DSA4199 Honours Project in Data Science & Analytics

Differentiable Samplers: Time-Embedded Annealed Flow Transport

Steven Lester Seah Wei Han

8 March 2024

Summary

This thesis covers the Sequential Monte Carlo (SMC) sampling algorithm [MDJ06] and its variants in the context of generating weighted samples for the estimation of expectations and normalizing constants of unnormalized probability density functions. In particular, we focus on two novel variants in the literature that incorporate normalizing flows to the SMC algorithm, namely Annealed Flow Transport (AFT) [AMD21] and Continual Repeated Annealed Flow Transport (CRAFT) [MARD22]. Importance sampling and normalizing flows are briefly introduced as part of the background in section 2. We discuss the details of these sampling algorithms, as well as verify that the algorithms indeed produce weighted samples that are able to approximate expectations of the target distribution, as well as estimate normalizing constants in section 3. Section 4 introduces the contributions made by the thesis, which are Time-Embedded AFT (TE-AFT), Time-Embedded CRAFT (TE-CRAFT), and an adaptive variant of TE-AFT. TE-AFT and TE-CRAFT are modifications of AFT and CRAFT respectively, where the normalizing flows used in each iteration are replaced by a single normalizing flow that takes information about the current and previous iterations' intermediate distributions as input. Adaptive TE-AFT modifies the TE-AFT algorithm to adaptively select the intermediate distributions based on the current iteration's set of weighted samples and parameters of the normalizing flow, as a TE-AFT analogue of the adaptive SMC proposed by [ZJA16]. We introduce a modification of the conditional effective sample size (CESS) metric (used in adaptive SMC to quantify the similarity between intermediate distributions) into $\text{CESS}^{(f)}$ in order to account for the flow transport in adaptive TE-AFT. Finally, section 5 covers simulation experiments conducted to compare the proposed algorithms to SMC, AFT and CRAFT, and interprets the results of these experiments. Implementation of the Real-NVP normalizing flow [DSDB16], time-embedded flows, TE-AFT, TE-CRAFT, and adaptive TE-AFT, as well as the implementation of the training of Real-NVP on the two-moons example (see figure 1) and time-embedded flow example (see figure 3) are originally made.

Contents

1 Introduction

Monte Carlo methods Monte Carlo methods are a broad collection of algorithms that offer a computational approach to solving physical and mathematical problems through the use of numerical simulations to approximate solutions. These problems are primarily of the optimization, numerical integration, and sampling (from a probability distribution) variety [KBTB14]. Some simple and basic applications of Monte Carlo involve expressing the desired integral to be solved in the form of an expectation of some random variable, which, invoking the law of large numbers, can then be estimated by the empirical mean of a large number of independent samples of this random variable. A popular toy example demonstrating this concept is the approximation of $\frac{\pi}{4}$ through the numerical integration of the top-left quadrant of the unit circle, using samples uniformly drawn from the unit square (see [KW09]).

Markov Chain Monte Carlo As the complexity of problems (and their corresponding probabilistic interpretations) increase, there quickly comes a point where the distributions involved become too complex to directly sample from. Fortunately, by constructing a Markov transition kernel K that keeps the desired distribution, say π, invariant, we are able to draw samples of π using Markov Chain Monte Carlo (MCMC). By first generating a sample x_0 from an initial distribution π_0 and then iteratively sampling x_t from $K(x_{t-1}, \cdot)$ for $t \in \{1, 2, ..., n\}$ for some n, we can generate a Markov chain $(x_t)_{0 \geq t \geq n}$. After discarding the initial b states as "burn-in", the remainder $T = n - b$ states form an approximation of π,

$$\hat{\pi} = T^{-1} \sum_{t=1}^{T} \delta_{x_t}, \tag{1.1}$$

which converges to π as $T \to \infty$ [Num02], where δ_x is the Dirac probability distribution that assigns all of its probability mass at x. This concept is central to MCMC algorithms, which are designed to construct such Markov transition kernels and draw from them to ultimately produce samples of a desired distribution (for example, the Metropolis-Hastings [Has70] and Hamiltonian Monte Carlo [N+11] algorithms). MCMC has direct applications in a wide range of domains, such as statistical Physics [Kra06] [NB99], Chemistry [FS02], and the social sciences [Gil08]. Within the practice of Statistics, MCMC features prominently in Bayesian inference, where its ability to generate samples with only knowledge of a distribution's unnormalized density allows it to play a big role in conducting inference on posterior distributions.

Bayesian inference In Bayesian inference, parameters of interest are treated as random variables. Initial or prior knowledge about the values of these parameters are expressed as the prior distribution. Through the use of Bayes' theorem, the product of the likelihood of observed data and the density of the prior distribution can be taken to obtain the density of the parameters given the observed data, known as the posterior distribution, up to some normalizing constant. However, this normalizing constant often presents itself as an intractable integral, which tends to become a major obstacle in performing Bayesian inference analytically. To conduct Bayesian

inference numerically, one would require a method to generate samples from the posterior distribution, of which we only know its density function up to some intractable normalizing constant. For many cases, MCMC algorithms are typically able to fulfill this role, but they also come with serious drawbacks; estimating the normalizing constant from MCMC runs alone is difficult, and it can be time-consuming to tune the Markov transition kernel in order to obtain acceptable efficiency [DHJW22].

Annealed Flow Transport Monte Carlo Many sampling algorithms have been proposed since the development of MCMC that are capable of sampling from progressively complex distributions and providing increasingly accurate estimates for the normalizing constant of the unnormalized densities of these distributions. Two particular algorithms have emerged recently that demonstrate potential in advancing the state of the art; namely, annealed flow transport (AFT) [AMD21] and continual repeated annealed flow transport (CRAFT) [MARD22], which we collectively refer to as annealed flow transport Monte Carlo. In this thesis, we describe and introduce annealed flow transport Monte Carlo, as well as investigate its performance in delivering accurate estimates for normalizing constants using numerical simulations.

We also propose variants of each of the algorithms that reduce the number of normalizing flows used down to one by encoding the schedule time and feeding it into the flow, which we refer to as time-embedded AFT (TE-AFT) and time-embedded CRAFT (TE-CRAFT). This makes the use of more intermediate distributions less prohibitive as the number of parameters no longer scale with the number of intermediate distributions, and has the potential to facilitate and improve training of the flow parameters. Compared to CRAFT, TE-CRAFT seems to be more robust to poor choices of intermediate distributions, which could make the tuning of the choice of intermediate distributions easier. An adaptive variant of TE-AFT that performs on-line selection of intermediate distributions is also proposed, that empirically shows to able to make better choices of intermediate distributions than its AFT analogue. Our code implementations for the experiments conducted in this thesis are available publicly[1].

2 Background

2.1 Inference for latent variable models

Consider a latent variable model that depends on parameters θ, with probability density p_θ. Then we can get the likelihood of data $\mathbf{y}$ by taking

$$p_\theta(\mathbf{y}) = \int_E p_\theta(\mathbf{y}, x) dx, \tag{2.1}$$

where $x \in E$ is a latent variable over some (possibly multidimensional) support E. Now consider the case where $\mathbf{y}$ is known and fixed, and where the integral in (2.1) is intractable (which is often

[1] https://github.com/stevvseah/fyp_differentiable_samplers

the case for non-trivial problems). This makes it difficult to perform likelihood-based inference, since the likelihood $p_\theta(\mathbf{y})$ itself is no longer practical to compute. Fortunately, if we define an unnormalized density function of the posterior distribution by taking

$$f_{\cdot|\mathbf{y}}(x) = p_\theta(\mathbf{y}, x) = p_\theta(\mathbf{y}|x)p_\theta(x), \tag{2.2}$$

then it is clear that we would be able to retrieve the likelihood $p_\theta(\mathbf{y})$ as the normalizing constant of this density. Methods to produce good estimates of the normalizing constant of a given unnormalized density are thus imperative to allow likelihood-based inference to carry on, and will be in the focus of this thesis.

2.2 Distribution approximation

Let the target distribution we are interested in be π, over the (possibly multidimensional) support E. We define a particle approximation $\hat{\pi}$ of π as

$$\hat{\pi} = \sum_{i=1}^{N} W^{(i)} \delta_{X^{(i)}}, \tag{2.3}$$

where N is a large positive integer, $W^{(i)} \in [0, 1]$ and $X^{(i)} \in E$ for $i \in \{1, 2, ..., N\}$. We refer to each $W^{(i)}$ as weights and each $X^{(i)}$ as particles (or the position of the i^{th} particle). Put together, we refer to the collection $\{W^{(i)}, X^{(i)}\}_{i=1}^{N}$ as weighted particles. With a good particle approximation of π, we will be able to approximate the expectation of any arbitrary statistic $g(X)$ for $X \sim \pi$ by taking

$$\begin{aligned} \mathbb{E}_\pi[g(X)] &= \int_E g(x)\pi(x)dx \\ &\approx \int_E g(x) \sum_{i=1}^{N} W^{(i)} \delta_{X^{(i)}}(x)dx \\ &= \sum_{i=1}^{N} W^{(i)} g(X^{(i)}). \end{aligned} \tag{2.4}$$

The quality of the particle approximation relies on the weighted particles, and thus, the discussion on sampling algorithms in this thesis will also focus on obtaining weighted particles that can produce good particle approximations.

For the rest of this thesis, we shall denote π to be the target distribution (that we wish to obtain a particle approximation of) with unnormalized density $f : E \mapsto \mathbb{R}$ and normalizing constant Z.

2.3 Importance sampling

One of the most immediate ways to find a collection of weighted particles that satisfies (2.3) is by (self-normalized) importance sampling (IS). IS directly uses samples from a proposal distribution

π_0 as the particles $\{X^{(i)}\}_{i=1}^N$, and assigns the weights to be

$$W^{(i)} = \frac{f(X^{(i)})}{\pi_0(X^{(i)})} \Bigg/ \sum_{j=1}^{N} \left[\frac{f(X^{(j)})}{\pi_0(X^{(j)})}\right]. \tag{2.5}$$

Indeed, we have that the approximation $\sum_{i=1}^{N} W^{(i)}\delta_{X^{(i)}}$ constructed this way converges to π as $N \to \infty$, under some assumptions on the ratio of densities $\pi(x)/\pi_0(x)$ for $x \in E$ [Owe13], satisfying (2.3) for large N.

Obtaining a good estimate $\hat{Z}$ of Z by IS is also quite simple. Observe that

$$Z = \int_E f(x)dx = \int_E \frac{f(x)}{\pi_0(x)}\pi_0(x)dx = E_{\pi_0}\left[\frac{f(X)}{\pi_0(X)}\right]. \tag{2.6}$$

Then by the Law of Large Numbers, we have that the estimate

$$\hat{Z} = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X^{(i)})}{\pi_0(X^{(i)})}, \tag{2.7}$$

for $X^{(i)} \sim \pi_0$, $i \in \{1, 2, ..., N\}$, converges to Z as $N \to \infty$, where we often label

$$w(X) = \frac{f(X)}{\pi_0(X)} \tag{2.8}$$

as the unnormalized importance weight of $X \sim \pi_0$.

Typically, π_0 is chosen to be a distribution that is computationally cheap and simple to sample from, such as the Gaussian distribution. However, when π and π_0 differ too drastically, this approximation can be extremely poor [APSAS17] [CD18]. In such cases, the variance of the importance weights tends to be large, as most of the weights will be concentrated in a small number of particles. The large variance introduced from the weights causes estimates of expectations with respect to π in the form of (2.4) to become noisy and unreliable – in fact, the variance of these estimates is approximately proportional to $1 + \mathrm{Var}_{\pi_0}\{w(X)\}$ [Liu01]. This drawback is significantly more pronounced in high-dimensional settings, where it is already usually difficult to make a good choice of π_0 that does not differ too much from π, let alone for a non-trivial π [NLS+19].

Extensions of IS, like sequential importance sampling [HM69] [RC99] and annealed importance sampling [Nea01] [Jar97] were proposed to overcome these flaws through a strategy of sequentially estimating a sequence of "bridging" distributions between π_0 and π. This strategy involves estimating the next bridging distribution using the weighted particles that form the particle approximation of the current bridging distribution, until weighted particles that can form a good particle approximation of π are obtained. This strategy eventually developed into the Sequential Monte Carlo (SMC) algorithm [MDJ06], which we will describe in detail in a later section.

2.4 Normalizing Flows

Normalizing flows (NFs) were introduced in [TT13] generally as a composition of a variable number of simple maps. Its introduction was quickly followed by numerous developments in its application in density estimation [Ada13], image modelling and inference [DKB14], which demonstrated its expressiveness and potential in a wide range of applications. Similar to IS, NFs have the ability to generate samples from a complicated distribution using samples from a much more simpler distribution. However, unlike IS, it does not rely on weight-adjustments to accomplish this. Instead, NFs learn a diffeomorphism T that directly transports the particles $\{X^{(i)}\}_{i=1}^N$ sampled from π_0 to $\{T(X^{(i)})\}_{i=1}^N$ such that

$$\pi = T_{\#}\pi_0, \tag{2.9}$$

where $T_{\#}\pi_0$ is the distribution of $T(X)$ for $X \sim \pi_0$. [PNR+21] provides a constructive proof of the existence of such a T for any pair of well-behaved π and π_0. See also [BKM05] for a more formal argument.

For our purposes, we only require that the weighted particles $\{\frac{1}{N}, T(X^{(i)})\}_{i=1}^N$ can provide a good approximation of π, i.e.

$$\pi \approx \sum_{i=1}^N \frac{1}{N}\delta_{T(X^{(i)})}. \tag{2.10}$$

Typically, T is parameterized by neural networks. For T of this sort, there exist two popular approaches to learn T – the maximum likelihood approach, and the variational inference approach. Note that in this section, while we assume T to depend on parameters, say θ, we choose to write T as T_θ in the interest of brevity.

2.4.1 Maximum likelihood

Given samples $\{Y^{(i)}\}_{i=1}^N$ from an unknown target distribution π, it is commonly preferred to fit T by maximum likelihood. Using the invertibility of T, we can expose the density of π in terms of the density of a known proposal distribution π_0 as

$$\pi(y) = T_{\#}\pi_0(y) = \pi_0(T^{-1}(y))|\det J_{T^{-1}}(y)|, \tag{2.11}$$

where $J_{T^{-1}}$ is the Jacobian of T^{-1}. Then we can obtain the likelihood of the sample $\{Y^{(i)}\}_{i=1}^N$ to be

$$p\left(\{Y^{(i)}\}_{i=1}^N\right) = \prod_{i=1}^N \pi_0(T^{-1}(Y^{(i)}))|\det J_{T^{-1}}(Y^{(i)})|. \tag{2.12}$$

The parameters θ of T can then be learned to maximize the likelihood objective expressed in (2.12), typically using iterative gradient-based techniques.

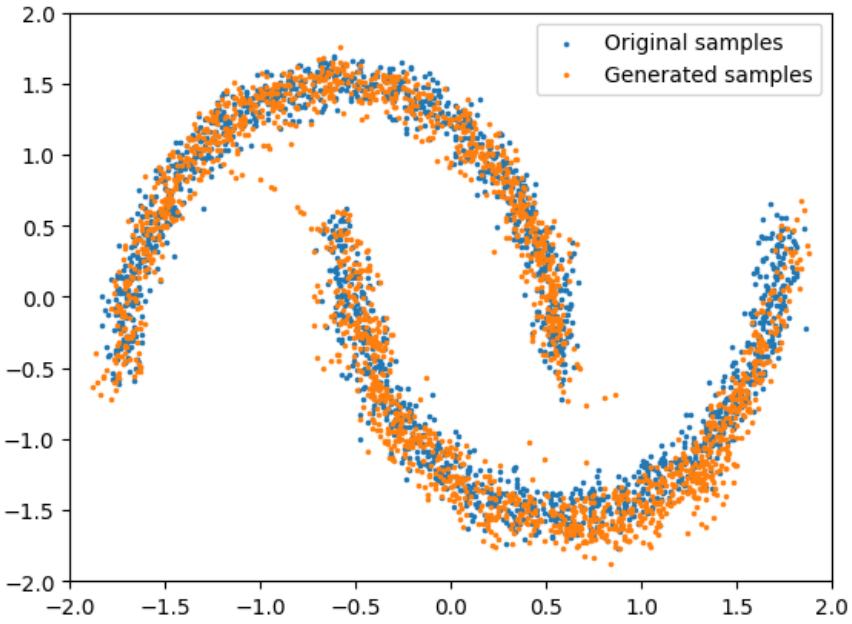

Figure 1: Samples generated from a trained Real-NVP model plotted over the original training samples of the Two Moons dataset obtained from the Scikit-Learn package.

In practice, this optimization problem is often framed as a minimization problem of the negative logarithm of the objective in (2.12), i.e. $\min_\theta \mathcal{L}(\theta)$, where

$$\begin{aligned}\mathcal{L}(\theta) &= -\frac{1}{N}\log p\left(\{Y^{(i)}\}_{i=1}^N\right) \\ &= -\frac{1}{N}\sum_{i=1}^N \left\{\log \pi_0(T^{-1}(Y^{(i)})) + \log\left|\det J_{T^{-1}}(Y^{(i)})\right|\right\}.\end{aligned} \tag{2.13}$$

Minimizing the above objective can also be interpreted as minimizing a Monte Carlo approximation of the forward Kullbeck-Leibler (KL) divergence between π and $T_\#\pi_0$ [PNR+21]. Observe that

$$\begin{aligned}D_{KL}(\pi||T_\#\pi_0) &= -\mathbb{E}_\pi[\log T_\#\pi_0(y)] + \text{const.} \\ &= -\mathbb{E}_\pi[\log \pi_0(T^{-1}(y)) + \log\left|\det J_{T^{-1}}(y)\right|] + \text{const.} \\ &\approx -\frac{1}{N}\sum_{i=1}^N \left\{\log \pi_0(T^{-1}(Y^{(i)})) + \log\left|\det J_{T^{-1}}(Y^{(i)})\right|\right\} + \text{const.},\end{aligned} \tag{2.14}$$

where the last line is essentially equivalent to the objective in (2.13) up to an additive constant.

We demonstrate the effectiveness of learning T by maximum likelihood with a toy example, where we implemented the real-valued non-volume-preserving (Real-NVP) flow [DSDB16] model

in Python using the Jax [BFH+18], Flax [HLO+23] and Optax [DBB+20] packages, and trained it on the Two Moons dataset obtained from the Scikit-Learn [PVG+11] package (see Figure 1).

2.4.2 Variational inference

Suppose that we have the unnormalized density function f of the target distribution π up to some (possibly intractable) normalizing constant Z. Then we are able to perform variational inference to learn the parameters θ of the transport map T by minimizing the reverse KL divergence between π and $T_{\#}\pi_0$ [RM15]. Observe that

$$\begin{aligned} D_{KL}(T_{\#}\pi_0||\pi) &= \mathbb{E}_{T_{\#}\pi_0}[\log T_{\#}\pi_0(y) - \log \pi(y)] \\ &= \mathbb{E}_{\pi_0}[\log \pi_0(x) - \log|\det J_T(x)| - \log \pi(T(x))] \\ &= \mathbb{E}_{\pi_0}[\log \pi_0(x) - \log|\det J_T(x)| - \log f(T(x))] + \log Z, \end{aligned} \tag{2.15}$$

where a change of variable was made in the variable the expectation was taken with respect to, from $y \sim T_{\#}\pi_0$, to $x = T^{-1}(y) \sim \pi_0$. Since Z does not depend on T's parameters θ, θ can instead be learned by minimizing the objective

$$\mathcal{L}(\theta) = \mathbb{E}_{\pi_0}[\log \pi_0(x) - \log|\det J_T(x)| - \log f(T(x))], \tag{2.16}$$

eliminating the need for Z to be computed to learn θ. Similar to the maximum likelihood approach, θ can be optimized using gradient-based techniques. In our context, we will focus on the variational inference approach to learning T, as we always assume to have the unnormalized density of the target distribution π, making this approach very appropriate.

2.4.3 Computing the Jacobian

In both approaches to training NFs, the determinant of the Jacobian of T appears as an important component of the objective function. Generally, for a flow T with d-dimensional inputs and outputs, the computation of this determinant has a time complexity of $\mathcal{O}(d^3)$, even when using d passes of either the forward-mode or reverse-mode automatic differentiation. This is too costly to pursue in a high-dimensional setting. Fortunately, the architecture of the neural network that parameterizes T can be cleverly designed to allow the determinant of the Jacobian to be computed in as quickly as $\mathcal{O}(d)$ time [PNR+21].

2.4.4 Variants and extensions

As diffeomorphisms, NFs face a major challenge due to their preservation of the topological properties of their input. [CCDD20] showed that Bi-Lipschitz constraints are induced when a unimodal π_0 is mapped onto a well-separated multi-modal π. A simple demonstration was implemented by [WKN20], where a mapping of a unimodal Gaussian base distribution onto a bimodal distribution with affine coupling layers still maintained a connection between the density modes of the bimodal distribution. Several variants of flow models have been proposed to overcome this, like by boosting the expressiveness of the coupling layer [DBMP19], usage of real-and-discrete mixtures of flows [DSDLP19], or by augmenting the input space [DDT19] [HDC20].

2.5 Sequential Monte Carlo samplers

SMC originated from the signal processing literature as particle filters, which were designed to perform online inference in non-linear state space models [Kit93] [GSS93] [SMJ92] [CMR09]. As attention shifted to applying these methods for approximate inference in more complex models, the strategy of propagating samples through a sequence of intermediate distributions quickly made its way into sampling algorithms as a remedy for the deficiencies of IS, in algorithms such as sequential importance sampling [RC99], iterated batch importance sampling [Cho02], and annealed importance sampling [Nea01]. Eventually, SMC samplers [MDJ06] were proposed, extending these ideas while improving accuracy and amenability to parallel computing [DHJW22].

The SMC sampler is a sampling algorithm that propagates particles sampled from a proposal distribution π_0 through a sequence of bridging distributions $\{\pi_t\}_{t=1}^n$, where $\pi_n = \pi$, to ultimately produce weighted particles $\{W_n^{(i)}, X_n^{(i)}\}_{i=1}^N$ that can form the particle approximation

$$\hat{\pi}_n = \sum_{i=1}^{N} W_n^{(i)} \delta_{X_n^{(i)}}, \tag{2.17}$$

such that $\hat{\pi}_n \approx \pi_n$.

2.5.1 Description of algorithm

After the initialization of the particles to be propagated, the SMC sampler algorithm consists of a main loop of subroutines that we will describe as the importance sampling step, the conditional resampling step, and the mutation step (which are carried out in this order as well). In this section, we will go describe each of these components in further detail. We describe the full algorithm in algorithm 1.

Initialization Initialize the weighted particles by drawing N samples $\{X_0^{(i)}\}_{i=1}^N$ from π_0, each with corresponding weights $\{W_0^{(i)}\}_{i=1}^N$, where $W_0^{(i)} = \frac{1}{N}$ for all $i \in \{1, 2, ..., N\}$, so that we have

$$\pi_0 \approx \hat{\pi}_0 = \sum_{i=1}^{N} W_0^{(i)} \delta_{X_0^{(i)}}. \tag{2.18}$$

Importance sampling step The importance sampling step serves to update the weights of the weighted particles, that is, obtain $W_t^{(i)}$ from $W_{t-1}^{(i)}$ for each $i \in \{1, 2, ..., N\}$. At the start of an importance sampling step, we have the weighted particles $\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N$ that form the particle approximation

$$\hat{\pi}_{t-1} = \sum_{i=1}^{N} W_{t-1}^{(i)} \delta_{X_{t-1}^{(i)}}, \tag{2.19}$$

Algorithm 1: Sequential Monte Carlo Sampler

Input: Number of particles N, bridging distributions $\{\pi_t\}_{t=1}^n$ with unnormalized density functions $\{f_t\}_{t=1}^n$, and resampling threshold $S \in (1, N)$.

Output: Approximations $(\hat{\pi}_n, \hat{Z}_n)$ to (π_n, Z_n).

Initialization step:

Initialize weighted particles $\{W_0^{(i)}, X_0^{(i)}\}_{i=1}^N$, where $X_0^{(i)} \sim \pi_0$ and $W_0^{(i)} = \frac{1}{N}$ for $i \in \{1, 2, ..., N\}$.

Initialize evidence estimate $\hat{Z}_0 = Z_0$ from proposal distribution.

for $t = 1, 2, ..., n$ **do**

Importance sampling step:

Set $\tilde{w}_t^{(i)} = \frac{f_t(X_{t-1}^{(i)})}{f_{t-1}(X_{t-1}^{(i)})}$ for $i = 1, 2, ..., N$.

Set $W_t^{(i)} = \frac{W_{t-1}^{(i)} \tilde{w}_t^{(i)}}{\sum_{j=1}^N W_{t-1}^{(j)} \tilde{w}_t^{(j)}}$ for $i = 1, 2, ..., N$.

Set $\hat{Z}_t = \hat{Z}_{t-1} \sum_{i=1}^N W_{t-1}^{(i)} \tilde{w}_t^{(i)}$.

Conditional resampling step:

Compute $ESS_t = \left\{ \sum_{i=1}^N \left(W_t^{(i)} \right)^2 \right\}^{-1}$.

if $ESS_t < S$ **then**

Resample $X_{t-1}^{(1)}, ..., X_{t-1}^{(N)} \sim \text{Multinomial}(\{X_{t-1}^{(i)}\}_{i=1}^N; \{W_t^{(i)}\}_{i=1}^N)$.

Set $W_t^{(1)} = W_t^{(2)} = ... = W_t^{(N)} = \frac{1}{N}$.

end

Mutation step:

Draw $X_t^{(i)} \sim K_t(X_{t-1}^{(i)}, \cdot)$ for $i = 1, 2, ..., N$.

end

and we wish to get $\{W_t^{(i)}\}_{i=1}^N$ such that

$$\pi_t \approx \sum_{i=1}^{N} W_t^{(i)} \delta_{X_{t-1}^{(i)}}. \tag{2.20}$$

To obtain the updated weights, we first re-weight the particles by their importance weights to the current bridging distribution, i.e. we compute the incremental weight

$$\tilde{w}_t^{(i)} = \frac{f_t(X_{t-1}^{(i)})}{f_{t-1}(X_{t-1}^{(i)})}, \tag{2.21}$$

where f_t and f_{t-1} are the unnormalized densities of π_t and π_{t-1} with normalizing constants Z_t and Z_{t-1} respectively. Then we assign the weights

$$W_t^{(i)} = \frac{W_{t-1}^{(i)} \tilde{w}_t^{(i)}}{\sum_{j=1}^{N} W_{t-1}^{(j)} \tilde{w}_t^{(j)}}, \tag{2.22}$$

for each particle $i \in \{1, 2, ..., N\}$, for iteration $t \in \{1, 2, ..., n\}$.

Conditional resampling step In this step, we replace the current set of weighted particles $\{W_t^{(i)}, X_t^{(i)}\}_{i=1}^N$ with $\{W_t^{*(i)}, X_t^{*(i)}\}_{i=1}^N$, where $W_t^{*(i)} = \frac{1}{N}$, and $X_t^{*(i)}$ is randomly drawn from $\{X_t^{(j)}\}_{j=1}^N$ by a multinomial distribution with weights $\{W_t^{(j)}\}_{j=1}^N$, for all particles $i \in \{1, 2, ..., N\}$ in iteration $t \in \{1, 2, ..., n\}$ of the main loop of SMC. We refer to this as resampling. Resampling eliminates particles with small weights, and generate copies of particles with large weights, producing a set of equally weighted particles that mostly populate high density regions of the current bridging distribution. This helps to reduce the variance of the weights, which is crucial, as this variance can cause a degeneracy of the particle approximation of the current bridging distribution if it is allowed to become too large [MARD22]. However, resampling also introduces additional variance from the multinomial sampling. To mitigate this effect, we only run the resampling procedure if the degeneracy of the particle approximation exceeds a certain threshold. To quantify the extent of degeneracy, we use the effective sample size (ESS) criterion $\{\sum_{i=1}^N (W_t^{(i)})^2\}^{-1}$ [LC98], and trigger the resampling subroutine only when ESS falls below a chosen threshold $S \in (1, N)$. The ESS is large when the variance of the weights are small, up to a maximum of N when all particles hold equal weight, and small when the variance of the weights are large, down to a minimum of 1 when only one particle holds all of the weight. Other resampling schemes have been proposed that introduce less variance, such as stratified resampling [Kit96] and residual resampling [LC98], but in this thesis we will focus on using the simple multinomial scheme described here.

Mutation step In the mutation step, we perturb the particles by sampling $X_t^{(i)} \sim K_t(X_{t-1}^{(i)}, \cdot)$ for each $i \in \{1, 2, ..., N\}$, where K_t is a MCMC kernel that leaves the current bridging distribution π_t invariant, for iteration t. Not only does this adjust the distribution of the particles to

become more similar to that of π_t, but it also helps to spread out overlapping particles if resampling has been triggered immediately before this mutation step, reducing the degeneracy of the approximation $\hat{\pi}_t$ stemming from the presence of too many identical particles [MARD22]. We will show in the next section that the weights $\{W_t^{(i)}\}_{i=1}^N$ sufficiently account for the mutation step, preserving the approximation

$$\pi_t \approx \hat{\pi}_t = \sum_{i=1}^{N} W_t^{(i)} \delta_{X_t^{(i)}}. \tag{2.23}$$

2.5.2 Correctness of weights

At the end of each iteration t of the main loop of the algorithm (i.e. the end of the t^{th} mutation step), we assumed that the produced weighted particles $\{W_t^{(i)}, X_t^{(i)}\}_{i=1}^N$ approximated π_t as in (2.23). This section serves to verify the correctness of this assumption.

Suppose we have weighted particles $\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N$ such that the resulting particle approximation $\hat{\pi}_{t-1}$ approximates π_{t-1} well. Let K_t be an arbitrary 'forward' Markov kernel and L_{t-1} be an arbitrary 'backward' Markov kernel, for some $t \in \{1, 2, ..., n\}$.

Also, define the joint densities $\bar{\pi}^{(f)}$ and $\bar{\pi}^{(b)}$ to be

$$\bar{\pi}^{(f)}(x, y) = \pi_{t-1}(x) K_t(x, y), \tag{2.24}$$

and

$$\bar{\pi}^{(b)}(x, y) = \pi_t(y) L_{t-1}(y, x), \tag{2.25}$$

for $x \in X$ and $y \in Y$, where X and Y are the (possibly multidimensional) supports of π_{t-1} and π_t respectively. Note that by this construction, $\bar{\pi}^{(b)}(x, y)$ admits π_t as marginal on y, i.e.

$$\int_X \bar{\pi}^{(b)}(x, y) dx = \int_X \pi_t(y) L_{t-1}(y, x) dx = \pi_t(y). \tag{2.26}$$

By drawing particles $X_t^{(i)} \sim K_t(X_{t-1}^{(i)}, \cdot)$ for $i \in \{1, 2, ..., N\}$ in the current iteration's mutation step, we obtain a particle approximation of $\bar{\pi}^{(f)}$ from $\{W_{t-1}^{(i)}, X_{t-1}^{(i)}, X_t^{(i)}\}_{i=1}^N$, i.e.

$$\bar{\pi}^{(f)} \approx \sum_{i=1}^{N} W_{t-1}^{(i)} \delta_{X_{t-1}^{(i)}, X_t^{(i)}}. \tag{2.27}$$

Then we can approximate $\bar{\pi}^{(b)}$ by taking

$$\begin{aligned} \bar{\pi}^{(b)}(x, y) &= \frac{\bar{\pi}^{(b)}(x, y)}{\bar{\pi}^{(f)}(x, y)} \bar{\pi}^{(f)}(x, y) \\ &\approx \frac{\bar{\pi}^{(b)}(x, y)}{\bar{\pi}^{(f)}(x, y)} \sum_{i=1}^{N} W_{t-1}^{(i)} \delta_{X_{t-1}^{(i)}, X_t^{(i)}}(x, y) \\ &= \sum_{i=1}^{N} \frac{\bar{\pi}^{(b)}(X_{t-1}^{(i)}, X_t^{(i)})}{\bar{\pi}^{(f)}(X_{t-1}^{(i)}, X_t^{(i)})} W_{t-1}^{(i)} \delta_{X_{t-1}^{(i)}, X_t^{(i)}}(x, y). \end{aligned} \tag{2.28}$$

Using (2.26), we can obtain an approximation of π_t by marginalizing (2.28):

$$\begin{aligned}\pi_t(y) &\approx \int_X \sum_{i=1}^N \frac{\bar{\pi}^{(b)}(X_{t-1}^{(i)}, X_t^{(i)})}{\bar{\pi}^{(f)}(X_{t-1}^{(i)}, X_t^{(i)})} W_{t-1}^{(i)} \delta_{X_{t-1}^{(i)}, X_t^{(i)}}(x, y) dx \\ &= \sum_{i=1}^N \frac{\bar{\pi}^{(b)}(X_{t-1}^{(i)}, X_t^{(i)})}{\bar{\pi}^{(f)}(X_{t-1}^{(i)}, X_t^{(i)})} W_{t-1}^{(i)} \delta_{X_t^{(i)}}(y).\end{aligned} \tag{2.29}$$

Note that, so far, K_t and L_{t-1} have just been arbitrary Markov kernels. By choosing K_t to be a reversible Markov kernel that keeps π_t invariant, detailed balance allows us to have

$$\pi_t(y) K_t(y, x) = \pi_t(x) K_t(x, y). \tag{2.30}$$

We can then choose L_{t-1} to be the time reversal of K_t (as done in [Jar97] and [Nea01]), i.e.

$$L_{t-1}(y, x) = \frac{\pi_t(x) K_t(x, y)}{\pi_t(y)}. \tag{2.31}$$

This lets us simplify (2.29) to be

$$\pi_t \approx \sum_{i=1}^N \frac{\pi_t(X_{t-1}^{(i)})}{\pi_{t-1}(X_{t-1}^{(i)})} W_{t-1}^{(i)} \delta_{X_t^{(i)}}. \tag{2.32}$$

We thus have a particle approximation of π_t from $\{\frac{\pi_t(X_{t-1}^{(i)})}{\pi_{t-1}(X_{t-1}^{(i)})} W_{t-1}^{(i)}, X_t^{(i)}\}_{i=1}^N$.

Recall that the densities π_t and π_{t-1} tend to be intractable. As such, it is often necessary for us to approximate the weights of (2.32) as well. Defining f_t and f_{t-1} to be the unnormalized densities of π_t and π_{t-1} with normalizing constants Z_t and Z_{t-1} respectively, we get

$$\begin{aligned}\frac{\pi_t(X_{t-1}^{(i)})}{\pi_{t-1}(X_{t-1}^{(i)})} W_{t-1}^{(i)} &= W_{t-1}^{(i)} \frac{f_t(X_{t-1}^{(i)})}{f_{t-1}(X_{t-1}^{(i)})} \Big/ \left\{\frac{Z_t}{Z_{t-1}}\right\} \\ &\approx W_{t-1}^{(i)} \frac{f_t(X_{t-1}^{(i)})}{f_{t-1}(X_{t-1}^{(i)})} \Big/ \left\{\widehat{\frac{Z_t}{Z_{t-1}}}\right\} \\ &= \frac{W_{t-1}^{(i)} \tilde{w}_t^{(i)}}{\sum_{j=1}^N W_{t-1}^{(j)} \tilde{w}_t^{(j)}} \\ &= W_t^{(i)},\end{aligned} \tag{2.33}$$

as defined in (2.22), where we used $\tilde{w}_t^{(i)}$ as per (2.21) and the approximation for $\frac{Z_t}{Z_{t-1}}$ in (2.36).

Indeed, this definition of $W_t^{(i)}$ gets us

$$\sum_{i=1}^N W_t^{(i)} \delta_{X_t^{(i)}} \approx \sum_{i=1}^N \frac{\pi_t(X_{t-1}^{(i)})}{\pi_{t-1}(X_{t-1}^{(i)})} W_{t-1}^{(i)} \delta_{X_t^{(i)}} \approx \pi_t. \tag{2.34}$$

Note that more optimal choices of kernels K_t and L_t capable of minimizing variance from the mutation step exists; a more in-depth discussion is provided in [MDJ06].

2.5.3 Estimation of evidence

We can use the weights of the previous bridging distribution and the incremental weights obtained in the current iteration's importance sampling step to estimate the ratio of the evidences of the current bridging distribution and the previous bridging distribution. Observe that, for $t \in \{1, ..., n\}$,

$$\begin{aligned} \frac{Z_t}{Z_{t-1}} &= \int_E \frac{f_t(x)}{f_{t-1}(x)} \pi_{t-1}(x) dx = \mathbb{E}_{\pi_{t-1}} \left[\frac{f_t(X)}{f_{t-1}(X)} \right] \\ &\approx \sum_{i=1}^{N} W_{t-1}^{(i)} \frac{f_t(X_{t-1}^{(i)})}{f_{t-1}(X_{t-1}^{(i)})} = \sum_{i=1}^{N} W_{t-1}^{(i)} \tilde{w}_t^{(i)}. \end{aligned} \tag{2.35}$$

This gives us the estimate

$$\widehat{\frac{Z_t}{Z_{t-1}}} = \sum_{i=1}^{N} W_{t-1}^{(i)} \tilde{w}_t^{(i)}, \tag{2.36}$$

which allows us to get

$$\widehat{\frac{Z_n}{Z_0}} = \prod_{t=1}^{n} \left\{ \sum_{i=1}^{N} W_{t-1}^{(i)} \tilde{w}_t^{(i)} \right\}. \tag{2.37}$$

With a choice of proposal distribution π_0 that allows for exact computation of the normalizing constant Z_0, we can easily obtain the evidence estimate $\hat{Z}_n$ by multiplying Z_0 to the above estimate.

2.5.4 Variants and extensions

Due to its reliance on IS weights, SMC suffers flaws similar to that of IS. While the use of bridging distributions $\{\pi_t\}_{t=1}^n$ can help to reduce the discrepancies between each pair of distributions that IS weights are computed for, large variance can still be introduced into estimators when there are big enough dissimilarities between any pair of consecutive bridging distributions [AMD21]. This effect is amplified in high-dimensional settings, as KL divergences between π_0 and π tend to increase exponentially with the number of dimensions [DHJW22]. Fortunately, proper choice and tuning of $\{\pi_t\}_{t=1}^n$ can help to alleviate this. [JSDT11] proposes a method to adaptively select the bridging distributions based on controlling the decay of the ESS in every iteration of the SMC's main loop. Variants of this strategy have been proposed by [HR19] and [ZJA16], where alternative criteria are used in place of ESS.

Another potentially troublesome issue is that SMC approximations of expectations with respect to π are biased. [ADH10] proposes the use of SMC in the outer loop of a Particle MCMC algorithm in order to mitigate this.

A more recent approach incorporates deterministic moves to the stochastic transition kernels of SMC in the form of NFs in Annealed Flow Transport (AFT) Monte Carlo [AMD21] and Continual Repeated Annealed Flow Transport (CRAFT) Monte Carlo [MARD22]. These algorithms use NFs in each iteration of the SMC main loop to transport the particles closer to their current bridging distribution before taking IS weights, thereby closing the gap between bridging distributions and further reducing the variance of the IS weights. We will take a closer look at AFT and CRAFT in the next section.

3 Annealed Flow Transport Monte Carlo

The use of deterministic transformations to supplement the annealing approach has been proposed as early as [VJ11]. This concept is further explored by [ECMAW20] in the context of SMC, and has been applied in algorithms such as the nudged particle filter [AM20] and Gibbs flow [HDP21]. To extend these ideas and leverage on the vast and, in recent times, burgeoning literature of NFs [PNR+21], [AMD21] proposed in AFT Monte Carlo the use of NFs to perform deterministic transformations on the particles before IS weights are computed. Unfortunately, the training of the flows under AFT is hindered by its design of using only one pass through the bridging distributions – [MARD22] identifies the issue that the sample complexity of estimation can be higher than the finite number of available particles, and proposes an alternative training scheme in CRAFT to rectify this. Further details of AFT and CRAFT will be described in this section.

3.1 Annealed Flow Transport

AFT is a sampling algorithm that incorporates variational inference with NFs into SMC. The NFs are trained to learn a deterministic map that transports particles between bridging distributions by minimizing the KL divergence between the current and previous bridging distributions. The optimization objective is estimated using the weighted particles, as we will expound upon later. By combining NFs with SMC, AFT has the potential to overcome the limitations of each of its components; the use of NFs to adjust the particles in each iteration can reduce the variance of the incremental weights, while the MCMC steps can lighten the representational burden on NFs [MARD22].

3.1.1 Description of algorithm

AFT has a very similar structure to SMC – it has a main loop of subroutines after the initialization of the weighted particles that consists of the transport step, the importance sampling step, the conditional resampling step, and the mutation step (also carried out in this order). The full algorithm is provided in algorithm 2.

Initialization Initialize the weighted particles $\{W_0^{(i)}, X_0^{(i)}\}_{i=1}^N$, where $X_0^{(i)} \sim \pi_0$ and $W_0^{(i)} = \frac{1}{N}$ for $i \in \{1, 2, ..., N\}$, as well as the parameters $\{\phi_t\}_{t=1}^n$ for each of $\{T_t\}_{t=1}^n$. As with SMC, (2.18)

Algorithm 2: Annealed Flow Transport

Input: Number of particles N, bridging distributions $\{\pi_t\}_{t=1}^n$ with unnormalized density functions $\{f_t\}_{t=1}^n$, resampling threshold $S \in (1, N)$, and the initial flow parameters $\{\phi_t^{(0)}\}_{t=1}^n$.

Output: Approximations $(\hat{\pi}_n, \hat{Z}_n)$ to (π_n, Z_n).

Initialization step:

Initialize weighted particles $\{W_0^{(i)}, X_0^{(i)}\}_{i=1}^N$, where $X_0^{(i)} \sim \pi_0$ and $W_0^{(i)} = \frac{1}{N}$ for $i \in \{1, 2, ..., N\}$.

Initialize evidence estimate $\hat{Z}_0 = Z_0$ from proposal distribution.

for $t = 1, 2, ..., n$ **do**

Transport step:

Minimize $\tilde{\mathcal{L}}_t(\phi, \{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N)$ (from (3.8)) over ϕ by a gradient descent method (e.g. SGD), using initial flow parameters $\phi_t^{(0)}$.

Set the parameters of T_t to be the flow parameters obtained from the gradient descent method.

Set $\tilde{X}_{t-1}^{(i)} = T_t(X_{t-1}^{(i)})$ for $i = 1, 2, ..., N$.

Importance sampling step:

Set $\tilde{w}_t^{(i)} = \frac{f_t(\tilde{X}_{t-1}^{(i)}) |\det J_{T_t}(X_{t-1}^{(i)})|}{f_{t-1}(X_{t-1}^{(i)})}$ for $i = 1, 2, ..., N$.

Set $W_t^{(i)} = \frac{W_{t-1}^{(i)} \tilde{w}_t^{(i)}}{\sum_{j=1}^N W_{t-1}^{(j)} \tilde{w}_t^{(j)}}$ for $i = 1, 2, ..., N$.

Set $\hat{Z}_t = \hat{Z}_{t-1} \sum_{i=1}^N W_{t-1}^{(i)} \tilde{w}_t^{(i)}$.

Conditional resampling step:

Compute $ESS_t = \left\{ \sum_{i=1}^N \left(W_t^{(i)} \right)^2 \right\}^{-1}$.

if $ESS_t < S$ **then**

Resample $\tilde{X}_{t-1}^{(1)}, ..., \tilde{X}_{t-1}^{(N)} \sim \text{Multinomial}(\{\tilde{X}_{t-1}^{(i)}\}_{i=1}^N; \{W_t^{(i)}\}_{i=1}^N)$.

Set $W_t^{(1)} = W_t^{(2)} = ... = W_t^{(N)} = \frac{1}{N}$.

end

Mutation step:

Draw $X_t^{(i)} \sim K_t(\tilde{X}_{t-1}^{(i)}, \cdot)$ for $i = 1, 2, ..., N$.

end

holds here as well.

Transport step At the beginning of iteration t of the main loop, we have the weighted particles $\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N$ that form the particle approximation in (2.19). In this step, a transport map T_t is learned by minimizing an estimate of the KL divergence of $T_{t\#}\pi_{t-1}$ and π_t, by optimizing over the parameters ϕ_t of T_t using gradient descent methods, so that the distribution of the transported particles alone are close to that of π_t, i.e. we hope to get the approximation

$$\pi_t \approx \sum_{i=1}^N W_{t-1}^{(i)} \delta_{T_t(X_{t-1}^{(i)})}. \tag{3.1}$$

Once learned, the particles are transported using the flow from $X_{t-1}^{(i)}$ to $\tilde{X}_{t-1}^{(i)} = T_t(X_{t-1}^{(i)})$ for each $i \in \{1, 2, ..., N\}$.

Importance sampling step Just like in SMC, the purpose of the importance sampling step is to update the weights of the current set of weighted particles. However, since this step takes place after the transport step, we instead begin with the particle approximation

$$T_{t\#}\hat{\pi}_{t-1} = \sum_{i=1}^N W_{t-1}^{(i)} \delta_{\tilde{X}_{t-1}^{(i)}}, \tag{3.2}$$

and wish to obtain $\{W_t^{(i)}\}_{i=1}^N$ such that

$$\pi_t \approx \sum_{i=1}^N W_t^{(i)} \delta_{\tilde{X}_{t-1}^{(i)}}. \tag{3.3}$$

The easy computation of the Jacobian of T_t allows us to compute IS weights without a closed form for the density of $T_{t\#}\pi_{t-1}$, enabling the use of NFs that are as complex as one wishes. In this step, we first compute the incremental weights

$$\tilde{w}_t^{(i)} = \frac{f_t(\tilde{X}_{t-1}^{(i)}) |\det J_{T_t}(X_{t-1}^{(i)})|}{f_{t-1}(X_{t-1}^{(i)})} \tag{3.4}$$

and then assign the weights

$$W_t^{(i)} = \frac{W_{t-1}^{(i)} \tilde{w}_t^{(i)}}{\sum_{j=1}^N W_{t-1}^{(j)} \tilde{w}_t^{(j)}}, \tag{3.5}$$

for each particle $i \in \{1, 2, ..., N\}$, for iteration $t \in \{1, 2, ..., n\}$.

Conditional resampling step This step is essentially identical to the conditional resampling step in SMC. Given a threshold $S \in (1, N)$, we resample $\tilde{X}_{t-1}^{(i)} \sim \text{Multinomial}(\{W_t^{(j)}\}_{j=1}^N, \{\tilde{X}_{t-1}^{(j)}\}_{j=1}^N)$ and then reset the weights to $W_t^{(i)} = \frac{1}{N}$ for all particles $i \in \{1, 2, ..., N\}$ if the ESS $\{\sum_{i=1}^N (W_t^{(i)})^2\}^{-1}$ falls below S.

Mutation step Finally, we sample $X_t^{(i)} \sim K_t(X_{t-1}^{(i)}, \cdot)$ for some π_t-invariant MCMC kernel K_t, for each $i \in \{1, 2, ..., N\}$. Indeed, as in SMC, the resultant set of weighted particles $\{W_t^{(i)}, X_t^{(i)}\}_{i=1}^N$ at this point satisfies (2.23), which we will verify later.

3.1.2 Flow training

To obtain the objective function for learning T_t, we can decompose the KL divergence $D_{KL}(T_{t\#}\pi_{t-1}||\pi_t)$ in a similar fashion as (2.15) to get

$$D_{KL}(T_{t\#}\pi_{t-1}||\pi_t) = \mathbb{E}_{\pi_{t-1}}[\log f_{t-1}(X) - \log|\det J_{T_t}(X)| - \log f_t(T_t(X))] + \log\frac{Z_t}{Z_{t-1}}. \tag{3.6}$$

Since $\log\frac{Z_t}{Z_{t-1}}$ does not depend on T_t's parameters ϕ_t, we can remove it and put the remaining terms in the objective function, i.e. we have the objective function $\mathcal{L}_t$ to be

$$\mathcal{L}_t(\phi_t) = \mathbb{E}_{\pi_{t-1}}[\log f_{t-1}(X) - \log|\det J_{T_t}(X)| - \log f_t(T_t(X))]. \tag{3.7}$$

Using the weighted particles $\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N$, we can get an IS estimate of the objective function as

$$\begin{aligned}\tilde{\mathcal{L}}_t(\phi_t, \{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N) &= \sum_{i=1}^N W_{t-1}^{(i)} h_t(X_{t-1}^{(i)}),\\ h_t(x) &= \log f_{t-1}(x) - \log|\det J_{T_t}(x)| - \log f_t(T_t(x)).\end{aligned} \tag{3.8}$$

This objective function can then be minimized as described in the transport step in the previous section. Note that a global minimizer need not be obtained in the training phase of T_t, as shown in the convergence results derived in [AMD21].

3.1.3 Correctness of weights

At the mutation step of iteration $t \in \{1, 2, ..., n\}$, we have weighted particles $\{W_t^{(i)}, \tilde{X}_{t-1}^{(i)}\}_{i=1}^N$ that satisfy the approximation

$$T_{t\#}\pi_{t-1} \approx T_{t\#}\hat{\pi}_{t-1} = \sum_{i=1}^N W_t^{(i)} \delta_{\tilde{X}_{t-1}^{(i)}} \tag{3.9}$$

Assuming a fixed flow T_t (obtained after the flow is learned), we then follow an argument identical to 2.5.2 up to (2.32), where we use $T_{t\#}\pi_{t-1}$ in place of π_{t-1} to get the approximation

$$\begin{aligned}\pi_t &\approx \sum_{i=1}^N \frac{\pi_t(\tilde{X}_{t-1}^{(i)})}{T_{t\#}\pi_{t-1}(\tilde{X}_{t-1}^{(i)})} W_{t-1}^{(i)} \delta_{X_t^{(i)}}\\ &= \sum_{i=1}^N \frac{\pi_t(\tilde{X}_{t-1}^{(i)})}{\pi_{t-1}(X_{t-1}^{(i)})|\det J_{T_t}(X_{t-1}^{(i)})|^{-1}} W_{t-1}^{(i)} \delta_{X_t^{(i)}}\\ &= \sum_{i=1}^N \frac{\pi_t(\tilde{X}_{t-1}^{(i)})|\det J_{T_t}(X_{t-1}^{(i)})|}{\pi_{t-1}(X_{t-1}^{(i)})} W_{t-1}^{(i)} \delta_{X_t^{(i)}}.\end{aligned} \tag{3.10}$$

For intractable π_{t-1} and π_t, we must approximate the weight term above to get

$$\begin{aligned}\frac{\pi_t(\tilde{X}_{t-1}^{(i)})|\det J_{T_t}(X_{t-1}^{(i)})|}{\pi_{t-1}(X_{t-1}^{(i)})}W_{t-1}^{(i)} &= W_{t-1}^{(i)}\frac{f_t(\tilde{X}_{t-1}^{(i)})|\det J_{T_t}(X_{t-1}^{(i)})|}{f_{t-1}(X_{t-1}^{(i)})} \Big/ \left\{\frac{Z_t}{Z_{t-1}}\right\} \\ &\approx W_{t-1}^{(i)}\frac{f_t(\tilde{X}_{t-1}^{(i)})|\det J_{T_t}(X_{t-1}^{(i)})|}{f_{t-1}(X_{t-1}^{(i)})} \Big/ \left\{\frac{\widehat{Z_t}}{Z_{t-1}}\right\} \\ &= \frac{W_{t-1}^{(i)}\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} W_{t-1}^{(j)}\tilde{w}_t^{(j)}} \\ &= W_t^{(i)},\end{aligned} \tag{3.11}$$

as defined in (3.5), where we used $\tilde{w}_t^{(i)}$ as per (3.4) and the approximation $\frac{Z_t}{Z_{t-1}}$ in (3.12).

3.1.4 Estimation of evidence

At iteration $t \in \{1, 2, ..., n\}$ of the main loop, we can approximate the ratio $\frac{Z_t}{Z_{t-1}}$ using the weights $\{W_{t-1}^{(i)}\}_{i=1}^N$ and the incremental weights $\{\tilde{w}_t^{(i)}\}_{i=1}^N$. Observe that

$$\begin{aligned}\frac{Z_t}{Z_{t-1}} &= \frac{1}{Z_{t-1}}\int_E f_t(y)dy = \int_E \frac{1}{Z_{t-1}} f_t(T_t(x))dT_t(x) = \int_E \frac{1}{Z_{t-1}} f_t(T_t(x))|\det J_{T_t}(x)|dx \\ &= \int_E \frac{f_t(T_t(x))|\det J_{T_t}(x)|}{f_{t-1}(x)}\pi_{t-1}(x)dx = \mathbb{E}_{\pi_{t-1}}\left[\frac{f_t(T_t(X))|\det J_{T_t}(X)|}{f_{t-1}(X)}\right] \\ &\approx \sum_{i=1}^{N} W_{t-1}^{(i)}\frac{f_t(T_t(X_{t-1}^{(i)}))|\det J_{T_t}(X_{t-1}^{(i)})|}{f_{t-1}(X_{t-1}^{(i)})} = \sum_{i=1}^{N} W_{t-1}^{(i)}\tilde{w}_t^{(i)},\end{aligned} \tag{3.12}$$

i.e. we have the estimate

$$\frac{\widehat{Z_t}}{Z_{t-1}} = \sum_{i=1}^{N} W_{t-1}^{(i)}\tilde{w}_t^{(i)}. \tag{3.13}$$

This allows us to estimate the ratio of the evidences $\frac{Z_t}{Z_0}$ by taking the product

$$\frac{\widehat{Z_t}}{Z_0} = \prod_{t=1}^{n}\left\{\sum_{i=1}^{N} W_{t-1}^{(i)}\tilde{w}_t^{(i)}\right\}, \tag{3.14}$$

from which it is straightforward to obtain an estimate $\hat{Z}_t$ of Z_t given the exact evidence (or an approximation of) Z_0. This can be simply satisfied by a choice of π_0 that allows for tractable computation or approximation of Z_0.

Algorithm 3: Practical Annealed Flow Transport

Input: Number of training, validation and test particles N_{train}, N_{val}, N_{test}, bridging distributions $\{\pi_t\}_{t=1}^n$ with unnormalized density functions $\{f_t\}_{t=1}^n$, resampling thresholds $S_{train} \in (1, N_{train})$, $S_{val} \in (1, N_{val})$, $S_{test} \in (1, N_{test})$, the initial flow parameters $\{\phi_t^{(0)}\}_{t=1}^n$, and the number of training iterations J.

Output: Approximations $(\hat{\pi}_n, \hat{Z}_n)$ to (π_n, Z_n).

Initialization step:

Initialize weighted particles $\{W_0^{(i),a}, X_0^{(i),a}\}_{i=1}^{N_a}$, where $X_0^{(i),a} \sim \pi_0$ and $W_0^{(i),a} = \frac{1}{N_a}$ for $i \in \{1, 2, ..., N_a\}$, for $a \in \{train, val, test\}$.

Initialize evidence estimate $\hat{Z}_0 = Z_0$ from proposal distribution.

for $t = 1, 2, ..., n$ **do**

Transport step:

for $j = 1, 2, ..., J$ **do**

Perform one iteration of a gradient descent method to minimize $\tilde{\mathcal{L}}_t(\phi, \{W_{t-1}^{(i),train}, X_{t-1}^{(i),train}\}_{i=1}^N)$ over ϕ using $\phi_t^{(j-1)}$.

Set $\phi_t^{(j)}$ to be the flow parameters obtained from the gradient descent iteration.

end

Set the parameters of T_t to be $\phi_t = \arg\min_\phi \tilde{\mathcal{L}}_t(\phi_t^{(j)}, \{W_{t-1}^{(i),val}, X_{t-1}^{(i),val}\}_{i=1}^N)$.

Set $\tilde{X}_{t-1}^{(i),a} = T_t(X_{t-1}^{(i),a})$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Importance sampling step:

Set $\tilde{w}_t^{(i),a} = \frac{f_t(\tilde{X}_{t-1}^{(i),a}) |\det J_{T_t}(X_{t-1}^{(i),a})|}{f_{t-1}(X_{t-1}^{(i),a})}$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Set $W_t^{(i),a} = \frac{W_{t-1}^{(i),a} \tilde{w}_t^{(i),a}}{\sum_{j=1}^{N_a} W_{t-1}^{(j),a} \tilde{w}_t^{(j),a}}$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Set $\hat{Z}_t = \hat{Z}_{t-1} \sum_{i=1}^{N_{test}} W_{t-1}^{(i),test} \tilde{w}_t^{(i),test}$.

Conditional resampling step:

for $a \in \{train, val, test\}$ **do**

Compute $ESS_t^a = \left\{ \sum_{i=1}^{N_a} \left(W_t^{(i),a} \right)^2 \right\}^{-1}$.

if $ESS_t^a < S_a$ **then**

Resample $\tilde{X}_{t-1}^{(1),a}, ..., \tilde{X}_{t-1}^{(N_a),a} \sim \text{Multinomial}(\{\tilde{X}_{t-1}^{(i),a}\}_{i=1}^{N_a}; \{W_t^{(i),a}\}_{i=1}^{N_a})$.

Set $W_t^{(1),a} = W_t^{(2),a} = ... = W_t^{(N_a),a} = \frac{1}{N_a}$.

end

end

Mutation step:

Draw $X_t^{(i),a} \sim K_t(\tilde{X}_{t-1}^{(i),a}, \cdot)$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

end

3.1.5 Variants and extensions

By training $\{T_t\}_{t=1}^n$ on the same set of particles to be propagated through the AFT algorithm, the flows become dependent on these particles, resulting in biased estimates of the evidences $\{\hat{Z}_t\}_{t=1}^n$ [AMD21]. This is described in [MARD22] as the sample replenishment problem, where sample complexity of estimation exceeds the finite number of available particles, causing the flows $\{T_t\}_{t=1}^n$ to over-fit to the particles and produce heavily biased estimates of evidence and expectations. [AMD21] proposes an adjustment to AFT called practical AFT, where an additional training set and validation set of particles are generated and propagated through the algorithm along with the original set of particles. The training set is used to train $\{T_t\}_{t=1}^n$ by gradient descent methods, and the validation set is used to evaluate the objective of the flow. In an iteration of the main loop, once training of the flow is completed, the set of flow parameters that obtained the lowest objective value on the validation set is fixed as the parameters of the flow. All three sets of particles are then propagated through the iteration as in the original AFT. Practical AFT is described in algorithm 3. Due to its one-pass design, it is similar enough to SMC to extend it in similar ways, such as incorporating an adaptive selection of bridging distributions to maintain a fixed percentage decrease in ESS [JSDT11] [SC13] [BJKT16] [ZJA16].

3.2 Continual Repeated Annealed Flow Transport

To address the sample replenishment problem of AFT, [MARD22] proposes CRAFT, which borrows the same principle of extending the SMC algorithm by interleaving NFs between the MCMC transitions, but abandons the one-pass regime in favour of training the NFs through multiple passes of independent batches of generated particles. This allows CRAFT to introduce significantly more particles to match or exceed the sample complexity of estimating the training gradients of the NFs as compared to AFT. Under the CRAFT training scheme, the algorithm is more aligned to the traditional machine learning training paradigm, making it easier to adopt and more amenable to parallel computing [MARD22].

3.2.1 Description of algorithm

To learn the flows, CRAFT repeats a slightly modified AFT algorithm in an 'outer' loop. Other than some adjustments to the transport step, this modified AFT algorithm features an identical initialization step, importance sampling step, conditional resampling step, and mutation step as described in 3.1.1, with the adjusted transport, importance sampling, conditional resampling, and mutation steps wrapped in a 'main' loop as is done in AFT (which we will refer to as the inner loop of the CRAFT training phase). Once trained, the NFs are fixed, and one more pass of the modified AFT algorithm is performed, where the produced approximations $\hat{\pi}_n$ and $\hat{Z}_n$ are returned. The full CRAFT algorithm is presented in algorithm 4.

Modified transport step At iteration t of the inner loop, we transport each of the particles $X_{t-1}^{(i)}$ to $\tilde{X}_{t-1}^{(i)} = T_t(X_{t-1}^{(i)})$ for $i \in \{1, 2, ..., N\}$. Then update the flow parameters by performing one iteration of a gradient descent method to minimize $\tilde{\mathcal{L}}_t(\phi, \{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N)$ (from (3.8)) over

Algorithm 4: Continual Repeated Annealed Flow Transport

Input: Number of particles N, bridging distributions $\{\pi_t\}_{t=1}^n$ with unnormalized density functions $\{f_t\}_{t=1}^n$, resampling threshold $S \in (1, N)$, the initial flow parameters $\{\phi_t^{(0)}\}_{t=1}^n$, and the number of training iterations J.

Output: Approximations $(\hat{\pi}_n, \hat{Z}_n)$ to (π_n, Z_n).

for $j = 1, 2, ..., J+1$ **do**

/* The extra pass after J training iterations of the outer loop serves as the evaluation run of CRAFT. */

Initialization step:

Initialize weighted particles $\{W_0^{(i)}, X_0^{(i)}\}_{i=1}^N$, where $X_0^{(i)} \sim \pi_0$ and $W_0^{(i)} = \frac{1}{N}$ for $i \in \{1, 2, ..., N\}$.

Initialize evidence estimate $\hat{Z}_0 = Z_0$ from proposal distribution.

for $t = 1, 2, ..., n$ **do**

Transport step:

Set $\tilde{X}_{t-1}^{(i)} = T_t(X_{t-1}^{(i)})$ and $\gamma_t^{(i)} = |\det J_{T_t}(X_{t-1}^{(i)})|$ for $i = 1, 2, ..., N$.

if $j \leq J$ **then**

Perform one iteration of a gradient descent method to minimize $\tilde{\mathcal{L}}_t(\phi, \{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N)$ over ϕ using $\phi_t^{(j-1)}$, and set $\phi_t^{(j)}$ to be the flow parameters obtained.

Set the parameters of T_t to be $\phi_t^{(j)}$.

end

Importance sampling step:

Set $\tilde{w}_t^{(i)} = \frac{f_t(\tilde{X}_{t-1}^{(i)})\gamma_t^{(i)}}{f_{t-1}(X_{t-1}^{(i)})}$ for $i = 1, 2, ..., N$.

Set $W_t^{(i)} = \frac{W_{t-1}^{(i)}\tilde{w}_t^{(i)}}{\sum_{j=1}^N W_{t-1}^{(j)}\tilde{w}_t^{(j)}}$ for $i = 1, 2, ..., N$.

Set $\hat{Z}_t = \hat{Z}_{t-1} \sum_{i=1}^N W_{t-1}^{(i)}\tilde{w}_t^{(i)}$.

Conditional resampling step:

Compute $ESS_t = \left\{\sum_{i=1}^N \left(W_t^{(i)}\right)^2\right\}^{-1}$.

if $ESS_t < S$ **then**

Resample $\tilde{X}_{t-1}^{(1)}, ..., \tilde{X}_{t-1}^{(N)} \sim \text{Multinomial}(\{\tilde{X}_{t-1}^{(i)}\}_{i=1}^N; \{W_t^{(i)}\}_{i=1}^N)$.

Set $W_t^{(1)} = W_t^{(2)} = ... = W_t^{(N)} = \frac{1}{N}$.

end

Mutation step:

Draw $X_t^{(i)} \sim K_t(\tilde{X}_{t-1}^{(i)}, \cdot)$ for $i = 1, 2, ..., N$.

end

end

ϕ using the current flow parameters. This order of operations is important as we want to only apply the updated flow parameters in the next iteration of the outer loop (i.e. the next pass of the modified AFT).

3.2.2 Flow training

The learning objective in CRAFT can be taken to have the form

$$H(\phi) = \sum_{t=1}^{n} D_{KL}(T_t(\phi)_{\#}\pi_{t-1}||\pi_t), \tag{3.15}$$

where the sum runs over each transition between bridging distributions. Separating the terms of the sum into its constituent n KL divergences, we obtain the objectives of identical form as (3.6) for each bridging distribution. Removing the log-evidence ratio term independent of the flow parameters, we get a function of the form (3.7) (which has an IS estimate of (3.8)) that can be used as an objective to train the flow parameters of each iteration of the inner loop with. Decomposing the objective in this fashion can reduce the mode-seeking tendencies that come with optimizing the reverse KL divergence [MARD22]. Indeed, we end up with the same objectives to minimize the flow parameters in each iteration of the inner loop just like in the main loop of AFT. The main difference lies in when the parameters updated by gradient descent methods are applied; in AFT, the updated parameters used immediately after obtaining them in the same iteration of the main loop, whereas in CRAFT, the updated parameters are only used in the next pass.

3.2.3 Correctness of weights and evidence estimation

Once the flows $\{T_t\}_{t=1}^n$ are learned, they are fixed and essentially become deterministic maps. Thus, identical arguments and results of 3.1.3 and 3.1.4 apply here too.

3.2.4 Variants and extensions

Notice that, due to CRAFT's multiple-pass training regime, the bridging distributions must be determined before training begins (or at the very latest, as $\{T_t\}_{t=1}^n$ and $\{\phi_t\}_{t=1}^n$ are initialized). This impedes the direct application of on-line adaptive methods to select the bridging distributions (see 3.1.5), unlike AFT and SMC. Instead, variants using off-line methods of selecting the bridging distributions can be pursued, such as that proposed by [NSPD16]. [MSS+23] proposes an alternate training scheme in their Flow Annealed Importance Sampling Bootstrap variant of CRAFT, where the flows $\{T_t\}_{t=1}^n$ are trained to minimize the variance of the importance sampling weights, in order to further stamp out the mode-seeking phenomenon associated with optimizing KL divergence objectives.

4 Time Embedded Annealed Flow Transport Monte Carlo

In this section, we introduce time-embedded AFT (TE-AFT) and time-embedded CRAFT (TE-CRAFT), extensions of the annealed flow transport Monte Carlo methods that seek to make more

efficient use of the flow component in AFT and CRAFT, as well as improve their scalability with the number of bridging distributions, by sharing the flow parameters across all of the NFs $\{T_t\}_{t=1}^n$. Just like AFT and CRAFT, TE-AFT and TE-CRAFT are sampling algorithms that build on the SMC structure, incorporating the use of NFs and Markov kernels to propagate samples from an initial distribution π_0 through a sequence of bridging distributions to produce weighted samples that can form a particle approximation $\hat{\pi}_n$ of the target distribution π_n.

4.1 Choice of bridging distributions

The bridging distributions $\{\pi_t\}_{t=1}^n$ are a key component that characterizes the SMC approach to sampling algorithms. A poor choice of $\{\pi_t\}_{t=1}^n$ can lead to high variance in estimators produced from an SMC sampler if there are big enough differences between any two consecutive bridging distributions [AMD21]. A popular way to choose $\{\pi_t\}_{t=1}^n$ is by the geometric annealing schedule [GM98] [Nea01] [MDJ06] [AMD21] [MARD22], where we have the density

$$\pi_t(x) \propto \pi_0(x)^{1-\beta_t}\pi_n(x)^{\beta_t}, \tag{4.1}$$

with $0 = \beta_0 < \beta_1 < ... < \beta_n = 1$. β_t is commonly referred to as the annealing temperature, or the schedule time. In the context of TE-AFT and TE-CRAFT, the choice of bridging distributions will be based on the geometric annealing schedule, as will all discussion on SMC, AFT, and CRAFT, including the simulation experiments, unless stated otherwise explicitly. Note that, however, any general sequence of distributions $\{\pi_t\}_{t=1}^n$ smoothly bridging π_0 to π_n that has a way of indexing the distributions using a variable $\beta_t \in [0, 1], t \in \{0, 1, 2, ..., n\}$ can be used in place of the geometric annealing schedule as well.

4.2 Time embedding

Since the parameters for the flows in TE-AFT and TE-CRAFT are shared, information about the schedule time must be injected into the flows in order for them to perform different transformations based on the annealing temperature. We take inspiration from the positional embedding method used in [VSP+17] and encode schedule time β as a d-dimensional vector TE_β such that

$$\begin{aligned} TE_{\beta,2k} &= \sin\left(\frac{10\beta}{10000^{2k/d}}\right), \\ TE_{\beta,2k+1} &= \cos\left(\frac{10\beta}{10000^{2k/d}}\right), \end{aligned} \tag{4.2}$$

where $k \in \{k \in \mathbb{N} : 2k \leq d, 2k+1 \leq d\}$. Unlike [VSP+17], d can be tuned as a hyperparameter – it need not be of the same dimension as the particles being propagated, as we do not intend to sum the time embeddings with the particle positions. Instead, the embeddings are fed as separate inputs to the networks of the flow that determine the transformation, such as a dense layer that also takes in masked inputs to output the scaling and shifting factors of an affine transformation in an affine coupling layer of a RealNVP flow [DSDB16] (see figure 2).

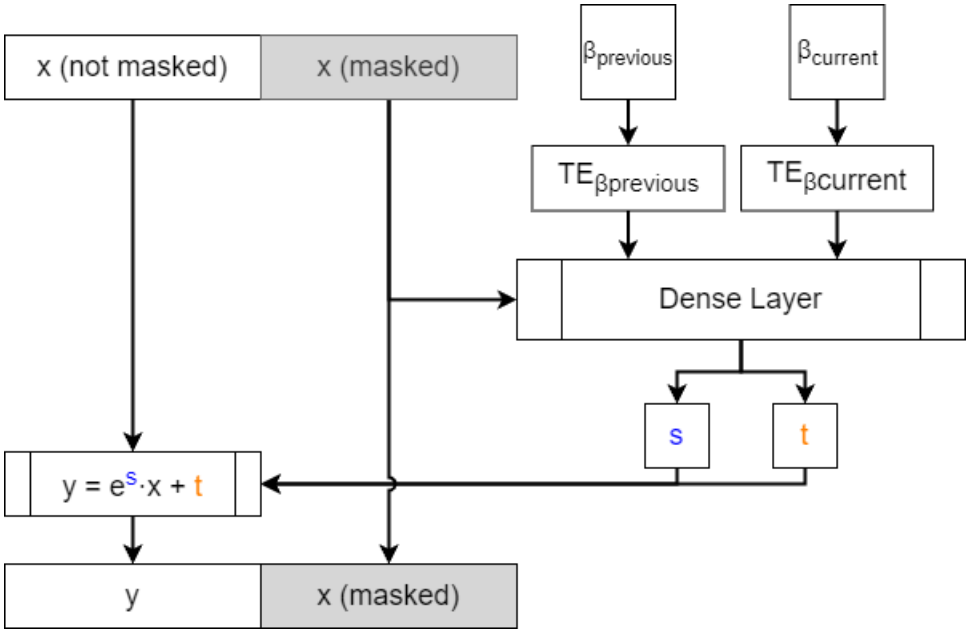

Figure 2: Time-embedded affine coupling layer with input particle x. β_{previous} and β_{current} are the previous and current annealing temperatures respectively.

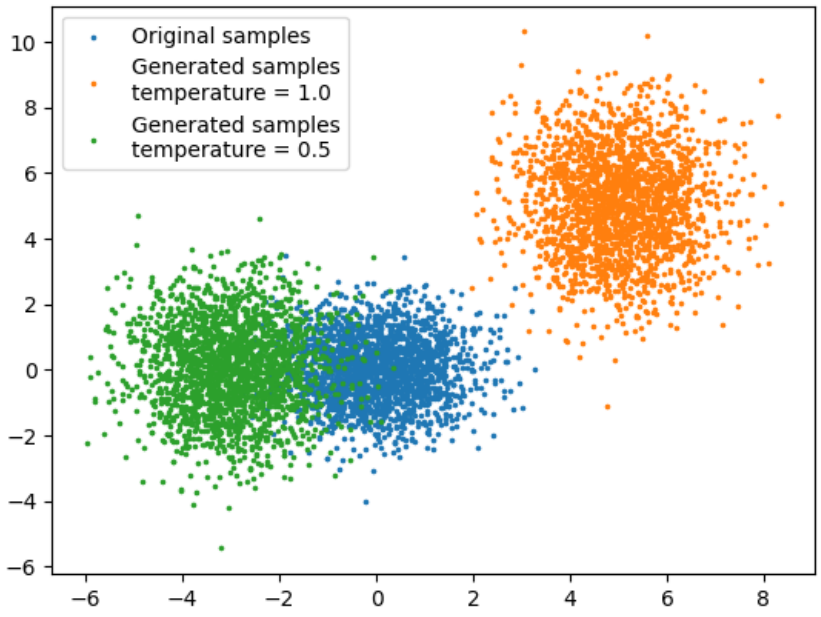

Figure 3: Samples generated from a learned time-embedded diagonal affine flow transporting samples from $N_2[\mathbf{0}, \mathbf{I}]$ to $N_2[(5,5)^T, \mathbf{I}]$ at annealing temperature 0.5, and to $N_2[(-3,0)^T, \mathbf{I}]$ at annealing temperature 1.

To demonstrate the effectiveness of learning separate transformations using schedule-time embeddings, we implement a simple toy example in Python, using the Jax, Flax, and Optax packages, where a diagonal affine flow is tasked to learn to transform particles drawn from $N_2[\mathbf{0}, \mathbf{I}]$ to $N_2[(5,5)^T, \mathbf{I}]$ at annealing temperature 0.5, and to $N_2[(-3,0)^T, \mathbf{I}]$ at annealing temperature 1. The annealing temperatures of the input and output distributions are encoded and input to a dense layer which outputs the scaling and shifting factors of an affine transformation that is applied onto the input samples. The samples generated from the learned flow is shown in figure 3.

4.3 Time Embedded Annealed Flow Transport

In AFT, if the same type of flow model is used to learn each of the transport maps $\{T_t\}_{t=1}^n$, then we can re-interpret the AFT algorithm to be using the same NF to transport particles in each iteration of the main loop, with the flow parameters being reset at the start of a new iteration. TE-AFT builds on this interpretation, with the main changes being that the flow parameters are no longer reset in each iteration, and the embedding and insertion of the previous and current schedule times into the flow. Once the maps are learned and the flow is fixed at inference time, the flow can be treated as a deterministic transformation (that takes as input the previous and current schedule times) in each iteration of the main loop, and thus, the arguments for the correctness of the weights and evidence estimate in AFT apply to TE-AFT as well.

4.3.1 Description of algorithm

Due to the sample replenishment problem, we developed TE-AFT as an extension to practical AFT (algorithm 3) rather than AFT itself. The one-pass design, overall structure (such as the main loop), and all of the subroutines remain identical to that of practical AFT, except for the transport step, where we train the same flow used in the previous iteration (i.e. begin training with the final flow parameters chosen in the previous iteration) instead of training a freshly initialized flow. The full TE-AFT algorithm is shown in algorithm 5.

4.3.2 Flow Training

Under TE-AFT, the flows take as input the previous and current schedule times. Thus, we can label the shared flow as T, and set $T_t(x) = T(x, \beta_{t-1}, \beta_t)$ for $x \in E$ and $t = 1, 2, ..., n$. Using this notation, we can define the IS estimate of the objective function identically to (3.8) for each schedule time, which will be optimized to train the flow parameters as done in practical AFT; i.e. trained using a training set of particles on the objective function at the current schedule time, with the final parameters chosen based on the minimum objective score obtained using a validation set of particles.

From the perspective of using separate flows at each schedule time, we can interpret TE-AFT as initializing the parameters for the current schedule time's flow by copying the parameters of the previous schedule time's flow. This leverages on the structures and relations in the hidden

Algorithm 5: Time Embedded Annealed Flow Transport

Input: Number of training, validation and test particles N_{train}, N_{val}, N_{test}, bridging distributions $\{\pi_t\}_{t=1}^n$ with unnormalized density functions $\{f_t\}_{t=1}^n$, resampling thresholds $S_{train} \in (1, N_{train})$, $S_{val} \in (1, N_{val})$, $S_{test} \in (1, N_{test})$, the initial flow parameters ϕ_0, and the number of training iterations J.

Output: Approximations $(\hat{\pi}_n, \hat{Z}_n)$ to (π_n, Z_n).

Initialization step:

Initialize weighted particles $\{W_0^{(i),a}, X_0^{(i),a}\}_{i=1}^{N_a}$, where $X_0^{(i),a} \sim \pi_0$ and $W_0^{(i),a} = \frac{1}{N_a}$ for $i \in \{1, 2, ..., N_a\}$, for $a \in \{train, val, test\}$.

Initialize evidence estimate $\hat{Z}_0 = Z_0$ from proposal distribution.

for $t = 1, 2, ..., n$ **do**

Transport step:

Set $\phi_t^{(0)} = \phi_{t-1}$.

for $j = 1, 2, ..., J$ **do**

Perform one iteration of a gradient descent method to minimize $\tilde{\mathcal{L}}_t(\phi, \{W_{t-1}^{(i),train}, X_{t-1}^{(i),train}\}_{i=1}^N)$ (as redefined in 4.3.2) over ϕ using $\phi_t^{(j-1)}$.

Set $\phi_t^{(j)}$ to be the flow parameters obtained from the gradient descent iteration.

end

Set the parameters of T_t to be $\phi_t = \arg\min_\phi \tilde{\mathcal{L}}_t(\phi_t^{(j)}, \{W_{t-1}^{(i),val}, X_{t-1}^{(i),val}\}_{i=1}^N)$.

Set $\tilde{X}_{t-1}^{(i),a} = T_t(X_{t-1}^{(i),a})$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Importance sampling step:

Set $\tilde{w}_t^{(i),a} = \frac{f_t(\tilde{X}_{t-1}^{(i),a})|\det J_{T_t}(X_{t-1}^{(i),a})|}{f_{t-1}(X_{t-1}^{(i),a})}$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Set $W_t^{(i),a} = \frac{W_{t-1}^{(i),a}\tilde{w}_t^{(i),a}}{\sum_{j=1}^{N_a} W_{t-1}^{(j),a}\tilde{w}_t^{(j),a}}$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Set $\hat{Z}_t = \hat{Z}_{t-1} \sum_{i=1}^{N_{test}} W_{t-1}^{(i),test}\tilde{w}_t^{(i),test}$.

Conditional resampling step:

for $a \in \{train, val, test\}$ **do**

Compute $ESS_t^a = \left\{\sum_{i=1}^{N_a}\left(W_t^{(i),a}\right)^2\right\}^{-1}$.

if $ESS_t^a < S_a$ **then**

Resample $\tilde{X}_{t-1}^{(1),a}, ..., \tilde{X}_{t-1}^{(N_a),a} \sim \text{Multinomial}(\{\tilde{X}_{t-1}^{(i),a}\}_{i=1}^{N_a}; \{W_t^{(i),a}\}_{i=1}^{N_a})$.

Set $W_t^{(1),a} = W_t^{(2),a} = ... = W_t^{(N_a),a} = \frac{1}{N_a}$.

end

end

Mutation step:

Draw $X_t^{(i),a} \sim K_t(\tilde{X}_{t-1}^{(i),a}, \cdot)$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

end

layers learned in the previous flow, which can facilitate training in the current flow as compared to randomly generated initial parameters. This effect is especially useful when the transformation to be learned at the current schedule time is very similar to the transformation learned at the previous schedule time.

4.4 Time-Embedded Continual Repeated Annealed Flow Transport

TE-CRAFT is an extension of CRAFT, where the flows used in each annealing temperature are replaced by a single time-embedded flow. This allows the TE-CRAFT algorithm to maintain the same number of parameters regardless of the number of bridging distributions used, as compared to the linear scaling of the number of parameters with the number of bridging distributions in CRAFT. Exposing the schedule times to the flow can also allow it to learn the transport map for unseen schedule times as well, opening up the possibility of a variant of CRAFT with adaptive selection of schedule times. Just like how TE-AFT behaves identically to AFT at inference time, TE-CRAFT becomes identical to CRAFT when the flows have finished training are fixed as deterministic transformations, allowing the same results for the correctness of weights and evidence estimate to apply to TE-CRAFT.

4.4.1 Description of Algorithm

TE-CRAFT uses the same structure of an outer loop repeating a modified AFT algorithm to train the time-embedded flow (with the inner loop being the main loop of the modified AFT algorithm). The main difference between TE-CRAFT and CRAFT is a modification to the transport step in the inner loop, where the gradient descent and parameter updates are performed in a separate update step instead – the transport step is reduced to a simple feedforward of the particles through the time-embedded flow. We thus shall focus on the new update step in this section. We lay out the TE-CRAFT algorithm in algorithm 6.

Update step Instead of performing an iteration of a gradient descent method and updating the flow parameters in the transport step as done in CRAFT, TE-CRAFT outsources this to an update step after the inner loop is completed. TE-CRAFT collects the objective values from each iteration of the inner loop in a sum, and optimizes the time-embedded flow parameters to minimize this sum by performing one iteration of a gradient descent method. The flow used in the inner loop of the next outer loop iteration will then use the parameters updated at the end of the previous outer loop iteration.

4.4.2 Flow Training

Using the same notation for the time-embedded flow at each annealing temperature as in 4.3.2, the learning objective in TE-CRAFT has the same expression as (3.2.2). However, in TE-CRAFT, this sum is no longer split into its constituent KL divergences to optimize over in each inner loop iteration. While we still use the particles at each schedule time to form IS estimates of $D_{KL}(T_{t\#}\pi_{t-1}||\pi_t)$ for each $t \in \{1, 2, ..., n\}$, we now sum these terms up at the end of the inner

Algorithm 6: Time-Embedded Continual Repeated Annealed Flow Transport

Input: Number of particles N, bridging distributions $\{\pi_t\}_{t=1}^n$ with unnormalized density functions $\{f_t\}_{t=1}^n$, resampling threshold $S \in (1, N)$, the initial flow parameters $\phi^{(0)}$, and the number of training iterations J.

Output: Approximations $(\hat{\pi}_n, \hat{Z}_n)$ to (π_n, Z_n).

Set the parameters of the flow T to be the initial parameters $\phi^{(0)}$.

for $j = 1, 2, ..., J+1$ **do**

/* The extra pass after J training iterations of the outer loop serves as the evaluation run of CRAFT. */

Initialization step:

Initialize weighted particles $\{W_0^{(i)}, X_0^{(i)}\}_{i=1}^N$, where $X_0^{(i)} \sim \pi_0$ and $W_0^{(i)} = \frac{1}{N}$ for $i \in \{1, 2, ..., N\}$.

Initialize evidence estimate $\hat{Z}_0 = Z_0$ from proposal distribution.

for $t = 1, 2, ..., n$ **do**

Transport step:

Set $\tilde{X}_{t-1}^{(i)} = T_t(X_{t-1}^{(i)})$ and $\gamma_t^{(i)} = |\det J_{T_t}(X_{t-1}^{(i)})|$ for $i = 1, 2, ..., N$.

Importance sampling step:

Set $\tilde{w}_t^{(i)} = \frac{f_t(\tilde{X}_{t-1}^{(i)})\gamma_t^{(i)}}{f_{t-1}(X_{t-1}^{(i)})}$ for $i = 1, 2, ..., N$.

Set $W_t^{(i)} = \frac{W_{t-1}^{(i)}\tilde{w}_t^{(i)}}{\sum_{j=1}^N W_{t-1}^{(j)}\tilde{w}_t^{(j)}}$ for $i = 1, 2, ..., N$.

Set $\hat{Z}_t = \hat{Z}_{t-1} \sum_{i=1}^N W_{t-1}^{(i)}\tilde{w}_t^{(i)}$.

Conditional resampling step:

Compute $ESS_t = \left\{\sum_{i=1}^N \left(W_t^{(i)}\right)^2\right\}^{-1}$.

if $ESS_t < S$ **then**

Resample $\tilde{X}_{t-1}^{(1)}, ..., \tilde{X}_{t-1}^{(N)} \sim \text{Multinomial}(\{\tilde{X}_{t-1}^{(i)}\}_{i=1}^N; \{W_t^{(i)}\}_{i=1}^N)$.

Set $W_t^{(1)} = W_t^{(2)} = ... = W_t^{(N)} = \frac{1}{N}$.

end

Mutation step:

Draw $X_t^{(i)} \sim K_t(\tilde{X}_{t-1}^{(i)}, \cdot)$ for $i = 1, 2, ..., N$.

end

Update step:

if $j \leq J$ **then**

Perform one iteration of a gradient descent method to optimize $\tilde{H}^{(j)}(\phi)$ using $\phi_t^{(j-1)}$, and set $\phi^{(j)}$ to be the updated flow parameters.

Set the parameters of T to be $\phi^{(j)}$.

end

end

loop of each outer loop iteration to form an estimate for (3.2.2), with which we run an iteration of a gradient descent method on to train the time-embedded flow parameters on. In other words, we define an estimate $\tilde{H}$ of the objective as

$$\begin{aligned} \tilde{H}^{(j)}(\phi) &= \sum_{t=1}^{n} \sum_{i=1}^{N} W_{t-1}^{(i),(j)} h_t(X_{t-1}^{(i),(j)}), \\ h_t(x) &= \log f_{t-1}(x) - \log |\det J_{T_t}(x)| - \log f_t(T_t(x)), \end{aligned} \tag{4.3}$$

where $\{W_t^{(i),(j)}, X_t^{(i),(j)}\}_{i=1}^N$ are the weighted particles obtained at iteration t of the inner loop, at the j^{th} iteration of the outer loop. We then run one iteration of a gradient descent method to optimize $\tilde{H}^{(j)}(\phi)$ using the parameters $\phi^{(j-1)}$ to get the updated parameters $\phi^{(j)}$.

4.5 Adaptive Time Embedded Annealed Flow Transport

In [ZJA16], a variant of SMC that uses an on-line method of choosing the annealing temperatures adaptively was proposed, where each annealing temperature is chosen such that each pair of consecutive bridging distributions will have a common level of discrepancy. We extend this concept to account for a flow component, and develop an analogous variant for TE-AFT, which we call adaptive TE-AFT. This section serves to briefly introduce the method proposed by [ZJA16] and present the details of adaptive TE-AFT.

4.5.1 Adaptive sequential Monte Carlo

The idea of adaptively selecting the next bridging distribution based on maintaining the similarity between consecutive bridging distributions throughout an SMC algorithm was first proposed by [JSDT11]. In their method, they quantified the discrepancy between bridging distributions using the ESS criterion [LC98]. In each iteration of the main loop of SMC, the next bridging distribution will be chosen such that the ESS that would be obtained at that bridging distribution will be of a fixed value that is determined beforehand. The ESS at the next potential bridging distribution can be checked using the particles obtained at the current iteration. However, ESS does not accurately quantify the accumulated mismatch (reflected in the weights of the particles) between π_0 and π_n if resampling is not done at every iteration [ZJA16]. In view of this, [ZJA16] proposes an adaptive SMC algorithm that chooses each bridging distribution to preserve the value of the conditional ESS (CESS) criterion instead, which has the form

$$\begin{aligned} CESS_{\beta_t}(\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N) &= \left[\sum_{i=1}^{N} N W_{t-1}^{(i)} \left(\frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N N W_{t-1}^{(j)} \tilde{w}_t^{(j)}} \right)^2 \right]^{-1} \\ &= \frac{N(\sum_{i=1}^N W_{t-1}^{(i)} \tilde{w}_t^{(i)})^2}{\sum_{j=1}^N W_{t-1}^{(j)} (\tilde{w}_t^{(j)})^2}, \\ \tilde{w}_t^{(i)} &= \frac{f_t(X_{t-1}^{(i)})}{f_{t-1}(X_{t-1}^{(i)})}, \end{aligned} \tag{4.4}$$

at the t^{th} iteration of the main loop of SMC. CESS only becomes equal to ESS when resampling is performed in every iteration. Using a geometric annealing schedule, to find the next bridging distribution from iteration t, the bisection method is used to find the appropriate annealing temperature such that the weighted particles obtained at that temperature produces a CESS equal to the fixed value determined beforehand.

4.5.2 Adjustment to CESS

To account for the flow component of TE-AFT, we make a slight adjustment to the CESS criterion by replacing the $w_t^{(i)}$ terms as defined in SMC with the definition used for AFT (and its variants) in (3.4); i.e. we define the criterion

$$\begin{aligned} CESS_{\beta_t}^{(f)}(\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N) &= \frac{N(\sum_{i=1}^N W_{t-1}^{(i)} \tilde{w}_t^{(i)})^2}{\sum_{j=1}^N W_{t-1}^{(j)} (\tilde{w}_t^{(j)})^2}, \\ \tilde{w}_t^{(i)} &= \frac{f_t(T_t(X_{t-1}^{(i)}))|\det J_{T_t}(X_{t-1}^{(i)})|}{f_{t-1}(X_{t-1}^{(i)})}, \\ T_t(\cdot) &= T(\cdot, \beta_{t-1}, \beta_t), \end{aligned} \tag{4.5}$$

where $\text{CESS}_{\beta_t}^{(f)}$ is the CESS at schedule time β_t, accounting for a fixed flow transport map.

4.5.3 Description of algorithm

Adaptive TE-AFT adds an additional search step on top of the TE-AFT algorithm, where the algorithm searches for the schedule time for the next iteration of the main loop so that the $\text{CESS}^{(f)}$ is maintained close to a pre-determined value using the bisection method. This occurs at the start of each iteration of the main loop of TE-AFT. We focus on the details of the new search step here, with the full algorithm described in algorithm 7.

Search step At iteration t, we start with the weighted particles $\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N$. This allows us to compute $\text{CESS}_\alpha^{(f)}$ for any $\alpha \in [\beta_{t-1}, 1]$, which lets us use the bisection method to search for a β_t starting from the interval $[\beta_{t-1}, 1]$ such that $\text{CESS}_{\beta_t}^{(f)}(\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^N) = S_{CESS}$, where $S_{CESS} \in [1, N]$ is a pre-determined threshold. Should the solution found from the bisection method search be larger than 1, we will set $\beta_t = 1$, ending the TE-AFT algorithm after the current iteration completes. Note that $\text{CESS}_\alpha^{(f)}$ is only equal to CESS_α for annealing temperature $\alpha \in [\beta_{t-1}, 1]$ if the time-embedded flow is fixed. This means that during the search for the next annealing temperature, the values of $\text{CESS}_\alpha^{(f)}$ evaluated in the bisection step are the $\text{CESS}_\alpha^{(f)}$ values where the flow parameters do not change. Thus, the final $\text{CESS}_{\beta_t}^{(f)}$ will typically be different from that as computed in the search step, as the flow parameters are trained and updated within the next iteration's transport step. While this means that the final $\text{CESS}^{(f)}$ values are not kept constant in the end, it allows the adaptive temperature selection mechanism to make a choice of annealing temperatures such that the discrepancy between the previous and current bridging distributions is maintained at the start of each training loop for the flow parameters (in the transport step of

Algorithm 7: Adaptive Time Embedded Annealed Flow Transport

Input: Number of training, validation and test particles N_{train}, N_{val}, N_{test}, resampling thresholds $S_{train} \in (1, N_{train})$, $S_{val} \in (1, N_{val})$, $S_{test} \in (1, N_{test})$, initial flow parameters ϕ_0, number of training iterations J, and CESS$^{(f)}$ threshold S_{CESS}.

Output: Approximations $(\hat{\pi}_n, \hat{Z}_n)$ to (π_n, Z_n).

Initialization step:

Initialize $\{W_0^{(i),a}, X_0^{(i),a}\}_{i=1}^{N_a}$, where $X_0^{(i),a} \sim \pi_0$ and $W_0^{(i),a} = \frac{1}{N_a}$ for $i \in \{1, 2, ..., N_a\}$, for $a \in \{train, val, test\}$, evidence estimate $\hat{Z}_0 = Z_0$, $t = 0$ and $\beta_0 = 0$.

while $\beta_t < 1$ **do**

Set $t = t + 1$.

Search step:

Use the bisection method to find α^* such that $\text{CESS}_{\alpha^*}^{(f)}(\{W_{t-1}^{(i)}, X_{t-1}^{(i)}\}_{i=1}^{N}) = S_{CESS}$ in the interval $[\beta_{t-1}, 1]$ and set $\beta_t = \min(\alpha^*, 1)$.

Transport step:

Set $\phi_t^{(0)} = \phi_{t-1}$.

for $j = 1, 2, ..., J$ **do**

Perform one iteration of a gradient descent method to minimize $\tilde{\mathcal{L}}_t(\phi, \{W_{t-1}^{(i),train}, X_{t-1}^{(i),train}\}_{i=1}^{N})$ (as redefined in 4.3.2) over ϕ using $\phi_t^{(j-1)}$.

Set $\phi_t^{(j)}$ to be the flow parameters obtained from the gradient descent iteration.

end

Set the parameters of T_t to be $\phi_t = \arg\min_\phi \tilde{\mathcal{L}}_t(\phi_t^{(j)}, \{W_{t-1}^{(i),val}, X_{t-1}^{(i),val}\}_{i=1}^{N})$.

Set $\tilde{X}_{t-1}^{(i),a} = T_t(X_{t-1}^{(i),a})$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Importance sampling step:

Set $\tilde{w}_t^{(i),a} = \frac{f_t(\tilde{X}_{t-1}^{(i),a}) |\det J_{T_t}(X_{t-1}^{(i),a})|}{f_{t-1}(X_{t-1}^{(i),a})}$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Set $W_t^{(i),a} = \frac{W_{t-1}^{(i),a} \tilde{w}_t^{(i),a}}{\sum_{j=1}^{N_a} W_{t-1}^{(j),a} \tilde{w}_t^{(j),a}}$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

Set $\hat{Z}_t = \hat{Z}_{t-1} \sum_{i=1}^{N_{test}} W_{t-1}^{(i),test} \tilde{w}_t^{(i),test}$.

Conditional resampling step:

for $a \in \{train, val, test\}$ **do**

Compute $ESS_t^a = \left\{ \sum_{i=1}^{N_a} \left(W_t^{(i),a} \right)^2 \right\}^{-1}$.

if $ESS_t^a < S_a$ **then**

Resample $\tilde{X}_{t-1}^{(1),a}, ..., \tilde{X}_{t-1}^{(N_a),a} \sim \text{Multinomial}(\{\tilde{X}_{t-1}^{(i),a}\}_{i=1}^{N_a}; \{W_t^{(i),a}\}_{i=1}^{N_a})$.

Set $W_t^{(1),a} = W_t^{(2),a} = ... = W_t^{(N_a),a} = \frac{1}{N_a}$.

end

end

Mutation step:

Draw $X_t^{(i),a} \sim K_t(\tilde{X}_{t-1}^{(i),a}, \cdot)$ for $i = 1, 2, ..., N_a$, for $a \in \{train, val, test\}$.

end

each iteration). This can facilitate training if S_{CESS} is tuned such that the discrepancy between each successive bridging distribution is similar enough that the transport map to be learned is simple and achievable by the flow.

4.5.4 Incompatibility with AFT

Note that this adaptive variant may be inapplicable to AFT, as the flow parameters do not carry over to the next iteration; if a bisection method were used to find the next annealing temperature α such that $\text{CESS}_\alpha^{(f)} = S_{CESS}$, the $\text{CESS}_\alpha^{(f)}$ values evaluated during the bisection method would reflect the $\text{CESS}_\alpha^{(f)}$ if the flow used was identical to the final trained flow in the current iteration, which would not be accurate at all since the flow parameters are reset at the start of each iteration and a completely new flow transport would be learned. An analogous variant for AFT would thus have to use the original CESS defined by [ZJA16], which would not acknowledge and tap on the effects of the flow component in AFT, unlike $\text{CESS}^{(f)}$ and TE-AFT. Nonetheless, we still test the performance of applying the $\text{CESS}^{(f)}$ adaptive variant on AFT in our experiments as described in the next section.

5 Experiments

To empirically investigate the performance of the proposed TE-AFT and TE-CRAFT, we conducted simulation experiments where the sampling algorithms are trained to draw samples from challenging target distributions. We use the evidence estimates obtained from the drawn samples to judge the quality of the sampled particles, where it is naturally desirable to obtain evidence estimates as close to the true value of the estimate as possible. The target distributions used in the experiments are Neal's funnel distribution and the log Gaussian Cox process, both of which we will describe in the following subsections. AFT, CRAFT and SMC are also used in the experiments, as baselines with which we can compare the performance of TE-AFT and TE-CRAFT to.

5.1 Neal's funnel

Neal's funnel [Nea03] is a ten-dimensional distribution with a narrow 'funnel-like' geometry that bends into one direction. An $x_{0:9} \in \mathbb{R}^{10}$ drawn from Neal's funnel has the distribution

$$x_0 \sim N(0, \sigma_f^2), \quad x_{1:9}|x_0 \sim N_{10}[\mathbf{0}, \exp(x_0)\mathbf{I}], \tag{5.1}$$

where we set $\sigma_f^2 = 9$ in our experiments. It has proven to be challenging for many MCMC algorithms, due to the variety of length scales (that depend on the value x_0), as well as the heavy tails of the marginal distribution of $x_{1:9}$ [AMD21]. In our experiments, we used the normalized density of Neal's funnel, and so the true evidence value is 0.

5.2 Log Gaussian Cox process

The target distribution described here refers to the log Gaussian Cox process used in [MSW98] to model the positions of pine saplings in a forest in Finland. We discretize the modelled 'forest' area into a 32×32 grid, resulting in the (unnormalized) density function

$$f(x) = N_{1024}(x; \mu, V) \prod_{i \in [1:32]^2} \exp(x_i y_i - a \exp(x_i)), \tag{5.2}$$

where μ is a constant vector of $\log(126) - 1.91^2$, covariance $V(u, v) = 1.91^2 \exp\left(\frac{33}{32}||u - v||_2\right)$, $a = \frac{1}{1024}$, and y_i is the number of pine saplings in grid $i \in [1 : 32]^2$ as recorded in the data obtained by [MSW98]. These values are used to match with the model fitted in [MSW98] and the log Gaussian Cox process experiments run in [AMD21] and [MARD22]. While we did not have access to the exact value of the true evidence, we were able to obtain a 'gold standard' estimate using the Cholesky whitening of the distribution (which makes the distribution trivial for SMC to sample from), and taking the average evidence estimate obtained from 200 SMC runs using 100 annealing temperatures, similar to how [AMD21] acquired their gold standard evidence estimate for this log Gaussian Cox process.

5.3 Experiment configuration

In all our experiments, each sampler is run 200 times, with resampling threshold at 0.3 times the number of evaluation particles. We draw from the multivariate standard normal as the initial distribution π_0, and use 1 iteration of the Hamiltonian Monte Carlo (HMC) [N+11] with 10 leapfrog steps as the Markov kernel. The HMC step sizes used follow a schedule provided by the code implementation of [AMD21] and [MARD22], where the step sizes to use at a given temperature is obtained by linearly interpolating within a set of step sizes tuned to maintain a reasonable acceptance probability on preliminary runs of SMC on each target distribution for some fixed number of temperatures. For Neal's funnel, the HMC step size schedule use the interpolation points (for annealing temperature) $[0, 0.25, 0.5, 0.75, 1]$ with corresponding step sizes $[0.9, 0.7, 0.6, 0.5, 0.4]$, whereas the log Gaussian Cox process used interpolation points $[0, 0.25, 0.5, 1]$ and corresponding step sizes $[0.3, 0.3, 0.2, 0.2]$. RealNVP [DSDB16] with two affine coupling layers is used for the flows in AFT and CRAFT, while TE-AFT and TE-CRAFT use a RealNVP flow with two time-embedded affine coupling layers (see figure 2). For a number of evaluation particles N, we maintain a budget for the memory used to store particles for training purposes by allowing CRAFT and TE-CRAFT to generate N particles for each training iteration, and assigning AFT and TE-AFT (and its adaptive variant) $N/2$ training and $N/2$ validation particles. The ADAM optimizer [KB17] was used for all neural network training involved. All of the experiments are implemented using Python and the Jax [BFH+18], Flax [HLO+23] and Optax [DBB+20] libraries, where we borrow much of the implementation of AFT, CRAFT and the Log Gaussian Cox process target distribution from [AMD21] and [MARD22] from their repository[2]. For the purpose of recording compute times, the experiments were run on an AMD Ryzen 5 5600X 6-Core processor, and the average

[2] https://github.com/google-deepmind/annealed_flow_transport

times over 3 runs were recorded. Full details of the experiment setup can be found within the `configs` directory of our code implementation[3].

5.4 Results and discussion

Two types of experiments are carried out sampling from each of the target distributions; the fixed-temperature experiment, and the adaptive experiment. This section explains each type of experiment, and discusses the results obtained from them.

5.4.1 Fixed-temperature experiments

We first test the performance of the sampling algorithms under a fixed number of annealing temperatures that are linearly spaced in the interval $[0, 1]$. The sampling algorithms implemented and assessed here are SMC, practical AFT, CRAFT, TE-AFT and TE-CRAFT. As done in [AMD21] and [MARD22], practical AFT is used in place of AFT as it is more robust, avoids overfitting the flows, and provides unbiased evidence estimates. Just like in those studies, we use the number of temperatures as a proxy for computational effort, so that we may observe the performance of the sampling algorithms when given varying levels of computational resources, while still having a common basis for comparison. However, we would also like to point out that a sampler does not necessarily perform better under a larger number of temperatures. Indeed, as we will observe in the experiment results later, a more optimal choice of temperatures can be obtained using a small number of temperatures. In fact, the choice of temperatures in this experiment (linearly spaced in $[0, 1]$ for some number of temperatures) do not guarantee an optimal choice of temperatures in any way. We thus treat the results obtained here to reflect the performance of the samplers under poor choices of annealing schedule times.

Neal's funnel In the first fixed-temperature experiment using Neal's funnel as the target distribution, 2000 particles were sampled from the target distribution using each sampling algorithm. AFT, TE-AFT, CRAFT, and TE-CRAFT trained for 200 training iterations. The results of this experiment are shown in figure 4. The compute time of each sampler is tabulated in table 1.

Note that the compute time required for CRAFT and TE-CRAFT is substantially higher than SMC, AFT and TE-AFT. This is due to CRAFT and TE-CRAFT propagating particles through the Markov kernel J times as much as the other algorithms for J training iterations. This can be problematic, since propagation of the particles through the Markov kernel can be very costly, especially when sampling from extremely complicated target distributions with densities that are already expensive to evaluate on their own, as Markov kernels typically require multiple evaluations of the target density. A fairer experiment yielding more practical results can be conducted by providing SMC, AFT and TE-AFT with more resources so that the compute time used across all algorithms will be more equal. This would indicate a more equal cost in terms of computational resources across the sampling algorithms. We thus repeat the fixed-temperature experiment with a more

[3] https://github.com/stevvseah/fyp_differentiable_samplers

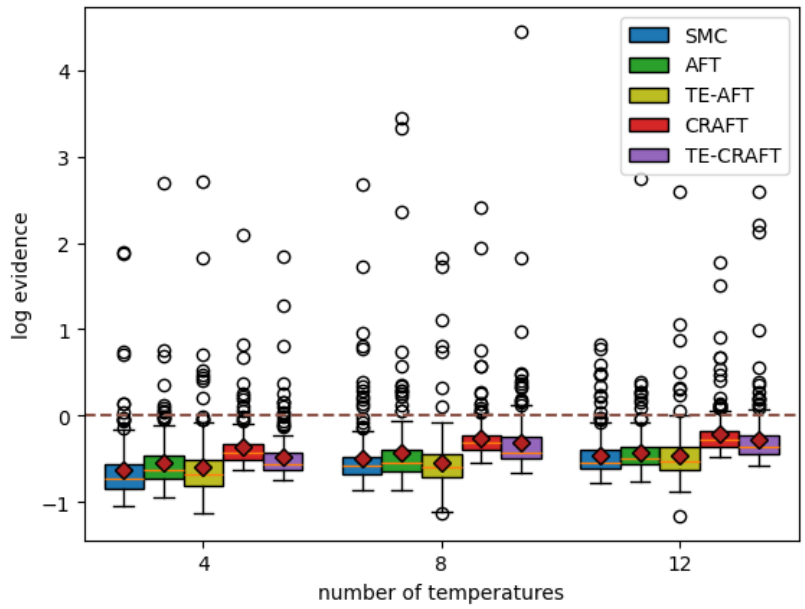

Figure 4: Fixed-temperature experiment on the Neal's funnel target, using 2000 evaluation particles and 200 training iterations. Orange lines denote the median and red diamonds denote the mean. The brown dashed line marks the true value of the evidence, which is 0.

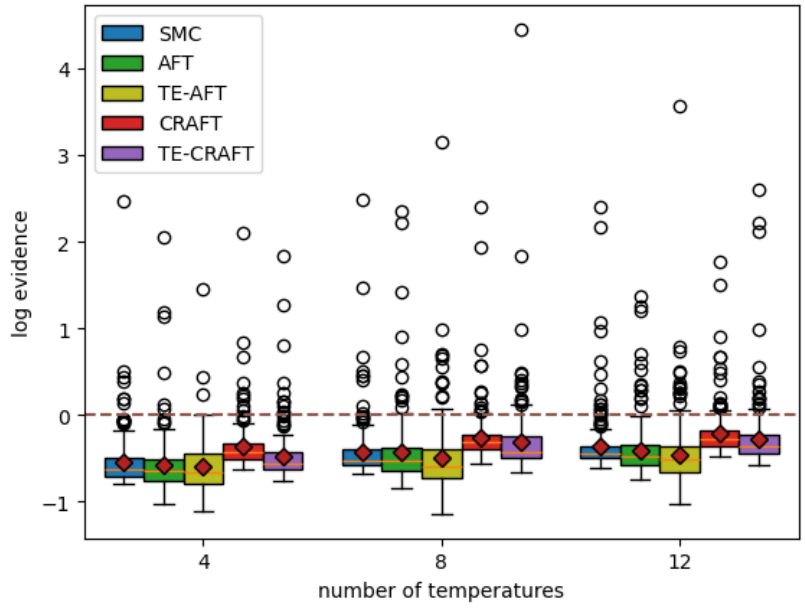

Figure 5: Fixed-temperature experiment on the Neal's funnel target, with adjusted SMC, AFT and TE-AFT. Orange lines denote the median and red diamonds denote the mean. The brown dashed line marks the true value of the evidence, which is 0.

equitable configuration, giving SMC 12 times more particles, and AFT and TE-AFT 10 times more training iterations. The results of this experiment are plotted in figure 5.

We observe that TE-AFT and TE-CRAFT perform very similarly to their non-time-embedded counterparts, if not slightly worse. This suggests that swapping out the NFs in each iteration for a single time-embedded flow may harm the quality of the generated particles in AFT and CRAFT. However, this slight decrease in performance may be an acceptable trade-off to scale the sampling algorithms for a large number of temperatures when memory is a concern, since TE-AFT and TE-CRAFT do not scale at all with the number of temperatures used.

Neal's funnel experiment compute times

n. temps	SMC	AFT	TE-AFT	adj. SMC	adj. AFT	adj. TE-AFT	CRAFT	TE-CRAFT
4	2.06	10.03	11.06	20.86	15.52	17.04	20.09	19.68
8	2.13	10.60	11.76	24.66	21.16	23.71	25.51	25.40
12	2.18	11.89	12.70	25.45	27.11	29.71	27.74	27.86

Table 1: Compute times (in seconds) used by each sampling algorithm for the tested number of temperatures in the fixed-temperature experiment conducted on the Neal's funnel target. The timings are averaged over 3 runs. The adjusted samplers (e.g., adj. SMC) refer to the samplers (of the corresponding name) that were given more computational resources.

Log Gaussian Cox process Similar to the above experiment using Neal's funnel, we first use an equal number of evaluation particles and training iterations across all the samplers. For our experiments on the log Gaussian Cox process target distribution, we used 200 evaluation particles and 100 training iterations. Figure 6 displays a box plot of the results, and table 2 shows an estimate of the compute time used by each sampler. To match the compute time used by CRAFT and TE-CRAFT, we repeat the experiment, giving SMC 60 times more particles, and AFT and TE-AFT 3 times more training iterations. The results are shown in figure 7.

TE-AFT and TE-CRAFT also seem to exhibit similar performance to AFT and CRAFT under the Cox process target distribution. However, we observe that the performance of CRAFT noticeably deteriorates as the number of temperatures increase, while TE-CRAFT improves. This is possibly due to the larger number of flow parameters that have to be trained in CRAFT as compared to TE-CRAFT as more temperatures are used – this could indicate that TE-CRAFT is able to use the paltry 100 training iterations more efficiently than CRAFT. These results demonstrate the robustness of TE-CRAFT to poor choices of annealing temperatures as compared to CRAFT, which suggests that TE-CRAFT could prove to be easier to tune when it comes to the annealing schedule.

5.4.2 Adaptive experiments

In the adaptive experiments, we implement the adaptive variant of SMC described in [ZJA16], as well as (practical) AFT and TE-AFT analogues of this adaptive SMC algorithm and assess the ac-

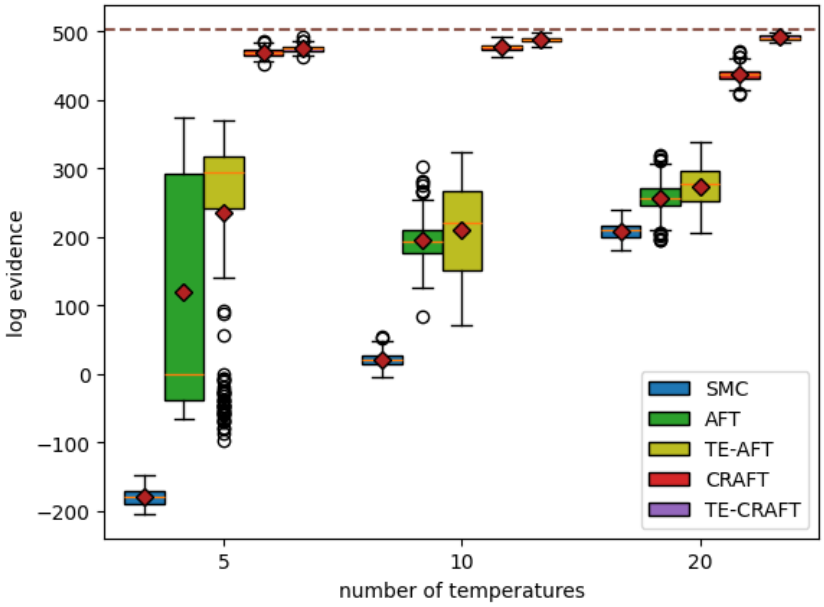

Figure 6: Results of fixed-temperature experiment on the Cox process target, using 200 evaluation particles and 100 training iterations. Orange lines denote the median, and red diamonds denote the mean. The brown dashed line marks a gold-standard estimate of the evidence at 503.14.

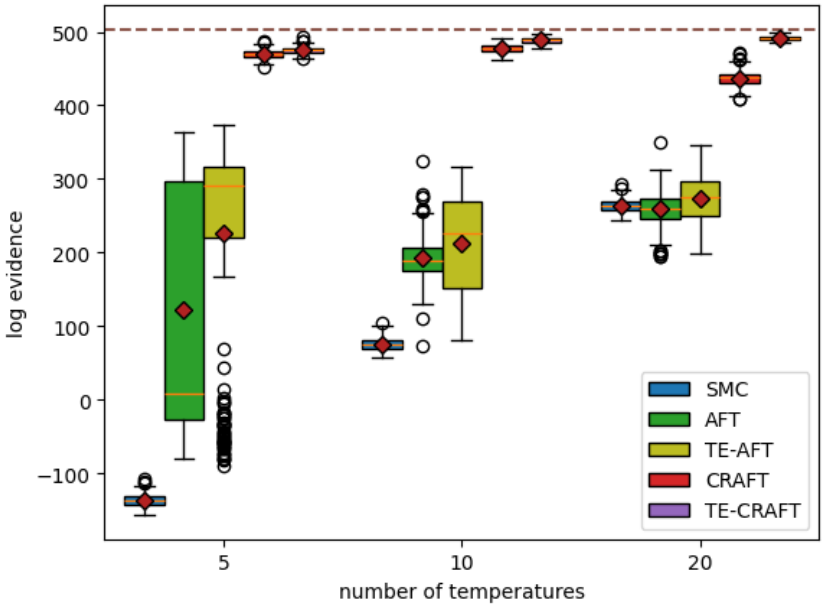

Figure 7: Results of fixed-temperature experiment on the Cox process target, with adjusted SMC, AFT and TE-AFT. Orange lines denote the median, and red diamonds denote the mean. The brown dashed line marks a gold-standard estimate of the evidence at 503.14.

Log Gaussian Cox Process experiment compute times

n. temps	SMC	AFT	TE-AFT	adj. SMC	adj. AFT	adj. TE-AFT	CRAFT	TE-CRAFT
5	2.42	15.63	16.86	21.92	26.37	27.11	23.66	20.69
10	2.63	21.19	22.74	40.84	41.69	43.02	43.77	42.23
20	2.84	32.24	33.91	79.43	75.04	75.39	87.74	81.98

Table 2: Compute times (in seconds) used by each sampling algorithm for the tested number of temperatures in the fixed-temperature experiment conducted on the Cox process target. The timings are averaged over 3 runs. The adjusted samplers (e.g., adj. SMC) refer to the samplers (of the corresponding name) that were given more computational resources.

curacy of the evidence estimates produced by each algorithm. Two adaptive variants of practical AFT were implemented, which we will refer to as AFT(f) and AFT(n). AFT(f) is simply an adjustment of adaptive TE-AFT that returns to using separate flows (like in AFT) at each annealing temperature instead of the time-embedded flow shared across all temperatures, using the CESS$^{(f)}$ criterion to search for annealing temperatures. AFT(n) is the same algorithm as AFT(f), except that CESS is used instead of CESS$^{(f)}$ to adaptively select temperatures. AFT(f) and AFT(n) are implemented to investigate the effects of using the CESS$^{(f)}$ criterion when a new flow is used for each iteration instead of using the same time-embedded flow in all iterations. Each sampling algorithm is run for varying CESS thresholds (the target CESS or CESS$^{(f)}$ for the selection of temperatures). In all adaptive experiments, 8 iterations of the bisection method are used in each search step.

Neal's funnel For the adaptive experiment on the Neal's funnel target, we give SMC 24000 particles, and the AFT(n), AFT(f), and TE-AFT samplers 2000 particles and 2000 training iterations, i.e. the configuration from the fixed-temperature experiment where SMC, AFT and TE-AFT use roughly the same amount of compute time for the tested number of temperatures. A box plot of the results are shown in figure 8, and we plot the corresponding number of temperatures used in figure 9.

At CESS thresholds $0.2N$ and $0.5N$, where N is the number of evaluation particles for the sampler, all adaptive samplers seem to perform similarly to each other. At a CESS threshold of 0.9, adaptive TE-AFT is able to significantly outdo all the other samplers. We find it remarkable that adaptive TE-AFT is able to consistently find annealing schedules that only contain 2 annealing temperatures (excluding 0, which represents the initial distribution) throughout all the CESS thresholds used in the experiment. This suggests that the combination of the time-embedded flow and the use of CESS$^{(f)}$ in place of CESS allows the adaptive temperature selection mechanism to choose annealing temperatures that greatly complement the increased expressiveness (over adaptive SMC) afforded by the NF component. Adaptive TE-AFT thus seems to make the best use of NFs as compared to both adaptive AFT algorithms. We also find it interesting to note that the AFT(f) algorithm is able to match the performance of AFT(n), which might suggest that the use of CESS$^{(f)}$ can still result in decent choices of annealing temperatures, even when the

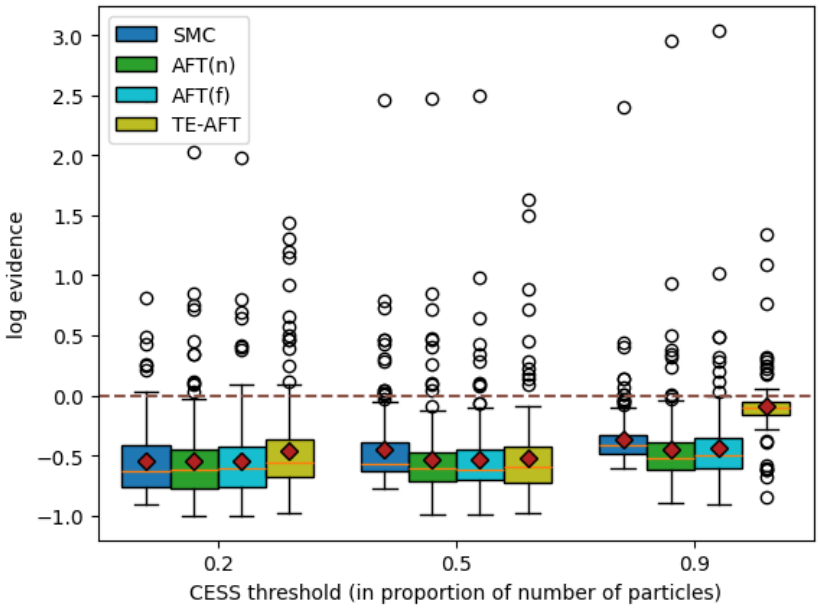

Figure 8: Results of the adaptive experiment on the Neal's funnel target. Orange lines denote the median and red diamonds denote the mean. The brown dashed line marks the true value of the evidence, which is 0.

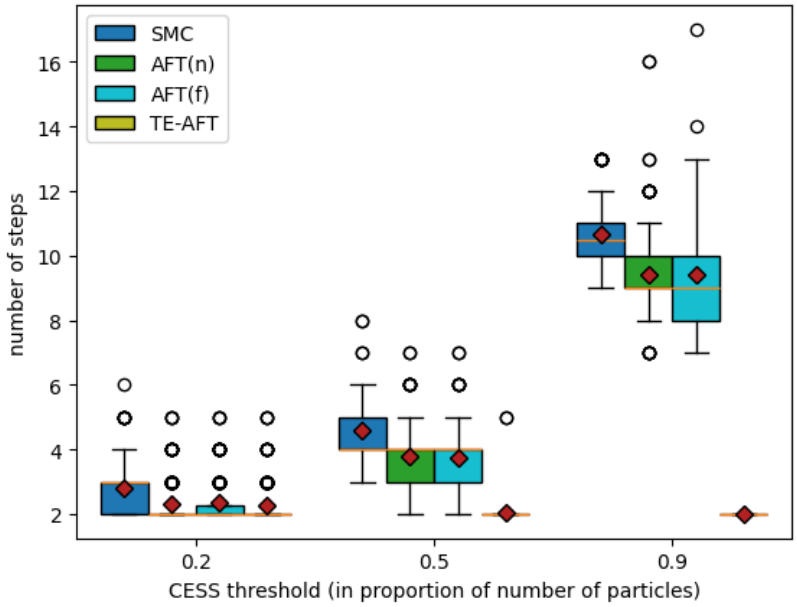

Figure 9: Number of annealing temperatures used in the adaptive experiment on the Neal's funnel target. Orange lines denote the median and red diamonds denote the mean.

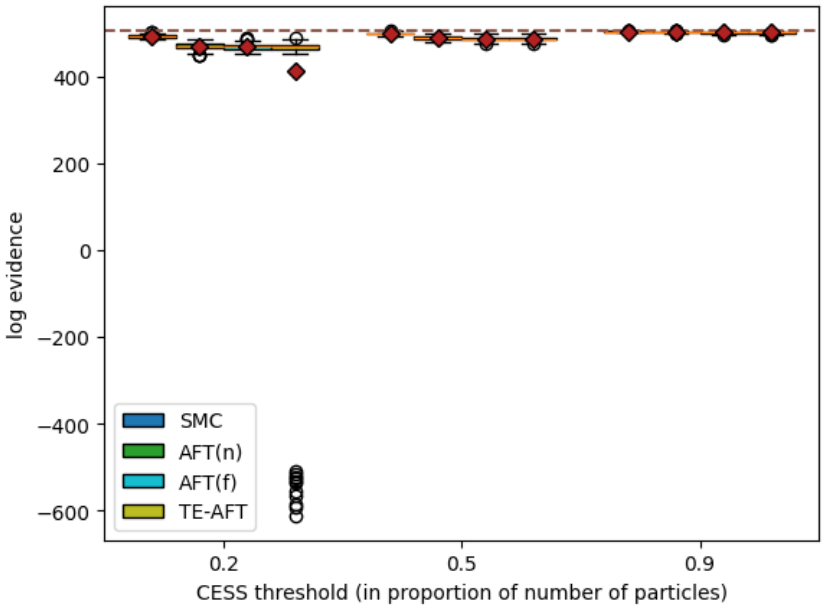

Figure 10: Results of the adaptive experiment on the Cox process target. Orange lines denote the median and red diamonds denote the mean. The brown dashed line marks a gold-standard estimate of the evidence at 503.14.

flow parameters are reset in the next iteration.

Log Gaussian Cox process For the Cox process target distribution, SMC is given 2000 particles, and AFT(n), AFT(f) and TE-AFT are given 200 evaluation particles with 300 training iterations. The results of this experiment are displayed in figure 10, with corresponding box plot of number of temperatures used shown in figure 12. We zoom in on the box plots in figure 11, where only the lower outliers of adaptive TE-AFT at CESS threshold $0.2N$ cannot be seen.

Adaptive TE-AFT seems to occasionally break down at the CESS threshold of $0.2N$, where a very negative log evidence is obtained using 6 annealing temperatures. This could just be due to the poorer choices of annealing temperatures made at lower CESS thresholds. Otherwise, all samplers seem to perform well in this experiment at all CESS thresholds. It seems that adaptive TE-AFT is again able to perform similarly to the other samplers while generally using less temperatures. We also observe that adaptive SMC seems to be able to generate more accurate and consistent log evidence estimates than the other samplers, although it ends up using the most number of temperatures. This could be just a matter of the CESS thresholds corresponding to different levels of discrepancy of consecutive bridging distributions between different sampling algorithms; perhaps a slightly larger CESS threshold could yield similar performance in the AFT-based samplers using more annealing temperatures, matching the amount used in adaptive SMC.

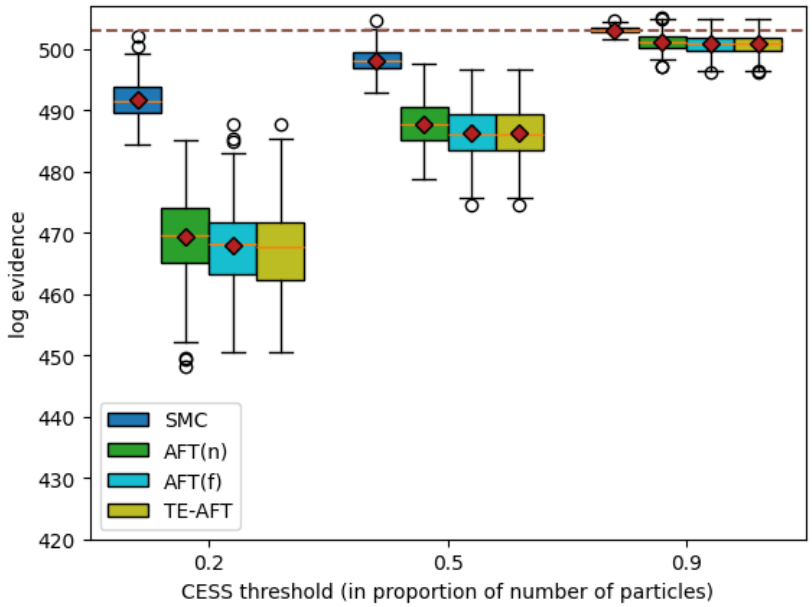

Figure 11: Results of the adaptive experiment on the Cox process target, where we zoom in on the plots. Orange lines denote the median and red diamonds denote the mean. The brown dashed line marks a gold-standard estimate of the evidence at 503.14.

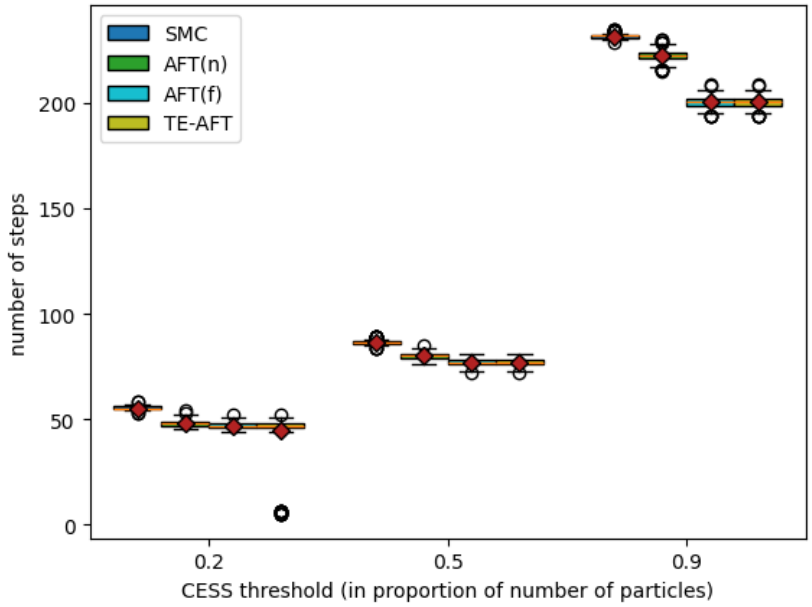

Figure 12: Number of annealing temperatures used in the adaptive experiment on the Cox process target. Orange lines denote the median and red diamonds denote the mean.

6 Conclusion

In this thesis, we briefly review and analyze SMC, AFT, CRAFT, and the relevant sampling techniques that they are built upon. We proposed time-embedded variants of AFT and CRAFT, as well as an adaptive variant of the time-embedded AFT that uses an on-line mechanism to adaptively determine the annealing schedule. Simulation experiments revealed that the adaptive TE-AFT is able to make better choices of annealing temperatures that synergize well with the training scheme of the time-embedded flow, as compared to other AFT analogues of adaptive SMC. TE-CRAFT performs similarly to its default counterpart, contributing a more space-efficient implementation of CRAFT that is more robust to poor choices of annealing temperatures, as the number of flow parameters it uses is independent from the number of annealing temperatures used.

7 Acknowledgements

This thesis would not have been possible without A/P Alexandre Thiery. Under his guidance, studying and implementing all of the sampling algorithms presented in this thesis was intuitive and meaningful. The independent-learning and research skills he emphasises and imparts in his pedagogy were integral to the development of this thesis and are deeply appreciated.

References

[Ada13] Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arxiv:1302.5125*, 2013.

[ADH10] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(3):269–342, 2010.

[AM20] Ömer Deniz Akyildiz and Joaquín Míguez. Nudging the particle filter. *Statistics and Computing*, 30:305–330, 2020.

[AMD21] Michael Arbel, Alex Matthews, and Arnaud Doucet. Annealed flow transport monte carlo. In *International Conference on Machine Learning*, pages 318–330. PMLR, 2021.

[APSAS17] S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. M. Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32(3):405–431, 2017.

[BFH+18] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[BJKT16] Alexandros Beskos, Ajay Jasra, Nikolas Kantas, and Alexandre Thiery. On the convergence of adaptive sequential Monte Carlo methods. *The Annals of Applied Probability*, 26(2):1111 – 1146, 2016.

[BKM05] Vladimir Igorevich Bogachev, Aleksandr Viktorovich Kolesnikov, and Kirill Vladimirovich Medvedev. Triangular transformations of measures. *Sbornik: Mathematics*, 196(3):309, 2005.

[CCDD20] Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International conference on machine learning*, pages 2133–2143. PMLR, 2020.

[CD18] Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

[Cho02] Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

[CMR09] Olivier Cappé, Eric Moulines, and Tobias Rydén. Inference in hidden markov models. In *Proceedings of EUSFLAT conference*, pages 14–16, 2009.

[DBB+20] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[DBMP19] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

[DDT19] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in neural information processing systems*, 32, 2019.

[DHJW22] Chenguang Dai, Jeremy Heng, Pierre E Jacob, and Nick Whiteley. An invitation to sequential monte carlo samplers. *Journal of the American Statistical Association*, 117(539):1587–1600, 2022.

[DKB14] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[DSDB16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[DSDLP19] Laurent Dinh, Jascha Sohl-Dickstein, Hugo Larochelle, and Razvan Pascanu. A rad approach to deep mixture models. *arXiv preprint arXiv:1903.07714*, 2019.

[ECMAW20] Richard G Everitt, Richard Culliford, Felipe Medina-Aguayo, and Daniel J Wilson. Sequential monte carlo with transformations. *Statistics and computing*, 30:663–676, 2020.

[FS02] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Academic Press San Diego, 2002.

[Gil08] J. Gill. *Bayesian Methods: A Social and Behavioral Sciences Approach, Second Edition*. Chapman & Hall/CRC Statistics in the Social and Behavioral Sciences. Taylor & Francis, 2008.

[GM98] Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185, 1998.

[GSS93] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140:107–113(6), April 1993.

[Has70] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 04 1970.

[HDC20] Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv preprint arXiv:2002.07101*, 2020.

[HDP21] Jeremy Heng, Arnaud Doucet, and Yvo Pokern. Gibbs flow for approximate transport with applications to bayesian computation. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83(1):156–187, 2021.

[HLO+23] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023.

[HM69] Johannes Edmund Handschin and David Q Mayne. Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International journal of control*, 9(5):547–559, 1969.

[HR19] Jonathan H. Huggins and Daniel M. Roy. Sequential Monte Carlo as approximate sampling: bounds, adaptive resampling via ∞-ESS, and an application to particle Gibbs. *Bernoulli*, 25(1):584 – 622, 2019.

[Jar97] Christopher Jarzynski. Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78(14):2690, 1997.

[JSDT11] Ajay Jasra, David A Stephens, Arnaud Doucet, and Theodoros Tsagaris. Inference for lévy-driven stochastic volatility models via adaptive sequential monte carlo. *Scandinavian Journal of Statistics*, 38(1):1–22, 2011.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[KBTB14] Dirk P. Kroese, Tim Brereton, Thomas Taimre, and Zdravko I. Botev. Why the monte carlo method is so important today. *WIREs Computational Statistics*, 6(6):386–392, 2014.

[Kit93] Genshiro Kitagawa. A monte carlo filtering and smoothing method for non-gaussian nonlinear state space models. In *Proceedings of the 2nd US-Japan joint seminar on statistical time series analysis*, volume 2, pages 110–131, 1993.

[Kit96] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[Kra06] Werner Krauth. *Statistical mechanics: algorithms and computations*, volume 13. OUP Oxford, 2006.

[KW09] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods*. Wiley, 2009.

[LC98] Jun S Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998.

[Liu01] Jun S. Liu. *Monte Carlo strategies in scientific computing*, volume Springer series in statistics. Springer, New York, 2001.

[MARD22] Alex Matthews, Michael Arbel, Danilo Jimenez Rezende, and Arnaud Doucet. Continual repeated annealed flow transport monte carlo. In *International Conference on Machine Learning*, pages 15196–15219. PMLR, 2022.

[MDJ06] 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.

[MSS+23] Laurence Illing Midgley, Vincent Stimper, Gregor N. C. Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap, 2023.

[MSW98] Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log gaussian cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998.

[N+11] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[NB99] Mark EJ Newman and Gerard T Barkema. *Monte Carlo methods in statistical physics*. Clarendon Press, 1999.

[Nea01] Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

[Nea03] Radford M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705 – 767, 2003.

[NLS+19] Christian A Naesseth, Fredrik Lindsten, Thomas B Schön, et al. Elements of sequential monte carlo. *Foundations and Trends® in Machine Learning*, 12(3):307–392, 2019.

[NSPD16] Thi Le Thu Nguyen, François Septier, Gareth W. Peters, and Yves Delignon. Efficient sequential monte-carlo samplers for bayesian inference. *IEEE Transactions on Signal Processing*, 64(5):1305–1319, 2016.

[Num02] Esa Nummelin. Mc's for mcmc'ists. *International Statistical Review*, 70(2):215–240, 2002.

[Owe13] Art B. Owen. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

[PNR+21] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(1), jan 2021.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RC99] Christian P Robert and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

[RM15] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[SC13] Christian Schäfer and Nicolas Chopin. Sequential monte carlo on large binary sampling spaces. *Statistics and Computing*, 23:163–184, 2013.

[SMJ92] Leland Stewart and Perry McCarty Jr. Use of bayesian belief networks to fuse continuous and discrete information for target recognition, tracking, and situation assessment. In *Signal Processing, Sensor Fusion, and Target Recognition*, volume 1699, pages 177–185. SPIE, 1992.

[TT13] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[VJ11] Suriyanarayanan Vaikuntanathan and Christopher Jarzynski. Escorted free energy simulations. *The Journal of chemical physics*, 134(5), 2011.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[WKN20] Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *Advances in Neural Information Processing Systems*, 33:5933–5944, 2020.

[ZJA16] Yan Zhou, Adam M Johansen, and John AD Aston. Toward automatic model comparison: an adaptive sequential monte carlo approach. *Journal of Computational and Graphical Statistics*, 25(3):701–726, 2016.