Stability-Optimized High Strong Order Methods and Stiffness Detection for Stochastic Differential Equations with Additive and Diagonal Noise

Chris Rackauckas and Qing Nie

August 21, 2017

1 Introduction

Stochastic differential equations have seen increasing use in scientific fields such as biology and climate science due to their ability to capture the randomness inherent in physical systems. These equations are 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 studied for their qualitatative behavior: scientists are interested not in numerical predictions exact to many decimal places but to understand the phonomena 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 noise. This results in $g(t, X_t)$ being the diagonal matrix $\left(\sigma_i X_t^i\right)$ and thus falling into the more general category of diagonal noise.

The unique features of stochastic models in many cases are pathwise-dependent. The mean of a chemical reaction network may stay at a constant steady state, but in the presence of randomness this may be switching between various states. These pathwise properties are of interest because they capture the effects which cannot be found in deterministic models. However, these same effects exhibit numerical difficulties. Almost by definition these features exist in the single trajectories of the random processes and thus must be controlled individually. These trajectories display large, transient and random switching behavior which in a given trajectory causes stochastic bursts of numerical stiffness, a phonomena which we will denote pathwise stiffness. In previous work, the authors have shown that by using adaptive timestepping a stochastic reaction network of 19 reactants is able to be solve with an average timestep 100,000x larger than the value that was found necessary for stability during the random stiff events. 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 two means. On one end we develop

stability-optimized Stochastic Runge-Kutta (SRK) methods which have the property of having drastically enlarged stability regions. Similar to the Runge-Kutta Chebyschev (and the S-ROCK extension to the stochastic case), these methods are designed to be efficient for equations which display stiffness without fully commiting to implicit solvers. On the otherhand, to handle extreme stiffness we develop an implicit RK method for additive noise problems. We show that this method is A-L stable in a generalization of these terms to additive noise SDEs. To extend the utility of these methods, we derive and extension of the methods for additive SDEs to multiplicative SDEs through a transformation. In addition to the new methods, we display a novel scalable mechanism for the derivation of "optimal" Runge-Kutta methods, and use it to design stability-optimized methods for diagonal noise SDEs which would otherwise be analytically intractable due to the few million terms in the stability equation. Lastly, we show that on test problems that these methods are no less efficient than existing SRK methods when one only requires a few decimal places of accuracy ($> 10^{-6}$), but show that these methods have two to three times the stability region, allowing them to dramatically speed up computations on stability-bound problems.

2 Adaptive Strong Order 1.0/1.5 SRK Methods

The class of methods we wish to example are the Strong Order 1.5 SRK methods due to Rossler. 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

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

with stages

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

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

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

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

with stages

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

The tuple of coefficients $\left(A^{(j)}, B^{(j)}, \beta^{(j)}, \alpha\right)$ thus fully determine the SRK method. These coefficients are subject to the constraint equations described in Appendix X in order to receive strong order 1.5. Rackauckas and Nie constructed a systematic method for developing an embedded strong order 1.0 method from any SRK tableau and showed that adaptive timestepping using this estimator is fundamental for the efficient

simulation of pathwise stiff models. Thus unlike in the theory of ordinary differential equations, 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 in time.

3 Optimized-Stability High Order SRK Methods with Additive Noise

Using the terms as defined by Kloden and Platen, we define 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 $\delta \in (0, \Delta_0)$

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

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 $\bar{X}_n^\delta$. For additive noise, we consider the complex-valued linear test equations

$$dX_t = \mu X_t dt + dW_t$$

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

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

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

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

where

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

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

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

This set of equations decouples to the form of Equation ## 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

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

which one may notice is the stability equation of the drift tableau applied to a deterministic ODE. 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 were developed to satisfy higher order deterministic order constraints which do not apply. We will instead look to maximize stability while satisfying the stochastic order constraints.

3.1 Stability-Optimal 2-Stage Explicit SRA Methods

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

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

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 $A^{(0)}, \alpha$ 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 $z = \frac{3}{4}\left(1 - \sqrt{\frac{19}{3}}\right) \approx -1.13746$. We see that 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. Thus the SRA1 is the stability-optimized 2-stage explicit method which achieves the most robust error estimator.

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

3.2 Stability-Optimal 3-Stage Explicit SRA Methods

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

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

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. In the Mathematica notebooks we should that one root condition can be discarded, but the other two had difficulties optimizing. 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 $z \in \mathbb{R}$. Define the function:

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

Notice that for $D \to \mathbb{C}$, f is the area of the stability region. Thus we define the stability-optimized diagonal SRK method as the set of coefficients which achieves

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

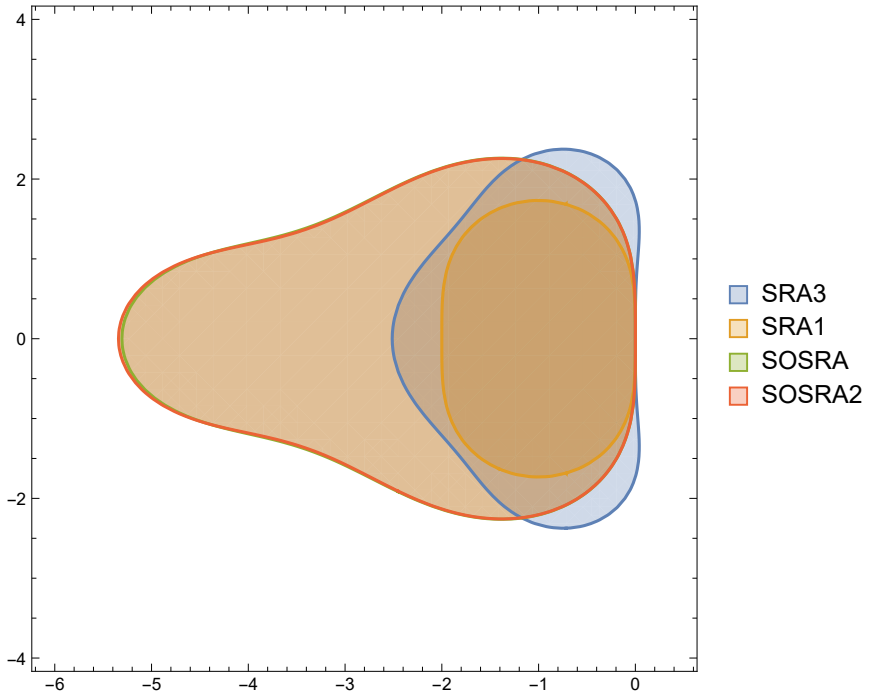

Figure 1: SOSRA Stability Regions. The stability regions ($|G(z)| < 1$) are plotted in the (x, y)-plane for $z = x + iy$. Shown are the SRA methods SRA1 and SRA3 in addition to the SOSRA and SOSRA2 methods. Note that SOSRA and SOSRA2 have nearly identical stability regions, making it hard to decern the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

In all cases we impose $0 < c_i^{(0)}, c_i^{(1)} < 1$. We use the order constraints to simplify the problem to an 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 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 X in the appendix. 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. The method generated with this extra constraint is SOSRA2. These methods have their stability regions compared to SRA1 and SRA3 in Figure X where it is shown that the SOSRA methods more than doubles the allowed real eigenvalues.

3.3 An L-Stable SRA Method: Additive TRBDF2 (ATRBDF2)

Lastly, we wished to find an L-stable method capable of handling highly stiff additive SDEs. Given the connection between SRA and deterministic stability, we looked to find an appropriate generalization of an L-stable determininstic Runge-Kutta method. Since the strong order would not exceed 1.5, we restricted our search to low order implicit Runge-Kutta methods. Extensive numerical testing in the development of DifferentialEquations.jl has shown that the explicit first step singly diagonal implicit Runge-Kutta (ESDIRK) method TRBDF2 (standing for a composition between the Trapezoid and the BDF2 methods) is a very efficient low order implicit Runge-Kutta method. Particularly nice qualities include the fact that it's 2nd order stiffly accurate (and thus applicable to DAEs), has an explicit first stage which is auto-computed via the FSAL (first same as last) property and thus is effectively a two stage method, has very accurate stage predictors, has a stabilization of its error estimate, and performs well on equations which are traditionally difficult stiff test problems like the Van Der Pol equation and the Robertson equation.

Letting $A^{(0)}$, α, and $c^{(0)}$ be the TRBDF2 tableau, our goal is to find suitable values for $B^{(0)}$, $\beta^{(1)}$, $\beta^{(2)}$, and $c^{(1)}$.

4 Optimized-Stability Methods for Multiplicative 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, which states that the SDE of the form

$$dX_t = f(t, X_t)dt + \sigma(t, X_t)R(t)dW_t$$

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

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

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

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

with

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

This is easily verified using Ito's Lemma. An example of such a transformation is multidimensional geometric Brownian motion, where $A = diag(a_1, a_2)$, $\sigma = diag(X_1, X_2)$, and $R = r_{ij}$. Then in this case, $Z = \psi(X) = \log(X)$ and

$$d\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} a_1 - \frac{1}{2}\left(r_{11}^2 + r_{12}^2\right) \\ a_2 - \frac{1}{2}\left(r_{21}^2 + r_{22}^2\right) \end{bmatrix} dt + \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} dW_t.$$

This transformation requires that $\sigma_i^{-1}(t, X_{i,t})$ is one-to-one, and thus does not exist in general for diagonal noise. However, in the case of mixed multiplicative and additive noise:

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

where σ is a constant diagonal matrix, then

$$d\log X_t = \tilde{f}(t, X_t)dt + dW_t$$
$$\tilde{f}(t, X_t) = \frac{f(t, X_t)}{\sigma X_t}$$

where the division is considered element-wise. Thus we can modify the additive SRK method to be in the form

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

with stages

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

Back-transforming this, we get

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

where the exponentiation is interpreted element-wise.

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

$$dX_t = \mu X_t dt + \sigma X_t dW_t.$$

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

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

with stages

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

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

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

By the derivation in the appendix, we receive the equation

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

We apply the substitutions from the Appendix and let

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

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. 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 additive and 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

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

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

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

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 with the NLopt backend. A mixed approach using many solutions of the semi-local optimizer LN_AUGLAG_EQ and fewer solutions from the global optimizer GN_ISRES were used to approximate the optimality of solutions. Optimization was run many times in parallel until many results produced methods with similar optimiality, indicating that we were likely obtained values near the true minimum.

The parameters N and M are the bounds on the stability region, but also represent a tradeoff 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 SRIOpt1, 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 SRIOpt1.

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 simliar to SRIOpt1 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 label SRIOpt2, resulted in similar stability to SRIOpt1 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 X.

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

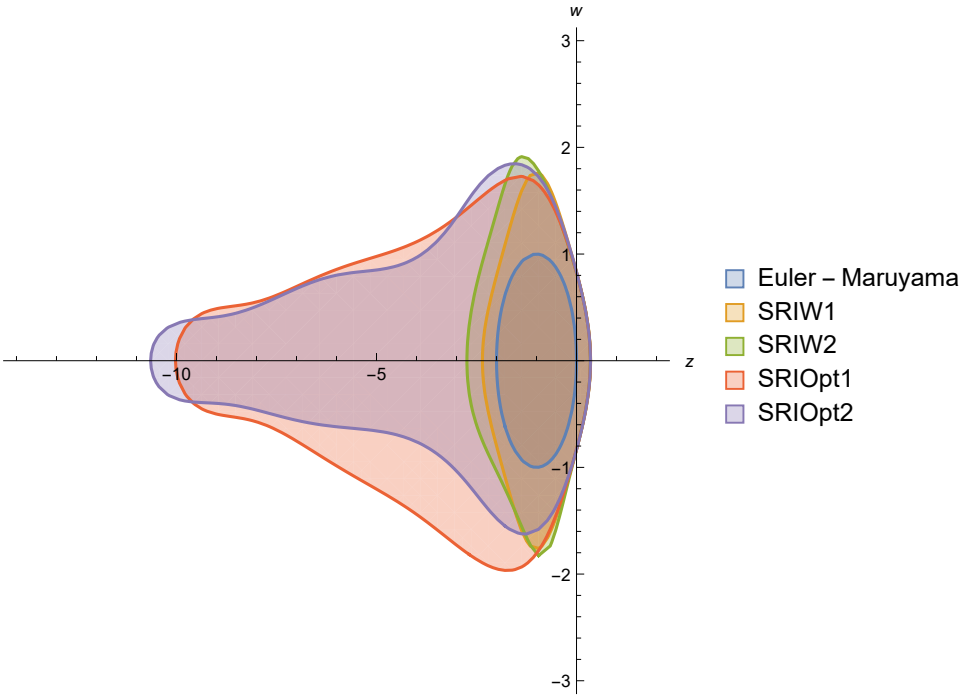

Figure 2: SOSRI Stability Regions. The stability regions for the SOSRI methods are plotted in the (z, w)-plane. In addition, the regions for Euler-Maruyama, SRIW1, and SRIW2 are shown for comparison.

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 tradeoff, we investigated how our methods allow for switching between methods which optimize for the different situations.

The basis for our method is a straight-forward extension of a method proposed for deterministic differential equations. 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,

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

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 timepoint, $c_i^{(k)} = c_j^{(k)}$, then the following serves as a local Jacobian estimate:

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

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

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

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 SRIOpt2, a natural extension is

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

Given that these values are all part of the actual step calculations, this stiffness estimate is free. By comparing these values to the stability plot in Figure X, we use the following heristic to decide if SRIOpt2 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 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 tradeoff 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 method is non-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 SRSRA2 has the same property, allowing stiffness detection via

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

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

6 Numerical Results

6.1 SOSRA Numerical Experiments

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

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

where $\alpha = \frac{1}{10}$ and $\beta = \frac{1}{20}$. Actual Solution:

$$X_t = \frac{1}{\sqrt{1+t}}X_0 + \frac{\beta}{\sqrt{1+t}}\left(t + \alpha W_t\right). \tag{2}$$

To test the efficiency we plotted a work-precision diagram against the SRA1, SRA2, SRA3, and the fixed timestep Euler-Maruyama and a Runge-Kutta Milstein scheme. The results, as depicted in Figure X, 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 siginificantly more efficient than the lower order algorithms when the required error is $< 10^{-4}$. To show that the SOSRA methods indeed achieve the appropriate strong order, a convergence test was conducted and it was shown that four separate measures of error converged at a rate 2.0 (it has previously been noted that SRA methods have a higher convergence rate, at least on this problem. The authors have noticed this phonomena elsewere as well). These results show that in the regime of mild accuracy, these methods are much more efficient than low order methods yet achieve the same efficiency as the non-stability optimized SRA variants.

To test how efficiently the algorithms could achieve their goal, we choose to analyze the qualtative results of the driven Van der Pol equation. The driven Van der Pol equation is given by the system:

$$dy = \mu((1-x^2)y - x)dt$$
$$dx = ydt$$

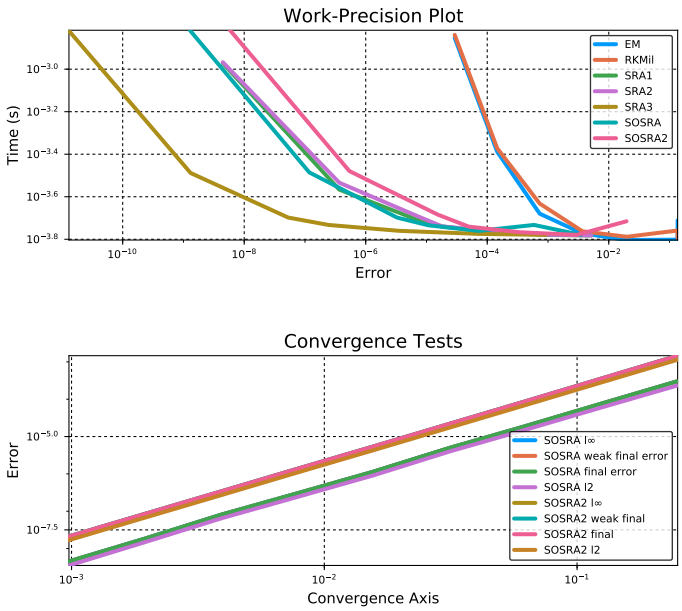

Figure 3: SOSRA efficiency and correctness tests. **(A)** Shown are the work-precision plots for the methods on the additive noise test problem. Each of the adaptive timestepping methods solved the problem on the interval using changing values of tolerances, with $tol = abstol = reltol$ starting at 10^2 and ending at 10^{-4} going in increments of 10. The fixed timestepping methods used timesteps 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 timeseries. The time is the avarage time to compute a trajectory and is averaged over 1000 runs at the same tolerance / timestep size. **(B)** Depicted are the results of a convergence test on the SOSRA and SOSRA2 methods. The test used a fixed timestep $h = 1/2^{-2}$ to $h = 1/2^{-10}$. Shown are four separate error estimates: the strong l_∞ error, the strong l_2 error, the weak error at the final timepoint, and the strong error at the final timepoint. The strong error is averaged over 10 trajectories, which is the same number of trajectories used to compute the weak error estimate.

Algorithm	Runtime	Relative Time (vs SOSRA)
SOSRA	520.054837	1x
SOSRA2	520.847214	1.002x
SRA1	2488.683401	4.785x
SRA3	2033. 793730	3.911x

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

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, 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. Figure X shows the ODE solved using the Tsit5 explicit Runge-Kutta algorithm and demonstrates the mild stiffness which is still well-handled by explicit methods. We extend this model to the driven Van der Pol model with additive noise:

$$dy = \mu((1 - x^2)y - x)dt + \rho dW_t^{(1)}$$
$$dx = y + \rho dW_t^{(2)}$$

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 as shown in Figure X, and SOSRA2 producing similar results at the same tolerances a factor of two lower. However, even at tolerances of $abstol = 1/2^3$,$reltol = 1/2^3$ SRA3 was unable to reproduce the correct qualitative behavior as shown in Figure X. Thus we decreased the tolerances by factors of 2 until it was able to reproduce the correct qualitative results as shown in Figure X. This shows that the SOSRA are more reliable on models with transient stiffness. To test the impact on the runtime of tha algorithms, each of the algorithms were run 100 times with the tolerance setup that allows them to most efficiently generate correct qualitative results. The runtimes are shown in Table X, which show that SRA1 takes nearly 5 times and SRA3 nearly 4 times as long as the SOSRA methods to generate a correct result.

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 are shown in Figure X and The authors note that the 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.

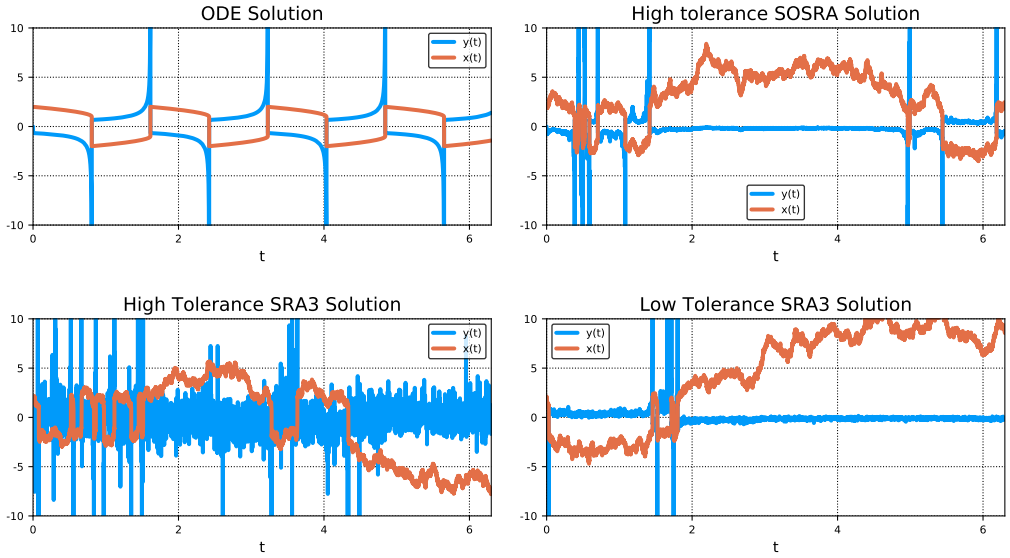

Figure 4: Representative trajectories for solutions to the Van der Pol equations with additive noise. **(A)** The solution to the ODE with the explicit Runge-Kutta method Tsit5. **(B)** The solution to the SDE with tolerance $abstol = 1, reltol = 1/2^1$ from SOSRA. **(C)** Solution to the SDE with tolerances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

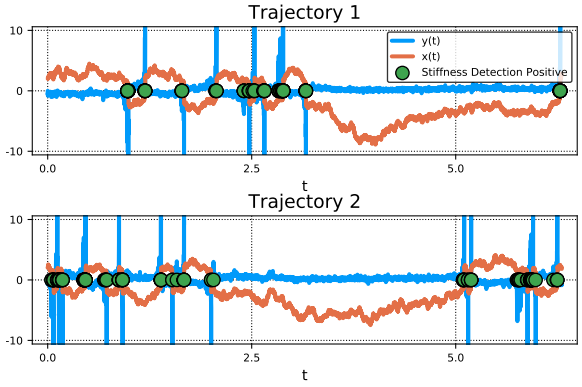

Figure 5: Stiffness detection in the Van der Pol equations with additive noise. Two representative trajectories to the Van der Pol equations are plotted. The green dots indicate timepoints where the stiffness detection algorithm detected stiffness.

6.2 SOSRI Numerical Experiments

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

$$dX_t = \alpha X_t dt + \beta X_t dW_t \;\; X_0 = \frac{1}{2}, \tag{3}$$

where $\alpha = \frac{1}{10}$ and $\beta = \frac{1}{20}$. Actual Solution:

$$X_t = X_0 e^{\left(\beta - \frac{\alpha^2}{2}\right)t + \alpha W_t}. \tag{4}$$

To test the efficiency we plotted a work-precision diagram against the SRIW1, SOSRI, SOSRI2, and the fixed timestep Euler-Maruyama and a Runge-Kutta Milstein schemes. The results, as depicted in Figure X, 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 siginificantly more efficient than the lower order algorithms when the required error is $< 10^{-2}$. To show that the SOSRA methods indeed achieve the appropriate strong order, a convergence test was conducted and it was shown that four separate measures of error converged at a rate 1.5 as depicted in Figure X. These results show that in the regime of mild accuracy, these methods are much more efficient than low order methods yet achieve the same efficiency as the non-stability optimized SRI variants.

To test the real consequences of the enhanced stability, we use the Oval2 model of 19 pathwise stiff reaction equations introduced in X and studied as a numerical test in Y. In the previous work it was noted that $t \in [0, 1]$ was a less stiff version of this model over the timespan $t \in [0, 500]$. 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. In order to test the raw efficiency of the methods, we used the same tableau-based implementation and only replaced the coefficients. The results are depcited in Table X. 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 SRI methods take nearly the same amount of time. 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. We note that this relative tolerance $2^{-18} \approx 3.9 \times 10^{-6}$ is essentially requiring 6 digits of accuracy (in strong error), which is far beyond the accuracy necessary in many cases. Instead, SOSRI can achieve around 2 digits of accuracy in each trajectory in a time that is an order of magnitude faster. Since weak order is higher and weak error is usually lower, this represents more than enough accuracy to adequately solve the problem. We note that the maximal tolerances which is stable for SOSRI2 is lower than that for SOSRI, meaning that it is able to handle less stiff problems as well, but note that it includes a stiffness detection algorithm similar to SOSRA2 and thus tradeoff some stiffness handling for the ability to switch to other algorithms when stiff. We note that timings for Euler-Mauyama and Runge-Kutta Milstein schemes are omitted since the tests were unable to finish. From the results of CITE we note that the average dt for SRIW1 on the edge of its stability had that the smallest dt was approximately 10^{-11}. Figure X shows that the stability region for fixed stepsize Euler-Maruyama is strictly smaller than SRIW1 and thus would require around 5×10^{12} timesteps (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.

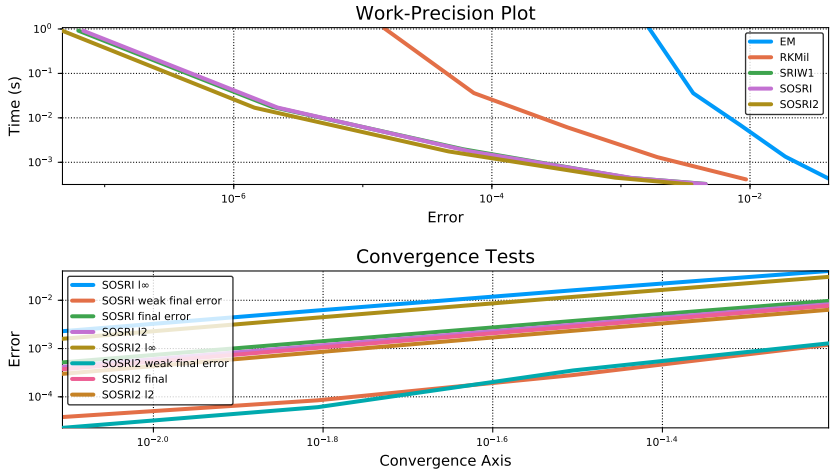

Figure 6: SOSRI efficiency and correctness tests. **(A)** Shown are the work-precision plots for the methods on the scalar noise test problem. Each of the adaptive timestepping methods solved the problem on the interval using changing values of tolerances, with $tol = abstol = reltol$ starting at 10^{-1} and ending at 10^{-5} going in increments of 10. The fixed timestepping methods used timesteps of size $h = 1/5^{-3}$ to $h = 1/5^{8}$, changing the value by factors of 5. The error is the strong l_2 error computed over the timeseries. The time is the avarage time to compute a trajectory and is averaged over 1000 runs at the same tolerance / timestep size. **(B)** Depicted are the results of a convergence test on the SOSRA and SOSRA2 methods. The test used a fixed timestep $h = 1/2^{-2}$ to $h = 1/2^{-10}$. Shown are four separate error estimates: the strong l_∞ error, the strong l_2 error, the weak error at the final timepoint, and the strong error at the final timepoint. The strong error is averaged over 10 trajectories, which is the same number of trajectories used to compute the weak error estimate.

Algorithm	Abstol	Reltol	Runtime	Relative Time (vs SOSRI)
SOSRI	2^{-7}	2^{-4}	4.456220255	1x
SOSRI	2^{-7}	2^{-6}	4.528014259	1.02x
SOSRI	2^{-12}	2^{-15}	15.154482346	3.401x
SOSRI	2^{-14}	2^{-18}	64.385613115	14.45x
SOSRI2	2^{-12}	2^{-15}	15.465410168	3.471x
SOSRI2	2^{-14}	2^{-18}	63.918306532	14.34x
SRIW1	2^{-14}	2^{-18}	68.196888282	15.30x

Table 2: SRI times for the the Oval2 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.

Algorithm	Abstol	Reltol	Runtime	Relative Time (vs SOSRI)
SOSRI	10^{-2}	10^{-2}	42.278439	1x
SOSRI	10^{-4}	10^{-4}	127.546408	3.017x
SOSRI	10^{-5}	10^{-3}	161.754319	3.826x
SOSRI2	10^{-4}	10^{-4}	133.714578	3.163x
SOSRI2	10^{-5}	10^{-3}	189.233673	3.163x
SRIW1	10^{-5}	10^{-3}	357.097802	8.446x

Table 3: SRI times for the the Oval2 model on $t \in [0, 500]$. The equations were solve 10,000 times with the given tolerances to completion and the elapsed time was recorded.

Algorithm	Abstol	Reltol	Runtime	Relative Time (vs SOSRI)
SOSRI	10^{-1}	10^{-1}	4150.078598	1x
SOSRI2	10^{-1}	10^{-1}	17563.2690785	4.232x
SRIW1	10^{-5}	10^{-3}	41407.3344385	9.977x

Table 4: SRI times for the the RA SPDE model on $t \in [0, 50]$. The equations were solved twice with the given tolerances to completion and the elapsed time was recorded. Note that none of the timings varied by more than 1% of the total runtime.

We then timed the runtime to solve 10 trajectories in the $t \in [0, 500]$ case. The results are shown in Table X. This time we found the optimal tolerance in terms of powers of 10. Once again, SRIW1 needed a lower tolerance than is necessary in order to stay stable. SOSRI is able to solve the problem only asking for 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 again, and both are nearly twice as fast as SRIW1 when matching tolerances.

As another test we applied the methods to a method of lines (MOL) discretization of a stochastic partial differential equation (SPDE) describing the spatial regulation of the zebrafish hindbrain via retinoic acid signaling. The model was solved nearly to steady state using a second order Rosenbrock method on the PDE described by the drift term. Starting from 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 are noted in Table X. Since this problem starts from the only positive steady state, it is only semi-stiff. Yet we receieve similar results as the other problems, 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.

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 multiplicative 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 kernal 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 weak noise estimation. 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 an A-L stable method for additive (and thus multiplicative) 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. However, the results clearly showed that the methods with higher number of stages were more efficient, a phonomena seen in other types of Runge-Kutta methods like symplectic RK methods. Thus one interesting avenue of future research would be in the development of 3-stage implicit methods for additive noise equations. While the complexity of the constraints would be higher, the same general approach should yield satisfactory results.

Our timings convincingly 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 neccessary solutions in order to satisfy stability constraints. For additive noise problems we were able to obtain solutions about 4x faster and for diagonal noise 8x-15x than the current methods. We note that the popular Euler-Maruyama and (derivative-free) Milstein methods were previously compared in X and were shown to be more than 12x slower than the SRIW1 method on the Oval2 model for $t \in [0, 1]$. We have also shown that these methods are very robust even at high tolerances and have a tendency to produce the correct qualitative results (via plots) even when the user chosen accuracy is low. Given that the user input is minimal, we see these as very strong candidates for 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 these methods are not efficient at low tolerances, and other methods such as SRA3 (or higher order SRK methods) should be used instead.

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 nonstiff 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: SRK Order Conditions

8.1 Order Conditions for Rößler-SRI Methods

The coefficients
$(A_0, B_0, \beta^{(i)}, \alpha)$ 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 $f, g \in C^{1,2}(\mathcal{I} \times \mathbb{R}^d, \mathbb{R}^d)$, $c^{(i)} = A^{(i)} e$, $e = (1, 1, 1, 1)^T$ [?].

8.2 Order Conditions for Rößler-SRA Methods

The coefficients
$(A_0, B_0, \beta^{(i)}, \alpha)$ 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 $g \in C^1(\mathcal{I}, \mathbb{R}^d)$ [**?**]. From these conditions he proposed the following Strong Order 1.5 scheme known as SRA1:

$c^{(0)}$	$A^{(0)}$	$B^{(0)}$	
	α^T	${\beta^{(1)}}^T$	${\beta^{(2)}}^T$

0					
$\frac{3}{4}$	$\frac{3}{4}$		$\frac{1}{2}$		
	$\frac{1}{3}$	$\frac{2}{3}$	1	0	-1	1

9 Appendix II: Derivation Details

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

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

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

Thus we substitute in the Wiktorsson approximations

$$I_{(1,1)} = \frac{1}{2}\left(\Delta W^2 - h\right)$$
$$I_{(1,1,1)} = \frac{1}{6}\left(\Delta W^3 - 3h\Delta W\right)$$
$$I_{(1.0)} = \frac{1}{2}h\left(\Delta W + \frac{1}{\sqrt{3}}\Delta Z\right)$$

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

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