Asymptotically exact inference in differentiable generative models

Matthew M. Graham and Amos J. Storkey

School of Informatics
University of Edinburgh
e-mail: m.m.graham@ed.ac.uk*;* a.storkey@ed.ac.uk

Abstract: Many generative models can be expressed as a differentiable function applied to input variables sampled from a known probability distribution. This framework includes both the generative component of learned parametric models such as variational autoencoders and generative adversarial networks, and also procedurally defined simulator models which involve only differentiable operations. Though the distribution on the input variables to such models is known, often the distribution on the output variables is only implicitly defined. We present a method for performing efficient Markov chain Monte Carlo inference in such models when conditioning on observations of the model output. For some models this offers an asymptotically exact inference method where approximate Bayesian computation might otherwise be employed. We use the intuition that computing conditional expectations is equivalent to integrating over a density defined on the manifold corresponding to the set of inputs consistent with the observed outputs. This motivates the use of a constrained variant of Hamiltonian Monte Carlo which leverages the smooth geometry of the manifold to coherently move between inputs exactly consistent with observations. We validate the method by performing inference experiments in a diverse set of models.

MSC 2010 subject classifications: Primary 65C05; secondary 62F15.
Keywords and phrases: Generative models, Implicit models, Markov chain Monte Carlo, Approximate Bayesian computation.

Contents

1 Introduction 1
2 Notation 2
3 Problem definition 3
4 Differentiable generative models 3
5 Model parameterisation 5
6 Directed and undirected generative models 6
7 Approximate Bayesian Computation 7
8 Inference in the input space 14
9 Constrained Hamiltonian Monte Carlo 18
10 Implementation details 26
11 Related work 30
12 Numerical experiments 32
13 Discussion 52
References 54

1. Introduction

There has been a long interest in probabilistic models which are defined *implicitly* [10, 40, 27] - that is where we can generate random values for the latent and observed variables in the model, but we cannot tractably evaluate a density function for the probability distribution on those variables. This is in contrast to the more typical case where the probabilistic model of interest is defined by specifying an explicit (potentially unnormalised) probability density function on latent and observed variables, often via a graphical model such as a Bayesian network [74], Markov random field [45] or factor graph [32].

Although implicit models are challenging to work with from an inferential perspective, they are ubiquitous in science and engineering in the form of probabilistic models defined by the computational simulation of a physical or biological system. Typically simulator models are specified procedurally in code with any stochasticity introduced by drawing values from a *pseudo-random number generator* (PRNG). The complexity of the function mapping from random inputs to simulated outputs means that computing a probability density on the outputs is usually at best computationally challenging and often intractable.

Implicit models also arise through use of distributions defined by their quantile function (inverse of the *cumulative distribution function* (CDF)) [42, 95]. Independent samples can be easily generated from such distributions by mapping standard uniform variates through the quantile function. Although these *quantile distributions* can offer very flexible descriptions of shape of a distribution [36] often the quantile function will not have a closed-form inverse meaning their CDF and so density function cannot be evaluated analytically.

Recently implicit generative models have also been the subject of much interest in the machine learning community due to the significant gains in modelling flexibility offered by dropping the requirement to be able to compute an explicit density function on model outputs [62, 93]. For instance *generative-adversarial networks* (GANs) [38] have become a popular approach in unsupervised machine learning for training models which can generate plausible simulated data points, typically images, given a large collection of data points to learn from. The *generator* of a GAN takes the form of differentiable network which receives as input a vector of values drawn from a simple distribution such as the standard normal and outputs for example a generated image. The generator function will typically be non-injective however, meaning that we cannot tractably evaluate the probability density of the generated outputs as this requires integrating over implicitly defined sets of inputs consistent with a particular output.

A lack of an explicit density on the variables in a generative model makes it non-trivial to apply approximate inference approaches such as *Markov chain Monte Carlo* (MCMC) to infer the values of unobserved variables given known values for a set of observed variables in the model. This has spurred the development of inference approaches specifically targeted at implicit generative models such as indirect inference [40] and *approximate Bayesian computation* (ABC) [10].

In both indirect inference and ABC, inferences about plausible values of the unobserved variables are made by computing distances between simulated observed variables and data. At a qualitative level, values of the unobserved variables associated with simulated observations that are 'near' to the data are viewed to be more plausible. This approximation that the simulated observations are only close but not equal to the observed data makes the inference problem more tractable but also biases the inference output. Further distance measures tend to become increasingly less informative as the dimensionality of a space increases, making it difficult to use these approaches to perform inference in models with large numbers of unobserved variables [56].

In this article we describe a *Hamiltonian Monte Carlo* (HMC) [28, 55] method for performing inference in a sub-class of implicitly-defined generative models where the mapping from random inputs to the model to simulated observed and latent variables is differentiable. Unlike existing approaches, this method allows inference to be performed by conditioning the observed variables in the model to be arbitrarily close to data values. This means that subject to the usual conditions on the Markov chain being irreducible and aperiodic, asymptotically exact inference can be performed: conditional expectation estimates computed using the method will converge in the asymptotic limit to their true values. Further by exploiting gradient information the approach is able to perform efficient inference in models with large numbers of unobserved variables and conditioning on all observed data rather than low-dimensional summaries.

2. Notation

We will briefly summarise the notation used in the rest of this article. Lower-case bold-faced characters are used to represent vector quantities, e.g. $\boldsymbol{x}$, with upper case bold-face reserved for matrix quantities, e.g. $\boldsymbol{A}$. The lower triangular Cholesky factor of a positive definite matrix $\boldsymbol{A}$ is $\mathrm{chol}(\boldsymbol{A})$. The determinant of a matrix $\boldsymbol{A}$ is $|\boldsymbol{A}|$. The Euclidean norm of a vector $\boldsymbol{x}$ is $\|\boldsymbol{x}\|_2$ and the infinity norm is $\|\boldsymbol{x}\|_\infty$. Calligraphic characters are used for sets for example $\mathcal{A}$ and script characters for σ-algebras on a set, e.g. $\mathscr{E}$. The Borel σ-algebra on a topological space $\mathcal{X}$ is $\mathfrak{B}(\mathcal{X})$. The indicator function on a set $\mathcal{A}$ is $\mathbb{I}_{\mathcal{A}}$. Sans-serif variants of characters are used for random variables (or vectors), for example $\boldsymbol{\mathsf{x}}$ is a random vector. The expectation of a random variable x is $\mathbb{E}[\mathsf{x}]$ while $\mathbb{E}[\mathsf{x} \,|\, \mathsf{y} = y]$ is the conditional expectation of x given $\mathsf{y} = y$. The distribution of a random variable x is P_x. For two random variables x and y their joint distribution is $\mathsf{P}_{\mathsf{x},\mathsf{y}}$ and the regular conditional distribution on x given $\mathsf{y} = y$ is denoted $\mathsf{P}_{\mathsf{x}|\mathsf{y}}(\cdot \,|\, y)$. The density of a distribution P_x of a real-valued random variable x with respect to a reference measure μ is $\mathsf{p}_\mathsf{x} = \frac{\mathrm{d}\mathsf{P}_\mathsf{x}}{\mathrm{d}\mu}$. The joint density on random variables x and y and conditional density on x given y are similarly denoted by $\mathsf{p}_{\mathsf{x},\mathsf{y}}$ and $\mathsf{p}_{\mathsf{x}|\mathsf{y}}$. The density of a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ at a point $\boldsymbol{x}$ is $\mathcal{N}(\boldsymbol{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. The D-dimensional Hausdorff measure on a metric space is $\mathcal{H}^D$. All integrals without an explicit measure indicated should be assumed to be with respect to the Lebesgue measure λ. The gradient of a function $f : \mathbb{R}^D \to \mathbb{R}$ is $\nabla f : \mathbb{R}^D \to \mathbb{R}^D$ with $[\nabla f(\boldsymbol{x})]_i = \frac{\partial f}{\partial x_i}$ and the Jacobian of a function $\boldsymbol{g} : \mathbb{R}^N \to \mathbb{R}^M$ is $\mathbf{J}_{\boldsymbol{g}} : \mathbb{R}^N \to \mathbb{R}^{M \times N}$ with $[\mathbf{J}_{\boldsymbol{g}}(\boldsymbol{x})]_{m,n} = \frac{\partial g_m}{\partial x_n}$.

3. Problem definition

Let $(\mathcal{S}, \mathscr{E}, \mathsf{P})$ be a probability space, and $(\mathcal{X}, \mathscr{G})$, $(\mathcal{Z}, \mathscr{H})$ be two measurable spaces. We denote the vector of observed random variables in the model of interest as $\mathbf{x} : \mathcal{S} \to \mathcal{X}$ and the vector of unobserved random variables that we wish to infer $\mathbf{z} : \mathcal{S} \to \mathcal{Z}$. Our objective is to be able to compute conditional expectations of arbitrary measurable functions $f : \mathcal{Z} \to \mathcal{F}$ of the unobserved variables given known values for the observed variables $\mathbf{x}$, where the conditional expectation $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x}\,] : \mathcal{X} \to \mathcal{F}$ is defined as a measurable function satisfying

$$\int_{\mathcal{A}} \mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{x}] \, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \int_{\mathcal{A} \times \mathcal{Z}} f(\boldsymbol{z}) \, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) \qquad \forall \mathcal{A} \in \mathscr{G}, \tag{1}$$

with this identity uniquely defining $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x}\,]$ up to $\mathsf{P}_{\mathbf{x}}$-null sets.

In models where the joint distribution $\mathsf{P}_{\mathbf{x},\mathbf{z}}$ is specified by an explicit density $\mathsf{p}_{\mathbf{x},\mathbf{z}}$ with respect to a product measure $\mu_{\mathbf{x}} \times \mu_{\mathbf{z}}$, then we have the standard result that the conditional expectation can be expressed as

$$\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{x}] = \frac{1}{\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})} \int_{\mathcal{Z}} f(\boldsymbol{z}) \, \mathsf{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{z}) \, \mu_{\mathbf{z}}(\mathrm{d}\boldsymbol{z}) \qquad \forall \boldsymbol{x} \in \mathcal{X} : \mathsf{p}_{\mathbf{x}}(\boldsymbol{x}) > 0. \tag{2}$$

Although we typically cannot analytically evaluate this integral or the marginal density $\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})$, having access to the joint density $\mathsf{p}_{\mathbf{x},\mathbf{z}}$ is sufficient to allow us to apply approximate methods such as MCMC and variational inference to estimate conditional expectations. In this article we consider the problem of computing conditional expectations in models where we can generate samples from the joint distribution $\mathsf{P}_{\mathbf{x},\mathbf{z}}$ but we cannot evaluate the joint density $\mathsf{p}_{\mathbf{x},\mathbf{z}}$.

4. Differentiable generative models

Any probabilistic model that we can programmatically generate values from can be expressed in the form of a deterministic function which takes as input a vector of random inputs sampled from a known distribution. This observation just corresponds to stating that we can track all of the calls to a PRNG in a program, and that given the values sampled from the PRNG all of the operations then performed by the program are deterministic[1]. To formalise this idea below we give a concrete definition for what we will consider as constituting a generative model for a set of observed variables $\mathbf{x}$ and unobserved variables $\mathbf{z}$.

Definition 1 (Generative model)**:** *Let* $\mathbf{u} : \mathcal{S} \to \mathcal{U}$ *be a random vector taking on values in a measurable space* $(\mathcal{U}, \mathscr{F})$*. We require that the distribution* $\mathsf{P}_{\mathbf{u}}$ *has a density* $\mathsf{p}_{\mathbf{u}}$ *with respect to a reference measure* μ *which we can evalulate and that it is tractable to generate independent samples from* $\mathsf{P}_{\mathbf{u}}$*. If* $\boldsymbol{g}_{\mathbf{x}} : \mathcal{U} \to \mathcal{X}$ *and* $\boldsymbol{g}_{\mathbf{z}} : \mathcal{U} \to \mathcal{Z}$ *are measurable functions such that*

$$\mathbf{x}(s) = \boldsymbol{g}_{\mathbf{x}} \circ \mathbf{u}(s) \quad \textit{and} \quad \mathbf{z}(s) = \boldsymbol{g}_{\mathbf{z}} \circ \mathbf{u}(s) \quad \forall s \in \mathcal{S} \tag{3}$$

[1]For the purposes of clarity of exposition we consider the outputs of the PRNG as truly random, even though in reality they are deterministically computed.

then we define $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{u}}, \mu, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{g}_{\mathbf{z}})$ *as a generative model for* $\mathbf{x}$ *and* $\mathbf{z}$*. We refer to* $(\mathcal{U}, \mathscr{F})$ *as the* input space *of the generative model,* $(\mathcal{X}, \mathscr{G})$ *the* observed output space *and* $(\mathcal{Z}, \mathscr{G})$ *the* unobserved output space*. Further we term* $\boldsymbol{g}_{\mathbf{x}}$ *as the* generator of $\mathbf{x}$ *and likewise* $\boldsymbol{g}_{\mathbf{z}}$ *the* generator of $\mathbf{z}$*. The random vector* $\mathbf{u}$ *is the* random inputs *and the density* $\mathsf{p}_{\mathbf{u}}$ *the* input density*.*

The input vector $\mathbf{u}$ encapsulates all of the values drawn from a PRNG in the code of a generative model and the generator functions $\boldsymbol{g}_{\mathbf{x}}$ and $\boldsymbol{g}_{\mathbf{z}}$ represent the operations used to generate values for $\mathbf{x}$ and $\mathbf{z}$ respectively given values for the random inputs $\mathbf{u}$. In some cases the number of random inputs used in a generator evaluation will depend on the values of the random inputs themselves, for example if there is a branching statement which depends on a random input and the operations in each branch use different random inputs. Although implementationally more challenging, we can still consider this case within the above definition by enumerating the random inputs required in all possible control flow paths through the generator code and mapping each to a different element in $\mathbf{u}$. In interpreted languages, this can be done lazily by detecting if a call to a PRNG object has occurred at the same point in a execution trace previously and if so matching to same element in $\mathbf{u}$ as used previously otherwise matching to a new $\mathbf{u}$ element.

In this article we concentrate on a restricted class of generative models which we term *differentiable generative models.*

Definition 2 (Differentiable generative model)**:** *Let* $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{u}}, \mu, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{g}_{\mathbf{z}})$ *be a generative model for* $\mathbf{x}$ *and* $\mathbf{z}$ *as specified in Definition 1. Then if the following conditions are satisfied*

1. $\mathcal{U} \subseteq \mathbb{R}^{D_{\mathbf{u}}}$*,* $\mathscr{F} = \mathfrak{B}(\mathcal{U})$ *and* $\mathcal{X} \subseteq \mathbb{R}^{D_{\mathbf{x}}}$*,* $\mathscr{G} = \mathfrak{B}(\mathcal{X})$*,*
2. $\mathsf{P}_{\mathbf{u}}$ *has a density* $\mathsf{p}_{\mathbf{u}}$ *with respect to the Lebesgue measure* $\mu = \lambda^{D_{\mathbf{u}}}$*,*
3. *the input density gradient* $\nabla \mathsf{p}_{\mathbf{u}}$ *exists* $\mathsf{P}_{\mathbf{u}}$*-almost everywhere,*
4. *the generator Jacobian* $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ *exists* $\mathsf{P}_{\mathbf{u}}$*-almost everywhere.*

we describe $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{u}}, \mu, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{g}_{\mathbf{z}})$ *as a differentiable generative model.*

These requirements are quite severe: for example they exclude any models with discrete random inputs and those in which branch statements in the generator code introduce discontinuities. This means the proposed method is not for instance applicable to models with discrete latent variables which are commonly the target of existing ABC applications. However there are still a large class of interesting models which do meet these conditions: for example simulator models based on approximate integration of *ordinary differential equations* (ODEs) (combined with a stochastic observation model) or *stochastic differential equation* (SDE) models without a jump-process component. Similarly as differentiability is usually a requirement for training the generative models used in machine learning, many such models will also fall in to this class.

A further restriction we will typically assume is that the Jacobian $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ is full row-rank $\mathsf{P}_{\mathbf{u}}$-almost everywhere, which also necessarily means that $D_{\mathbf{u}} \geq D_{\mathbf{x}}$ i.e the number of random inputs is at least as many as the number of observed variables that will be conditioned on. In cases where this does not hold the implicitly defined probability distribution $\mathsf{P}_{\mathbf{x}}$ will not be absolutely continuous with respect to the $D_{\mathbf{x}}$-dimensional Lebesgue measure on $\mathcal{X}$. Instead $\mathsf{P}_{\mathbf{x}}$ will only have support on a sub-manifold of dimension locally equal to the rank of $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ and conditioning on arbitrary $\boldsymbol{x} \in \mathcal{X}$ is not a well-defined operation.

Although we only required the existence of the input density gradient $\nabla \mathsf{p}_{\mathbf{u}}$ and generator Jacobian $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ above, unsurprisingly this is motivated by the need to evaluate these terms in the proposed method. Although this may seem a limiting requirement for complex models, the availability of efficient general-purpose *automatic differentiation* (AD) libraries [8] means it is possible to automatically calculate the necessary derivatives given just the code defining the forward functions $\mathsf{p}_{\mathbf{u}}$ and $\boldsymbol{g}_{\mathbf{x}}$. For generative models implemented in existing code this will typically require re-coding using an appropriate AD framework.

When applying reverse-mode AD [89, 54] to a function $\boldsymbol{h} : \mathbb{R}^K \to \mathbb{R}^L$ the Jacobian $\mathbf{J}_{\boldsymbol{h}}$ can be calculated at an operation-count cost which is at most cL times the corresponding cost of evaluating the function $\boldsymbol{h}$ itself. The constant factor c guaranteed to be less than six and more typically around two to three [8]. The gradient $\nabla \mathsf{p}_{\mathbf{u}}$ can therefore be evaluated at a cost proportional to evaluating the density itself and the Jacobian $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ can be evaluated at a cost which is proportional to $D_{\mathbf{x}}$ times the cost of a single evaluation of the generator $\boldsymbol{g}_{\mathbf{x}}$.

5. Model parameterisation

A generative model $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{u}}, \mu, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{g}_{\mathbf{z}})$ for $\mathbf{x}$ and $\mathbf{z}$ will not uniquely define the resulting joint distribution $\mathsf{P}_{\mathbf{x},\mathbf{z}}$. As a simple example if $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{u}}, \mu, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{g}_{\mathbf{z}})$ is a differentiable generative model and $\boldsymbol{f} : \mathcal{U} \to \mathcal{U}$ is a diffeomorphism, then we can reparameterise the random inputs as $\mathbf{v} = \boldsymbol{f}^{-1}(\mathbf{u})$. Using the change of variables formula, the corresponding input density is $\mathsf{p}_{\mathbf{v}}(\boldsymbol{v}) = |\mathbf{J}_{\boldsymbol{f}}(\boldsymbol{v})|\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{f}(\boldsymbol{v}))$ and $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{v}}, \mu, \boldsymbol{g}_{\mathbf{x}} \circ \boldsymbol{f}, \boldsymbol{g}_{\mathbf{z}} \circ \boldsymbol{f})$ is also a generative model for $\mathbf{x}$ and $\mathbf{z}$.

The MCMC method we propose performs updates in the input space to the generator, therefore the ability to reparameterise a generative model can be exploited to endow the input density with properties favourable for MCMC inference. For example it will generally be desirable to reparameterise variables with bounded support to transformed variables with unbounded support, for example reparameterising in terms of the logarithm of a strictly positive variable. In general performing updates to unbounded variables simplifies MCMC inference by preventing the need to check transitions respect bounding constraints. Probabilistic programming frameworks such as Stan [34] and PyMC3 [85] make use of a range of such transformations within their MCMC implementations. Choosing parameterisations in terms input variables with a common prior scale, for example using unit variance distributions, is also a useful heuristic as it will typically simplify the choice of scale parameters of MCMC updates.

Although we motivated our definition of $\mathbf{u}$ by saying it could be constructed by tracking all the draws from a PRNG, in general we will not want to parameterise $\mathbf{u}$ in terms of low-level uniform draws, but instead use the output of higher-level functions for producing samples from standard densities. This is important as if for example we defined as inputs the uniform draws used in the rejection sampling routines typically used to generate Gamma random variables, the resulting $\boldsymbol{g}_{\mathbf{x}}$ would be non-differentiable. If we instead use the generated Gamma variable itself as the input by including an appropriate Gamma density factor in $\mathsf{p}_{\mathbf{u}}$ we side step this issue.

In some cases using the outputs of higher-level PRNG routines as the input variables will introduce dependencies between the variables in the input density $\mathsf{p}_{\mathbf{u}}$. In particular if u_i corresponds to the output of a routine which is passed arguments depending on one or more previous random inputs $\{\mathsf{u}_j\}_{j \in \mathcal{J}}$, then an appropriate conditional density factor on u_i given $\{\mathsf{u}_j\}_{j \in \mathcal{J}}$ will need to be included in $\mathsf{p}_{\mathbf{u}}$. By using alternative parameterisations it may be possible to avoid introducing such dependencies; for example a random input v_i generated from a normal distribution with mean μ and standard deviation σ which depend on previous random inputs $\{\mathsf{u}_j\}_{j \in \mathcal{J}}$ can instead be parameterised in terms of an independent random variable u_i distributed with a standard normal density $\mathcal{N}(0, 1)$ and v_i computed as $\sigma \mathsf{u}_i + \mu$ in the generator. Such *non-centred parameterisations* [78, 18, 73] (also known by the 'reparameterisation trick' in the machine learning literature [46, 82]) are available for example for all location-scale family distributions. Whether it is necessarily helpful to remove dependencies in $\mathsf{p}_{\mathbf{u}}$ like this for the proposed method is an open question and will likely be model specific; it has previously been found that non-centred parameterisations can be beneficial when performing MCMC inference in hierarchical models when the unobserved variables are only weakly identified by observations [72, 73, 14].

6. Directed and undirected generative models

So far we have considered generative models where both the observed and unobserved variables are jointly generated from $\mathbf{u}$ without assuming any particular relationship between $\mathbf{x}$ and $\mathbf{z}$. This structure is shown as a factor graph in Figure 1a and a corresponding factor graph for just $\mathbf{x}$ and $\mathbf{z}$ with $\mathbf{u}$ marginalised out shown in Figure 1b.

A common special case is when the input space decomposes as $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2$ and the unobserved variables $\mathbf{z}$ are generated from a subset of the random inputs $\mathbf{u}_1 : \mathcal{S} \to \mathcal{U}_1$ (e.g. corresponding to sampling from a prior distribution over the parameters of a simulator model), with the observed variables $\mathbf{x}$ then generated from a function $\boldsymbol{g}_{\mathbf{x}|\mathbf{z}} : \mathcal{Z} \times \mathcal{U}_2 \to \mathcal{X}$ which takes as input both the generated unobserved variables $\mathbf{z}$ and the remaining random variables $\mathbf{u}_2 : \mathcal{S} \to \mathcal{U}_2$, i.e. $\mathbf{x} = \boldsymbol{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z}, \mathbf{u}_2) = \boldsymbol{g}_{\mathbf{x}|\mathbf{z}}(\boldsymbol{g}_{\mathbf{z}}(\mathbf{u}_1), \mathbf{u}_2)$. This is illustrated as a factor graph in Figure 1c. Again a corresponding factor graph with $\mathbf{u}$ marginalised out is shown in Figure 1d, with in this case the structure of the generator making a directed factorisation in terms $\mathsf{p}_{\mathbf{z}}$ and $\mathsf{p}_{\mathbf{x}|\mathbf{z}}$ natural.

We will therefore term models with this structure as *directed generative models* (with the more general case termed *undirected* for symmetry). The method we propose are equally applicable to undirected and directed generative models, though often the extra structure present in the directed case can allow

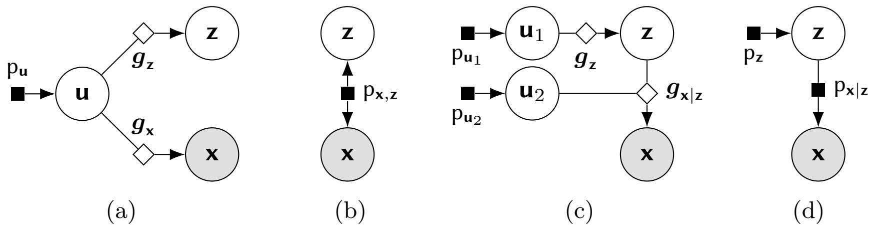


Fig 1: Factor graphs visualising the structure of models considered in this paper. Circular nodes represent random variables, filled square nodes probabilistic factors and unfilled diamonds deterministic factors. Shaded circular nodes are observed. Panel (a) shows the more general undirected model case in which observed variables $\mathbf{x}$ and latent variables $\mathbf{z}$ are jointly generated from random inputs $\mathbf{u}$ by mapping through functions $\boldsymbol{g}_\mathbf{x}$ and $\boldsymbol{g}_\mathbf{z}$, with (b) showing an equivalent factor graph after marginalising out the random inputs. Panel (c) shows the directed model case in which we first generate the latent variables $\mathbf{z}$ from a subset of the random inputs $\mathbf{u}_1$ then generate the observed variables $\mathbf{x}$ from $\mathbf{z}$ and the remaining random inputs $\mathbf{u}_2$, with (d) showing resulting natural directed factorisation of joint distribution when marginalising out $\mathbf{u}_1$ and $\mathbf{u}_2$.

computational gains. Most ABC inference methods concentrate on directed generative models. Typically the marginal density $\mathsf{p_z}$ (i.e. the density of the prior distribution on the unobserved variables) will be tractable to explicitly compute, such that it is only the conditional density $\mathsf{p_{x|z}}$ which cannot be evaluated. As this conditional density is often referred to as the *likelihood*, an alternative designation of *likelihood-free inference* is sometimes used for ABC and related methods.

7. Approximate Bayesian Computation

We will now review the ABC approach to inference in generative models in order to help motivate our proposed method. We will assume here that the observed variables in the generative model of interest are real-valued, i.e. that $\mathcal{X} \subseteq \mathbb{R}^{D_\mathsf{x}}$, with inference in generative models with discrete observations being in general simpler from a theoretical perspective (though not necessarily computationally). The auxiliary-variable description we give of ABC is non-standard, but is consistent with the algorithms used in practice and will help illustrate the relation of our proposed approach to existing ABC methods.

We introduce an auxiliary $\mathcal{X}$-valued random vector $\mathbf{y}$ which depends on the observed random vector $\mathbf{x}$ via a regular conditional distribution $\mathsf{P_{y|x}}$ we term the *kernel* which has a conditional density $k_\epsilon : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ with respect to the Lebesgue measure,

$$\mathsf{P_{y|x}}(\mathcal{A} \,|\, \boldsymbol{x}) = \int_\mathcal{A} k_\epsilon(\boldsymbol{y}; \boldsymbol{x}) \,\mathrm{d}\boldsymbol{y} \qquad \forall \mathcal{A} \in \mathfrak{B}(\mathcal{X}). \tag{4}$$

The kernel density k_ϵ is parameterised by a *tolerance* ϵ and chosen such that the following conditions holds for arbitrary measurable functions $f : \mathcal{X} \to \mathbb{R}$

$$\lim_{\epsilon \to 0} \int_{\mathcal{X}} f(\boldsymbol{y})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathrm{d}\boldsymbol{y} = f(\boldsymbol{x}) \tag{5}$$

$$\text{and} \quad \lim_{\epsilon \to 0} \int_{\mathcal{X}} f(\boldsymbol{x})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = f(\boldsymbol{y}). \tag{6}$$

For kernels meeting these condition (5) we have that $\forall \mathcal{A} \in \mathfrak{B}(\mathcal{X})$

$$\begin{aligned} \lim_{\epsilon \to 0} \mathsf{P}_{\mathbf{y}}(\mathcal{A}) &= \lim_{\epsilon \to 0} \int_{\mathcal{X}} \mathsf{P}_{\mathbf{y}|\mathbf{x}}(\mathcal{A} \,|\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) && (7)\\ &= \lim_{\epsilon \to 0} \int_{\mathcal{X}} \int_{\mathcal{X}} \mathbb{I}_{\mathcal{A}}(\boldsymbol{y})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathrm{d}\boldsymbol{y}\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) && (8)\\ &= \int_{\mathcal{X}} \mathbb{I}_{\mathcal{A}}(\boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \mathsf{P}_{\mathbf{x}}(\mathcal{A}), && (9) \end{aligned}$$

i.e. that in the limit $\epsilon \to 0$, $\mathbf{y}$ has the same distribution as $\mathbf{x}$. Intuitively, as we decrease the tolerance ϵ we increasingly tightly constrain $\mathbf{y}$ and $\mathbf{x}$ to have similar distributions. Two common choices of kernels satisfying (5) and (6) are the *uniform ball* and *Gaussian* kernels which respectively have densities

$$k_\epsilon(\boldsymbol{y}; \boldsymbol{x}) \propto \frac{1}{\epsilon^{D_{\mathbf{x}}}} \mathbb{I}_{[0,\epsilon]}(\|\boldsymbol{y} - \boldsymbol{x}\|_2) \qquad \text{(uniform ball kernel)}, \tag{10}$$

$$\text{and} \quad k_\epsilon(\boldsymbol{y}; \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{y} \,|\, \boldsymbol{x},\, \epsilon^2 \mathbf{I}\big) \qquad \text{(Gaussian kernel)}. \tag{11}$$

The marginal distribution of $\mathbf{y}$ can be written $\forall \mathcal{A} \in \mathfrak{B}(\mathcal{X})$ as

$$\mathsf{P}_{\mathbf{y}}(\mathcal{A}) = \int_{\mathcal{X} \times \mathcal{Z}} \mathsf{P}_{\mathbf{y}|\mathbf{x}}(\mathcal{A} \,|\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \int_{\mathcal{A}} \int_{\mathcal{X}} k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x})\, \mathrm{d}\boldsymbol{y}, \tag{12}$$

from which we have that $\mathsf{P}_{\mathbf{y}}$ has a density with respect to the Lebesgue measure

$$\mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) = \int_{\mathcal{X}} k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \int_{\mathcal{X} \times \mathcal{Z}} k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) \qquad \forall \boldsymbol{y} \in \mathcal{X}. \tag{13}$$

The density $\mathsf{p}_{\mathbf{y}}$ exists for $\epsilon > 0$ irrespective of whether $\mathsf{P}_{\mathbf{x}}$ has a density with respect to the Lebesgue measure (it may not for example if $\mathsf{P}_{\mathbf{x}}$ only has support on a sub-manifold of $\mathcal{X}$). Using this definition of the density $\mathsf{p}_{\mathbf{y}}$ we have that for any measurable function $f : \mathcal{Z} \to \mathcal{F}$ and for all $\mathcal{A} \in \mathfrak{B}(\mathcal{X})$ that

$$\begin{aligned} \int_{\mathcal{A} \times \mathcal{Z}} f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{y},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}\boldsymbol{z}) &= \int_{\mathcal{A} \times \mathcal{X} \times \mathcal{Z}} f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{y},\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) && (14)\\ &= \int_{\mathcal{A}} \int_{\mathcal{X} \times \mathcal{Z}} f(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})\, \mathrm{d}\boldsymbol{y} && (15)\\ &= \int_{\mathcal{A}^*} \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{\mathcal{X} \times \mathcal{Z}} f(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})\, \mathsf{P}_{\mathbf{y}}(\mathrm{d}\boldsymbol{y}) && (16) \end{aligned}$$

where we define $\mathcal{A}^* = \{\boldsymbol{y} \in \mathcal{A} : \mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) > 0\}$. Comparing this to the definition of the conditional expectation in (1) we have that $\forall \boldsymbol{y} \in \mathcal{X} : \mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) > 0$

$$\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon] = \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{\mathcal{X} \times \mathcal{Z}} f(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) \tag{17}$$

$$= \frac{\int_{\mathcal{X} \times \mathcal{Z}} f(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}{\int_{\mathcal{X} \times \mathcal{Z}} k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}. \tag{18}$$

For the case of a model in which $\mathsf{P}_{\mathbf{z}}$ has a density $\mathsf{p}_{\mathbf{z}}$ with respect to the Lebesgue measure, then if we use $f = \mathbb{I}_{\mathcal{A}}$ for $\mathcal{A} \in \mathfrak{B}(\mathcal{Z})$ in (18) and the definition of a regular conditional distribution $\mathsf{P}_{\mathbf{z}|\mathbf{y}}(\mathcal{A} \,|\, \boldsymbol{y}) \equiv \mathbb{E}[\mathbb{I}_{\mathcal{A}}(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y}]$ we have

$$\mathsf{P}_{\mathbf{z}|\mathbf{y}}(\mathcal{A} \,|\, \boldsymbol{y}) = \int_{\mathcal{A}} \frac{\int_{\mathcal{X}} k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \,|\, \boldsymbol{z})\, \mathsf{p}_{\mathbf{z}}(\boldsymbol{z})}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})}\, \mathrm{d}\boldsymbol{z}. \tag{19}$$

In this case the regular conditional distribution $\mathsf{P}_{\mathbf{z}|\mathbf{y}}$ has a conditional density $\mathsf{p}_{\mathbf{z}|\mathbf{y}}$ with respect to the Lebesgue measure,

$$\mathsf{p}_{\mathbf{z}|\mathbf{y}}(\boldsymbol{z} \,|\, \boldsymbol{y}) = \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{\mathcal{X}} k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \,|\, \boldsymbol{z})\, \mathsf{p}_{\mathbf{z}}(\boldsymbol{z}). \tag{20}$$

In reference to terminology of Bayesian inference, the density $\mathsf{p}_{\mathbf{z}|\mathbf{y}}$ is termed the ABC posterior density, and therefore conditional expectations of the form of (18) which correspond to an integral with respect to this ABC posterior, are termed ABC posterior expectations.

We now consider how $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]$ is related to the conditional expectation we are interested in evaluating $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{y}]$. If we assume that $\mathsf{P}_{\mathbf{x},\mathbf{z}}$ is absolutely continuous with respect to the Lebesgue measure with density $\mathsf{p}_{\mathbf{x},\mathbf{z}}$ and using (6) we have that $\forall \boldsymbol{y} \in \mathcal{X} : \mathsf{p}_{\mathbf{x}}(\boldsymbol{y}) > 0$

$$\lim_{\epsilon \to 0} \mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon] = \lim_{\epsilon \to 0} \frac{\int_{\mathcal{Z}} f(\boldsymbol{z}) \int_{\mathcal{X}} k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{z})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{z}}{\int_{\mathcal{Z}} \int_{\mathcal{X}} k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{z})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{z}} \tag{21}$$

$$= \frac{\int_{\mathcal{Z}} f(\boldsymbol{z})\, \mathsf{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{y}, \boldsymbol{z})\, \mathrm{d}\boldsymbol{z}}{\int_{\mathcal{Z}} \mathsf{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{y}, \boldsymbol{z})\, \mathrm{d}\boldsymbol{z}} \tag{22}$$

$$= \mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{y}]. \tag{23}$$

We therefore have that conditional expectations $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y};\, \epsilon]$ converge as $\epsilon \to 0$ to the conditional expectations we wish to be able to estimate $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x}\,]$. Crucially we also have that the numerator and denominator of (18) take the forms of expectations of known functions of $\mathbf{x}$ and $\mathbf{z}$, i.e.

$$\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon] = \frac{\mathbb{E}[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y};\, \mathbf{x})]}{\mathbb{E}[k_\epsilon(\boldsymbol{y};\, \mathbf{x})]}. \tag{24}$$

Generating Monte Carlo estimates of these expectations only requires us to be able to generate samples from $\mathsf{P}_{\mathbf{x},\mathbf{z}}$ without any requirement to be able to evaluate

$\mathsf{p}_{\mathbf{x},\mathbf{z}}$ and therefore can be achieved in the implicit generative models of interest. We can therefore estimate $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]$ by generating a set of independent pairs of random vectors $\{\mathbf{x}_s,\, \mathbf{z}_s\}_{s=1}^{S}$ from $\mathsf{P}_{\mathbf{x},\mathbf{z}}$[2] and computing Monte Carlo estimates of the numerator and denominator in (24), which gives the following estimator for the conditional expectation $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]$

$$\hat{\mathsf{f}}_{S,\epsilon} = \frac{\sum_{s=1}^{S} (f(\mathbf{z}_s)\, k_\epsilon(\boldsymbol{y};\, \mathbf{x}_s))}{\sum_{s=1}^{S} (k_\epsilon(\boldsymbol{y};\, \mathbf{x}_s))} \tag{25}$$

This directly corresponds to an importance sampling estimator for expectations with respect to $\mathsf{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ using $\mathsf{P}_{\mathbf{x},\mathbf{z}}$ as the proposal distribution. Therefore if both $f(\mathbf{z})\, k_\epsilon(\boldsymbol{y}; \mathbf{x})$ and $k_\epsilon(\boldsymbol{y}; \mathbf{x})$ have finite variance, then the estimator $\hat{\mathsf{f}}_{S,\epsilon}$ will be consistent,

$$\lim_{S\to\infty} \mathbb{E}\left[\hat{\mathsf{f}}_{S,\epsilon}\right] = \mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]. \tag{26}$$

If the kernel used is the uniform ball kernel (10), the estimator can be manipulated in to a particularly intuitive form

$$\hat{\mathsf{f}}_{S,\epsilon} = \frac{1}{|\mathcal{A}|} \sum_{s\in\mathcal{A}} (f(\mathbf{z}_s)) \quad \text{with} \quad \mathcal{A} = \{s \in \{1 \ldots S\} : \|\boldsymbol{y} - \mathbf{x}_s\|_2 < \epsilon\} \tag{27}$$

which corresponds to averaging the values of sampled unobserved variables $\mathbf{z}_s$ where the corresponding samples of model observed variables $\mathbf{x}_s$ are within a distance ϵ of the observed data $\boldsymbol{y}$. The is the standard ABC rejection algorithm [84, 90, 33, 96, 79] , with $\mathcal{A}$ corresponding to the indices of the set of accepted samples, with the other samples being ‘rejected’ as the simulated observations $\mathbf{x}_s$ are more than a distance ϵ from the observed data $\boldsymbol{y}$. As an instance of a rejection sampler, conditioned on the acceptance set containing at least one sample, i.e. $|\mathcal{A}| > 0$, (27) is an unbiased estimator for $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]$.

If we instead use a Gaussian kernel (11), then as for the general case for importance sampling, the estimator (25) is no longer unbiased. In the Gaussian kernel case we more highly weight samples if the simulated observed variables are closer to the data which may be viewed as preferable to equally weighting all values within a fixed tolerance as in ABC reject. However as it has support on all of $\mathcal{X}$, the Gaussian kernel also gives non-zero weights to all of the samples, with typically most making little contribution to the expectation. This may be considered somewhat wasteful of computation versus the rejection scheme which creates a sparse set of samples to compute expectations over [10]. Kernels with bounded support but non-flat densities such as the *Epanechnikov kernel* [29] which has a parabolic density in a bounded region, offer some of the advantages of both the uniform ball and Gaussian kernels.

Irrespective of the kernel chosen, the estimate formed is only consistent for the ABC posterior expectation $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]$ rather than the actual posterior expectation $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{y}]$ we are directly interested in. As $\epsilon \to 0$, $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]$

[2]As ABC is usually applied to directed models this is usually considered as generating $\mathbf{z}$ from a prior then simulating $\mathbf{x}$ given $\mathbf{z}$ however more generally we can sample from the joint.

converges to $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{y}]$, however for reject ABC we also have that as $\epsilon \to 0$ the proportion of accepted samples will tend to zero meaning that we need to expend increasing computational effort to get an estimator for $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]$ with a similar variance (which by a standard Monte Carlo argument is inversely proportional to the number of accepted samples).

In the more general importance sampling case, although we do not explicitly reject any samples if using a kernel with unbounded support, we instead have that as $\epsilon \to 0$ that the kernel weightings in (25) will becoming increasingly dominated by the few samples closest to the observed data and so the contribution from to the estimator (25) from all but a few will be negligible, again leading to an increasing number of samples being needed to keep the variance of the estimator reasonable. For the exact $\epsilon = 0$ case we would only accept (or equivalently put non-zero weight on) samples for which $\mathbf{x}_s$ is exactly equal to $\boldsymbol{y}$. For $\mathcal{X} \subseteq \mathbb{R}^{D_{\mathbf{x}}}$ if $\mathsf{P}_{\mathbf{x}}$ is absolutely continuous with respect to the Lebesgue measure, the event $\mathbf{x} = \boldsymbol{y}$ has zero measure under $\mathsf{P}_{\mathbf{x},\mathbf{z}}$ and so some degree of approximation due to a $\epsilon > 0$ is always required in practice in these simple Monte Carlo ABC schemes.

When the dimensionality of the observed variable vector $\mathbf{x}$ is high it quickly becomes impractical to reduce the variance of these naive Monte Carlo estimators for (18) to reasonable levels without using large ϵ which introduces significant approximation error. The ABC rejection method is well known to scale poorly with dimensionality due to curse of dimensionality effects [16, 56, 77]. Although often discussed specifically in the context of ABC, the issues faced are much the same as encountered when trying to use any simple rejection or importance sampling scheme to approximate expectations with respect to a probability distribution on a high-dimensional space. If the proposal distribution ($\mathsf{P}_{\mathbf{x},\mathbf{z}}$ here) is significantly more diffuse than the target distribution ($\mathsf{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ here) an exponentially small proportion of the probability mass of the proposal distribution will lie in the typical set of the target distribution and so very few samples will be accepted or have non-negligible importance weights.

Rather than conditioning on the full observed data most applications of ABC methods therefore instead use *summary statistics* to extract lower dimensional representations of the observed data [77]. A function $\boldsymbol{s} : \mathcal{X} \to \mathcal{T}$ is defined which computes summary statistics from simulated observed outputs $\mathbf{x}$ and observed data $\boldsymbol{y}$ with the dimensionality of the summaries, $\dim(\mathcal{T})$, typically much smaller than $D_{\mathbf{x}}$. The ABC posterior expectation is then computed using

$$\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{s} = \boldsymbol{s}(\boldsymbol{y});\, \epsilon] = \frac{\int_{\mathcal{X}\times\mathcal{Z}} f(\boldsymbol{z})\, k_\epsilon(\boldsymbol{s}(\boldsymbol{y}); \boldsymbol{s}(\boldsymbol{x}))\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}{\int_{\mathcal{X}\times\mathcal{Z}} k_\epsilon(\boldsymbol{s}(\boldsymbol{y}); \boldsymbol{s}(\boldsymbol{x}))\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}, \tag{28}$$

with now the variable conditioned on the $\mathcal{T}$-valued variable $\mathbf{s}$ with

$$\mathsf{P}_{\mathbf{s}|\mathbf{x}}(\mathcal{A} \,|\, \boldsymbol{x}) = \int_{\mathcal{A}} k_\epsilon(\boldsymbol{s}; \boldsymbol{s}(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{s} \qquad \forall \mathcal{A} \in \mathfrak{B}(\mathcal{T}),\, \boldsymbol{x} \in \mathcal{X}. \tag{29}$$

In general the statistics used will not be *sufficient* - the posterior distribution on $\mathbf{z}$ will differ when conditioning on $\boldsymbol{s}(\mathbf{x})$ compared to conditioning on $\mathbf{x}$ directly. By a data processing inequality argument we know that the mutual information between $\mathbf{z}$ and $\boldsymbol{s}(\mathbf{x})$ will be less than or equal to the mutual information between $\mathbf{z}$ and $\mathbf{x}$ therefore we would expect for the posterior distribution on $\mathbf{z}$ given $\boldsymbol{s}(\mathbf{x})$ to be less informative about $\mathbf{z}$ than the posterior distribution given $\mathbf{x}$ [5]. This means that even in the limit of $\epsilon \to 0$ estimates of the ABC summary statistics posterior expectation $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{s} = \boldsymbol{s}(\boldsymbol{y});\, \epsilon]$ will generally not converge to the true posterior expectations $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{y}]$ of interest.

ABC methods therefore trade-off between the approximation errors introduced due to using summary statistics and a non-zero tolerance ϵ, and the Monte Carlo error from using a finite number of samples in the estimates. If informative summary statistics can be found then typically the approximation error can be kept to a more reasonable level compared to the conditioning on the full data without the Monte Carlo error becoming impractically large by allowing a smaller ϵ to be used while maintaining a reasonable accept rate. Finding informative low-dimensional summaries is often critical to getting ABC methods to work well in practice and there is a wide literature on methods for choosing summary statistics - see [77] and [17] for reviews.

In some cases use of summary statistics might not be viewed just as a computational aid, but as a purposeful exercise in removing 'irrelevant' information from the data. For example if inferring plausible parameter values for a dynamic model of a system given observed sequences of the system state showing periodic behaviour, then we might view the phase of observed state sequences as an irrelevant artefact of the arbitrary point at which observations were started. In this case conditioning on the exact observed data could be viewed as over constraining the model to reproduce features of the data which are only incidental, and using summary statistics which are invariant to phase could be preferable to conditioning on the full data [99].

Similarly the introduction of a kernel in ABC need not be viewed as simply a method for making inference tractable, but instead as part of the modelling process [98]. In general we will expect any observed data to be subject to some amount of measurement noise (at the very least it will include some quantification noise) and so conditioning the model to reproduce the exact values of the data is not necessarily desirable. In this context we can consider $\mathbf{y}$ the noisy measured version of an underlying state $\mathbf{x}$ and the kernel $\mathsf{P}_{\mathbf{y}|\mathbf{x}}$ as representing the measurement noise model. We might also instead view the kernel $\mathsf{P}_{\mathbf{y}|\mathbf{x}}$ as accounting for the mismatch between our proposed model for how the observed values are generated and the true data generating process [81, 98]. In both these cases we could then consider ϵ as a further unobserved variable to be inferred.

These examples demonstrate that in some cases there may be a modelling motivation for introducing summary statistics or a 'noise' kernel. In practice however the summary statistics and tolerance ϵ are more typically chosen on grounds of computational tractability [56, 83, 77]. Therefore inference methods which are able to maintain tractability when conditioning on higher-dimensional summaries or in some cases all observations, and when using smaller tolerance ϵ values are of significant practical interest.

As an alternative to the simple Monte Carlo ABC inference schemes so far described, methods have also been proposed to utilise more scalable inference methods to estimate the approximate expectation (17) including sequential Monte Carlo methods [87, 92], population Monte Carlo [9], expectation propagation [7] and variational Bayes [94]. Of particular relevance to our work is the use of MCMC within an ABC framework [57, 86]. As is standard in ABC methods, ABC MCMC approaches are generally targeted at directed models where the unobserved variables have a known marginal density $\mathsf{p}_{\mathbf{z}}$ but we can only generate samples from the conditional distribution $\mathsf{P}_{\mathbf{x}|\mathbf{z}}$. If a Markov chain is constructed on a $(\mathbf{x}, \mathbf{z})$ state pair with unique stationary distribution

$$\mathsf{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}(\mathcal{A},\, \mathcal{B} \,|\, \boldsymbol{y}) = \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{\mathcal{B}} \int_{\mathcal{A}} \mathsf{p}_{\mathbf{z}}(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \,|\, \boldsymbol{z})\, \mathrm{d}\boldsymbol{z} \tag{30}$$

then we can compute consistent MCMC estimators for (18) by computing averages over the unobserved variable $\mathbf{z}$ components of the chain states.

The standard ABC MCMC approach [57] uses a Metropolis–Hastings scheme which perturbatively updates the unobserved variables but independently resamples the observed variables. A new chain state $(\boldsymbol{x}^*,\, \boldsymbol{z}^*)$ is proposed given the current state $(\boldsymbol{x}_s,\, \boldsymbol{z}_s)$ by sampling $\boldsymbol{z}^*$ from a Markov kernel with density $q : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$ and then generating a new $\boldsymbol{x}^*$ by sampling from $\mathsf{P}_{\mathbf{x}|\mathbf{z}}(\cdot \,|\, \boldsymbol{z}^*)$. With probability

$$\alpha(\boldsymbol{x}^*, \boldsymbol{z}^* \,|\, \boldsymbol{x}_s, \boldsymbol{z}_s) = \min\left\{1,\, \frac{q(\boldsymbol{z}_s \,|\, \boldsymbol{z}^*)\, \mathsf{p}_{\mathbf{z}}(\boldsymbol{z}^*)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x}^*)}{q(\boldsymbol{z}^* \,|\, \boldsymbol{z}_s)\, \mathsf{p}_{\mathbf{z}}(\boldsymbol{z}_s)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x}_s)}\right\}, \tag{31}$$

the proposed $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ pair is accepted such that $(\boldsymbol{x}_{s+1}, \boldsymbol{z}_{s+1}) \leftarrow (\boldsymbol{x}^*, \boldsymbol{z}^*)$ otherwise a rejection occurs and $(\boldsymbol{x}_{s+1}, \boldsymbol{z}_{s+1}) \leftarrow (\boldsymbol{x}_s, \boldsymbol{z}_s)$. The transition operator defined by this process leaves (30) invariant, and under a suitable choice of proposal density for the $\mathbf{z}$ updates will be aperiodic and irreducible and so have (30) as its unique stationary distribution.

By making small changes to the unobserved variables $\mathbf{z}$ and so making use of information from the previous state about plausible values for $\mathbf{z}$ under $\mathsf{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ rather than independently sampling them from $\mathsf{P}_{\mathbf{z}}$ as in the simpler Monte Carlo schemes, ABC MCMC can often increase efficiency in generative models with large numbers of unobserved variables to infer [86]. This potential improved efficiency comes at a cost of introducing the usual challenges associated with MCMC methods compared to simpler Monte Carlo methods of dependence between successive samples and the difficulty of assessing convergence.

Further ABC MCMC chains can be prone to 'sticking' pathologies - suffering large series of rejections visible as the variables being stuck at a fixed value in traces of the chain state. Though small moves are proposed to $\mathbf{z}$, proposed updates to the simulated observations $\mathbf{x}$ are sampled independently of the previous simulated observations. Generally the conditional distribution $\mathsf{P}_{\mathbf{x}|\mathbf{z},\mathbf{y}}$, i.e. describing the plausible values for $\mathbf{x}$ given the observed data *and* proposed $\mathbf{z}$ values, will be much more concentrated than the distribution $\mathsf{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ and so proposing updates to $\mathbf{x}$ from the latter will often lead to proposed values for $(\mathbf{x}, \mathbf{z})$ with a very low acceptance probability. The ABC MCMC Metropolis–Hastings scheme is an instance of a pseudo-marginal MCMC method [10, 4] where such sticking artifacts are a well known problem [65].

8. Inference in the input space

To try to overcome some of the limitations of the standard ABC MCMC approach, we now consider reparameterising the inference problem using the formulation of a generative model as a deterministic transformation of random inputs introduced in Definition 1 in Section 4. For a generative model $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{u}}, \mu, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{g}_{\mathbf{z}})$ for observed variables $\mathbf{x}$ and unobserved variables $\mathbf{z}$, the ABC posterior expectation (18) can be reparameterised as

$$\begin{aligned}\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon] &= \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})}\mathbb{E}[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y};\, \mathbf{x})] &(32)\\ &= \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})}\mathbb{E}[f(\boldsymbol{g}_{\mathbf{z}}(\mathbf{u}))\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{g}_{\mathbf{x}}(\mathbf{u}))] &(33)\\ &= \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})}\int_{\mathcal{U}} f \circ \boldsymbol{g}_{\mathbf{z}}(\boldsymbol{u})\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}))\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}. &(34)\end{aligned}$$

Crucially this reparameterisation takes the form of an integral of a function $f \circ \boldsymbol{g}_{\mathbf{z}}$ against an *explicit* probability density

$$\pi_\epsilon(\boldsymbol{u}) = \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})} k_\epsilon(\boldsymbol{y};\, \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}))\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u}), \tag{35}$$

that we can evaluate up to an unknown normalising constant $\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})$. This is the typical setting for approximate inference in (explicit) probabilistic models, and so is amenable to applying standard variants of methods such as MCMC and variational inference. In the common special case (and typical ABC setting) of a directed generative model with a tractable marginal density on the unobserved variables $\mathsf{p}_{\mathbf{z}}$, using the notation introduced in Section 6 we have that

$$\begin{aligned}\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon] &= \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})}\mathbb{E}\Big[f(\mathbf{z})\, k_\epsilon\Big(\boldsymbol{y};\, \boldsymbol{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z},\, \mathbf{u}_2)\Big)\Big] &(36)\\ &= \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})}\int_{\mathcal{Z}}\int_{\mathcal{U}_2} f(\boldsymbol{z})\, k_\epsilon\Big(\boldsymbol{y};\, \boldsymbol{g}_{\mathbf{x}|\mathbf{z}}(\boldsymbol{z},\, \boldsymbol{u}_2)\Big)\, \mathsf{p}_{\mathbf{z}}(\boldsymbol{z})\mathsf{p}_{\mathbf{u}_2}(\boldsymbol{u}_2)\, \mathrm{d}\boldsymbol{u}_2\, \mathrm{d}\boldsymbol{z} &(37)\end{aligned}$$

with now the explicit target density for inference being

$$\pi_\epsilon(\boldsymbol{z}, \boldsymbol{u}_2) = \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})} k_\epsilon\Big(\boldsymbol{y};\, \boldsymbol{g}_{\mathbf{x}|\mathbf{z}}(\boldsymbol{z},\, \boldsymbol{u}_2)\Big)\, \mathsf{p}_{\mathbf{z}}(\boldsymbol{z})\, \mathsf{p}_{\mathbf{u}_2}(\boldsymbol{u}_2). \tag{38}$$

This latter form is directly comparable to the reparameterisation suggested in [67] for pseudo-marginal inference problems. There it is applied to construct a MCMC method which uses slice sampling transition operators to iterate between updating the unobserved variables $\mathbf{z}$ given random inputs $\mathbf{u}_2$ and vice versa. For models in which the target density (35) is continuous with respect to both arguments, the slice sampling updates will be almost surely move the state a non-zero distance, therefore the chain will not 'stick'. Related approaches using Metropolis updates instead have also been proposed [24, 25].

In reparameterising inference in terms of evaluating an integral over the input space we have still so far required the definition of a kernel k_ϵ and tolerance ϵ, with the integral being estimated the ABC posterior expectation $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y},\, \epsilon]$ (18) rather than exact posterior expectation $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x}\,]$ we are directly interested in. We now consider in the specific case of differentiable generative models how to perform inference without introducing an ABC kernel.

We begin an initial intuition for the approach, by considering taking the limit of $\epsilon \to 0$ in the integral (32) corresponding to evaluating the ABC posterior expectation in the generator input space. We previously showed in (21) that the approximate expectation $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon]$ converges as $\epsilon \to 0$ to the conditional expectation of interest $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{y}]$, providing that the implicit distribution of the observed variables in the generative model $\mathsf{P}_{\mathbf{x}}$ is absolutely continuous with respect to the Lebesgue measure with density $\mathsf{p}_{\mathbf{x}}$. Informally for kernels meeting the conditions (5) and (6), in the limit of $\epsilon \to 0$ the kernel density $k_\epsilon(\boldsymbol{y};\, \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}))$ tends to a Dirac delta $\delta(\boldsymbol{y} - \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}))$ and so

$$\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{y}] = \lim_{\epsilon \to 0} \mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon] \tag{39}$$

$$\simeq \frac{\int_{\mathcal{U}} f \circ \boldsymbol{g}_{\mathbf{z}}(\boldsymbol{u})\, \delta(\boldsymbol{y} - \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}))\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}}{\int_{\mathcal{U}} \delta(\boldsymbol{y} - \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}))\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}}. \tag{40}$$

The Dirac delta term restricts the integral across the input space $\mathcal{U}$ to a $D_{\mathbf{u}} - D_{\mathbf{x}}$ dimensional, implicitly-defined manifold corresponding to the fibre of $\boldsymbol{y}$ under $\boldsymbol{g}_{\mathbf{x}}$ (i.e. the pre-image under $\boldsymbol{g}_{\mathbf{x}}$ of the singleton set $\{\boldsymbol{y}\}$),

$$\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{y}] \equiv \{\boldsymbol{u} \in \mathcal{U} : \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}) = \boldsymbol{y}\}. \tag{41}$$

It is not necessarily immediately clear how to define the required probability density on the manifold for arbitrary non-injective $\boldsymbol{g}_{\mathbf{x}}$. In differentiable generative models we can however use a derivation equivalent to that given by Diaconis, Holmes and Shahshahani in [26] for conditional densities on a manifold to find an expression for the conditional expectation consistent with definition given in (1). The key result we use is Federer's *co-area formula* [30, §3.2.12]. This generalises Fubini's theorem for iterated integrals on spaces defined by a Cartesian product to more general foliations of a space.

Theorem 1 (Co-area formula)**:** *Let $\mathcal{V} \subseteq \mathbb{R}^L$ and $\mathcal{W} \subseteq \mathbb{R}^K$ with $L \geq K$, and let $\boldsymbol{m} : \mathcal{V} \to \mathcal{W}$ be a Lipschitz function and $h : \mathcal{V} \to \mathbb{R}$ a Lebesgue measurable function. Then*

$$\int_{\mathcal{V}} h(\boldsymbol{v})\, J_{\boldsymbol{m}}(\boldsymbol{v})\, \lambda^L(\mathrm{d}\boldsymbol{v}) = \int_{\mathcal{W}} \int_{\boldsymbol{m}^{-1}[\boldsymbol{w}]} h(\boldsymbol{v})\, \mathcal{H}^{L-K}(\mathrm{d}\boldsymbol{v})\, \lambda^K(\mathrm{d}\boldsymbol{w}) \tag{42}$$

with $\mathcal{H}^{L-K}$ denoting the $L-K$-dimensional Hausdorff measure and $J_{\boldsymbol{m}}(\boldsymbol{v})$ denoting the generalised Jacobian determinant for 'wide' rectangular Jacobian matrices

$$J_{\boldsymbol{m}}(\boldsymbol{v}) \equiv \left| \mathbf{J}_{\boldsymbol{m}}(\boldsymbol{u}) \mathbf{J}_{\boldsymbol{m}}(\boldsymbol{u})^T \right|^{\frac{1}{2}}. \tag{43}$$

More immediately applicable in our case is the following corollary.

Corollary 1: *If Q is a probability measure on $\mathcal{V}$ with density q with respect to the Lebesgue measure λ^L and $\mathbf{J}_{\boldsymbol{m}}$ is full row-rank Q-almost everywhere, then for Lebesgue measurable $f : \mathcal{V} \to \mathbb{R}$*

$$\int_{\mathcal{V}} f(\boldsymbol{v})\, q(\boldsymbol{v})\, \lambda^L(\mathrm{d}\boldsymbol{v}) = \int_{\mathcal{W}} \int_{\boldsymbol{m}^{-1}[\boldsymbol{w}]} f(\boldsymbol{v})\, q(\boldsymbol{v})\, J_{\boldsymbol{m}}(\boldsymbol{v})^{-1}\, \mathcal{H}^{L-K}(\mathrm{d}\boldsymbol{v})\, \lambda^K(\mathrm{d}\boldsymbol{w}). \tag{44}$$

This can be shown by setting $h(\boldsymbol{v}) = f(\boldsymbol{v})\, q(\boldsymbol{v}) J_{\boldsymbol{m}}(\boldsymbol{v})^{-1}$ in (42) and using the equivalence of Lebesgue integrals in which the integrand differs only zero-measure sets. We now show that distribution of the observed variables $\mathsf{P}_{\mathbf{x}}$ has a density $\mathsf{p}_{\mathbf{x}}$ with respect to the Lebesgue measure.

Proposition 1 (Change of variables in a differentiable generative model)**:** *For a differentiable generative model $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{u}}, \mu, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{g}_{\mathbf{z}})$ as defined in Definition 2, then if the generator $\boldsymbol{g}_{\mathbf{x}}$ is Lipschitz and the Jacobian $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ has full row-rank $\mathsf{P}_{\mathbf{u}}$-almost everywhere, the observed vector $\mathbf{x}$ has a density with respect to the Lebesgue measure satisfying*

$$\mathsf{p}_{\mathbf{x}}(\boldsymbol{x}) = \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathbf{u}}-D_{\mathbf{x}}}(\mathrm{d}\boldsymbol{u}) \qquad \forall \boldsymbol{x} \in \mathcal{X}. \tag{45}$$

Proof. From Definitions 1 and 2 we have that $\mathbf{x} = \boldsymbol{g}_{\mathbf{x}}(\mathbf{u})$ and $\frac{\mathrm{d}\mathsf{P}_{\mathbf{u}}}{\mathrm{d}\lambda^{D_{\mathbf{u}}}} = \mathsf{p}_{\mathbf{u}}$ and so

$$\mathsf{P}_{\mathbf{x}}(\mathcal{A}) = \int_{\mathcal{U}} \mathbb{I}_{\mathcal{A}}(\boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \int_{\mathcal{U}} \mathbb{I}_{\mathcal{A}} \circ \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u})\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, \lambda^{D_{\mathbf{u}}}(\mathrm{d}\boldsymbol{u}) \qquad \forall \mathcal{A} \in \mathscr{G}. \tag{46}$$

As $\boldsymbol{g}_{\mathbf{x}}$ is Lipschitz and $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ has full row-rank $\mathsf{P}_{\mathbf{u}}$-almost everywhere we can apply Corollary 1, and so we have that $\forall \mathcal{A} \in \mathscr{G}$

$$\mathsf{P}_{\mathbf{x}}(\mathcal{A}) = \int_{\mathcal{X}} \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} \mathbb{I}_{\mathcal{A}} \circ \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u})\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathbf{u}}-D_{\mathbf{x}}}(\mathrm{d}\boldsymbol{u})\, \lambda^{D_{\mathbf{x}}}(\mathrm{d}\boldsymbol{x}). \tag{47}$$

The term $\mathbb{I}_{\mathcal{A}} \circ \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u})$ inside the inner integral is equal to $\mathbb{I}_{\mathcal{A}}(\boldsymbol{x})$ across all points on the fibre $\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$ being integrated across and so can be taken outside the inner integral to give

$$\mathsf{P}_{\mathbf{x}}(\mathcal{A}) = \int_{\mathcal{X}} \mathbb{I}_{\mathcal{A}}(\boldsymbol{x}) \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathbf{u}}-D_{\mathbf{x}}}(\mathrm{d}\boldsymbol{u})\, \lambda^{D_{\mathbf{x}}}(\mathrm{d}\boldsymbol{x}) \tag{48}$$

$$= \int_{\mathcal{A}} \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathbf{u}}-D_{\mathbf{x}}}(\mathrm{d}\boldsymbol{u})\, \lambda^{D_{\mathbf{x}}}(\mathrm{d}\boldsymbol{x}). \tag{49}$$

By definition the density $\mathsf{p}_{\mathbf{x}}$ of a probability measure $\mathsf{P}_{\mathbf{x}}$ with respect to the Lebesgue measure $\lambda^{D_{\mathbf{x}}}$ satisfies

$$\mathsf{P}_{\mathbf{x}}(\mathcal{A}) = \int_{\mathcal{A}} \mathsf{p}_{\mathbf{x}}(\boldsymbol{x})\, \lambda^{D_{\mathbf{x}}}(\mathrm{d}\boldsymbol{x}) \qquad \forall \mathcal{A} \in \mathscr{G} \tag{50}$$

$\therefore$ $\mathsf{P}_{\mathbf{x}}$ has a density corresponding to (45) with respect to $\lambda^{D_{\mathbf{x}}}$. □

This is a generalisation of the standard change of variables formula under a diffeomorphism. We now derive a result for the conditional expectation.

Proposition 2 (Conditional expectations in a differentiable generative model)**:** *For a differentiable generative model* $(\mathcal{U}, \mathscr{F}, \mathsf{p}_{\mathbf{u}}, \mu, \boldsymbol{g}_{\mathbf{x}}, \boldsymbol{g}_{\mathbf{z}})$ *as defined in Definition 2 and satisfying the conditions in Proposition 1, then for Lebesgue measurable functions* $f : \mathcal{X} \to \mathbb{R}$ *and* $\boldsymbol{x} \in \mathcal{X}$ *such that* $\mathsf{p}_{\mathbf{x}}(\boldsymbol{x}) > 0$ *we have that*

$$\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{x}] = \frac{1}{\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})} \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} f \circ \boldsymbol{g}_{\mathbf{z}}(\boldsymbol{u})\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathsf{u}}-D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{u}). \tag{51}$$

Proof. From Definition 1 we have that $\mathbf{x} = \boldsymbol{g}_{\mathbf{x}}(\mathbf{u})$ and $\mathbf{z} = \boldsymbol{g}_{\mathbf{z}}(\mathbf{u})$ and so $\forall \mathcal{A} \in \mathscr{G}$

$$\int_{\mathcal{A}\times\mathcal{Z}} f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) = \int_{\mathcal{X}\times\mathcal{Z}} \mathbb{I}_{\mathcal{A}}(\boldsymbol{x})\, f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) \tag{52}$$

$$= \int_{\mathcal{U}} \mathbb{I}_{\mathcal{A}} \circ \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u})\, f \circ \boldsymbol{g}_{\mathbf{z}}(\boldsymbol{u})\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, \lambda^{D_{\mathsf{u}}}(\mathrm{d}\boldsymbol{u}). \tag{53}$$

Applying the co-area corollary (44) to the right-hand side and again noting $\mathbb{I}_{\mathcal{A}} \circ \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u})$ is constant across the fibre being integrated on, we have that $\forall \mathcal{A} \in \mathscr{G}$

$$\int_{\mathcal{A}\times\mathcal{Z}} f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) \tag{54}$$

$$= \int_{\mathcal{X}} \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} \mathbb{I}_{\mathcal{A}} \circ \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u})\, f \circ \boldsymbol{g}_{\mathbf{z}}(\boldsymbol{u})\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathsf{u}}-D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{u})\, \lambda^{D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{x}) \tag{55}$$

$$= \int_{\mathcal{X}} \mathbb{I}_{\mathcal{A}}(\boldsymbol{x}) \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} f \circ \boldsymbol{g}_{\mathbf{z}}(\boldsymbol{u})\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathsf{u}}-D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{u})\, \lambda^{D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{x}) \tag{56}$$

$$= \int_{\mathcal{A}} \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} f \circ \boldsymbol{g}_{\mathbf{z}}(\boldsymbol{u})\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathsf{u}}-D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{u})\, \lambda^{D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{x}). \tag{57}$$

Finally using that $\mathsf{P}_{\mathbf{x}}$ has a density $\mathsf{p}_{\mathbf{x}}$ with respect to the Lebesgue measure as shown in the previous proposition and so $\mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \mathsf{p}_{\mathbf{x}}(\boldsymbol{x})\lambda^{D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{x})$, we have that

$$\begin{aligned} &\int_{\mathcal{A}\times\mathcal{Z}} f(\boldsymbol{z})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) = \\ &\qquad \int_{\mathcal{A}} \frac{1}{\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})} \int_{\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} f \circ \boldsymbol{g}_{\mathbf{z}}(\boldsymbol{u})\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u})\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathcal{H}^{D_{\mathsf{u}}-D_{\mathsf{x}}}(\mathrm{d}\boldsymbol{u})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}). \end{aligned} \tag{58}$$

Here we ignore the points for which $\mathsf{p}_{\mathbf{x}}(\boldsymbol{x}) = 0$ as the set of all such points has zero measure under $\mathsf{P}_{\mathbf{x}}$ and so does not contribute to integrals against the probability measure $\mathsf{P}_{\mathbf{x}}$. Comparing to the definition of the conditional expectation (1) we have that (51) satisfies the definition. □

The expression derived for the conditional expectation has the form of an integral of function $f \circ \boldsymbol{g}_{\mathbf{z}}$ integrated against a density

$$\pi(\boldsymbol{u}) = \frac{1}{\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})}\, J_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathsf{p}_{\mathbf{u}}(\boldsymbol{u}) \tag{59}$$

which we can evaluate up to an unknown normalising constant $\mathsf{p}_{\mathbf{x}}(\boldsymbol{x})$. The key complicating factor is that the integral is now not across a Euclidean space, but an implicitly defined manifold corresponding to the fibre $\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$. However if we can construct a Markov transition operator which has an invariant distribution with density (59) with respect to the Hausdorff measure on the manifold, then we can use samples of the chain states $\{\mathbf{u}_s\}_{s=1}^{S}$ to compute an estimator

$$\hat{\mathsf{f}}_S = \frac{1}{S}\sum_{s=1}^{S}(f \circ \boldsymbol{g}_{\mathbf{z}}(\mathbf{u}_s)) \tag{60}$$

which providing the chain is also aperiodic and irreducible will be a consistent estimator for $\mathbb{E}[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{x}]$. Although constructing a Markov transition operator with the required properties is non-trivial, there is a significant body of existing work on methods for defining Markov chains on manifolds. We propose here to use a constrained Hamiltonian Monte Carlo method.

9. Constrained Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) [28, 68] is an auxiliary variable MCMC method which exploits the gradient of the density of the target distribution. The vector variable of interest, here the generator random inputs $\mathbf{u} \in \mathbb{R}^{D_{\mathbf{u}}}$ and that for the purposes of this section we will refer to as the *configuration state*, is augmented with a vector of auxiliary *momentum* variables $\mathbf{p} \in \mathbb{R}^{D_{\mathbf{u}}}$. Typically the momenta are specified to be independent of the configuration state with a zero mean normal distribution with covariance $\boldsymbol{M}$ termed the mass matrix, i.e

$$\mathsf{p}_{\mathbf{p}}(\boldsymbol{p}) = \mathcal{N}(\boldsymbol{p} \,|\, \mathbf{0}, \boldsymbol{M}) \propto \exp\left(-\frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p}\right). \tag{61}$$

The density π of the target distribution on the configuration state is used to define a *potential energy* function $\phi : \mathcal{U} \to \mathbb{R}$

$$\phi(\boldsymbol{u}) = -\log \pi(\boldsymbol{u}) - \log C \iff \pi(\boldsymbol{u}) = \frac{1}{C}\exp(-\phi(\boldsymbol{u})) \tag{62}$$

with C a normalising constant, and similarly the quadratic form $\frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p}$ corresponding to the negative logarithm of the unnormalised density on the momenta is termed the *kinetic energy*. The joint distribution on $\mathbf{u}$ and $\mathbf{p}$ then has a density proportional to $\exp(-h(\boldsymbol{u}, \boldsymbol{p}))$ where the Hamiltonian $h(\boldsymbol{u}, \boldsymbol{p})$ is

$$h(\boldsymbol{u}, \boldsymbol{p}) = \phi(\boldsymbol{u}) + \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p}. \tag{63}$$

The canonical Hamiltonian dynamic is then described by the system of ODEs

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \frac{\partial h}{\partial \boldsymbol{p}}^{\mathsf{T}} = \boldsymbol{M}^{-1}\boldsymbol{p}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial h}{\partial \boldsymbol{u}}^{\mathsf{T}} = -\nabla\phi(\boldsymbol{u}). \tag{64}$$

This dynamic is time-reversible, volume-preserving and exactly conserves the Hamiltonian. Symplectic integrators allow approximate integration of the Hamiltonian dynamic while maintaining the time-reversibility and volume-preservation properties. Further subject to stability bounds on the time-step, symplectic integrators will exactly conserve a 'nearby' Hamiltonian, and so the change in the Hamiltonian will remain bounded even over long simulated trajectories [50].

These properties make simulated Hamiltonian dynamics an ideal proposal mechanism for a Metropolis MCMC method. The Metropolis accept ratio for a proposal $(\boldsymbol{u}_\mathrm{p},\, \boldsymbol{p}_\mathrm{p})$ generated by simulating the dynamic N_s time steps forward from $(\boldsymbol{u},\, \boldsymbol{p})$ with a symplectic integrator and then negating the momentum, is simply $\exp\big(h(\boldsymbol{u},\, \boldsymbol{p}) - h(\boldsymbol{u}_\mathrm{p},\, \boldsymbol{p}_\mathrm{p})\big)$. Typically the change in the Hamiltonian will be small and so the probability of acceptance high. To ensure ergodicity, simulated dynamic moves are interleaved with updates independently sampling new momentum values from $\mathcal{N}(\boldsymbol{0},\, \boldsymbol{M})$.

In our case the target distribution on the configuration state u is defined on an implicitly defined manifold embedded in a Euclidean space $\mathcal{U} = \mathbb{R}^{D_\mathsf{u}}$. Intuitively we can consider the manifold as representing the allowable configurations of mechanical system subject to a constraint. By simulating a constrained Hamiltonian dynamic we can therefore construct a HMC transition operator analogous to that just described but that generates chains on an implicitly defined manifold rather than an unconstrained Euclidean space.

The use of constrained Hamiltonian dynamics within a MCMC method has been proposed by multiple authors. In the molecular dynamics literature, Hartmann and Schutte [41] and Lelièvre, Rousset and Stoltz [52] used simulated constrained Hamiltonian dynamics within a HMC framework to estimate free-energy profiles of molecular systems. Most relevantly for our case, Brubaker, Salzmann and Urtasun [19] proposed a constrained HMC algorithm for performing inference in target distributions defined on implicitly defined embedded manifolds. We will concentrate on the algorithm proposed in [19] here.

To simplify notation and emphasise the generality of the approach beyond our specific setting, we define the following notation for the vector *constraint function* on the system and corresponding Jacobian

$$\boldsymbol{c}(\boldsymbol{u}) = \boldsymbol{g}_\mathsf{x}(\boldsymbol{u}) - \boldsymbol{x}, \qquad \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u}) = \mathbf{J}_{\boldsymbol{g}_\mathsf{x}}(\boldsymbol{u}). \tag{65}$$

The *constraint manifold* is then defined as the zero level-set of $\boldsymbol{c}$, in our case corresponding to the fibre of $\boldsymbol{x}$ under the generator $\boldsymbol{g}_\mathsf{x}$

$$\mathcal{C} = \{\boldsymbol{u} \in \mathbb{R}^M : \boldsymbol{c}(\boldsymbol{u}) = \boldsymbol{0}\} = \boldsymbol{g}_\mathsf{x}^{-1}[\boldsymbol{x}]. \tag{66}$$

Defining as previously the potential energy ϕ as the negative logarithm of the unnormalised target density and the kinetic energy as a quadratic form $\frac{1}{2}\boldsymbol{p}^\mathsf{T}\boldsymbol{M}^{-1}\boldsymbol{p}$, the Hamiltonian for the constrained system can be written as

$$h(\boldsymbol{u}, \boldsymbol{p}) = \phi(\boldsymbol{u}) + \frac{1}{2}\boldsymbol{p}^\mathsf{T}\boldsymbol{M}^{-1}\boldsymbol{p} + \boldsymbol{c}(\boldsymbol{u})^\mathsf{T}\boldsymbol{\lambda}, \tag{67}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrangian multipliers for the constraints.

The constrained Hamiltonian dynamic is then defined by

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \frac{\partial h}{\partial \boldsymbol{p}} = \boldsymbol{M}^{-1}\boldsymbol{p}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial h}{\partial \boldsymbol{u}}^{\mathsf{T}} = -\nabla\phi(\boldsymbol{u})^{\mathsf{T}} - \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}, \tag{68}$$

with the Lagrange multipliers taking values to ensure the system constraints are satisfied. In addition to the configuration constraint $\boldsymbol{c}(\boldsymbol{u}) = \mathbf{0}$ there is a corresponding implied constraint on the momenta $\boldsymbol{p}$ requiring that the configuration velocity $\boldsymbol{M}^{-1}\boldsymbol{p}$ is always tangential to the constraint manifold at the current configuration, or equivalently that the momenta are in the *tangent space* to the constraint manifold. The tangent space $\mathscr{T}_{\boldsymbol{u}}\mathcal{C}$ at a configuration $\boldsymbol{u}$ is defined as

$$\mathscr{T}_{\boldsymbol{u}}\mathcal{C} = \left\{\boldsymbol{p} \in \mathbb{R}^M : \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})\boldsymbol{M}^{-1}\boldsymbol{p} = \mathbf{0}\right\}. \tag{69}$$

The complete set of valid configuration–momentum state pairs is termed the *tangent bundle* $\mathscr{T}\mathcal{C}$ of the constraint manifold and defined as

$$\mathscr{T}\mathcal{C} = \left\{\boldsymbol{u}, \boldsymbol{p} \in \mathbb{R}^M \times \mathbb{R}^M : \boldsymbol{c}(\boldsymbol{u}) = \mathbf{0},\ \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})\boldsymbol{M}^{-1}\boldsymbol{p} = \mathbf{0}\right\}. \tag{70}$$

The solution at time t to the initial value problem defined by the ODEs (68) defines a flow map $\boldsymbol{\gamma}_t : \mathscr{T}\mathcal{C} \to \mathscr{T}\mathcal{C}$ between states in the tangent bundle of the constraint manifold. As with the unconstrained Hamiltonian dynamics encountered previously, this flow map exactly conserves the Hamiltonian and is reversible under negation of the momenta. Further the flow map of the constrained dynamic is symplectic and conserves the volume element of the constraint manifold tangent bundle [50].

Importantly there exist symplectic integrators which can be used to approximate the constrained Hamiltonian dynamic flow map and which map between states exactly in the constraint manifold tangent bundle (modulo numerical error due to finite precision arithmetic). The approximate flow maps defined by these integrators are reversible and conserve the tangent bundle volume element. They also exhibit the bounded change in the Hamiltonian over simulated trajectories discussed previously for the unconstrained case.

A popular symplectic numerical integrator for constrained Hamiltonian dynamics is the RATTLE method [3, 51]. This a generalisation of the Störmer–Verlet or leapfrog integrator typically used to integrate the Hamiltonian dynamics in standard HMC, with additional steps to project the states on to the tangent bundle of the constraint manifold. A RATTLE step is composed of three component maps. The first map is defined by

$$\begin{gathered}\hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t}(\boldsymbol{u}, \boldsymbol{p}) = \Big(\boldsymbol{u} + \delta t \boldsymbol{M}^{-1}(\boldsymbol{p} - \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}),\ \boldsymbol{p} - \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}\Big) \\ \text{solving for } \boldsymbol{\lambda} \text{ such that } \boldsymbol{c}\Big(\boldsymbol{u} + \delta t \boldsymbol{M}^{-1}(\boldsymbol{p} - \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda})\Big) = \mathbf{0}.\end{gathered} \tag{71}$$

This defines an approximate *geodesic* step on the constraint manifold: the configuration $\boldsymbol{u}$ is incremented in the direction of the current velocity $\boldsymbol{M}^{-1}\boldsymbol{p}$ and then the new configuration state projected back on to the constraint manifold by solving a non-linear system of equations for the Lagrange multipliers $\boldsymbol{\lambda}$.

The second component map updates the momenta with a 'kick' in the direction of the potential energy gradient

$$\hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\delta t}(\boldsymbol{u},\boldsymbol{p}) = \Big(\boldsymbol{u},\, \boldsymbol{p} - \delta t \nabla\phi(\boldsymbol{u})^{\mathsf{T}}\Big). \tag{72}$$

Though both $\hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t}$ and $\hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\delta t}$ steps will map between configurations in the constraint manifold (trivially in the case of $\hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\delta t}$ as the configurations are kept fixed), the corresponding momenta will not be confined to the tangent spaces to the manifold. The final component map projects the momenta in to the tangent space of the constraint manifold at the current configuration. It is defined by

$$\begin{gathered}\hat{\boldsymbol{\gamma}}^{\mathrm{P}}(\boldsymbol{u},\boldsymbol{p}) = \Big(\boldsymbol{u},\, \boldsymbol{p} - \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}\Big) \\ \text{solving for } \boldsymbol{\lambda} \text{ such that } \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})\boldsymbol{M}^{-1}(\boldsymbol{p} - \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})^{\mathsf{T}}\boldsymbol{\lambda}) = \mathbf{0}.\end{gathered} \tag{73}$$

In this case the system of equation needing to be solved is linear and has an analytic solution, giving the following closed-form definition

$$\hat{\boldsymbol{\gamma}}^{\mathrm{P}}(\boldsymbol{u},\boldsymbol{p}) = \Big(\boldsymbol{u},\, \boldsymbol{p} - \mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})^{\mathsf{T}}(\mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})\boldsymbol{M}^{-1}\mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})^{\mathsf{T}})^{-1}\mathbf{J}_{\boldsymbol{c}}(\boldsymbol{u})\boldsymbol{M}^{-1}\boldsymbol{p}\Big). \tag{74}$$

An overall RATTLE step is then defined by the composition

$$\hat{\boldsymbol{\gamma}}^{\mathrm{R}}_{\delta t} = \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}}. \tag{75}$$

In practice the intermediate momentum projection steps $\hat{\boldsymbol{\gamma}}^{\mathrm{P}}$ are redundant [58] and so typically the momentum is only projected back in to the tangent space at the end of the step, giving the following update

$$\hat{\boldsymbol{\gamma}}^{\mathrm{R}}_{\delta t} = \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}}. \tag{76}$$

Solving the non-linear constraint equations in the geodesic step $\hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\delta t}$ is computationally challenging, with closed form solutions generally not available and so an iterative approach required. Further the system of equations are not guaranteed to have a unique solution: if the step size δt is too large there can be multiple or no solutions [50]. It is important therefore to keep the step size small enough to avoid the iterative solver converging to an incorrect solution or not converging at all. Often the resulting step size will be smaller than required however in terms of controlling the Hamiltonian error over a simulated trajectory. An alternative to the standard RATTLE integrator is therefore to perform $N_g > 1$ inner geodesic steps $\hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\frac{\delta t}{N_g}}$ for each outer pair of momentum kick steps $\hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}}$

$$\hat{\boldsymbol{\gamma}}^{\mathrm{G}}_{\delta t} = \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}} \circ \big(\hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{A}}_{\frac{\delta t}{N_g}}\big)^{N_g} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{P}} \circ \hat{\boldsymbol{\gamma}}^{\mathrm{B}}_{\frac{\delta t}{2}}. \tag{77}$$

This *geodesic integrator* [49, 48] scheme can reduce the number of potential energy gradient evaluations required by using a larger step size for the momentum kick updates while still maintaining a sufficiently small step size to avoid convergence issues in the geodesic step.

Assuming the iterative solving of the projections to constraint manifold in the geodesic steps converge correctly, the approximate flow map defined by iterating RATTLE or geodesic integrator steps preserves the volume element of $\mathscr{T}\mathcal{C}$ and is reversible under negation of the momenta. We can therefore use the composition of the approximate flow map with a momentum reversal operator to define a volume-preserving involution between states in $\mathscr{T}\mathcal{C}$. We can then use this involution as a proposal generating mechanism for a Metropolis accept step to correct for the Hamiltonian error in the approximate flow map.

As in the standard HMC algorithm, Metropolis updates with approximate flow map proposals are interleaved with updates in which the momenta are independently resampled. To ensure the momenta remain in the tangent space $\mathscr{T}_{\boldsymbol{u}}\mathcal{C}$ to the constraint manifold after generating new values from $\mathcal{N}(\mathbf{0}, \boldsymbol{M})$, the momenta are projected in to the tangent space using the projection operator defined in (74). The overall constrained HMC transition operator defined by this combination of momentum resampling and Metropolis accept step with a constrained dynamic proposal, leaves invariant the distribution with negative log density defined by the Hamiltonian in (67) on the constraint manifold tangent bundle $\mathscr{T}\mathcal{C}$, and so marginally leaves the target distribution on $\mathcal{C}$ invariant.

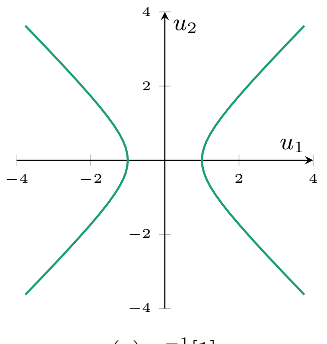
Ensuring ergodicity of chains generated by the constrained HMC transition operator is in general more challenging than for HMC on Euclidean spaces due to the often complex geometry of the constraint manifold $\mathcal{C}$ and potential for numerical issues in the projection steps. In [19] it is shown that if[3]

- $\mathcal{C}$ is a connected, smooth differentiable manifold,
- $\mathbf{J}_{\boldsymbol{c}}$ has full row-rank everywhere,
- and $\pi(\boldsymbol{u}) \propto \exp(-\phi(\boldsymbol{u}))$ is smooth and strictly positive on $\mathcal{C}$

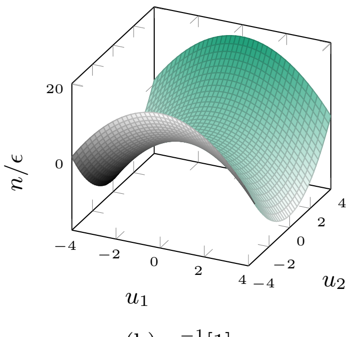
for a constrained HMC transition using an approximate flow map defined by a symplectic integrator with step size δt, if the integrator step size δt is set sufficiently small such that there is a unique solution to the choice of Lagrange multipliers $\boldsymbol{\lambda}$ in each geodesic step (71) and the iterative method employed converges to this solution in every step, that the overall transition operator will be irreducible, aperiodic and leave the target distribution on $\mathcal{C}$ invariant.

These conditions put stricter requirements on the generator $\boldsymbol{g}_{\mathbf{x}}$ of a differentiable generative model than those specified in Definition 2 and Proposition 2 if we wish to use a constrained HMC method to estimate conditional expectations under the model. The requirement that $\mathcal{C} = \boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$ is a smooth and connected manifold is likely to be challenging to check for complex generators. If the fibre of $\boldsymbol{x}$ under the generator $\boldsymbol{g}_{\mathbf{x}}$ consists of multiple disconnected components then the constrained Hamiltonian dynamic will remain confined to just one of them. Although problematic, this issue is similar to that faced by other MCMC methods in target distributions with multiple separated modes. The requirement that the Jacobian $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ is defined and full row-rank everywhere is also stricter than previously required.

[3]We give only a loose statement of the full conditions here for brevity; for complete details see Theorems 1 to 4 in [19].



(a) $\boldsymbol{g}_{\mathsf{x}}^{-1}[1]$



(b) $\boldsymbol{g}_{\mathsf{y}}^{-1}[1]$

Fig 2: Visualisations of the hyperbola fibre $\boldsymbol{g}_{\mathsf{x}}^{-1}[1]$ of the generator $\boldsymbol{g}_{\mathsf{x}}$ defined in (79) consisting of two disconnected components and the corresponding connected hyperbolic paraboloid fibre $\boldsymbol{g}_{\mathsf{y}}^{-1}[1]$ of the noisy generator.

If we define an augmented 'noisy' generator

$$\boldsymbol{g}_{\mathbf{y}}(\boldsymbol{u}, \boldsymbol{n}) = \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}) + \epsilon \boldsymbol{n} \tag{78}$$

with $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and ϵ a small positive constant, then if $\boldsymbol{g}_{\mathbf{x}}$ is differentiable everywhere then the Jacobian of the augmented generator $\mathbf{J}_{\boldsymbol{g}_{\mathsf{y}}}$ will be full row-rank everywhere. Further in some cases the fibres under the noisy generator $\boldsymbol{g}_{\mathbf{y}}^{-1}[\boldsymbol{x}]$ will be connected when the fibres under the original generator $\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$ are not. As a simple example consider

$$\boldsymbol{g}_{\mathsf{x}}(\boldsymbol{u}) = u_1^2 - u_2^2, \qquad \boldsymbol{g}_{\mathsf{x}}(\boldsymbol{u}, n) = u_1^2 - u_2^2 + \epsilon n. \tag{79}$$

The fibres $\boldsymbol{g}_{\mathsf{x}}^{-1}[x]$ are hyperbola in $\mathbb{R}^2$, for $x \neq 0$ consisting of two disconnected components as shown in Figure 2a. The fibres of $\boldsymbol{g}_{\mathsf{y}}^{-1}[x]$ are connected hyperbolic paraboloids in $\mathbb{R}^3$ as shown in Figure 2b.

This noisy augmentation of the generator corresponds to using an ABC approach with a Gaussian kernel with tolerance ϵ, and so we could instead perform standard HMC in the ABC posterior density in the generator input space (35). The potential energy function corresponding to (35) in this case is

$$\phi(\boldsymbol{u}) = \frac{1}{2\epsilon^2}(\boldsymbol{x} - \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}))^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u})) - \log \mathsf{p}_{\mathbf{u}}(\boldsymbol{u}), \tag{80}$$

The energy function combines a term favouring inputs $\mathbf{u}$ which generate outputs close to the observed data $\boldsymbol{x}$ and prior term favouring input values which are plausible under $\mathsf{P}_{\mathbf{u}}$. Typically the input density $\mathsf{p}_{\mathbf{u}}$ will have a simple form e.g. standard normal $\mathcal{N}(\boldsymbol{u} \,|\, \mathbf{0}, \mathbf{I})$ in which case the main complexity in the target density arises from the term due to the Gaussian kernel k_ϵ and generator function $\boldsymbol{g}_{\mathbf{x}}$; this term puts high density on inputs close to the fibre $\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$ of the observed data $\boldsymbol{x}$ under the generator function $\boldsymbol{g}_{\mathbf{x}}$. For small ϵ this will mean the distribution

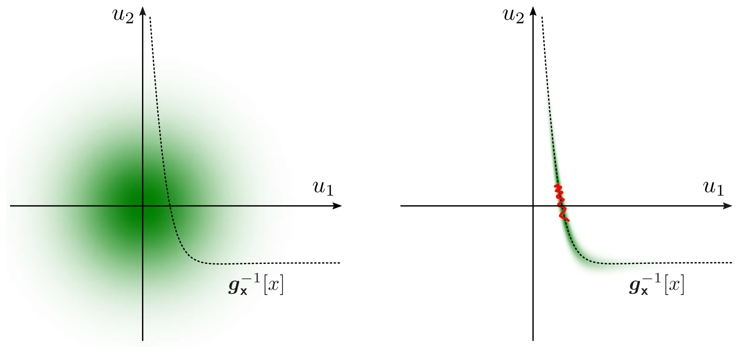

u_2
u_1
$\boldsymbol{g}_{\mathsf{x}}^{-1}[x]$
u_2
u_1
$\boldsymbol{g}_{\mathsf{x}}^{-1}[x]$

Fig 3: Illustration of oscillatory behaviour in HMC trajectories when using an ABC posterior density (35) in the input space to a generative model. The left axis shows the two-dimensional input space $\mathcal{U}$ of a differentiable generative model with a Gaussian input density $\mathsf{p}_{\mathbf{u}}$ (green shading). The dashed curve shows the one-dimensional manifold corresponding to the pre-image under the generator function $\boldsymbol{g}_{\mathsf{x}}$ of an observed output x. The right axis shows the same input space with now the green shading showing the density proportional to $k_\epsilon(x; \boldsymbol{g}_{\mathsf{x}}(\boldsymbol{u}))\,\mathsf{p}_{\mathbf{u}}(\boldsymbol{u})$ with a Gaussian k_ϵ. The red curve shows a corresponding simulated HMC trajectory: the large magnitude density gradients normal to the manifold cause high-frequency oscillations and slows movement along the manifold (which corresponds to variation in the latent variable z).

in the input space is increasingly tightly concentrated in a narrow 'ridge' around the manifold embedded in the input space corresponding to $\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$. Although the gradient-based Hamiltonian dynamic is able to propose moves which remain within this high-density region, the strong gradients normal to the manifold tends to produce trajectories which oscillate back and forth across the ridge, limiting the motion tangential to the manifold and requiring a small integrator step-size for stability; this is illustrated in a simple model with a two dimensional input space in Figure 3. In some cases (examples of which will be shown in the numerical experiments in Section 12) applying constrained HMC with the noisy generator $\boldsymbol{g}_{\mathbf{y}}$ can therefore be more efficient than running standard HMC in the ABC target density, despite the much higher per-step costs, as constrained HMC updates are able to use a much larger integrator step size when using small ϵ.

Riemannian manifold Hamiltonian Monte Carlo (RMHMC) [37] extends the standard HMC algorithm by introducing momenta with a configuration-dependent covariance matrix $\boldsymbol{G} : \mathbb{R}^{D_{\mathbf{u}}} \to \mathbb{R}^{D_{\mathbf{u}} \times D_{\mathbf{u}}}$, typically termed the metric. The metric, which is required to be positive-definite almost-everywhere in the configuration space, is able to condition the momenta to adjust for locally varying curvature in the target density, potentially significantly improving the ability of the simulated Hamiltonian dynamic to explore the configuration space. An alternative approach to remedying the issues with performing standard HMC in the generator input space is therefore to apply a RMHMC algorithm using a metric exploiting the geometry of the target density to improve the behaviour of the simulated dynamic. For example the metric

$$\boldsymbol{G}(\boldsymbol{u}) = \frac{1}{\epsilon^2}\mathbf{J}_{\boldsymbol{g}_\mathbf{x}}(\boldsymbol{u})^\mathsf{T}\mathbf{J}_{\boldsymbol{g}_\mathbf{x}}(\boldsymbol{u}) + \mathbf{I} \tag{81}$$

is positive definite everywhere and equal to the Hessian of the potential energy (80) for $\boldsymbol{u} \in \boldsymbol{g}_\mathbf{x}^{-1}[\boldsymbol{x}]$. Using this metric, for small ϵ and inputs $\boldsymbol{u}$ generating outputs close to the data $\boldsymbol{x}$ i.e. small values of $\frac{1}{\epsilon}\|\boldsymbol{g}_\mathbf{x}(\boldsymbol{u}) - \boldsymbol{x}\|_2$, the velocity in the RMHMC dynamic $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{G}(\boldsymbol{u})^{-1}\boldsymbol{p}$ will tend to be higher along the directions tangential to the fibre $\boldsymbol{g}_\mathbf{x}^{-1}[\boldsymbol{x}]$, reducing the tendency for the dynamic to oscillate normal to the fibre. RMHMC requires use of a computationally costly implicit integrator due to the non-separable Hamiltonian and so like the constrained HMC method proposed here has a significantly higher computational cost per sample than the standard HMC algorithm. However as with constrained HMC the potential for improved exploration of the space for small ϵ may compensate for the more costly updates. We do not explore this idea further here but it may be an interesting avenue for future work.

Geodesic Monte Carlo [20] also considers applying a HMC scheme to sample from non-linear manifolds embedded in a Euclidean space. Similarly to [19] however the motivation is performing inference with respect to distributions explicitly defined on a manifold such as directional statistics. The method presented in [20] uses an exact solution for the geodesic flow on the manifold. The use of a geodesic integration scheme within a constrained HMC update as discussed here can be considered an extension for cases when an exact geodesic solution is not available. Instead the geodesic flow is approximately simulated while still maintaining the required volume-preservation and reversibility properties for validity of the overall HMC scheme.

An alternative Metropolis method for sampling from densities defined on manifolds embedded in a Euclidean space is proposed in [100]. Compared to constrained HMC this alleviates the requirements to calculate the gradient of (the logarithm of) the target density on the manifold, though still requires evaluation of the constraint function Jacobian. As discussed in Section 4, using reverse-mode AD the gradient of the target density can be computed at a cost proportional to evaluation of the target density itself. In general we would expect exploiting the gradient of the target density on the manifold within a simulated Hamiltonian dynamic to lead to more coherent exploration of the target distribution, instead of the more random-walk behaviour of a non-gradient based Metropolis update, and so for the gradient evaluation overhead to be worthwhile.

There is extensive theoretical discussion of the issues involved in sampling from distributions defined on manifolds in [26], including a derivation of conditional densities on a manifold using the co-area formula which directly motivated our earlier derivations of expressions for conditional expectations under a differentiable generative model. The experiments in [26] are mainly concentrated on expository examples using simple parameterised manifolds such as a torus embedded in $\mathbb{R}^3$ and conditional testing in exponential family distributions.

Algorithm 1 Constrained HMC in a differentiable generative model

Input:

$\boldsymbol{g}_{\mathbf{x}}$: observed variable generator function;
ϕ : potential energy function $\phi(\boldsymbol{u}) = -\log \mathsf{p}_{\mathbf{u}}(\boldsymbol{u}) + \frac{1}{2}\log|\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})|$;
$\boldsymbol{x}$: observed data values being conditioned on;
$\boldsymbol{u}$: current chain state (model inputs) with $\|\boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}) - \boldsymbol{x}\|_\infty < \epsilon$;
$(\varphi, \boldsymbol{J}, \boldsymbol{L})$: cached values of ϕ, $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ and $\mathrm{chol}\big(\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}{}^{\mathsf{T}}\big)$ evaluated at $\boldsymbol{u}$;
ϵ : convergence tolerance for Newton iteration;
I : number of Newton iterations to try before rejecting for non-convergence;
δt : integrator time step; N_s : number of time steps to simulate;
N_g : number of geodesic steps per time step.

Output:

$\boldsymbol{u}_{\mathrm{n}}$: new chain state with $\|\boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}_{\mathrm{n}}) - \boldsymbol{x}\|_\infty < \epsilon$;
$(\varphi_{\mathrm{n}}, \boldsymbol{J}_{\mathrm{n}}, \boldsymbol{L}_{\mathrm{n}})$: values of ϕ, $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ and $\mathrm{chol}\big(\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}{}^{\mathsf{T}}\big)$ evaluated at new $\boldsymbol{u}_{\mathrm{n}}$.

```
$\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\boldsymbol{p} \leftarrow$ ProjectMom$(\boldsymbol{n}, \boldsymbol{J}, \boldsymbol{L})$
$\boldsymbol{u}_{\mathrm{p}}, \boldsymbol{p}_{\mathrm{p}}, \boldsymbol{J}_{\mathrm{p}}, \boldsymbol{L}_{\mathrm{p}} \leftarrow$ SimDyn$(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
$\varphi_{\mathrm{p}} \leftarrow \phi(\boldsymbol{u})$
$r \sim \mathcal{U}(0,1)$
$p_a \leftarrow \exp\big(\varphi + \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{p} - \varphi_{\mathrm{p}} - \frac{1}{2}\boldsymbol{p}_{\mathrm{p}}^{\mathsf{T}}\boldsymbol{p}_{\mathrm{p}}\big)$
if $r < p_a$
    $\boldsymbol{u}_{\mathrm{n}}, \varphi_{\mathrm{n}}, \boldsymbol{J}_{\mathrm{n}}, \boldsymbol{L}_{\mathrm{n}} \leftarrow \boldsymbol{u}_{\mathrm{p}}, \varphi_{\mathrm{p}}, \boldsymbol{J}_{\mathrm{p}}, \boldsymbol{L}_{\mathrm{p}}$
else
    $\boldsymbol{u}_{\mathrm{n}}, \varphi_{\mathrm{n}}, \boldsymbol{J}_{\mathrm{n}}, \boldsymbol{L}_{\mathrm{n}} \leftarrow \boldsymbol{u}, \varphi, \boldsymbol{J}, \boldsymbol{L}$

function SimDyn$(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
    $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \frac{\delta t}{2}\nabla\phi(\boldsymbol{u})^{\mathsf{T}}$
    $\boldsymbol{p} \leftarrow$ ProjectMom$(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
    $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L} \leftarrow$ SimGeo$(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
    for $s \in \{2 \dots N_s\}$
        $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \delta t \nabla\phi(\boldsymbol{u})^{\mathsf{T}}$
        $\boldsymbol{p} \leftarrow$ ProjectMom$(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
        $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L} \leftarrow$ SimGeo$(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
    $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \frac{\delta t}{2}\nabla\phi(\boldsymbol{u})^{\mathsf{T}}$
    $\boldsymbol{p} \leftarrow$ ProjectMom$(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
    return $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L}$

function ProjectMom$(\boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
    return $\boldsymbol{p} - \boldsymbol{J}^{\mathsf{T}}\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{J}\boldsymbol{p}$

function ProjectPos$(\boldsymbol{u}, \boldsymbol{J}, \boldsymbol{L})$
    $\boldsymbol{\delta} \leftarrow \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}) - \boldsymbol{x}$
    $i \leftarrow 0$
    while $\|\boldsymbol{\delta}\|_\infty > \epsilon$ and $i < I$
        $\boldsymbol{u} \leftarrow \boldsymbol{u} - \boldsymbol{J}^{\mathsf{T}}\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{\delta}$
        $\boldsymbol{\delta} \leftarrow \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}) - \boldsymbol{x}$
        $i \leftarrow i + 1$
    if $i = I$
        raise RejectMove
    return $\boldsymbol{u}$

function SimGeo$(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
    for $i \in \{1 \dots N_g\}$
        $\tilde{\boldsymbol{u}} \leftarrow \boldsymbol{u} + \frac{\delta t}{N_g}\boldsymbol{p}$
        $\boldsymbol{u}' \leftarrow$ ProjectPos$(\tilde{\boldsymbol{u}}, \boldsymbol{J}, \boldsymbol{L})$
        $\boldsymbol{J} \leftarrow \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u}')$
        $\boldsymbol{L} \leftarrow \mathrm{chol}\big(\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}\big)$
        $\tilde{\boldsymbol{p}} \leftarrow \frac{N_g}{\delta t}(\boldsymbol{u}' - \boldsymbol{u})$
        $\boldsymbol{p} \leftarrow$ ProjectMom$(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
        $\boldsymbol{u}_r \leftarrow \boldsymbol{u}' - \frac{\delta t}{N_g}\boldsymbol{p}$
        $\boldsymbol{u}_r \leftarrow$ ProjectPos$(\boldsymbol{u}_r, \boldsymbol{J}, \boldsymbol{L})$
        if $\|\boldsymbol{u} - \boldsymbol{u}_r\|_\infty > \sqrt{\epsilon}$
            raise RejectMove
        $\boldsymbol{u} \leftarrow \boldsymbol{u}'$
    return $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L}$
```

10. Implementation details

The constrained HMC implementation we propose for performing inference in differentiable generative models is shown in Algorithm 1. This algorithm differs in some details from that proposed [19] and we discuss these differences and computational issues specific to our setting in the following subsections.

10.1. Iterative solver for projection on to manifold

Rather than the RATTLE integrator used in [19], we use the geodesic integrator generalisation discussed in the previous section to simulate the constrained dynamic. This gives increased flexibility in balancing the need for an appropriately small step-size to ensure convergence of the iterative solution of the equations projecting on to the constraint manifold and using a more efficient larger step size for updates to the momentum due to the potential energy gradient. We assume $\boldsymbol{M} = \mathbf{I}$ here; other mass matrix choices can be implemented by reparameterising the model with an initial linear transformation stage in the generator.

The projection on to the constraint manifold in the geodesic steps is performed in the function ProjectPos in Algorithm 1. We use a quasi-Newton method for solving for $\boldsymbol{\lambda}$ the system of equations $\boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u} + (^{\delta t}/_{N_g})\boldsymbol{p} - \boldsymbol{J}^\mathsf{T}\boldsymbol{\lambda}) = \boldsymbol{x}$ where $\boldsymbol{J} = \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u})$. Expressing directly in terms of the configuration state $\boldsymbol{u}$ rather than the Lagrange multipliers, the full Newton update would be

$$\boldsymbol{u}' \leftarrow \boldsymbol{u}' - \boldsymbol{J}^\mathsf{T}\left(\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}(\boldsymbol{u}')\boldsymbol{J}^\mathsf{T}\right)^{-1}(\boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}') - \boldsymbol{x}). \tag{82}$$

This requires recalculating the Jacobian and solving a dense linear system within the optimisation loop. Instead as proposed in [6] we use a symmetric quasi-Newton update,

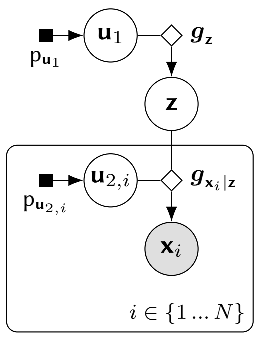
$$\boldsymbol{u}' \leftarrow \boldsymbol{u}' - \boldsymbol{J}^\mathsf{T}\left(\boldsymbol{J}\boldsymbol{J}^\mathsf{T}\right)^{-1}(\boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}') - \boldsymbol{x}). \tag{83}$$

The Jacobian product $\boldsymbol{J}\boldsymbol{J}^\mathsf{T}$ is used to condition the moves. This matrix is positive-definite and a Cholesky decomposition can be calculated outside the optimisation loop allowing cheaper quadratic cost solves within the loop.

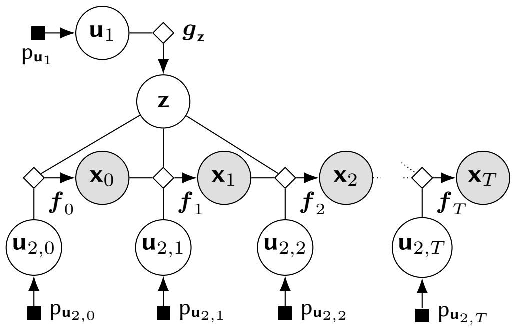
Convergence of the quasi-Newton iteration is signalled when the maximum absolute difference between the generated observed variables and the observed data is below a tolerance ϵ, i.e. $\|\boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}) - \boldsymbol{x}\|_\infty < \epsilon$. The tolerance is analogous to the ϵ parameter in ABC methods, however here we can set this value close to machine precision ($\epsilon = 10^{-8}$ in the experiments here) and so the error introduced is comparable to that otherwise incurred for using non-exact arithmetic.

In some cases the quasi-Newton iteration will fail to converge. We use a fixed upper limit on the number of iterations and reject the move (line 34 in Algorithm 1) if convergence is not achieved within this limit. To ensure reversibility, once we have solved for a forward geodesic step on the manifold in SimGeo, we then check if the corresponding reverse step (with the momentum negated) returns to the original position and reject if not. This involves running a second Newton iteration, though as it reuses the same Jacobian $\boldsymbol{J}$ and Cholesky factor $\boldsymbol{L}$, the evaluation of which tend to be the dominant costs in the algorithm, we found the overhead introduced tended to be quite small (around a 20% increase in run-time compared to only performing the forward step). A similar scheme for ensuring reversibility is proposed in [100].

The square root of the tolerance ϵ used for the Newton convergence check *in the output space of generator* (line 29 in Algorithm 1) is used for the reverse-step



(a) Independent $\mathbf{x}_i$



(b) Markovian $\mathbf{x}_i$

Fig 4: Factor graphs of examples of structured directed generative models.

check on *the inputs* (line 48 in Algorithm 1) based on standard recommendations for checking convergence in optimisation routines [22]. In the implementation used in the experiments, we fall back to a MINPACK [63] implementation of Powell's Hybrid method [76] if the quasi-Newton iteration fails to converge, with a rejection then only occurring if both iterative solvers fail. In practice we found if the step size δt and number of geodesic steps N_g are chosen appropriately then rejections due to non-convergence or non-reversible steps occur rarely.

10.2. Exploiting model structure

For larger systems, the Cholesky decomposition of the Jacobian matrix product $\mathbf{J}_{\boldsymbol{g}_\mathbf{x}}\mathbf{J}_{\boldsymbol{g}_\mathbf{x}}{}^{\mathsf{T}}$ (line 42) will become a dominant cost, generally scaling cubically with $D_\mathbf{x}$. In many models however conditional independency structure will mean that not all observed variables $\mathbf{x}$ are dependent on all of the input variables $\mathbf{u}$ and so the Jacobian $\mathbf{J}_{\boldsymbol{g}_\mathbf{x}}$ has a sparse structure which can be exploited to reduce this worst-case cost. In particular two common cases are directed generative models in which the observed variables $\mathbf{x}$ can be split into groups $\{\mathbf{x}_i\}_{i=1}^{G}$ such that all of the $\mathbf{x}_i$ are either conditionally independent given the latent variables $\mathbf{z} = \boldsymbol{g}_\mathbf{z}(\mathbf{u}_1)$ (for example a model for a *independent and identically distributed* (IID) dataset), or each $\mathbf{x}_i$ is conditionally independent of all $\{\mathbf{x}_j\}_{j<i-1}$ given $\mathbf{x}_{i-1}$ and $\mathbf{z}$ (most commonly Markov chains for example from simulation of a SDE model, though more general tree structured dependencies can also be ordered into this form).

Figure 4 shows factor graphs for directed generative models with these two structures, with the conditional independencies corresponding to each $\mathbf{x}_i$ being generated as a function of only a subset $\mathbf{u}_{2,i}$ of the random input variables $\mathbf{u}_2$. We assume here each $\mathbf{x}_i$ vector has the same dimensionality as the corresponding random input vector $\mathbf{u}_{2,i}$. For models with these structures the generator Jacobian

$$\mathbf{J}_{\boldsymbol{g}_\mathbf{x}} = \left[\frac{\partial \boldsymbol{g}_\mathbf{x}}{\partial \boldsymbol{u}_1} \middle| \frac{\partial \boldsymbol{g}_\mathbf{x}}{\partial \boldsymbol{u}_2} \right] \tag{84}$$

has a component ${}^{\partial \boldsymbol{g}_{\mathbf{x}}}/_{\partial \boldsymbol{u}_2}$ which is either block-diagonal (independent) or block-triangular (Markovian). Considering first the simplest case where each $(\mathbf{x}_i, \mathbf{u}_{2,i})$ pair are single dimensional, the Cholesky decomposition of

$$\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}} \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}^{\mathsf{T}} = \frac{\partial \boldsymbol{g}_{\mathbf{x}}}{\partial \boldsymbol{u}_1} \frac{\partial \boldsymbol{g}_{\mathbf{x}}}{\partial \boldsymbol{u}_1}^{\mathsf{T}} + \frac{\partial \boldsymbol{g}_{\mathbf{x}}}{\partial \boldsymbol{u}_2} \frac{\partial \boldsymbol{g}_{\mathbf{x}}}{\partial \boldsymbol{u}_2}^{\mathsf{T}} \tag{85}$$

can then be computed by low-rank Cholesky updates of the triangular or diagonal matrix ${}^{\partial \boldsymbol{g}_{\mathbf{x}}}/_{\partial \boldsymbol{u}_2}$ with each of the columns of ${}^{\partial \boldsymbol{g}_{\mathbf{x}}}/_{\partial \boldsymbol{u}_1}$. As $\dim(\boldsymbol{u}_1) = L$ is often significantly less than the number of observations being conditioned on $D_{\mathbf{x}}$, the resulting $\mathcal{O}(LD_{\mathbf{x}}^2)$ cost of the low-rank Cholesky updates is a significant improvement over the original $\mathcal{O}(D_{\mathbf{x}}^3)$.

For cases in which each $(\mathbf{x}_i, \mathbf{u}_{2,i})$ pair are both vectors of dimension D and so ${}^{\partial \boldsymbol{g}_{\mathbf{x}}}/_{\partial \boldsymbol{u}_2}$ is block diagonal or triangular, then the Cholesky factorisation of $({}^{\partial \boldsymbol{g}_{\mathbf{x}}}/_{\partial \boldsymbol{u}_2})({}^{\partial \boldsymbol{g}_{\mathbf{x}}}/_{\partial \boldsymbol{u}_2})^{\mathsf{T}}$ can be computed at a cost $\mathcal{O}(GD^3)$ for block diagonal, and $\mathcal{O}(G^2D^3)$ for block triangular ${}^{\partial \boldsymbol{g}_{\mathbf{x}}}/_{\partial \boldsymbol{u}_2}$, with then again $\mathcal{O}(LD_{\mathbf{x}}^2)$ cost low-rank updates of this Cholesky factor by the columns of ${}^{\partial \boldsymbol{g}_{\mathbf{x}}}/_{\partial \boldsymbol{u}_1}$ performed.

10.3. Efficiently evaluating the potential energy and gradient

The Metropolis accept step and momentum updates in the SimDyn routine require evaluating the potential energy corresponding to (59) and its gradient respectively. Although this can by achieved by directly using the expression given in (59) (and applying reverse-mode AD to get the gradient), both the potential energy and its gradient can be more efficiently calculated by reusing the Cholesky decomposition of the constraint Jacobian Gram matrix computed in line 42.

Dropping the dependence of the Jacobian on $\boldsymbol{u}$ for brevity we have that the potential energy ϕ corresponding to the negative logarithm of the unnormalised target density on the manifold (59) is

$$\phi(\boldsymbol{u}) = \frac{1}{2} \log \left| \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}} \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}^{\mathsf{T}} \right| - \log \mathsf{p}_{\mathbf{u}}(\boldsymbol{u}) \tag{86}$$

In general evaluating the determinant $|\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}} \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}^{\mathsf{T}}|$ has computational cost which scales as $\mathcal{O}(D_{\mathbf{u}} D_{\mathbf{x}}^2)$. However the lower-triangular Cholesky decomposition $\boldsymbol{L}$ of $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}} \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}^{\mathsf{T}}$ is already calculated in the SimGeo routine in Algorithm 1. Using basic properties of the matrix determinant

$$\phi(\boldsymbol{u}) = \sum_{i=1}^{D_{\mathbf{x}}} \log(L_{ii}) - \log \mathsf{p}_{\mathbf{u}}(\boldsymbol{u}). \tag{87}$$

Given the Cholesky factor $\boldsymbol{L}$ we can therefore can evaluate the potential energy ϕ at a marginal computational cost that scales linearly with $D_{\mathbf{x}}$. For the gradient we can use reverse-mode AD to calculate the derivative of (87) with respect to $\boldsymbol{u}$. This requires propagating derivatives through the Cholesky decomposition [64]; implementations for this are present in many AD frameworks.

Alternatively using the standard result for the derivative of a log determinant and the invariance of the trace to cyclic permutations we have that the gradient of the log determinant term in (86) can be manipulated in to the form

$$\frac{1}{2}\frac{\partial}{\partial u_i}\log\left|\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}{}^\mathsf{T}\right| = \operatorname{trace}\left(\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}{}^\mathsf{T}\left(\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}{}^\mathsf{T}\right)^{-1}\frac{\partial \mathbf{J}_{\boldsymbol{g}_\mathsf{x}}}{\partial u_i}\right) \tag{88}$$

We denote the matrix vectorisation operator vec such that for a $M \times N$ matrix $\boldsymbol{A}$, we have $\operatorname{vec}(\boldsymbol{A}) = [A_{1,1}, \dots, A_{M,1}, A_{1,2}, \dots, A_{N,M}]^\mathsf{T}$. Then as the trace of a matrix product defines an inner product we have that $\operatorname{trace}(\boldsymbol{A}\boldsymbol{B}) = \operatorname{vec}(\boldsymbol{A})^\mathsf{T}\operatorname{vec}(\boldsymbol{B})$. We can therefore write the gradient of the log determinant term as

$$\frac{1}{2}\frac{\partial}{\partial \boldsymbol{u}}\log\left|\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}{}^\mathsf{T}\right| = \operatorname{vec}\left(\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}{}^\mathsf{T}\left(\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}{}^\mathsf{T}\right)^{-1}\right)^\mathsf{T}\frac{\partial \operatorname{vec}\left(\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}\right)}{\partial \boldsymbol{u}} \tag{89}$$

The matrix inside the left vec operator can be computed once by reusing the Cholesky factorisation of $\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}{}^\mathsf{T}$ to solve the system of equations by forward and backward substitution. We then have an expression in the form of a vector-Jacobian product which is provided as an efficient primitive in many AD frameworks, e.g. as `Lop` in Theano, and like the gradient (which is actually a special case) can be evaluated at cost which is a constant over head of evaluating the forward function (i.e. the cost of evaluating $\mathbf{J}_{\boldsymbol{g}_\mathsf{x}}$ here).

10.4. Initialising the state

A final implementation detail is the requirement to find an initial $\boldsymbol{u}$ satisfying $\boldsymbol{g}_\mathsf{x}(\boldsymbol{u}) = \boldsymbol{x}$ to initialise the chain at. In directed generative models with one of the structures described in Section 10.2, a method we found worked well in the experiments was to sample a $\boldsymbol{u}_1$, $\boldsymbol{u}_2$ pair from $\mathsf{P}_\mathbf{u}$ and then keeping the $\boldsymbol{u}_1$ values fixed, solve $\boldsymbol{g}_{\mathsf{x}|\mathsf{z}}(\boldsymbol{g}_\mathsf{z}(\boldsymbol{u}_1), \boldsymbol{u}_2) = \boldsymbol{x}$ for $\boldsymbol{u}_2$ using for example Newton's method or by directly minimising the Euclidean norm $\|\boldsymbol{g}_\mathsf{x}(\boldsymbol{g}_\mathsf{z}(\boldsymbol{u}_1), \boldsymbol{u}_2) - \boldsymbol{x}\|_2^2$ with respect to $\boldsymbol{u}_2$ by gradient descent. In more general cases one strategy is to randomly sample affine subspaces by generating a $D_\mathbf{u} \times D_\mathbf{x}$ matrix $\boldsymbol{P}$ and $D_\mathbf{u}$ dimensional vector $\boldsymbol{b}$ and then attempt to find any intersections with the manifold by iteratively solving $\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{P}\boldsymbol{v} + \boldsymbol{b})$ for $\boldsymbol{v}$, sampling a new subspace if no roots are found.

11. Related work

Several related approaches to applying gradient-based Monte Carlo inference methods within a ABC setting have been proposed. The *pseudo-marginal HMC* algorithm of [53] is particularly closely related to our approach, the authors proposing use of a HMC transition operator to jointly update the target variables $\mathbf{z}$ being inferred and auxiliary random input variables $\mathbf{u}_2$ used in computing the density estimate in pseudo-marginal inference problems. The authors discuss the specific relevance of their approach to an ABC setting, though formulate the method in terms of the wider context of the pseudo-marginal framework for MCMC inference using an unbiased density estimator [4].

Compared to the suggestion in Section 8 to directly apply a standard HMC transition operator to the ABC posterior density in the input space (35), the method proposed in [53] assumes extra structure in the models considered. Specifically the auxiliary random inputs $\mathbf{u}_2$ are assumed to marginally be independent standard normal variables, with this additional structure leveraged in a more efficient symplectic integrator compared to the standard leapfrog method that gives improved scaling to problems where the dimensionality of the auxiliary random inputs $\mathbf{u}_2$ is high.

Unlike the constrained HMC approach suggested here, the pseudo-marginal HMC method still requires use of a non-zero ϵ tolerance in ABC inference problems, and the complex 'narrow-ridge' geometry typical of the ABC posterior densities in the input space will often require use of a small integrator step-size as illustrated in Figure 3. This limits the gains in sampling efficiency from using a gradient-based approach and in the experiments of [53] it was found the proposed pseudo-marginal HMC method performed comparably to using non gradient-based elliptical slice sampling [66] updates to the target variables $\mathbf{z}$ and auxiliary random input variables $\mathbf{u}_2$ as proposed in [67].

Hamiltonian ABC [60], also proposes applying HMC to perform inference in simulator models. Rather than using reverse-mode AD to exactly calculate gradients of the generator function, Hamiltonian ABC uses a stochastic gradient estimator calculated using a *simultaneous perturbation stochastic approximation* (SPSA) scheme [88]. This is based on previous work considering methods for using a stochastic gradients within HMC [97, 21]. It has been suggested however that the use of stochastic gradients can compromise the favourable properties of Hamiltonian dynamics which enable coherent exploration of high dimensional state spaces [12]. The approach proposed in [60] also differs from that discussed in this paper in using a *synthetic likelihood* based ABC method [99] as opposed to the kernel-based formulation used here and described in Section 7. The synthetic likelihood method generates multiple simulated observed variables $\mathbf{x}$ for each evaluation of the approximated posterior density, using the empirical mean and standard deviation estimates of the set of simulated observations given the current unobserved variables $\mathbf{z}$ to fit a 'synthetic' normal model for the conditional density $\mathsf{p}_{\mathbf{x}|\mathbf{z}}$ (the likelihood). In [60] this is motivated by the observation that SPSA estimates of the gradients of the synthetic likelihood ABC posterior density are lower variance than the corresponding SPSA gradient estimator for a kernel-based ABC posterior density, albeit at the introduction of further bias compared to the gradients of the exact posterior density of interest.

The authors of Hamiltonian ABC also observe that representing the generative model as a deterministic function by fixing the random inputs to the generator is a useful method for improving exploration of the state space. This is achieved by including the state of the PRNG in the chain state however rather than directly updating the random inputs. As pointed out by the authors, this formulation puts minimal requirements on the model implementation with most numerical computing libraries having some facility to control the internal state of the PRNG being used, simplifying the application of the method with existing legacy code. In comparison the approach we propose will generally require some re-coding in a framework supporting reverse-mode AD and explicitly enumerating the random inputs used in the generator code.

Also related is *Optimization Monte Carlo* [61]. The authors propose using an optimiser to find parameters of a simulator model consistent with observed data (to within some tolerance ϵ) given fixed random inputs sampled independently. The optimisation is not volume-preserving and so the Jacobian of the map is approximated with finite differences to weight the samples. Our proposed constrained HMC method also uses an optimiser to find inputs consistent with the observations, however by using a volume-preserving dynamic we avoid having to re-weight samples. Our method also differs in treating all inputs to a generator equivalently; while the Optimization Monte Carlo authors similarly identify the simulator models as deterministic functions they distinguish between parameters and random inputs, optimising the first and independently sampling the latter. This can lead to random inputs being sampled for which no parameters can be found consistent with the observations (even with a within ϵ constraint). Although optimisation failure is also potentially an issue for our method, we found this occurred rarely in practice if an appropriate step size is chosen.

12. Numerical experiments

To evaluate the performance of the MCMC methods proposed in Sections 8 and 9 we performed inference experiments with three implicit generative models: a quantile distribution model for an IID dataset, a Lotka–Volterra predator-prey SDE simulator model, and a differentiable generator network model for human poses. In all experiments Theano [91], a Python computation graph framework providing reverse-mode AD, was used to specify the generator functions and compute derivatives. All experiments were run on a Intel Core i5-2400 quad-core CPU. Python code for the experiments is available at `https://git.io/dgm`.

12.1. Quantile distribution inference

As a first example we consider inferring the parameters of quantile distribution model for a IID dataset of univariate values. The generalised Tukey lambda distribution [80, 31] is a four parameter family of distributions defined via its quantile function. It has very flexible form which can describe distributions with a range of shapes, including close approximations of standard distributions such as the normal but also allowing asymmetric distributions with more general skewness and kurtosis. This flexibility has supported it use for statistical modelling in a diverse range of settings, including for example finance [23], climatology [69], control engineering [70] and material science [15].

Using the inverse CDF transform method it is simple to generate samples given a quantile function by mapping standard uniform samples through the quantile function. The quantile function does not have an analytic inverse however so the

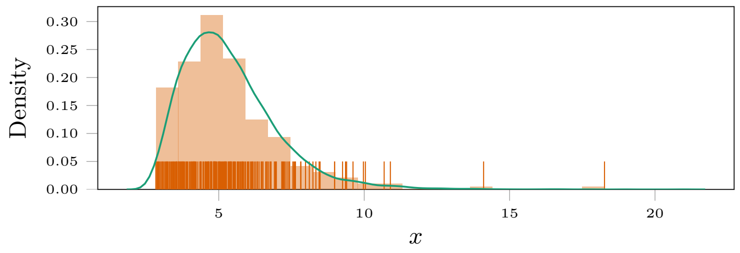


Fig 5: Histogram of generated generalised lambda distribution dataset used in experiments with $N = 250$ points generated using the quantile function parameterisation in (90) with parameters $z_1 = 5$, $z_2 = 1$, $z_3 = 0.4$ and $z_4 = -0.1$. The light orange region shows the histogram of the generated data with the orange ticks along the x axis indicating the actual data points. The green curve shows a kernel density estimate of the density of the distribution using a separate set of 10 000 independent samples.

CDF and corresponding density function do not have explicit forms. The use of ABC to perform Bayesian inference using quantile distributions was suggested by Allingham, King and Mengersen [2], with they employing a pseudo-marginal ABC MCMC approach based on order statistics of the observation in their experiments. McVinish [59] proposed a more efficient 'modified' ABC MCMC scheme specifically tailored to quantile distributions, with interval bisection used to identify an efficient proposal distribution for updates to the auxiliary uniform variables mapped through the quantile function.

We follow [59] in parameterising the quantile function of the generalised lambda distribution as

$$q_{\mathrm{GL}}(p \,|\, \boldsymbol{z}) = z_1 + \frac{1}{z_2}\left(\frac{p^{z_3} - 1}{z_3} + \frac{(1-p)^{z_4} - 1}{z_4}\right), \tag{90}$$

with z_1 a location parameter, z_2 a positive scale parameter and z_3 and z_4 shape parameters. In the experiments in [59], a synthetic dataset $\boldsymbol{x}$ of $N = 250$ independent samples is generated from a generalised lambda distribution using the quantile function (90) with parameters $z_1 = 5$, $z_2 = 1$, $z_3 = 0.4$ and $z_4 = -0.1$. The task considered in [59] is then inferring the posterior distribution on the parameters $\mathbf{z}$ given observed (synthetic) data $\boldsymbol{x}$.

A prior density on $\mathbf{z}$ is defined as

$$\mathsf{p}_{\mathbf{z}}(\boldsymbol{z}) = \lambda \exp(-\lambda z_2)\mathbb{I}_{[0,\infty)}(z_2)\,\mathcal{N}\big(z_1 \,|\, 0, \sigma^2\big)\,\mathcal{N}\big(z_3 \,|\, 0, \sigma^2\big)\,\mathcal{N}\big(z_4 \,|\, 0, \sigma^2\big) \tag{91}$$

corresponding to independent normal priors on each of the location and shape parameters and an exponential prior on the location parameter. In the experiments in [59] the prior hyperparameters are chosen as $\sigma = 10$ and $\lambda = {}^1/_{10}$. In [59] the proposed modified ABC MCMC algorithm is compared to a standard ABC MCMC approach and a population Monte Carlo ABC method [9]. The proposed modified ABC MCMC algorithm was found to significantly outperform the other two approaches, and so we focus on comparing to this method.

We compare a Cython [11] implementation of the modified ABC MCMC algorithm to two of the algorithms discussed in previous sections: an ABC approach with a Gaussian kernel k_ϵ, running HMC in the input space to a differentiable generator for the model as discussed in Section 8; the constrained HMC method described in Algorithm 1, conditioning the output of a differentiable generator to be exactly equal to observed data. As in [59] we use $N = 250$ generated data points using the parameters $\boldsymbol{z} = [5,\ 1,\ 0.4,\ -0.1]^\mathsf{T}$ with the generated data used in our experiments shown in Figure 5.

We formulate the quantile distribution model as a directed differentiable generative model as follows. We define a generator $\boldsymbol{g}_\mathbf{z}$ for the parameters $\mathbf{z}$ by

```
function $\boldsymbol{g}_\mathbf{z}(\mathbf{u}_1)$
    $\mathsf{z}_1 \leftarrow \sigma \mathsf{u}_{1,1}$
    $\mathsf{z}_3 \leftarrow \sigma \mathsf{u}_{1,2}$
    $\mathsf{z}_4 \leftarrow \sigma \mathsf{u}_{1,3}$
    $\mathsf{z}_2 \leftarrow \frac{1}{\lambda} \log\left(1 + \exp\left(\frac{\pi}{\sqrt{3}} \mathsf{u}_{1,4}\right)\right)$
    return $[\mathsf{z}_1,\ \mathsf{z}_2,\ \mathsf{z}_3,\ \mathsf{z}_4]^\mathsf{T}$
```

Here the input variables $\mathsf{u}_{1,1}$, $\mathsf{u}_{1,2}$ and $\mathsf{u}_{1,3}$ are assumed to have independent standard normal distributions $\mathcal{N}(0, 1)$. The input variable $\mathsf{u}_{1,4}$, which maps to the scale parameter z_2, has a zero-mean and unit-variance logistic distribution

$$\mathsf{p}_{\mathsf{u}_{1,4}}(u_{1,4}) = \frac{\pi}{4\sqrt{3}} \cosh\left(\frac{\pi u_{1,4}}{2\sqrt{3}}\right)^{-2}. \tag{92}$$

Given a vector of inputs $\mathbf{u}_1$ with these distributions, $\boldsymbol{g}_\mathbf{z}$ outputs a parameter vector $\mathbf{z}$ distributed according to the prior density (91).

The generator for the observed variables $\mathbf{x}$ given the parameters $\mathbf{z}$ and additional random inputs $\mathbf{u}_2$ is then specified by

```
function $\boldsymbol{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z},\ \mathbf{u}_2)$
    for $n \in \{1 \dots N\}$
        $\mathsf{p}_n \leftarrow \left(1 + \exp\left(-\frac{\pi}{\sqrt{3}} \mathsf{u}_{2,n}\right)\right)^{-1}$
        $\mathsf{x}_n \leftarrow q_{\mathrm{GL}}(\mathsf{p}_n \,|\, \mathbf{z})$
    return $[\mathsf{x}_1,\ \mathsf{x}_2,\ \dots\ ,\ \mathsf{x}_N]^\mathsf{T}$
```

Here the input variables $\mathbf{u}_2$ have independent zero-mean and unit-variance logistic distributions with density as in (92). These are transformed to standard uniform variables via a logistic sigmoid function, with these uniform variables then mapped through the quantile function to generate values from the generalised lambda quantile distribution given the provided parameter values $\mathbf{z}$.

As the generated observed variables $\mathbf{x}$ are conditionally independent given the parameters $\mathbf{z}$, the Jacobian of the overall generator $\boldsymbol{g}_\mathbf{x}(\mathbf{u}) = \boldsymbol{g}_{\mathbf{x}|\mathbf{z}}(\boldsymbol{g}_\mathbf{z}(\mathbf{u}_1), \mathbf{u}_2)$ has the block structure discussed in Section 10.2, with a dense matrix block corresponding to the partial derivatives of the generated $\mathbf{x}$ with respect to the inputs $\mathbf{u}_1$ mapping to parameters, and a diagonal matrix block corresponding to the partial derivatives of the generated $\mathbf{x}$ with respect to the inputs $\mathbf{u}_2$. As described in Section 10.2 this allows efficient computation of the Jacobian product Cholesky factor in the constrained HMC algorithm.

The modified ABC MCMC method uses a proposal kernel to generate updates to the parameters $\mathbf{z}$ which are then accepted or rejected in a Metropolis–Hastings step. We follow the experiments of [59] and use a uniform random-walk proposal density $\mathcal{U}(z' \mid z - s, z + s)$ where s is a step-size parameter, which was tuned to give an average accept rate of approximately 0.25 in pilot runs, with $s = 0.075$ used in our experiments. The interval bisection method used to construct the proposed updates to the auxiliary uniform variables has a free parameter m defining the number of bisection iterations; following the experiments of [59] we use $m = 16$. The ABC kernel used in the modified ABC MCMC algorithm is uniform across a cubic region specified by an infinity norm tolerance

$$k_\epsilon(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{\epsilon^D} \mathbb{I}_{[0,\epsilon]}(\|\boldsymbol{y} - \boldsymbol{x}\|_\infty) = \frac{1}{\epsilon^D} \prod_{d=1}^{D} \mathbb{I}_{[0,\epsilon]}(|y_i - x_i|) \tag{93}$$

with the product decomposition of this kernel being central to the proposed efficient update to the auxiliary variables in [59]. We follow [59] in using a tolerance of $\epsilon = 0.1$ in the experiments.

In pilot runs with the modified ABC MCMC algorithm, we found that when initialising chains from the normal–exponential prior (91) with hyperparameters $\sigma = 10$ and $\lambda = 1/10$, that some chains failed to converge, remaining at the initial state for long series of rejections even with very small step sizes and in some cases failing completely due to numerical overflow. By generating additional synthetic datasets using parameters sampled from a prior with $\sigma = 10$ and $\lambda = 1/10$ it was found that this prior choice put significant mass on settings leading to very extreme sampled values and in some cases producing values beyond the maximum range of double precision floating point. As such extreme variation in the target distribution seem implausible a-priori, we use a more informative choice of prior in our experiments with $\sigma = \lambda = 1$, with this choice giving a more plausible range of variation for simulated datasets. We found the regularisation provided by this choice to significantly improve the stability of all the methods tested while having a negligible impact on the inferred posteriors.

For all the approaches tested, the chains for the parameter values $\mathbf{z}$, or correspondingly the input variables $\mathbf{u}_1$ in the case of the methods parameterised in the generator input space, were initialised from values sampled from the prior, with the same 5 independently sampled initial states used for all chains. For the constrained HMC chains, the initial states of the remaining $\mathbf{u}_2$ input variables were set by using an optimisation routine to solve for values of these variables giving generated observed outputs within an maximum elementwise distance of 10^{-8} of the observed data values. These same optimised $\mathbf{u}_2$ initial states were also used for the unconstrained HMC chains. This optimisation was a negligible overhead (less than one second) and so not included in the run time estimates.

For the constrained HMC chains we used an integrator step size $\delta t = 0.6$ and $N_g = 4$ inner geodesic steps per overall time step. These values were chosen based on pilot runs to give an average accept rate in the range 0.6 to 0.9 [13] and to minimise the occurrence of any rejections due to non-reversible geodesic steps or convergence failure in the iterative solver. The number of integrator steps N_s for each constrained HMC update was uniformly sampled from $[5, 10]$.

For the unconstrained HMC chains using a Gaussian kernel ABC target density in the generator input space (35), we ran sets of chains for $\epsilon = 0.25$ and $\epsilon = 0.05$ (due to the different kernel from that used in the modified ABC MCMC method the tolerance values cannot be directly compared between the two methods). For $\epsilon = 0.25$ we used a integrator step size $\delta t = 2.5 \times 10^{-3}$ and for $\epsilon = 0.05$, $\delta t = 5 \times 10^{-4}$, again chosen based on trying to achieve a target accept rate in $[0.6, 0.9]$. We found however that the sensitivity of the stability of the updates to δt made it challenging to meet this requirement, with values for δt giving reasonable accept rates below 0.9 for some chains leading to other having very low accept rates, and so the chosen δt values gave accept rates closer to 0.95 in most cases. We sampled the number of leapfrog steps L for each update uniformly from $[20, 40]$ for the $\epsilon = 0.25$ chains and $[40, 80]$ for the $\epsilon = 0.05$ chains; these values were chosen relatively arbitrarily and performance could likely be improved by tuning these values or using the adaptive NUTS algorithm [43].

For all chains we ran initial warm-up phases which were excluded from the later estimates to allow for convergence to the posterior typical set and reduce the estimator bias. The number of warm-up iterations for each chain was hand-tuned based on visualising traces of the chains and setting the number of warm-up iterations to remove any obvious initial transient behaviour in the chains. For the constrained and unconstrained HMC chains we found it helped stability to use a smaller integrator step size and fewer integrator steps in the warm-up phase. The initial states have atypically high potential energy and so the momenta quickly grow large in the simulated dynamics in the early chain iterations, in some cases leading to stability issues with the step size. Using a smaller initial step size and smaller number of integration steps and so more frequent momentum resampling operations where the momenta are restored to values with more reasonable magnitudes helps to alleviate this issue.

We used $\delta t = 0.05$ and $N_s = 2$ in 200 warm up iterations for each constrained HMC chain; $\delta t = 10^{-3}$, $L = 10$ for 1000 warm up iterations for each $\epsilon = 0.25$ HMC chain; and $\delta t = 2.5 \times 10^{-4}$ and $L = 20$ for 5000 warm up iterations for each $\epsilon = 0.05$ HMC chain. For the modified ABC MCMC chains we used 5000 warm up iterations (using the same $s = 0.075$ step size as in the main runs). We ran the main sampling phase for 1000 iterations for the constrained HMC chains, 30 000 iterations for the $\epsilon = 0.25$ HMC chains, 15 000 iterations for the $\epsilon = 0.05$ HMC chains and 100 000 iterations for the modified ABC MCMC chains; in all cases this leading to chains taking roughly five minutes to run each in our implementations (we recorded exact run times for each chain *including the warm-up iterations* to use in normalising efficiency estimates). Although performance of the different methods is somewhat implementation dependent, in all cases the use of efficient compiled updates for the main computational bottlenecks (either via Cython for the modified ABC MCMC implementation or Theano for the two HMC algorithms) meant that the interpreter overhead from using Python was at least minimal, with all chains fully utilising a single CPU core when running.

The estimated parameter posterior distributions using the samples from all of the chains run for each of the approaches tested are shown in Figure 6. We can see that the marginal posteriors generally concentrate relatively tightly around the values of the parameters used to generate the data (shown by dashed lines), with the constrained HMC and modified ABC MCMC algorithms showing tighter estimated distributions than the Gaussian kernel HMC chains, with the $\epsilon = 0.25$ case being the most diffuse as expected. The estimated posterior marginals from the $\epsilon = 0.05$ HMC chains show spurious appearing irregularities not evident in the results from the other chains, which is indicative of convergence issues in the chains. The estimated *potential scale reduction factor* (PSRF) statistic [35] for the $\epsilon = 0.05$ HMC chains was $\hat{R} = 1.21$ which also suggests convergence problems ($\hat{R}$ values close one are indicative of chains having converged); for both the constrained HMC and modified ABC MCMC chains $\hat{R} = 1.00$ while for the $\epsilon = 0.25$ HMC chains $\hat{R} = 1.03$.

Figure 7 shows estimates of the *effective sample size* (ESS) for the posterior means of each model parameter (calculated using the `CODA` package in `R` [75]) normalised by the chain run time in seconds and grouped by chains of each the four approaches tested. The coloured bars show the mean values across the five independent chains for each method and the black ticks ± 1 standard error of mean. Although as noted above the real-time performance of the methods is somewhat implementation dependent, it seems that the proposed constrained HMC method performs broadly about as well as the modified ABC MCMC approach here in terms of sampling efficiency, while the methods performing HMC in the generator input space using a Gaussian ABC kernel are significantly less efficient.

That the proposed constrained HMC method works about as well as an algorithm custom tuned to this particular problem is encouraging. Further given the generally improved relative performance of HMC methods compared to random-walk Metropolis based methods as the dimension of the target distribution grows, it seems plausible that the comparison would be even more positive towards the constrained HMC method in models with larger numbers of parameters. It is interesting to note that both approaches use iterative optimisation methods within the inner loop of the algorithms: in the modified ABC MCMC method interval bisection is used to find a relatively tight bounding box on the allowable values of the auxiliary uniform variables used to generate the simulated data given the current parameter values, while in our constrained HMC approach a quasi-Newton iteration is used to project on to the constraint manifold in the input space corresponding to the fibre of observed data under the generator function $\boldsymbol{g}_{\mathbf{x}}$. In both cases this helps overcome the curse of dimensionality effects typically experienced when conditioning on high-dimensional observed data in ABC inference problems.

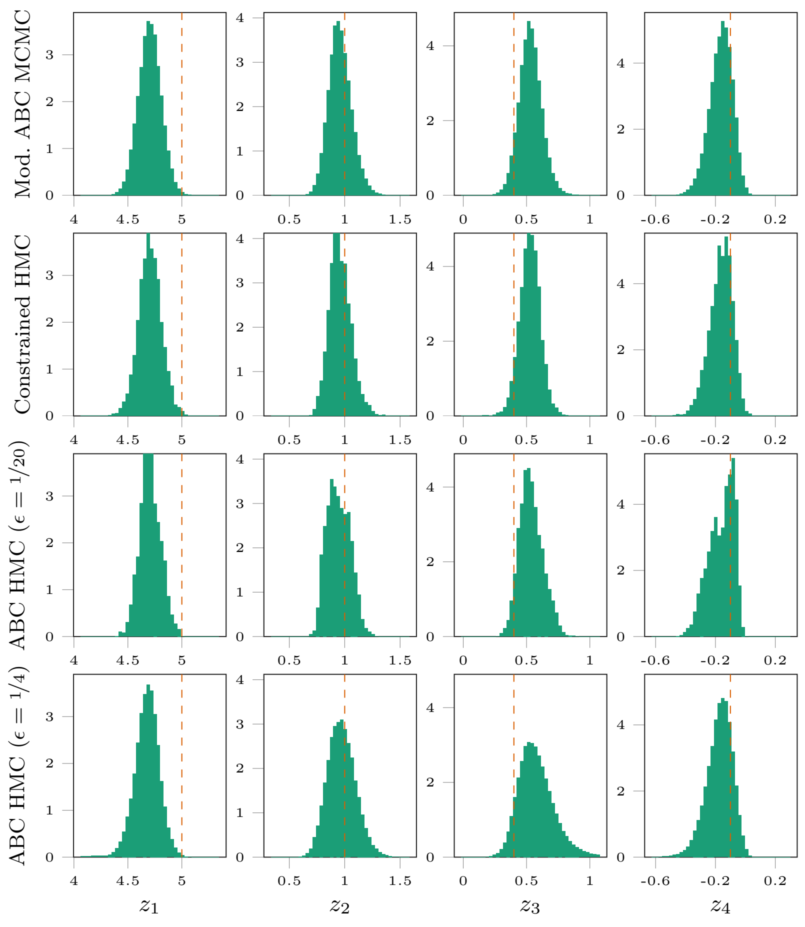


Fig 6: Estimated marginal posterior distributions of generalised lambda model parameters. Each row corresponds to samples from five independent chains for MCMC method labelled to left of plot, while each column corresponds to one of the four distribution parameters, labelled to bottom of plot. The orange dashed line on each axis indicates the value of the parameter used to generate the data.

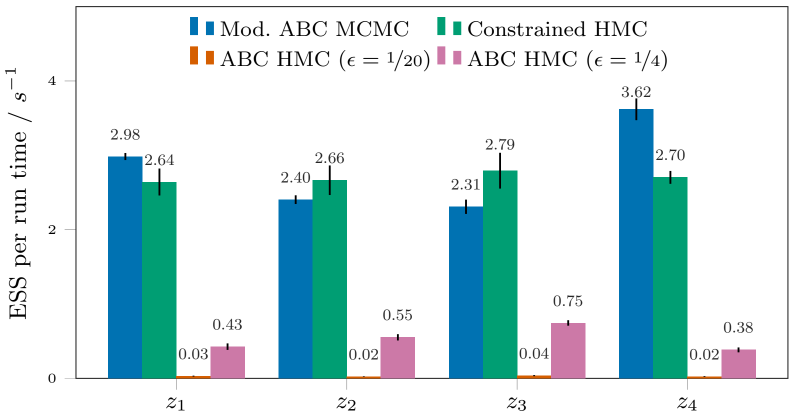


Fig 7: Estimated ESS for posterior means of each generalised lambda model parameter normalised by chain run time. Each coloured set of bars corresponds to mean estimated ESS per run time across five independent chains for the method indicated in the legend. The ticks on the bars show ±1 standard error of mean.

12.2. Lotka–Volterra parameter inference

As a second test case we considered inferring the parameters of a SDE variant of the Lotka–Volterra predator–prey model, a common example problem in the ABC literature e.g. [61, 71]. In particular given observed predator–prey population data we consider inferring the parameters of the following pair of SDEs

$$\mathrm{d}r = (z_1 r - z_2 r f)\mathrm{d}t + \mathrm{d}n_r, \quad \mathrm{d}f = (z_4 r f - z_3 f)\mathrm{d}t + \mathrm{d}n_f, \tag{94}$$

where r represents the prey population, f the predator population, $\{z_i\}_{i=1}^4$ the system parameters and n_r and n_f are zero-mean white noise processes.

A simulator for these SDEs can be formed by using an Euler–Maruyama [47] integration scheme to generate simulated realisations of the stochastic process at discrete time points. If the white-noise processes n_r and n_f have variances σ_r^2 and σ_f^2 respectively, then an Euler–Maruyama discretisation of S time points of the SDE 94 with an integrator time step δt and initial system state (r_0, f_0) can be generated given a vector of standard normal random variates $\mathbf{u}_2$ as defined in the following pseudo-code.

```
function $\boldsymbol{g}_{\mathsf{x}|\mathsf{z}}(\mathbf{z}, \mathbf{u}_2)$
    $\mathsf{r}_0 \leftarrow r_0$
    $\mathsf{f}_0 \leftarrow f_0$
    for $s \in \{1 \dots S\}$
        $\mathsf{r}_s \leftarrow \mathsf{r}_{s-1} + \delta t(\mathsf{z}_1\mathsf{r}_{s-1} - \mathsf{z}_2\mathsf{r}_{s-1}\mathsf{f}_{s-1}) + \sqrt{\delta t}\sigma_r \mathsf{u}_{2,2s}$
        $\mathsf{f}_s \leftarrow \mathsf{f}_{s-1} + \delta t(\mathsf{z}_4\mathsf{r}_{s-1}\mathsf{f}_{s-1} - \mathsf{z}_3\mathsf{f}_{s-1}) + \sqrt{\delta t}\sigma_f \mathsf{u}_{2,2s+1}$
    $\mathbf{x} \leftarrow [\mathsf{r}_1, \mathsf{f}_1, \dots \mathsf{r}_S, \mathsf{f}_S]$
    return $\mathbf{x}$
```

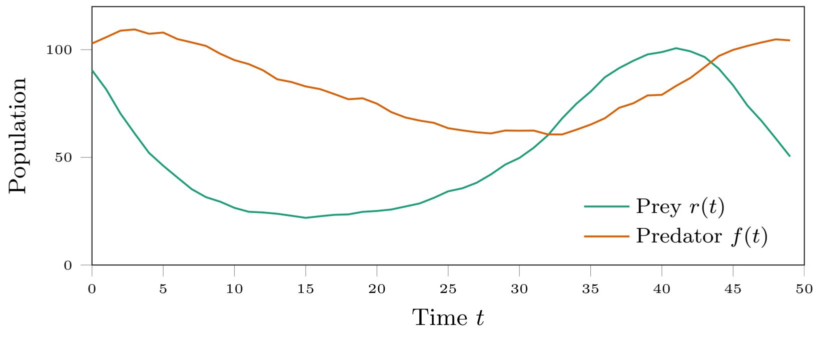


Fig 8: Traces of generated realisations of Lotka–Volterra SDE model (94) used as the observations in the experiments.

As suggested by the notation we can consider this Euler–Maruyama integration as defining the observed generator $\boldsymbol{g}_{\mathbf{x}|\mathbf{z}}$ of a directed generative model, mapping from the unobserved parameter variables $\mathbf{z}$ and an auxiliary vector of standard normal random inputs $\mathbf{u}_2$ to a vector formed by the concatenation of the simulated state sequences. This mapping is differentiable with respect to $\mathbf{z}$ and $\mathbf{u}_2$ and so can be used to define a differentiable generative model. The generator in this case has the Markovian structure discussed in Section 10.2 allowing efficient computation of the Cholesky factor of the Jacobian matrix product $\mathbf{J}_{\boldsymbol{g}_\mathbf{x}}\mathbf{J}_{\boldsymbol{g}_\mathbf{x}}^\mathsf{T}$.

In the Lotka–Volterra SDE parameterisation used in (94), all of the parameter variables $\mathbf{z}$ are required to be positive. A simple suitable choice of a prior distribution on the parameters is therefore a log-normal distribution

$$\mathsf{p}_\mathbf{z}(\boldsymbol{z}) = \prod_{i=1}^{4} \mathrm{LogNormal}(z_i \,|\, m_i, s_i). \tag{95}$$

A generator function for the parameters can then be defined by

```
function $\boldsymbol{g}_\mathbf{z}(\mathbf{u}_1)$
    $\mathbf{z} \leftarrow \exp(\boldsymbol{s} \odot \mathbf{u}_1 + \boldsymbol{m})$
    return $\mathbf{z}$
```

where $\mathbf{u}_1$ is an input vector of standard normal variables.

For the experiments we generated a synthetic observed data set $\boldsymbol{x}$ of $S = 50$ simulated time points of predator–prey population state sequences using the Euler–Maruyama generator function defined above with an integrator time step $\delta t = 1$, white noise process standard deviations $\sigma_f = \sigma_r = 1$, initial conditions $r_0 = f_0 = 100$ and model parameter values $z_1 = 0.4$, $z_2 = 0.005$, $z_3 = 0.05$ and $z_4 = 0.001$ (chosen to give stable, oscillatory dynamics). The generated sequences used in the experiments are shown in Figure 8. We then considered the problem of inferring the 'unknown' model parameters $\mathbf{z}$ (with the initial states, integrator time step and noise variances assumed to be known) given the observed data $\boldsymbol{x}$.

For the log-normal prior, we used location hyperparameters $m_i = -2 \;\; \forall i \in \{1 \dots 4\}$ and scale hyperparameters to $s_i = 1 \;\; \forall i \in \{1 \dots 4\}$. As in the generalised lambda distribution experiments in the previous section this choice of a relatively informative prior was motivated by trying to minimise the prior probability mass put on parameters corresponding to implausible generated sequences, with in particular in this case the Lotka–Volterra dynamics being unstable for many parameter settings, with an exponential blow-up in the prey population if the predator population 'dies off'. Biasing the prior towards smaller values was found to favour more plausible appearing sequences with stable dynamics.

We first tested several standard ABC approaches to perform inference, conditioning on the full observed data sequences i.e. without use of summary statistics. ABC rejection using a uniform ball kernel failed catastrophically, with no acceptances in 10^6 samples even with a very large tolerance $\epsilon = 1000$. A standard (pseudo-marginal) ABC MCMC method with a Gaussian random-walk proposal distribution also performed very poorly with the dynamic having zero acceptances over multiple runs of 10^5 updates for $\epsilon = 100$ and getting stuck at points in parameter space over thousands of iterations for larger $\epsilon = 1000$, even with very small proposal steps. Similar issues were also observed when attempting to run pseudo-marginal ABC MCMC chains using a Gaussian kernel. This poor performance is not unexpected, but highlights the challenges of working with high-dimensional observations in standard ABC approaches.

We next attempted to reduce the dimensionality of the observed data and generated observations by using a set of summary statistics. We used the nine summary statistics employed in a similar Lotka–Volterra inference problem in [71] - the means and log variances of the two sequences, lag-one and lag-two autocorrelation coefficients and cross-correlation coefficient of the sequences. Even when reducing to this much lower dimensional space, ABC reject continued to give zero accepts unless a non-informatively large tolerance was used. Using this set of summary statistics we were however able to successfully run ABC MCMC chains which appeared to converge (PSRF statistic of $\hat{R} = 1.02$ across five independent chains of 500 000 samples) when using a uniform ball kernel with $\epsilon = 2.5$ on the nine-dimensional summary statistics.

A histogram of the resulting estimated marginal posteriors on the model parameters from the last 250 000 samples of five 500 000 sample chains is shown in Figure 9 with the orange dashed lines indicating the values of the parameters used to generate the data. The estimated marginal posteriors of z_1 and z_2 are concentrated around the values of the parameters used to generate the data, however this is not the case for the estimated marginal posteriors of the z_3 and z_4 parameters. Although there is nothing to guarantee that the true posterior is centred at the parameters used to generate the data[4] the degree of discrepancy between where the posterior mass is located and the parameters used to generate the data is potentially concerning. We will see in later results that the posterior distributions conditioned on the summary statistics of generated observations

[4]In general we would only expect the parameters to be a plausible sample under the posterior; if the parameters were sampled from the prior then they would represent an exact sample from the posterior given the generated data.

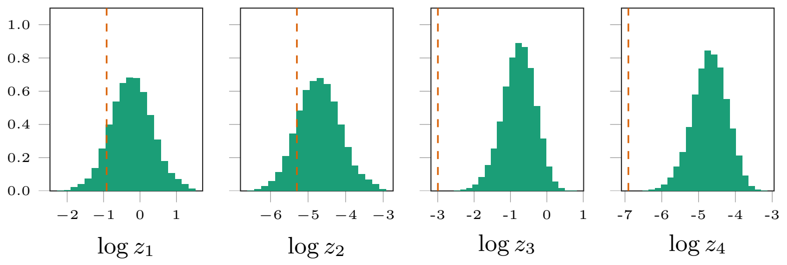


Fig 9: Estimated marginal posterior distributions of Lotka–Volterra model parameters using random-walk Metropolis pseudo-marginal ABC MCMC chains with nine-dimensional summary statistics and a uniform ball kernel with $\epsilon = 2.5$. Each histogram corresponds to last 250 000 samples from five independent chains of 500 000 samples. The orange dashed line on each axis indicates the value of the parameter used to generate the data.

exactly matching the data summary statistics appears to be concentrated around the data generating parameters as does the ABC posterior when conditioning on all of the data using a uniform ball kernel.

It therefore seems here that it may be the combined use of summary statistics and an ABC kernel which is causing a potentially non-representative posterior distribution (in the sense of being representative of the true posterior we are interested in, the estimated posterior may in fact be reflective of the true location of the mass of the distribution conditioned on the summary statistics of the data being within a distance of $\epsilon = 2.5$ of the data). These issues highlight the challenges in assessing the impact of the choice of summary statistics and tolerance on the inferred posterior in ABC methods.

As the generative model here is differentiable we are able to apply our proposed constrained HMC method in the input space of the generator to construct chains directly targeting the posterior distribution of interest, constraining the output of the generator to be equal to the observed data (to within a 10^{-8} infinity norm distance used as the convergence tolerance in the Newton iteration). We ran ten independent constrained HMC chains of 1000 samples, using an integrator step size $\delta t = 0.25$, the number of integrator time steps per proposed update N_s uniformly sampled from $[4, 8]$ on each iteration, $N_i = 3$ inner geodesic steps per update and a Newton convergence tolerance of $\epsilon = 10^{-8}$. As in the experiments in the previous section, the initial states for the chains were computed by sampling random values of the input variables $\mathbf{u}_1$ corresponding to the model parameters and then solving for the values of the remaining random inputs $\mathbf{u}_2$ giving a generated output equal to the observed data using the `fsolve` optimisation routine in `SciPy` [44]. Based on visualisation of traces, the first ten iterations of each chain were removed as 'warm-up' iterations.

To test how informative the nine summary statistics used in the ABC MCMC runs were, we also ran constrained HMC chains in the posterior distribution formed by constraining the summary statistics of the generated observed variables

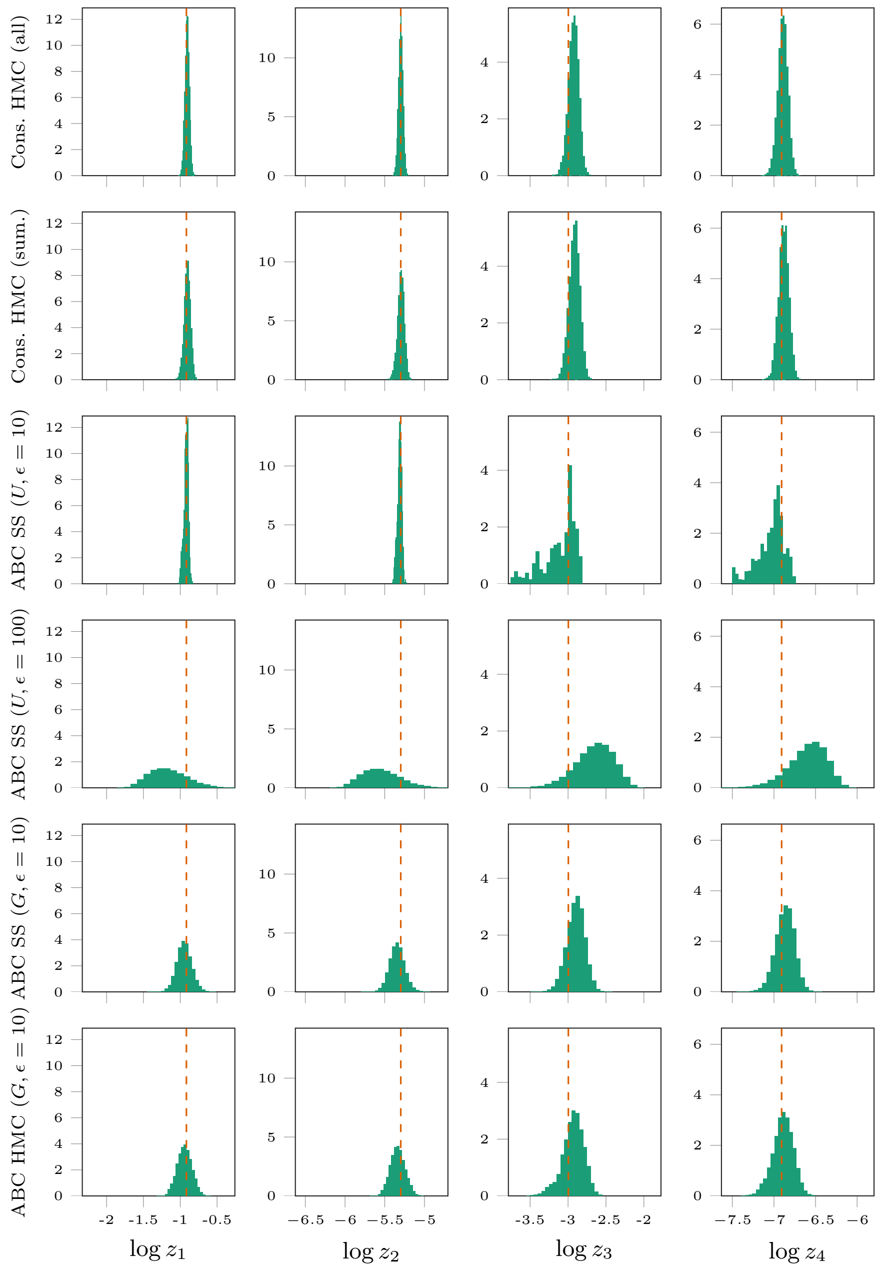


Fig 10: Estimated marginal posterior distributions of Lotka–Volterra model parameters. Each row corresponds to samples from ten independent chains for MCMC method labelled to left of plot, while each column corresponds to one of the four distribution parameters, labelled to bottom of plot. The orange dashed line on each axis indicates the value of the parameter used to generate the data.

$\mathbf{x}$ to be equal to the summary statistics of the observed data $\boldsymbol{x}$. As the summary statistics are all differentiable functions of the generated observations here, we can simply defined an augmented generator which outputs summaries rather than full observations and use this in the constrained HMC update in Algorithm 1. We used the same initial input states and algorithm settings for these chains as for the full data case.

The resulting estimates of the marginal posteriors on the model parameters formed using the samples from the constrained HMC chains run in these two cases are shown in the top two rows in Figure 10, with the top row (labelled 'Cons. HMC (all)') corresponding to the posterior distribution conditioned on all of the data, and the second row (labelled 'Cons. HMC (sum.)') corresponding to the posterior distribution conditioned on just the summary statistics of the data. It can be seen that in both cases the estimated posteriors are concentrated around the parameter values used to generate the data (indicated by orange dashed lines), unlike the previous results in Figure 9. Interestingly there seems to be minimal loss of information about the parameters when conditioning on the summary statistics rather than the full data here, with the estimated marginal posteriors for the summary statistics chains only slightly more diffuse than the corresponding marginal posterior estimates for the full data chains. In both cases the estimated PSRF statistics across the 10 chains were $\hat{R} = 1.00$.

We also tested the proposed approach of performing ABC inference by running chains in the input space to the generator, as discussed in Section 8. Encouragingly we found that by performing MCMC updates to the random inputs to the generator, we are able to tractably perform ABC inference when conditioning on the full observed data, even when using relatively simple non-gradient based MCMC methods. In particular based on the pseudo-marginal slice sampling of [67], we tried using alternating elliptical slice sampling updates of the random inputs $\mathbf{u}_1$ used to generate the parameters, i.e. $\mathbf{z} = \boldsymbol{g}_{\mathbf{z}}(\mathbf{u}_1)$, and remaining random inputs $\mathbf{u}_2$ used to generate the simulated observations given parameters, i.e. $\mathbf{x} = \boldsymbol{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z}, \mathbf{u}_2)$. Using this method, which had zero free settings to tune, we were able to construct chains which appeared to converge to a reasonable approximate posterior both when using a uniform ball kernel with $\epsilon = 100$ and $\epsilon = 10$ and when using a Gaussian kernel with $\epsilon = 10$. We also ran HMC chains with a $\epsilon = 10$ Gaussian kernel, using an integrator step size $\delta t = 2.5 \times 10^{-3}$ and a number of leapfrog steps per proposed update L sampled uniformly from $[10, 20]$. For the slice sampling approaches we ran 10 independent chains of 60 000 samples for each kernel and tolerance combination, discarding the first 30 000 samples as warm-up iterations, and for the HMC case we ran 10 independent chains of 10 000 samples, discarding the first 5000 samples as warm-up iterations.

Estimates of the marginal posterior distributions on the parameters for these chains are shown in the last four rows of Figure 10, with the label 'ABC SS' indicating elliptical slice sampling chains and 'ABC HMC' the HMC chains and a U in parenthesis in the label indicating use of uniform ball kernel and a G in parenthesis in the label G a Gaussian kernel, with in both cases the corresponding ϵ tolerance also given in the parentheses. It is immediately evident that the estimated ABC marginal posteriors here are more diffuse than for the constrained

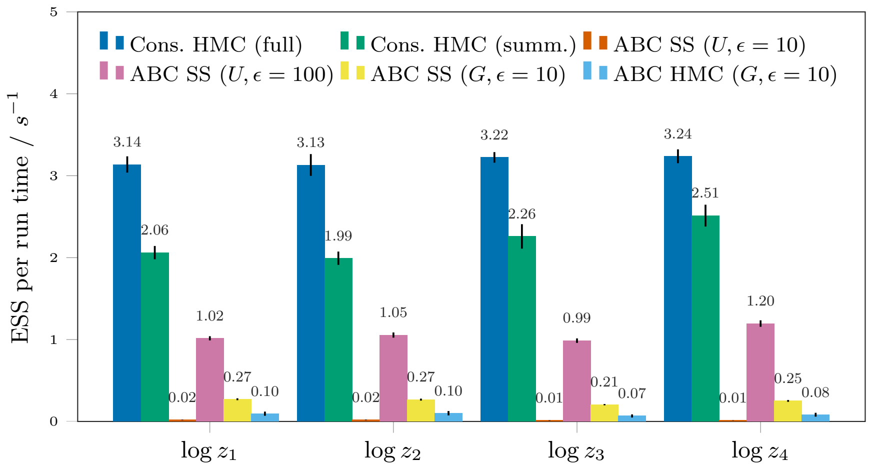


Fig 11: Estimated ESS for posterior means of each Lotka–Volterra model parameters normalised by chain run time. Each coloured set of bars corresponds to mean estimated ESS per run time across ten independent chains for the method indicated in the legend (key: SS - slice sampling, summ. - summary statistics, G - Gaussian kernel, U - uniform ball kernel). The ticks on the bars show ± 1 standard error of mean.

HMC chains, particularly for the slice sampling chains using a uniform ball kernel with $\epsilon = 100$ (fourth row), though the estimated posteriors are still significantly more concentrated than for the summary statistic based ABC posterior shown in Figure 9 (note the difference in the horizontal scales compared to Figure 10). Unlike the summary-statistic based pseudo-marginal ABC MCMC estimates in Figure 9 though, all of the marginal posterior estimates for these ABC methods using the full set of observations are concentrated around the region of the parameters used to generate the data, and seem to broadly consistent, albeit more diffuse, with the constrained HMC estimates of the true posterior.

Between the four different approaches, the estimates of the ABC posterior using a uniform ball kernel with $\epsilon = 10$ (row three) seem to be the closest to the constrained HMC estimates of the true posterior (first row), with in particular the marginal estimates for z_1 and z_2 visually very similar. For the z_3 and z_4 marginal posterior estimates in the uniform ball kernel $\epsilon = 10$ case there are spurious appearing peaks however suggesting possible convergence issues and this is backed up by a PSRF statistic of $\hat{R} = 1.89$ across the 10 chains. Although not as visible in the marginal posterior estimates, the Gaussian kernel HMC chains also suffered convergence issues, with a PSRF statistic of $\hat{R} = 1.33$ across the 10 chains. The $\epsilon = 10$ Gaussian kernel and $\epsilon = 100$ uniform ball kernel slice sampling chains both had estimated PSRF statistics of $\hat{R} = 1.01$.

We also measured the sampling efficiency of the chains generated using the different approaches by computing ESS estimates for each parameter and normalising by the total chain run-time (including warm-up iterations), with the results shown in Figure 11. As there are quite significant differences between the distributions being targeted by the chains of most of the methods, as well as potential differences in the relative efficiencies of the implementations, the run time normalised ESS estimates can only give a rough indication of relative performance. Subject to those provisos however, the results suggest the constrained HMC methods are potentially significantly more efficient than the alternative approaches here, despite also giving in some sense the 'best' estimates of the true posterior. Although not as efficient as the constrained HMC methods here, the slice sampling approaches are an attractively simple and 'black-box' method, with no free parameters to tune and no requirement to propagate derivatives through the generative models of interest.

Somewhat counter-intuitively perhaps, the constrained HMC chains conditioned on the full data showed higher sampling efficiency than those conditioned on the reduced dimension summary statistics. Given the earlier statement that a dominant cost in the constrained HMC algorithm is the computation of the Cholesky decomposition of the generator Jacobian product, $\operatorname{chol} \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}} \mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}^{\mathsf{T}}$, which in general will scale cubically with the dimension of the generator output, it might be expected that projecting the generator output to a lower-dimensional space would lead to lower-cost updates and so improved efficiency. While this may be the case in some settings, here there is the additional factor that the generator Jacobian for the full observations case has the triangular block structure discussed in Section 10.2, which allows a more efficient computation of the Cholesky factor using low-rank updates. This structure is lost when projecting down to lower dimensions, and so a standard cubic-cost Cholesky factorisation routine needs to be used. In models without such structure however and with large numbers of observed variables, projecting down to lower-dimensional summaries could be an important method for improving the scalability of the constrained HMC approach, and as shown in the example here, in some cases may entail minimal loss of information about the unobserved variables being inferred.

12.3. Human pose and camera model inference

For the final experiment we consider inference in an differentiable generative model for human poses. In particular we consider the task of inferring a three-dimensional human pose given binocular two-dimensional projections of joint positions, using a learnt prior model of poses from motion capture and anthropometric data, and a simple projective pin-hole camera model.

We parameterised the poses used a 19 joint skeleton model, with degrees of freedom specifying the angular configurations of each joint and the lengths of the bones between joints. In total the model has 47 local joint angles $\mathbf{z}_a$ (with some joints, for example those corresponding to knees and elbows, not having a full three degrees of freedom). The prior over the joint angles was specified by a Gaussian *variational autoencoder* (VAE) model [46, 82] trained on the *PosePrior* motion caption dataset [1]. The circular topology of the angular data is poorly

Algorithm 2 Human pose model generator functions.

Input:

$\{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=0}^{L}$: parameters of pose angle differentiable network;
$\boldsymbol{\mu}_b$, $\boldsymbol{\Sigma}$: mean and covariance of skeleton bone lengths;
$\boldsymbol{\mu}_{c,:2}$, $\boldsymbol{\sigma}_{c,:2}$: camera x, y coordinates normal prior parameters;
$\mu_{c,2}$, $\sigma_{c,2}$: camera z coordinate log-normal prior parameters;
JointPositions : maps pose angles and bone lengths to joint positions;
CameraMatrices : maps camera parameters to a pair of camera matrices;
Project : uses camera matrix to map world to image coordinates;
Partition : partitions a vector in a specified number of equal length parts;
Flatten : flattens a multidimensional array to a vector.

```
function $\boldsymbol{g}_{\mathbf{z}}([\mathbf{u}_h;\, \mathbf{u}_1;\, \mathbf{u}_2;\, \mathbf{u}_b;\, \mathbf{u}_c])$
    $\mathbf{h}_L \leftarrow$ DifferentiableNetwork$(\mathbf{u}_h)$
    $\mathbf{m}_1, \mathbf{k}_1, \mathbf{m}_2, \mathbf{k}_2 \leftarrow$ Partition$(\mathbf{h}_L, 4)$
    $\mathbf{r}_1 \leftarrow \exp(\mathbf{k}_1) \odot \mathbf{u}_1 + \mathbf{m}_1$
    $\mathbf{r}_2 \leftarrow \exp(\mathbf{k}_2) \odot \mathbf{u}_2 + \mathbf{m}_2$
    $\mathbf{z}_a \leftarrow \mathrm{atan2}(\mathbf{r}_2, \mathbf{r}_1)$
    $\mathbf{z}_b \leftarrow \exp(\boldsymbol{\mu}_b + \boldsymbol{\Sigma}_b \mathbf{u}_b)$
    $\mathbf{z}_{c,:2} \leftarrow \boldsymbol{\sigma}_{c,:2} \odot \mathbf{u}_{c,:2} + \boldsymbol{\mu}_{c,:2}$
    $\mathrm{z}_{c,2} \leftarrow \exp(\sigma_{c,2} \mathrm{u}_{c,2} + \mu_{c,2})$
    return $[\mathbf{z}_a;\, \mathbf{z}_b;\, \mathbf{z}_c]$
function DifferentiableNetwork$(\mathbf{u}_h)$
    $\mathbf{h}_0 \leftarrow \tanh(\boldsymbol{W}_0 \mathbf{u}_h + \boldsymbol{b}_0)$
    for $\ell \in \{1 \dots L-1\}$
        $\mathbf{h}_\ell \leftarrow \tanh(\boldsymbol{W}_\ell \mathbf{h}_{\ell-1} + \boldsymbol{b}_\ell) + \mathbf{h}_{\ell-1}$
    return $\boldsymbol{W}_L \mathbf{h}_{L-1} + \boldsymbol{b}_L$
function $\boldsymbol{g}_{\mathbf{x}|\mathbf{z}}([\mathbf{z}_a;\, \mathbf{z}_b;\, \mathbf{z}_c])$
    $\mathbf{P} \leftarrow$ JointPositions$(\mathbf{z}_a, \mathbf{z}_b)$
    $\mathbf{C}_1,\, \mathbf{C}_2 \leftarrow$ CameraMatrices$(\mathbf{z}_c)$
    $\mathbf{X}_1 \leftarrow$ Project$(\mathbf{C}_1, \mathbf{P})$
    $\mathbf{X}_2 \leftarrow$ Project$(\mathbf{C}_2, \mathbf{P})$
    return Flatten$([\mathbf{X}_1; \mathbf{X}_2])$
```

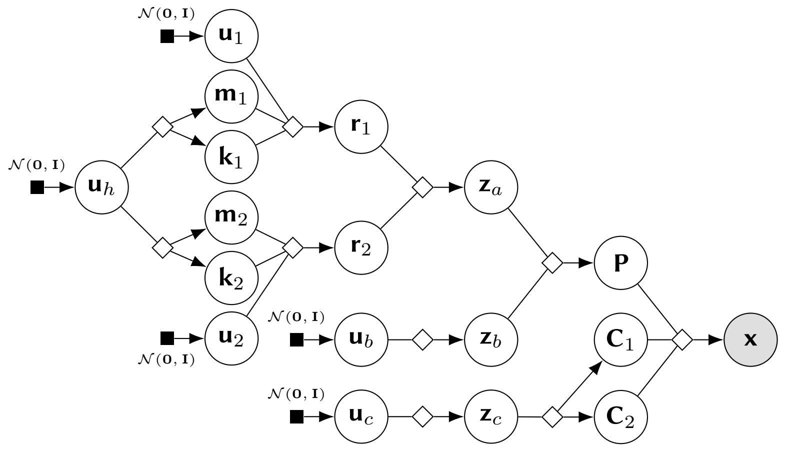


Fig 12: Factor graph of human pose differentiable generative model. The operations corresponding to the deterministic nodes ($\diamond$) in the graph are described in Algorithm 2.

matched by the Euclidean space a Gaussian VAE typically learns a distribution on, and simply 'unwrapping' the angles to e.g. $[-\pi, \pi)$ leads to discontinuities at the $\pm\pi$ cut-point, this both making the initial learning problem challenging (as there is no in-built prior knowledge of continuity across the cut-point) and tending to lead to a learned latent space less amenable to MCMC inference as 'nearby' poses with joint angles on opposite sides of the cut-point will likely end up corresponding to points far apart in the latent space.

During training we therefore mapped each vector of 47 joint angles $\boldsymbol{z}_a^{(i)}$ (corresponding to a single motion capture datapoint) to a pair of 47-dimensional vectors $(\boldsymbol{r}_1^{(i)}, \boldsymbol{r}_2^{(i)})$ by sampling a Gaussian random vector $\boldsymbol{n}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then computing $\boldsymbol{r}_1^{(i)} = \exp \boldsymbol{n}^{(i)} \odot \cos \boldsymbol{z}_a^{(i)}$ and $\boldsymbol{r}_2^{(i)} = \exp \boldsymbol{n}^{(i)} \odot \sin \boldsymbol{z}_a^{(i)}$ and training the VAE to maximise (a variational lower bound) on the joint marginal density of the $\{\boldsymbol{r}_1^{(i)}, \boldsymbol{r}_2^{(i)}\}_i$ pairs. At the cost of doubling the dimension, this leads to an embedding in a Euclidean space which does not introduce any arbitrary cut-points and empirically seemed to lead to better sample quality from the learned generative model compared to learning the angles directly. Given the trained model we can generate a vector of angles $\mathbf{z}_a$ using the model by sampling a Gaussian code (latent representation) vector $\mathbf{u}_h$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ then sampling a pair of 47-dimensional vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ from the learnt Gaussian decoder model given $\mathbf{u}_h$ (and further Gaussian random input vectors $\mathbf{u}_1$ and $\mathbf{u}_2$), and finally recovering an angle by computing $\mathbf{z}_a = \operatorname{atan2}(\mathbf{r}_2, \mathbf{r}_1)$. The resulting distribution on $\mathbf{z}_a$ is only implicitly defined, but the overall generative model is differentiable with respect to the input vectors $\mathbf{u}_h$, $\mathbf{u}_1$ and $\mathbf{u}_2$.

The *PosePrior* motion capture data includes recordings from only a relatively small number of distinct actors and so limited variation in the 'bone-lengths' of the skeleton model. Therefore a separate log-normal model for the bone lengths $\mathbf{z}_b$ was fitted using data from the *ANSUR* anthropometric data-set [39], due to symmetry in the skeleton thirteen independent lengths being specified.

A simple pin-hole projective camera model with three position parameters $\mathbf{z}_c$ and fixed focal-length was used. The camera orientation is fixed to avoid replicating the degrees of freedom specified by the angular orientation of the root joint of the skeleton: only the relative camera–skeleton orientation is important. A log-normal prior distribution was placed on the depth co-ordinate $\mathrm{z}_{c,2}$ to enforce positivity with normal priors on the other two co-ordinates $\mathrm{z}_{c,0}$ and $\mathrm{z}_{c,1}$.

Given a generated triplet of joint-angles, bone length and camera parameters $\mathbf{z}_a$, $\mathbf{z}_b$ and $\mathbf{z}_c$, a binocular pair of simulated two-dimensional projection of the skeleton $\mathbf{x}$ are generated by first mapping the joint-angles and bone-lengths to a 4×19 matrix of joint positions $\mathbf{P}$ in homogeneous world-coordinates by recursing through the skeleton tree. Two 3×4 projective camera matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ are generated from $\mathbf{z}_c$ (with a known fixed offset between the camera centres) and then used to project the world-coordinate joint positions to two 2×19 matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ of joint positions in two-dimensional image-coordinates. The projected positions matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ are flattened to a vector to give the $19 \times 2 \times 2 = 76$ dimensional observed vector $\mathbf{x}$. The overall corresponding model generator functions $\boldsymbol{g}_{\mathbf{x}|\mathbf{z}}$ and $\boldsymbol{g}_{\mathbf{z}}$ are described procedurally in Algorithm 2 and a factor graph summarising the model structure shown in Figure 12.

The generator $\boldsymbol{g}_{\mathbf{x}}$ here has a complex form, not corresponding to either the independent or Markovian observations structures discussed in Section 10.2. The total input dimension is $D_{\mathbf{u}} = 140$ and output dimension $D_{\mathbf{x}} = 76$. The resulting generator Jacobian $\mathbf{J}_{\boldsymbol{g}_{\mathbf{x}}}$ is not full row-rank across the input space; for this binocular observation case there are a maximum of $19 \times 3 = 57$ true degrees of freedom in the skeleton model (three degrees of freedom for each joint) versus the $D_{\mathbf{x}} = 76$ observed dimensions. We therefore define an 'augmented' noisy generator $\boldsymbol{g}_{\mathbf{y}}(\mathbf{u}, \mathbf{n}) = \boldsymbol{g}_{\mathbf{x}}(\mathbf{u}) + \epsilon\mathbf{n}$ to use to perform inference with as discussed in Section 8. We set the noise standard deviation $\epsilon = 0.01$ which produces a non-obvious level of perturbation in visualisations of the generated two-dimensional projections. Similar to the earlier discussion of ABC kernels, we can either consider this additional noise as a computational approximation or as part of the model, representing for example the measurement noise that would be present in two-dimensional joint positions derived from image data.

Under this noisy generator model, the joint density on the generated outputs $\mathbf{y}$ and inputs $\mathbf{u}$ has an explicit form

$$\mathsf{p}_{\mathbf{y},\mathbf{u}}(\boldsymbol{y}, \boldsymbol{u}) = \mathcal{N}\big(\boldsymbol{y} \,|\, \boldsymbol{g}_{\mathbf{x}}(\boldsymbol{u}),\, \epsilon^2\mathbf{I}\big)\mathcal{N}(\boldsymbol{u} \,|\, \mathbf{0}, \mathbf{I}), \tag{96}$$

and so the resulting posterior density $\mathsf{p}_{\mathbf{u}|\mathbf{y}}$ on the model inputs $\mathbf{u}$ given observed $\mathbf{y}$ values can be evaluated up to a normalising constant. This is equivalent to an ABC posterior density in the input space (35) when using a Gaussian kernel.

We generated three sets of binocular two-dimensional joint position projections to use as the observed data for the inference experiments using the *noisy* generator function; these are shown in the left column of Figure 13. Given each of these

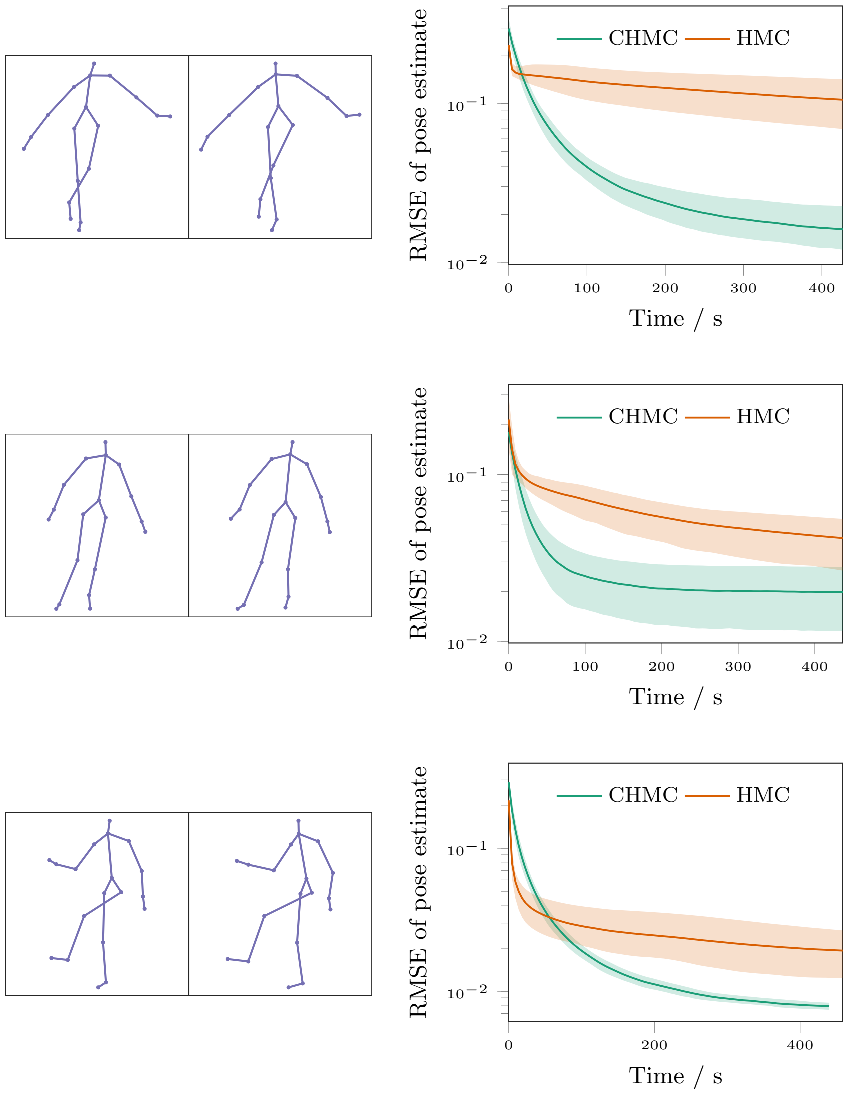


Fig 13: Results of binocular pose inference experiments. In each row the left column shows the generated binocular two-dimensional joint position projections used as the observed data for inference. The right column shows plots of the RMSE of posterior mean estimates of true 3D pose used to generate the binocular projections, using samples from a constrained HMC chains (green) versus standard HMC chains (orange). The horizontal axes of the plot show computation time to produce number of samples in the estimate. Solid curves are averages of RMSE over five chains independently initialised from the prior and shaded regions show ±3 standard deviations.

observed binocular joint projections, we then attempted to infer plausible values for the model inputs $\mathbf{u}$ and so (as they are a deterministic function of the inputs) the latent variables $\mathbf{z}_a$, $\mathbf{z}_b$ and $\mathbf{z}_c$ describing the three-dimensional scene.

We ran chains using the proposed constrained HMC method with the noisy generator $\boldsymbol{g}_{\mathbf{y}}$, the chain state in this case being defined as both $\mathbf{u}$ and $\mathbf{n}$, and chains using standard HMC transitions on the posterior density $\mathsf{p}_{\mathbf{u}|\mathbf{y}}$ on the model inputs $\mathbf{u}$. We used a integrator step size of $\delta t = 0.05$ for the constrained HMC chains, $N_g = 8$ inner geodesic steps and a number of integrator steps per proposed update N_s uniformly sampled from $[10, 20]$. We ran five chains of 200 samples each from independent initialisations. To compute the initial states for the chains, we generated $\mathbf{u}$ values independently from the $\mathcal{N}(\mathbf{0}, \mathbf{I})$ prior and then set the $\mathbf{n}$ values to $\frac{1}{\epsilon}\big(\boldsymbol{x} - \boldsymbol{g}_{\mathbf{y}}(\boldsymbol{u})\big)$ where $\boldsymbol{u}$ are the values sampled from the prior and $\boldsymbol{x}$ is the observed data being conditioned on.

For the standard HMC chains we used an integrator step size of $\delta t = 2 \times 10^{-4}$ and a number of leapfrog steps L per proposed update randomly sampled from $[100, 200]$. We again ran five chains, independently sampling the initial $\mathbf{u}$ state from the normal prior, and running each chain for 1200 samples. The small integrator step size used here was the result of higher step sizes leading to some chains accepting very few or in some cases no updates. This issue was also encountered when using smaller numbers of leapfrog steps per update, including just one integrator step, with these chains however showing marginally slower overall run-time adjusted convergence. Given the small size of ϵ here and the resulting tight concentration of the posterior mass around the fibre $\boldsymbol{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$, the need for a small step size when using an unconstrained HMC approach is not surprising. However the ability of the constrained HMC updates to support a much larger step size provides evidence for the earlier assertion in Section 8 that the constrained HMC method can offer improved performance in such cases.

Due to the high-dimensional nature of the latent variables being inferred it is challenging to assess the convergence of the chains here. As a proxy measure for convergence, we computed the *root mean squared error* (RMSE) of an estimate of the three-dimensional joint positions compared the true positions, computed using the mean of the three-dimensional joint positions generated from the posterior $\mathbf{u}$ input samples. In this binocular case, the disparity in projected joint positions between the two projections gives information about the distances of the corresponding joints from the image plane in the depth direction and so we would expect the posterior distribution on the three-dimensional pose to be concentrated around the true values used to generate the observations.

The right column of Figure 13 shows the RMSE values as a function of the computation time taken to generate the number of samples included in the estimate, for each of the three generated scenes (with the projection visualisations corresponding to the RMSE plots in each row). The constrained HMC chains (green curves) consistently give position estimates which converge more quickly towards the true positions. In this case standard HMC (orange curves) performs relatively poorly despite the significantly cheaper cost of each integrator step compared to the constrained dynamics. The posterior distribution on the model inputs appeared to be multimodal here, with the chains often seeming to converge to

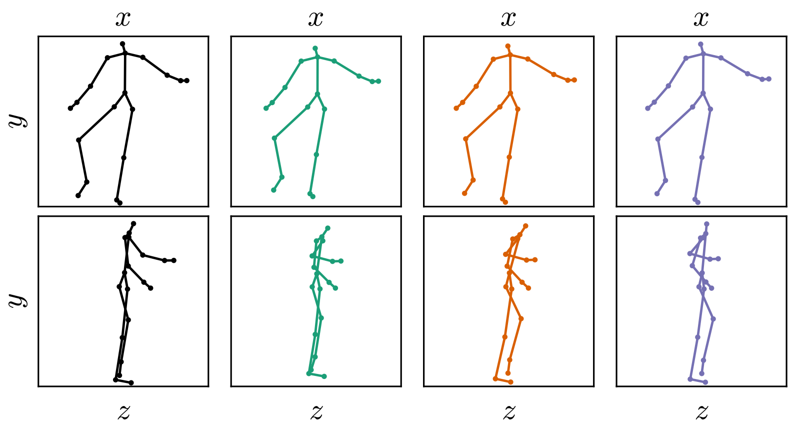


Fig 14: Orthographic projections (top: front view, bottom: side view) of 3D poses consistent with monocular projections. Left most pair (black) shows pose used to generate observations, right three show constrained HMC samples.

slightly different modes. Visually inspecting the sampled poses and individual run traces (not shown) it seems that there are a number of local modes corresponding to a small subset of joints being ‘incorrectly’ positioned; both the HMC and constrained HMC chains are similarly susceptible to getting stuck in local modes, as neither dynamic is likely to overcome large energy barriers.

To highlight the generality of the approach, we also considered inferring three-dimensional scene information from a single two-dimensional projection. Monocular projection is inherently information destroying with significant uncertainty to the true pose and camera parameters which generated the observations. Figure 14 shows pairs of orthographic projections of 3D poses: the left most column is the pose used to generate the projection conditioned on and the right three columns are poses sampled using constrained HMC consistent with the observations. The top row shows front x–y views, corresponding to the camera view though with a orthographic rather than the perspective projection used in the generative model, the bottom row shows side z–y views with the z axis the depth from the camera. The dynamic is able to move between a range of plausible poses consistent with the observations while reflecting the inherent depth ambiguity from the monocular projection.

13. Discussion

We have presented a generally applicable framework for performing inference in differentiable generative models. Though the proposed constrained HMC method is computationally costly, the resulting coherent gradient-based exploration of the state space can lead to significantly improved sampling efficiency over alternative methods as seen in the three inference experiments in the previous section.

Further our approach allows asymptotically exact inference in differentiable generative models where ABC methods might otherwise be used. We suggest an approach for dealing with two of the key issues in ABC methods — enabling inference in continuous spaces as ϵ collapses to zero and allowing efficient inference when conditioning on high-dimensional observations without the need for dimensionality reduction with summary statistics (and the resulting task of choosing appropriate summary statistics).

As well as being of practical importance itself, this approach can be useful in providing 'ground truth' inferences in more complex models to assess the affect of the approximations used in ABC methods on the quality of the inferences. The application of constrained HMC to a generator function outputting summary statistics in the Lotka–Volterra experiments was an interesting example of this: that the posterior conditioning exactly on the summaries without use of an ABC kernel would be nearly as informative about the parameters as conditioning on all of the data was not necessarily obvious a-priori.

In molecular simulations, constrained dynamics are often used to improve efficiency with intra-molecular motion is removed by fixing bond lengths. This allows a larger time-step to be used due to the removal of high-frequency bond oscillations [48]. An analogous effect is present when performing inference in an ABC setting with a ϵ kernel 'soft-constraint' to enforce consistency between the inputs and observed outputs. As $\epsilon \to 0$ the scales over which the posterior density changes value in directions orthogonal to the constraint manifold and along directions tangential to the manifold increasingly differ. To stay within the soft constraint a small step size needs to be used. Using a constrained dynamic decouples the motion on the constraint manifold from the steps to project on to it, allowing more efficient larger steps to be used for moving on the manifold.

A significant limitation of the proposed method is the requirement of differentiability of the generator. This prevents applying our approach to generative models which use discontinuous operations or discrete random inputs. In some cases conditioned on fixed values of discrete random inputs the generator may still be differentiable and so the proposed method can be used to update the continuous random inputs given values of the discrete inputs. To ensure ergodicity of the overall chainl, this would need to be alternated with updates to the discrete inputs, which would require devising methods for updating the discrete inputs to the generator while constraining its output to exactly match observations.

Acknowledgements

Our thanks to Iain Murray for several useful discussions and to Ben Leimkuhler for pointing out the potential of using a geodesic integration scheme. We are also grateful to the anonymous reviewers of this paper and a earlier conference version for their helpful feedback. This work was supported in part by grants EP/F500385/1 and BB/F529254/1 for the University of Edinburgh School of Informatics Doctoral Training Centre in Neuroinformatics and Computational Neuroscience (www.anc.ac.uk/dtc) from the UK Engineering and Physical Sciences Research Council (EPSRC), UK Biotechnology and Biological Sciences Research Council (BBSRC), and the UK Medical Research Council (MRC).

References

[1] I. Akhter and M. J. Black. Pose-conditioned joint angle limits for 3D human pose reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[2] D. Allingham, R. King, and K. L. Mengersen. Bayesian estimation of quantile distributions. *Statistics and Computing*, 19(2):189–201, 2009.

[3] H. C. Andersen. RATTLE: A velocity version of the SHAKE algorithm for molecular dynamics calculations. *Journal of Computational Physics*, 1983.

[4] C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 2009.

[5] C. P. Barnes, S. Filippi, M. P. H. Stumpf, and T. Thorne. Considerate approaches to constructing summary statistics for ABC model selection. *Statistics and Computing*, 22(6):1181–1197, 2012.

[6] E. Barth, K. Kuczera, B. Leimkuhler, and R. D. Skeel. Algorithms for constrained molecular dynamics. *Journal of computational chemistry*, 1995.

[7] S. Barthelmé and N. Chopin. Expectation propagation for likelihood-free inference. *Journal of the American Statistical Association*, 109(505):315–333, 2014.

[8] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*, 2015.

[9] M. A. Beaumont, J.-M. Cornuet, J.-M. Marin, and C. P. Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009.

[10] M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 2002.

[11] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

[12] M. Betancourt. The fundamental incompatibility of scalable Hamiltonian Monte Carlo and naive data subsampling. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[13] M. Betancourt, S. Byrne, and M. Girolami. Optimizing the integrator step size for Hamiltonian Monte Carlo. *arXiv preprint arXiv:1411.6669*, 2014.

[14] M. Betancourt and M. Girolami. Hamiltonian Monte Carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, 79:30, 2015.

[15] M. Bigerelle, D. Najjar, B. Fournier, N. Rupin, and A. Iost. Application of lambda distributions and bootstrap analysis to the prediction of fatigue lifetime and confidence intervals. *International Journal of Fatigue*, 28(3):223–236, 2006.

[16] M. G. Blum. Approximate Bayesian computation: a nonparametric perspective. *Journal of the American Statistical Association*, 105(491):1178–1187, 2010.

[17] M. G. Blum, M. A. Nunes, D. Prangle, and S. A. Sisson. A comparative review of dimension reduction methods in approximate Bayesian

computation. *Statistical Science*, 28(2):189–208, 2013.
[18] G. Bonnet. Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire. *Annals of Telecommunications*, 19(9):203–220, 1964.
[19] M. A. Brubaker, M. Salzmann, and R. Urtasun. A family of MCMC methods on implicitly defined manifolds. In *International Conference on Artificial Intelligence and Statistics*, 2012.
[20] S. Byrne and M. Girolami. Geodesic Monte Carlo on embedded manifolds. *Scandinavian Journal of Statistics*, 2013.
[21] T. Chen, E. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.
[22] T. Christensen, A. Hurn, and K. Lindsay. The devil is in the detail: hints for practical optimisation. *Economic Analysis and Policy*, 38(2):345–368, 2008.
[23] C. J. Corrado et al. Option pricing based on the generalized lambda distribution. *Journal of Futures Markets*, 21(3):213–236, 2001.
[24] J. Dahlin, F. Lindsten, J. Kronander, and T. B. Schön. Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. *arXiv preprint arXiv:1511.05483*, 2015.
[25] G. Deligiannidis, A. Doucet, M. K. Pitt, and R. Kohn. The correlated pseudo-marginal method. *arXiv preprint arXiv:1511.04992*, 2015.
[26] P. Diaconis, S. Holmes, and M. Shahshahani. Sampling from a manifold. In *Advances in Modern Statistical Theory and Applications*, pages 102–125. Institute of Mathematical Statistics, 2013.
[27] P. J. Diggle and R. J. Gratton. Monte Carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 193–227, 1984.
[28] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 1987.
[29] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.
[30] H. Federer. *Geometric measure theory.* Springer, 2014.
[31] M. Freimer, G. Kollia, G. S. Mudholkar, and C. T. Lin. A study of the generalized Tukey lambda family. *Communications in Statistics-Theory and Methods*, 17(10):3547–3567, 1988.
[32] B. J. Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 257–264. Morgan Kaufmann Publishers Inc., 2002.
[33] Y.-X. Fu and W.-H. Li. Estimating the age of the common ancestor of a sample of DNA sequences. *Molecular biology and evolution*, 14(2):195–199, 1997.
[34] A. Gelman, D. Lee, and J. Guo. Stan: A probabilistic programming language for bayesian inference and optimization. *Journal of Educational*

and Behavioral Statistics, 40(5):530–543, 2015.

[35] A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, pages 457–472, 1992.

[36] W. Gilchrist. *Statistical Modelling with Quantile Functions.* CRC Press, 2000.

[37] M. Girolami and B. Calderhead. Riemann-manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[38] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[39] C. C. Gordon, T. Churchill, C. E. Clauser, B. Bradtmiller, J. T. McConville, I. Tebbets, and R. A. Walker. Anthropometric survey of US army personell: Final report. Technical report, United States Army, 1988.

[40] C. Gourieroux, A. Monfort, and E. Renault. Indirect inference. *Journal of applied econometrics*, 8(S1):S85–S118, 1993.

[41] C. Hartmann and C. Schutte. A constrained hybrid Monte-Carlo algorithm and the problem of calculating the free energy in several variables. *ZAMM-Zeitschrift fur Angewandte Mathematik und Mechanik*, 2005.

[42] C. Hastings Jr, F. Mosteller, J. W. Tukey, and C. P. Winsor. Low moments for small samples: a comparative study of order statistics. *The Annals of Mathematical Statistics*, pages 413–426, 1947.

[43] M. D. Hoffman and A. Gelman. The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 2014.

[44] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

[45] R. Kindermann and L. Snell. *Markov random fields and their applications.* American Mathematical Society, 1980.

[46] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2013.

[47] P. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations.* Applications of Mathematics. Springer-Verlag, 1992.

[48] B. Leimkuhler and C. Matthews. Efficient molecular dynamics using geodesic integration and solvent–solute splitting. In *Proc. R. Soc. A.* The Royal Society, 2016.

[49] B. Leimkuhler and G. W. Patrick. A symplectic integrator for Riemannian manifolds. *Journal of Nonlinear Science*, 6(4):367–384, 1996.

[50] B. Leimkuhler and S. Reich. *Simulating Hamiltonian dynamics.* Cambridge University Press, 2004.

[51] B. J. Leimkuhler and R. D. Skeel. Symplectic numerical integrators in constrained Hamiltonian systems. *Journal of Computational Physics*, 1994.

[52] T. Lelièvre, M. Rousset, and G. Stoltz. Langevin dynamics with constraints and computation of free energy differences. *Mathematics of computation*, 2012.

[53] F. Lindsten and A. Doucet. Pseudo-marginal hamiltonian monte carlo. *arXiv preprint arXiv:1607.02516*, 2016.

[54] S. Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160, 1976.

[55] D. J. MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[56] J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 2012.

[57] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 2003.

[58] R. I. McLachlan, K. Modin, O. Verdier, and M. Wilkins. Geometric generalisations of SHAKE and RATTLE. *Foundations of Computational Mathematics*, 14(2):339–370, 2014.

[59] R. McVinish. Improving abc for quantile distributions. *Statistics and Computing*, 22(6):1199–1207, 2012.

[60] E. Meeds, R. Leenders, and M. Welling. Hamiltonian ABC. In *Proceedings of 31st Conference of Uncertainty in Artificial Intelligence*, 2015.

[61] T. Meeds and M. Welling. Optimization Monte Carlo: Efficient and embarrassingly parallel likelihood-free inference. In *Advances in Neural Information Processing Systems*, 2015.

[62] S. Mohamed and B. Lakshminarayanan. Learning in implicit generative models. In *Proceedings of the International Conference on Learning Representations*, 2017.

[63] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. *User Guide for MINPACK-1*. ANL-80-74, Argonne National Laboratory, 1980.

[64] I. Murray. Differentiation of the Cholesky decomposition. *arXiv preprint arXiv:1602.07527*, 2016.

[65] I. Murray and R. P. Adams. Slice sampling covariance hyperparameters of latent Gaussian models. In *Advances in Neural Information Processing Systems*, 2010.

[66] I. Murray, R. P. Adams, and D. J. MacKay. Elliptical slice sampling. In *The Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, volume 9 of *JMLR: W&CP*, pages 541–548, 2010.

[67] I. Murray and M. Graham. Pseudo-marginal slice sampling. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 911–919, 2016.

[68] R. M. Neal. *MCMC using Hamiltonian dynamics*, chapter 5, pages 113–162. Chapman & Hall/CRC, 2011.

[69] A. Öztürk and R. Dale. A study of fitting the generalized lambda distribution to solar radiation data. *Journal of Applied Meteorology*, 21(7):995–1004, 1982.

[70] S. Pal. Evaluation of nonnormal process capability indices using generalized lambda distribution. *Quality Engineering*, 17(1):77–85, 2004.

[71] G. Papamakarios and I. Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. *Advances in Neural*

Information Processing Systems 29, 2016.

[72] O. Papaspiliopoulos, G. O. Roberts, and M. Sköld. Non-centered parameterisations for hierarchical models and data augmentation. In *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, volume 307. Oxford University Press, USA, 2003.

[73] O. Papaspiliopoulos, G. O. Roberts, and M. Sköld. A general framework for the parametrization of hierarchical models. *Statistical Science*, pages 59–73, 2007.

[74] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* Morgan Kaufmann, 1988.

[75] M. Plummer, N. Best, K. Cowles, and K. Vines. CODA: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11, 2006.

[76] M. J. D. Powell. *Numerical Methods for Nonlinear Algebraic Equations*, chapter A Hybrid Method for Nonlinear Equations. Gordon and Breach, 1970.

[77] D. Prangle. Summary statistics in approximate Bayesian computation. *arXiv preprint arXiv:1512.05633*, 2015.

[78] R. Price. A useful theorem for nonlinear devices having Gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.

[79] J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman. Population growth of human Y chromosomes: a study of Y chromosome microsatellites. *Molecular biology and evolution*, 16(12):1791–1798, 1999.

[80] J. S. Ramberg and B. W. Schmeiser. An approximate method for generating asymmetric random variables. *Communications of the ACM*, 17(2):78–82, 1974.

[81] O. Ratmann, C. Andrieu, C. Wiuf, and S. Richardson. Model criticism based on likelihood-free inference, with an application to protein network evolution. *Proceedings of the National Academy of Sciences*, 2009.

[82] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

[83] C. P. Robert, K. Mengersen, and C. Chen. Model choice versus model criticism. *Proceedings of the National Academy of Sciences of the United States of America*, 2010.

[84] D. B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, 12(4):1151–1172, 1984.

[85] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2016.

[86] S. A. Sisson and Y. Fan. *Likelihood-free MCMC*, chapter 12, pages 313–333. Chapman & Hall/CRC, 2011.

[87] S. A. Sisson, Y. Fan, and M. M. Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765, 2007.

[88] J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic*

control, 37(3):332–341, 1992.

[89] B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, 1980.

[90] S. Tavaré, D. J. Balding, R. C. Griffiths, and P. Donnelly. Inferring coalescence times from DNA sequence data. *Genetics*, 145(2):505–518, 1997.

[91] Theano development team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.

[92] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.

[93] D. Tran, R. Ranganath, and D. M. Blei. Deep and hierarchical implicit models. *arXiv preprint arXiv:1702.08896*, 2017.

[94] M.-N. Tran, D. J. Nott, and R. Kohn. Variational bayes with intractable likelihood. *Journal of Computational and Graphical Statistics*, 2017.

[95] J. W. Tukey. Practical relationship between the common transformations of percentages or fractions and of amounts. Technical Report 36, Statistical Research Group,Princeton, 1960.

[96] G. Weiss and A. von Haeseler. Inference of population history using a likelihood approach. *Genetics*, 149(3):1539–1546, 1998.

[97] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[98] R. D. Wilkinson. Approximate Bayesian computation (ABC) gives exact results under the assumption of model error. *Statistical applications in genetics and molecular biology*, 2013.

[99] S. N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, 2010.

[100] E. Zappa, M. Holmes-Cerfon, and J. Goodman. Monte Carlo on manifolds: sampling densities and integrating functions. *arXiv preprint arXiv:1702.08446*, 2017.