GENERALISED PARALLEL TEMPERING: FLEXIBLE REPLICA EXCHANGE VIA FLOWS AND DIFFUSIONS

Leo Zhang[*†] **Peter Potaptchik**[*†] **Arnaud Doucet**[†] **Hai-Dang Dau**[‡] **Saifuddin Syed**[†]
[†]University of Oxford, [‡]National University of Singapore
[*]Equal contribution.

ABSTRACT

Parallel Tempering (PT) is a classical MCMC algorithm designed for leveraging parallel computation to sample efficiently from high-dimensional, multimodal or otherwise complex distributions via annealing. One limitation of the standard formulation of PT is the growth of computational resources required to generate high-quality samples, as measured by effective sample size or round trip rate, for increasingly challenging distributions. To address this issue, we propose the framework: Generalised Parallel Tempering (GePT) which allows for the incorporation of recent advances in modern generative modelling, such as normalising flows and diffusion models, within Parallel Tempering, while maintaining the same theoretical guarantees as MCMC-based methods. For instance, we show that this allows us to utilise diffusion models in a parallelised manner, bypassing the usual computational cost of a large number of steps to generate quality samples. Further, we empirically demonstrate that GePT can improve sample quality and reduce the growth of computational resources required to handle complex distributions over the classical algorithm.

1 INTRODUCTION

Sampling from a complex probability distribution π over a state-space $\mathcal{X}$, whose density $\pi(x)$ is only known up to a normalising constant, is a fundamental task in modern statistical inference. Markov Chain Monte Carlo (MCMC) methods are a popular class of algorithms employed for such purposes that rely on constructing a Markov chain $(X_t)_{t\in\mathbb{N}}$ over $\mathcal{X}$ using a sequence of local moves which leave the target distribution invariant. While MCMC algorithms are guaranteed to converge asymptotically, standard approaches, in practice, struggle when the target distribution is complex with multiple well-separated modes. This is due to the low likelihood of the Markov chain crossing low-density regions (Łatuszyński et al., 2025). Unfortunately, multimodality is ubiquitous in challenging real-world modelling problems (Papamarkou et al., 2022; Hénin et al., 2022), limiting the usefulness of standard MCMC methods. To handle such cases, *Parallel Tempering* (PT) (Geyer, 1991) (also known as the Replica-Exchange Monte Carlo (Swendsen & Wang, 1986; Hukushima & Nemoto, 1996)) is a popular class of MCMC algorithms designed to improve the global mixing of locally efficient MCMC algorithms.

PT works by considering an *annealing path* $\pi_0, \pi_1, \dots, \pi_N$ of distributions over $\mathcal{X}$ interpolating between a simple reference distribution π_0 (e.g. a Gaussian), which we can efficiently sample and evaluate the normalised density, and the target distribution $\pi_N = \pi$. PT algorithms involve constructing a Markov chain $(\mathbf{X}_t)_{t\in\mathbb{N}}$ on the extended state-space $\mathcal{X}^{N+1}$, targeting the joint annealing distribution $\boldsymbol{\pi}(\mathbf{x}) = \pi_0(x^0)\cdots\pi_N(x^N)$. The Markov dynamics of the PT chain $\mathbf{X}_t = (X_t^0, \dots, X_t^N)$ alternate between (1) a *local exploration phase* where the n-th chain [1] of $\mathbf{X}_t$ is updated according to a π_n-invariant MCMC algorithm; and (2) a *communication phase* which proposes a scheduled sequence of swaps between neighbouring states accepted according to a Metropolis-Hastings correction ensuring invariance. Crucially, PT is designed to offset the additional computation burden of

[1]Following the PT literature, we also refer to components of $\mathbf{X}_t$ as chains.

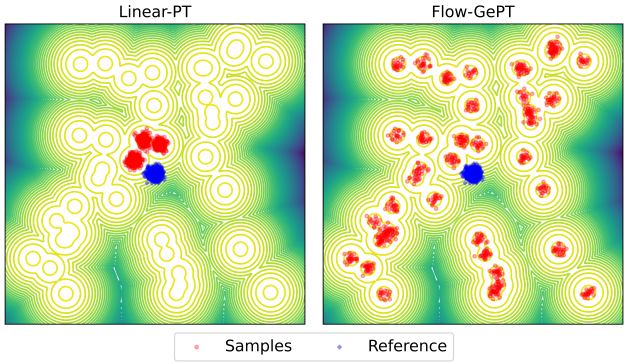

Figure 1: One thousand samples from Linear-PT (left) and Flow-GePT (right) for GMM-2 using six chains. Linear-PT struggles to capture all modes, primarily sampling near the reference, while Flow-GePT successfully generates samples from all modes, demonstrating improved exploration. See Section 4.1 for details.

simulating the extended N chains through the use of *parallel computation*, allowing for a similar effective computational cost as a single chain when implemented in a maximally parallelised manner.

Typically, the chains X_t^n mix faster when closer to the reference and struggle closer towards the target. Therefore, communication between the reference and target, facilitated through swaps transporting samples between neighbouring chains, can induce rapid mixing between modes of the target component (Woodard et al., 2009; Surjanovic et al., 2024). The importance of the swapping mechanism for PT has led to a literature dedicated to optimising communication between the reference and target by tuning the swapping schedule (Syed et al., 2022), annealing path (Syed et al., 2021), and reference distribution (Surjanovic et al., 2022). However, such works still rely on the same naive, inflexible replica-exchange mechanism originally formalised in Geyer (1991).

As an alternative to PT, recent work (Midgley et al., 2023; Vargas et al., 2023) has explored *neural samplers* which use more flexible approximate methods, such as normalising flows and diffusion models (Papamakarios et al., 2021; Song et al., 2020), for sampling. However, neural-sampling methods usually incur a bias, foregoing the asymptotic consistency and theoretical guarantees of MCMC, and can be expensive to implement and train. Despite its inflexibility, it has been empirically shown in He et al. (2025) that PT provides a strong baseline for such methods.

In this paper, in order to combine the benefits of neural samplers with the robustness of PT, we mathematically formalise the framework introduced by Ballard & Jarzynski (2009; 2012) in the statistical physics literature for designing more flexible replica-exchange mechanisms; we term the resulting algorithm, *Generalised Parallel Tempering* (GePT) (see Algorithm 1 and Figure 1). GePT preserves the same mixing and asymptotic consistency properties of classical PT and can be easily incorporated into existing PT implementations. We then provide two concrete instances of GePT, *Flow-GePT* and *Diff-GePT*, which respectively utilise methods from normalising flows and diffusion models to construct flexible swap moves. This strengthens PT with the ability of flows and diffusion models to effectively learn transport maps between neighbouring distributions. Further, we demonstrate how these algorithms can use flows and diffusions in a *parallelised* manner. This circumvents the usual computational burden of requiring a large number of steps to generate samples while retaining the asymptotic guarantees of MCMC methods. Finally, we empirically demonstrate that Flow-GePT and Diff-GePT outperform PT by accelerating the communication between the reference and target states, even when controlling for sequential computation.

Related work Previous work on neural samplers can be roughly organised as being diffusion/flow-based (Vargas et al., 2023; Zhang & Chen, 2022; Zhang et al., 2023; Berner et al., 2024; Akhound-Sadegh et al., 2024; Nüsken et al., 2024; Tian et al., 2024; Albergo & Vanden-Eijnden, 2024), normalising flows-based (Albergo et al., 2019; Noé et al., 2019; Gabrié et al., 2022; Midgley et al., 2023) and Sequential Monte Carlo (SMC)-based (Arbel et al., 2021; Doucet et al., 2022; Matthews et al., 2022; Phillips et al., 2024). We note that these classifications are not strict - for example, Phillips et al. (2024) uses techniques from both diffusion models and SMC. A relevant prior work

integrating PT with normalising flows is Invernizzi et al. (2022). Their approach involves training a normalising flow to directly map configurations from the highest-temperature reference distribution of a molecular system to the lowest-temperature target distribution, effectively bypassing the intermediate annealing distributions. By contrast, our framework leverages normalising flows to facilitate exchanges between all neighbouring temperature levels, thereby enhancing sampling efficiency across the entire annealing path and providing a more stable training objective.

2 BACKGROUND

2.1 PARALLEL TEMPERING

Let $(\mathcal{X}, \mathcal{F})$ be a measurable space. Let $\pi_0, \pi_1, \dots, \pi_N$ be an annealing path which admits densities $\pi_n(\mathrm{d}x) = \tilde{\pi}_n(x)\mathrm{d}x/Z_n$ with respect to a base measure $\mathrm{d}x$. We can evaluate the un-normalised density $\tilde{\pi}_n(x) > 0$ but typicially do not know the normalising constant $Z_n = \int_{\mathcal{X}} \tilde{\pi}_n(x)\mathrm{d}x$. There is considerable flexibility in choosing the annealing distributions (Syed et al., 2021; Phillips et al., 2024). However, a common choice is the *linear path* $\pi_n(x) \propto \pi_0(x)^{1-s_n}\pi_N(x)^{s_n}$, which linearly interpolates between the reference and target densities in log-space, according to an annealing schedule $0 = s_0 < s_1 \cdots < s_N = 1$.

Given an annealing path, the PT algorithm constructs a Markov chain $\mathbf{X}_t = (X_t^0, \dots, X_t^N)$ on the extended state-space $\mathcal{X}^{N+1}$ invariant to the joint distribution $\boldsymbol{\pi}(\mathbf{x}) = \boldsymbol{\pi}(\mathrm{d}x^0, \dots, \mathrm{d}x^N) = \prod_{n=0}^N \pi_i(\mathrm{d}x^i)$. We construct $\mathbf{X}_t$ from $\mathbf{X}_{t-1}$ through a *local exploration* move followed by a *communication* move, encoded by the local exploration and communication Markov kernels $\mathbf{K}^{\mathrm{expl}}$, and $\mathbf{K}_t^{\mathrm{comm}}$ respectively on $\mathcal{X}^{N+1}$- i.e. $\mathbf{X}_t \sim \mathbf{K}_t^{\mathrm{comm}}\mathbf{K}^{\mathrm{expl}}(\mathbf{X}_{t-1}, \mathrm{d}\mathbf{x}_t)$. We informally describe how these kernels act on $\mathcal{X}^{N+1}$ below and formalise their construction in Appendix A.

Local exploration Given a state $\mathbf{x} = (x^0, \dots, x^N)$ on the extended state-space $\mathcal{X}^{N+1}$, the local exploration kernel $\mathbf{K}^{\mathrm{expl}}(\mathbf{x}, \mathrm{d}\mathbf{x}') = \prod_{n=0}^N K_n(x^n, \mathrm{d}x'^n)$ defined on $\mathcal{X}^{N+1}$ updates x^n using a π_n-invariant kernel $K_n(x^n, \mathrm{d}x'^n)$ on $\mathcal{X}$. In general, we will assume that $K_0(x, \mathrm{d}x') = \pi_0(\mathrm{d}x')$ generates an independent sample from the reference, and $K_n(x, \mathrm{d}x')$ corresponds to a MCMC move targeting π_n for $n > 0$. Notably, each coordinate can be updated in parallel.

Communication For $n = 1, \dots, N$, we define the PT swap kernel $\mathbf{K}_{n-1,n}(\mathbf{x}, \mathrm{d}\mathbf{x}')$ as swapping the $n-1$-th and n-th coordinates of $\mathbf{x} \in \mathcal{X}^{N+1}$ with probability $\alpha_n(x^{n-1}; x^n)$ equal to,

$$\alpha_{n-1,n}(x^{n-1}; x^n) = 1 \wedge \frac{[\mathrm{d}\pi_n/\mathrm{d}\pi_{n-1}](x^{n-1})}{[\mathrm{d}\pi_n/\mathrm{d}\pi_{n-1}](x^n)} \tag{1}$$

$$= 1 \wedge \frac{\tilde{\pi}_{n-1}(x^n)\tilde{\pi}_n(x^{n-1})}{\tilde{\pi}_{n-1}(x^{n-1})\tilde{\pi}_n(x^n)}, \tag{2}$$

where $\mathrm{d}\pi_n/\mathrm{d}\pi_{n-1}$ is the Radon-Nikodym derivative of π_n with respect to π_{n-1}.

The swap kernel $\mathbf{K}_{n-1,n}$ leaves $\boldsymbol{\pi}$ invariant. Therefore, we can obtain a valid communication move by composing a sequence of swap kernels $\mathbf{K}_{n-1,n}$. In practice, it is advantageous to use the scheme from *non-reversible* PT (NRPT) (Okabe et al., 2001; Syed et al., 2022), which performs as $\mathbf{K}_t^{\mathrm{comm}} = \prod_{n \text{ even}} \mathbf{K}_{n-1,n}$ when t is even and $\mathbf{K}_t^{\mathrm{comm}} = \prod_{n \text{ odd}} \mathbf{K}_{n-1,n}$ when t is odd. Finally, we note that the swap moves outlined above can be applied in parallel, allowing for distributed implementations of PT to leverage parallel computation to accelerate sampling (Surjanovic et al., 2023).

Performance metrics for PT While the effective sample size (ESS) of samples generated by a Markov chain is the gold standard for evaluating the performance of MCMC algorithms, in our setting, we are mainly interested in improving the swap kernel within PT. As ESS measures the intertwined performance of the local exploration and swap kernels, we are instead interested in measuring performance in terms of the *round trip rate* (Katzgraber et al., 2006; Lingenheil et al., 2009). This metric is used in the PT literature to track the rate at which independent samples from the reference are transported to the target (Surjanovic et al., 2024). Notably a higher round trip rate is obtained if fewer proposed swaps are rejected. For further details, see Appendix C.

Normalising flows Normalising flows (Tabak & Vanden-Eijnden, 2010; Rezende & Mohamed, 2015; Dinh et al., 2017) provide a flexible framework for approximating complex probability distributions by transforming a simple base distribution (e.g., a Gaussian) through a sequence of invertible, differentiable mappings. Formally, a normalising flow models a target distribution π using a bijective function $F_\theta : \mathcal{X} \to \mathcal{X}$, typically a neural network parameterized by θ, where samples are generated as $x = F_\theta(z)$, where $z \sim p_z$, a simple prior distribution. The density of the transformed distribution is given by the change of variables formula:

$$p(x) = p_z(F_\theta^{-1}(x)) \left|\det J_{F_\theta}(F_\theta^{-1}(x))\right|^{-1},$$

where J_{F_θ} is the Jacobian of F_θ. If samples from the target distribution are available, normalising flows can be trained via maximum likelihood estimation (MLE). Alternatively, when only the unnormalised density is known, training can be done by minimizing the reverse Kullback-Leibler (KL) divergence: $\mathrm{KL}(p\|\pi) = \int p(x) \log \frac{p(x)}{\pi(x)}\, dx$.

Diffusion models Given a target distribution π, the Variance-Preserving (VP) diffusion process is defined by the following stochastic differential equation (SDE) $dY_s = -\beta_s Y_s \mathrm{d}s + \sqrt{2\beta_s}\mathrm{d}W_s$, where $Y_0 \sim \pi$ and $\beta_s : [0,1] \to \mathbb{R}^+$. The dynamics defined the SDE induces a path of distributions $(\pi_s^{\mathrm{VP}})_{s\in[0,1]}$ such that $Y_{1-s} \sim \pi_s^{\mathrm{VP}}$ and π_0^{VP} is close to a standard Gaussian[2]. Diffusion models aim to learn an approximation to the score $\nabla \log \pi_s^{\mathrm{VP}}$ through the score-matching objective (Vincent, 2011; Song et al., 2020) to exploit the fact that the time-reversal SDE $(X_s)_{s\in[0,1]} = (Y_{1-s})_{s\in[0,1]}$ has the form $dX_s = [\beta_{1-s}X_s + 2\beta_{1-s}\nabla \log \pi_s^{\mathrm{VP}}(X_s)]\mathrm{d}s + \sqrt{2\beta_{1-s}}\mathrm{d}B_s$ with $X_0 \sim \pi_0^{\mathrm{VP}}$ in order to generate approximate samples from π by solving the reverse SDE with the learned score.

3 Generalised Parallel Tempering

A limitation of PT is the inflexibility of the swap kernels, which can only propose samples that can be directly exchanged between distributions, since the classic PT swap in Equation (1) only considers the relative densities of π_{n-1} and π_n. It cannot exploit any knowledge of flows that transport one measure to another. For instance, if we have $X^{n-1} \sim \pi_{n-1}$, $X^n \sim \pi_n$ and have an invertible mapping $F_n : \mathcal{X} \to \mathcal{X}$ such that $F_n(X^{n-1})$ is π_n-distributed (and thus $F_n^{-1}(X^n)$ is π_{n-1}-distributed), we would like to be able to propose 'generalised swaps' of the form $(x^{n-1}, x^n) \to (F_n^{-1}(x^n), F_n(x^{n-1}))$ with 100% acceptance rate.

To address this limitation, we propose Generalised Parallel Tempering (GePT) which only differs from PT through the use of *generalised swap* kernels $\bar{\mathbf{K}}_{n-1,n}$. Further, we provide specific examples of our method which incorporate techniques from the generative modelling literature to achieve high acceptance rates and accelerate sampling.

3.1 Generalised PT Swap

In addition to the annealing path, suppose that we have two user-chosen Markov kernels $P_n(x^{n-1}, \mathrm{d}x_*^n)$ and $Q_{n-1}(x^n, \mathrm{d}x_*^{n-1})$ on $\mathcal{X}$ for $n = 1, \ldots, N$, which we will refer to as the *forward* and *backward* kernels respectively. For $n \in \{1, \ldots, N\}$, a *generalised swap* kernel $\bar{\mathbf{K}}_{n-1,n}(\mathbf{x}, d\mathbf{x}')$ starts by generating a proposal x_*^n given x^{n-1} using the forward kernel P_n to generate a forward proposal x_*^{n-1} given x^n, and uses the backward kernel Q_{n-1} to generate a backward proposal x_*^{n-1}. We then replace x^{n-1}, and x^n in $\mathbf{x}$ with the proposed states x_*^{n-1} and x_*^n respectively with probability $\bar{\alpha}_n(x^{n-1}, x^n; x_*^{n-1}, x_*^n)$ equal to

$$\bar{\alpha}_n(x^{n-1}, x^n; x_*^{n-1}, x_*^n) = 1 \wedge \frac{[\mathrm{d}\bar{\pi}_n^Q/\mathrm{d}\bar{\pi}_{n-1}^P](x^{n-1}, x_*^n)}{[\mathrm{d}\bar{\pi}_n^Q/\mathrm{d}\bar{\pi}_{n-1}^P](x_*^{n-1}, x^n)}, \tag{3}$$

where $\bar{\pi}_{n-1}^P = \pi_{n-1} \otimes P_n$ and $\bar{\pi}_n^Q = \pi_n \otimes Q_{n-1}$ are the forward and backward extensions of π_{n-1} and π_n respectively defining probability measures on $\mathcal{X} \times \mathcal{X}$ equal to,

See Appendix A.3 for a formal description of the GePT swap kernel $\bar{\mathbf{K}}_{n-1,n}$. By using $\bar{\mathbf{K}}_{n-1,n}$ in place of PT swap kernel kernel $\mathbf{K}_{n-1,n}$ we obtain the GePT algorithm described in Algorithm 1.

[2]We reverse the time convention of diffusion models to align with the notation from the PT literature

Algorithm 1 Generalised Parallel Tempering (GePT)

Initialise $\mathbf{X}_0 = (X_0^0, \dots, X_0^N)$;
for $t = 1, \dots, T$ **do**
  $\mathbf{X}_t = (X_t^0, \dots, X_t^N), \quad X_t^n \sim K_n(X_{t-1}^n, \mathrm{d}x^n)$ ▷ Local exploration (see Section 2.1)
  if t is even, $\mathcal{S} = \{n \text{ even}\}$ **else** $\mathcal{S} = \{n \text{ odd}\}$ ▷ Communication (see Section 2.1)
  for $n \in \mathcal{S}$ **do**
    $X_*^{n-1} \sim Q_{n-1}(X_t^n, \mathrm{d}x_*^{n-1})$ ▷ Backward proposal
    $X_*^n \sim P_n(X_t^{n-1}, \mathrm{d}x_*^n)$ ▷ Forward proposal
    Simulate $U \sim \mathrm{Uniform}([0,1])$
    if $U < \bar{\alpha}(X_t^n, X_t^n, X_*^{n-1}, X_*^n)$ **then** ▷ See Equation (3).
      $X_t^{n-1}, X_t^n \leftarrow X_*^{n-1}, X_*^n$ ▷ Generalised PT swap (see Section 3.1)
    end if
  end for
end for
Output: Return: $\mathbf{X}_1, \dots, \mathbf{X}_T$

Notably, the acceptance rate is equal to 1 if $\bar{\pi}_{n-1}^P \equiv \bar{\pi}_n^Q$. Therefore, we aim to choose P and Q so that the two distributions are as close as possible. Moreover, we recover the classical PT swap move when the forward and backward kernels are the identity kernels. Finally, Proposition 1 shows that the generalised swap kernel $\bar{\mathbf{K}}_{n-1,n}$ formalised above is valid and can be used within a PT algorithm instead of the classic swap kernel $\mathbf{K}_{n-1,n}$. See Appendix B for the proof.

Proposition 1. *Suppose that the two measures $\bar{\pi}_{n-1}^P$ and $\bar{\pi}_n^Q$ defined in Equation* (4) *are mutually absolutely continuous. Then $\bar{\mathbf{K}}_{n-1,n}$ keeps the distribution $\boldsymbol{\pi}$ invariant.*

$$\begin{aligned} \bar{\pi}_{n-1}^P(\mathrm{d}x^{n-1}, \mathrm{d}x^n) &= \pi_{n-1}(\mathrm{d}x^{n-1}) P_n(x^{n-1}, \mathrm{d}x^n), \\ \bar{\pi}_n^Q(\mathrm{d}x^{n-1}, \mathrm{d}x^n) &= \pi_n(\mathrm{d}x^n) Q_{n-1}(x^n, \mathrm{d}x^{n-1}). \end{aligned} \tag{4}$$

3.2 GePT with Deterministic flows

We now return to the motivation outlined at the beginning of this section and consider forward and backward kernels defined by a mapping $F_n : \mathcal{X} \to \mathcal{X}$. The following proposition, of which the proof is postponed to Appendix B.3, gives a readily calculable expression for $\bar{\alpha}_n$ in this case.

Proposition 2. *Let $\mathcal{X} = \mathbb{R}^d$ and suppose that π_{n-1} and π_n admit strictly positive densities $\tilde{\pi}_{n-1}$ and $\tilde{\pi}_n$ with respect to the Lebesgue measure. Let $F_n : \mathbb{R}^d \to \mathbb{R}^d$ be a diffeomorphism with Jacobian matrix $J_{F_n}(x)$. If we choose forward and backward kernels P_n and Q_{n-1} such that*

$$P_n(x^{n-1}, \mathrm{d}x_*^n) = \delta_{F_n(x^{n-1})}(\mathrm{d}x_*^n) \qquad Q_{n-1}(x^n, \mathrm{d}x_*^{n-1}) = \delta_{F_n^{-1}(x^n)}(\mathrm{d}x_*^{n-1}), \tag{5}$$

then we have the following expression for $\bar{\alpha}_{n-1,n}$ given in Equation (3)*:*

$$\bar{\alpha}_{n-1,n}(x^{n-1}, x^n; x_*^{n-1}, x_*^n) = 1 \wedge \left[\frac{\tilde{\pi}_{n-1}(x_*^{n-1}) \tilde{\pi}_n(x_*^n)}{\tilde{\pi}_{n-1}(x^{n-1}) \tilde{\pi}_n(x_n)} \cdot \frac{|\det(J_{F_n}(x^{n-1}))|}{|\det(J_{F_n}(x_*^{n-1}))|} \right]. \tag{6}$$

We refer to an instance of GePT using the swap kernel described above as Flow-GePT. We note that Arbel et al. (2021); Matthews et al. (2022) use normalising flows in a similar manner to map between annealing distribution, but in the context of SMC samplers (Del Moral et al., 2006).

Training An advantage of the GePT framework is its flexibility in training. One can either run standard PT (without flows) or GePT (incorporating flows), where F_n is initialized at the identity transformation so that early training behaves like PT. After a sufficient burn-in period, samples from various distributions along the annealing path can be used to train the flows, gradually improving their ability to approximate transport between distributions. For further details, see Appendix E.

3.3 GePT with Diffusion Swaps

If we assume that we have access to the path of distributions $(\pi_s^{\mathrm{VP}})_{s \in [0,1]}$ induced by a VP diffusion process, we can construct an annealing path $\pi_n = \pi_{s_n}^{\mathrm{VP}}$ for GePT, given some annealing schedule

$0 = s_0 < s_1 \cdots < s_N = 1$. We note that the joint distribution $\pi^{\text{VP}}_{s_{n-1},s_n}(\mathrm{d}x^{n-1}, \mathrm{d}x^n)$ has the factorisations $\pi_{n-1}(\mathrm{d}x^{n-1})P_n(x^{n-1}, \mathrm{d}x^n) = \pi_n(\mathrm{d}x^n)Q_{n-1}(x^n, \mathrm{d}x^{n-1})$ where it is well-known that Q_{n-1} is a closed-form Gaussian kernel and P_n can be well-approximated with a Gaussian kernel derived from the form of the time-reversal SDE. Therefore, it is a natural choice to define P_n, Q_{n-1} by such kernels to ensure an improved acceptance probability. We note that the use of such a path and kernels is similar to Phillips et al. (2024) which employs these components within a SMC sampler instead. The following proposition, of which the proof is postponed to Appendix B.4, gives a readily calculable expression for $\bar{\alpha}_n$ in this case.

Proposition 3. *Let $\mathcal{X} = \mathbb{R}^d$ and suppose that π_{n-1} and π_n admit strictly positive densities $\tilde{\pi}_{n-1}$ and $\tilde{\pi}_n$ with respect to the Lebesgue measure. If we choose forward and backward kernels P_n and Q_{n-1} such that they admit strictly positive densities $\tilde{P}_n$ and $\tilde{Q}_{n-1}$ with respect to the Lebesgue measure, then we have the following expression for $\bar{\alpha}_n$ given in Equation* (3)*:*

$$\bar{\alpha}(x^{n-1}, x^n; x_*^{n-1}, x_*^n) = 1 \wedge \left[\frac{\tilde{\pi}_{n-1}(x_*^{n-1})\tilde{P}_n(x_*^{n-1}, x^n)\tilde{\pi}_n(x_*^n)\tilde{Q}_{n-1}(x_*^n, x^{n-1})}{\tilde{\pi}_{n-1}(x^{n-1})\tilde{P}_n(x^{n-1}, x_*^n)\tilde{\pi}_n(x_n)\tilde{Q}_{n-1}(x^n, x_*^{n-1})} \right].$$

The expression in the denominator is the density of the quadruple $(x^{n-1}, x^n; x_*^{n-1}, x_*^n)$ before the swap. The expression in the numerator can be thought of informally as a hypothetical 'post-swap density', i.e. when (x^{n-1}, x^n) becomes (x_*^{n-1}, x_*^n) and vice versa.

We refer to an instance of GePT using the swap kernel described above as Diff-GePT. We also note that the above kernel can be readily generalised to the case where we use $m > 1$ Markov kernels to generate our proposed swap states via Nilmeier et al. (2011). We refer to this algorithm as m-step Diff-GePT where $m = 1$ coincides with using the above kernel. For further details, see Appendix D.

Training To implement this algorithm in practice, as we do not have access, in general, to the analytical form of the diffusion path $(\pi_s^{\text{VP}})_{s\in[0,1]}$, we parametrise an energy-based model (Salimans & Ho, 2021) π_s^θ using a modified version of the parametrisation from Phillips et al. (2024) which satisfies the boundary conditions $\pi_0^\theta = \mathcal{N}(0, \mathrm{I})$ and $\pi_1^\theta = \pi$. We can then iteratively run Diff-GePT with the approximation $(\pi_s^\theta)_{s\in[0,1]}$ in order to generate samples from π that are then used to improve the approximation π_s^θ via the score-matching objective. We stress that a parallelised implementation of Diff-GePT allows for generating a sample from π with a similar effective cost as a single discretisation step solving the time-reversal SDE. For further details, see Appendix F.

4 Experiments

We evaluate our proposed algorithms: Flow-GePT and Diff-GePT, against the baseline of PT using the linear path with reference $\pi_0 = \mathcal{N}(0, \mathrm{I})$ (Linear-PT), on the commonly used 40-mode Gaussian mixture model (GMM-d) distribution in d dimensions (Midgley et al., 2023). See Appendix G.1 for further details about the target distribution we use.

4.1 Flow-GePT Experiments

We compare Flow-GePT on GMM-2 with Linear-PT using six chains (see Figure 1 for the generated samples). We observe that six chains are insufficient for PT to effectively sample from GMM-2, as most modes are missed, with samples primarily generated in modes that remain close to the reference. In contrast, samples from Flow-GePT, shown on the right of Figure 1, demonstrate that incorporating flows facilitates sampling from all modes of the distribution. Additionally, the swap accept-reject mechanism ensures that the local shapes are accurately preserved. See Appendix G.2 for more details.

4.2 Diff-GePT Experiments

We compare m-step Diff-GePT ($m = 1, 2, 5$) against the baselines of Linear-PT and PT with the diffusion path (Diff-PT) in terms of round trip rate to assess the improved communication of states by the Diff-GePT swap kernels. In addition, to control for the additional sequential computation that m-step Diff-GePT requires (when considering parallelised implementations of all algorithms), we

also report the *compute-normalised round trip rate*. Further, to ensure a fair comparison, we tune the annealing schedule with the algorithm from Syed et al. (2021) and use the same local exploration kernels for all methods. For further details, see Appendix G.3.

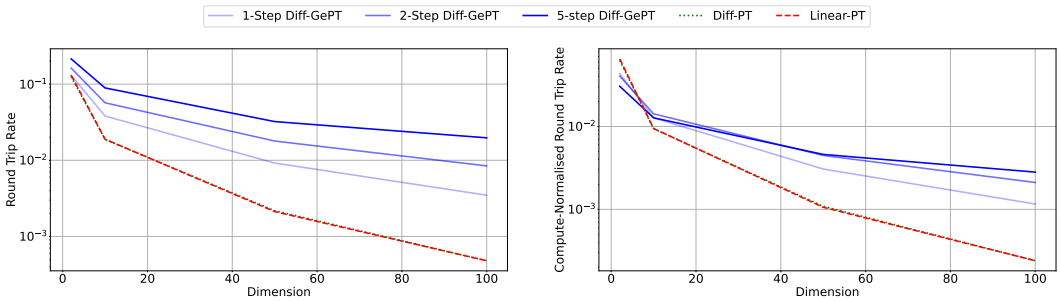

Figure 2: Round trip metrics for m-step Diff-GePT ($m = 1, 2, 5$) and Diff-PT using the true diffusion path, and Linear-PT targeting GMM-d for $d = 2, 10, 50, 100$ when using 30 chains. (Left) Round trip rate against d. (Right) Compute-normalised round trip rate against d.

Scaling of (ideal) Diff-GePT with dimensions To understand the theoretical performance of m-step Diff-GePT for a fixed number of chains when we scale d, we use the fact that the path of distributions $(\pi_s^{\text{VP}})_{s\in[0,1]}$ induced by a VP diffusion process initialised at GMM-d is analytically tractable, in order to run m-step Diff-GePT with the true diffusion path. In Figure 2, we compare the resulting round trip rate and compute-normalised round trip rate of the different algorithms when we fix the number of chains to 30.

We see that, for all values of d, the round trip rate of Diff-GePT monotonically increases over Diff-PT and Linear-PT as we increase m, with greater gains for larger values of d, demonstrating that our proposed Diff-GePT swap kernels improve the communication of states over the standard swap kernel. Further, while there is a smaller difference between algorithms and we do see that Linear-PT and Diff-PT is able to outperform Diff-GePT for $d = 2$, overall, we note a similar trend with the compute-normalised round trip rate, suggesting the additional computation for Diff-GePT is justified.

Table 1: Compute-normalised round trip rate of m-step Diff-GePT ($m = 1, 2, 5$) and Diff-PT using a learned diffusion path, and Linear-PT targeting GMM-d for $d = 2, 10, 50$ where the number of chains is 10, 30 and 60.

Methods	GMM-2			GMM-10			GMM-50		
	10	30	60	10	30	60	10	30	60
1-Diff-GePT	0.0368	0.0391	0.0391	0.00532	0.0120	**0.0132**	0.00	0.00319	0.00486
2-Diff-GePT	0.0333	0.0345	0.0331	0.00749	**0.0126**	0.0123	**0.00045**	**0.00425**	**0.00534**
5-Diff-GePT	0.0237	0.0227	0.0219	**0.00790**	0.0102	0.0072	0.00140	0.00379	0.00394
Diff-PT	0.0529	0.0604	0.0636	0.00339	0.00914	**0.0132**	0.00	0.00108	0.00203
Linear-PT	**0.0584**	**0.0654**	**0.0695**	0.00330	0.00944	0.0129	0.00	0.00106	0.00195

Learned Diff-GePT We also compare the performance of m-step Diff-GePT and Diff-PT with a learned diffusion path targeting GMM-d when we scale d and N; for details about training, see Appendix G.3. In Table 1, we report the compute-normalised round trip rate of each method where we scale d in $\{2, 10, 50\}$ and the number of chains in $\{10, 30, 60\}$.

We see that increasing the number of chains tends to increase compute-normalised round trip rates, however with diminishing returns due to the corresponding increase in distance that states have to traverse. In addition, we see that the performance of m-step Diff-GePT for a fixed number of chains, on the whole, largely mirrors the trends seen from Diff-GePT with the true diffusion path in Figure 2, but with reduced rates. For example, 1-step Diff-GePT with 30 chains targeting GMM-50 provides a $3\times$ improvement on rates compared to the corresponding Linear-PT samples. These results demonstrate that our Diff-GePT swap kernels can still improve communication with the

additional cost justified, and with substantial gains in higher dimensions, even with an imperfect approximation to $(\pi_s^{\mathrm{VP}})_{s\in[0,1]}$.

5 Conclusion

In this work, we have introduced and formalised the design of more flexible swap kernels within PT under the framework of Generalised Parallel Tempering. In addition, we have proposed two concrete examples, Flow-GePT and Diff-GePT which utilise normalising flow and diffusion models to construct swaps. Empirically, we have demonstrated the ability of these methods to improve sample quality over classical PT even when controlling for additional computational cost. In the future, we intend to validate GePT on more complex distributions and to further refine the training procedure involved in GePT.

Acknowledgments

The authors are grateful to George Deligiannidis for helpful discussions. LZ and PP are supported by the EPSRC CDT in Modern Statistics and Statistical Machine Learning (EP/S023151/1). SS acknowledges support from the EPSRC CoSInES grant (EP/R034710/1) and the NSERC Postdoctoral Fellowship.

References

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, Nicolas Malkin, and Alexander Tong. Iterated denoising energy matching for sampling from Boltzmann densities. In *International Conference on Machine Learning*, 2024.

M. S. Albergo, G. Kanwar, and P. E. Shanahan. Flow-based generative models for markov chain monte carlo in lattice field theory. *Phys. Rev. D*, 100:034515, Aug 2019. doi: 10.1103/PhysRevD.100.034515. URL https://link.aps.org/doi/10.1103/PhysRevD.100.034515.

Michael S Albergo and Eric Vanden-Eijnden. Nets: A non-equilibrium transport sampler. *arXiv preprint arXiv:2410.02711*, 2024.

Michael Arbel, Alex Matthews, and Arnaud Doucet. Annealed flow transport Monte Carlo. In *International Conference on Machine Learning*, pp. 318–330. PMLR, 2021.

Andrew J Ballard and Christopher Jarzynski. Replica exchange with nonequilibrium switches. *Proceedings of the National Academy of Sciences*, 106(30):12224–12229, 2009.

Andrew J Ballard and Christopher Jarzynski. Replica exchange with nonequilibrium switches: Enhancing equilibrium sampling by increasing replica overlap. *The Journal of Chemical Physics*, 136(19), 2012.

Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *Transactions on Machine Learning Research*, 2024.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

Arnaud Doucet, Will Grathwohl, Alexander G Matthews, and Heiko Strathmann. Score-based diffusion meets annealed importance sampling. *Advances in Neural Information Processing Systems*, 35:21482–21494, 2022.

Marylou Gabrié, Grant M. Rotskoff, and Eric Vanden-Eijnden. Adaptive monte carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, 119(10):e2109420119, 2022. doi: 10.1073/pnas.2109420119. URL https://www.pnas.org/doi/abs/10.1073/pnas.2109420119.

Charles J Geyer. Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, 1991.

Jiajun He, Yuanqi Du, Francisco Vargas, Dinghuai Zhang, Shreyas Padhy, RuiKang OuYang, Carla Gomes, and José Miguel Hernández-Lobato. No trick, no treat: Pursuits and challenges towards simulation-free training of neural samplers. *arXiv preprint arXiv:2502.06685*, 2025.

Jérôme Hénin, Tony Lelièvre, Michael R Shirts, Omar Valsson, and Lucie Delemotte. Enhanced sampling methods for molecular dynamics simulations. *Living Journal of Computational Molecular Science*, 4(1), 2022.

Koji Hukushima and Koji Nemoto. Exchange Monte Carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608, 1996.

Michele Invernizzi, Andreas Kramer, Cecilia Clementi, and Frank Noe. Skipping the replica exchange ladder with normalizing flows. *The Journal of Physical Chemistry Letters*, 13(50):11643–11649, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Helmut G Katzgraber, Simon Trebst, David A Huse, and Matthias Troyer. Feedback-optimized parallel tempering Monte Carlo. *Journal of Statistical Mechanics: Theory and Experiment*, 2006 (03):P03018, 2006.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krzysztof Łatuszyński, Matthew T Moores, and Timothée Stumpf-Fétizon. MCMC for multi-modal distributions. *arXiv preprint arXiv:2501.05908*, 2025.

Martin Lingenheil, Robert Denschlag, Gerald Mathias, and Paul Tavan. Efficiency of exchange schemes in replica exchange. *Chemical Physics Letters*, 478(1-3):80–84, 2009.

Alexander G. D. G. Matthews, Michael Arbel, Danilo J. Rezende, and Arnaud Doucet. Continual repeated annealed flow transport Monte Carlo. In *International Conference on Machine Learning*, 2022.

Laurence Illing Midgley, Vincent Stimper, Gregor NC Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. In *International Conference on Learning Representations*, 2023.

Radford M Neal. MCMC using Hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

Jerome P Nilmeier, Gavin E Crooks, David DL Minh, and John D Chodera. Nonequilibrium candidate Monte Carlo is an efficient tool for equilibrium simulation. *Proceedings of the National Academy of Sciences*, 108(45):E1009–E1018, 2011.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Nikolas Nüsken, Francisco Vargas, Shreyas Padhy, and Denis Blessing. Transport meets variational inference: Controlled Monte Carlo diffusions. In *International Conference on Learning Representations*, 2024.

Tsuneyasu Okabe, Masaaki Kawata, Yuko Okamoto, and Masuhiro Mikami. Replica-exchange Monte Carlo method for the isobaric–isothermal ensemble. *Chemical Physics Letters*, 335(5-6): 435–439, 2001.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Theodore Papamarkou, Jacob Hinkle, M Todd Young, and David Womble. Challenges in Markov chain Monte Carlo for Bayesian neural networks. *Statistical Science*, 37(3):425–442, 2022.

Angus Phillips, Hai-Dang Dau, Michael John Hutchinson, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Particle denoising diffusion sampler. In *International Conference on Machine Learning*, 2024.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363, 1996.

Tim Salimans and Jonathan Ho. Should EBMs model the energy or the score? In *Energy Based Models Workshop-ICLR 2021*, 2021.

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

Nikola Surjanovic, Saifuddin Syed, Alexandre Bouchard-Côté, and Trevor Campbell. Parallel tempering with a variational reference. *Advances in Neural Information Processing Systems*, 35: 565–577, 2022.

Nikola Surjanovic, Miguel Biron-Lattes, Paul Tiede, Saifuddin Syed, Trevor Campbell, and Alexandre Bouchard-Côté. Pigeons. jl: Distributed sampling from intractable distributions. *arXiv preprint arXiv:2308.09769*, 2023.

Nikola Surjanovic, Saifuddin Syed, Alexandre Bouchard-Côté, and Trevor Campbell. Uniform ergodicity of parallel tempering with efficient local exploration. *arXiv preprint arXiv:2405.11384*, 2024.

Robert H. Swendsen and Jian-Sheng Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57:2607–2609, 1986.

Saifuddin Syed, Vittorio Romaniello, Trevor Campbell, and Alexandre Bouchard-Côté. Parallel tempering on optimized paths. In *International Conference on Machine Learning*, 2021.

Saifuddin Syed, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Non-reversible parallel tempering: a scalable highly parallel MCMC scheme. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(2):321–350, 2022.

Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

Yifeng Tian, Nishant Panda, and Yen Ting Lin. Liouville flow importance sampler. In *International Conference on Machine Learning*, 2024.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. In *International Conference on Learning Representations*, 2023.

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

Dawn B Woodard, Scott C Schmidler, and Mark Huber. Conditions for rapid mixing of parallel and simulated tempering on multimodal distributions. *The Annals of Applied Probability*, 19(2): 617–640, 2009.

Dinghuai Zhang, Ricky TQ Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *arXiv preprint arXiv:2310.02679*, 2023.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. In *International Conference on Machine Learning*, 2022.

A PT KERNELS

A.1 LOCAL EXPLORATION

The local exploration kernel $\mathbf{K}^{\text{expl}}(\mathbf{x}, \mathrm{d}x')$ generates sample $\mathbf{x}' = (x'^0, \ldots, x'^N)$ by updating each coordinate of $\mathbf{x} = (x^0, \ldots, x^N)$ using a $K_n(x, \mathrm{d}x')$ is a π_n-invariant Markov kernel. We will further assume that $K_0(x, \mathrm{d}x') = \pi_0(\mathrm{d}x')$ draws an independent sample from the reference. Formally, the local exploration kernel equals,

$$\mathbf{K}^{\text{expl}}(\mathbf{x}, \mathrm{d}\mathbf{x}') = \prod_{n=0}^{N} K_n(x^n, \mathrm{d}x'^n) = \pi_0(\mathrm{d}x'^0) \prod_{n=1}^{N} K_n(x^n, \mathrm{d}x'^n)$$

We can simulate from $\mathbf{K}_{n-1,n}$ using Algorithm 2.

Algorithm 2 Local exploration

Input: $\mathbf{x} = (x^0, \ldots, x^N)$
$x'^0 \sim \pi_0(\mathrm{d}x'^0)$ ▷ Generate sample from reference
for $n = 1, \ldots, N$ **do** ▷ Parallelisable
  $x'^n \sim K_n(x^n, \mathrm{d}x'^n)$ ▷ Run MCMC targeting π_n
end for
Output: $(x'^0, \ldots, x'^N)$

A.2 PT SWAP

For $n = 1, \ldots, N$ define the swap function $S_{n-1,n}(\mathbf{x}, x_*^{n-1}, x_*^n)$ that replaces the $n-1$ and n-th coordinates of $\mathbf{x} = (x^0, \ldots, x^N)$ with x_*^{n-1}, x_*^n respectively,

$$S_{n-1,n}(\mathbf{x}, x_*^{n-1}, x_*^n) = (x^0, \ldots, x^{n-2}, x_*^{n-1}, x_*^n, x^{n+1}, \ldots, x^N).$$

The swap kernel $\mathbf{K}_{n-1,n}(\mathbf{x}, \mathrm{d}\mathbf{x}')$ deterministically proposes the state $S_{n-1,n}(\mathbf{x}, x^n, x^{n-1})$ swapping components of x^{n-1} and x^n in $\mathbf{x}'$, and accepts with probability $\alpha_{n-1,n}(\mathbf{x})$ depending only on x^{n-1} and x^n,

$$\alpha_{n-1,n}(\mathbf{x}) = 1 \wedge \frac{[\mathrm{d}\pi_n/\mathrm{d}\pi_{n-1}](x^{n-1})}{[\mathrm{d}\pi_n/\mathrm{d}\pi_{n-1}](x^n)}.$$

Formally, the PT swap kernel equals,

$$\mathbf{K}_{n-1,n}(\mathbf{x}, \mathrm{d}\mathbf{x}') = \alpha_{n-1,n}(\mathbf{x})\delta_{S_{n-1,n}(\mathbf{x}, x^n, x^{n-1})}(\mathrm{d}\mathbf{x}') + (1 - \alpha_{n-1,n}(\mathbf{x}))\delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}').$$

We can simulating from $\mathbf{K}_{n-1,n}$ using Algorithm 3.

Algorithm 3 PT swap between $n-1$ and n

Input: $\mathbf{x} = (x^0, \ldots, x^N)$
$U \sim \text{Uniform}[0, 1]$
if $U < \alpha_{n-1,n}(\mathbf{x})$ **then**
  $x^{n-1}, x^n \leftarrow x^n, x^{n-1}$ ▷ Swap components $n-1$ and n of $\mathbf{x}$
end if
Output: $\mathbf{x}$

A.3 GENERALISED PT SWAP

The generalised PT swap kernel $\bar{\mathbf{K}}_{n-1,n}(\mathbf{x}, \mathrm{d}x')$ proposes a state $\mathbf{x}_* = (x_*^0, \ldots, x_*^N)$ generated by $\mathbf{Q}_{n-1,n}$ which replaces the $n-1$ and n-th coordinates of $\mathbf{x} = (x^0, \ldots, x^N)$ using the backwards kernels and forward kernels applied to x^n and x^{n-1} respectively,

$$\mathbf{Q}_{n-1,n}(\mathbf{x}, \mathrm{d}\mathbf{x}_*) = Q_n(x^n, \mathrm{d}x_*^{n-1})P_n(x^{n-1}, \mathrm{d}x_*^n) \prod_{m \neq n-1,n} \delta_{x^m}(\mathrm{d}x_*^m) \tag{7}$$

The proposed state $\mathbf{x}_*$ is then accepted with probability $\bar{\alpha}_{n-1,n}(\mathbf{x};\mathbf{x}_*)$ depending only on the $n-1$ and n-th coordinates of $\mathbf{x}$ and $\mathbf{x}_*$,

$$\bar{\alpha}_{n-1,n}(\mathbf{x};\mathbf{x}_*) = 1 \wedge \frac{[\mathrm{d}\bar{\pi}_n^Q/\mathrm{d}\bar{\pi}_{n-1}^P](x^{n-1}, x_*^n)}{[\mathrm{d}\bar{\pi}_n^Q/\mathrm{d}\bar{\pi}_{n-1}^P](x_*^{n-1}, x^n)},$$

where recall $\bar{\pi}_{n-1}^P = \pi_{n-1} \times P_n$ and $\bar{\pi}_n^Q = \pi_n \times Q_{n-1}$ are the forward and backward extensions of π_{n-1} and π_n respectively defined in Equation (4). Formally, the generalised PT swap kernel equals,

$$\begin{aligned}\bar{\mathbf{K}}_{n-1,n}(\mathbf{x}, \mathrm{d}\mathbf{x}') = &\int \bar{\alpha}_{n-1,n}(\mathbf{x};\mathbf{x}_*)\mathbf{Q}_{n-1,n}(\mathbf{x}, \mathrm{d}\mathbf{x}_*)\delta_{S_{n-1,n}(\mathbf{x}, x_*^{n-1}, x_*^n)}(\mathrm{d}\mathbf{x}') \\ &+ \int (1 - \bar{\alpha}_{n-1,n}(\mathbf{x};\mathbf{x}_*))\mathbf{Q}_{n-1,n}(\mathbf{x}, \mathrm{d}\mathbf{x}_*)\delta_{\mathbf{x}}(\mathrm{d}\mathbf{x}')\end{aligned}$$

We can simulate from $\bar{\mathbf{K}}_{n-1,n}$ using Algorithm 4.

Algorithm 4 GePT swap between $n-1$ and n

Input: $\mathbf{x} = (x^0, \ldots, x^N)$
  $x_*^{n-1} \sim Q_{n-1}(x^n, \mathrm{d}x_*^{n-1})$ ▷ Backward proposal
  $x_*^n \sim P_n(x^{n-1}, \mathrm{d}x_*^n)$ ▷ Forward proposal
  $U \sim \mathrm{Uniform}([0,1])$
  if $U < \alpha(x^{n-1}, x^n; x_*^{n-1}, x_*^n)$ **then**
    $x^{n-1}, x^n \leftarrow x_*^{n-1}, x_*^n$ ▷ Swap x^{n-1}, x^n with x_*^{n-1}, x_*^n.
  end if
Output: $\mathbf{x}$

B PROOFS

B.1 LEMMA 1

Lemma 1 shows applying the swap kernel $\mathbf{K}_{n-1,n}$ leaves $\boldsymbol{\pi}$ invariant.

Lemma 1. *Suppose that the two measures π_{n-1} and π_n are mutually absolutely continuous. Then $\mathbf{K}_{n-1,n}$ keeps the distribution $\boldsymbol{\pi}$ invariant.*

Remark. The correctness of the swap step is well-known for the form of acceptance probability given by the ratio of products of densities (see Equation (2)). In this lemma, we stress on the general form of acceptance probability given by the ratio of Radon-Nikodym derivatives (see Equation (1)). This allows us to cast the *generalised* swap kernel as an instance of the *classic* swap kernel (see the proof of Proposition 1 given in Appendix B.2). More importantly it facilitates the derivation of the generalised swap probability for GePT with deterministic flows (see Section 3.2) by unifying it in the same framework as stochastic GePT (Section 3.3).

Proof. Let (X^{n-1}, X^n) be distributed according to $\pi_{n-1}(\mathrm{d}x^{n-1})\pi_n(\mathrm{d}x^n)$. The swap step is performed by first simulating a random variable U uniformly in $[0,1]$, then updating the states as

$$(X_{\text{new}}^{n-1}, X_{\text{new}}^n) = \begin{cases}(X^n, X^{n-1}) & \text{if } U < \alpha_n(x^{n-1}, x^n) \\ (X^{n-1}, X^n) & \text{otherwise.}\end{cases}$$

To prove that the swap keeps $\boldsymbol{\pi}$ invariant, we consider a continuous bounded test function $\varphi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and show that

$$\mathbb{E}[\varphi(X_{\text{new}}^{n-1}, X_{\text{new}}^n)] = \mathbb{E}[\varphi(X^{n-1}, X^n)].$$

We write, using the shorthand $\alpha = \alpha_n(X^{n-1}, X^n)$ where necessary,

$$\begin{aligned}\mathbb{E}[\varphi(X_{\text{new}}^{n-1}, X_{\text{new}}^n)] &= \mathbb{E}[\varphi(X_{\text{new}}^{n-1}, X_{\text{new}}^n)\mathbb{1}_{U<\alpha}] + \mathbb{E}[\varphi(X_{\text{new}}^{n-1}, X_{\text{new}}^n)\mathbb{1}_{U>\alpha}] \\ &= \mathbb{E}[\varphi(X^n, X^{n-1})\alpha(X^{n-1}, X^n)] + \mathbb{E}[\varphi(X^{n-1}, X^n)(1 - \alpha(X^{n-1}, X^n))] \\ &= \mathbb{E}[\varphi(X^{n-1}, X^n)] + \left\{\mathbb{E}[\varphi(X^n, X^{n-1})\alpha] - \mathbb{E}[\varphi(X^{n-1}, X^n)\alpha]\right\}.\end{aligned} \tag{8}$$

We have

$$\begin{aligned}\mathbb{E}[\varphi(X^n, X^{n-1})\alpha] &= \mathbb{E}_{x^{n-1}, x^n \overset{\text{iid}}{\sim} \pi_{n-1}}\left[\varphi(X^n, X^{n-1})\frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(X^n)\alpha\right] \\ &= \mathbb{E}_{x^{n-1}, x^n \overset{\text{iid}}{\sim} \pi_{n-1}}\left[\varphi(X^n, X^{n-1})\left(\frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(X^n) \wedge \frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(X^{n-1})\right)\right].\end{aligned} \tag{9}$$

Similarly we also have

$$\begin{aligned}\mathbb{E}[\varphi(X^{n-1}, X^n)\alpha] &= \mathbb{E}_{x^{n-1}, x^n \overset{\text{iid}}{\sim} \pi_{n-1}}\left[\varphi(X^{n-1}, X^n)\left(\frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(X^n) \wedge \frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(X^{n-1})\right)\right] \\ &= \mathbb{E}_{x^{n-1}, x^n \overset{\text{iid}}{\sim} \pi_{n-1}}\left[\varphi(X^n, X^{n-1})\left(\frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(X^n) \wedge \frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(X^{n-1})\right)\right]\end{aligned} \tag{10}$$

where the last equality holds thanks to the symmetry of the measure

$$\pi_{n-1}(\mathrm{d}x^{n-1})\pi_{n-1}(\mathrm{d}x^n)\left(\frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(x^n) \wedge \frac{\mathrm{d}\pi_n}{\mathrm{d}\pi_{n-1}}(x^{n-1})\right)$$

with respect to x^{n-1} and x^n. Combining Equation (8), Equation (9), and Equation (10), we have

$$\mathbb{E}[\varphi(X^{n-1}_{\text{new}}, X^n_{\text{new}})] = \mathbb{E}[\varphi(X^{n-1}, X^n)]$$

which concludes the proof. □

B.2 Proof of Proposition 1

Proof. The main idea is to express the *generalised* PT swap kernel as the *classic* PT swap kernel of another appropriately-defined PT problem.

Given $(X^{n-1}, X^n) \sim \pi^{n-1} \otimes \pi^n$, the first step of the generalised swap kernel simulates $X^{n-1}_* \sim Q_{n-1}(X^n, \mathrm{d}x^{n-1})$ and $X^n_* \sim P_n(X^{n-1}, \mathrm{d}x^n)$. Consider now a *conceptual* parallel tempering problem *with only two* annealing distributions

$$\begin{aligned}\mu_1(\mathrm{d}x^{n-1}, \mathrm{d}x^n) &:= \pi_{n-1}(\mathrm{d}x^{n-1})P_n(x^{n-1}, \mathrm{d}x^n) \\ \mu_2(\mathrm{d}x^{n-1}, \mathrm{d}x^n) &:= \pi_n(\mathrm{d}x^n)Q_{n-1}(x^n, \mathrm{d}x^{n-1}).\end{aligned} \tag{11}$$

We have that $(X^{n-1}, X^n_*) \sim \mu_1$ and $(X^{n-1}_*, X^n) \sim \mu_2$. Moreover (X^{n-1}, X^n_*) is independent from (X^{n-1}_*, X^n). As such, $(X^{n-1}, X^n_*; X^{n-1}_*, X^n)$ is a stationary state of the parallel tempering problem given by Equation (11).

We now apply the *classic* swap kernel $\mathbf{K}_{n-1,n}$ to this particular parallel tempering problem. The kernel simulates $(Y^{n-1}, Y^n_*; Y^{n-1}_*, Y^n)$ such that

$$(Y^{n-1}, Y^n_*; Y^{n-1}_*, Y^n) = \begin{cases}(X^{n-1}_*, X^n; X^{n-1}, X^n_*) & \text{with probability } \alpha_{12} \\ (X^{n-1}, X^n_*; X^{n-1}_*, X^n) & \text{with probability } 1 - \alpha_{12}\end{cases} \tag{12}$$

where

$$\alpha_{12} := \frac{\mathrm{d}\mu_2/\mathrm{d}\mu_1(x^{n-1}, x^n_*)}{\mathrm{d}\mu_2/\mathrm{d}\mu_1(x^{n-1}_*, x^n)}.$$

Notice that Equation (12) can be marginalized on the first and the last components to read

$$(Y^{n-1}, Y^n) = \begin{cases}(X^{n-1}_*, X^n_*) & \text{with probability } \alpha_{12} \\ (X^{n-1}, X^n) & \text{with probability } 1 - \alpha_{12}.\end{cases}$$

Additionally, it is readily checked that the definition of α_{12} coincides with Equation (3).

Therefore, the *generalised* swap applied to the original PT problem coincides with the *classic* swap applied to the artificial PT problem defined by Equation (11). Lemma 1 asserts that the classic swap is invariant, i.e.

$$(Y^{n-1}, Y^n_*; Y^{n-1}_*, Y^n) \sim \mu_1(\mathrm{d}y^{n-1}, \mathrm{d}y^n_*) \otimes \mu_2(\mathrm{d}y^{n-1}_*, \mathrm{d}y^n).$$

In particular

$$(Y^{n-1}, Y^n) \sim \pi_{n-1}(\mathrm{d}y^{n-1}) \otimes \pi_n(\mathrm{d}y^n).$$

As a result, the generalised PT swap keeps $\pi_{n-1} \times \pi_n$, hence $\boldsymbol{\pi}$, invariant. □

B.3 Proof of Proposition 2

Proof. Let $S := \{(z^{n-1}, z^n) \in \mathbb{R}^d \times \mathbb{R}^d$ such that $F_n(z^{n-1}) = z^n\}$. Recall the definition of Equation (4):

$$\bar{\pi}_n^Q(\mathrm{d}z^{n-1}, \mathrm{d}z^n) = \pi_n(\mathrm{d}z^n) Q_{n-1}(z^n, \mathrm{d}z^{n-1}).$$

Moreover for $(z^{n-1}, z^n) \in S$,

$$\bar{\pi}_{n-1}^P(\mathrm{d}z^{n-1}, \mathrm{d}z^n) = (F_n \# \pi_{n-1})(\mathrm{d}z^n) Q_{n-1}(z^n, \mathrm{d}z^{n-1}).$$

Therefore

$$\begin{aligned}\frac{\mathrm{d}\bar{\pi}_n^Q}{\mathrm{d}\bar{\pi}_{n-1}^P}(z^{n-1}, z^n) &= \frac{\pi_n(\mathrm{d}z^n)}{(F_n \# \pi_{n-1})(\mathrm{d}z^n)} \\ &= \frac{\tilde{\pi}_n(z_n)}{\tilde{\pi}_{n-1}(F_n^{-1}(z^n))|\det(J_{F_n^{-1}}(z^n))|} = \frac{\tilde{\pi}_n(z^n)|\det(J_{F_n}(z^{n-1}))|}{\tilde{\pi}_{n-1}(z^{n-1})}. \quad (13)\end{aligned}$$

Applying this identity at $(z^{n-1}, z^n) = (x^{n-1}, x_*^n) \in S$ gives

$$\frac{\mathrm{d}\bar{\pi}_n^Q}{\mathrm{d}\bar{\pi}_{n-1}^P}(x^{n-1}, x_*^n) = \frac{\tilde{\pi}_n(x_*^n)|\det(J_{F_n}(x^{n-1}))|}{\tilde{\pi}_{n-1}(x^{n-1})}. \quad (14)$$

Similarly, applying Equation (13) at $(z^{n-1}, z^n) = (x_*^{n-1}, x^n) \in S$ gives

$$\frac{\mathrm{d}\bar{\pi}_n^Q}{\mathrm{d}\bar{\pi}_{n-1}^P}(x_*^{n-1}, x^n) = \frac{\tilde{\pi}_n(x^n)|\det(J_{F_n}(x_*^{n-1}))|}{\tilde{\pi}_{n-1}(x_*^{n-1})}. \quad (15)$$

Together Equation (14) and Equation (15) establish the proposition. □

B.4 Proof of Proposition 3

Proof. Since the relevant measures and kernels have strictly positive densities with respect to the Lebesgue measure, we have, for any $(z^{n-1}, z^n) \in \mathbb{R}^d \times \mathbb{R}^d$,

$$\frac{\mathrm{d}\bar{\pi}_n^Q}{\mathrm{d}\bar{\pi}_{n-1}^P}(z^{n-1}, z^n) = \frac{\tilde{\pi}_n(z^n)\tilde{Q}_{n-1}(z^n, z^{n-1})}{\tilde{\pi}_{n-1}(z^{n-1})\tilde{P}_n(z^{n-1}, z^n)}. \quad (16)$$

Applying this identity at $(z^{n-1}, z^n) = (x^{n-1}, x_*^n)$ gives

$$\frac{\mathrm{d}\bar{\pi}_n^Q}{\mathrm{d}\bar{\pi}_{n-1}^P}(x^{n-1}, x_*^n) = \frac{\tilde{\pi}_n(x_*^n)\tilde{Q}_{n-1}(x_*^n, x^{n-1})}{\tilde{\pi}_{n-1}(x^{n-1})\tilde{P}_n(x^{n-1}, x_*^n)}. \quad (17)$$

Similarly, applying Equation (16) at $(z^{n-1}, z^n) = (x_*^{n-1}, x^n)$ gives

$$\frac{\mathrm{d}\bar{\pi}_n^Q}{\mathrm{d}\bar{\pi}_{n-1}^P}(x_*^{n-1}, x^n) = \frac{\tilde{\pi}_n(x^n)\tilde{Q}_{n-1}(x^n, x_*^{n-1})}{\tilde{\pi}_{n-1}(x_*^{n-1})\tilde{P}_n(x_*^{n-1}, x^n)}. \quad (18)$$

Together Equation (17) and Equation (18) establish the proposition. □

C Round Trip Rate

In order to define the round trip rate used within PT, we consider the Markov chain $(\mathbf{X}_t)_{t\in\mathbf{N}}$ constructed by PT, and imagine an auxiliary Markov chain $(\mathbf{I}_t)_{t\in\mathbf{N}}$ initialised at the state $\mathbf{I}_0 = (0, 1, \ldots, N)$. The subsequent values of $\mathbf{I}_t$ are then determined by the swap moves carried out by $\mathbf{K}_t^{\text{comm}}$ on $\mathbf{X}_t$ - i.e. we apply the same swaps to the components of $\mathbf{I}_t$ that are proposed and accepted by $\mathbf{K}_t^{\text{comm}}$ on the components of $\mathbf{X}_t$. This results in $\mathbf{I}_t$ defining some sequence of permutations of $[N] = \{0, \ldots, N\}$ tracking the underlying communication of states in $\mathbf{X}_t$. Hence, for a realisation of $\mathbf{X}_t$ with T steps, we define the *round trips for index* $n \in [N]$ as the number of times which the index n in $\mathbf{I}_t$ completes the journey from 0-th component, then to the N-th component and then back to the 0-th component - i.e. completes the round trip from the reference to the target and back again. The (overall) *round trips* is then defined as the sum of round trips for index $n \in [N]$ over all index values and the *round trip rate* is defined as the round trips divided by T.

D MULTISTEP GEPT SWAP KERNEL

We can extend the swap kernel presented in Section 3.3 to instead utilise a sequence of $m > 1$ Markov kernels to generate the proposed states.

Specifically, given m forward kernels P_n^j, $j = 1, \ldots, m$; and m backward kernels Q_{n-1}^k for $k = 1, \ldots, m$ having strictly positive densities with respect to the Lebesgue measure, the m-step GePT kernel $\bar{\mathbf{K}}_{n-1,n}^m(\mathbf{x}, d\mathbf{x}')$ proceeds as follows:

- Simulate the sequence $(x^{(n-1,j)})_{j=0}^m$ by setting $x^{(n-1,0)} = x^{n-1}$ and sequentially generate $x^{(n-1,j)} \sim P_n^j(x^{(n-1,j-1)}, \mathrm{d}x')$, for $j = 1, \ldots, m$;
- Simulate the sequence $(x^{(n,k)})_{k=0}^m$ by setting $x^{(n,m)} = x^n$ and sequentially generate $x^{(n,k)} \sim Q_{n-1}^{k+1}(x^{(n,k+1)}, \mathrm{d}x')$ for $k = 0, \ldots, m-1$;
- Replace components $n-1$ and n of $\mathbf{x}$ with $x^{(n,0)}$ and $x^{(n-1,m)}$ with probability equal to

$$\bar{\alpha}(\{x^{(n-1,0:m)}\}; \{x^{(n,0:m)}\}) = 1 \wedge \left[\frac{\pi_n(x^{(n-1,m)}) \prod_{k=0}^{m-1} Q_{n-1}^{k+1}(x^{(n-1,k+1)}, x^{(n-1,k)})}{\pi_{n-1}(x^{(n-1,0)}) \prod_{j=1}^{m} P_n^j(x^{(n-1,j-1)}, x^{(n-1,j)})} \times \right.$$
$$\left. \times \frac{\pi_{n-1}(x^{(n,0)}) \prod_{j=1}^{m} P_n^j(x^{(n,j-1)}, x^{(n,j)})}{\pi_n(x^{(n,m)}) \prod_{k=0}^{m-1} Q_{n-1}^{k+1}(x^{(n,k+1)}, x^{(n,k)})} \right]. \quad (19)$$

We note that when $m = 1$, this kernel coincides with the swap move presented in Section 3.3.

Proposition 4. *The multistep GePT swap kernel keeps invariant the distribution* $\boldsymbol{\pi}$.

Proof. We consider the two probability measures defined on $(\mathbb{R}^d)^{n+1}$:

$$\bar{\pi}_{(n-1,m)}^P(\mathrm{d}z^0, \ldots, \mathrm{d}z^m) = \pi_{n-1}(\mathrm{d}z^0) \prod_{j=1}^{m} P_n^j(z^{j-1}, \mathrm{d}z^j)$$
$$\bar{\pi}_{(n,m)}^Q(\mathrm{d}z^0, \ldots, \mathrm{d}z^m) = \pi_n(\mathrm{d}z^m) \prod_{k=0}^{m-1} Q_{n-1}^k(z^{k+1}, \mathrm{d}z_k).$$

Before the swap step, the two variables $x^{(n-1,0:m)}$ and $x^{(n,0:m)}$ are independent and are distributed according to $\bar{\pi}_{(n-1,m)}^P$ and $\bar{\pi}_{(n,m)}^Q$ respectively. This allows us to apply Lemma 1 to these two measures. In particular, Equation (1) reduces to

$$1 \wedge \frac{\mathrm{d}\bar{\pi}_{(n,m)}^Q / \mathrm{d}\bar{\pi}_{(n-1,m)}^P(x^{(n-1,0:m)})}{\mathrm{d}\bar{\pi}_{(n,m)}^Q / \mathrm{d}\bar{\pi}_{(n-1,m)}^P(x^{(n,0:m)})}$$

which, upon elementary algebra, coincides with Equation (19) and concludes the proof. □

E FURTHER DETAILS ON FLOW-GEPT

Since GePT provides approximate samples from both the target and prior densities at each flow iteration, it enables a range of training objectives. Some choices include maximum likelihood estimation (MLE) (equivalent to forward KL), reverse KL, and symmetric KL, which averages the two. Each has trade-offs: forward KL promotes mode covering, reverse KL is more mode-seeking, and symmetric KL balances both. GePT's parallel structure makes symmetric KL particularly effective by providing access to samples at each intermediate annealing distribution, a feature many other methods lack. For example, sequential Monte Carlo (SMC)-based approaches such as FAB (Midgley et al., 2023) and CRAFT (Matthews et al., 2022) rely on samples from only one side, limiting their choice of loss functions.

Additionally, we explore loss functions based on GePT's rejection rates, as the round trip rate is analytically linked to them. Since higher round trip rates indicate more efficient mixing, optimizing

for this metric improves sampling performance. Empirically, we found that combining reverse KL with an inverse round trip rate loss yielded the most stable and robust results across different settings. This combination was therefore used in our final experiments.

Specifically, the flows $\{F_n\}_{n=1}^N$, where each F_n aims to transport samples from π_{n-1} to π_n, are trained by optimizing the following loss:

$$\mathcal{L}(\{F_n\}) = \sum_{n=1}^{N} D_{\text{KL}}(F_n^{-1}\#\pi_n||\pi_{n-1}) + \sum_{n=1}^{N} \frac{\text{Rej}_n}{1-\text{Rej}_n}, \tag{20}$$

where $D_{\text{KL}}(F_n^{-1}\#\pi_n||\pi_{n-1})$ is equal to the reverse KL divergence between the target distribution π_n and the flow-induced distribution $F_n\#\pi_{n-1}$. The second term accounts for the expected swap rejection rates, defined as:

$$\text{Rej}_n = \mathbb{E}\left[1 - \bar{\alpha}_{n-1,n}(X^{n-1}, X^n; F_n^{-1}(X^n), F_n(X^{n-1}))\right]$$

where $X^{n-1} \sim \pi_{n-1}, X^n \sim \pi_n$, and $\bar{\alpha}_{n-1,n}$ is the acceptance probability from the deterministic GePT swap mechanism Equation (6). We note that the second term in Equation (20) favors minimizing Rej_n, the expected probability of rejection between chains. This formulation is motivated by an analytic expression for the round trip rate in terms of the rejection rates under simplifying assumptions (Syed et al., 2021). We reiterate that this loss formulation requires access to approximate samples from both π_{n-1} and π_n. This is naturally facilitated by GePT, whereas other methods typically rely on samples from only one side of the annealing path, limiting their flexibility in training.

The loss in Equation (20) is minimized if and only if the flows $\{F_n\}$ perfectly transport π_{n-1} to π_n, ensuring zero KL divergence and a rejection rate of zero. However, we emphasize that the correctness of the GePT framework—i.e., the invariance of the stationary distribution—is maintained even when the flows are imperfect. This guarantees asymptotically unbiased sampling regardless of flow accuracy, though better flow approximations improve efficiency.

F Further Details on Diff-GePT

Diffusion process We use the following Variance-Preserving (VP) diffusion process

$$dY_s = -\beta_s Y_s \mathrm{d}s + \sqrt{2\beta_s}\mathrm{d}W_s, \quad Y_0 \sim \pi,$$

with the choice of schedule $\beta_s = \frac{1}{2(1-s)}$ to define the path of distributions $(\pi_s^{\text{VP}})_{s\in(0,1]}$ where $Y_s \sim \pi_{1-s}^{\text{VP}}$ Due to the singularity at $s = 0$, π_0 is not defined by the path, but we can define this point to be a standard Gaussian as the path converges to this distribution in the limit as s approaches 0 in order to define the full annealing path $(\pi_s^{\text{VP}})_{s\in[0,1]}$.

Swap kernel The Markov kernel $Q_n(x^n, \mathrm{d}x')$ is well-known to have the closed-form expression given by $\mathcal{N}(\mathrm{d}x'; \sqrt{1-\alpha_{n-1}}x^n, \alpha_{n-1}\mathrm{I})$ where $\alpha_{n-1} = 1 - \exp(-2\int_{1-s_n}^{1-s_{n-1}} \beta_s \mathrm{d}s)$ which transports the distribution $\pi_{s_n}^{\text{VP}}$ to $\pi_{s_{n-1}}^{\text{VP}}$. The Markov kernel $P_n(x^{n-1}, \mathrm{d}x')$ can be given by any discretisation of the time-reversal SDE approximating the dynamics transporting $\pi_{s_{n-1}}^{\text{VP}}$ to $\pi_{s_n}^{\text{VP}}$. In particular, we use the exponential integrator for this approximation given by the kernel $P_n(x^{n-1}, \mathrm{d}x') = \mathcal{N}(\mathrm{d}x'; \mu_n(x^{n-1}), \alpha_{n-1}\mathrm{I})$ where

$$\mu_n(x) = \sqrt{1-\alpha_{n-1}}x + 2(1-\sqrt{1-\alpha_{n-1}})(x + \nabla \log \pi_{s_{n-1}}^{\text{VP}}(x)).$$

For the case of the m-step GePT swap kernel, we simply divide the interval $[s_{n-1}, s_n]$ into a finer discretisation $l_0 = s_{n-1} < l_1 \ldots < l_m = s_n$ and define P_n^j, Q_{n-1}^k as taking the same form as above but defined on the new intervals $[l_{j-1}, l_j]$ moving samples from $\pi_{s_{n-1}}^{\text{VP}}$ to $\pi_{s_n}^{\text{VP}}$ through the intermediate $\pi_{l_j}^{\text{VP}}$ distributions step by step (according to the direction encoded by these kernels - i.e. following the VP diffusion SDE or the time-reversal SDE). In practice, we simply take $\{l_j\}$ to be a uniform discretisation of $[s_{n-1}, s_n]$.

Network parametrisation We parametrise an energy-based model as outlined in Phillips et al. (2024) but modified to ensure that $\pi_0^\theta(x) \propto N(x; 0, \mathrm{I})$. For completeness, we specifically take $\log \pi_s^\theta(x) = \log g_s^\theta(x) - \frac{1}{2}||x||^2$ where

$$\log g_\theta(x, s) = [r_\theta(1) - r_\theta(s)][r_\theta(s) - r_\theta(0)]\langle N_\theta(x, s), x\rangle + [1 + r_\theta(1) - r_\theta(s)] \log g_0(\sqrt{s}x),$$

where $g_1(x) \propto \pi(x)N(x; 0, \mathrm{I})$. Here, r is a scalar-valued neural network and N is a vector-valued function in $\mathbb{R}^d$. We also note that $\pi_1^\theta(x) \propto \pi$, hence π_s^θ serves as a valid annealing path between $N(0, \mathrm{I})$ and π.

G Experimental Details

G.1 Target distributions

GMM-d We take the 40-mode Gaussian mixture model (GMM-2) in 2 dimensions from Midgley et al. (2023) where to extend this distribution to higher dimensions d, we extend the means with zero padding to a vector in $\mathbb{R}^d$ and keep the covariances as the identity matrix but now within $\mathbb{R}^d$, to allow for the visualisation of samples in higher dimensions. Following previous work (Akhound-Sadegh et al., 2024; Phillips et al., 2024), we also scale the distribution GMM-d by a factor of 40 to ensure the modes are contained within the range $[-1, 1]$ for the Diff-GePT experiments, where we use the same scaling for the Linear-PT baseline to ensure a fair comparison.

G.2 Algorithmic Details for Flow-GePT Experiments

Annealing path We use the linear annealing path, $\pi_n \propto \pi_0^{1-\frac{n}{N}} \pi^{\frac{n}{N}}$, with $N = 5$ (for a total of 6 chains) and a uniform discretisation. We do not fine-tune the annealing schedule here as our goal is to demonstrate that Flow-GePT can enhance performance even with a suboptimal choice.

Local communication kernels Flow-GePT and Linear-PT each use a single step of the Metropolis-adjusted Langevin algorithm (MALA) (Roberts & Tweedie, 1996) for the local exploration kernels K_n.

Training We use 1000 independent copies of Flow-GePT (i.e., there are 6000 total samples at any point in time) with normalising flows parametrised by 20 NVP layers Dinh et al. (2017) with 16 hidden units each and initialised at the identity function.

We then train for 4000 iterations with all samples initialised from the reference distribution. Each iteration consists of five Flow-GePT steps using the current flows, after which the samples from the fifth step are used to update the flows via the loss in Equation (20).

We train using a single batch using all samples and optimize with Adam (Kingma & Ba, 2014) using a learning rate of 10^{-3} and global norm clipping at 1.0.

G.3 Algorithmic Details for Diff-GePT Experiments

Compute-normalised round trip rate In a fully parallelised implementation of PT and m-step Diff-GePT, a hardware-agnostic measure of the effective computational cost required to generate a single sample from the target distribution π is the number of function evaluations of the annealing path π_n (or its score) that a single machine needs to implement, as this constitutes the sequential computation of the algorithm. For example, each step of PT requires each machine to compute two evaluations of π_n and m-step Diff-GePT requires $2 + m$ evaluations. Hence, we define the *compute-normalised round trip rate* to be the round trips divided by the total number of function evaluations of π_n that a single machine implements when considering a fully parallelised implementation. For example, given a run of m-step Diff-GePT with T steps which generates r round trips, the compute-normalised round trip rate equals $\frac{r}{(2+m)T}$.

Annealing schedule initialisation For Diff-GePT and Diff-PT, we initialise the annealing schedule using the following scheme:

$$s_n = 1 - \left(s_{\text{max}}^{\frac{1}{\rho}} + \frac{n}{N-1}(s_{\text{min}}^{\frac{1}{\rho}} - s_{\text{max}}^{\frac{1}{\rho}}) \right)^{\rho} \quad \text{for } n < N, \quad s_N = 1 \tag{21}$$

based off of equation 5 in Karras et al. (2022), where we take $s_{\text{min}} = 1 \times 10^{-6}, s_{\text{max}} = 1$ and $\rho = 3$. We note that the modification of the scheme from Karras et al. (2022) is due to the reversal of the usual diffusion time convention to align with the indexing convention of the PT literature. For Linear-PT, we initialise the annealing schedule to be a uniform discretisation of $[0, 1]$.

Local communication kernels For each method, we define the local exploration kernels K_n as using 5 Hamiltonian Monte Carlo (Neal, 2012) steps with 2 leapfrog steps and a step size of 0.03 each - i.e. we use the same local exploration kernel for each component of PT/GePT.

Annealing schedule tuning The choice of annealing schedule $s_0 = 0 < s_1 \ldots < s_N = 1$ is empirically very important for the performance of PT and GePT. In order to tune this component at sampling time, for a fair comparison between each method, we use 10 rounds of the tuning algorithm proposed in Syed et al. (2021) with 100 burn in steps and 500 samples.

Training To train the learned diffusion path, we run 32 independent copies of PT with $(\pi_s^\theta)_{s\in[0,1]}$ as an annealing path, where we initialise states along the annealing path at the reference distribution and take 200 steps of burn in. Afterwards, we run the outer loop for M steps which takes 16 steps of the PT Markov chain to give 512 samples at the target distribution π. We then run the inner loop where we optimise the score-matching objective with the above batch of 512 samples for 20 steps, where we sample the times s according to the continuous version of the annealing schedule in Equation (21). Further, we retune the annealing schedule used during sampling every 200 outer steps with the current collection of samples.

For the optimiser, we use Adam with default hyper-parameters and a learning rate of 5×10^{-4}, global norm clipping at 2.0 and EMA with decay parameter of 0.9. We present further training hyper-parameters in Table 2.

Table 2: Training hyper-parameters for the learned Diff-GePT models targeting GMM-d for $d = 2, 10, 50$.

Model	Number of Chains	M
GMM-2	10	3000
GMM-10	30	6000
GMM-50	60	10000

G.4 Further Results for Diff-GePT Experiments

Round trip rate Table 3 presents the round trip rate derived from the results presented in Table 1.

Table 3: Corresponding round trip rate of the results presented in Table 1.

Methods	GMM-2			GMM-10			GMM-50		
	10	30	60	10	30	60	10	30	60
1-Diff-GePT	0.110	0.117	0.117	0.0160	0.0360	0.0395	0.00	0.00956	0.0146
2-Diff-GePT	0.133	0.138	0.132	0.0230	0.0504	0.0492	0.0018	0.0170	0.0214
5-Diff-GePT	**0.166**	**0.159**	**0.154**	**0.0553**	**0.0713**	**0.0504**	**0.0098**	**0.0265**	**0.0276**
Diff-PT	0.106	0.121	0.127	0.00678	0.0183	0.0263	0.00	0.00216	0.00406
Linear-PT	0.117	0.131	0.139	0.00660	0.0189	0.0259	0.00	0.00212	0.00390

Normalisation constant estimation From the samples generated by m-step Diff-GePT, Diff-PT and Linear-PT in Table 1, we take 5 consecutive chunks of 2000 samples to estimate the log-normalisation constant of the corresponding GMM-d distribution 5 times via the stepping stone estimator. We present the following distributions of log-normalisation constants estimators in Figure 3.

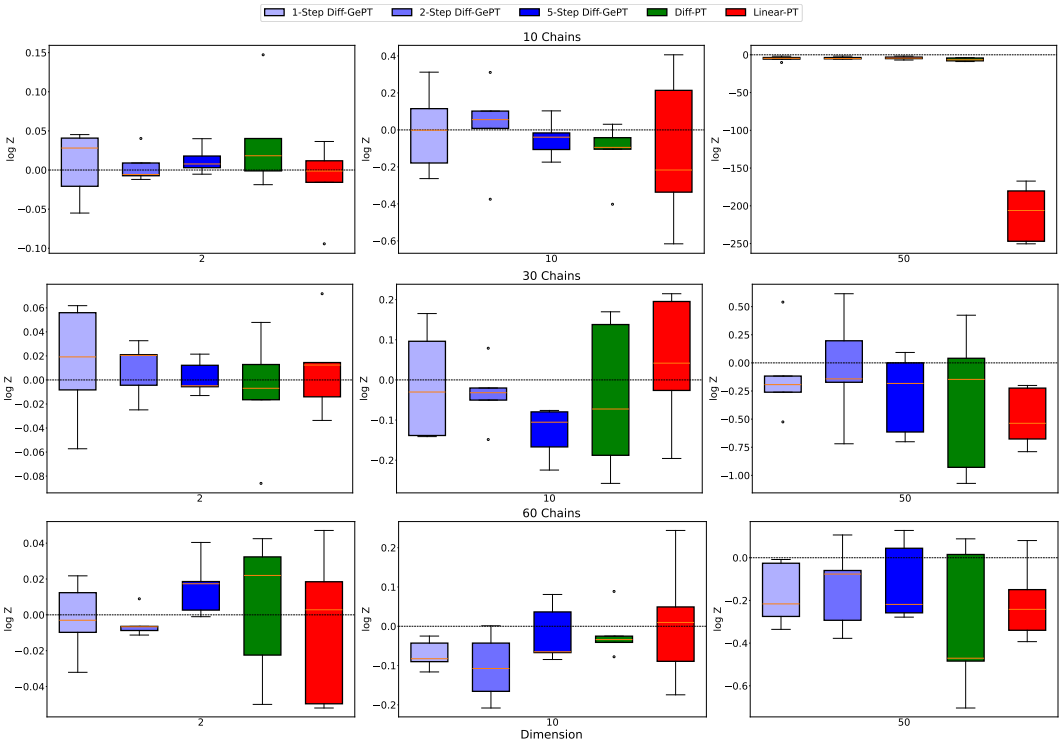

Figure 3: Estimates of the log-normalisation constant of GMM-d for $d = 2, 10, 50$ by m-step Diff-GePT ($m = 1, 2, 5$), Diff-PT and Linear-PT using 2000 samples. Each box consists of 5 estimates. The black dotted line denotes the ground-true log-normalisation constant.

W2 distance We follow the same setting as above to compute the Wasserstein-2 (W2) distance with respect to 2000 independent samples from the corresponding GMM-d distributions. We present the following distributions of W2 distances in Figure 4.

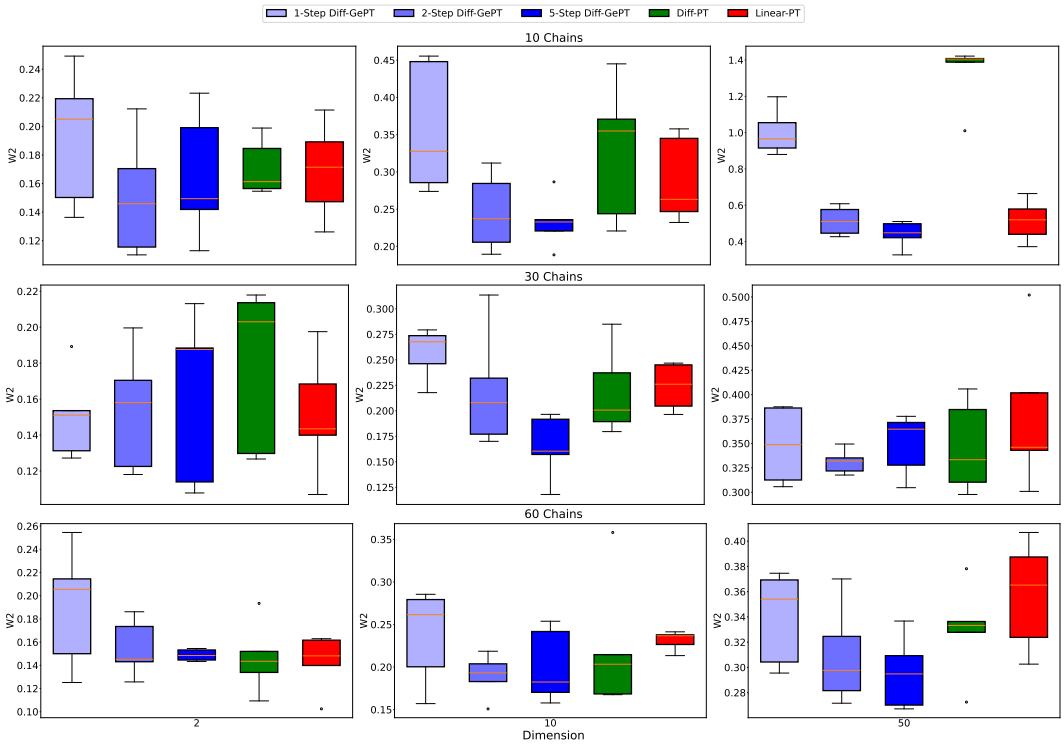

Figure 4: W2 distance between samples from GMM-d for $d = 2, 10, 50$ and m-step Diff-GePT ($m = 1, 2, 5$), Diff-PT and Linear-PT using 2000 samples. Each box consists of 5 estimates.