

IMPERIAL

Imperial College London
Department of Mathematics

**Sequential Monte Carlo Guided
Diffusion Sampling for General
Optimisation**

Brendan Dowling

CID: 02458886

Supervised by Dr O. Deniz Akyildiz

August 29, 2024

Submitted in partial fulfilment of the requirements for the MSc in
Statistics at Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: Brendan Dowling

Date: August 29, 2024

Acknowledgements

I want to give special thanks to my supervisor, Dr Deniz Akyildiz, who has provided excellent guidance over the course of this work. The dedication of his time towards answering my questions and suggesting avenues to explore went beyond expectations and was greatly appreciated. I also want to thank Benjamin Boys for meeting with me several times, providing insightful and guiding discussions related to this research, and for providing access to his code. I also want to give special thanks to my family for their continued support of my studies and for providing me the resources needed to excel. Lastly, I want to thank my fiancée for her support over this challenging past year; without her this thesis could not have happened.

Abstract

Diffusion models have been shown to learn underlying data distributions, even in high-dimensional settings as demonstrated by their state-of-the-art performance on image, video and molecular synthesis tasks. From a Bayesian perspective, this makes them suitable as prior models. Recent research has explored how to accurately *guide* the diffusion process for posterior sampling. Such methods have primarily focused on solving ill-posed inverse problems by approximating the time conditional score function to use in the reverse sampling steps. In this paper, we present a novel and general framework, **SMCDiffOpt**, for guiding diffusion models with sequential Monte Carlo methods. The framework considers inverse problems as a special case of optimisation, where we've reframed the optimisation task as a sampling problem from an annealed Boltzmann-Gibbs distribution defined by the objective, using the diffusion model as a prior. This framework can be applied even in settings where gradients are not available, making it both computationally efficient and applicable in black-box optimisation settings. Relatedly, this framework does not necessitate conditional score approximations though can flexibly accommodate them for better sampling efficiency. We demonstrate the efficacy of the algorithm through experiments on synthetic and real-world inverse problems and optimisation tasks.

Contents

| | |
|---|------------|
| 1. Introduction | 1 |
| 2. Background | 3 |
| 2.1. Inverse Problems | 3 |
| 2.2. General Optimisation through Sampling | 4 |
| 2.3. Diffusion Models | 5 |
| 2.4. Sequential Monte Carlo | 9 |
| 2.5. Related Work | 11 |
| 3. SMC-Guided Conditional Diffusion Sampling | 13 |
| 3.1. SMCDiffOpt for General Optimisation Problems | 13 |
| 3.2. Inverse Problems as a Special Case | 16 |
| 3.3. Related Work | 19 |
| 4. Experiments and Results | 21 |
| 4.1. Gaussian Mixture Model Inverse Problem | 21 |
| 4.2. Branin Function Optimisation | 24 |
| 4.3. Black-Box Optimisation | 27 |
| 5. Discussion | 29 |
| References | 32 |
| Appendices | A.1 |
| A. Additional Tables and Figures | A.1 |

| | | |
|------|---|------|
| B. | Proofs | A.4 |
| C. | Further Experimental Details | A.7 |
| D. | Additional Background and Related Works | A.8 |
| D.1. | Diffusion Models | A.8 |
| D.2. | Non-Linear Case | A.9 |
| D.3. | FPS-SMC Observation Noising | A.10 |

1. Introduction

Score-based diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Y. Song & Ermon, 2020) have garnered significant attention for their ability to model complex data distributions, particularly in high-dimensional spaces such as image (Dhariwal & Nichol, 2021; Rombach et al., 2021), video (Ho et al., 2022), and molecular (Xu et al., 2022) synthesis. These models operate by iteratively transforming noise into structured data, using a learned (Stein) score function, to effectively enable sampling from the underlying data distribution. Consequently, diffusion models have recently been considered as prior models for Bayesian posterior sampling tasks such as solving inverse problems (Y. Song & Ermon, 2020) and, more recently, optimisation (Guo et al., 2024; Kong et al., 2024; Krishnamoorthy et al., 2023).

There are roughly two core approaches to this posterior sampling task: training a conditional score network for the specific inverse problem (Nichol et al., 2021; Saharia et al., 2021, 2022; Y. Song & Ermon, 2020) or optimisation task (Guo et al., 2024; Krishnamoorthy et al., 2023); taking a pre-trained score network and guiding the generative process using approximations of the conditional score for inverse problems (Boys et al., 2023; Chung et al., 2022; J. Song et al., 2023) or gradients of the objective for optimisation (Kong et al., 2024). We focus on the latter approach since it enables zero shot inference — we can take any diffusion model and use it for any inverse/optimisation tasks without needing access to labelled samples.

We can view inverse problems as a special case of optimisation where the objective is the posterior density (i.e. maximum *a posteriori* estimation of the state (data) from the

observations/conditioning variable). Hence, through the optimisation lens, we may view the prior diffusion model as imposing constraints on the optimisation procedure. These constraints ensure that outcomes of the optimisation procedure lie on or near the data manifold, representing realistic configurations (Kong et al., 2024).

Recent work has considered using sequential Monte Carlo (SMC) methods (Chopin & Papaspiliopoulos, 2020) to correct for the approximation errors incurred during the guiding process (Cardoso et al., 2023; Dou & Song, 2023; Trippe et al., 2023; Wu et al., 2023). However, such frameworks have strictly focused on solving linear inverse problems. In this paper, we introduce **SMCDiffOpt**, a versatile framework that combines diffusion models with SMC methods to address both general optimisation and inverse problems. Our framework views optimisation as a posterior sampling problem from an annealed Boltzmann distribution, using diffusion models as priors and employing SMC methods to more effectively target the posterior distribution. It's also flexible, operating in zero-shot, accommodating different proposal distributions, and being able to operate without access to gradients of the objective — this makes it particularly effective in black-box optimisation settings where only function evaluations are possible.

We contrast **SMCDiffOpt** with standard gradient-based optimisers on a synthetic optimisation task, highlighting its advantage at navigating complex optimisation landscapes, including those with multiple optima. We also test it on the real-world SuperConductor black-box optimisation task (Trabucco et al., 2022), benchmarking it against other state-of-the-art approaches. Additionally, we demonstrate its performance and flexibility in solving inverse problems by comparing its posterior sampling efficiency to alternative methods on a synthetic Gaussian mixture model example.

The paper is organised as follows: Section 2 reviews the relevant background on optimisation, inverse problems, diffusion models, and SMC methods, as well as mentioning some related and alternative approaches. Section 3 introduces the **SMCDiffOpt** framework, detailing its application to both general optimisation tasks and inverse problems. Section 4 presents experimental results, and Section 5 discusses the broader implications of our findings and suggests directions for future research.

2. Background

2.1. Inverse Problems

Definition 2.1.1 (Inverse Problem with Gaussian Noise). Given a data-point $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$, denote some lossy measurement by

$$\mathbf{y} = m(\mathbf{x}) + \sigma_y \epsilon \in \mathcal{Y}, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}_{d_y}) \quad (1)$$

where $m : \mathcal{X} \rightarrow \mathcal{Y} \subseteq \mathbb{R}^{d_y}$ is some *measurement operator*, and $\sigma_y \in \mathbb{R}^+$ controls the measurement noise. The goal of an inverse problem is to recover \mathbf{x} from \mathbf{y} . We call \mathcal{X} the *state space* and \mathcal{Y} the *measurement space*.

Definition 2.1.2 (Linear Inverse Problem). A linear inverse problem is an inverse problem where the measurement operator is such that $h(\mathbf{x}) = A\mathbf{x}$ for some matrix $A \in \mathbb{R}^{d_y \times d_x}$ called the *measurement matrix*.

Example 2.1.1 (Gaussian Deblurring). Suppose we have some \mathbf{x} representing a flattened $h \times w$ image (ignoring channels for simplicity). Let $h(\mathbf{x})$ represent the convolution of \mathbf{x} with a Gaussian kernel. Given the discrete nature of images, this kernel can be represented as a $k \times k$ matrix. Given that the convolution operator is linear, we can form some matrix A from the kernel matrix (by using it to form a block Toeplitz matrix) so that $h(\mathbf{x}) = A\mathbf{x}$. It follows that if we have some blurry image, \mathbf{y} , generated according to [Equation 1](#), the goal of inferring the unblurred image, \mathbf{x} , represents a linear inverse problem.

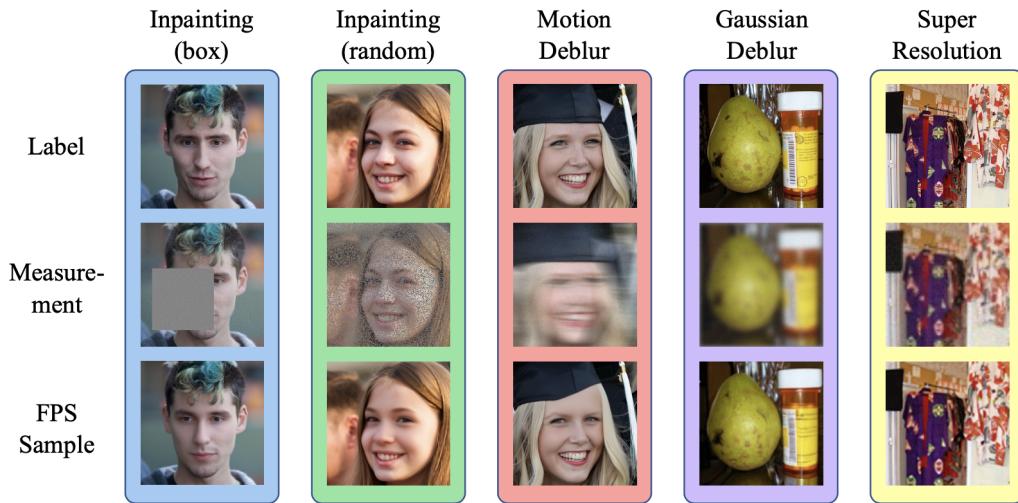


Figure 2.1.: Examples of inverse problems for image data (Dou & Song, 2023).

Remark 2.1.1 (Bayesian Solution to Ill-posed Inverse Problems). Typically $d_x > d_y$, leading to a many-to-one $\mathbf{x} \rightarrow \mathbf{y}$ mapping (Chung et al., 2022), and requiring some *prior* information about \mathbf{x} . We call such an inverse problem *ill-posed*. If we assume some prior distribution of the data, $p(\mathbf{x})$, “solving” the inverse problem amounts to sampling from some posterior

$$p(\mathbf{x} \mid \mathbf{y}) \propto p(\mathbf{x})g(\mathbf{y} \mid \mathbf{x}) = p(\mathbf{x})\mathcal{N}(\mathbf{y} \mid m(\mathbf{x}), \sigma_y^2 \mathbf{I}_{d_y}).$$

For many problems, $p(\mathbf{x})$ is generally unknown or does not conjugate with $g(\mathbf{y} \mid \mathbf{x})$, necessitating numerical methods to sample from $p(\mathbf{x} \mid \mathbf{y})$. \diamond

2.2. General Optimisation through Sampling

Definition 2.2.1 (Gibbs Measure / Boltzmann Distribution). Let $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ be a state space and let $h : \mathcal{X} \rightarrow \mathbb{R}$ be a measurable function, called the *energy function* or

Hamiltonian. The Gibbs measure is the probability density function over \mathcal{X} given by

$$q(\mathbf{x}; \beta) = \frac{\exp\{-\beta h(\mathbf{x})\}}{Z(\beta)} \quad (2)$$

where $\beta > 0$ is the *inverse temperature* parameter and $Z(\beta)$ is the normalizing constant (also known as the partition function), defined by

$$Z(\beta) = \int_{\mathcal{X}} \exp\{-\beta h(\mathbf{x})\} d\mathbf{x}.$$

Proposition 2.2.1 (Optimisation via Sampling (Hwang, 1980; Kong et al., 2024)). Assume $h \in C^3(\mathbb{R}^{d_x}, \mathbb{R})$ with $\{\mathbf{x}_i^*\}_{i=1}^M$ the set of its minimisers. Let p be a density on \mathbb{R}^d such that there exists $i_0 \in \{1, \dots, M\}$ with $p(\mathbf{x}_{i_0}^*) > 0$. Then Q_β , the distribution with density w.r.t the Lebesgue measure $\propto q(\mathbf{x}; \beta)p(\mathbf{x})$, weakly converges to Q_∞ as $\beta \rightarrow \infty$ and we have that

$$Q_\infty = \frac{\sum_{i=1}^M a_i \delta_{\mathbf{x}_i^*}}{\sum_{i=1}^M a_i} \quad (3)$$

with $a_i = p(\mathbf{x}_i^*) \det(\nabla^2 h(\mathbf{x}_i^*))^{-1/2}$

Remark 2.2.1 (Annealing). Proposition 2.2.1 tells us that by properly *tempering* β (i.e. slowly increasing it), we can globally optimise (minimise or maximise via negation) the function h . Note here that because β is the *inverse* temperature, we refer to increasing it as *annealing* (lowering the temperature). The density function p can be interpreted as some *prior* distribution we use to sample the points \mathbf{x} (ideally such that more ‘mass’ is placed near the optima, $\{\mathbf{x}_i^*\}_{i=1}^M$, *a priori*). \diamond

2.3. Diffusion Models

Definition 2.3.1 (Stein Score). Given a probability density function $p(\mathbf{x})$ on \mathbb{R}^{d_x} , the Stein score function is given by $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, the gradient of the log-density with respect

to the data (*not* the distribution's parameters). Going forwards, we refer to the Stein score simply as the *score*, in-line with convention in related literature.

Definition 2.3.2 (Forwards Noising Process (Dou & Song, 2023)). Let \mathbf{x}_0 be a sample from some p_{data} distribution. Define the *forward noising process* as a Markov chain

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^T \mathcal{N}(a_t \mathbf{x}_{t-1}, b_t^2 \mathbf{I}_{d_x})$$

where $\{(a_t, b_t)\}_{t=1}^T$ differ depending on formulation of the problem, and T is typically quite large (≈ 1000). a_t and b_t can be interpreted as the *memory retention factor* and *noise magnitude* of the forward process which typically decrease and increase, respectively, with time.

Definition 2.3.3 (Forward Marginal (Dou & Song, 2023)). Given some forward noising process defined by $\{(a_t, b_t)\}_{t=1}^T$, the *forward marginal* is given by

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(c_t \mathbf{x}_0, d_t^2 \mathbf{I}_{d_x})$$

where c_t, d_t are derived from a_t and b_t . c_t essentially represents the signal strength (from the original sample) while d_t represents the cumulative noise. It follows that we should have $c_T \approx 0, d_T \approx 1$ since the goal is to fully destructure the data into $\mathcal{N}(0, \mathbf{I}_{d_x})$ noise.

Definition 2.3.4 (Backwards Denoising Process (Dou & Song, 2023)). Given some forward noising process, we assume¹ some backwards process ($t = T \rightarrow 0$) to be likewise a Markov chain with one-step backwards transition kernel given by

$$p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(u_t \cdot \hat{\mathbf{x}}_0(\mathbf{x}_t) + v_t \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t), w_t^2 \mathbf{I}_{d_x}) \quad (4)$$

¹This assumption is well-founded for large T since the forward processes associated SDE (see Remark D.1) can be analytically reversed using the score thanks to the result of Anderson (1982). This, along with reducing the discretisation error of sampling from the SDE, is why we take T as large as feasible.

with u_t, v_t some functions of a_t, b_t , $q(\mathbf{x}_t) = \int q(\mathbf{x}_t | \mathbf{x}_0) p(\mathbf{x}_0) d\mathbf{x}_0$, and

$$\hat{\mathbf{x}}_0(\mathbf{x}_t) := \mathbb{E} \{ \mathbf{x}_0 | \mathbf{x}_t \} = \frac{\mathbf{x}_t + d_t^2 \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)}{c_t} \quad (5)$$

derived from Tweedie's formula.

Remark 2.3.1 (DDPM Representation). Under the de-noising diffusion probabilistic model (DDPM) of Ho et al. (2020), we take

$$a_t = \sqrt{\alpha_t} \quad b_t = \sqrt{\beta_t} \quad c_t = \sqrt{\alpha_t} \quad d_t = \sqrt{1 - \bar{\alpha}_t} \quad (6)$$

and

$$u_t = \sqrt{\bar{\alpha}_{t-1}}^2 \quad v_t = -\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1}) \quad w_t = \sqrt{\beta_t \cdot \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}} \quad (7)$$

with $\alpha_t = 1 - \beta_t$, for some $\beta_t \in (0, 1)$ known as the *noise schedule*, and $\bar{\alpha}_t = \prod_{\tau=1}^t \alpha_\tau$. For this paper, we primarily consider the DDPM formulation for simplicity. However, other formulations, such as de-noising diffusion implicit models (DDIM) of J. Song et al. (2020) (see Remark D.2) and Predictor-Corrector (PC) Y. Song et al. (2021), are equally applicable to our proposed framework; the only requirement is that the formulations conform to the above representations of the forwards and backwards process, and enable sampling at discrete time points. \diamond

Remark 2.3.2 (Noise Schedule). Under the DDPM framework, the core parameter to tune is the noising schedule, β_t . Typically, we take this as an increasing function from some very low β_{\min} to some β_{\max} . A linear schedule is the obvious and common choice, but other schedules, such as the cosine schedule of Nichol and Dhariwal (2021), can lead to better sampling performance. For this paper, the exact details of the schedule are not of significant importance; for experiments we use the linear schedule. \diamond

Given Definition 2.3.4, it follows that if we have access to the score, we can sample from p_{data} by sampling $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I}_{d_x})$ ($\approx q(\mathbf{x}_T | \mathbf{x}_0)$) and iterating backwards in time according to Equation 4 until $t = 0$.

²In Dou and Song (2023) this is incorrectly given as $\sqrt{\alpha_{t-1}}$.

Unfortunately, $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$ is intractable³. As such, we train a score-approximating (Y. Song et al., 2021) neural network, denoted $s_\theta(\mathbf{x}_t, t)$. We defer to the extensive literature (foundationally Ho et al. (2020), Y. Song et al. (2021) and Nichol and Dhariwal (2021)) on how precisely to train such a network, but generally this is achieved via minimisation of a “re-weighted variant of the evidence lower bound” (Y. Song et al., 2021)

$$\theta^* = \arg \min_{\theta} \sum_{t=1}^T (1 - \alpha_t) \mathbb{E}_{\mathbf{x}_0 \sim \hat{p}_{\text{data}}} \mathbb{E}_{\mathbf{x}_t | \mathbf{x}_0 \sim q_{\alpha_t}} [\|s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q_{\alpha_t}(\mathbf{x}_t | \mathbf{x}_0)\|_2^2]$$

where \hat{p}_{data} is the empirical data distribution (i.e. based on our training samples), and q_{α_t} is the forward marginal at time t .

Plugging in this score network to Equation 4, we yield an approximate *backwards transition kernel*, $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) \approx q(\mathbf{x}_{t-1} | \mathbf{x}_t)$, which we use to sample backwards in time. Formally, we sample according to

$$p_\theta(\mathbf{x}_0) = \int p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) d\mathbf{x}_{1:T}, \quad p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I}_{d_x}) \quad (8)$$

with, given a well trained network, $p_\theta(\mathbf{x}_0) \approx p_{\text{data}}(\mathbf{x}_0)$. Denote the mean of this transition kernel by $\mu_t(\mathbf{x}_t, \theta)$ and the variance by Σ_t . For DDPM, we have

$$\mu_t(\mathbf{x}_t, \theta) = \frac{1}{\sqrt{\alpha_t}} \mathbf{x}_t + \frac{\beta_t}{\sqrt{\alpha_t}} s_\theta(\mathbf{x}_t, t) \quad (9)$$

$$\Sigma_t = \frac{\beta_t(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \cdot \mathbf{I} \quad (10)$$

Remark 2.3.3 (De-noising). Ho et al. (2020) originally consider learning a model to predict the cumulative noise added by the forward process; hence, the name “De-noising Diffusion Probabilistic Models”. Using D.1, the two approaches are essentially equivalent; we refer to DDPM by the parametrisation in 2.3.1 rather than learning a noise-estimating model. \diamond

³Note this is not $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0)$, which is tractable but “useless” for sampling since we don’t know \mathbf{x}_0 a priori (it’s what we’re trying to sample); this quantity is useful, however, for training where we *do* know \mathbf{x}_0 .

2.4. Sequential Monte Carlo

Definition 2.4.1 (State Space Model). Let $\mathbf{x}_t \in \mathcal{X}$ be a *state vector* at time t , which encapsulates all the information about some system necessary to predict future states (perhaps stochastically). Let $\mathbf{y}_t \in \mathcal{Y}$ be some observation vector at time t , representing (potentially noisy) measurements of the system. Let $\phi : \mathcal{X} \rightarrow \mathcal{X}$ be some function we call the *transition function*. Let $m : \mathcal{X} \rightarrow \mathcal{Y}$ be a measurement operator. Then the following dynamical system represents a (Gaussian) *state space model* (SSM):

$$\mathbf{X}_t | \mathbf{X}_{t-1} = \mathbf{x}_{t-1} \sim \mathcal{N}(\phi(\mathbf{x}_{t-1}), \sigma_x^2 \mathbf{I}_{d_x}) \quad (11)$$

$$\mathbf{Y}_t | \mathbf{X}_t = \mathbf{x}_t \sim \mathcal{N}(m(\mathbf{x}_t), \sigma_y^2 \mathbf{I}_{d_y}) \quad (12)$$

We refer to Equation 11 as the *transition kernel*, with density $f(\mathbf{x}_t | \mathbf{x}_{t-1})$, and Equation 12 as the *likelihood*, with density $g(\mathbf{y}_t | \mathbf{x}_t)$.

Definition 2.4.2 (Bayesian Filtering). Suppose we have access to measurements $\mathbf{y}_{0:t}$ emitted from some SSM. The task of using such measurements to infer the hidden states, $\mathbf{x}_{0:t}$, is known as *Bayesian filtering*. We may be interested in the *joint filtering distribution*, with density $p(\mathbf{x}_{0:t} | \mathbf{y}_{0:t})$, or the *filtering distribution*, with density $p(\mathbf{x}_t | \mathbf{y}_{0:t})$. Focusing on the latter, given the setup of Definition 2.4.1, we can derive the two-step recursive relationship:

$$\begin{aligned} p(\mathbf{x}_t | \mathbf{y}_{0:t-1}) &= \int p(\mathbf{x}_t, \mathbf{x}_{t-1} | \mathbf{y}_{0:t-1}) d\mathbf{x}_{t-1} \\ &= \int f(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{y}_{0:t-1}) d\mathbf{x}_{t-1} \end{aligned} \quad (13)$$

$$\begin{aligned} p(\mathbf{x}_t | \mathbf{y}_{0:t}) &\propto p(\mathbf{x}_t, \mathbf{y}_t | \mathbf{y}_{0:t-1}) \\ &\propto p(\mathbf{x}_t | \mathbf{y}_{0:t-1}) g(\mathbf{y}_t | \mathbf{x}_t) \end{aligned} \quad (14)$$

Generally, however, the integral in Equation 13 and normalising constant in Equation 14 are intractable, necessitating approximate methods.

Definition 2.4.3 (Sequential Monte Carlo for Filtering⁴). Sequential Monte Carlo (SMC) provides a method for approximately sampling from the sequence of distributions as described in [Definition 2.4.2](#) (Chopin & Papaspiliopoulos, 2020). It works by generating N particles, $\{\mathbf{x}_t^{(i)}\}_{i=1}^N$, according to some initial distribution, $r_0(\cdot)$, moving them according to some *proposal distribution*, $r_t(\cdot | \mathbf{x}_{t-1}, \mathbf{y}_t)$, weighting the particles according to some weighting function, $\omega_t(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{y}_t)$, and resampling⁵ the particles according to their (self-normalised) weights ($W_t^{(i)}$). That is, the algorithm takes the following steps from $t = 1, \dots, T$:

- **Propose:** $\tilde{\mathbf{x}}_t^{(i)} \sim r_t(\cdot | \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t), \quad i = 1, \dots, N$
- **Weight:** $W_t^{(i)} \leftarrow \omega_t^{(i)} \leftarrow \omega_t(\tilde{\mathbf{x}}_t^{(i)}, \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t) \quad i = 1, \dots, N$
- **Resample:** $\{\mathbf{x}_{0:t}^{(i)}\}_{i=1}^N \sim \text{Multinomial}\left(\{\tilde{\mathbf{x}}_{0:t}^{(i)}\}_{i=1}^N; \{W^{(i)}\}_{i=1}^N\right)$

At the last step, we can yield an empirical distribution over the particles

$$\hat{p}(\mathbf{x}_T | \mathbf{y}_{0:T}) = \sum_{i=1}^N W_T^{(i)} \delta_{\mathbf{x}_T^{(i)}}(\mathbf{x}_T)$$

which asymptotically (in N) approximates the true posterior distribution, $p(\mathbf{x}_T | \mathbf{y}_{0:T})$ (Chopin & Papaspiliopoulos, 2020; Del Moral, 2011).

Remark 2.4.1 (Weight Function). The weight function's purpose is to correct for discrepancies between the proposal and true posterior, $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{y}_t)$, which it aims to approximate. In the filtering case for an SSM as in [Definition 2.4.1](#), the weight function is given by the ratio of the two (Chopin & Papaspiliopoulos, 2020):

$$\omega_t^{(i)} = \frac{p(\tilde{\mathbf{x}}_t^{(i)} | \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)}{r_t(\tilde{\mathbf{x}}_t^{(i)} | \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)} \propto \frac{f(\tilde{\mathbf{x}}_t | \mathbf{x}_{t-1}^{(i)})g(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(i)})}{r_t(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{y}_t)}. \quad (15)$$

◊

⁴Also known as Particle Filtering.

⁵We present multinomial resampling for simplicity, but in practice alternative methods, such as systematic resampling, are used due to inefficiencies in the multinomial scheme (Chopin & Papaspiliopoulos, 2020)

Remark 2.4.2 (Resampling). The purpose of resampling is to avoid the issue of weight-degeneracy. Without resampling, after several iterations (i.e. for large T), most particles will end up having negligible weight (since without resampling we need to multiply all previous weights for the particle to get its time t weight) and eventually a single particle will dominate the approximation. Without resampling, we would not get the desired asymptotic results. \diamond

2.5. Related Work

The goal of conditionally sampling from p_{data} (learnt via a diffusion model) based on some \mathbf{y} has been the center of significant recent research. The principle approach of most methods is to approximate $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t \mid \mathbf{y})$, and plug this in to [Equation 4](#) in place of its unconditional counterpart. The obvious approach is to train a conditional model to learn some $s_\theta(\mathbf{x}_t, \mathbf{y}, t)$ and use this. Indeed, this is the approach that modern text-to-image, image-to-image, and image-to-video applications take ([Li et al., 2023](#); [Nichol et al., 2021](#); [Rombach et al., 2021](#); [Saharia et al., 2021](#)). This approach generally achieves the best conditional sampling but is limited in application; it requires training a model specific to each particular inverse problem (of which the aforementioned are examples) which may be costly. Furthermore, such training requires having labelled data which may impose limitations in certain settings and for certain tasks.

A separate branch of research, and the area with which this work aligns, considers taking a pre-trained *unconditional* diffusion model (i.e. one where we just have a parametrised estimator of $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$) and *guide* the diffusion process to enable conditional sampling. This is typically achieved by exploiting the relationship

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y} \mid \mathbf{x}_t). \quad (16)$$

Replacing the first term with our pre-trained score network, optimal guidance can be achieved then by well-approximating $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{y} \mid \mathbf{x}_t)$ (which is intractable). Such ap-

proximation methods are generally non-specific to the inverse problem, enabling taking any unconditional diffusion model and using it to sample from the desired posterior distribution. The diffusion-posterior-sampling method of Chung et al. (2022) was foundational, and can be applied to solve general inverse problems. J. Song et al. (2023) and, recently, Boys et al. (2023) have iterated on this with more analytically refined⁶ approaches (though these are only applicable to linear inverse problems).

Given the difficulty of such approximations, and the sequential nature of the sampling procedure from diffusion models, significant recent research (Cardoso et al., 2023; Dou & Song, 2023; Janati et al., 2024; Trippe et al., 2023; Wu et al., 2023) has considered using SMC methods (with time running from $t = T \rightarrow 0$) to instead more directly target the posterior (or rather, the intermediate distributions $p(\mathbf{x}_t | \mathbf{y})$) and circumvent the need to compute this term. Our framework is most heavily inspired by the scheme of Cardoso et al. (2023); in Section 3.3 we demonstrate how their scheme is very closely related to our framework. In fact, each of these methods can be applied under our framework since their differences essentially lie primarily in their choice of proposal distribution.

However, our framework is generalised to solving any optimisation task (not just inverse problems). Through this lens, our framework’s objective and formulation aligns with that of Kong et al., 2024, with both employing Proposition 2.2.1 and essentially targeting the distribution induced by the diffusion model and the objective’s associated annealed Boltzmann distribution. However, where they guide the diffusion based on the score of the objective (with further corrections via Metropolis-adjusted Langevin dynamics), we instead use interacting particle systems (i.e. SMC). This circumvents the need to compute gradients of the objective function, making our method even better suited to black-box optimisation tasks since it removes the need to train a surrogate model.

⁶See Boys et al., 2023 Section 5 for an excellent summary of how the methods co-relate.

3. SMC-Guided Conditional Diffusion Sampling

3.1. SMCDiffOpt for General Optimisation Problems

Consider some function $f : \mathcal{X} \rightarrow \mathbb{R}$ whose minima, $\{\mathbf{x}_i^*\}_{i=1}^M$, we wish to find. Consider $q(\mathbf{x}; \gamma) \propto \exp\{-\gamma f(\mathbf{x})\}$. Suppose we have some pre-trained score network which we can use to sample from p_{data} via some p_θ in T time-steps. Take p_θ as a prior for use in Proposition 2.2.1. Write $\mathbf{x}_0 := \mathbf{x}$. Naive targeting of $\pi(\mathbf{x}_0) \propto q(\mathbf{x}_0; \beta)p_\theta(\mathbf{x}_0)$ amounts to sampling \mathbf{x}_0 according to Equation 8, and using some form of MCMC-type sampling with annealing for β to target Equation 3. In cases where f is non-convex and/or induces a highly rugged loss landscape, sampling can be either highly inefficient or challenging to tune (Kong et al., 2024).

Instead, we consider using the intermediate unconditional transition functions of the diffusion model, $p_\theta(\mathbf{x}_t | \mathbf{x}_{t+1})$, and guide these with the objective function. Using some annealing schedule, $\gamma(t)$, we construct a sequence $\{q_t(\mathbf{x}_t; \gamma_t)\}_{t=0}^T$. Loosely (though concretely in Section 3.2), we may view q_t as a likelihood, with f emitting observations \mathbf{y}_t based on \mathbf{x}_t . That is, we may consider $q_t(\mathbf{x}_t; \gamma_t)$ as some $g(\mathbf{y}_t | \mathbf{x}_t)$. The task is now frameable through a Bayesian filtering lens, with the intermediate targets being

$$p(\mathbf{x}_t | \mathbf{y}_{T:t}) \propto p(\mathbf{x}_t | \mathbf{y}_{T:t+1})g(\mathbf{y}_t | \mathbf{x}_t) \quad (17)$$

(as in [Equation 14](#) with time reversed). SMC methods provide a principled method of targeting these and ultimately $p(\mathbf{x}_0 \mid \mathbf{y}_{0:T}) \approx q_\infty \propto q(\mathbf{x}; \infty) p_\theta(\mathbf{x})$ of [Proposition 2.2.1](#), yielding $\{\mathbf{x}_i^*\}_{i=1}^M$.

Remark 3.1.1 (Diffusion Models Learn Data Manifolds). A key feature which enables p_θ to act as a good prior is the ability of diffusion models to implicitly learn the data manifold (De Bortoli et al., 2021; Pidstrigach, 2022; Wenliang & Moran, 2023). In the context of optimisation, the diffusion model essentially enables us to only consider optimisation over ‘sensible’ values of \mathbf{x} ; a feasibility constraint. Put differently, the unconditional diffusion prior acts as a regularisation to encourage optimisation of f over $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ as opposed to over all of \mathbb{R}^{d_x} (Guo et al., 2024). \diamond

If we consider the *bootstrap* proposal, $r(\cdot \mid \mathbf{x}_{t+1}) = p_\theta(\cdot \mid \mathbf{x}_{t+1})$, then the weight function in [Equation 15](#) reduces to $g(\mathbf{y}_t \mid \mathbf{x}_t) = q_t(\mathbf{x}; \gamma_t)$, the ‘likelihood’. In the case where $q_t(\mathbf{x}_t; \gamma_t)$ is ‘peaky’ over its support (which is guaranteed for t close to 0 assuming reasonable annealing), this will generally be a poor proposal (Chopin & Papaspiliopoulos, 2020), as it is akin to the ‘observations’, \mathbf{y}_t , being informative about the ‘states’, \mathbf{x}_t . The obvious remedy is to choose a better proposal. In the case of general optimisation, however, this is a non-trivial task.

Instead we consider using an *auxiliary variable*, adjusting the weight for each particle based on their *previous* likelihood. Heuristically, this implies we weight more heavily those particles whose likelihood *significantly increased* compared to their previous position. This imposes a form of regularisation that implicitly discourages concentration on high likelihood regions near the initial distribution, $p(\mathbf{x}_T)$. Geometrically, this regularisation may be viewed as dynamically flattening the optimisation landscape for each particle based on its position. The role of $\gamma(t)$ is to provide global annealing of the landscape, gradually attenuating this dynamic effect and ultimately applying [Proposition 2.2.1](#) to enable sampling from Q_∞ . The full algorithm is described in [Algorithm 1](#).

Remark 3.1.2 (Auxiliary Variable). The concept of using an auxiliary variable to adjust the weights is the foundation of the *auxiliary particle filter* (Chopin & Papaspiliopoulos,

Algorithm 1 SMCDiffOpt for General Optimisation

Require: $f : \mathcal{X} \rightarrow \mathbb{R}$, an objective function to *minimise* (take $-f$ to *maximise*).
Require: $s_\theta(\mathbf{x}_t, t)$, a pre-trained score network (or noise-predictor and apply [Proposition D.1](#)).
Require: $r(\cdot | \mathbf{x}_{t+1})$, a proposal distribution. Default to $p_\theta(\cdot | \mathbf{x}_{t+1})$.
Require: $\gamma(t)$, some annealing schedule.
Require: N , number of particles to use.

Draw $\mathbf{x}_T^{(i)} \sim \mathcal{N}(0, \mathbf{I}_{d_x})$, $i = 1, \dots, N$

for $t \leftarrow T - 1$ to 0 **do**

- for** $i \leftarrow 1$ to N **do**
- Move:** $\tilde{\mathbf{x}}_t^{(i)} \sim r(\cdot | \mathbf{x}_{t+1})$.
- Compute weight:**

$$\omega_t^{(i)} \leftarrow \frac{\exp\left\{-\gamma(t)f(\tilde{\mathbf{x}}_t^{(i)})\right\}}{\exp\left\{-\gamma(t+1)f(\mathbf{x}_{t+1}^{(i)})\right\}}. \quad (18)$$

Normalise weight: $W_t^{(i)} \leftarrow \frac{\omega_t^{(i)}}{\sum_{i=1}^N \omega_t^{(i)}}$.

Resample: $\{\mathbf{x}_{T:t}^{(i)}\}_{i=1}^N \sim \text{Multinomial}\left(\{\tilde{\mathbf{x}}_{T:t}^{(i)}\}_{i=1}^N; \{W_t^{(i)}\}_{i=1}^N\right)$.

end for

end for

return $\{\mathbf{x}_0^{(i)}\}_{i=1}^N$, some summary statistic computed using $\{w_0^{(i)}\}_{i=1}^N$, or some $\{\mathbf{x}_0^{(i)}\}_{i \in \mathcal{I}}$ subset which achieve the (perhaps within a threshold) minimal value amongst all particles.

2020) and can be justified analytically. The exact procedure for auxiliary particle filters differs very slightly from that described in [Definition 2.4.3](#), mainly pertaining to the order of operations and how the intermediate distributions are approximated from the particles. \diamond

Remark 3.1.3 (Gradient-free). Note that we don't require any gradient information about f . This makes [Algorithm 1](#) suitable in black-box optimisation settings where we might only be able to evaluate f and can't access its gradient (by back-propagation or analytically). However, where we can compute the gradient, $\nabla_{\mathbf{x}_t} f(\mathbf{x}_t)$, we can instead consider using these gradients to 'twist' (Wu et al., 2023) the proposal distribution, yielding a proposal $r_t(\cdot | \mathbf{x}_{t+1})$ closer to the true 'posterior', $p(\cdot | \mathbf{x}_{t+1}, \mathbf{y}_t)$, reducing the variance of the weights and making the sampler more efficient (enabling lower number

of particles, N). We discuss alternative proposals in more detail in Section 3.3, though these typically are only applicable in inverse problem settings. Alternatively, we may consider adding additional steps after moving the particles, nudging (perhaps a subset of) the particles as in Akyildiz and Míguez (2020). Note here though that even with access to gradients, the algorithm offers substantial advantages over standard gradient-based optimisation in cases where f has steep gradients or is highly non-convex. We demonstrate this experimentally in Section 4.2. \diamond

Remark 3.1.4 (Annealing-only). In principle we could remove the auxiliary variable, and just tune the $\gamma(t)$ annealing schedule. In practice, however, this is extremely challenging and adds even heavier dependence on the nature of f ; in cases where it has steep gradients, the annealing schedule needs to be very gentle, which is often infeasible due to the fixed T time-steps that the diffusion sampler takes (one would need to consider intermediate diffusions with, for example, extra Langevin steps in a fashion like Janati et al. (2024) to work around). The auxiliary variable helps make the tuning of $\gamma(t)$ much easier by mitigating its importance in adequately flattening the landscape. This was observed strongly in Section 4.2, for example. \diamond

Remark 3.1.5 (Adaptive Resampling (Chopin & Papaspiliopoulos, 2020)). In practice, we don't usually resample at every iteration in Algorithm 1, instead resampling only when particle diversity falls below some certain threshold (typically using effective sample size as a metric). Given this, the weight formula in Equation 18 is adjusted by pre-multiplying by the previous un-normalised weight (or 1 if resampled); the procedure is otherwise identical. The choice of, say, ESS ratio is critical though for Algorithm 1 and can significantly alleviate the difficulty in tuning $\gamma(t)$. \diamond

3.2. Inverse Problems as a Special Case

Proposition 3.2.1 (Obvservation Generation). Let $\mathbf{y} = \mathbf{y}_0 = A\mathbf{x}_0 + \sigma_y \epsilon$ be some noisy linear measurement we have about a desired sample, \mathbf{x}_0 . Construct a sequence $\{\mathbf{y}_t\}_{t=1}^T$

by $\mathbf{y}_t = a_t \cdot \mathbf{y}_{t-1}$ with a_t of Definition 2.3.2. It follows that

$$\mathbf{y}_t = c_t \cdot \mathbf{y} \quad (19)$$

from which we may conclude

$$\mathbf{Y}_t \mid \mathbf{X}_0 = \mathbf{x}_0 \sim \mathcal{N}(c_t \cdot A\mathbf{x}_0, c_t^2 \sigma_y^2 \cdot \mathbf{I}_{d_y})$$

and

$$\mathbf{Y}_t \mid \mathbf{X}_t = \mathbf{x}_t \sim \mathcal{N}(A\mathbf{x}_t, c_t^2 \sigma_y^2 \cdot \mathbf{I}_{d_y} + d_t^2 \cdot AA^\top) \quad (20)$$

with $g(\mathbf{y}_0 \mid \mathbf{x}_0) = g(\mathbf{y} \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{y}; A\mathbf{x}_0, \sigma_y^2 \cdot \mathbf{I}_{d_y})$ exactly the measurement density.

Proof. See Appendix B. □

Remark 3.2.1 (Shrinking). The important characteristic of Equation 19 is that it ‘shrinks’ the observations towards 0 (in tandem, theoretically, with how \mathbf{x}_0 is shrunk towards zero according to its forward process). This essentially creates an *aligned* observation sequence so that the likelihoods (i.e. density evaluations of Equation 20) are sensibly sized, enabling their usage in SMC algorithms. ◊

Note that this generation does not depend on the clean sample, \mathbf{x}_0 . As such, it can be generated prior to running the guided sampling and we may view sampling from $p(\mathbf{x}_0 \mid \mathbf{y})$ as a Bayesian filtering task. This is described succinctly in Dou and Song (2023):

We can generate a sample \mathbf{x}_0 from the Bayesian posterior distribution $p_\theta(\mathbf{x}_0 \mid \mathbf{y}_0)$ by first sampling from $p_\theta(\mathbf{y}_{1:T} \mid \mathbf{y}_0)$ before sampling from $p_\theta(\mathbf{x}_0 \mid \mathbf{y}_{0:T})$.

This is because $p_\theta(\mathbf{x}_0 \mid \mathbf{y}_0) = \int p_\theta(\mathbf{x}_0 \mid \mathbf{y}_{0:T}) p_\theta(\mathbf{y}_{1:T} \mid \mathbf{y}_0) d\mathbf{y}_{1:T}$.

Proposition 3.2.2 (Inverse Problem Objective). Consider the objective function

$$f(\mathbf{x}_t) = -\frac{1}{2\sigma_y^2 c_t^2} \|\mathbf{y}_t - A\mathbf{x}_t\|^2 \quad (21)$$

with \mathbf{y}_t constructed from \mathbf{y} according to Equation 19. Plugging this in to Equation 18, ignoring particle indices, the weight function reduces to

$$\omega_t \leftarrow \frac{\mathcal{N}(\mathbf{y}_t; A\tilde{\mathbf{x}}_t, c_t^2 \sigma_y^2 \cdot \mathbf{I}_{d_y} + d_t^2 \cdot AA^\top)^{\gamma(t)}}{\mathcal{N}(\mathbf{y}_{t+1}; A\mathbf{x}_{t+1}, c_{t+1}^2 \sigma_y^2 \cdot \mathbf{I}_{d_y} + d_{t+1}^2 \cdot AA^\top)^{\gamma(t+1)}} \quad (22)$$

We see this is exactly the ratio of the likelihoods, annealed according to $\gamma(t)$. Its interpretation is exactly as in Section 3.1 except the notion of observations and posterior densities is now exact.

Remark 3.2.2 (MAP). The choice of f in Equation 21 essentially states that we're choosing to maximise the likelihood, $g(\mathbf{y}_t | \mathbf{x}_t)$, subject to the constraint/regularisation imposed by the prior diffusion model through $p_\theta(\cdot | \mathbf{x}_{t+1})$. Based on 2.4.3, it follows that running Algorithm 1 seeks ultimately to yield those samples $\{\mathbf{x}_0^{(i)}\}_{i=1}^N$ which are the *maximum a posteriori* estimates of \mathbf{x}_0 , exactly as desired for solving an inverse problem.

◊

Remark 3.2.3 (Inner Tempering). By symmetry of the Gaussian distribution, we have

$$\begin{aligned} \mathcal{N}(\mathbf{y}_t; A\mathbf{x}_t, c_t^2 \sigma_y^2 \cdot \mathbf{I}_{d_y} + d_t^2 \cdot AA^\top) &= \mathcal{N}(A\mathbf{x}_t; \mathbf{y}_t, c_t^2 \sigma_y^2 \cdot \mathbf{I}_{d_y} + d_t^2 \cdot AA^\top) \\ &= \mathcal{N}(A\mathbf{x}_t; c_t \mathbf{y}, c_t^2 \sigma_y^2 \cdot \mathbf{I}_{d_y} + d_t^2 \cdot AA^\top) \end{aligned}$$

We see that there is essentially an inner-level of tempering according to $c(t)$, defined by the unconditional diffusion model. As such, for inverse problems we can generally take $\gamma(t) = 1 \forall t$.

◊

3.3. Related Work

Taking $\mu_{t+1}(\mathbf{x}_{t+1}, \theta)$, Σ_{t+1} as the mean (e.g. Equation 9) and variance (e.g. Equation 10) of the backwards transition kernel, $p_\theta(\mathbf{x}_t | \mathbf{x}_{t+1})$, and Ξ_t as the covariance of the Monte Carlo Guided Diffusion (MCGdiff) (Cardoso et al., 2023) and Filter Posterior Sampling SMC (FPS-SMC) (Dou & Song, 2023) frameworks consider essentially equivalent proposals

$$r(\tilde{\mathbf{x}}_t | \mathbf{x}_{t+1}, \mathbf{y}_t) \propto p_\theta(\tilde{\mathbf{x}}_t | \mathbf{x}_{t+1})g(\mathbf{y}_t | \tilde{\mathbf{x}}_t) = \mathcal{N}(\mu_{t+1}^*(\mathbf{x}_{t+1}, \mathbf{y}_t, \theta), \Sigma_{t+1}^*)$$

(by Normal-Normal conjugacy) with

$$\begin{aligned}\Sigma_{t+1}^* &= (\Sigma_{t+1}^{-1} + A^\top \Xi^{-1} A)^{-1} \\ \mu_{t+1}^*(\mathbf{x}_{t+1}, \mathbf{y}_t, \theta) &= \Sigma_{t+1}^* \cdot (\Sigma_{t+1}^{-1} \mu_{t+1}(\mathbf{x}_{t+1}, \theta) + A^\top \Xi^{-1} \mathbf{y}_t)\end{aligned}$$

and \mathbf{y}_t constructed by Equation 19¹ and g as in Equation 20. Plugging this in to Equation 15, we get

$$\begin{aligned}\omega_t &= \int p_\theta(\tilde{\mathbf{x}}_t | \mathbf{x}_{t+1})g(\mathbf{y}_t | \tilde{\mathbf{x}}_t) d\mathbf{x}_t = g(\mathbf{y}_t | \mathbf{x}_{t+1}) \\ &= \mathcal{N}(\mathbf{y}_t; A\mu_{t+1}(\mathbf{x}_{t+1}, \theta), \Xi_t + A\Sigma_{t+1}A^\top)\end{aligned}$$

where we apply Proposition B.1. Note this *only* holds when we have a linear inverse problem; otherwise $g(\mathbf{y}_t | \mathbf{x}_{t+1})$ is intractable. In the case of FPS-SMC, these are exactly the weights used in the SMC procedure; in the case of MCGDiff they similarly use the previous likelihood, $g(\mathbf{y}_{t+1} | \mathbf{x}_{t+1})$, as an auxiliary variable. Through this lens, we may view MCGDiff as a special case of Algorithm 1 where we choose

$$f(\tilde{\mathbf{x}}_t) = g(\mathbf{y}_t | \mathbf{x}_{t+1}) = \mathcal{N}(\mathbf{y}_t; A\mu_{t+1}(\mathbf{x}_{t+1}, \theta), \Xi_t + A\Sigma_{t+1}A^\top)$$

¹Dou and Song (2023) considers a slightly different construction which employs “noise-sharing” with the state’s forward-process; see D.

but keep the auxiliary variable (abusing notation) as $f(\mathbf{x}_{t+1}) = g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})$. The interpretation is very similar to before except rather than more strongly weighting those particles whose likelihood *increased* relative to their previous position, we more strongly weight those whose likelihood *will increase with the next transition* relative to their previous position more strongly. The idea is then to properly implement an auxiliary particle filter, and weight particles *before* moving them. This generally should improve the performance since it's essentially ‘forward-looking’. However, it's important to note this can only be applied to *linear inverse problems* due to the intractability of $g(\mathbf{y}_t \mid \mathbf{x}_{t+1})$ generally; it similarly cannot be applied for general optimisation tasks.

With that said, there is nothing prohibiting us from using other proposals, such as that of MCGDiff/FPS-SMC, in [Algorithm 1](#), re-deriving the weight function for [Equation 22](#) from [Equation 15](#) but still using the previous likelihood auxiliary variable. As mentioned in [Remark 3.1.3](#), we could consider using gradients to construct the proposals. Wu et al. (2023) consider $\nabla_{\mathbf{x}_t} \log p_{\mathbf{y} \mid \hat{\mathbf{x}}_0(\mathbf{x}_t, \theta)}$ (with $\hat{\mathbf{x}}_0(\mathbf{x}_t, \theta)$ as in [Equation 5](#)) computed using back-propagation through the network, s_θ , to twist the proposal. Boys et al. (2023) consider $\nabla_{\mathbf{x}_t} \hat{\mathbf{x}}_0(\mathbf{x}_t, \theta)$ to choose a more optimal covariances in $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. Such methods incur additional computational cost and are, again, restricted to linear inverse problems. In such cases, however, the added efficiency of the proposal can (significantly) reduce the number of particles, N , mitigating some of this added computational cost.

In the context of general optimisation, an alternative approach considered by Kong et al. (2024) is to run a guided diffusion using the gradient of the objective, in order to roughly sample regions on the data manifold near to the optima, and then run Langevin steps (i.e. MALA) after to yield better optima. This has the advantage over `SMCDiffOpt` of yielding an independent sampler ($N = 1$), enabling it to be *embarrassingly parallelised*. The downside to this method is its reliance on the gradient meaning it necessitates training a surrogate model for the objective function in black-box settings. Of course, the idea of supplementary sampling steps after running the guided diffusion is trivially applicable to our method if desired to yield better estimates of the optima.

4. Experiments and Results

4.1. Gaussian Mixture Model Inverse Problem

Beginning with an inverse problem, we consider the synthetic experiment described in Boys et al. (2023) and Cardoso et al. (2023). Consider a Gaussian mixture model (GMM), p_{data} , with 25 equally weighted ($\omega_{i,j} \propto 1$) d_x -dimensional Gaussian random variables with means $\mu_{i,j} := (8i, 8j, \dots, 8i, 8j) \in \mathbb{R}^{d_x}$ for $(i, j) \in \{-2, \dots, 2\}^2$ and unit variance. We generate some d_y -dimensional measurement, \mathbf{y} , according to the following process:

- Draw $\tilde{A} \sim \mathcal{N}(0, 1)^{d_y \times d_x} \in \mathbb{R}^{d_y \times d_x}$ and compute its SVD decomposition, USV^\top .
- Sample $s_{i,j} \sim \mathcal{U}[0, 1]$ for $(i, j) \in \{-2, \dots, 2\}^2$.
- Set $A := U \text{diag}(\{s_{i,j}\}) V^\top$.
- Draw $\mathbf{x}_* \sim p_{\text{data}}$ and set $\mathbf{y} := A\mathbf{x}_* + \sigma_y \epsilon$, $\epsilon \sim \mathcal{N}(\mathbf{0}_{d_y}, \mathbf{I}_{d_y})$.

The goal of the inverse problem is to take \mathbf{y} (with A and σ_y known), and use it to infer \mathbf{x}_* . We consider $d_y < d_x$, making the problem ill-posed. Per Remark 2.1.1, we tackle this by considering $p_{\text{data}}(\mathbf{x})$ as a prior distribution, $g(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}(\mathbf{y}; A\mathbf{x}, \sigma_y \mathbf{I}_{d_y})$ as the measurement density, and then aim to sample from $p(\mathbf{x} \mid \mathbf{y})$.

| d_x | d_y | SMCDiffOpt | DPS | ΠIGD | TMPD |
|-------|-------|-----------------|-----------------|-----------------|-----------------|
| 8 | 1 | 1.35 ± 1.0 | 8.18 ± 7.5 | 3.16 ± 2.72 | 3.31 ± 2.86 |
| | 2 | 0.55 ± 0.43 | 2.72 ± 2.62 | 0.94 ± 0.89 | 1.34 ± 1.25 |
| | 4 | 0.19 ± 0.09 | 0.95 ± 0.9 | 0.09 ± 0.03 | 0.37 ± 0.33 |
| 80 | 1 | 1.65 ± 1.41 | 5.03 ± 4.63 | 3.08 ± 2.71 | 2.42 ± 1.71 |
| | 2 | 1.19 ± 1.07 | 3.14 ± 3.02 | 1.68 ± 1.62 | 1.35 ± 1.22 |
| | 4 | 1.11 ± 0.93 | 1.32 ± 1.21 | 0.84 ± 0.79 | 1.14 ± 0.9 |

Table 4.1.: Sliced-Wasserstein distances of samples from particle samples.

For this model, given p_{data} is available in closed form, the posterior, $p(\mathbf{x}_* | \mathbf{y})$, is actually available in closed form as another GMM with

$$c_{i,j} := \mathcal{N}(\Sigma (\sigma_y^{-2} \cdot A^\top \mathbf{y} + \mu_{i,j}), \Sigma)$$

$$\tilde{\omega}_{i,j} \propto \omega_{i,j} \mathcal{N}(\mathbf{y}; A\mu_{i,j}, \mathbf{I}_{d_x} + AA^\top)$$

where $\Sigma := (\mathbf{I}_{d_x} + \sigma_y^{-2} \cdot A^\top A)^{-1}$. This enables us to accurately benchmark the different numerical methods.

Supposing p_{data} was not available in closed form / easy to sample from, we may consider using a diffusion model to sample from it. As p_{data} is a GMM, the backwards marginal is also available in closed form allowing us to avoid training a score network, being able to use simple auto-differentiation of the marginal to yield the exact score at intermediate time-steps in a diffusion process. Then using [Equation 4](#), we can sample from p_{data} in some T time-steps. In [Figure A.2a](#) we show prior samples from such an unconditional model (compared with exact posterior samples for a particular inverse problem).

We can then apply our method, as described in [Section 3.2](#) with $\gamma(t) = 1$, to guide the process to enable sampling from the posterior, $p(\mathbf{x} | \mathbf{y})$. We can contrast **SMCDiffOpt** numerically with the non-particle based approaches of [Boys et al. \(2023\)](#), [Chung et al. \(2022\)](#), and [J. Song et al. \(2023\)](#)¹, using sliced-Wasserstein distance ([Boys et al., 2023](#); [Cardoso et al., 2023](#)) to measure how well each method approximates the true posterior distribution based on samples drawn directly from it.

¹See [Section C](#) for discussion as why MCGdiff comparison was excluded.

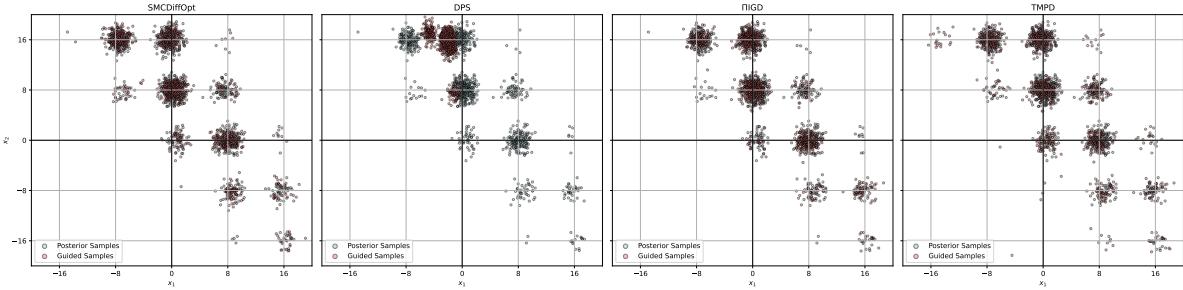


Figure 4.1.: Comparison of samples against true posterior samples; $d_x = 8$; $d_y = 2$.

We experiment for $d_x \in \{8, 80\}$, $d_y \in \{1, 2, 4\}$, and $\sigma_y \in \{0.01, 0.1, 1.0\}$, comparing the performance of **SMCDiffOpt**, diffusion-posterior-sampling (DPS; Chung et al., 2022), pseudo-inverse-guided-diffusion (PIIGD; J. Song et al., 2023), and Tweedie-moment-projected-diffusion (TMPD; Boys et al., 2023). We run the experiment for 10 different seeds, generating 10 different measurement models according to the above steps, using 1000 particles for **SMCDiffOpt** and generating 1000 independent samples for the other methods, all in 1000 time-steps under DDPM sampling. For **SMCDiffOpt**, we resample only when the ESS ratio drops below 0.8.

The results are shown in Table 4.1 rolled up on σ_y ; in Table A.1 we show the full-results split out by σ_y ². The values shown are the median and ranges for the sliced-Wasserstein distance over the seeds. We see that **SMCDiffOpt** outperforms the other methods. In Figure 4.1 we plot the first two co-ordinates of the samples generated by each method overlaying true posterior samples for a specific measurement model with $d_x = 8$, $d_y = 2$, providing visual confirmation. Figure A.2 better highlights how **SMCDiffOpt** is able to accurately guide the unconditional sampler to instead sample from the posterior.

In Figure 4.2 we show the evolution of the true (log) posterior density evaluated on the sample at step t , $p_{\mathbf{x}_0|\mathbf{y}}(\mathbf{x}_t \mid \mathbf{y})$, and (log) likelihood, $g(\mathbf{y}_t \mid \mathbf{x}_t)$, over the course of the guided diffusions under each method. Figure 4.2a shows how well the methods ultimately target the posterior distribution; we see that all but DPS follow similar trajectories. Examining Figure 4.2b in conjunction, we see that the methods appear to operate by first optimising the likelihood to a point before turning to optimise the posterior density,

²This table roughly aligns with Table 4 of Boys et al. (2023), providing some level of validation.

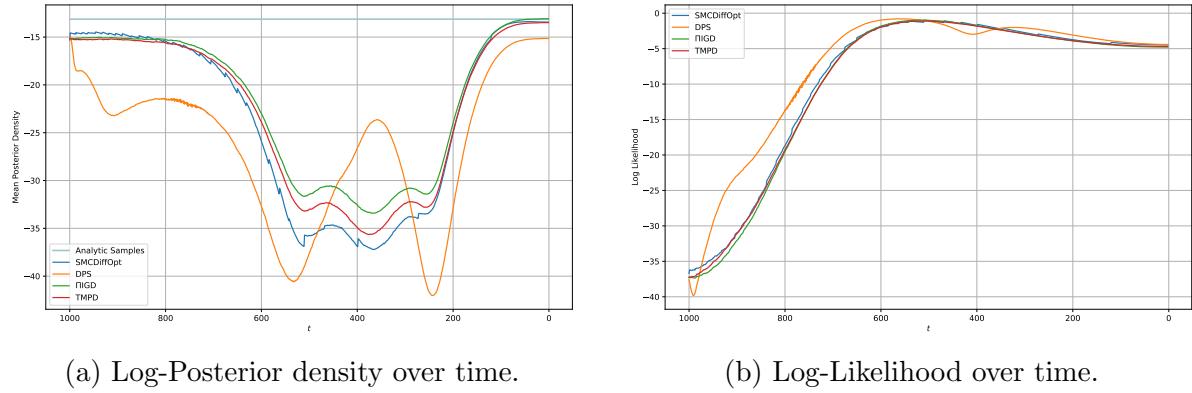


Figure 4.2.: Mean log-posterior and log-likelihood of samples over time with respect to observations, $\{\mathbf{y}_t\}_{t=1}^T$, constructed per Equation 19; associated with Figure 4.1.

allowing the likelihood to decline. Essentially, Figure 4.2 highlights the regularisation effect of the diffusion prior model while optimising Equation 21.

Note that the numeric comparison of **SMCDiffOpt** with the other three methods isn't entirely fair since the former is an interacting-particle algorithm; **SMCDiffOpt** requires several particles to operate whereas the others are independent runs. Reducing the number of particles will almost definitely reduce the relative outperformance of **SMCDiffOpt** over the others. This provides an interesting avenue for future research in the form of an ablation study.

4.2. Branin Function Optimisation

For a general optimisation task, we consider optimising the Branin function as an objective over some constrained region (the data manifold) as detailed in Kong et al. (2024). The Branin function is defined by

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t) \cos(x_1) + s \quad (23)$$

where $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$, $s = 10$, $t = \frac{1}{8\pi}$. It has three global minima located at $(-\pi, 12.275)$, $(\pi, 2.275)$ and $(9.42478, 2.475)$, with a value of 0.397887. We

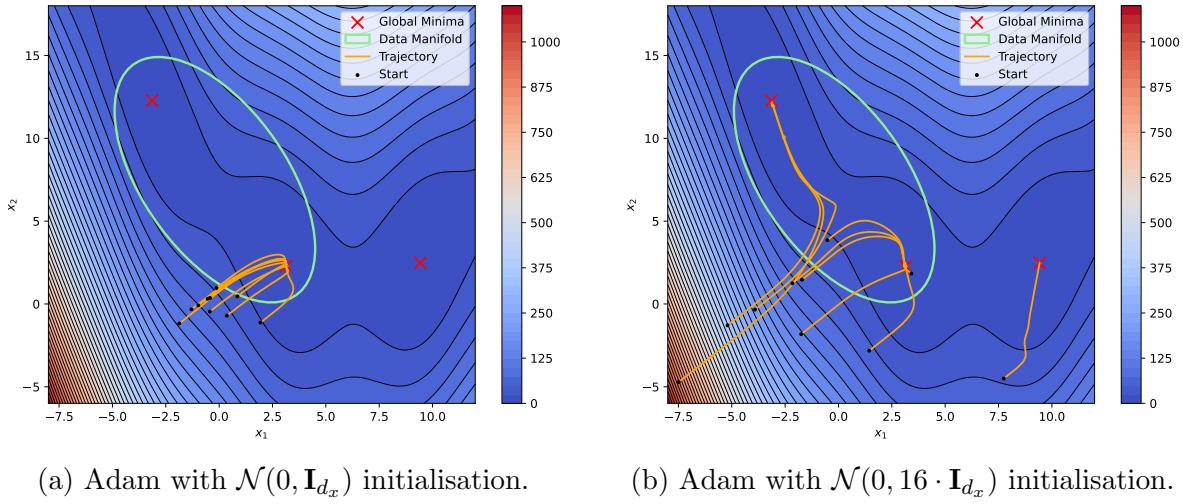


Figure 4.3.: Pitfalls of naive gradient optimisation of the Branin function.

consider some uniform prior sampling distribution, p_{data} , over an ellipse region centred at $(-0.2, 7.5)$ with semi-axis lengths $(3.6, 8)$, tilted 25° . This region covers two of the global minima $(-\pi, 12.275)$ and $(\pi, 2.275)$. See Figure 4.3 to visualise.

Similar to in Section 4.1, here we can easily sample from p_{data} by inverse-transform sampling. Again, we suppose now that we can't do so and suppose also that we're in a high-dimensional setting where performing a simple visual analysis to quickly infer the geometry of the objective function and data manifold is not possible. Our stated objective is to optimise the function over the data manifold, ideally finding the $(-\pi, 12.275)$ and $(\pi, 2.275)$ points, but *not* the $(9.42478, 2.475)$ point (since it's outside the constrained region / not a *valid configuration*).

The obvious approach might be to use a gradient-based optimiser, such as the popular Adam (Kingma & Ba, 2017), adaptive-moment-estimating optimiser. However, without prior knowledge of the data manifold, choosing initial points for the optimiser is challenging. In Figure 4.3a we show choosing 10 (for visual clarity) starting points randomly from a $\mathcal{N}(0, \mathbf{I}_{d_x})$ distribution. Using a base learning rate of 0.05 and 1000 iterations of the optimiser, we see that in this case the optimiser does find one of the optima, but has essentially no chance of finding the other. In Figure 4.3b we re-run the optimiser but with initial points randomly from a $\mathcal{N}(0, 16 \cdot \mathbf{I}_{d_x})$ distribution instead. Here we see that

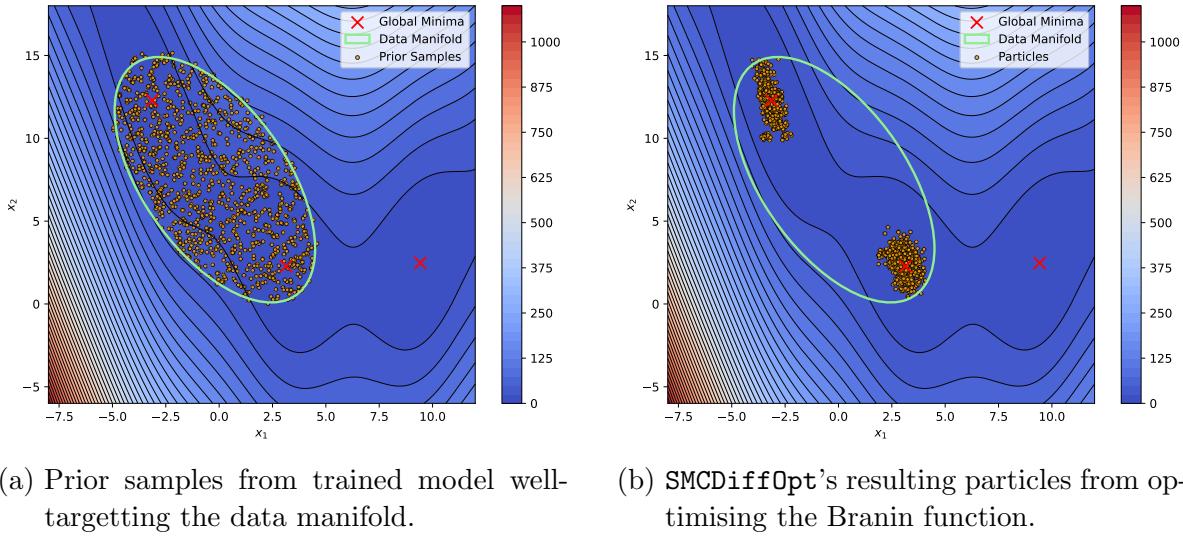


Figure 4.4.: Prior versus optimised samples under guidance through SMCDiffOpt.

while the optimiser found the two desired points, it also found one of the points outside the constrained region — an invalid configuration. Furthermore, we see that one of trajectories didn't converge to an optima; this sample headed towards the saddle-point region between the two optima and essentially got stuck there requiring many iterations to escape. The key issue here is that the optimiser has no guidance towards the data manifold and relies solely on the gradient. Obviously, if we could sample from p_{data} analytically, choosing this as the distribution to pick initial samples is optimal. Since we can't generally, we resort to generative techniques, such as diffusion models.

To apply SMCDiffOpt, we sample training data from p_{data} ³, and use this to train a simple score-based diffusion model (details in [Section C](#)). With such a model, we can now approximately generate samples from p_{data} . More importantly, we can use the model as a prior, using it as transition kernel in [Algorithm 1](#), with the ‘likelihood’ being an (annealed) induced Boltzmann distribution from the Branin function. Choosing $\gamma(t) = 1 - e^{-\zeta T}$ for some small, tunable ζ (e.g. 0.001), we run [Algorithm 1](#) with 1000 particles, resampling when the ESS ratio drops below 0.5. In [Figure 4.4b](#), we see that the procedure has well-discovered the two desired optima; using the final particle weights the

³Obviously, we are supposing this sampling isn't generally possible; we are in a generative modelling situation where we pre-suppose access to a finite number of samples from p_{data} a priori, and train a model on these, hoping it enables general sampling from p_{data} .

precise global minima can be easily discovered. This is further highlighted in [Figure A.2](#) where we show how the mean Branin value across the particles evolves over time.

The remarkable feature of [Algorithm 1](#) is that it can achieve this optimisation without requiring access to the gradient of the objective. As mentioned in [Chapter 3](#), if we did have access to these gradients we could consider using them with [Algorithm 1](#) either as an additional step after moving the particles (“gradient nudging” (Akyildiz & Míguez, 2020)) or as a means of picking a better proposal distribution (“twisting” (Wu et al., 2023)). In such cases, the requisite number of particles and the reliance on the $\gamma(t)$ schedule should likely reduce.

4.3. Black-Box Optimisation

For a black-box optimisation task, we consider the SuperConductor task of Trabucco et al. (2022). Originally proposed in Hamidieh (2018), the goal is “to design a chemical formula for a superconducting material that has a high critical temperature” (Trabucco et al., 2022). The dataset comprises 21,263 86-dimensional vectors and their associated critical temperatures. Each vector represents the “mixture of elements by number of atoms in the chemical formula of each superconductor” (Trabucco et al., 2022). A provided oracle model takes any configuration and gives the associated temperature. This is the black-box function we wish to optimise; we want to find the vector which maximises the oracle.

As suggested in Trabucco et al. (2022), and applied in Kong et al. (2024) and Krishnamoorthy et al. (2023), we use an 80% subset of the data for training. This holds out rows with the top and bottom 10% of temperatures. Training a score network on this data, the hope is this models well the data manifold (allowed configurations) enabling it to act as a strong prior model for optimisation. In [Figure A.2](#) we show an example of a true sample ([Figure A.2a](#)) versus a sample generated by the model ([Figure A.2b](#)). Details of the model’s architecture are described in [Section C](#).

| CbAS | CMA-ES | Gradient Ascent | MINs | REINFORCE | DDOM | DiffOpt | SMCDiffOpt |
|---------------|---------------|-----------------|---------------|---------------|---------------|---------------|----------------------|
| 0.433 ± 0.027 | 0.474 ± 0.021 | 0.510 ± 0.009 | 0.473 ± 0.003 | 0.483 ± 0.015 | 0.560 ± 0.044 | 0.614 ± 0.029 | 0.644 ± 0.024 |

Table 4.2.: Normalised scores for SuperConductor experiment (higher values are better).

Using this model, we can employ **SMCDiffOpt** for finding an optimal configuration. We can determine the performance of **SMCDiffOpt** by yielding an optimised temperature from its final particles (e.g. via sampling and evaluation, or evaluation then averaging; we choose latter for benchmark comparisons). We min-max normalise such a score based on the full dataset to give a more interpretable score; the maximum temperature in the full dataset is 185 and the minimum is ≈ 0 . We run the algorithm with annealing as in Section 4.2, using only 100 particles, resampling when the ESS ratio drops below 0.5.

As in Krishnamoorthy et al. (2023), we benchmark our approach with multiple baselines including “canonical approaches” like gradient ascent and REINFORCE (Sutton et al., 1999), the CMA-ES (Hansen, 2006) evolutionary strategy, and the CbAS (Brookes et al., 2019) and MINs (Kumar & Levine, 2020) recent deep learning methods. We also include the diffusion-based DDOM (Krishnamoorthy et al., 2023) and DiffOpt (Kong et al., 2024) methods in our comparison. In Table 4.2 we present for each method the mean min-max normalised scores and their 95% CLT confidence intervals based on 5 runs of each on different random seeds. We see that **SMCDiffOpt** comfortably outperforms each method, yielding state-of-the-art performance. We show an example configuration generated by the optimiser in Figure A.2c.

Like in Section 4.1, the comparison is not exactly fair since our method uses 100 interacting particles compared to the other methods which are independent ‘samplers’. In black-box settings, this can be a major downside since we may have a strict *query budget* (Krishnamoorthy et al., 2023), only being allowed to evaluate the oracle a finite number of times. In such settings, **SMCDiffOpt** is generally ill-suited; time-respacing, gradient-nudging, or supplementary Langevin steps may provide remedies but we defer investigating applying these ideas to future research.

5. Discussion

We proposed **SMCDiffOpt**, a novel approach for general optimisation based on sequential Monte Carlo guided diffusion models. The method operates in zero-shot, taking any pre-trained unconditional diffusion model and enabling its usage for optimisation and inverse problem tasks. Existing similar diffusion-based methods have been sub-divided into those which solve inverse problems (Boys et al., 2023; Cardoso et al., 2023; Chung et al., 2022; Dou & Song, 2023; J. Song et al., 2023; Wu et al., 2023) and those which solve optimisation tasks (Guo et al., 2024; Kong et al., 2024; Krishnamoorthy et al., 2023). Our approach bridges these domains by framing optimisation as a posterior sampling task and inverse problem solving as a special case of optimisation. For inverse problems, we described how the method can be applied by constructing a sequence of aligned measurements and likelihoods, taking maximisation of these likelihoods as the optimisation objective. We tested **SMCDiffOpt** on a variety of tasks and showed its ability to yield state-of-the-art performance.

Our approach offers several advantages over existing methods. It yields asymptotically exact posterior samples as the number of particle increases, giving significant performance gains over independent samplers in suitable settings. It also operates without requiring access to gradients, making it suitable for black-box optimisation tasks. It's also flexible, accommodating different proposals and incorporation of gradients via nudging. It's also extensible, supporting intermediate or terminal steps for improved sampling.

SMCDiffOpt is not without some limitations. The method inherits the slow sampling speed of diffusion models compared to other generative approaches. More importantly,

the method requires multiple particles to work effectively — in cases of certain inverse tasks more efficient proposals may mitigate this drawback but never fully remove it. Additionally, in the context of general optimisation, the final particles may not strictly adhere to valid configurations — this problem is illustrated in Guo et al. (2024). In many settings, such as image generation, particles being ‘close’ to the data manifold may be sufficient; in others, such as chemical generation, the requirements may be more stringent, with the results of `SMCDiffOpt` requiring further analysis and modification. `SMCDiffOpt` also offers only approximate methods for solving non-linear inverse tasks (see subsection D.2), though it should be noted this is true even of DPS (Boys et al., 2023; Chung et al., 2022). Finally, in certain tasks, tuning of $\gamma(t)$ and the resampling rate (i.e. minimum ESS) may be challenging to tune.

There are several potential avenues for future work. First, a comparison of the relative performance of `SMCDiffOpt` versus other SMC-based samplers (Cardoso et al., 2023; Dou & Song, 2023; Wu et al., 2023) for linear inverse problems would be insightful (see Section C), including plugging their twisted proposals in to Algorithm 1 and measuring the relative efficiency gains. Relatedly, our work lacks an ablation study on the impact of the number of particles, the noising schedule, the resampling rate and the annealing schedule which are the key variables in Algorithm 1. Examining the performance and viability of our method on higher-dimensional settings, such as image generation, would also be fruitful. Similarly, for non-linear inverse problems using the approximate methods described in subsection D.2. An examination the performance of `SMCDiffOpt` under a time-respaced diffusion model would be insightful and if positive suggest better applicability under query budget constraints for black-box optimisation. Fully benchmarking `SMCDiffOpt` on the design-bench black-box optimisation tasks must be carried out to validate the state-of-the-art performance of the algorithm over previous works. Devising a new scoring method which accounts for diversity of optimised samples should be considered too, since this is one of the key benefits of `SMCDiffOpt`. Lastly, an analysis of the theoretical guarantees of running Algorithm 1 should be conducted.

Endmatter

All code for the research is openly available on a [GitHub repository](#). All results in this paper can be easily reproduced by following the instructions of said repository's `README`. The SuperConductor experiment code is separately available on [this repository](#) due to complex dependency requirements. Code was run on an NVIDIA RTX3080.

Black-box data and oracle models are available originally through the [design-bench repository](#).

References

- Akyildiz, Ö. D., & Míguez, J. (2020). Nudging the particle filter. *Statistics and Computing*, 30(2), 305–330. <https://doi.org/10.1007/s11222-019-09884-y>
- Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3), 313–326. [https://doi.org/10.1016/0304-4149\(82\)90051-5](https://doi.org/10.1016/0304-4149(82)90051-5)
- Boys, B., Girolami, M., Pidstrigach, J., Reich, S., Mosca, A., & Akyildiz, O. D. (2023, November 22). *Tweedie Moment Projected Diffusions For Inverse Problems*. arXiv: 2310.06721 [stat]. Retrieved June 23, 2024, from <http://arxiv.org/abs/2310.06721>
- Brookes, D., Park, H., & Listgarten, J. (2019, June 9–15). Conditioning by adaptive sampling for robust design. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (pp. 773–782, Vol. 97). PMLR. <https://proceedings.mlr.press/v97/brookes19a.html>
- Cardoso, G., Idrissi, Y. J. E., Corff, S. L., & Moulaines, E. (2023, October 25). *Monte Carlo guided Diffusion for Bayesian linear inverse problems*. arXiv: 2308.07983 [cs, stat]. <https://doi.org/10.48550/arXiv.2308.07983>
- Chopin, N., & Papaspiliopoulos, O. (2020). *An introduction to sequential Monte Carlo*. Springer.
- Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., & Ye, J. C. (2022). Diffusion Posterior Sampling for General Noisy Inverse Problems. <https://doi.org/10.48550/ARXIV.2209.14687>

- De Bortoli, V., Thornton, J., Heng, J., & Doucet, A. (2021, June 1). *Diffusion Schrödinger Bridge with Applications to Score-Based Generative Modeling*. arXiv.org. Retrieved August 19, 2024, from <https://arxiv.org/abs/2106.01357v5>
- Del Moral, P. (2011). Central Limit Theorems. In *Feynman-Kac formulae: Genealogical and interacting particle systems with applications* (pp. 291–330). Springer. OCLC: 1063493341.
- Dhariwal, P., & Nichol, A. (2021, June 1). *Diffusion Models Beat GANs on Image Synthesis*. arXiv: 2105.05233 [cs, stat]. <https://doi.org/10.48550/arXiv.2105.05233>
- Dou, Z., & Song, Y. (2023). Diffusion Posterior Sampling for Linear Inverse Problem Solving: A Filtering Perspective. Retrieved June 8, 2024, from [https://openreview.net/forum?id=tplXNcHZs1&referrer=%5Bthe%20profile%20of%20Yang%20Song%5D\(%2Fprofile%3Fid%3D~Yang_Song1\)](https://openreview.net/forum?id=tplXNcHZs1&referrer=%5Bthe%20profile%20of%20Yang%20Song%5D(%2Fprofile%3Fid%3D~Yang_Song1))
- Guo, Y., Yuan, H., Yang, Y., Chen, M., & Wang, M. (2024, April 23). *Gradient Guidance for Diffusion Models: An Optimization Perspective*. arXiv: 2404.14743 [cs, stat]. Retrieved July 26, 2024, from <http://arxiv.org/abs/2404.14743>
- Hamidieh, K. (2018). A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154, 346–354. <https://doi.org/10.1016/j.commatsci.2018.07.052>
- Hansen, N. (2006). The CMA evolution strategy: A comparing review. In J. A. Lozano, P. Larrañaga, I. Inza, & E. Bengoetxea (Eds.), *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms* (pp. 75–102). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-32494-1_4
- Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. <https://doi.org/10.48550/ARXIV.2006.11239>
- Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., & Fleet, D. J. (2022, June 22). *Video Diffusion Models*. arXiv: 2204.03458 [cs]. Retrieved August 21, 2024, from <http://arxiv.org/abs/2204.03458>
- Hwang, C.-R. (1980). Laplace's Method Revisited: Weak Convergence of Probability Measures. *The Annals of Probability*, 8(6). <https://doi.org/10.1214/aop/1176994579>

- Janati, Y., Durmus, A., Moulines, E., & Olsson, J. (2024, March 17). *Divide-and-Conquer Posterior Sampling for Denoising Diffusion Priors*. arXiv: 2403.11407 [cs, stat]. Retrieved May 24, 2024, from <http://arxiv.org/abs/2403.11407>
- Kingma, D. P., & Ba, J. (2017, January 29). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs]. <https://doi.org/10.48550/arXiv.1412.6980>
- Kong, L., Du, Y., Mu, W., Neklyudov, K., De Bortoli, V., Wang, H., Wu, D., Ferber, A., Ma, Y.-A., Gomes, C. P., & Zhang, C. (2024, April 29). *Diffusion Models as Constrained Samplers for Optimization with Unknown Constraints*. arXiv: 2402.18012 [cs]. Retrieved July 26, 2024, from <http://arxiv.org/abs/2402.18012>
- Krishnamoorthy, S., Mashkaria, S. M., & Grover, A. (2023, August 21). *Diffusion Models for Black-Box Optimization*. arXiv: 2306.07180 [cs]. Retrieved July 31, 2024, from <http://arxiv.org/abs/2306.07180>
- Kumar, A., & Levine, S. (2020). Model inversion networks for model-based optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 5126–5137, Vol. 33). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/373e4c5d8edfa8b74fd4b6791d0cf6dc-Paper.pdf
- Li, X., Ren, Y., Jin, X., Lan, C., Wang, X., Zeng, W., Wang, X., & Chen, Z. (2023). Diffusion Models for Image Restoration and Enhancement – A Comprehensive Survey. <https://doi.org/10.48550/ARXIV.2308.09388>
- Nichol, A., & Dhariwal, P. (2021, February 18). *Improved Denoising Diffusion Probabilistic Models*. arXiv: 2102.09672 [cs, stat]. Retrieved May 29, 2024, from <http://arxiv.org/abs/2102.09672>
- Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., & Chen, M. (2021). GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. <https://doi.org/10.48550/ARXIV.2112.10741>
- Pidstrigach, J. (2022). Score-Based Generative Models Detect Manifolds. <https://doi.org/10.48550/ARXIV.2206.01018>

- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-Resolution Image Synthesis with Latent Diffusion Models. <https://doi.org/10.48550/ARXIV.2112.10752>
- Saharia, C., Chan, W., Chang, H., Lee, C. A., Ho, J., Salimans, T., Fleet, D. J., & Norouzi, M. (2021). Palette: Image-to-Image Diffusion Models. <https://doi.org/10.48550/ARXIV.2111.05826>
- Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., & Norouzi, M. (2022). Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. <https://doi.org/10.48550/ARXIV.2205.11487>
- Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015, July 7–9). Deep unsupervised learning using nonequilibrium thermodynamics. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd international conference on machine learning* (pp. 2256–2265, Vol. 37). PMLR. <https://proceedings.mlr.press/v37/sohl-dickstein15.html>
- Song, J., Meng, C., & Ermon, S. (2020). Denoising Diffusion Implicit Models. <https://doi.org/10.48550/ARXIV.2010.02502>
- Song, J., Vahdat, A., Mardani, M., & Kautz, J. (2023). Pseudoinverse-guided diffusion models for inverse problems. *International Conference on Learning Representations*. https://openreview.net/forum?id=9_gsMA8MRKQ
- Song, Y., & Ermon, S. (2020, October 10). *Generative Modeling by Estimating Gradients of the Data Distribution*. arXiv: 1907.05600 [cs, stat]. <https://doi.org/10.48550/arXiv.1907.05600>
- Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2021, February 10). *Score-Based Generative Modeling through Stochastic Differential Equations*. arXiv: 2011.13456 [cs, stat]. Retrieved May 31, 2024, from <http://arxiv.org/abs/2011.13456>
- Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, & K. Müller (Eds.), *Advances in neural information processing systems* (Vol. 12).

- MIT Press. https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf
- Trabucco, B., Geng, X., Kumar, A., & Levine, S. (2022, February 17). *Design-Bench: Benchmarks for Data-Driven Offline Model-Based Optimization*. arXiv: 2202.08450 [cs]. <https://doi.org/10.48550/arXiv.2202.08450>
- Trippe, B. L., Yim, J., Tischer, D., Baker, D., Broderick, T., Barzilay, R., & Jaakkola, T. (2023, March 19). *Diffusion probabilistic modeling of protein backbones in 3D for the motif-scaffolding problem*. arXiv: 2206.04119 [cs, q-bio, stat]. Retrieved June 5, 2024, from <http://arxiv.org/abs/2206.04119>
- Wenliang, L. K., & Moran, B. (2023, November 16). *Score-based generative models learn manifold-like structures with constrained mixing*. arXiv: 2311.09952 [cs, stat]. <https://doi.org/10.48550/arXiv.2311.09952>
- Wu, L., Trippe, B. L., Naesseth, C. A., Blei, D. M., & Cunningham, J. P. (2023, June 30). *Practical and Asymptotically Exact Conditional Sampling in Diffusion Models*. arXiv: 2306.17775 [cs, q-bio, stat]. Retrieved June 5, 2024, from <http://arxiv.org/abs/2306.17775>
- Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., & Tang, J. (2022, March 6). *GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation*. arXiv: 2203.02923 [cs, q-bio]. Retrieved August 21, 2024, from <http://arxiv.org/abs/2203.02923>

Appendices

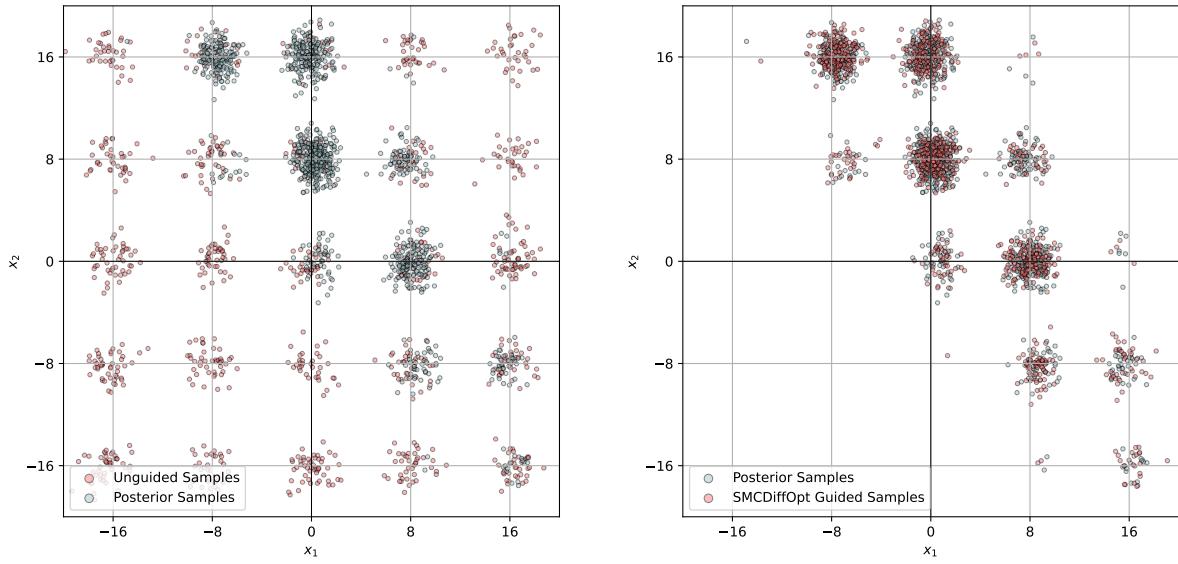
A. Additional Tables and Figures

| σ_y | d_x | d_y | SMCDiffOpt | DPS | IIIGD | TMPD |
|------------|-------|-------|-------------|-------------|-------------|-------------|
| 0.0 | 8 | 1 | 0.98 ± 0.44 | 8.22 ± 7.32 | 3.15 ± 2.58 | 3.5 ± 2.67 |
| | | 2 | 0.55 ± 0.43 | 0.41 ± 0.29 | 0.33 ± 0.27 | 0.44 ± 0.34 |
| | | 4 | 0.21 ± 0.07 | 0.12 ± 0.06 | 0.09 ± 0.03 | 0.08 ± 0.04 |
| | 80 | 1 | 0.75 ± 0.31 | 2.45 ± 1.79 | 3.18 ± 2.6 | 2.66 ± 1.47 |
| | | 2 | 1.22 ± 1.04 | 1.35 ± 1.21 | 0.33 ± 0.27 | 0.81 ± 0.68 |
| | | 4 | 0.26 ± 0.05 | 0.97 ± 0.86 | 0.08 ± 0.02 | 0.68 ± 0.44 |
| | 0.1 | 8 | 1.13 ± 0.51 | 8.18 ± 7.5 | 3.22 ± 2.67 | 3.11 ± 2.31 |
| | | 2 | 0.2 ± 0.07 | 0.38 ± 0.28 | 0.19 ± 0.13 | 0.43 ± 0.34 |
| | | 4 | 0.15 ± 0.05 | 0.16 ± 0.06 | 0.07 ± 0.01 | 0.06 ± 0.01 |
| | 80 | 1 | 1.25 ± 1.02 | 2.51 ± 2.11 | 2.93 ± 2.56 | 2.56 ± 1.18 |
| | | 2 | 0.45 ± 0.32 | 1.27 ± 1.14 | 0.62 ± 0.56 | 0.66 ± 0.54 |
| | | 4 | 0.23 ± 0.06 | 1.03 ± 0.9 | 0.08 ± 0.02 | 0.63 ± 0.34 |
| 1.0 | 8 | 1 | 1.35 ± 1.0 | 6.62 ± 4.96 | 1.16 ± 0.72 | 1.35 ± 0.9 |
| | | 2 | 0.44 ± 0.28 | 2.92 ± 2.42 | 0.94 ± 0.89 | 1.44 ± 1.15 |
| | | 4 | 0.15 ± 0.03 | 1.19 ± 0.66 | 0.1 ± 0.03 | 0.45 ± 0.26 |
| | 80 | 1 | 1.69 ± 1.37 | 5.26 ± 4.4 | 1.2 ± 0.79 | 1.23 ± 0.53 |
| | | 2 | 0.89 ± 0.78 | 3.21 ± 2.95 | 1.68 ± 1.62 | 1.41 ± 1.16 |
| | | 4 | 1.12 ± 0.92 | 1.54 ± 0.99 | 0.89 ± 0.74 | 1.37 ± 0.67 |

Table A.1.: GMM Sliced-Wasserstein distances for different σ_y .

| CbAS | CMA-ES | Gradient Ascent | MINs | REINFORCE | DDOM | DiffOpt | SMCDiffOpt |
|----------------|----------------|-----------------|----------------|----------------|-----------------|-----------------|-----------------|
| 80.082 ± 4.933 | 87.774 ± 3.965 | 94.414 ± 1.719 | 87.483 ± 0.481 | 89.351 ± 2.708 | 103.600 ± 8.139 | 113.545 ± 5.322 | 119.059 ± 4.497 |

Table A.2.: Raw scores for SuperConductor experiment (higher values are better).



(a) GMM prior samples under DDPM with analytic score; $d_x = 8, d_y = 2$.

(b) SMCDiffOpt samples for a particular measurement model; $d_x = 8, d_y = 2$.

Figure A.2.: GMM Prior versus posterior samples under guidance through SMCDiffOpt.

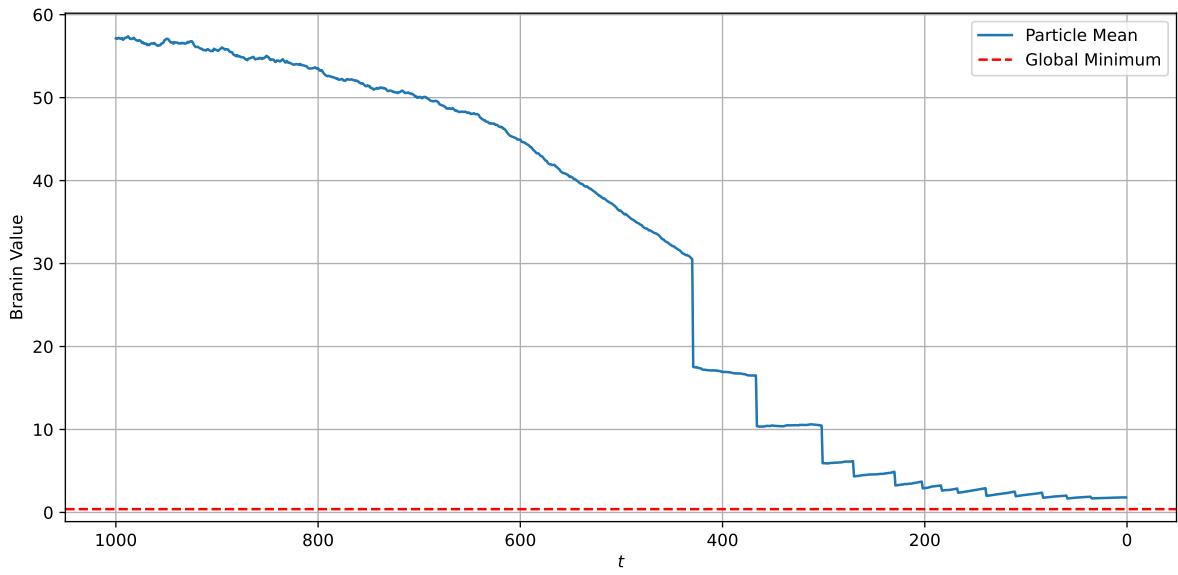


Figure A.2.: Mean value of Equation 23 on particles over time.

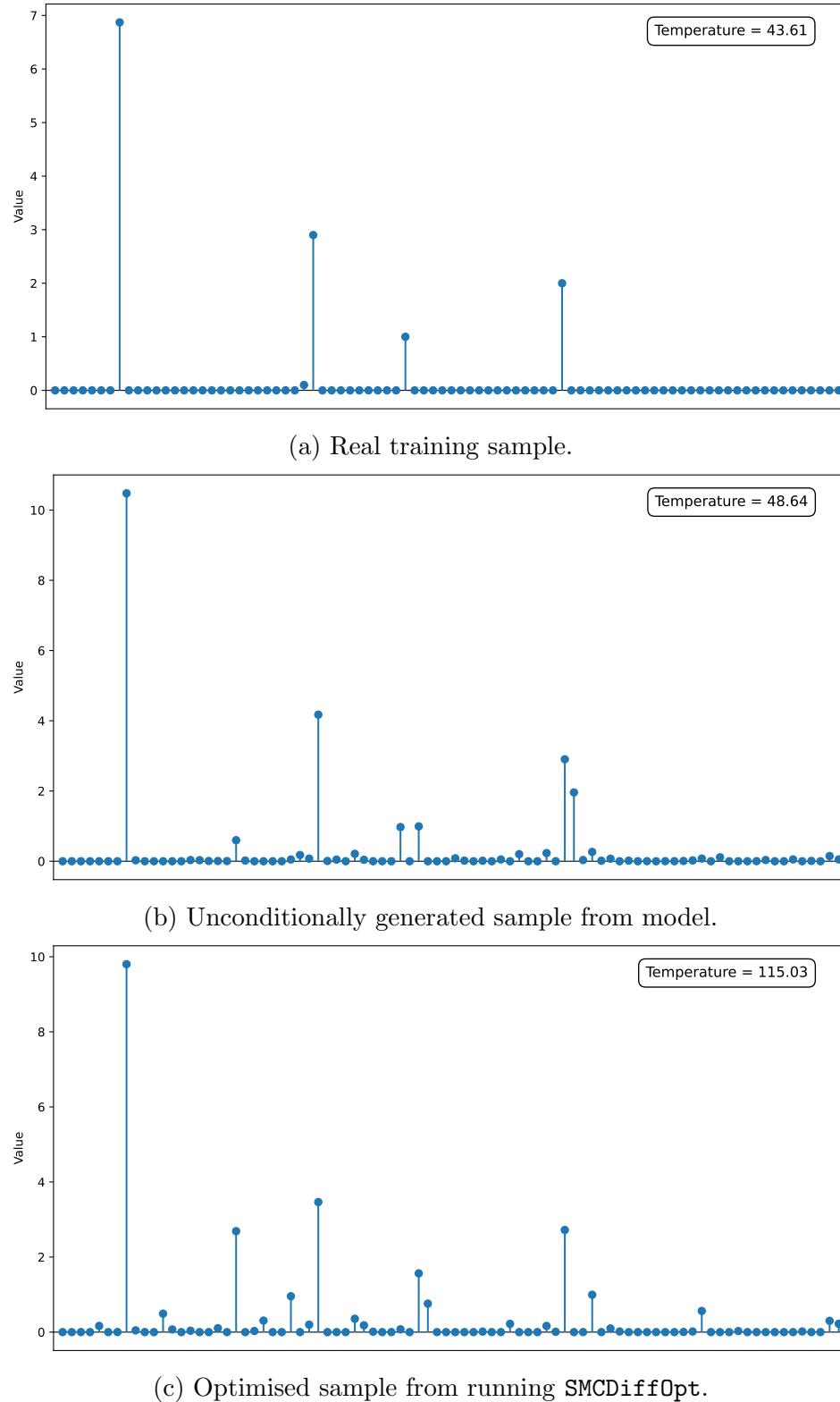


Figure A.2.: Real, synthetic, and optimised SuperConductor samples along with associated temperature values from evaluating configuration with oracle.

B. Proofs

Proof of Proposition 3.2.1. Given

$$\begin{aligned}\mathbf{X}_t \mid \mathbf{X}_0 = \mathbf{x}_0 &\sim \mathcal{N}(c_t \cdot \mathbf{x}_0, d_t^2 \cdot \mathbf{I}_{d_x}) \\ \mathbf{Y}_t \mid \mathbf{Y}_0 = \mathbf{y}_0 &\sim \delta(\mathbf{y}_t - c_t \cdot \mathbf{y}_0) \\ \mathbf{Y}_0 \mid \mathbf{X}_0 = \mathbf{x}_0 &\sim \mathcal{N}(A\mathbf{x}_t, \sigma_y^2 \mathbf{I}_{d_y})\end{aligned}$$

we have

$$\begin{aligned}\mathbb{E}\{\mathbf{Y}_t \mid \mathbf{X}_0\} &= \mathbb{E}\{\mathbb{E}\{\mathbf{Y}_t \mid \mathbf{Y}_0\} \mid \mathbf{X}_0\} \\ &= \mathbb{E}\{c_t \cdot \mathbf{Y}_0 \mid \mathbf{X}_0\} \\ &= c_t \cdot A\mathbf{X}_0\end{aligned}$$

and

$$\begin{aligned}\mathbb{V}\{\mathbf{Y}_t \mid \mathbf{X}_0\} &= \mathbb{E}\{\mathbb{V}\{\mathbf{Y}_t \mid \mathbf{Y}_0\} \mid \mathbf{X}_0\} + \mathbb{V}\{\mathbb{E}\{\mathbf{Y}_t \mid \mathbf{Y}_0\} \mid \mathbf{X}_0\} \\ &= \mathbb{V}\{c_t \cdot \mathbf{Y}_0 \mid \mathbf{X}_0\} \\ &= c_t^2 \sigma_y^2 \cdot \mathbf{I}_{d_y}.\end{aligned}$$

Hence,

$$\mathbf{Y}_t \mid \mathbf{X}_0 = \mathbf{x}_0 \sim \mathcal{N}(c_t \cdot A\mathbf{x}_0, \sigma_y^2 c_t^2 \cdot \mathbf{I}_{d_y}).$$

Further,

$$\begin{aligned}A\mathbf{X}_t \mid \mathbf{X}_0 = \mathbf{x}_0 &\sim \mathcal{N}(c_t \cdot A\mathbf{x}_0, d_t^2 \cdot AA^\top) \\ \implies \mathbf{Y}_t - A\mathbf{X}_t \mid \mathbf{X}_0 = \mathbf{x}_0 &\sim \mathcal{N}(0, \sigma_y^2 c_t^2 \cdot \mathbf{I}_{d_y} + d_t^2 \cdot AA^\top) \\ \implies \mathbf{Y}_t \mid \mathbf{X}_t = \mathbf{x}_t &\sim \mathcal{N}(A\mathbf{x}_t, \sigma_y^2 c_t^2 \cdot \mathbf{I}_{d_y} + d_t^2 \cdot AA^\top)\end{aligned}$$

□

Proposition B.1 (Gaussian-Gaussian Exactness for Linear Observations). Suppose

$$\mathbf{X}_t \mid \mathbf{X}_{t+1} \sim \mathcal{N}(\mu_{t+1}(\mathbf{x}_{t+1}), \Sigma_{t+1})$$

$$\mathbf{Y}_t \mid \mathbf{X}_t = \mathbf{x}_t \sim \mathcal{N}(A\mathbf{x}_t, \Xi_t)$$

Consider some proposal

$$p_t(\mathbf{x}_t \mid \mathbf{y}_t, \mathbf{x}_{t+1}) \propto p_t(\mathbf{x}_t \mid \mathbf{x}_{t+1}) g_t(\mathbf{y}_t \mid \mathbf{x}_t).$$

Then,

$$\mathbf{Y}_t \mid \mathbf{X}_{t+1} = \mathbf{x}_{t+1} \sim \mathcal{N}(A\mu_{t+1}, \Xi_t + A\Sigma_{t+1}A^\top).$$

Proof. By Gaussian conjugacy, it immediately follows that

$$\mathbf{X}_t \mid \mathbf{Y}_t = \mathbf{y}_t, \mathbf{X}_{t+1} = \mathbf{x}_{t+1} \sim \mathcal{N}(\mu_{t+1}^P, \Sigma_{t+1}^P)$$

where

$$\begin{aligned}\mu_{t+1}^P &= \Sigma_{t+1}^P (\Sigma_{t+1}^{-1} \mu_{t+1}(\mathbf{x}_{t+1}) + A^\top \Xi_t^{-1} \mathbf{y}_t) \\ \Sigma_{t+1}^P &= (\Sigma_{t+1}^{-1} + A^\top \Xi_t^{-1} A)^{-1}.\end{aligned}$$

The normalizing constant of the proposal is

$$\int p_t(\mathbf{x}_t \mid \mathbf{x}_{t+1}) g_t(\mathbf{y}_t \mid \mathbf{x}_t) d\mathbf{x}_t = g_t(\mathbf{y}_t \mid \mathbf{x}_{t+1})$$

Note that

$$\mathbb{E} \{\mathbf{Y}_t \mid \mathbf{X}_{t+1}\} = \mathbb{E} \{\mathbb{E} \{\mathbf{Y}_t \mid \mathbf{X}_t\} \mid \mathbf{X}_{t+1}\} \quad \dots \text{tower property}$$

$$= \mathbb{E} \{A\mathbf{X}_t \mid \mathbf{X}_{t+1}\}$$

$$= A\mathbb{E} \{\mathbf{X}_t \mid \mathbf{X}_{t+1}\}$$

$$= A\mu_{t+1}(\mathbf{X}_{t+1})$$

$$\mathbb{V} \{\mathbf{Y}_t \mid \mathbf{X}_{t+1}\} = \mathbb{E} \{\mathbb{V} \{\mathbf{Y}_t \mid \mathbf{X}_t\} \mid \mathbf{X}_{t+1}\} + \mathbb{V} \{\mathbb{E} \{\mathbf{Y}_t \mid \mathbf{X}_t\} \mid \mathbf{X}_{t+1}\} \quad \dots \text{total variance}$$

$$= \mathbb{E} \{\Xi_t \mid \mathbf{X}_{t+1}\} + \mathbb{V} \{A\mathbf{X}_t \mid \mathbf{X}_{t+1}\}$$

$$= \Xi_t + A\mathbb{V} \{\mathbf{X}_t \mid \mathbf{X}_{t+1}\} A^\top$$

$$= \Xi_t + A\Sigma_{t+1}A^\top$$

Since \mathbf{Y}_t is an affine transformation, it follows then that

$$\mathbf{Y}_t \mid \mathbf{X}_{t+1} = \mathbf{x}_{t+1} \sim \mathcal{N} (A\mu_{t+1}, \Xi_t + A\Sigma_{t+1}A^\top).$$

□

C. Further Experimental Details

Gaussian Mixture Model We aimed to include MCGdiff (Cardoso et al., 2023) in our comparison. However, we were unable to reproduce their results, even using the code on their repository. Indeed, we found that the resulting sliced Wasserstein distances from MCGdiff were almost uniformly larger than those of SMCDiffOpt on each experimental run (with identical setups and pseudo-RNG seeds). This is doubly unexpected since the raw results don’t align with those found in their paper, and since theoretically their proposal should be more optimal than ours for this particular task; if we take the values they give in Table 1 of their paper and compare them with those in [Table 4.1](#) we see this is indeed the case. Given the time constraints, the exhaustion of all obvious potential author-errors, and assuming the validity of their results, we opted to omit our results from a comparison. However, we provide in [Table C.3](#) the results including MCGdiff but note with caution that we lack confidence in such figures.

Branin Optimisation We train a noise estimating neural network and then employ REF to convert it to a score approximating network, $s_\theta(\mathbf{x}_t, t)$. The model is a 4 layer MLP with 256 hidden dimensions in each layer and ReLU activation function. We also add some simple Fourier based time embeddings to encode the current time-step. The model is trained on 6000 datapoints — uniformly sampled from the elliptical region of interest — in 20,000 epochs. We use an Adam optimiser for training model parameters with a base learning rate of 10^{-3} . The diffusion is parametrised with $\beta_{\min} = 10^{-4}$, $\beta_{\max} = 0.02$, $T = 1000$, with linear schedule and applying DDPM sampling.

SuperConductor Optimisation Our model architecture is essentially identical to that used in the Branin optimisation. For training, we used a 90:10 train:evaluation split of the 17,014 subset of vectors, using a 256 batch size in 40,000 epochs, monitoring evaluation loss for early stopping. We use an AdaBelief optimiser for training model parameters with a base learning rate of 3×10^{-4} . We similarly parametrise the diffusion as in the Branin experiment.

| σ_y | d_x | d_y | SMCDiffOpt | MCGDiff | DPS | PIIGD | TMPD |
|------------|-------|-------|-------------|-------------|-------------|-------------|-------------|
| 0.0 | 8 | 1 | 0.98 ± 0.44 | 0.95 ± 0.44 | 8.22 ± 7.32 | 3.15 ± 2.58 | 3.5 ± 2.67 |
| | | 2 | 0.55 ± 0.43 | 0.39 ± 0.33 | 0.41 ± 0.29 | 0.33 ± 0.27 | 0.44 ± 0.34 |
| | | 4 | 0.21 ± 0.07 | 0.15 ± 0.09 | 0.12 ± 0.06 | 0.09 ± 0.03 | 0.08 ± 0.04 |
| | 80 | 1 | 0.75 ± 0.31 | 1.06 ± 0.77 | 2.45 ± 1.79 | 3.18 ± 2.6 | 2.66 ± 1.47 |
| | | 2 | 1.22 ± 1.04 | 2.31 ± 2.21 | 1.35 ± 1.21 | 0.33 ± 0.27 | 0.81 ± 0.68 |
| | | 4 | 0.26 ± 0.05 | 2.22 ± 1.92 | 0.97 ± 0.86 | 0.08 ± 0.02 | 0.68 ± 0.44 |
| | 0.1 | 1 | 1.13 ± 0.51 | 0.79 ± 0.62 | 8.18 ± 7.5 | 3.22 ± 2.67 | 3.11 ± 2.31 |
| | | 2 | 0.2 ± 0.07 | 1.15 ± 1.09 | 0.38 ± 0.28 | 0.19 ± 0.13 | 0.43 ± 0.34 |
| | | 4 | 0.15 ± 0.05 | 0.25 ± 0.12 | 0.16 ± 0.06 | 0.07 ± 0.01 | 0.06 ± 0.01 |
| | 80 | 1 | 1.25 ± 1.02 | 1.26 ± 0.87 | 2.51 ± 2.11 | 2.93 ± 2.56 | 2.56 ± 1.18 |
| | | 2 | 0.45 ± 0.32 | 3.06 ± 2.93 | 1.27 ± 1.14 | 0.62 ± 0.56 | 0.66 ± 0.54 |
| | | 4 | 0.23 ± 0.06 | 3.34 ± 3.01 | 1.03 ± 0.9 | 0.08 ± 0.02 | 0.63 ± 0.34 |
| 1.0 | 8 | 1 | 1.35 ± 1.0 | 1.14 ± 0.7 | 6.62 ± 4.96 | 1.16 ± 0.72 | 1.35 ± 0.9 |
| | | 2 | 0.44 ± 0.28 | 1.11 ± 1.04 | 2.92 ± 2.42 | 0.94 ± 0.89 | 1.44 ± 1.15 |
| | | 4 | 0.15 ± 0.03 | 0.12 ± 0.06 | 1.19 ± 0.66 | 0.1 ± 0.03 | 0.45 ± 0.26 |
| | 80 | 1 | 1.69 ± 1.37 | 1.49 ± 0.75 | 5.26 ± 4.4 | 1.2 ± 0.79 | 1.23 ± 0.53 |
| | | 2 | 0.89 ± 0.78 | 2.08 ± 1.97 | 3.21 ± 2.95 | 1.68 ± 1.62 | 1.41 ± 1.16 |
| | | 4 | 1.12 ± 0.92 | 4.52 ± 2.55 | 1.54 ± 0.99 | 0.89 ± 0.74 | 1.37 ± 0.67 |

Table C.3.: Sliced-Wasserstein distances for GMM experiment with MCGdiff included. These results do not align with those in Cardoso et al. (2023), despite running exactly the script in their repository.

D. Additional Background and Related Works

D.1. Diffusion Models

Remark D.1 (SDE Representation). The DDPM approach corresponds to a discretised time rescaled Ornstein-Uhlenbeck process (Boys et al., 2023; Y. Song et al., 2021):

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)}d\mathbf{w}_t$$

and is often referred to as the variance-preserving SDE (Y. Song et al., 2021). When it comes to sampling, we can then analytically reverse the diffusion model using the result of Anderson (1982) and use any numeric sampler. The choice of sampling strategy corresponds to slightly different discretised equivalents and is discussed in detail in Y.

Song et al. (2021) in detail. Regardless, this SDE representation is what analytically justifies the backwards process (see footnote 1). \diamond

Remark D.2 (DDIM Representation). Under DDIM, take

$$u_t = \sqrt{\alpha_{t-1}} \quad v_t = -\sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \sqrt{1 - \bar{\alpha}_t} \quad w_t = \sigma_t$$

with $\{\sigma_t\}_{t=1}^T$ an arbitrary conditional variance sequence. It's common to consider

$$\sigma_t(\eta) = \eta \sqrt{\frac{1 - \alpha_{t-1}}{1 - \alpha_t}} \sqrt{1 - \frac{\alpha_t}{\alpha_{t-1}}}, \quad \eta \in [0, 1]$$

with $\eta = 1$ ultimately reducing u_t, v_t, w_t to the DDPM values, and $\eta = 0$ corresponding to deterministic generation (J. Song et al., 2020). \diamond

Remark D.3 (Time Reslicing). One of the remarkable features of the DDIM algorithm is it enabling *time re-spacing* whereby we can sample from the backwards process in fewer time-steps. In this paper, we don't consider such re-spacing though in principle our methodology does not prohibit it. \diamond

Proposition D.1 (Score to Noise Conversion). Let $\mathbf{x}_t = c_t \mathbf{x}_0 + d_t \epsilon_t$ be a forward noised observation. Then

$$\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \approx -\frac{\epsilon_t}{d_t}.$$

Proof. Follows from forward marginal and Tweedie's formula (Equation 5). \square

D.2. Non-Linear Case

In the case of non-linear measurement operator, m , we must resort to approximate methods. Assuming $\mathbf{X}_0 \mid \mathbf{X}_t$ is Gaussian (this is the *Moment Projection* assumption

(Boys et al., 2023)), we can easily show:

$$\mathbf{Y}_t \mid \mathbf{X}_t = \mathbf{x}_t \sim \mathcal{N} \left(c_t \cdot \mathbb{E} \{ m(\mathbf{X}_0) \mid \mathbf{X}_t = \mathbf{x}_t \}, c_t^2 (\sigma_y^2 \cdot \mathbf{I}_{d_y} + \mathbb{V} \{ m(\mathbf{X}_0) \mid \mathbf{X}_t = \mathbf{x}_t \}) \right)$$

and then use a first-order Taylor expansion (linearisation) so that

$$m(\mathbf{X}_0) \approx m(\mu_{\mathbf{X}_0|\mathbf{x}_t}) + \nabla m(\mu_{\mathbf{X}_0|\mathbf{x}_t}) \cdot (\mathbf{X}_0 - \mu_{\mathbf{X}_0|\mathbf{x}_t}) \quad (24)$$

to approximate the likelihood by:

$$\mathbf{Y}_t \mid \mathbf{X}_t = \mathbf{x}_t \sim \mathcal{N} \left(c_t \cdot m(\mu_{\mathbf{X}_0|\mathbf{x}_t}), c_t^2 (\sigma_y^2 \cdot \mathbf{I}_{d_y} + \nabla m(\mu_{\mathbf{X}_0|\mathbf{x}_t}) \Sigma_{\mathbf{X}_0|\mathbf{x}_t} \nabla m(\mu_{\mathbf{X}_0|\mathbf{x}_t})^\top) \right)$$

where $\mu_{\mathbf{X}_0} \approx \hat{\mathbf{x}}_0(\mathbf{x}_t)$, and we approximate $\Sigma_{\mathbf{X}_0|\mathbf{x}_t}$ by $r_t^2 \mathbf{I}_{d_x}$ with r_t^2 fixed based on the diffusion parameters (Ho et al., 2020) or learnt (Nichol & Dhariwal, 2021), or better approximated with a second-order Tweedie formula (Boys et al., 2023).

Alternatively, we can make the approximate assumption $\mathbf{X}_t \approx c_t \mathbf{X}_0$, so that

$$\mathbf{Y}_t \mid \mathbf{X}_t = \mathbf{x}_t \sim \mathcal{N} \left(c_t \cdot m \left(\frac{\mathbf{x}_t}{c_t} \right), c_t^2 \sigma_y^2 \mathbf{I}_{d_y} \right).$$

Referring to 20, this essentially ignores the second term in the variance. More concretely, it assumes we deterministically ‘noise’ the data exactly as in 19. This approximation can work well in cases where g is not significantly non-linear, such as in Section 4.1.

D.3. FPS-SMC Observation Noising

Rather than Equation 19, Dou and Song (2023) instead considers a “shared-noise” or “duplex” diffusion.

Proposition D.2 (Obvbservation Diffusion). Dou and Song (2023, Proposition B.1). Let \mathbf{x}_t be generated according to the forward marginal of some diffusion process. Let

$\mathbf{y} = \mathbf{y}_0$ be some (linear) measurement we have about the clean sample, \mathbf{x}_0 . Construct a sequence $\{\mathbf{y}_t\}_{t=1}^T$ by

$$\mathbf{y}_t = a_t \cdot \mathbf{y}_{t-1} + b_t \cdot A\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \mathbf{I}_{d_y})$$

where ϵ_t is the *same* as used for forward noising \mathbf{x}_0 at time-step t . It follows that

$$\mathbf{Y}_t \mid \mathbf{X}_t = \mathbf{x}_t \sim \mathcal{N}(A\mathbf{x}_t, \sigma_y^2 c_t^2 \cdot \mathbf{I}_{d_y}) \quad (25)$$

with $g(\mathbf{y}_0 \mid \mathbf{x}_0) = g(\mathbf{y} \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{y}; A\mathbf{x}_0, \sigma_y^2 \cdot \mathbf{I}_{d_y})$ exactly the measurement density.

Note because of the noise-sharing, we get cancellation in the covariance of the likelihood, dropping the $d_t^2 \cdot AA^\top$ term as in [Equation 20](#).

Based on [Proposition D.2](#) and [Dou and Song \(2023, Remark B.1\)](#), we can generate some $\{\mathbf{y}_t\}_{t=0}^T$ either forwards or *backwards*. The latter follows because where in the backwards process for \mathbf{x}_t we use an estimate of \mathbf{x}_0 (via Tweedie's formula), for the observations we have \mathbf{y}_0 , and an expression can be analytically derived.