REX: A FAMILY OF REVERSIBLE EXPONENTIAL (STOCHASTIC) RUNGE-KUTTA SOLVERS

Zander W. Blasingame*
AITHYRA, Clarkson University
zblasingame@aithyra.at

Chen Liu
Clarkson University
cliu@clarkson.edu

ABSTRACT

Deep generative models based on neural differential equations have quickly become the state-of-the-art for numerous generation tasks across many different applications. These models rely on ODE/SDE solvers which integrate from a prior distribution to the data distribution. In many applications it is highly desirable to then integrate in the other direction. The standard solvers, however, accumulate discretization errors which don't align with the forward trajectory, thereby prohibiting an *exact inversion*. In applications where the precision of the generative model is paramount this inaccuracy in inversion is often unacceptable. Current approaches to solving the inversion of these models results in significant downstream issues with poor stability and low-order of convergence; moreover, they are strictly limited to the ODE domain. In this work, we propose a new family of reversible exponential (stochastic) Runge-Kutta solvers which we refer to as *Rex* developed by an application of Lawson methods to convert any explicit (stochastic) Runge-Kutta scheme into a reversible one. In addition to a rigorous theoretical analysis of the proposed solvers, we also empirically demonstrate the utility of *Rex* on improving the sampling of Boltzmann distributions with flow models, and improving image generation and editing capabilities with diffusion models.

1 INTRODUCTION

Deep generative models based on *neural differential equations* (Kidger, 2022) have quickly become the state-of-the-art in generation tasks across many varied modalities from image generation (Rombach et al., 2022), protein generation (Skreta et al., 2025), Boltzmann sampling (Rehman et al., 2026a), and biometrics (Blasingame & Liu, 2024d). These models use the language of a neural Itô *stochastic differential equation* (SDE) (Kidger et al., 2021) or neural *ordinary differential equation* (ODE) (Chen et al., 2018)—sometimes referred to as a *continuous normalizing flow* (CNF)—to describe the (stochastic) mapping from a prior source distribution to the target data distribution. This can be achieved through a variety of numerical schemes (Lu et al., 2022b; Zhang et al., 2023; Zhang & Chen, 2023; Gonzalez et al., 2024) which integrates from the source to target distribution. The *exact* inversion of this numerical method, *i.e.*, going back from the target distribution back to the source distribution, is invaluable in several key applications where precision is concerned. *E.g.*, gradient descent through these models (Ben-Hamu et al., 2024; Blasingame & Liu, 2024a; McCallum & Foster, 2024) for training, fine-tuning, and differentiable rewards; image editing (Wallace et al., 2023; Wang et al., 2024); and calculating accurate likelihoods of the generative models (Rehman et al., 2026a) enabling sampling from Boltzmann distributions.

Whilst useful, designing such inversion methods is very tricky, as such solvers are plagued by issues of low order of convergence, lack of stability, amongst other undesirable properties; moreover, it is *even* more difficult to construct such schemes for SDEs. Recent work has developed exact inversion methods specifically for diffusion models (Wallace et al., 2023; Zhang et al., 2024; Wang et al., 2024). Unfortunately, these schemes suffer from poor numerical stability which can hamper their real world utility, particularly in contexts like editing of real samples. Moreover, the previous approaches do not support the often useful SDE formulation of diffusion models.

*Work partially completed while a Ph.D. student at Clarkson University.

To address these challenges we propose *Rex*, a family of **r**eversible **ex**ponential (stochastic) Runge-Kutta solvers for diffusion models. Our contributions are:

- We show that *Rex* is the reversible version of many popular solvers for diffusion models.
- *Rex* is an *algebraically reversible* solver for diffusion SDEs.
- The *Rex* ODE solvers obtain an arbitrarily high order of convergence and have a non-zero region of stability.
- We empirically show that *Rex* outperforms previous methods developed for the *exact inversion* of diffusion models.
- We demonstrate that *Rex* improves Boltzmann sampling by ensuring invertibility.

2 Preliminaries

2.1 Reversible solvers for neural differential equations

Recently, researchers studying *neural differential equations* have begun to propose several *algebraically reversible solvers*. Consider some prototypical neural ODE of the form $\dot{\boldsymbol{x}}_t = \boldsymbol{u}_\theta(t, \boldsymbol{x}_t)$ with vector field $\boldsymbol{u}_\theta \in \mathcal{C}^r(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^d)$ which satisfies the usual regularity conditions. Then consider a single-step numerical scheme of the form $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \boldsymbol{\Phi}_h(t_n, \boldsymbol{x}_n, \boldsymbol{u}_\theta)$. Every numerical scheme $\boldsymbol{\Phi}$ is reversible in the sense that we can rewrite the forward step as an implicit scheme of the form $\boldsymbol{x}_n = \boldsymbol{x}_{n+1} - \boldsymbol{\Phi}_h(t_n, \boldsymbol{x}_n, \boldsymbol{u}_\theta)$; however, this requires fixed point iteration[1] and is both *approximate* and computationally *expensive*. This type of reversibility is known as *analytic reversibility* within the neural differential equations community (Kidger, 2022, Section 5.3.2.1). What we would prefer, however, is a form of reversibility that can be expressed in *closed-form*.

There are only a few such *algebraically reversible* solvers which have been proposed within the last few years (Zhuang et al., 2021; Kidger et al., 2021; McCallum & Foster, 2024) and only one of these works for SDEs, namely, the work of Kidger et al. (2021). Whilst only for ODEs the recent work of McCallum & Foster (2024) is highly interesting as it is the *only* algebraically reversible with a non-zero region of stability. We refer to the method proposed in McCallum & Foster (2024) as the *McCallum-Foster* method and summarize it below in Definition 2.1.

Definition 2.1. Initialize $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ and let $\zeta \in (0, 1]$. Consider a step size of h, then a forward step of the McCallum-Foster method is defined as

$$\boldsymbol{x}_{n+1} = \zeta \boldsymbol{x}_n + (1 - \zeta)\hat{\boldsymbol{x}}_n + \boldsymbol{\Phi}_h(t_n, \hat{\boldsymbol{x}}_n), \tag{1a}$$

$$\hat{\boldsymbol{x}}_{n+1} = \hat{\boldsymbol{x}}_n - \boldsymbol{\Phi}_{-h}(t_{n+1}, \boldsymbol{x}_{n+1}), \tag{1b}$$

and the backward step is given as

$$\hat{\boldsymbol{x}}_n = \hat{\boldsymbol{x}}_{n+1} + \boldsymbol{\Phi}_{-h}(t_{n+1}, \boldsymbol{x}_{n+1}), \tag{2a}$$

$$\boldsymbol{x}_n = \zeta^{-1}\boldsymbol{x}_{n+1} + (1 - \zeta^{-1})\hat{\boldsymbol{x}}_n - \zeta^{-1}\boldsymbol{\Phi}_h(t_n, \hat{\boldsymbol{x}}_n). \tag{2b}$$

2.2 Diffusion models

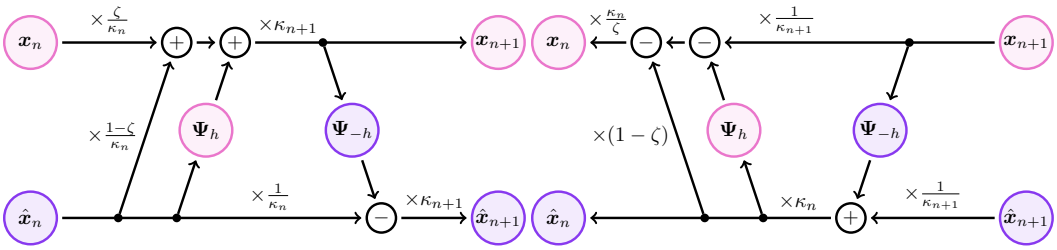
Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021a;b) have quickly become one of the most popular paradigms for constructing generative models. Consider the following Itô *stochastic differential equation* (SDE) defined on time interval $[0, T]$:

$$\mathrm{d}\boldsymbol{X}_t = f(t)\boldsymbol{X}_t \,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{W}_t, \tag{3}$$

where $f, g \in \mathcal{C}^\infty([0, T])$[2] form the drift and diffusion coefficients of the SDE and where $\{\boldsymbol{W}_t\}_{t\in[0,T]}$ is the standard Brownian motion on the time interval. The coefficients f, g are chosen such that the SDE maps clean samples from the data distribution $\boldsymbol{X}_0 \sim q(\boldsymbol{X})$ at time 0 to an isotropic Gaussian at time T. More specifically, for a *noise schedule* $\alpha_t, \sigma_t \in \mathcal{C}^\infty([0, T]; \mathbb{R}_{\geq 0})$ consisting of a strictly

[1] If the step size h is small enough.

[2] We let $\mathcal{C}^r(X; Y)$ denote the class of r-th differentiable functions from X to Y. If Y is omitted then $Y = \mathbb{R}$.



(a) Forward step (b) Backward step

Figure 1: An overview of the *Rex* solver. Here $\boldsymbol{\Psi}_h$ denotes the *Princeps* scheme exponential (stochastic) Runge-Kutta scheme (see Section 3.1), $\zeta \in (0, 1)$ is a coupling parameter, and $\{\kappa_n\}_{n=1}^N$ denotes the set of weighting variables derived from the exponential schemes. The particular values of κ_n are discussed in Proposition 3.1.

monotonically decreasing function α_t and strictly monotonically increasing function σ_t, the drift and diffusion coefficients are found to be

$$f(t) = \frac{\dot{\alpha}_t}{\alpha_t}, \qquad g^2(t) = \dot{\sigma}_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t}\sigma_t^2, \tag{4}$$

where with abuse of notation $\dot{\sigma}_t^2$ denotes the time derivative of the function σ_t^2 (Lu et al., 2022b; Kingma et al., 2021)—this ensures that $\boldsymbol{X}_t \sim \mathcal{N}(\alpha_t \boldsymbol{X}_0, \sigma_t^2 \boldsymbol{I})$. However, we wish to map from *noise* back to *data*, as such we employ the result of Anderson (1982) to construct the *reverse-time* diffusion SDE of Equation (16), which is found to be

$$\mathrm{d}\boldsymbol{X}_t = [f(t)\boldsymbol{X}_t - g^2(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{X}_t)]\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{\boldsymbol{W}}_t, \tag{5}$$

where $\mathrm{d}t$ is a *negative* timestep, $\{\overline{\boldsymbol{W}}_t\}_{t\in[0,T]}$ is the standard Brownian motion in reverse-time, and $p_t(\boldsymbol{x}) \coloneqq p(t, \boldsymbol{x})$ is the marginal density function. Then, if we can learn the *score function* $(t, \boldsymbol{x}) \mapsto \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ (Song et al., 2021b)—or some other *equivalent* reparameterization, *e.g.*, noise prediction (Song et al., 2021a; Ho et al., 2020) or data prediction (Kingma et al., 2021)—we can then draw samples from our data distribution $q(\boldsymbol{X})$ by first sampling some $\boldsymbol{X}_T \sim p(\boldsymbol{X})$ from the Gaussian prior and then employing a numerical SDE solver, *e.g.*, Euler-Maruyama, to solve Equation (18) in reverse-time.

3 Rex: A Family of Reversible Exponential (Stochastic) Runge-Kutta Solvers

In this section we introduce the *Rex* family of reversible exponential Runge-Kutta solvers. This family of *bespoke* numerical schemes is specifically tailored to exploit the semi-linear structure of the diffusion ODE/SDE. The elegance of *Rex* is that it can be built rather generally from many popular pre-existing ODE/SDE solvers. Let $\boldsymbol{\Phi}$ denote our explicit (S)RK scheme of choice, *e.g.*, Euler or the Dormand-Prince method. Then we massage the probability flow ODE and reverse-time SDE into a sufficiently *nice* form and apply $\boldsymbol{\Phi}$—we refer to this construction as *Princeps*, $\boldsymbol{\Psi}$. Lastly, we construct a reversible scheme from this nice form of $\boldsymbol{\Psi}$. We summarize this three step process below.

Rex recipe.

1. Select an explicit (S)RK scheme, $\boldsymbol{\Phi}$.
2. Build *Princeps* from the RK scheme, $\boldsymbol{\Psi}$.
3. Build *Rex* out of *Princeps*, $\boldsymbol{\Upsilon}$.

In Figure 1 we present an overview of the *Rex* computational graph. *N.B.*, the graph for both the ODE and SDE formulations are identical with the only difference being the weighting terms $\{\kappa_n\}$ and the underlying numerical scheme $\boldsymbol{\Psi}_h$. The rest of this section is organized as follows: first we construct the *Princeps* scheme $\boldsymbol{\Psi}$ from some explicit (S)RK scheme $\boldsymbol{\Phi}$ in Section 3.1, then we construct *Rex*, $\boldsymbol{\Upsilon}$, from *Princeps* in Section 3.2, and lastly in Section 3.3 we discuss the theoretical properties of *Rex*.

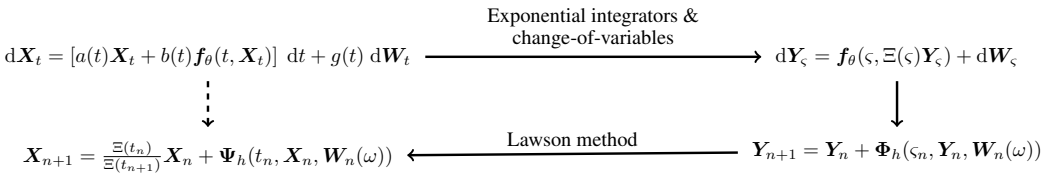


Figure 2: Overview of the construction of the *Princeps*, $\boldsymbol{\Psi}$, for the probability flow ODE from an underlying explicit stochastic Runge-Kutta scheme $\boldsymbol{\Phi}$ for the SDE in Equation (6). This graph holds for the ODE formulation *mutatis mutandis*.

3.1 Princeps

In this section we discuss how to build the underlying scheme, $\boldsymbol{\Psi}$, from which we construct the reversible *Rex* scheme. As this scheme is *one that is first* we hereafter refer to it as the *Princeps* scheme. For simplicity of presentation we derive *Princeps* from the reverse-time diffusion SDE in Equation (18); however, this framework generalizes the *probability flow* ODE formulation, see Equation (19), with the explicit derivations detailed in Appendix C.1.

As the data and noise prediction formulations differ we will derive *Princeps* from a rather general view of the problem and elide the particular details, reserving them for Appendix C.2. Recall that the Itô SDE in Equation (18) has a semi-linear drift term and additive noise, this nice structure allows us to greatly simplify our discussion. We will rewrite Equation (18) as

$$\mathrm{d}\boldsymbol{X}_t = [a(t)\boldsymbol{X}_t + b(t)\boldsymbol{f}_\theta(t, \boldsymbol{X}_t)]\ \mathrm{d}t + g(t)\ \mathrm{d}\overline{\boldsymbol{W}}_t, \tag{6}$$

where $a(t), b(t)$ are the appropriate generalizations for the data and noise parameterizations and $\boldsymbol{f}_\theta$ denotes either the noise or data prediction model (see Appendix C *cf*. Section 2.2). Let $\Xi(t) \coloneqq \exp \int_0^t a(\tau)\ \mathrm{d}\tau$ be the reciprocal of the integrating factor of Equation (6), which we can then rewrite as a time-changed SDE of the form

$$\mathrm{d}\boldsymbol{Y}_\varsigma = \boldsymbol{f}_\theta(\varsigma, \Xi(\varsigma)\boldsymbol{Y}_\varsigma)\ \mathrm{d}\varsigma + \mathrm{d}\boldsymbol{W}_{\varsigma^2}, \tag{7}$$

where $\boldsymbol{Y}_t = \Xi^{-1}(t)\boldsymbol{X}_t$ and $\varsigma_t = \int \Xi^{-1}(t)b(t)\ \mathrm{d}t$. Now we merely need to construct *Princeps* from a stochastic Runge-Kutta scheme for Equation (7). The question is, however, what stochastic Runge-Kutta formulation to use, as unlike in the ODE case there are many different formulations to choose from.

Stochastic Runge-Kutta. Constructing a numerical scheme for SDEs is greatly more complicated than ODEs due to the complexities of stochastic processes and in particular stochastic integrals. Unlike numerical schemes for ODEs which are usually built upon truncated Taylor expansions, SDEs require constructing truncated Itô or Stratonovich-Taylor expansions (Kloeden & Platen, 1991) which results in numerous iterated stochastic integrals. Approximating these iterated integrals, or equivalently Lévy areas, of Brownian motion is quite difficult (Clark & Cameron, 2005; Mrongowius & Rößler, 2022); however, SDEs with certain constraints on the diffusion term may use specialized solvers to further achieve a strong order of convergence with simple approximations of these iterated stochastic integrals. As such there are several ways to express SRK methods depending on the choice of approximating these iterated integrals. We choose to follow the work of Foster et al. (2024) which makes usage of the *space-time Lévy area* in constructing such methods. The space-time Lévy area (see Foster et al., 2020, Definition 3.5; *cf*. Rößler, 2010) is defined below in Definition 3.1.

Definition 3.1 (Space-time Lévy area)**.** The rescaled space-time Lévy area of a Brownian motion $\{W_t\}$ on the interval $[s, t]$ corresponds to the signed area of the associated bridge process

$$H_{s,t} \coloneqq \frac{1}{h}\int_s^t \left(W_{s,u} - \frac{u-s}{h}W_{s,t}\right)\ \mathrm{d}u, \tag{8}$$

where $h \coloneqq t - s$ and $W_{s,u} = W_u - W_s$ for $u \in [s, t]$.

In particular, for additive-noise SDEs which our SDE in Equation (7) is, the Itô and Stratonovich integrals coincide and the numerical scheme is significantly simpler, for more details we refer to Appendix B.

For the Itô SDE in Equation (7) we follow Foster et al. (2024) and write an s-stage SRK as

$$\boldsymbol{f}_\theta^i = \boldsymbol{f}_\theta(\varsigma_n + c_i h, \Xi(\varsigma_n + c_i h)\boldsymbol{Z}_i), \tag{9a}$$

$$\boldsymbol{Z}_i = \boldsymbol{Y}_n + h\left(\sum_{j=1}^{i-1} a_{ij}\boldsymbol{f}_\theta^j\right) + a_i^W \boldsymbol{W}_n + a_i^H \boldsymbol{H}_n, \tag{9b}$$

$$\boldsymbol{Y}_{n+1} = \boldsymbol{Y}_n + h\left(\sum_{i=1}^{s} b_i \boldsymbol{f}_\theta^i\right) + b^W \boldsymbol{W}_n + b^H \boldsymbol{H}_n, \tag{9c}$$

where $h = \varsigma_{n+1} - \varsigma_n$ is the step size and $\boldsymbol{W}_n := \boldsymbol{W}_{t_n,t_{n+1}}$ and $\boldsymbol{H}_n := \boldsymbol{H}_{t_n,t_{n+1}}$ are the Brownian and Lévy increments respectively; and where $a_{ij}, a_i^W, a_i^H \in \mathbb{R}^{s\times s}$, $b_i, b^W, b^H \in \mathbb{R}^s$, and $c_i \in \mathbb{R}^s$ for the coefficients for an *extended* Butcher tableau (Rößler, 2025; Foster et al., 2024). (*cf*. for the ODE case Stewart, 2022, Section 6.1.4). Then by straightforward substitution for the identity of $\boldsymbol{Y}$ we arrive at the *Princeps* scheme denoted $\boldsymbol{\Psi}$,

$$\boldsymbol{f}_\theta^i = \boldsymbol{f}_\theta(\varsigma_n + c_i h, \Xi(\varsigma_n + c_i h)\boldsymbol{Z}_i), \tag{10a}$$

$$\boldsymbol{Z}_i = \Xi^{-1}(\varsigma_n)\boldsymbol{X}_n + h\left(\sum_{j=1}^{i-1} a_{ij}\boldsymbol{f}_\theta^j\right) + a_i^W \boldsymbol{W}_n + a_i^H \boldsymbol{H}_n, \tag{10b}$$

$$\boldsymbol{X}_{n+1} = \frac{\Xi(\varsigma_{n+1})}{\Xi(\varsigma_n)}\boldsymbol{X}_n + \Xi(\varsigma_{n+1})\underbrace{\left[h\left(\sum_{i=1}^{s} b_i \boldsymbol{f}_\theta^i\right) + b^W \boldsymbol{W}_n + b^H \boldsymbol{H}_n\right]}_{:=\boldsymbol{\Psi}}, \tag{10c}$$

from which we will build the *Rex* scheme from.

3.2 Rex

Equipped with both Equation (10) we are now ready to construct Rex. The key idea is to construct a reversible scheme from an explicit (S)RK scheme (we provide more detail in Appendix B) for the reparameterized differential equation using the McCallum-Foster method and then apply Lawson methods to bring the scheme back to the original state variable, *cf*. Figure 2. Unfortunately, the full construction of *Rex* is quite long and does not fit within the main body of the paper so we provide a brief summary of it and encourage the reader to read the full scheme in Appendix C.3.

Proposition 3.1 (Rex). *Without loss of generality let $\boldsymbol{\Phi}$ denote an explicit SRK scheme for the SDE in Equation (7) with extended Butcher tableau $a_{ij}, b_i, c_i, a_i^W, a_i^H, b^W, b^H$. Fix an $\omega \in \Omega$ and let $\boldsymbol{W}$ be the Brownian motion over time variable ς. Then the reversible solver constructed from $\boldsymbol{\Phi}$ in terms of the underlying state variable $\boldsymbol{X}_t$ is given by the forward step*

$$\begin{aligned}\boldsymbol{X}_{n+1} &= \frac{\kappa_{n+1}}{\kappa_n}\left(\zeta\boldsymbol{X}_n + (1-\zeta)\hat{\boldsymbol{X}}_n\right) + \kappa_{n+1}\boldsymbol{\Psi}_h(\varsigma_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_n(\omega)),\\ \hat{\boldsymbol{X}}_{n+1} &= \frac{\kappa_{n+1}}{\kappa_n}\hat{\boldsymbol{X}}_n - \kappa_{n+1}\boldsymbol{\Psi}_{-h}(\varsigma_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_n(\omega)),\end{aligned} \tag{11}$$

and backward step

$$\begin{aligned}\hat{\boldsymbol{X}}_n &= \frac{\kappa_n}{\kappa_{n+1}}\hat{\boldsymbol{X}}_{n+1} + \kappa_n\boldsymbol{\Psi}_{-h}(\varsigma_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_n(\omega)),\\ \boldsymbol{X}_n &= \frac{\kappa_n}{\kappa_{n+1}}\zeta^{-1}\boldsymbol{X}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{X}}_n - \kappa_n\zeta^{-1}\boldsymbol{\Psi}_h(\varsigma_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_n(\omega)),\end{aligned} \tag{12}$$

with step size $h := \varsigma_{n+1} - \varsigma_n$ and where $\boldsymbol{\Psi}$ follows Equation (10) with $\kappa_n = \Xi(\varsigma_n)$.

Remark 3.2. We can recover all four cases with the appropriate choice of weighting coefficient and time variable. For data prediction SDEs we have, $\kappa_n = \frac{\sigma_n}{\gamma_n}$ and $\varsigma_t = \frac{\sigma_n^2}{\alpha_n^2}$; the ODE case is recovered for an explicit RK scheme with $\kappa_n = \sigma_n$ and $\varsigma_t = \frac{\alpha_n}{\sigma_n}$ For noise prediction models we have $\boldsymbol{f}_\theta$ denoting the noise prediction model with $\kappa_n = \alpha_n$ and $\varsigma_t = \frac{\sigma_n}{\alpha_n}$.

An important aspect of reversibility with SDEs is the key idea is to use the *same* realization of the Brownian motion in both the forward pass or backward pass. We discuss how to implement this in Appendix G.3 via the Brownian interval.

3.3 Properties of Rex

Convergence order. We show that *Rex* can achieve an arbitrarily high-order of convergence in Theorem 3.2 with the proof provided in Appendix D.2.

Theorem 3.2 (Rex is a k-th order solver)**.** *Let $\boldsymbol{\Phi}$ be a k-th order explicit Runge-Kutta scheme for the reparameterized probability flow ODE in Equation (80) with variance preserving noise schedule (α_t, σ_t). Then Rex constructed from $\boldsymbol{\Phi}$ is a k-th order solver,* i.e.*, given the reversible solution $\{\boldsymbol{x}_n, \hat{\boldsymbol{x}}_n\}_{n=1}^N$ and true solution $\boldsymbol{x}_{t_n}$ we have*

$$\|\boldsymbol{x}_n - \boldsymbol{x}_{t_n}\| \leq Ch^k, \tag{13}$$

for constants $C, h_{max} > 0$ and for step sizes $h \in [0, h_{max}]$.

We can show a similar result for *Princeps* with the full proof provided in Appendix D.3.

Theorem 3.3 (Convergence order for *Princeps*)**.** *Let $\boldsymbol{\Phi}$ be a SRK scheme with strong order of convergence $\xi > 0$ for the reparameterized reverse-time diffusion SDE in Equation (127) with variance preserving noise schedule (α_t, σ_t) and $\alpha_T > 0$. Then $\boldsymbol{\Psi}$ constructed from $\boldsymbol{\Phi}$ has strong order of convergence ξ.*

Relation to existing solvers. Next we show that several variants of *Rex* are actually the *reversible versions* of several well-known solvers in the literature for diffusion models, *e.g.*, the DPM-Solvers (Lu et al., 2022b). We state this result below in Theorem 3.4 with the full details and proofs in Appendix E.

Theorem 3.4 (*Princeps* subsumes previous solvers)**.** *Princeps subsumes the following solvers for diffusion models*

1. *DDIM (Song et al., 2021a),*
2. *DPM-Solver-1, DPM-Solver-2, DPM-Solver-12 (Lu et al., 2022b),*
3. *DPM-Solver++1, DPM-Solver++(2S), SDE-DPM-Solver-1, SDE-DPM-Solver++1 (Lu et al., 2022a),*
4. *SEEDS-1 (Gonzalez et al., 2024), and*
5. *gDDIM (Zhang et al., 2023).*

Corollary 3.4.1. *Rex is the reversible revision of the well-known solvers for diffusion models in Theorem 3.4.*

Stability. One drawback of reversible solvers is their rather unimpressive stability, in fact until the work of McCallum & Foster (2024) there were no reversible methods which had a non-zero region of stability. We discuss this more in detail Appendix A.3 along with illustrating the poor stability characteristics of BDIA and O-BELM (see Corollaries A.4.1 and A.3.2). However, since *Rex* is built upon the McCallum-Foster method the ODE solver has a non-zero region of stability.[3] This will prove valuable later in our empirical studies.

4 Empirical Results

In this section we conduct a number of empirical studies to illustrate the utility of *Rex* in a wide variety of contexts. We primarily explore this within in two contexts: 1) image generation and editing with exact inversion, and 2) ensuring invertibility for accurate likelihood calculations for Boltzmann sampling. Unless stated otherwise $\zeta = 0.999$ for all experiments.

[3] *I.e.*, in the sense of the linear test equation, see Appendix A.3 for more details.

Table 1: Quantitative comparison of different reversible solvers for unconditional image generation with a pre-trained DDPM model on CelebA-HQ (256×256) with the non-reversible DDIM as a baseline. Our reversible ODE solvers are in pink and reversible SDE solvers in mauve.

Steps	Solver	FD (↓)	FD$_\infty$ (↓)	Precision (↑)	Recall (↑)	Density (↑)	Coverage (↑)
10	EDICT	1042.89	1034.82	0.49	0.10	0.19	0.11
	BDIA†	900.95	894.23	0.61	0.10	0.28	0.14
	BDIA‡	1284.48	1274.46	0.41	0.00	0.14	0.05
	O-BELM	**605.52**	**596.47**	0.78	0.18	0.56	0.34
	Rex (Midpoint)	**607.20**	**597.04**	0.78	0.21	0.60	**0.37**
	Rex (RK4)	633.90	617.11	**0.81**	**0.22**	**0.64**	0.36
	Rex (Euler-Maruyama)	**610.16**	**598.56**	0.79	0.10	0.61	**0.37**
	DDIM	727.75	716.41	0.75	0.14	0.49	0.27
20	EDICT	752.68	743.89	0.68	0.15	0.36	0.21
	BDIA†	611.47	601.37	0.76	0.19	0.50	0.30
	BDIA‡	982.30	968.62	0.54	0.10	0.22	0.10
	O-BELM	489.94	477.82	0.82	0.23	0.71	0.43
	Rex (Midpoint)	539.96	527.85	0.81	0.26	0.66	0.41
	Rex (RK4)	547.24	533.30	0.82	**0.27**	0.71	0.43
	Rex (Euler-Maruyama)	**460.42**	**447.01**	**0.86**	0.21	**0.91**	**0.51**
	DDIM	570.11	555.26	0.79	0.20	0.62	0.38
50	EDICT	551.13	534.73	0.78	0.24	0.60	0.37
	BDIA†	500.79	489.24	0.82	0.27	0.70	0.44
	BDIA‡	798.47	790.17	0.71	0.12	0.39	0.18
	O-BELM	476.29	463.07	0.84	**0.29**	0.77	0.45
	Rex (Midpoint)	505.67	494.94	0.81	**0.29**	0.70	0.44
	Rex (RK4)	511.17	498.94	0.80	0.27	0.69	0.44
	Rex (Euler-Maruyama)	**391.93**	**381.01**	**0.87**	0.28	**0.98**	**0.56**
	DDIM	490.88	479.87	0.80	0.26	0.67	0.45

4.1 Image generation

Unconditional image generation. Following prior works (Wang et al., 2024; Wallace et al., 2023) we begin by exploring the ability of Rex to function as a traditionally solver for diffusion models. To evaluate this we drew 10^4 samples using a DDPM model (Ho et al., 2020) pre-trained on the CelebA-HQ (Karras et al., 2018) dataset with the various solvers each using the same fixed seed. We report the performance in terms of the *Fréchet inception distance* (FID) (Heusel et al., 2017) which calculates the Fréchet distance between the real and generate distributions within one of the layers of an Inception neural network (Szegedy et al., 2016). Following Stein et al. (2023), we report the *Fréchet distance* (FD) with DINOv2 (Oquab et al., 2023), along with additional metrics described in Appendix I.1.2. In Table 1 we compare pre-existing methods for exact inversion with diffusion models against Rex, along with including the non-reversible DDIM solver as a baseline. We observe that the Rex family of reversible solvers performs exceedingly well, surpassing the baseline non-reversible DDIM scheme, handily beating EDICT and BDIA, and often outperforming O-BELM. We observe that our reversible SDE scheme consistently performs quite well outside of the very few step-size regime (a well known limitation of SDE schemes). *N.B.*, that unlike the results reported for the other reversible solvers we did not search for the optimal hyperparameters for Rex for the sampling task. In Figure 3 we present a visual qualitative comparison of the different solvers using the same initial noise.

(a) DDIM (b) EDICT (c) BDIA (d) O-BELM (e) Rex

Figure 3: Qualitative comparison of unconditional sampling with different solvers with a pre-trained DDPM model on CelebA-HQ (256×256) with the non-reversible DDIM as a baseline. Each method used 10 discretization steps.

Conditional image generation. To further evaluate Rex we drew text-conditioned samples using Stable Diffusion v1.5 (Rombach et al., 2022) with a set of 1000 randomly selected captions from COCO (Lin et al., 2014) with the various solvers each using the same fixed seed. We report performance in terms of the state-of-the-art text-to-image scoring function PickScore (Kirstain

et al., 2023); and in terms of the state-of-the-art Image Reward metric (Xu et al., 2023) which assigns a score that reflects human preferences, namely, aesthetic quality and prompt adherence. In Table 2 we compare pre-existing methods for exact inversion with diffusion models against Rex, along with including the non-reversible DDIM solver as a baseline. We observe that Rex does very well compared to other reversible solvers, and in particular the stochastic variants of Rex perform *extremely* well. In Appendix J.1 we present Figure 6, a visual qualitative comparison of the different solvers using the same initial noise. We provide additional experimental details in Appendix I.2.

Table 2: Quantitative comparison of different reversible solvers in terms of average Image Reward and PickScore for conditional text-to-image generation with Stable Diffusion v1.5 (512×512) with the non-reversible DDIM as a baseline. Our reversible ODE solvers are in pink and reversible SDE solvers in mauve. Best results in **bold**, second best under-lined.

	Image Reward (↑)			**PickScore** (↑)		
Solver / Steps	10	20	50	10	20	50
DDIM	0.033	0.136	0.247	21.06	21.29	21.04
EDICT	-1.219	-0.134	-0.055	19.52	20.84	21.05
BDIA	-0.006	0.055	0.066	20.98	21.16	21.21
O-BELM	0.051	0.105	0.160	20.88	21.00	21.16
Rex (Midpoint)	0.119	0.179	0.198	21.28	21.38	21.41
Rex (RK4)	0.156	0.187	0.195	21.35	21.40	21.41
Rex (Euler-Maruyama)	0.222	0.239	**0.264**	**21.50**	**21.66**	21.70
Rex (ShARK)	**0.239**	**0.249**	**0.263**	**21.51**	**21.66**	**21.72**

4.2 Image Editing

To test the round-trip image editing capabilities we follow Brooks et al. (2022) and use the `pix2pix` dataset which was designed for editing. The curated dataset consists of image and text caption pairs. The text captions include a description of the image, *e.g.*, "a girl riding a horse" and an instruction, *e.g.*, "have her ride a dragon". Our experimental is nearly identical to the last section except we also evaluate the LPIPS (Zhang et al., 2018) to measure the difference between the original and edited image for perceptual similarity. For all the fixed step-size solvers we choose 100 steps to help give the less stable schemes a leg up. We noticed that both the fixed step-size and adaptive step-size *Rex* schemes performed very well even outperforming the baseline non-reversible DDIM. *N.B.*, we omitted EDICT from the table because the model completely *failed* to edit and only learned the identity map.

4.3 Boltzmann Sampling

We evaluate the usefulness of *Rex* on equilibrium conformation sampling of tri-alanine. In particular, we are interested in drawing samples from a target Boltzmann distribution p_{target} defined on $\mathbb{R}^d$ as

$$p_{\text{target}}(\boldsymbol{x}) \propto \exp\left(-\mathcal{E}(\boldsymbol{x})\right), \quad (14)$$

where $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}$ is the energy of the system which can be efficiently computed for any $\boldsymbol{x}$. The Boltzmann is notoriously difficult to sample with classical simulation-based techniques such as molecular dynamics, rather recent work has turned to using deep generative models equipped with exact likelihoods trained on a small biased dataset. This biased model can then be corrected using self-normalized importance sampling (Liu & Liu, 2001). Chen et al. (2018) showed that the exact likelihood of a neural ODE with learnt vector field $\boldsymbol{u}_t^\theta$ can be found as the solution to the following augmented ODE

Table 3: Quantitative comparison of different reversible solvers image editing with the non-reversible DDIM as a baseline. Our reversible ODE solvers are in pink and adaptive ODE solvers are in teal. Best results in **bold**, second best under-lined.

Solver	**Image Reward** (↑)	**CLIP score** (↑)	**PickScore** (↑)	**LPIPS** (↓)
DDIM	-0.564	**19.17**	18.367	0.214
BDIA	-2.205	18.57	16.956	0.885
O-BELM	-0.639	19.16	18.416	0.140
Rex (Euler)	-0.551	**19.17**	**18.721**	0.109
Rex (Dopri5)	**-0.547**	19.16	18.698	**0.107**

$$\begin{bmatrix} \boldsymbol{x}_t \\ \log p_t^\theta(\boldsymbol{x}_t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_0 \\ \log p_0^\theta(\boldsymbol{x}_0) \end{bmatrix} + \int_0^t \begin{bmatrix} \boldsymbol{u}_s^\theta(\boldsymbol{x}_s) \\ -\left\langle \nabla_{\boldsymbol{x}}, \boldsymbol{u}_s^\theta(\boldsymbol{x}_s) \right\rangle \end{bmatrix} \mathrm{d}s. \quad (15)$$

In practice, however, one has to use a discretized numerical scheme $\boldsymbol{\Phi}$, yet its inverse $\boldsymbol{\Phi}^{-1}$ may not exist! This means that the change-of-variables used to find the probabilities may no longer be valid, introducing errors—as discussed in Rehman et al. (2026a) this can poses a significant problem if not

adequately addressed. *Rex* Υ, however, being reversible ensures that Υ^{-1} exists, thereby ensuring proper probabilities (up to a discretization error).

Baselines. We compare against a broad variety of standard baselines which includes equivariant CNFs (Klein et al., 2023). In particular, we compare against an improved version, dubbed ECNF++, proposed by Tan et al. (2025) which several modifications across flow matching loss; deeper architecture; improved optimizer and learning rate schedule; and exponential moving average. We additionally compare to discrete normalizing flows in RegFlow (Rehman et al., 2026b) and the state-of-the-art *Sequential Boltzmann Generator* (SBG) (Tan et al., 2025). Following, Rehman et al. (2026a) we train our own *diffusion transformer* (DiT) (Peebles & Xie, 2023) on tri-alanine which represents our last baseline. All the CNFs used the Dormand-Prince method (Dormand & Prince, 1980) which is a 5th order Runge-Kutta method with an embedded 4th order method for adaptive step sizing, the Butcher tableau is from Shampine (1986), and we use atol = rtol = 10^{-5}.

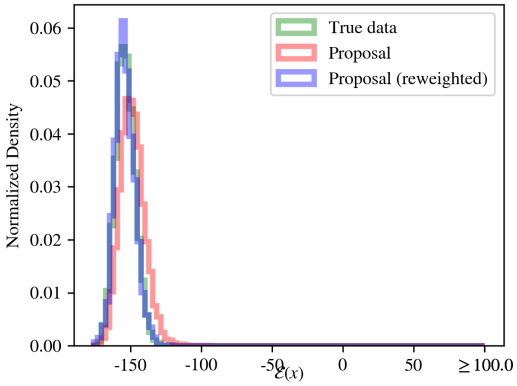


Figure 4: True MD energy distribution with unweighted and reweighted proposals created using Rex(Dopri5).

Results. In Table 4 we report the results of the sampling from the Boltzmann distribution in terms of the *effective sample size* (ESS) and the 2-Wasserstein distance between both the energy distributions ($\mathcal{E}$-$\mathcal{W}_2$) and dihedral angles ($\mathbb{T}$-$\mathcal{W}_2$)—further info on these metrics is provided in Appendix I.5.3. We see that applying the reversible *Rex* (Dopri5) to the DiT *vastly* improves the $\mathcal{E}$-$\mathcal{W}_2$ metric whilst only decreasing the other two metrics by a very small amount, still keeping them within the state-of-the-art. We note that for this experiment we choose $\zeta = 0.001$ as we don't need to properly invert the solve, but rather guarantee that the scheme is invertible, thus we are free to optimize for stability (*cf*. Appendix A.3).

Table 4: Quantitative results on tri-alanine. Evaluations are conducted over 10^4 samples. Best results in **bold**, second best under-lined.

Model	Numerical scheme	ESS (↑)	$\mathcal{E}$-$\mathcal{W}_2$ (↓)	$\mathbb{T}$-$\mathcal{W}_2$ (↓)
RegFlow	-	0.029	1.051	1.612
SBG (IS)	-	0.052	0.758	0.502
SBG (SMC)	-	-	0.598	0.503
ECNF++	Dopri5	0.003	2.206	0.962
DiT	Dopri5	**0.140**	0.737	**0.468**
DiT	Rex (Dopri5)	0.104	**0.495**	0.497

5 Conclusion

We propose *Rex* a family of reversible exponential (stochastic) Runge-Kutta solvers for diffusion models which can obtain arbitrarily a high order of convergence (for the ODE case). This scheme also naturally admits reversible adaptive step-size solvers enabling key ai4science applications. Moreover, we propose (to the best of our knowledge) the first method for the exact inversion for diffusion SDEs without storing the entire trajectory of the Brownian motion. We also showed that *Princeps* subsumes several previous and popular solvers showing that *Rex* is the reversible version of these beloved schemes. Our empirical studies show that not only does Rex have nice theoretical properties but it also functions as a capable and robust numerical scheme for doing interesting things with diffusion models. The proposed method can be incorporated into pre-existing applications wherein preserving the bijections of flow maps is important, leading to many exciting possible applications.

References

Iyabo Ann Adamu. *Numerical approximation of SDEs & the stochastic Swift-Hohenberg equation*. Ph.d. thesis, Heriot-Watt University, 2011. URL https://www.ros.hw.ac.uk/

bitstream/handle/10399/2460/AdamuIA_0711_macs.pdf.

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=SE20BFqj6J.

Zander W. Blasingame and Chen Liu. AdjointDEIS: Efficient gradients for diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=fAlcxvrOEX.

Zander W. Blasingame and Chen Liu. Fast-dim: Towards fast diffusion morphs. *IEEE Security & Privacy*, 2024b.

Zander W. Blasingame and Chen Liu. Greedy-dim: Greedy algorithms for unreasonably effective face morphs. In *2024 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–11, 2024c. doi: 10.1109/IJCB62174.2024.10744517.

Zander W. Blasingame and Chen Liu. Leveraging diffusion for strong and high quality face morphing attacks. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 6(1):118–131, 2024d.

Zander W. Blasingame and Chen Liu. Greed is good: A unifying perspective on guided generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=s14pdQgoLb.

Paul Bourgade. Stochastic analysis, 2010. URL https://cims.nyu.edu/~bourgade/SA2010/StochasticAnalysis.pdf?utm_source=chatgpt.com.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022.

Kevin Burrage and Pamela M Burrage. Order conditions of stochastic runge–kutta methods by b-series. *SIAM Journal on Numerical Analysis*, 38(5):1626–1646, 2000.

John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016. Third Edition.

Girolamo Cardano. *Artis Magnæ, Sive de Regulis Algebraicis, Lib. unus.* 1545.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find them. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6070–6079, 2020.

Koen Claessen and Michał H Pałka. Splittable pseudorandom number generators using cryptographic hashing. *ACM SIGPLAN Notices*, 48(12):47–58, 2013.

John MC Clark and RJ Cameron. The maximum rate of convergence of discrete approximations for stochastic differential equations. In *Stochastic Differential Systems Filtering and Control: Proceedings of the IFIP-WG 7/1 Working Conference Vilnius, Lithuania, USSR, Aug. 28–Sept. 2, 1978*, pp. 162–171. Springer, 2005.

M Crouzeix and FJ Lisbona. The convergence of variable-stepsize, variable-formula, multistep methods. *SIAM journal on numerical analysis*, 21(3):512–534, 1984.

Kristian Debrabant, Anne Kværnø, and Nicky Cordua Mattsson. Runge–kutta lawson schemes for stochastic differential equations. *BIT Numerical Mathematics*, 61(2):381–409, 2021.

Lisa DeBruine and Benedict Jones. Face Research Lab London Set, 5 2017. URL https://figshare.com/articles/dataset/Face_Research_Lab_London_Set/5047666.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation, 2015. URL https://arxiv.org/abs/1410.8516.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xQBRrtQM8u.

J. R. Dormand and P. J. Prince. A family of embedded Runge–Kutta formulae. *J. Comp. Appl. Math*, 6:19–26, 1980.

DC Dowson and BV666017 Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

Lester E Dubins and Gideon Schwarz. On continuous martingales. *Proceedings of the National Academy of Sciences*, 53(5):913–916, 1965.

Kang Feng. On difference schemes and symplectic geometry. In *Proceedings of the 5th international symposium on differential geometry and differential equations*, 1984.

James Foster, Terry Lyons, and Harald Oberhauser. An optimal polynomial approximation of brownian motion. *SIAM Journal on Numerical Analysis*, 58(3):1393–1421, 2020.

James M Foster. *Numerical approximations for stochastic differential equations*. Ph.d. thesis, University of Oxford, 2020. URL https://ora.ox.ac.uk/objects/uuid:775fc3f5-501c-425f-8b43-fc5a7b2e4310.

James M Foster, Goncalo Dos Reis, and Calum Strange. High order splitting methods for sdes satisfying a commutativity condition. *SIAM Journal on Numerical Analysis*, 62(1):500–532, 2024.

Peter K Friz and Martin Hairer. *A course on rough paths*. Springer, 2020.

Jessica G Gaines and Terry J Lyons. Variable step size control in the numerical solution of stochastic differential equations. *SIAM Journal on Applied Mathematics*, 57(5):1455–1484, 1997.

Martin Gonzalez, Nelson Fernandez Pinto, Thuy Tran, Hatem Hajri, Nader Masmoudi, et al. Seeds: Exponential sde solvers for fast high-quality sampling from diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9307–9315, 2024.

Andraž Jelinčič, James Foster, and Patrick Kidger. Single-seed generation of brownian paths and integrals for adaptive and high order sde solvers. *arXiv preprint arXiv:2405.06464*, 2024.

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast with score-based generative models. In *The Symbiosis of Deep Learning and Differential Equations*, 2021. URL https://openreview.net/forum?id=gEoVDSASC2h.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk99zCeAb.

Patrick Kidger. *On Neural Differential Equations*. Ph.d. thesis, Oxford University, 2022. Available at https://arxiv.org/abs/2202.02435.

Patrick Kidger, James Foster, Xuechen Chen Li, and Terry Lyons. Efficient and accurate gradients for neural sdes. *Advances in Neural Information Processing Systems*, 34:18747–18761, 2021.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=G5RwHpBUv0.

Leslie Kish. Confidence intervals for clustered samples. *American Sociological Review*, 22(2): 154–165, 1957.

Leon Klein and Frank Noé. Transferable boltzmann generators, 2025. URL https://arxiv.org/abs/2406.14426.

Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36:59886–59910, 2023.

Peter E Kloeden and Eckhard Platen. Stratonovich and itô stochastic taylor expansions. *Mathematische Nachrichten*, 151(1):33–50, 1991.

Peter E. Kloeden and Eckhard Platen. *Stochastic Differential Equations*, pp. 103–160. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992. ISBN 978-3-662-12616-5. doi: 10.1007/978-3-662-12616-5_4. URL https://doi.org/10.1007/978-3-662-12616-5_4.

Kei Kobayashi. Stochastic calculus for a time-changed semimartingale and the associated stochastic differential equations. *Journal of Theoretical Probability*, 24(3):789–820, 2011.

Yoshio Komori, David Cohen, and Kevin Burrage. Weak second order explicit exponential runge–kutta methods for stochastic differential equations. *SIAM Journal on Scientific Computing*, 39(6): A2857–A2878, 2017.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of imagenet classes in fréchet inception distance. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=4oXTQ6m_ws8.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3870–3882. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/li20i.html.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 5404–5411, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Jun S Liu and Jun S Liu. *Monte Carlo strategies in scientific computing*, volume 10. Springer, 2001.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

George Lowther. Time-changed brownian motion, 2010. URL https://almostsuremath.com/2010/04/20/time-changed-brownian-motion/.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022a.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=2uAaGwlP_V.

Terry J Lyons. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*, 14(2):215–310, 1998.

Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker–planck equations through gradient–log–density estimation. *Entropy*, 22(8):802, 2020.

Takashi Matsubara, Yuto Miyatake, and Takaharu Yaguchi. Symplectic adjoint method for exact gradient of neural ode with minimal memory. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 20772–20784. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/adf8d7f8c53c8688e63a02bfb3055497-Paper.pdf.

Sam McCallum and James Foster. Efficient, accurate and stable gradients for neural odes. *arXiv preprint arXiv:2410.11648*, 2024.

Jan Mrongowius and Andreas Rößler. On the approximation and simulation of iterated stochastic integrals and the corresponding lévy areas in terms of a multidimensional brownian motion. *Stochastic Analysis and Applications*, 40(3):397–425, 2022.

Ulrich Mutze. An asynchronous leapfrog method ii. *arXiv preprint arXiv:1311.6602*, 2013.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International conference on machine learning*, pp. 7176–7185. PMLR, 2020.

Shen Nie, Hanzhong Allan Guo, Cheng Lu, Yuhao Zhou, Chenyu Zheng, and Chongxuan Li. The blessing of randomness: SDE beats ODE in general diffusion-based image editing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=DesYwmUG00.

Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Universitext. Springer Berlin Heidelberg, Berlin, Germany, jul 2003. ISBN 9783662036204. doi: 10.1007/978-3-642-14394-6.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Jiachun Pan, Hanshu Yan, Jun Hao Liew, Jiashi Feng, and Vincent YF Tan. Towards accurate guided diffusion sampling through symplectic adjoint method. *arXiv preprint arXiv:2312.12030*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Martin Redmann and Sebastian Riedel. Runge-kutta methods for rough differential equations. *arXiv preprint arXiv:2003.12626*, 2020.

Danyal Rehman, Tara Akhound-Sadegh, Artem Gazizov, Yoshua Bengio, and Alexander Tong. FALCON: Few-step accurate likelihoods for continuous flows. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=FbssShlI4N.

Danyal Rehman, Oscar Davis, Jiarui Lu, Jian Tang, Michael Bronstein, Yoshua Bengio, Alexander Tong, and Avishek Joey Bose. Efficient regression-based training of normalizing flows for boltzmann generators. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=ctdnzPxDI3.

Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*, volume 293. Springer Science & Business Media, 2013.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

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

Andreas Rößler. A class of stochastic runge-kutta methods for stochastic differential equations converging with order 1 in L^p-norm. *arXiv preprint arXiv:2506.22657*, 2025.

W Rüemelin. Numerical treatment of stochastic differential equations. *SIAM Journal on Numerical Analysis*, 19(3):604–613, 1982.

Ronald D Ruth. A canonical integration technique. *IEEE Trans. Nucl. Sci.*, 30(CERN-LEP-TH-83-14):2669–2671, 1983.

Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *Advances in neural information processing systems*, 31, 2018.

John K Salmon, Mark A Moraes, Ron O Dror, and David E Shaw. Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of 2011 international conference for high performance computing, networking, storage and analysis*, pp. 1–12, 2011.

L. F. Shampine. Stability of the leapfrog/midpoint method. *Applied Mathematics and Computation*, 208(1):293–298, 2009.

Lawrence F. Shampine. Some practical Runge-Kutta formulas. *Mathematics of Computation*, 46 (173):135–150, 1986. doi: https://doi.org/10.2307/2008219.

Daniil Shmelev and Cristopher Salvi. Explicit and effectively symmetric schemes for neural sdes. *arXiv preprint arXiv:2509.20599*, 2025.

Daniil Shmelev, Kurusch Ebrahimi-Fard, Nikolas Tapia, and Cristopher Salvi. Explicit and effectively symmetric runge-kutta methods. *arXiv preprint arXiv:2507.21006*, 2025.

Marta Skreta, Lazar Atanackovic, Joey Bose, Alexander Tong, and Kirill Neklyudov. The superposition of diffusion models using the itô density estimator. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2o58Mbqkd2.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 2256–2265. JMLR.org, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/song23a.html.

George Stein, Jesse Cresswell, Rasa Hosseinzadeh, Yi Sui, Brendan Ross, Valentin Villecroze, Zhaoyan Liu, Anthony L Caterini, Eric Taylor, and Gabriel Loaiza-Ganem. Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

David E Stewart. *Numerical analysis: A graduate course*, volume 258. Springer, 2022.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Charlie B Tan, Avishek Joey Bose, Chen Lin, Leon Klein, Michael M Bronstein, and Alexander Tong. Scalable equilibrium sampling with sequential boltzmann generators. *arXiv preprint arXiv:2502.18462*, 2025.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

René J. De Vogelaere. Methods of integration which preserve the contact transformation property of the hamilton equations. Report NO. 4, University of Notre Dame, 1956.

Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22532–22541, 2023.

Fangyikang Wang, Hubery Yin, Yue-Jiang Dong, Huminhao Zhu, Chao Zhang, Hanbin Zhao, Hui Qian, and Chen Li. BELM: Bidirectional explicit linear multi-step sampler for exact inversion in diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ccQ4fmwLDb.

Chen Henry Wu and Fernando De la Torre. A latent space of stochastic diffusion models for zero-shot image editing and guidance. In *ICCV*, 2023.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: learning and evaluating human preferences for text-to-image generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 15903–15935, 2023.

Guoqiang Zhang, J. P. Lewis, and W. Bastiaan Kleijn. Exact diffusion inversion via bidirectional integration approximation. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LVII*, pp. 19–36, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-72997-3. doi: 10.1007/978-3-031-72998-0_2. URL https://doi.org/10.1007/978-3-031-72998-0_2.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Loek7hfb46P.

Qinsheng Zhang, Molei Tao, and Yongxin Chen. gDDIM: Generalized denoising diffusion implicit models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1hKE9qjvz-.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Juntang Zhuang, Nicha C Dvornek, sekhar tatikonda, and James s Duncan. MALI: A memory efficient and reverse accurate integrator for neural ODEs. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=blfSjHeFM_e.

Appendices

A Detailed discussion on related works . . . 20

A.1 Diffusion models . . . 20

A.2 Reversible solvers . . . 20

Asynchronous leapfrog method • Reversible Heun method • McCallum-Foster method • Explicit and effectively symmetric schemes

A.3 A note on stability . . . 23

A.4 Exact inversion of diffusion models . . . 24

EDICT sampler • BDIA sampler • BELM sampler • CycleDiffusion • Summary

A.5 SDE solvers for diffusion models . . . 27

Comparison with SEEDS

B Stochastic Runge-Kutta methods . . . 29

B.1 Foster-Reis-Strange SRK Scheme . . . 30

B.2 Independence of the Brownian and Lévy increments . . . 30

B.3 ShARK . . . 30

C Derivation of Rex . . . 31

C.1 Rex (ODE) . . . 31

Proof of Proposition C.1 • Data prediction • Noise prediction

C.2 Rex (SDE) . . . 36

Time-changed Brownian motion • Proof of reparametrized SDE for data prediction models • Proof of reparametrized SDE for noise prediction models • Derivation of Rex (SDE)

C.3 Proof of Proposition 3.1 . . . 44

D Convergence order proofs . . . 45

D.1 Assumptions . . . 45

D.2 Proof of Theorem 3.2 . . . 45

D.3 Proof of Theorem 3.3 . . . 46

E Relation to other solvers for diffusion models . . . 47

E.1 Rex as reversible ODE solvers . . . 48

Euler • Second-order methods • Third-order methods

E.2 Rex as reversible SDE solvers . . . 52

Euler-Maruyama

E.3 Rex as reversible SEEDS-1 . . . 54

F A brief note on the theory of rough paths . . . 55

G Implementation details . . . 55

G.1 Closed form expressions of the noise schedule . . . 55

Variance Preserving SDEs • OT flow matching

G.2 Some other inverse functions . . . 58

G.3 Numerical simulation of Brownian motion . . . 60

Methods

G.4 Implementation . . . 61

H Code . . . 62

I *Experimental details* 65
I.1 *Unconditional image generation* 65
Diffusion model • *Metrics* • *Hyperparameters*
I.2 *Conditional image generation* 66
Diffusion model • *Metrics* • *Hyperparameters*
I.3 *Interpolation* 67
I.4 *Image editing* 67
I.5 *Boltzmann sampling* 67
Datasets • *Training details* • *Metrics*
I.6 *Hardware* 69
I.7 *Repositories* 69
J *Additional results* 69
J.1 *Conditional image generation* 69
J.2 *Interpolation* 69

OVERVIEW OF THEORETICAL RESULTS

3.1 Proposition (Rex) 5
3.2 Theorem (Rex is a k-th order solver) 6
3.3 Theorem (Convergence order for *Princeps*) 6
3.4 Theorem (*Princeps* subsumes previous solvers) 6
A.1 Theorem (Convergence order of the McCallum-Foster method) 22
A.2 Theorem (Region of stability for the McCallum-Foster method) 24
A.3 Proposition (BDIA is the leapfrog/midpoint method) 25
A.3.1 Corollary (BDIA is a first-order method) 25
A.3.2 Corollary (BDIA is nowhere linearly stable) 26
A.4 Theorem (O-BELM is the leapfrog/midpoint method) 27
A.4.1 Corollary (O-BELM is nowhere linearly stable) 27
C.2 Lemma (Rex (ODE) for data prediction models) 33
C.3 Lemma (Rex (ODE) for noise prediction models) 35
C.4 Theorem (Dambis-Dubins-Schwarz representation theorem) 36
C.5 Theorem (Multi-dimensional Dambis-Dubins-Schwarz representation theorem) 37
C.8 Lemma (Time reparametrization of the reverse-time diffusion SDE) 39
C.9 Lemma (Time reparametrization of the reverse-time diffusion SDE for noise prediction models) 40
C.10 Lemma (Rex (SDE) for data prediction models) 42
C.11 Lemma (Rex (SDE) for noise prediction models) 43
C.12 Proposition (Rex) 44
3.2 Theorem (Rex is a k-th order solver) 45

3.3 Theorem (Convergence order for *Princeps*) 46
3.4 Theorem (*Princeps* subsumes previous solvers) 47
E.1 Proposition (Rex (Euler) is reversible DPM-Solver++1) 48
E.1.1 Corollary (Rex (Euler) is reversible deterministic DDIM for data prediction models) 49
E.2 Proposition (Rex (Euler) is reversible DPM-Solver-1) 49
E.2.1 Corollary (Rex (Euler) is reversible deterministic DDIM for noise prediction models) 50
E.3 Proposition (Rex (generic second-order) is reversible DPM-Solver++(2S)) . . . 50
E.4 Proposition (Rex (generic second-order) is reversible DPM-Solver-2)) 51
E.5 Proposition (Rex (Euler-Midpoint) is DPM-Solver-12) 51
E.6 Proposition (Rex (Euler-Maruyama) is reversible SDE-DPM-Solver++1) 52
E.6.1 Corollary (Rex (Euler-Maruyama) is reversible stochastic DDIM) 53
E.7 Proposition (Rex (Euler-Maruyama) is reversible SDE-DPM-Solver-1) 53
E.7.1 Corollary (Rex (Euler-Maruyama) is reversible stochastic DDIM for noise prediction models) 54
E.8 Proposition (Rex is reversible SEEDS-1) 54
E.8.1 Corollary (Rex (Euler-Maruyama) is reversible gDDIM) 54
G.1 Proposition (Inverse function of γ_t for linear noise schedule) 56
G.1.1 Corollary (Inverse function of χ_t for linear noise schedule) 56
G.1.2 Corollary (Inverse function of ϱ_t for linear noise schedule) 56
G.2 Proposition (Inverse function of γ_t for scaled linear noise schedule) 57
G.2.1 Corollary (Inverse function of χ_t for scaled linear noise schedule) 58
G.2.2 Corollary (Inverse function of ϱ_t for scaled linear noise schedule) 58
G.3 Proposition (Inverse function of γ_t in OT flow matching) 58
G.3.1 Corollary (Inverse function of χ_t in OT flow matching) 58

A DETAILED DISCUSSION ON RELATED WORKS

In this section we provide a detailed comparison with relevant related works. We begin in Appendix A.1 by discussing diffusion models before discussing algebraically reversible solvers in Appendix A.2. Then in Appendix A.3 we introduce the stability of an ODE solver, a helpful tool in comparing reversible solvers. Using this tool along with examining the convergence order we compare a variety of reversible solvers for diffusion models in Appendix A.4. Lastly, in Appendix A.5 we explore related work on constructing SDE solvers for diffusion models.

A.1 DIFFUSION MODELS

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021a;b) have quickly become one of the most popular paradigms for constructing generative models. Consider the following Itô *stochastic differential equation* (SDE) defined on time interval $[0, T]$:

$$\mathrm{d}\boldsymbol{X}_t = f(t)\boldsymbol{X}_t \,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{W}_t, \tag{16}$$

where $f, g \in \mathcal{C}^\infty([0, T])$[4] form the drift and diffusion coefficients of the SDE and where $\{\boldsymbol{W}_t\}_{t\in[0,T]}$ is the standard Brownian motion on the time interval. The coefficients f, g are chosen such that the SDE maps clean samples from the data distribution $\boldsymbol{X}_0 \sim q(\boldsymbol{X})$ at time 0 to an isotropic Gaussian at time T. More specifically, for a *noise schedule* $\alpha_t, \sigma_t \in \mathcal{C}^\infty([0, T]; \mathbb{R}_{\geq 0})$ consisting of a strictly monotonically decreasing function α_t and strictly monotonically increasing function σ_t, the drift and diffusion coefficients are found to be

$$f(t) = \frac{\dot{\alpha}_t}{\alpha_t}, \qquad g^2(t) = \dot{\sigma}_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t}\sigma_t^2, \tag{17}$$

where with abuse of notation $\dot{\sigma}_t^2$ denotes the time derivative of the function σ_t^2 (Lu et al., 2022b; Kingma et al., 2021)—this ensures that $\boldsymbol{X}_t \sim \mathcal{N}(\alpha_t \boldsymbol{X}_0, \sigma_t^2 \boldsymbol{I})$. However, we wish to map from *noise* back to *data*, as such we employ the result of Anderson (1982) to construct the *reverse-time* diffusion SDE of Equation (16), which is found to be

$$\mathrm{d}\boldsymbol{X}_t = [f(t)\boldsymbol{X}_t - g^2(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{X}_t)]\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{\boldsymbol{W}}_t, \tag{18}$$

where $\mathrm{d}t$ is a *negative* timestep, $\{\overline{\boldsymbol{W}}_t\}_{t\in[0,T]}$ is the standard Brownian motion in reverse-time, and $p_t(\boldsymbol{x}) := p(t, \boldsymbol{x})$ is the marginal density function. Then, if we can learn the *score function* $(t, \boldsymbol{x}) \mapsto \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ (Song et al., 2021b)—or some other *equivalent* reparameterization, *e.g.*, noise prediction (Song et al., 2021a; Ho et al., 2020) or data prediction (Kingma et al., 2021)—we can then draw samples from our data distribution $q(\boldsymbol{X})$ by first sampling some $\boldsymbol{X}_T \sim p(\boldsymbol{X})$ from the Gaussian prior and then employing a numerical SDE solver, *e.g.*, Euler-Maruyama, to solve Equation (18) in reverse-time. Notably, through careful massaging of the Fokker-Planck-Kolomogorov equation for the marginal density, one can construct an ODE which is equivalent in *distribution* to Equation (18) (Song et al., 2021b; Maoutsa et al., 2020), yielding the *highly* popular *probability flow ODE*

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = f(t)\boldsymbol{x}_t - \frac{g^2(t)}{2}\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t). \tag{19}$$

A.2 REVERSIBLE SOLVERS

The earliest work on reversible solvers can be traced back to the pioneering work on symplectic integrators by Vogelaere (1956); Ruth (1983); Feng (1984). Due to symplectic integrators being developed for solving Hamiltonian systems they are intrinsically reversible by construction (Greydanus et al., 2019). More recently, Matsubara et al. (2021) explored the use of symplectic solvers for solving the continuous adjoint equations. Likewise, work by Pan et al. (2023) extended this idea, making use of symplectic solvers for solving the continous adjoint equations for diffusion models. However, in this section we will focus on non-symplectic reversible solvers.

Throughout this section we consider solving the following d-dimensional IVP:

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)), \tag{20}$$

over the time interval $[0, T]$ with numerical solution $\{\boldsymbol{x}_n\}_{n=0}^N$.

[4] We let $\mathcal{C}^r(X; Y)$ denote the class of r-th differentiable functions from X to Y. If Y is omitted then $Y = \mathbb{R}$.

A.2.1 ASYNCHRONOUS LEAPFROG METHOD

To the best of our knowledge the *asynchronous leapfrog definition* was the first algebraically reversible non-symplectic solver, initially proposed by Mutze (2013) and popularized in a modern deep learning context by Zhuang et al. (2021). The asynchronous leapfrog method is a modification of the leapfrog method which converts it from a multi-step to single-step method. The method keeps track of a second state, $\{\boldsymbol{v}_n\}$ which is supposed to be *sufficiently close* to the value of the vector field. We define the method below in Definition A.1.

Definition A.1 (Asynchronous leapfrog method)**.** Initialize $\boldsymbol{v}_0 = \boldsymbol{f}(0, \boldsymbol{x}_0)$. Consider a step size of h and let $\hat{t}_n = t_n + h/2$, then a forward step of the asynchronous leapfrog method is defined as

$$\begin{aligned} \hat{\boldsymbol{x}}_n &= \boldsymbol{x}_n + \frac{1}{2}\boldsymbol{v}_n h, \\ \boldsymbol{v}_{n+1} &= 2\boldsymbol{f}(\hat{t}_n, \hat{\boldsymbol{x}}_n) - \boldsymbol{v}_n, \\ \boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + \boldsymbol{f}(\hat{t}_n, \hat{\boldsymbol{x}}_n)h, \end{aligned} \tag{21}$$

and the backward step is given as

$$\begin{aligned} \hat{\boldsymbol{x}}_n &= \boldsymbol{x}_{n+1} - \frac{1}{2}\boldsymbol{v}_{n+1} h, \\ \boldsymbol{x}_n &= \boldsymbol{x}_{n+1} - \boldsymbol{f}(\hat{t}_n, \hat{\boldsymbol{x}}_n)h, \\ \boldsymbol{v}_n &= 2\boldsymbol{f}(\hat{t}_n, \hat{\boldsymbol{x}}_n) - \boldsymbol{v}_{n+1}. \end{aligned} \tag{22}$$

Remark A.2. The method is a second-order solver (Zhuang et al., 2021, Theorem 3.1).

A.2.2 REVERSIBLE HEUN METHOD

Later work by Kidger et al. (2021) proposed the *reversible Heun method*, a general purpose reversible solver which is symmetric and is an algebraically reversible SDE solver in addition to being a reversible ODE solver. This solver keeps track of an auxiliary state variable $\hat{\boldsymbol{x}}_n$ and an extra copy of previous evaluations of the drift and diffusion coefficients. We present this method below in Definition A.3.

Definition A.3 (Reversible Heun method for ODEs)**.** Initialize $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$. Consider a step size of h, then a forward step of the reversible Heun method is defined as

$$\begin{aligned} \hat{\boldsymbol{x}}_{n+1} &= 2\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n + \boldsymbol{f}(t_n, \hat{\boldsymbol{x}}_n)h, \\ \boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + \frac{1}{2}\left(\boldsymbol{f}(t_{n+1}, \hat{\boldsymbol{x}}_{n+1}) + \boldsymbol{f}(t_n, \hat{\boldsymbol{x}}_n)\right)h. \end{aligned} \tag{23}$$

and the backward step is given as

$$\begin{aligned} \hat{\boldsymbol{x}}_n &= 2\boldsymbol{x}_{n+1} - \hat{\boldsymbol{x}}_{n+1} - \boldsymbol{f}(t_{n+1}, \hat{\boldsymbol{x}}_{n+1})h, \\ \boldsymbol{x}_n &= \boldsymbol{x}_{n+1} - \frac{1}{2}\left(\boldsymbol{f}(t_{n+1}, \hat{\boldsymbol{x}}_{n+1}) + \boldsymbol{f}(t_n, \hat{\boldsymbol{x}}_n)\right)h. \end{aligned} \tag{24}$$

Remark A.4. This method is a second-order solver (Kidger, 2022, Theorem 5.18).

Recall that simulating SDEs in reverse-time is much trickier than simulating ODEs in reverse-time. This observation is even more true of algebraically reversible methods for SDEs. To the best of our knowledge, the only general reversible solver for SDEs is the reversible Heun method. The main idea of the SDE formulation of the reversible Heun method is to extend the Euler-Heun method[5] like how Heun's method was extended to the reversible Heun solver for ODEs. We define the method in Kidger et al. (2021, Algorithm 1) below in Definition A.5.

[5]This converges with strong order $\frac{1}{2}$ in the Stratonovich sense (Rüemelin, 1982).

Definition A.5 (Reversible Heun method for SDEs). Initialize $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$. Consider a step size of h and let $\boldsymbol{W}_h \coloneqq \boldsymbol{W}_{t_{n+1}} - \boldsymbol{W}_{t_n}$, then a forward step of the reversible Heun method is defined as

$$\begin{aligned} \hat{\boldsymbol{x}}_{n+1} &= 2\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n + \boldsymbol{\mu}(t_n, \hat{\boldsymbol{x}}_n)h + \boldsymbol{\sigma}(t_n, \hat{\boldsymbol{x}}_n)\boldsymbol{W}_h, \\ \boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + \frac{1}{2}\left(\boldsymbol{\mu}(t_{n+1}, \hat{\boldsymbol{x}}_{n+1}) + \boldsymbol{\mu}(t_n, \hat{\boldsymbol{x}}_n)\right)h \\ &\quad + \frac{1}{2}\left(\boldsymbol{\sigma}(t_{n+1}, \hat{\boldsymbol{x}}_{n+1}) + \boldsymbol{\sigma}(t_n, \hat{\boldsymbol{x}}_n)\right)\boldsymbol{W}_h. \end{aligned} \tag{25}$$

and the backward step is given as

$$\begin{aligned} \hat{\boldsymbol{x}}_n &= 2\boldsymbol{x}_{n+1} - \hat{\boldsymbol{x}}_{n+1} - \boldsymbol{\mu}(t_{n+1}, \hat{\boldsymbol{x}}_{n+1})h - \boldsymbol{\sigma}(t_n, \hat{\boldsymbol{x}}_n)\boldsymbol{W}_h, \\ \boldsymbol{x}_n &= \boldsymbol{x}_{n+1} - \frac{1}{2}\left(\boldsymbol{\mu}(t_{n+1}, \hat{\boldsymbol{x}}_{n+1}) + \boldsymbol{\mu}(t_n, \hat{\boldsymbol{x}}_n)\right)h \\ &\quad - \frac{1}{2}\left(\boldsymbol{\sigma}(t_{n+1}, \hat{\boldsymbol{x}}_{n+1}) + \boldsymbol{\sigma}(t_n, \hat{\boldsymbol{x}}_n)\right)\boldsymbol{W}_h. \end{aligned} \tag{26}$$

Remark A.6. This method requires some tractable solution for recalculating the Brownian motion from a splittable PRNG.

A.2.3 McCallum-Foster method

Recent work by McCallum & Foster (2024) created a general method for constructing n-th order solvers from preexisting explicit single-step solvers while also addressing the stability issues that earlier methods suffered from. As McCallum & Foster (2024) simply refer to their method as *reversible X* where *X* is the underlying single-step solver we opt to refer to their method as the *McCallum-Foster method*. We restate the definition below.

Definition A.7 (McCallum-Foster method). Initialize $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ and let $\zeta \in (0, 1]$. Consider a step size of h, then a forward step of the McCallum-Foster method is defined as

$$\boldsymbol{x}_{n+1} = \zeta\boldsymbol{x}_n + (1-\zeta)\hat{\boldsymbol{x}}_n + \boldsymbol{\Phi}_h(t_n, \hat{\boldsymbol{x}}_n), \tag{27a}$$

$$\hat{\boldsymbol{x}}_{n+1} = \hat{\boldsymbol{x}}_n - \boldsymbol{\Phi}_{-h}(t_{n+1}, \boldsymbol{x}_{n+1}), \tag{27b}$$

and the backward step is given as

$$\hat{\boldsymbol{x}}_n = \hat{\boldsymbol{x}}_{n+1} + \boldsymbol{\Phi}_{-h}(t_{n+1}, \boldsymbol{x}_{n+1}), \tag{28a}$$

$$\boldsymbol{x}_n = \zeta^{-1}\boldsymbol{x}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{x}}_n - \zeta^{-1}\boldsymbol{\Phi}_h(t_n, \hat{\boldsymbol{x}}_n). \tag{28b}$$

Remark A.8. *N.B.*, the ζ and ζ^{-1} terms in the forward and backward steps determine the stability of the system.

Interestingly, McCallum & Foster (2024, Theorem 2.1) showed that this reversible method inherits the convergence order of single-step solver $\boldsymbol{\Phi}_h$ enabling the construction of an arbitrarily high-order reversible solver. We restate this result below in Theorem A.1.

Theorem A.1 (Convergence order of the McCallum-Foster method). *Consider the ODE in Equation (20) over $[0, T]$ with fixed time horizon $T > 0$. Let $T = Nh$ where $N > 0$ is the number of discretization steps and $h > 0$ is the step size. Let $\boldsymbol{\Phi}$ be a k-th order ODE solver such that it satisfies the Lipschitz condition*

$$\|\boldsymbol{\Phi}_\eta(\cdot, \boldsymbol{a}) - \boldsymbol{\Phi}_\eta(\cdot, \boldsymbol{b})\| \leq L|\eta|\|\boldsymbol{a} - \boldsymbol{b}\|, \tag{29}$$

for all $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$ and $\eta \in [-h_{max}, h_{max}]$ for some $h_{max} > 0$. Consider the reversible solution $\{\boldsymbol{x}_n, \hat{\boldsymbol{x}}_n\}_{nin\mathbb{N}}$ admitted by Equation (27). Then there exists constants $h_{max} > 0$, $C > 0$, such that, for $h \in (0, h_{max}]$,

$$\|\boldsymbol{x}_n - \boldsymbol{x}(t_n)\| \leq Ch^k. \tag{30}$$

A.2.4 EXPLICIT AND EFFECTIVELY SYMMETRIC SCHEMES

Contemporary work by Shmelev & Salvi (2025) explores an *explicit and effectively symmetric* (EES) Runge-Kutta schemes (Shmelev et al., 2025) for neural SDEs. The *key* difference is that these schemes are only *approximately* invertible rather than *exactly* invertible. The other large difference is they construct the scheme for *rough differential equations* (RDEs) driven by a α-Hölder branched rough path, $\alpha \in (0, 1]$. More concretely, for some driving signal $\boldsymbol{X} : [0, T] \to \mathbb{R}^d$, *e.g.*, a semi-martingale, lifted to a *rough path* $\mathcal{X}$ (*cf*. Appendix F), we consider the rough differential equation

$$\mathrm{d}\boldsymbol{Y}_t = \boldsymbol{f}(\boldsymbol{Y}_t)\,\mathrm{d}\mathcal{X}_t, \tag{31}$$

where $\boldsymbol{f}$ is sufficiently smooth and bounded with bounded derivatives. Following Redmann & Riedel (2020) assume that there exists smooth paths $\{\boldsymbol{X}^h\}_{h>0}$ for step-sizes $h > 0$ whose natural lifts to branch rough paths $\{\mathcal{X}^h\}_{h>0}$ converge almost surely to $\mathcal{X}$ under the metric of α-Hölder rough paths as $h \to 0$. Then we have the solution with drive $\mathcal{X}^h$ given by

$$\mathrm{d}\boldsymbol{Y}_t^h = \boldsymbol{f}(\boldsymbol{Y}_t^h)\,\mathrm{d}\mathcal{X}_t^h. \tag{32}$$

Shmelev & Salvi (2025) then uses the following Runge-Kutta scheme for the RDE given by

$$\boldsymbol{y}_{n+1}^h = \boldsymbol{y}_n^h + \sum_{m=1}^{d}\sum_{i=1}^{s} b_i \boldsymbol{f}_m(\boldsymbol{k}_i)\boldsymbol{X}_{t_n,t_{n+1}}^{(m)}, \tag{33a}$$

$$\boldsymbol{k}_i = \boldsymbol{y}_n^h + \sum_{m=1}^{d}\sum_{i=1}^{s} a_{ij} \boldsymbol{f}_m(\boldsymbol{k}_i)\boldsymbol{X}_{t_n,t_{n+1}}^{(m)}, \tag{33b}$$

where $\boldsymbol{X}_{t_n,t_{n+1}}$ denotes the increment of $\boldsymbol{X}^h$ over $[t_n, t_{n+1}]$. With an appropriate choice of coefficients a_{ij} and b_i the Runge-Kutta scheme for the RDE is *approximately* reversible (Shmelev & Salvi, 2025). Cleary, this scheme is quite different from the SRK schemes we study and construct *exactly* reversible schemes from (*cf*. Appendix B).

A.3 A NOTE ON STABILITY

Historically, the stability properties of reversible solvers has been one of their weakest attributes (Kidger, 2022), limiting their use in practical applications. We formally introduce the notation of stability following Kidger (2022, Definition C.39), which we rewrite below in Definition A.9.

Definition A.9 (Region of stability). Fix some numerical differential equation solver and let $\{\boldsymbol{x}_n^{\lambda,h}\}_{n\in\mathbb{N}}$ be the solution admitted by the numerical scheme solving the linear (or Dahlquist) test equation

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \lambda \boldsymbol{x}(t), \tag{34}$$

where $\lambda \in \mathbb{C}$, $h > 0$ is the step size, and $\boldsymbol{x}_0 \in \mathbb{R}^d$ is a non-zero initial condition. The region of stability is defined as

$$\{h\lambda \in \mathbb{C} : \{\boldsymbol{x}_n^{\lambda,h}\}_{n\in\mathbb{N}} \text{ is uniformly bounded over } t_n\}. \tag{35}$$

I.e., there exists a constant C depending on λ and h but independent of t_n such that $\|\boldsymbol{x}_n^{\lambda,h}\| < C$.

With the linear test equation Equation (34) the ODE converges asymptotically when $\Re(\lambda) \leq 0$,[6] and thus we are interested in numerical schemes which are bounded when the underlying analytical solution converges. Ideally, a numerical scheme would converge for all $h\lambda$ with $\Re(\lambda) < 0$.[7] Thus, the larger the region of stability the larger the step size we can take, wherein the numerical scheme still converges.

Remark A.10. Regrettably, the reversible Heun, leapfrog, and asynchronous leapfrog methods have poor stability properties. Specifically, the region of stability for all the methods is the complex interval $[-i, i]$, see Kidger (2022, Theorem 5.20) for reversible Heun, Shampine (2009, Section 2) for leapfrog, and Zhuang et al. (2021, Appendix A.4) for asynchronous leapfrog.

[6] The ODE converges to 0 when $\Re(\lambda) < 0$.

[7] A region of stability which satisfies is known as a region of absolute stability.

In other words, all previous reversible solvers are nowhere linearly stable for any step size h.[8] The instability in both asynchronous leapfrog and reversible Heun can be attributed to a step of general form $2A - B$, *i.e.*, we can write the source of instability as

$$\begin{aligned} &2\boldsymbol{f}(\hat{t}_n, \hat{\boldsymbol{x}}_n) - \boldsymbol{v}_n, && \text{(asynchronous leapfrog)} \\ &2\boldsymbol{x}_{n+1} - \hat{\boldsymbol{x}}_{n+1}. && \text{(reversible Heun)} \end{aligned}$$

Thus the instability in these reversible schemes is caused by a decoupling between $\boldsymbol{v}_n$ and $\boldsymbol{f}(t_n, \boldsymbol{x}_n)$ (asynchronous leapfrog); and $\boldsymbol{x}_n$ and $\hat{\boldsymbol{x}}_n$ (reversible Heun). The strategy of McCallum & Foster (2024) is to couple $\boldsymbol{x}_n$ and $\hat{\boldsymbol{x}}_n$ together with the coupling parameter ζ. Using this strategy, they showed that it was possible to construct a reversible solver with a non-trivial region of convergence. Let $\boldsymbol{\Phi}_h(t_n, \boldsymbol{x}_n) = R(h\lambda)\boldsymbol{x}_n$ and let $R(h\lambda)$ denote the *transfer function* used in analysis of Runge-Kutta methods with step size h (see Stewart, 2022). We restate McCallum & Foster (2024, Theorem 2.3) below.

Theorem A.2 (Region of stability for the McCallum-Foster method). *Let $\boldsymbol{\Phi}$ be given by an explicit Runge-Kutta solver. Then the reversible numerical solution $\{\boldsymbol{x}_n, \hat{\boldsymbol{x}}_n\}_{n\in\mathbb{N}}$ given by Equation (27) is linearly stable iff*

$$|\Gamma| < 1 + \zeta, \tag{36}$$

where

$$\Gamma = 1 + \zeta - (1 - \zeta)R(-h\lambda) - R(-h\lambda)R(h\lambda). \tag{37}$$

Remark A.11. The McCallum-Foster method when constructed from explicit Runge-Kutta methods have a *non-trivial* region of stability. Note, however, that this region of stability is smaller than the original region of stability from the original Runga-Kutta method.

A.4 Exact inversion of diffusion models

Independent of the work on reversible solvers for neural ODEs several researchers have developed reversible methods for solving the probability flow ODE—often in the literature on diffusion models this is called the *exact inversion* of diffusion models.

A.4.1 EDICT sampler

The first work to explore this topic of exact inversion with diffusion models was that of Wallace et al. (2023), who inspired by coupling layers in normalizing flows (Dinh et al., 2015) proposed a reversible solver which they refer to as *exact diffusion inversion via coupled transformations* (EDICT). Like all reversible solvers this method keeps track of an extra state, denoted by $\{\boldsymbol{y}_n\}_{n\in\mathbb{N}}$, with $\boldsymbol{y}_0 = \boldsymbol{x}_0$. Letting $a_n = \frac{\alpha_{n+1}}{\alpha_n}$ and $b_n = \sigma_{n+1} - \frac{\alpha_{n+1}}{\alpha_n}\sigma_n$, this numerical scheme can be described as

$$\begin{aligned} \boldsymbol{x}_n^{\text{inter}} &= a_n \boldsymbol{x}_n + b_n \boldsymbol{x}^{\theta}_{T|t_n}(\boldsymbol{y}_n), \\ \boldsymbol{y}_n^{\text{inter}} &= a_n \boldsymbol{y}_n + b_n \boldsymbol{x}^{\theta}_{T|t_n}(\boldsymbol{x}_n^{\text{inter}}), \\ \boldsymbol{x}_{n+1} &= \xi \boldsymbol{x}_n^{\text{inter}} + (1-\xi)\boldsymbol{y}_n^{\text{inter}} \\ \boldsymbol{y}_{n+1} &= \xi \boldsymbol{x}_n^{\text{inter}} + (1-\xi)\boldsymbol{x}_{n+1}, \end{aligned} \tag{38}$$

[8] Linearly stability refers to stability for linear test equations with $\Re(\lambda) < 0$.

where $\xi \in (0, 1)$ is a mixing parameter.[9] This method can be inverted to obtain a closed form expression for backward step:

$$\begin{aligned} \boldsymbol{y}_n^{\text{inter}} &= \frac{\boldsymbol{y}_{n+1} - (1-\xi)\boldsymbol{x}_{n+1}}{\xi}, \\ \boldsymbol{x}_n^{\text{inter}} &= \frac{\boldsymbol{y}_{n+1} - (1-\xi)\boldsymbol{y}_n^{\text{inter}}}{\xi}, \\ \boldsymbol{y}_n &= \frac{\boldsymbol{y}_n^{\text{inter}} - b_n \boldsymbol{x}_{T|t_n}^{\theta}(\boldsymbol{x}_n^{\text{inter}})}{a_n}, \\ \boldsymbol{x}_n &= \frac{\boldsymbol{x}_n^{\text{inter}} - b_n \boldsymbol{x}_{T|t_n}^{\theta}(\boldsymbol{y}_n)}{a_n}. \end{aligned} \tag{39}$$

Notably, the EDICT solver was developed in the context of discrete-time diffusion models and the connection to reversible solvers for ODEs was not considered in the original work. *N.B.*, to the best of our knowledge our work is the first to draw the connection between the work on reversible ODE solvers and exact inversion with diffusion models. Unfortunately, this method suffers from poor convergence issues (see Remark A.12) and generally has poor performance when used to perform sampling with diffusion models, thereby limiting its utility in practice (Zhang et al., 2024; Wang et al., 2024).

Remark A.12. Later work by Wang et al. (2024, Proposition 6) showed that EDICT is actually a zero-order method, *i.e.*, the local truncation error is $\mathcal{O}(h)$, making it generally unsuitable in practice.

A.4.2 BDIA SAMPLER

Later work by Zhang et al. (2024) proposed a reversible solver for the probability flow ODE which they call *bidirectional integration approximation* (BDIA). The core idea is to use both single-step methods $\boldsymbol{\Phi}_{t_n,t_{n-1}}$ and $\boldsymbol{\Phi}_{t_n,t_{n+1}}$ to induce reversibility.[10] Then using these two approximations—both of which are computed from a discretization centered around $\boldsymbol{x}_n$—the process is update via a multistep process with a forward step of[11]

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_{n-1} - \boldsymbol{\Phi}_{t_n,t_{n-1}}(\boldsymbol{x}_n) + \boldsymbol{\Phi}_{t_n,t_{n+1}}(\boldsymbol{x}_n). \tag{40}$$

The backwards step can easily be expressed as

$$\boldsymbol{x}_{n-1} = \boldsymbol{x}_{n+1} + \boldsymbol{\Phi}_{t_n,t_{n-1}}(\boldsymbol{x}_n) + \boldsymbol{\Phi}_{t_n,t_{n+1}}(\boldsymbol{x}_n). \tag{41}$$

In practice, BDIA uses the DDIM solver (*i.e.*, Euler) for $\boldsymbol{\Phi}$, but in theory one could use a higher-order method—this was not explored in Zhang et al. (2024).

Proposition A.3 (BDIA is the leapfrog/midpoint method)**.** *The BDIA method described in Equation* (40) *is the leapfrog/midpoint method when* $\boldsymbol{\Phi}_h(t, \boldsymbol{x}) = h\boldsymbol{u}_t^{\theta}(\boldsymbol{x})$, i.e.*, the Euler step.*

Proof. This can be shown rather straightforwardly by substitution, *i.e.*,

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_{n-1} + 2h\boldsymbol{u}_{t_n}^{\theta}(\boldsymbol{x}_n). \tag{42}$$

□

Corollary A.3.1 (BDIA is a first-order method)**.** *BDIA is first-order method,* i.e.*, the local truncation error is* $\mathcal{O}(h^2)$.

Remark A.13. This result was also observed in Wang et al. (2024, Proposition 6).

[9] In practice, when used for image editing the authors found that the parameter ξ controlled how closely the EDICT sampler aligned with the original sample, with lower values corresponding to higher agreement with the original sample.

[10] *N.B.*, in the original paper, Zhang et al. (2024) use quite different notation for explaining their idea; however, we find our presentation to be simpler for the reader as it more easily enables comparison to other methods.

[11] In some sense, this is reminiscent of the idea from the more general McCallum-Foster method; however, this approach results in a multi-step method unlike the single-step method of McCallum & Foster (2024).

Corollary A.3.2 (BDIA is nowhere linearly stable)**.** *BDIA is nowhere linearly stable,* i.e.*, the region of stability is the complex interval* $[-i, i]$.

Proof. This follows straightforwardly from Proposition A.3 and Shampine (2009, Section 2). □

Zhang et al. (2024) introduce a hyperparameter $\gamma \in [0, 1]$ which is used below

$$\hat{\boldsymbol{\Phi}}_{t_n,t_{n-1}}(\boldsymbol{x}_n) = (1-\gamma)(\boldsymbol{x}_{n-1} - \boldsymbol{x}_n) + \gamma \boldsymbol{\Phi}_{t_n,t_{n-1}}(\boldsymbol{x}_n), \tag{43}$$

to modify the BDIA update rule in Equation (40). Thus, γ can be viewed as a parameter which interpolates between the midpoint and Euler schemes. For image editing applications the authors found this parameter to control how closely the BDIA sampler aligned with the original image, with lower values corresponding to higher agreement with the original image (making it similar to the ξ parameter from BDIA).

A.4.3 BELM SAMPLER

Recently, Wang et al. (2024) proposed a linear multi-step reversible solver for the probability flow ODE called the *bidirectional explicit linear multi-step* (BELM) sampler. First, they reparameterize the probability flow ODE as

$$\mathrm{d}\overline{\boldsymbol{x}}(t) = \overline{\boldsymbol{x}}^{\theta}_{T|\overline{\sigma}_t}(\overline{\boldsymbol{x}}(t))\, \mathrm{d}\overline{\sigma}_t, \tag{44}$$

where $\overline{\boldsymbol{x}}(t) \coloneqq \boldsymbol{x}(t)/\alpha_t$, $\overline{\sigma}(t) \coloneqq \sigma_t/\alpha_t$, and $\overline{\boldsymbol{x}}^{\theta}_{T|\overline{\sigma}_t}(\overline{\boldsymbol{x}}(t)) = \boldsymbol{x}^{\theta}_{T|t}(\boldsymbol{x}(t))$.[12] The BELM sampler makes use of the variable-stepsize-variable-formula (VSVF) linear multi-step methods (Crouzeix & Lisbona, 1984) to construct the numerical solver. The k-step VSVF linear multi-step method for solving the reparameterized probability flow ODE in Equation (44) is given by

$$\overline{\boldsymbol{x}}_{n+1} = \sum_{m=1}^{k} a_{n,m} \overline{\boldsymbol{x}}_{n+1-m} \tag{45}$$

$$+ \sum_{m=1}^{k-1} b_{n,m} h_{n+1-m} \overline{\boldsymbol{x}}^{\theta}_{T|\overline{\sigma}_{n+1-m}}(\overline{\boldsymbol{x}}_{n+1-m}). \tag{46}$$

where $a_{n,m} \neq 0$,[13] and $b_{n,m}$ are coefficients chosen using dynamic multi-step formulæ to find the coefficients (Crouzeix & Lisbona, 1984); and h_n are step sizes chosen beforehand. This scheme can be reversed via the backward step

$$\overline{\boldsymbol{x}}_{n+1-k} = \frac{1}{a_{n,k}} \overline{\boldsymbol{x}}_{n+1} - \sum_{m=1}^{k-1} \frac{a_{n,m}}{a_{n,k}} \overline{\boldsymbol{x}}_{n+1-m} \tag{47}$$

$$- \sum_{m=1}^{k-1} \frac{b_{n,m}}{a_{n,k}} h_{n+1-m} \overline{\boldsymbol{x}}^{\theta}_{T|\overline{\sigma}_{n+1-m}}(\overline{\boldsymbol{x}}_{n+1-m}). \tag{48}$$

Remark A.14. The BELM samplers require $k-1$ extra to be stored in memory in order to be reversible. In contrast, McCallum & Foster (2024) only requires storing one extra states, irregardless of the desired convergence order. Additionally, poor stability is a concern with such linear multi-step methods (see Kidger, 2022, Remark 5.24).

Remark A.15. Interestingly, the earlier EDICT and BDIA methods can be viewed as instances of the BELM method (Wang et al., 2024, Appendicies A.7 and A.8).

By solving the multi-step formulæ to minimize the local truncation error Wang et al. (2024) propose an instance of the BELM solver which they refer to as *O-BELM* defined as[14]

$$\overline{\boldsymbol{x}}_{n+1} = \frac{h_n^2}{h_{n-1}^2} \overline{\boldsymbol{x}}_{n-1} + \frac{h_{n-1}^2 + h_n^2}{h_{n-1}^2} \overline{\boldsymbol{x}}_n - \frac{h_n(h_n + h_{n+1})}{h_{n+1}} \overline{\boldsymbol{x}}_{0|\overline{\sigma}_n}(\overline{\boldsymbol{x}}_n). \tag{49}$$

Notably, the O-BELM sampler can also be viewed as instance of the leapfrog/midpoint method.

[12] *N.B.*, this is a popular parameterization of diffusion models and affine conditional flows. This can be done *mutatis mutandis* for target prediction models retrieving (Blasingame & Liu, 2025, Proposition D.2).

[13] This is to ensure that the method is reversible.

[14] *N.B.*, the original equation in Wang et al. (2024, Equation (18)) had a sign difference for the coefficient of $b_{i,1}$; however, this is due to differences in convention in handling integration in reverse-time.

Table 5: Comparison of different (non-symplectic) reversible solvers. We note that some of the solvers were developed particularly for the probability flow ODE (an affine conditional flow) whilst others work for general ODEs/SDEs. In the first column we denote the number of extra states the numerical scheme needs to keep in memory to ensure algebraic reversibility. For BELM k denotes the number of steps and for McCallum-Foster k denotes the convergence order of the underlying single-step solver. For the column labelled *region of linear stability* we mean there exists some subset of $\mathbb{C}$ which is the region of stability and the set is not a null set. The proof of convergence for BELM is only provided for the special case (called *O-BELM* in Wang et al. (2024)) with $k = 2$.

Solver	**SDE**	**Exponential integrators**	**Number of extra states**	**Local truncation error**	**Region of linear stability**	**Proof of convergence**
Asynchronous leapfrog	✗	✗	1	$\mathcal{O}(h^3)$	✗	✓
Reversible Heun	✓	✗	1	$\mathcal{O}(h^3)$	✗	✓
McCallum-Foster	✗	✗	1	$\mathcal{O}(h^{k+1})$	✓	✓
EDICT	✗	✗	1	$\mathcal{O}(h)$	✗	✗
BDIA	✗	✗	1	$\mathcal{O}(h^2)$	✗	✗
BELM	✗	✓	$k-1$	$\mathcal{O}(h^{k+1})$	✗	$\sim$
Rex	✓	✓	1	$\mathcal{O}(h^{k+1})$	✓	✓

Theorem A.4 (O-BELM is the leapfrog/midpoint method)**.** *Fix a step size $h_n = h$ for all n, then O-BELM is the leapfrog/midpoint method.*

Proof. This follows from substitution of $h_n = h$. □

Corollary A.4.1 (O-BELM is nowhere linearly stable)**.** *Fix a step size $h_n = h$, then O-BELM is nowhere linearly stable,* i.e.*, the region of stability is the complex interval $[-i, i]$.*

A.4.4 CycleDiffusion

To our knowledge, the *only* other work to propose exact inversion with the SDE formulation of the diffusion models is the work of Wu & la Torre (2023). However, there a *several* noticeable distinctions, the largest being that they store the entire solution trajectory in memory. Given a particular realization of the Wiener process that admits $\boldsymbol{x}_t \sim \mathcal{N}(\alpha_t \boldsymbol{x}_0 \mid \sigma_t^2 \mathbf{I})$, then given $\boldsymbol{x}_s$ and noise $\boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ we can calculate

$$\boldsymbol{x}_t = \frac{\alpha_t}{\alpha_s}\boldsymbol{x}_s + 2\sigma_t(e^h - 1)\hat{\boldsymbol{x}}_{T|s}(\boldsymbol{x}_s) + \sigma_t\sqrt{e^{2h} - 1}\boldsymbol{\epsilon}_s. \tag{50}$$

Wu & la Torre (2023) propose to invert this by first calculating, for two samples $\boldsymbol{x}_t$ and $\boldsymbol{x}_s$, the noise $\boldsymbol{\epsilon}_s$. This can be calculated by rearranging the previous equation to find

$$\boldsymbol{\epsilon}_s = \frac{\boldsymbol{x}_t - \frac{\alpha_t}{\alpha_s}\boldsymbol{x}_s + 2\sigma_t(e^h - 1)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_s, \boldsymbol{z}, s)}{\sigma_t\sqrt{e^{2h} - 1}} \tag{51}$$

With this the sequence $\{\boldsymbol{\epsilon}_{t_i}\}_{i=1}^N$ of added noises can be calculated which can be used to reconstruct the original input from the initial realization of the Wiener process. However, unlike our approach, this process requires storing the entire realization in memory.

A.4.5 Summary

We present a summary of related works on either *exact inversion* or *reversible solvers* below in Table 5. *N.B.*, we omit *CycleDiffusion* because it is more orthogonal to the general concept of a reversible solver and is only reversible in the trivial sense.

A.5 SDE solvers for diffusion models

Next we discuss related works on SDE solvers for the reverse-time diffusion SDE in Equation (18). Now there are numerous *stochastic Runge-Kutta* (SRK) methods in the literature all tailor to specific

types of SDEs, which we can distinguish by the their strong order of convergence (see Definition D.1) and strong order conditions. For example the classic Euler-Maruyama scheme (Kloeden & Platen, 1992) has strong order of convergence of 0.5 and was straightforwardlly applied to the reverse-time diffusion SDE in Jolicoeur-Martineau et al. (2021) as a baseline. Song et al. (2021b) proposed an ancestral sampling scheme for a discretization of the forward-time diffusion SDE in Equation (16) with additional Langevin dynamics; likewise, the DDIM solver from Song et al. (2021a) can be viewed a sort of Euler-Maruyama scheme. Other classic SDE schemes like SRA1/SRA2/SRA3 schemes (Rößler, 2010) all have strong order of convergence 1.5 for additive noise SDEs and were tested for diffusion models in Jolicoeur-Martineau et al. (2021).

More recently, researchers have explored exponential solvers for SDEs, *e.g.*, the exponential Euler-Maruyama method (Komori et al., 2017) and the *stochastic Runge-Kutta Lawson* (SRKL) schemes (Debrabant et al., 2021). From an initial inspection the SRKL schemes of Debrabant et al. (2021, Algorithm 1) is somewhat similar to our method for constructing $\boldsymbol{\Psi}$; however, upon closer inspection they are some key fundamental differences.[15] The largest of these is how the underlying SRK schemes are represented. In particular the SRKL schemes choose to follow the conventions of Burrage & Burrage (2000) (for Stratonovich SDEs) in constructing the underlying SRK schemes; whereas we follow the SRK schemes outlined by Foster et al. (2024) (*cf*. Appendix B). These differences stem from how one chooses to handle the the iterated stochastic integrals from the Stratonovich-Taylor (or Itô-Taylor) expansions.

A.5.1 Comparison with SEEDS

Mostly directly relevant to our work on constructing a stochastic $\boldsymbol{\Psi}$ is the SEEDS family of solvers proposed by Gonzalez et al. (2024). Similar to us, they also approach using exponential methods to simplify the expression of diffusion models Gonzalez et al. (2024, Appendix B.1). There are two *key* distinctions, namely, 1) that they use the *stochastic exponential time differencing* (SETD) method (Adamu, 2011), whereas, we construct stochastic Lawson schemes;[16] and 2) that they use a different technique for modeling the iterated stochastic integrals for high-order solvers. In particular, SEEDS introduces a decomposition for the iterated stochastic integrals produced by the Itô-Taylor expansions of Equation (18) such that the decomposition preserves the Markov property, *i.e.*, the random variables used to construct model the Brownian increments from iterated integrals are independent on non-overlapping intervals and dependent on overlapping intervals (see Gonzalez et al., 2024, Proposition 4.3). By making use of the SRK schemes of Foster et al. (2024) developed from using the space-time Lévy area to construct high-order splitting methods we have an alternative method for ensuring this property. This results in our solver based on ShARK (see Appendix B.3, *cf*. Theorem 3.3) having a strong order of convergence of 1.5; whereas, SEEDS-3 only achieves a *weak* order of convergence of 1.

This brings us to another large difference, the SEEDS solvers focus on the *weak* approximation to Equation (18); whereas, as we are concerned with the *strong* approximation to Equation (18). The difference between these two is that the weak convergence is considered with the precisions of the *moments*; whereas, strong convergence is concerned with the precision of the *path*. Moreover, by definition a strong order of convergence implies a weak order of convergence, the converse is not true. In particular, for our application of developing *reversible* schemes this strong order of convergence is particularly important as we care about the path. Thus the technique SEEDS uses to replace iterated Itô integrals with other random variables with equivalent moment conditions is *wholely unsuitable* for our purposes as we desire a *strong* approximation.

[15] *N.B.*, in general Debrabant et al. (2021) consider full stochastic Lawson schemes where the integrating factor is a stochastic process given by the matrix exponential applied to linear terms in the drift and diffusion coefficients; conversely, the drift stochastic Lawson schemes are more similar to what we study.

[16] *N.B.*, for certain scenarios these two different viewpoints converge, particularly, in the deterministic case. See our discussion on the family of DPM-Solvers which also use (S)ETD in Appendix E.

B STOCHASTIC RUNGE-KUTTA METHODS

Recall that the general Butcher tableau for a s-stage explicit RK scheme (Stewart, 2022, Section 6.1.4) for a generic ODE is written as

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s(s-1)} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} = \begin{array}{c|c} c & a \\ \hline & b \end{array}. \tag{52}$$

E.g., the famous 4-th order Runge-Kutta (RK4) method is given by

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}. \tag{53}$$

However, for SDEs this is much trickier due to the presense of iterated stochastic integrals in the Itô-Taylor or Stratonovich-Taylor expansions (Kloeden & Platen, 1992). Consider a d-dimensional *Stratonovich* SDE driven by d_w-dimensional Brownian motion $\{\boldsymbol{W}_t\}_{t\in[0,T]}$ defined as

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{\mu}_\theta(t, \boldsymbol{X}_t)\,\mathrm{d}t + \boldsymbol{\sigma}_\theta(t, \boldsymbol{X}_t) \circ \mathrm{d}\boldsymbol{W}_t, \tag{54}$$

where $\boldsymbol{\mu}_\theta \in \mathcal{C}^2(\mathbb{R}\times\mathbb{R}^d;\mathbb{R}^d)$ and $\boldsymbol{\sigma}_\theta \in \mathcal{C}^3(\mathbb{R}\times\mathbb{R}^d;\mathbb{R}^{d\times d_w})$ satisfy the usual regularity conditions for Stratonovich SDEs (Øksendal, 2003, Theorem 5.2.1) and where $\circ\mathrm{d}\boldsymbol{W}_t$ denotes integration in the Stratonovich sense.

Rößler (2025) write one such class of an s-stage explicit SRK methods (*cf*. Burrage & Burrage, 2000; Rößler, 2010) for Equation (54) as

$$\begin{aligned} \boldsymbol{Z}_i^{(0)} &= \boldsymbol{X}_n + h\sum_{j=1}^{i-1} a_{ij}^{(0)}\boldsymbol{\mu}_\theta(t_n + c_j^{(0)}, \boldsymbol{Z}_j^{(0)}), \\ \boldsymbol{Z}_i^{(k)} &= \boldsymbol{X}_n + h\sum_{j=1}^{i-1} a_{ij}^{(1)}\boldsymbol{\mu}_\theta(t_n + c_j^{(0)}, \boldsymbol{Z}_j^{(0)}) + \sum_{j=1}^{i-1}\sum_{l=1}^{d_w} a_{ij}^{(2)}\boldsymbol{I}_{(l,k),n}\boldsymbol{\sigma}_\theta(t_n + c_j^{(1)}, \boldsymbol{Z}_i^{(l)}), \\ \boldsymbol{X}_{n+1} &= \boldsymbol{X}_n + h\sum_{i=1}^{s} b_i^{(0)}\boldsymbol{\mu}_\theta(t_n + c_i^{(0)}, \boldsymbol{Z}_j^{(0)}) + \sum_{i=1}^{s}\sum_{k=1}^{d_w}\left(b_i^{(1)}\boldsymbol{I}_{(k),n} + b_i^{(2)}\right)\boldsymbol{\sigma}_\theta(t_n + c_j^{(1)}, \boldsymbol{Z}_i^{(k)}), \end{aligned} \tag{55}$$

for $k = 1, \ldots, d_w$ and where

$$\boldsymbol{I}_{(k),n} = \int_{t_n}^{t_{n+1}} \circ\mathrm{d}\boldsymbol{W}_u^k = \boldsymbol{W}_{t_{n+1}}^k - \boldsymbol{W}_{t_n}^k, \tag{56}$$

$$\boldsymbol{I}_{(l,k),n} = \int_{t_n}^{t_{n+1}} \int_{t_n}^{u} \circ\mathrm{d}\boldsymbol{W}_v^l \circ \mathrm{d}\boldsymbol{W}_u^k, \tag{57}$$

let $\hat{\boldsymbol{I}}$ denote the iterated integrals for the Itô case *mutatis mutandis*. This scheme is described the by the *extended* Butcher tableau (Rößler, 2025)

$$\begin{array}{c|c|c|c} c^{(0)} & a^{(0)} & & \\ \hline c^{(1)} & a^{(1)} & a^{(2)} & \\ \hline & b^{(0)} & b^{(1)} & b^{(2)} \end{array}. \tag{58}$$

These iterated integrals $\boldsymbol{I}_{(l,k),n}$ are very tricky to work with and can raise up many practical concerns. As alluded to earlier (*cf*. Appendix A.5.1) it is common to use a weak approximation of such integrals via a random variables with corresponding moments. This results in two drawbacks: 1) the resulting SDE scheme only converges in the *weak* sense and 2) the solution yielding by the scheme is not a Markov chain in general. SEEDS overcomes the second issue by using a special decomposition to preserve the Markov property, see the ablations in Gonzalez et al. (2024) for more details on this topic in practice.

B.1 Foster-Reis-Strange SRK Scheme

Conversely, Foster et al. (2024) propose another SRK scheme based on higher-order splitting methods for Stratonovich SDEs. For the Stratonovich SDE in Equation (54) Foster et al. (2024) write an s-stage SRK as

$$\begin{aligned}
\boldsymbol{\mu}_\theta^i &= \boldsymbol{\mu}_\theta(t_n + c_i h, \boldsymbol{Z}_i), \\
\boldsymbol{\sigma}_\theta^i &= \boldsymbol{\sigma}_\theta(t_n + c_i h, \boldsymbol{Z}_i), \\
\boldsymbol{Z}_i &= \boldsymbol{X}_n + h\left(\sum_{j=1}^{i-1} a_{ij}\boldsymbol{\mu}_\theta^j\right) + \boldsymbol{W}_n\left(\sum_{j=1}^{i-1} a_{ij}^W \boldsymbol{\sigma}_\theta^j\right) + \boldsymbol{H}_n\left(\sum_{j=1}^{i-1} a_{ij}^H \boldsymbol{\sigma}_\theta^j\right), \\
\boldsymbol{X}_{n+1} &= \boldsymbol{X}_n + h\left(\sum_{i=1}^{s} b_i\boldsymbol{\mu}_\theta^i\right) + \boldsymbol{W}_n\left(\sum_{i=1}^{s} b_i^W \boldsymbol{\sigma}_\theta^i\right) + \boldsymbol{H}_n\left(\sum_{i=1}^{s} b_i^H \boldsymbol{\sigma}_\theta^i\right),
\end{aligned} \tag{59}$$

where $h = t_{n+1} - t_n$ is the step size and $\boldsymbol{W}_n \coloneqq \boldsymbol{W}_{t_n,t_{n+1}}$ and $\boldsymbol{H}_n \coloneqq \boldsymbol{H}_{t_n,t_{n+1}}$ are the Brownian and space-time Lévy increments (*cf*. Definition 3.1) respectively; and where $a_{ij}, a_{ij}^W, a_{ij}^H \in \mathbb{R}^{s\times s}$, $b_i, b_i^W, b_i^H \in \mathbb{R}^s$, and $c_i \in \mathbb{R}^s$ for the coefficients for an *extended* Butcher tableau (Foster et al., 2024) which is given as

$$\begin{array}{c|c|c|c}
c & a & a^W & a^H \\
\hline
 & b & b^W & b^H
\end{array}. \tag{60}$$

E.g., we can write the famous Euler-Maruyama scheme as

$$\begin{array}{c|c|c|c}
0 & 0 & 0 & 0 \\
\hline
 & 1 & 1 & 0
\end{array}. \tag{61}$$

B.2 Independence of the Brownian and Lévy Increments

Remarkably, in Foster et al. (2020, Theorem 2.2) present a polynomial Karhunen-Loève theorem for the Brownian bridge (*cf*. Definition G.1)—picture an stochastic analogue to the Fourier series of a function on a bounded interval—which leads to a most useful remark (Foster et al., 2020, Remark 3.6) which we restate below.

Remark B.1. We have that $H_{s,t} \sim \mathcal{N}(0, \frac{1}{12}h)$ is independent of $\kappa_{s,t}$ when $d = 1$, likewise, since the coordinate processes of a Brownian motion are independent, one can write $\boldsymbol{W}_{s,t} \sim (\boldsymbol{0}, h\boldsymbol{I})$ and $\boldsymbol{H}_{s,t} \sim \mathcal{N}(\boldsymbol{0}, \frac{1}{12}h\boldsymbol{I})$ are independent.

Thus we have found another remedy to the problem of independent increments, whilst still being able to obtain a *strong* approximation of the SDE.

B.3 ShARK

Recently, Foster et al. (2024) developed *shifted additive-noise Runge-Kutta* (ShARK) for additive noise SDEs which is based on Foster et al. (2024, Equation (6.1)). This scheme has converges strongly with order 1.5 for additive-noise SDEs and makes two evaluations of the drift and diffusion per step.

ShARK is described via the following extended Butcher tableau

$$\begin{array}{c|cc|c|c} 0 & & & 0 & 1 \\ \frac{5}{6} & \frac{5}{6} & & \frac{5}{6} & 1 \\ \hline & 0.4 & 0.6 & 1 & 0 \\ & -0.6 & 0.6 & & \end{array}. \tag{62}$$

The second row for the b variable describes the coefficients used for adaptive-step size solvers to approximate the error at each step. The Butcher tableau for this scheme can be found here: https://github.com/patrick-kidger/diffrax/blob/main/diffrax/_solver/shark.py.

C Derivation of Rex

We derive the Rex scheme presented in Proposition 3.1 in the main paper.

Nomenclature. As alluded to earlier, there exist two popular reparameterizations of the score function which are used widely in practice, namely the noise prediction (Ho et al., 2020) and data prediction (Kingma et al., 2021) formulations. Following the conventions of Lipman et al. (2024) we write noise prediction model as $\boldsymbol{x}_{T|t}(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{X}_T|\boldsymbol{X}_t = \boldsymbol{x}]$ and write data prediction model as $\boldsymbol{x}_{0|t}(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{X}_0|\boldsymbol{X}_t = \boldsymbol{x}]$. Throughout this paper we will assume the existence of *sufficiently trained* diffusion models denoted $\boldsymbol{x}^\theta_{T|t}(\boldsymbol{x}) \approx \boldsymbol{x}_{T|t}(\boldsymbol{x})$ and $\boldsymbol{x}^\theta_{0|t}(\boldsymbol{x}) \approx \boldsymbol{x}_{0|t}(\boldsymbol{x})$.

C.1 Rex (ODE)

In this section we derive the Rex scheme for the probability flow ODE. We present derivations for both the data prediction and noise prediction formulations. It is well known (Blasingame & Liu, 2025, Equation (19); *cf*. Domingo-Enrich et al., 2025, Equation (8)) that the ODE in Equation (19) can be rewritten as

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \frac{\dot{\beta}_t}{\beta_t}\boldsymbol{x}_t + \frac{\sigma_t\dot{\alpha}_t - \dot{\sigma}_t\alpha_t}{\beta_t}\boldsymbol{f}_\theta(t, \boldsymbol{x}_t), \tag{63}$$

where $\beta_t = -\alpha_t$ for noise prediction with and $\beta_t = \sigma_t$ for target prediction. This choice of β and $\boldsymbol{f}_\theta$ thus depends on the particulars of the noise or data reparameterization.

Remark C.1. Without loss of generality any of the results for the probability flow ODE apply to any arbitrary flow model which models an *affine probability path* (Lipman et al., 2024) with the correct conversions to the flow matching conventions.[17]

It is well observed that the structure of the ODE in Equation (63) can be greatly simplified via *exponential integrators* (Lu et al., 2022b; Zhang & Chen, 2023; Blasingame & Liu, 2024a). We make use of this insight to rewrite the ODE in a form which eliminates the discretization error in the $f(t)\boldsymbol{x}_t$ linear term along with a time reparameterization which will simplify the construction of the reversible solver. To accomplish this, we use exponential integrators in the form of a change-of-variables with $\boldsymbol{y}_t = \exp(-\int_T^t \frac{\dot{\beta}_u}{\beta_u}\,\mathrm{d}u)\boldsymbol{x}_t = \frac{\beta_T}{\beta_t}\boldsymbol{x}_t$. *N.B.*, we integrate from time T to t because the ODE in Equation (63) is defined in *reverse-time*. To achieve the time reparametrization we introduce a new variable ς_t defined as the *signal-to-noise ratio* (SNR) α_t/σ_t for the data prediction formulation and defined as the inverse SNR σ_t/α_t for the noise prediction formulation. Using this time change we find Proposition C.1, in Appendix C.1.1 we provide the full derivation of this result.

C.1.1 Proof of Proposition C.1

We state Proposition C.1 below.

[17] *I.e.*, sampling in forward-time such that $\boldsymbol{X}_1 \sim q(\boldsymbol{X})$ and $\boldsymbol{X}_0 \sim p(\boldsymbol{X})$.

Proposition C.1. *The probability flow ODE in Equation (63) can be rewritten in ς_t as*

$$\frac{\mathrm{d}\boldsymbol{y}_\varsigma}{\mathrm{d}\varsigma} = |\beta_T| \boldsymbol{f}_\theta\left(\varsigma, \frac{\beta_\varsigma}{\beta_T}\boldsymbol{y}_\varsigma\right), \tag{64}$$

where $\boldsymbol{y}_t = \frac{\beta_T}{\beta_t}\boldsymbol{x}_t$ and $\varsigma_t = \frac{\beta_t^2}{\alpha_t \sigma_t}$.

Proof. Recall that from Equation (63) we have that the ODE is given by

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \frac{\dot{\beta}_t}{\beta_t}\boldsymbol{x}_t + \frac{\sigma_t\dot{\alpha}_t - \dot{\sigma}_t\alpha_t}{\beta_t}\boldsymbol{f}_\theta(t, \boldsymbol{x}_t). \tag{65}$$

We can use the technique of exponential integrators to rewrite the ODE as

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[e^{\int_T^t -\frac{\dot{\beta}_u}{\beta_u}\,\mathrm{d}u}\boldsymbol{x}_t\right] = e^{\int_T^t -\frac{\dot{\beta}_u}{\beta_u}\,\mathrm{d}u}\frac{\sigma_t\dot{\alpha}_t - \dot{\sigma}_t\alpha_t}{\beta_t}\boldsymbol{f}_\theta(t, \boldsymbol{x}_t), \tag{66}$$

recalling that we integrate from initial time T in reverse-time. Then the exponential terms simplify to

$$e^{\int_T^t -\frac{\dot{\beta}_u}{\beta_u}\,\mathrm{d}u} = \frac{\beta_0}{\beta_T}. \tag{67}$$

We introduce a *change-of-variables* $\boldsymbol{y}_t = \frac{\beta_0}{\beta_T}\boldsymbol{x}_t$ to rewrite the ODE as

$$\frac{\mathrm{d}\boldsymbol{y}_t}{\mathrm{d}t} = \underbrace{\frac{\beta_T}{\beta_t}\frac{\sigma_t\dot{\alpha}_t - \dot{\sigma}_t\alpha_t}{\beta_t}}_{=\upsilon_t}\boldsymbol{f}_\theta\left(t, \frac{\beta_T}{\beta_t}\boldsymbol{y}_t\right). \tag{68}$$

Next we define

$$\dot{\varsigma}_t = \mathrm{sgn}(\beta_T)\frac{\sigma_t\dot{\alpha}_t - \sigma\dot{\alpha}_t}{\beta_t^2}, \tag{69}$$

which we will now justify. Now recall that β_t is either $-\alpha_t$ or σ_t depending on the whether $\boldsymbol{f}_\theta$ denotes the data or noise prediction model. Moreover we know that α_t is a strictly monotonically decreasing in t and that σ_t is a strictly monotonically increasing in t. We will now prove that there exists and inverse function for ς_t such that $t_\varsigma(\varsigma_t) = t$ for both cases.

Case $\beta_t = -\alpha_t$. We can write υ_t as

$$\kappa_t = \alpha_T\frac{\dot{\sigma}_t\alpha_t - \sigma_t\dot{\alpha}_t}{\alpha_t^2}, \tag{70}$$

$$\overset{(i)}{=} \alpha_T\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\sigma_t}{\alpha_t}\right), \tag{71}$$

where (i) holds by the quotient role. Clearly, we have that

$$\dot{\varsigma}_t = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\sigma_t}{\alpha_t}\right), \tag{72}$$

$$\varsigma_t = \frac{\sigma_t}{\alpha_t}, \tag{73}$$

It follows from (α_t, σ_t) that ς_t is strictly monotonically increasing in t and thus we can construct its inverse.

Case $\beta_t = \sigma_t$. We can write υ_t as

$$\kappa_t = \sigma_T\frac{\sigma_t\dot{\alpha}_t - \dot{\sigma}_t\alpha_t}{\sigma_t^2}, \tag{74}$$

$$\overset{(i)}{=} \sigma_T\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\alpha_t}{\sigma_t}\right), \tag{75}$$

where (i) holds by the quotient role. Clearly, we have that

$$\dot{\varsigma}_t = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\alpha_t}{\sigma_t}\right), \tag{76}$$

$$\varsigma_t = \frac{\alpha_t}{\sigma_t}, \tag{77}$$

It follows from (α_t, σ_t) that ς_t is strictly monotonically decreasing in t and thus we can construct its inverse.

Thus we can rewrite the ODE via a time-change to find

$$\frac{\mathrm{d}\boldsymbol{y}_\varsigma}{\mathrm{d}\varsigma} = |\beta_T| \boldsymbol{f}_\theta\left(\varsigma, \frac{\beta_T}{\beta_\varsigma}\boldsymbol{y}_\varsigma\right), \tag{78}$$

with the usual *abuse-of-notation* $\boldsymbol{y}_\varsigma \coloneqq \boldsymbol{y}_{t_\varsigma(\varsigma)}$, $\beta_\varsigma \coloneqq \beta_{t_\varsigma(\varsigma)}$, *&c.* □

Remark C.2. When in the noise prediction formulation with Proposition C.1 we recover the following reparameterization of Equation (63)

$$\frac{\mathrm{d}\boldsymbol{z}_\chi}{\mathrm{d}\chi} = \alpha_T \boldsymbol{x}^\theta_{T|\chi}\left(\frac{\alpha_\chi}{\alpha_T}\boldsymbol{z}_\chi\right), \tag{79}$$

where $\alpha_T > 0$, $\boldsymbol{z}_t = \frac{\alpha_T}{\alpha_T}\boldsymbol{x}_t$ and $\chi_t = \frac{\sigma_t}{\alpha_t}$, which has been observed by numerous prior works (see Song et al., 2021a, Equation (14); Pan et al., 2023, Equation (11); Wang et al., 2024, Equation (6)).

Remark C.3. When in the data prediction formulation, Proposition C.1 recovers Blasingame & Liu (2025, Proposition D.2) which states that Equation (63) can be written as

$$\frac{\mathrm{d}\boldsymbol{y}_\gamma}{\mathrm{d}\gamma} = \sigma_T \boldsymbol{x}^\theta_{0|\gamma}\left(\frac{\sigma_\gamma}{\sigma_T}\boldsymbol{y}_\gamma\right), \tag{80}$$

where $\boldsymbol{y}_t = \frac{\sigma_T}{\sigma_t}\boldsymbol{x}_t$ and $\gamma_t = \frac{\alpha_t}{\sigma_t}$.

C.1.2 Data prediction

We present this derivation in the form of Lemma C.2 below.

Lemma C.2 (Rex (ODE) for data prediction models)**.** *Let $\boldsymbol{\Phi}$ be an explicit Runge-Kutta solver for the ODE in Equation (80) with Butcher tableau a_{ij}, b_i, c_i. The reversible solver for $\boldsymbol{\Phi}$ in terms of the original state $\boldsymbol{x}_t$ is given by the forward step*

$$\begin{aligned}\boldsymbol{x}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n}\left(\zeta\boldsymbol{x}_n + (1-\zeta)\hat{\boldsymbol{x}}_n\right) + \sigma_{n+1}\boldsymbol{\Psi}_h(\gamma_n, \hat{\boldsymbol{x}}_n),\\ \hat{\boldsymbol{x}}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n}\hat{\boldsymbol{x}}_n - \sigma_{n+1}\boldsymbol{\Psi}_{-h}(\gamma_{n+1}, \boldsymbol{x}_{n+1}),\end{aligned} \tag{81}$$

and backward step

$$\begin{aligned}\hat{\boldsymbol{x}}_n &= \frac{\sigma_n}{\sigma_{n+1}}\hat{\boldsymbol{x}}_{n+1} + \sigma_n\boldsymbol{\Psi}_{-h}(\gamma_{n+1}, \boldsymbol{x}_{n+1}),\\ \boldsymbol{x}_n &= \frac{\sigma_n}{\sigma_{n+1}}\zeta^{-1}\boldsymbol{x}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{x}}_n - \sigma_n\zeta^{-1}\boldsymbol{\Psi}_h(\gamma_n, \hat{\boldsymbol{x}}_n),\end{aligned} \tag{82}$$

with step size $h \coloneqq \gamma_{n+1} - \gamma_n$ and where $\boldsymbol{\Psi}$ denotes the following scheme

$$\begin{aligned}\hat{\boldsymbol{z}}_i &= \frac{1}{\sigma_n}\boldsymbol{x}_n + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{x}^\theta_{0|\gamma_n+c_jh}(\sigma_{\gamma_n+c_jh}\hat{\boldsymbol{z}}_j),\\ \boldsymbol{\Psi}_h(\gamma_n, \boldsymbol{x}_n) &= h\sum_{i=1}^{s} b_i\boldsymbol{x}^\theta_{0|\gamma_n+c_ih}(\sigma_{\gamma_n+c_ih}\hat{\boldsymbol{z}}_i),\end{aligned} \tag{83}$$

Proof. Recall that the forward step of the McCallum-Foster method for Equation (80) given $\mathbf{\Phi}$ is given as

$$\begin{aligned} \boldsymbol{y}_{n+1} &= \zeta \boldsymbol{y}_n + (1-\zeta)\hat{\boldsymbol{y}}_n + \mathbf{\Phi}_h(\gamma_n, \hat{\boldsymbol{y}}_n), \\ \hat{\boldsymbol{y}}_{n+1} &= \hat{\boldsymbol{y}}_n - \mathbf{\Phi}_{-h}(\gamma_{n+1}, \boldsymbol{y}_{n+1}), \end{aligned} \tag{84}$$

with step size $h = \gamma_{n+1} - \gamma_n$. We use the definition of $\boldsymbol{y}_t = \frac{\sigma_T}{\sigma_t}\boldsymbol{x}_t$ to rewrite the forward pass as

$$\begin{aligned} \boldsymbol{x}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n}\left(\zeta \boldsymbol{x}_n + (1-\zeta)\hat{\boldsymbol{x}}_n\right) + \frac{\sigma_{n+1}}{\sigma_T}\mathbf{\Phi}_h\left(\gamma_n, \frac{\sigma_T}{\sigma_n}\hat{\boldsymbol{x}}_n\right), \\ \hat{\boldsymbol{x}}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n}\hat{\boldsymbol{x}}_n - \frac{\sigma_{n+1}}{\sigma_T}\mathbf{\Phi}_{-h}\left(\gamma_{n+1}, \frac{\sigma_T}{\sigma_{n+1}}\boldsymbol{x}_{n+1}\right). \end{aligned} \tag{85}$$

Mutatis mutandis we find the backward step in $\boldsymbol{x}_t$ to be given as

$$\begin{aligned} \hat{\boldsymbol{x}}_n &= \frac{\sigma_n}{\sigma_{n+1}}\hat{\boldsymbol{x}}_{n+1} + \frac{\sigma_n}{\sigma_T}\mathbf{\Phi}_{-h}\left(\gamma_{n+1}, \frac{\sigma_T}{\sigma_{n+1}}\boldsymbol{x}_{n+1}\right), \\ \boldsymbol{x}_n &= \frac{\sigma_n}{\sigma_{n+1}}\zeta^{-1}\boldsymbol{x}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{x}}_n - \frac{\sigma_n}{\sigma_T}\zeta^{-1}\mathbf{\Phi}_h\left(\gamma_n, \frac{\sigma_T}{\sigma_n}\hat{\boldsymbol{x}}_n\right), \end{aligned} \tag{86}$$

Next we simplify the explicit RK scheme $\mathbf{\Phi}(\gamma_n, \boldsymbol{y}_n)$ for the time-changed probability flow ODE in Equation (80). Recall that the RK scheme can be written as

$$\begin{aligned} \boldsymbol{z}_i &= \boldsymbol{y}_n + h\sum_{j=1}^{i-1} a_{ij}\sigma_T \boldsymbol{x}_{0|\gamma_n + c_j h}\left(\frac{\sigma_{\gamma_n + c_j h}}{\sigma_T}\boldsymbol{z}_j\right), \\ \mathbf{\Phi}_h(\gamma_n, \boldsymbol{y}_n) &= h\sum_{i=1}^{s} b_i \sigma_T \boldsymbol{x}_{0|\gamma_n + c_i h}\left(\frac{\sigma_{\gamma_n + c_i h}}{\sigma_T}\boldsymbol{z}_i\right). \end{aligned} \tag{87}$$

Next, we replace $\boldsymbol{y}_t$ back with $\boldsymbol{x}_t$ which yields

$$\begin{aligned} \boldsymbol{z}_i &= \sigma_T\left(\frac{1}{\sigma_n}\boldsymbol{x}_n + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{x}_{0|\gamma_n + c_j h}\left(\frac{\sigma_{\gamma_n + c_j h}}{\sigma_T}\boldsymbol{z}_j\right)\right), \\ \mathbf{\Phi}_h\left(\gamma_n, \frac{\sigma_T}{\sigma_n}\boldsymbol{x}_n\right) &= \sigma_T h\sum_{i=1}^{s} b_i \boldsymbol{x}_{0|\gamma_n + c_i h}\left(\frac{\sigma_{\gamma_n + c_i h}}{\sigma_T}\boldsymbol{z}_i\right). \end{aligned} \tag{88}$$

To further simplify let $\sigma_T \hat{\boldsymbol{z}}_i = \boldsymbol{z}_i$ and define $\mathbf{\Psi}_h(\gamma_n, \boldsymbol{x}_n) := \sigma_T \mathbf{\Phi}(\gamma_n, \frac{\sigma_T}{\sigma_n}\boldsymbol{x}_n)$.

Thus we can write the following reversible scheme with forward step

$$\begin{aligned} \boldsymbol{x}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n}\left(\zeta \boldsymbol{x}_n + (1-\zeta)\hat{\boldsymbol{x}}_n\right) + \sigma_{n+1}\mathbf{\Psi}_h(\gamma_n, \hat{\boldsymbol{x}}_n), \\ \hat{\boldsymbol{x}}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n}\hat{\boldsymbol{x}}_n - \sigma_{n+1}\mathbf{\Psi}_{-h}(\gamma_{n+1}, \boldsymbol{x}_{n+1}), \end{aligned} \tag{89}$$

and the backward step

$$\begin{aligned} \hat{\boldsymbol{x}}_n &= \frac{\sigma_n}{\sigma_{n+1}}\hat{\boldsymbol{x}}_{n+1} + \sigma_n\mathbf{\Psi}_{-h}(\gamma_{n+1}, \boldsymbol{x}_{n+1}), \\ \boldsymbol{x}_n &= \frac{\sigma_n}{\sigma_{n+1}}\zeta^{-1}\boldsymbol{x}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{x}}_n - \sigma_n\zeta^{-1}\mathbf{\Psi}_h(\gamma_n, \hat{\boldsymbol{x}}_n), \end{aligned} \tag{90}$$

with the numerical scheme

$$\begin{aligned} \hat{\boldsymbol{z}}_i &= \frac{1}{\sigma_n}\boldsymbol{x}_n + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{x}^{\theta}_{0|\gamma_n + c_j h}(\sigma_{\gamma_n + c_j h}\hat{\boldsymbol{z}}_j), \\ \mathbf{\Psi}_h(\gamma_n, \boldsymbol{x}_n) &= h\sum_{i=1}^{s} b_i \boldsymbol{x}^{\theta}_{0|\gamma_n + c_i h}(\sigma_{\gamma_n + c_i h}\hat{\boldsymbol{z}}_i). \end{aligned} \tag{91}$$

□

C.1.3 Noise prediction

We present this derivation in the form of Lemma C.3 below.

Lemma C.3 (Rex (ODE) for noise prediction models). *Let $\mathbf{\Phi}$ be an explicit Runge-Kutta solver for the ODE in Equation (79) with Butcher tableau a_{ij}, b_i, c_i. The reversible solver for $\mathbf{\Phi}$ in terms of the original state $\boldsymbol{x}_t$ is given by the forward step*

$$\begin{aligned}\boldsymbol{x}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n}\left(\zeta\boldsymbol{x}_n + (1-\zeta)\hat{\boldsymbol{x}}_n\right) + \alpha_{n+1}\mathbf{\Psi}_h(\chi_n, \hat{\boldsymbol{x}}_n),\\ \hat{\boldsymbol{x}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n}\hat{\boldsymbol{x}}_n - \alpha_{n+1}\mathbf{\Psi}_{-h}(\chi_{n+1}, \boldsymbol{x}_{n+1}),\end{aligned} \tag{92}$$

and backward step

$$\begin{aligned}\hat{\boldsymbol{x}}_n &= \frac{\alpha_n}{\alpha_{n+1}}\hat{\boldsymbol{x}}_{n+1} + \alpha_n\mathbf{\Psi}_{-h}(\chi_{n+1}, \boldsymbol{x}_{n+1}),\\ \boldsymbol{x}_n &= \frac{\alpha_n}{\alpha_{n+1}}\zeta^{-1}\boldsymbol{x}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{x}}_n - \alpha_n\zeta^{-1}\mathbf{\Psi}_h(\chi_n, \hat{\boldsymbol{x}}_n),\end{aligned} \tag{93}$$

with step size $h \coloneqq \chi_{n+1} - \chi_n$ and where $\mathbf{\Psi}$ denotes the following scheme

$$\begin{aligned}\hat{\boldsymbol{z}}_i &= \frac{1}{\alpha_n}\boldsymbol{x}_n + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{x}^{\theta}_{T|\chi_n+c_jh}(\alpha_{\chi_n+c_jh}\hat{\boldsymbol{z}}_j),\\ \mathbf{\Psi}_h(\chi_n, \boldsymbol{x}_n) &= h\sum_{i=1}^{s} b_i\boldsymbol{x}^{\theta}_{T|\chi_n+c_ih}(\alpha_{\chi_n+c_ih}\hat{\boldsymbol{z}}_i),\end{aligned} \tag{94}$$

Proof. Recall that the forward step of the McCallum-Foster method for Equation (79) given $\mathbf{\Phi}$ is given as

$$\begin{aligned}\boldsymbol{z}_{n+1} &= \zeta\boldsymbol{z}_n + (1-\zeta)\hat{\boldsymbol{z}}_n + \mathbf{\Phi}_h(\chi_n, \hat{\boldsymbol{z}}_n),\\ \hat{\boldsymbol{z}}_{n+1} &= \hat{\boldsymbol{z}}_n - \mathbf{\Phi}_{-h}(\chi_{n+1}, \boldsymbol{z}_{n+1}),\end{aligned} \tag{95}$$

with step size $h = \chi_{n+1} - \chi_n$. We use the definition of $\boldsymbol{z}_t = \frac{\alpha_T}{\alpha_t}\boldsymbol{x}_t$ to rewrite the forward pass as

$$\begin{aligned}\boldsymbol{x}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n}\left(\zeta\boldsymbol{x}_n + (1-\zeta)\hat{\boldsymbol{x}}_n\right) + \frac{\alpha_{n+1}}{\alpha_T}\mathbf{\Phi}_h\left(\chi_n, \frac{\alpha_T}{\alpha_n}\hat{\boldsymbol{x}}_n\right),\\ \hat{\boldsymbol{x}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n}\hat{\boldsymbol{x}}_n - \frac{\alpha_{n+1}}{\alpha_T}\mathbf{\Phi}_{-h}\left(\chi_{n+1}, \frac{\alpha_T}{\alpha_{n+1}}\boldsymbol{x}_{n+1}\right).\end{aligned} \tag{96}$$

Mutatis mutandis we find the backward step in $\boldsymbol{x}_t$ to be given as

$$\begin{aligned}\hat{\boldsymbol{x}}_n &= \frac{\alpha_n}{\alpha_{n+1}}\hat{\boldsymbol{x}}_{n+1} + \frac{\alpha_n}{\alpha_T}\mathbf{\Phi}_{-h}\left(\chi_{n+1}, \frac{\alpha_T}{\alpha_{n+1}}\boldsymbol{x}_{n+1}\right),\\ \boldsymbol{x}_n &= \frac{\alpha_n}{\alpha_{n+1}}\zeta^{-1}\boldsymbol{x}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{x}}_n - \frac{\alpha_n}{\alpha_T}\zeta^{-1}\mathbf{\Phi}_h\left(\chi_n, \frac{\alpha_T}{\alpha_n}\hat{\boldsymbol{x}}_n\right),\end{aligned} \tag{97}$$

Next we simplify the explicit RK scheme $\mathbf{\Phi}(\chi_n, \boldsymbol{z}_n)$ for the time-changed probability flow ODE in Equation (80). Recall that the RK scheme can be written as

$$\begin{aligned}\boldsymbol{z}_i &= \boldsymbol{z}_n + h\sum_{j=1}^{i-1} a_{ij}\alpha_T\boldsymbol{x}_{0|\chi_n+c_jh}\left(\frac{\alpha_{\chi_n+c_jh}}{\alpha_T}\boldsymbol{z}_j\right),\\ \mathbf{\Phi}_h(\chi_n, \boldsymbol{z}_n) &= h\sum_{i=1}^{s} b_i\alpha_T\boldsymbol{x}_{0|\chi_n+c_ih}\left(\frac{\alpha_{\chi_n+c_ih}}{\alpha_T}\boldsymbol{z}_i\right).\end{aligned} \tag{98}$$

Next, we replace $\boldsymbol{z}_t$ back with $\boldsymbol{x}_t$ which yields

$$\begin{aligned}\boldsymbol{z}_i &= \alpha_T\left(\frac{1}{\alpha_n}\boldsymbol{x}_n + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{x}_{0|\chi_n+c_jh}\left(\frac{\alpha_{\chi_n+c_jh}}{\alpha_T}\boldsymbol{z}_j\right)\right),\\ \mathbf{\Phi}_h\left(\chi_n, \frac{\alpha_T}{\alpha_n}\boldsymbol{x}_n\right) &= \alpha_T h\sum_{i=1}^{s} b_i\boldsymbol{x}_{0|\chi_n+c_ih}\left(\frac{\alpha_{\chi_n+c_ih}}{\alpha_T}\boldsymbol{z}_i\right).\end{aligned} \tag{99}$$

To further simplify let $\alpha_T \hat{\boldsymbol{z}}_i = \boldsymbol{z}_i$ and define $\boldsymbol{\Psi}_h(\chi_n, \boldsymbol{x}_n) \coloneqq \alpha_T \boldsymbol{\Phi}(\chi_n, \frac{\alpha_T}{\alpha_n}\boldsymbol{x}_n)$.

Thus we can write the following reversible scheme with forward step

$$\begin{aligned} \boldsymbol{x}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n}\left(\zeta \boldsymbol{x}_n + (1-\zeta)\hat{\boldsymbol{x}}_n\right) + \alpha_{n+1}\boldsymbol{\Psi}_h(\chi_n, \hat{\boldsymbol{x}}_n), \\ \hat{\boldsymbol{x}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n}\hat{\boldsymbol{x}}_n - \alpha_{n+1}\boldsymbol{\Psi}_{-h}(\chi_{n+1}, \boldsymbol{x}_{n+1}), \end{aligned} \tag{100}$$

and the backward step

$$\begin{aligned} \hat{\boldsymbol{x}}_n &= \frac{\alpha_n}{\alpha_{n+1}}\hat{\boldsymbol{x}}_{n+1} + \alpha_n \boldsymbol{\Psi}_{-h}(\chi_{n+1}, \boldsymbol{x}_{n+1}), \\ \boldsymbol{x}_n &= \frac{\alpha_n}{\alpha_{n+1}}\zeta^{-1}\boldsymbol{x}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{x}}_n - \alpha_n\zeta^{-1}\boldsymbol{\Psi}_h(\chi_n, \hat{\boldsymbol{x}}_n), \end{aligned} \tag{101}$$

with the numerical scheme

$$\begin{aligned} \hat{\boldsymbol{z}}_i &= \frac{1}{\alpha_n}\boldsymbol{x}_n + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{x}^{\theta}_{T|\chi_n + c_j h}(\alpha_{\chi_n + c_j h}\hat{\boldsymbol{z}}_j), \\ \boldsymbol{\Psi}_h(\chi_n, \boldsymbol{x}_n) &= h\sum_{i=1}^{s} b_i \boldsymbol{x}^{\theta}_{T|\chi_n + c_i h}(\alpha_{\chi_n + c_i h}\hat{\boldsymbol{z}}_i). \end{aligned} \tag{102}$$

□

C.2 REX (SDE)

In this section we derive the Rex scheme for the reverse-time diffusion SDE along with several helper derivations. We begin by deriving the time-changed Brownian motions before diving into the reparameterization of Equation (6) in **??** before detailing the data prediction scenario in Appendix C.2.2 and then performing an analogous derivation for the noise prediction scenario in Appendix C.2.3.

C.2.1 TIME-CHANGED BROWNIAN MOTION

Before detailing this proof we first review some necessary preliminary results about continuous local martingales and Brownian motion. In particular we will show that we can simplify the stochastic integrals in Equation (6) and the corresponding reparameterization with data/noise prediction models.

Dambis-Dubins-Schwarz representation theorem. We restate the Dambis-Dubins-Schwarz representation theorem (Dubins & Schwarz, 1965) which shows that continuous local martingales can be represented as time-changed Brownian motions.

Theorem C.4 (Dambis-Dubins-Schwarz representation theorem)**.** *Let M be a continuous local martingale adapted to a filtration $\{\mathcal{F}_t\}_{t\geq 0}$ beginning at 0 (*i.e.*, $M_0 = 0$) such that $\langle M\rangle_\infty = \infty$ almost surely. Define the random variables $\{\tau_t\}_{t\geq 0}$ by*

$$\tau_t = \inf\{s \geq 0 : \langle M\rangle_s > t\} = \sup\{s \geq 0 : \langle M\rangle_s = t\}. \tag{103}$$

Then for any given t the random variable τ_t is an almost surely finite stopping time, and the process[a] *$B_t = M_{\tau_t}$ is a Brownian motion w.r.t. the filtration $\{\mathcal{G}_t\}_{t\geq 0} = \{\mathcal{F}_{\tau_t}\}_{t\geq 0}$. Moreover,*

$$M_t = B_{\langle M\rangle_t}. \tag{104}$$

[a] Defined up to a null set.

A multi-dimensional version of the Dambis-Dubins-Schwarz representation theorem. In our work we are interested in a d-dimensional local martingale $\boldsymbol{M} \coloneqq (M^1, \dots M^d)$. As such we discuss a multi-dimensional extension of Theorem C.4 which requires that the d-dimensional continuous local martingale if the quadratic (covariation) matrix $\langle \boldsymbol{M}\rangle_t^{ij} = \left\langle M^i, M^j\right\rangle_t$ is proportional to the identity matrix. We adapt the following theorem from Lowther (2010, Theorem 2) and Bourgade (2010, Theorem 4.13) (*cf*. Revuz & Yor, 2013).

Theorem C.5 (Multi-dimensional Dambis-Dubins-Schwarz representation theorem)**.** *Let $\boldsymbol{M} = (M^1, \ldots, M^d)$ be a collection of continuous local martingales with $\boldsymbol{M}_0 = \mathbf{0}$ such that for any $1 \leq 1 \leq d$, $\langle \boldsymbol{M} \rangle_\infty^{ii} = \infty$ almost surely. Suppose, furthermore, that $\langle M^i, M^j \rangle_t = \delta_{ij} A_t$, where δ denotes the Kronecker delta, for some process A and all $1 \leq i, j \leq d$ and $t \geq 0$. Then there is a d-dimensional Brownian motion $\boldsymbol{B}$ w.r.t. a filtration $\{\mathcal{G}_t\}_{t \geq 0}$ such that for each $t \geq 0$, $\omega \mapsto A_t(\omega)$ is a $\mathcal{G}$-stopping time and*

$$\boldsymbol{M}_t = \boldsymbol{B}_{A_t}. \tag{105}$$

Enlargement of the probability space. Recall that in Theorems C.4 and C.5 we stated that quadratic variation of the continuous local martingale needed to tend towards infinity as $t \to \infty$. What when $\langle M \rangle_\infty$ has a nonzero probability of being finite? It can be shown that Theorems C.4 and C.5 holds under an enlargement of the probability space (not the filtration). Consider both our original probability space $(\Omega, \mathcal{F}, P)$ and another probability space $(\Omega', \mathcal{F}', P')$ along with a measurable surjection $f : \Omega' \to \Omega$ preserving probabilities such that $P(A) = P'(f^{-1}(A))$ for all $A \in \mathcal{F}$, *i.e.*, $f_* P'$ is a pushforward measure. Thus any process on the original probability space can be *lifted* to $(\Omega', \mathcal{F}', P')$ and likewise the filtration is also lifted to $\mathcal{F}'_t = \{f^{-1}(A) : A \in \mathcal{F}_t\}$. Therefore, it is possible to enlarge the probability space so that Brownian motion is defined. *E.g.*, if $(\Omega'', \mathcal{F}'', P'')$ is probability space on which there is a Brownian motion defined, we can take $\Omega' = \Omega \times \Omega''$, $\mathcal{F}' = \mathcal{F} \otimes \mathcal{F}''$, and $P' = P \otimes P''$ for the enlargement, and $f :' \Omega \to \Omega$ is just the projection onto Ω.

Data prediction. We now present a lemma for rewriting the stochastic differential in Equation (126) using the Dambis-Dubins-Schwarz representation theorem. Recall that in Equation (126) we denote the reverse-time d-dimensional Brownian motion as $\overline{\boldsymbol{W}}_t$, *i.e.*, by Lévy's characterization we have $\overline{\boldsymbol{W}}_T = \mathbf{0}$ and

$$\overline{\boldsymbol{W}}_t - \overline{\boldsymbol{W}}_s \sim -\mathcal{N}(\mathbf{0}, (t-s)\boldsymbol{I}) = \mathcal{N}(\mathbf{0}, (t-s)\boldsymbol{I}), \tag{106}$$

for $0 \leq t < s \leq T$. With this in mind we present Lemma C.6 below.

Lemma C.6. *The stochastic differential $\sqrt{-\frac{\mathrm{d}\varrho_t}{\mathrm{d}t}}\, \mathrm{d}\overline{\boldsymbol{W}}_t$ can be rewritten as a time-changed Brownian motion of the form*

$$\sqrt{-\frac{\mathrm{d}\varrho_t}{\mathrm{d}t}}\, \mathrm{d}\overline{\boldsymbol{W}}_t = \mathrm{d}\boldsymbol{W}_\varrho, \tag{107}$$

where $\varrho_t = \gamma_t^2$.

Proof. To simplify the stochastic integral term we first define a continuous local martingale $\boldsymbol{M}_t$ via the stochastic integral

$$\boldsymbol{M}_t := \int_T^t \sqrt{-\frac{\mathrm{d}\varrho}{\mathrm{d}t}}\, \mathrm{d}\overline{\boldsymbol{W}}_t. \tag{108}$$

We choose time T as our starting point for the martingale rather than 0 and then integrate in *reverse-time*. However, due to the negative sign within the square root term it is more convenient to work with $\boldsymbol{W}_t$, *i.e.*, the standard d-dimensional Brownian motion defined in forward time. Recall that the standard d-dimensional Brownian motion in *reverse-time* with starting point T is defined as

$$\overline{\boldsymbol{W}}_t = \boldsymbol{W}_T - \boldsymbol{W}_t \tag{109}$$

which is distributed like $\boldsymbol{W}_t$ in time $T - t$. Define the function $\boldsymbol{f}(t, \boldsymbol{W}_t) = \overline{\boldsymbol{W}}_t$. Then by Itô's lemma we have

$$\mathrm{d}\boldsymbol{f}(t, \boldsymbol{W}_t) = \partial_t \boldsymbol{f}(t, \boldsymbol{W}_t)\, \mathrm{d}t + \sum_{i=1}^d \partial_{\boldsymbol{x}_i} \boldsymbol{f}(t, \boldsymbol{W}_t)\, \mathrm{d}\boldsymbol{W}_t^i + \sum_{i,j=1}^d \partial_{\boldsymbol{x}_i, \boldsymbol{x}_j} \boldsymbol{f}(t, \boldsymbol{W}_t)\, \mathrm{d}\left\langle \boldsymbol{W}^i, \boldsymbol{W}^j \right\rangle_t, \tag{110}$$

which simplifies to

$$\mathrm{d}\boldsymbol{f}(t, \boldsymbol{W}_t) = \mathrm{d}\overline{\boldsymbol{W}}_t = -\mathrm{d}\boldsymbol{W}_t. \tag{111}$$

Thus we can rewrite Equation (108) as

$$\boldsymbol{M}_t = -\int_T^t \sqrt{-\frac{\mathrm{d}\varrho}{\mathrm{d}t}}\,\mathrm{d}\boldsymbol{W}_t. \tag{112}$$

Next we establish a few properties of this martingale. First, $\boldsymbol{M}_T = \mathbf{0}$ by construction. Second, since the integral consists of scalar noise we have that $\langle M^i, M^j\rangle_t = 0$ for all $i \neq j$. Thus, the quadratic variation of $\langle \boldsymbol{M}_t\rangle^{ii}$ for each i is found to be

$$\langle \boldsymbol{M}\rangle_t^{ii} = A_t = -\int_T^t \left(\sqrt{-\frac{\mathrm{d}\varrho_\tau}{\mathrm{d}\tau}}\right)^2 \mathrm{d}\tau, \tag{113}$$

$$= \int_T^t \frac{\mathrm{d}\varrho_\tau}{\mathrm{d}\tau}\,\mathrm{d}\tau, \tag{114}$$

$$= \varrho_t - \varrho_T = \frac{\alpha_t^2}{\sigma_t^2} - \frac{\alpha_T^2}{\sigma_T^2}. \tag{115}$$

Now we have a deterministic mapping from the original time to our new time via A_t. Now in general for any valid choice of (α_t, σ_t) we don't necessarily have that $\langle \boldsymbol{M}\rangle_\infty^{ii} = \infty$ almost surely and as such we may need to enlarge the underlying probability space. Our constructed martingale can be expressed as time-changed Brownian motion, see Theorem C.5, such that $\boldsymbol{M}_t = \boldsymbol{W}_{A_t}$ were $\boldsymbol{W}_\varrho$ is the standard d-dimensional Brownian motion with time variable ϱ.

Now we can rewrite Equation (112) in differential form as

$$\mathrm{d}\boldsymbol{M}_t = \mathrm{d}\boldsymbol{W}_{A_t}. \tag{116}$$

Because Brownian motion is time-shift invariant we can then write

$$\mathrm{d}\boldsymbol{M}_t = \mathrm{d}\boldsymbol{W}_{\varrho_t}. \tag{117}$$

□

Remark C.4. Lemma C.6 can similarly be found via Øksendal (2003, Theorem 8.5.7) and Kobayashi (2011, Lemma 2.3); however, do to the oddness of the *reverse-time* integration we found it easier to tackle the problem via the Dambis-Dubins-Schwarz theorem.

Remark C.5. Under the common scenario where $\sigma_0 = 0$ then we have that $\langle \boldsymbol{M}\rangle_\infty^{ii} = \infty$ almost surely.

Noise prediction. Next we discuss the stochastic integral used in the noise prediction formulation.

Lemma C.7. *Let $\alpha_T > 0$. Then the stochastic differential $\sqrt{\frac{\mathrm{d}}{\mathrm{d}t}(\chi_t^2)}\,\mathrm{d}\overline{\boldsymbol{W}}_t$ can be rewritten as a time-changed Brownian motion of the form*

$$\sqrt{\frac{\mathrm{d}}{\mathrm{d}t}(\chi_t^2)}\,\mathrm{d}\overline{\boldsymbol{W}}_t = \mathrm{d}\overline{\boldsymbol{W}}_{\chi^2}, \tag{118}$$

where $\chi_t = \frac{\sigma_t}{\alpha_t}$.

Proof. To simplify the stochastic integral term we first define a continuous local martingale $\boldsymbol{M}_t$ via the stochastic integral

$$\boldsymbol{M}_t \coloneqq \int_T^t \sqrt{\frac{\mathrm{d}}{\mathrm{d}t}(\chi_t^2)}\,\mathrm{d}\overline{\boldsymbol{W}}_t. \tag{119}$$

We choose time T as our starting point for the martingale rather than 0 and then integrate in *reverse-time*, hence the negative sign. Next we establish a few properties of this martingale. First, $\boldsymbol{M}_T = \mathbf{0}$ by construction. Second, since the integral consists of scalar noise we have that $\langle M^i, M^j\rangle_t = 0$ for

all $i \neq j$. Thus, the quadratic variation of $\langle \boldsymbol{M}_t \rangle^{ii}$ for each i is found to be

$$\langle \boldsymbol{M} \rangle_t^{ii} = A_t = \int_T^t \left(\sqrt{\frac{\mathrm{d}}{\mathrm{d}\tau} (\chi_\tau^2)} \right)^2 \mathrm{d}\tau, \tag{120}$$

$$= \int_T^t \frac{\mathrm{d}}{\mathrm{d}\tau} (\chi_t^2) \ \mathrm{d}\tau, \tag{121}$$

$$= \chi_t^2 - \chi_T^2 = \frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_T^2}{\alpha_T^2}. \tag{122}$$

Now we have a deterministic mapping from the original time to our new time via A_t. Now in general for any valid choice of (α_t, σ_t) we don't necessarily have that $\langle \boldsymbol{M} \rangle_\infty^{ii} = \infty$ almost surely and as such we may need to enlarge the underlying probability space. Our constructed martingale can be expressed as time-changed Brownian motion, see Theorem C.5, such that $\boldsymbol{M}_t = \overline{\boldsymbol{W}}_{A_t}$ were $\overline{\boldsymbol{W}}_{\chi^2}$ is the standard d-dimensional Brownian motion with time variable χ^2 in reverse-time.

Now we can rewrite Equation (108) in differential form as

$$\mathrm{d}\boldsymbol{M}_t = \mathrm{d}\overline{\boldsymbol{W}}_{A_t}. \tag{123}$$

Because Brownian motion is time-shift invariant we can then write

$$\mathrm{d}\boldsymbol{M}_t = \mathrm{d}\overline{\boldsymbol{W}}_{\chi_t^2}. \tag{124}$$

□

Remark C.6. The constraint of $\alpha_T > 0$ is important to ensure that χ_T is finite which is necessary due

$$\overline{\boldsymbol{W}}_{\chi_t^2} = \boldsymbol{W}_{\chi_T^2} - \boldsymbol{W}_{\chi_t^2}. \tag{125}$$

In practice this is satisfied with a number of noise schedules of diffusion models (*cf*. Appendix G.1).

C.2.2 Proof of reparametrized SDE for data prediction models

It is well known (Lu et al., 2022a) that the reverse-time diffusion SDE in Equation (18) can be rewritten in terms of the data prediction model as

$$\mathrm{d}\boldsymbol{X}_t = \left[\left(f(t) + \frac{g^2(t)}{\sigma_t^2} \right) \boldsymbol{X}_t - \frac{\alpha_t g^2(t)}{\sigma_t^2} \boldsymbol{x}_{0|t}^\theta(\boldsymbol{X}_t) \right] \mathrm{d}t + g(t) \ \mathrm{d}\overline{\boldsymbol{W}}_t. \tag{126}$$

Remarkably, following a similar derivation to the one above for the probability flow ODE yields a time-changed SDE with a very similar form to the one above, sans the Brownian motion term and different weighting terms. We present this result in Lemma C.8 with the full proof in Appendix C.2.2.

Lemma C.8 (Time reparametrization of the reverse-time diffusion SDE)**.** *The reverse-time SDE in Equation* (126) *can be rewritten in terms of the data prediction model as*

$$\mathrm{d}\boldsymbol{Y}_\varrho = \frac{\sigma_T}{\gamma_T} \boldsymbol{x}_{0|\varrho}^\theta \left(\frac{\gamma_T \sigma_\varrho}{\sigma_T \gamma_\varrho} \boldsymbol{Y}_\varrho \right) \mathrm{d}\varrho + \frac{\sigma_T}{\gamma_T} \mathrm{d}\boldsymbol{W}_\varrho, \tag{127}$$

where $\boldsymbol{Y}_t = \frac{\sigma_T^2 \alpha_t}{\sigma_t^2 \alpha_T} \boldsymbol{X}_t$ *and* $\varrho_t \coloneqq \frac{\alpha_t^2}{\sigma_t^2}$.

Proof. We rewrite Equation (18) in terms of the data prediction model, using the identity

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) = -\frac{1}{\sigma_t^2} \boldsymbol{x} + \frac{\alpha_t}{\sigma_t^2} \boldsymbol{x}_{0|t}(\boldsymbol{x}), \tag{128}$$

to find

$$\mathrm{d}\boldsymbol{X}_t = \left[\underbrace{\left(f(t) + \frac{g^2(t)}{\sigma_t^2} \right)}_{=a(t)} \boldsymbol{X}_t + \underbrace{\left(-\frac{\alpha_t g^2(t)}{\sigma_t^2} \right)}_{=b(t)} \boldsymbol{x}_{0|t}(\boldsymbol{X}_t) \right] \mathrm{d}t + g(t) \ \mathrm{d}\overline{\boldsymbol{W}}_t, \tag{129}$$

where

$$f(t) = \frac{\dot{\alpha}_t}{\alpha_t}, \quad g^2(t) = \dot{\sigma}_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t}\sigma_t^2 = -2\sigma_t^2 \frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}. \tag{130}$$

Next we find the integrating factor $\Xi_t = \exp - \int_T^t a(u)\,\mathrm{d}u$,

$$\begin{aligned}
\Xi_t &= \exp\left(\int_t^T \frac{\mathrm{d}\log\alpha_u}{\mathrm{d}u} + \frac{g^2(u)}{\sigma_u^2}\,\mathrm{d}u\right), && (131)\\
&= \exp\left(\int_t^T \frac{\mathrm{d}\log\alpha_u}{\mathrm{d}u} - 2\frac{\mathrm{d}\log\gamma_u}{\mathrm{d}u}\,\mathrm{d}u\right), && (132)\\
&= \exp\left(\int_t^T \frac{\mathrm{d}\log\alpha_u}{\mathrm{d}u} - 2\left[\frac{\mathrm{d}\log\alpha_u}{\mathrm{d}u} - \frac{\mathrm{d}\log\sigma_u}{\mathrm{d}u}\right]\,\mathrm{d}u\right), && (133)\\
&= \exp\left(\int_t^T \frac{\mathrm{d}\log\sigma_u^2}{\mathrm{d}u} - \frac{\mathrm{d}\log\alpha_u}{\mathrm{d}u}\,\mathrm{d}u\right), && (134)\\
&= \exp\left(\log\sigma_T^2 - \log\sigma_t^2 - (\log\alpha_T - \log\alpha_t)\right), && (135)\\
&= \frac{\sigma_T^2\alpha_t}{\sigma_t^2\alpha_T}. && (136)
\end{aligned}$$

We can write the integrating factor in terms of γ_t as

$$\Xi_t = \frac{\sigma_T\gamma_t}{\sigma_t\gamma_T}. \tag{137}$$

Moreover we can further simplify $b(t)$ as

$$\begin{aligned}
b(t) &= \frac{-\alpha_t g^2(t)}{\sigma_t^2}, && (138)\\
&= 2\alpha_t \frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}. && (139)
\end{aligned}$$

Thus we can rewrite the SDE in Equation (129) as

$$\begin{aligned}
\mathrm{d}\left[\frac{\sigma_T}{\gamma_T}\frac{\gamma_t}{\sigma_t}\boldsymbol{X}_t\right] &= 2\frac{\sigma_T}{\gamma_T}\frac{\alpha_t}{\sigma_t}\gamma_t\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}\boldsymbol{x}_{0|t}(\boldsymbol{X}_t)\,\mathrm{d}t + \frac{\sigma_T}{\gamma_T}\frac{\gamma_t}{\sigma_t}\sqrt{-2\sigma_t^2\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (140)\\
\mathrm{d}\boldsymbol{Y}_t &\overset{(i)}{=} 2\frac{\sigma_T}{\gamma_T}\frac{\alpha_t}{\sigma_t}\gamma_t\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}\boldsymbol{x}_{0|t}\left(\frac{\gamma_T\sigma_t}{\sigma_T\gamma_t}\boldsymbol{Y}_t\right)\,\mathrm{d}t + \frac{\sigma_T}{\gamma_T}\frac{\gamma_t}{\sigma_t}\sqrt{-2\sigma_t^2\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (141)\\
\mathrm{d}\boldsymbol{Y}_t &= \frac{\sigma_T}{\gamma_T}\frac{\mathrm{d}\gamma_t^2}{\mathrm{d}t}\boldsymbol{x}_{0|t}\left(\frac{\gamma_T\sigma_t}{\sigma_T\gamma_t}\boldsymbol{Y}_t\right)\,\mathrm{d}t + \frac{\sigma_T}{\gamma_T}\sqrt{-\gamma_t^2\frac{\mathrm{d}\log\gamma_t^2}{\mathrm{d}t}}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (142)\\
\mathrm{d}\boldsymbol{Y}_t &= \frac{\sigma_T}{\gamma_T}\frac{\mathrm{d}\gamma_t^2}{\mathrm{d}t}\boldsymbol{x}_{0|t}\left(\frac{\gamma_T\sigma_t}{\sigma_T\gamma_t}\boldsymbol{Y}_t\right)\,\mathrm{d}t + \frac{\sigma_T}{\gamma_T}\sqrt{-\frac{\mathrm{d}\gamma_t^2}{\mathrm{d}t}}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (143)\\
\mathrm{d}\boldsymbol{Y}_\varrho &\overset{(ii)}{=} \frac{\sigma_T}{\gamma_T}\boldsymbol{x}_{0|\varrho}\left(\frac{\gamma_T\sigma_\varrho}{\sigma_T\gamma_\varrho}\boldsymbol{Y}_\varrho\right)\,\mathrm{d}\varrho + \frac{\sigma_T}{\gamma_T}\,\mathrm{d}\boldsymbol{W}_\varrho, && (144)
\end{aligned}$$

where (i) holds by the change-of-variables $\boldsymbol{Y}_t = \frac{\sigma_T\gamma_t}{\gamma_T\sigma_t}\boldsymbol{X}_t$ and (ii) holds by Lemma C.6. □

C.2.3 Proof of reparametrized SDE for noise prediction models

Lemma C.9 (Time reparametrization of the reverse-time diffusion SDE for noise prediction models)**.** *The reverse-time SDE in Equation (18) can be rewritten in terms of the noise prediction model as*

$$\mathrm{d}\boldsymbol{Y}_\chi = 2\alpha_T\boldsymbol{x}_{T|\chi}^\theta\left(\frac{\alpha_\chi}{\alpha_T}\boldsymbol{Y}_\chi\right)\,\mathrm{d}\chi + \alpha_T\,\mathrm{d}\overline{\boldsymbol{W}}_{\chi^2}, \tag{145}$$

where $\boldsymbol{Y}_t = \frac{\alpha_t}{\alpha_T}\boldsymbol{X}_t$ *and* $\chi_t \coloneqq \frac{\sigma_t}{\alpha_t}$.

Proof. We rewrite Equation (18) in terms of the noise prediction model to find

$$\mathrm{d}\boldsymbol{X}_t = \left[f(t)\boldsymbol{X}_t + \frac{g^2(t)}{\sigma_t}\boldsymbol{x}^\theta_{T|t}(\boldsymbol{X}_t)\right]\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{\boldsymbol{W}}_t, \tag{146}$$

where

$$f(t) = \frac{\dot\alpha_t}{\alpha_t}, \quad g^2(t) = \dot\sigma_t^2 - 2\frac{\dot\alpha_t}{\alpha_t}\sigma_t^2 = -2\sigma_t^2\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}. \tag{147}$$

Next we find the integrating factor to be $\exp -\int_T^t f(u)\,\mathrm{d}u = \frac{\alpha_T}{\alpha_t}$. Moreover, we can further simplify $\frac{g^2(t)}{\sigma_t}$ as

$$\begin{aligned}
\frac{g^2(t)}{\sigma_t} &= -2\sigma_t\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}, && (148)\\
&= -2\sigma_t\frac{\dot\gamma_t}{\gamma_t}, && (149)\\
&= -2\frac{\sigma_t}{\gamma_t}\frac{\dot\alpha_t\sigma_t - \alpha_t\dot\sigma_t}{\sigma_t^2}, && (150)\\
&= -2\frac{\sigma_t^2}{\alpha_t}\frac{\dot\alpha_t\sigma_t - \alpha_t\dot\sigma_t}{\sigma_t^2}, && (151)\\
&= 2\frac{\sigma_t^2}{\alpha_t}\frac{\alpha_t\dot\sigma_t - \dot\alpha_t\sigma_t}{\sigma_t^2}, && (152)\\
&= 2\frac{\alpha_t\dot\sigma_t - \dot\alpha_t\sigma_t}{\alpha_t}, && (153)\\
& && (154)
\end{aligned}$$

Let $\chi_t \coloneqq \frac{\sigma_t}{\alpha_t} = \frac{1}{\gamma_t}$. Thus we can rewrite the SDE in Equation (146) as

$$\begin{aligned}
\mathrm{d}\left[\frac{\alpha_T}{\alpha_t}\boldsymbol{X}_t\right] &= \frac{\alpha_T}{\alpha_t}\frac{g^2(t)}{\sigma_t^2}\boldsymbol{x}^\theta_{T|t}(\boldsymbol{X}_t)\,\mathrm{d}t + \frac{\alpha_T}{\alpha_t}\sqrt{-2\sigma_t^2\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (155)\\
\mathrm{d}\boldsymbol{Y}_t &\overset{(i)}{=} \frac{\alpha_T}{\alpha_t}\frac{g^2(t)}{\sigma_t^2}\boldsymbol{x}^\theta_{T|t}\left(\frac{\alpha_t}{\alpha_T}\boldsymbol{Y}_t\right)\,\mathrm{d}t + \frac{\alpha_T}{\alpha_t}\sqrt{-2\sigma_t^2\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (156)\\
\mathrm{d}\boldsymbol{Y}_t &= 2\alpha_T\frac{\alpha_t\dot\sigma_t - \dot\alpha_t\sigma_t}{\alpha_t^2}\boldsymbol{x}^\theta_{T|t}\left(\frac{\alpha_t}{\alpha_T}\boldsymbol{Y}_t\right)\,\mathrm{d}t + \frac{\alpha_T}{\alpha_t}\sqrt{-2\sigma_t^2\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (157)\\
\mathrm{d}\boldsymbol{Y}_t &\overset{(ii)}{=} 2\alpha_T\dot\chi_t\boldsymbol{x}^\theta_{T|t}\left(\frac{\alpha_t}{\alpha_T}\boldsymbol{Y}_t\right)\,\mathrm{d}t + \alpha_T\sqrt{-2\frac{\sigma_t^2}{\alpha_t^2}\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t}}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (158)\\
\mathrm{d}\boldsymbol{Y}_t &= 2\alpha_T\dot\chi_t\boldsymbol{x}^\theta_{T|t}\left(\frac{\alpha_t}{\alpha_T}\boldsymbol{Y}_t\right)\,\mathrm{d}t + \alpha_T\sqrt{\dot\chi_t^2}\,\mathrm{d}\overline{\boldsymbol{W}}_t, && (159)\\
\mathrm{d}\boldsymbol{Y}_\chi &\overset{(iii)}{=} 2\alpha_T\boldsymbol{x}^\theta_{T|\chi}\left(\frac{\alpha_\chi}{\alpha_T}\boldsymbol{Y}_\chi\right)\,\mathrm{d}\chi + \alpha_T\,\mathrm{d}\overline{\boldsymbol{W}}_{\chi^2}, && (160)\\
& && (161)
\end{aligned}$$

where (i) holds by the change-of-variables $\boldsymbol{Y}_t = \frac{\alpha_T}{\alpha_t}\boldsymbol{X}_t$, (ii) holds by

$$\begin{aligned}
-2\frac{\sigma_t^2}{\alpha_t^2}\frac{\mathrm{d}\log\gamma_t}{\mathrm{d}t} &= \frac{\sigma_t^2}{\alpha_t^2}\frac{\mathrm{d}(-2\log\gamma_t)}{\mathrm{d}t}, && (162)\\
&= \frac{\sigma_t^2}{\alpha_t^2}\frac{\mathrm{d}\log\chi_t^2}{\mathrm{d}t}, && (163)\\
&= \frac{\sigma_t^2}{\alpha_t^2}\frac{\dot\chi_t^2}{\chi_t^2}, && (164)\\
&= \dot\chi_t^2, && (165)
\end{aligned}$$

and (iii) holds by Lemma C.6 *mutatis mutandis* for χ_t. □

C.2.4 Derivation of Rex (SDE)

We present derivations for both the data prediction and noise prediction formulations.

Data prediction. We present this derivation in the form of Lemma C.10 below.

Lemma C.10 (Rex (SDE) for data prediction models)**.** *Let $\boldsymbol{\Phi}$ be an explicit stochastic Runge-Kutta solver for the additive noise SDE in Equation (127), we construct the following reversible scheme for diffusion models*

$$\begin{aligned}\boldsymbol{X}_{n+1} &= \frac{\sigma_{n+1}\gamma_n}{\gamma_{n+1}\sigma_n}(\zeta\boldsymbol{X}_n + (1-\zeta)\hat{\boldsymbol{X}}_n) + \frac{\sigma_{n+1}}{\gamma_{n+1}}\boldsymbol{\Psi}_h(\varrho_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_\varrho(\omega)),\\ \hat{\boldsymbol{X}}_{n+1} &= \frac{\sigma_{n+1}\gamma_n}{\gamma_{n+1}\sigma_n}\hat{\boldsymbol{X}}_n - \frac{\sigma_{n+1}}{\gamma_{n+1}}\boldsymbol{\Psi}_{-h}(\varrho_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_\varrho(\omega)),\end{aligned} \tag{166}$$

and the backward step is given as

$$\begin{aligned}\hat{\boldsymbol{X}}_n &= \frac{\sigma_n\gamma_{n+1}}{\gamma_n\sigma_{n+1}}\hat{\boldsymbol{X}}_n + \frac{\sigma_n}{\gamma_n}\boldsymbol{\Psi}_{-h}(\varrho_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_\varrho(\omega)),\\ \boldsymbol{X}_n &= \frac{\sigma_n\gamma_{n+1}}{\gamma_n\sigma_{n+1}}\zeta^{-1}\boldsymbol{X}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{X}}_n - \frac{\sigma_n}{\gamma_n}\zeta^{-1}\boldsymbol{\Psi}_h(\varrho_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_\varrho(\omega)),\end{aligned} \tag{167}$$

with step size $h \coloneqq \varrho_{n+1} - \varrho_n$ and where $\boldsymbol{\Psi}$ denotes the following scheme

$$\begin{aligned}\hat{\boldsymbol{Z}}_i &= \frac{\gamma_n}{\sigma_n}\boldsymbol{X}_n + h\sum_{j=1}^{i-1}\left[a_{ij}\boldsymbol{x}_{0|\varrho_n+c_jh}\left(\frac{\sigma_{\varrho_n+c_jh}}{\gamma_{\varrho_n+c_jh}}\hat{\boldsymbol{Z}}_j\right)\right] + a_i^W\boldsymbol{W}_n + a_i^H\boldsymbol{H}_n,\\ \boldsymbol{\Psi}_h(\varrho_n, \boldsymbol{X}_n, \boldsymbol{W}_\varrho(\omega)) &= h\sum_{j=1}^{s}\left[b_i\boldsymbol{x}_{0|\varrho_n+c_ih}\left(\frac{\sigma_{\varrho_n+c_ih}}{\gamma_{\varrho_n+c_jh}}\hat{\boldsymbol{Z}}_j\right)\right] + b^W\boldsymbol{W}_n + b^H\boldsymbol{H}_n.\end{aligned} \tag{168}$$

Proof. We write the SRK scheme for the time-changed reverse-time SDE in Equation (127) to construct the following SRK scheme

$$\begin{aligned}\boldsymbol{Z}_i &= \boldsymbol{Y}_n + h\sum_{j=1}^{i-1}\left[a_{ij}\frac{\sigma_T}{\gamma_T}\boldsymbol{x}_{0|\varrho_n+c_jh}\left(\frac{\gamma_T\sigma_{\varrho_n+c_jh}}{\sigma_T\gamma_{\varrho_n+c_jh}}\boldsymbol{Z}_j\right)\right] + \frac{\sigma_T}{\gamma_T}(a_i^W\boldsymbol{W}_n + a_i^H\boldsymbol{H}_n),\\ \boldsymbol{Y}_{n+1} &= \boldsymbol{Y}_n + h\sum_{i=1}^{s}\left[b_i\frac{\sigma_T}{\gamma_T}\boldsymbol{x}_{0|\varrho_n+c_ih}\left(\frac{\gamma_T\sigma_{\varrho_n+c_ih}}{\sigma_T\gamma_{\varrho_n+c_ih}}\boldsymbol{Z}_i\right)\right] + \frac{\sigma_T}{\gamma_T}(b^W\boldsymbol{W}_n + b^H\boldsymbol{H}_n),\end{aligned} \tag{169}$$

with step size $h = \varrho_{n+1} - \varrho_n$. Next, we replace $\boldsymbol{Y}_t$ back with $\boldsymbol{X}_t$ which yields

$$\begin{aligned}\boldsymbol{Z}_i &= \frac{\sigma_T}{\gamma_T}\left(\frac{\gamma_n}{\sigma_n}\boldsymbol{X}_n + h\sum_{j=1}^{i-1}\left[a_{ij}\boldsymbol{x}_{0|\varrho_n+c_jh}\left(\frac{\gamma_T\sigma_{\varrho_n+c_jh}}{\sigma_T\gamma_{\varrho_n+c_jh}}\boldsymbol{Z}_j\right)\right]\right)\\ &\quad + \frac{\sigma_T}{\gamma_T}(a_i^W\boldsymbol{W}_n + a_i^H\boldsymbol{H}_n),\\ \frac{\sigma_T\gamma_{n+1}}{\gamma_T\sigma_{n+1}}\boldsymbol{X}_{n+1} &= \frac{\sigma_T\gamma_n}{\gamma_T\sigma_n}\boldsymbol{X}_n\\ &\quad + \frac{\sigma_T}{\gamma_T}\underbrace{h\sum_{i=1}^{s}\left[b_i\frac{\sigma_T}{\gamma_T}\boldsymbol{x}_{0|\varrho_n+c_ih}\left(\frac{\gamma_T\sigma_{\varrho_n+c_ih}}{\sigma_T\gamma_{\varrho_n+c_ih}}\boldsymbol{Z}_i\right)\right] + \frac{\sigma_T}{\gamma_T}(b^W\boldsymbol{W}_n + b^H\boldsymbol{H}_n)}_{=\boldsymbol{\Psi}_h(\varrho_n, \boldsymbol{X}_n, \boldsymbol{W}_\varrho)}.\end{aligned} \tag{170}$$

To further simplify let $\frac{\sigma_T}{\gamma_T}\hat{\boldsymbol{Z}}_i = \boldsymbol{Z}_i$, then we construct the reversible scheme with forward pass:

$$\begin{aligned}\boldsymbol{X}_{n+1} &= \frac{\sigma_{n+1}\gamma_n}{\gamma_{n+1}\sigma_n}(\zeta\boldsymbol{X}_n + (1-\zeta)\hat{\boldsymbol{X}}_n) + \frac{\sigma_{n+1}}{\gamma_{n+1}}\boldsymbol{\Psi}_h(\varrho_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_\varrho(\omega)),\\ \hat{\boldsymbol{X}}_{n+1} &= \frac{\sigma_{n+1}\gamma_n}{\gamma_{n+1}\sigma_n}\hat{\boldsymbol{X}}_n - \frac{\sigma_{n+1}}{\gamma_{n+1}}\boldsymbol{\Psi}_{-h}(\varrho_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_\varrho(\omega)),\end{aligned} \tag{171}$$

and backward pass

$$\begin{aligned}\hat{\boldsymbol{X}}_n &= \frac{\sigma_n\gamma_{n+1}}{\gamma_n\sigma_{n+1}}\hat{\boldsymbol{X}}_n + \frac{\sigma_n}{\gamma_n}\boldsymbol{\Psi}_{-h}(\varrho_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_\varrho(\omega)),\\ \hat{\boldsymbol{X}}_{n+1} &= \frac{\sigma_n\gamma_{n+1}}{\gamma_n\sigma_{n+1}}\zeta^{-1}\boldsymbol{X}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{X}}_n - \frac{\sigma_n}{\gamma_n}\zeta^{-1}\boldsymbol{\Psi}_h(\varrho_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_\varrho(\omega)),\end{aligned} \tag{172}$$

with step size $h \coloneqq \varrho_{n+1} - \varrho_n$

$$\begin{aligned}\hat{\boldsymbol{Z}}_i &= \frac{\gamma_n}{\sigma_n}\boldsymbol{X}_n + h\sum_{j=1}^{i-1}\left[a_{ij}\boldsymbol{x}_{0|\varrho_n+c_jh}\left(\frac{\sigma_{\varrho_n+c_jh}}{\gamma_{\varrho_n+c_jh}}\hat{\boldsymbol{Z}}_j\right)\right] + a_i^W\boldsymbol{W}_n + a_i^H\boldsymbol{H}_n,\\ \boldsymbol{\Psi}_h(\varrho_n, \boldsymbol{X}_n, \boldsymbol{W}_\varrho(\omega)) &= h\sum_{j=1}^{s}\left[b_i\boldsymbol{x}_{0|\varrho_n+c_ih}\left(\frac{\sigma_{\varrho_n+c_ih}}{\gamma_{\varrho_n+c_jh}}\hat{\boldsymbol{Z}}_j\right)\right] + b^W\boldsymbol{W}_n + b^H\boldsymbol{H}_n.\end{aligned} \tag{173}$$

□

Noise prediction. We present this derivation in the form of Lemma C.11 below.

Lemma C.11 (Rex (SDE) for noise prediction models)**.** *Let $\boldsymbol{\Phi}$ be an explicit stochastic Runge-Kutta solver for the additive noise SDE in Equation (145), we construct the following reversible scheme for diffusion models*

$$\begin{aligned}\boldsymbol{X}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n}(\zeta\boldsymbol{X}_n + (1-\zeta)\hat{\boldsymbol{X}}_n) + \alpha_{n+1}\boldsymbol{\Psi}_h(\chi_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_{\chi^2}(\omega)),\\ \hat{\boldsymbol{X}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n}\hat{\boldsymbol{X}}_n - \alpha_{n+1}\boldsymbol{\Psi}_{-h}(\chi_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_{\chi^2}(\omega)),\end{aligned} \tag{174}$$

and the backward step is given as

$$\begin{aligned}\hat{\boldsymbol{X}}_n &= \frac{\alpha_n}{\alpha_{n+1}}\hat{\boldsymbol{X}}_n + \alpha_n\boldsymbol{\Psi}_{-h}(\chi_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_{\chi^2}(\omega)),\\ \boldsymbol{X}_n &= \frac{\alpha_n}{\alpha_n+1}\zeta^{-1}\boldsymbol{X}_{n+1} + (1-\zeta^{-1})\hat{\boldsymbol{X}}_n - \alpha_n\zeta^{-1}\boldsymbol{\Psi}_h(\chi_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_{\chi^2}(\omega)),\end{aligned} \tag{175}$$

with step size $h \coloneqq \chi_{n+1} - \chi_n$ and where $\boldsymbol{\Psi}$ denotes the following scheme

$$\begin{aligned}\hat{\boldsymbol{Z}}_i &= \frac{1}{\alpha_n}\boldsymbol{X}_n + h\sum_{j=1}^{i-1}\left[2a_{ij}\boldsymbol{x}^\theta_{T|\chi_n+c_jh}\left(\alpha_{\chi_n+c_jh}\hat{\boldsymbol{Z}}_j\right)\right] + a_i^W\boldsymbol{W}_n + a_i^H\boldsymbol{H}_n,\\ \boldsymbol{\Psi}_h(\chi_n, \boldsymbol{X}_n, \boldsymbol{W}_\chi(\omega)) &= h\sum_{j=1}^{s}\left[2b_i\boldsymbol{x}^\theta_{T|\chi_n+c_ih}\left(\alpha_{\chi_n+c_ih}\hat{\boldsymbol{Z}}_j\right)\right] + b^W\boldsymbol{W}_n + b^H\boldsymbol{H}_n.\end{aligned} \tag{176}$$

Proof. We write the SRK scheme for the time-changed reverse-time SDE in Equation (145) to construct the following SRK scheme

$$\begin{aligned}\boldsymbol{Z}_i &= \boldsymbol{Y}_n + h\sum_{j=1}^{i-1}\left[2a_{ij}\alpha_T\boldsymbol{x}_{T|\chi_n+c_jh}\left(\frac{\alpha_{\chi_n+c_jh}}{\alpha_T}\boldsymbol{Z}_j\right)\right] + \alpha_T(a_i^W\boldsymbol{W}_n + a_i^H\boldsymbol{H}_n),\\ \boldsymbol{Y}_{n+1} &= \boldsymbol{Y}_n + h\sum_{i=1}^{s}\left[2b_i\alpha_T\boldsymbol{x}_{T|\chi_n+c_ih}\left(\frac{\alpha_{\chi_n+c_ih}}{\alpha_T}\boldsymbol{Z}_i\right)\right] + \alpha_T(b^W\boldsymbol{W}_n + b^H\boldsymbol{H}_n),\end{aligned} \tag{177}$$

with step size $h = \chi_{n+1} - \chi_n$. Next, we replace $\boldsymbol{Y}_t$ back with $\boldsymbol{X}_t$ which yields

$$\begin{aligned}
\boldsymbol{Z}_i &= \alpha_T \left(\frac{1}{\alpha_n} \boldsymbol{X}_n + h \sum_{j=1}^{i-1} \left[2a_{ij} \boldsymbol{x}_{T|\chi_n + c_j h} \left(\frac{\alpha_{\chi_n + c_j h}}{\alpha_T} \boldsymbol{Z}_j \right) \right] \right) \\
&\quad + \alpha_T (a_i^W \boldsymbol{W}_n + a_i^H \boldsymbol{H}_n), \\
\frac{\alpha_{n+1}}{\alpha_T} \boldsymbol{X}_{n+1} &= \frac{\alpha_T}{\alpha_n} \boldsymbol{X}_n \\
&\quad + \alpha_T \underbrace{h \sum_{i=1}^{s} \left[2b_i \alpha_T \boldsymbol{x}_{T|\chi_n + c_i h} \left(\frac{\alpha_{\chi_n + c_i h}}{\alpha_T} \boldsymbol{Z}_i \right) \right] + \alpha_T (b^W \boldsymbol{W}_n + b^H \boldsymbol{H}_n)}_{= \boldsymbol{\Psi}_h(\chi_n, \boldsymbol{X}_n, \boldsymbol{W}_\chi)} .
\end{aligned} \tag{178}$$

To further simplify let $\alpha_T \hat{\boldsymbol{Z}}_i = \boldsymbol{Z}_i$, then we construct the reversible scheme with forward pass:

$$\begin{aligned}
\boldsymbol{X}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} (\zeta \boldsymbol{X}_n + (1-\zeta) \hat{\boldsymbol{X}}_n) + \alpha_{n+1} \boldsymbol{\Psi}_h(\chi_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_\chi(\omega)), \\
\hat{\boldsymbol{X}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} \hat{\boldsymbol{X}}_n - \alpha_{n+1} \boldsymbol{\Psi}_{-h}(\chi_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_\chi(\omega)),
\end{aligned} \tag{179}$$

and backward pass

$$\begin{aligned}
\hat{\boldsymbol{X}}_n &= \frac{\alpha_n}{\alpha_{n+1}} \hat{\boldsymbol{X}}_n + \alpha_n \boldsymbol{\Psi}_{-h}(\chi_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_\chi(\omega)), \\
\hat{\boldsymbol{X}}_{n+1} &= \frac{\alpha_n}{\alpha_{n+1}} \zeta^{-1} \boldsymbol{X}_{n+1} + (1-\zeta^{-1}) \hat{\boldsymbol{X}}_n - \alpha_n \zeta^{-1} \boldsymbol{\Psi}_h(\chi_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_\chi(\omega)),
\end{aligned} \tag{180}$$

with step size $h \coloneqq \chi_{n+1} - \chi_n$

$$\begin{aligned}
\hat{\boldsymbol{Z}}_i &= \frac{\gamma_n}{\sigma_n} \boldsymbol{X}_n + h \sum_{j=1}^{i-1} \left[2a_{ij} \boldsymbol{x}_{T|\chi_n + c_j h} \left(\alpha_{\chi_n + c_j h} \hat{\boldsymbol{Z}}_j \right) \right] + a_i^W \boldsymbol{W}_n + a_i^H \boldsymbol{H}_n, \\
\boldsymbol{\Psi}_h(\chi_n, \boldsymbol{X}_n, \boldsymbol{W}_\chi(\omega)) &= h \sum_{j=1}^{s} \left[2b_i \boldsymbol{x}_{T|\chi_n + c_i h} \left(\alpha_{\chi_n + c_i h} \hat{\boldsymbol{Z}}_j \right) \right] + b^W \boldsymbol{W}_n + b^H \boldsymbol{H}_n.
\end{aligned} \tag{181}$$

N.B., $\boldsymbol{W}_n = \overline{\boldsymbol{W}}_{\chi_{n+1}^2} - \overline{\boldsymbol{W}}_{\chi_n^2}$. □

C.3 Proof of Proposition 3.1

We now can construct Rex.

Proposition C.12 (Rex). *Without loss of generality let $\boldsymbol{\Phi}$ denote an explicit SRK scheme for the SDE in Equation (127) with extended Butcher tableau $a_{ij}, b_i, c_i, a_i^W, a_i^H, b^W, b^H$. Fix an $\omega \in \Omega$ and let $\boldsymbol{W}$ be the Brownian motion over time variable ς. Then the reversible solver constructed from $\boldsymbol{\Phi}$ in terms of the underlying state variable $\boldsymbol{X}_t$ is given by the forward step*

$$\begin{aligned}
\boldsymbol{X}_{n+1} &= \frac{w_{n+1}}{w_n} \left(\zeta \boldsymbol{X}_n + (1-\zeta) \hat{\boldsymbol{X}}_n \right) + w_{n+1} \boldsymbol{\Psi}_h(\varsigma_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_n(\omega)), \\
\hat{\boldsymbol{X}}_{n+1} &= \frac{w_{n+1}}{w_n} \hat{\boldsymbol{X}}_n - w_{n+1} \boldsymbol{\Psi}_{-h}(\varsigma_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_n(\omega)),
\end{aligned} \tag{182}$$

and backward step

$$\begin{aligned}
\hat{\boldsymbol{X}}_n &= \frac{w_n}{w_{n+1}} \hat{\boldsymbol{X}}_{n+1} + w_n \boldsymbol{\Psi}_{-h}(\varsigma_{n+1}, \boldsymbol{X}_{n+1}, \boldsymbol{W}_n(\omega)), \\
\boldsymbol{X}_n &= \frac{w_n}{w_{n+1}} \zeta^{-1} \boldsymbol{X}_{n+1} + (1-\zeta^{-1}) \hat{\boldsymbol{X}}_n - w_n \zeta^{-1} \boldsymbol{\Psi}_h(\varsigma_n, \hat{\boldsymbol{X}}_n, \boldsymbol{W}_n(\omega)),
\end{aligned} \tag{183}$$

with step size $h := \varsigma_{n+1} - \varsigma_n$ *and where* $\boldsymbol{\Psi}$ *denotes the following scheme*

$$\hat{\boldsymbol{Z}}_i = \frac{1}{w_n}\boldsymbol{X}_n + h\sum_{j=1}^{i-1}\left[a_{ij}\boldsymbol{f}_\theta\left(\varsigma_n + c_j h, w_{\varsigma_n + c_j h}\hat{\boldsymbol{Z}}_j\right)\right] + a_i^W \boldsymbol{W}_n(\omega) + a_i^H \boldsymbol{H}_n(\omega),$$

$$\boldsymbol{\Psi}_h(\varsigma_n, \boldsymbol{X}_n, \boldsymbol{W}_\varrho(\omega)) = h\sum_{j=1}^{s}\left[b_i\boldsymbol{f}_\theta\left(\varsigma_n + c_i h, w_{\varsigma_n + c_i h}\hat{\boldsymbol{Z}}_j\right)\right] + b^W \boldsymbol{W}_n(\omega) + b^H \boldsymbol{H}_n(\omega), \tag{184}$$

where $\boldsymbol{f}_\theta$ *denotes the data prediction model,* $w_n = \frac{\sigma_n}{\gamma_n}$ *and* $\varsigma_t = \varrho_t$*. The ODE case is recovered for an explicit RK scheme* $\boldsymbol{\Phi}$ *for the ODE in Equation (80) with* $w_n = \sigma_n$ *and* $\varsigma_t = \gamma_t$ *For noise prediction models we have* $\boldsymbol{f}_\theta$ *denoting the noise prediction model with* $w_n = \alpha_n$ *and* $\varsigma_t = \frac{\sigma_n}{\alpha_n}$*.*

Proof. This follows by Lemmas C.2, C.3, C.10 and C.11 *mutatis mutandis*. □

D Convergence order proofs

D.1 Assumptions

Beyond the general regularity conditions imposed on the learned diffusion model itself (see Lu et al., 2022b; Blasingame & Liu, 2024a; 2025) we also assert that in the noise prediction setting that $\alpha_T > 0$. In practice most commonly used diffusion noise schedules like the linear or scaled linear schedule satisfy this, (see Appendix G.1; *cf*. Lin et al., 2024).

D.2 Proof of Theorem 3.2

Theorem 3.2 (Rex is a k-th order solver)**.** *Let* $\boldsymbol{\Phi}$ *be a* k*-th order explicit Runge-Kutta scheme for the reparameterized probability flow ODE in Equation (80) with variance preserving noise schedule* (α_t, σ_t)*. Then Rex constructed from* $\boldsymbol{\Phi}$ *is a* k*-th order solver,* i.e.*, given the reversible solution* $\{\boldsymbol{x}_n, \hat{\boldsymbol{x}}_n\}_{n=1}^N$ *and true solution* $\boldsymbol{x}_{t_n}$ *we have*

$$\|\boldsymbol{x}_n - \boldsymbol{x}_{t_n}\| \leq Ch^k, \tag{13}$$

for constants $C, h_{max} > 0$ *and for step sizes* $h \in [0, h_{max}]$*.*

Proof. We will prove this for both the data prediction and noise prediction formulations.

Data prediction. By Theorem A.1 we have that reversible $\boldsymbol{\Phi}$ is a k-th order solver, and thus

$$\|\boldsymbol{y}_n - \boldsymbol{y}_{t_n}\| \leq Ch^k. \tag{185}$$

We use the change of variables from Equation (80) to find

$$\left\|\frac{\sigma_T}{\sigma_n}\boldsymbol{x}_n - \frac{\sigma_T}{\sigma_n}\boldsymbol{x}_{t_n}\right\| \leq Ch^k, \tag{186}$$

which simplifies to

$$\|\boldsymbol{x}_n - \boldsymbol{x}_{t_n}\| \leq \frac{\sigma_n}{\sigma_T}Ch^k. \tag{187}$$

Now by definition for variance preserving type diffusion SDEs we have that $\sigma_t \leq 1$ for all t. Thus we can write

$$\|\boldsymbol{x}_n - \boldsymbol{x}_{t_n}\| \leq C_1 h^k, \tag{188}$$

where $C_1 = \frac{C}{\sigma_T}$.

Noise prediction. By Theorem A.1 we have that reversible $\boldsymbol{\Phi}$ is a k-th order solver, and thus

$$\|\boldsymbol{y}_n - \boldsymbol{y}_{t_n}\| \leq Ch^k. \tag{189}$$

We use the change of variables from Equation (79) to find

$$\left\| \frac{\alpha_T}{\alpha_n} \boldsymbol{x}_n - \frac{\alpha_T}{\alpha_n} \boldsymbol{x}_{t_n} \right\| \leq Ch^k, \tag{190}$$

which simplifies to

$$\|\boldsymbol{x}_n - \boldsymbol{x}_{t_n}\| \leq \frac{\alpha_n}{\alpha_T} Ch^k. \tag{191}$$

Now by definition we have $\alpha_t \leq 1$ for all t and we assume that $\alpha_T > 0$. Thus we can write

$$\|\boldsymbol{x}_n - \boldsymbol{x}_{t_n}\| \leq C_1 h^k, \tag{192}$$

where $C_1 = \frac{C}{\sigma_T}$. □

D.3 Proof of Theorem 3.3

Definition D.1 (Strong order of convergence). Suppose an SDE solver admits a numerical solution $\boldsymbol{X}_n$ and we have a true solution $\boldsymbol{X}_{t_n}$. If

$$\sup_{0 \leq n \leq N} \mathbb{E}\|\boldsymbol{X}_n - \boldsymbol{X}_{t_n}\|^2 \leq Ch^{2\alpha}, \tag{193}$$

where $C > 0$ is a constant and h is the step size, then the SDE solver strongly converges with order α.

Theorem 3.3 (Convergence order for *Princeps*). *Let $\boldsymbol{\Phi}$ be a SRK scheme with strong order of convergence $\xi > 0$ for the reparameterized reverse-time diffusion SDE in Equation (127) with variance preserving noise schedule (α_t, σ_t) and $\alpha_T > 0$. Then $\boldsymbol{\Psi}$ constructed from $\boldsymbol{\Phi}$ has strong order of convergence ξ.*

Proof. We will prove this for both the data prediction and noise prediction formulations.

Data prediction. By definition we have $\boldsymbol{\Phi}$ has strong order of convergence ξ and thus,

$$\sup_{0 \leq n \leq N} \mathbb{E}\|\boldsymbol{Y}_n - \boldsymbol{Y}_{t_n}\|^2 \leq Ch^{2\xi}, \tag{194}$$

where $h = \frac{\sigma_{n+1}^2}{\alpha_{n+1}} - \frac{\sigma_n^2}{\alpha_n}$. We use the change of variables from Equation (127) to find

$$\sup_{0 \leq n \leq N} \mathbb{E} \left\| \frac{\sigma_T^2 \alpha_n}{\sigma_n^2 \alpha_T} \boldsymbol{X}_n - \frac{\sigma_T^2 \alpha_n}{\sigma_n^2 \alpha_T} \boldsymbol{X}_{t_n} \right\|^2 \leq Ch^{2\xi}, \tag{195}$$

which simplifies to

$$\sup_{0 \leq n \leq N} \mathbb{E}\|\boldsymbol{X}_n - \boldsymbol{X}_{t_n}\|^2 \leq \frac{\sigma_n \sqrt{\alpha_T}}{\sigma_T \sqrt{\alpha_n}} Ch^{2\xi}. \tag{196}$$

Since by definition of α_n is a monotonically decreasing function, σ_n is a monotonically increasing function, $\alpha_T > 0$, and $\sigma_T \leq 1$ we can write

$$\sup_{0 \leq n \leq N} \mathbb{E}\|\boldsymbol{X}_n - \boldsymbol{X}_{t_n}\|^2 \leq Ch^{2\xi}, \tag{197}$$

as

$$\frac{\sigma_n \sqrt{\alpha_T}}{\sigma_T \sqrt{\alpha_n}} \leq 1. \tag{198}$$

Noise prediction. By definition we have $\boldsymbol{\Phi}$ has strong order of convergence ξ and thus,

$$\sup_{0 \leq n \leq N} \mathbb{E}\|\boldsymbol{Y}_n - \boldsymbol{Y}_{t_n}\|^2 \leq Ch^{2\xi}, \tag{199}$$

where $h = \frac{\sigma_{n+1}}{\alpha_{n+1}} - \frac{\sigma_n}{\alpha_n}$. We use the change of variables from Equation (145) to find

$$\sup_{0 \leq n \leq N} \mathbb{E} \left\| \frac{\alpha_n}{\alpha_T} \boldsymbol{X}_n - \frac{\alpha_n}{\alpha_T} \boldsymbol{X}_{t_n} \right\|^2 \leq Ch^{2\xi}, \tag{200}$$

which simplifies to

$$\sup_{0\leq n\leq N} \mathbb{E}\|\boldsymbol{X}_n - \boldsymbol{X}_{t_n}\|^2 \leq \frac{\sqrt{\alpha_T}}{\sqrt{\alpha_n}} C h^{2\xi}. \tag{201}$$

Since by definition of α_n is a monotonically decreasing function strictly less than 1 and $\alpha_T > 0$ we can write

$$\sup_{0\leq n\leq N} \mathbb{E}\|\boldsymbol{X}_n - \boldsymbol{X}_{t_n}\|^2 \leq C h^{2\xi}. \tag{202}$$

□

E Relation to other solvers for diffusion models

While this paper primarily focused on Rex and the family of reversible solvers created by it, we wish to discuss the relation between the underlying scheme $\boldsymbol{\Psi}$ constructed from our method and other existing solvers for diffusion models.

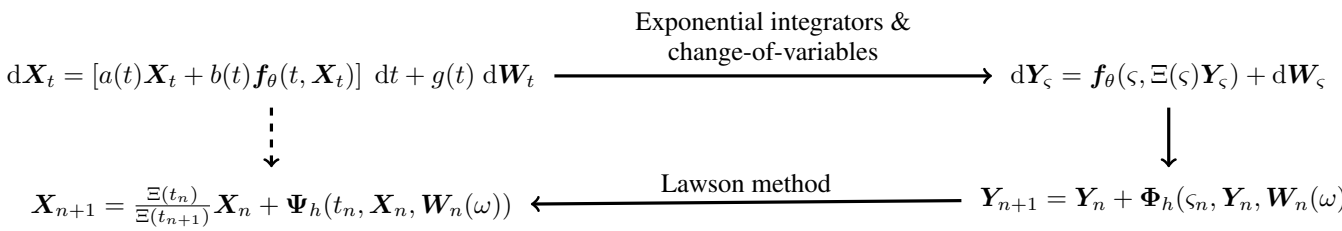


Figure 5: Overview of the construction of $\boldsymbol{\Psi}$ for the probability flow ODE from an underlying RK scheme $\boldsymbol{\Phi}$ for the reparameterized ODE. This graph holds for the SDE case *mutatis mutandis*.

Surprisingly, we discover that using Lawson methods outlined in Figure 5 (*cf*. Figure 2 from the main paper) is a surprisingly generalized methodology for construing numerical schemes for diffusion modes, and that it subsumes previous works. This means that several of the reversible schemes we presented here are reversible variants of well known schemes in the literature in diffusion models.

Theorem 3.4 (*Princeps* subsumes previous solvers)**.** *Princeps subsumes the following solvers for diffusion models*

1. *DDIM (Song et al., 2021a),*
2. *DPM-Solver-1, DPM-Solver-2, DPM-Solver-12 (Lu et al., 2022b),*
3. *DPM-Solver++1, DPM-Solver++(2S), SDE-DPM-Solver-1, SDE-DPM-Solver++1 (Lu et al., 2022a),*
4. *SEEDS-1 (Gonzalez et al., 2024), and*
5. *gDDIM (Zhang et al., 2023).*

Proof. We prove the connection to each solver in the list within a set of separate propositions for easier readability. The statement holds true via Propositions E.1 to E.8 and Corollaries E.1.1 to E.6.1. □

Corollary E.0.1. *Rex is the reversible revision of the well-known solvers for diffusion models in Theorem 3.4.*

Remark E.1. The SDE solvers constructed from Foster-Reis-Strange SRK schemes are wholly unique (with the exception of the trivial Euler-Maruyama scheme) and have no existing counterpart in the literature in diffusion models. Thus Rex (ShARK) is not only a novel reversible solver, but a novel solver for diffusion models in general.

E.1 REX AS REVERSIBLE ODE SOLVERS

Here we discuss Rex as reversible versions for well-known numerical schemes for diffusion models. Recall that the general Butcher tableau for a s-stage explicit RK scheme (Stewart, 2022, Section 6.1.4) is written as

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{(s-1)s} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} = \begin{array}{c|c} c & a \\ \hline & b \end{array}. \tag{203}$$

Embedded methods for adaptive step sizing are of the form

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{(s-1)s} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \\ & b_1^* & b_2^* & \cdots & b_{s-1}^* & b_s^* \end{array}, \tag{204}$$

where the lower-order step is given by the coefficients b_i^*.

E.1.1 EULER

In this section we explore the numerical schemes produced by choosing the Euler scheme for $\mathbf{\Phi}$. The Butcher tableau for the Euler method is

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}. \tag{205}$$

Proposition E.1 (Rex (Euler) is reversible DPM-Solver++1)**.** *The underlying scheme of Rex (Euler) for the data prediction parameterization of diffusion models in Equation (80) is the DPM-Solver++1 from Lu et al. (2022a).*

Proof. Apply in the Butcher tableau for the Euler scheme to $\mathbf{\Psi}$ constructed from Equation (79) to find

$$\boldsymbol{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n}\boldsymbol{x}_n + \sigma_{n+1} h \boldsymbol{x}^{\theta}_{0|\gamma_n}(\boldsymbol{x}_n), \tag{206}$$

with $h = \gamma_{n+1} - \gamma_n$. We can rewrite the step size as

$$\begin{aligned}
\sigma_{n+1} h &= \sigma_{n+1} \left(\frac{\alpha_{n+1}}{\sigma_{n+1}} - \frac{\alpha_n}{\sigma_n} \right), && (207)\\
&= \left(\alpha_{n+1} - \alpha_n \frac{\sigma_{n+1}}{\sigma_n} \right), && (208)\\
&= \left(\alpha_{n+1} \frac{\alpha_{n+1}}{\alpha_{n+1}} - \frac{\alpha_n}{\alpha_{n+1}} \frac{\sigma_{n+1}}{\sigma_n} \right), && (209)\\
&= -\alpha_{n+1} \left(\frac{\alpha_n}{\alpha_{n+1}} \frac{\sigma_{n+1}}{\sigma_n} - 1 \right), && (210)\\
&= -\alpha_{n+1} \left(\frac{\gamma_n}{\gamma_{n+1}} - 1 \right), && (211)\\
&= -\alpha_{n+1} \left(e^{\log \frac{\gamma_n}{\gamma_{n+1}}} - 1 \right), && (212)\\
&= -\alpha_{n+1} \left(e^{\log \gamma_n - \log \gamma_{n+1}} - 1 \right), && (213)\\
&\overset{(i)}{=} -\alpha_{n+1} \left(e^{\lambda_n - \lambda_{n+1}} - 1 \right), && (214)\\
&\overset{(ii)}{=} -\alpha_{n+1} \left(e^{-h_\lambda} - 1 \right), && (215)
\end{aligned}$$

where (i) holds by the letting $\lambda_t = \log \gamma_t$ following the notation of Lu et al. (2022b;a) and (ii) holds by letting $h_\lambda = \lambda_{n+1} - \lambda_n$. Plugging this back into Equation (206) yields

$$\boldsymbol{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} \boldsymbol{x}_n - \alpha_{n+1} \left(e^{-h_\lambda} - 1 \right) \boldsymbol{x}^\theta_{0|t_n}(\boldsymbol{x}_n), \tag{216}$$

which is the DPM-Solver++1 from Lu et al. (2022a). □

Corollary E.1.1 (Rex (Euler) is reversible deterministic DDIM for data prediction models)**.** *The underlying scheme of Rex (Euler) for the data prediction parameterization of diffusion models in Equation (80) is the deterministic DDIM solver from Song et al. (2021a).*

Proof. This holds because DPM-Solver++1 is DDIM see Lu et al. (2022a, Equation (21)) with $\eta = 0$. □

Proposition E.2 (Rex (Euler) is reversible DPM-Solver-1)**.** *The underlying scheme of Rex (Euler) for the data prediction parameterization of diffusion models in Equation (79) is the DPM-Solver-1 from Lu et al. (2022b, Equation (3.7)).*

Proof. Apply in the Butcher tableau for the Euler scheme to $\boldsymbol{\Psi}$ from Rex (see Proposition 3.1) to find

$$\boldsymbol{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n} \boldsymbol{x}_n + \alpha_{n+1} h \boldsymbol{x}^\theta_{T|\chi_n}(\boldsymbol{x}_n), \tag{217}$$

with $h = \chi_{n+1} - \chi_n$. We can rewrite step size as

$$\begin{aligned}
\alpha_{n+1} h &= \alpha_{n+1} \left(\frac{\sigma_{n+1}}{\alpha_{n+1}} - \frac{\sigma_n}{\alpha_n} \right), && (218)\\
&= \left(\sigma_{n+1} - \sigma_n \frac{\alpha_{n+1}}{\alpha_n} \right), && (219)\\
&= \left(\sigma_{n+1} \frac{\sigma_{n+1}}{\sigma_{n+1}} - \frac{\sigma_n}{\sigma_{n+1}} \frac{\alpha_{n+1}}{\alpha_n} \right), && (220)\\
&= -\sigma_{n+1} \left(\frac{\sigma_n}{\sigma_{n+1}} \frac{\alpha_{n+1}}{\alpha_n} - 1 \right), && (221)\\
&= -\sigma_{n+1} \left(\frac{\chi_n}{\chi_{n+1}} - 1 \right), && (222)\\
&= -\sigma_{n+1} \left(e^{\log \frac{\chi_n}{\chi_{n+1}}} - 1 \right), && (223)\\
&= -\sigma_{n+1} \left(e^{\log \chi_n - \log \chi_{n+1}} - 1 \right), && (224)\\
&\overset{(i)}{=} -\sigma_{n+1} \left(e^{-\lambda_n + \lambda_{n+1}} - 1 \right), && (225)\\
&\overset{(ii)}{=} -\sigma_{n+1} \left(e^{h_\lambda} - 1 \right), && (226)
\end{aligned}$$

where (i) holds by the letting $\lambda_t = \log \gamma_t = -\log \chi_t$ following the notation of Lu et al. (2022b;a) and (ii) holds by letting $h_\lambda = \lambda_{n+1} - \lambda_n$. Plugging this back into Equation (206) yields

$$\boldsymbol{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n} \boldsymbol{x}_n - \sigma_{n+1} \left(e^{h_\lambda} - 1 \right) \boldsymbol{x}^\theta_{T|t_n}(\boldsymbol{x}_n), \tag{227}$$

which is the DPM-Solver-1 from Lu et al. (2022b). □

Corollary E.2.1 (Rex (Euler) is reversible deterministic DDIM for noise prediction models)**.** *The underlying scheme of Rex (Euler) for the noise prediction parameterization of diffusion models in Equation* (79) *is the deterministic DDIM solver from Song et al. (2021a).*

Proof. This holds because DPM-Solver-1 is DDIM see Lu et al. (2022b, Equation (4.1)). □

E.1.2 Second-order methods

In this section we explore the numerical schemes produced by choosing the explicit midpoint method for $\boldsymbol{\Phi}$. We can write a generic second-order method as

$$\begin{array}{c|cc}
0 & & \\
\eta & \eta & \\
\hline
 & 1 - \frac{1}{2\eta} & \frac{1}{2\eta}
\end{array}, \tag{228}$$

for $\eta \neq 0$ (Butcher, 2016). The choice of $\eta = \frac{1}{2}$ yields the explicit midpoint, $\eta = \frac{2}{3}$ gives Ralston's second-order method, and $\eta = 1$ gives Heun's second-order method.

Proposition E.3 (Rex (generic second-order) is reversible DPM-Solver++(2S))**.** *The underlying scheme of Rex (generic second-order) for the data prediction parameterization of diffusion models in Equation* (80) *is the DPM-Solver++(2S) from Lu et al. (2022a, Algorithm 1).*

Proof. The DPM-Solver++(2S) (Lu et al., 2022a, Algorithm 1) is defined as

$$\begin{aligned}
\boldsymbol{u} &= \frac{\sigma_p}{\sigma_n} \boldsymbol{x}_n - \alpha_p \left(e^{-r_\lambda h_\lambda} - 1 \right) \boldsymbol{x}^\theta_{0|t_n}(\boldsymbol{x}_n),\\
\boldsymbol{D} &= \left(1 - \frac{1}{2r_\lambda} \right) \boldsymbol{x}^\theta_{0|t_n}(\boldsymbol{x}_n) + \frac{1}{2r_\lambda} \boldsymbol{x}^\theta_{0|t_p}(\boldsymbol{u}), && (229)\\
\boldsymbol{x}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n} \boldsymbol{x}_n - \alpha_{n+1} \left(e^{-h_\lambda} - 1 \right) \boldsymbol{D},
\end{aligned}$$

for some intermediate timestep $t_n > t_p > t_{n+1}$ and with $r_\lambda = \frac{\lambda_p - \lambda_n}{\lambda_{n+1} - \lambda_n}$. Notice that r_λ describes the location of the midpoint time in the λ-domain as a ratio, *i.e.*, we could say

$$\lambda_p = \lambda_n + r_\lambda h_\lambda, \tag{230}$$

where $r_\lambda \in (0, 1)$ denotes the interpolation point between the initial timestep λ_n and terminal timestep λ_{n+1}. Thus we fix $\eta = r_\lambda$ as the step size ratio of the intermediate point.

Now we return to the underlying scheme of Rex applied to the generic second-order scheme, see Equation (228), Apply in the Butcher tableau for generic second-order scheme to $\boldsymbol{\Psi}$ constructed from Equation (79) to find

$$\begin{aligned} \boldsymbol{z} &= \frac{1}{\sigma_n}\boldsymbol{x}_n + \eta h \boldsymbol{x}^\theta_{0|\gamma_n}(\boldsymbol{x}_n), \\ \boldsymbol{x}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n}\boldsymbol{x}_n + \sigma_{n+1} h \left(\left(1 - \frac{1}{2\eta}\right) \boldsymbol{x}^\theta_{0|\gamma_n}(\boldsymbol{x}_n) + \frac{1}{2\eta}\boldsymbol{x}^\theta_{0|\gamma_n+\eta h}(\sigma_p \boldsymbol{z}) \right), \end{aligned} \tag{231}$$

with $h = \gamma_{n+1} - \gamma_n$ and $\sigma_p = \sigma_{\gamma_n + \eta h}$ with $\gamma_p = \gamma_n + \eta h$. We can write

$$\sigma_p \boldsymbol{z} = \frac{\sigma_p}{\sigma_n}\boldsymbol{x}_n + \sigma_p \eta h \boldsymbol{x}^\theta_{0|\gamma_n}(\boldsymbol{x}_n). \tag{232}$$

Plugging this back into Equation (231) yields

$$\begin{aligned} \sigma_p \boldsymbol{z} &= \frac{\sigma_p}{\sigma_n}\boldsymbol{x}_n + \sigma_p \eta h \boldsymbol{x}^\theta_{0|\gamma_n}(\boldsymbol{x}_n), \\ \boldsymbol{x}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n}\boldsymbol{x}_n + \sigma_{n+1} h \underbrace{\left(\left(1 - \frac{1}{2\eta}\right) \boldsymbol{x}^\theta_{0|\gamma_n}(\boldsymbol{x}_n) + \frac{1}{2\eta}\boldsymbol{x}^\theta_{0|\gamma_n+\eta h}(\sigma_p \boldsymbol{z}) \right)}_{=\hat{\boldsymbol{D}}}, \end{aligned} \tag{233}$$

which is the DPM-Solver++1 from Lu et al. (2022a). Now recall from Proposition E.1 that

$$\sigma_{n+1} h = -\alpha_{n+1} \left(e^{-h_\lambda} - 1) \right), \tag{234}$$

it follows that

$$\sigma_p \eta h = -\alpha_p \left(e^{-r_\lambda h_\lambda} - 1 \right), \tag{235}$$

due to $\lambda_p - \lambda_n = r_\lambda h_\lambda$ and $\eta h = \lambda_p - \lambda_n$. Thus by letting $\sigma_p \boldsymbol{z} = \boldsymbol{u}$ and $\hat{\boldsymbol{D}} = \boldsymbol{D}$ we recover the DPM-Solver++(2S) solver. □

Proposition E.4 (Rex (generic second-order) is reversible DPM-Solver-2)**.** *The underlying scheme of Rex (generic second-order) for the noise prediction parameterization of diffusion models in Equation* (79) *is the DPM-Solver-2 from Lu et al. (2022b, Algorithm 4* cf. *Algorithm 1).*

Proof. This follows as straightforward derivation from Proposition E.2 and Proposition E.3. □

Proposition E.5 (Rex (Euler-Midpoint) is DPM-Solver-12)**.** *The underlying scheme of Rex (Euler-Midpoint) for the noise prediction parameterization of diffusion models in Equation* (79) *is the DPM-Solver-12 from Lu et al. (2022b).*

Proof. The explicit midpoint method with embedded Euler method for adaptive step sizing is given by the Butcher tableau

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \\ & 1 & 0 \end{array}. \tag{236}$$

From Proposition E.2 and Proposition E.4 we have shown that Rex (Euler) and Rex (Midpoint) correspond to DPM-Solver-1 and DPM-Solver-2 respectively. Thus the Butcher tableau above outlines DPM-Solver-12. □

E.1.3 THIRD-ORDER METHODS

For third-order solvers like DPM-Solver-3 (Lu et al., 2022b, Algorithm 5) our constructed scheme differs from solvers derived using ETD methods due to the presence of φ_2 terms where

$$\varphi_{k+1}(t) = \int_0^1 e^{(1-\delta)t} \frac{\delta^k}{k!} \, \mathrm{d}\delta, \tag{237}$$

this also reasoning extends to the DPM-Solver-4 from Gonzalez et al. (2024, Algorithm 7).

E.2 REX AS REVERSIBLE SDE SOLVERS

In this section we discuss the connections between Rex and preexisting SDE solvers for diffusion models.

E.2.1 EULER-MARUYAMA

The extended Butcher tableau for the Euler-Maruyama scheme is given by

$$\begin{array}{c|c|c|c} 0 & 0 & 0 & 0 \\ \hline & 1 & 1 & 0 \end{array}. \tag{238}$$

Proposition E.6 (Rex (Euler-Maruyama) is reversible SDE-DPM-Solver++1)**.** *The underlying scheme of Rex (Euler-Maruyama) for the data prediction parameterization of diffusion models in Equation (127) is the SDE-DPM-Solver++1 from Lu et al. (2022a, Equation (18)).*

Proof. Apply in the Butcher tableau for the Euler-Maruyama scheme to $\mathbf{\Psi}$ constructed from Equation (145) to find

$$\boldsymbol{x}_{n+1} = \frac{\sigma_{n+1}^2 \alpha_n}{\sigma_n^2 \alpha_{n+1}} \boldsymbol{x}_n + \frac{\sigma_{n+1}^2}{\alpha_{n+1}} h \boldsymbol{x}_{0|\varrho_n}^\theta(\boldsymbol{x}_n) + \frac{\sigma_{n+1}^2}{\alpha_{n+1}} \boldsymbol{W}_n, \tag{239}$$

with $h = \varrho_{n+1} - \varrho_n$. We can rewrite the step size as

$$\begin{aligned}
\frac{\sigma_{n+1}^2}{\alpha_{n+1}} h &= \frac{\sigma_{n+1}^2}{\alpha_{n+1}} \left(\frac{\alpha_{n+1}^2}{\sigma_{n+1}^2} - \frac{\alpha_n^2}{\sigma_n^2} \right), && (240)\\
&= \left(\alpha_{n+1} - \frac{\alpha_n^2}{\alpha_{n+1}} \frac{\sigma_{n+1}^2}{\sigma_n^2} \right), && (241)\\
&= \alpha_{n+1} \left(1 - \frac{\alpha_n^2}{\alpha_{n+1}^2} \frac{\sigma_{n+1}^2}{\sigma_n^2} \right), && (242)\\
&= \alpha_{n+1} \left(1 - \frac{\varrho_n}{\varrho_{n+1}} \right), && (243)\\
&= \alpha_{n+1} \left(1 - e^{2 \log \frac{\gamma_n}{\gamma_{n+1}}} \right), && (244)\\
&= \alpha_{n+1} \left(1 - e^{2 \log \gamma_n - 2 \log \gamma_{n+1}} \right), && (245)\\
&\overset{(i)}{=} \alpha_{n+1} \left(1 - e^{2\lambda_n - 2\lambda_{n+1}} \right), && (246)\\
&\overset{(ii)}{=} \alpha_{n+1} \left(1 - e^{-2h_\lambda} \right), && (247)
\end{aligned}$$

where (i) holds by the letting $\lambda_t = \log \gamma_t$ following the notation of Lu et al. (2022b;a) and (ii) holds by letting $h_\lambda = \lambda_{n+1} - \lambda_n$. Now recall that

$$\frac{\sigma_{n+1}^2 \alpha_n}{\sigma_n^2 \alpha_{n+1}} = \frac{\sigma_{n+1}}{\sigma_n} e^{-h_\lambda}. \tag{248}$$

Plugging these back into Equation (239) yields

$$\boldsymbol{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} e^{-h_\lambda} \boldsymbol{x}_n + \alpha_{n+1} \left(1 - e^{-2h_\lambda} \right) \boldsymbol{x}_{0|t_n}^\theta(\boldsymbol{x}_n) + \frac{\sigma_{n+1}^2}{\alpha_n} \boldsymbol{W}_n. \tag{249}$$

Now recall that the Brownian increment $\boldsymbol{W}_n \coloneqq \boldsymbol{W}_{\varrho_{n+1}} - \boldsymbol{W}_{\varrho_n}$ has variance h. Thus via the Itô isometry we can write

$$\boldsymbol{W}_n \sim \sqrt{h}\boldsymbol{\epsilon}, \tag{250}$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Then we have

$$\frac{\sigma_{n+1}^2}{\alpha_{n+1}}\sqrt{h} = \frac{\sigma_{n+1}^2}{\alpha_{n+1}}\sqrt{\frac{\alpha_{n+1}^2}{\sigma_{n+1}^2} - \frac{\alpha_n^2}{\sigma_n^2}}, \tag{251}$$

$$= \sqrt{\sigma_{n+1}^2 - \frac{\alpha_n^2}{\alpha_{n+1}^2}\frac{\sigma_{n+1}^4}{\sigma_n^2}}, \tag{252}$$

$$= \sigma_{n+1}\sqrt{1 - \frac{\alpha_n^2}{\alpha_{n+1}^2}\frac{\sigma_{n+1}^2}{\sigma_n^2}}, \tag{253}$$

$$= \sigma_{n+1}\sqrt{1 - \frac{\varrho_n}{\varrho_{n+1}}}, \tag{254}$$

$$= \sigma_{n+1}\sqrt{1 - e^{-2h_\lambda}}. \tag{255}$$

Thus we have re-derived the noise term of the SDE-DPM-Solver++1, and putting everything together we have obtained the SDE-DPM-Solver++1 from Lu et al. (2022a) which is

$$\boldsymbol{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n}e^{-h_\lambda}\boldsymbol{x}_n + \alpha_{n+1}\left(1 - e^{-2h_\lambda}\right)\boldsymbol{x}_{0|t_n}^\theta(\boldsymbol{x}_n) + \sigma_{n+1}\sqrt{1 - e^{-2h_\lambda}}\boldsymbol{\epsilon}. \tag{256}$$

Thus we have shown that the SDE-DPM-Solver++1 is the same as the underlying scheme of Rex (Euler-Maruyama). □

Corollary E.6.1 (Rex (Euler-Maruyama) is reversible stochastic DDIM)**.** *The underlying scheme of Rex (Euler-Maruyama) for the data prediction parameterization of diffusion models in Equation* (127) *is the stochastic DDIM solver from Song et al. (2021a) with* $\eta = \sigma_t\sqrt{1 - e^{-2h_\lambda}}$.

Proof. This holds because SDE-DPM-Solver-1 is DDIM see Lu et al. (2022a, Section 6.1). □

Proposition E.7 (Rex (Euler-Maruyama) is reversible SDE-DPM-Solver-1)**.** *The underlying scheme of Rex (Euler-Maruyama) for the noise prediction parameterization of diffusion models in Equation* (145) *is the SDE-DPM-Solver-1 from Lu et al. (2022a, Equation (17)).*

Proof. Apply in the Butcher tableau for the Euler scheme to $\boldsymbol{\Psi}$ from Rex (see Proposition 3.1) to find

$$\boldsymbol{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n}\boldsymbol{x}_n + 2\alpha_{n+1}h\boldsymbol{x}_{T|\chi_n}^\theta(\boldsymbol{x}_n) + \alpha_{n+1}\boldsymbol{W}_n, \tag{257}$$

with $h = \chi_{n+1} - \chi_n$. Recall from Proposition E.2 that we can rewrite the step size

$$\alpha_{n+1}h = -\sigma_{n+1}\left(e^{h_\lambda} - 1\right). \tag{258}$$

Now recall that the Brownian increment $\boldsymbol{W}_n \coloneqq \overline{\boldsymbol{W}}_{\chi_{n+1}^2} - \overline{\boldsymbol{W}}_{\chi_n^2}$ has variance $\chi_n^2 - \chi_{n+1}^2$.[18] Thus via the Itô isometry we can write

$$\boldsymbol{W}_n \sim \sqrt{\chi_n^2 - \chi_{n+1}^2}\boldsymbol{\epsilon}, \tag{259}$$

[18]This is because $\overline{\boldsymbol{W}}_\chi^2$ is defined in reverse-time.

with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Then we have

$$\alpha_{n+1}\sqrt{\chi_n^2 - \chi_{n+1}^2} = \alpha_{n+1}\sqrt{\frac{\sigma_n^2}{\alpha_n^2} - \frac{\sigma_{n+1}^2}{\alpha_{n+1}^2}}, \tag{260}$$

$$= \sqrt{\frac{\sigma_n^2\alpha_{n+1}^2}{\alpha_n^2} - \sigma_{n+1}^2}, \tag{261}$$

$$= \sigma_{n+1}\sqrt{\frac{\sigma_n^2\alpha_{n+1}^2}{\sigma_{n+1}^2\alpha_n^2} - 1}, \tag{262}$$

$$= \sigma_{n+1}\sqrt{\frac{\chi_n^2}{\chi_{n+1}^2} - 1}, \tag{263}$$

$$= \sigma_{n+1}\sqrt{e^{\log\frac{\chi_n^2}{\chi_{n+1}^2}} - 1}, \tag{264}$$

$$= \sigma_{n+1}\sqrt{e^{\log\chi_n^2 - \log\chi_{n+1}^2} - 1}, \tag{265}$$

$$= \sigma_{n+1}\sqrt{e^{-2\log\gamma_n + 2\log\gamma_{n+1}} - 1}, \tag{266}$$

$$= \sigma_{n+1}\sqrt{e^{2\log\lambda_{n+1} - 2\log\lambda_n} - 1}, \tag{267}$$

$$= \sigma_{n+1}\sqrt{e^{2h_\lambda} - 1}. \tag{268}$$

Plugging Equations (258) and (268) back into Equation (257) yields

$$\boldsymbol{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n}\boldsymbol{x}_n - 2\sigma_{n+1}\left(e^{h_\lambda} - 1\right)\boldsymbol{x}_{T|\chi_n}^\theta(\boldsymbol{x}_n) + \sigma_{n+1}\sqrt{e^{2h_\lambda} - 1}\boldsymbol{\epsilon}, \tag{269}$$

which is the SDE-DPM-Solver-1 from Lu et al. (2022a). □

Corollary E.7.1 (Rex (Euler-Maruyama) is reversible stochastic DDIM for noise prediction models)**.** *The underlying scheme of Rex (Euler-Maruyama) for the noise prediction parameterization of diffusion models in Equation* (145) *is the stochastic DDIM solver from Song et al. (2021a) with* $\eta = \sigma_t\sqrt{e^{-2h_\lambda} - 1}$.

Proof. This follows from a straightforwardly from Corollary E.6.1 and Lu et al. (2022b, Equation (4.1)). □

E.3 Rex as reversible SEEDS-1

Proposition E.8 (Rex is reversible SEEDS-1)**.** *The choice of Euler or Euler-Maruyama for the underlying scheme of Rex with either the noise prediction parameterization of diffusion models in Equations* (79) *and* (145) *or data prediction in Equations* (79) *and* (127) *yields the four variants of SEEDS-1 outlined in Gonzalez et al. (2024, Equations (28-31)).*

Proof. This follows straightforwardly from Propositions E.1, E.2, E.6 and E.7 by definition of SEEDS-1. □

Corollary E.8.1 (Rex (Euler-Maruyama) is reversible gDDIM)**.** *The underlying scheme of Rex (Euler-Maruyama) for the data prediction parameterization of diffusion models in Equation* (127) *is the gDDIM solver in Zhang et al. (2023, Theorem 1) for* $\ell = 1$.

Proof. This follows as an immediate consequence of Proposition E.8 since by Gonzalez et al. (2024, Proposition 4.5) gDDIM is SEEDS-1. □

As mentioned earlier in Appendix A.5.1 high-order variants of SEEDS use a Markov-preserving noise decomposition to approximate the iterated stochastic integrals. However, we follow Foster et al. (2024) and use the space-time Lévy area resulting in numerical schemes that are quite different beyond the first-order case, albeit that Rex exhibits better convergence properties.

F A BRIEF NOTE ON THE THEORY OF ROUGH PATHS

To perform reversibility it is useful to consider the pathwise interpretation of SDEs (Lyons, 1998), as such we introduce a few notations from rough path theory. Let $\{\boldsymbol{W}_t\}$ be a d_w-dimensional Brownian motion and let $\boldsymbol{W}$ be enhanced by

$$\mathbb{W}_{s,t} = \int_s^t \boldsymbol{W}_{s,r} \otimes \circ \mathrm{d}\boldsymbol{W}_r, \tag{270}$$

where $\otimes$ is the tensor product. Then, the pair $\mathcal{W} := (\boldsymbol{W}, \mathbb{W})$ is the *Stratonovich enhanced Brownian rough path*.[19] Thus consider the d_x-dimensional *rough differential equation* RDE of the form:

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{\mu}(t, \boldsymbol{X}_t)\,\mathrm{d}t + \boldsymbol{\sigma}(t, \boldsymbol{X}_t)\,\mathrm{d}\mathcal{W}_t, \qquad \boldsymbol{X}_0 = \boldsymbol{x}_0. \tag{271}$$

where $\boldsymbol{\mu} : [0, T] \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ is Lipschitz continuous in its second argument and $\boldsymbol{\sigma} \in \mathcal{C}_b^{1,3}([0, T] \times \mathbb{R}^{d_x}; \mathcal{L}(\mathbb{R}^{d_w}, \mathbb{R}^{d_x}))$ (Friz & Hairer, 2020, Theorem 9.1).[20] Fix an $\omega \in \Omega$, then almost surely $\mathcal{W}(\omega)$ admits a unique solution to the RDE $(\boldsymbol{X}_t(\omega), \boldsymbol{\sigma}(t, \boldsymbol{X}_t(\omega)))$ and $\boldsymbol{X}_t = \boldsymbol{X}_t(\omega)$ is a strong solution to the Stratonovich SDE[21] started at $\boldsymbol{X}_0 = \boldsymbol{x}_0$. To elucidate, consider the commutative diagram below

$$\boldsymbol{W} \overset{\Psi}{\longmapsto} (\boldsymbol{W}, \mathbb{W}) \overset{S}{\longmapsto} \boldsymbol{X}, \tag{272}$$

where Ψ is a map which merely lifts Brownian motion into a rough path (could be Itô or Stratonovich), the second map, S, is known as the *Itô-Lyons map* (Lyons, 1998); this map is purely deterministic and is also a *continuous map* w.r.t. to initial condition and driving signal. Thus for a fixed realization of the Brownian motion we have a pathwise interpretation of the Stratonovich SDE.

G IMPLEMENTATION DETAILS

G.1 CLOSED FORM EXPRESSIONS OF THE NOISE SCHEDULE

G.1.1 VARIANCE PRESERVING SDES

In practice, popular libraries like the `diffusers` library define the noise schedule for diffusion models as a discrete schedule $\{\beta_n\}_{n=1}^N$ following Ho et al. (2020); Song et al. (2021a) as an arithemetic sequence of the form

$$\beta_n = \frac{\beta_0}{N} + \frac{n-1}{N(N-1)}(\beta_1 - \beta_0), \tag{273}$$

with hyperparameters $\beta_0, \beta_1 \in \mathbb{R}_{\geq 0}$. Song et al. (2021b) defines the continuous-time schedule as

$$\beta_t = \beta_0 + t(\beta_1 - \beta_0), \tag{274}$$

for all $t \in [0, 1]$ in the limit of $N \to \infty$. Thus one can write the forward-time diffusion (variance preserving) SDE as

$$\mathrm{d}\boldsymbol{X}_t = -\frac{1}{2}\beta_t \boldsymbol{X}_t\,\mathrm{d}t + \sqrt{\beta_t}\,\mathrm{d}\boldsymbol{W}_t. \tag{275}$$

Thus we can express the noise schedule (α_t, σ_t) as

$$\alpha_t = \exp\left(-\frac{1}{2}\int \beta_t\,\mathrm{d}t\right), \tag{276a}$$

$$\sigma_t = \sqrt{1 - \alpha_t^2}. \tag{276b}$$

N.B., often the hyperparmeters in libraries like `diffusers` are expressed as $\hat{\beta}_0 = \frac{\beta_0}{N}$ and $\hat{\beta}_1 = \frac{\beta_1}{N}$, often with $N = 1000$.

[19] See, Friz & Hairer (2020, Chapter 3) for more details.

[20] Here $\mathcal{L}(V, W)$ denotes the set of continuous maps from V to W, a Banach space.

[21] If $\boldsymbol{X}_t$ and $\partial_{\boldsymbol{x}} \boldsymbol{X}_t$ are adapted and $\langle \boldsymbol{X}, \boldsymbol{W} \rangle_t$ exists, then almost surely

$$\int_0^T \boldsymbol{X}\mathrm{d}\mathcal{W}_t = \int_0^T \boldsymbol{X} \circ \mathrm{d}\boldsymbol{W}_t.$$

Linear noise schedule. For the linear noise schedule in Equation (274) used by DDPMs (Ho et al., 2020), the schedule (α_t, σ_t) is written as

$$\begin{aligned}\alpha_t &= \exp\left(-\frac{\beta_1-\beta_0}{4}t^2 - \frac{\beta_0}{2}t\right),\\ \sigma_t &= \sqrt{1-\alpha_t^2},\end{aligned} \tag{277}$$

for $t \in [0,1]$ with hyperparameters β_0 and β_1.

Proposition G.1 (Inverse function of γ_t for linear noise schedule)**.** *For the linear noise schedule used by DDPMs (Ho et al., 2020) the inverse function of γ_t denoted t_γ can be expressed in closed form as*

$$t_\gamma(\gamma) = \frac{-\beta_0 + \sqrt{\beta_0^2 + 2(\beta_1-\beta_0)\log(\gamma^{-2}+1)}}{\beta_1-\beta_0}. \tag{278}$$

Proof. Let α_t be denoted by $\alpha_t = e^{a_t}$ where

$$a_t = -\frac{\beta_1-\beta_0}{4}t^2 - \frac{\beta_0}{2}t. \tag{279}$$

Then by definition of γ_t we can write

$$\gamma_t = \frac{e^{a_t}}{\sqrt{1-e^{2a_t}}}, \tag{280}$$

and with a little more algebra we find

$$\sqrt{1-e^{2a_t}} = \frac{e^{a_t}}{\gamma_t}, \tag{281}$$

$$1-e^{2a_t} = \frac{e^{2a_t}}{\gamma_t^2}, \tag{282}$$

$$e^{-2a_t} - 1 = \gamma_t^{-2}, \tag{283}$$

$$e^{-2a_t} = \gamma_t^{-2} + 1, \tag{284}$$

$$-2a_t = \log(\gamma_t^{-2}+1). \tag{285}$$

Then by substituting in the definition of a_t and letting γ denote the variable produced by γ_t we have

$$\frac{\beta_1-\beta_0}{2}t^2 + \beta_0 t - \log(\gamma^{-2}+1) = 0. \tag{286}$$

We then use the quadratic formula to find the roots of the polynomial of t to find

$$t = \frac{-\beta_0 \pm \sqrt{\beta_0^2 + 2(\beta_1-\beta_0)\log(\gamma^{-2}+1)}}{\beta_1-\beta_0}. \tag{287}$$

Since $t \in [0,1]$ we only take the positive root and thus

$$t = \frac{-\beta_0 + \sqrt{\beta_0^2 + 2(\beta_1-\beta_0)\log(\gamma^{-2}+1)}}{\beta_1-\beta_0}. \tag{288}$$

□

Corollary G.1.1 (Inverse function of χ_t for linear noise schedule)**.** *It follows by a straightforward substitution from Proposition G.1 that t_χ can be written as*

$$t_\chi(\chi) = \frac{-\beta_0 + \sqrt{\beta_0^2 + 2(\beta_1-\beta_0)\log(\chi^{2}+1)}}{\beta_1-\beta_0}. \tag{289}$$

Corollary G.1.2 (Inverse function of ϱ_t for linear noise schedule)**.** *It follows by a straightforward substitution from Proposition G.1 that t_ϱ can be written as*

$$t_\varrho(\varrho) = \frac{-\beta_0 + \sqrt{\beta_0^2 + 2(\beta_1-\beta_0)\log(\varrho^{-1}+1)}}{\beta_1-\beta_0}. \tag{290}$$

Scaled linear schedule. The *scaled linear schedule* is used widely by *latent diffusion models* (LDMs) (Rombach et al., 2022) and takes the discrete form of

$$\beta_n = \left(\sqrt{\hat{\beta}_0} + \frac{n-1}{N-1}\left(\sqrt{\hat{\beta}_1} - \sqrt{\hat{\beta}_0}\right)\right)^2. \tag{291}$$

Thus following a similar approach to Song et al. (2021b) we write the scaled linear schedule as a function of t,

$$\beta_t = (\beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0)t^2 + 2t(\sqrt{\beta_1\beta_0} - \beta_0) + \beta_0. \tag{292}$$

Then using Equation (276) we find the noise schedule (α_t, σ_t) to be defined as

$$\begin{aligned} \alpha_t &= \exp\left(-\frac{\beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0}{6}t^3 - \frac{\sqrt{\beta_1\beta_0} - \beta_0}{2}t^2 - \frac{\beta_0}{2}t\right), \\ \sigma_t &= \sqrt{1-\alpha_t^2}. \end{aligned} \tag{293}$$

Next we will derive the inverse function for γ_t

Proposition G.2 (Inverse function of γ_t for scaled linear noise schedule)**.** *For the scaled linear noise schedule commonly used by LDMs (Rombach et al., 2022) the inverse function of γ_t denoted t_γ can be expressed in closed form as*

$$t_\gamma(\gamma) = \frac{\beta_0 - \sqrt{\beta_1\beta_0} - \sqrt[3]{2(\sqrt{\beta_1\beta_0} - \beta_0)^3 - 3\beta_0\Delta(\sqrt{\beta_1\beta_0} - \beta_0) - 3\Delta^2\log(\gamma^{-2}+1)}}{\Delta}, \tag{294}$$

where

$$\Delta = \beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0. \tag{295}$$

Proof. Let α_t be denoted by $\alpha_t = e^{a_t}$ where

$$a_t = -\frac{\beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0}{6}t^3 - \frac{\sqrt{\beta_1\beta_0} - \beta_0}{2}t^2 - \frac{\beta_0}{2}t. \tag{296}$$

Then by definition of γ_t we can write

$$\gamma_t = \frac{e^{a_t}}{\sqrt{1-e^{2a_t}}}, \tag{297}$$

and with a little more algebra we find

$$\sqrt{1-e^{2a_t}} = \frac{e^{a_t}}{\gamma_t}, \tag{298}$$

$$1 - e^{2a_t} = \frac{e^{2a_t}}{\gamma_t^2}, \tag{299}$$

$$e^{-2a_t} - 1 = \gamma_t^{-2}, \tag{300}$$

$$e^{-2a_t} = \gamma_t^{-2} + 1, \tag{301}$$

$$-2a_t = \log(\gamma_t^{-2} + 1). \tag{302}$$

Then by substituting in the definition of a_t and letting γ denote the variable produced by γ_t we have

$$\frac{\beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0}{3}t^3 + (\sqrt{\beta_1\beta_0} - \beta_0)t^2 + \beta_0 t - \log(\gamma^{-2}+1) = 0. \tag{303}$$

We then use the cubic formula (Cardano, 1545) to find the roots of the polynomial of t. The only real root is given by

$$t_\gamma(\gamma) = \frac{\beta_0 - \sqrt{\beta_1\beta_0} - \sqrt[3]{2(\sqrt{\beta_1\beta_0} - \beta_0)^3 - 3\beta_0\Delta(\sqrt{\beta_1\beta_0} - \beta_0) - 3\Delta^2\log(\gamma^{-2}+1)}}{\Delta}, \tag{304}$$

where

$$\Delta = \beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0. \tag{305}$$

□

Corollary G.2.1 (Inverse function of χ_t for scaled linear noise schedule)**.** *It follows by a straightforward substitution from Proposition G.2 that t_χ can be written as*

$$t_\chi(\chi) = \frac{\beta_0 - \sqrt{\beta_1\beta_0} - \sqrt[3]{2(\sqrt{\beta_1\beta_0} - \beta_0)^3 - 3\beta_0\Delta(\sqrt{\beta_1\beta_0} - \beta_0) - 3\Delta^2 \log(\chi^2 + 1)}}{\Delta}, \tag{306}$$

where

$$\Delta = \beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0. \tag{307}$$

Corollary G.2.2 (Inverse function of ϱ_t for scaled linear noise schedule)**.** *It follows by a straightforward substitution from Proposition G.2 that t_ϱ can be written as*

$$t_\varrho(\varrho) = \frac{\beta_0 - \sqrt{\beta_1\beta_0} - \sqrt[3]{2(\sqrt{\beta_1\beta_0} - \beta_0)^3 - 3\beta_0\Delta(\sqrt{\beta_1\beta_0} - \beta_0) - 3\Delta^2 \log(\varrho^{-1} + 1)}}{\Delta}, \tag{308}$$

where

$$\Delta = \beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0. \tag{309}$$

G.1.2 OT FLOW MATCHING

Within OT flow matching (Tong et al., 2024) framework these expressions become much simpler.[22] Within this framework we have

$$\alpha_t = t, \tag{310a}$$
$$\sigma_t = 1 - t. \tag{310b}$$

Consequently, we have the following simple proposition.

Proposition G.3 (Inverse function of γ_t in OT flow matching)**.** *Within the OT flow matching context the inverse function of γ_t denoted t_γ can be expressed in closed form as*

$$t_\gamma(\gamma) = \frac{\gamma}{1 + \gamma}. \tag{311}$$

Proof. By definition of γ_t we write

$$\gamma_t = \frac{t}{1 - t}, \tag{312}$$
$$(1 - t)\gamma_t = t, \tag{313}$$
$$\gamma_t = (1 + \gamma_t)t, \tag{314}$$
$$t = \frac{\gamma_t}{1 + \gamma_t}. \tag{315}$$

□

Corollary G.3.1 (Inverse function of χ_t in OT flow matching)**.** *It follows by a straightforward substitution from Proposition G.3 that t_χ can be written as*

$$t_\chi(\chi) = \frac{1}{1 + \chi}. \tag{316}$$

G.2 SOME OTHER INVERSE FUNCTIONS

$\gamma \mapsto \sigma$. Additionally, we need to be able to extract the weighting terms from the time integration variable. For the ODE case we need the function $\sigma_\gamma(\gamma)$ which describes the map $\gamma \mapsto \sigma$. By the

[22] We will use the flow matching conventions where $\boldsymbol{X}_1 \sim p_{\text{data}}$.

definition of γ we have

$$\gamma = \frac{\alpha}{\sigma}, \tag{317}$$
$$\gamma \overset{(i)}{=} \frac{\sqrt{1-\sigma^2}}{\sigma}, \tag{318}$$
$$\sigma\gamma = \sqrt{1-\sigma^2}, \tag{319}$$
$$\sigma^2\gamma^2 = 1-\sigma^2, \tag{320}$$
$$\sigma^2\gamma^2 = 1-\sigma^2, \tag{321}$$
$$\gamma^2 = \sigma^{-2} - 1, \tag{322}$$
$$\gamma^2 + 1 = \sigma^{-2}, \tag{323}$$
$$\sigma^2 = \frac{1}{\gamma^2+1} \tag{324}$$
$$\sigma_\gamma(\gamma) = \frac{1}{\sqrt{\gamma^2+1}}, \tag{325}$$

where (i) hold by $\sigma^2 = 1 - \alpha^2$ for VP type diffusion SDEs.

$\varrho \mapsto \frac{\sigma}{\gamma}$. Likewise, for the SDE case we need the function which maps $\varrho \mapsto \frac{\sigma}{\gamma}$. Recall that (note we drop the subscript t for the derivation)

$$\varrho = \frac{\alpha^2}{\sigma^2}, \tag{326}$$

thus we have

$$\varrho \overset{(i)}{=} \frac{\alpha^2}{1-\alpha^2}, \tag{327}$$
$$(1-\alpha^2)\varrho = \alpha^2, \tag{328}$$
$$\alpha^{-2} - 1 = \varrho^{-1}, \tag{329}$$
$$\alpha^{-2} = \varrho^{-1} + 1, \tag{330}$$
$$\alpha = \frac{1}{\sqrt{\varrho^{-1}+1}}, \tag{331}$$

where (i) hold by $\sigma^2 = 1 - \alpha^2$ for VP type diffusion SDEs. Then we can write

$$\frac{\sigma}{\gamma} = \frac{\sigma^2}{\alpha}, \tag{332}$$
$$= \frac{\sigma^2}{\alpha}\frac{\alpha}{\alpha}, \tag{333}$$
$$= \frac{\sigma^2}{\alpha^2}\alpha, \tag{334}$$
$$= \varrho^{-1}\alpha, \tag{335}$$
$$= \frac{1}{\rho\sqrt{\rho^{-1}+1}}. \tag{336}$$

$\chi \mapsto \alpha$. Lastly, for the noise prediction models we need the map $\chi \mapsto \alpha$ denoted $\alpha_\chi(\chi)$. By definition of χ we have

$$\chi = \frac{\sigma}{\alpha}, \tag{337}$$
$$\chi \overset{(i)}{=} \frac{\sqrt{1-\alpha^2}}{\alpha}, \tag{338}$$
$$\alpha_\chi(\chi) \overset{(ii)}{=} \frac{1}{\sqrt{\chi^2+1}}, \tag{339}$$

where (i) hold by $\sigma^2 = 1 - \alpha^2$ for VP type diffusion SDEs and (ii) holds by the derivation for $\sigma_\gamma(\gamma)$ *mutatis mutandis*.

G.3 Numerical simulation of Brownian motion

Earlier we mentioned that for reversible methods we need to be able to compute both the *same* realization of the Brownian motion. Now sampling Brownian motion is quite simple—recall Lévy's characterization of Brownian motion (Øksendal, 2003, Theorem 8.6.1)—and can be sampled by drawing independent Gaussian increments during the numerical solve of an SDE. A common choice for an adaptive solver is to use Lévy's Brownian bridge formula (Revuz & Yor, 2013).

Definition G.1 (Lévy's Brownian bridge)**.** Given the standard d_w-dimensional Brownian motion $\{\boldsymbol{W}_t : t \geq 0\}$ and for any $0 \leq s < t < u$, the Brownian bridge is defined as

$$\boldsymbol{W}_t | \boldsymbol{W}_s, \boldsymbol{W}_u \sim \mathcal{N}\left(\boldsymbol{W}_s + \frac{t-s}{u-s}(\boldsymbol{W}_u - \boldsymbol{W}_s), \frac{(u-t)(t-s)}{u-s}\boldsymbol{I}\right), \tag{340}$$

and this quantity is conditionally independent of $\boldsymbol{W}_v$ for $v < s$ or $v > u$.

Sampling the Brownian motion in reverse-time, however, is more complicated as it is only adapted to the natural filtration defined in forward time. The naïve approach to sampling Brownian motion, called the *Brownian path*, is to simply store the entire realization of the Brownian motion from the forward pass in memory and use Equation (340) when necessary (for adaptive step size methods). This results in a query time of $\mathcal{O}(1)$, but with a memory cost of $\mathcal{O}(nd_w)$, where n is the number of samples.

G.3.1 Methods

Virtual Brownian Tree. Seminal work on neural SDEs by Li et al. (2020) introduced the *Virtual Brownian Tree* which extends the concept of Brownian trees introduced by Gaines & Lyons (1997). The Brownian tree recursively applies Equation (340) to sample the Brownian motion at any midpoint, constructing a tree structure; however, storing such a tree would be memory intensive. By making use of splittable *pseudo-random number generators* PRNGs (Salmon et al., 2011; Claessen & Pałka, 2013) which can deterministically generate two random seeds given an existing seed. Then making use of a splittable PRNG one can evaluate the Brownian motion at any point by recursively applying the Brownian tree constructing to rebuild the tree until the recursive midpoint time t_r is suitable *close* to the desired timestep t, *i.e.*, $|t - t_r| < \epsilon$ for some fixed error threshold $\epsilon > 0$. This requires constant $\mathcal{O}(1)$ memory but takes $\mathcal{O}(\log(1/\epsilon))$ time and is only *approximate*.

Brownian Interval. Closely related work by Kidger et al. (2021) introduces the *Brownian Interval* which offers exact sampling with $\mathcal{O}(1)$ query times. The primary difference between this method and Virtual Brownian Trees is that this method focuses on intervals rather than particular sample points. To elucidate, let $\boldsymbol{W}_{s,t} = \boldsymbol{W}_t - \boldsymbol{W}_s$ denote an interval of Brownian motion. Then the formula for Lévy's Brownian bridge (340) can be rewritten in terms of Brownian intervals as

$$\boldsymbol{W}_{s,t} | \boldsymbol{W}_{s,u} \sim \mathcal{N}\left(\frac{t-s}{u-s}\boldsymbol{W}_{s,u}, \frac{(u-t)(s-u)}{u-s}\boldsymbol{I}\right). \tag{341}$$

Then, the method constructs a tree with stump being the global interval $[0, T]$ and a random seed for a splittable PRNG. New leaf nodes are constructed when queries over intervals are made; this provides the advantage of the tree being query-dependent unlike the Virtual Brownian Tree which has a fixed dyadic structure. Further computational improvements are made to improve implementation with the details being found in Kidger (2022, Section 5.5.3). Beyond the numerical efficiency in computing intervals over points is that we regularly need use intervals in numeric schemes and not single sample points. Often, solvers which approximate higher-order integrals (*e.g.*, stochastic Runge-Kutta) require samples of the Lévy area[23] which would require the Brownian interval to construct.[24]

[23] *I.e.*, for a d_w-dimensional Brownian motion over $[s, t]$ the Lévy area is

$$2\boldsymbol{L}^{i,j}_{s,t} := \int_s^t \boldsymbol{W}^i_{s,u} \mathrm{d}\boldsymbol{W}^j_u - \int_s^t \boldsymbol{W}^j_{s,u} \mathrm{d}\boldsymbol{W}^i_u.$$

[24] The interested reader can find more details in James Foster's thesis (Foster, 2020).

Updated Virtual Brownian Tree. Recent work by Jelinčič et al. (2024) improves upon the Virtual Brownian Tree (Li et al., 2020) by using an interpolation strategy between query points.[25] This enables the updated algorithm to exactly match the distribution of Brownian motion and Lévy areas at all query times as long as each query time is at least ϵ apart.

G.4 Implementation

We used the Brownian interval (Kidger et al., 2021) provided by the `torchsde` library. In general we would recommend the virtual Brownian tree from Jelinčič et al. (2024) over the Brownian interval, an implementation of this can be found in the `diffrax` library. However, as our code base made extensive used of prior projects developed in pytorch and `diffrax` is a jax library it made more sense to use `torchsde` for this project.

[25] This algorithm is a part of the popular `Diffrax` library.

H CODE

In this section we provide some example code for the core components of the model to help illustrate the core ideas.

```
def rex_forward(model_func, scheduler, xt, xt_hat, timesteps, solver='euler', coupling=0.999,
↪  low_order_final_n_steps=0, bm=None, pred_type='data', sched_type='linear'):
    """
    Based on McCallum & Foster's reversible ODE solver and adapted for diffusion models.
    """

    # Choose underlying solver
    is_sde = (solver in SDE_SOLVERS)
    psi = SOLVER_DICT[solver]

    if not is_sde:
        _t_to_gamma, _gamma_to_t = _gen_time_funcs(sched_type=sched_type, pred_type=pred_type)
        t_to_gamma = _t_to_gamma
        gamma_to_t = _gamma_to_t
        gamma_to_sigma = _gamma_to_sigma if pred_type == 'data' else _chi_to_alpha
    else:
        _t_to_rho, _rho_to_t = _gen_time_funcs(sched_type=sched_type, rho=True, pred_type=pred_type)
        t_to_gamma = _t_to_rho
        gamma_to_t = _rho_to_t
        gamma_to_sigma = _rho_to_siggamma if pred_type == 'data' else _chi_to_alpha


    # create timesteps in gamma, alt gamma^2 = rho for SDEs
    gammas = t_to_gamma(scheduler, timesteps)

    # Push gamma reparam back to time t and convert noise pred to data pred
    if pred_type == 'data':
        wrap_model = lambda gamma, x: _convert_noise_to_data(scheduler, model_func,
        ↪  gamma_to_t(scheduler, gamma), x, sched_type=sched_type)
    else:
        p = 2 if is_sde else 1
        wrap_model = lambda gamma, x: p * model_func(gamma_to_t(scheduler, gamma), x)

    xt.to(torch.float32)
    xt_hat.to(torch.float32)

    for n in tqdm(range(len(gammas)-1)):
        gamma_n = gammas[n]
        gamma_n1 = gammas[n+1]
        h = gamma_n1 - gamma_n

        sigma_n = gamma_to_sigma(gamma_n)
        sigma_n1 = gamma_to_sigma(gamma_n1)

        if n < (len(gammas) - 1 - low_order_final_n_steps):
            if not is_sde:
                _psi = lambda t, x, h: psi(wrap_model, t, x, h)
            else:
                _psi = lambda t, x, h: psi(wrap_model, t, x, h, bm, pred_type=pred_type)
        else:
            if not is_sde:
                _psi = lambda t, x, h: euler(wrap_model, t, x, h)
            else:
                _psi = lambda t, x, h: euler_maruyama(wrap_model, t, x, h, bm, pred_type=pred_type)


        xt = (sigma_n1 / sigma_n) * (coupling * xt + (1-coupling) * xt_hat) + sigma_n1 * _psi(gamma_n,
        ↪  xt_hat, h)
        xt_hat = (sigma_n1 / sigma_n) * xt_hat - sigma_n1 * _psi(gamma_n1, xt, -h)

    return xt, xt_hat
```

```python
def rex_backward(model_func, scheduler, xt, xt_hat, timesteps, solver='euler', coupling=0.999,
↪  low_order_final_n_steps=0, bm=None, pred_type='data', sched_type='linear'):
    """
    Based on McCallum & Foster's reversible ODE solver and adapted for diffusion models.
    """

    # Choose underlying solver
    is_sde = (solver in SDE_SOLVERS)
    psi = SOLVER_DICT[solver]

    if not is_sde:
        _t_to_gamma, _gamma_to_t = _gen_time_funcs(sched_type=sched_type, pred_type=pred_type)
        t_to_gamma = _t_to_gamma
        gamma_to_t = _gamma_to_t
        gamma_to_sigma = _gamma_to_sigma if pred_type == 'data' else _chi_to_alpha
    else:
        _t_to_rho, _rho_to_t = _gen_time_funcs(sched_type=sched_type, rho=True, pred_type=pred_type)
        t_to_gamma = _t_to_rho
        gamma_to_t = _rho_to_t
        gamma_to_sigma = _rho_to_siggamma if pred_type == 'data' else _chi_to_alpha


    # create timesteps in gamma, alt gamma^2 = rho for SDEs
    gammas = t_to_gamma(scheduler, timesteps)

    # Push gamma reparam back to time t and convert noise pred to data pred
    if pred_type == 'data':
        wrap_model = lambda gamma, x: _convert_noise_to_data(scheduler, model_func,
        ↪  gamma_to_t(scheduler, gamma), x, sched_type=sched_type)
    else:
        p = 2 if is_sde else 1
        wrap_model = lambda gamma, x: p * model_func(gamma_to_t(scheduler, gamma), x)

    xt.to(torch.float32)
    xt_hat.to(torch.float32)

    coupling_inv = 1. / coupling

    for n in tqdm(range(len(gammas) - 2, -1, -1)):
        gamma_n = gammas[n]
        gamma_n1 = gammas[n+1]
        h = gamma_n1 - gamma_n

        sigma_n = gamma_to_sigma(gamma_n)
        sigma_n1 = gamma_to_sigma(gamma_n1)

        if n < (len(gammas) - 1 - low_order_final_n_steps):
            if not is_sde:
                _psi = lambda t, x, h: psi(wrap_model, t, x, h)
            else:
                _psi = lambda t, x, h: psi(wrap_model, t, x, h, bm, pred_type=pred_type)
        else:
            if not is_sde:
                _psi = lambda t, x, h: euler(wrap_model, t, x, h)
            else:
                _psi = lambda t, x, h: euler_maruyama(wrap_model, t, x, h, bm, pred_type=pred_type)

        xt_hat = (sigma_n / sigma_n1) * xt_hat + sigma_n * _psi(gamma_n1, xt, -h)
        xt = (sigma_n / sigma_n1) * (coupling_inv * xt) + (1 - coupling_inv) * xt_hat - sigma_n *
        ↪  coupling_inv * _psi(gamma_n, xt_hat, h)

    return xt, xt_hat
```

In Code H.3 we provide an implementation of the ShARK method. The official implementation can be found at https://github.com/patrick-kidger/diffrax/blob/main/diffrax/_solver/shark.py.

```python
def ShARK(model, time_var, x, h, bm, pred_type='data'):
    t_to_w = _rho_to_siggamma if pred_type == 'data' else _chi_to_alpha

    x_sg = x / t_to_w(time_var)

    if pred_type == 'data':
        a, b = time_var, time_var + h
    else:
        a, b = time_var.pow(2), (time_var + h).pow(2)

    if h < 0:
        a, b = b, a

    h_corr = h if pred_type == 'data' else (time_var + h).pow(2) - time_var.pow(2)

    W, U = bm(a, b, return_U=True)
    W, U = W.to(x.device), U.to(x.device)

    if h < 0:
        H = U / (-h_corr) - 0.5 * W
        W = -W
    else:
        H = U / (-h_corr) - 0.5 * W

    Z1 = x_sg + H

    f1 = model(time_var, t_to_w(time_var) * Z1)

    Z2 = x_sg + h * (5/6) * f1 + (5/6) * W + H
    f2 = model(time_var + 5/6 * h, t_to_w(time_var + 5/6 * h) * Z2)

    return h * (0.4 * f1 + 0.6 * f2) + W
```

I EXPERIMENTAL DETAILS

We provide additional details for the empirical studies conducted in Section 4. *N.B.*, for all experiments we used fixed random seeds between the different software components to ensure a fair comparision.

I.1 UNCONDITIONAL IMAGE GENERATION

I.1.1 DIFFUSION MODEL

We make use of a pre-trained DDPM (Ho et al., 2020) model trained on the CelebA-HQ 256×256 dataset (Karras et al., 2018). The linear noise schedule from (Ho et al., 2020) is given as

$$\beta_i = \frac{\hat{\beta}_0}{T} + \frac{i-1}{T(T-1)}(\hat{\beta}_1 - \hat{\beta}_0). \tag{342}$$

We convert this into a continuous time representation via the details in Appendix G.1 following Song et al. (2021b). For this experiment we used $\hat{\beta}_0 = 0.0001$ and $\hat{\beta}_1 = 0.2$. To ensure numerical stability due to $\frac{1}{\sigma_t}$ terms we solve the probability flow ODE in reverse-time on the time interval $[\epsilon, 1]$ with $\epsilon = 0.0002$. This is a common choice to make in practice see Song et al. (2023).

I.1.2 METRICS

We use several metrics to assess the performance in unconditional image generation following Stein et al. (2023) by using a DINOv2 feature extractor (Oquab et al., 2023), all of which are calculated using the 10k generated samples and 30k real samples from the CelebA-HQ dataset. Throughout this section we will let $\{\boldsymbol{x}_i\}_{i=1}^n$ denote an empirical distribution drawn from our generated distribution $\mathbb{P}_\theta$ and let $\{\hat{\boldsymbol{x}}_i\}_{i=1}^m$ denote an empirical distribution drawn from the data distribution $\mathbb{P}_{data}$.

FD. The *Fréchet distance* (FD) (Dowson & Landau, 1982) is measured using the sample mean and covariance of the real $\mathbb{P}_{data}$ and generated $\mathbb{P}_\theta$ distributions denoted

$$\text{FD}(\mathbb{P}_{data} \| \mathbb{P}_\theta) = \|\mu_{data} - \mu_\theta\|_2^2 + \text{Tr}\left(\Sigma_{data} + \Sigma_\theta - 2(\Sigma_{data}\Sigma_\theta)^{\frac{1}{2}}\right), \tag{343}$$

where $(\mu_\cdot, \Sigma_\cdot)$ denote the sample mean and covariances. This metric corresponds two the 2-Wasserstein distance between two multivariate Gaussians and is thus a valid metric between the first two moments. Heusel et al. (2017) popularized the use of this metric within the feature layer of an Inception-V3 network (Szegedy et al., 2016) to assess the fidelity of unconditional image generation, this metric is referred to as the *Fréchet inception distance* or FID. Recent works have challenged the use of the Inception-V3 network as the feature extractor (Stein et al., 2023; Jayasumana et al., 2024; Kynkäänniemi et al., 2023) showing that the Inception-V3 network is poorly suited for capturing a semantic view of images which correlates well to human judgment. In particular, Stein et al. (2023) shows that using DINOv2 (Oquab et al., 2023) for the feature extractor results in a metric which is significantly more aligned with human judgment.

FD$_\infty$. FD$_\infty$ proposed by Chong & Forsyth (2020) is a modification of FD which aims to remove the inherent bias induced by using a finite number of empirical samples. The samples is determined by evaluating FD over 15 regular intervals over the number of total samples and fitting a linear trend to the 15 data points to infer a trend for FD as the number of empirical samples, $N \to \infty$.

Precision, recall, density and coverage. The density metric (Naeem et al., 2020) is used as a proxy to measure sample fidelity and improves upon the earlier precision metric (Kynkäänniemi et al., 2019; Sajjadi et al., 2018). The metric is based upon nearest neighbours distance computed in a representation space and counts how many real-sample neighbourhood balls contain the generated sample. Likewise to quantify sample diversity we use the coverage metric (Naeem et al., 2020) which improves upon the earlier recall metric (Kynkäänniemi et al., 2019; Sajjadi et al., 2018). The density metric is given by

$$\text{density}(\mathbb{P}_{data}, \mathbb{P}_\theta) = \frac{1}{kn}\sum_{i=1}^{n}\sum_{j=1}^{m} 1_{B(\hat{\boldsymbol{x}}_j, \delta^k(\hat{\boldsymbol{x}}_j))}(\boldsymbol{x}_i), \tag{344}$$

where $1_A(\cdot)$ denotes the indicator function for set A, $B(\boldsymbol{x}, r)$ constructs a Euclidean ball centered at $\boldsymbol{x}$ with radius r, and $\delta^k(\hat{\boldsymbol{x}}_j)$ is the distance to the k-th nearest neighbour in $\{\hat{\boldsymbol{x}}_i\}_{i=1}^m$, excluding itself. The precision metric is given by

$$\text{precision}(\mathbb{P}_{data}, \mathbb{P}_\theta) = \frac{1}{n} \sum_{i=1}^{n} 1_{\bigcup_{j=1}^m B(\hat{\boldsymbol{x}}_j, \delta^k(\hat{\boldsymbol{x}}_j))}(\boldsymbol{x}_i). \tag{345}$$

Similarly, coverage is given by

$$\text{coverage}(\mathbb{P}_{data}, \mathbb{P}_\theta) = \frac{1}{m} \sum_{j=1}^{m} \max_{i=1,\ldots,n} 1_{B(\hat{\boldsymbol{x}}_j, \delta^k(\hat{\boldsymbol{x}}_j))}(\boldsymbol{x}_i). \tag{346}$$

Likewise, the recall metric is given by

$$\text{recall}(\mathbb{P}_{data}, \mathbb{P}_\theta) = \frac{1}{m} \sum_{j=1}^{m} 1_{\bigcup_{i=1}^n B(\boldsymbol{x}_i, \delta^k(\boldsymbol{x}_i))}(\hat{\boldsymbol{x}}_j). \tag{347}$$

We used $k = 5$ and 10k samples throughtout, as standard.

On reporting. When reporting on these metrics like in Table 1 we use **bold font** to denote the best performance with a 1% error range. More formally, suppose we have a series of n data points $\{x_i\}_{i=1}^n$ that is totally ordered by some relation R. We say will denote a query point x_i with **bold font** if the *range-normalized absolute percentage error* is less than $\epsilon > 0$, *i.e.*,

$$\frac{|\max_j x_j - x_i|}{\max_j x_j - \min_k x_k} < \epsilon. \tag{348}$$

In our experiments we report $\epsilon = 0.01$.

I.1.3 Hyperparameters

We follow the suggestion of Wallace et al. (2023) and report results with EDICT using the hyperparameter $p = 0.93$. For BDIA, the original paper recommends $\gamma = 1.0$ for unconditional image generation (Zhang et al., 2024, Section 6.1). However, we found $\gamma = 0.5$ to yield better performance, this corroborates with the findings of Wang et al. (2024).

I.2 Conditional image generation

I.2.1 Diffusion model

We make use of Stable Diffusion v1.5 (Rombach et al., 2022) a pre-trained *latent diffusion model* (LDM) model. We also use the scaled linear noise schedule given as

$$\beta_i = \left(\sqrt{\frac{\hat{\beta}_0}{T}} + \frac{i-1}{\sqrt{T}(T-1)} \left(\sqrt{\hat{\beta}_1} - \sqrt{\hat{\beta}_0} \right) \right)^2. \tag{349}$$

We convert this into a continuous time representation via the details in Appendix G.1 following Song et al. (2021b). For this experiment we used $\hat{\beta}_0 = 0.00085$ and $\hat{\beta}_1 = 0.012$. To ensure numerical stability due to $\frac{1}{\sigma_t}$ terms we solve the probability flow ODE in reverse-time on the time interval $[\epsilon, 1]$ with $\epsilon = 0.0002$. This is a common choice to make in practice see Song et al. (2023).

Numerical schemes. We set the last two steps of Rex schemes to be either Euler or Euler-Maruyama for better stability near time 0.

I.2.2 Metrics

As mentioned in the main paper we use the CLIP Score (Hessel et al., 2021) PickScore (Kirstain et al., 2023), and Image Reward metrics (Xu et al., 2023) to asses the ability of the text-to-image conditional generation task. We calculate each by comparing the sampled image and the given text prompt used to produce the image. We then report the average over the 1000 samples.

CLIP score. The CLIP score measures the cosine similarity between the text and visual embeddings with pretrained CLIP model (Radford et al., 2021) denoted as

$$\text{CLIPScore}(\boldsymbol{x}, \boldsymbol{c}) = \max\left\{\frac{\langle \mathcal{E}_I(\boldsymbol{x}), \mathcal{E}_C(\boldsymbol{c})\rangle}{\|\mathcal{E}_I(\boldsymbol{x})\|\|\mathcal{E}_C(\boldsymbol{c})\|}, 0\right\}, \tag{350}$$

where $\mathcal{E}_I : \mathbb{R}^d \to V$ is the image embedder and $\mathcal{E}_C : \mathbb{R}^{d'} \to V$ is the caption embedder; and where $\boldsymbol{x}$ is the query image and $\boldsymbol{c}$ is the query caption. Thus this metric aims to measure how well our generated images align with their prompt. In particular, we use the `ViT-L/14` backbone trained by OpenAI.

PickScore. Similar to CLIP score, PickScore finetunes a CLIP-H model on their proposed Pick-a-Pic dataset which purportedly aligns better with human preference over CLIP score.

Image Reward. Image Reward (Xu et al., 2023) is the newest of the three metrics and uses BLIP (Li et al., 2022) over CLIP as the backbone and finetunes the model using reward model training. The resulting metrics achieves state-of-the-art alignment with human preferences.

On reporting. When reporting on these metrics like in Table 2 we use **bold font** to denote the best performance with a 1% error range. In our experiments we report $\epsilon = 0.01$.

I.2.3 Hyperparameters

We follow the suggestion of Wallace et al. (2023) and report results with EDICT using the hyperparameter $p = 0.93$. For BDIA, the original paper recommends $\gamma = 0.5$ for text-to-image generation (Zhang et al., 2024, Section 6.1). We also ran BDIA with $\gamma = 0.96$ as suggested by Wang et al. (2024).

I.3 Interpolation

Diffusion model. We make use of a pre-trained DDPM (Ho et al., 2020) model trained on the CelebA-HQ 256×256 dataset (Karras et al., 2018). We used linear noise schedule from (Ho et al., 2020). We convert this into a continuous time representation via the details in Appendix G.1 following Song et al. (2021b). For this experiment we used $\hat{\beta}_0 = 0.0001$ and $\hat{\beta}_1 = 0.2$. For the face pairings we followed Blasingame & Liu (2024a;c) and used the FRLL (DeBruine & Jones, 2017) dataset.

Notably, we used the noise prediction parameterization rather than data prediction as we found that it performed better for editing. This is likely due to the singularity of the $\frac{1}{\sigma_t}$ terms as $t \to 0$. Within this parameterization we could use the time interval $[0, 1]$ instead of $[\epsilon, 1]$ like in previous experiments with data prediction models.

I.4 Image editing

For our experiments we drew used 100 text-image pairs with edit instructions and evaluated all the metrics over that.

I.5 Boltzmann sampling

I.5.1 Datasets

We follow the same training, validation, and test split used by Tan et al. (2025) to evaluate *Rex* on equilibrium conformation sampling tasks, with a focus on tri-alanine. These datasets are obtained from implicit solvent molecular dynamics (MD) simulations. In particular, a single MCMC chain is decomposed into 10^5, 2×10^4, and 10^4 samples for training, validation, and testing. The training and validation data are each taken from contiguous regions of the chain to simulate the realistic scenario wherein a pre-existing MCMC trajectory exists and one would like to use a Boltzmann generator to continue generating samples. Earlier parts of the trajectory under-sampled specific modes enabling a biased training set which we aim to debias through access to the energy function and SNIS. All MD simulations were run for 1 µs with a timestep of 1 fs at temperatures of 300K and 310K for alanine dipeptide and tri-alanine mirroring those done by Klein & Noé (2025).

Tri-alanine. For the tri-alanine dataset, we follow the splitting procedure of Tan et al. (2025). The first 100,000 datapoints after burn-in are used as the training set, the next 10,000 points in the (subsampled) chain are used for validation, and a random selection of the rest of the chain is used as test samples. This creates a biased training set relative to the test set, and is realistic in the setting where we would like to draw samples more efficiently from an existing MD chain.

I.5.2 Training details

Architecture. We adopt a DiT backbone (Peebles & Xie, 2023) with the details shown below in Table 6.

Table 6: Overview of architecture configurations.

Parameter	DiT
Hidden size	192
Blocks	6
Heads	6
Conditional dimension	64
# of Parameters (M)	3.1 M

Training configuration. All models were trained us an exponential moving average on the weights with a decay rate of 0.999. For evaluation we generated 10^4 proposal samples and we used the same number of re-sampling and computing all metrics.

Hyperparameters. We used AdamW algorithm (Loshchilov & Hutter, 2017) to perform gradient descent with a learning rate of 5×10^{-4}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 10^{-4}. A cosine annealing schedule was applied to the learning rate with a warm-up phase covering 5% of the training iterations. We trained for 3000 epochs.

I.5.3 Metrics

We evaluate model performance using both sample-based metrics and metrics that assess energy distributions. We enumerate these in greater detail below.

Effective sample size. We compute the effective sample size (ESS) using Kish's formula (Kish, 1957), *i.e.*, given $N \in \mathbb{N}$ generated particles with unnormalized importance weights $\{w_i\}_{i=1}^N \subsetneq \mathbb{R}_{\geq 0}$ we have

$$ESS(\{w_i\}_i = 1^N) \coloneqq \frac{1}{N} \frac{1}{\sum_{i=1}^N w_i^2} \left(\sum_{i=1}^N w_i \right)^2. \tag{351}$$

The ESS is a measure of how many independent and equally-weighted samples would provide equivalent statistical power to the weighted sample.

2-Wasserstein energy distance ($\mathcal{E}$-$\mathcal{W}_2$). To compare energy distributions we measure the 2-Wasserstein distance between them which for two probability measures μ, ν on $\mathbb{R}$ over energy values is given as

$$\mathcal{E}\text{-}\mathcal{W}_2(\mu, \nu)^2 = \inf_{\gamma \in \Pi(\mu,\nu)} \int_{\mathbb{R}\times\mathbb{R}} |x - y|^2 \, \mathrm{d}\gamma(x, y), \tag{352}$$

where $\Pi(\mu, \nu)$ is the set of all couplings whose marginals are μ and ν. The 2-Wasserstein distance is an integral probability metric which measures captures both the difference in *location* and *shape* of two distributions.

Torus 2-Wasserstein distance ($\mathbb{T}$-$\mathcal{W}_2$). To measure structural similarity in torsional space, we compute the 2-Wasserstein distance over dihedral angles. For a molecule with $L \in \mathbb{N}$ residues, we define the dihedral vector as

$$\text{Dihedrals}(\boldsymbol{x}) = (\phi_1, \psi_1, \ldots, \phi_{L-1}, \psi_{L-1}) \in [0, 2\pi)^{2(L-1)}. \tag{353}$$

Thus given the torus geometry a natural cost function arises as the minimal signed angle difference, *i.e.*,

$$c_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})^2 = \sum_{i=1}^{2} \left[(\text{Dihedrals}(\boldsymbol{x})_i - \text{Dihedrals}(\boldsymbol{y})_i + \pi) \mod 2\pi - \pi \right]^2. \tag{354}$$

Thus the torus 2-Wasserstein between two distributions μ, ν on $[0, 2\pi)^{2(L-1)}$ is then

$$\mathbb{T}\text{-}\mathcal{W}_2(\mu, \nu)^2 = \inf_{\gamma \in \Pi(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} c_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})^2 \, \mathrm{d}\gamma(\boldsymbol{x}, \boldsymbol{y}). \tag{355}$$

I.6 Hardware

All experiments were run using a single NVIDIA H100 80 GB GPU.

I.7 Repositories

In our empirical studies we made use of the following resources and repositories:

1. `google/ddpm-celebahq-256` (DDPM Model)
2. `stable-diffusion-v1-5/stable-diffusion-v1-5` (Stable Diffusion v1.5)
3. `zituitui/BELM` (Implementation of BELM, EDICT, and BDIA)
4. `google-research/torchsde` (Brownian Interval)
5. `layer6ai-labs/dgm-eval` (FD, FD_∞, KD, Density, and Coverage metrics)
6. `torchmetrics` (CLIP score)
7. `zai-org/ImageReward` (Image Reward)
8. `yuvalkirstain/pickscore` (PickScore)
9. `timbrooks/instructpix2pix-clip-filtered` (InstructPix2Pix dataset)

J Additional Results

J.1 Conditional Image Generation

J.2 Interpolation

We explore interpolating between the inversions of two images, a difficult problem as the inverted space is often non-Gaussian (Blasingame & Liu, 2024b). We illustrate an example of this in Figure 7 exploring interpolation with an unconditional DDPM model. We notice the that stochastic Rex has much better interpolations properties than both ODE inversions corroborating with Nie et al. (2024). Both ODE variants seem to fail quite noticeably, unable to smoothly interpolate between the two samples. *N.B.*, we noticed that the inverted samples with ShARK had variance much closer to one, whereas the other inverted samples had much larger variance, likely contributing to the distortions.

(a) DDIM (b) EDICT (c) BDIA (d) BELM (e) Rex

Figure 6: Qualitative comparison of text-to-image conditional sampling with different reversible solvers with Stable Diffusion v1.5 (512×512) and 10 discretization steps. Prompts from top to bottom are: "White plate with fried fish and lemons sitting on top of it.", "A lady enjoying a meal of some sort.", and "A young boy riding skis with ski poles.".

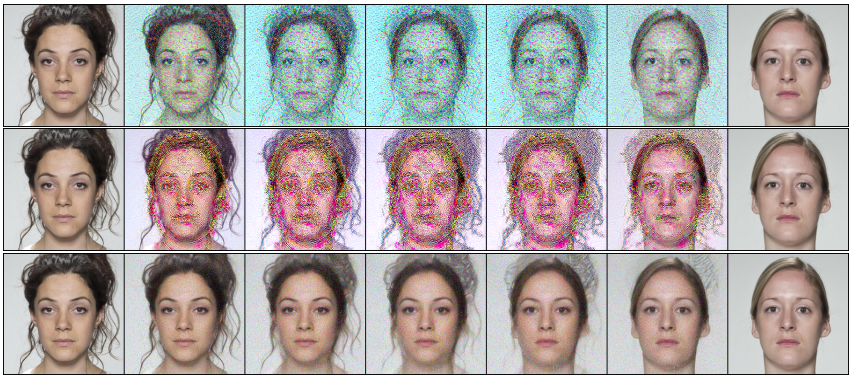


Figure 7: Unconditional interpolation between two real images from FRLL (DeBruine & Jones, 2017) with a DDPM model trained on CelebA-HQ. Top row is BELM, middle is Rex (Euler), and bottom is Rex (ShARK). 50 steps used for each method.