was performed using global optimizers in massive parallel using a hand-optimized CUDA kernel for the numerical integral of the characteristic function, replacing man-hours with core-hours and effectively optimizing the method. The clear next steps are to find SRA and SRI methods with minimal error estimates and sensible stability regions for the cases in which lower strong error matters, and similar optimizations on SRK methods developed for small noise problems. We note that high strong order methods were investigated because of their better trajectory-wise convergence, allowing for a more robust solution and error estimation since our application to transiently pathwise stiff equations requires such properties.

In this work we also derived L-stable methods for additive (and thus multiplicative and affine) noise equations, and computationally could not find an A-B-L stable method. While our method does not prove that no 2-stage A-B-L method exists, we have at least drastically narrowed down its possibility. Additionally an extension of a well-known ESDIRK method to additive noise was developed. This method has an implicit-explicit (IMEX) extension and the stochastic extension was compatible with both tableaus. We showed that this IMEX version of the method could numerically converge at order 2.0 on a test problem (matching the other SRA methods), indicating that it may achieve the sufficient condition. As an adaptive high order IMEX method, the ODE version of the method is a common choice for large discretizations of PDEs. Thus this method presents itself as a potentially automatic and efficient option for discretizations of large affine noise SPDEs by being able to use a low number of time steps while minimizing the amount of work required to solve the implicit equation. More work could compare this method to other implicit solvers for additive noise SDEs to determine the problem domain which is is efficient.

Our timings show that the current high order SRK methods are stability-bound and that when scientific studies are only looking for small amounts of accuracy in stochastic simulations, most of the computational effort is lost to generating more accurate than necessary solutions in order to satisfy stability constraints. For additive noise problems we were able to obtain solutions about 20x-60x faster and for diagonal noise approximately 6x than the current adaptive methods (SRA1, SRA3, SRIW1), while common methods like Euler-Maruyama and Drift-Implicit Runge-Kutta Milstein were 100x slower or in many cases could not even finish. We have also shown that these methods are very robust even at high tolerances and have a tendency to produce the correct qualitative results on semi-stiff equations (via plots) even when the user chosen accuracy

is low. Given that the required user input is minimal, we see these as very strong candidates for default general purpose solvers for problem-solving environments such as MATLAB and Julia since they can easily and efficiently produce results which are sufficiently correct. We note that the SOSRA and SOSRA2 methods are not as efficient at low tolerances as SRA3, so SRA3 should be used when high accuracy is necessary (on additive or affine noise problems). However, in many cases like integrating to find steady distributions of bistable parameter regimes or generating trajectories of phonomenological models, this ability to quickly get a more course estimate is valuable.

We note that the stiffness detection in SDEs is a novel addition which we have demonstrated can act very robustly. It has a control parameter ω which can be used to control the false positive and false negative rate as needed. Note that stiff methods can achieve similar largest eigenvalue estimates directly from the Jacobians of f (and g) given that the methods are implicit (or in the case of Rosenbrock methods, the Jacobian must still be computed), and thus this can be paired with a stiff solver to allow for automatic switching between stiff and non-stiff solvers. Given that the cost for such stiffness checks is minimal, we are interested in future studies on the efficiency of such methods due to the stochastic nature of stiffness in SDEs.

8 Appendix I: Extended Test Equations

8.1 Epithelial-Mesenchymal Transition Model

The Epithelial-Mesenchymal Transition (EMT) model is given by the following system of SDEs which correspond to a chemical reaction network modeled via mass-action kinetics with Hill functions for the feedbacks. This model was introduced in [14].

$$
\begin{aligned}
A &= \left(\left[TGF\right] + \left[TGF0\right](t)\right)/J0_{snail}\right)^{n0_{snail}} + \left(\left[OVOL2\right]/J1_{snail}\right)^{n1_{snail}} \\
\frac{d\left[snail1\right]_t}{dt} &= k0_{snail} + k_{snail}\frac{\left(\left(\left[TGF\right] + \left[TGF0\right](t)\right)/J0_{snail}\right)^{n0_{snail}}}{\left(1+A\right)\left(1+\left[SNAIL\right]/J2_{snail}\right)} \\
&- kd_{snail}\left(\left[snail1\right] - \left[SR\right]\right) - kd_{SR}\left[SR\right] \\
\frac{d\left[SNAIL\right]}{dt} &= k_{SNAIL}\left(\left[snail1\right] - \left[SR\right]\right) - kd_{SNAIL}\left[SNAIL\right] \\
\frac{d\left[miR34\right]}{dt} &= kO_{34} + \frac{k_{34}}{1 + \left(\left[SNAIL\right]/J1_{34}\right)^{n1_{34}} + \left(\left[ZEB\right]/J2_{34}\right)^{n2_{34}}} \\
&- kd_{34}\left(\left[miR34\right] - \left[SR\right]\right) - \left(1-\lambda_{SR}\right)kd_{SR}\left[SR\right] \\
\frac{d\left[SR\right]}{dt} &= Tk\left(K_{SR}\left(\left[snail1\right] - \left[SR\right]\right)\left(\left[miR34\right] - \left[SR\right]\right) - \left[SR\right]\right) \\
\frac{d\left[zeb\right]}{dt} &= k0_{zeb} + k_{zeb}\frac{\left(\left[SNAIL\right]/J1_{zeb}\right)^{n1_{zeb}}}{1 + \left(\left[SNAIL\right]/J1_{zeb}\right)^{n1_{zeb}} + \left(\left[OVOL2\right]/J2_{zeb}\right)^{n2_{zeb}}} \\
&- kd_{zeb}\left(\left[zeb\right] - \sum_{i=1}^{5} C_5^i\left[ZR\right]\right) - \sum_{i=1}^{5} kd_{ZR_i}C_5^i\left[ZR_i\right] \\
\frac{d\left[ZEB\right]}{dt} &= k_{ZEB}\left(\left[zeb\right] - \sum_{i=1}^{5} C_5^i\left[ZR_i\right]\right) - kd_{ZEB}\left[ZEB\right] \\
\frac{d\left[miR200\right]}{dt} &= k0_{200} + \frac{k_{200}}{1 + \left(\left[SNAIL\right]/J1_{200}\right)^{n1_{200}} + \left(\left[ZEB\right]/J2_{200}\right)^{n2_{200}}} \\
&- kd_{200}\left(\left[miR200\right] - \sum_{i=1}^{5} iC_5^i\left[ZR_i\right] - \left[TR\right]\right)
\end{aligned}
$$

$$
\begin{aligned}
& - \quad \sum_{i=1}^{5} (1-\lambda_i)\, kd_{ZR_i} C_5^i i\,[ZR_i] - (1-\lambda_{TR})\, kd_{TR}\,[TR] \\
\frac{d\,[ZR_1]}{dt} &= Tk\left(K_1\left([miR200] - \sum_{i=1}^{5} iC_5^i\,[ZR_i] - [TR]\right)\right. \\
&\quad \left.\left([zeb] - \sum_{i=1}^{5} C_5^i\,[ZR_i]\right) - [ZR_1]\right) \\
\frac{d\,[ZR_2]}{dt} &= Tk\left(K_2\left([miR200] - \sum_{i=1}^{5} iC_5^i\,[ZR_i] - [TR]\right)[ZR_1] - [ZR_2]\right) \\
\frac{d\,[ZR_3]}{dt} &= Tk\left(K_3\left([miR200] - \sum_{i=1}^{5} iC_5^i\,[ZR_i] - [TR]\right)[ZR_1] - [ZR_3]\right) \\
\frac{d\,[ZR_4]}{dt} &= Tk\left(K_4\left([miR200] - \sum_{i=1}^{5} iC_5^i\,[ZR_i] - [TR]\right)[ZR_1] - [ZR_4]\right) \\
\frac{d\,[ZR_5]}{dt} &= Tk\left(K_5\left([miR200] - \sum_{i=1}^{5} iC_5^i\,[ZR_i] - [TR]\right)[ZR_1] - [ZR_5]\right) \\
\frac{d\,[tgf]}{dt} &= k_{tgf} - kd_{tgf}\,([tgf] - [TR]) - kd_{TR}\,[TR] \\
\frac{d\,[TGF]}{dt} &= k0_{TGF} + k_{TGF}\,([tgf] - [TR]) - kd_{TGF}\,[TGF] \\
\frac{d\,[TR]}{dt} &= Tk\left(K_{TR}\left([miR200] - \sum_{i=1}^{5} iC_5^i\,[ZR_i] - [TR]\right)([tgf] - [TR]) - [TR]\right) \\
\frac{d\,[Ecad]}{dt} &= k0_E + \frac{k_{E1}}{1 + ([SNAIL]/J1_E)^{n1_E}} + \frac{k_{E2}}{1 + ([ZEB]/J2_E)^{n2_E}} - kd_E\,[Ecad] \\
B &= k_{V1}\frac{([SNAIL]/J1_V)^{n1_V}}{1 + ([SNAIL]/J1_V)^{n1_V}} + k_{V2}\frac{([ZEB]/J2_V)^{n2_V}}{1 + ([ZEB]/J2_V)^{n2_V}} \\
\frac{d\,[Vim]}{dt} &= k0_V + \frac{B}{(1 + [OVOL2]/J3_V)} - kd_V\,[Vim] \\
\frac{d\,[OVOL2]}{dt} &= k0_0 + k_0\frac{1}{1 + ([ZEB]/J_0)^{n_0}} - kd_O\,[OVOL2] \\
\frac{d\,[OVOL2]_p}{dt} &= k_{Op}\,[OVOL2] - kd_{Op}\,[OVOL2]_p
\end{aligned}
$$

where

$$
\begin{aligned}
\sum_{i=1}^{5} iC_5^i\,[ZR_i] &= 5\,[ZR_1] + 20\,[ZR_2] + +30\,[ZR_3] + 20\,[ZR_4] + 5\,[ZR_5]\,, \\
\sum_{i=1}^{5} C_5^i\,[ZR_i] &= 5\,[ZR_1] + 10\,[ZR_2] + 10\,[ZR_3] + 5\,[ZR_4] + [ZR_5]\,, \\
[TGF0]\,(t) &= \begin{cases} \frac{1}{2} & t > 100 \\ 0 & o.w. \end{cases}
\end{aligned}
$$

The parameter values are given in Table 5.

Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
$J1_{200}$	3	$J1_E$	0.1	K_2	1	$k0_O$	0.35
$J2_{200}$	0.2	$J2_E$	0.3	K_3	1	kO_{200}	0.0002
$J1_{34}$	0.15	$J1_V$	0.4	K_4	1	kO_{34}	0.001
$J2_{34}$	0.35	$J2_V$	0.4	K_5	1	kd_{snail}	0.09
J_O	0.9	$J3_V$	2	K_{TR}	20	kd_{tgf}	0.1
$J0_{snail}$	0.6	$J1_{zeb}$	3.5	K_{SR}	100	kd_{zeb}	0.1
$J1_{snail}$	0.5	$J2_{zeb}$	0.9	$TGF0$	0	kd_{TGF}	0.9
$J2_{snail}$	1.8	K_1	1	Tk	1000	kd_{ZEB}	1.66
$k0_{snail}$	0.0005	$k0_{zeb}$	0.003	λ_1	0.5	$k0_{TGF}$	1.1
$n1_{200}$	3	$n1_{snail}$	2	λ_2	0.5	$k0_E$	5
$n2_{200}$	2	$n1_E$	2	λ_3	0.5	$k0_V$	5
$n1_{34}$	2	$n2_E$	2	λ_4	0.5	k_{E1}	15
$n2_{34}$	2	$n1_V$	2	λ_5	0.5	k_{E2}	5
n_O	2	$n2_V$	2	λ_{SR}	0.5	k_{V1}	2
$n0_{snail}$	2	$n2_{zeb}$	6	λ_{TR}	0.5	k_{V2}	5
k_O	1.2	k_{200}	0.02	k_{34}	0.01	k_{tgf}	0.05
k_{zeb}	0.06	k_{TGF}	1.5	k_{SNAIL}	16	k_{ZEB}	16
kd_{ZR_1}	0.5	kd_{ZR_2}	0.5	kd_{ZR_3}	0.5	kd_{ZR_4}	0.5
kd_{ZR_5}	0.5	kd_O	1.0	kd_{200}	0.035	kd_{34}	0.035
kd_{SR}	0.9	kd_E	0.05	kd_V	0.05	k_{Op}	10
						kd_{Op}	10

Table 5: Table of Parameter Values for the EMT Model.

Parameter	Value	Parameter	Value	Parameter	Value
$\sigma_{RA_{in}}$,σ_{RA-RAR},$\sigma_{RA_{out}}$	0.03	ω	100	u	0.01
b	0.17	γ	3.0	d	0.1
α	10000	δ	0.0013	e	1
β_0	1	η	0.0001	a	1
c	0.1	r	0.0001	ζ	0.02
ν	0.85	λ	0.85	D	250.46

Table 6: Table of Parameter Values for the EMT Model.

8.2 Retinoic Acid SPDE Model

$$
\begin{aligned}
d\left[RA_{out}\right] &= \left(\beta(x) + D\Delta\left[RA_{out}\right] - b\left[RA_{out}\right] + c\left[RA_{in}\right]\right)dt + \sigma_{RA_{out}} dW_t^{out} \\
d\left[RA_{in}\right] &= \left(b\left[RA_{out}\right] + \delta\left[BP\right]\left[RA - RAR\right] - \left(\gamma\left[BP\right] + \eta + \frac{\alpha\left[RA-RAR\right]}{\omega + \left[RA-RAR\right]} - c\right)\left[RA_{in}\right]\right)dt \\
d\left[RA - BP\right] &= \left(\gamma\left[BP\right]\left[RA_{in}\right] + \lambda\left[BP\right]\left[RA - RAR\right] - \left(\delta + \nu\left[RAR\right]\right)\left[RA - BP\right]\right)dt \\
d\left[RA - RAR\right] &= \left(\nu\left[RA - BP\right]\left[RAR\right] - \lambda\left[BP\right]\left[RA - RAR\right]\right)dt + \sigma_{RA-RAR}\left[RA - RAR\right] dW_t^{RA-RAR} \\
d\left[BP\right] &= \left(a - \lambda\left[BP\right]\left[RA - RAR\right] - \gamma\left[BP\right]\left[RA_{in}\right] + \left(\delta + \nu\left[RAR\right]\right)\left[RA - BP\right] - u\left[BP\right] + \frac{d\left[RA-RAR\right]}{e + \left[RA-RAR\right]}\right)dt \\
d\left[RAR\right] &= \left(\zeta - \nu\left[RA - BP\right]\left[RAR\right] + \lambda\left[BP\right]\left[RA - RAR\right] - r\left[RAR\right]\right)dt
\end{aligned}
$$

where $\beta(x) = \beta_0 H(x-40)$ with H the Heaviside step function and $x = 40$ is the edge of retinoic acid production. The space was chosen as $[-100, 400] \times [0, 100]$ with $\Delta x = \Delta y = 5$. The boundary conditions were no-flex on every side except the right side which had leaky boundary conditions with parameter $kA = 0.002$, though full no-flux does not noticably change the results. The parameter values are given in Table 6.

9 Appendix I: SOSRA and SOSRI Tableaus

All entries not listed are zero.

9.1 SOSRA

Coefficient	Value	Coefficient	Value
α_1	0.2889874966892885	$\beta_3^{(1)}$	0.27753845684143835
α_2	0.6859880440839937	$\beta_1^{(2)}$	0.4237535769069274
α_3	0.025024459226717772	$\beta_2^{(2)}$	0.6010381474428539
$c_1^{(0)}$	0	$\beta_3^{(2)}$	-1.0247917243497813
$c_2^{(0)}$	0.6923962376159507	$A_{2,1}^{(0)}$	0.6923962376159507
$c_3^{(0)}$	1	$A_{3,1}^{(0)}$	-3.1609142252828395
$c_1^{(1)}$	0	$A_{3,2}^{(0)}$	4.1609142252828395
$c_2^{(1)}$	0.041248171110700504	$B_{2,1}^{(0)}$	1.3371632704399763
$c_3^{(1)}$	1	$B_{3,1}^{(0)}$	1.442371048468624
$\beta_1^{(1)}$	-16.792534242221663	$B_{3,2}^{(0)}$	1.8632741501139225
$\beta_2^{(1)}$	17.514995785380226		

9.2 SOSRA2

Coefficient	Value	Coefficient	Value
α_1	0.4999999999999998	$\beta_3^{(1)}$	0.07561967854316998
α_2	-0.9683897375354181	$\beta_1^{(2)}$	1
α_3	1.4683897375354185	$\beta_2^{(2)}$	-0.8169981105823436
$c_1^{(0)}$	0	$\beta_3^{(2)}$	-0.18300188941765633
$c_2^{(0)}$	1	$A_{2,1}^{(0)}$	1
$c_3^{(0)}$	1	$A_{3,1}^{(0)}$	0.9511849235504364
$c_1^{(1)}$	0	$A_{3,2}^{(0)}$	0.04881507644956362
$c_2^{(1)}$	1	$B_{2,1}^{(0)}$	0.7686101171003622
$c_3^{(1)}$	1	$B_{3,1}^{(0)}$	0.43886792994934987
$\beta_1^{(1)}$	0	$B_{3,2}^{(0)}$	0.7490415909204886
$\beta_2^{(1)}$	0.92438032145683		

9.3 SOSRI

Coefficient	Value	Coefficient	Value
$A^{(0)}_{2,1}$	-0.04199224421316468	α_3	0.4736296532772559
$A^{(0)}_{3,1}$	2.842612915017106	α_4	0.026404498125060714
$A^{(0)}_{3,2}$	-2.0527723684000727	$c^{(0)}_2$	-0.04199224421316468
$A^{(0)}_{4,1}$	4.338237071435815	$c^{(0)}_3$	0.7898405466170333
$A^{(0)}_{4,2}$	-2.8895936137439793	$c^{(0)}_4$	3.7504010171562823
$A^{(0)}_{4,3}$	2.3017575594644466	$c^{(1)}_1$	0
$A^{(1)}_{2,1}$	0.26204282091330466	$c^{(1)}_2$	0.26204282091330466
$A^{(1)}_{3,1}$	0.20903646383505375	$c^{(1)}_3$	0.05879875232001766
$A^{(1)}_{3,2}$	-0.1502377115150361	$c^{(1)}_4$	0.758661169101175
$A^{(1)}_{4,1}$	0.05836595312746999	$\beta^{(1)}_1$	-1.8453464565104432
$A^{(1)}_{4,2}$	0.6149440396332373	$\beta^{(1)}_2$	2.688764531100726
$A^{(1)}_{4,3}$	0.08535117634046772	$\beta^{(1)}_3$	-0.2523866501071323
$B^{(0)}_{2,1}$	-0.21641093549612528	$\beta^{(1)}_4$	0.40896857551684956
$B^{(0)}_{3,1}$	1.5336352863679572	$\beta^{(2)}_1$	0.4969658141589478
$B^{(0)}_{3,2}$	0.26066223492647056	$\beta^{(2)}_2$	-0.5771202869753592
$B^{(0)}_{4,1}$	-1.0536037558179159	$\beta^{(2)}_3$	-0.12919702470322217
$B^{(0)}_{4,2}$	1.7015284721089472	$\beta^{(2)}_4$	0.2093514975196336
$B^{(0)}_{4,3}$	-0.20725685784180017	$\beta^{(3)}_1$	2.8453464565104425
$B^{(1)}_{2,1}$	-0.5119011827621657	$\beta^{(3)}_2$	-2.688764531100725
$B^{(1)}_{3,1}$	2.67767339866713	$\beta^{(3)}_3$	0.2523866501071322
$B^{(1)}_{3,2}$	-4.9395031322250995	$\beta^{(3)}_4$	-0.40896857551684945
$B^{(1)}_{4,1}$	0.15580956238299215	$\beta^{(4)}_1$	0.11522663875443433
$B^{(1)}_{4,2}$	3.2361551006624674	$\beta^{(4)}_2$	-0.57877086147738
$B^{(1)}_{4,3}$	-1.4223118283355949	$\beta^{(4)}_3$	0.2857851028163886
α_1	1.140099274172029	$\beta^{(4)}_4$	0.17775911990655704
α_2	-0.6401334255743456		

9.4 SOSRI2

Coefficient	Value	Coefficient	Value
$A^{(0)}_{2,1}$	0.13804532298278663	α_3	0.686995463807979
$A^{(0)}_{3,1}$	0.5818361298250374	α_4	-0.2911544680711602
$A^{(0)}_{3,2}$	0.4181638701749618	$c^{(0)}_2$	0.13804532298278663
$A^{(0)}_{4,1}$	0.4670018408674211	$c^{(0)}_3$	1
$A^{(0)}_{4,2}$	0.8046204792187386	$c^{(0)}_4$	1
$A^{(0)}_{4,3}$	-0.27162232008616016	$c^{(1)}_1$	0
$A^{(1)}_{2,1}$	0.45605532163856893	$c^{(1)}_2$	0.45605532163856893
$A^{(1)}_{3,1}$	0.7555807846451692	$c^{(1)}_3$	1
$A^{(1)}_{3,2}$	0.24441921535482677	$c^{(1)}_4$	1
$A^{(1)}_{4,1}$	0.6981181143266059	$\beta^{(1)}_1$	-0.45315689727309133
$A^{(1)}_{4,2}$	0.3453277086024727	$\beta^{(1)}_2$	0.8330937231303951
$A^{(1)}_{4,3}$	-0.04344582292908241	$\beta^{(1)}_3$	0.3792843195533544
$B^{(0)}_{2,1}$	0.08852381537667678	$\beta^{(1)}_4$	0.24077885458934192
$B^{(0)}_{3,1}$	1.0317752458971061	$\beta^{(2)}_1$	-0.4994383733810986
$B^{(0)}_{3,2}$	0.4563552922077882	$\beta^{(2)}_2$	0.9181786186154077
$B^{(0)}_{4,1}$	1.73078280444124	$\beta^{(2)}_3$	-0.25613778661003145
$B^{(0)}_{4,2}$	-0.46089678470929774	$\beta^{(2)}_4$	-0.16260245862427797
$B^{(0)}_{4,3}$	-0.9637509618944188	$\beta^{(3)}_1$	1.4531568972730915
$B^{(1)}_{2,1}$	0.6753186815412179	$\beta^{(3)}_2$	-0.8330937231303933
$B^{(1)}_{3,1}$	-0.07452812525785148	$\beta^{(3)}_3$	-0.3792843195533583
$B^{(1)}_{3,2}$	-0.49783736486149366	$\beta^{(3)}_4$	-0.24077885458934023
$B^{(1)}_{4,1}$	-0.5591906709928903	$\beta^{(4)}_1$	-0.4976090683622265
$B^{(1)}_{4,2}$	0.022696571806569924	$\beta^{(4)}_2$	0.9148155835648892
$B^{(1)}_{4,3}$	-0.8984927888368557	$\beta^{(4)}_3$	-1.4102107084476505
α_1	-0.15036858140642623	$\beta^{(4)}_4$	0.9930041932449877
α_2	0.7545275856696072		

10 Appendix II: SRK Order Conditions

10.1 Order Conditions for Rößler-SRI Methods

The coefficients
$\left(A_0, B_0, \beta^{(i)}, \alpha\right)$ must satisfy the following order conditions to achieve order .5:

1. $\alpha^T e = 1$
2. ${\beta^{(1)}}^T e = 1$
3. ${\beta^{(2)}}^T e = 0$
4. ${\beta^{(3)}}^T e = 0$
5. ${\beta^{(4)}}^T e = 0$

additionally, for order 1:

1. $\beta^{(1)^T} B^{(1)} e = 0$
2. $\beta^{(2)^T} B^{(1)} e = 1$
3. $\beta^{(3)^T} B^{(1)} e = 0$
4. $\beta^{(4)^T} B^{(1)} e = 0$

and lastly for order 1.5:

1. $\alpha^T A^{(0)} e = \frac{1}{2}$
2. $\alpha^T B^{(0)} e = 1$
3. $\alpha^T \left(B^{(0)} e\right)^2 = \frac{3}{2}$
4. $\beta^{(1)^T} A^{(1)} e = 1$
5. $\beta^{(2)^T} A^{(1)} e = 0$
6. $\beta^{(3)^T} A^{(1)} e = -1$
7. $\beta^{(4)^T} A^{(1)} e = 0$
8. $\beta^{(1)^T} \left(B^{(1)} e\right)^2 = 1$
9. $\beta^{(2)^T} \left(B^{(1)} e\right)^2 = 0$
10. $\beta^{(3)^T} \left(B^{(1)} e\right)^2 = -1$
11. $\beta^{(4)^T} \left(B^{(1)} e\right)^2 = 2$
12. $\beta^{(1)^T} \left(B^{(1)} \left(B^{(1)} e\right)\right) = 0$
13. $\beta^{(2)^T} \left(B^{(1)} \left(B^{(1)} e\right)\right) = 0$
14. $\beta^{(3)^T} \left(B^{(1)} \left(B^{(1)} e\right)\right) = 0$
15. $\beta^{(4)^T} \left(B^{(1)} \left(B^{(1)} e\right)\right) = 1$
16. $\frac{1}{2}\beta^{(1)^T} \left(A^{(1)} \left(B^{(0)} e\right)\right) + \frac{1}{3}\beta^{(3)^T} \left(A^{(1)} \left(B^{(0)} e\right)\right) = 0$

where $f, g \in C^{1,2}(\mathcal{I} \times \mathbb{R}^d, \mathbb{R}^d)$, $c^{(i)} = A^{(i)} e$, $e = (1, 1, 1, 1)^T$ [30].

10.2 Order Conditions for Rößler-SRA Methods

The coefficients
$\left(A_0, B_0, \beta^{(i)}, \alpha\right)$ must satisfy the conditions for order 1:

1. $\alpha^T e = 1$
2. $\beta^{(1)^T} e = 1$
3. $\beta^{(2)^T} e = 0$

and the additional conditions for order 1.5:

1. $\alpha^T B^{(0)} e = 1$
2. $\alpha^T A^{(0)} e = \frac{1}{2}$
3. $\alpha^T \left(B^{(0)} e\right)^2 = \frac{3}{2}$
4. $\beta^{(1)^T} c^{(1)} = 1$
5. $\beta^{(2)^T} c^{(1)} = -1$

where $c^{(0)} = A^{(0)} e$ with $f \in C^{1,3}(\mathcal{I} \times \mathbb{R}^d, \mathbb{R}^d)$ and $g \in C^1(\mathcal{I}, \mathbb{R}^d)$ [30].

11 Appendix III: Derivation Details

$$
\begin{aligned}
\left(I-\mu\Delta tA^{(0)}\right)H^{(0)} &= U_n+\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}\left(U_n+\mu\Delta tA^{(1)}H^{(0)}\right),\\
\left(I-\mu\Delta tA^{(0)}\right)H^{(0)}-\left[\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}\right]\mu\Delta tA^{(1)}H^{(0)} &= U_n+\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}U_n\\
\left(I-\mu\Delta tA^{(0)}-\mu\Delta tA^{(1)}\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}\right)H^{(0)} &= \left(I+\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}\right)U_n\\
H^{(0)} &= \left(I-\mu\Delta tA^{(0)}-\mu\sigma I_{(1,0)}A^{(1)}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}\right)^{-1}\\
&\quad\left(I+\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}\right)U_n
\end{aligned}
$$

$$
\begin{aligned}
\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)H^{(1)} &= U_n+\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\left(U_n+\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}H^{(1)}\right)\\
\left(I-\sigma\sqrt{\Delta t}B^{(1)}-\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\right)H^{(1)} &= U_n+\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}U_n\\
\left(I-\sigma\sqrt{\Delta t}B^{(1)}-\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\right)H^{(1)} &= \left(I+\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\right)U_n\\
H^{(1)} &= \left(I-\sigma\sqrt{\Delta t}B^{(1)}-\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\right)^{-1}\\
&\quad\left(I+\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\right)U_n
\end{aligned}
$$

$$
\begin{aligned}
U_{n+1} = U_n &+\mu\Delta t\left(\alpha\cdot\left[\left(I-\mu\Delta tA^{(0)}-\mu\sigma I_{(1,0)}A^{(1)}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}\right)^{-1}\left(I+\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\left(I-\sigma\sqrt{\Delta t}B^{(1)}\right)^{-1}\right)\right]U_n\right)\\
&+\sigma I_{(1)}\left(\beta^{(1)}\cdot\left[\left(I-\sigma\sqrt{\Delta t}B^{(1)}-\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\right)^{-1}\left(I+\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\right)\right]U_n\right)\\
&+\sigma\frac{I_{(1,1)}}{\sqrt{\Delta t}}\left(\beta^{(2)}\cdot\left[\left(I-\sigma\sqrt{\Delta t}B^{(1)}-\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\right)^{-1}\left(I+\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\right)\right]U_n\right)\\
&+\sigma\frac{I_{(1,0)}}{\Delta t}\left(\beta^{(3)}\cdot\left[\left(I-\sigma\sqrt{\Delta t}B^{(1)}-\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\right)^{-1}\left(I+\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\right)\right]U_n\right)\\
&+\sigma\frac{I_{(1,1,1)}}{\Delta t}\left(\beta^{(4)}\cdot\left[\left(I-\sigma\sqrt{\Delta t}B^{(1)}-\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\sigma\frac{I_{(1,0)}}{\Delta t}B^{(0)}\right)^{-1}\left(I+\mu\Delta tA^{(1)}\left(I-\mu\Delta tA^{(0)}\right)^{-1}\right)\right]U_n\right)
\end{aligned}
$$

Thus we substitute in the Wiktorsson approximations

$$
\begin{aligned}
I_{(i,i)} &= \frac{1}{2}\left(\Delta W^2-h\right)\\
I_{(i,i,i)} &= \frac{1}{6}\left(\Delta W^3-3h\Delta W\right)\\
I_{(i.0)} &= \frac{1}{2}h\left(\Delta W+\frac{1}{\sqrt{3}}\Delta Z\right)
\end{aligned}
$$

where $\Delta Z\sim N(0,h)$ is independent of $\Delta W\sim N(0,h)$. By the properties of the normal distribution, we have that

$$E\left[(\Delta W)^n\right]=0$$

for any odd n and

$$\begin{aligned} E\left[(\Delta W)^2\right] &= h \\ E\left[(\Delta W)^4\right] &= 3h^2 \\ E\left[(\Delta W)^6\right] &= 15h^3 \\ E\left[(\Delta W)^8\right] &= 105h^4, \end{aligned}$$

and similarly for ΔZ.

References

[1] A. Abdulle and S. Cirilli. S-rock: Chebyshev methods for stiff stochastic differential equations. *SIAM Journal on Scientific Computing*, 30(2):997–1014, 2008.

[2] David I. Ketcheson Ahmadia and Aron J. Optimal stability polynomials for numerical integration of initial value problems. *CAMCOS*, 7(2):247–271, 2012.

[3] E. G. Birgin and J. M. Martinez. Improving ultimate convergence of an augmented lagrangian method. *Optimization Methods and Software*, 23(2):177–195, 2008.

[4] Kevin Burrage and J. C. Butcher. Non-linear stability of a general class of differential equation methods. *BIT Numerical Mathematics*, 20(2):185–203, 1980.

[5] J. C. Butcher. A history of runge-kutta methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.

[6] J. C. Butcher. Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics*, 125(1â2):1–29, 2000.

[7] A. Conn, N. Gould, and P. Toint. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[8] J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[9] I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[10] Hagen Gilsing and Tony Shardlow. Sdelab: A package for solving stochastic differential equations in matlab. *Journal of Computational and Applied Mathematics*, 205(2):1002–1018, 2007.

[11] E. Hairer, S. P. NÃžrsett, and Gerhard Wanner. *Solving ordinary differential equations I : nonstiff problems*. Springer series in computational mathematics,. Springer, Heidelberg ; London, 2nd rev. edition, 2009.

[12] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems*. Springer, 1991.

[13] Ernst Hairer and Gerhard Wanner. Stiff differential equations solved by radau methods. *Journal of Computational and Applied Mathematics*, 111(1):93–111, 1999.

[14] Tian Hong, Kazuhide Watanabe, Catherine Ha Ta, Alvaro Villarreal-Ponce, Qing Nie, and Xing Dai. An ovol2-zeb1 mutual inhibitory circuit governs bidirectional and multi-step transition between epithelial and mesenchymal states. *PLoS Comput Biol*, 11(11):e1004569, 2015.

[15] M. E. Hosea and L. F. Shampine. Analysis and implementation of tr-bdf2. *Applied Numerical Mathematics*, 20(1):21–37, 1996.

[16] Aleksander Janicki, Adam Izydorczyk, and Przemyslaw Gradalski. *Computer Simulation of Stochastic Models with SDE-Solver Software Package*, pages 361–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[17] Steven G. Johnson. The nlopt nonlinear-optimization package.

[18] Christopher A. Kennedy and Mark H. Carpenter. Additive runge-kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44(1):139–181, 2003.

[19] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer Berlin Heidelberg, 2011.

[20] Yoshio Komori and Kevin Burrage. Weak second order s-rock methods for stratonovich stochastic differential equations. *Journal of Computational and Applied Mathematics*, 236(11):2895–2908, 2012.

[21] Yoshio Komori and Kevin Burrage. Strong first order s-rock methods for stochastic differential equations. *Journal of Computational and Applied Mathematics*, 242(Supplement C):261–274, 2013.

[22] F Shampine Lawrence. Some practical runge-kutta formulas. *Math. Comput.*, 46(173):135–150, 1986.

[23] J. Lawson. An order five runge-kutta process with extended region of stability. *SIAM Journal on Numerical Analysis*, 3(4):593–597, 1966.

[24] Xuerong Mao. The truncated eulerâmaruyama method for stochastic differential equations. *Journal of Computational and Applied Mathematics*, 290(Supplement C):370–384, 2015.

[25] Jan Kloppenborg Moller and Henrik Madsen. From state dependent diffusion to constant diffusion in stochastic differential equations by the lamperti transform. Report, Technical University of Denmark, DTU Informatics, Building 321, 2010.

[26] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis, Proceedings of the 6th Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, volume 275, pages 51–67. Kluwer Academic Publishers.

[27] Christopher Rackauckas and Qing Nie. Mean-independent noise attenuation via intermediate states. *Submitted*.

[28] Christopher Rackauckas and Qing Nie. Adaptive methods for stochastic differential equations via natural embeddings and rejection sampling with memory. *Discrete and Continuous Dynamical Systems - Series B*, 22(7):2731–2761, 2016.

[29] Christopher Rackauckas and Qing Nie. Differentialequations.jl - a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1):15, 2017.

[30] Andreas Rossler. Runge kutta methods for the strong approximation of solutions of stochastic differential equations. *SIAM Journal on Numerical Analysis*, 48(3):922–952, 2010.

[31] T. P. Runarsson and Yao Xin. Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(2):233–243, 2005.

[32] Thomas Schaffter. *From genes to organisms: Bioinformatics System Models and Software*. Thesis, 2014.

[33] Gustaf SÃ¶derlind and Lina Wang. Evaluating numerical ode/dae methods, algorithms and software. *Journal of Computational and Applied Mathematics*, 185(2):244–260, 2006.

[34] L. F. Shampine. Stiffness and nonstiff differential equation solvers, ii: Detecting stiffness with runge-kutta methods. *ACM Trans. Math. Softw.*, 3(1):44–53, 1977.

[35] L. F. Shampine and K. L. Hiebert. Detecting stiffness with the fehlberg (4, 5) formulas. *Computers Mathematics with Applications*, 3(1):41–46, 1977.

[36] Wayne H. Enright Sharp, Desmond J. Higham, Brynjulf Owren, and Philip W. A survey of the explicit runge-kutta method. 1995.

[37] Ch Tsitouras. Runge-kutta pairs of order 5(4) satisfying only the first column simplifying assumption. *Computers Mathematics with Applications*, 62(2):770–775, 2011.

[38] P. J. van der Houwen. Explicit runge-kutta formulas with increased stability boundaries. *Numerische Mathematik*, 20(2):149–164, 1972.

[39] Magnus Wiktorsson. Joint characteristic function and simultaneous simulation of iterated ito integrals for multiple independent brownian motions. *The Annals of Applied Probability*, pages 470–487, 2001.