

000
001
002
003

REX: REVERSIBLE SOLVERS FOR DIFFUSION MODELS

004 **Anonymous authors**
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053

Paper under double-blind review

ABSTRACT

Diffusion models have quickly become the state-of-the-art for numerous generation tasks across many different applications. Encoding samples from the data distribution back into the model’s underlying prior distribution, often called the *inversion* of diffusion models, is an important task that arises from many downstream applications. Prior approaches for solving this task, however, are often simple heuristic solvers that come with several drawbacks in practice. In this work, we propose a new family of solvers for diffusion models by exploiting the connection between this task and the broader study of *algebraically reversible* solvers for differential equations. In particular, we construct a family of reversible solvers using an application of Lawson methods to construct exponential Runge–Kutta methods for the diffusion models; we call this family of reversible exponential solvers *Rex*. In addition to a rigorous theoretical analysis of the proposed solvers, we also demonstrate the utility of the methods through a variety of empirical illustrations.

1 INTRODUCTION

Diffusion models have quickly become the state-of-the-art in generation tasks across many varied modalities from images (Rombach et al., 2022) and video (Blattmann et al., 2023) to protein generation (Skreta et al., 2025b) and biometrics (Blasingame & Liu, 2024d). The sampling process of diffusion models is done through numerically solving an Itô *stochastic differential equation* (SDE) or related *ordinary differential equation* (ODE) which describes the evolution of a sample drawn for some prior noise distribution to the data distribution. Inversion of the sampling procedure, *i.e.*, constructing a bijective map from the data distribution back to the prior distribution, is invaluable for many downstream applications.

While the true (stochastic) flow maps of diffusion models do provide such a bijection, in practice we need to solve such models numerically, thereby incurring truncation errors breaking the bijection. Thus to obtain the *exact inversion* of a diffusion model we are looking for a scheme which is algebraically reversible. *I.e.*, we would like a numerical scheme which enables us to move between the data and prior distribution without any reconstruction errors. Recently, several works have explored solving this problem for the probability flow ODE, namely, EDICT (Wallace et al., 2023), BDIA (Zhang et al., 2024), and BELM (Wang et al., 2024).

However, 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. To the best of our knowledge there does not currently exist a scheme for exact inversion for diffusion SDEs *without* storing the entire trajectory of the Brownian motion in memory *à la* Wu & la Torre (2023) which is trivially reversible, but not the type of reversibility we are interested with.

To address these issues we propose *Rex*, a family of reversible solvers for diffusion models which can

1. Work for both the probability flow ODE and reverse-time SDE with both data and noise prediction parameterizations,
2. Obtain an arbitrarily high order of convergence (in the ODE case), and
3. Exactly invert a diffusion SDE *without* storing the entire realization Brownian motion in memory.

054 **2 PRELIMINARIES**

056 **Diffusion models.** Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al.,
 057 2021a;b) have quickly become one of the most popular paradigms for constructing *generative models*.
 058 Consider the following Itô stochastic differential equation (SDE) defined on time interval $[0, T]$:

$$d\mathbf{X}_t = f(t)\mathbf{X}_t dt + g(t) d\mathbf{W}_t, \quad (1)$$

060 where $f, g \in \mathcal{C}^\infty([0, T])$ ¹ form the drift and diffusion coefficients of the SDE and where $\{\mathbf{W}_t\}_{t \in [0, T]}$
 061 is the standard Brownian motion on the time interval. The coefficients f, g are chosen such that the
 062 SDE maps clean samples from the data distribution $\mathbf{X}_0 \sim q(\mathbf{X})$ at time 0 to an isotropic Gaussian
 063 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
 064 monotonically decreasing function α_t and strictly monotonically increasing function σ_t , the drift and
 065 diffusion coefficients are found to be

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

066 where with abuse of notation $\dot{\sigma}_t^2$ denotes the time derivative of the function σ_t^2 (Lu et al., 2022b;
 067 Kingma et al., 2021)—this ensures that $\mathbf{X}_t \sim \mathcal{N}(\alpha_t \mathbf{X}_0, \sigma_t^2 \mathbf{I})$. However, we wish to map from *noise*
 068 back to *data*, as such we employ the result of Anderson (1982) to construct the *reverse-time* diffusion
 069 SDE of Equation (1), which is found to be

$$d\mathbf{X}_t = [f(t)\mathbf{X}_t - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{X}_t)] dt + g(t) d\bar{\mathbf{W}}_t, \quad (3)$$

070 where dt is a *negative* timestep, $\{\bar{\mathbf{W}}_t\}_{t \in [0, T]}$ is the standard Brownian motion in reverse-time,
 071 and $p_t(\mathbf{x}) := p(t, \mathbf{x})$ is the marginal density function. Then, if we can learn the *score function*
 072 $(t, \mathbf{x}) \mapsto \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ (Song et al., 2021b)—or some other *equivalent* reparameterization, e.g., noise
 073 prediction (Song et al., 2021a; Ho et al., 2020) or data prediction (Kingma et al., 2021)—we can then
 074 draw samples from our data distribution $q(\mathbf{X})$ by first sampling some $\mathbf{X}_T \sim p(\mathbf{X})$ from the Gaussian
 075 prior and then employing a numerical SDE solver, e.g., Euler-Maruyama, to solve Equation (3) in
 076 reverse-time. Notably, through careful massaging of the Fokker-Planck-Kolomogorov equation for
 077 the marginal density, one can construct an ODE which is equivalent in *distribution* to Equation (3)
 078 (Song et al., 2021b; Maoutsu et al., 2020), yielding the *highly popular probability flow ODE*

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{g^2(t)}{2}\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t). \quad (4)$$

079 **Reversible solvers for neural differential equations.** Recently, researchers studying *neural*
 080 *differential equations* have begun to propose several *algebraically reversible solvers* as an alternative
 081 to both traditional discretize-then-optimize and optimize-then-discretize (the continuous adjoint
 082 equations) (Kidger, 2022, Chapters 5.1 & 5.2) which are used to perform backpropagation through
 083 the neural differential equation. Consider some prototypical neural ODE of the form $\dot{\mathbf{x}}_t = \mathbf{u}_\theta(t, \mathbf{x}_t)$
 084 with vector field $\mathbf{u}_\theta \in \mathcal{C}^r(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^d)$ which satisfies the usual regularity conditions. Then consider
 085 a single-step numerical scheme of the form $\mathbf{x}_{n+1} = \mathbf{x}_n + \Phi_h(t_n, \mathbf{x}_n, \mathbf{u}_\theta)$. Every numerical scheme
 086 Φ is reversible in the sense that we can rewrite the forward step as an implicit scheme of the form
 087 $\mathbf{x}_n = \mathbf{x}_{n+1} - \Phi_h(t_n, \mathbf{x}_n, \mathbf{u}_\theta)$; however, this requires fixed point iteration² and is both *approximate*
 088 and computationally *expensive*. This type of reversibility is known as *analytic reversibility* within
 089 the neural differential equations community (Kidger, 2022, Section 5.3.2.1). What we would prefer,
 090 however, is a form of reversibility that can be expressed in *closed-form*.

091 Beyond symplectic solvers (Vogelaere, 1956) which are trivially reversible³, several algebraically
 092 reversible solvers have been proposed in light of the large popularity of neural ODEs. Namely, the
 093 following methods have been proposed: the *asynchronous leapfrog method* (Mutze, 2013; Zhuang
 094 et al., 2021), *reversible Heun method* (Kidger et al., 2021), and *McCallum-Foster method* (McCallum
 095 & Foster, 2024). The last of these is of particular interest to us, as it is the *only* algebraically
 096 reversible ODE solver to have a non-trivial region of stability and arbitrarily high convergence
 097 order. As McCallum & Foster (2024) simply refer to their method as *reversible X* where X is the
 098 underlying single-step solver, we opt to refer to their method as the *McCallum-Foster method* which
 099 we summarize below in Definition 2.1.

100 ¹We let $\mathcal{C}^r(X; Y)$ denote the class of r -th differentiable functions from X to Y . If Y is omitted then $Y = \mathbb{R}$.

101 ²If the step size h is small enough.

102 ³Due to symplectic integrators being developed for solving Hamiltonian systems, they are intrinsically
 103 reversible by construction (Greydanus et al., 2019).

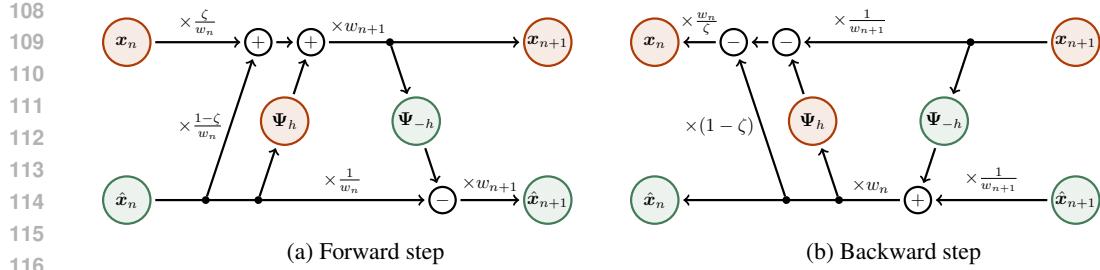


Figure 1: The computation graph of the Rex solver. Here Ψ_h denotes an exponentially weighted Runge-Kutta scheme (cf. Section 3.1) or exponential stochastic Runge-Kutta scheme (cf. Section 3.2), $\zeta \in (0, 1]$ is a coupling parameter, and $\{w_n\}_{n=1}^N$ denotes the set of weighting variables derived from the exponential schemes. The particular values of w_n are discussed in Proposition 3.3. The visualization of the computation graph is inspired by [McCallum & Foster \(2024, Figure 2\)](#).

Definition 2.1 (McCallum-Foster method). Initialize $\hat{x}_0 = x_0$ and let $\zeta \in (0, 1]$. Consider a step size of h , then a forward step of the McCallum-Foster method is defined as

$$\begin{aligned} x_{n+1} &= \zeta x_n + (1 - \zeta) \hat{x}_n + \Phi_h(t_n, \hat{x}_n), \\ \hat{x}_{n+1} &= \hat{x}_n - \Phi_{-h}(t_{n+1}, x_{n+1}), \end{aligned} \quad (5)$$

and the backward step is given as

$$\begin{aligned} \hat{x}_n &= \hat{x}_{n+1} + \Phi_{-h}(t_{n+1}, x_{n+1}), \\ x_n &= \zeta^{-1} x_{n+1} + (1 - \zeta^{-1}) \hat{x}_n - \zeta^{-1} \Phi_h(t_n, \hat{x}_n). \end{aligned} \quad (6)$$

3 REX

In this section we introduce the *Rex* family of reversible solvers for diffusion models. Whilst one could straightforwardly apply a pre-existing reversible solver like asynchronous leapfrog, reversible Heun, or the McCallum-Foster method directly to the probability flow ODE in Equation (4), there are several reasons to consider an alternative approach. Stepping back from reversible solvers for a moment, we consider the broader literature of constructing numerical schemes for diffusion models. It is well known that we can exploit the structure of the drift and diffusion coefficients, *i.e.*, $f(t)$ and $g(t)$, to remove the discretization error from the linear term and transform the stiff ODE into a non-stiff form ([Lu et al., 2022b; Zhang & Chen, 2023](#)); a similar idea also holds for the reverse-time-diffusion SDE (see [Lu et al., 2022a; Gonzalez et al., 2024; Blasingame & Liu, 2024a](#)). Moreover, recall that the definitions of the drift and diffusion coefficients contain the time derivatives of the noise schedule (α_t, σ_t) , this structure enables us to greatly simplify the ODE/SDE and express a number of terms in closed-form again reducing approximation errors.

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 $\{w_n\}$ and the underlying numerical scheme Ψ_h . The rest of this section is organized as follows: first we discuss applying the exponential integrators to the probability flow ODEs (see Section 3.1), then the reverse-time SDEs (see Section 3.2), and lastly we present the general Rex scheme (see Section 3.3).

3.1 PROBABILITY-FLOW ODE

Before constructing Rex we must first discuss the construction of Ψ_h from Φ_h and how to derive the reparameterized ODE, *i.e.*, step 1 in Figure 2. In this section we review how to reparameterize the ODE in Equation (4) into this more convenient form.

Generalized nomenclature for data and noise prediction models. 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 $x_{T|t}(\mathbf{x}) = \mathbb{E}[\mathbf{X}_T | \mathbf{X}_t =$

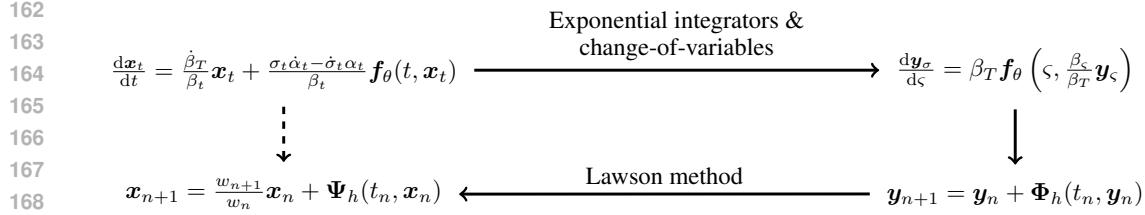


Figure 2: Overview of the construction of Ψ for the probability flow ODE from an underlying Runge-Kutta scheme Φ for the reparameterized ODE in Equation (8). The parameters β_t and ς_t are chosen to suit the data or noise prediction parameterizations (cf. Section 3.1). The graph holds for the SDE case *mutatis mutandis*.

and write data prediction model as $x_{0|t}(x) = \mathbb{E}[X_0 | X_t = x]$. In this work we consider *both* a trained noise and data prediction model which we will denote generally by the neural network $f_\theta(t, x)$. Additionally, we place the usual regularity constraints (cf. Lu et al., 2022b, Appendix B.1) on the model to ensure the existence and uniqueness of the ODE/SDE solutions. It is well known (Blasingame & Liu, 2025, Equation (19)) that the ODE in Equation (4) can be rewritten as

$$\frac{dx_t}{dt} = \frac{\dot{\beta}_t}{\beta_t} x_t + \frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\beta_t} f_\theta(t, x_t), \quad (7)$$

where $\beta_t = -\alpha_t$ for noise prediction with and $\beta_t = \sigma_t$ for target prediction. This choice of β and f_θ thus depends on the particulars of the noise or data reparameterization.

Remark 3.1. 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.⁴

It is well observed that the structure of the ODE in Equation (7) 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 $f(t)x_t$ linear term along with a time reparameterization which will simplify the construction of the reversible solver. To achieve the time reparameterization 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 3.1, in Section C.1.1 we provide the full derivation of this result.

Proposition 3.1 (Reparameterization of the probability flow ODE). *The probability flow ODE in Equation (7) can be rewritten in ς_t as*

$$\frac{dy_\varsigma}{d\varsigma} = \beta_T f_\theta\left(\varsigma, \frac{\beta_\varsigma}{\beta_\varsigma} y_\varsigma\right), \quad (8)$$

where $y_t = \frac{\beta_T}{\beta_t} x_t$.

The remaining step to constructing Rex is to perform a similar process but for an underlying explicit Runge-Kutta scheme by making use of Lawson methods (a particular class of exponential integrators) (Lawson, 1967; Hochbruck et al., 2020). However, since *both* the ODE and SDE version of Rex share the same computational graph, we will delay this presentation until we have discussed the SDE case.

3.2 REVERSE-TIME DIFFUSION SDE

Unlike with the ODE scenario the forms of the data and noise prediction formulations differ more significantly. As such we opt to focus only on the data prediction formulation which slightly less complicated and leave the details on the noise prediction formulation to Appendix C.2. It is well known (Lu et al., 2022a) that the reverse-time diffusion SDE in Equation (3) can be rewritten in terms of the data prediction model as

$$dX_t = \left[\left(f(t) + \frac{g^2(t)}{\sigma_t^2} \right) X_t - \frac{\alpha_t g^2(t)}{\sigma_t^2} x_{0|t}^\theta(X_t) \right] dt + g(t) d\bar{W}_t. \quad (9)$$

⁴I.e., sampling in forward-time such that $X_1 \sim q(X)$ and $X_0 \sim p(X)$.

216 Remarkably, following a similar derivation to the one above for the probability flow ODE yields a
 217 time-changed SDE with a very similar form to the one above, sans the Brownian motion term and
 218 different weighting terms. We present this result in Proposition 3.2 with the full proof in Section C.2.2.
 219

220 **Proposition 3.2** (Time reparameterization of the reverse-time diffusion SDE). *The reverse-time SDE
 221 in Equation (9) can be rewritten in terms of the data prediction model as*

$$222 \quad d\mathbf{Y}_\varrho = \frac{\sigma_T}{\gamma_T} \mathbf{x}_{0|\varrho}^\theta \left(\frac{\gamma_T \sigma_\varrho}{\sigma_T \gamma_\varrho} \mathbf{Y}_\varrho \right) d\varrho + \frac{\sigma_T}{\gamma_T} d\mathbf{W}_\varrho, \quad (10)$$

224 where $\mathbf{Y}_t = \frac{\sigma_T^2 \alpha_t}{\sigma_t^2 \alpha_T} \mathbf{X}_t$ and $\varrho_t := \frac{\alpha_t^2}{\sigma_t^2}$.
 225

226 **Stochastic Runge-Kutta.** Before constructing a reversible solver for the reverse-time SDE in
 227 Equation (10), we will zoom out to contextualize the discussion within the study of neural SDEs
 228 and to introduce *stochastic Runge-Kutta* (SRK) methods. Constructing a numerical scheme for
 229 SDEs is greatly more complicated than ODEs due to the complexities of stochastic processes
 230 and in particular stochastic integrals. Unlike numerical schemes for ODEs which are usually
 231 built upon truncated Taylor expansions, SDEs require constructing truncated Itô or Stratonovich-
 232 Taylor expansions (Kloeden & Platen, 1991) which results in numerous iterated stochastic integrals.
 233 Approximating these iterated integrals, or equivalently Lévy areas, of Brownian motion is quite
 234 difficult (Clark & Cameron, 2005; Mrongowius & Rößler, 2022); however, SDEs with certain
 235 constraints on the diffusion term may use specialized solvers to further achieve a strong order of
 236 convergence with simple approximations of these iterated stochastic integrals. As such there are
 237 several ways to express SRK methods depending on the choice of approximating these iterated
 238 integrals. We choose to follow the work of Foster et al. (2024) which makes usage of the *space-time*
 239 *Lévy area* in constructing such methods. The space-time Lévy area (see Foster et al., 2020, Definition
 240 3.5; cf. Rößler, 2010) is defined below in Definition 3.2.

240 **Definition 3.2** (Space-time Lévy area). The rescaled space-time Lévy area of a Brownian motion
 241 $\{W_t\}$ on the interval $[s, t]$ corresponds to the signed area of the associated bridge process

$$242 \quad H_{s,t} := \frac{1}{h} \int_s^t \left(W_{s,u} - \frac{u-s}{h} W_{s,t} \right) du, \quad (11)$$

243 where $h := t - s$ and $W_{s,u} = W_u - W_s$ for $u \in [s, t]$.
 244

246 In particular, for additive-noise SDEs which our SDE in Equation (10) is, the Itô and Stratonovich
 247 integrals coincide and the numerical scheme is significantly simpler, for more details we refer to
 248 Appendix B.
 249

250 3.3 THE REX SOLVER

252 Equipped with both Proposition 3.1 and Proposition 3.2 we are now ready to construct Rex. The key
 253 idea is to construct a reversible scheme from an explicit (S)RK scheme (we provide more detail in
 254 Appendix B) for the reparameterized differential equation using the McCallum-Foster method and
 255 then apply Lawson methods to bring the scheme back to the original state variable, cf. Figure 2.

256 We present the full scheme for the Rex solver below in Proposition 3.3 with the full derivation found
 257 in Appendix C.

258 **Proposition 3.3** (Rex). *Without loss of generality let Φ denote an explicit SRK scheme for the SDE
 259 in Equation (10) 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
 260 \mathbf{W} be the Brownian motion over time variable ς . Then the reversible solver constructed from Φ in
 261 terms of the underlying state variable \mathbf{X}_t is given by the forward step*

$$262 \quad \mathbf{X}_{n+1} = \frac{w_{n+1}}{w_n} \left(\zeta \mathbf{X}_n + (1 - \zeta) \hat{\mathbf{X}}_n \right) + w_{n+1} \Psi_h(\varsigma_n, \hat{\mathbf{X}}_n, \mathbf{W}_n(\omega)), \\ 263 \quad \hat{\mathbf{X}}_{n+1} = \frac{w_{n+1}}{w_n} \hat{\mathbf{X}}_n - w_{n+1} \Psi_{-h}(\varsigma_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_n(\omega)), \quad (12)$$

266 and backward step

$$267 \quad \hat{\mathbf{X}}_n = \frac{w_n}{w_{n+1}} \hat{\mathbf{X}}_{n+1} + w_n \Psi_{-h}(\varsigma_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_n(\omega)), \\ 268 \quad \mathbf{X}_n = \frac{w_n}{w_{n+1}} \zeta^{-1} \mathbf{X}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{X}}_n - w_n \zeta^{-1} \Psi_h(\varsigma_n, \hat{\mathbf{X}}_n, \mathbf{W}_n(\omega)), \quad (13)$$

270 with step size $h := \varsigma_{n+1} - \varsigma_n$ and where Ψ denotes the following scheme
 271

$$272 \hat{\mathbf{Z}}_i = \frac{1}{w_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[a_{ij} \mathbf{f}^\theta \left(\varsigma_n + c_j h, w_{\varsigma_n + c_j h} \hat{\mathbf{Z}}_j \right) \right] + a_i^W \mathbf{W}_n(\omega) + a_i^H \mathbf{H}_n(\omega), \\ 273 \Psi_h(\varsigma_n, \mathbf{X}_n, \mathbf{W}_\varrho(\omega)) = h \sum_{j=1}^s \left[b_i \mathbf{f}^\theta \left(\varsigma_n + c_i h, w_{\varsigma_n + c_i h} \hat{\mathbf{Z}}_j \right) \right] + b^W \mathbf{W}_n(\omega) + b^H \mathbf{H}_n(\omega), \\ 274 \quad 275 \quad 276 \quad 277 \quad 278 \quad 279 \quad 280 \quad 281 \quad 282 \quad 283 \quad 284 \quad 285 \quad 286 \quad 287 \quad 288 \quad 289 \quad 290 \quad 291 \quad 292 \quad 293 \quad 294 \quad 295 \quad 296 \quad 297 \quad 298 \quad 299 \quad 300$$

288 where \mathbf{f}^θ denotes the data prediction model, $w_n = \frac{\sigma_n}{\gamma_n}$ and $\varsigma_t = \varrho_t$. The ODE case is recovered
 289 for an explicit RK scheme Φ for the ODE in Equation (70) with $w_n = \sigma_n$ and $\varsigma_t = \gamma_t$. For noise
 290 prediction models we have \mathbf{f}^θ denoting the noise prediction model with $w_n = \alpha_n$ and $\varsigma_t = \frac{\sigma_n}{\alpha_n}$.
 291

292 We still have yet to address how to construct an *algebraically reversible* scheme for a *stochastic*
 293 process, but merely stated it above in Proposition 3.3, we will now, however, justify our design
 294 decisions above. The key idea is to use the *same* realization of the Brownian motion in both the
 295 forward pass or backward pass. This has been explored in prior works studying the continuous
 296 adjoint equations for neural SDEs (Li et al., 2020; Kidger et al., 2021) and essentially amounts to
 297 fixing the realization of the Brownian motion along with clever strategies for reconstructing the same
 298 realization. Formally, let $(\Omega, \mathcal{F}, \mathbb{P})$ be the probability space and let $W_t : \Omega \rightarrow \mathbb{R}^{d_w}$ be the standard
 299 Brownian motion on $[0, T]$. Then for each reversible solve we fix an $\omega \in \Omega$. This can be justified if
 300 we view the SDE from a roughs path perspective, *i.e.*, the Itô-Lyons map (Lyons, 1998) provides a
 301 deterministic continuous map from the initial condition of the SDE and realization of the Brownian
 302 motion to the solution trajectory, see Appendix F for a more detailed explanation.
 303

304 **Numerical simulation of the Brownian motion.** The naïve way to fix the realization of the
 305 Brownian motion for both the forward pass is to simply store the entire realization of the Brownian
 306 motion in system memory, *i.e.*, record $\{W_n(\omega)\}_{n=1}^N$ à la Wu & la Torre (2023).⁵ However, recent
 307 work by Li et al. (2020); Kidger et al. (2021); Jelinčić et al. (2024) have proposed much more elegant
 308 solutions which enable one to recalculate *any* realization of the Brownian motion from a single seed
 309 given access to a splittable *pseudo-random number generator* (PRNG) (Salmon et al., 2011). *N.B.*,
 310 we discuss the more nuanced technical details of such approaches in Appendix G, for now it suffices
 311 to say we adopt a more elegant solution to reconstructing the Brownian motion in the backward step.
 312

313 4 THEORETICAL RESULTS

314 4.1 CONVERGENCE ORDER AND STABILITY

315 A nice property of the McCallum-Foster is that the the convergence order of the underlying explicit
 316 RK scheme Φ is inherited by the resulting reversible scheme McCallum & Foster (2024, Theorem
 317 2.1). However, does this property hold true for Rex? Fortunately, it does indeed hold true which we
 318 show in Theorem 4.1 with the proof provided in Appendix D.2.

319 **Theorem 4.1** (Rex is a k -th order solver). *Let Φ be a k -th order explicit Runge-Kutta scheme for*
 320 *the reparameterized probability flow ODE in Equation (70) with variance preserving noise schedule*
 321 *(α_t, σ_t) . Then Rex constructed from Φ is a k -th order solver, *i.e.*, given the reversible solution*
 322 *$\{\mathbf{x}_n, \hat{\mathbf{x}}_n\}_{n=1}^N$ and true solution \mathbf{x}_{t_n} we have*

$$323 \quad 324 \quad 325 \quad 326 \quad 327 \quad 328 \quad 329 \quad 330 \quad 331 \quad 332 \quad 333 \quad 334 \quad 335 \quad 336 \quad 337 \quad 338 \quad 339 \quad 340 \quad 341 \quad 342 \quad 343 \quad 344 \quad 345 \quad 346 \quad 347 \quad 348 \quad 349 \quad 350 \quad 351 \quad 352 \quad 353 \quad 354 \quad 355 \quad 356 \quad 357 \quad 358 \quad 359 \quad 360 \quad 361 \quad 362 \quad 363 \quad 364 \quad 365 \quad 366 \quad 367 \quad 368 \quad 369 \quad 370 \quad 371 \quad 372 \quad 373 \quad 374 \quad 375 \quad 376 \quad 377 \quad 378 \quad 379 \quad 380 \quad 381 \quad 382 \quad 383 \quad 384 \quad 385 \quad 386 \quad 387 \quad 388 \quad 389 \quad 390 \quad 391 \quad 392 \quad 393 \quad 394 \quad 395 \quad 396 \quad 397 \quad 398 \quad 399 \quad 400 \quad 401 \quad 402 \quad 403 \quad 404 \quad 405 \quad 406 \quad 407 \quad 408 \quad 409 \quad 410 \quad 411 \quad 412 \quad 413 \quad 414 \quad 415 \quad 416 \quad 417 \quad 418 \quad 419 \quad 420 \quad 421 \quad 422 \quad 423 \quad 424 \quad 425 \quad 426 \quad 427 \quad 428 \quad 429 \quad 430 \quad 431 \quad 432 \quad 433 \quad 434 \quad 435 \quad 436 \quad 437 \quad 438 \quad 439 \quad 440 \quad 441 \quad 442 \quad 443 \quad 444 \quad 445 \quad 446 \quad 447 \quad 448 \quad 449 \quad 450 \quad 451 \quad 452 \quad 453 \quad 454 \quad 455 \quad 456 \quad 457 \quad 458 \quad 459 \quad 460 \quad 461 \quad 462 \quad 463 \quad 464 \quad 465 \quad 466 \quad 467 \quad 468 \quad 469 \quad 470 \quad 471 \quad 472 \quad 473 \quad 474 \quad 475 \quad 476 \quad 477 \quad 478 \quad 479 \quad 480 \quad 481 \quad 482 \quad 483 \quad 484 \quad 485 \quad 486 \quad 487 \quad 488 \quad 489 \quad 490 \quad 491 \quad 492 \quad 493 \quad 494 \quad 495 \quad 496 \quad 497 \quad 498 \quad 499 \quad 500 \quad 501 \quad 502 \quad 503 \quad 504 \quad 505 \quad 506 \quad 507 \quad 508 \quad 509 \quad 510 \quad 511 \quad 512 \quad 513 \quad 514 \quad 515 \quad 516 \quad 517 \quad 518 \quad 519 \quad 520 \quad 521 \quad 522 \quad 523 \quad 524 \quad 525 \quad 526 \quad 527 \quad 528 \quad 529 \quad 530 \quad 531 \quad 532 \quad 533 \quad 534 \quad 535 \quad 536 \quad 537 \quad 538 \quad 539 \quad 540 \quad 541 \quad 542 \quad 543 \quad 544 \quad 545 \quad 546 \quad 547 \quad 548 \quad 549 \quad 550 \quad 551 \quad 552 \quad 553 \quad 554 \quad 555 \quad 556 \quad 557 \quad 558 \quad 559 \quad 560 \quad 561 \quad 562 \quad 563 \quad 564 \quad 565 \quad 566 \quad 567 \quad 568 \quad 569 \quad 570 \quad 571 \quad 572 \quad 573 \quad 574 \quad 575 \quad 576 \quad 577 \quad 578 \quad 579 \quad 580 \quad 581 \quad 582 \quad 583 \quad 584 \quad 585 \quad 586 \quad 587 \quad 588 \quad 589 \quad 590 \quad 591 \quad 592 \quad 593 \quad 594 \quad 595 \quad 596 \quad 597 \quad 598 \quad 599 \quad 600 \quad 601 \quad 602 \quad 603 \quad 604 \quad 605 \quad 606 \quad 607 \quad 608 \quad 609 \quad 610 \quad 611 \quad 612 \quad 613 \quad 614 \quad 615 \quad 616 \quad 617 \quad 618 \quad 619 \quad 620 \quad 621 \quad 622 \quad 623 \quad 624 \quad 625 \quad 626 \quad 627 \quad 628 \quad 629 \quad 630 \quad 631 \quad 632 \quad 633 \quad 634 \quad 635 \quad 636 \quad 637 \quad 638 \quad 639 \quad 640 \quad 641 \quad 642 \quad 643 \quad 644 \quad 645 \quad 646 \quad 647 \quad 648 \quad 649 \quad 650 \quad 651 \quad 652 \quad 653 \quad 654 \quad 655 \quad 656 \quad 657 \quad 658 \quad 659 \quad 660 \quad 661 \quad 662 \quad 663 \quad 664 \quad 665 \quad 666 \quad 667 \quad 668 \quad 669 \quad 670 \quad 671 \quad 672 \quad 673 \quad 674 \quad 675 \quad 676 \quad 677 \quad 678 \quad 679 \quad 680 \quad 681 \quad 682 \quad 683 \quad 684 \quad 685 \quad 686 \quad 687 \quad 688 \quad 689 \quad 690 \quad 691 \quad 692 \quad 693 \quad 694 \quad 695 \quad 696 \quad 697 \quad 698 \quad 699 \quad 700 \quad 701 \quad 702 \quad 703 \quad 704 \quad 705 \quad 706 \quad 707 \quad 708 \quad 709 \quad 710 \quad 711 \quad 712 \quad 713 \quad 714 \quad 715 \quad 716 \quad 717 \quad 718 \quad 719 \quad 720 \quad 721 \quad 722 \quad 723 \quad 724 \quad 725 \quad 726 \quad 727 \quad 728 \quad 729 \quad 730 \quad 731 \quad 732 \quad 733 \quad 734 \quad 735 \quad 736 \quad 737 \quad 738 \quad 739 \quad 740 \quad 741 \quad 742 \quad 743 \quad 744 \quad 745 \quad 746 \quad 747 \quad 748 \quad 749 \quad 750 \quad 751 \quad 752 \quad 753 \quad 754 \quad 755 \quad 756 \quad 757 \quad 758 \quad 759 \quad 760 \quad 761 \quad 762 \quad 763 \quad 764 \quad 765 \quad 766 \quad 767 \quad 768 \quad 769 \quad 770 \quad 771 \quad 772 \quad 773 \quad 774 \quad 775 \quad 776 \quad 777 \quad 778 \quad 779 \quad 780 \quad 781 \quad 782 \quad 783 \quad 784 \quad 785 \quad 786 \quad 787 \quad 788 \quad 789 \quad 790 \quad 791 \quad 792 \quad 793 \quad 794 \quad 795 \quad 796 \quad 797 \quad 798 \quad 799 \quad 800 \quad 801 \quad 802 \quad 803 \quad 804 \quad 805 \quad 806 \quad 807 \quad 808 \quad 809 \quad 810 \quad 811 \quad 812 \quad 813 \quad 814 \quad 815 \quad 816 \quad 817 \quad 818 \quad 819 \quad 820 \quad 821 \quad 822 \quad 823 \quad 824 \quad 825 \quad 826 \quad 827 \quad 828 \quad 829 \quad 830 \quad 831 \quad 832 \quad 833 \quad 834 \quad 835 \quad 836 \quad 837 \quad 838 \quad 839 \quad 840 \quad 841 \quad 842 \quad 843 \quad 844 \quad 845 \quad 846 \quad 847 \quad 848 \quad 849 \quad 850 \quad 851 \quad 852 \quad 853 \quad 854 \quad 855 \quad 856 \quad 857 \quad 858 \quad 859 \quad 860 \quad 861 \quad 862 \quad 863 \quad 864 \quad 865 \quad 866 \quad 867 \quad 868 \quad 869 \quad 870 \quad 871 \quad 872 \quad 873 \quad 874 \quad 875 \quad 876 \quad 877 \quad 878 \quad 879 \quad 880 \quad 881 \quad 882 \quad 883 \quad 884 \quad 885 \quad 886 \quad 887 \quad 888 \quad 889 \quad 890 \quad 891 \quad 892 \quad 893 \quad 894 \quad 895 \quad 896 \quad 897 \quad 898 \quad 899 \quad 900 \quad 901 \quad 902 \quad 903 \quad 904 \quad 905 \quad 906 \quad 907 \quad 908 \quad 909 \quad 910 \quad 911 \quad 912 \quad 913 \quad 914 \quad 915 \quad 916 \quad 917 \quad 918 \quad 919 \quad 920 \quad 921 \quad 922 \quad 923 \quad 924 \quad 925 \quad 926 \quad 927 \quad 928 \quad 929 \quad 930 \quad 931 \quad 932 \quad 933 \quad 934 \quad 935 \quad 936 \quad 937 \quad 938 \quad 939 \quad 940 \quad 941 \quad 942 \quad 943 \quad 944 \quad 945 \quad 946 \quad 947 \quad 948 \quad 949 \quad 950 \quad 951 \quad 952 \quad 953 \quad 954 \quad 955 \quad 956 \quad 957 \quad 958 \quad 959 \quad 960 \quad 961 \quad 962 \quad 963 \quad 964 \quad 965 \quad 966 \quad 967 \quad 968 \quad 969 \quad 970 \quad 971 \quad 972 \quad 973 \quad 974 \quad 975 \quad 976 \quad 977 \quad 978 \quad 979 \quad 980 \quad 981 \quad 982 \quad 983 \quad 984 \quad 985 \quad 986 \quad 987 \quad 988 \quad 989 \quad 990 \quad 991 \quad 992 \quad 993 \quad 994 \quad 995 \quad 996 \quad 997 \quad 998 \quad 999 \quad 1000 \quad 1001 \quad 1002 \quad 1003 \quad 1004 \quad 1005 \quad 1006 \quad 1007 \quad 1008 \quad 1009 \quad 1010 \quad 1011 \quad 1012 \quad 1013 \quad 1014 \quad 1015 \quad 1016 \quad 1017 \quad 1018 \quad 1019 \quad 1020 \quad 1021 \quad 1022 \quad 1023 \quad 1024 \quad 1025 \quad 1026 \quad 1027 \quad 1028 \quad 1029 \quad 1030 \quad 1031 \quad 1032 \quad 1033 \quad 1034 \quad 1035 \quad 1036 \quad 1037 \quad 1038 \quad 1039 \quad 1040 \quad 1041 \quad 1042 \quad 1043 \quad 1044 \quad 1045 \quad 1046 \quad 1047 \quad 1048 \quad 1049 \quad 1050 \quad 1051 \quad 1052 \quad 1053 \quad 1054 \quad 1055 \quad 1056 \quad 1057 \quad 1058 \quad 1059 \quad 1060 \quad 1061 \quad 1062 \quad 1063 \quad 1064 \quad 1065 \quad 1066 \quad 1067 \quad 1068 \quad 1069 \quad 1070 \quad 1071 \quad 1072 \quad 1073 \quad 1074 \quad 1075 \quad 1076 \quad 1077 \quad 1078 \quad 1079 \quad 1080 \quad 1081 \quad 1082 \quad 1083 \quad 1084 \quad 1085 \quad 1086 \quad 1087 \quad 1088 \quad 1089 \quad 1090 \quad 1091 \quad 1092 \quad 1093 \quad 1094 \quad 1095 \quad 1096 \quad 1097 \quad 1098 \quad 1099 \quad 1100 \quad 1101 \quad 1102 \quad 1103 \quad 1104 \quad 1105 \quad 1106 \quad 1107 \quad 1108 \quad 1109 \quad 1110 \quad 1111 \quad 1112 \quad 1113 \quad 1114 \quad 1115 \quad 1116 \quad 1117 \quad 1118 \quad 1119 \quad 1120 \quad 1121 \quad 1122 \quad 1123 \quad 1124 \quad 1125 \quad 1126 \quad 1127 \quad 1128 \quad 1129 \quad 1130 \quad 1131 \quad 1132 \quad 1133 \quad 1134 \quad 1135 \quad 1136 \quad 1137 \quad 1138 \quad 1139 \quad 1140 \quad 1141 \quad 1142 \quad 1143 \quad 1144 \quad 1145 \quad 1146 \quad 1147 \quad 1148 \quad 1149 \quad 1150 \quad 1151 \quad 1152 \quad 1153 \quad 1154 \quad 1155 \quad 1156 \quad 1157 \quad 1158 \quad 1159 \quad 1160 \quad 1161 \quad 1162 \quad 1163 \quad 1164 \quad 1165 \quad 1166 \quad 1167 \quad 1168 \quad 1169 \quad 1170 \quad 1171 \quad 1172 \quad 1173 \quad 1174 \quad 1175 \quad 1176 \quad 1177 \quad 1178 \quad 1179 \quad 1180 \quad 1181 \quad 1182 \quad 1183 \quad 1184 \quad 1185 \quad 1186 \quad 1187 \quad 1188 \quad 1189 \quad 1190 \quad 1191 \quad 1192 \quad 1193 \quad 1194 \quad 1195 \quad 1196 \quad 1197 \quad 1198 \quad 1199 \quad 1200 \quad 1201 \quad 1202 \quad 1203 \quad 1204 \quad 1205 \quad 1206 \quad 1207 \quad 1208 \quad 1209 \quad 1210 \quad 1211 \quad 1212 \quad 1213 \quad 1214 \quad 1215 \quad 1216 \quad 1217 \quad 1218 \quad 1219 \quad 1220 \quad 1221 \quad 1222 \quad 1223 \quad 1224 \quad 1225 \quad 1226 \quad 1227 \quad 1228 \quad 1229 \quad 1230 \quad 1231 \quad 1232 \quad 1233 \quad 1234 \quad 1235 \quad 1236 \quad 1237 \quad 1238 \quad 1239 \quad 1240 \quad 1241 \quad 1242 \quad 1243 \quad 1244 \quad 1245 \quad 1246 \quad 1247 \quad 1248 \quad 1249 \quad 1250 \quad 1251 \quad 1252 \quad 1253 \quad 1254 \quad 1255 \quad 1256 \quad 1257 \quad 1258 \quad 1259 \quad 1260 \quad 1261 \quad 1262 \quad 1263 \quad 1264 \quad 1265 \quad 1266 \quad 1267 \quad 1268 \quad 1269 \quad 1270 \quad 1271 \quad 1272 \quad 1273 \quad 1274 \quad 1275 \quad 1276 \quad 1277 \quad 1278 \quad 1279 \quad 1280 \quad 1281 \quad 1282 \quad 1283 \quad 1284 \quad 1285 \quad 1286 \quad 1287 \quad 1288 \quad 1289 \quad 1290 \quad 1291 \quad 1292 \quad 1293 \quad 1294 \quad 1295 \quad 1296 \quad 1297 \quad 1298 \quad 1299 \quad 1300 \quad 1301 \quad 1302 \quad 1303 \quad 1304 \quad 1305 \quad 1306 \quad 1307 \quad 1308 \quad 1309 \quad 1310 \quad 1311 \quad 1312 \quad 1313 \quad 1314 \quad 1315 \quad 1316 \quad 1317 \quad 1318 \quad 1319 \quad 1320 \quad 1321 \quad 1322 \quad 1323 \quad 1324 \quad 1325 \quad 1326 \quad 1327 \quad 1328 \quad 1329 \quad 1330 \quad 1331 \quad 1332 \quad 1333 \quad 1334 \quad 1335 \quad 1336 \quad 1337 \quad 1338 \quad 1339 \quad 1340 \quad 1341 \quad 1342 \quad 1343 \quad 1344 \quad 1345 \quad 1346 \quad 1347 \quad 1348 \quad 1349 \quad 1350 \quad 1351 \quad 1352 \quad 1353 \quad 1354 \quad 1355 \quad 1356 \quad 1357 \quad 1358 \quad 1359 \quad 1360 \quad 1361 \quad 1362 \quad 1363 \quad 1364 \quad 1365 \quad 1366 \quad 1367 \quad 1368 \quad 1369 \quad 1370 \quad 1371 \quad 1372 \quad 1373 \quad 1374 \quad 1375 \quad 1376 \quad 1377 \quad 1378 \quad 1379 \quad 1380 \quad 1381 \quad 1382 \quad 1383 \quad 1384 \quad 1385 \quad 1386 \quad 1387 \quad 1388 \quad 1389 \quad 1390 \quad 1391 \quad 1392 \quad 1393 \quad 1394 \quad 1395 \quad 1396 \quad 1397 \quad 1398 \quad 1399 \quad 1400 \quad 1401 \quad 1402 \quad 1403 \quad 1404 \quad 1405 \quad 1406 \quad 1407 \quad 1408 \quad 1409 \quad 1410 \quad 1411 \quad 1412 \quad 1413 \quad 1414 \quad 1415 \quad 1416 \quad 1417 \quad 1418 \quad 1419 \quad 1420 \quad 1421 \quad 1422 \quad 1423 \quad 1424 \quad 1425 \quad 1426 \quad 1427 \quad 1428 \quad 1429 \quad 1430 \quad 1431 \quad 1432 \quad 1433 \quad 1434 \quad 1435 \quad 1436 \quad 1437 \quad 1438 \quad 1439 \quad 1440 \quad 1441 \quad 1442 \quad 1443 \quad 1444 \quad 1445 \quad 1446 \quad 1447 \quad 1448 \quad 1449 \quad 1450 \quad 1451 \quad 1452 \quad 1453 \quad 1454 \quad 1455 \quad 1456 \quad 1457 \quad 1458 \quad 1459 \quad 1460 \quad 1461 \quad 1462 \quad 1463 \quad 1464 \quad 1465 \quad 1466 \quad 1467 \quad 1468 \quad 1469 \quad 1470 \quad 1471 \quad 1472 \quad 1473 \quad 1474 \quad 1475 \quad 1476 \quad 1477 \quad 1478 \quad 1479 \quad 1480 \quad 1481 \quad 1482 \quad 1483 \quad 1484 \quad 1485 \quad 1486 \quad 1487 \quad 1488 \quad 1489 \quad 1490 \quad 1491 \quad 1492 \quad 1493 \quad 1494 \quad 1495 \quad 1496 \quad 1497 \quad 1498 \quad 1499 \quad 1500 \quad 1501 \quad 1502 \quad 1503 \quad 1504 \quad 1505 \quad 1506 \quad 1507 \quad 1508 \quad 1509 \quad 1510 \quad 1511 \quad 1512 \quad 1513 \quad 1514 \quad 1515 \quad 1516 \quad 1517 \quad 1518 \quad 1519 \quad 1520 \quad 1521 \quad 1522 \quad 1523 \quad 1524 \quad 1525 \quad 1526 \quad 1527 \quad 1528 \quad 1529 \quad 1530 \quad 1531 \quad 1532 \quad 1533 \quad 1534 \quad 1535 \quad 1536 \quad 1537 \quad 1538 \quad 1539 \quad 1540 \quad 1541 \quad 1542 \quad 1543 \quad 1544 \quad 1545 \quad 1546 \quad 1547 \quad 1548 \quad 1549 \quad 1550 \quad 1551 \quad 1552 \quad 1553 \quad 1554 \quad 1555 \quad 1556 \quad 1557 \quad 1558 \quad 1559 \quad 1560 \quad 1561 \quad 1562 \quad 1563 \quad 1564 \quad 1565 \quad 1566 \quad 1567 \quad 1568 \quad 1569 \quad 1570 \quad 1571 \quad 1572 \quad 1573 \quad 1574 \quad 1575 \quad 1576 \quad 1577 \quad 1578 \quad 1579 \quad 1580 \quad 1581 \quad 1582 \quad 1583 \quad 1584 \quad 1585 \quad 1586 \quad 1587 \quad 1588 \quad 1589 \quad 1590 \quad 1591 \quad 1592 \quad 1593 \quad 1594 \quad 1595 \quad 1596 \quad 1597 \quad 1598 \quad 1599 \quad 1600 \quad 1601 \quad 1602 \quad 1603 \quad 1604 \quad 1605 \quad 1606 \quad 1607 \quad 1608 \quad 1609 \quad 1610 \quad 1611 \quad 1612 \quad 1613 \quad 1614 \quad 1615 \quad 1616 \quad 1617 \quad 1618 \quad 1619 \quad 1620 \quad 1621 \quad 1622 \quad 1623 \quad 1624 \quad 1625 \quad 1626 \quad 1627 \quad 1628 \quad 1629 \quad 1630 \quad 1631 \quad 1632 \quad 1633 \quad 1634 \quad 1635 \quad 1636 \quad 1637 \quad 1638 \quad 1639 \quad 1640 \quad 1641 \quad 1642 \quad 1643 \quad 1644 \quad 1645 \quad 1646 \quad 1647 \quad 1648 \quad 1649 \quad 1650 \quad 1651 \quad 1652 \quad 1653 \quad 1654 \quad 1655 \quad 1656 \quad 1657 \quad 1658 \quad 1659 \quad 1660 \quad 1661 \quad 1662 \quad 1663 \quad 1664 \quad 1665 \quad 1666 \quad 1667 \quad 1668 \quad 1669 \quad 1670 \quad 1671 \quad 1672 \quad 1673 \quad 1674 \quad 1675 \quad 1676 \quad 1677 \quad 1678 \quad 1679 \quad 1680 \quad 1681 \quad 1682 \quad 1683 \quad 1684 \quad 1685 \quad 1686 \quad 1687 \quad 1688 \quad 1689 \quad 1690 \quad 1691 \quad 1692 \quad 1693 \quad 1694 \quad 1695 \quad 1696 \quad 1697 \quad 1698 \quad 1699 \quad 1700 \quad 1701 \quad 1702 \quad 1703 \quad 1704 \quad 1705 \quad 1706 \quad 1707 \quad 1708 \quad 1709 \quad 1710 \quad 1711 \quad 1712 \quad 1713 \quad 1714 \quad 1715 \quad 1716 \quad 1717 \quad 1718 \quad 1719 \quad 1720 \quad 1721 \quad 1722 \quad 1723 \quad 1724 \quad 1725 \quad 1726 \quad 1727 \quad 1728 \quad 1729 \quad 1730 \quad 1731 \quad 1732 \quad 1733 \quad 1734 \quad 1735 \quad 1736 \quad 1737 \quad 1738 \quad 1739 \quad 1740 \quad 1741 \quad 1742 \quad 1743 \quad 1744 \quad 1745 \quad 1746 \quad 1747 \quad 17$$



Figure 3: Qualitative comparison of unconditional sampling with different reversible 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.

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-trivial region of stability. We discuss this more in detail [Appendix A.2](#) 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 some stability.⁶

4.2 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 4.3](#) with the full details and proofs in [Appendix E](#).

Theorem 4.3 (Rex subsumes previous solvers). *The underlying scheme used Ψ in Rex given by*

$$\begin{aligned} \hat{\mathbf{Z}}_i &= \frac{1}{w_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[a_{ij} \mathbf{f}^\theta \left(\varsigma_n + c_j h, w_{\varsigma_n + c_j h} \hat{\mathbf{Z}}_j \right) \right] + a_i^W \mathbf{W}_n(\omega) + a_i^H \mathbf{H}_n(\omega), \\ \mathbf{X}_{n+1} &= \frac{w_{n+1}}{w_n} \mathbf{X}_n + w_{n+1} \left(h \sum_{j=1}^s \left[b_i \mathbf{f}^\theta \left(\varsigma_n + c_i h, w_{\varsigma_n + c_i h} \hat{\mathbf{Z}}_j \right) \right] + b^W \mathbf{W}_n(\omega) + b^H \mathbf{H}_n(\omega) \right), \end{aligned} \quad (16)$$

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 4.3.1 (Rex is reversible version of previous solvers). *Rex is the reversible revision of the well-known solvers for diffusion models in [Theorem 4.3](#).*

5 EMPIRICAL RESULTS

5.1 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 traditional solver for diffusion models. To evaluate this we drew 10,240 samples using a DDPM model ([Ho et al., 2020](#)) pretrained on the

⁶*I.e.*, in the sense of the linear test equation, see [Appendix A.2](#) for more details.

378
 379 Table 1: Quantitative comparison of different reversible solvers for unconditional image generation
 380 with a pre-trained DDPM model on CelebA-HQ (256×256) with the non-reversible DDIM as a
 381 baseline. \dagger denotes $\gamma = 0.5$ and \ddagger denotes $\gamma = 1.0$ for BDIA hyperparameter.

382	Steps	Solver	FD (↓)	FD _∞ (↓)	Precision (↑)	Recall (↑)	Density (↑)	Coverage (↑)
383	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 (RK4)	633.90	617.11	0.81	0.22	0.64	0.36
		Rex (Midpoint)	607.20	597.04	0.78	0.21	0.60	0.37
		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 (RK4)	547.24	533.30	0.82	0.27	0.71	0.43
		Rex (Midpoint)	539.96	527.85	0.81	0.26	0.66	0.41
		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 (RK4)	511.17	498.94	0.80	0.27	0.69	0.44
		Rex (Midpoint)	505.67	494.94	0.81	0.29	0.70	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

402
 403 CelebA-HQ (Karras et al., 2018) dataset with the various solvers each using the same fixed seed.
 404 Following Stein et al. (2023), we report the *Fréchet distance* (FD) with DINOv2 (Oquab et al., 2023)
 405 feature extractor along with FD_∞ (Chong & Forsyth, 2020). We also report the precision and recall
 406 metrics (Kynkäanniemi et al., 2019); along with density and coverage metrics (Naeem et al., 2020)
 407 which serve as a proxy for fidelity and sample diversity respectively. We provide more details on
 408 these metrics in Section I.1.2. In Table 1 we compare pre-existing methods for exact inversion with
 409 diffusion models against Rex, along with including the non-reversible DDIM solver as a baseline. We
 410 observe that the Rex family of reversible solvers performs exceedingly well, surpassing the baseline
 411 non-reversible DDIM scheme, handily beating EDICT and BDIA, and often outperforming O-BELM.
 412 We observe that our reversible SDE scheme consistently performs quite well outside of the very few
 413 step-size regime (a well known limitation of SDE schemes). *N.B.*, that unlike the results reported
 414 for the other reversible solvers we did not search for the optimal hyperparameters for Rex for the
 415 sampling task. In Figure 3 we present a visual qualitative comparison of the different solvers using
 416 the same initial noise. We provide additional experimental details in Appendix I.1.

417 Table 2: Quantitative comparison of different reversible solvers in terms of average CLIP score,
 418 Image Reward, and PickScore. for conditional text-to-image generation with Stable Diffusion v1.5
 419 (512×512) with the non-reversible DDIM as a baseline.

421	Solver / Steps	CLIP score (↑)			Image Reward (↑)			PickScore (↑)		
		10	20	50	10	20	50	10	20	50
422	EDICT	27.97	31.04	31.17	-1.219	-0.134	-0.055	19.52	20.84	21.05
423	BDIA $\gamma = 0.96$	31.11	31.52	31.54	-0.111	0.067	0.087	20.52	21.01	21.19
424	BDIA $\gamma = 0.5$	31.57	31.48	31.48	-0.006	0.055	0.066	20.98	21.16	21.21
425	O-BELM	31.47	31.43	31.51	0.051	0.105	0.160	20.88	21.00	21.16
426	Rex (Midpoint)	31.62	31.64	31.60	0.119	0.179	0.198	21.28	21.38	21.41
427	Rex (RK4)	31.69	31.60	31.57	0.156	0.187	0.195	21.35	21.40	21.41
428	Rex (Euler-Maruyama)	31.68	31.56	31.33	0.222	0.239	0.264	21.50	21.66	21.70
429	Rex (ShARK)	31.55	31.56	31.39	0.239	0.249	0.263	21.51	21.66	21.72
430	DDIM	31.78	31.76	31.24	0.033	0.136	0.247	21.06	21.29	21.04

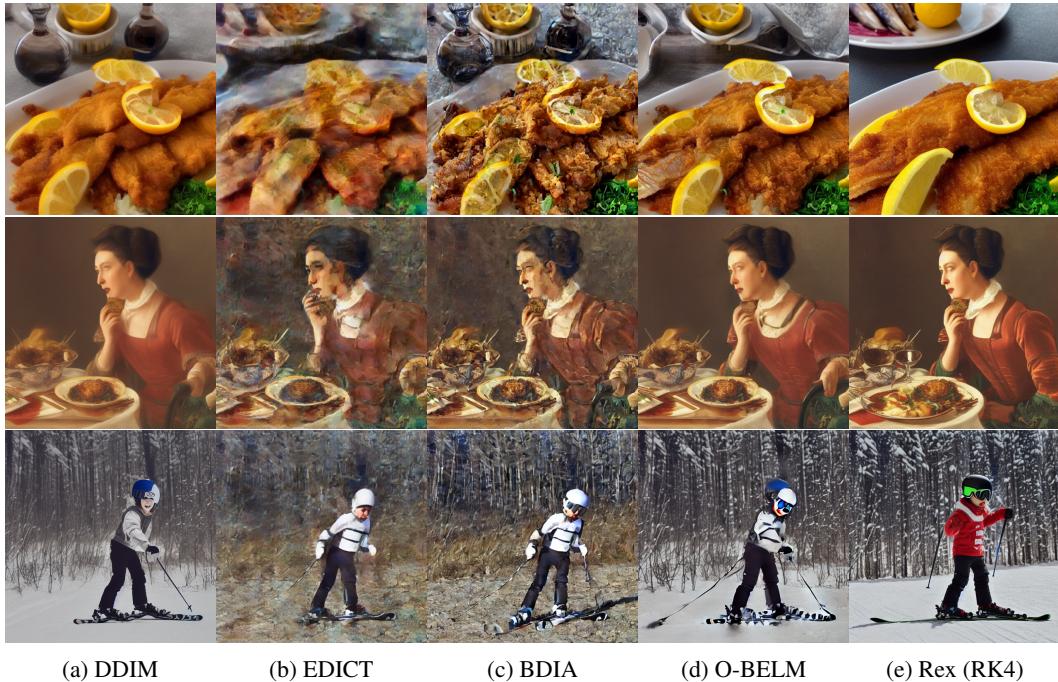


Figure 4: Qualitative comparison of text-to-image conditional sampling with different reversible solvers with Stable Diffusion v1.5 (512×512) and 10 discretization steps. Prompts from top to bottom are: “White plate with fried fish and lemons sitting on top of it.”, “A lady enjoying a meal of some sort.”, and “A young boy riding skis with ski poles.”.

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 CLIP Score (Hessel et al., 2021); 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. The later metric was recently become a popular metric for evaluating the performance of diffusion models (Skreta et al., 2025a). 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 Figure 4 we present a visual qualitative comparison of the different solvers using the same initial noise. We provide additional experimental details in Appendix I.2.

5.2 IMAGE 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 5 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, we discuss this more in Appendix K.

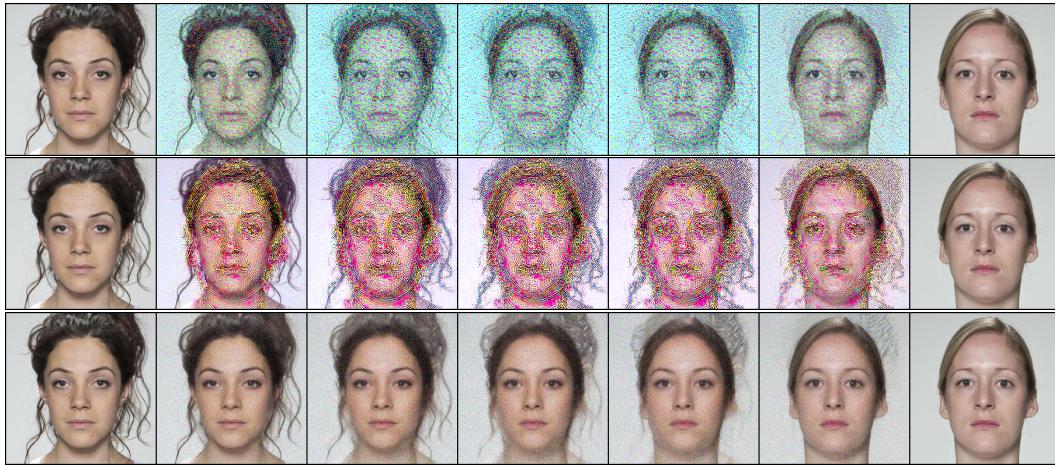


Figure 5: Unconditional interpolation between two real images from FRLL (DeBruine & Jones, 2017) with a DDPM model trained on CelebA-HQ. Top row is BELM, middle is Rex (Euler), and bottom is Rex (ShARK). 50 steps used for each method.

6 CONCLUSION

We propose *Rex* a family of algebraically reversible solvers for diffusion models which can obtain arbitrarily a high order of convergence (for the ODE case). Moreover, we propose (to the best of our knowledge) the first method for exact inversion for diffusion SDEs without storing the entire trajectory of the Brownian motion. Our empirical illustrations show that not only does *Rex* have nice theoretical properties but it also functions as a capable numerical scheme for sampling with diffusion models. The proposed method can be incorporated into preexisting applications wherein preserving the bijections of flow maps is important, leading to many exciting possible applications.

ETHICS STATEMENT

We recognize that *Rex* as numerical scheme for sampling with diffusion models could potentially be misused used for malicious applications particularly when used in editing pipelines.

REPRODUCIBILITY STATEMENT

To aid with reproducibility we include detailed derivations of *Rex* in Appendix C along with additional proofs in Appendix D. We draw connections between *Rex* and other solver for diffusion models in Appendix E. We include through implementation details in Appendix H and experimental details in Appendix I; in particular, we mention all code repositories and datasets we used in Appendix I.5. Moreover, we provide code illustrations of the core components of *Rex* in Appendix J.

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.
- Zander W. Blasingame and Chen Liu. Adjointdeis: Efficient gradients for diffusion models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomeczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 2449–2483. Curran Associates, Inc., 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/04badd3b048315c8c3a0ca17eff723d7-Paper-Conference.pdf.

- 540 Zander W. Blasingame and Chen Liu. Fast-dim: Towards fast diffusion morphs. *IEEE Security &*
 541 *Privacy*, 2024b.
- 542
- 543 Zander W. Blasingame and Chen Liu. Greedy-dim: Greedy algorithms for unreasonably effective
 544 face morphs. In *2024 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–11, 2024c.
 545 doi: 10.1109/IJCB62174.2024.10744517.
- 546 Zander W. Blasingame and Chen Liu. Leveraging diffusion for strong and high quality face morphing
 547 attacks. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 6(1):118–131, 2024d.
- 548
- 549 Zander W. Blasingame and Chen Liu. Greed is good: A unifying perspective on guided generation.
 550 In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL
 551 <https://openreview.net/forum?id=s14pdQgoLb>.
- 552 Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and
 553 Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models.
 554 In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.
 555 22563–22575, 2023.
- 556
- 557 Paul Bourgade. Stochastic analysis, 2010. URL https://cims.nyu.edu/~bourgade/SA2010/StochasticAnalysis.pdf?utm_source=chatgpt.com.
- 558
- 559 Kevin Burrage and Pamela M Burrage. Order conditions of stochastic runge–kutta methods by
 560 b-series. *SIAM Journal on Numerical Analysis*, 38(5):1626–1646, 2000.
- 561
- 562 John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons,
 563 2016. Third Edition.
- 564
- 565 Girolamo Cardano. *Artis Magnæ, Sive de Regulis Algebraicis, Lib. unus.* 1545.
- 566
- 567 Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find
 568 them. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp.
 6070–6079, 2020.
- 569
- 570 Koen Claessen and Michał H Pałka. Splittable pseudorandom number generators using cryptographic
 571 hashing. *ACM SIGPLAN Notices*, 48(12):47–58, 2013.
- 572
- 573 John MC Clark and RJ Cameron. The maximum rate of convergence of discrete approximations
 574 for stochastic differential equations. In *Stochastic Differential Systems Filtering and Control: Procedings of the IFIP-WG 7/1 Working Conference Vilnius, Lithuania, USSR, Aug. 28–Sept. 2, 1978*, pp. 162–171. Springer, 2005.
- 575
- 576 M Crouzeix and FJ Lisbona. The convergence of variable-stepsize, variable-formula, multistep
 577 methods. *SIAM journal on numerical analysis*, 21(3):512–534, 1984.
- 578
- 579 Kristian Debrabant, Anne Kværnø, and Nicky Cordua Mattsson. Runge–kutta lawson schemes for
 580 stochastic differential equations. *BIT Numerical Mathematics*, 61(2):381–409, 2021.
- 581
- 582 Lisa DeBruine and Benedict Jones. Face Research Lab London Set, 5 2017. URL
 583 https://figshare.com/articles/dataset/Face_Research_Lab_London_Set/5047666.
- 584
- 585 Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components
 586 estimation, 2015. URL <https://arxiv.org/abs/1410.8516>.
- 587
- 588 DC Dowson and BV666017 Landau. The fréchet distance between multivariate normal distributions.
 589 *Journal of multivariate analysis*, 12(3):450–455, 1982.
- 590
- 591 Lester E Dubins and Gideon Schwarz. On continuous martingales. *Proceedings of the National
 592 Academy of Sciences*, 53(5):913–916, 1965.
- 593
- Kang Feng. On difference schemes and symplectic geometry. In *Proceedings of the 5th international
 594 symposium on differential geometry and differential equations*, 1984.

- 594 James Foster, Terry Lyons, and Harald Oberhauser. An optimal polynomial approximation of
 595 brownian motion. *SIAM Journal on Numerical Analysis*, 58(3):1393–1421, 2020.
 596
- 597 James M Foster. *Numerical approximations for stochastic differential equations*. Ph.d. thesis,
 598 University of Oxford, 2020. URL <https://ora.ox.ac.uk/objects/uuid:775fc3f5-501c-425f-8b43-fc5a7b2e4310>.
 599
- 600 James M Foster, Goncalo Dos Reis, and Calum Strange. High order splitting methods for sdes
 601 satisfying a commutativity condition. *SIAM Journal on Numerical Analysis*, 62(1):500–532, 2024.
 602
- 603 Peter K Friz and Martin Hairer. *A course on rough paths*. Springer, 2020.
 604
- 605 Jessica G Gaines and Terry J Lyons. Variable step size control in the numerical solution of stochastic
 606 differential equations. *SIAM Journal on Applied Mathematics*, 57(5):1455–1484, 1997.
 607
- 608 Martin Gonzalez, Nelson Fernandez Pinto, Thuy Tran, Hatem Hajri, Nader Masmoudi, et al. Seeds:
 609 Exponential sde solvers for fast high-quality sampling from diffusion models. *Advances in Neural
 610 Information Processing Systems*, 36, 2024.
- 611 Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In
 612 H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (eds.),
 613 *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.,
 614 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf.
 615
- 616 Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-
 617 free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.
 618
- 619 Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans
 620 trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural
 621 information processing systems*, 30, 2017.
- 622 Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in
 623 neural information processing systems*, 33:6840–6851, 2020.
 624
- 625 Marlis Hochbruck, Jan Leibold, and Alexander Ostermann. On the convergence of lawson methods
 626 for semilinear stiff problems. *Numerische Mathematik*, 145(3):553–580, 2020.
 627
- 628 Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and
 629 Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In
 630 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.
 631 9307–9315, 2024.
- 632 Andraž Jelinčič, James Foster, and Patrick Kidger. Single-seed generation of brownian paths and
 633 integrals for adaptive and high order sde solvers. *arXiv preprint arXiv:2405.06464*, 2024.
 634
- 635 Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta
 636 go fast with score-based generative models. In *The Symbiosis of Deep Learning and Differential
 637 Equations*, 2021. URL <https://openreview.net/forum?id=gEoVDSASC2n>.
 638
- 639 Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for
 640 improved quality, stability, and variation. In *International Conference on Learning Representations*,
 641 2018. URL <https://openreview.net/forum?id=Hk99zCeAb>.
 642
- 643 Patrick Kidger. *On Neural Differential Equations*. Ph.d. thesis, Oxford University, 2022. Available at
<https://arxiv.org/abs/2202.02435>.
 644
- 645 Patrick Kidger, James Foster, Xuechen Chen Li, and Terry Lyons. Efficient and accurate gradients
 646 for neural sdes. *Advances in Neural Information Processing Systems*, 34:18747–18761, 2021.
 647
- Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances
 648 in neural information processing systems*, 34:21696–21707, 2021.

- 648 Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy.
 649 Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Thirty-seventh*
 650 *Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=G5RwHpBUv0>.
 651
- 652 Peter E Kloeden and Eckhard Platen. Stratonovich and itô stochastic taylor expansions. *Mathematische*
 653 *Nachrichten*, 151(1):33–50, 1991.
 654
- 655 Peter E. Kloeden and Eckhard Platen. *Stochastic Differential Equations*, pp. 103–160. Springer
 656 Berlin Heidelberg, Berlin, Heidelberg, 1992. ISBN 978-3-662-12616-5. doi: 10.1007/978-3-662-
 657 12616-5_4. URL https://doi.org/10.1007/978-3-662-12616-5_4.
 658
- 659 Kei Kobayashi. Stochastic calculus for a time-changed semimartingale and the associated stochastic
 660 differential equations. *Journal of Theoretical Probability*, 24(3):789–820, 2011.
 661
- 662 Yoshio Komori, David Cohen, and Kevin Burrage. Weak second order explicit exponential runge–
 663 kutta methods for stochastic differential equations. *SIAM Journal on Scientific Computing*, 39(6):
 664 A2857–A2878, 2017.
 665
- 666 Tuomas Kynkänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved
 667 precision and recall metric for assessing generative models. *Advances in neural information*
 668 *processing systems*, 32, 2019.
 669
- 670 Tuomas Kynkänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of im-
 671 agenet classes in fréchet inception distance. In *The Eleventh International Conference on Learning*
 672 *Representations*, 2023. URL https://openreview.net/forum?id=4oXTQ6m_ws8.
 673
- 674 J Douglas Lawson. Generalized runge-kutta processes for stable systems with large lipschitz constants.
 675 *SIAM Journal on Numerical Analysis*, 4(3):372–380, 1967.
 676
- 677 Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-
 678 training for unified vision-language understanding and generation. In *International conference on*
 679 *machine learning*, pp. 12888–12900. PMLR, 2022.
 680
- 681 Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients
 682 for stochastic differential equations. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings*
 683 *of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108
 684 *of Proceedings of Machine Learning Research*, pp. 3870–3882. PMLR, 26–28 Aug 2020. URL
 685 <https://proceedings.mlr.press/v108/li20i.html>.
 686
- 687 Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and
 688 sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of*
 689 *computer vision*, pp. 5404–5411, 2024.
 690
- 691 Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr
 692 Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European*
 693 *conference on computer vision*, pp. 740–755. Springer, 2014.
 694
- 695 Yaron Lipman, Marton Havasi, Peter Holderith, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen,
 696 David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint*
 697 *arXiv:2412.06264*, 2024.
 698
- 699 George Lowther. Time-changed brownian motion, 2010. URL <https://almostsuremath.com/2010/04/20/time-changed-brownian-motion/>.
 700
- 701 Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast
 702 solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*,
 703 2022a.
 704
- 705 Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A
 706 fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In Alice H. Oh,
 707 Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information*
 708 *Processing Systems*, 2022b. URL https://openreview.net/forum?id=2uAaGwlP_V.
 709

- 702 Terry J Lyons. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*,
 703 14(2):215–310, 1998.
- 704
- 705 Dimitra Maoutsas, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker-
 706 planck equations through gradient-log-density estimation. *Entropy*, 22(8):802, 2020.
- 707
- 708 Takashi Matsubara, Yuto Miyatake, and Takaharu Yaguchi. Symplectic adjoint method
 709 for exact gradient of neural ode with minimal memory. In M. Ranzato, A. Beygelz-
 710 imer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural In-
 711 formation Processing Systems*, volume 34, pp. 20772–20784. Curran Associates, Inc.,
 712 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/adf8d7f8c53c8688e63a02bfb3055497-Paper.pdf.
- 713
- 714 Sam McCallum and James Foster. Efficient, accurate and stable gradients for neural odes. *arXiv
 preprint arXiv:2410.11648*, 2024.
- 715
- 716 Jan Mrongowius and Andreas Rößler. On the approximation and simulation of iterated stochastic
 717 integrals and the corresponding lévy areas in terms of a multidimensional brownian motion.
 718 *Stochastic Analysis and Applications*, 40(3):397–425, 2022.
- 719
- 720 Ulrich Mutze. An asynchronous leapfrog method ii. *arXiv preprint arXiv:1311.6602*, 2013.
- 721
- 722 Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable
 723 fidelity and diversity metrics for generative models. In *International conference on machine
 learning*, pp. 7176–7185. PMLR, 2020.
- 724
- 725 Shen Nie, Hanzhong Allan Guo, Cheng Lu, Yuhao Zhou, Chenyu Zheng, and Chongxuan Li. The
 726 blessing of randomness: SDE beats ODE in general diffusion-based image editing. In *The Twelfth
 727 International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=DesYwmUG00>.
- 728
- 729 Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Universitext.
 730 Springer Berlin Heidelberg, Berlin, Germany, jul 2003. ISBN 9783662036204. doi: 10.1007/978-
 731 3-642-14394-6.
- 732
- 733 Maxime Oquab, Timothée Darcret, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov,
 734 Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning
 robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- 735
- 736 Jiachun Pan, Hanshu Yan, Jun Hao Liew, Jiashi Feng, and Vincent YF Tan. Towards accurate guided
 737 diffusion sampling through symplectic adjoint method. *arXiv preprint arXiv:2312.12030*, 2023.
- 738
- 739 Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal,
 740 Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever.
 741 Learning transferable visual models from natural language supervision. In Marina Meila and Tong
 742 Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume
 743 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL
 744 <https://proceedings.mlr.press/v139/radford21a.html>.
- 745
- 746 Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*, volume 293. Springer
 747 Science & Business Media, 2013.
- 748
- 749 Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-
 750 resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF confer-
 751 ence on computer vision and pattern recognition*, pp. 10684–10695, 2022.
- 752
- 753 Andreas Rößler. Runge–kutta methods for the strong approximation of solutions of stochastic
 754 differential equations. *SIAM Journal on Numerical Analysis*, 48(3):922–952, 2010.
- 755
- 756 Andreas Rößler. A class of stochastic runge-kutta methods for stochastic differential equations
 757 converging with order 1 in L^p -norm. *arXiv preprint arXiv:2506.22657*, 2025.
- 758
- 759 W Rüemelin. Numerical treatment of stochastic differential equations. *SIAM Journal on Numerical
 760 Analysis*, 19(3):604–613, 1982.

- 756 Ronald D Ruth. A canonical integration technique. *IEEE Trans. Nucl. Sci.*, 30(CERN-LEP-TH-83-
 757 14):2669–2671, 1983.
- 758
- 759 Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing
 760 generative models via precision and recall. *Advances in neural information processing systems*, 31,
 761 2018.
- 762 John K Salmon, Mark A Moraes, Ron O Dror, and David E Shaw. Parallel random numbers: as
 763 easy as 1, 2, 3. In *Proceedings of 2011 international conference for high performance computing,
 764 networking, storage and analysis*, pp. 1–12, 2011.
- 765
- 766 L. F. Shampine. Stability of the leapfrog/midpoint method. *Applied Mathematics and Computation*,
 767 208(1):293–298, 2009.
- 768
- 769 Marta Skreta, Tara Akhoud-Sadegh, Viktor Ohanesian, Roberto Bondesan, Alan Aspuru-Guzik,
 770 Arnaud Doucet, Rob Brekelmans, Alexander Tong, and Kirill Neklyudov. Feynman-kac correc-
 771 tors in diffusion: Annealing, guidance, and product of experts. In *Forty-second International
 772 Conference on Machine Learning*, 2025a. URL <https://openreview.net/forum?id=Vhc0KrcqWu>.
- 773
- 774 Marta Skreta, Lazar Atanackovic, Joey Bose, Alexander Tong, and Kirill Neklyudov. The superposi-
 775 tion of diffusion models using the \hat{t} density estimator. In *The Thirteenth International Confer-
 776 ence on Learning Representations*, 2025b. URL <https://openreview.net/forum?id=2o58Mbqkd2>.
- 777
- 778 Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised
 779 learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Confer-
 780 ence on International Conference on Machine Learning - Volume 37*, ICML’15, pp. 2256–2265.
 781 JMLR.org, 2015.
- 782
- 783 Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Inter-
 784 national Conference on Learning Representations*, 2021a. URL <https://openreview.net/forum?id=St1giarCHLP>.
- 785
- 786 Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben
 787 Poole. Score-based generative modeling through stochastic differential equations. In *International
 788 Conference on Learning Representations*, 2021b. URL <https://openreview.net/forum?id=PxTIG12RRHS>.
- 789
- 790 Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas
 791 Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan
 792 Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume
 793 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 23–29 Jul 2023.
 794 URL <https://proceedings.mlr.press/v202/song23a.html>.
- 795
- 796 George Stein, Jesse Cresswell, Rasa Hosseinzadeh, Yi Sui, Brendan Ross, Valentin Villecroze,
 797 Zhaoyan Liu, Anthony L Caterini, Eric Taylor, and Gabriel Loaiza-Ganem. Exposing flaws of
 798 generative model evaluation metrics and their unfair treatment of diffusion models. In *Advances in
 799 Neural Information Processing Systems*, volume 36, 2023.
- 800
- 801 David E Stewart. *Numerical analysis: A graduate course*, volume 258. Springer, 2022.
- 802
- 803 Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking
 804 the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer
 805 vision and pattern recognition*, pp. 2818–2826, 2016.
- 806
- 807 René J. De Vogelaere. Methods of integration which preserve the contact transformation property of
 808 the hamilton equations. Report NO. 4, University of Notre Dame, 1956.
- 809
- 810 Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transfor-
 811 mations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*,
 812 pp. 22532–22541, 2023.

- 810 Fangyikang Wang, Hubery Yin, Yue-Jiang Dong, Huminhao Zhu, Chao Zhang, Hanbin Zhao, Hui
 811 Qian, and Chen Li. BELM: Bidirectional explicit linear multi-step sampler for exact inversion
 812 in diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing*
 813 *Systems*, 2024. URL <https://openreview.net/forum?id=ccQ4fmwLDb>.
- 814 Chen Henry Wu and Fernando De la Torre. A latent space of stochastic diffusion models for zero-shot
 815 image editing and guidance. In *ICCV*, 2023.
- 817 Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao
 818 Dong. Imagereward: learning and evaluating human preferences for text-to-image generation. In
 819 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp.
 820 15903–15935, 2023.
- 821 Guoqiang Zhang, J. P. Lewis, and W. Bastiaan Kleijn. Exact diffusion inversion via bidirectional
 822 integration approximation. In *Computer Vision – ECCV 2024: 18th European Conference, Milan,
 823 Italy, September 29–October 4, 2024, Proceedings, Part LVII*, pp. 19–36, Berlin, Heidelberg,
 824 2024. Springer-Verlag. ISBN 978-3-031-72997-3. doi: 10.1007/978-3-031-72998-0_2. URL
 825 https://doi.org/10.1007/978-3-031-72998-0_2.
- 826 Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator.
 827 In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=Loek7hfb46P>.
- 830 Qinsheng Zhang, Molei Tao, and Yongxin Chen. gDDIM: Generalized denoising diffusion implicit
 831 models. In *The Eleventh International Conference on Learning Representations*, 2023. URL
 832 <https://openreview.net/forum?id=1hKE9qjvz->.
- 833 Juntang Zhuang, Nicha C Dvornek, sekhar tatikonda, and James s Duncan. MALI: A memory
 834 efficient and reverse accurate integrator for neural ODEs. In *International Conference on Learning*
 835 *Representations*, 2021. URL https://openreview.net/forum?id=blfSjHeFM_e.
- 837
 838
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 860
 861
 862
 863

864	APPENDICES		
865			
866	A	<i>Related works</i>	20
867		<i>A.1 Reversible solvers</i>	20
868		<i>Asynchronous leapfrog method • Reversible Heun method • McCallum-Foster method</i>	
869			
870		<i>A.2 A note on stability</i>	22
871		<i>A.3 Exact inversion of diffusion models</i>	23
872		<i>EDICT sampler • BDIA sampler • BELM sampler • CycleDiffusion • Summary</i>	
873			
874		<i>A.4 SDE solvers for diffusion models</i>	26
875		<i>Comparison with SEEDS</i>	
876			
877	B	<i>Stochastic Runge-Kutta methods</i>	27
878		<i>B.1 Foster-Reis-Strange SRK Scheme</i>	28
879		<i>B.2 Independence of the Brownian and Lévy increments</i>	29
880		<i>B.3 ShARK</i>	29
881			
882	C	<i>Derivation of Rex</i>	29
883		<i>C.1 Rex (ODE)</i>	29
884		<i>Proof of Proposition 3.1 • Data prediction • Noise prediction</i>	
885			
886		<i>C.2 Rex (SDE)</i>	34
887		<i>Time-changed Brownian motion • Proof of Proposition 3.2 • Proof of reparameterized SDE for noise prediction models • Derivation of Rex (SDE)</i>	
888			
889		<i>C.3 Proof of Proposition 3.3</i>	41
890			
891	D	<i>Convergence order proofs</i>	42
892		<i>D.1 Assumptions</i>	42
893		<i>D.2 Proof of Theorem 4.1</i>	42
894		<i>D.3 Proof of Theorem 4.2</i>	43
895			
896	E	<i>Relation to other solvers for diffusion models</i>	44
897		<i>E.1 Rex as reversible ODE solvers</i>	45
898		<i>Euler • Second-order methods • Third-order methods</i>	
899			
900		<i>E.2 Rex as reversible SDE solvers</i>	49
901		<i>Euler-Maruyama</i>	
902			
903		<i>E.3 Rex as reversible SEEDS-1</i>	51
904			
905	F	<i>A brief note on the theory of rough paths</i>	52
906			
907	G	<i>Numerical simulation of Brownian motion</i>	52
908			
909	H	<i>Implementation details</i>	53
910		<i>H.1 Closed form expressions of the noise schedule</i>	53
911		<i>Linear noise schedule • Scaled linear schedule</i>	
912			
913		<i>H.2 Some other inverse functions</i>	56
914		<i>H.3 Brownian motion</i>	57
915	I	<i>Experimental details</i>	57
916		<i>I.1 Unconditional image generation</i>	57
917		<i>Diffusion model • Metrics • Hyperparameters</i>	

918	<i>I.2 Conditional image generation</i>	59
919	<i>Diffusion model • Metrics • Hyperparameters</i>	
920		
921	<i>I.3 Interpolation</i>	60
922		
923	<i>I.4 Hardware</i>	60
924		
925	<i>I.5 Repositories</i>	60
926		
927	<i>J Code</i>	60
928		
929	<i>K Visualization of inversion and the latent space</i>	64
930		
931	<i>L Additional results</i>	65
932		
933	<i>L.1 Unconditional image generation</i>	65
934		
935	<i>L.2 Conditional image generation</i>	65
936		

OVERVIEW OF THEORETICAL RESULTS

937	<i>3.1 Proposition (Reparameterization of the probability flow ODE)</i>	4
938	<i>3.2 Proposition (Time reparameterization of the reverse-time diffusion SDE)</i>	5
939	<i>3.3 Proposition (Rex)</i>	5
940		
941	<i>4.1 Theorem (Rex is a k-th order solver)</i>	6
942	<i>4.2 Theorem (Convergence order for stochastic Ψ)</i>	6
943	<i>4.3 Theorem (Rex subsumes previous solvers)</i>	7
944		
945	<i>4.3.1 Corollary (Rex is reversible version of previous solvers)</i>	7
946		
947	<i>A.1 Theorem (Convergence order of the McCallum-Foster method)</i>	22
948	<i>A.2 Theorem (Region of stability for the McCallum-Foster method)</i>	23
949	<i>A.3 Proposition (BDIA is the leapfrog/midpoint method)</i>	24
950		
951	<i>A.3.1 Corollary (BDIA is a first-order method)</i>	24
952	<i>A.3.2 Corollary (BDIA is nowhere linearly stable)</i>	24
953		
954	<i>A.4 Theorem (O-BELM is the leapfrog/midpoint method)</i>	25
955	<i>A.4.1 Corollary (O-BELM is nowhere linearly stable)</i>	25
956		
957	<i>3.1 Proposition (Reparameterization of the probability flow ODE)</i>	29
958	<i>C.1 Lemma (Rex (ODE) for data prediction models)</i>	31
959	<i>C.2 Lemma (Rex (ODE) for noise prediction models)</i>	32
960		
961	<i>C.3 Theorem (Dambis-Dubins-Schwarz representation theorem)</i>	34
962	<i>C.4 Theorem (Multi-dimensional Dambis-Dubins-Schwarz representation theorem)</i>	34
963		
964	<i>3.2 Proposition (Time reparameterization of the reverse-time diffusion SDE)</i>	37
965	<i>C.7 Proposition (Time reparameterization of the reverse-time diffusion SDE for noise prediction models)</i>	38
966		
967	<i>C.8 Lemma (Rex (SDE) for data prediction models)</i>	39
968		
969	<i>C.9 Lemma (Rex (SDE) for noise prediction models)</i>	40
970		
971	<i>3.3 Proposition (Rex)</i>	42
	<i>4.1 Theorem (Rex is a k-th order solver)</i>	42

972	4.2	Theorem (Convergence order for stochastic Ψ)	43
973	4.3	Theorem (Rex subsumes previous solvers)	44
974	4.3.1	Corollary (Rex is reversible version of previous solvers)	45
975	E.1	Proposition (Rex (Euler) is reversible DPM-Solver++1)	45
976	E.1.1	Corollary (Rex (Euler) is reversible deterministic DDIM for data prediction models)	46
977	E.2	Proposition (Rex (Euler) is reversible DPM-Solver-1)	46
978	E.2.1	Corollary (Rex (Euler) is reversible deterministic DDIM for noise prediction models)	47
979	E.3	Proposition (Rex (generic second-order) is reversible DPM-Solver++(2S))	47
980	E.4	Proposition (Rex (generic second-order) is reversible DPM-Solver-2))	48
981	E.5	Proposition (Rex (Euler-Midpoint) is DPM-Solver-12)	48
982	E.6	Proposition (Rex (Euler-Maruyama) is reversible SDE-DPM-Solver++1)	49
983	E.6.1	Corollary (Rex (Euler-Maruyama) is reversible stochastic DDIM)	50
984	E.7	Proposition (Rex (Euler-Maruyama) is reversible SDE-DPM-Solver-1)	50
985	E.7.1	Corollary (Rex (Euler-Maruyama) is reversible stochastic DDIM for noise prediction models)	51
986	E.8	Proposition (Rex is reversible SEEDS-1)	51
987	E.8.1	Corollary (Rex (Euler-Maruyama) is reversible gDDIM)	51
988	H.1	Proposition (Inverse function of γ_t for linear noise schedule)	54
989	H.1.1	Corollary (Inverse function of χ_t for linear noise schedule)	54
990	H.1.2	Corollary (Inverse function of ϱ_t for linear noise schedule)	54
991	H.2	Proposition (Inverse function of γ_t for scaled linear noise schedule)	55
992	H.2.1	Corollary (Inverse function of χ_t for scaled linear noise schedule)	56
993	H.2.2	Corollary (Inverse function of ϱ_t for scaled linear noise schedule)	56
1000			
1001			
1002			
1003			
1004			
1005			
1006			
1007			
1008			
1009			
1010			
1011			
1012			
1013			
1014			
1015			
1016			
1017			
1018			
1019			
1020			
1021			
1022			
1023			
1024			
1025			

1026

A RELATED WORKS

1027

1028 In this section we provide a detailed comparison with relevant related works. We begin in Ap-
1029 pendix A.1 by providing an overview of algebraically reversible solvers. Then in Appendix A.2
1030 we introduce the stability of an ODE solver, a helpful tool in comparing reversible solvers. Using
1031 this tool along with examining the convergence order we compare a variety of reversible solvers for
1032 diffusion models in Appendix A.3. Lastly, in Appendix A.4 we explore related work on constructing
1033 SDE solvers for diffusion models.

1034

A.1 REVERSIBLE SOLVERS

1035

1036 The earliest work on reversible solvers can be traced back to the pioneering work on symplectic
1037 integrators by [Vogelaere \(1956\)](#); [Ruth \(1983\)](#); [Feng \(1984\)](#). Due to symplectic integrators being
1038 developed for solving Hamiltonian systems they are intrinsically reversible by construction ([Grey-](#)
1039 [danus et al., 2019](#)). More recently, [Matsubara et al. \(2021\)](#) explored the use of symplectic solvers
1040 for solving the continuous adjoint equations. Likewise, work by [Pan et al. \(2023\)](#) extended this idea,
1041 making use of symplectic solvers for solving the continuous adjoint equations for diffusion models.
1042 However, in this section we will focus on non-symplectic reversible solvers.

1043 Throughout this section we consider solving the following d -dimensional IVP:

1044
$$\mathbf{x}(0) = \mathbf{x}_0, \quad \frac{d\mathbf{x}}{dt}(t) = \mathbf{f}(t, \mathbf{x}(t)), \quad (17)$$
1045

1046 over the time interval $[0, T]$ with numerical solution $\{\mathbf{x}_n\}_{n=0}^N$.

1047

A.1.1 ASYNCHRONOUS LEAPFROG METHOD

1048

1049 To the best of our knowledge the *asynchronous leapfrog definition* was the first algebraically reversible
1050 non-symplectic solver, initially proposed by [Mutze \(2013\)](#) and popularized in a modern deep learning
1051 context by [Zhuang et al. \(2021\)](#). The asynchronous leapfrog method is a modification of the leapfrog
1052 method which converts it from a multi-step to single-step method. The method keeps track of a
1053 second state, $\{\mathbf{v}_n\}$ which is supposed to be *sufficiently close* to the value of the vector field. We
1054 define the method below in Definition A.1.

1055 **Definition A.1** (Asynchronous leapfrog method). Initialize $\mathbf{v}_0 = \mathbf{f}(0, \mathbf{x}_0)$. Consider a step size of h
1056 and let $\hat{t}_n = t_n + h/2$, then a forward step of the asynchronous leapfrog method is defined as

1057
$$\begin{aligned} \hat{\mathbf{x}}_n &= \mathbf{x}_n + \frac{1}{2} \mathbf{v}_n h, \\ \mathbf{v}_{n+1} &= 2\mathbf{f}(\hat{t}_n, \hat{\mathbf{x}}_n) - \mathbf{v}_n, \\ \mathbf{x}_{n+1} &= \mathbf{x}_n + \mathbf{f}(\hat{t}_n, \hat{\mathbf{x}}_n)h, \end{aligned} \quad (18)$$
1058

1059 and the backward step is given as

1060
$$\begin{aligned} \hat{\mathbf{x}}_n &= \mathbf{x}_{n+1} - \frac{1}{2} \mathbf{v}_{n+1} h, \\ \mathbf{x}_n &= \mathbf{x}_{n+1} - \mathbf{f}(\hat{t}_n, \hat{\mathbf{x}}_n)h, \\ \mathbf{v}_n &= 2\mathbf{f}(\hat{t}_n, \hat{\mathbf{x}}_n) - \mathbf{v}_{n+1}. \end{aligned} \quad (19)$$
1061

1062 **Remark A.2.** The method is a second-order solver ([Zhuang et al., 2021](#), Theorem 3.1).

1063

A.1.2 REVERSIBLE HEUN METHOD

1064

1065 Later work by [Kidger et al. \(2021\)](#) proposed the *reversible Heun method*, a general purpose reversible
1066 solver which is symmetric and is an algebraically reversible SDE solver in addition to being a
1067 reversible ODE solver. This solver keeps track of an auxiliary state variable $\hat{\mathbf{x}}_n$ and an extra copy
1068 of previous evaluations of the drift and diffusion coefficients. We present this method below in
1069 Definition A.3.

1080
1081 **Definition A.3** (Reversible Heun method for ODEs). Initialize $\hat{\mathbf{x}}_0 = \mathbf{x}_0$. Consider a step size of h ,
1082 then a forward step of the reversible Heun method is defined as

$$1083 \hat{\mathbf{x}}_{n+1} = 2\mathbf{x}_n - \hat{\mathbf{x}}_n + \mathbf{f}(t_n, \hat{\mathbf{x}}_n)h, \\ 1084 \mathbf{x}_{n+1} = \mathbf{x}_n + \frac{1}{2} (\mathbf{f}(t_{n+1}, \hat{\mathbf{x}}_{n+1}) + \mathbf{f}(t_n, \hat{\mathbf{x}}_n))h. \quad (20)$$

1085 and the backward step is given as

$$1086 \hat{\mathbf{x}}_n = 2\mathbf{x}_{n+1} - \hat{\mathbf{x}}_{n+1} - \mathbf{f}(t_{n+1}, \hat{\mathbf{x}}_{n+1})h, \\ 1087 \mathbf{x}_n = \mathbf{x}_{n+1} - \frac{1}{2} (\mathbf{f}(t_{n+1}, \hat{\mathbf{x}}_{n+1}) + \mathbf{f}(t_n, \hat{\mathbf{x}}_n))h. \quad (21)$$

1088 **Remark A.4.** This method is a second-order solver (Kidger, 2022, Theorem 5.18).

1089 Recall that simulating SDEs in reverse-time is much trickier than simulating ODEs in reverse-time.
1090 This observation is even more true of algebraically reversible methods for SDEs. To the best of our
1091 knowledge, the only general reversible solver for SDEs is the reversible Heun method. The main
1092 idea of the SDE formulation of the reversible Heun method is to extend the Euler-Heun method⁷ like
1093 how Heun's method was extended to the reversible Heun solver for ODEs. We define the method in
1094 Kidger et al. (2021, Algorithm 1) below in Definition A.5.

1095 **Definition A.5** (Reversible Heun method for SDEs). Initialize $\hat{\mathbf{x}}_0 = \mathbf{x}_0$. Consider a step size of h
1096 and let $\mathbf{W}_h := \mathbf{W}_{t_{n+1}} - \mathbf{W}_{t_n}$, then a forward step of the reversible Heun method is defined as

$$1097 \hat{\mathbf{x}}_{n+1} = 2\mathbf{x}_n - \hat{\mathbf{x}}_n + \boldsymbol{\mu}(t_n, \hat{\mathbf{x}}_n)h + \boldsymbol{\sigma}(t_n, \hat{\mathbf{x}}_n)\mathbf{W}_h, \\ 1098 \mathbf{x}_{n+1} = \mathbf{x}_n + \frac{1}{2} (\boldsymbol{\mu}(t_{n+1}, \hat{\mathbf{x}}_{n+1}) + \boldsymbol{\mu}(t_n, \hat{\mathbf{x}}_n))h \\ 1099 + \frac{1}{2} (\boldsymbol{\sigma}(t_{n+1}, \hat{\mathbf{x}}_{n+1}) + \boldsymbol{\sigma}(t_n, \hat{\mathbf{x}}_n))\mathbf{W}_h. \quad (22)$$

1100 and the backward step is given as

$$1101 \hat{\mathbf{x}}_n = 2\mathbf{x}_{n+1} - \hat{\mathbf{x}}_{n+1} - \boldsymbol{\mu}(t_{n+1}, \hat{\mathbf{x}}_{n+1})h - \boldsymbol{\sigma}(t_n, \hat{\mathbf{x}}_n)\mathbf{W}_h, \\ 1102 \mathbf{x}_n = \mathbf{x}_{n+1} - \frac{1}{2} (\boldsymbol{\mu}(t_{n+1}, \hat{\mathbf{x}}_{n+1}) + \boldsymbol{\mu}(t_n, \hat{\mathbf{x}}_n))h \\ 1103 - \frac{1}{2} (\boldsymbol{\sigma}(t_{n+1}, \hat{\mathbf{x}}_{n+1}) + \boldsymbol{\sigma}(t_n, \hat{\mathbf{x}}_n))\mathbf{W}_h. \quad (23)$$

1104 **Remark A.6.** This method requires some tractable solution for recalculating the Brownian motion
1105 from a splittable PRNG.

1106 A.1.3 MCCALLUM-FOSTER METHOD

1107 Recent work by McCallum & Foster (2024) created a general method for constructing n -th order
1108 solvers from preexisting explicit single-step solvers while also addressing the stability issues that
1109 earlier methods suffered from. As McCallum & Foster (2024) simply refer to their method as
1110 *reversible X* where X is the underlying single-step solver we opt to refer to their method as the
1111 *McCallum-Foster method*. We restate the definition below.

1112 **Definition 2.1** (McCallum-Foster method). Initialize $\hat{\mathbf{x}}_0 = \mathbf{x}_0$ and let $\zeta \in (0, 1]$. Consider a step
1113 size of h , then a forward step of the McCallum-Foster method is defined as

$$1114 \hat{\mathbf{x}}_{n+1} = \zeta \mathbf{x}_n + (1 - \zeta) \hat{\mathbf{x}}_n + \Phi_h(t_n, \hat{\mathbf{x}}_n), \\ 1115 \mathbf{x}_{n+1} = \hat{\mathbf{x}}_n - \Phi_{-h}(t_{n+1}, \mathbf{x}_{n+1}), \quad (5)$$

1116 and the backward step is given as

$$1117 \hat{\mathbf{x}}_n = \hat{\mathbf{x}}_{n+1} + \Phi_{-h}(t_{n+1}, \mathbf{x}_{n+1}), \\ 1118 \mathbf{x}_n = \zeta^{-1} \mathbf{x}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{x}}_n - \zeta^{-1} \Phi_h(t_n, \hat{\mathbf{x}}_n). \quad (6)$$

1119 **Remark A.7.** N.B., the ζ and ζ^{-1} terms in the forward and backward steps determine the stability of
1120 the system.

1121
1122 ⁷This converges with strong order $\frac{1}{2}$ in the Stratonovich sense (Rüemelin, 1982).

Interestingly, [McCallum & Foster \(2024, Theorem 2.1\)](#) showed that this reversible method inherits the convergence order of single-step solver Φ_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 (17) 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 Φ be a k -th order ODE solver such that it satisfies the Lipschitz condition*

$$\|\Phi_\eta(\cdot, \mathbf{a}) - \Phi_\eta(\cdot, \mathbf{b})\| \leq L|\eta|\|\mathbf{a} - \mathbf{b}\|, \quad (24)$$

for all $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ and $\eta \in [-h_{\max}, h_{\max}]$ for some $h_{\max} > 0$. Consider the reversible solution $\{\mathbf{x}_n, \hat{\mathbf{x}}_n\}_{n \in \mathbb{N}}$ admitted by Equation (5). Then there exists constants $h_{\max} > 0$, $C > 0$, such that, for $h \in (0, h_{\max}]$,

$$\|\mathbf{x}_n - \mathbf{x}(t_n)\| \leq Ch^k. \quad (25)$$

A.2 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.8](#).

Definition A.8 (Region of stability). Fix some numerical differential equation solver and let $\{\mathbf{x}_n^{\lambda, h}\}_{n \in \mathbb{N}}$ be the solution admitted by the numerical scheme solving the linear (or Dahlquist) test equation

$$\mathbf{x}(0) = \mathbf{x}_0, \quad \frac{d\mathbf{x}}{dt} = \lambda \mathbf{x}(t), \quad (26)$$

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

$$\{h\lambda \in \mathbb{C} : \{\mathbf{x}_n^{\lambda, h}\}_{n \in \mathbb{N}} \text{ is uniformly bounded over } t_n\}. \quad (27)$$

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

With the linear test equation Equation (26) the ODE converges asymptotically when $\Re(\lambda) \leq 0$,⁸ 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 $\Re(\lambda) < 0$.⁹ Thus, the larger the region of stability the larger the step size we can take, wherein the numerical scheme still converges.

Remark A.9. 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.

In other words, all previous reversible solvers are nowhere linearly stable for any step size h .¹⁰ 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

$2\mathbf{f}(\hat{t}_n, \hat{\mathbf{x}}_n) - \mathbf{v}_n,$	(asynchronous leapfrog)
$2\mathbf{x}_{n+1} - \hat{\mathbf{x}}_{n+1}.$	(reversible Heun)

Thus the instability in these reversible schemes is caused by a decoupling between \mathbf{v}_n and $\mathbf{f}(t_n, \mathbf{x}_n)$ (asynchronous leapfrog); and \mathbf{x}_n and $\hat{\mathbf{x}}_n$ (reversible Heun). The strategy of [McCallum & Foster \(2024\)](#) is to couple \mathbf{x}_n and $\hat{\mathbf{x}}_n$ 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 $\Phi_h(t_n, \mathbf{x}_n) = R(h\lambda)\mathbf{x}_n$ 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.

⁸The ODE converges to 0 when $\Re(\lambda) < 0$.

⁹A region of stability which satisfies is known as a region of absolute stability.

¹⁰Linearly stability refers to stability for linear test equations with $\Re(\lambda) < 0$.

1188
 1189 **Theorem A.2** (Region of stability for the McCallum-Foster method). *Let Φ be given by an explicit*
 1190 *Runge-Kutta solver. Then the reversible numerical solution $\{\mathbf{x}_n, \hat{\mathbf{x}}_n\}_{n \in \mathbb{N}}$ given by Equation (5) is*
 1191 *linearly stable iff*

$$1192 \quad |\Gamma| < 1 + \zeta, \quad (28)$$

1193 *where*

$$1194 \quad \Gamma = 1 + \zeta - (1 - \zeta)R(-h\lambda) - R(-h\lambda)R(h\lambda). \quad (29)$$

1195 **Remark A.10.** The McCallum-Foster method when constructed from explicit Runge-Kutta methods
 1196 have a *non-trivial* region of stability. Note, however, that this region of stability is smaller than the
 1197 original region of stability from the original Runge-Kutta method.

1199 A.3 EXACT INVERSION OF DIFFUSION MODELS

1201 Independent of the work on reversible solvers for neural ODEs several researchers have developed
 1202 reversible methods for solving the probability flow ODE—often in the literature on diffusion models
 1203 this is called the *exact inversion* of diffusion models.

1204 A.3.1 EDICT SAMPLER

1206 The first work to explore this topic of exact inversion with diffusion models was that of Wallace et al.
 1207 (2023), who inspired by coupling layers in normalizing flows (Dinh et al., 2015) proposed a reversible
 1208 solver which they refer to as *exact diffusion inversion via coupled transformations* (EDICT). Like
 1209 all reversible solvers this method keeps track of an extra state, denoted by $\{\mathbf{y}_n\}_{n \in \mathbb{N}}$, with $\mathbf{y}_0 = \mathbf{x}_0$.
 1210 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

$$1212 \quad \mathbf{x}_n^{\text{inter}} = a_n \mathbf{x}_n + b_n \mathbf{x}_{T|t_n}^\theta(\mathbf{y}_n), \\ 1213 \quad \mathbf{y}_n^{\text{inter}} = a_n \mathbf{y}_n + b_n \mathbf{x}_{T|t_n}^\theta(\mathbf{x}_n^{\text{inter}}), \\ 1214 \quad \mathbf{x}_{n+1} = \xi \mathbf{x}_n^{\text{inter}} + (1 - \xi) \mathbf{y}_n^{\text{inter}} \\ 1215 \quad \mathbf{y}_{n+1} = \xi \mathbf{x}_n^{\text{inter}} + (1 - \xi) \mathbf{x}_{n+1}, \quad (30)$$

1218 where $\xi \in (0, 1)$ is a mixing parameter.¹¹ This method can be inverted to obtain a closed form
 1219 expression for backward step:

$$1220 \quad \mathbf{y}_n^{\text{inter}} = \frac{\mathbf{y}_{n+1} - (1 - \xi) \mathbf{x}_{n+1}}{\xi}, \\ 1221 \quad \mathbf{x}_n^{\text{inter}} = \frac{\mathbf{y}_{n+1} - (1 - \xi) \mathbf{y}_n^{\text{inter}}}{\xi}, \\ 1222 \quad \mathbf{y}_n = \frac{\mathbf{y}_n^{\text{inter}} - b_n \mathbf{x}_{T|t_n}^\theta(\mathbf{x}_n^{\text{inter}})}{a_n}, \\ 1223 \quad \mathbf{x}_n = \frac{\mathbf{x}_n^{\text{inter}} - b_n \mathbf{x}_{T|t_n}^\theta(\mathbf{y}_n)}{a_n}. \quad (31)$$

1230 Notably, the EDICT solver was developed in the context of discrete-time diffusion models and the
 1231 connection to reversible solvers for ODEs was not considered in the original work. *N.B.*, to the best
 1232 of our knowledge our work is the first to draw the connection between the work on reversible ODE
 1233 solvers and exact inversion with diffusion models. Unfortunately, this method suffers from poor
 1234 convergence issues (see Remark A.11) and generally has poor performance when used to perform
 1235 sampling with diffusion models, thereby limiting its utility in practice (Zhang et al., 2024; Wang
 1236 et al., 2024).

1237 **Remark A.11.** Later work by Wang et al. (2024, Proposition 6) showed that EDICT is actually a
 1238 zero-order method, *i.e.*, the local truncation error is $\mathcal{O}(h)$, making it generally unsuitable in practice.

1239 ¹¹In practice, when used for image editing the authors found that the parameter ξ controlled how closely the
 1240 EDICT sampler aligned with the original sample, with lower values corresponding to higher agreement with the
 1241 original sample.

1242 A.3.2 BDIA SAMPLER
1243

1244 Later work by [Zhang et al. \(2024\)](#) proposed a reversible solver for the probability flow ODE which
1245 they call *bidirectional integration approximation* (BDIA). The core idea is to use both single-step
1246 methods $\Phi_{t_n, t_{n-1}}$ and $\Phi_{t_n, t_{n+1}}$ to induce reversibility.¹² Then using these two approximations—
1247 both of which are computed from a discretization centered around \mathbf{x}_n —the process is update via a
1248 multistep process with a forward step of¹³

$$1249 \quad \mathbf{x}_{n+1} = \mathbf{x}_{n-1} - \Phi_{t_n, t_{n-1}}(\mathbf{x}_n) + \Phi_{t_n, t_{n+1}}(\mathbf{x}_n). \quad (32)$$

1250 The backwards step can easily be expressed as

$$1252 \quad \mathbf{x}_{n-1} = \mathbf{x}_{n+1} + \Phi_{t_n, t_{n-1}}(\mathbf{x}_n) + \Phi_{t_n, t_{n+1}}(\mathbf{x}_n). \quad (33)$$

1254 In practice, BDIA uses the DDIM solver (*i.e.*, Euler) for Φ , but in theory one could use a higher-order
1255 method—this was not explored in [Zhang et al. \(2024\)](#).

1256 **Proposition A.3** (BDIA is the leapfrog/midpoint method). *The BDIA method described in Equation*
1257 *(32) is the leapfrog/midpoint method when $\Phi_h(t, \mathbf{x}) = h\mathbf{u}_t^\theta(\mathbf{x})$, i.e., the Euler step.*

1259 *Proof.* This can be shown rather straightforwardly by substitution, *i.e.*,

$$1261 \quad \mathbf{x}_{n+1} = \mathbf{x}_{n-1} + 2h\mathbf{u}_{t_n}^\theta(\mathbf{x}_n). \quad (34)$$

□

1264 **Corollary A.3.1** (BDIA is a first-order method). *BDIA is first-order method*, *i.e.*, the local truncation
1265 error is $\mathcal{O}(h^2)$.

1266 **Remark A.12.** This result was also observed in [Wang et al. \(2024\)](#), Proposition 6).

1268 **Corollary A.3.2** (BDIA is nowhere linearly stable). *BDIA is nowhere linearly stable*, *i.e.*, the region
1269 of stability is the complex interval $[-i, i]$.

1271 *Proof.* This follows straightforwardly from Proposition A.3 and [Shampine \(2009\)](#), Section 2). □

1273 [Zhang et al. \(2024\)](#) introduce a hyperparameter $\gamma \in [0, 1]$ which is used below

$$1275 \quad \hat{\Phi}_{t_n, t_{n-1}}(\mathbf{x}_n) = (1 - \gamma)(\mathbf{x}_{n-1} - \mathbf{x}_n) + \gamma\Phi_{t_n, t_{n-1}}(\mathbf{x}_n), \quad (35)$$

1277 to modify the BDIA update rule in Equation (32). Thus, γ can be viewed as a parameter which
1278 interpolates between the midpoint and Euler schemes. For image editing applications the authors
1279 found this parameter to control how closely the BDIA sampler aligned with the original image, with
1280 lower values corresponding to higher agreement with the original image (making it similar to the ξ
1281 parameter from BDIA).

1282 A.3.3 BELM SAMPLER
1283

1284 Recently, [Wang et al. \(2024\)](#) proposed a linear multi-step reversible solver for the probability flow
1285 ODE called the *bidirectional explicit linear multi-step* (BELM) sampler. First, they reparameterize
1286 the probability flow ODE as

$$1287 \quad d\bar{\mathbf{x}}(t) = \bar{\mathbf{x}}_{T|\bar{\sigma}_t}^\theta(\bar{\mathbf{x}}(t)) d\bar{\sigma}_t, \quad (36)$$

1288 where $\bar{\mathbf{x}}(t) := \mathbf{x}(t)/\alpha_t$, $\bar{\sigma}(t) := \sigma_t/\alpha_t$, and $\bar{\mathbf{x}}_{T|\bar{\sigma}_t}^\theta(\bar{\mathbf{x}}(t)) = \mathbf{x}_{T|t}^\theta(\mathbf{x}(t))$.¹⁴ The BELM sampler
1289 makes use of the variable-step-size-variable-formula (VSVF) linear multi-step methods ([Crouzeix](#)

1291 ¹²*N.B.*, in the original paper, [Zhang et al. \(2024\)](#) use quite different notation for explaining their idea; however,
1292 we find our presentation to be simpler for the reader as it more easily enables comparison to other methods.

1293 ¹³In some sense, this is reminiscent of the idea from the more general McCallum-Foster method; however,
1294 this approach results in a multi-step method unlike the single-step method of [McCallum & Foster \(2024\)](#).

1295 ¹⁴*N.B.*, this is a popular parameterization of diffusion models and affine conditional flows. This can be done
1296 *mutatis mutandis* for target prediction models retrieving ([Blasingame & Liu, 2025](#), Proposition D.2).

& Lisbona, 1984) to construct the numerical solver. The k -step VSVF linear multi-step method for solving the reparameterized probability flow ODE in Equation (36) is given by

$$\bar{\mathbf{x}}_{n+1} = \sum_{m=1}^k a_{n,m} \bar{\mathbf{x}}_{n+1-m} \quad (37)$$

$$+ \sum_{m=1}^{k-1} b_{n,m} h_{n+1-m} \bar{\mathbf{x}}_{T|\bar{\sigma}_{n+1-m}}^\theta(\bar{\mathbf{x}}_{n+1-m}). \quad (38)$$

where $a_{n,m} \neq 0$,¹⁵ 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

$$\bar{\mathbf{x}}_{n+1-k} = \frac{1}{a_{n,k}} \bar{\mathbf{x}}_{n+1} - \sum_{m=1}^{k-1} \frac{a_{n,m}}{a_{n,k}} \bar{\mathbf{x}}_{n+1-m} \quad (39)$$

$$- \sum_{m=1}^{k-1} \frac{b_{n,m}}{a_{n,k}} h_{n+1-m} \bar{\mathbf{x}}_{T|\bar{\sigma}_{n+1-m}}^\theta(\bar{\mathbf{x}}_{n+1-m}). \quad (40)$$

Remark A.13. 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.14. Interestingly, the earlier EDICT and BDIA methods can be viewed as instances of the BELM method (Wang et al., 2024, Appendices 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 as¹⁶

$$\bar{\mathbf{x}}_{n+1} = \frac{h_n^2}{h_{n-1}^2} \bar{\mathbf{x}}_{n-1} + \frac{h_{n-1}^2 + h_n^2}{h_{n-1}^2} \bar{\mathbf{x}}_n - \frac{h_n(h_n + h_{n+1})}{h_{n+1}} \bar{\mathbf{x}}_{0|\bar{\sigma}_n}(\bar{\mathbf{x}}_n). \quad (41)$$

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

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 $h_n = h$. \square

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.3.4 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 $\mathbf{x}_t \sim \mathcal{N}(\alpha_t \mathbf{x}_0 \mid \sigma_t^2 \mathbf{I})$, then given \mathbf{x}_s and noise $\epsilon_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ we can calculate

$$\mathbf{x}_t = \frac{\alpha_t}{\alpha_s} \mathbf{x}_s + 2\sigma_t(e^h - 1) \hat{\mathbf{x}}_{T|s}(\mathbf{x}_s) + \sigma_t \sqrt{e^{2h} - 1} \epsilon_s. \quad (42)$$

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

$$\epsilon_s = \frac{\mathbf{x}_t - \frac{\alpha_t}{\alpha_s} \mathbf{x}_s + 2\sigma_t(e^h - 1) \hat{\mathbf{x}}_{T|s}(\mathbf{x}_s, \mathbf{z}, s)}{\sigma_t \sqrt{e^{2h} - 1}} \quad (43)$$

With this the sequence $\{\epsilon_{t_i}\}_{i=1}^N$ 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.

¹⁵This is to ensure that the method is reversible.

¹⁶N.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.

1350
 1351 Table 3: Comparison of different (non-symplectic) reversible ODE solvers. We note that some of
 1352 the solvers were developed particularly for the probability flow ODE (an affine conditional flow)
 1353 whilst others work for general ODEs. In the first column we denote the number of extra states the
 1354 numerical scheme needs to keep in memory to ensure algebraic reversibility. For BELM k denotes
 1355 the number of steps and for McCallum-Foster k denotes the convergence order of the underlying
 1356 single-step solver. For the column labeled *region of linear stability* we mean there exists some subset
 1357 of \mathbb{C} which is the region of stability and the set is not a null set. The proof of convergence for BELM
 1358 is only provided for the special case (called *O-BELM* in Wang et al. (2024)) with $k = 2$.
 1359
 1360

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

A.3.5 SUMMARY

We present a summary of related works on either *exact inversion* or *reversible solvers* below in Table 3. *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.4 SDE SOLVERS FOR DIFFUSION MODELS

Next we discuss related works on SDE solvers for the reverse-time diffusion SDE in Equation (3). 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 straightforwardly 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 (1) 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 Ψ ; however, upon closer inspection they are some key fundamental differences.¹⁷ 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.

¹⁷*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.

1404 A.4.1 COMPARISON WITH SEEDS
1405

1406 Mostly directly relevant to our work on constructing a stochastic Ψ is the SEEDS family of solvers
1407 proposed by [Gonzalez et al. \(2024\)](#). Similar to us, they also approach using exponential methods to
1408 simplify the expression of diffusion models [Gonzalez et al. \(2024, Appendix B.1\)](#). There are two *key*
1409 distinctions, namely, 1) that they use the *stochastic exponential time differencing* (SETD) method
1410 ([Adamu, 2011](#)), whereas, we construct stochastic Lawson schemes;¹⁸ and 2) that they use a different
1411 technique for modeling the iterated stochastic integrals for high-order solvers. In particular, SEEDS
1412 introduces a decomposition for the iterated stochastic integrals produced by the Itô-Taylor expansions
1413 of Equation (3) such that the decomposition preserves the Markov property, *i.e.*, the random variables
1414 used to construct model the Brownian increments from iterated integrals are independent on non-
1415 overlapping intervals and dependent on overlapping intervals (see [Gonzalez et al., 2024, Proposition](#)
1416 4.3). By making use of the SRK schemes of [Foster et al. \(2024\)](#) developed from using the space-time
1417 Lévy area to construct high-order splitting methods we have an alternative method for ensuring this
1418 property. This results in our solver based on ShARK (see Appendix B.3, *cf.* Theorem 4.2) having a
1419 strong order of convergence of 1.5; whereas, SEEDS-3 only achieves a *weak* order of convergence of
1420 1.

1421 This brings us to another large difference, the SEEDS solvers focus on the *weak* approximation to
1422 Equation (3); whereas, as we are concerned with the *strong* approximation to Equation (3). The
1423 difference between these two is that the weak convergence is considered with the precisions of the
1424 *moments*; whereas, strong convergence is concerned with the precision of the *path*. Moreover, by
1425 definition a strong order of convergence implies a weak order of convergence, the converse is not true.
1426 In particular, for our application of developing *reversible* schemes this strong order of convergence is
1427 particularly important as we care about the path. Thus the technique SEEDS uses to replace iterated
1428 Itô integrals with other random variables with equivalent moment conditions is *wholly unsuitable*
1429 for our purposes as we desire a *strong* approximation.

1430 B STOCHASTIC RUNGE-KUTTA METHODS
1431

1432 Recall that the general Butcher tableau for a s -stage explicit RK scheme ([Stewart, 2022](#), Section
1433 6.1.4) for a generic ODE is written as

$$\begin{array}{c|ccccc} c_1 & & & & & \\ \hline c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s(s-1)} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} = \begin{array}{c|c} c & a \\ \hline & b \end{array}. \quad (44)$$

1442 *E.g.*, the famous 4-th order Runge-Kutta (RK4) method is given by
1443

$$\begin{array}{c|ccccc} 0 & & & & & \\ \hline \frac{1}{2} & \frac{1}{2} & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & \\ 1 & 0 & 0 & 1 & & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \end{array}. \quad (45)$$

1451 However, for SDEs this is much trickier due to the presence of iterated stochastic integrals in the
1452 Itô-Taylor or Stratonovich-Taylor expansions ([Kloeden & Platen, 1992](#)). Consider a d -dimensional
1453 Stratonovich SDE driven by d_w -dimensional Brownian motion $\{W_t\}_{t \in [0, T]}$ defined as
1454

$$d\mathbf{X}_t = \boldsymbol{\mu}_\theta(t, \mathbf{X}_t) dt + \boldsymbol{\sigma}_\theta(t, \mathbf{X}_t) \circ dW_t, \quad (46)$$

1455 ¹⁸*N.B.*, for certain scenarios these two different viewpoints converge, particularly, in the deterministic case.
1456 See our discussion on the family of DPM-Solvers which also use (S)ETD in Appendix E.

where $\mu_\theta \in \mathcal{C}^2(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^d)$ and $\sigma_\theta \in \mathcal{C}^3(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^{d \times d_w})$ satisfy the usual regularity conditions for Stratonovich SDEs (Øksendal, 2003, Theorem 5.2.1) and where $\circ d\mathbf{W}_t$ 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 (46) as

$$\begin{aligned} \mathbf{Z}_i^{(0)} &= \mathbf{X}_n + h \sum_{j=1}^{i-1} a_{ij}^{(0)} \mu_\theta(t_n + c_j^{(0)}, \mathbf{Z}_j^{(0)}), \\ \mathbf{Z}_i^{(k)} &= \mathbf{X}_n + h \sum_{j=1}^{i-1} a_{ij}^{(1)} \mu_\theta(t_n + c_j^{(0)}, \mathbf{Z}_j^{(0)}) + \sum_{j=1}^{i-1} \sum_{l=1}^{d_w} a_{ij}^{(2)} \mathbf{I}_{(l,k),n} \sigma_\theta(t_n + c_j^{(1)}, \mathbf{Z}_i^{(l)}), \\ \mathbf{X}_{n+1} &= \mathbf{X}_n + h \sum_{i=1}^s b_i^{(0)} \mu_\theta(t_n + c_i^{(0)}, \mathbf{Z}_j^{(0)}) + \sum_{i=1}^s \sum_{k=1}^{d_w} \left(b_i^{(1)} \mathbf{I}_{(k),n} + b_i^{(2)} \right) \sigma_\theta(t_n + c_i^{(1)}, \mathbf{Z}_i^{(k)}), \end{aligned} \quad (47)$$

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

$$\mathbf{I}_{(k),n} = \int_{t_n}^{t_{n+1}} \circ d\mathbf{W}_u^k = \mathbf{W}_{t_{n+1}}^k - \mathbf{W}_{t_n}^k, \quad (48)$$

$$\mathbf{I}_{(l,k),n} = \int_{t_n}^{t_{n+1}} \int_{t_n}^u \circ d\mathbf{W}_v^l \circ d\mathbf{W}_u^k, \quad (49)$$

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

$$\begin{array}{c|c|c|c} c^{(0)} & a^{(0)} & & \\ \hline c^{(1)} & a^{(1)} & a^{(2)} & \\ \hline & b^{(0)} & b^{(1)} & b^{(2)} \end{array} \quad (50)$$

These iterated integrals $\mathbf{I}_{(l,k),n}$ are very tricky to work with and can raise up many practical concerns. As alluded to earlier (cf. Section A.4.1) 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 (46) Foster et al. (2024) write an s -stage SRK as

$$\begin{aligned} \mu_\theta^i &= \mu_\theta(t_n + c_i h, \mathbf{Z}_i), \\ \sigma_\theta^i &= \sigma_\theta(t_n + c_i h, \mathbf{Z}_i), \\ \mathbf{Z}_i &= \mathbf{X}_n + h \left(\sum_{j=1}^{i-1} a_{ij} \mu_\theta^j \right) + \mathbf{W}_n \left(\sum_{j=1}^{i-1} a_{ij}^W \sigma_\theta^j \right) + \mathbf{H}_n \left(\sum_{j=1}^{i-1} a_{ij}^H \sigma_\theta^j \right), \\ \mathbf{X}_{n+1} &= \mathbf{X}_n + h \left(\sum_{i=1}^s b_i \mu_\theta^i \right) + \mathbf{W}_n \left(\sum_{i=1}^s b_i^W \sigma_\theta^i \right) + \mathbf{H}_n \left(\sum_{i=1}^s b_i^H \sigma_\theta^i \right), \end{aligned} \quad (51)$$

where $h = t_{n+1} - t_n$ is the step size and $\mathbf{W}_n := \mathbf{W}_{t_n, t_{n+1}}$ and $\mathbf{H}_n := \mathbf{H}_{t_n, t_{n+1}}$ are the Brownian and space-time Lévy increments (cf. Definition 3.2) 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

$$\begin{array}{c|c|c|c} c & a & a^W & a^H \\ \hline & b & b^W & b^H \end{array} \quad (52)$$

1512 *E.g.*, we can write the famous Euler-Maruyama scheme as
 1513

$$\begin{array}{c|c|c|c} 0 & 0 & 0 & 0 \\ \hline & 1 & 1 & 0 \end{array} \quad (53)$$

1517 B.2 INDEPENDENCE OF THE BROWNIAN AND LÉVY INCREMENTS

1519 Remarkably, in [Foster et al. \(2020, Theorem 2.2\)](#) present a polynomial Karhunen-Loëve theorem for
 1520 the Brownian bridge (*cf.* Definition [G.1](#))—picture an stochastic analogue to the Fourier series of a
 1521 function on a bounded interval—which leads to a most useful remark ([Foster et al., 2020, Remark](#)
 1522 [3.6](#)) which we restate below.

1523 **Remark B.1.** We have that $H_{s,t} \sim \mathcal{N}(0, \frac{1}{12}h)$ is independent of $W_{s,t}$ when $d = 1$, likewise, since
 1524 the coordinate processes of a Brownian motion are independent, one can write $\mathbf{W}_{s,t} \sim (0, h\mathbf{I})$ and
 1525 $\mathbf{H}_{s,t} \sim \mathcal{N}(0, \frac{1}{12}h\mathbf{I})$ are independent.

1527 Thus we have found another remedy to the problem of independent increments, whilst still being able
 1528 to obtain a *strong* approximation of the SDE.

1530 B.3 SHARK

1532 Recently, [Foster et al. \(2024\)](#) developed *shifted additive-noise Runge-Kutta* (ShARK) for additive
 1533 noise SDEs which is based on [Foster et al. \(2024, Equation \(6.1\)\)](#). This scheme has converges
 1534 strongly with order 1.5 for additive-noise SDEs and makes two evaluations of the drift and diffusion
 1535 per step.

1536 ShARK is described via the following extended Butcher tableau

$$\begin{array}{c|c|c|c} 0 & & 0 & 1 \\ \hline \frac{5}{6} & \frac{5}{6} & \frac{5}{6} & 1 \\ \hline & 0.4 & 0.6 & 1 \\ & -0.6 & 0.6 & 0 \end{array} \quad (54)$$

1544 The second row for the b variable describes the coefficients used for adaptive-step size
 1545 solvers to approximate the error at each step. The Butcher tableau for this scheme
 1546 can be found here: https://github.com/patrick-kidger/diffrax/blob/main/diffrax/_solver/shark.py.

1548 C DERIVATION OF REX

1551 We derive the Rex scheme presented in Proposition [3.3](#) in the main paper.

1553 C.1 REX (ODE)

1555 In this section we derive the Rex scheme for the probability flow ODE. We present derivations for
 1556 both the data prediction and noise prediction formulations.

1557 C.1.1 PROOF OF PROPOSITION 3.1

1559 We restate Proposition [3.1](#) below.

1560 **Proposition 3.1** (Reparameterization of the probability flow ODE). *The probability flow ODE in*
 1561 *Equation (7) can be rewritten in ς_t as*

$$\frac{d\mathbf{y}_\varsigma}{d\varsigma} = \beta_T \mathbf{f}_\theta \left(\varsigma, \frac{\beta_\varsigma}{\beta_t} \mathbf{y}_\varsigma \right), \quad (8)$$

1563 where $\mathbf{y}_t = \frac{\beta_T}{\beta_t} \mathbf{x}_t$.

1566 *Proof.* Recall that from Equation (7) we have that the ODE is given by
 1567

$$\frac{d\mathbf{x}_t}{dt} = \frac{\dot{\beta}_t}{\beta_t} \mathbf{x}_t + \frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\beta_t} \mathbf{f}_\theta(t, \mathbf{x}_t). \quad (55)$$

1570 We can use the technique of exponential integrators to rewrite the ODE as
 1571

$$\frac{d}{dt} \left[e^{\int_T^t -\frac{\dot{\beta}_u}{\beta_u} du} \mathbf{x}_t \right] = e^{\int_T^t -\frac{\dot{\beta}_u}{\beta_u} du} \frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\beta_t} \mathbf{f}_\theta(t, \mathbf{x}_t), \quad (56)$$

1573 recalling that we integrate from initial time T in reverse-time. Then the exponential terms simplify to
 1574

$$e^{\int_T^t -\frac{\dot{\beta}_u}{\beta_u} du} = \frac{\beta_0}{\beta_T}. \quad (57)$$

1577 We introduce a *change-of-variables* $\mathbf{y}_t = \frac{\beta_0}{\beta_T} \mathbf{x}_t$ to rewrite the ODE as
 1578

$$\frac{d\mathbf{y}_t}{dt} = \frac{\beta_T}{\beta_t} \underbrace{\frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\beta_t}}_{=\kappa_t} \mathbf{f}_\theta \left(t, \frac{\beta_T}{\beta_t} \mathbf{y}_t \right). \quad (58)$$

1581 Next we define
 1582

$$\dot{\varsigma}_t = \text{sgn}(\beta_T) \frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\beta_t^2}, \quad (59)$$

1585 which we will now justify. Now recall that β_t is either $-\alpha_t$ or σ_t depending on the whether \mathbf{f}_θ
 1586 denotes the data or noise prediction model. Moreover we know that α_t is a strictly monotonically
 1587 decreasing in t and that σ_t is a strictly monotonically increasing in t . We will now prove that there
 1588 exists and inverse function for ς_t such that $t_\varsigma(\varsigma_t) = t$ for both cases.

1589 **Case $\beta_t = -\alpha_t$.** We can write κ_t as
 1590

$$\kappa_t = \alpha_T \frac{\dot{\sigma}_t \alpha_t - \dot{\sigma}_t \dot{\alpha}_t}{\alpha_t^2}, \quad (60)$$

$$\stackrel{(i)}{=} \alpha_T \frac{d}{dt} \left(\frac{\sigma_t}{\alpha_t} \right), \quad (61)$$

1595 where (i) holds by the quotient rule. Clearly, we have that
 1596

$$\dot{\varsigma}_t = \frac{d}{dt} \left(\frac{\sigma_t}{\alpha_t} \right), \quad (62)$$

$$\varsigma_t = \frac{\sigma_t}{\alpha_t}, \quad (63)$$

1600 It follows from (α_t, σ_t) that ς_t is strictly monotonically increasing in t and thus we can construct its
 1601 inverse.

1602 **Case $\beta_t = \sigma_t$.** We can write κ_t as
 1603

$$\kappa_t = \sigma_T \frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\sigma_t^2}, \quad (64)$$

$$\stackrel{(i)}{=} \sigma_T \frac{d}{dt} \left(\frac{\alpha_t}{\sigma_t} \right), \quad (65)$$

1608 where (i) holds by the quotient rule. Clearly, we have that
 1609

$$\dot{\varsigma}_t = \frac{d}{dt} \left(\frac{\alpha_t}{\sigma_t} \right), \quad (66)$$

$$\varsigma_t = \frac{\alpha_t}{\sigma_t}, \quad (67)$$

1614 It follows from (α_t, σ_t) that ς_t is strictly monotonically decreasing in t and thus we can construct its
 1615 inverse.

1616 Thus we can rewrite the ODE via a time-change to find
 1617

$$\frac{d\mathbf{y}_\varsigma}{d\varsigma} = \beta_0 \mathbf{f}_\theta \left(\varsigma, \frac{\beta_T}{\beta_\varsigma} \mathbf{y}_\varsigma \right), \quad (68)$$

1618 with the usual *abuse-of-notation* $\mathbf{y}_\varsigma := \mathbf{y}_{t_\varsigma(\varsigma)}$, $\beta_\varsigma := \beta_{t_\varsigma(\varsigma)}$, &c. □
 1619

1620
1621 **Remark C.1.** When in the noise prediction formulation with Proposition 3.1 we recover the following
1622 reparameterization of Equation (7)

1623
$$\frac{d\mathbf{z}_\chi}{d\chi} = \alpha_T \mathbf{x}_{T|\chi}^\theta \left(\frac{\alpha_\chi}{\alpha_T} \mathbf{z}_\chi \right), \quad (69)$$

1624

1625 where $\alpha_T > 0$, $\mathbf{z}_t = \frac{\alpha_T}{\alpha_t} \mathbf{x}_t$ and $\chi_t = \frac{\sigma_t}{\alpha_t}$, which has been observed by numerous prior works (see
1626 [Song et al., 2021a](#), Equation (14); [Pan et al., 2023](#), Equation (11); [Wang et al., 2024](#), Equation (6)).

1627 **Remark C.2.** When in the data prediction formulation, Proposition 3.1 recovers [Blasingame & Liu](#)
1628 ([2025](#), Proposition D.2) which states that Equation (7) can be written as
1629

1630
$$\frac{d\mathbf{y}_\gamma}{d\gamma} = \sigma_T \mathbf{x}_{0|\gamma}^\theta \left(\frac{\sigma_\gamma}{\sigma_T} \mathbf{y}_\gamma \right), \quad (70)$$

1631

1632 where $\mathbf{y}_t = \frac{\sigma_T}{\sigma_t} \mathbf{x}_t$ and $\gamma_t = \frac{\alpha_t}{\sigma_t}$.

1634 C.1.2 DATA PREDICTION
1635

1636 We present this derivation in the form of Lemma C.1 below.

1637 **Lemma C.1** (Rex (ODE) for data prediction models). *Let Φ be an explicit Runge-Kutta solver for
1638 the ODE in Equation (70) with Butcher tableau a_{ij} , b_i , c_i . The reversible solver for Φ in terms of the
1639 original state \mathbf{x}_t is given by the forward step*

1640
$$\mathbf{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} (\zeta \mathbf{x}_n + (1 - \zeta) \hat{\mathbf{x}}_n) + \sigma_{n+1} \Psi_h(\gamma_n, \hat{\mathbf{x}}_n), \quad (71)$$

1641
1642
$$\hat{\mathbf{x}}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} \hat{\mathbf{x}}_n - \sigma_{n+1} \Psi_{-h}(\gamma_{n+1}, \mathbf{x}_{n+1}),$$

1643

1644 and backward step

1645
$$\hat{\mathbf{x}}_n = \frac{\sigma_n}{\sigma_{n+1}} \hat{\mathbf{x}}_{n+1} + \sigma_n \Psi_{-h}(\gamma_{n+1}, \mathbf{x}_{n+1}), \quad (72)$$

1646
1647
$$\mathbf{x}_n = \frac{\sigma_n}{\sigma_{n+1}} \zeta^{-1} \mathbf{x}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{x}}_n - \sigma_n \zeta^{-1} \Psi_h(\gamma_n, \hat{\mathbf{x}}_n),$$

1648

1649 with step size $h := \gamma_{n+1} - \gamma_n$ and where Ψ denotes the following scheme

1650
$$\hat{\mathbf{z}}_i = \frac{1}{\sigma_n} \mathbf{x}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{x}_{0|\gamma_n+c_j h}^\theta (\sigma_{\gamma_n+c_j h} \hat{\mathbf{z}}_j), \quad (73)$$

1651
1652
$$\Psi_h(\gamma_n, \mathbf{x}_n) = h \sum_{i=1}^s b_i \mathbf{x}_{0|\gamma_n+c_i h}^\theta (\sigma_{\gamma_n+c_i h} \hat{\mathbf{z}}_i),$$

1653

1654 *Proof.* Recall that the forward step of the McCallum-Foster method for Equation (70) given Φ is
1655 given as

1656
$$\begin{aligned} \mathbf{y}_{n+1} &= \zeta \mathbf{y}_n + (1 - \zeta) \hat{\mathbf{y}}_n + \Phi_h(\gamma_n, \hat{\mathbf{y}}_n), \\ \hat{\mathbf{y}}_{n+1} &= \hat{\mathbf{y}}_n - \Phi_{-h}(\gamma_{n+1}, \mathbf{y}_{n+1}), \end{aligned} \quad (74)$$

1657

1658 with step size $h = \gamma_{n+1} - \gamma_n$. We use the definition of $\mathbf{y}_t = \frac{\sigma_T}{\sigma_t} \mathbf{x}_t$ to rewrite the forward pass as
1659

1660
$$\begin{aligned} \mathbf{x}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n} (\zeta \mathbf{x}_n + (1 - \zeta) \hat{\mathbf{x}}_n) + \frac{\sigma_{n+1}}{\sigma_T} \Phi_h \left(\gamma_n, \frac{\sigma_T}{\sigma_n} \hat{\mathbf{x}}_n \right), \\ \hat{\mathbf{x}}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n} \hat{\mathbf{x}}_n - \frac{\sigma_{n+1}}{\sigma_T} \Phi_{-h} \left(\gamma_{n+1}, \frac{\sigma_T}{\sigma_{n+1}} \mathbf{x}_{n+1} \right). \end{aligned} \quad (75)$$

1661

1662 *Mutatis mutandis* we find the backward step in \mathbf{x}_t to be given as
1663

1664
$$\begin{aligned} \hat{\mathbf{x}}_n &= \frac{\sigma_n}{\sigma_{n+1}} \hat{\mathbf{x}}_{n+1} + \frac{\sigma_n}{\sigma_T} \Phi_{-h} \left(\gamma_{n+1}, \frac{\sigma_T}{\sigma_{n+1}} \mathbf{x}_{n+1} \right), \\ \mathbf{x}_n &= \frac{\sigma_n}{\sigma_{n+1}} \zeta^{-1} \mathbf{x}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{x}}_n - \frac{\sigma_n}{\sigma_T} \zeta^{-1} \Phi_h \left(\gamma_n, \frac{\sigma_T}{\sigma_n} \hat{\mathbf{x}}_n \right), \end{aligned} \quad (76)$$

1665

1674 Next we simplify the explicit RK scheme $\Phi(\gamma_n, \mathbf{y}_n)$ for the time-changed probability flow ODE in
 1675 Equation (70). Recall that the RK scheme can be written as
 1676

$$1677 \quad \mathbf{z}_i = \mathbf{y}_n + h \sum_{j=1}^{i-1} a_{ij} \sigma_T \mathbf{x}_{0|\gamma_n+c_j h} \left(\frac{\sigma_{\gamma_n+c_j h}}{\sigma_T} \mathbf{z}_j \right), \quad (77)$$

$$1680 \quad \Phi_h(\gamma_n, \mathbf{y}_n) = h \sum_{i=1}^s b_i \sigma_T \mathbf{x}_{0|\gamma_n+c_i h} \left(\frac{\sigma_{\gamma_n+c_i h}}{\sigma_T} \mathbf{z}_i \right).$$

1683 Next, we replace \mathbf{y}_t back with \mathbf{x}_t which yields

$$1684 \quad \mathbf{z}_i = \sigma_T \left(\frac{1}{\sigma_n} \mathbf{x}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{x}_{0|\gamma_n+c_j h} \left(\frac{\sigma_{\gamma_n+c_j h}}{\sigma_T} \mathbf{z}_j \right) \right), \quad (78)$$

$$1688 \quad \Phi_h \left(\gamma_n, \frac{\sigma_T}{\sigma_n} \mathbf{x}_n \right) = \sigma_T h \sum_{i=1}^s b_i \mathbf{x}_{0|\gamma_n+c_i h} \left(\frac{\sigma_{\gamma_n+c_i h}}{\sigma_T} \mathbf{z}_i \right).$$

1691 To further simplify let $\sigma_T \hat{\mathbf{z}}_i = \mathbf{z}_i$ and define $\Psi_h(\gamma_n, \mathbf{x}_n) := \sigma_T \Phi(\gamma_n, \frac{\sigma_T}{\sigma_n} \mathbf{x}_n)$.

1692 Thus we can write the following reversible scheme with forward step

$$1694 \quad \mathbf{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} (\zeta \mathbf{x}_n + (1 - \zeta) \hat{\mathbf{x}}_n) + \sigma_{n+1} \Psi_h(\gamma_n, \hat{\mathbf{x}}_n), \quad (79)$$

$$1696 \quad \hat{\mathbf{x}}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} \hat{\mathbf{x}}_n - \sigma_{n+1} \Psi_{-h}(\gamma_{n+1}, \mathbf{x}_{n+1}),$$

1698 and the backward step

$$1700 \quad \hat{\mathbf{x}}_n = \frac{\sigma_n}{\sigma_{n+1}} \hat{\mathbf{x}}_{n+1} + \sigma_n \Psi_{-h}(\gamma_{n+1}, \mathbf{x}_{n+1}), \quad (80)$$

$$1702 \quad \mathbf{x}_n = \frac{\sigma_n}{\sigma_{n+1}} \zeta^{-1} \mathbf{x}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{x}}_n - \sigma_n \zeta^{-1} \Psi_h(\gamma_n, \hat{\mathbf{x}}_n),$$

1704 with the numerical scheme

$$1705 \quad \hat{\mathbf{z}}_i = \frac{1}{\sigma_n} \mathbf{x}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{x}_{0|\gamma_n+c_j h}^\theta (\sigma_{\gamma_n+c_j h} \hat{\mathbf{z}}_j), \quad (81)$$

$$1709 \quad \Psi_h(\gamma_n, \mathbf{x}_n) = h \sum_{i=1}^s b_i \mathbf{x}_{0|\gamma_n+c_i h}^\theta (\sigma_{\gamma_n+c_i h} \hat{\mathbf{z}}_i).$$

1711 \square

1713 NOISE PREDICTION

1714 We present this derivation in the form of Lemma C.2 below.

1716 **Lemma C.2** (Rex (ODE) for noise prediction models). *Let Φ be an explicit Runge-Kutta solver for
 1717 the ODE in Equation (69) with Butcher tableau a_{ij} , b_i , c_i . The reversible solver for Φ in terms of the
 1718 original state \mathbf{x}_t is given by the forward step*

$$1720 \quad \mathbf{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n} (\zeta \mathbf{x}_n + (1 - \zeta) \hat{\mathbf{x}}_n) + \alpha_{n+1} \Psi_h(\chi_n, \hat{\mathbf{x}}_n), \quad (82)$$

$$1722 \quad \hat{\mathbf{x}}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n} \hat{\mathbf{x}}_n - \alpha_{n+1} \Psi_{-h}(\chi_{n+1}, \mathbf{x}_{n+1}),$$

1723 and backward step

$$1725 \quad \hat{\mathbf{x}}_n = \frac{\alpha_n}{\alpha_{n+1}} \hat{\mathbf{x}}_{n+1} + \alpha_n \Psi_{-h}(\chi_{n+1}, \mathbf{x}_{n+1}), \quad (83)$$

$$1726 \quad \mathbf{x}_n = \frac{\alpha_n}{\alpha_{n+1}} \zeta^{-1} \mathbf{x}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{x}}_n - \alpha_n \zeta^{-1} \Psi_h(\chi_n, \hat{\mathbf{x}}_n),$$

1728 with step size $h := \chi_{n+1} - \chi_n$ and where Ψ denotes the following scheme
1729

$$1730 \quad \hat{z}_i = \frac{1}{\alpha_n} \mathbf{x}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{x}_{T|\chi_n+c_j h}^\theta (\alpha_{\chi_n+c_j h} \hat{z}_j), \quad (84)$$

$$1733 \quad \Psi_h(\chi_n, \mathbf{x}_n) = h \sum_{i=1}^s b_i \mathbf{x}_{T|\chi_n+c_i h}^\theta (\alpha_{\chi_n+c_i h} \hat{z}_i),$$

1736 *Proof.* Recall that the forward step of the McCallum-Foster method for Equation (69) given Φ is
1737 given as

$$1738 \quad \begin{aligned} z_{n+1} &= \zeta z_n + (1 - \zeta) \hat{z}_n + \Phi_h(\chi_n, \hat{z}_n), \\ 1739 \quad \hat{z}_{n+1} &= \hat{z}_n - \Phi_{-h}(\chi_{n+1}, z_{n+1}), \end{aligned} \quad (85)$$

1741 with step size $h = \chi_{n+1} - \chi_n$. We use the definition of $\mathbf{z}_t = \frac{\alpha_T}{\alpha_t} \mathbf{x}_t$ to rewrite the forward pass as

$$1742 \quad \begin{aligned} 1743 \quad \mathbf{x}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} (\zeta \mathbf{x}_n + (1 - \zeta) \hat{\mathbf{x}}_n) + \frac{\alpha_{n+1}}{\alpha_T} \Phi_h \left(\chi_n, \frac{\alpha_T}{\alpha_n} \hat{\mathbf{x}}_n \right), \\ 1744 \quad \hat{\mathbf{x}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} \hat{\mathbf{x}}_n - \frac{\alpha_{n+1}}{\alpha_T} \Phi_{-h} \left(\chi_{n+1}, \frac{\alpha_T}{\alpha_{n+1}} \mathbf{x}_{n+1} \right). \end{aligned} \quad (86)$$

1747 *Mutatis mutandis* we find the backward step in \mathbf{x}_t to be given as

$$1748 \quad \begin{aligned} 1749 \quad \hat{\mathbf{x}}_n &= \frac{\alpha_n}{\alpha_{n+1}} \hat{\mathbf{x}}_{n+1} + \frac{\alpha_n}{\alpha_T} \Phi_{-h} \left(\chi_{n+1}, \frac{\alpha_T}{\alpha_{n+1}} \mathbf{x}_{n+1} \right), \\ 1750 \quad \mathbf{x}_n &= \frac{\alpha_n}{\alpha_{n+1}} \zeta^{-1} \mathbf{x}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{x}}_n - \frac{\alpha_n}{\alpha_T} \zeta^{-1} \Phi_h \left(\chi_n, \frac{\alpha_T}{\alpha_n} \hat{\mathbf{x}}_n \right), \end{aligned} \quad (87)$$

1754 Next we simplify the explicit RK scheme $\Phi(\chi_n, z_n)$ for the time-changed probability flow ODE in
1755 Equation (70). Recall that the RK scheme can be written as

$$1756 \quad \begin{aligned} 1757 \quad \mathbf{z}_i &= \mathbf{z}_n + h \sum_{j=1}^{i-1} a_{ij} \alpha_T \mathbf{x}_{0|\chi_n+c_j h} \left(\frac{\alpha_{\chi_n+c_j h}}{\alpha_T} \mathbf{z}_j \right), \\ 1758 \quad \Phi_h(\chi_n, \mathbf{z}_n) &= h \sum_{i=1}^s b_i \alpha_T \mathbf{x}_{0|\chi_n+c_i h} \left(\frac{\alpha_{\chi_n+c_i h}}{\alpha_T} \mathbf{z}_i \right). \end{aligned} \quad (88)$$

1762 Next, we replace \mathbf{z}_t back with \mathbf{x}_t which yields

$$1763 \quad \begin{aligned} 1764 \quad \mathbf{z}_i &= \alpha_T \left(\frac{1}{\alpha_n} \mathbf{x}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{x}_{0|\chi_n+c_j h} \left(\frac{\alpha_{\chi_n+c_j h}}{\alpha_T} \mathbf{z}_j \right) \right), \\ 1765 \quad \Phi_h \left(\chi_n, \frac{\alpha_T}{\alpha_n} \mathbf{x}_n \right) &= \alpha_T h \sum_{i=1}^s b_i \mathbf{x}_{0|\chi_n+c_i h} \left(\frac{\alpha_{\chi_n+c_i h}}{\alpha_T} \mathbf{z}_i \right). \end{aligned} \quad (89)$$

1770 To further simplify let $\alpha_T \hat{z}_i = \mathbf{z}_i$ and define $\Psi_h(\chi_n, \mathbf{x}_n) := \alpha_T \Phi(\chi_n, \frac{\alpha_T}{\alpha_n} \mathbf{x}_n)$.
1771

1772 Thus we can write the following reversible scheme with forward step

$$1773 \quad \begin{aligned} 1774 \quad \mathbf{x}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} (\zeta \mathbf{x}_n + (1 - \zeta) \hat{\mathbf{x}}_n) + \alpha_{n+1} \Psi_h(\chi_n, \hat{\mathbf{x}}_n), \\ 1775 \quad \hat{\mathbf{x}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} \hat{\mathbf{x}}_n - \alpha_{n+1} \Psi_{-h}(\chi_{n+1}, \mathbf{x}_{n+1}), \end{aligned} \quad (90)$$

1777 and the backward step

$$1778 \quad \begin{aligned} 1779 \quad \hat{\mathbf{x}}_n &= \frac{\alpha_n}{\alpha_{n+1}} \hat{\mathbf{x}}_{n+1} + \alpha_n \Psi_{-h}(\chi_{n+1}, \mathbf{x}_{n+1}), \\ 1780 \quad \mathbf{x}_n &= \frac{\alpha_n}{\alpha_{n+1}} \zeta^{-1} \mathbf{x}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{x}}_n - \alpha_n \zeta^{-1} \Psi_h(\chi_n, \hat{\mathbf{x}}_n), \end{aligned} \quad (91)$$

1782 with the numerical scheme
 1783

$$1784 \quad \hat{\mathbf{z}}_i = \frac{1}{\alpha_n} \mathbf{x}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{x}_T^{\theta} |_{\chi_n + c_j h} (\alpha_{\chi_n + c_j h} \hat{\mathbf{z}}_j), \quad (92)$$

$$1787 \quad \Psi_h(\chi_n, \mathbf{x}_n) = h \sum_{i=1}^s b_i \mathbf{x}_T^{\theta} |_{\chi_n + c_i h} (\alpha_{\chi_n + c_i h} \hat{\mathbf{z}}_i).$$

□

1792 C.2 REX (SDE)

1793 In this section we derive the Rex scheme for the reverse-time diffusion SDE along with several helper
 1794 derivations. We begin by deriving the reparameterization of Equation (9) in Section C.2.2 and then
 1795 performing an analogous derivation for the noise prediction scenario in Section C.2.3.

1797 C.2.1 TIME-CHANGED BROWNIAN MOTION

1799 Before detailing this proof we first review some necessary preliminary results about continuous local
 1800 martingales and Brownian motion. In particular we will show that we can simplify the stochastic
 1801 integrals in Equation (9) and the corresponding reparameterization with noise prediction models.

1802 **Dambis-Dubins-Schwarz representation theorem.** We restate the Dambis-Dubins-Schwarz rep-
 1803 resentation theorem (Dubins & Schwarz, 1965) which shows that continuous local martingales can
 1804 be represented as time-changed Brownian motions.

1805 **Theorem C.3** (Dambis-Dubins-Schwarz representation theorem). *Let M be a continuous local
 1806 martingale adapted to a filtration $\{\mathcal{F}_t\}_{t \geq 0}$ beginning at 0 (i.e., $M_0 = 0$) such that $\langle M \rangle_\infty = \infty$
 1807 almost surely. Define the random variables $\{\tau_t\}_{t \geq 0}$ by*

$$1809 \quad \tau_t = \inf \{s \geq 0 : \langle M \rangle_s > t\} = \sup \{s \geq 0 : \langle M \rangle_s = t\}. \quad (93)$$

1810 Then for any given t the random variable τ_t is an almost surely finite stopping time, and the process¹⁹
 1811 $B_t = M_{\tau_t}$ is a Brownian motion w.r.t. the filtration $\{\mathcal{G}_t\}_{t \geq 0} = \{\mathcal{F}_{\tau_t}\}_{t \geq 0}$. Moreover,

$$1813 \quad M_t = B_{\langle M \rangle_t}. \quad (94)$$

1815 **A multi-dimensional version of the Dambis-Dubins-Schwarz representation theorem.** In our
 1816 work we are interested in a d -dimensional local martingale $M := (M^1, \dots, M^d)$. As such we discuss
 1817 a multi-dimensional extension of Theorem C.3 which requires that the d -dimensional continuous
 1818 local martingale if the quadratic (covariation) matrix $\langle M \rangle_t^{ij} = \langle M^i, M^j \rangle_t$ is proportional to the
 1819 identity matrix. We adapt the following theorem from Lowther (2010, Theorem 2) and Bourgade
 1820 (2010, Theorem 4.13) (cf. Revuz & Yor, 2013).

1821 **Theorem C.4** (Multi-dimensional Dambis-Dubins-Schwarz representation theorem). *Let $M =$
 1822 (M^1, \dots, M^d) be a collection of continuous local martingales with $M_0 = \mathbf{0}$ such that for any
 1823 $1 \leq i \leq d$, $\langle M \rangle_\infty^{ii} = \infty$ almost surely. Suppose, furthermore, that $\langle M^i, M^j \rangle_t = \delta_{ij} A_t$, where δ
 1824 denotes the Kronecker delta, for some process A and all $1 \leq i, j \leq d$ and $t \geq 0$. Then there is a
 1825 d -dimensional Brownian motion 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
 1826 a \mathcal{G} -stopping time and*

$$1827 \quad M_t = B_{A_t}. \quad (95)$$

1828 **Enlargement of the probability space.** Recall that in Theorems C.3 and C.4 we stated that
 1829 quadratic variation of the continuous local martingale needed to tend towards infinity as $t \rightarrow \infty$.
 1830 What when $\langle M \rangle_\infty$ has a nonzero probability of being finite? It can be shown that Theorems C.3
 1831 and C.4 holds under an enlargement of the probability space (not the filtration). Consider both
 1832 our original probability space (Ω, \mathcal{F}, P) and another probability space $(\Omega', \mathcal{F}', P')$ along with a
 1833 measurable surjection $f : \Omega' \rightarrow \Omega$ preserving probabilities such that $P(A) = P'(f^{-1}(A))$ for all
 1834 $A \in \mathcal{F}$, i.e., $f_* P'$ is a pushforward measure. Thus any process on the original probability space

1835 ¹⁹Defined up to a null set.

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 $f' : \Omega \rightarrow \Omega$ is just the projection onto Ω .

We now present a lemma for rewriting the stochastic differential in Equation (9) using the Dambis-Dubins-Schwarz representation theorem. Recall that in Equation (9) we denote the reverse-time d -dimensional Brownian motion as $\bar{\mathbf{W}}_t$, *i.e.*, by Lévy's characterization we have $\bar{\mathbf{W}}_T = \mathbf{0}$ and

$$\bar{\mathbf{W}}_t - \bar{\mathbf{W}}_s \sim -\mathcal{N}(\mathbf{0}, (t-s)\mathbf{I}) = \mathcal{N}(\mathbf{0}, (t-s)\mathbf{I}), \quad (96)$$

for $0 \leq t < s \leq T$. With this in mind we present Lemma C.5 below.

Lemma C.5. *The stochastic differential $\sqrt{-\frac{d\varrho_t}{dt}} d\bar{\mathbf{W}}_t$ can be rewritten as a time-changed Brownian motion of the form*

$$\sqrt{-\frac{d\varrho_t}{dt}} d\bar{\mathbf{W}}_t = d\mathbf{W}_\varrho, \quad (97)$$

where $\varrho_t = \gamma_t^2$.

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

$$\mathbf{M}_t := \int_T^t \sqrt{-\frac{d\varrho}{dt}} d\bar{\mathbf{W}}_s. \quad (98)$$

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 \mathbf{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

$$\bar{\mathbf{W}}_t = \mathbf{W}_T - \mathbf{W}_t \quad (99)$$

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

$$d\mathbf{f}(t, \mathbf{W}_t) = \partial_t \mathbf{f}(t, \mathbf{W}_t) dt + \sum_{i=1}^d \partial_{\mathbf{x}_i} \mathbf{f}(t, \mathbf{W}_t) d\mathbf{W}_t^i + \sum_{i,j=1}^d \partial_{\mathbf{x}_i, \mathbf{x}_j} \mathbf{f}(t, \mathbf{W}_t) d\langle \mathbf{W}^i, \mathbf{W}^j \rangle_t, \quad (100)$$

which simplifies to

$$d\mathbf{f}(t, \mathbf{W}_t) = d\bar{\mathbf{W}}_t = -d\mathbf{W}_t. \quad (101)$$

Thus we can rewrite Equation (98) as

$$\mathbf{M}_t = - \int_T^t \sqrt{-\frac{d\varrho}{dt}} d\mathbf{W}_s. \quad (102)$$

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

$$\langle \mathbf{M} \rangle_t^{ii} = A_t = - \int_T^t \left(\sqrt{-\frac{d\varrho_\tau}{d\tau}} \right)^2 d\tau, \quad (103)$$

$$= \int_T^t \frac{d\varrho_\tau}{d\tau} d\tau, \quad (104)$$

$$= \varrho_t - \varrho_T = \frac{\alpha_t^2}{\sigma_t^2} - \frac{\alpha_T^2}{\sigma_T^2}. \quad (105)$$

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 $\langle \mathbf{M} \rangle_\infty^{ii} = \infty$ almost surely and as such we may need to enlarge the underlying probability space. Our constructed martingale can be

1890 expressed as time-changed Brownian motion, see Theorem C.4, such that $M_t = \bar{W}_{A_t}$ were \bar{W}_ϱ is
 1891 the standard d -dimensional Brownian motion with time variable ϱ .

1892 Now we can rewrite Equation (102) in differential form as

$$1894 \quad dM_t = d\bar{W}_{A_t}. \quad (106)$$

1895 Because Brownian motion is time-shift invariant we can then write

$$1897 \quad dM_t = d\bar{W}_{\varrho_t}. \quad (107)$$

1900 **Remark C.3.** Lemma C.5 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
 1901 tackle the problem via the Dambis-Dubins-Schwarz theorem.

1903 **Remark C.4.** Under the common scenario where $\sigma_0 = 0$ then we have that $\langle M \rangle_\infty^{ii} = \infty$ almost
 1904 surely.

1905 **Lemma C.6.** *Let $\alpha_T > 0$. Then the stochastic differential $\sqrt{\frac{d}{dt}(\chi_t^2)} d\bar{W}_t$ can be rewritten as a
 1906 time-changed Brownian motion of the form*

$$1908 \quad \sqrt{\frac{d}{dt}(\chi_t^2)} d\bar{W}_t = d\bar{W}_{\chi^2}, \quad (108)$$

1910 where $\chi_t = \frac{\sigma_t}{\alpha_t}$.

1912 *Proof.* To simplify the stochastic integral term we first define a continuous local martingale M_t via
 1913 the stochastic integral

$$1915 \quad M_t := \int_T^t \sqrt{\frac{d}{dt}(\chi_\tau^2)} d\bar{W}_\tau. \quad (109)$$

1917 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, $M_T = 0$
 1918 by construction. Second, since the integral consists of scalar noise we have that $\langle M^i, M^j \rangle_t = 0$ for
 1919 all $i \neq j$. Thus, the quadratic variation of $\langle M_t \rangle^{ii}$ for each i is found to be

$$1921 \quad \langle M \rangle_t^{ii} = A_t = \int_T^t \left(\sqrt{\frac{d}{d\tau}(\chi_\tau^2)} \right)^2 d\tau, \quad (110)$$

$$1924 \quad = \int_T^t \frac{d}{d\tau}(\chi_\tau^2) d\tau, \quad (111)$$

$$1926 \quad = \chi_t^2 - \chi_T^2 = \frac{\sigma_t^2}{\alpha_t^2} - \frac{\sigma_T^2}{\alpha_T^2}. \quad (112)$$

1928 Now we have a deterministic mapping from the original time to our new time via A_t . Now in general
 1929 for any valid choice of (α_t, σ_t) we don't necessarily have that $\langle M \rangle_\infty^{ii} = \infty$ almost surely and as
 1930 such we may need to enlarge the underlying probability space. Our constructed martingale can be
 1931 expressed as time-changed Brownian motion, see Theorem C.4, such that $M_t = \bar{W}_{A_t}$ were \bar{W}_{χ^2} is
 1932 the standard d -dimensional Brownian motion with time variable χ^2 in reverse-time.

1933 Now we can rewrite Equation (98) in differential form as

$$1935 \quad dM_t = d\bar{W}_{A_t}. \quad (113)$$

1936 Because Brownian motion is time-shift invariant we can then write

$$1938 \quad dM_t = d\bar{W}_{\chi_t^2}. \quad (114)$$

1940 **Remark C.5.** The constraint of $\alpha_T > 0$ is important to ensure that χ_T is finite which is necessary
 1941 due

$$1943 \quad \bar{W}_{\chi_t^2} = \bar{W}_{\chi_T^2} - \bar{W}_{\chi_t^2}. \quad (115)$$

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

1944 C.2.2 PROOF OF PROPOSITION 3.2
19451946 In this section we provide the proof for Proposition 3.2 along with associated derivations. We restate
1947 Proposition 3.2 below.1948 **Proposition 3.2** (Time reparameterization of the reverse-time diffusion SDE). *The reverse-time SDE
1949 in Equation (9) can be rewritten in terms of the data prediction model as*

1950
1951
$$d\mathbf{Y}_\varrho = \frac{\sigma_T}{\gamma_T} \mathbf{x}_{0|\varrho}^\theta \left(\frac{\gamma_T \sigma_\varrho}{\sigma_T \gamma_\varrho} \mathbf{Y}_\varrho \right) d\varrho + \frac{\sigma_T}{\gamma_T} d\mathbf{W}_\varrho, \quad (10)$$

1952

1953 where $\mathbf{Y}_t = \frac{\sigma_T^2 \alpha_t}{\sigma_t^2 \alpha_T} \mathbf{X}_t$ and $\varrho_t := \frac{\alpha_t^2}{\sigma_t^2}$.
19541956 *Proof.* We rewrite Equation (3) in terms of the data prediction model, using the identity
1957

1958
1959
$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = -\frac{1}{\sigma_t^2} \mathbf{x} + \frac{\alpha_t}{\sigma_t^2} \mathbf{x}_{0|t}(\mathbf{x}), \quad (116)$$

1960

1961 to find

1962
1963
$$d\mathbf{X}_t = \left[\underbrace{\left(f(t) + \frac{g^2(t)}{\sigma_t^2} \right) \mathbf{X}_t}_{=a(t)} + \underbrace{\left(-\frac{\alpha_t g^2(t)}{\sigma_t^2} \right) \mathbf{x}_{0|t}(\mathbf{X}_t)}_{=b(t)} \right] dt + g(t) d\bar{\mathbf{W}}_t, \quad (117)$$

1964
1965
1966

1967 where

1968
1969
$$f(t) = \frac{\dot{\alpha}_t}{\alpha_t}, \quad g^2(t) = \dot{\sigma}_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t} \sigma_t^2 = -2\sigma_t^2 \frac{d \log \gamma_t}{dt}. \quad (118)$$

1970

1971 Next we find the integrating factor $\Xi_t = \exp - \int_T^t a(u) du$,
1972

1973
1974
$$\Xi_t = \exp \left(\int_t^T \frac{d \log \alpha_u}{du} + \frac{g^2(u)}{\sigma_u^2} du \right), \quad (119)$$

1975

1976
1977
$$= \exp \left(\int_t^T \frac{d \log \alpha_u}{du} - 2 \frac{d \log \gamma_u}{du} du \right), \quad (120)$$

1978

1979
1980
$$= \exp \left(\int_t^T \frac{d \log \alpha_u}{du} - 2 \left[\frac{d \log \alpha_u}{du} - \frac{d \log \sigma_u}{du} \right] du \right), \quad (121)$$

1981

1982
1983
$$= \exp \left(\int_t^T \frac{d \log \sigma_u^2}{du} - \frac{d \log \alpha_u}{du} du \right), \quad (122)$$

1984

1985
$$= \exp (\log \sigma_T^2 - \log \sigma_t^2 - (\log \alpha_T - \log \alpha_t)), \quad (123)$$

1986
1987
$$= \frac{\sigma_T^2 \alpha_t}{\sigma_t^2 \alpha_T}. \quad (124)$$

1988 We can write the integrating factor in terms of γ_t as
1989

1990
1991
$$\Xi_t = \frac{\sigma_T \gamma_t}{\sigma_t \gamma_T}. \quad (125)$$

1992 Moreover we can further simplify $b(t)$ as
1993

1994
1995
$$b(t) = \frac{-\alpha_t g^2(t)}{\sigma_t^2}, \quad (126)$$

1996

1997
$$= 2\alpha_t \frac{d \log \gamma_t}{dt}. \quad (127)$$

1998 Thus we can rewrite the SDE in Equation (117) as
 1999

2000
$$d \left[\frac{\sigma_T \gamma_t}{\gamma_T \sigma_t} \mathbf{X}_t \right] = 2 \frac{\sigma_T}{\gamma_T} \frac{\alpha_t}{\sigma_t} \gamma_t \frac{d \log \gamma_t}{dt} \mathbf{x}_{0|t}(\mathbf{X}_t) dt + \frac{\sigma_T}{\gamma_T} \frac{\gamma_t}{\sigma_t} \sqrt{-2\sigma_t^2 \frac{d \log \gamma_t}{dt}} d\bar{\mathbf{W}}_t, \quad (128)$$

2003
$$d\mathbf{Y}_t \stackrel{(i)}{=} 2 \frac{\sigma_T}{\gamma_T} \frac{\alpha_t}{\sigma_t} \gamma_t \frac{d \log \gamma_t}{dt} \mathbf{x}_{0|t} \left(\frac{\gamma_T \sigma_t}{\sigma_T \gamma_t} \mathbf{Y}_t \right) dt + \frac{\sigma_T}{\gamma_T} \frac{\gamma_t}{\sigma_t} \sqrt{-2\sigma_t^2 \frac{d \log \gamma_t}{dt}} d\bar{\mathbf{W}}_t, \quad (129)$$

2005
$$d\mathbf{Y}_t = \frac{\sigma_T}{\gamma_T} \frac{d\gamma_t^2}{dt} \mathbf{x}_{0|t} \left(\frac{\gamma_T \sigma_t}{\sigma_T \gamma_t} \mathbf{Y}_t \right) dt + \frac{\sigma_T}{\gamma_T} \sqrt{-\gamma_t^2 \frac{d \log \gamma_t^2}{dt}} d\bar{\mathbf{W}}_t, \quad (130)$$

2008
$$d\mathbf{Y}_t = \frac{\sigma_T}{\gamma_T} \frac{d\gamma_t^2}{dt} \mathbf{x}_{0|t} \left(\frac{\gamma_T \sigma_t}{\sigma_T \gamma_t} \mathbf{Y}_t \right) dt + \frac{\sigma_T}{\gamma_T} \sqrt{-\frac{d\gamma_t^2}{dt}} d\bar{\mathbf{W}}_t, \quad (131)$$

2011
$$d\mathbf{Y}_\varrho \stackrel{(ii)}{=} \frac{\sigma_T}{\gamma_T} \mathbf{x}_{0|\varrho} \left(\frac{\gamma_T \sigma_\varrho}{\sigma_T \gamma_\varrho} \mathbf{Y}_\varrho \right) d\varrho + \frac{\sigma_T}{\gamma_T} d\mathbf{W}_\varrho, \quad (132)$$

2013 where (i) holds by the change-of-variables $\mathbf{Y}_t = \frac{\sigma_T \gamma_t}{\gamma_T \sigma_t} \mathbf{X}_t$ and (ii) holds by Lemma C.5. \square
 2014

2015 C.2.3 PROOF OF REPARAMETERIZED SDE FOR NOISE PREDICTION MODELS

2018 **Proposition C.7** (Time reparameterization of the reverse-time diffusion SDE for noise prediction
 2019 models). *The reverse-time SDE in Equation (3) can be rewritten in terms of the noise prediction
 2020 model as*

2021
$$d\mathbf{Y}_\chi = 2\alpha_T \mathbf{x}_{T|\chi}^\theta \left(\frac{\alpha_\chi}{\alpha_T} \mathbf{Y}_\chi \right) d\chi + \alpha_T d\bar{\mathbf{W}}_{\chi^2}, \quad (133)$$

2023 where $\mathbf{Y}_t = \frac{\alpha_t}{\alpha_T} \mathbf{X}_t$ and $\chi_t := \frac{\sigma_t}{\alpha_t}$.

2026 *Proof.* We rewrite Equation (3) in terms of the noise prediction model to find

2028
$$d\mathbf{X}_t = \left[f(t) \mathbf{X}_t + \frac{g^2(t)}{\sigma_t} \mathbf{x}_{T|t}^\theta(\mathbf{X}_t) \right] dt + g(t) d\bar{\mathbf{W}}_t, \quad (134)$$

2030 where

2032
$$f(t) = \frac{\dot{\alpha}_t}{\alpha_t}, \quad g^2(t) = \dot{\sigma}_t^2 - 2\frac{\dot{\alpha}_t}{\alpha_t} \sigma_t^2 = -2\sigma_t^2 \frac{d \log \gamma_t}{dt}. \quad (135)$$

2034 Next we find the integrating factor to be $\exp - \int_T^t f(u) du = \frac{\alpha_T}{\alpha_t}$. Moreover, we can further simplify
 2035 $\frac{g^2(t)}{\sigma_t}$ as
 2036

2038
$$\frac{g^2(t)}{\sigma_t} = -2\sigma_t \frac{d \log \gamma_t}{dt}, \quad (136)$$

2040
$$= -2\sigma_t \frac{\dot{\gamma}_t}{\gamma_t}, \quad (137)$$

2042
$$= -2 \frac{\sigma_t}{\gamma_t} \frac{\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t}{\sigma_t^2}, \quad (138)$$

2044
$$= -2 \frac{\sigma_t^2}{\alpha_t} \frac{\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t}{\sigma_t^2}, \quad (139)$$

2046
$$= 2 \frac{\sigma_t^2}{\alpha_t} \frac{\alpha_t \dot{\sigma}_t - \dot{\alpha}_t \sigma_t}{\sigma_t^2}, \quad (140)$$

2048
$$= 2 \frac{\alpha_t \dot{\sigma}_t - \dot{\alpha}_t \sigma_t}{\alpha_t}, \quad (141)$$

2050
$$= 2 \frac{\alpha_t \dot{\sigma}_t - \dot{\alpha}_t \sigma_t}{\alpha_t}, \quad (142)$$

2052 Let $\chi_t := \frac{\sigma_t}{\alpha_t} = \frac{1}{\gamma_t}$. Thus we can rewrite the SDE in Equation (134) as
2053

$$2054 \quad d \left[\frac{\alpha_T}{\alpha_t} \mathbf{X}_t \right] = \frac{\alpha_T}{\alpha_t} \frac{g^2(t)}{\sigma_t^2} \mathbf{x}_{T|t}^\theta(\mathbf{X}_t) dt + \frac{\alpha_T}{\alpha_t} \sqrt{-2\sigma_t^2 \frac{d \log \gamma_t}{dt}} d\bar{\mathbf{W}}_t, \quad (143)$$

$$2055 \quad d\mathbf{Y}_t \stackrel{(i)}{=} \frac{\alpha_T}{\alpha_t} \frac{g^2(t)}{\sigma_t^2} \mathbf{x}_{T|t}^\theta \left(\frac{\alpha_t}{\alpha_T} \mathbf{Y}_t \right) dt + \frac{\alpha_T}{\alpha_t} \sqrt{-2\sigma_t^2 \frac{d \log \gamma_t}{dt}} d\bar{\mathbf{W}}_t, \quad (144)$$

$$2056 \quad d\mathbf{Y}_t = 2\alpha_T \frac{\alpha_t \dot{\sigma}_t - \dot{\alpha}_t \sigma_t}{\alpha_t^2} \mathbf{x}_{T|t}^\theta \left(\frac{\alpha_t}{\alpha_T} \mathbf{Y}_t \right) dt + \frac{\alpha_T}{\alpha_t} \sqrt{-2\sigma_t^2 \frac{d \log \gamma_t}{dt}} d\bar{\mathbf{W}}_t, \quad (145)$$

$$2057 \quad d\mathbf{Y}_t \stackrel{(ii)}{=} 2\alpha_T \dot{\chi}_t \mathbf{x}_{T|t}^\theta \left(\frac{\alpha_t}{\alpha_T} \mathbf{Y}_t \right) dt + \alpha_T \sqrt{-2 \frac{\sigma_t^2}{\alpha_t^2} \frac{d \log \gamma_t}{dt}} d\bar{\mathbf{W}}_t, \quad (146)$$

$$2058 \quad d\mathbf{Y}_t = 2\alpha_T \dot{\chi}_t \mathbf{x}_{T|t}^\theta \left(\frac{\alpha_t}{\alpha_T} \mathbf{Y}_t \right) dt + \alpha_T \sqrt{\dot{\chi}_t^2} d\bar{\mathbf{W}}_t, \quad (147)$$

$$2059 \quad d\mathbf{Y}_\chi \stackrel{(iii)}{=} 2\alpha_T \mathbf{x}_{T|\chi}^\theta \left(\frac{\alpha_\chi}{\alpha_T} \mathbf{Y}_\chi \right) d\chi + \alpha_T d\bar{\mathbf{W}}_{\chi^2}, \quad (148)$$

$$2060 \quad (149)$$

2070 where (i) holds by the change-of-variables $\mathbf{Y}_t = \frac{\alpha_T}{\alpha_t} \mathbf{X}_t$, (ii) holds by
2071

$$2072 \quad -2 \frac{\sigma_t^2}{\alpha_t^2} \frac{d \log \gamma_t}{dt} = \frac{\sigma_t^2}{\alpha_t^2} \frac{d(-2 \log \gamma_t)}{dt}, \quad (150)$$

$$2073 \quad = \frac{\sigma_t^2}{\alpha_t^2} \frac{d \log \chi_t^2}{dt}, \quad (151)$$

$$2074 \quad = \frac{\sigma_t^2}{\alpha_t^2} \frac{\dot{\chi}_t^2}{\chi_t^2}, \quad (152)$$

$$2075 \quad = \dot{\chi}_t^2, \quad (153)$$

2076 and (iii) holds by Lemma C.5 *mutatis mutandis* for χ_t . \square
2077

2078 C.2.4 DERIVATION OF REX (SDE)

2079 We present derivations for both the data prediction and noise prediction formulations.
2080

2081 **Data prediction.** We present this derivation in the form of Lemma C.8 below.
2082

2083 **Lemma C.8** (Rex (SDE) for data prediction models). *Let Φ be an explicit stochastic Runge-Kutta
2084 solver for the additive noise SDE in Equation (10), we construct the following reversible scheme for
2085 diffusion models*

$$2086 \quad \mathbf{X}_{n+1} = \frac{\sigma_{n+1} \gamma_n}{\gamma_{n+1} \sigma_n} (\zeta \mathbf{X}_n + (1 - \zeta) \hat{\mathbf{X}}_n) + \frac{\sigma_{n+1}}{\gamma_{n+1}} \Psi_h(\varrho_n, \hat{\mathbf{X}}_n, \mathbf{W}_\varrho(\omega)), \\ 2087 \quad \hat{\mathbf{X}}_{n+1} = \frac{\sigma_{n+1} \gamma_n}{\gamma_{n+1} \sigma_n} \hat{\mathbf{X}}_n - \frac{\sigma_{n+1}}{\gamma_{n+1}} \Psi_{-h}(\varrho_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_\varrho(\omega)), \quad (154)$$

2088 and the backward step is given as
2089

$$2090 \quad \hat{\mathbf{X}}_n = \frac{\sigma_n \gamma_{n+1}}{\gamma_n \sigma_{n+1}} \hat{\mathbf{X}}_n + \frac{\sigma_n}{\gamma_n} \Psi_{-h}(\varrho_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_\varrho(\omega)), \\ 2091 \quad \mathbf{X}_n = \frac{\sigma_n \gamma_{n+1}}{\gamma_n \sigma_{n+1}} \zeta^{-1} \mathbf{X}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{X}}_n - \frac{\sigma_n}{\gamma_n} \zeta^{-1} \Psi_h(\varrho_n, \hat{\mathbf{X}}_n, \mathbf{W}_\varrho(\omega)), \quad (155)$$

2092 with step size $h := \varrho_{n+1} - \varrho_n$ and where Ψ denotes the following scheme
2093

$$2094 \quad \hat{\mathbf{Z}}_i = \frac{\gamma_n}{\sigma_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[a_{ij} \mathbf{x}_{0|\varrho_n+c_j h} \left(\frac{\sigma_{\varrho_n+c_j h}}{\gamma_{\varrho_n+c_j h}} \hat{\mathbf{Z}}_j \right) \right] + a_i^W \mathbf{W}_n + a_i^H \mathbf{H}_n, \\ 2095 \quad \Psi_h(\varrho_n, \mathbf{X}_n, \mathbf{W}_\varrho(\omega)) = h \sum_{j=1}^s \left[b_i \mathbf{x}_{0|\varrho_n+c_i h} \left(\frac{\sigma_{\varrho_n+c_i h}}{\gamma_{\varrho_n+c_i h}} \hat{\mathbf{Z}}_j \right) \right] + b^W \mathbf{W}_n + b^H \mathbf{H}_n. \quad (156)$$

2106 *Proof.* We write the SRK scheme for the time-changed reverse-time SDE in Equation (10) to construct
2107 the following SRK scheme
2108

$$\begin{aligned} 2109 \quad \mathbf{Z}_i &= \mathbf{Y}_n + h \sum_{j=1}^{i-1} \left[a_{ij} \frac{\sigma_T}{\gamma_T} \mathbf{x}_{0|\varrho_n+c_j h} \left(\frac{\gamma_T \sigma_{\varrho_n+c_j h}}{\sigma_T \gamma_{\varrho_n+c_j h}} \mathbf{Z}_j \right) \right] + \frac{\sigma_T}{\gamma_T} (a_i^W \mathbf{W}_n + a_i^H \mathbf{H}_n), \\ 2110 \\ 2111 \quad \mathbf{Y}_{n+1} &= \mathbf{Y}_n + h \sum_{i=1}^s \left[b_i \frac{\sigma_T}{\gamma_T} \mathbf{x}_{0|\varrho_n+c_i h} \left(\frac{\gamma_T \sigma_{\varrho_n+c_i h}}{\sigma_T \gamma_{\varrho_n+c_i h}} \mathbf{Z}_i \right) \right] + \frac{\sigma_T}{\gamma_T} (b^W \mathbf{W}_n + b^H \mathbf{H}_n), \end{aligned} \quad (157)$$

2114 with step size $h = \varrho_{n+1} - \varrho_n$. Next, we replace \mathbf{Y}_t back with \mathbf{X}_t which yields
2115

$$\begin{aligned} 2116 \quad \mathbf{Z}_i &= \frac{\sigma_T}{\gamma_T} \left(\frac{\gamma_n}{\sigma_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[a_{ij} \mathbf{x}_{0|\varrho_n+c_j h} \left(\frac{\gamma_T \sigma_{\varrho_n+c_j h}}{\sigma_T \gamma_{\varrho_n+c_j h}} \mathbf{Z}_j \right) \right] \right) \\ 2117 \\ 2118 &\quad + \frac{\sigma_T}{\gamma_T} (a_i^W \mathbf{W}_n + a_i^H \mathbf{H}_n), \\ 2119 \quad \frac{\sigma_T \gamma_{n+1}}{\gamma_T \sigma_{n+1}} \mathbf{X}_{n+1} &= \frac{\sigma_T \gamma_n}{\gamma_T \sigma_n} \mathbf{X}_n \\ 2120 \\ 2121 &\quad + \frac{\sigma_T}{\gamma_T} h \underbrace{\sum_{i=1}^s \left[b_i \frac{\sigma_T}{\gamma_T} \mathbf{x}_{0|\varrho_n+c_i h} \left(\frac{\gamma_T \sigma_{\varrho_n+c_i h}}{\sigma_T \gamma_{\varrho_n+c_i h}} \mathbf{Z}_i \right) \right]}_{= \Psi_h(\varrho_n, \mathbf{X}_n, \mathbf{W}_\varrho)} + \frac{\sigma_T}{\gamma_T} (b^W \mathbf{W}_n + b^H \mathbf{H}_n). \\ 2122 \\ 2123 \\ 2124 \\ 2125 \\ 2126 \\ 2127 \\ 2128 \end{aligned} \quad (158)$$

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

$$\begin{aligned} 2129 \quad \mathbf{X}_{n+1} &= \frac{\sigma_{n+1} \gamma_n}{\gamma_{n+1} \sigma_n} (\zeta \mathbf{X}_n + (1 - \zeta) \hat{\mathbf{X}}_n) + \frac{\sigma_{n+1}}{\gamma_{n+1}} \Psi_h(\varrho_n, \hat{\mathbf{X}}_n, \mathbf{W}_\varrho(\omega)), \\ 2130 \\ 2131 \quad \hat{\mathbf{X}}_{n+1} &= \frac{\sigma_{n+1} \gamma_n}{\gamma_{n+1} \sigma_n} \hat{\mathbf{X}}_n - \frac{\sigma_{n+1}}{\gamma_{n+1}} \Psi_{-h}(\varrho_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_\varrho(\omega)), \end{aligned} \quad (159)$$

2133 and backward pass
2134

$$\begin{aligned} 2135 \quad \hat{\mathbf{X}}_n &= \frac{\sigma_n \gamma_{n+1}}{\gamma_n \sigma_{n+1}} \hat{\mathbf{X}}_n + \frac{\sigma_n}{\gamma_n} \Psi_{-h}(\varrho_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_\varrho(\omega)), \\ 2136 \\ 2137 \quad \hat{\mathbf{X}}_{n+1} &= \frac{\sigma_n \gamma_{n+1}}{\gamma_n \sigma_{n+1}} \zeta^{-1} \mathbf{X}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{X}}_n - \frac{\sigma_n}{\gamma_n} \zeta^{-1} \Psi_h(\varrho_n, \hat{\mathbf{X}}_n, \mathbf{W}_\varrho(\omega)), \end{aligned} \quad (160)$$

2138 with step size $h := \varrho_{n+1} - \varrho_n$
2139

$$\begin{aligned} 2140 \quad \hat{\mathbf{Z}}_i &= \frac{\gamma_n}{\sigma_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[a_{ij} \mathbf{x}_{0|\varrho_n+c_j h} \left(\frac{\sigma_{\varrho_n+c_j h}}{\gamma_{\varrho_n+c_j h}} \hat{\mathbf{Z}}_j \right) \right] + a_i^W \mathbf{W}_n + a_i^H \mathbf{H}_n, \\ 2141 \\ 2142 \\ 2143 \quad \Psi_h(\varrho_n, \mathbf{X}_n, \mathbf{W}_\varrho(\omega)) &= h \sum_{j=1}^s \left[b_i \mathbf{x}_{0|\varrho_n+c_i h} \left(\frac{\sigma_{\varrho_n+c_i h}}{\gamma_{\varrho_n+c_i h}} \hat{\mathbf{Z}}_j \right) \right] + b^W \mathbf{W}_n + b^H \mathbf{H}_n. \\ 2144 \\ 2145 \\ 2146 \\ 2147 \end{aligned} \quad (161)$$

□

2148 **Noise prediction.** We present this derivation in the form of Lemma C.9 below.

2149 **Lemma C.9** (Rex (SDE) for noise prediction models). *Let Φ be an explicit stochastic Runge-Kutta
2150 solver for the additive noise SDE in Equation (133), we construct the following reversible scheme for
2151 diffusion models*

$$\begin{aligned} 2152 \quad \mathbf{X}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} (\zeta \mathbf{X}_n + (1 - \zeta) \hat{\mathbf{X}}_n) + \alpha_{n+1} \Psi_h(\chi_n, \hat{\mathbf{X}}_n, \mathbf{W}_{\chi^2}(\omega)), \\ 2153 \\ 2154 \quad \hat{\mathbf{X}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} \hat{\mathbf{X}}_n - \alpha_{n+1} \Psi_{-h}(\chi_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_{\chi^2}(\omega)), \end{aligned} \quad (162)$$

2155 and the backward step is given as
2156

$$\begin{aligned} 2157 \quad \hat{\mathbf{X}}_n &= \frac{\alpha_n}{\alpha_{n+1}} \hat{\mathbf{X}}_n + \alpha_n \Psi_{-h}(\chi_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_{\chi^2}(\omega)), \\ 2158 \\ 2159 \quad \mathbf{X}_n &= \frac{\alpha_n}{\alpha_n + 1} \zeta^{-1} \mathbf{X}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{X}}_n - \alpha_n \zeta^{-1} \Psi_h(\chi_n, \hat{\mathbf{X}}_n, \mathbf{W}_{\chi^2}(\omega)), \end{aligned} \quad (163)$$

2160 with step size $h := \chi_{n+1} - \chi_n$ and where Ψ denotes the following scheme
2161

$$\begin{aligned} 2162 \quad \hat{\mathbf{Z}}_i &= \frac{1}{\alpha_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[2a_{ij} \mathbf{x}_{T|\chi_n+c_j h}^\theta \left(\alpha_{\chi_n+c_j h} \hat{\mathbf{Z}}_j \right) \right] + a_i^W \mathbf{W}_n + a_i^H \mathbf{H}_n, \\ 2163 \\ 2164 \quad \Psi_h(\chi_n, \mathbf{X}_n, \mathbf{W}_\chi(\omega)) &= h \sum_{j=1}^s \left[2b_i \mathbf{x}_{T|\chi_n+c_i h}^\theta \left(\alpha_{\chi_n+c_i h} \hat{\mathbf{Z}}_j \right) \right] + b^W \mathbf{W}_n + b^H \mathbf{H}_n. \end{aligned} \quad (164)$$

2168 *Proof.* We write the SRK scheme for the time-changed reverse-time SDE in Equation (133) to
2169 construct the following SRK scheme
2170

$$\begin{aligned} 2171 \quad \mathbf{Z}_i &= \mathbf{Y}_n + h \sum_{j=1}^{i-1} \left[2a_{ij} \alpha_T \mathbf{x}_{T|\chi_n+c_j h} \left(\frac{\alpha_{\chi_n+c_j h}}{\alpha_T} \mathbf{Z}_j \right) \right] + \alpha_T (a_i^W \mathbf{W}_n + a_i^H \mathbf{H}_n), \\ 2172 \\ 2173 \quad \mathbf{Y}_{n+1} &= \mathbf{Y}_n + h \sum_{i=1}^s \left[2b_i \alpha_T \mathbf{x}_{T|\chi_n+c_i h} \left(\frac{\alpha_{\chi_n+c_i h}}{\alpha_T} \mathbf{Z}_i \right) \right] + \alpha_T (b^W \mathbf{W}_n + b^H \mathbf{H}_n), \end{aligned} \quad (165)$$

2177 with step size $h = \chi_{n+1} - \chi_n$. Next, we replace \mathbf{Y}_t back with \mathbf{X}_t which yields
2178

$$\begin{aligned} 2179 \quad \mathbf{Z}_i &= \alpha_T \left(\frac{1}{\alpha_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[2a_{ij} \mathbf{x}_{T|\chi_n+c_j h} \left(\frac{\alpha_{\chi_n+c_j h}}{\alpha_T} \mathbf{Z}_j \right) \right] \right) \\ 2180 \\ 2181 \quad &+ \alpha_T (a_i^W \mathbf{W}_n + a_i^H \mathbf{H}_n), \\ 2182 \\ 2183 \quad \frac{\alpha_{n+1}}{\alpha_T} \mathbf{X}_{n+1} &= \frac{\alpha_T}{\alpha_n} \mathbf{X}_n \\ 2184 \\ 2185 \quad &+ \alpha_T \underbrace{h \sum_{i=1}^s \left[2b_i \alpha_T \mathbf{x}_{T|\chi_n+c_i h} \left(\frac{\alpha_{\chi_n+c_i h}}{\alpha_T} \mathbf{Z}_i \right) \right] + \alpha_T (b^W \mathbf{W}_n + b^H \mathbf{H}_n)}_{= \Psi_h(\chi_n, \mathbf{X}_n, \mathbf{W}_\chi)}. \end{aligned} \quad (166)$$

2190 To further simplify let $\alpha_T \hat{\mathbf{Z}}_i = \mathbf{Z}_i$, then we construct the reversible scheme with forward pass:
2191

$$\begin{aligned} 2192 \quad \mathbf{X}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} (\zeta \mathbf{X}_n + (1 - \zeta) \hat{\mathbf{X}}_n) + \alpha_{n+1} \Psi_h(\chi_n, \hat{\mathbf{X}}_n, \mathbf{W}_\chi(\omega)), \\ 2193 \\ 2194 \quad \hat{\mathbf{X}}_{n+1} &= \frac{\alpha_{n+1}}{\alpha_n} \hat{\mathbf{X}}_n - \alpha_{n+1} \Psi_{-h}(\chi_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_\chi(\omega)), \end{aligned} \quad (167)$$

2196 and backward pass

$$\begin{aligned} 2197 \quad \hat{\mathbf{X}}_n &= \frac{\alpha_n}{\alpha_{n+1}} \hat{\mathbf{X}}_n + \alpha_n \Psi_{-h}(\chi_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_\chi(\omega)), \\ 2198 \\ 2199 \quad \mathbf{X}_{n+1} &= \frac{\alpha_n}{\alpha_{n+1}} \zeta^{-1} \mathbf{X}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{X}}_n - \alpha_n \zeta^{-1} \Psi_h(\chi_n, \hat{\mathbf{X}}_n, \mathbf{W}_\chi(\omega)), \end{aligned} \quad (168)$$

2202 with step size $h := \chi_{n+1} - \chi_n$

$$\begin{aligned} 2203 \quad \hat{\mathbf{Z}}_i &= \frac{\gamma_n}{\sigma_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[2a_{ij} \mathbf{x}_{T|\chi_n+c_j h} \left(\alpha_{\chi_n+c_j h} \hat{\mathbf{Z}}_j \right) \right] + a_i^W \mathbf{W}_n + a_i^H \mathbf{H}_n, \\ 2204 \\ 2205 \quad \Psi_h(\chi_n, \mathbf{X}_n, \mathbf{W}_\chi(\omega)) &= h \sum_{j=1}^s \left[2b_i \mathbf{x}_{T|\chi_n+c_i h} \left(\alpha_{\chi_n+c_i h} \hat{\mathbf{Z}}_j \right) \right] + b^W \mathbf{W}_n + b^H \mathbf{H}_n. \end{aligned} \quad (169)$$

2209 *N.B.*, $\mathbf{W}_n = \bar{\mathbf{W}}_{\chi_{n+1}^2} - \bar{\mathbf{W}}_{\chi_n^2}$. □
2210

2212 C.3 PROOF OF PROPOSITION 3.3

2213 We now can construct Rex.

2214
 2215 **Proposition 3.3** (Rex). *Without loss of generality let Φ denote an explicit SRK scheme for the SDE*
 2216 *in Equation (10) 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*
 2217 *W be the Brownian motion over time variable ς . Then the reversible solver constructed from Φ in*
 2218 *terms of the underlying state variable \mathbf{X}_t is given by the forward step*

$$\begin{aligned} 2219 \quad \mathbf{X}_{n+1} &= \frac{w_{n+1}}{w_n} \left(\zeta \mathbf{X}_n + (1 - \zeta) \hat{\mathbf{X}}_n \right) + w_{n+1} \Psi_h(\varsigma_n, \hat{\mathbf{X}}_n, \mathbf{W}_n(\omega)), \\ 2220 \quad \hat{\mathbf{X}}_{n+1} &= \frac{w_{n+1}}{w_n} \hat{\mathbf{X}}_n - w_{n+1} \Psi_{-h}(\varsigma_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_n(\omega)), \end{aligned} \quad (12)$$

2221 and backward step

$$\begin{aligned} 2224 \quad \hat{\mathbf{X}}_n &= \frac{w_n}{w_{n+1}} \hat{\mathbf{X}}_{n+1} + w_n \Psi_{-h}(\varsigma_{n+1}, \mathbf{X}_{n+1}, \mathbf{W}_n(\omega)), \\ 2225 \quad \mathbf{X}_n &= \frac{w_n}{w_{n+1}} \zeta^{-1} \mathbf{X}_{n+1} + (1 - \zeta^{-1}) \hat{\mathbf{X}}_n - w_n \zeta^{-1} \Psi_h(\varsigma_n, \hat{\mathbf{X}}_n, \mathbf{W}_n(\omega)), \end{aligned} \quad (13)$$

2228 with step size $h := \varsigma_{n+1} - \varsigma_n$ and where Ψ denotes the following scheme

$$\begin{aligned} 2230 \quad \hat{\mathbf{Z}}_i &= \frac{1}{w_n} \mathbf{X}_n + h \sum_{j=1}^{i-1} \left[a_{ij} \mathbf{f}^\theta \left(\varsigma_n + c_j h, w_{\varsigma_n + c_j h} \hat{\mathbf{Z}}_j \right) \right] + a_i^W \mathbf{W}_n(\omega) + a_i^H \mathbf{H}_n(\omega), \\ 2231 \quad \Psi_h(\varsigma_n, \mathbf{X}_n, \mathbf{W}_\varrho(\omega)) &= h \sum_{j=1}^s \left[b_i \mathbf{f}^\theta \left(\varsigma_n + c_i h, w_{\varsigma_n + c_i h} \hat{\mathbf{Z}}_j \right) \right] + b^W \mathbf{W}_n(\omega) + b^H \mathbf{H}_n(\omega), \end{aligned} \quad (14)$$

2236 where \mathbf{f}^θ denotes the data prediction model, $w_n = \frac{\sigma_n}{\gamma_n}$ and $\varsigma_t = \varrho_t$. The ODE case is recovered
 2237 for an explicit RK scheme Φ for the ODE in Equation (70) with $w_n = \sigma_n$ and $\varsigma_t = \gamma_t$. For noise
 2238 prediction models we have \mathbf{f}^θ denoting the noise prediction model with $w_n = \alpha_n$ and $\varsigma_t = \frac{\sigma_n}{\alpha_n}$.
 2239

2240 *Proof.* This follows by Lemmas C.1, C.2, C.8 and C.9 *mutatis mutandis*. \square

D CONVERGENCE ORDER PROOFS

D.1 ASSUMPTIONS

2246 Beyond the general regularity conditions imposed on the learned diffusion model itself (see Lu
 2247 et al., 2022b; Blasingame & Liu, 2024a; 2025) we also assert that in the noise prediction setting that
 2248 $\alpha_T > 0$. In practice most commonly used diffusion noise schedules like the linear or scaled linear
 2249 schedule satisfy this, (see Appendix H.1; cf. Lin et al., 2024).

D.2 PROOF OF THEOREM 4.1

2253 **Theorem 4.1** (Rex is a k -th order solver). *Let Φ be a k -th order explicit Runge-Kutta scheme for*
 2254 *the reparameterized probability flow ODE in Equation (70) with variance preserving noise schedule*
 2255 *(α_t, σ_t). Then Rex constructed from Φ is a k -th order solver, i.e., given the reversible solution*
 2256 *$\{\mathbf{x}_n, \hat{\mathbf{x}}_n\}_{n=1}^N$ and true solution \mathbf{x}_{t_n} we have*

$$\|\mathbf{x}_n - \mathbf{x}_{t_n}\| \leq Ch^k, \quad (15)$$

2259 for constants $C, h_{\max} > 0$ and for step sizes $h \in [0, h_{\max}]$.

2261 *Proof.* We will prove this for both the data prediction and noise prediction formulations.

2262 **Data prediction.** By Theorem A.1 we have that reversible Φ is a k -th order solver, and thus

$$\|\mathbf{y}_n - \mathbf{y}_{t_n}\| \leq Ch^k. \quad (170)$$

2265 We use the change of variables from Equation (70) to find

$$\left\| \frac{\sigma_T}{\sigma_n} \mathbf{x}_n - \frac{\sigma_T}{\sigma_n} \mathbf{x}_{t_n} \right\| \leq Ch^k, \quad (171)$$

2268 which simplifies to
 2269
 2270

$$\|\mathbf{x}_n - \mathbf{x}_{t_n}\| \leq \frac{\sigma_n}{\sigma_T} Ch^k. \quad (172)$$

2271 Now by definition for variance preserving type diffusion SDEs we have that $\sigma_t \leq 1$ for all t . Thus
 2272 we can write
 2273

$$\|\mathbf{x}_n - \mathbf{x}_{t_n}\| \leq C_1 h^k, \quad (173)$$

2274 where $C_1 = \frac{C}{\sigma_T}$.
 2275

2276 **Noise prediction.** By Theorem A.1 we have that reversible Φ is a k -th order solver, and thus
 2277

$$\|\mathbf{y}_n - \mathbf{y}_{t_n}\| \leq Ch^k. \quad (174)$$

2279 We use the change of variables from Equation (69) to find
 2280

$$\left\| \frac{\alpha_T}{\alpha_n} \mathbf{x}_n - \frac{\alpha_T}{\alpha_n} \mathbf{x}_{t_n} \right\| \leq Ch^k, \quad (175)$$

2283 which simplifies to
 2284

$$\|\mathbf{x}_n - \mathbf{x}_{t_n}\| \leq \frac{\alpha_n}{\alpha_T} Ch^k. \quad (176)$$

2286 Now by definition we have $\alpha_t \leq 1$ for all t and we assume that $\alpha_T > 0$. Thus we can write
 2287

$$\|\mathbf{x}_n - \mathbf{x}_{t_n}\| \leq C_1 h^k, \quad (177)$$

2289 where $C_1 = \frac{C}{\sigma_T}$.
 2290

2291 D.3 PROOF OF THEOREM 4.2

2293 **Definition D.1** (Strong order of convergence). Suppose an SDE solver admits a numerical solution
 2294 \mathbf{X}_n and we have a true solution \mathbf{X}_{t_n} . If
 2295

$$\sup_{0 \leq n \leq N} \mathbb{E} \|\mathbf{X}_n - \mathbf{X}_{t_n}\|^2 \leq Ch^{2\alpha}, \quad (178)$$

2298 where $C > 0$ is a constant and h is the step size, then the SDE solver strongly converges with order
 2299 α .

2300 **Theorem 4.2** (Convergence order for stochastic Ψ). *Let Φ be a SRK scheme with strong order of
 2301 convergence $\xi > 0$ for the reparameterized reverse-time diffusion SDE in Equation (10) with variance
 2302 preserving noise schedule (α_t, σ_t) and $\alpha_T > 0$. Then Ψ constructed from Φ has strong order of
 2303 convergence ξ .*

2305 *Proof.* We will prove this for both the data prediction and noise prediction formulations.

2306 **Data prediction.** By definition we have Φ has strong order of convergence ξ and thus,
 2307

$$\sup_{0 \leq n \leq N} \mathbb{E} \|\mathbf{Y}_n - \mathbf{Y}_{t_n}\|^2 \leq Ch^{2\xi}, \quad (179)$$

2310 where $h = \frac{\sigma_{n+1}^2}{\alpha_{n+1}} - \frac{\sigma_n^2}{\alpha_n}$. We use the change of variables from Equation (10) to find
 2311

$$\sup_{0 \leq n \leq N} \mathbb{E} \left\| \frac{\sigma_T^2 \alpha_n}{\sigma_n^2 \alpha_T} \mathbf{X}_n - \frac{\sigma_T^2 \alpha_n}{\sigma_n^2 \alpha_T} \mathbf{X}_{t_n} \right\|^2 \leq Ch^{2\xi}, \quad (180)$$

2315 which simplifies to
 2316

$$\sup_{0 \leq n \leq N} \mathbb{E} \|\mathbf{X}_n - \mathbf{X}_{t_n}\|^2 \leq \frac{\sigma_n \sqrt{\alpha_T}}{\sigma_T \sqrt{\alpha_n}} Ch^{2\xi}. \quad (181)$$

2319 Since by definition of α_n is a monotonically decreasing function, σ_n is a monotonically increasing
 2320 function, $\alpha_T > 0$, and $\sigma_T \leq 1$ we can write
 2321

$$\sup_{0 \leq n \leq N} \mathbb{E} \|\mathbf{X}_n - \mathbf{X}_{t_n}\|^2 \leq Ch^{2\xi}, \quad (182)$$

2322

as

$$\frac{\sigma_n \sqrt{\alpha_T}}{\sigma_T \sqrt{\alpha_n}} \leq 1. \quad (183)$$

2325

Noise prediction. By definition we have Φ has strong order of convergence ξ and thus,

2327

$$\sup_{0 \leq n \leq N} \mathbb{E} \|\mathbf{Y}_n - \mathbf{Y}_{t_n}\|^2 \leq Ch^{2\xi}, \quad (184)$$

2329

2330 where $h = \frac{\sigma_{n+1}}{\alpha_{n+1}} - \frac{\sigma_n}{\alpha_n}$. We use the change of variables from Equation (133) to find

2331

$$\sup_{0 \leq n \leq N} \mathbb{E} \left\| \frac{\alpha_n}{\alpha_T} \mathbf{X}_n - \frac{\alpha_n}{\alpha_T} \mathbf{X}_{t_n} \right\|^2 \leq Ch^{2\xi}, \quad (185)$$

2334

which simplifies to

2335

$$\sup_{0 \leq n \leq N} \mathbb{E} \|\mathbf{X}_n - \mathbf{X}_{t_n}\|^2 \leq \frac{\sqrt{\alpha_T}}{\sqrt{\alpha_n}} Ch^{2\xi}. \quad (186)$$

2337

2338 Since by definition of α_n is a monotonically decreasing function strictly less than 1 and $\alpha_T > 0$ we
2339 can write

2340

$$\sup_{0 \leq n \leq N} \mathbb{E} \|\mathbf{X}_n - \mathbf{X}_{t_n}\|^2 \leq Ch^{2\xi}. \quad (187)$$

2341

□

2342

2343

E RELATION TO OTHER SOLVERS FOR DIFFUSION MODELS

2345

2346

2347 While this paper primarily focused on Rex and the family of reversible solvers created by it, we wish
2348 to discuss the relation between the underlying scheme Ψ constructed from our method and other
2349 existing solvers for diffusion models.

2350

2351

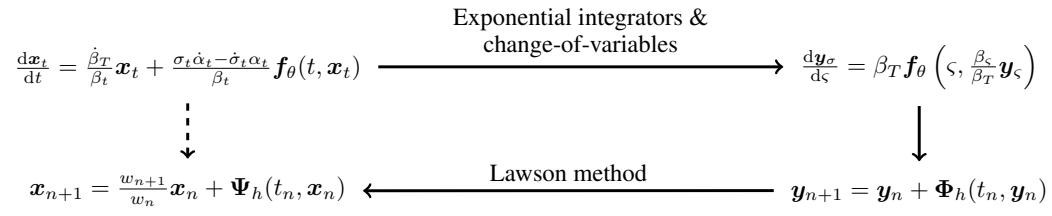
2352

2353

2354

2355

2356



2357

2358 Figure 6: Overview of the construction of Ψ for the probability flow ODE from an underlying RK
2359 scheme Φ for the reparameterized ODE. This graph holds for the SDE case *mutatis mutandis*.

2360

2361 Surprisingly, we discover that using Lawson methods outlined in Figure 6 (cf. Figure 2 from the main
2362 paper) is a surprisingly generalized methodology for construing numerical schemes for diffusion
2363 modes, and that it subsumes previous works. This means that several of the reversible schemes we
2364 presented here are reversible variants of well known schemes in the literature in diffusion models.

2365

Theorem 4.3 (Rex subsumes previous solvers). *The underlying scheme used Ψ in Rex given by*

2366

2367

2368

2369

2370

2371

2372

2373

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

$$\mathbf{X}_{n+1} = \frac{w_{n+1}}{w_n} \mathbf{X}_n + w_{n+1} \left(h \sum_{j=1}^s \left[b_{ij} \mathbf{f}^\theta \left(\varsigma_n + c_i h, w_{\varsigma_n + c_i h} \hat{\mathbf{Z}}_j \right) \right] + b^W \mathbf{W}_n(\omega) + b^H \mathbf{H}_n(\omega) \right), \quad (16)$$

subsumes the following solvers for diffusion models

2374

2375

1. DDIM (Song et al., 2021a),
2. DPM-Solver-1, DPM-Solver-2, DPM-Solver-12 (Lu et al., 2022b),

- 2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
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. \square

Corollary 4.3.1 (Rex is reversible version of previous solvers). *Rex is the reversible revision of the well-known solvers for diffusion models in Theorem 4.3.*

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

$$\begin{array}{c|ccccc} c_1 & & & & & \\ \hline c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \hline c_s & a_{s1} & a_{s2} & \cdots & a_{(s-1)s} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} = \begin{array}{c|c} c & a \\ \hline & b \end{array}. \quad (188)$$

Embedded methods for adaptive step sizing are of the form

$$\begin{array}{c|ccccc} c_1 & & & & & \\ \hline c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \hline c_s & a_{s1} & a_{s2} & \cdots & a_{(s-1)s} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \\ & b_1^* & b_2^* & \cdots & b_{s-1}^* & b_s^* \end{array}, \quad (189)$$

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 Φ . The Butcher tableau for the Euler method is

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}. \quad (190)$$

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 (70) is the DPM-Solver++1 from [Lu et al. \(2022a\)](#).*

Proof. Apply in the Butcher tableau for the Euler scheme to Ψ constructed from Equation (69) to find

$$\mathbf{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} \mathbf{x}_n + \sigma_{n+1} h \mathbf{x}_{0|\gamma_n}^\theta(\mathbf{x}_n), \quad (191)$$

2430 with $h = \gamma_{n+1} - \gamma_n$. We can rewrite the step size as
 2431

$$2433 \quad \sigma_{n+1}h = \sigma_{n+1} \left(\frac{\alpha_{n+1}}{\sigma_{n+1}} - \frac{\alpha_n}{\sigma_n} \right), \quad (192)$$

$$2435 \quad = \left(\alpha_{n+1} - \alpha_n \frac{\sigma_{n+1}}{\sigma_n} \right), \quad (193)$$

$$2437 \quad = \left(\alpha_{n+1} \frac{\alpha_{n+1}}{\alpha_{n+1}} - \frac{\alpha_n}{\alpha_{n+1}} \frac{\sigma_{n+1}}{\sigma_n} \right), \quad (194)$$

$$2440 \quad = -\alpha_{n+1} \left(\frac{\alpha_n}{\alpha_{n+1}} \frac{\sigma_{n+1}}{\sigma_n} - 1 \right), \quad (195)$$

$$2442 \quad = -\alpha_{n+1} \left(\frac{\gamma_n}{\gamma_{n+1}} - 1 \right), \quad (196)$$

$$2444 \quad = -\alpha_{n+1} \left(e^{\log \frac{\gamma_n}{\gamma_{n+1}}} - 1 \right), \quad (197)$$

$$2446 \quad = -\alpha_{n+1} \left(e^{\log \gamma_n - \log \gamma_{n+1}} - 1 \right), \quad (198)$$

$$2448 \quad \stackrel{(i)}{=} -\alpha_{n+1} \left(e^{\lambda_n - \lambda_{n+1}} - 1 \right), \quad (199)$$

$$2449 \quad \stackrel{(ii)}{=} -\alpha_{n+1} \left(e^{-h_\lambda} - 1 \right), \quad (200)$$

2451 where (i) holds by the letting $\lambda_t = \log \gamma_t$ following the notation of [Lu et al. \(2022b;a\)](#) and (ii) holds
 2452 by letting $h_\lambda = \lambda_{n+1} - \lambda_n$. Plugging this back into Equation (191) yields
 2453

$$2455 \quad \mathbf{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} \mathbf{x}_n - \alpha_{n+1} \left(e^{-h_\lambda} - 1 \right) \mathbf{x}_{0|t_n}^\theta(\mathbf{x}_n), \quad (201)$$

2458 which is the DPM-Solver++1 from [Lu et al. \(2022a\)](#). □
 2459

2462 **Corollary E.1.1** (Rex (Euler) is reversible deterministic DDIM for data prediction models). *The
 2463 underlying scheme of Rex (Euler) for the data prediction parameterization of diffusion models in
 2464 Equation (70) is the deterministic DDIM solver from [Song et al. \(2021a\)](#).*
 2465

2466 *Proof.* This holds because DPM-Solver++1 is DDIM see [Lu et al. \(2022a, Equation \(21\)\)](#) with
 2467 $\eta = 0$. □
 2468

2474 **Proposition E.2** (Rex (Euler) is reversible DPM-Solver-1). *The underlying scheme of Rex (Euler)
 2475 for the data prediction parameterization of diffusion models in Equation (69) is the DPM-Solver-1
 2476 from [Lu et al. \(2022b, Equation \(3.7\)\)](#).*
 2477

2481 *Proof.* Apply in the Butcher tableau for the Euler scheme to Ψ from Rex (see Proposition 3.3) to find
 2482

$$2483 \quad \mathbf{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n} \mathbf{x}_n + \alpha_{n+1} h \mathbf{x}_{T|\chi_n}^\theta(\mathbf{x}_n), \quad (202)$$

2484 with $h = \chi_{n+1} - \chi_n$. We can rewrite step size as
 2485

$$\alpha_{n+1}h = \alpha_{n+1} \left(\frac{\sigma_{n+1}}{\alpha_{n+1}} - \frac{\sigma_n}{\alpha_n} \right), \quad (203)$$

$$= \left(\sigma_{n+1} - \sigma_n \frac{\alpha_{n+1}}{\alpha_n} \right), \quad (204)$$

$$= \left(\sigma_{n+1} \frac{\sigma_{n+1}}{\sigma_{n+1}} - \frac{\sigma_n}{\sigma_{n+1}} \frac{\alpha_{n+1}}{\alpha_n} \right), \quad (205)$$

$$= -\sigma_{n+1} \left(\frac{\sigma_n}{\sigma_{n+1}} \frac{\alpha_{n+1}}{\alpha_n} - 1 \right), \quad (206)$$

$$= -\sigma_{n+1} \left(\frac{\chi_n}{\chi_{n+1}} - 1 \right), \quad (207)$$

$$= -\sigma_{n+1} \left(e^{\log \frac{\chi_n}{\chi_{n+1}}} - 1 \right), \quad (208)$$

$$= -\sigma_{n+1} \left(e^{\log \chi_n - \log \chi_{n+1}} - 1 \right), \quad (209)$$

$$\stackrel{(i)}{=} -\sigma_{n+1} \left(e^{-\lambda_n + \lambda_{n+1}} - 1 \right), \quad (210)$$

$$\stackrel{(ii)}{=} -\sigma_{n+1} \left(e^{h_\lambda} - 1 \right), \quad (211)$$

2504 where (i) holds by the letting $\lambda_t = \log \gamma_t = -\log \chi_t$ following the notation of [Lu et al. \(2022b;a\)](#)
 2505 and (ii) holds by letting $h_\lambda = \lambda_{n+1} - \lambda_n$. Plugging this back into Equation (191) yields
 2506

$$\mathbf{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n} \mathbf{x}_n - \sigma_{n+1} \left(e^{h_\lambda} - 1 \right) \mathbf{x}_{T|t_n}^\theta(\mathbf{x}_n), \quad (212)$$

2507 which is the DPM-Solver-1 from [Lu et al. \(2022b\)](#). \square
 2508

2511 **Corollary E.2.1** (Rex (Euler) is reversible deterministic DDIM for noise prediction models). *The
 2512 underlying scheme of Rex (Euler) for the noise prediction parameterization of diffusion models in
 2513 Equation (69) is the deterministic DDIM solver from [Song et al. \(2021a\)](#).*

2514 *Proof.* This holds because DPM-Solver-1 is DDIM see [Lu et al. \(2022b, Equation \(4.1\)\)](#). \square
 2515

2517 E.1.2 SECOND-ORDER METHODS

2518 In this section we explore the numerical schemes produced by choosing the explicit midpoint method
 2519 for Φ . We can write a generic second-order method as
 2520

$$\begin{array}{c|cc} 0 & & \\ \hline \eta & \eta & \\ \hline 1 - \frac{1}{2\eta} & \frac{1}{2\eta} & \end{array}, \quad (213)$$

2521 for $\eta \neq 0$ ([Butcher, 2016](#)). The choice of $\eta = \frac{1}{2}$ yields the explicit midpoint, $\eta = \frac{2}{3}$ gives Ralston's
 2522 second-order method, and $\eta = 1$ gives Heun's second-order method.
 2523

2524 **Proposition E.3** (Rex (generic second-order) is reversible DPM-Solver++(2S)). *The underlying
 2525 scheme of Rex (generic second-order) for the data prediction parameterization of diffusion models in
 2526 Equation (70) is the DPM-Solver++(2S) from [Lu et al. \(2022a, Algorithm 1\)](#).*
 2527

2528 *Proof.* The DPM-Solver++(2S) ([Lu et al., 2022a](#), Algorithm 1) is defined as
 2529

$$\begin{aligned} \mathbf{u} &= \frac{\sigma_p}{\sigma_n} \mathbf{x}_n - \alpha_p \left(e^{-r_\lambda h_\lambda} - 1 \right) \mathbf{x}_{0|t_n}^\theta(\mathbf{x}_n), \\ \mathbf{D} &= \left(1 - \frac{1}{2r_\lambda} \right) \mathbf{x}_{0|t_n}^\theta(\mathbf{x}_n) + \frac{1}{2r_\lambda} \mathbf{x}_{0|t_p}^\theta(\mathbf{u}), \\ \mathbf{x}_{n+1} &= \frac{\sigma_{n+1}}{\sigma_n} \mathbf{x}_n - \alpha_{n+1} \left(e^{-h_\lambda} - 1 \right) \mathbf{D}, \end{aligned} \quad (214)$$

2538 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
 2539 the location of the midpoint time in the λ -domain as a ratio, *i.e.*, we could say
 2540

$$2541 \quad \lambda_p = \lambda_n + r_\lambda h_\lambda, \quad (215)$$

2542 where $r_\lambda \in (0, 1)$ denotes the interpolation point between the initial timestep λ_n and terminal
 2543 timestep λ_{n+1} . Thus we fix $\eta = r_\lambda$ as the step size ratio of the intermediate point.
 2544

2545 Now we return to the underlying scheme of Rex applied to the generic second-order scheme, see
 2546 Equation (213), Apply in the Butcher tableau for generic second-order scheme to Ψ constructed from
 2547 Equation (69) to find

$$2548 \quad \mathbf{z} = \frac{1}{\sigma_n} \mathbf{x}_n + \eta h \mathbf{x}_{0|\gamma_n}^\theta(\mathbf{x}_n), \\ 2549 \quad \mathbf{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} \mathbf{x}_n + \sigma_{n+1} h \left(\left(1 - \frac{1}{2\eta} \right) \mathbf{x}_{0|\gamma_n}^\theta(\mathbf{x}_n) + \frac{1}{2\eta} \mathbf{x}_{0|\gamma_n+\eta h}^\theta(\sigma_p \mathbf{z}) \right), \quad (216)$$

2553 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
 2554

$$2555 \quad \sigma_p \mathbf{z} = \frac{\sigma_p}{\sigma_n} \mathbf{x}_n + \sigma_p \eta h \mathbf{x}_{0|\gamma_n}^\theta(\mathbf{x}_n). \quad (217)$$

2557 Plugging this back into Equation (216) yields

$$2558 \quad \sigma_p \mathbf{z} = \frac{\sigma_p}{\sigma_n} \mathbf{x}_n + \sigma_p \eta h \mathbf{x}_{0|\gamma_n}^\theta(\mathbf{x}_n), \\ 2559 \quad \mathbf{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} \mathbf{x}_n + \sigma_{n+1} h \underbrace{\left(\left(1 - \frac{1}{2\eta} \right) \mathbf{x}_{0|\gamma_n}^\theta(\mathbf{x}_n) + \frac{1}{2\eta} \mathbf{x}_{0|\gamma_n+\eta h}^\theta(\sigma_p \mathbf{z}) \right)}_{= \hat{\mathbf{D}}}, \quad (218)$$

2564 which is the DPM-Solver++1 from Lu et al. (2022a). Now recall from Proposition E.1 that
 2565

$$2566 \quad \sigma_{n+1} h = -\alpha_{n+1} (e^{-h_\lambda} - 1), \quad (219)$$

2567 it follows that

$$2568 \quad \sigma_p \eta h = -\alpha_p (e^{-r_\lambda h_\lambda} - 1), \quad (220)$$

2569 due to $\lambda_p - \lambda_n = r_\lambda h_\lambda$ and $\eta h = \lambda_p - \lambda_n$. Thus by letting $\sigma_p \mathbf{z} = \mathbf{u}$ and $\hat{\mathbf{D}} = \mathbf{D}$ we recover the
 2570 DPM-Solver++(2S) solver. \square
 2571

2573 **Proposition E.4** (Rex (generic second-order) is reversible DPM-Solver-2)). *The underlying scheme*
 2574 *of Rex (generic second-order) for the noise prediction parameterization of diffusion models in*
 2575 *Equation (69) is the DPM-Solver-2 from Lu et al. (2022b, Algorithm 4 cf. Algorithm 1).*

2576 *Proof.* This follows as straightforward derivation from Proposition E.2 and Proposition E.3. \square

2579 **Proposition E.5** (Rex (Euler-Midpoint) is DPM-Solver-12). *The underlying scheme of Rex (Euler-
 2580 Midpoint) for the noise prediction parameterization of diffusion models in Equation (69) is the
 2581 DPM-Solver-12 from Lu et al. (2022b).*

2583 *Proof.* The explicit midpoint method with embedded Euler method for adaptive step sizing is given
 2584 by the Butcher tableau

$$2585 \quad \begin{array}{c|cc} 0 & & \\ \hline \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \\ & 1 & 0 \end{array}. \quad (221)$$

2590 From Proposition E.2 and Proposition E.4 we have shown that Rex (Euler) and Rex (Midpoint)
 2591 correspond to DPM-Solver-1 and DPM-Solver-2 respectively. Thus the Butcher tableau above
 2592 outlines DPM-Solver-12. \square

2592 E.1.3 THIRD-ORDER METHODS
25932594 For third-order solvers like DPM-Solver-3 (Lu et al., 2022b, Algorithm 5) our constructed scheme
2595 differs from solvers derived using ETD methods due to the presence of φ_2 terms where

2596
$$\varphi_{k+1}(t) = \int_0^1 e^{(1-\delta)t} \frac{\delta^k}{k!} d\delta, \quad (222)$$

2597
2598

2599 this also reasoning extends to the DPM-Solver-4 from Gonzalez et al. (2024, Algorithm 7).
26002601 E.2 REX AS REVERSIBLE SDE SOLVERS
26022603 In this section we discuss the connections between Rex and preexisting SDE solvers for diffusion
2604 models.2605 E.2.1 EULER-MARUYAMA
26062607 The extended Butcher tableau for the Euler-Maruyama scheme is given by
2608

2609
$$\begin{array}{c|cc|c} 0 & 0 & 0 & 0 \\ \hline & 1 & 1 & 0 \end{array}. \quad (223)$$

2610
2611

2612 **Proposition E.6** (Rex (Euler-Maruyama) is reversible SDE-DPM-Solver++1). *The underlying
2613 scheme of Rex (Euler-Maruyama) for the data prediction parameterization of diffusion models in
2614 Equation (10) is the SDE-DPM-Solver++1 from Lu et al. (2022a, Equation (18)).*2615 *Proof.* Apply in the Butcher tableau for the Euler-Maruyama scheme to Ψ constructed from Equation
2616 (133) to find
2617

2618
$$\mathbf{x}_{n+1} = \frac{\sigma_{n+1}^2 \alpha_n}{\sigma_n^2 \alpha_{n+1}} \mathbf{x}_n + \frac{\sigma_{n+1}^2}{\alpha_{n+1}} h \mathbf{x}_{0|\varrho_n}^\theta(\mathbf{x}_n) + \frac{\sigma_{n+1}^2}{\alpha_{n+1}} \mathbf{W}_n, \quad (224)$$

2619
2620

2621 with $h = \varrho_{n+1} - \varrho_n$. We can rewrite the step size as
2622

2623
$$\frac{\sigma_{n+1}^2}{\alpha_{n+1}} h = \frac{\sigma_{n+1}^2}{\alpha_{n+1}} \left(\frac{\alpha_{n+1}^2}{\sigma_{n+1}^2} - \frac{\alpha_n^2}{\sigma_n^2} \right), \quad (225)$$

2624

2625
$$= \left(\alpha_{n+1} - \frac{\alpha_n^2}{\alpha_{n+1}} \frac{\sigma_{n+1}^2}{\sigma_n^2} \right), \quad (226)$$

2626

2627
$$= \alpha_{n+1} \left(1 - \frac{\alpha_n^2}{\alpha_{n+1}^2} \frac{\sigma_{n+1}^2}{\sigma_n^2} \right), \quad (227)$$

2628

2629
$$= \alpha_{n+1} \left(1 - \frac{\varrho_n}{\varrho_{n+1}} \right), \quad (228)$$

2630

2631
$$= \alpha_{n+1} \left(1 - e^{2 \log \frac{\varrho_n}{\varrho_{n+1}}} \right), \quad (229)$$

2632

2633
$$= \alpha_{n+1} \left(1 - e^{2 \log \varrho_n - 2 \log \varrho_{n+1}} \right), \quad (230)$$

2634

2635
$$\stackrel{(i)}{=} \alpha_{n+1} \left(1 - e^{2 \lambda_n - 2 \lambda_{n+1}} \right), \quad (231)$$

2636

2637
$$\stackrel{(ii)}{=} \alpha_{n+1} \left(1 - e^{-2h_\lambda} \right), \quad (232)$$

2638

2639 where (i) holds by the letting $\lambda_t = \log \varrho_t$ following the notation of Lu et al. (2022b;a) and (ii) holds
2640 by letting $h_\lambda = \lambda_{n+1} - \lambda_n$. Now recall that

2641
$$\frac{\sigma_{n+1}^2 \alpha_n}{\sigma_n^2 \alpha_{n+1}} = \frac{\sigma_{n+1}}{\sigma_n} e^{-h_\lambda}. \quad (233)$$

2642

2643 Plugging these back into Equation (224) yields
2644

2645
$$\mathbf{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} e^{-h_\lambda} \mathbf{x}_n + \alpha_{n+1} \left(1 - e^{-2h_\lambda} \right) \mathbf{x}_{0|t_n}^\theta(\mathbf{x}_n) + \frac{\sigma_{n+1}^2}{\alpha_n} \mathbf{W}_n. \quad (234)$$

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

$$\mathbf{W}_n \sim \sqrt{h} \boldsymbol{\epsilon}, \quad (235)$$

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

$$\frac{\sigma_{n+1}^2}{\alpha_{n+1}} \sqrt{h} = \frac{\sigma_{n+1}^2}{\alpha_{n+1}} \sqrt{\frac{\alpha_{n+1}^2}{\sigma_{n+1}^2} - \frac{\alpha_n^2}{\sigma_n^2}}, \quad (236)$$

$$= \sqrt{\sigma_{n+1}^2 - \frac{\alpha_n^2}{\alpha_{n+1}^2} \frac{\sigma_{n+1}^4}{\sigma_n^2}}, \quad (237)$$

$$= \sigma_{n+1} \sqrt{1 - \frac{\alpha_n^2}{\alpha_{n+1}^2} \frac{\sigma_{n+1}^2}{\sigma_n^2}}, \quad (238)$$

$$= \sigma_{n+1} \sqrt{1 - \frac{\varrho_n}{\varrho_{n+1}}}, \quad (239)$$

$$= \sigma_{n+1} \sqrt{1 - e^{-2h_\lambda}}. \quad (240)$$

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

$$\mathbf{x}_{n+1} = \frac{\sigma_{n+1}}{\sigma_n} e^{-h_\lambda} \mathbf{x}_n + \alpha_{n+1} (1 - e^{-2h_\lambda}) \mathbf{x}_{0|t_n}^\theta(\mathbf{x}_n) + \sigma_{n+1} \sqrt{1 - e^{-2h_\lambda}} \boldsymbol{\epsilon}. \quad (241)$$

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

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 (10) 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. \(2022a, Section 6.1\)](#). \square

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 (133) is the SDE-DPM-Solver-1 from [Lu et al. \(2022a, Equation \(17\)\)](#).*

Proof. Apply in the Butcher tableau for the Euler scheme to Ψ from Rex (see Proposition 3.3) to find

$$\mathbf{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n} \mathbf{x}_n + 2\alpha_{n+1} h \mathbf{x}_{T|\chi_n}^\theta(\mathbf{x}_n) + \alpha_{n+1} \mathbf{W}_n, \quad (242)$$

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

$$\alpha_{n+1} h = -\sigma_{n+1} (e^{h_\lambda} - 1). \quad (243)$$

Now recall that the Brownian increment $\mathbf{W}_n := \overline{\mathbf{W}}_{\chi_{n+1}^2} - \overline{\mathbf{W}}_{\chi_n^2}$ has variance $\chi_n^2 - \chi_{n+1}^2$.²⁰ Thus via the Itô isometry we can write

$$\mathbf{W}_n \sim \sqrt{\chi_n^2 - \chi_{n+1}^2} \boldsymbol{\epsilon}, \quad (244)$$

²⁰This is because $\overline{\mathbf{W}}_\chi$ is defined in reverse-time.

2700 with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then we have
 2701

$$\alpha_{n+1} \sqrt{\chi_n^2 - \chi_{n+1}^2} = \alpha_{n+1} \sqrt{\frac{\sigma_n^2}{\alpha_n^2} - \frac{\sigma_{n+1}^2}{\alpha_{n+1}^2}}, \quad (245)$$

$$= \sqrt{\frac{\sigma_n^2 \alpha_{n+1}^2}{\alpha_n^2} - \sigma_{n+1}^2}, \quad (246)$$

$$= \sigma_{n+1} \sqrt{\frac{\sigma_n^2 \alpha_{n+1}^2}{\sigma_{n+1}^2 \alpha_n^2} - 1}, \quad (247)$$

$$= \sigma_{n+1} \sqrt{\frac{\chi_n^2}{\chi_{n+1}^2} - 1}, \quad (248)$$

$$= \sigma_{n+1} \sqrt{e^{\log \frac{\chi_n^2}{\chi_{n+1}^2}} - 1}, \quad (249)$$

$$= \sigma_{n+1} \sqrt{e^{\log \chi_n^2 - \log \chi_{n+1}^2} - 1}, \quad (250)$$

$$= \sigma_{n+1} \sqrt{e^{-2 \log \gamma_n + 2 \log \gamma_{n+1}} - 1}, \quad (251)$$

$$= \sigma_{n+1} \sqrt{e^{2 \log \lambda_{n+1} - 2 \log \lambda_n} - 1}, \quad (252)$$

$$= \sigma_{n+1} \sqrt{e^{2h_\lambda} - 1}. \quad (253)$$

2722 Plugging Equations (243) and (253) back into Equation (242) yields
 2723

$$\mathbf{x}_{n+1} = \frac{\alpha_{n+1}}{\alpha_n} \mathbf{x}_n - 2\sigma_{n+1} (e^{h_\lambda} - 1) \mathbf{x}_{T|\chi_n}^\theta(\mathbf{x}_n) + \sigma_{n+1} \sqrt{e^{2h_\lambda} - 1} \epsilon, \quad (254)$$

2726 which is the SDE-DPM-Solver-1 from Lu et al. (2022a). \square

2727 **Corollary E.7.1** (Rex (Euler-Maruyama) is reversible stochastic DDIM for noise prediction models).
 2728 *The underlying scheme of Rex (Euler-Maruyama) for the noise prediction parameterization of*
 2729 *diffusion models in Equation (133) is the stochastic DDIM solver from Song et al. (2021a) with*
 2730 $\eta = \sigma_t \sqrt{e^{-2h_\lambda} - 1}$.
 2731

2732 *Proof.* This follows from a straightforwardly from Corollary E.6.1 and Lu et al. (2022b, Equation
 2733 (4.1)). \square

2735 E.3 REX AS REVERSIBLE SEEDS-1

2737 **Proposition E.8** (Rex is reversible SEEDS-1). *The choice of Euler or Euler-Maruyama for the*
 2738 *underlying scheme of Rex with either the noise prediction parameterization of diffusion models in*
 2739 *Equations (69) and (133) or data prediction in Equations (10) and (69) yields the four variants of*
 2740 *SEEDS-1 outlined in Gonzalez et al. (2024, Equations (28-31)).*

2742 *Proof.* This follows straightforwardly from Propositions E.1, E.2, E.6 and E.7 by definition of
 2743 SEEDS-1. \square

2744 **Corollary E.8.1** (Rex (Euler-Maruyama) is reversible gDDIM). *The underlying scheme of Rex*
 2745 *(Euler-Maruyama) for the data prediction parameterization of diffusion models in Equation (10) is*
 2746 *the gDDIM solver in Zhang et al. (2023, Theorem 1) for $\ell = 1$.*
 2747

2748 *Proof.* This follows as an immediate consequence of Proposition E.8 since by Gonzalez et al. (2024),
 2749 Proposition 4.5) gDDIM is SEEDS-1. \square

2751 As mentioned earlier in Section A.4.1 high-order variants of SEEDS use a Markov-preserving noise
 2752 decomposition to approximate the iterated stochastic integrals. However, we follow Foster et al.
 2753 (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.

2754 **F A BRIEF NOTE ON THE THEORY OF ROUGH PATHS**
 2755

2756 To perform reversibility it is useful to consider the pathwise interpretation of SDEs (Lyons, 1998), as
 2757 such we introduce a few notations from rough path theory. Let $\{\mathbf{W}_t\}$ be a d_w -dimensional Brownian
 2758 motion and let \mathbf{W} be enhanced by

2759
$$\mathbb{W}_{s,t} = \int_s^t \mathbf{W}_{s,r} \otimes \circ d\mathbf{W}_r, \quad (255)$$

 2760

2762 where \otimes is the tensor product. Then, the pair $\mathcal{W} := (\mathbf{W}, \mathbb{W})$ is the *Stratonovich enhanced Brownian*
 2763 *rough path*.²¹ Thus consider the d_x -dimensional *rough differential equation* RDE of the form:

2765
$$d\mathbf{X}_t = \boldsymbol{\mu}(t, \mathbf{X}_t) dt + \boldsymbol{\sigma}(t, \mathbf{X}_t) d\mathcal{W}_t, \quad \mathbf{X}_0 = \mathbf{x}_0. \quad (256)$$

2766 where $\boldsymbol{\mu} : [0, T] \times \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x}$ is Lipschitz continuous in its second argument and $\boldsymbol{\sigma} \in \mathcal{C}_b^{1,3}([0, T] \times$
 2767 $\mathbb{R}^{d_x}; \mathcal{L}(\mathbb{R}^{d_w}, \mathbb{R}^{d_x}))$ (Friz & Hairer, 2020, Theorem 9.1).²² Fix an $\omega \in \Omega$, then almost surely $\mathcal{W}(\omega)$
 2768 admits a unique solution to the RDE $(\mathbf{X}_t(\omega), \boldsymbol{\sigma}(t, \mathbf{X}_t(\omega)))$ and $\mathbf{X}_t = \mathbf{X}_t(\omega)$ is a strong solution to
 2769 the Stratonovich SDE²³ started at $\mathbf{X}_0 = \mathbf{x}_0$. To elucidate, consider the commutative diagram below

2771
$$\mathbf{W} \xrightarrow{\Psi} (\mathbf{W}, \mathbb{W}) \xrightarrow{S} \mathbf{X}, \quad (257)$$

 2772

2773 where Ψ is a map which merely lifts Brownian motion into a rough path (could be Itô or Stratonovich),
 2774 the second map, S , is known as the *Itô-Lyons map* (Lyons, 1998); this map is purely deterministic
 2775 and is also a *continuous map* w.r.t. to initial condition and driving signal. Thus for a fixed realization
 2776 of the Brownian motion we have a pathwise interpretation of the Stratonovich SDE.

2777 **G NUMERICAL SIMULATION OF BROWNIAN MOTION**
 2778

2779 Earlier we mentioned that for reversible methods we need to be able to compute both the *same*
 2780 realization of the Brownian motion. Now sampling Brownian motion is quite simple—recall Lévy’s
 2781 characterization of Brownian motion (Øksendal, 2003, Theorem 8.6.1)—and can be sampled by
 2782 drawing independent Gaussian increments during the numerical solve of an SDE. A common choice
 2783 for an adaptive solver is to use Lévy’s Brownian bridge formula (Revuz & Yor, 2013).

2784 **Definition G.1** (Lévy’s Brownian bridge). Given the standard d_w -dimensional Brownian motion
 2785 $\{\mathbf{W}_t : t \geq 0\}$ and for any $0 \leq s < t < u$, the Brownian bridge is defined as

2786
$$\mathbf{W}_t | \mathbf{W}_s, \mathbf{W}_u \sim \mathcal{N} \left(\mathbf{W}_s + \frac{t-s}{u-s} (\mathbf{W}_u - \mathbf{W}_s), \frac{(u-t)(t-s)}{u-s} \mathbf{I} \right), \quad (258)$$

 2787

2788 and this quantity is conditionally independent of \mathbf{W}_v for $v < s$ or $v > u$.

2789 Sampling the Brownian motion in reverse-time, however, is more complicated as it is only adapted
 2790 to the natural filtration defined in forward time. The naïve approach to sampling Brownian motion,
 2791 called the *Brownian path*, is to simply store the entire realization of the Brownian motion from the
 2792 forward pass in memory and use Equation (258) when necessary (for adaptive step size methods).
 2793 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
 2794 samples.

2795 **Virtual Brownian Tree.** Seminal work on neural SDEs by Li et al. (2020) introduced the *Virtual*
 2796 *Brownian Tree* which extends the concept of Brownian trees introduced by Gaines & Lyons (1997).
 2797 The Brownian tree recursively applies Equation (258) to sample the Brownian motion at any midpoint,
 2798 constructing a tree structure; however, storing such a tree would be memory intensive. By making

2799 ²¹See, Friz & Hairer (2020, Chapter 3) for more details.

2800 ²²Here $\mathcal{L}(V, W)$ denotes the set of continuous maps from V to W , a Banach space.

2801 ²³If \mathbf{X}_t and $\partial_{\mathbf{x}} \mathbf{X}_t$ are adapted and $\langle \mathbf{X}, \mathbf{W} \rangle_t$ exists, then almost surely

2802
$$\int_0^T \mathbf{X} d\mathcal{W}_t = \int_0^T \mathbf{X} \circ d\mathbf{W}_t.$$

 2803

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 $\mathbf{W}_{s,t} = \mathbf{W}_t - \mathbf{W}_s$ denote an interval of Brownian motion. Then the formula for Lévy's Brownian bridge (258) can be rewritten in terms of Brownian intervals as

$$\mathbf{W}_{s,t} | \mathbf{W}_{s,u} \sim \mathcal{N} \left(\frac{t-s}{u-s} \mathbf{W}_{s,u}, \frac{(u-t)(s-u)}{u-s} \mathbf{I} \right). \quad (259)$$

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 area²⁴ which would require the Brownian interval to construct.²⁵

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.²⁶ 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.

H IMPLEMENTATION DETAILS

H.1 CLOSED FORM EXPRESSIONS OF THE NOISE SCHEDULE

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 arithmetic sequence of the form

$$\beta_n = \frac{\beta_0}{N} + \frac{n-1}{N(N-1)}(\beta_1 - \beta_0), \quad (260)$$

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

$$\beta_t = \beta_0 + t(\beta_1 - \beta_0), \quad (261)$$

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

$$d\mathbf{X}_t = -\frac{1}{2}\beta_t \mathbf{X}_t dt + \sqrt{\beta_t} d\mathbf{W}_t. \quad (262)$$

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

$$\begin{aligned} \alpha_t &= \exp \left(-\frac{1}{2} \int \beta_t dt \right), \\ \sigma_t &= \sqrt{1 - \alpha_t^2}. \end{aligned} \quad (263)$$

²⁴I.e., for a d_w -dimensional Brownian motion over $[s, t]$ the Lévy area is

$$2\mathbf{L}_{s,t}^{i,j} := \int_s^t \mathbf{W}_{s,u}^i d\mathbf{W}_u^j - \int_s^t \mathbf{W}_{s,u}^j d\mathbf{W}_u^i.$$

²⁵The interested reader can find more details in James Foster's thesis (Foster, 2020).

²⁶This algorithm is a part of the popular `Diffrax` library.

2862 *N.B.*, often the hyperparameters in libraries like `diffusers` are expressed as $\hat{\beta}_0 = \frac{\beta_0}{N}$ and $\hat{\beta}_1 = \frac{\beta_1}{N}$,
 2863 often with $N = 1000$.
 2864

2865 H.1.1 LINEAR NOISE SCHEDULE

2866 For the linear noise schedule in Equation (261) used by DDPMs (Ho et al., 2020), the schedule
 2867 (α_t, σ_t) is written as

$$2869 \quad \alpha_t = \exp \left(-\frac{\beta_1 - \beta_0}{4} t^2 - \frac{\beta_0}{2} t \right), \quad (264)$$

$$2870 \quad \sigma_t = \sqrt{1 - \alpha_t^2},$$

2871 for $t \in [0, 1]$ with hyperparameters β_0 and β_1 .
 2872

2873 **Proposition H.1** (Inverse function of γ_t for linear noise schedule). *For the linear noise schedule used
 2874 by DDPMs (Ho et al., 2020) the inverse function of γ_t denoted t_γ can be expressed in closed form as*

$$2876 \quad t_\gamma(\gamma) = \frac{-\beta_0 + \sqrt{\beta_0^2 + 2(\beta_1 - \beta_0) \log(\gamma^{-2} + 1)}}{\beta_1 - \beta_0}. \quad (265)$$

2877 *Proof.* Let α_t be denoted by $\alpha_t = e^{a_t}$ where
 2878

$$2879 \quad a_t = -\frac{\beta_1 - \beta_0}{4} t^2 - \frac{\beta_0}{2} t. \quad (266)$$

2880 Then by definition of γ_t we can write
 2881

$$2882 \quad \gamma_t = \frac{e^{a_t}}{\sqrt{1 - e^{2a_t}}}, \quad (267)$$

2883 and with a little more algebra we find
 2884

$$2885 \quad \sqrt{1 - e^{2a_t}} = \frac{e^{a_t}}{\gamma_t}, \quad (268)$$

$$2886 \quad 1 - e^{2a_t} = \frac{e^{2a_t}}{\gamma_t^2}, \quad (269)$$

$$2887 \quad e^{-2a_t} - 1 = \gamma_t^{-2}, \quad (270)$$

$$2888 \quad e^{-2a_t} = \gamma_t^{-2} + 1, \quad (271)$$

$$2889 \quad -2a_t = \log(\gamma_t^{-2} + 1). \quad (272)$$

2890 Then by substituting in the definition of a_t and letting γ denote the variable produced by γ_t we have
 2891

$$2892 \quad \frac{\beta_1 - \beta_0}{2} t^2 + \beta_0 t - \log(\gamma^{-2} + 1) = 0. \quad (273)$$

2893 We then use the quadratic formula to find the roots of the polynomial of t to find
 2894

$$2895 \quad t = \frac{-\beta_0 \pm \sqrt{\beta_0^2 + 2(\beta_1 - \beta_0) \log(\gamma^{-2} + 1)}}{\beta_1 - \beta_0}. \quad (274)$$

2896 Since $t \in [0, 1]$ we only take the positive root and thus
 2897

$$2898 \quad t = \frac{-\beta_0 + \sqrt{\beta_0^2 + 2(\beta_1 - \beta_0) \log(\gamma^{-2} + 1)}}{\beta_1 - \beta_0}. \quad (275)$$

2899 \square

2900 **Corollary H.1.1** (Inverse function of χ_t for linear noise schedule). *It follows by a straightforward
 2901 substitution from Proposition H.1 that t_χ can be written as*

$$2902 \quad t_\chi(\chi) = \frac{-\beta_0 + \sqrt{\beta_0^2 + 2(\beta_1 - \beta_0) \log(\chi^2 + 1)}}{\beta_1 - \beta_0}. \quad (276)$$

2903 **Corollary H.1.2** (Inverse function of ϱ_t for linear noise schedule). *It follows by a straightforward
 2904 substitution from Proposition H.1 that t_ϱ can be written as*

$$2905 \quad t_\varrho(\varrho) = \frac{-\beta_0 + \sqrt{\beta_0^2 + 2(\beta_1 - \beta_0) \log(\varrho^{-1} + 1)}}{\beta_1 - \beta_0}. \quad (277)$$

2916 H.1.2 SCALED LINEAR SCHEDULE
29172918 The *scaled linear schedule* is used widely by *latent diffusion models* (LDMs) (Rombach et al., 2022)
2919 and takes the discrete form of

2920
$$\beta_n = \left(\sqrt{\hat{\beta}_0} + \frac{n-1}{N-1} \left(\sqrt{\hat{\beta}_1} - \sqrt{\hat{\beta}_0} \right) \right)^2. \quad (278)$$

2922

2923 Thus following a similar approach to Song et al. (2021b) we write the scaled linear schedule as a
2924 function of t ,

2925
$$\beta_t = (\beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0)t^2 + 2t(\sqrt{\beta_1\beta_0} - \beta_0) + \beta_0. \quad (279)$$

2926

2927 Then using Equation (263) we find the noise schedule (α_t, σ_t) to be defined as

2928
$$\alpha_t = \exp \left(-\frac{\beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0}{6}t^3 - \frac{\sqrt{\beta_1\beta_0} - \beta_0}{2}t^2 - \frac{\beta_0}{2}t \right), \quad (280)$$

2929
2930
$$\sigma_t = \sqrt{1 - \alpha_t^2}.$$

2931

2932 Next we will derive the inverse function for γ_t 2933 **Proposition H.2** (Inverse function of γ_t for scaled linear noise schedule). *For the scaled linear noise*
2934 *schedule commonly used by LDMs (Rombach et al., 2022) the inverse function of γ_t denoted t_γ can*
2935 *be expressed in closed form as*

2936
$$t_\gamma(\gamma) = \frac{\beta_0 - \sqrt{\beta_1\beta_0} - \sqrt[3]{2(\sqrt{\beta_1\beta_0} - \beta_0)^3 - 3\beta_0\Delta(\sqrt{\beta_1\beta_0} - \beta_0) - 3\Delta^2 \log(\gamma^{-2} + 1)}}{\Delta}, \quad (281)$$

2937

2938 where

2939
$$\Delta = \beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0. \quad (282)$$

2940

2941 *Proof.* Let α_t be denoted by $\alpha_t = e^{a_t}$ where

2943
$$a_t = -\frac{\beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0}{6}t^3 - \frac{\sqrt{\beta_1\beta_0} - \beta_0}{2}t^2 - \frac{\beta_0}{2}t. \quad (283)$$

2944

2945 Then by definition of γ_t we can write

2946
$$\gamma_t = \frac{e^{a_t}}{\sqrt{1 - e^{2a_t}}}, \quad (284)$$

2947

2948 and with a little more algebra we find

2949
$$\sqrt{1 - e^{2a_t}} = \frac{e^{a_t}}{\gamma_t}, \quad (285)$$

2950

2951
$$1 - e^{2a_t} = \frac{e^{2a_t}}{\gamma_t^2}, \quad (286)$$

2952

2953
$$e^{-2a_t} - 1 = \gamma_t^{-2}, \quad (287)$$

2954

2955
$$e^{-2a_t} = \gamma_t^{-2} + 1, \quad (288)$$

2956

2957
$$-2a_t = \log(\gamma_t^{-2} + 1). \quad (289)$$

2958

2959 Then by substituting in the definition of a_t and letting γ denote the variable produced by γ_t we have

2960
$$\frac{\beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0}{3}t^3 + (\sqrt{\beta_1\beta_0} - \beta_0)t^2 + \beta_0t - \log(\gamma^{-2} + 1) = 0. \quad (290)$$

2961

2963 We then use the cubic formula (Cardano, 1545) to find the roots of the polynomial of t . The only real
2964 root is given by

2965
$$t_\gamma(\gamma) = \frac{\beta_0 - \sqrt{\beta_1\beta_0} - \sqrt[3]{2(\sqrt{\beta_1\beta_0} - \beta_0)^3 - 3\beta_0\Delta(\sqrt{\beta_1\beta_0} - \beta_0) - 3\Delta^2 \log(\gamma^{-2} + 1)}}{\Delta}, \quad (291)$$

2966

2967 where

2968
$$\Delta = \beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0. \quad (292)$$

2969

□

2970
2971 **Corollary H.2.1** (Inverse function of χ_t for scaled linear noise schedule). *It follows by a straightfor-
2972 ward substitution from Proposition H.2 that t_χ can be written as*
2973

$$2974 t_\chi(\chi) = \frac{\beta_0 - \sqrt{\beta_1\beta_0} - \sqrt[3]{2(\sqrt{\beta_1\beta_0} - \beta_0)^3 - 3\beta_0\Delta(\sqrt{\beta_1\beta_0} - \beta_0) - 3\Delta^2 \log(\chi^2 + 1)}}{\Delta}, \quad (293)$$

2975 where
2976

$$2977 \Delta = \beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0. \quad (294)$$

2978 **Corollary H.2.2** (Inverse function of ϱ_t for scaled linear noise schedule). *It follows by a straightfor-
2979 ward substitution from Proposition H.2 that t_ϱ can be written as*
2980

$$2981 t_\varrho(\varrho) = \frac{\beta_0 - \sqrt{\beta_1\beta_0} - \sqrt[3]{2(\sqrt{\beta_1\beta_0} - \beta_0)^3 - 3\beta_0\Delta(\sqrt{\beta_1\beta_0} - \beta_0) - 3\Delta^2 \log(\varrho^{-1} + 1)}}{\Delta}, \quad (295)$$

2983 where
2984

$$2985 \Delta = \beta_1 - 2\sqrt{\beta_1\beta_0} + \beta_0. \quad (296)$$

2986 H.2 SOME OTHER INVERSE FUNCTIONS 2987

2988 **Gamma to sigma.** Additionally, we need to be able to extract the weighting terms from the time
2989 integration variable. For the ODE case we need the function $\sigma_\gamma(\gamma)$ which describes the map $\gamma \mapsto \sigma$.
2990 By the definition of γ we have

$$2991 \gamma = \frac{\alpha}{\sigma}, \quad (297)$$

$$2994 \gamma \stackrel{(i)}{=} \frac{\sqrt{1 - \sigma^2}}{\sigma}, \quad (298)$$

$$2996 \sigma\gamma = \sqrt{1 - \sigma^2}, \quad (299)$$

$$2997 \sigma^2\gamma^2 = 1 - \sigma^2, \quad (300)$$

$$2999 \sigma^2\gamma^2 = 1 - \sigma^2, \quad (301)$$

$$3000 \gamma^2 = \sigma^{-2} - 1, \quad (302)$$

$$3001 \gamma^2 + 1 = \sigma^{-2}, \quad (303)$$

$$3003 \sigma^2 = \frac{1}{\gamma^2 + 1} \quad (304)$$

$$3005 \sigma_\gamma(\gamma) = \frac{1}{\sqrt{\gamma^2 + 1}}, \quad (305)$$

3007 where (i) hold by $\sigma^2 = 1 - \alpha^2$ for VP type diffusion SDEs.
3008

3009 **Rho to sigma over gamma.** Likewise, for the SDE case we need the function which maps $\varrho \mapsto \frac{\sigma}{\gamma}$.
3010 Recall that (note we drop the subscript t for the derivation)

$$3012 \varrho = \frac{\alpha^2}{\sigma^2}, \quad (306)$$

3014 thus we have
3015

$$3016 \varrho \stackrel{(i)}{=} \frac{\alpha^2}{1 - \alpha^2}, \quad (307)$$

$$3018 (1 - \alpha^2)\varrho = \alpha^2, \quad (308)$$

$$3020 \alpha^{-2} - 1 = \varrho^{-1}, \quad (309)$$

$$3021 \alpha^{-2} = \varrho^{-1} + 1, \quad (310)$$

$$3023 \alpha = \frac{1}{\sqrt{\varrho^{-1} + 1}}, \quad (311)$$

3024 where (i) hold by $\sigma^2 = 1 - \alpha^2$ for VP type diffusion SDEs. Then we can write
 3025

$$\frac{\sigma}{\gamma} = \frac{\sigma^2}{\alpha}, \quad (312)$$

$$= \frac{\sigma^2}{\alpha} \frac{\alpha}{\alpha}, \quad (313)$$

$$= \frac{\sigma^2}{\alpha^2} \alpha, \quad (314)$$

$$= \varrho^{-1} \alpha, \quad (315)$$

$$= \frac{1}{\rho \sqrt{\rho^{-1} + 1}}. \quad (316)$$

3036 **Chi to alpha.** Lastly, for the noise prediction models we need the map $\chi \mapsto \alpha$ denoted $\alpha_\chi(\chi)$. By
 3037 definition of χ we have
 3038

$$\chi = \frac{\sigma}{\alpha}, \quad (317)$$

$$\chi \stackrel{(i)}{=} \frac{\sqrt{1 - \alpha^2}}{\alpha}, \quad (318)$$

$$\alpha_\chi(\chi) \stackrel{(ii)}{=} \frac{1}{\sqrt{\chi^2 + 1}}, \quad (319)$$

3045 where (i) hold by $\sigma^2 = 1 - \alpha^2$ for VP type diffusion SDEs and (ii) holds by the derivation for $\sigma_\gamma(\gamma)$
 3046 *mutatis mutandis*.
 3047

3048 H.3 BROWNIAN MOTION

3050 We used the Brownian interval (Kidger et al., 2021) provided by the `torchsde` library. In general
 3051 we would recommend the virtual Brownian tree from Jelinčić et al. (2024) over the Brownian interval,
 3052 an implementation of this can be found in the `diffjax` library. However, as our code base made
 3053 extensive used of prior projects developed in pytorch and `diffjax` is a jax library it made more
 3054 sense to use `torchsde` for this project.
 3055

3056 I EXPERIMENTAL DETAILS

3058 We provide additional details for the empirical studies conducted in Section 5. *N.B.*, for all ex-
 3059 periments we used fixed random seeds between the different software components to ensure a fair
 3060 comparision.
 3061

3062 I.1 UNCONDITIONAL IMAGE GENERATION

3064 I.1.1 DIFFUSION MODEL

3065 We make use of a pre-trained DDPM (Ho et al., 2020) model trained on the CelebA-HQ 256×256
 3066 dataset (Karras et al., 2018). The linear noise schedule from (Ho et al., 2020) is given as
 3067

$$\beta_i = \frac{\hat{\beta}_0}{T} + \frac{i-1}{T(T-1)}(\hat{\beta}_1 - \hat{\beta}_0). \quad (320)$$

3070 We convert this into a continuous time representation via the details in Appendix H.1 following Song
 3071 et al. (2021b). For this experiment we used $\hat{\beta}_0 = 0.0001$ and $\hat{\beta}_1 = 0.2$. To ensure numerical stability
 3072 due to $\frac{1}{\sigma_t}$ terms we solve the probability flow ODE in reverse-time on the time interval $[\epsilon, 1]$ with
 3073 $\epsilon = 0.0002$. This is a common choice to make in practice see Song et al. (2023).
 3074

3075 I.1.2 METRICS

3076 We use several metrics to assess the performance in unconditional image generation following Stein
 3077 et al. (2023) by using a DINOv2 feature extractor (Oquab et al., 2023), all of which are calculated

3078 using the 10k generated samples and 30k real samples from the CelebA-HQ dataset. Throughout this
 3079 section we will let $\{\mathbf{x}_i\}_{i=1}^n$ denote an empirical distribution drawn from our generated distribution
 3080 \mathbb{P}_θ and let $\{\hat{\mathbf{x}}_i\}_{i=1}^m$ denote an empirical distribution drawn from the data distribution \mathbb{P}_{data} .
 3081

3082 **FD.** The *Fréchet distance* (FD) (Dowson & Landau, 1982) is measured using the sample mean and
 3083 covariance of the real \mathbb{P}_{data} and generated \mathbb{P}_θ distributions denoted

$$3084 \text{FD}(\mathbb{P}_{data} \| \mathbb{P}_\theta) = \|\mu_{data} - \mu_\theta\|_2^2 + \text{Tr} \left(\Sigma_{data} + \Sigma_\theta - 2(\Sigma_{data} \Sigma_\theta)^{\frac{1}{2}} \right), \quad (321)$$

3086 where (μ, Σ) denote the sample mean and covariances. This metric corresponds to the 2-
 3087 Wasserstein distance between two multivariate Gaussians and is thus a valid metric between the first
 3088 two moments. Heusel et al. (2017) popularized the use of this metric within the feature layer of an
 3089 Inception-V3 network (Szegedy et al., 2016) to assess the fidelity of unconditional image generation,
 3090 this metric is referred to as the *Fréchet inception distance* or FID. Recent works have challenged the
 3091 use of the Inception-V3 network as the feature extractor (Stein et al., 2023; Jayasumana et al., 2024;
 3092 Kynkänniemi et al., 2023) showing that the Inception-V3 network is poorly suited for capturing a
 3093 semantic view of images which correlates well to human judgment. In particular, Stein et al. (2023)
 3094 shows that using DINOv2 (Oquab et al., 2023) for the feature extractor results in a metric which is
 3095 significantly more aligned with human judgment.

3096 **FD_∞.** FD_∞ proposed by Chong & Forsyth (2020) is a modification of FD which aims to remove
 3097 the inherent bias induced by using a finite number of empirical samples. The samples is determined
 3098 by evaluating FD over 15 regular intervals over the number of total samples and fitting a linear trend
 3099 to the 15 data points to infer a trend for FD as the number of empirical samples, $N \rightarrow \infty$.

3100 **Precision, recall, density and coverage.** The density metric (Naeem et al., 2020) is used as a
 3101 proxy to measure sample fidelity and improves upon the earlier precision metric (Kynkänniemi
 3102 et al., 2019; Sajjadi et al., 2018). The metric is based upon nearest neighbours distance computed in a
 3103 representation space and counts how many real-sample neighbourhood balls contain the generated
 3104 sample. Likewise to quantify sample diversity we use the coverage metric (Naeem et al., 2020) which
 3105 improves upon the earlier recall metric (Kynkänniemi et al., 2019; Sajjadi et al., 2018). The density
 3106 metric is given by

$$3107 \text{density}(\mathbb{P}_{data}, \mathbb{P}_\theta) = \frac{1}{kn} \sum_{i=1}^n \sum_{j=1}^m 1_{B(\hat{\mathbf{x}}_j, \delta^k(\hat{\mathbf{x}}_j))}(\mathbf{x}_i), \quad (322)$$

3110 where $1_A(\cdot)$ denotes the indicator function for set A , $B(\mathbf{x}, r)$ constructs a Euclidean ball centered at
 3111 \mathbf{x} with radius r , and $\delta^k(\hat{\mathbf{x}}_j)$ is the distance to the k -th nearest neighbour in $\{\hat{\mathbf{x}}_i\}_{i=1}^m$, excluding itself.
 3112 The precision metric is given by

$$3113 \text{precision}(\mathbb{P}_{data}, \mathbb{P}_\theta) = \frac{1}{n} \sum_{i=1}^n 1_{\bigcup_{j=1}^m B(\hat{\mathbf{x}}_j, \delta^k(\hat{\mathbf{x}}_j))}(\mathbf{x}_i). \quad (323)$$

3116 Similarly, coverage is given by

$$3117 \text{coverage}(\mathbb{P}_{data}, \mathbb{P}_\theta) = \frac{1}{m} \sum_{j=1}^m \max_{i=1, \dots, n} 1_{B(\hat{\mathbf{x}}_j, \delta^k(\hat{\mathbf{x}}_j))}(\mathbf{x}_i). \quad (324)$$

3120 Likewise, the recall metric is given by

$$3121 \text{recall}(\mathbb{P}_{data}, \mathbb{P}_\theta) = \frac{1}{m} \sum_{j=1}^m 1_{\bigcup_{i=1}^n B(\hat{\mathbf{x}}_j, \delta^k(\hat{\mathbf{x}}_j))}(\hat{\mathbf{x}}_j). \quad (325)$$

3124 We used $k = 5$ and 10k samples throughout, as standard.

3125 **On reporting.** When reporting on these metrics like in Table 1 we use **bold font** to denote the
 3126 best performance with a 1% error range. More formally, suppose we have a series of n data points
 3127 $\{\mathbf{x}_i\}_{i=1}^n$ that is totally ordered by some relation R . We say will denote a query point x_i with **bold
 3128 font** if the *range-normalized absolute percentage error* is less than $\epsilon > 0$, i.e.,

$$3129 \frac{|\max_j x_j - x_i|}{\max_j x_j - \min_k x_k} < \epsilon. \quad (326)$$

3131 In our experiments we report $\epsilon = 0.01$.

3132 I.1.3 HYPERPARAMETERS
3133

3134 We follow the suggestion of [Wallace et al. \(2023\)](#) and report results with EDICT using the hyper-
3135 parameter $p = 0.93$. For BDIA, the original paper recommends $\gamma = 1.0$ for unconditional image
3136 generation ([Zhang et al., 2024](#), Section 6.1). However, we found $\gamma = 0.5$ to yield better performance,
3137 this corroborates with the findings of [Wang et al. \(2024\)](#).

3138 I.2 CONDITIONAL IMAGE GENERATION
31393140 I.2.1 DIFFUSION MODEL
3141

3142 We make use of Stable Diffusion v1.5 ([Rombach et al., 2022](#)) a pre-trained *latent diffusion model*
3143 (LDM) model. We also use the scaled linear noise schedule given as

$$3144 \beta_i = \left(\sqrt{\frac{\hat{\beta}_0}{T}} + \frac{i-1}{\sqrt{T(T-1)}} \left(\sqrt{\hat{\beta}_1} - \sqrt{\hat{\beta}_0} \right) \right)^2. \quad (327)$$

3145 We convert this into a continuous time representation via the details in Appendix H.1 following [Song](#)
3146 [et al. \(2021b\)](#). For this experiment we used $\hat{\beta}_0 = 0.00085$ and $\hat{\beta}_1 = 0.012$. To ensure numerical
3147 stability due to $\frac{1}{\sigma_t}$ terms we solve the probability flow ODE in reverse-time on the time interval $[\epsilon, 1]$
3148 with $\epsilon = 0.0002$. This is a common choice to make in practice see [Song et al. \(2023\)](#).

3149 **Numerical schemes.** We set the last two steps of Rex schemes to be either Euler or Euler-Maruyama
3150 for better stability near time 0.

3151 I.2.2 METRICS
3152

3153 As mentioned in the main paper we use the CLIP Score ([Hessel et al., 2021](#)) PickScore ([Kirstain](#)
3154 [et al., 2023](#)), and Image Reward metrics ([Xu et al., 2023](#)) to asses the ability of the text-to-image
3155 conditional generation task. We calculate each by comparing the sampled image and the given text
3156 prompt used to produce the image. We then report the average over the 1000 samples.

3157 **CLIP score.** The CLIP score measures the cosine similarity between the text and visual embeddings
3158 with pretrained CLIP model ([Radford et al., 2021](#)) denoted as

$$3159 \text{CLIPScore}(\mathbf{x}, \mathbf{c}) = \max \left\{ \frac{\langle \mathcal{E}_I(\mathbf{x}), \mathcal{E}_C(\mathbf{c}) \rangle}{\|\mathcal{E}_I(\mathbf{x})\| \|\mathcal{E}_C(\mathbf{c})\|}, 0 \right\}, \quad (328)$$

3160 where $\mathcal{E}_I : \mathbb{R}^d \rightarrow V$ is the image embedder and $\mathcal{E}_C : \mathbb{R}^{d'} \rightarrow V$ is the caption embedder; and where
3161 \mathbf{x} is the query image and \mathbf{c} is the query caption. Thus this metric aims to measure how well our
3162 generated images align with their prompt. In particular, we use the ViT-L/14 backbone trained by
3163 OpenAI.

3164 **PickScore.** Similar to CLIP score, PickScore finetunes a CLIP-H model on their proposed Pick-a-
3165 Pic dataset which purportedly aligns better with human preference over CLIP score.

3166 **Image Reward.** Image Reward ([Xu et al., 2023](#)) is the newest of the three metrics and uses BLIP
3167 ([Li et al., 2022](#)) over CLIP as the backbone and finetunes the model using reward model training. The
3168 resulting metrics achieves state-of-the-art alignment with human preferences.

3169 **On reporting.** When reporting on these metrics like in Table 2 we use **bold font** to denote the best
3170 performance with a 1% error range. In our experiments we report $\epsilon = 0.01$.

3171 I.2.3 HYPERPARAMETERS
3172

3173 We follow the suggestion of [Wallace et al. \(2023\)](#) and report results with EDICT using the hyper-
3174 parameter $p = 0.93$. For BDIA, the original paper recommends $\gamma = 0.5$ for text-to-image generation
3175 ([Zhang et al., 2024](#), Section 6.1). We also ran BDIA with $\gamma = 0.96$ as suggested by [Wang et al.](#)
3176 ([2024](#)).

3186 I.3 INTERPOLATION
3187

3188

3189 **Diffusion model.** We make use of a pre-trained DDPM (Ho et al., 2020) model trained on the
3190 CelebA-HQ 256×256 dataset (Karras et al., 2018). We used linear noise schedule from (Ho et al.,
3191 2020). We convert this into a continuous time representation via the details in Appendix H.1 following
3192 Song et al. (2021b). For this experiment we used $\hat{\beta}_0 = 0.0001$ and $\hat{\beta}_1 = 0.2$. For the face pairings
3193 we followed Blasingame & Liu (2024a;c) and used the FRLL (DeBruine & Jones, 2017) dataset.3194 Notably, we used the noise prediction parameterization rather than data prediction as we found that it
3195 performed better for editing. This is likely due to the singularity of the $\frac{1}{\sigma_t}$ terms as $t \rightarrow 0$. Within this
3196 parameterization we could use the time interval $[0, 1]$ instead of $[\epsilon, 1]$ like in previous experiments
3197 with data prediction models.

3198

3199

3200

3201 I.4 HARDWARE
3202

3203

3204 All experiments were run using a single NVIDIA H100 80 GB GPU.

3205

3206

3207

3208 I.5 REPOSITORIES
3209

3210

3211 In our empirical studies we made use of the following resources and repositories:
3212

3213

- 1.
- [google/ddpm-celebahq-256](#)
- (DDPM Model)
-
- 3214
-
- 2.
- [stable-diffusion-v1-5/stable-diffusion-v1-5](#)
- (Stable Diffusion v1.5)
-
- 3215
-
- 3.
- [zituitui/BELM](#)
- (Implementation of BELM, EDICT, and BDIA)
-
- 3216
-
- 4.
- [google-research/torchsde](#)
- (Brownian Interval)
-
- 3217
-
- 3218
-
- 5.
- [layer6ai-labs/dgm-eval](#)
- (FD,
- FD_∞
- , KD, Density, and Coverage metrics)
-
- 3219
-
- 3220
-
- 3221
-
- 6.
- [torchmetrics](#)
- (CLIP score)
-
- 3222
-
- 3223
-
- 3224
-
- 7.
- [zai-org/ImageReward](#)
- (Image Reward)
-
- 3225
-
- 3226
-
- 3227
-
- 3228
-
- 3229
-
- 3230
-
- 3231
-
- 3232
-
- 3233
-
- 3234

3235

J CODE
3236

3237

3238

3239

In this section we provide some example code for the core components of the model to help illustrate
the core ideas.

3240

Code J.1: Rex forward step

```

3242 def rex_forward(model_func, scheduler, xt, xt_hat, timesteps, solver='euler', coupling=0.999,
3243     low_order_final_n_steps=0, bm=None, pred_type='data', sched_type='linear'):
3244     """
3245     Based on McCallum & Foster's reversible ODE solver and adapted for diffusion models.
3246     """
3247
3248     # Choose underlying solver
3249     is_sde = (solver in SDE_SOLVERS)
3250     psi = SOLVER_DICT[solver]
3251
3252     if not is_sde:
3253         _t_to_gamma, _gamma_to_t = _gen_time_funcs(sched_type=sched_type, pred_type=pred_type)
3254         t_to_gamma = _t_to_gamma
3255         gamma_to_t = _gamma_to_t
3256         gamma_to_sigma = _gamma_to_sigma if pred_type == 'data' else _chi_to_alpha
3257     else:
3258         _t_to_rho, _rho_to_t = _gen_time_funcs(sched_type=sched_type, rho=True, pred_type=pred_type)
3259         t_to_gamma = _t_to_rho
3260         gamma_to_t = _rho_to_t
3261         gamma_to_sigma = _rho_to_siggamma if pred_type == 'data' else _chi_to_alpha
3262
3263     # create timesteps in gamma, alt  $\gamma^2 = \rho$  for SDEs
3264     gammas = t_to_gamma(scheduler, timesteps)
3265
3266     # Push gamma reparam back to time t and convert noise pred to data pred
3267     if pred_type == 'data':
3268         wrap_model = lambda gamma, x: _convert_noise_to_data(scheduler, model_func,
3269             gamma_to_t(scheduler, gamma), x, sched_type=sched_type)
3270     else:
3271         p = 2 if is_sde else 1
3272         wrap_model = lambda gamma, x: p * model_func(gamma_to_t(scheduler, gamma), x)
3273
3274     xt.to(torch.float32)
3275     xt_hat.to(torch.float32)
3276
3277     for n in tqdm(range(len(gammas)-1)):
3278         gamma_n = gammas[n]
3279         gamma_n1 = gammas[n+1]
3280         h = gamma_n1 - gamma_n
3281
3282         sigma_n = gamma_to_sigma(gamma_n)
3283         sigma_n1 = gamma_to_sigma(gamma_n1)
3284
3285         if n < (len(gammas) - 1 - low_order_final_n_steps):
3286             if not is_sde:
3287                 _psi = lambda t, x, h: psi(wrap_model, t, x, h)
3288             else:
3289                 _psi = lambda t, x, h: psi(wrap_model, t, x, h, bm, pred_type=pred_type)
3290             else:
3291                 if not is_sde:
3292                     _psi = lambda t, x, h: euler(wrap_model, t, x, h)
3293                 else:
3294                     _psi = lambda t, x, h: euler_maruyama(wrap_model, t, x, h, bm, pred_type=pred_type)
3295
3296             xt = (sigma_n1 / sigma_n) * (coupling * xt + (1-coupling) * xt_hat) + sigma_n1 * _psi(gamma_n,
3297                 xt_hat, h)
3298             xt_hat = (sigma_n1 / sigma_n) * xt_hat - sigma_n1 * _psi(gamma_n1, xt, -h)
3299
3300     return xt, xt_hat

```

```

3294
3295     Code J.2: Rex backward step
3296
3297     def rex_backward(model_func, scheduler, xt, xt_hat, timesteps, solver='euler', coupling=0.999,
3298     ↪ low_order_final_n_steps=0, bm=None, pred_type='data', sched_type='linear'):
3299     """
3300     Based on McCallum & Foster's reversible ODE solver and adapted for diffusion models.
3301     """
3302
3303     # Choose underlying solver
3304     is_sde = (solver in SDE_SOLVERS)
3305     psi = SOLVER_DICT[solver]
3306
3307     if not is_sde:
3308         _t_to_gamma, _gamma_to_t = _gen_time_funcs(sched_type=sched_type, pred_type=pred_type)
3309         t_to_gamma = _t_to_gamma
3310         gamma_to_t = _gamma_to_t
3311         gamma_to_sigma = _gamma_to_sigma if pred_type == 'data' else _chi_to_alpha
3312     else:
3313         _t_to_rho, _rho_to_t = _gen_time_funcs(sched_type=sched_type, rho=True, pred_type=pred_type)
3314         t_to_rho = _t_to_rho
3315         rho_to_t = _rho_to_t
3316         gamma_to_sigma = _rho_to_siggamma if pred_type == 'data' else _chi_to_alpha
3317
3318     # create timesteps in gamma, alt gamma^2 = rho for SDEs
3319     gammas = t_to_gamma(scheduler, timesteps)
3320
3321     # Push gamma reparam back to time t and convert noise pred to data pred
3322     if pred_type == 'data':
3323         wrap_model = lambda gamma, x: _convert_noise_to_data(scheduler, model_func,
3324         ↪ gamma_to_t(scheduler, gamma), x, sched_type=sched_type)
3325     else:
3326         p = 2 if is_sde else 1
3327         wrap_model = lambda gamma, x: p * model_func(gamma_to_t(scheduler, gamma), x)
3328
3329     xt.to(torch.float32)
3330     xt_hat.to(torch.float32)
3331
3332     coupling_inv = 1. / coupling
3333
3334     for n in tqdm(range(len(gammas) - 2, -1, -1)):
3335         gamma_n = gammas[n]
3336         gamma_n1 = gammas[n+1]
3337         h = gamma_n1 - gamma_n
3338
3339         sigma_n = gamma_to_sigma(gamma_n)
3340         sigma_n1 = gamma_to_sigma(gamma_n1)
3341
3342         if n < (len(gammas) - 1 - low_order_final_n_steps):
3343             if not is_sde:
3344                 _psi = lambda t, x, h: psi(wrap_model, t, x, h)
3345             else:
3346                 _psi = lambda t, x, h: psi(wrap_model, t, x, h, bm, pred_type=pred_type)
3347         else:
3348             if not is_sde:
3349                 _psi = lambda t, x, h: euler(wrap_model, t, x, h)
3350             else:
3351                 _psi = lambda t, x, h: euler_maruyama(wrap_model, t, x, h, bm, pred_type=pred_type)
3352
3353         xt_hat = (sigma_n / sigma_n1) * xt_hat + sigma_n * _psi(gamma_n1, xt, -h)
3354         xt = (sigma_n / sigma_n1) * (coupling_inv * xt) + (1 - coupling_inv) * xt_hat - sigma_n *
3355         ↪ coupling_inv * _psi(gamma_n, xt_hat, h)
3356
3357     return xt, xt_hat
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4209
4210
4211
4212
4213
4214
4215
4216
4217
4217
4218
4219
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4229
4230
4231
4232
4233
4234
4235
4236
4237
4237
4238
4239
4239
4240
4241
4242
4243
4244
4245
4245
4246
4247
4247
4248
4249
4249
4250
4251
4252
4253
4254
4255
4255
4256
4257
4257
4258
4259
4259
4260
4261
4262
4263
4264
4264
4265
4266
4266
4267
4268
4268
4269
4269
4270
4271
4271
4272
4273
4273
4274
4275
4275
4276
4277
4277
4278
4278
4279
4279
4280
4281
4281
4282
4283
4283
4284
4285
4285
4286
4286
4287
4287
4288
4288
4289
4289
4290
4290
4291
4291
4292
4292
4293
4293
4294
4294
4295
4295
4296
4296
4297
4297
4298
4298
4299
4299
4300
4300
4301
4301
4302
4302
4303
4303
4304
4304
4305
4305
4306
4306
4307
4307
4308
4308
4309
4309
4310
4310
4311
4311
4312
4312
4313
4313
4314
4314
4315
4315
4316
4316
4317
4317
4318
4318
4319
4319
4320
4320
4321
4321
4322
4322
4323
4323
4324
4324
4325
4325
4326
4326
4327
4327
4328
4328
4329
4329
4330
4330
4331
4331
4332
4332
4333
4333
4334
4334
4335
4335
4336
4336
4337
4337
4338
4338
4339
4339
4340
4340
4341
4341
4342
4342
4343
4343
4344
4344
4345
4345
4346
4346
4347
4347
4348
4348
4349
4349
4350
4350
4351
4351
4352
4352
4353
4353
4354
4354
4355
4355
4356
4356
4357
4357
4358
4358
4359
4359
4360
4360
4361
4361
4362
4362
4363
4363
4364
4364
4365
4365
4366
4366
4367
4367
4368
4368
4369
4369
4370
4370
4371
4371
4372
4372
4373
4373
4374
4374
4375
4375
4376
4376
4377
4377
4378
4378
4379
4379
4380
4380
4381
4381
4382
4382
4383
4383
4384
4384
4385
4385
4386
4386
4387
4387
4388
4388
4389
4389
4390
4390
4391
4391
4392
4392
4393
4393
4394
4394
4395
4395
4396
4396
4397
4397
4398
4398
4399
4399
4400
4400
4401
4401
4402
4402
4403
4403
4404
4404
4405
4405
4406
4406
4407
4407
4408
4408
4409
4409
4410
4410
4411
4411
4412
4412
4413
4413
4414
4414
4415
4415
4416
4416
4417
4417
4418
4418
4419
4419
4420
4420
4421
4421
4422
4422
4423
4423
4424
4424
4425
4425
4426
4426
4427
4427
4428
4428
4429
4429
4430
4430
4431
4431
4432
4432
4433
4433
4434
4434
4435
4435
4436
4436
4437
4437
4438
4438
4439
4439
4440
4440
4441
4441
4442
4442
4443
4443
4444
4444
4445
4445
4446
4446
4447
4447
4448
4448
4449
4449
4450
4450
4451
4451
4452
4452
4453
4453
4454
4454
4455
4455
4456
4456
4457
4457
4458
4458
4459
4459
4460
4460
4461
4461
4462
4462
4463
4463
4464
4464
4465
4465
4466
4466
4467
4467
4468
4468
4469
4469
4470
4470
4471
4471
4472
4472
4473
4473
4474
4474
4475
4475
4476
4476
4477
4477
4478
4478
4479
4479
4480
4480
4481
4481
4482
4482
4483
4483
4484
4484
4485
4485
4486
4486
4487
4487
4488
4488
4489
4489
4490
4490
4491
4491
4492
4492
4493
4493
4494
4494
4495
4495
4496
4496
4497
4497
4498
4498
4499
4499
4500
4500
4501
4501
4502
4502
4503
4503
4504
4504
4505
4505
4506
4506
4507
4507
4508
4508
4509
4509
4510
4510
4511
4511
4512
4512
4513
4513
4514
4514
4515
4515
4516
4516
4517
4517
4518
4518
4519
4519
4520
4520
4521
4521
4522
4522
4523
4523
4524
4524
4525
4525
4526
4526
4527
4527
4528
4528
4529
4529
4530
4530
4531
4531
4532
4532
4533
4533
4534
4534
4535
4535
4536
4536
4537
4537
4538
4538
4539
4539
4540
4540
4541
4541
4542
4542
4543
4543
4544
4544
4545
4545
4546
4546
4547
4547
4548
4548
4549
4549
4550
4550
4551
4551
4552
4552
4553
4553
4554
4554
4555
4555
4556
4556
4557
4557
4558
4558
4559
4559
4560
4560
4561
4561
4562
4562
4563
4563
4564
4564
4565
4565
4566
4566
4567
4567
4568
4568
4569
4569
4570
4570
4571
4571
4572
4572
4573
4573
4574
4574
4575
4575
4576
4576
4577
4577
4578
4578
4579
4579
4580
4580
4581
4581
4582
4582
4583
4583
4584
4584
4585
4585
4586
4586
4587
4587
4588
4588
4589
4589
4590
4590
4591
4591
4592
4592
4593
4593
4594
4594
4595
4595
4596
4596
4597
4597
4598
4598
4599
4599
4600
4600
4601
4601
4602
4602
4603
4603
4604
4604
4605
4605
4606
4606
4607
4607
4608
4608
4609
4609
4610
4610
4611
4611
4612
4612
4613
4613
4614
4614
4615
4615
4616
4616
4617
4617
4618
4618
4619
4619
4620
4620
4621
4621
4622
4622
4623
4623
4624
4624
4625
4625
4626
4626
4627
4627
4628
4628
4629
4629
4630
4630
4631
4631
4632
4632
4633
4633
4634
4634
4635
4635
4636
4636
4637
4637
4638
4638
4639
4639
4640
4640
4641
4641
4642
4642
4643
4643
4644
4644
4645
4645
4646
4646
4647
4647
4648
4648
4649
4649
4650
4650
4651
4651
4652
4652
4653
4653
4654
4654
4655
4655
4656
4656
4657
4657
4658
4658
4659
4659
4660
4660
4661
4661
4662
4662
4663
4663
4664
4664
4665
4665
4666
4666
4667
4667
4668
4668
4669
4669
4670
4670
4671
4671
4672
4672
4673
4673
4674
4674
4675
4675
4676
4676
4677
4677
4678
4678
4679
4679
4680
4680
4681
4681
4682
4682
4683
4683
4684
4684
4685
4685
4686
4686
4687
4687
4688
4688
4689
4689
4690
4690
4691
4691
4692
4692
4693
4693
4694
4694
4695
4695
4696
4696
4697
4697
4698
4698
4699
4699
4700
4700
4701
4701
4702
4702
4703
4703
4704
4704
4705
4705
4706
4706
4707
4707
4708
4708
4709
4709
4710
4710
4711
4711
4712
4712
4713
4713
4714
4714
4715
4715
4716
4716
4717
4717
4718
4718
4719
4719
4720
4720
4721
4721
4722
4722
4723
4723
4724
4724
4725
4725
4726
4726
4727
4727
4728
4728
4729
4729
4730
4730
4731
4731
4732
4732
4733
4733
4734
4734
4735
4735
4736
4736
4737
4737
4738
4738
4739
4739
4740
4740
4741
4741
4742
4742
4743
4743
4744
4744
4745
4745
4746
4746
4747
4747
4748
4748
4749
4749
4750
4750
4751
4751
4752
4752
4753
4753
4754
4754
4755
4755
4756
4756
4757
4757
4758
4758
4759
4759
4760
4760
4761
4761
4762
4762
4763
4763
4764
4764
4765
4765
4766
4766
4767
4767
4768
4768
4769
4769
4770
4770
4771
4771
4772
4772
4773
4773
4774
4774
4775
4775
4776
4776
4777
4777
4778
4778
4779
4779
4780
4780
4781
4781
4782
4782
4783
4783
4784
4784
4785
4785
4786
4786
4787
4787
4788
4788
4789
4789
4790
4790
4791
4791
4792
4792
4793
4793
4794
4794
4795
4795
4796
4796
4797
4797
4798
4798
4799
4799
4800
4800
4801
4801
4802
4802
4803
4803
4804
4804
4805
4805
4806
4806
4807
4807
4808
4808
4809
4809
4810
4810
4811
4811
4812
4812
4813
4813
4814
4814
4815
4815
4816
4816
4817
4817
4818
4818
4819
4819
4820
4820
4821
4821
4822
4822
4823
4823
4824
4824
4825
4825
4826
4826
4827
4827
4828
4828
4829
4829
4830
4830
4831
4831
4832
4832
4833
4833
4834
4834
4835
4
```

```

3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401

```

Code J.3: ShARK

```

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

```

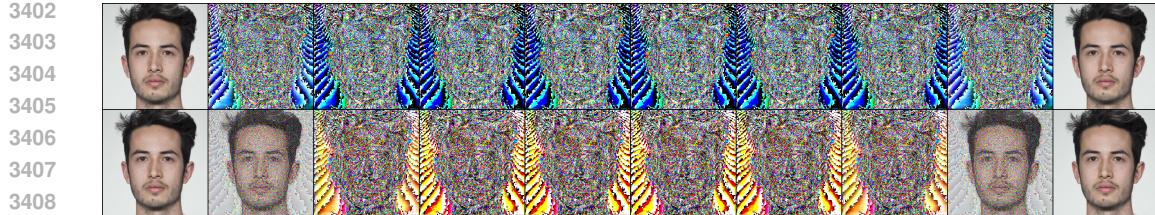


Figure 7: Inversion followed by sampling with Rex (Euler) 5 steps, $\zeta = 0.999$. Data prediction. Top row tracks \mathbf{x}_n , bottom row $\hat{\mathbf{x}}_n$.

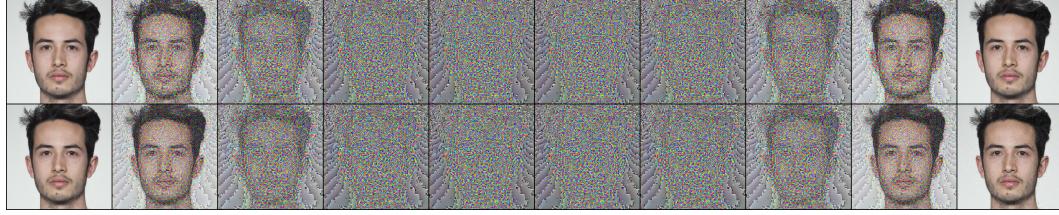


Figure 8: Inversion followed by sampling with Rex (Euler) 5 steps, $\zeta = 0.999$. Noise prediction. Top row tracks \mathbf{x}_n , bottom row $\hat{\mathbf{x}}_n$.

K VISUALIZATION OF INVERSION AND THE LATENT SPACE

We conduct a further qualitative study of the latent space produced by inversion and the impact various design parameters play. First in Figure 7 we show the process of inverting and then reconstructing a real sample. Notice that while the data prediction formulation worked great in sampling and still possesses the correct reconstruction, *i.e.*, it is still reversible, the latent space is all messed up. The variance of $(\mathbf{x}_n, \hat{\mathbf{x}}_n)$ tends to about 10^7 , many orders of magnitude too large! We did observe that raising $\zeta = 1 - 10^{-9}$ did help reduce this, but it was still relatively unstable. *N.B.*, these trends hold in a large number of discretization steps (we tested up to 250); however, for visualization purposes we chose fewer steps.

Conversely, the noise prediction formulation is much more stable, see Figure 8. The variance of $(\mathbf{x}_n, \hat{\mathbf{x}}_n)$ is on the right order of magnitude this time, however, there are strange artefacting and it is clear the latent variables are not normally distributed.

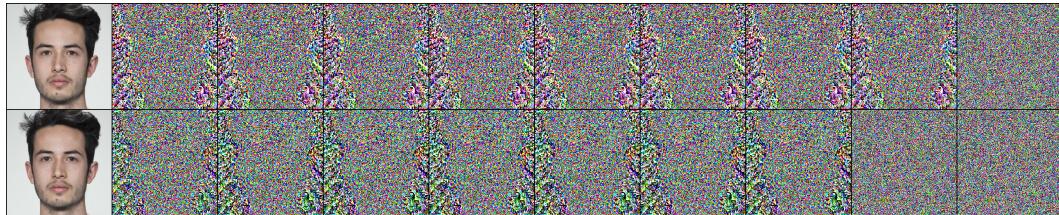


Figure 9: FAILURE CASE! Inversion followed by sampling with Rex (ShARK) 5 steps, $\zeta = 0.999$. Data prediction. Top row tracks x_n , bottom row \hat{x}_n .

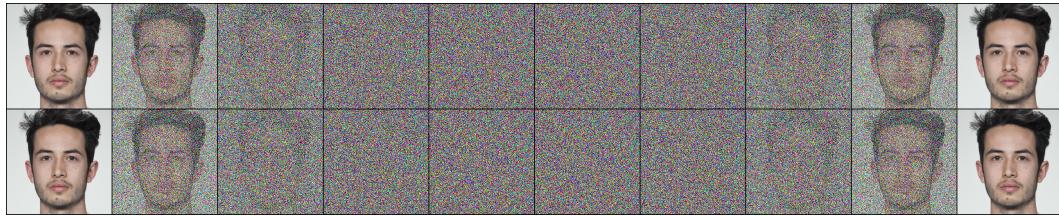


Figure 10: Inversion followed by sampling with Rex (ShARK) 5 steps, $\zeta = 0.999$. Noise prediction. Top row tracks x_n , bottom row \hat{x}_n .

Moving to the SDE case with ShARK in Figure 9, we see that the data prediction formulation is so unstable in forward-time that we ran into overflow errors and can no longer achieve algebraic reversibility. However, the noise parameterization with ShARK, see Figure 10, works very well with the latent variables appearing to be close to normally distributed.

L ADDITIONAL RESULTS

L.1 UNCONDITIONAL IMAGE GENERATION

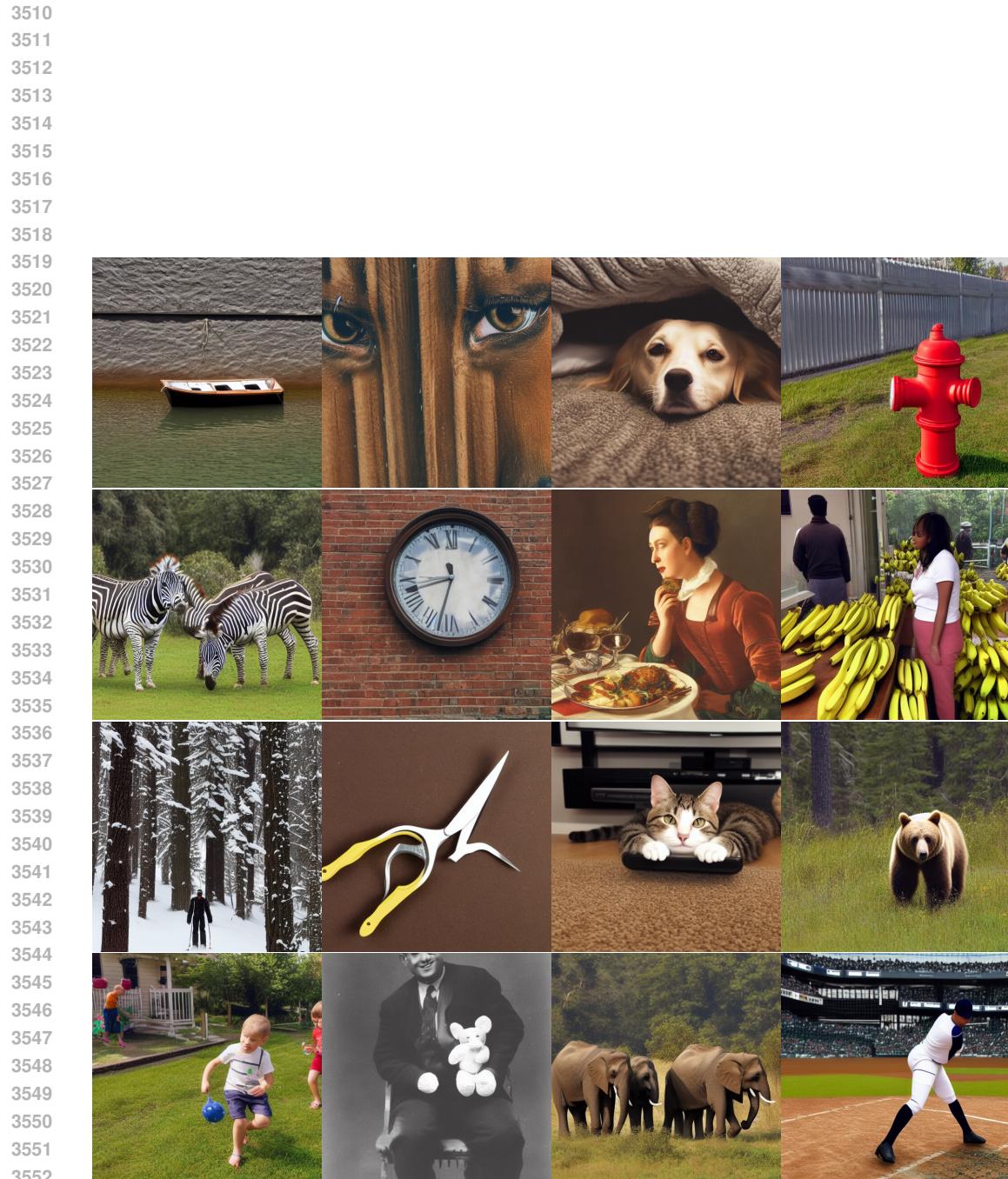
We present some additional ablations on the underlying solver for Rex in Table 4.

Table 4: Quantitative comparison of different underlying schemes Φ used in Rex in terms of FID (\downarrow for unconditional image generation with a pre-trained DDPM model on CelebA-HQ (256×256)).

Steps	Solver				
	Euler	Midpoint	RK4	Euler-Maruyama	ShARK
10	36.65	x	31.00	40.79	59.89
20	24.63	23.36	23.49	27.80	32.18
50	21.45	21.45	21.35	19.77	21.93

L.2 CONDITIONAL IMAGE GENERATION

We present some uncrated samples using Rex with various underlying solvers and discretization steps.



3553
 3554 Figure 11: Uncurated samples created using Rex (RK4) and Stable Diffusion v1.5 (512 × 512) and
 10 discretization steps.
 3555
 3556
 3557
 3558
 3559
 3560
 3561
 3562
 3563

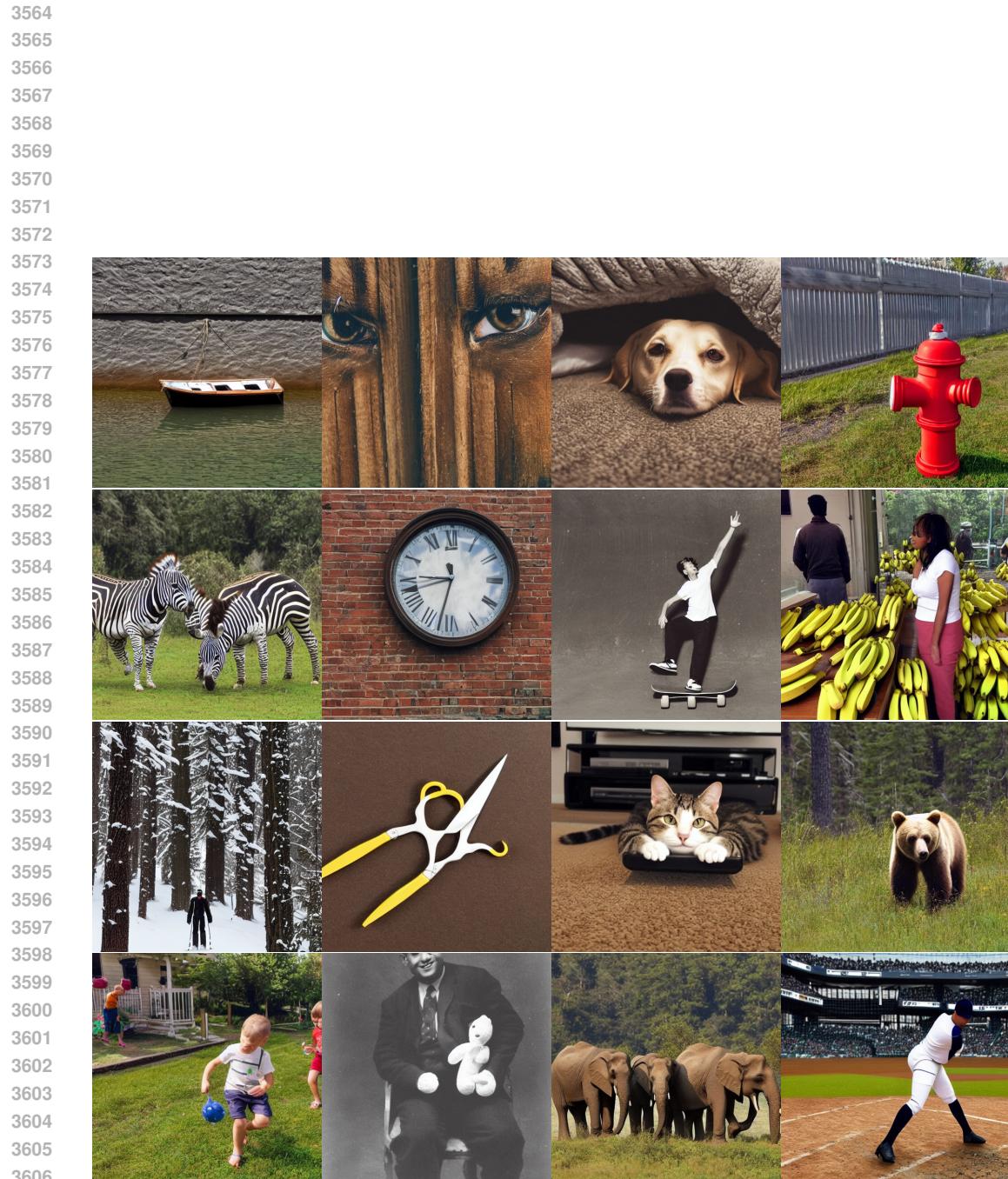


Figure 12: Uncurated samples created using Rex (RK4) and Stable Diffusion v1.5 (512 × 512) and 50 discretization steps.

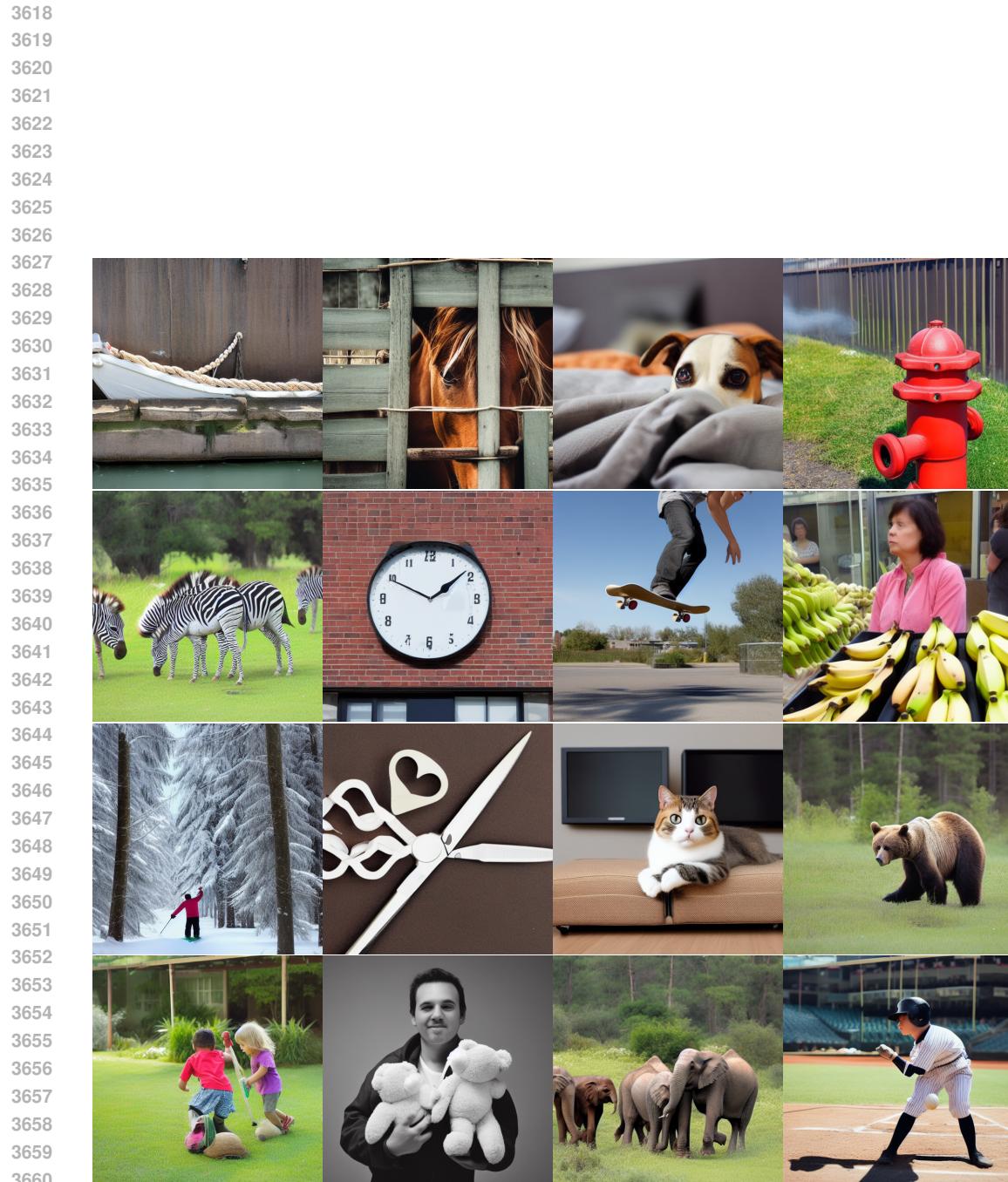


Figure 13: Uncurated samples created using Rex (ShARK) and Stable Diffusion v1.5 (512 × 512) and 10 discretization steps.

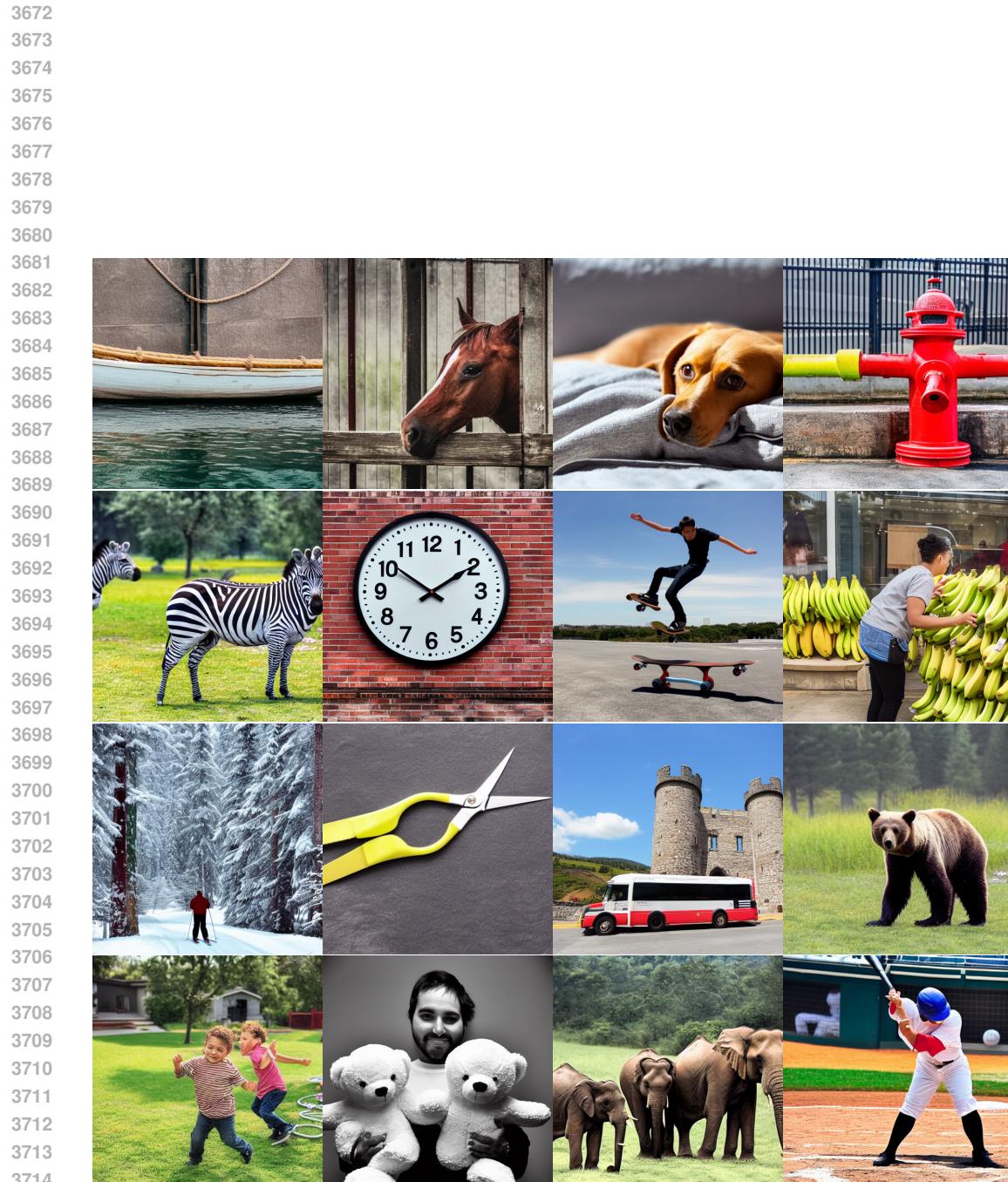


Figure 14: Uncurated samples created using Rex (ShARK) and Stable Diffusion v1.5 (512 × 512) and 50 discretization steps.

3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725