Variational inference for nonlinear ordinary differential equations

Sanmitra Ghosh[1] **Paul J. Birrell**[2,1] **Daniela De Angelis**[1,2]
{sanmitra.ghosh,paul.birrell,daniela.deangelis}@mrc-bsu.cam.ac.uk
[1]MRC Biostatistics Unit, University of Cambridge, [2]Public Health England, Colindale

Abstract

We apply the reparameterisation trick to obtain a variational formulation of Bayesian inference in nonlinear ODE models. By invoking the linear noise approximation we also extend this variational formulation to a stochastic kinetic model. Our proposed inference method does not depend on any emulation of the ODE solution and only requires the extension of automatic differentiation to an ODE. We achieve this through a novel and holistic approach that uses both forward and adjoint sensitivity analysis techniques. Consequently, this approach can cater to both small and large ODE models efficiently. Upon benchmarking on some widely used mechanistic models, the proposed inference method produced a reliable approximation to the posterior distribution, with a significant reduction in execution time, in comparison to MCMC.

1 Introduction

A major practical hindrance to the use of Bayesian inference for nonlinear ordinary differential equation (ODE) models is rooted in the computational burden of MCMC algorithms which require repeated numerical integration of an ODE. For complex models such computational burden appears prohibitive. Despite its potential usefulness in ODE inference problems, variational inference (Jordan et al., 1999; Wainwright et al., 2008) has seldom been the chosen method for inference in comparison to MCMC. The fundamental limitation arises from the fact that variational inference requires expectations of the likelihood with respect to the approximating density. In ODE inference problems this expectation is intractable. In more broader terms this is the case for any likelihood distribution that is parameterised using a nonlinear function. Thus, all classical applications of variational inference were limited to probabilistic models where this expectation is analytically tractable.

In this paper we build on recent advances (Kingma et al., 2014; Kucukelbir et al., 2017) in variational inference that enables Bayesian inference of parameters of nonlinear functions such as a neural network, which can be evaluated and differentiated using automatic differentiation (AD) (Baydin et al., 2017), and posit the ODE parameter estimation task as a model agnostic (as our method remains same for any ODE model and likelihood distribution) variational inference problem.

A variety of ODE based methods exist (see Chapter 9 in Särkkä and Solin (2019)) for approximating the transition distribution of a stochastic model. Our proposed variational formulation can be easily adopted to carry out inference in such models using any chosen ODE based approximation. We demonstrate this through the application of the linear noise approximation (LNA) (Kampen, 2007) of a stochastic kinetic model.

2 Inference of ODE parameters

We begin by first introducing the Bayesian framework for inference in a coupled non-linear ODE defined as

$$\frac{dX(t)}{dt} = \boldsymbol{f}\big(X(t), \boldsymbol{\theta}\big) \tag{1}$$

where $X(t) \in \mathbb{R}^K$ is the solution, at each time point, of the system composed of K coupled ODEs – the state vector – and $\boldsymbol{\theta} \in \mathbb{R}^D$ is the parameter vector that we wish to infer. $\boldsymbol{f}(\cdot)$ is a non-linear function representing the vector field. Let $\boldsymbol{X}(\mathbf{x_0}; \boldsymbol{\theta})$ denote the solution of the above system of equations at some specified time points given a set of parameters $\boldsymbol{\theta}$ and initial conditions $\mathbf{x_0}$. In certain models the initial conditions are unknown and these are then estimated along with the model parameters. In these cases we simply extend the parameter vector $\boldsymbol{\theta}$ to include the unknown initial

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

conditions. Also, for notational clarity we omit the dependence of $\mathbf{x_0}$ and denote the parameterised ODE solution as $\boldsymbol{X}(\boldsymbol{\theta}) := \boldsymbol{X}(\mathbf{x_0}; \boldsymbol{\theta})$ throughout the rest of this paper.

Consider a set of noisy experimental observations $\boldsymbol{y} \in \mathbb{R}^{T \times K}$ observed at T experimental time points, $\{t_i\}_{i=0}^{T-1}$, for the K states. We can obtain the likelihood $p(\boldsymbol{y}|\boldsymbol{X}(\boldsymbol{\theta}))$ and combine that with a prior distribution $p(\boldsymbol{\theta})$ on the parameters, using the Bayes theorem, to obtain the posterior density as

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{1}{Z} p(\boldsymbol{y}|\boldsymbol{X}(\boldsymbol{\theta}))p(\boldsymbol{\theta}), \tag{2}$$

where $Z = \int p(\boldsymbol{y}|\boldsymbol{X}(\boldsymbol{\theta}))p(\boldsymbol{\theta})d\boldsymbol{\theta}$ is the intractable marginal likelihood. Due to this intractability we resort to approximate inference and apply MCMC.

2.1 Variational inference

Using variational inference we can approximate $p(\boldsymbol{\theta}|\boldsymbol{y})$ with a tractable distribution $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ from a family of distributions $q(\cdot|\boldsymbol{\lambda})$, indexed by $\boldsymbol{\lambda}$, by minimising the Kullback-Leibler divergence $\mathrm{KL}\big(q(\boldsymbol{\theta}|)||p(\boldsymbol{\theta}|\boldsymbol{y})\big)$. Direct optimisation of $\mathrm{KL}(q||p)$ is intractable. Alternatively, an equivalent quantity, a lower bound to the marginal likelihood, is maximised. This quantity often referred to as the *evidence lower bound* (ELBO) (Jordan et al., 1999) is a tractable objective given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda}) &= \mathbb{E}[\log p(\boldsymbol{y}|\boldsymbol{X}(\boldsymbol{\theta}))p(\boldsymbol{\theta})] - \mathbb{E}[\log q(\boldsymbol{\theta}|\boldsymbol{\lambda})] \\ &= \mathbb{E}[\log p(\boldsymbol{y}|\boldsymbol{X}(\boldsymbol{\theta}))p(\boldsymbol{\theta})] + \mathbb{H}[q(\boldsymbol{\theta}|\boldsymbol{\lambda})]\end{aligned} \tag{3}$$

where $\mathbb{H}$ denotes the entropy of a distribution and the above expectations are with respect to $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$. The goal is then to maximise the ELBO with respect to $\boldsymbol{\lambda}$ in order to find a (possibly unique) $\boldsymbol{\lambda}^*$ that gives the tightest bound.

Note that for an ODE model the first term in Equation (3) that is the expectation of the joint density is intractable. Thus, classical mean-field variational inference algorithms, relying upon this expectation, cannot be applied directly to an ODE problem. Next, we show how this intractability can be overcome using Monte Carlo (MC) quadrature.

3 Variational inference for ODEs

Note that if gradient of the ELBO, w.r.t the variational parameter $\boldsymbol{\lambda}$, is available then variational inference can be formulated as a simple gradient descent problem as follows:

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \eta \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}), \tag{4}$$

where η is a learning rate. For an ODE model this gradient formulation requires the gradient of an intractable expectation. Using the reparameterisation trick Kingma et al. (2014); Rezende et al. (2014); Titsias and Lázaro-Gredilla (2014) we can express the variational approximation $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ as a *non-centred parameterisation* (Papaspiliopoulos and Roberts, 2003). That is we can write $\boldsymbol{\theta}$ as the output of an invertible, differentiable function $g(\boldsymbol{\lambda}, \boldsymbol{\epsilon})$ using a parameter-free distribution $p(\boldsymbol{\epsilon})$. Using this reparameterisation of $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ we can push the gradient inside the expectation, which in turn can be approximated by MC quadrature. The resulting MC estimate, using L samples, of the gradient of the ELBO is given by

$$\begin{aligned}&\nabla_{\boldsymbol{\lambda}} \mathcal{L}_{MC}(\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}} \mathbb{H}\big[q(\boldsymbol{\theta}|\boldsymbol{\lambda})\big] \\ &\quad + \frac{1}{L} \sum_{l=1}^{L} \nabla_{\boldsymbol{\theta}} \left[\log \left\{p(\boldsymbol{y}|\boldsymbol{X}(\boldsymbol{\theta}^l))p(\boldsymbol{\theta}^l)\right\}\right] \nabla_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}, \boldsymbol{\epsilon}^{(l)})\end{aligned} \tag{5}$$

where $\boldsymbol{\theta}^l = g(\boldsymbol{\lambda}, \boldsymbol{\epsilon}^{(l)})$ and $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$. Using the MC estimation of the derivative of the ELBO, and considering the fact that we will be using differentiable probability distributions for the likelihood and priors, we can find the desired $\boldsymbol{\lambda}^*$ using the following update:

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \eta \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{MC}(\boldsymbol{\lambda}). \tag{6}$$

Note that a variety of stochastic gradient descent algorithms (the stochasticity in this case results from the MC estimation) can be employed to carry out the optimisation.

3.1 Constraint on the choice of the approximation

For a large number of inference problems we may constrain the ODE parameters to be positive. For this purpose we generally put prior distributions with support on positive reals. Such priors immediately restrict the class of available distributions for $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$. In order to alleviate this constraint, Kucukelbir et al. (2017) proposed to transform the support of the latent variable, in our case the ODE parameters $\boldsymbol{\theta}$, to the unconstrained real line $\mathbb{R}^D$. To do this we define, following (Kucukelbir et al., 2017), a diffeomorphism, $T : \mathbb{R}^D_{>0} \to \mathbb{R}^D$, and subsequently the transformed parameter vector $\boldsymbol{\phi} = T(\boldsymbol{\theta})$. The posterior density $p(\boldsymbol{\theta}|\boldsymbol{y})$, with the above transformation, is given by

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{X}(T^{-1}(\boldsymbol{\phi})))p(T^{-1}(\boldsymbol{\phi})) \,\big|\det J_{T^{-1}}(\boldsymbol{\phi})\,\big|, \tag{7}$$

where $J_{T^{-1}}(\boldsymbol{\phi})$ is the Jacobian of the inverse of T. This transformation lets us choose an approximating distribution $q(\boldsymbol{\phi}|\boldsymbol{\lambda})$ with unconstrained support, such as a Gaussian. Samples on the original parameter space can be recovered by inverting the transformed parameter samples: $\boldsymbol{\theta} = T^{-1}(\boldsymbol{\phi}),\ \ \boldsymbol{\phi} \sim q(\boldsymbol{\phi}|\boldsymbol{\lambda})$. Thus, in summary we need to choose and apply two transformation $T(\boldsymbol{\theta})$

and $g(\boldsymbol{\lambda}, \boldsymbol{\epsilon})$, the former for transforming the support of the model parameters and the latter for the non-centred parameterisation.

Considering these transformations, we can now re-write the expression for the gradient of the ELBO as follows:

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{MC}(\boldsymbol{\lambda}) &= \nabla_{\boldsymbol{\lambda}}\mathbb{H}\big[q(\boldsymbol{\phi}|\boldsymbol{\lambda})\big] \\ &+ \frac{1}{L}\sum_{l=1}^{L}\nabla_{\boldsymbol{\phi}^{(l)}}\big[\log p(\boldsymbol{y}|\boldsymbol{X}(T^{-1}(\boldsymbol{\phi}^{(l)})))p(T^{-1}(\boldsymbol{\phi}^{(l)})) \\ &+ \log\big|\det J_{T^{-1}}(\boldsymbol{\phi}^{(l)})\big|\big]\nabla_{\boldsymbol{\lambda}}g(\boldsymbol{\lambda}, \boldsymbol{\epsilon}^{(l)})\end{aligned} \tag{8}$$

where $\boldsymbol{\theta} = T^{-1}(\boldsymbol{\phi})$ and $\boldsymbol{\phi}^l = g(\boldsymbol{\lambda}, \boldsymbol{\epsilon}^{(l)}), \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$.

AD, after extending to an ODE, can be used to obtain this gradient to carry out the update in Equation (6). The computational graph for evaluating the ELBO and its gradient is shown in Figure 1(a).

3.2 Full-rank Gaussian variational approximation

A common choice of variational approximation is a factorised (among the dimensions of $\boldsymbol{\phi}$) distribution $q(\boldsymbol{\phi}) = \prod_i^D q_i(\phi_i)$, for example a Normal distribution with a diagonal covariance. However, these factorised distributions would always lack the ability to capture the strong correlation among the parameters of a nonlinear ODE. Instead, we use a full-rank Gaussian distribution as the variational approximation: $q(\boldsymbol{\phi}|\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\phi}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where the vector $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ concatenates the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ and represents the variational parameters. To ensure that the covariance matrix $\boldsymbol{\Sigma}$ remains positive semidefinite we can parameterise the covariance using Cholesky factorisation, $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$, where we use A^T to denote the transpose of the matrix A with a slight abuse of notation. Thus, the variational approximation becomes: $q(\boldsymbol{\phi}|\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\phi}|\boldsymbol{\mu}, \boldsymbol{L}\boldsymbol{L}^T)$. Furthermore, to have a unique $\boldsymbol{L}$ we parameterise the diagonal elements by taking their logarithm. Hence, the variational parameters $\boldsymbol{\lambda} = \big(\boldsymbol{\mu}, (\boldsymbol{L}_{off}, \boldsymbol{L}_{diag})^T\big)$ live in the unconstrained space $\mathbb{R}^{D+D(D+1)/2}$, where we use the shorthands $\boldsymbol{L}_{diag} := \log\operatorname{diag}(\boldsymbol{L})$ and $\boldsymbol{L}_{off} := \boldsymbol{L}_{i,j\in i\neq j}$ to denote the $D(D+1)/2$ real-valued entries parameterising $\boldsymbol{L}$. The required re-parameterisation, $\boldsymbol{\phi} = g(\boldsymbol{\lambda}, \boldsymbol{\epsilon})$, then simply follows as the location-scale transform $\boldsymbol{\phi} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \mathrm{I})$.

The true posterior distribution for an ODE might not resemble a Gaussian density. However, modelling $\boldsymbol{\phi}$ with a Gaussian density in the unconstrained space will induce a non-Gaussian density on the constrained parameters $\boldsymbol{\theta}$. Moreover, a full-rank Gaussian has the potential to capture, adequately, the nonlinear correlation structure of the true posterior distribution.

4 Inference in stochastic kinetic models

Consider a reaction network with u species $X_1, \ldots, X_u$ and v reactions $R_1, \ldots, R_v$. We can write a typical reaction R_i as:

$$R_i: \quad p_{i1}X_1 + \ldots + p_{iu}X_u \to q_{i1}X_1 + \ldots + q_{iu}X_u. \tag{9}$$

Let the vector $\mathcal{X}_t = (\mathcal{X}_1(t), \ldots, \mathcal{X}_u(t))$ denote the number of molecules of each of the u species. The model dynamics can be described by defining a rate (or hazard) function $h_i(\mathcal{X}_t, c_i)$, depending on a rate constant c_i giving the overall hazard of the occurrence of reaction R_i. Under the assumption of mass action kinetics this hazard function is given by

$$h_i(\mathcal{X}_t, c_i) = c_i \prod_{j=1}^{u} \binom{\mathcal{X}_j(t)}{p_{ij}}, \quad i = 1, \ldots, v. \tag{10}$$

We are primarily interested in the inference of the vector of rate constants $\boldsymbol{c} = (c_1, \ldots, c_v)$. This system can be modelled as a Markov Jump Process (MJP), where the probability of a reaction of type i occurring within an infinitesimal time interval $(t, t + dt]$ is $h_i(\mathcal{X}_t, c_i)$. With the occurrence of a type i reaction the system state changes via the i-th row of the net effect matrix $A \in \mathbb{R}^{u\times v}$, with (i, j)-th component: $A_{i,j} = q_{ij} - p_{ij}$. Conveniently, we will use the stoichiometry matrix $S = A^T$. With a specified vector of rate constants $\boldsymbol{c}$ and an initial system state $\mathcal{X}_0$, an MJP can be exactly simulated using the *stochastic simulation algorithm* (SSA) (Gillespie, 1977).

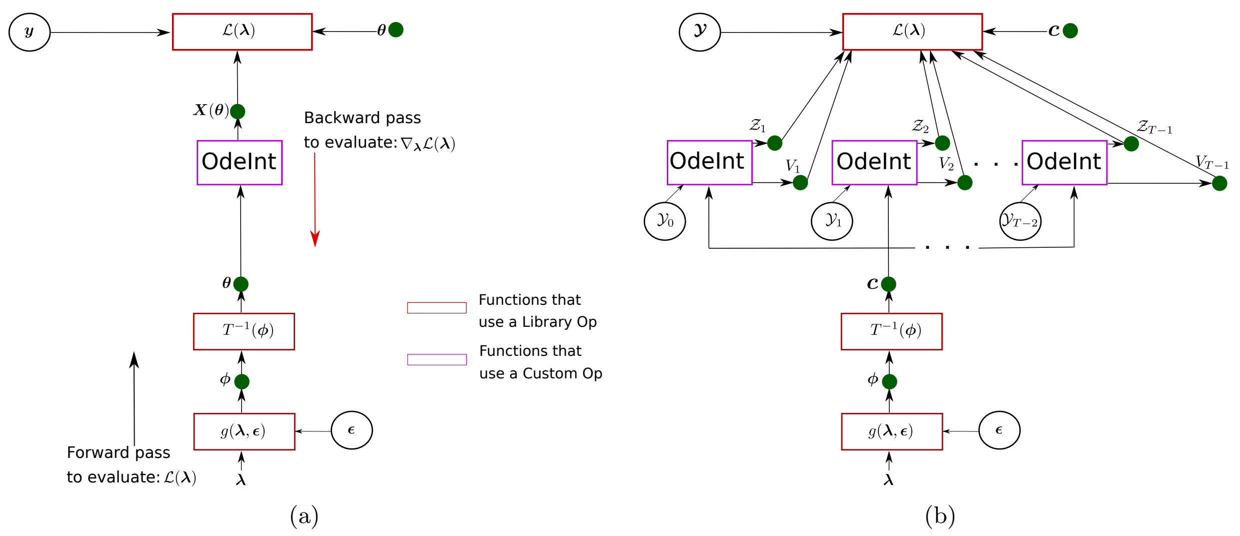
Let us now consider the problem of inferring $\boldsymbol{c}$ given a realisation of the time evolution of the system state $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{T\times u}$ observed at T discrete time points. Note that the MJP has an intractable likelihood. Thus, to arrive at a posterior distribution $p(\boldsymbol{c}|\boldsymbol{\mathcal{Y}})$, often an ABC approximation

$$p(\boldsymbol{c}|\boldsymbol{\mathcal{Y}}) \approx p_\epsilon(\boldsymbol{c}|\boldsymbol{\mathcal{Y}}) = p\Big(\boldsymbol{c}\Big|\Big\{\frac{1}{M}\sum_{m=1}^{M}\mathbb{1}\big(\rho(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{X}}_m) \le \epsilon\big)\Big\}\Big), \tag{11}$$

is used where $\boldsymbol{\mathcal{X}} := \{\mathcal{X}_t\}_{t=0}^{T-1}$ denotes a simulated trajectory obtained through SSA. $\rho(\cdot)$ denotes a distance function, such as the Euclidean distance between the trajectories $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\mathcal{X}}$. $\mathbb{1}(\cdot)$ is the indicator function, ϵ a chosen tolerance and M is the chosen number of simulations. We can obtain the exact posterior when $\epsilon \to 0$. Similar to MCMC, all ABC methods are simulation intensive and thus a faster alternative is desirable.

4.1 Linear noise approximation

Assuming a constant hazard over the infinitesimal time interval $(t, t + dt]$, the MJP can be approximated (see



y
$\mathcal{L}(\boldsymbol{\lambda})$
$\boldsymbol{\theta}$
$\boldsymbol{X}(\boldsymbol{\theta})$
OdeInt
Backward pass
to evaluate: $\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})$
$\boldsymbol{\theta}$
$T^{-1}(\boldsymbol{\phi})$
$\boldsymbol{\phi}$
$g(\boldsymbol{\lambda}, \boldsymbol{\epsilon})$
$\boldsymbol{\epsilon}$
Forward pass
to evaluate: $\mathcal{L}(\boldsymbol{\lambda})$
$\boldsymbol{\lambda}$
Functions that use a Library Op
Functions that use a Custom Op
(a)
$\mathcal{Y}$
$\mathcal{L}(\boldsymbol{\lambda})$
$\boldsymbol{c}$
$\mathcal{Z}_1$
V_1
$\mathcal{Z}_2$
V_2
$\mathcal{Z}_{T-1}$
V_{T-1}
OdeInt
OdeInt
OdeInt
$\mathcal{Y}_0$
$\mathcal{Y}_1$
$\mathcal{Y}_{T-2}$
$\boldsymbol{c}$
$T^{-1}(\boldsymbol{\phi})$
$\boldsymbol{\phi}$
$g(\boldsymbol{\lambda}, \boldsymbol{\epsilon})$
$\boldsymbol{\epsilon}$
$\boldsymbol{\lambda}$
(b)

Figure 1: Computational graph for evaluating the ELBO: The function OdeInt$(\cdot)$ represents a numerical ODE solver that can produce a continuous (dense) solution. A nonlinear ODE solution does not have an analytical expression, thus we implement a custom Op (see Figure 2) to facilitate this function. (a) ELBO for inference in an ODE model. (b) ELBO for the LNA approximation that involves one ODE solve between each pair of observation. In this case each ODE is initialised with value of the observation at the previous time point.

Appendix A) using the following Ito stochastic differential equation, known as *the chemical Langevin equation* (CLE) (Gillespie, 2000):

$$d\mathcal{X}_t = Sh(\mathcal{X}_t, \boldsymbol{c})dt + \sqrt{S\,\mathrm{diag}(h(\mathcal{X}_t, \boldsymbol{c}))S^T}dW_t, \quad (12)$$

where W_t is a u-dimensional vector of standard Brownian motion. The linear noise approximation (LNA) further approximates the MJP by linearising the drift and the noise (diffusion) terms in the CLE.

We begin by replacing the hazard in Equation (12) with a rescaled form: $\Omega f(\mathcal{X}_t/\Omega, \boldsymbol{c})$, where Ω is the volume of the container in which reactions take place. We thus have the rescaling of the CLE as:

$$d\mathcal{X}_t = \Omega S f(\mathcal{X}_t/\Omega, \boldsymbol{c})dt + \sqrt{\Omega S\,\mathrm{diag}(f(\mathcal{X}_t/\Omega, \boldsymbol{c}))S^T}dW_t. \quad (13)$$

We now write the solution of the CLE as a deterministic term plus a residual noise as follows (Kampen, 2007): $\mathcal{X}_t = \Omega\mathcal{Z}_t + \sqrt{\Omega}\mathcal{M}_t$. The evolution of $\mathcal{Z}_t$ and $\mathcal{M}_t$ are given by the set of an ODE and SDE (see Appendix A) as follows:

$$\begin{aligned} \frac{d\mathcal{Z}_t}{dt} &= Sf(\mathcal{Z}_t, \boldsymbol{c}), \\ d\mathcal{M}_t &= SF\mathcal{M}_t dt + \sqrt{S\,\mathrm{diag}(f(\mathcal{Z}_t, \boldsymbol{c}))S^T}dW_t, \end{aligned} \quad (14)$$

where $F_t \in \mathbb{R}^{v\times u}$ is the Jacobian matrix with (i,j)-th components: $\frac{\partial f_i(\mathcal{Z}_i(t), \boldsymbol{c})}{\partial \mathcal{Z}_j(t)}$. The expansion of $\mathcal{X}_t$ into $\mathcal{Z}_t, \mathcal{M}_t$ along with Equation (14) comprises the LNA approximation. Given a Gaussian initial value $\mathcal{M}_0 \sim \mathcal{N}(m_0, V_0)$, the residual at time t is given, upon solving the linear SDE explicitly, by $\mathcal{M}_t \sim \mathcal{N}(m_t, V_t)$ where the mean and covariance are given by the following ODEs (Golightly and Gillespie, 2013):

$$\begin{aligned} \frac{dm_t}{dt} &= SF_t m_t, \\ \frac{dV_t}{dt} &= V_t F_t^T S^T + S\,\mathrm{diag}(h(\mathcal{Z}_t, \boldsymbol{c}))S^T + SF_t V_t. \end{aligned} \quad (15)$$

The ODEs for $\mathcal{Z}_t, m_t, V_t$ need to be solved numerically with inital values: $\mathcal{Z}_0 = \mathcal{X}_0/\Omega, m_0 = (\mathcal{X}_0 - \Omega\mathcal{Z}_0)/\sqrt{\Omega}, V_0 = \mathbf{0}_{u\times u}$, $\mathcal{X}_0$ is a user supplied initial number of molecules. The state at time t is then obtained as $\mathcal{X}_t \sim \mathcal{N}(\Omega\mathcal{Z}_t + \sqrt{\Omega}m_t, \Omega V_t)$. The linear noise approximation of the system state $\mathcal{X}_t$ is shown to produce close approximation of the MJP, for high concentration scenarios (Golightly and Gillespie, 2013).

4.2 Inference using LNA

A realisation, using the LNA, of an MJP can be obtained at discrete time points by sampling $\mathcal{X}_t$ from a Gaussian parameterised by the solution of the ODEs for $\mathcal{Z}_t$, m_t and V_t. However, solving the ODE for $\mathcal{Z}_t$ once for the entire time horizon may lead to a poor approximation (Giagos, 2010). Thus, to alleviate this problem $\mathcal{Z}_t$ can be set to $\mathcal{X}_t/\Omega$ at each t and solved along with V_t within each time interval. In that case, $m_t = 0$ for all t and need not be solved. This results in $\mathcal{X}_t \sim \mathcal{N}(\Omega\mathcal{Z}_t, \Omega V_t)$ for each t. Given an observed noise-free realisation (data) $\boldsymbol{\mathcal{Y}} \equiv \boldsymbol{\mathcal{X}}$ at T experimental time points $\{t_i\}_{i=0}^{T-1}$, the desired posterior distribution for $\boldsymbol{c}$ is then given by

$$\begin{aligned} p_{LNA}(\boldsymbol{c}|\boldsymbol{\mathcal{Y}}) &\propto p(\boldsymbol{c})p(\boldsymbol{\mathcal{Y}}|\boldsymbol{c}) \\ &= p(\boldsymbol{c})\prod_{i=1}^{T-1} p(\mathcal{Y}_{t_i}|\mathcal{Y}_{t_{i-1}}, \boldsymbol{c}) \\ &= p(\boldsymbol{c})\prod_{i=1}^{T-1} \mathcal{N}(\mathcal{Y}_{t_i}|\Omega\mathcal{Z}_{t_i}, \Omega V_{t_i}), \end{aligned} \tag{16}$$

where due to the noise-free nature of the observations we solve the ODEs for $\mathcal{Z}_{t_i}, V_{t_i}$ using $\mathcal{Y}_{t_{i-1}}$ as the initial number of molecules. Thus, we solve these ODEs once between each pair of observations. Pseudocode for calculating the likelihood above is given below in Algorithm 1.

Algorithm 1 LNA likelihood calculation

```
Input: $\boldsymbol{\mathcal{Y}}$, $\boldsymbol{c}$, $\Omega$, $\{t_i\}_{i=0}^{T-1}$.
Initialise $i = 0$, $\mathcal{Z}_{t_0} = \mathcal{Y}_{t_0}/\Omega$ and $V_{t_0} = \mathbf{0}_{u\times u}$.
for $i = 1$ to $T - 1$ do
    Solve the ODEs for $\mathcal{Z}_{t_i}, V_{t_i}$ given by Equation (14) and (15) within $[t_{i-1}, t_i]$ with initial values: $\mathcal{Z}_{t_{i-1}}, V_{t_{i-1}}$.
    Obtain likelihood factors: $p(\mathcal{Y}_{t_i}|\mathcal{Y}_{t_{i-1}}, \boldsymbol{c}) = \mathcal{N}(\mathcal{Y}_{t_i}|\Omega\mathcal{Z}_{t_i}, \Omega V_{t_i})$.
    Set $\mathcal{Z}_{t_i} = \mathcal{Y}_{t_i}/\Omega$ and set $V_{t_i}$ as a $\mathbf{0}_{u\times u}$.
end for
Obtain likelihood as product of the factors: $p(\boldsymbol{\mathcal{Y}}|\boldsymbol{c}) = \prod_{i=1}^{T-1} p(\mathcal{Y}_{t_i}|\mathcal{Y}_{t_{i-1}}, \boldsymbol{c})$.
Result: $p(\boldsymbol{\mathcal{Y}}|\boldsymbol{c})$
```

Note that $p_{LNA}(\boldsymbol{c}|\boldsymbol{\mathcal{Y}})$ is differentiable and thus the variational formulation derived earlier can be easily applied. Plugging this posterior in Equation (8) we have the ELBO for LNA given by

$$\begin{aligned} &\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{MC}(\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}}\mathbb{H}\big[q(\boldsymbol{\phi}|\boldsymbol{\lambda})\big] \\ &+ \frac{1}{L}\sum_{l=1}^{L}\nabla_{\boldsymbol{\phi}^{(l)}}\big[\log p(\boldsymbol{\mathcal{Y}}|T^{-1}(\boldsymbol{\phi}^{(l)}))p(T^{-1}(\boldsymbol{\phi}^{(l)})) \\ &+ \log\big|\det J_{T^{-1}}(\boldsymbol{\phi}^{(l)})\big|\big]\nabla_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}, \boldsymbol{\epsilon}^{(l)}) \end{aligned} \tag{17}$$

where $\boldsymbol{c} = T^{-1}(\boldsymbol{\phi})$ and $\boldsymbol{\phi}^l = g(\boldsymbol{\lambda}, \boldsymbol{\epsilon}^{(l)})$, $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$. The corresponding computational graph for evaluating the ELBO, for the LNA approximation, is shown in Figure 1(b).

5 ODE sensitivity analysis

Suppose we are given a cost function, such as the ELBO, on the ODE solution at the measurement times:

$$C(X(t)) = \sum_i c(X(t_i;\boldsymbol{\theta})). \tag{18}$$

Sensitivity analysis can be used to obtain the gradient of the cost function w.r.t the ODE parameters. By the chain rule $\frac{\partial C(X(t))}{\partial \boldsymbol{\theta}}$ requires the model sensitivities $\frac{\partial X(t)}{\partial \boldsymbol{\theta}}$. In *forward sensitivity analysis* we can obtain the model sensitivities, at each time point, as the solution to an initial value problem by augmenting the state-space of system ODEs (Equation (1)) with the sensitivity terms, $S_{kd}(t)$:

$$S_{kd}(t) = \frac{d}{dt}\left\{\frac{\partial X_k(t)}{\partial \theta_d}\right\} = \frac{\partial \boldsymbol{f}}{\partial X}\frac{\partial X}{\partial \theta_d} + \frac{\partial \boldsymbol{f}}{\partial \theta_d}, \tag{19}$$

where $\frac{\partial \boldsymbol{f}}{\partial X}$ is the Jacobian of the velocity function $\boldsymbol{f}$ with respect to the current state $X(t)$, and $\frac{\partial \boldsymbol{f}}{\partial \theta_d}$ is the gradient of the velocity with respect to the d-th parameter. This augmented ODE $\big(X(t), S_{kd}(t)\big)$ can then be solved together using a chosen numerical method. In *adjoint sensitivity* analysis (Rackauckas et al., 2018) the gradient of a scalar-valued cost function $C(\cdot)$, whose input is the ODE solution, can be computed directly. The first step is to solve a backwards ODE, the adjoint problem:

$$\frac{d\boldsymbol{a}(t)}{dt} = -\boldsymbol{a}(t)^{\mathrm{T}}\frac{\partial \boldsymbol{f}}{\partial X}, \tag{20}$$

where at each experimental time point t_i this backward ODE is perturbed by $\frac{\partial c(X(t_i;\boldsymbol{\theta}))}{\partial X}$. The gradient of the cost function with respect to the ODE parameters can be evaluated by another quadrature as follows:

$$\frac{dC}{d\boldsymbol{\theta}} = \boldsymbol{a}(t_0)^{\mathrm{T}}\frac{\partial \boldsymbol{f}\big(X(t_0),\boldsymbol{\theta}\big)}{\partial \boldsymbol{\theta}} + \sum_i \int_{t_i}^{t_{i+1}} \boldsymbol{a}(t)^{\mathrm{T}}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}dt. \tag{21}$$

Note that a continuous solution of the system and the adjoint states is required for this scheme which can be memory intensive if K is large. Chen et al. (2018) avoided this bottleneck by solving the system, the adjoint and the cost function ODEs simultaneously and backward in time. However, this approach implicitly assumes reversibility of ODE solvers. One can also use a *checkpointing* (of a forward solve of $X(t)$, at each t_i), scheme that uses backward integration for the adjoint and cost function only. See Appendix B for more details on this.

Forward sensitivity requires $K \times D$ ODEs to be solved, in comparison to $2K + D$ for adjoint, which becomes infeasible for systems with large parameter dimension. However, as shown in Rackauckas et al. (2018) forward sensitivity can be much more efficient for small number of parameters. Interestingly, many widely used mechanistic models fall in this category. See Appendix B for a discussion of the choice between these methods.

5.1 Extending AD to an ODE

In automatic differentiation a function composition is broken down to a sequence of elementary operations, or *ops* in short, applied to the inputs as well as other intermediate variables that have analytical expressions for outputs and derivatives. Clearly for any composition of function involving an ODE, without analytical expressions, it is impossible to express them using the supported (library) ops of most standard AD packages. To overcome this limitation we will give a recipe for defining a custom (user supplied) operation that utilises sensitivity analysis techniques described previously.

To illustrate the functionality of the custom op let us introduce a simple example where we want to evaluate an arbitrary differentiable function $\xi : \mathbb{R}^{T \times K} \to \mathbb{R}$, whose input is a parameterised ODE solution, $\boldsymbol{X}(\boldsymbol{\theta})$. We want to compute its function value $z = \xi(\boldsymbol{X}(\boldsymbol{\theta}))$ and its gradient $\frac{\partial z}{\partial \boldsymbol{\theta}}$ with respect to the parameter vector using a custom ODE op and we further assume that both the function $\xi(\boldsymbol{X}(\boldsymbol{\theta}))$ and its gradient $\boldsymbol{X}(\boldsymbol{\theta})$ can be expressed analytically. The custom op receives two arguments (see Figure 2). The first is simply the ODE parameters $\boldsymbol{\theta}$ and the second is the *adjoint*, $\frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})}$, where the latter being analytically tractable is evaluated by library ops. The first input is used to calculate the custom op's output, that is the ODE solution, while the latter is used to evaluate the action of the op's Jacobian on the adjoint:

$$\frac{\partial z}{\partial \boldsymbol{\theta}} = \frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})}^{T} \cdot \frac{\partial \boldsymbol{X}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \tag{22}$$

the *vector-Jacobian product* (VJP), which is returned as and when the custom op's gradient is queried. Note that this is also the down-stream adjoint when $\boldsymbol{\theta}$ is the output of another function.

VJP using forward sensitivity: In this case we multiply the ODE sensitivity $\frac{\partial \boldsymbol{X}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, obtained along-with the numerical solution, to the op's second input, the adjoint $\frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})}$, directly to obtain the desired VJP given by Equation (22). Since we are numerically implementing the VJP we need to rearrange the adjoint $\frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})}$ and the sensitivity $\frac{\partial \boldsymbol{X}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ to produce a vector and a matrix respectively using the $vec : \mathbb{R}^{n \times m} \to \mathbb{R}^{nm}$ operation which stacks the column of an $n \times m$ matrix to produce an nm-dimensional vector. Applying $vec\left(\frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})}\right)$ we obtain the flattened TK-dimensional adjoint vector and subsequently we rearrange the sensitivities to obtain a $TK \times D$ matrix. We can now evaluate the desired numerical VJP for the custom op as a matrix-vector product.

VJP using adjoint sensitivity: By repeatedly solving the cost function ODE given by Equation (21), and noting that the cost function ODE in this case is $\frac{\partial z}{\partial \boldsymbol{\theta}}$, the VJP can be directly evaluated, without obtaining the Jacobian explicitly as in forward sensitivity. The custom op's second input, the adjoint $\frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})} = (\boldsymbol{a}(t_0), \ldots, \boldsymbol{a}(t_T))$, at each time point is used to perturb the adjoint ODE, in Equation (20).

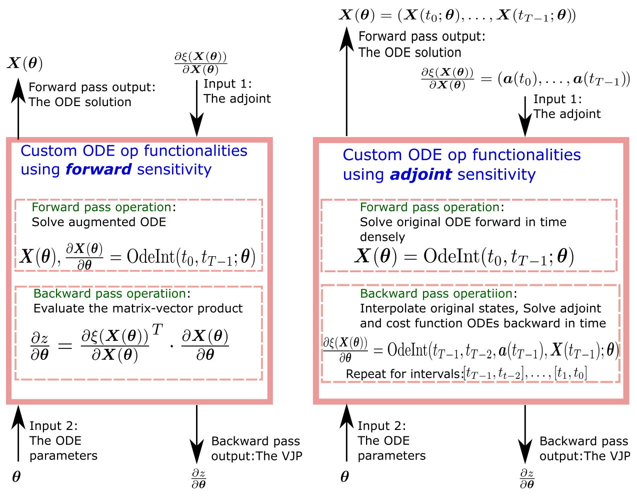

$\boldsymbol{X}(\boldsymbol{\theta})$
Forward pass output: The ODE solution
$\frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})}$
Input 1: The adjoint
Custom ODE op functionalities using ***forward*** sensitivity
Forward pass operation:
Solve augmented ODE
$\boldsymbol{X}(\boldsymbol{\theta}), \frac{\partial \boldsymbol{X}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathrm{OdeInt}(t_0, t_{T-1}; \boldsymbol{\theta})$
Backward pass operatiion:
Evaluate the matrix-vector product
$\frac{\partial z}{\partial \boldsymbol{\theta}} = \frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})}^{T} \cdot \frac{\partial \boldsymbol{X}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
Input 2: The ODE parameters
$\boldsymbol{\theta}$
Backward pass output:The VJP
$\frac{\partial z}{\partial \boldsymbol{\theta}}$
$\boldsymbol{X}(\boldsymbol{\theta}) = (\boldsymbol{X}(t_0; \boldsymbol{\theta}), \ldots, \boldsymbol{X}(t_{T-1}; \boldsymbol{\theta}))$
Forward pass output: The ODE solution
$\frac{\partial \xi(\boldsymbol{X}(\boldsymbol{\theta}))}{\partial \boldsymbol{X}(\boldsymbol{\theta})} = (\boldsymbol{a}(t_0), \ldots, \boldsymbol{a}(t_{T-1}))$
Input 1: The adjoint
Custom ODE op functionalities using ***adjoint*** sensitivity
Forward pass operation:
Solve original ODE forward in time densely
$\boldsymbol{X}(\boldsymbol{\theta}) = \mathrm{OdeInt}(t_0, t_{T-1}; \boldsymbol{\theta})$
Backward pass operatiion:
Interpolate original states, Solve adjoint and cost function ODEs backward in time
$\frac{\partial \xi(\boldsymbol{X}(\theta))}{\partial \theta} = \mathrm{OdeInt}(t_{T-1}, t_{T-2}, \boldsymbol{a}(t_{T-1}), \boldsymbol{X}(t_{T-1}); \boldsymbol{\theta})$
Repeat for intervals:$[t_{T-1}, t_{t-2}], \ldots, [t_1, t_0]$
Input 2: The ODE parameters
$\boldsymbol{\theta}$
Backward pass output:The VJP
$\frac{\partial z}{\partial \boldsymbol{\theta}}$

Figure 2: Block diagram showing the functionalities of a custom ODE op, for a simple illustrative example.

Using a custom ODE op as described above (see Figure 2) we can now evaluate the gradient of the ELBO in Equation (8) w.r.t $\boldsymbol{\lambda}$, end-to-end using AD, as long as differentiable density functions are used for the probabilistic components. Our custom op recipe can be used with most AD package and any off-the-shelf ODE solver. Note that either the system Jacobians $\frac{\partial \boldsymbol{f}}{\partial X}$, $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}$, or the corresponding matrix-vector products in (20), (21) can be calculated in a number of ways including AD. We have discussed these options in Appendix B.

6 Related work

Variational inference for ODEs: Meeds et al. (2019) proposed an amortised inference scheme applicable to ODEs. However, this method relied on backpropagation through an ODE solver steps which introduces numerical errors and high memory overload. Using our AD approach for ODEs, high precision off-the-shelf solvers can be easily used. Gorbach et al. (2017) applied the mean-field variational inference to ODEs using *gradient matching*. However, this method suffers from the limitations of gradient-matching (see below). Moreover, mean-field approximation is inadequate to represent the complex ODE posteriors.

Gradient matching using a Gaussian process (GP) to emulate the velocity field, (Macdonald et al., 2016; Wenk et al., 2019) can speed-up inference in ODEs. However, such approaches require complete observability [1] of the ODE states. This is the case in a handful

[1]By observability we mean that observations are avail-

of models. Moreover, this approach is only applicable to Gaussian likelihood. Our variational formulation is applicable to problems with partially observed systems and non-Gaussian likelihoods (see section 7.1).

Linear noise approximation has been used previously for Bayesian inference, using MCMC, in stochastic models in Fearnhead et al. (2014); Giagos (2010); Golightly and Wilkinson (2015). By introducing the variational formulation along with LNA, we provide a faster alternative to these approaches.

AD for ODE solvers using adjoint sensitivity analysis was proposed in Chen et al. (2018) for ODEs with a velocity field described by neural networks. This method is designed around the constraints posed by an extremely large system. Our approach is more holistic as we propose a framework useful for both small and large systems.

7 Benchmarking

In order to evaluate the efficacy of variation inference we used two models: i) the SIR and ii) protein transduction model. To test our inference method using LNA we have used the stochastic Lotka-Volterra predator-prey model.

For the first two problems we have compared the variational inference estimate against a corresponding "gold-standard" MCMC estimate, where we have used the No-U-Turn (NUTS) algorithm (Hoffman and Gelman, 2014) because of its widespread use. For the stochastic model we have compared variational inference using our LNA approach to the ABC-SMC (Toni et al., 2009) algorithm with a SSA based simulator of the MJP.

We have used Pytorch to implement our custom ODE op and have used the LSODA solver (Petzold, 1983) through SciPy's Python wrapper. To obtain the system Jacobians, $\frac{\partial \boldsymbol{f}}{\partial X}$, $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}$, we have used the SymPy package as this has deep integration with Numpy and was found to be faster than AD. However, we have provided code to obatain these using AD as well. We have also used the Pyro (Bingham et al., 2019) probabilistic programming package, to easily compare variational inference with the NUTS algorithm. For the ABC-SMC we have written bespoke Python code with the SSA simulator written in C++. The code can be accessed from `https://github.com/sg5g10/VBODE`

In all the examples, for variational inference, we have used a stochastic gradient algorithm proposed in Kucukelbir et al. (2017) which is a mixture of RmsProp (Tieleman and Hinton, 2016) and AdaGrad (Duchi et al., 2011) for carrying out the optimisation. We set the learning rate scale to 0.5 and have used 10,000 iterations with $L = 1$ throughout.

We considered 1000 independent samples generated using MCMC representing, pointwise, the underlying true posterior as the "gold standard". For the first two examples, two chains of NUTS is run for 1000 iterations after an initial 500 warmup iterations. The NUTS samples, from the two chains, are then thinned to obtain the 1000 samples. NUTS being a high ESS sampler needs 1000 post-warmup iterations to produce roughly ≈ 1000 independent samples from the posterior. For plotting and summarising the posterior distributions, and comparing to the "gold standard" we used 1000 samples from the variational approximation throughout. We have used **VI** to refer to the variational inference method.

7.1 The SIR compartmental model

The SIR model (Anderson et al., 1992) of infectious disease models the number of susceptible (S), infected (I), and recovered (R) people in a population subjected to an epidemic. The SIR model, for a population of N people, is defined by the following ODE system:

$$\frac{dS}{dt} = -\beta S\frac{I}{N}, \quad \frac{dI}{dt} = \beta S\frac{I}{N} - \gamma I, \quad \frac{dR}{dt} = \gamma I, \quad (23)$$

where the infection β and recovery γ rates are unknown parameters. Also, we have $N = S(t) + I(t) + R(t)$. To illustrate the model fitting, we used data on common colds infections in Tristan da Cunha obtained from Toni et al. (2009). This data consists of the number of infections $\boldsymbol{I_{obs}}$ and number of recoveries $\boldsymbol{R_{obs}}$ for a period of 21 days. The population size is $N = 300$. We used only the number of infections time-series as data for inference. In addition to β, γ we also estimated the initial susceptibility, $s_0 = S(t = 0)/N$, assuming the initial recovered fraction $r_0 = 0$ and thus $i_0 = 1 - s_0$. As this is count data we have used a Poisson likelihood, $y(t)|\beta, \gamma, s_0 \sim \text{Poisson}(I(t))$, and placed the following priors: $\beta \sim \text{Gamma}(2, 1)$, $\gamma \sim \text{Gamma}(2, 1)$ and $s_0 \sim \text{Beta}(0.5, 0.5)$.

We summarise the posterior marginals in Table 1. Table 2 compares the run-time of **VI** (using both the sensitivity techniques for the VJP) to **NUTS**. We noticed a significant speed-up for **VI** and a good match to the **NUTS** estimates. The model fit to the data is shown in Figure 3. The density plots are shown in Appendix C.

7.2 Protein transduction model

This model was first introduced in Vyshemirsky and Girolami (2008), and has been used consistently to benchmark gradient-matching algorithms (see Wenk

able for all the ODE states.

Table 1: We summarise the **mean ± standard deviation** of the posterior distribution of each parameter for the three examples. We used **VI-FOR**, **VI-ADJ** to denote estimates from **VI** run with the forward and adjoint based VJP. Similarly, we denote **NUTS-FOR**, **NUTS-ADJ** to point out the associated VJP type.

		The SIR model			
$\boldsymbol{\theta}$	True Value	**VI-FOR**	**VI-ADJ**	**NUTS-FOR**	**NUTS-ADJ**
β	–	1.7182 ± 0.1171	1.7182 ± 0.1171	1.7099 ± 0.1171	1.7182 ± 0.1163
γ	–	1.2077 ± 0.0979	1.2077 ± 0.0979	1.2077 ± 0.0760	1.2088 ± 0.0799
s_0	–	0.9960 ± 0.0013	0.9960 ± 0.0013	0.9959 ± 0.0014	0.9960 ± 0.0012

		The Protein Transduction model			
$\boldsymbol{\theta}$	True Value	**VI-FOR**	**VI-ADJ**	**NUTS-FOR**	**NUTS-ADJ**
p_1	0.07	0.0682 ± 0.0017	0.0683 ± 0.0016	0.0681 ± 0.0015	0.0680 ± 0.0015
p_2	0.6	0.5855 ± 0.0168	0.5865 ± 0.0162	0.5876 ± 0.0147	0.5877 ± 0.0153
p_3	0.05	0.0497 ± 0.0107	0.0498 ± 0.0091	0.0507 ± 0.0095	0.0513 ± 0.0094
p_4	0.3	0.2943 ± 0.0060	0.2964 ± 0.0061	0.2955 ± 0.0051	0.2958 ± 0.0052
p_5	0.017	0.0203 ± 0.0020	0.0205 ± 0.0020	0.0197 ± 0.0018	0.0198 ± 0.0018
p_6	0.3	0.4218 ± 0.0819	0.4301 ± 0.0868	0.3997 ± 0.0769	0.4063 ± 0.0765

		The Stochastic Lotka-Volterra model		
$\boldsymbol{c}$	True Value	**VI-FOR**	**VI-ADJ**	**ABC-SMC**
c_1	0.53	$.5356 \pm 0.0209$	0.5356 ± 0.0209	0.5314 ± 0.0226
$100 \times c_2$	0.25	0.2484 ± 0.0092	0.2484 ± 0.0092	0.2434 ± 0.0096
c_3	0.3	0.2988 ± 0.0091	0.2988 ± 0.0091	0.3227 ± 0.0163

et al. (2019) and references there in). This system is described by the following ODEs:

$$\begin{aligned}
\dot{S} &= -p_1 S - p_2 SR + p_3 RS, \quad \dot{dS} = p_1 S,\\
\dot{R} &= -p_2 SR + p_3 R_s + p_5 \frac{R_{pp}}{p_6 + R_{pp}},\\
\dot{R_s} &= p_2 SR - p_3 R_s - p_4 R_s,\\
\dot{R_{pp}} &= p_4 R_s - p_5 \frac{R_{pp}}{p_6 + R_{pp}},
\end{aligned} \tag{24}$$

where the unknown parameters $\boldsymbol{\theta} = (p_1, \ldots, p_6)$ need to be estimated. We reproduced the experimental setup introduced in Dondelinger et al. (2013) where the system was simulated within the time interval $[0, 100]$, and the states $X(t) = (S(t), dS(t), R(t), Rs(t), R_{pp}(t))$ at discrete time points $t = [0, 1, 2, 4, 5, 7, 10, 15, 20, 30, 40, 50, 60, 80, 100]$ were corrupted with a Gaussian noise with $\sigma = 0.01$, to form the observed data $\boldsymbol{y}$. The parameters for the simulation was set to $\boldsymbol{\theta} = (0.07, 0.6, 0.05, 0.3, 0.017, 0.3)$. The initial values $\mathbf{x_0}$ were set to $[1, 0, 1, 0, 0]$ and assumed to be known, as done in previous benchmarking studies cited above. The likelihood is a Gaussian, $p(\boldsymbol{y}|\boldsymbol{\theta}) = \prod_i \mathcal{N}(X(t_i), \sigma^2 \mathbb{I})$, where $\mathbb{I}$ is a 5×5 identity matrix. We place a Gamma$(1, 2)$ prior on all the parameters.

The posterior estimates are summarised in Table 1 and the run-times are again furnished in Table 2. The density and model-fit plots are shown in Appendix C. In this case, for both forward and adjoint **VI** runs, we noticed a significant (almost dramatic) speed-up. This model is known to be challenging and thus the **NUTS** algorithm tunes to a small step-size and slows down. We repeated the inference process using two more realisations of the artificial noise. Additional results are summarised in Appendix C.

7.3 Stochastic Lotka-Volterra model

The stochastic Lotka–Volterra model (Wilkinson, 2018) has been widely used for benchmarking (see Fearnhead et al. (2014); Giagos (2010)). This model describes a population comprising of two competing species: *predators* which die with rate c_2 and reproduce with rate c_1 by consuming prey, which in turn reproduce with rate c_3. This system can be defined through the following list of reactions:

$$\begin{aligned}
R_1 &: \quad X_1 \xrightarrow{c_1} 2X_1\\
R_2 &: \quad X_1 + X_2 \xrightarrow{c_2} 2X_2\\
R_3 &: \quad X_2 \xrightarrow{c_2} \emptyset,
\end{aligned} \tag{25}$$

where we denote by X_1, X_2 the prey and predator species respectively. We further denote the corresponding numbers of the species as the system state $\mathcal{X}_t = (\mathcal{X}_1(t), \mathcal{X}_2(t))$. The hazard vector for this system is $h(\mathcal{X}_t, \boldsymbol{c}) = \big(c_1\mathcal{X}_1(t), c_2\mathcal{X}_1(t)\mathcal{X}_2(t), c_3\mathcal{X}_2(t)\big)$. Following, Fearnhead et al. (2014) we assume that $\Omega = 1$. Thus, $f(\mathcal{X}_t, \boldsymbol{c}) = h(\mathcal{X}_t, \boldsymbol{c})$. The stoichiometry S and the Jacobian F_t matrices are given by

$$S = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}, \quad F_t = \begin{pmatrix} c_1 & 0 \\ c_2\mathcal{Z}_2(t) & c_2\mathcal{Z}_1(t) \\ 0 & c_3. \end{pmatrix} \tag{26}$$

We set the initial values as $\mathcal{X}_0 = (100, 100)$ and the rate constants as $\boldsymbol{c} = (0.5, 0.0025, 0.3)$. With these settings we generate the observed data $\boldsymbol{\mathcal{Y}}$, for the discrete time points $t = [0 : 5 : 50]$, by simulating the MJP. For inference we place the following priors on the rates: $c_1 \sim \text{Beta}(2, 1)$ $c_2 \times 100 \sim \text{Half}\,\mathcal{N}(0, 1)$ and $c_3 \sim \text{Beta}(1, 2)$. We carry out variational inference with the LNA approximation as described in section 4.2. We also ran the ABC-SMC algorithm with the SSA simulator. For ABC-SMC we used 1000 particles with an adaptive tolerance schedule and a multivariate perturbation kernel. We set the number of repeated simulations $M = 10$ (see Equation (11)).

Summaries of the posterior marginals are furnished in Table 1. Table 2 compares the run-times. Similar to the previous examples we noticed a significant speed-up for **VI** in comparison to the ABC-SMC. The density and model-fit plots are again shown in Appendix C.

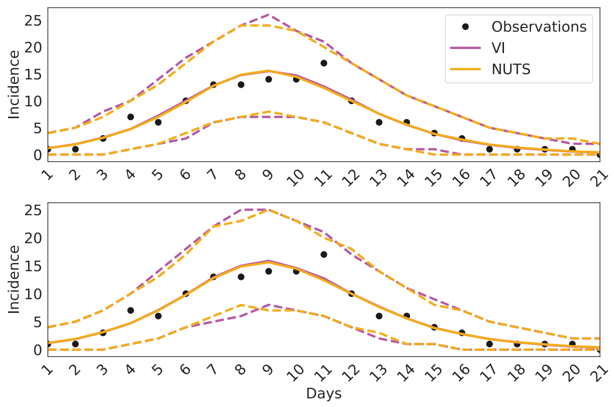


Figure 3: Model fit plots for inference using **Forward (top)** and **Adjoint sensitivty (bottom)** with the SIR model on the Tristan da Cunha common colds data (Toni et al., 2009). Mean (solid lines) and the 95% credible intervals (dashed lines) of the posterior predictive distributions are plotted.

7.4 Discussion

Throughout we did not notice an underestimation of the posterior variances, a pathology of mean-field variational approximation. In fact, the joint distributions (see Appendix C) from variational inference were almost indistinguishable to their MCMC/ABC counterparts. This means that the full-rank approximation is able to capture the correlation structure of the parameters well.

Table 2: Run-times in **seconds**, rounded-off to nearest integer, for all the different inference methods run on the three model examples. These were run on a 3.6 GHz machine with 16 GB memory.

METHODS	SIR	LOTKA-VOLTERRA	PROTEIN
VI-FOR	**171**	**7904**	**446**
VI-ADJ	**398**	**5272**	**1012**
NUTS-FOR	344	–	3707
NUTS-ADJ	827	–	10303
ABS-SMC	−−	13072	−−

Interestingly, the adjoint based **VI** appears to be faster than the forward one for the LNA example. The LNA likelihood, unlike the first two examples, requires same number of calls to the ODE solver for both sensitivity method, since the system needs to be solved between each pair of measurements. The adjoint VJP calculation involves fewer ($2u + v = 15$ vs $u \times v = 18$) coupled ODEs in comparison to forward VJP. Thus, adjoint is faster in this case.

The protein model is the trickiest due to identifiability problems. For this model, through a few pilot runs, we found that up to 10,000 iterations are required. We identified convergence by monitoring the ELBO, and also comparing the marginal, joint distributions with the corresponding estimates obtained through MCMC. We then simply fixed this number for the other models, although fewer iterations for the SIR, Lotka-Volterra models would have sufficed.

8 Conclusion

We presented a variational formulation of Bayesian inference of ODE parameters. We applied this formulation to stochastic models using the LNA approximation. Furthermore, we proposed a holistic framework for extending automatic differentiation to ODEs. We benchmarked the benefits of this approach by evaluating our proposed approach on three biological mechanistic models where we observed a significant speed-up of the parameter estimation process, without any significant lack in the quality of estimates in comparison to sampling based (MCMC/ABC) alternative inference methods.

Acknowledgements

We like to thank the anonymous reviewers for their helpful comments and suggestions. We also like to thank Gary Mirams, Martin Robinson and the PINTS team for discussion and feedback on some aspects of this work. SG was supported by the Medical Research Council (Unit programme number MC UU 00002/11).

References

Roy M Anderson, B Anderson, and Robert M May. *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992.

Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1): 5595–5637, 2017.

Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

Frank Dondelinger, Maurizio Filippone, Simon Rogers, and Dirk Husmeier. ODE parameter inference using adaptive gradient matching with Gaussian processes. In *Proc. 16th Int'l Conf. on Artificial Intelligence and Statistics*, pages 216–228, 2013.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

Paul Fearnhead, Vasilieos Giagos, and Chris Sherlock. Inference for reaction networks using the linear noise approximation. *Biometrics*, 70(2):457–466, 2014.

Vasileios Giagos. *Inference for auto-regulatory genetic networks using diffusion process approximations*. PhD thesis, Lancaster University, 2010.

Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

Daniel T Gillespie. The chemical langevin equation. *The Journal of Chemical Physics*, 113(1):297–306, 2000.

Andrew Golightly and Colin S Gillespie. Simulation of stochastic kinetic models. In *In Silico Systems Biology*, pages 169–187. Springer, 2013.

Andrew Golightly and Darren J Wilkinson. Bayesian inference for markov jump processes with informative observations. *Statistical applications in genetics and molecular biology*, 14(2):169–188, 2015.

Nico S Gorbach, Stefan Bauer, and Joachim M Buhmann. Scalable variational inference for dynamical systems. In *Advances in Neural Information Processing Systems*, pages 4806–4815, 2017.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry (Third Edition)*. North-Holland, 2007.

D P Kingma, Max Welling, et al. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 1, 2014.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017.

Benn Macdonald, Mu Niu, Simon Rogers, Maurizio Filippone, and Dirk Husmeier. Approximate parameter inference in systems biology using gradient matching: a comparative evaluation. *BioMedical Engineering OnLine*, 2016.

Ted Meeds, Geoffrey Roeder, Paul Grant, Andrew Phillips, and Neil Dalchau. Efficient amortised Bayesian inference for hierarchical and nonlinear dynamical systems. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4445–4455. PMLR, 2019.

Omiros Papaspiliopoulos and Gareth Roberts. Non-centered parameterisations for hierarchical models and data augmentation. *Bayesian Statistics*, 7:307–326, 01 2003.

Linda Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM journal on scientific and statistical computing*, 4(1):136–148, 1983.

Christopher Rackauckas, Yingbo Ma, Vaibhav Dixit, Xingjian Guo, Mike Innes, Jarrett Revels, Joakim Nyberg, and Vijay Ivaturi. A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions. *arXiv preprint arXiv:1812.01892*, 2018.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Ma-*

chine Learning, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286. PMLR, 2014.

Simo Särkkä and Arno Solin. *Applied stochastic differential equations*. Cambridge University Press, 2019.

T. Tieleman and G Hinton. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning, 4*, 2016.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1971–1979. PMLR, 2014.

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

Vladislav Vyshemirsky and Mark A Girolami. Bayesian ranking of biochemical system models. *Bioinformatics*, 24(6):833–839, 2008.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Philippe Wenk, Alkis Gotovos, Stefan Bauer, Nico S. Gorbach, Andreas Krause, and Joachim M. Buhmann. Fast gaussian process based gradient matching for parameter identification in systems of nonlinear odes. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1351–1360, 2019.

Darren J Wilkinson. *Stochastic modelling for systems biology*. CRC press, 2018.