Practical and Asymptotically Exact Conditional Sampling in Diffusion Models

Anonymous Authors[1]

Abstract

Diffusion models have been successful on a range of conditional generation tasks including molecular design and text-to-image generation. However, these achievements have primarily depended on expensive, task-specific conditional training or error-prone heuristic approximations to them. Ideally, a conditional generation method should provide exact samples for a broad range of conditional distributions without requiring task-specific training. To this end, we introduce the *Twisted Diffusion Sampler*, or *TDS*, a sequential Monte Carlo (SMC) algorithm that targets the conditional distributions of diffusion models. The main idea is to use *twisting*, an SMC technique the enjoys good computational efficiency, to incorporate heuristic approximations without compromising asymptotic exactness. We study the properties of TDS on MNIST image inpainting and class-conditional generation tasks. TDS extends to Riemannian diffusion models, which are crucial for protein modeling. When applied to the motif-scaffolding problem, a core problem in protein design, TDS enables more flexible conditioning criteria than conditionally trained models, and provides state-of-the-art success rates on 9/12 problems in a benchmark set with scaffolds shorter than 100 residues.

1. Introduction

Conditional sampling is an essential primitive in the probabilistic machine learning toolkit. Generative models parameterize distributions $p_\theta(x)$ on data x. When combined with a likelihood $p(y \mid x)$, this implies conditional distributions $p_\theta(x \mid y)$ given conditioning information y. Conditional generation tasks that customize generative models' outputs to meet specified requirements can be formalized as conditional sampling problems. In protein design, $p_\theta(x)$ can represent a distribution of physically realizable protein structures, y a substructure that imparts a desired biochemical function, and samples from $p_\theta(x \mid y)$ are then physically realizable structures that contain the substructure of interest (e.g. Trippe et al., 2023; Watson et al., 2022). In computer vision, as another example, $p_\theta(x)$ is a distribution of images, y a classification label, and samples from $p_\theta(x \mid y)$ where y are then images likely to be classified as with label y (Ho and Salimans, 2022).

Diffusion models are a class of generative models that have demonstrated great successes in conditional generation tasks (Ho et al., 2020; Ramesh et al., 2022; Song et al., 2020; Watson et al., 2022). They parameterize complicated data distributions $p_\theta(x)$ through an iterative refinement process that builds up data from noise. All conditional generation approaches for diffusion models proceed by modifying the refinement process to account for conditioning at each step.

For example, conditionally trained diffusion models, which take additional conditioning information y as input (e.g. Ramesh et al., 2022), have provided most of the major successes of diffusion models to date. However, conditional training requires (i) assembling a large set of paired examples of the data and conditioning information (x, y), and (ii) designing and training a task-specific model when adapting to new conditioning tasks.

To help with this expense, a separate line of work uses heuristic approximations that directly operate on unconditional diffusion models. These approaches have had some success in *inpainting* problems (Meng et al., 2021; Song et al., 2020; Trippe et al., 2023), and other general inverse problems (Bansal et al., 2023; Chung et al., 2023; Ho et al., 2022; Song et al., 2023). But it is unclear how well these heuristics approximate the exact conditional distributions they are designed to mimic, and they often fail in practice (Zhang et al., 2023). These issues are particularly relevant for domains that require accurate conditionals. In molecular design, for example, even a small approximation error could result in atomic structures that have chemically implausible bond distances.

In this paper, we develop a practical and exact method for conditional sampling from a unconditional diffusion model. Our method employs sequential Monte Carlo (SMC), which

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the SPIGM workshop at ICML 2023. Do not distribute.

is a general tool for asymptotically exact inference in sequential probabilistic models (Chopin and Papaspiliopoulos, 2020; Doucet et al., 2001; Naesseth et al., 2019). SMC first simulates an ensemble of weighted trajectories, or *particles*, using a sequence of proposals and weighting mechanisms. It then uses the weighted particles to form an asymptotically exact approximation to a desired target distribution.

The premise of this work is to recognize that the sequential structure of diffusion models permits the application of SMC for sampling from conditional distributions $p_\theta(x \mid y)$. To this end, we develop an algorithm that is practical and asymptotically exact. The algorithm leverages *twisting functions*, an SMC technique that modifies proposals and weighting schemes to approach the optimal choices (Guarniero et al., 2017; Whiteley and Lee, 2014). While optimal twisting functions are analytically intractable, we effectively approximate them with recent heuristic approaches to conditional sampling (eg. Ho et al., 2022), and then correct the errors by the weighting mechanisms The algorithm maintains asymptotic exactness to $p_\theta(x \mid y)$ with many particles, and we find empirically that it improves beyond the heuristics alone with even a handful of particles.

We summarize our contributions as following: (i) We propose a practical SMC algorithm, *Twisted Diffusion Sampler* or TDS, for asymptotically exact conditional sampling from diffusion models; (ii) We show TDS applies to a range of conditional generation problems, and extends to Riemannian manifold diffusion models; (iii) We demonstrate TDS's empirical improvements on MNIST inpainting and class-conditional generation tasks, and (iv) We demonstrate that TDS provides greater flexibility and achieves higher success rates than state of the art on several protein motif-scaffolding problems.

2. Twisted Diffusion Sampler: SMC sampling for diffusion model conditionals

We develop the Twisted Diffusion Sampler (TDS), a practical SMC algorithm targeting $p_\theta(x^0 \mid y)$. Section 2.1 describes how the Markov structure of diffusion models permits a factorization of an extended conditional distribution to which SMC applies. Section 2.2 then shows how a diffusion model's denoising predictions support the application of *twisting*, an SMC technique for constructing a set of "nearly optimal" proposals and weighting functions. Lastly, Section 2.3 shows that TDS applies to a broader range of conditioning problems and extends to Riemannian diffusion models.

TDS builds on diffusion generative models and SMC; we provide background and specify the conventions we adopt for diffusion models and SMC in Appendix A. Empirical results are left to Appendix B.

2.1. Conditional diffusion sampling as an SMC procedure

We first show that the Markov structure of the diffusion model permits a factorization that is recognizable as the final target of an SMC algorithm. Assuming the conditional independence of $x^{1:T}$ and y given x^0, we may write the conditional distribution, extended to additionally include $x^{1:T}$, as

$$p_\theta(x^{0:T} \mid y) = \frac{1}{p_\theta(y)} \left[p(x^T) \prod_{t=0}^{T-1} p_\theta(x^t \mid x^{t+1}) \right] p(y; x^0) \tag{1}$$

with the desired marginal, $p_\theta(x^0 \mid y)$. Comparison of Equation (1) to the form of the final SMC target ν_0 in Equation (21) immediately suggests the application of SMC is possible.

For example, consider choosing proposals $r_T(x^T) = p(x^T)$, $r_t(x^t \mid x^{t+1}) = p_\theta(x^t \mid x^{t+1})$ for $t = 1, \dots, T$, and weighting functions $w_T(x^T) = w_t(x^t, x^{t+1}) = 1$ for $t = 1, \dots, T$ and $w_0(x^0, x^1) = p(y; x^0)$. Substituting these choices into Equation (21) results in the desired final target $\nu_0 = p_\theta(x^{0:T} \mid y)$ with normalizing constant $\mathcal{L}_0 = p_\theta(y)$. As a result, the associated SMC algorithm produces a final set of K weighted particles $\{x_k^0; w_k^0\}_{k=1}^K$ that represent approximate samples from $p_\theta(x^0 \mid y)$ satisfying

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^K w_k^0 \delta_{x_k^0}(x^0) = \nu_0(x^0) = p_\theta(x^0 \mid y) \tag{2}$$

with $x_k^0 \overset{i.i.d.}{\sim} p_\theta(x^0)$.

However, this sampler reduces to naive importance sampling with proposal $p_\theta(x^{0:T})$. Consequently, this approach will be impractical if $p_\theta(x^0 \mid y)$ is too dissimilar from $p_\theta(x^0)$: the number of particles required for accurate estimation of $p_\theta(x^{0:T} \mid y)$ is exponential in KL $\left[p_\theta(x^{0:T} \mid y) \| p_\theta(x^{0:T})\right]$ (Chatterjee and Diaconis, 2018).

2.2. Twisted diffusion sampler

Twisting is a technique in SMC literature intended to reduce the number of particles required for good approximation accuracy (Guarniero et al., 2017; Heng et al., 2020; Naesseth et al., 2019). Its key idea is to relieve the discrepancy between successive intermediate targets by bringing them closer to the final target. This is accomplished by introducing a sequence of *twisting functions* that define new *twisted* proposals and *twisted* weighting functions.

Optimal twisting functions. For example, define the twisted proposals r_t^* by multiplying the naive proposals r_t described in Section 2.1 by the twisting functions set as

$\psi_t^*(x^t) := p_\theta(y \mid x^t)$:

$$r_T^*(x^T) \propto r_T(x^T)\psi_T^*(x^T) \text{ and} \tag{3}$$
$$r_t^*(x^t \mid x^{t+1}) \propto r_t(x^t \mid x^{t+1})\psi_t^*(x^t) \text{ for } \quad 0 \le t < T. \tag{4}$$

In fact, ψ_t^* are *optimal* twisting functions because they permit an SMC sampler that draws exact samples from $p_\theta(x^{0:T} \mid y)$ even when run with a single particle.

To see that even a single particle provides an exact sample, note that by Bayes rule the proposals in Equation (3) reduce to

$$r_T^*(x^T) = p_\theta(x^T \mid y) \text{ and} \tag{5}$$
$$r_t^*(x^t \mid x^{t+1}) = p_\theta(x^t \mid x^{t+1}, y) \text{ for } \quad 0 \le t < T. \tag{6}$$

If we choose twisted weighting functions as $w_T^*(x^T) = w_t^*(x^t, x^{t+1}) = 1$ for $t = T-1, \ldots, 0$, then resulting twisted targets ν_t^* are all equal to $p_\theta(x^{0:T} \mid y)$. Specifically, substituting r_t^* into Equation (21) gives , for $t = 0, \cdots, T$,

$$\nu_t^*(x^{0:T}) = p_\theta(x^T \mid y) \prod_{t=0}^{T-1} p_\theta(x^t \mid x^{t+1}, y) = p_\theta(x^{0:T} \mid y), \tag{7}$$

However, we cannot generate samples from the optimally twisted proposals r_t^* in Equation (5).

Tractable twisting functions. Our key insight is to approximate optimal twisting functions by

$$\tilde{p}_\theta(y \mid x^t) := p(y; \hat{x}_0(x^t, t)) \approx p_\theta(y \mid x^t) = \psi^*(x^t), \tag{8}$$

where $\hat{x}_0(x^t)$ is the denoising estimate of x^0 at step t from the diffusion model. To see that Equation (8) is a reasonable approximation, recall that $\hat{x}_0(x^t) = \mathbb{E}_{p_\theta}[x^0 \mid x^t]$ if p_θ is optimized to exactly match the true distribution q. For $t = 0$ we set $\hat{x}_0(x^0) := x^0$ which makes the approximation to the optimal twisting function exact $\tilde{p}_\theta(y \mid x^0) = p(y \mid x^0) = \psi^*(x^0)$. This choice will ensure that the final target coincides with the exact conditional distribution.

We next use Equation (8) to develop a sequence of twisted proposals $\tilde{p}_\theta(x^t \mid x^{t+1}, y)$, to approximate the optimal proposals $p_\theta(x^t \mid x^{t+1}, y)$ in Equation (5). Specifically, we define twisted proposals as

$$\tilde{p}_\theta(x^t \mid x^{t+1}, y) := \mathcal{N}\left(x^t; x^{t+1} + \sigma_{t+1}^2 s_\theta(x^{t+1}, y); \eta_{t+1}^2\right), \tag{9}$$

$$\text{where} \quad s_\theta(x^t, y) := s_\theta(x^t) + \nabla_{x^t} \log \tilde{p}_\theta(y \mid x^t) \tag{10}$$

is an approximation of the conditional score, $s_\theta(x^t, y) \approx \nabla_{x^t} \log p_\theta(x^t \mid y)$ and η_t^2 is the variance of the step t proposal. For example, for simplicity one could choose each $\eta_t^2 = \sigma_t^2$ to match the unconditional diffusion variance.

The gradient in Equation (10) is computed by backpropagating through the denoising network $\hat{x}_0(x^t)$. Equation (9) builds on previous works (e.g. (Shi et al., 2022)) that seek to approximate the reversal of a conditional forward noising process. And the idea to backpropagate through the denoising network follows from earlier *reconstruction guidance* approaches (Ho et al., 2022; Song et al., 2023) (See related works discussion in Appendix C).

To see why $\tilde{p}_\theta(x^t \mid x^{t+1}, y)$ provides a reasonable approximation of $p_\theta(x^t \mid x^{t+1}, y)$, notice that if $s_\theta(x^t, y) = \nabla_{x^t} \log q(x^t \mid y)$, $\eta_t^2 = \sigma_t^2$, $p_\theta = q$, and $\tilde{p}_\theta(y \mid x^t) = p_\theta(y \mid x^t)$, then

$$\tilde{p}_\theta(x^t \mid x^{t+1}, y) = \mathcal{N}\left(x^t; x^{t+1} + \sigma_{t+1}^2 \nabla_{x^{t+1}} \log q(x^{t+1} \mid y), \sigma_{t+1}^2\right) \tag{11}$$
$$= q(x^t \mid x^{t+1}, y) = p_\theta(x^t \mid x^{t+1}, y). \tag{12}$$

In practice, however, we cannot expect to have $\tilde{p}_\theta(x^t \mid x^{t+1}, y) = p_\theta(x^t \mid x^{t+1}, y)$. So we must introduce non-trivial weighting functions to ensure the resulting SMC sampler converges to the desired final target. In particular, we define *twisted* weighting functions as

$$w_t(x^t, x^{t+1}) := \frac{p_\theta(x^t \mid x^{t+1})\tilde{p}_\theta(y \mid x^t)}{\tilde{p}_\theta(y \mid x^{t+1})\tilde{p}_\theta(x^t \mid x^{t+1}, y)} \tag{13}$$

for $t = 0, \ldots, T-1$, and $w_T(x^T) := \tilde{p}_\theta(x^T)$. The weighting functions in Equation (13) again recover the optimal, constant weighting functions if all other approximations at play are exact.

These twisted proposals and weighting functions define intermediate targets that gradually approach the final target $\nu_0 = p_\theta(x^{0:T} \mid y)$. Substituting into Equation (21) each $\tilde{p}_\theta(x^t \mid x^{t+1}, y)$ for $r_t(x^t \mid x^{t+1})$ and $w_t(x^t, x^{t+1})$ in Equation (18) for each $w_t(x^t, x^{t+1})$ and then simplifying we find

$$\nu_t(x^{0:T}) \propto p_\theta(x^{0:T} \mid y)\left[\frac{\tilde{p}_\theta(y \mid x^t)}{p_\theta(y \mid x^t)} \prod_{t'=0}^{t-1} \frac{\tilde{p}_\theta(x^{t'} \mid x^{t'+1}, y)}{p_\theta(x^{t'} \mid x^{t'+1}, y)}\right]. \tag{14}$$

The right-hand bracketed term in Equation (14) can be understood as the discrepancy of ν_t from the final target ν_0 accumulated from step t to 0 (see Appendix D.1 for a derivation). As t approaches 0, $\tilde{p}_\theta(y \mid x^t)$ improves as an approximation of $p_\theta(y \mid x^t)$, and the t-term product inside the bracket consists of fewer terms. Finally, at $t = 0$, because $\tilde{p}_\theta(y \mid x^0) = p_\theta(y \mid x^0)$ by construction, Equation (14) reduces to $\nu_0(x^{0:T}) = p_\theta(x^{0:T} \mid y)$, as desired.

The TDS algorithm and asymptotic exactness. Together, twisted proposals $\tilde{p}_\theta(x^t \mid x^{t+1}, y)$ and weighting functions $w_t(x^t, x^{t+1})$ lead to the *Twisted Diffusion Sampler*, or TDS, in Algorithm 1. While Algorithm 1 states multinomial resampling for simplicity, in practice other resampling strategies (e.g. systematic (Chopin and Papaspiliopoulos, 2020, Ch. 9)) may be used as well. Under additional conditions, TDS provides arbitrarily accurate estimates of $p_\theta(x^0 \mid y)$. Importantly, this guarantee does not rely on assumptions on the accuracy of the approximations used to derive the twisted proposals and weights.

Theorem 2.1. *(Informal) Let $\mathbb{P}_K = \sum_{k=1}^K w_k^t \delta_{x_k^0}$ denote the discrete measure defined by the particles and weights returned by Algorithm 1 with K particles. Under mild conditions on the twisted proposals and weighting functions, $\mathbb{P}_K$ converges weakly to $p_\theta(x^0 \mid y)$ as K approaches infinity.*

Appendix D provides a statement with complete conditions and proof.

2.3. TDS for inpainting, additional degrees of freedom, and Riemannian diffusion models

The previous decision to set twisting functions $\tilde{p}_\theta(y \mid x^t) := p(y; \hat{x}_0(x^t))$ is one convenient choice, and is sensible only when $p(y; \hat{x}_0(x^t))$ is differentiable and positive. We now show how alternative twisting functions lead to proposals and weighting functions that address inpainting problems and more flexible conditioning specifications. In these extensions, Algorithm 1 applies but with these new twisting functions. Lastly, we extend TDS to Riemannian diffusion models. Appendix D provides additional details, including the adaptation of TDS to variance preserving diffusion models (Song et al., 2020).

Inpainting. Consider the case that x^0 can be segmented into observed dimensions $\mathbf{M}$ and unobserved dimensions $\bar{\mathbf{M}}$ such that we may write $x^0 = [x^0_{\mathbf{M}}, x^0_{\bar{\mathbf{M}}}]$ and let $y = x^0_{\mathbf{M}}$. The goal, then, is to sample $p_\theta(x^0 \mid x^0_{\mathbf{M}} = y) = p_\theta(x^0_{\bar{\mathbf{M}}} \mid x^0_{\mathbf{M}})\delta_y(x^0_{\mathbf{M}})$. Here we define the twisting function as

$$\tilde{p}_\theta(y \mid x^t; \mathbf{M}) := \mathcal{N}\left(y; \hat{x}_0(x^t)_{\mathbf{M}}, \bar{\sigma}_t^2 I\right). \tag{15}$$

The variance in Equation (15) ideally should reflect $\mathrm{Var}_{p_\theta}[y \mid x^t]$, but may be chosen as $\bar{\sigma}_t^2 = \mathrm{Var}_{p_\theta}[x^t \mid x^0]$ for simplicity. The twisted proposals and weighting functions are chosen according to eq 9 and 13, except at $t = 0$. These final quantities are set as

$$\tilde{p}_\theta(x^0 \mid x^1, y; \mathbf{M}) := \delta_y(x^0_{\mathbf{M}}) p_\theta(x^0_{\bar{\mathbf{M}}} \mid x^1) \tag{16}$$

and $w_0(x^0, x^1) = 1$. One can verify the resulting final target is $p_\theta(x^0 \mid x^0_{\mathbf{M}} = y)$ according to Equation (21).

Inpainting with degrees of freedom. We next consider the case when we wish to condition on some observed dimensions, but have additional degrees of freedom. In particular, let $p_\theta(x^0 \mid y) = p_\theta(x^0 \mid \exists \mathbf{M} \in \mathcal{M} \text{ s.t. } x^0_{\mathbf{M}} = y)$, where $\mathcal{M}$ is a set of possible observed dimensions. For example in the context of motif-scaffolding in protein design, the event $\{\exists \mathbf{M} \in \mathcal{M} \text{ s.t. } x^0_{\mathbf{M}} = y)\}$ could represent a functional motif y appearing *anywhere* in a protein structure. In this case, we define

$$\tilde{p}_\theta(y \mid x^t; \mathcal{M}) := \sum_{\mathbf{M} \in \mathcal{M}} \tilde{p}_\theta(y \mid x^t; \mathbf{M}), \tag{17}$$

with each $\tilde{p}_\theta(y \mid x^t; \mathbf{M})$ defined as in Equation (15), and define the final proposal and weight as in Equation (16), with $w_0(x^0, x^1) = 1$ and

$$\tilde{p}_\theta(x^0 \mid x^1, y; \mathcal{M}) := \sum_{\mathbf{M} \in \mathcal{M}} p_\theta(x^0_{\mathbf{M}} \mid x^1; \mathbf{M})\delta_y(x^0_{\mathbf{M}}) p_\theta(x^0_{\bar{\mathbf{M}}} \mid x^1).$$

The cost of a naive approach to computing $\tilde{p}_\theta(y \mid x^t)$ in Equation (17) would grow linearly with $|\mathcal{M}|$, with each term requiring an expensive call to the neural network $\hat{x}_0$. But each term depends on x^t only through the $\hat{x}_0(x^t, t)$, so if the expensive denoising step is computed only once, effectively constant runtime obtains even with large $|\mathcal{F}|$.

TDS on Riemannian manifolds. TDS can be easily extended to diffusion models on Riemannian manifolds with slight modifications. Riemannian diffusion models [7] are set up similarly as in the Euclidean case, but with conditionals defined using tangent normal distributions parameterized with a score approximation followed by a projection step back to the manifold. When we assume that (as, e.g., in (Yim et al., 2023)) the model is associated with a denoising network $\hat{x}_0$, twisting functions are also constructed analogously. For conditional tasks defined by positive likelihoods, $\tilde{p}_\theta(y \mid x^t)$ in Equation (8) applies. For inpainting (and by extension, degrees of freedom), we propose $\tilde{p}_\theta(y \mid x^t) = \mathcal{TN}_{\hat{x}_0^M(x^t)}(y; 0, \bar{\sigma}_t^2)$, where $\mathcal{TN}_{\hat{x}_0^{\mathbf{M}}(x^t)}(0, \bar{\sigma}_t^2)$ is a tangent normal distribution centered on $\hat{x}_0^{\mathbf{M}}(x^t)$. As in the Euclidean case, $\tilde{p}_\theta(x^t \mid x^{t+1}, y)$ is defined with conditional score approximation $s_\theta(x^t, y) = s_\theta(x^t) + \nabla_{x^t} \log \tilde{p}(y \mid x^t)$, which is also computable by automatic differentiation. Appendix E provides details.

3. Discussion

A limitation of TDS is its requirement for additional computational resources to simulate multiple particles. While in some cases we see improved performance with as few as two particles, how many particles is enough is problem dependent. Furthermore, the computational efficiency depends on how closely the twisting functions approximate exact conditionals, which depends on both the unconditional model and the conditioning information. Moreover, choosing twisting functions for generic, nonlinear, constraints may be challenging. Addressing these limitations and improving the computational efficiency of the TDS is an important direction for future work.

References

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. *arXiv preprint arXiv:2302.07121*, 2023.

Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

Gregory Chirikjian and Marin Kobilarov. Gaussian approximation of non-linear measurement models on lie groups. In *53rd IEEE Conference on Decision and Control*, pages 6401–6406. IEEE, 2014.

Nicolas Chopin and Omiros Papaspiliopoulos. *An introduction to sequential Monte Carlo*. Springer, 2020.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, , Philip J Leung, Timothy Huddy, Sam Pellock, Doug Tischer, F Chan, Brian Koepnick, H Nguyen, Alex Kang, B Sankaran, Asim K. Bera, Neil P. King, and David Baker. Robust deep learning-based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, 2022.

Valentin De Bortoli, Emile Mathieu, Michael John Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In *Advances in Neural Information Processing Systems*, 2022.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 2021.

Arnaud Doucet, Nando De Freitas, and Neil James Gordon. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

Pieralberto Guarniero, Adam M Johansen, and Anthony Lee. The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647, 2017.

Jeremy Heng, Adrian N Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential Monte Carlo. *Annals of Statistics*, 48(5), 2020.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Jonathan Ho, Tim Salimans, Alexey A Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David A. Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583 – 589, 2021.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *arXiv preprint arXiv:2201.11793*, 2022.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Christian A Naesseth, Fredrik Lindsten, and Thomas B Schön. Elements of sequential Monte Carlo. *Foundations and Trends in Machine Learning*, 12(3):307–392, 2019.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022a.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022b.

Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Conditional simulation using diffusion Schrödinger bridges. In *Uncertainty in Artificial Intelligence*, 2022.

Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*, 2020.

Brian L Trippe, Jason Yim, Doug Tischer, Tamara Broderick, David Baker, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3D for the motif-scaffolding problem. In *International Conference on Learning Representations*, 2023.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Jue Wang, Sidney Lisanza, David Juergens, Doug Tischer, Joseph L Watson, Karla M Castro, Robert Ragotte, Amijai Saragovi, Lukas F Milles, Minkyung Baek, et al. Scaffolding protein functional sites using deep learning. *Science*, 377(6604), 2022.

Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *bioRxiv*, 2022.

Nick Whiteley and Anthony Lee. Twisted particle filters. *The Annals of Statistics*, 42(1):115–141, 2014.

Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. SE (3) diffusion model with application to protein backbone generation. In *International Conference on Machine Learning*, 2023.

Guanhua Zhang, Jiabao Ji, Yang Zhang, Mo Yu, Tommi Jaakkola, and Shiyu Chang. Towards coherent image inpainting using denoising diffusion implicit models. *arXiv preprint arXiv:2304.03322*, 2023.

A. Background: Diffusion models and sequential Monte Carlo

Diffusion models. Our goal in this work is to sample from $p_\theta(x^0|y) \propto p_\theta(x^0)p(y \mid x^0)$ where $p_\theta(x^0)$ is a fitted unconditional diffusion model and $p(y \mid x^0) := p(y; x^0)$ is a pre-specified conditional likelihood model.

A diffusion model generates a data point x^0 by iteratively refining a sequence of noisy data points x^t, starting from pure noise x^T. This procedure parameterizes a distribution of x^0 as the marginal of a length T Markov chain

$$p_\theta(x^0) = \int p(x^T) \prod_{t=T-1}^{0} p_\theta(x^t \mid x^{t+1}) dx^{1:T}, \tag{18}$$

where $p(x^T)$ is an easy-to-sample noise distribution, and each $p_\theta(x^t \mid x^{t+1})$ is the transition distribution defined by the $(T-t)^{\text{th}}$ refinement step.

Diffusion models p_θ are fitted to match a data distribution $q(x^0)$ from which we have samples. To achieve this goal, a *forward process* $q(x^0)\prod_{t=0}^{T-1} q(x^{t+1} \mid x^t)$ is set to gradually add noise to the data, where $q(x^{t+1} \mid x^t) = \mathcal{N}\left(x^{t+1}; x^t, \sigma_{t+1}^2\right)$, and $\sigma_1^2, \ldots, \sigma_T^2$ is a sequence of variances. To fit a diffusion model, one finds θ such that $p_\theta(x^t \mid x^{t+1}) \approx q(x^t|x^{t+1})$, which is the reverse conditional of the forward process. If this approximation is accomplished for all t, and if T is big enough that $p(x^T) \approx q(x^T)$, then we will have $p_\theta(x^0) \approx q(x^0)$.

In particular, when T is large enough then the reverse conditionals of q are approximately Gaussian,

$$q(x^t \mid x^{t+1}) \approx \mathcal{N}\left(x^t; x^{t+1} + \sigma_t^2 \nabla_{x^{t+1}} \log q(x^{t+1}), \sigma_{t+1}^2\right), \tag{19}$$

where $q(x^t) = \int q(x^0)q(x^t \mid x^0)dx^0$ and $\nabla_{x^t} \log q(x^t)$ is known as the (Stein) score (Song et al., 2020). To mirror Eq 19, diffusion models parameterize $p_\theta(x^t \mid x^{t+1})$ via a *score network* $s_\theta(x^t, t)$

$$p_\theta(x^t \mid x^{t+1}) := \mathcal{N}\left(x^t; x^{t+1} + \sigma_t^2 s_\theta(x^{t+1}, t+1), \sigma_{t+1}^2\right). \tag{20}$$

When $s_\theta(x^t, t)$ is trained to approximate $\nabla_{x^t} \log q(x^t)$, we have $p_\theta(x^t \mid x^{t+1}) \approx q(x^t \mid x^{t+1})$.

Notably, approximating the score is equivalent to learning a *denoising* neural network $\hat{x}_0(x^t, t; \theta)$ to approximate $\mathbb{E}_q[x^0 \mid x^t]$. The reason is that by Tweedie's formula $\nabla_{x^t} \log q(x^t) = \bar{\sigma}_t^{-2}(\mathbb{E}_q[x^0 \mid x^t] - x^t)$ for $\bar{\sigma}_t^2 := \sum_{t'=1}^{T} \sigma_{t'}^2$ and one can set $s_\theta(x^t, t) := \bar{\sigma}_t^{-2}(\hat{x}_0(x^t, t; \theta) - x^t)$. For a detailed training procedure see (Ho et al., 2020; Vincent, 2011). For the remainder of paper, we drop the dependence on θ in $\hat{x}_0$, and drop the argument t in $\hat{x}_0$ and s_θ when it is clear from context.

Sequential Monte Carlo. Sequential Monte Carlo (SMC) is a general tool to approximately sample from a sequence of distributions on variables $x^{0:T}$, terminating at a final target of the main interest (Gordon et al., 1993; Doucet et al., 2001; Naesseth et al., 2019; Chopin and Papaspiliopoulos, 2020). SMC approximates these targets by generating a collection of K *particles* $\{x_k^t\}_{k=1:K}$ across T steps of an iterative procedure. The key ingredients are proposals $r_T(x^T)$, $\{r_t(x^t \mid x^{t+1})\}_{t=0}^{T-1}$ and weighting functions $w_T(x^T)$, $\{w_t(x^t, x^{t+1})\}_{t=0}^{T-1}$. At step T, one draws $x_k^T \sim r_T(x^T)$ and sets $w_k^T := w_T(x_k^T)$, and sequentially repeats the following for $t = T-1, \ldots, 0$:

- *resample* $\{x_k^{t+1:T}\}_{k=1}^K \sim \text{Multinomial}(\{x_k^{t+1:T}\}_{k=1}^K; \{w_k^{t+1}\}_{k=1}^K)$
- *propose* $x_k^t \sim r_t(x^t \mid x_k^{t+1})$
- *weight* $w_k^t := w_t(x_k^t, x_k^{t+1})$.

The proposals and weighting functions together define a sequence of *intermediate target* distributions,

$$\nu_t(x^{0:T}) := \frac{1}{\mathcal{L}_t}\left[r_T(x^T)\prod_{t'=0}^{T-1} r_{t'}(x^{t'} \mid x^{t'+1})\right]\left[w_T(x^T)\prod_{t'=t}^{T-1} w_{t'}(x^{t'}, x^{t'+1})\right] \tag{21}$$

where $\mathcal{L}_t$ is a normalization constant. For example, a classic SMC setting is where ν_t is a target posterior $p(x^{0:T} \mid y^{t:T})$ after the first $T-t$ observations in a filtering equation where $p(x^T)$ and each $p(x^t \mid x^{t+1})$ and $p(y^t \mid x^t)$ are known but the posterior is intractable. This example also points to the connection to diffusion and to our conditional sampling problem of interest.

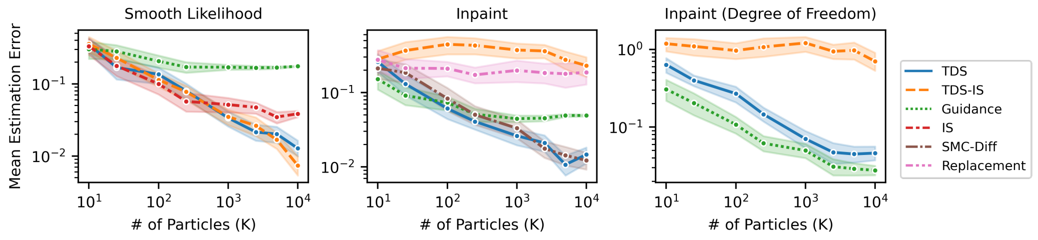


Figure A. Errors of conditional mean estimations with 2 SEM error bars averaged over 25 replicates. TDS applies to all three tasks and provides increasing accuracy with more particles.

The defining property of SMC is that the weighted particles at each t form discrete approximations $(\sum_{k'=1}^{K} w_{k'}^t)^{-1} \sum_{k=1}^{K} w_t^k \delta_{x_k^t}(x^t)$ (where δ is a Dirac measure) to $\nu_t(x^t)$ that become arbitrarily accurate in the limit that many particles are used (Chopin and Papaspiliopoulos, 2020, Proposition 11.4). So, by choosing r_t and w_t so that $\nu_0(x^0)$ matches the desired distribution, one can guarantee perfect accuracy in the large compute limit.

However, for SMC to be practical with finite compute, the weights must be sufficiently regular. Otherwise when only a small fraction of particles receive nontrivial weights, the rest will be discarded and the procedure provides a poor approximation of the target distributions.

B. Empirical Results

We first test the dependence of the accuracy of TDS on the number of particles in synthetic settings with tractable exact conditionals in Appendix B.1. We then demonstrate TDS's utility on an MNIST class-conditional generation task in Appendix B.2, and include an MNIST inpainting experiment in Appendix F. Finally, in Appendix B.3 we show that when applied to an unconditional model of protein backbones, TDS provides state-of-the-art performance on the motif-scaffolding problem. All experiment details are included in Appendix F.

Our evaluation includes: (i) *TDS*; (ii) *TDS-IS*, a variant of TDS without resampling steps; (iii) *Guidance*, equivalent to TDS with 1 particle (e.g. Chung et al., 2023; Ho et al., 2022; Song et al., 2023); (iv) *IS*, the naive importance sampler described in Equation (2); (v) *Replacement* method, a heuristic sampler for inpainting tasks that replaces $x_{\mathbf{M}}^t$ with a noisy version of the observed $x_{\mathbf{M}}^0$ at each t (Song et al., 2020); and (vi) *SMC-Diff*, an SMC algorithm with replacement method as proposals for inpainting tasks (Trippe et al., 2023). TDS and SMC-Diff are implemented with sytematic resampling. In all experiments, we follow the standard practice of returning the denoising mean on the final sample (Ho et al., 2020). See Appendix C for detailed method descriptions and other related works.

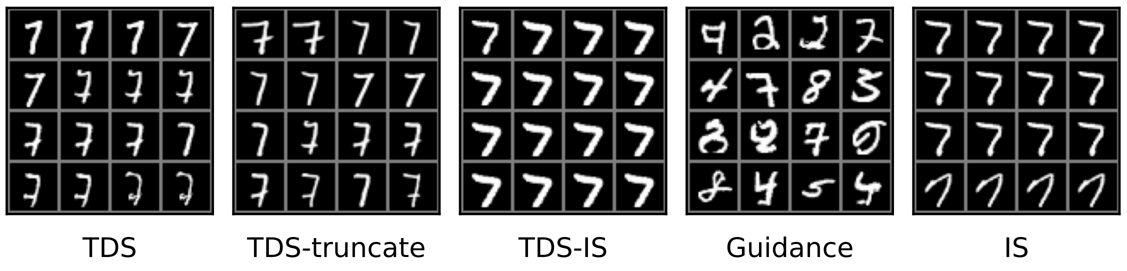
Each of the SMC samplers (TDS, TDS-IS, IS, and SMC-Diff) forms a discrete approximation to $p_\theta(x^0 \mid y)$ with K weighted particles $\{x_k^0, w_k^0\}_{k=1}^K$, i.e. $(\sum_{k'=1}^{K} w_{k'}^0)^{-1} \cdot \sum_{k=1}^{K} w_k^0 \delta_{x_k^0}(x^0)$. Guidance and replacement methods are considered to form a similar particle-based approximation with K independently drawn samples viewed as K particles with uniform weights.

B.1. Synthetic diffusion models

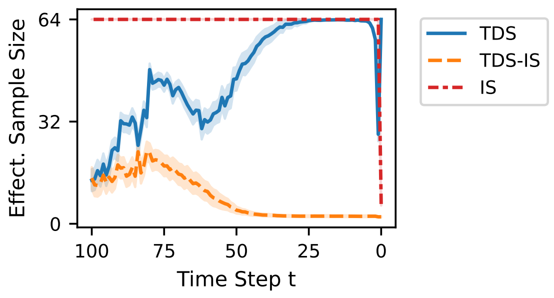
We explore two questions in this section: (i) what sorts of conditioning information can be handled by TDS as well as other baselines, and (ii) does TDS provide better conditional estimates?

To study these questions, we use an unconditional diffusion model p_θ to approximate a bivariate Gaussian q, where p_θ is defined by analytical score functions (see Appendix F). We consider three test cases defining the conditional information: (i) Smooth likelihood: $p_\theta(x^0 \mid y = 0) \propto p_\theta(x^0)p(y = 0; x^0)$ where $p(y; x^0) \propto e^{-\|x^0 - y\|_2}$; (ii) Inpainting: $p_\theta(x^0 \mid y = 1)$ where y representing the first dimension of x^0 is set to 1; and (iii) Inpainting with degrees-of-freedom: $p_\theta(x^0 \mid y = 0 \text{ or } 1)$ where y – first dimension of x^0 – can be either 0 or 1.

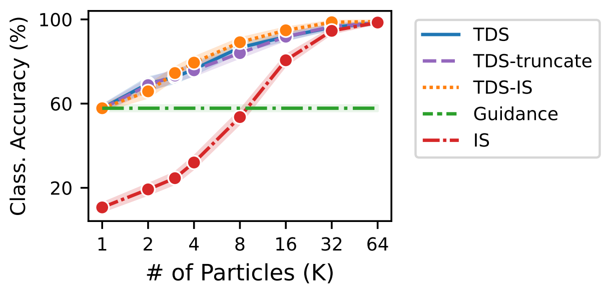
For the above problems, our choice of the diffusion model permits tractable conditional distributions which we can use as ground truth, and the exact score $\nabla_{x^t} \log q(x^t)$ is analytically available. Hence, we can analyze the performance without the influence of score network approximation errors. And the choice of a two-dimensional target distributions permits very close approximation of exact conditional distributions by numerical integration that can then be used as ground truth.



(a) Conditional samples of each method ran with 64 particles given class $y = 7$.



(b) ESS trace avg. over 100 runs ($K = 64$)



(c) Classification accuracy avg. over 1k runs.

Figure B. MNIST class-conditional generation. TDS generates a diverse set of desired digits, and has higher particle efficiency compared to other SMC samplers. Notably, TDS with $K = 2$ particles outperform the Guidance baseline in terms of classification accuracy.

Figure A reports the estimation error for the mean of the desired conditional distribution, i.e. $\|\sum w_k x_k^0 - \mathbb{E}_q[x^0 \mid y]\|_2$, where we approximate $\mathbb{E}_q[x^0 \mid y]$ numerically. TDS provides a computational-statistical trade-off: using more particles decreases estimation error at a roughly $O(1/\sqrt{K})$ rate (note the slopes of $\approx -1/2$ in log-log scale) as expected from standard SMC theory (Chopin and Papaspiliopoulos, 2020, Ch. 11). Moreover, TDS is applicable to all three different conditioning problems, and consistently outperforms other baselines.

B.2. MNIST class-conditional generation

We move on to study the performance of TDS on diffusion models where neural network approximations to score functions are employed. In particular, we study the class-conditional generation task on MNIST dataset, which involves sampling an image of the digit from $p_\theta(x^0 \mid y) \propto p_\theta(x^0)p(y; x^0)$, where y is a given class of the digit, and $p(y; \cdot)$ denotes the classification likelihood[1]. We compare TDS to TDS-IS, Guidance, and IS, all with $T = 100$ sampling steps. In addition we include a variation called *TDS-truncate* that truncates the TDS procedure at $t = 10$ and returns $\hat{x}_0(x^{10})$.

To assess the faithfulness of generation, we evaluate *classification accuracy* (CA) on predictions of conditional samples x^0 given y, made by the same classifier that specifies the likelihood. Another metric used for comparing between SMC samplers (namely TDS, TDS-IS and IS) is *effective sample size* (ESS) , which is defined as $(\sum_{k=1}^K w_k^t)^2 / (\sum_{k=1}^K (w_k^t)^2)$ for K weighted particles $\{x_k^t, w_k^t\}_{k=1}^K$. Note that ESS is always bounded between 0 and K.

The results are summarized in Figure B. To compare generation quality and diversity, we visualize conditional samples given class $y = 7$ in Figure Ba. Samples from Guidance have noticeable artifacts, and most of them do not resemble the digit 7, whereas the other 4 methods produce authentic and correct digits. owever, most samples from IS or TDS-IS are identical. By contrast, samples from TDS and TDS-truncate have greater diversity, with the latter exhibiting slightly more variations.

The ESS trace comparison in Figure Bb shows that TDS has a general upward trend of ESS approaching $K = 64$. Though in the final few steps ESS of TDS drops by a half, it is still higher than those of TDS-IS and IS that deteriorates to around 1

[1]We trained an unconditional diffusion model on 60k training images with 1k diffusion steps and the architecture proposed in (Dhariwal and Nichol, 2021), The likelihood id parameterized by a pretrained ResNet50 model taken from https://github.com/VSehwag/minimal-diffusion. See Appendix F for details.

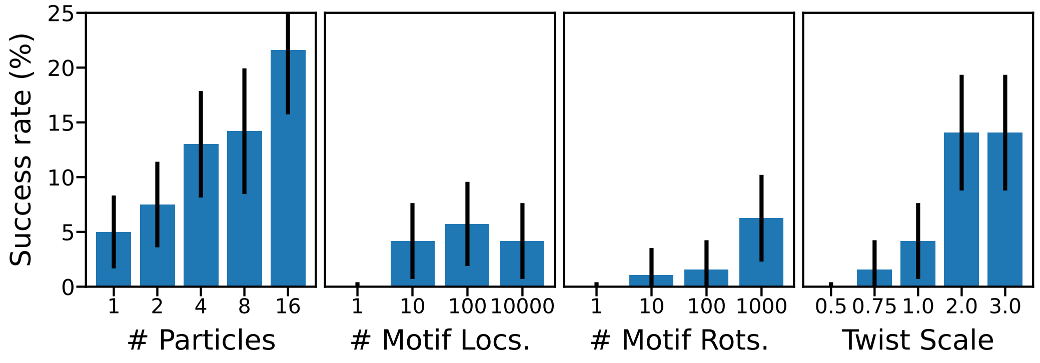


Figure C. Dependence of TDS motif-scaffolding success rate (test case `5IUS`) on (Left) number of particles, (Middle Left) number of motif location degrees of freedom, (Middle Right) number of rotation degrees of freedom, and twist scale.

and 6 respectively.

Finally, Figure Bc depicts the classification accuracy of all methods against # particles K. For all SMC samplers, more particles improve the accuracy, with $K = 64$ leading to nearly perfect accuracy. Given the same K, TDS and its variants have similar accuracy that outperforms others. This observation suggests that one can use TDS-truncate to avoid effective sample size drop and encourage the generation diversity, without compromising the performance.

TDS can be extended by exponentiating twisting functions with a *twist scale*. This extension is related to existing literature of classifier guidance (eg. Dhariwal and Nichol, 2021) that exponentiates the classification likelihood. We conduct an ablation study on twist scales, and find that moderately large twist scales enhance TDS's performance especially for small K. See Appendix F for details.

B.3. Protein design application: flexible conditioning and state-of-the-art motif-scaffolding

The biochemical functions of proteins are typically imparted by a small number of atoms (a *motif*), that are stabilized by the overall protein structure, known as the *scaffold* (Wang et al., 2022). A central task in protein design, then, is to identify stabilizing scaffolds in response to motifs known or theorized to confer biochemical function. Provided with a generative model supported on physically realizable protein structures $p_\theta(x^0)$, suitable scaffolds may be constructed by solving a conditional generative modeling problem (Trippe et al., 2023). Complete structures are viewed as segmented into a motif $x^0_{\mathbf{M}}$ and scaffold $x^0_{\bar{\mathbf{M}}}$, i.e. $x^0 = [x^0_{\mathbf{M}}, x^0_{\bar{\mathbf{M}}}]$. Putative compatible scaffolds are then identified by (approximately) sampling from $p_\theta(x^0_{\bar{\mathbf{M}}} \mid x^0_{\mathbf{M}})$.

While the conditional generative modeling approach to motif-scaffolding has produced functional, experimentally validated structures for certain motifs, the general problem remains open. For example, on a recently proposed benchmark set of motif-scaffolding problems, the state-of-the-art method (RFdiffusion, a conditionally trained diffusion model) provides low in silico success on a majority of test cases (Watson et al., 2022). Moreover, current methods for motif-scaffolding require a practitioner to specify of the location of the motif within the primary sequence of the full scaffold; this choice that can require significant expert knowledge and trial and error.

We hypothesized that greater flexibility and improved motif-scaffolding could be achieved through accurate conditional sampling. To this end, we applied TDS to FrameDiff, a Riemannian diffusion model that parameterizes protein backbones as a collection of rigid bodies (Yim et al., 2023). [2]

We first use TDS to address the requirement that motif location indices within the scaffold be specified in advance. While prior work randomly sampled this variable (Trippe et al., 2023; Wang et al., 2022; Watson et al., 2022), TDS can eliminate this degree of freedom and condition on the motif appearing at *any* combination of indices by applying the strategy described in Section 2.3 (see Appendix G for details). We are similarly able to eliminate the rotation of the motif as a degree of freedom as well. To circumvent the computational intractability of accounting for every possible motif location and orientation, we

[2] `https://github.com/jasonkyuyim/se3_diffusion`

subsample as needed; subsampling introduces the number of motif locations and rotations subsampled as a hyperparameter. Notably conditional training does not immediately address these degrees of freedom.

We follow a *self-consistency* evaluation criteria of generated backbones described in (Trippe et al., 2023) that (i) uses fixed backbone sequence design (inverse folding) to generate a putative amino acid sequence to encode the backbone, (ii) forward folds sequences to obtain backbone structure predictions, and (iii) judges the quality of initial backbones by their agreement (or self-consistency) with predicted structures. We inherit the specifics of our evaluation and success criteria set-up following (Watson et al., 2022), including using ProteinMPNN (Dauparas et al., 2022) for step (i) and AlphaFold (Jumper et al., 2021) on a single sequence (no multiple sequence alignment) for (ii). We first explore the impact of different hyper-parameters on a single problem in the benchmark, `5IUS`, before addressing the full benchmark set.

Figure C compares the impact of several TDS hyperparameters on success rates for a single motif (`5IUS`) from the benchmark set (Watson et al., 2022). We found success rate to increase monotonically with the number of particles used, with $K = 16$ providing a roughly 4 fold increase in empirical success rate (Figure C Left). Non-zero success rates in this setting with FrameDiff required accounting for motif locations (Figure C Left uses $1{,}000$ possible motif locations and 100 possible rotations). The success rate was 0% without accounting for these degrees of freedom, and increased with larger numbers of locations and rotations (Figure C Middle-Left and Middle-Right). Larger twist-scales also gave higher success rates on 5IUS (Figure C Right), though we found this trend was not monotonic for all problems (see Appendix G).

We next evaluated TDS on a benchmark set of 24 motif-scaffolding problems and compare its success rates to the previous state of the art, RFdiffusion (Watson et al., 2022). We ran TDS with K=1 and K=8 particles, twist-scale=2, and 100 rotations and $1{,}000$ motif location degrees of freedom (100,000 combinations total). TDS with 8 particles provides higher empirical success than guidance on most benchmark problems, including two cases (`5TPN` and `7MRX_128`) with zero success rate, and two problems (`6EXZ_long` and `1QJG`) on which the empirical success rate increased by at least 4-fold. We found variable sample efficiency (quantified by ESS) in different settings (see Appendix G). We next compare performance too RFdiffusion (Watson et al., 2022), the previous state of the art method. RFdiffusion operates on the same rigid body representation of protein backbones as FrameDiff, but is conditionally trained for the motif-scaffolding task. TDS provides higher success rates on half (11/22) of the problems on which either TDS and RFdiffusion have non-zero success rates. This performance is obtained despite the fact that FrameDiff is not trained on this task.

The division between problems on which each method performs well is primarily explained by total scaffold length, with TDS providing higher success rates on smaller scaffolds. TDS has higher success on 9/12 problems with scaffolds shorter than 100 residues and 2/10 problems with 100 residue or longer scaffolds; the successes in this latter group (`5UIS` and `1QJG`) both have discontiguous motif segments, on which the motif-placement degrees of freedom may be particularly helpful. We suspect the performance gap between TDS and RFdiffusion on longer scaffolds owes to (i) the underlying model; long backbones generated unconditionally by RFdiffusion are designable (backbone scRMSD¡2Å) with significantly higher frequency (Yim et al., 2023) and (ii) that unlike RFdiffusion, FrameDiff does not model and so can not condition on the fixed motif sequence.

C. Related work

There has been much recent work on conditional generation using diffusion models. But these prior works demand either task specific conditional training, or involve unqualified approximations and can suffer from poor performance in practice.

Approaches involving training with conditioning information. There are a variety of works that involve training a neural network on conditioning information to achieve approximate conditional sampling. These works include

- Conditional training with embeddings of conditioning information, e.g. for denoising or deblurring lower resolution images (Saharia et al., 2022b), and text-to-image generation (Ramesh et al., 2022)).
- Conditioning training with a subset of the state space, e.g. for protein design (Watson et al., 2022), and image inpainting (Saharia et al., 2022a).
- Classifier-guidance (Song et al., 2020; Dhariwal and Nichol, 2021), an approach for generating samples from a diffusion model that fall into a desired class. This strategy requires training a time-dependent classifier to approximate, for each $t,\ p(y \mid x^t)$. Training such a time-dependent classifier may be inconvenient and costly.
- Classifier-free guidance (Ho and Salimans, 2022), an approach that builds on the idea of classifier-guidance but does

not require training a separate classifier. Instead, this technique trains a diffusion model that takes class information as additional input.

These conditionally-trained models have the limitation that they do not apply to new types of conditioning information without additional training.

Reconstruction guidance. Outside of the context of SMC and the twisting technique, the use of denoising estimate $\hat{x}_0(x^t, t)$ to form an approximation $\tilde{p}_\theta(x^t \mid x^{t+1}, y)$ to $p_\theta(x^t \mid x^{t+1}, y)$ is called *reconstruction guidance* (Ho et al., 2022; Chung et al., 2023; Song et al., 2023). This technique involves sampling one trajectory from $\tilde{p}_\theta(x^t \mid x^{t+1}, y)$ from $T-1$ to 1, starting with $x^T \sim p(x^T)$. Notably, while this approximation is reasonable, there is no formal guarantee or evaluation criteria on how accurate the approximations, as well as the final conditional samples x^0, are. Also this approach has empirically been shown to have unreliable performance in image inpainting problems (Zhang et al., 2023).

Replacement method. The replacement method (Song et al., 2020) is the most widely used approach for approximate conditional sampling in an unconditionally trained diffusion model. In the inpainting problem, it replaces the observed dimensions of intermediate samples $x^t_{\mathbf{M}}$, with a noisy version of observation $x^0_{\mathbf{M}}$. However, it is a heuristic approximation and can lead to inconsistency between inpainted region and observed region (Lugmayr et al., 2022). Additionally, the replacement method is applicable only to inpainting problems. While recent work has extended the replacement method to linear inverse problems (e.g. Kawar et al., 2022), the heuristic approximation aspect persists, and it is unclear how to further extend to problems with smooth likelihoods.

SMC-Diff. Most closely related to the present work is SMC-Diff (Trippe et al., 2023), which uses SMC to provide asymptotically accurate conditional samples for the inpainting problem. However, this prior work (i) is limited to the inpainting case, and (ii) provides asymptotically accurate conditional samples only under the assumption that the learned diffusion model exactly matches the forward noising process at every step. Notably, the assumption in (ii) will not be satisfied in practice. Also, SMC-Diff does not leverage the idea of twisting functions.

D. Supplementary details on method

Algorithm 1: TDS: Twisted Diffusion Sampler

```
1  for $k = 1 : K$ do
2      $x_k^T \sim p(x^T)$
3      $\psi_k^T \leftarrow \texttt{twist}\,(x_k^T, T)$
4      $w_k \leftarrow \psi_k^T$
5  end
6  for $t = T-1, \cdots, 0$ do
7      $\{x_k^{t+1}, \psi_k^{t+1}\}_{k=1}^K \sim \text{Mult.}\left(\{x_k^{t+1}, \psi_k^{t+1}\}_{k=1}^K; \{w_k\}_{k=1}^K\right)$
8      for $k = 1 : K$ do
9          $s_k \leftarrow \bar{\sigma}_{t+1}^{-2}(\hat{x}_0(x_k^{t+1}) - x_k^{t+1})$
10         $s_k^\psi \leftarrow s_k + \nabla_{x_k^{t+1}} \log \psi_k^{t+1}$
11         $x_k^t \sim \tilde{p}_\theta(\cdot \mid x_k^{t+1}, y) := \mathcal{N}\left(x_k^{t+1} + \sigma_{t+1}^2 s_k^\psi, \eta_{t+1}^2\right)$
12         $\psi_k^t \leftarrow \texttt{twist}\,(x_k^t, t)$
13         $w_k \leftarrow \frac{p_\theta(x_k^t \mid x_k^{t+1})\psi_k^t}{\psi_k^{t+1}\tilde{p}_\theta(x_k^t \mid x_k^{t+1}, y)}$
14     end
15 end
```

Algorithm 2: Twisting Functions

```
1  Function twist $(x, t)$:
2      return $p(y; \hat{x}_0(x, t))$
3  Function twist2 $(x, t, y, \mathbf{M})$:
4      $\hat{y} \leftarrow \hat{x}_0(x, t)_{\mathbf{M}}$
5      return $\mathcal{N}(y; \hat{y}, \bar{\sigma}_t^2)$
6  Function twist3 $(x, t, y, \mathcal{M})$:
7      return $\sum_{\mathbf{M} \in \mathcal{M}}$ twist2$(x, t, y, \mathbf{M})$
```

TDS algorithm for variance exploding diffusion models. Our method developed in Section 2 is based on the variance exploding (VE) diffusion models. To provide a brief summary, VE diffusion models define the forward process q by

$$q(x^{1:T} \mid x^0) := \prod_{t=1}^{T} q(x^t \mid x^{t-1}), \qquad q(x^t \mid x^{t-1}) := \mathcal{N}(x^t; x^{t-1}, \sigma_t^2). \tag{22}$$

where σ_t^2 is an increasing sequence of variances such that $q(x^T) \approx \mathcal{N}(0, \bar{\sigma}_T^2)$, with $\bar{\sigma}_t^2 := \sum_{t'=1}^{t} \sigma_{t'}^2$ for $t = 1, \cdots, T$. And so one can set $p(x^T) = \mathcal{N}(0, \bar{\sigma}_T^2)$. Equation (22) gives the marginal conditional $q(x^t \mid x^0) = \mathcal{N}(x^t; x^0, \bar{\sigma}_t^2)$. The reverse diffusion process p_θ is parameterized as

$$p_\theta(x^{0:T}) := p(x^T) \prod_{t=T}^{1} p_\theta(x^{t-1} \mid x^t), \qquad p_\theta(x^{t-1} \mid x^t) := \mathcal{N}(x^{t-1}; x^t + \sigma_t^2 s_\theta(x^t, t), \sigma_t^2) \tag{23}$$

where the score network s_θ is modeled through a denoiser network $\hat{x}_0$ by $s_\theta(x^t, t) := \frac{\hat{x}_0(x^t, t; \theta) - x^t}{\bar{\sigma}_t^2}$.

The TDS algorithm for VE models is then described in Algorithm 1. Algorithm 2 covers twisting functions for different conditioning cases described in Sections 2.2 and 2.3.

Extension to variance preserving diffusion models. Another widely used diffusion framework is variance preserving (VP) diffusion models (Ho et al., 2020). VP models define the forward process q by

$$q(x^{1:T} \mid x^0) := \prod_{t=1}^{T} q(x^t \mid x^{t-1}), \qquad q(x^t \mid x^{t-1}) := \mathcal{N}(x^t; \sqrt{1 - \sigma_t^2} x^{t-1}, \sigma_t^2) \tag{24}$$

where σ_t^2 is a sequence of increasing variances chosen such that $q(x^T) \approx \mathcal{N}(0, 1)$, and so one can set $p(x^T) = \mathcal{N}(0, 1)$. Define $\alpha_t := 1 - \sigma_t^2$, $\bar{\alpha}_t := \prod_{t'=1}^{t} \alpha_{t'}$, and $\bar{\sigma}_t^2 := 1 - \bar{\alpha}_t$. Then the marginal conditional of Equation (22) is $q(x^t \mid x^0) = \mathcal{N}(x^t; \sqrt{\bar{\alpha}_t} x^0, \bar{\sigma}_t^2)$. The reverse diffusion process p_θ is parameterized as

$$p_\theta(x^{0:T}) := p(x^T) \prod_{t=T}^{1} p_\theta(x^{t-1} \mid x^t), \quad p_\theta(x^{t-1} \mid x^t) := \mathcal{N}(x^{t-1}; \sqrt{\alpha_t} x^t + \sigma_t^2 s_\theta(x^t, t), \sigma_t^2) \tag{25}$$

where s_θ is now defined through the denoiser $\hat{x}_0$ by $s_\theta(x^t, t) := \frac{\sqrt{\bar{\alpha}_t}\hat{x}_0(x^t, t; \theta) - x^t}{\bar{\sigma}_t^2}$.

TDS still applies to VP settings, with slight modifications of Algorithm 1 that Line 1 is changed to $s_k \leftarrow \frac{\sqrt{\bar{\alpha}_{t+1}}\hat{x}_0(x_k^{t+1}) - x_k^{t+1}}{\bar{\sigma}_{t+1}^2}$, and Line 1 is changed to $x_k^t \sim \tilde{p}_\theta(\cdot \mid x_k^{t+1}, y) := \mathcal{N}\left(\sqrt{\alpha_{t+1}} x_k^{t+1} + \sigma_{t+1}^2 s_k^\psi, \eta_{t+1}^2\right)$.

Proposal variance. In Line 1 of Algorithm 1, the proposal variance is a hyperparameter η_{t+1}^2. Unless otherwise specified, we set $\eta_{t+1}^2 := \mathrm{Var}_{p_\theta}[x^t \mid x_k^{t+1}]$.

Resampling strategy. The mulinomial resampling strategy in Line 1 of Algorithm 1 can be replaced by other strategies. In our experiments, we use the systematica resampling strategy.

In addition, one can consider setting an effective sample size (ESS) threshold (between 0 and 1), and only when the ESS is smaller than this threshold, the resampling step is implemented. Unless otherwise specified, TDS is implemented with ESS threshold equal to 1, i.e. always resampling.

D.1. Derivation of Equation (14)

We here provide a derivation of Equation (14), which illustrated how one can interpret the extended intermediate targets $\nu_t(x^{0:T})$ as approximations to the final target $p_\theta(x^{0:T} \mid y)$ that become increasingly accurate as t approaches 0. We obtain Equation (14) by substituting the proposal distributions in Equation (9) and weights in Equation (13) into Equation (21) and simplifying.

$$\nu_t(x^{0:T}) \propto \left[p(x^T)\prod_{t'=0}^{T}\tilde{p}_\theta(x^{t'} \mid x^{t'+1}, y)\right]\left[\tilde{p}(y \mid x^T)\prod_{t'=t}^{T-1}\frac{p_\theta(x^{t'} \mid x^{t'+1})\tilde{p}_\theta(y \mid x^{t'})}{\tilde{p}_\theta(y \mid x^{t'+1})\tilde{p}_\theta(x^{t'} \mid x^{t'+1}, y)}\right]$$

Rearrange and cancel p_θ and $\tilde{p}_\theta$ terms.

$$= \left[p(x^T)\prod_{t'=t}^{T}p_\theta(x^{t'} \mid x^{t'+1})\right]\left[\tilde{p}_\theta(y \mid x^{t'})\prod_{t'=0}^{t-1}\tilde{p}_\theta(x^{t'} \mid x^{t'+1}, y)\right]$$

Group p_θ terms by chain rule of probability.

$$= p_\theta(x^{t:T})\tilde{p}_\theta(y \mid x^t)\prod_{t'=0}^{t-1}\tilde{p}(x^{t'} \mid x^{t'+1}, y)$$

Apply Bayes' rule and note that $p_\theta(y \mid x^{t:T}) = p_\theta(y \mid x^t)$.

$$\propto p_\theta(x^{t:T} \mid y)\frac{\tilde{p}_\theta(y \mid x^t)}{p(y \mid x^t)}\prod_{t'=0}^{t-1}\tilde{p}(x^{t'} \mid x^{t'+1}, y)$$

Multiply by $\prod_{t'=0}^{t-1}p_\theta(x^{t'} \mid x^{t'+1}, y)/p_\theta(x^{t'} \mid x^{t'+1}, y) = 1$.

$$= p_\theta(x^{0:T} \mid y)\left[\frac{\tilde{p}_\theta(y \mid x^t)}{p(y \mid x^t)}\prod_{t'=0}^{t-1}\frac{\tilde{p}(x^{t'} \mid x^{t'+1}, y)}{p_\theta(x^{t'} \mid x^{t'+1}, y)}\right].$$

The final line is the desired expression.

D.2. Asymptotic accuracy of TDS – additional details and theorem statement

In this section we (i) characterize sufficient conditions under which TDS provides arbitrarily accurate estimates as the number of particles is increased and (ii) discuss when these conditions will hold in practice.

We begin with a theorem providing sufficient conditions for asymptotic accuracy of TDS.

Theorem D.1. *Let $p_\theta(x^{0:T})$ be a diffusion generative model (defined by Equations (18) and (20)) with*

$$p_\theta(x^t \mid x^{t+1}) = \mathcal{N}\left(x^t \mid x^{t+1} + \sigma_{t+1}^2 s_\theta(x^{t+1}), \sigma_{t+1}^2 I\right),$$

with variances $\sigma_1^2, \ldots, \sigma_T^2$. Let $\tilde{p}_\theta(y \mid x^t)$ be twisting functions, and

$$r_t(x^t \mid x^{t+1}) = \mathcal{N}\left(x^t \mid x^{t+1} + \sigma_{t+1}^2[s_\theta(x^{t+1}) + \nabla_{x^{t+1}}\log\tilde{p}_\theta(y \mid x^{t+1})], \eta_{t+1}^2 I\right)$$

be proposals distributions for $t = 0, \ldots, T-1$, and let $\mathbb{P}_K = \sum_{k=1}^K w_k^0 \delta_{x_k^0}$ for weighted particles $\{(x_k^0, w_k^0)\}_{k=1}^K$ returned by Algorithm 1 with K particles. Assume

(a) the final twisting function is the likelihood, $\tilde{p}_\theta(y \mid x^0) = p(y; x^0)$,

(b) the first twisting function $\tilde{p}_\theta(y \mid x^T)$, and the ratios of subsequent twisting functions $\tilde{p}_\theta(y \mid x^t)/\tilde{p}_\theta(y \mid x^{t+1})$ are positive and bounded,

(c) each $\nabla_{x^t}\log\tilde{p}_\theta(y \mid x^t)$ with $t > 0$ is continuous and has bounded gradients, and

(d) the proposal variances are larger than the model variances, i.e. for each t, $\eta_t^2 > \sigma_t^2$.

Then $\mathbb{P}_K$ converges weakly to $p_\theta(x^0 \mid y)$ with probability one, that is for every set A, $\lim_{K\to\infty}\mathbb{P}_K(A) = \int_A p_\theta(x^0 \mid y)dx^0$.

The assumptions of Theorem D.1 are mild. Assumption (a) may be satisfied by construction by, for example, choosing $\tilde{p}_\theta(y \mid x^0) = p(y; x^0)$ by enforcing that $\hat{x}_0(x^0) = x^0$.

Assumption (b) that $\tilde{p}_\theta(y \mid x^T)$ and each $\tilde{p}_\theta(y \mid x^t)/\tilde{p}_\theta(y \mid x^{t+1})$ are positive and bounded will typically be satisfied too; for example, $p(y; x)$ is smooth in x and everywhere positive, and if $\hat{x}_0(x^t)$ takes values in some compact domain. An alternative sufficient condition for the positive and bounded condition is for $p_\theta(y \mid x^0)$ to be positive and bounded away from zero; this will typically be the case when, for example, $p(y|x^0)$ is a classifier fit with some sort of regularization.

Next, Assumption (c) is the strongest assumption. It will be satisfied if, for example, if $p(y; x)$ and $\hat{x}_0(x^t)$ are smooth in x and x^t. While smoothness of $\hat{x}_0$ can be encouraged by the use of skip-connections and regularization, particularly at t close to zero, this $\hat{x}_0$ may present sharp transitions.

Lastly, Assumption (d), that the proposal variances satisfy $\eta_t^2 > \sigma_t^2$ is likely not needed for the result to hold, but permits using existing SMC theoretical results in the proof; in practice, our experiments use $\eta_t^2 = \sigma_t^2$, but alternatively the assumption could be met by inflating each η_t by some arbitrarily small δ without markedly impacting the behaviour of the sampler.

Proof of Theorem D.1: Theorem D.1 characterizes a set of conditions under which a standard result on the convergence of SMC algorithms. We restate this result below in our own notation, which differs from the alternative presentations in, for example, the reversal of time indices.

Theorem D.2 (Chopin and Papaspiliopoulos (2020) – Proposition 11.4)**.** *Let $\{(x_k^0, w_k^0)\}_{k=1}^K$ be the particles and weights returned at the last iteration of a sequential Monte Carlo algorithm with K particles using multinomial resampling. If each weighting function $w_t(x^t, x^{t+1})$ is positive and bounded, then for every bounded, ν_0-measurable function ϕ of x^t*

$$\lim_{K\to\infty} \sum_{k=1}^K w_k^0 \phi(x_k^0) = \int \phi(x^0)\nu_0(x^0)dx^0.$$

with probability one.

An immediate consequence of Theorem D.2 is the weak (i.e. set-wise) convergence of the discrete measures, $\hat{P}_K = \sum_{k=1}^K w_k^0 \delta_{x_k^0}$. This can be seen by taking for each $\phi(x) = \mathbb{I}[x \in A]$ for any set A. The theorem applies both in the Euclidean setting, where each $x_k^t \in \mathbb{R}^D$, as well as the Riemannian setting.

We now proceed to prove Theorem D.1.

Proof. To prove the theorem we show (i) the x^0 marginal final target ν_0 is $p_\theta(x^0 \mid y)$ and then (ii) $\mathbb{P}_K$ converges weakly to ν_0.

We first show (i) by manipulating ν_0 in Equation (21) to obtain $p_\theta(x^{0:T} \mid y)$. From Equation (21) we first have

$$\begin{aligned}
\nu_0(x^{0:T}) &= \frac{1}{\mathcal{L}_0}\left[r(x^T)\prod_{t'=0}^{T-1} r_{t'}(x^{t'} \mid x^{t'+1})\right]\left[w_T(x^T)\prod_{t'=0}^{T-1} w_{t'}(x^{t'}, x^{t'+1})\right] \\
&\quad \text{Substitute in proposals and weights from Eqs ...} \\
&= \frac{1}{\mathcal{L}_0}\left[p(x^T)\prod_{t'=0}^{T-1} r_t(x^t \mid x^{t+1})\right]\left[\tilde{p}_\theta(y \mid x^T)\prod_{t'=0}^{T-1} \frac{p_\theta(x^t \mid x^{t+1})\tilde{p}_\theta(y \mid x^t)}{\tilde{p}_\theta(y \mid x^{t+1}) r_t(x^t \mid x^{t+1})}\right] \\
&\quad \text{Rearrange } p_\theta \text{ and } r_t \text{ terms, and cancel out } r_t \text{ terms.} \\
&= \frac{1}{\mathcal{L}_0}\left[p(x^T)\prod_{t'=0}^{T-1} p_\theta(x^t \mid x^{t+1})\right]\left[\tilde{p}_\theta(y \mid x^T)\prod_{t'=0}^{T-1} \frac{\cancel{r_t(x^t \mid x^{t+1})}\tilde{p}_\theta(y \mid x^t)}{\tilde{p}_\theta(y \mid x^{t+1})\cancel{r_t(x^t \mid x^{t+1})}}\right] \\
&\quad \text{Collapse } p_\theta \text{ terms, rearrange and cancel } \tilde{p}_\theta \text{ terms.} \\
&= \frac{1}{\mathcal{L}_0} p_\theta(x^{0:T})\left[\prod_{t'=0}^{T-1} \frac{\cancel{\tilde{p}_\theta(y \mid x^t)}}{\cancel{\tilde{p}_\theta(y \mid x^{t+1})}}\right]\tilde{p}_\theta(y \mid x^1) \\
&\quad \text{Recognize } p_\theta(y \mid x^0) = p(y \mid x^0) \text{ by Assm.(a) and apply Bayes' rule with } \mathcal{L}_0 = p_\theta(y) \\
&= p_\theta(x^{0:T} \mid y).
\end{aligned}$$

The final line reveals that once we marginalize out $x^{1:T}$ we obtain $\nu_0(x^0) = p(x^0 \mid y)$ as desired.

We next show that $\mathbb{P}_K$ converges to ν_0 by applying Theorem D.2. To apply Theorem D.2 we need only to confirm that the weights at each step are upper bounded. Since there are a finite number of steps T, it is enough to show that each w_t is bounded. The first weight is the first twisting function, $w_T(x^T) = \tilde{p}_\theta(y \mid x^T)$, which is bounded by Assumption (b). So we proceed to intermediate weights.

To show that the weighting functions at subsequent steps are bounded, we decompose the log-weighting functions as

$$\log w_t(x^t, x^{t+1}) = \log \frac{\tilde{p}_\theta(y \mid x^t)}{\tilde{p}_\theta(y \mid x^{t+1})} + \log \frac{p_\theta(x^t \mid x^{t+1})}{r_t(x^t \mid x^{t+1})},$$

and show independently that $\log \tilde{p}_\theta(y \mid x^t)/\tilde{p}_\theta(y \mid x^{t+1})$ and $\log p_\theta(x^t \mid x^{t+1})/r_t(x^t \mid x^{t+1})$ are bounded. The first term $\log \tilde{p}_\theta(y \mid x^t)/\tilde{p}_\theta(y \mid x^{t+1})$ is again bounded by Assumption (b), and we proceed to the second.

That $\log p_\theta(x^t \mid x^{t+1})/r_t(x^t \mid x^{t+1})$ is bounded follows from Assumptions (c) and (d). First write

$$p_\theta(x^t \mid x^{t+1}) = \mathcal{N}\left(x^t \mid \hat{\mu}, \sigma_{t+1}^2 I\right)$$

with $\hat{\mu} = x^{t+1} + \sigma_{t+1}^2 s_\theta(x^{t+1})$, and

$$r_t(x^t \mid x^{t+1}) = \mathcal{N}\left(x^t \mid \hat{\mu}_\psi, \eta_{t+1}^2 I\right),$$

for $\hat{\mu}_\psi = \hat{\mu} + \sigma_{t+1}^2 \nabla_{x^{t+1}} \log \tilde{p}_\theta(y \mid x^{t+1})$. The log-ratio then simplifies as

$$\begin{aligned}
\log \frac{p_\theta(x^t \mid x^{t+1})}{r_t(x^t \mid x^{t+1})} &= \log \frac{|2\pi\sigma_{t+1}^2 I|^{-1/2} \exp\{-(2\sigma_{t+1}^2)^{-1}\|\hat{\mu} - x^t\|^2\}}{|2\pi\eta_{t+1}^2 I|^{-1/2} \exp\{-(2\eta_{t+1}^2)^{-1}\|\hat{\mu}_\psi - x^t\|^2\}} \\
&\text{Rearrange and let } C = \log |2\pi\sigma_{t+1}^2 I|^{-1/2}/|2\pi\eta_{t+1}^2 I|^{-1/2} \\
&= \frac{-1}{2}\left[\sigma_{t+1}^{-2}\|\hat{\mu} - x^t\|^2 - \eta_{t+1}^{-2}\|\hat{\mu}_\psi - x^t\|^2\right] + C \\
&\text{Expand and rearrange } \|\hat{\mu}_\psi - x^t\|^2 = \|\hat{\mu} - x^t\|^2 + 2\langle \hat{\mu}_\psi - \hat{\mu}, \hat{\mu} - x^t\rangle + \|\hat{\mu}_\psi - \hat{\mu}\|^2 \\
&= \frac{-1}{2}\left[(\sigma_{t+1}^{-2} - \eta_{t+1}^{-2})\|\hat{\mu} - x^t\|^2 - 2\eta_{t+1}^{-2}\langle \hat{\mu}_\psi - \hat{\mu}, \hat{\mu} - x^t\rangle - \eta_{t+1}^{2}\|\hat{\mu}_\psi - \hat{\mu}\|^2\right] + C \\
&\text{Let } C' = C - \frac{1}{2}\eta_{t+1}^2\|\hat{\mu}_\psi - \hat{\mu}\|^2 \text{ and rearrange} \\
&= \frac{-1}{2}(\sigma_{t+1}^{-2} - \eta_{t+1}^{-2})\|\hat{\mu} - x^t\|^2 + \eta_{t+1}^{-2}\langle \hat{\mu}_\psi - \hat{\mu}, \hat{\mu} - x^t\rangle + C' \\
&\text{Apply Cauchy-Schwarz} \\
&\leq \frac{-1}{2}(\sigma_{t+1}^{-2} - \eta_{t+1}^{-2})\|\hat{\mu} - x^t\|^2 + \eta_{t+1}^{-2}\|\hat{\mu}_\psi - \hat{\mu}\| \cdot \|\hat{\mu} - x^t\| + C' \\
&\text{Upper-bounding using that } \max_x \frac{-a}{2}x^2 + bx = \frac{b^2}{2a}. \\
&\leq \frac{1}{2}\frac{(\eta_{t+1}^{-2}\|\hat{\mu}_\psi - \hat{\mu}\|)^2}{\sigma_{t+1}^{-2} - \eta_{t+1}^{-2}} + C' \\
&= \frac{\eta_{t+1}^{-4}}{2(\sigma_{t+1}^{-2} - \eta_{t+1}^{-2})}\|\hat{\mu}_\psi - \hat{\mu}\|^2 + C' \\
&= \frac{\eta_{t+1}^{-4}}{2(\sigma_{t+1}^{-2} - \eta_{t+1}^{-2})}\sigma_{t+1}^2\|\nabla_x^t \log \tilde{p}_\theta(y \mid x^t)\|^2 + C' \\
&\leq C''.
\end{aligned}$$

The final line follows from Assumption (c), that the gradients of the twisting functions are bounded. The above derivation therefore provides that each w_t is bounded, concluding the proof. □

E. Extending Twisted Diffusion Sampler to Riemannian diffusion models

This section provides additional details on the extension of TDS to Riemannian diffusion models introduced in Section 2. We begin with some background on Riemannian diffusion models. Then we describe the extension of TDS to these models. Finally we show how Algorithm 1 modifies to this setting.

Riemannian diffusion models. Unconditional generation in Riemannian diffusion models proceeds through a geodesic random walk (De Bortoli et al., 2022).

At each step t, x^t is sampled from a tangent normal distribution (see e.g. (Chirikjian and Kobilarov, 2014; De Bortoli et al., 2022)) as

$$x^t \sim p(x^t \mid x^{t+1}) = \mathcal{TN}_{x^{t+1}}\left(x^t; \sigma_t^2 s_\theta(x^{t+1}), \sigma_t^2\right). \tag{26}$$

In particular we use $\mathcal{TN}_{x^{t+1}}(x^t; \mu, \sigma^2)$ with x^t and x^{t+1} in the manifold, $\mu \in \mathcal{T}_{x^{t+1}}$ (the tangent space at x^{t+1}), and $\sigma^2 > 0$ to denote the density of x^t implied by a two step procedure. The first step is to sample a variable $\bar{x}^t$ in $\mathcal{T}_{x^{t+1}}$; if $\{h_1, \ldots, h_D\}$ is an orthonormal basis of $\mathcal{T}_{x^{t+1}}$ one may generate

$$\bar{x}^t = \sigma_{t+1}^2 s_\theta(x^{t+1}) + \sum_{d=1}^{D} \sigma_{t+1}\epsilon_d \cdot h_d,$$

with $\epsilon_d \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$. The second step is to project $\bar{x}^t$ back onto the manifold to obtain $x^t = \exp_{x^{t+1}}\{\bar{x}^t\}$ where $\exp_x\{\cdot\}$ denotes the exponential map at x.

TDS for Riemannian diffusion models. To extend TDS, appropriate analogues of the twisted proposals and weights are all that is needed. For this extension we require that the diffusion model is also associated with a manifold-valued denoising estimate $\hat{x}_0$, as will be the case when, for example, $s_\theta(x^t, t) := \nabla_{x^t} \log q(x^t \mid x^0 = \hat{x}_0)$ for $\hat{x}_0(x^t, t)$. In contrast to the Euclidean case, a relationship between a denoising estimate and a computationally tractable score approximation may not always exist for arbitrary Riemannian manifolds; however for Lie groups when the the forward diffusion is the Brownian motion, tractable score approximations do exist (Yim et al., 2023, Proposition 3.2).

For positive and differentiable $p_\theta(y \mid x^t)$, we again choose $\tilde{p}_\theta(y \mid x^t) = p(y; \hat{x}_0(x^t))$.

Next the inpainting and inpainting with degrees of freedom cases. Here, assume that x^0 lives on a multidimensional manifold (e.g. $SE(3)^N$) and the unmasked observation $y = x^0_{\bar{\mathbf{M}}}$ with $\bar{\mathbf{M}} \subset \{1, \ldots, N\}$ on a lower-dimensional sub-manifold (e.g. $SE(3)^{|\bar{\mathbf{M}}|}$, with $M < N$). In this case, twisting functions are constructed exactly as in Section 2.3, except with the normal density in Equation (15) replaced with a Tangent normal.

We propose the twisting function and twisted proposal

$$\tilde{p}_\theta(y \mid x^t) = \mathcal{TN}_{\hat{x}_0^M(x^t)}(y; 0, \bar{\sigma}_t^2) \text{ and } \tilde{p}_\theta(x^t \mid x^{t+1}, y) = \mathcal{TN}_{x^{t+1}}\left(x^t; \sigma_{t+1}^2 s_\theta(x^{t+1}, y), \eta_{t+1}^2\right) \tag{27}$$

where as in the Euclidean case $s_\theta(x^t, y) = s_\theta(x^t) + \nabla_{x^t} \log \tilde{p}(y \mid x^t)$.

Note that the tangent Normal as we have defined it involves a change of variables (the exponential map). Therefore, to compute its density exactly one must compute the Jacobian of the exponential map. However since this term always occurs both the numerator and denominator of the weights, it cancels out at each step and so we do not need to compute it.

Weights at intermediate steps are computed as in the Euclidean case (Equation (13)). However, since the tangent normal includes a projection step back to the manifold, its density involves not only usual Gaussian density but also the Jacobian of the exponential map to account for the change of variables. For example, in the case that the ambient dimension of the manifold and the tangent space are the same dimension,

$$\mathcal{TN}_x(x'; \mu, \sigma^2) = \mathcal{N}(\exp_x^{-1}\{x'\}; \mu, \sigma^2) \left| \frac{\partial}{\partial x} \exp_x^{-1}\{x'\} \right|,$$

where $\exp_x^{-1}\{x'\}$ is the inverse of the exponential map at x from the manifold into $\mathcal{T}_x$, and $\left|\frac{\partial}{\partial x} \exp_x^{-1}\{x'\}\right|$ is the determinant of its Jacobian. The weights are then computed as

$$\begin{aligned} w_t(x^t, x^{t+1}) &:= \frac{p_\theta(x^t \mid x^{t+1}) \tilde{p}_\theta(y \mid x^t)}{\tilde{p}_\theta(y \mid x^{t+1}) \tilde{p}_\theta(x^t \mid x^{t+1}, y)} && (28) \\ &= \frac{\mathcal{TN}_{x^{t+1}}(x^t; \sigma_{t+1}^2 s_\theta(x^{t+1}), \sigma_{t+1}^2) \tilde{p}_\theta(y \mid x^t)}{\tilde{p}_\theta(y \mid x^{t+1}) \mathcal{TN}_{x^{t+1}}(x^t; \sigma_{t+1}^2 s_\theta(x^{t+1}, y), \eta_{t+1}^2)}. && (29) \end{aligned}$$

While the proposal and target contribute identical Jacobian determinant terms that cancel out, they remain in the twisting functions.

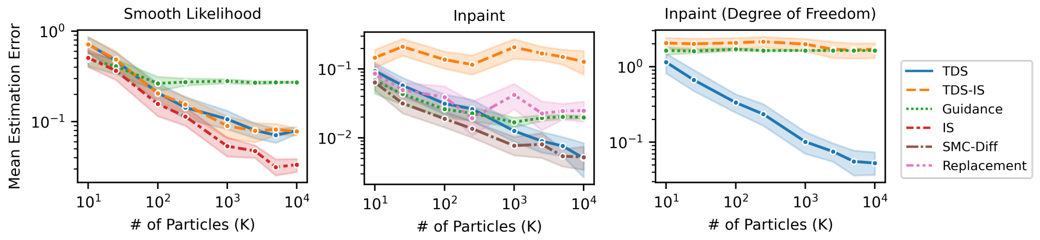


Figure D. Errors of conditional mean estimations with 2 SEM error bars averaged over 25 replicates on mixture of Gaussians unconditional target. TDS applies to all three tasks and provides increasing accuracy with more particles.

Adapting the TDS algorithm to the Riemannian setting. To translate the TDS algorithm to the Riemannian setting we require only two changes. The first is on Algorithm 1 Line 1. Here we assume that the score is computed through the transition density functions of the the forward processed used when training the diffusion model as

$$s_\theta(x^t, t) = \nabla_{x^t} \log q_{t\,|\,0}(x^t \mid \hat{x}_0) \quad \text{for } \hat{x}_0 = \hat{x}_0(x^t).$$

Line 1 is then replaced with

$$s_k \leftarrow \nabla_{x^t} \log q_{t\,|\,0}(x_k^t \mid \hat{x}_0) \quad \text{for } \hat{x}_0 = \hat{x}_0(x_k^t).$$

Notably, s_k (and similarly s_k^ψ on the next line) is a $\mathcal{T}_{x^t}$-valued Riemannian gradient.

The second change is to make the proposal on Algorithm 1 Line 1 a tangent normal, as defined in Equation (26).

F. Results supplementary details

F.1. Synthetic diffusion models

Forward process. Our forward process is variance preserving (as described in Appendix D) with $T = 100$ steps and a quadratic variance schedule. We set $\sigma_t^2 = \sigma_{\min}^2 + (\frac{t}{T})^2 \sigma_{\max}^2$ with $\sigma_{\min}^2 = 10^{-5}$ and $\sigma_{\max}^2 = 10^{-1}$.

Unconditional target distributions. We evaluate the different methods with two different unconditional target distributions:

1. A **bivariate Gaussian** with mean at $(\frac{1}{2}, \frac{1}{2})$ and covariance 0.9 and
2. A **Gaussian mixture** with three components with mixing proportions $[0.3, 0.5, 0.2],$, means $[(1.54, -0.29), (-2.18, 0.57), (-1.09, -1.40)]$, and 0.2 standard deviations.

Unconditional target distributions. We evaluate using three different conditional distributions:

1. **Smooth likelihood**, with $p(y; x^0) = \exp\{-\|x^0\}$,
2. **Inpainting** corresponding to $p(y; x^0) = \delta_y(x_1^0)$, for $y = 0$ and
3. **Inpainting with degrees-of-freedom** corresponding to $p(y; x^0) = \delta_y(x_1^0) + \delta_y(x_1^0 - 1)$.

Figure D provides results analogous to those in Figure A but with the mixture of Gaussians unconditional target. In this second example we evaluate a with a variation of the inpainting degrees of freedom case wherein we consider $y = 1$ and $\mathcal{M} = \{[1], [2]\}$, so that $p(y; x^0) = \delta_y(x_1^0) + \delta_y(x_2^0)$.

F.2. MNIST experiments

Setup. We set up an MNIST diffusion model using variance preserving framework. The model architecture is based on the guided diffusion codebase[3], with the following specifications: number of channels = 64, attention resolutions = ”28,14,7”,

[3] https://github.com/openai/guided-diffusion

number of residual blocks = 3, learn sigma (i.e. learning the variance of $p_\theta(x^{t-1} \mid x^t)$) = True, resblock updown = True, dropout = 0.1, variance schedule = "linear". We trained the model for 60k epochs with a batch size of 128 and a learning rate of 10^{-4} on 60k MNIST training images. The model uses $T = 1000$ for training and $T = 100$ for sampling.

The classifier used for class-conditional generation and evaluation is a pretrained ResNet50 model [4]. This classifier is trained on the same set of MNIST training iamges.

F.2.1. MNIST CLASS-CONDITIONAL GENERATION

Sample plots. To supplement the sample plot conditioned on class 7 in Figure Ba, we present samples conditioned on each of the remaining 9 classes. The observations are similar to Appendix B.2.

Ablation study on twist scales. We consider exponentiating and re-normalizing twisting functions by a *twist scale* γ, i.e. setting new twisting functions to $\tilde{p}_\theta(y \mid x^t; \gamma) \propto p(y; \hat{x}_0(x^t))^\gamma$. In particular, when $t = 0$, we set $p(y; x^0; \gamma) \propto p(y; x^0)^\gamma$. This modification suggests that the targeted conditional distribution is now

$$p_\theta(x^0 \mid y; \gamma) \propto p_\theta(x^0) p(y; x^0)^\gamma. \tag{30}$$

By setting $\gamma > 1$, the classification likelihood becomes sharper, which is potentially helpful for twisting the samples towards a specific class. The TDS algorithm (and likewise TDS-IS and Guidance) still applies with this new definition of twisting functions. The use of twist scale is similar to the *classifier scale* introduced in classifier-guidance literature, which is used to multiply the gradient of the log classification probability (Dhariwal and Nichol, 2021).

In Figure F, we examine the effect of varying twist scales on classification accuracy of TDS, TDS-IS and Guidance. We consider two ways to evaluate the accuracy. First, *classification accuracy* computed by a *neural network classifier*, where the evaluation setup is the same as in Appendix B.2. Second, the *human-rated* classification accuracy, where a human (one of the authors) checks if a generated digit has the right class and does not have artifacts. Since human evaluation is expensive, we only experiment with TDS ($K = 64, 2$), TDS-IS ($K = 64, 2$) and Guidance. In each class-conditional generation run, we randomly sample one particle out of K particles according to the associated weights. We conduct 64 runs for each class label, leading to a total of 640 samples for evaluation.

Figure Fa depicts the classification accuracy measured by a neural network classifier. We observe that in general larger twist scale improves the classification accuracy. For TDS and TDS-IS, the improvement is more significant for smaller number of particles K used.

Figure Fb depicts the human-rated accuracy. In this case, we find that larger twist scale is not necessarily better. A moderately large twist scale ($\gamma = 2, 3$) generally helps increasing the accuracy, while an excessively large twist scale ($\gamma = 10$) decreases the accuracy. An exception is TDS with $K = 64$ particles, where any $\gamma > 1$ leads to worse accuracy compared to the case of $\gamma = 1$. Study on twist scale aside, we find that using more particles K help improving human-rated accuracy (recall that Guidance is a special case of TDS with $K = 1$). And TDS and TDS-IS with $K = 64$ and $\gamma = 1$ both have almost perfect accuracy. This observation is consistent with previous findings with neural network classifier evaluation in Appendix B.2.

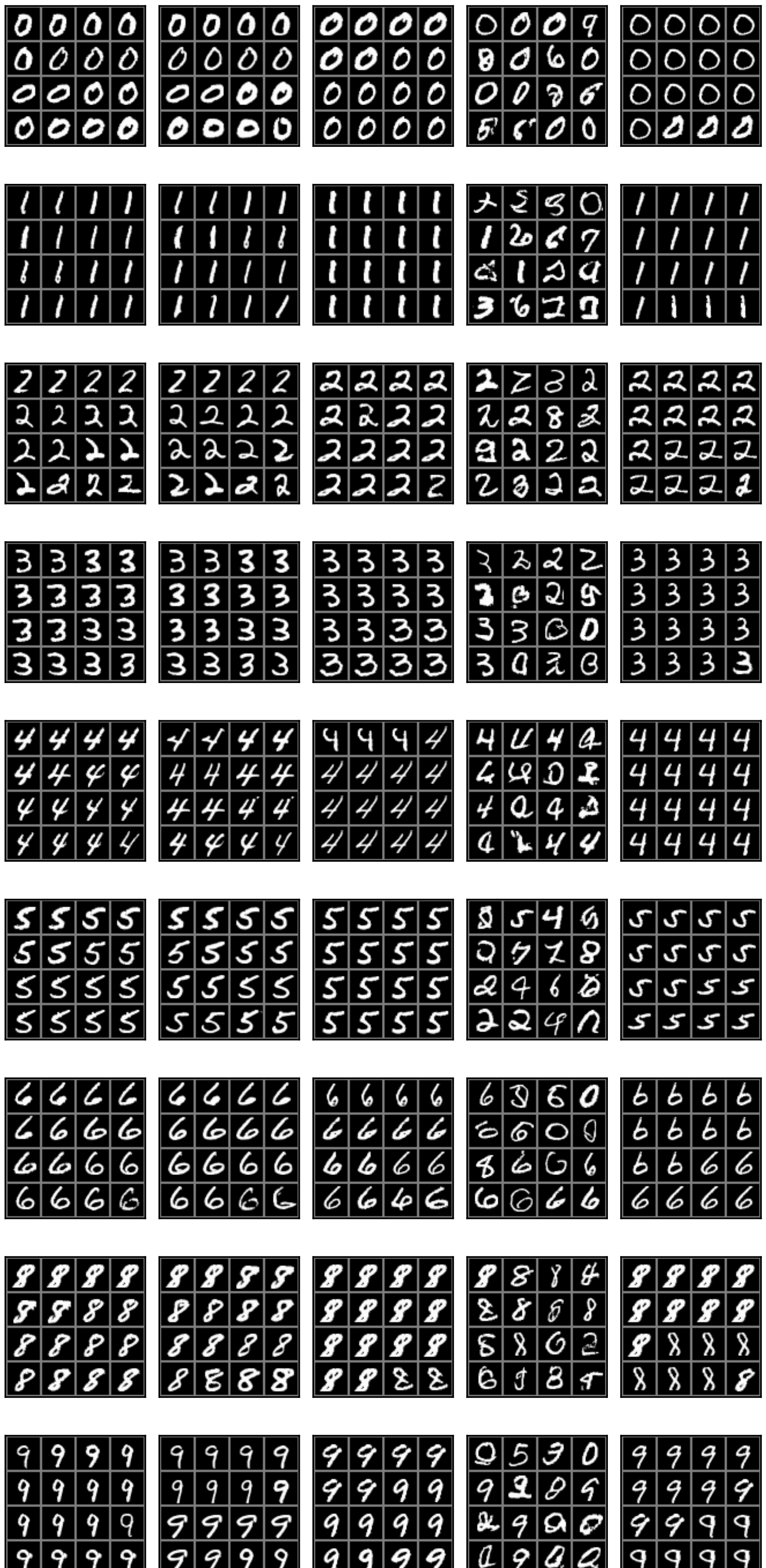
We note that there is a discrepancy on the effects of twist scales between neural network evaluation and human evaluation. We suspect that when overly large twist scale is used, the generated samples may fall out of the data manifold; however, they may still retain features recognizable to a neural network classifier, thereby leading to a low human-rated accuracy but a high classifier-rated accuracy. To validate this hypothesis, we present samples conditioned on class 6 in Figure G. For example, in Figure Ge, Guidance with $\gamma = 1$ has 31 good-quality samples out of 64, and the rest of the samples often resamble the shape of other digits, e.g. 3,4,8; and Guidance with $\gamma = 10$ has 34 good-quality samples, but most of the remaining samples resemble 6 with many artifacts.

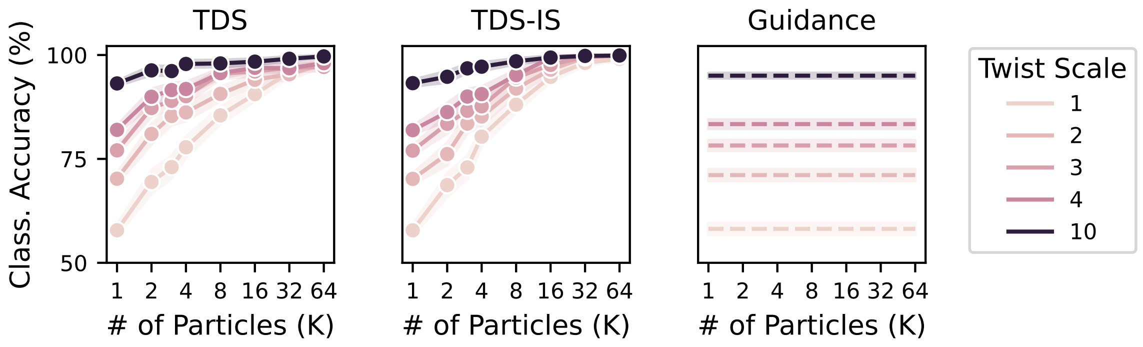
F.2.2. MNIST INPAINTING

The inpainting task is to sample images from $p_\theta(x^0 \mid x^0_{\mathbf{M}} = y)$ for some observed y, where $x^0 = [x^0_{\mathbf{M}}, x^0_{\bar{\mathbf{M}}}]$ is separated into observed dimensions $\mathbf{M}$ and unobserved dimensions $\bar{\mathbf{M}}$, as described in Section 2.3.

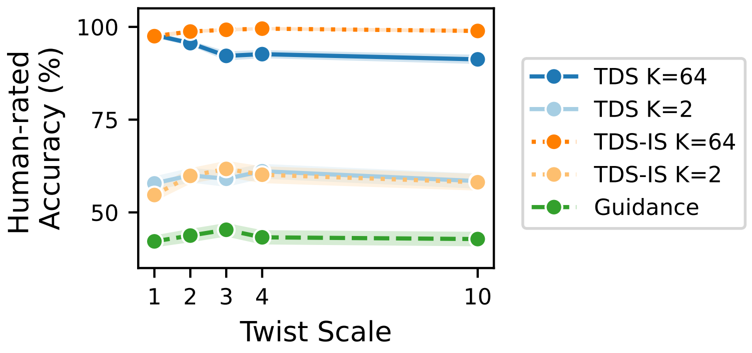
In this experiment, we consider two types of observed dimensions: (1) $\mathbf{M}$ = "half", where the left half of an image is observed, and (2) $\mathbf{M}$ = "quarter", where the upper left quarter of an image is observed.

[4]Downloaded from `https://github.com/VSehwag/minimal-diffusion`





(a) Classification accuracy (measured by the neural network classifier) v.s. number of particles K, under different twist scales. Results are averaged over 1k random runs with error bands indicating 2 standard errors. Across the panels we see that for all the methods, the larger the twist scale (i.e. the darker the line color), the higher the classification accuracy. For TDS and TDS-IS, this improvement is more significant for smaller value of K,



(b) Human-rated classification accuracy v.s. twist scale, for TDS ($K = 64, 2$), TDS-IS ($K = 64, 2$) and Guidance. Results are averaged over 640 randomly chosen samples with error bands indicating 2 standard errors. For TDS ($K = 2$), TDS-IS ($K = 64, 2$) and Guidance, a moderate increase in twist scale improves the human-rated accuracy; however, excessively large twist scale can hurt the accuracy. For TDS ($K = 64$), increasing the twist scale generally decreases the human-rated accuracy.

Figure F. MNIST class-conditional generation: classification accuracy under different twist scales, computed by a neural network classifier (top panel) and a human (bottom panel).

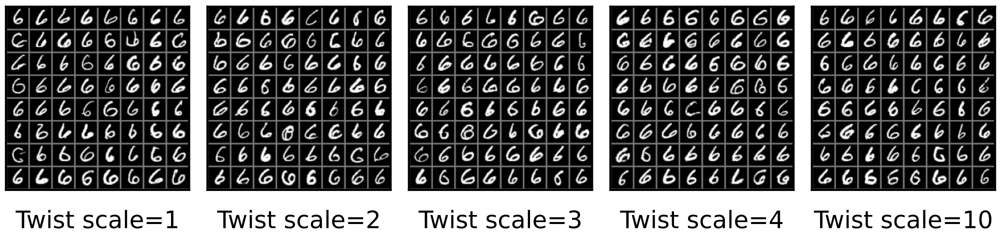
We run TDS, TDS-IS, Guidance, SMC-Diff, and Replacement to inpaint 10k validation images. We also include TDS-truncate that truncates the TDS procedure at $t = 10$ and returns $\hat{x}_0(x^{10})$.

For TDS and its variants, twisting functions are defined by `twist2` in Algorithm 2, with the modification that the variance σ_t^2 in line 5 is replaced by other *twist scales* γ_t. Here we set $\gamma_t := \frac{\tilde{\sigma}_t^2 \tau^2}{\tilde{\sigma}_t^2 + \tau^2}$, where $\tilde{\sigma}_t^2 := \frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}$ and $\tau^2 = 0.12$ is the population variance of x^0 estimated from training data.

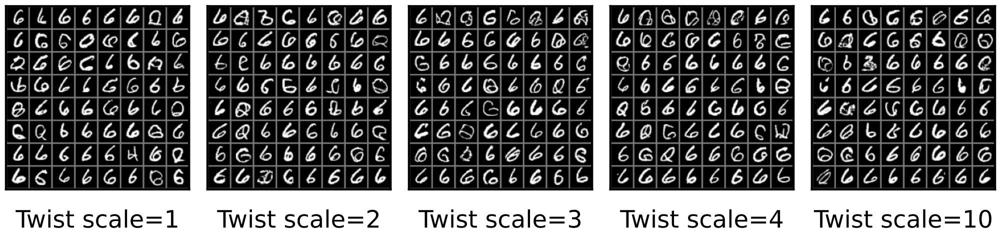
Metrics. As in Appendix B.2, we use effective sample size (ESS) to compare the particle efficiency among different SMC samplers (namely TDS, TDS-IS, and SMC-Diff).

In addition, we ground the performance of a sampler in the downstream task of classifying a partially observed image $x^0_{\mathrm{M}} = y$: we use a classifier to predict the class $\hat{z}$ of an inpainted image x^0, and compute the accuracy against the true class z^* of the unmasked image (z^* is provided in MNIST dataset). This prediction is made by the same classifier used in Appendix B.2.

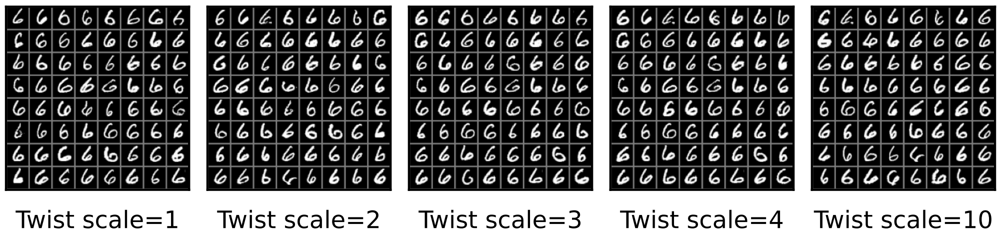
Consider K weighted particles $\{x_k^0; w_k^0\}_{k=1}^K$ generated from a sampler conditioned on $x^0_{\mathrm{M}} = y$ and assume the weights are



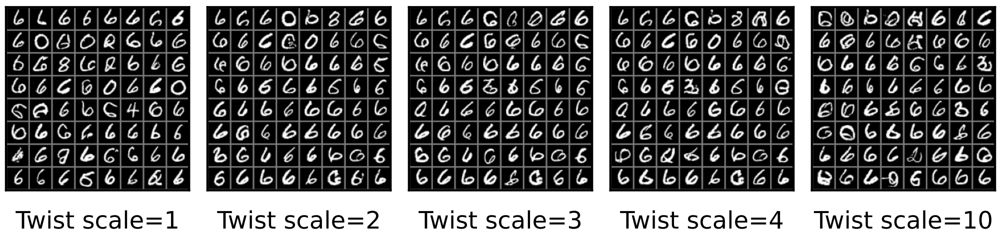
(a) TDS ($K = 64$). # of good-quality samples counted by human is 62, 61, 63, 60, 62, from left to right.



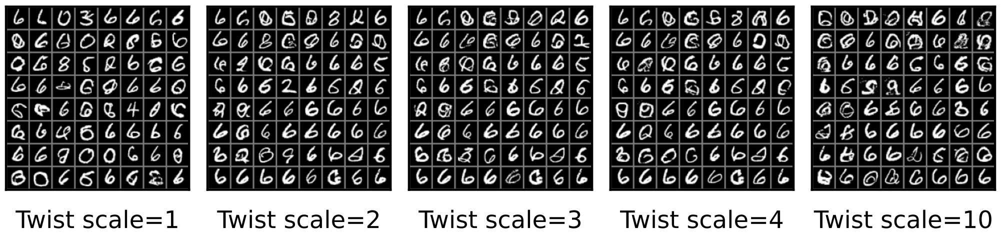
(b) TDS ($K = 2$). # of good-quality samples counted by human 49, 51, 51, 46, 45, from left to right.



(c) TDS-IS ($K = 64$). # of good-quality samples counted by a human is 62, 61, 64, 64, 61, from left to right.

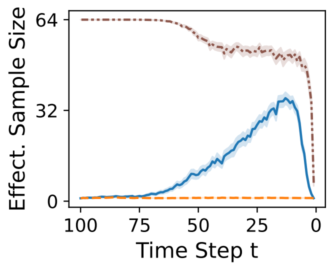


(d) TDS-IS ($K = 2$). # of good-quality samples counted by a human is 49, 52, 53, 52, 48, from left to right.

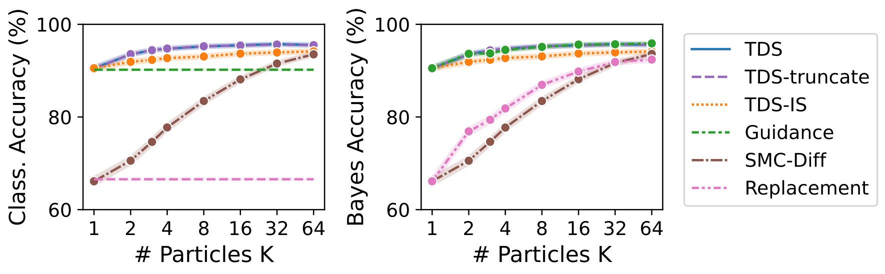


(e) Guidance. # of good-quality samples counted by a human is 31, 38, 41, 39, 34, from left to right.

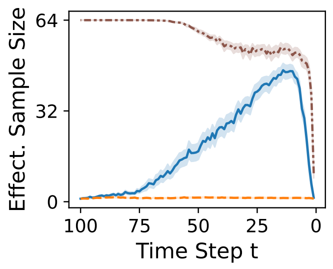
Figure G. MNIST class-conditional generation: random samples selected from 64 random runs conditioned on class 6 under different twist scales. Top to bottom: TDS ($K = 64$), TDS ($K = 2$), TDS-IS ($K = 64$), TDS-IS ($K = 2$) and Guidance. In general, moderately large twist scales improves the sample quality. However, overly large twist scale (e.g. 10) would distort the digit shape with more artifacts, though retaining useful features that may allow a neural network classifier to identity the class.



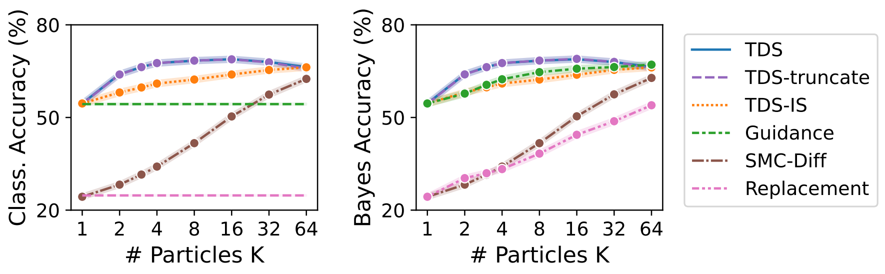
(a) $\mathbf{M}$ = “half”. ESS trace.



(b) $\mathbf{M}$ = “half”. Class. accuracy (left), and Bayes accuracy (right) v.s. K.



(c) $\mathbf{M}$ = “quarter”. ESS trace.



(d) $\mathbf{M}$. = “quarter”. Class. accuracy (left), and Bayes accuracy (right) v.s. K.

Figure H. MNIST image inpainting. Results for observed dimension $\mathbf{M}$ = “half” are shown in the top panel, and $\mathbf{M}$ = “quarter” in the bottom panel. (i) Left column: ESS traces are averaged over 100 different images inpainted by TDS, TDS-IS, and SMC-Diff, all with $K = 64$ particles. TDS's ESS is generally increasing untill the final 20 steps where it drops to around 1, suggesting significant particle collapse in the end of generation trajectory. TDS-IS's ESS is always around 1. SMC-Diff has higher particle efficiency compared to TDS. (ii) Right column: Classification accuracy and Bayes accuracy are averaged over 10k images. In general increasing the number of particles K would improve the performance of all samplers. TDS and TDS-truncate have the highest accuracy among all given the same K. (iii) Finally, comparison of top and bottom panels shows that in a harder inpainting problem where $\mathbf{M}$ = “quarter”, TDS's has a higher ESS but lower CA and BA.

normalized. We define the *Bayes accuracy* (BA) as

$$\mathbb{1}\left\{\hat{z}(y) = z^*\right\}, \quad \text{with} \quad \hat{z}(y) := \arg\max_{z=1:10} \sum_{k=1}^{K} w_k^0 p(z; x_k^0), \tag{31}$$

where $\hat{z}(y)$ is viewed as an approximation to the Bayes optimal classifier $\hat{z}^*(y)$ given by

$$\hat{z}^*(y) := \arg\max_{z=1:10} p_\theta(z \mid x_{\mathbf{M}}^0 = y) = \arg\max_{z=1:10} \int p_\theta(x^0 \mid x_{\mathbf{M}}^0 = y) p(z; x^0) dx^0. \tag{32}$$

(In Equation (32) we assume the classifier $p(\cdot; x^0)$ is the optimal classifier on full images.)

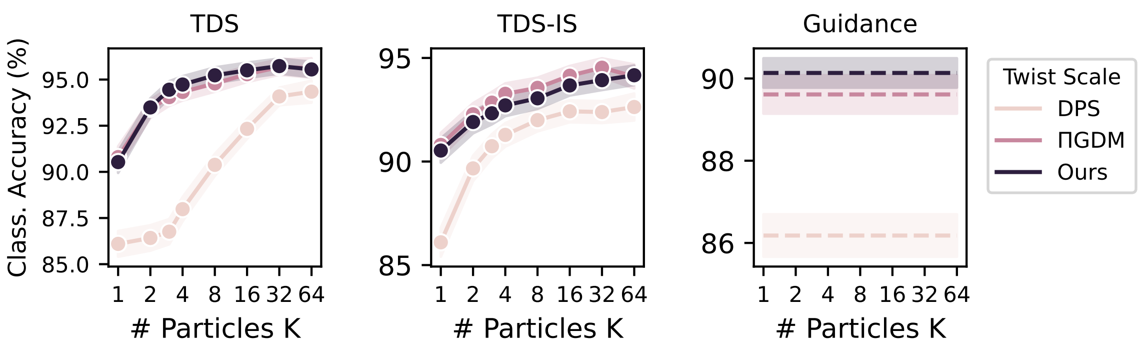
We also consider *classification accuracy* (CA) defined as the following

$$\sum_{k=1}^{K} w_k^0 \mathbb{1}\{\hat{z}(x_k^0) = z^*\}, \quad \text{with} \quad \hat{z}(x_k^0) := \arg\max_{z=1:10} p(z; x_k^0). \tag{33}$$

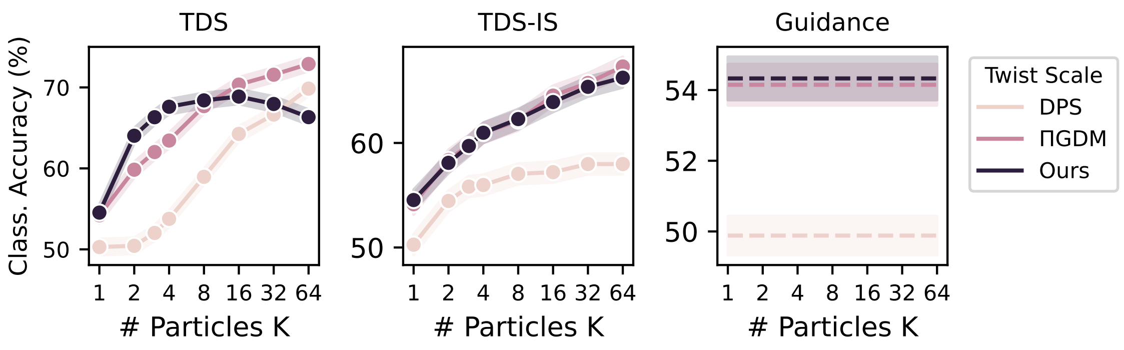
BA and CA evaluate different aspects of a sampler. BA is focused on the optimal prediction among multiple particles, whereas CA is focused on the their weighted averaged prediction.

Comparison results of different methods. Figure H depicts the ESS trace, BA and CA for different samplers. The overall observations are similar to the observations in the class-conditional generation task in Appendix B.2, except that SMC-Diff and Replacement methods are not available there.

Replacement has the lowest CA and BA across all settings. Comparing TDS to SMC-Diff, we find that SMC-Diff's ESS is consistently greater than TDS; however, SMC-Diff is outperformed by TDS in terms of both CA and BA.



(a) $\mathbf{M}$ = “half”. Classification accuracy under different twist scale schemes.



(b) $\mathbf{M}$ = “quarter”. Classification accuracy under different twist scale schemes.

Figure I. MNIST image inpainting: Ablation study on twist scales. Top: observed dimensions $\mathbf{M}$ = “half”. Bottom: $\mathbf{M}$ = “quarter”. In most case, the performance of our choice of twist scale is similar to that of ΠGDM, and is better compared to DPS.

We also note that despite Guidance’s CA is lower, its BA is comparable to TDS. This result is due to that as long as Guidance generates a few good-quality samples out of K particles, the optimal prediction can be accurate, thereby resulting in a high BA.

Ablation study on twist scales. We compare the following three twist scale schemes:

1. DPS: $\gamma_t := 2\|\hat{x}_0(x^t)_{\mathbf{M}} - y\|_2 \sigma_t^2$, adapted from (Chung et al., 2023) (see Algorithm 1).
2. ΠGDM: $\gamma_t := \sigma_t^2/\sqrt{\bar{\alpha}_t}$, adapted from (Song et al., 2023) (see Algorithm 1).
3. Ours: $\gamma_t := \frac{\tilde{\sigma}_t^2 \tau^2}{\tilde{\sigma}_t^2 + \tau^2}$ where $\tilde{\sigma}_t^2 := \frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}$ and $\tau^2 = 0.12$ is the population variance of x^0 estimated from training data.

Figure I shows the classification accuracy of TDS, TDS-IS and Guidance with different twist scale schemes. We find that our choice has similar similar performance to that of ΠGDM, and outperforms DPS in most cases. Exceptions are when $\mathbf{M}$ = “quarter” and for large K, TDS with twist scale choice of ΠGDM or DPS has higher CA, as is shown in the left panel in Figure Ib.

G. Supplementary motif-scaffolding methodological details

Unconditional model of protein backbones. We here use the FrameDiff, a diffusion generative model described by Yim et al. (2023). FrameDiff parameterizes protein N-residue protein backbones as a collection of rigid bodies defined by rotations and translations as $x^0 = [(R_1, z_1), \ldots, (R_N, z_N)] \in SE(3)^N$. $SE(3)$ is the special Euclidean group in three dimensions (a Riemannian manifold), each $R_n \in SO(3)$ is a rotation matrix and each $z_n \in \mathbb{R}^3$ in a translation. Together,

R_n and z_n describe how one obtains the coordinates of the backbone atoms $C, C-\alpha$, and N for each residue by translating and rotating the coordinates of an *idealized* residue with $C-\alpha$ carbon and the origin. The conditioning information is then a motif $y = x^0_{\mathbf{M}} \in SE(3)^{|\mathbf{M}|}$ for some $\mathbf{M} \subset \{1, \ldots, N\}$. FrameDiff is a continuous time diffusion model and includes the number of steps as a hyperparmeter; we use 200 steps in all experiments. We refer the reader to (Yim et al., 2023) for details on the neural network architecture, and details of the forward and reverse diffusion process.

Twisting on $SE(3)^N$. We use the Riemannian extension of TDS in Appendix E with twisting functions defined as $\tilde{p}_\theta(y \mid x^t) = \mathcal{TN}_{\hat{x}_0^M(x^t)}(y; 0, \bar{\sigma}_t^2)$. Because this approximation has isotropic covariance it factorizes across rotations and translations; for the translations (for the forward diffusion is variance preserving) the tangent normal is simply a usual normal distribution. For the rotational component of the twisting functions, in constructing the twisting functions we approximate the log density of the tangent normal using a the squared Frobenius norm, which provides a close approximation for t close to 0 (Watson et al., 2022). When running TDS, we use an effective sample size threshold of $K/2$ in all experiments; that is, we trigger resampling steps only once the effective sample size computed from intermediate weights drops below half the number of particles, as is commonly done to improve efficiency of SMC algorithms (see e.g. Naesseth et al., 2019, Chapter 2.2.2). See supplementary code for implementation details.

Evaluation details. For our self-consistency evaluation we use ProteinMPNN (Dauparas et al., 2022) with default settings to generate 8 sequences for each sampled backbone. Positions not indicated as resdesignable in (Watson et al., 2022, Methods Table 9) are held fixed. We use AlphaFold (Jumper et al., 2021) for forward folding. We define a "success" as a generated backbones for which at least one of the 8 sequences has backbone atom with both scRMSD ¡ 1 Å on the motif and scRMSD ¡ 2 Å on the full backbone. We benchmarked TDS on 24/25 problems in the benchmark set introduced (Watson et al., 2022, Methods Table 9). A 25th problem (6VW1) is excluded because it involves multiple chains, which cannot be represented by FrameDiff. Because `FrameDiff` requires specifying a total length of scaffolds. In all replicates, we fixed the scaffold the median of the `Total Length` range specified by Watson et al. (2022, Methods Table 9). For example, 116 becomes 125 and 62-83 becomes 75.

G.1. Additional degrees of freedom in motif location, rotation, and translation

Prior motif-scaffolding approaches have required of pre-specification of the location of the motif within the final scaffold, translation and rotation of the motif (Trippe et al., 2023; Wang et al., 2022; Watson et al., 2022). However, these nuisance degrees of freedom are ancillary to the biochemical function of a motif, which is determined by its internal geometry. We sought to eliminate these degrees of freedom by applying and extending the approach described in Section 2.

Motif locations degrees of freedom. We eliminate degrees of freedom associated with motif locations, by treating the sequence indices as a mask $\mathbf{M}$ and applying Equation (17). Because the number of motif placements grows combinatorially with number of residues in the backbone and motif, we (i) restrict to masks which place the motif indices in a pre-specified order that we do not permute, and (ii) when there are still too many possible placements sub-sample randomly to obtain $\mathcal{M}$ of at most some maximum length. When motif specifications include multiple contiguous residues from the same chain in the source PDB file we do not separate them. For each offset considered, we constrain the ordering of the motif segments, but do not enforce the spacing between segments specified in the "contig" description (Watson et al., 2022).

Motif orientation and translation degrees of freedom. To eliminate the translational degree of freedom, we simply remove the center of mass from the provided motif and the motif residues when computing the likelihood by redefining $p(y; x^0, \mathbf{M}) = \delta_{y-\bar{y}}(x^0_{\mathbf{M}} - \bar{x^0}_{\mathbf{M}})$. To eliminate the rotational degree of freedom, we wish to condition on the event that there exists a rotation R such that $y = R(x^0_{\mathbf{M}} - \bar{x^0}_{\mathbf{M}})$. Then the likelihood becomes $p(y; x^0, \mathbf{M}) = \int p_R(R)\delta_y(Rx^0_{\mathbf{M}})dR$, where $p_R(R)$ is the uniform distribution on the manifold of rotation matrices (the Haar measure). Because we cannot compute this integral exactly, we use an M sample Monte Carlo approximation from which we define the twisting function as $\tilde{p}_\theta(y \mid x^t) := M^{-1}\sum_{m=1}^M \tilde{p}_\theta(y \mid x^t; R_m, \mathbf{M})$, with $R_m \overset{iid}{\sim} p_R$. We use the same collection of R_m's for each particle and each step of generation. We suspect that future work may be able to handle this integral more accurately, and with greater compute efficiency, or eliminate the need for approximately including this integral altogether, by for example aligning predictions on the motif.

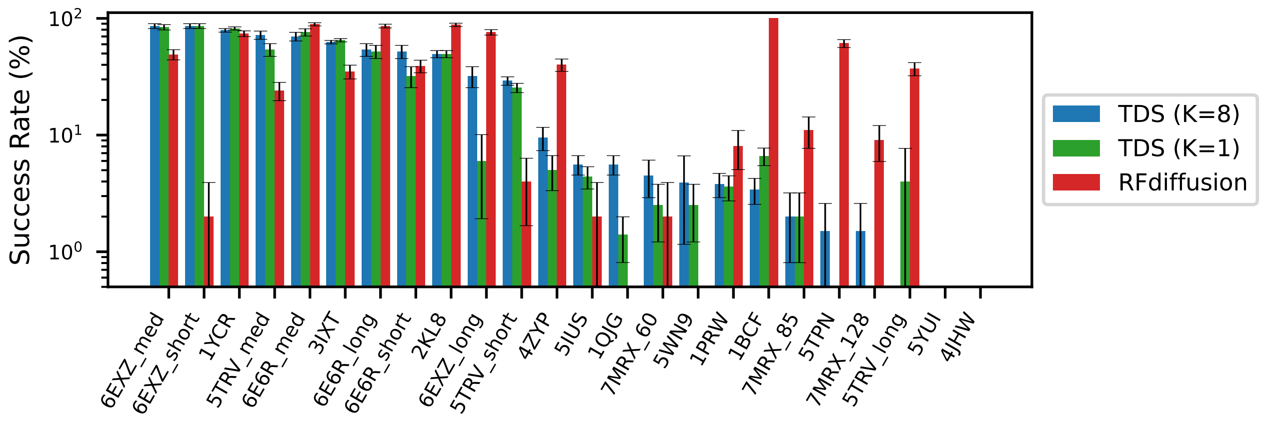


Figure J. Results on full motif-scaffolding benchmark set. The y-axis is the fraction of successes across 200 replicates. Error bars are ± 2 standard errors of the mean.

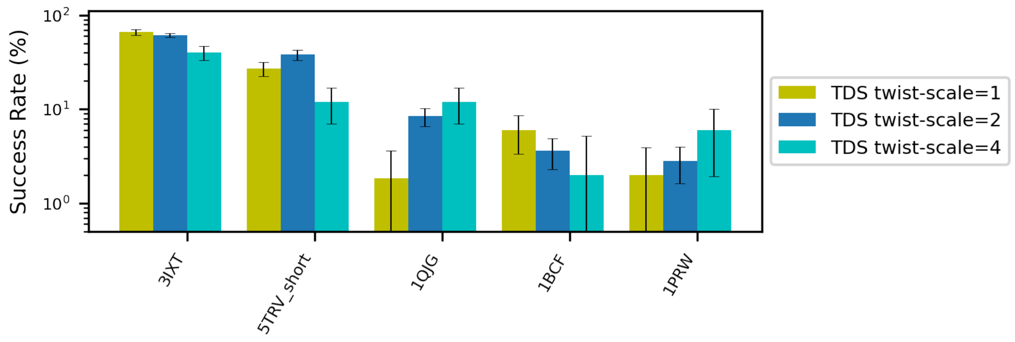


Figure K. Increasing the twist-scale has a different impact across motif-scaffolding benchmark problems.

G.2. Additional Results

Full benchmark results. Figure J compares the success rates of TDS with $K = 8$ and $K = 1$ particles to RFdiffusion on the benchmark set of 24 problems.

Impact of twist scale on additional motifs. Figure C showed monotonically increasing success rates with the twist-scale. However, this trend does not hold for every problem. Figure K demonstrates this by comparing the success rates with different twist-scales on five additional benchmark problems.

Effective sample size varies by problem. Figure L shows two example effective sample size traces over the course of sample generation. For `6EXZ-med` resampling was triggered 38 times (with 14 in the final 25 steps), and for `5UIS` resampling was triggered 63 times (with 13 in the final 25 steps). The traces are representative of the larger benchmark.

Application of TDS to RFdiffusion: We also tried applying TDS to RFdiffusion (Watson et al., 2022); RFdiffusion is a diffusion model that uses the same backbone structure representation as FrameDiff, and because it was trained with a mixture of conditional and unconditional training examples we reasoned that TDS should apply to RFdiffusion as well (when conditioning information if not provided as input). However, we were unable to compute numerically stable gradients (with respect to either the rotational or translational components of the backbone representation); this lead to twisted proposal distributions that were similarly unstable and trajectories that frequently diverged even with one particle. We suspect this instability owes to RFdiffusion's structure prediction pretraining and limited fine-tuning, which may allow it to achieve good performance without having fit a smooth score-approximation.

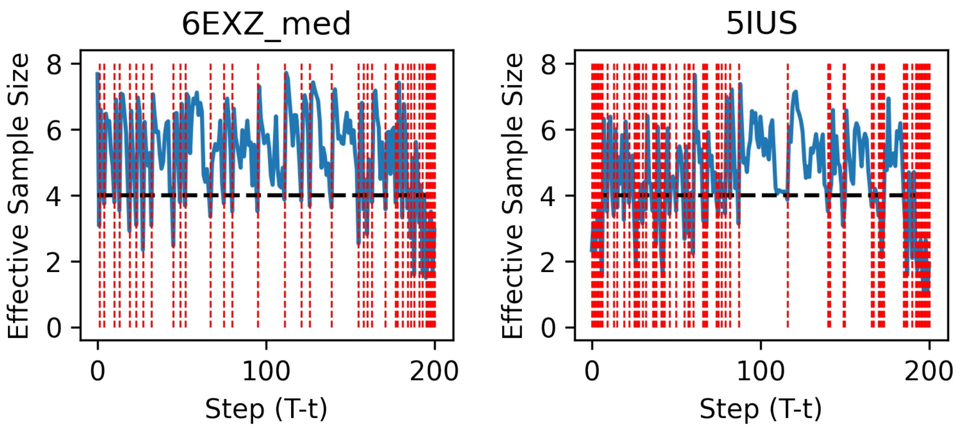


Figure L. Effective sample size traces for two motif-scaffolding examples (Left) `6EXZ-med` and (Right) `5IUS`. In both case $K = 8$ and a resampling threshold of $0.5K$ is used. Dashed red lines indicate resampling times.