Timonen, Juho; Siccha, Nikolas; Bales, Ben; Lähdesmäki, Harri; Vehtari, Aki

An importance sampling approach for reliable and efficient inference in Bayesian ordinary differential equation models

Published in:
Stat

DOI:
10.1002/sta4.614

Published: 18/09/2023

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
https://doi.org/10.1002/sta4.614
An importance sampling approach for reliable and efficient inference in Bayesian ordinary differential equation models

Juho Timonen | Nikolas Siccha | Ben Bales | Harri Lähdesmäki | Aki Vehtari

1Department of Computer Science, Aalto University, Espoo, 02150, Finland
2Earth Institute, University of Columbia, New York, New York, 10025, USA

Correspondence
Juho Timonen, Department of Computer Science, Aalto University, Espoo 02150, Finland.
Email: juho.timonen@aalto.fi

Funding information
Academy of Finland, Grant/Award Numbers: 340721, 311584, 328401

1 | INTRODUCTION

Implicitly defined quantities that depend on unknown parameters introduce challenges when they are involved in statistical models. Examples of such quantities are solutions to parameterized algebraic equations, optimization problems, integrals, or ordinary differential equations (ODEs). They generally do not have a closed form given the parameters, and to evaluate the model likelihood, they have to be approximated using numerical methods (Süli & Mayers, 2003). In the Bayesian context, model inference is commonly done by sampling the posterior distribution of the parameters using Markov chain Monte Carlo (MCMC) (Brooks et al., 2011) techniques. The theory behind MCMC assumes that the model likelihood can be computed exactly, but this is not the case if numerical approximations are required. Some numerical routines can estimate their own error, but as the true error is not known and its magnitude varies in different parts of the parameter space, it is difficult to predict how it affects the MCMC posterior draws. As this bias has not had a lot of attention in the literature, unaware users can blindly use default configurations of the numerical methods implemented in software packages.

Numerically solving an ODE system is a computationally intensive task and often dominates the cost of one unnormalized posterior density evaluation in Bayesian ODE models. Computational requirements are amplified by the fact that MCMC inference typically requires a large number of these ODE solutions, and this number can be several orders of magnitude larger than the posterior effective sample size (ESS) due to the need to warm-up the sampler and high auto-correlation of intermediate draws. The ratio of ESS to the number of unnormalized posterior density evaluations can be drastically increased by using gradient-based MCMC methods such as Hamiltonian Monte Carlo (HMC) (Duane et al., 1987), and using the...
gradient information is essential in the case of high-dimensional parameter spaces or complex posterior geometries in order to achieve good sampling performance. In modern statistical software such as Stan (Stan Development Team, 2022), PyMC (Salvatier et al., 2016), and Turing.jl (Ge et al., 2018), gradients are computed using automatic differentiation (AD) (Bell & Burke, 2008; Baydin et al., 2018). This poses a challenge for ODE models, for which computing the gradient is computationally significantly more demanding and subtle than the plain likelihood evaluation.

Classic numerical integrators for solving ODE systems are iterative methods that discretize the integral, and their accuracy and stability depend on the discretization step size (Hairer et al., 1993; Griffiths & Higham, 2010; Süli & Mayers, 2003). In theory, the error can be made arbitrarily small by using a step size that approaches zero, but this is not possible in practice due to limited computational resources and floating point arithmetic. Smaller step size means more evaluations of the ODE right-hand side (RHS) function, which means more computation. Selecting the step size therefore involves balancing between a reasonable computation time and good accuracy of the approximation.

Adaptive integrators remove the burden of selecting the step size, as they can tune it automatically. These methods estimate their own local error and adapt their step size so that given tolerances are satisfied. However, this does not give any guarantees about validity of the related statistical inference results. Moreover, requiring more accuracy typically causes the solver to adapt to smaller step size values, which leads to more computation. The problem of step size selection has thus been replaced by the problem of selecting the tolerances. Regardless, adaptive solvers are the most commonly used methods in statistical software and have been included in various probabilistic programming and machine learning frameworks that implement gradient-based MCMC samplers with AD.

To our best knowledge, there exist no generally applicable frameworks for validating the reliability of an approximate numerical method that is needed for posterior density evaluations in MCMC inference. For ODE solvers, one approach is to gradually use lower and lower step sizes (or stricter and stricter tolerances for adaptive solvers) during inference, until posterior estimates do not change appreciably anymore. However, repeating MCMC sampling like this quickly becomes computationally very expensive. Capistran et al. (2016) recognized the model that uses a numerical approximation as a different model than the true model with the exact ODE solution. In the special case of a Gaussian observation model and a certain type of fixed-step solver with step size $h$, they showed that the Bayes factor of the two models approaches one with the same rate as the numerical solution approaches the true solution, as $h \rightarrow 0$. Relying on this, they estimated the marginal likelihood of the true model based on first estimating it for approximate models, with different $h$, and then extrapolating to $h = 0$ using linear regression. However, the required several marginal likelihood approximations are difficult to perform in high dimensions, and Capistran et al. (2016) only demonstrated their method in one-dimensional parameter spaces.

Probabilistic numerical methods (Hennig et al., 2015) view the numerical problems probabilistically and can give uncertainty estimates for the solution. Teymur et al. (2021) used Gaussian process regression to estimate a distribution for the exact solution given a series of approximations of different accuracy. These methods are designed for performing a fixed numerical problem probabilistically, and it is not clear how to use them in Bayesian inference of models that contain numerical problems with unknown parameters. For the probabilistic parameter inference problem of ODE models, there exist strategies that completely avoid numerically integrating the ODE (see, e.g., Barber & Wang, 2014, and references therein). However, such approaches lack convergent numerical methods and thus are not asymptotically approximating the true ODE solutions.

We present an efficient, reliable, and generally applicable strategy for MCMC inference of models that require numerically approximating parameter-dependent quantities. It uses importance weights (Hastings, 1970) to correct the biases that result from using numerical approximations. These importance weights are cheap to compute compared with the cost of MCMC sampling, and we can diagnose their success using Pareto smoothed importance sampling (Vehtari et al., 2021). The proposed method is straightforward to implement in probabilistic programming languages, as it does not require modifications to standard MCMC algorithms or classic numerical solvers. We demonstrate its benefits in ODE model inference, using both adaptive and nonadaptive ODE solvers.

## METHODS

### ODEs

#### Initial value problems

ODE models are routinely used in various fields of science to model dynamic phenomena. A $D$-dimensional ODE system with state variables $y(t) \in \mathbb{R}^D$ is defined as

$$\frac{dy(t)}{dt} = f_{\psi}(y(t), t),$$

where the right-hand side function (RHS) $f_{\psi} : \mathbb{R}^D \times \mathbb{R} \rightarrow \mathbb{R}^D$ has parameters $\psi$. These parameters can be a subset of all parameters of a Bayesian model, in which ODE systems usually appear in the form of initial value problems (IVPs). This means that an initial value $y_0 := y(t_0)$ is explicitly
defined (either a known value or a model parameter), and evaluating the likelihood of the data requires solving \( y(t) \) at several time points \( t > t_0 \).

The solution is implicitly defined by Equation (1) and the initial value, and can be written using the integral formula:

\[
y(t) = y^{(0)} + \int_{t_0}^{t} f_{\nu}(y(t'), t') dt',
\]

which according to the Picard–Lindelöf theorem has a unique solution assuming some smoothness conditions\(^1\) for \( f_{\nu} \) (Hairer et al., 1993). However, the integral rarely has an explicit closed form and has to be approximated numerically.

### 2.1.2 | ODE solvers

A myriad of different methods exist for numerically approximating the integral in Equation (2). We use \( y^M(t) \) to denote the solution given by a numerical method \( M \). A general strategy used by method \( M(h) \) with fixed step size \( h > 0 \) is to first compute \( y^{M(h)}(t_j) \) on a grid \( t_j = t_0 + jh, j \in \{0,1,2,...\} \). This can be done by setting \( y^{M(h)}(t_0) = y^{(0)} \) and iteratively computing \( y^{M(h)}(t_{j+1}) \) for \( j \geq 0 \) using some update rule. After this, some interpolation method can be used if the solution is required at a time point \( t \) which is not on the grid (Hairer et al., 1993). Numerical methods are generally required to be convergent, meaning that the global error \( ||y^{M(h)}(t) - y(t)|| \) must approach zero as \( h \to 0 \). A method is called convergent of order \( p \) if this happens at rate \( O(h^p) \) (Hairer et al., 1993).

Smaller step sizes \( h \) will give a more accurate solution but require more iterations and therefore more computation. In practice, one would like to set the step size small enough to achieve good precision but large enough to avoid unnecessary computation. Adaptive step size methods try to automatically adapt the step size by estimating their own error. As the global error is difficult to estimate, software implementations are usually based on estimating the local truncation error, that is, the error induced by a single step (Griffiths & Higham, 2010). The step size is adapted so that user-supplied absolute \( (c_{abs}) \) and relative \( (c_{rel}) \) tolerances in the estimated local truncation error are satisfied. While these methods remove the burden of selecting \( h \) from the user, the user must still supply the absolute and relative tolerances. These tolerances have virtually the same trade-off as the step size selection itself; lower tolerances give better accuracy, but require smaller step sizes and therefore more computation.

ODE solvers are generally either explicit or implicit. For explicit solvers, the next state is computed explicitly based on the current state. Implicit solvers tend to perform significantly better for stiff problems (Hairer & Wanner, 1996), but the downside is that they require numerically solving a system of algebraic equations on each step. This has to be done using for example Newton iteration, which has its own stopping criterion that affects the result. The ODE solvers used in our experiments are described in more detail in Appendix S1.

### 2.1.3 | Sensitivity analysis

Gradient-based MCMC requires computing the gradient of the unnormalized posterior density. In modern probabilistic programming frameworks, gradients are computed using AD (Baydin et al., 2018; Margossian, 2019). This involves building a differentiable computation graph for the target density, where all operations on inputs are recorded. However, computing the sensitivities efficiently and reliably in AD frameworks is not straightforward when iterative numerical solvers are involved (Bell & Burke, 2008; Margossian, 2019; Rackauckas et al., 2021).

There are three main ways to integrate ODE solves into the computation graph built for AD. The direct method treats the ODE solve similarly as any other sequence of operations and directly records these operations into the computation graph. We use this method for all nonadaptive solvers in our experiments. The other two methods rely on continuous sensitivity analysis (Rackauckas et al., 2021). This approach is based on the fact that applying the chain rule of differentiation to the ODE system (Equation 1) gives

\[
\frac{d}{dt} \frac{dy(t)}{dp} = \frac{d}{dp} f_{\nu}(y(t), t) \frac{dy(t)}{dp} + \frac{\partial f_{\nu}(y(t), t)}{\partial y} \frac{\partial y(t)}{dp},
\]

from which we get an additional ODE system

\[
\frac{d}{dt} S(t) = \frac{\partial f_{\nu}(y(t), t)}{\partial y} S(t) + \frac{\partial f_{\nu}(y(t), t)}{\partial y} \frac{\partial y(t)}{dp},
\]

with DP dimensions. Forward continuous sensitivity analysis solves this extended system simultaneously with the original system using the same adaptive numerical ODE solver. This can be implemented so that also the extended system needs to satisfy the given tolerances, but more
sophisticated rules are used with implicit solvers that require also Newton iteration (Feehery et al., 1997). In our experiments, we use forward continuous sensitivity analysis for all adaptive solvers. Adjoint continuous sensitivity analysis (Margossian, 2019; Rackauckas et al., 2021) first solves only the original ODE system forward in time, and uses the obtained solution to solve a different additional system backward in time to get the sensitivities. In this method the additional system has only dimension $D$, so it theoretically scales better with respect to the number of parameters. However, this method is even harder to configure, and we leave it for future work.

### 2.2 Bayesian models with numerical approximations

We consider MCMC inference of models that define a posterior $p(\theta|D) \propto p(D|\theta)p(\theta)$, where $p(D|\theta)$ is the likelihood of data $D$ given parameters $\theta$, and $p(\theta)$ is the prior. The goal of inference is commonly the computation of expectations of the form

$$\mathbb{E}_{p(\theta|D)}[\varphi(\theta)] = \int \varphi(\theta)p(\theta|D)d\theta,$$

which can be for instance model predictions or parameter estimates, determined by the function $\varphi$. When MCMC is used to obtain posterior draws $\theta_s$, $s = 1, ..., S$, the integral can be estimated as

$$\mathbb{E}_{p(\theta|D)}[\varphi(\theta)] \approx \frac{1}{S} \sum_{s=1}^{S} \varphi(\theta_s).$$

We focus on models whose unnormalized posterior density depends on $N$ intermediate quantities $Y_n(\theta)$, collected in the list $Y(\theta) = \{Y_n(\theta)\}_{n=1}^{N}$. We define

$$P(Y(\theta), \theta) := p(D|\theta)p(\theta)$$

to denote the unnormalized density. Quantities $Y(\theta)$ can be defined implicitly through equations that have no closed form solution or for other reasons have to be numerically approximated as $Y_n(\theta) \approx Y_n(\theta)$ for each $n = 1, ..., N$, where $M$ denotes the approximation method. Since $Y(\theta)$ can only be approximated, $P(Y(\theta), \theta)$ cannot be evaluated exactly and therefore it is not possible to use MCMC to sample from $p(\theta|D)$ exactly. Instead, MCMC will sample from some distribution $p^M(\theta|D) \propto P(Y^M(\theta), \theta)$. It is therefore crucial to develop methods that can correct this bias and inform users if the approximation method $M$ is so inaccurate that it renders the correction impossible, and re-running MCMC with a more accurate method is needed.

Various numerical methods $M$ have some control parameters that can be used to tune their accuracy. Examples of such methods are ODE solvers $M = M(h)$, where $h > 0$ is the step size. In ODE solvers, the implicitly defined quantities are $Y_n(\theta) = y_\varphi(t_n)$, where $\varphi$ is a subset of all parameters $\theta$ and $y_\varphi(t_n)$ is the solution to an ODE initial value problem with parameters $\varphi$, at time $t_n$. The corresponding numerical approximate solution with method $M$ we denote $Y^M_\varphi(t_n) \approx y_\varphi(t_n)$.

### 2.3 Importance sampling approach

In importance sampling, draws $\theta_s$, $s = 1, ..., S$ are obtained from another distribution $q(\theta)$, and the expectation (Equation 5) can be estimated as

$$\mathbb{E}_{q(\theta|D)}[\varphi(\theta)] \approx \frac{\sum_{s=1}^{S} \varphi(\theta_s)}{\sum_{s=1}^{S} t_s},$$

where $t_s = \frac{p(\theta|D)}{q(\theta)}$ are called importance ratios/weights (Hastings, 1970). Our approach is to use MCMC importance sampling with $q(\theta) = P(Y^M(\theta), \theta)$, where $M$ is the approximation method used during MCMC. The importance ratios are

$$t_s^M = \frac{P(Y^M(\theta_s), \theta)}{P(Y^M(\theta), \theta)}$$

but as we cannot evaluate $P(Y(\theta_s), \theta)$, exact importance sampling is not possible. Instead, we use ratios
\[ r_{s}^{M^{*}} = \frac{p(Y_{T}^{M^{*}}(\theta), \theta)}{p(Y_{T}^{M}(\theta), \theta)} \]  

(10)

where \( M^{*} \) is a more accurate method than \( M \). We require that \( M^{*} \) is a convergent numerical method, meaning that for any fixed \( \theta \),

\[ Y_{n}^{M^{*}}(\theta) \rightarrow Y_{n}(\theta) \]  

(11)

for all \( n = 1, \ldots, N \), as \( M^{*} \) is made more and more accurate. This means that

\[ p(Y_{T}^{M^{*}}(\theta), \theta) \rightarrow p(Y(\theta), \theta) \]  

(12)

at each point \( \theta \). Consequently, the ratios \( r_{s}^{M^{*}} \) converge to the exact ratios \( r_{s}^{M} \), and posterior estimates (Equation 8) converge, too. Our approach therefore is to incrementally increase the accuracy of \( M^{*} \) until the maximum absolute error

\[ \text{MAE}^{M^{*}} = \max_{s} \left\{ \max_{\theta} \left\{ \| Y_{n}^{M^{*}}(\theta) - Y_{n}^{M}(\theta) \|_{\text{max}} \right\} \right\} \]  

(13)

has converged.\(^2\) In general high-dimensional parameter spaces of ODE models, this step can be done with negligible computational effort compared to the initial MCMC sampling. Furthermore, this analysis can be conveniently done simultaneously as we assess whether MCMC needs to be run again with a more accurate method \( M \).

2.4 Pareto smoothing and diagnostics

Importance sampling requires that the distribution \( q(\theta) \) is sufficiently similar to the target distribution \( p(\theta|\mathcal{D}) \), so that the nominator and denominator in Equation (8) would have finite variance (Geweke, 1989; 2005; Koopman et al., 2009). Pareto smoothed importance sampling (PSIS) (Vehtari et al., 2021) modifies the raw ratios so that the estimator of the expectation has finite variance and asymptotic normality in a wider range of problems (Vehtari et al., 2021). However, any modifications to the ratios cannot correct for a \( q(\theta) \) that is too far from \( p(\theta|\mathcal{D}) \), which in our case means that the method \( M \) used during MCMC sampling has to be sufficiently accurate. Importantly, PSIS provides a diagnostic that we can use to assess whether this is the case.

In PSIS, a generalized Pareto distribution (GPD) is fitted to match the tail of the empirical distribution of the ratios \( r_{s}^{M^{*}} \). The probability density function of the GPD is

\[ p_{\text{GPD}}(x|u, k, \sigma) = \begin{cases} \frac{1}{\sigma} \left(1 + \frac{k(x-u)}{\sigma} \right)^{-\frac{1}{k}}, & k \neq 0, \\ \frac{1}{\sigma} \exp \left( \frac{x-u}{\sigma} \right), & k = 0 \end{cases} \]  

(14)

where \( u \in \mathbb{R} \) is a location parameter, \( k \in \mathbb{R} \) is a shape parameter, and \( \sigma > 0 \) is a scale parameter. The cutoff value \( u = \bar{u} \) is first determined as explained in Vehtari et al. (2021), and \( k, \sigma \) are then fitted to the empirical distribution of the tail \( r_{s}^{M^{*}} > \bar{u} \). The latter part can be done using the method by Zhang and Stephens (2009), and overall fitting the GPD parameters is computationally very cheap.

The estimated shape parameter \( k \) can be used as a diagnostic to determine if the importance sampling estimate (Equation 8) is reliable (Vehtari et al., 2021). Values \( k < 0.7 \) indicate small errors with high probability and good convergence rate with increasing sample size (Vehtari et al., 2021). Moreover, as we increase the accuracy of \( M^{*} \), we can study the convergence of \( k \) as an additional safeguard metric to assess whether the distribution of importance ratios, and therefore any posterior estimates, has converged.

2.5 The proposed workflow

The proposed algorithm for MCMC inference of models that require numerical approximations can be summarized in the following steps.
1. Select a reasonable approximation method $M$.
2. Sample parameter draws $\theta_i, s = 1, \ldots, S$, using MCMC with $M$ as the approximation method.
3. Compute $\text{MAE}^M$ (Equation 13) and importance weights $r_s^M$ (Equation 10) using approximation method $M^*$. Fit $\hat{k}$ as explained in Section 2.4.
4. Increase the accuracy of $M^*$ and repeat Step 3 until $\text{MAE}^{M^*}$ and $\hat{k}$ converge. If $\hat{k}$ converges to a value larger than 0.7, increase the accuracy of $M$ and go back to Step 2.
5. Compute any posterior estimates using Equation (8), with $r_i$ being the values to which $r_s^M$ finally converged.

The algorithm can be used in two ways, depending on how Step 1 is done. First, it can be used to validate the reliability and correct the errors of a given method $M$, which can be for example a software default. Second, a smart initial selection of $M$ can provide substantial speed gains compared to often rather conservatively set software defaults, while still maintaining reliability of the inferences.

In the latter case, we generally recommend selecting $M$ initially so that sampling is as fast as possible. For example, for nonadaptive ODE solvers $M = M(h)$, one can first try using the largest sensible step size $h$ that does not result in immediate failure. Sometimes the solver returns infinite or NaN values or values that are inconsistent with the observation model and MCMC cannot proceed, but in this case, the sampler will fail fast and not much time is wasted. Selecting a good $M$ is more difficult in the case of adaptive solvers $M = M(\epsilon)$ whose adaptation rules are discontinuous and whose gradients also need to be approximated. In their case, using too large tolerances $\epsilon$ can cause inaccurate gradients and ragged likelihood surfaces, which can cause MCMC to struggle and become slower, and these issues are slower to detect. We discuss this effect in more detail in Appendix S2.

### RESULTS

We developed an R-package called `odemodeling` for fitting Bayesian ODE models using Stan (Stan Development Team, 2022), and it is available at https://github.com/jtimonen/odemodeling. In addition to the adaptive built-in ODE solvers of Stan, it implements two explicit Runge–Kutta methods using a fixed but tuneable number of steps. Furthermore, it implements our workflow for determining reliability of ODE model inference and other convenience functions.

We demonstrate our proposed workflow using various ODE solvers and two ODE models, first with simulated and then with real data. In all experiments, we use the dynamic HMC algorithm (Betancourt, 2018; Stan Development Team, 2022) implemented in Stan for MCMC sampling. We use initial leap frog step size 0.1 and otherwise default configuration, unless otherwise mentioned. In each experiment, we run four independent MCMC chains for 4000 iterations and the first 2000 iterations are discarded as warmup. Code for reproducing the experiments is available at https://github.com/jtimonen/numapprox-is. Full experiment details are in Appendix S3.

#### 3.1 Target-mediated drug disposition model

In the first experiment, the model is a target mediated drug disposition (TMDD) model (Aston et al., 2011; Mager & Jusko, 2001). This is a common pharmacokinetic–pharmacodynamic model for reversible binding of drug/ligand ($y_1$) with receptor ($y_2$), where a receptor–ligand complex ($y_3$) is formed. For solving the ODE system, we use the backward differentiation formulae method (BDF; see Appendix S1). We denote the BDF solver with absolute and relative tolerances $\epsilon_{\text{abs}} = \epsilon_{\text{rel}} = \epsilon$ by $BDF(\epsilon)$. The ODE solution $y^{BDF(\epsilon_{10^{-15}})}(t)$ using the simulation parameters (Appendix S3) and the simulated noisy data used in this experiment are visualized in Figure 1a.

To study the effect of tolerances, we first run MCMC sampling using $M = BDF(\epsilon)$ different $\epsilon$ values. Example ODE solutions from the posterior using $\epsilon = 0.02$ are shown in Figure 1b, and corresponding solutions with $\epsilon = 10^{-12}$ in Figure 1c. The tolerance value $\epsilon = 0.02$ is very high and the solutions look unstable compared to $\epsilon = 10^{-12}$.

The timing results in Figure 2 (black line) show that for sufficiently small values (roughly $\epsilon < 0.02$) of $\epsilon$, the MCMC runtime increases consistently as $\epsilon$ is decreased. For example, with $\epsilon = 10^{-10}$, which is the default in Stan, MCMC sampling takes around $10^3$ s. On the other hand, we can sample from the same distribution by first MCMC sampling with $M = BDF(0.02)$ and then performing importance sampling with $M^* = BDF(10^{-10})$ in just $10^2$ s. This time comes almost solely from the MCMC sampling, and we could change $M^*$ to even more accurate, with virtually no extra cost. We note that plain runtime of MCMC is not always a good measure of sampling efficiency. In this case, however, it is informative as we always use the same MCMC algorithm, the bulk and tail estimated effective sample sizes (Vehtari et al., 2021) are very similar in all cases, and convergence diagnostics are satisfactory (see Table S1).

Figure 1b,c demonstrated that the BDF(0.02) gives suspicious ODE solutions, indicating that we cannot trust it as such and need our reliability workflow. However, quantities in Figure 3 validate that importance sampling can be trusted for all tested tolerances of $M$, because the estimate of the shape parameter of the GPD $\hat{k}$ stabilizes at a value smaller than 0.5 as $M^*$ is made more and more accurate. Performing this reliability
check takes an insignificantly small amount of time compared to MCMC sampling. Additional metrics are provided in Figure S1. For example, the estimated relative efficiency of the importance sampling phase is close to 1 (Figure S1d), meaning that we do not significantly lose efficiency due to it.

Figure 4 illustrates the distribution of the importance ratios and fitting the GPD in different cases. We see that the tail of the ratios becomes less and less thick as tolerances of \( M \) are decreased.

The runtime starts increasing if the tolerances are made too large (approx. \( \epsilon > 0.03 \)). This means that we cannot start our workflow with \( \epsilon \) being arbitrarily large. To understand the reason for this, we have recorded some HMC NUTS metrics in Table S2. The runtime is explained almost completely by the number of ODE RHS evaluations with AD (Figure 2b), and too large tolerances cause the leap frog step size to adapt to smaller values, meaning that more leap frog steps and therefore more RHS evaluations with AD are needed. Potential reasons for this can be that the discontinuity of the ODE solver step size adaptation rule becomes more evident with large tolerances, which makes the likelihood surface more discontinuous and complicates sampling. The problems are amplified by the fact that the gradient error grows when tolerances are increased (See Section 2.1.3 and Appendix S2).

### 3.2 | Lotka–Volterra model

In the second experiment, we study a model of predator–prey dynamics between Canadian lynx and snowshoe hare. See Appendix S3 for details about the model and data. In this experiment, we first use the adaptive RK45 ODE solver (Appendix S1), and use RK45(\( \epsilon \)) to denote it with tolerances \( \epsilon_{\text{rel}} = \epsilon_{\text{abs}} = \epsilon \).
We first run MCMC sampling using $M = \text{RK4}5(\epsilon)$ with different $\epsilon$ values. The results in Figure 5a (black line) show that for small values (approx $\epsilon < 10^{-3}$) of $\epsilon$, the runtime increases as $\epsilon$ is decreased. For example, with $\epsilon = 10^{-6}$, which is the default in Stan, MCMC sampling takes around $10^{2.15} \approx 140$ s. On the other hand, we can sample from the same distribution by first MCMC sampling with $M = \text{BDF}(\epsilon)$ and then additionally computing the importance weights needed to correct resulting posterior estimates as if the draws were from $M = \text{BDF}(\epsilon)$, where $\epsilon$ is on the x-axis. The same MCMC algorithm and same number of chains and draws is used in all MCMC sampling. (b) The MCMC runtime is almost completely explained by the number of times we need to perform AD for the ODE RHS function. The red and blue dots each correspond to one MCMC chain, x-axis showing the number of needed RHS calls with AD and y-axis the chain runtime.

**FIGURE 2** Runtime comparison in the TMDD experiment. (a) The black line corresponds to running MCMC directly with $M = \text{BDF}(\epsilon)$, where $\epsilon$ is on the x-axis. The four colored lines correspond to first running MCMC with $M = \text{BDF}(0.05)$, $\text{BDF}(0.04)$, $\text{BDF}(0.03)$, or $\text{BDF}(0.02)$, and additionally computing the importance weights needed to correct resulting posterior estimates as if the draws were from $M = \text{BDF}(\epsilon)$, where $\epsilon$ is on the x-axis. The same MCMC algorithm and same number of chains and draws is used in all MCMC sampling. (b) The MCMC runtime is almost completely explained by the number of times we need to perform AD for the ODE RHS function. The red and blue dots each correspond to one MCMC chain, x-axis showing the number of needed RHS calls with AD and y-axis the chain runtime.

**FIGURE 3** Convergence of some metrics as the reference method $M^*$ is made more and more accurate in the TMDD experiment. The (a) row shows the maximum absolute error (MAE) between posterior ODE solutions $y^M$ and $y^{M^*}$ (Equation 13), as a function of the tolerance $\epsilon^*$ of $M^*$. The (b) row shows for Pareto smoothed importance sampling the $\hat{k}$ diagnostic. The columns correspond to different choices of the method $M$ used during MCMC.

We first run MCMC sampling using $M = \text{RK4}5(\epsilon)$ with different $\epsilon$ values. The results in Figure 5a (black line) show that for small values (approx $\epsilon < 10^{-3}$) of $\epsilon$, the runtime increases as $\epsilon$ is decreased. For example, with $\epsilon = 10^{-6}$, which is the default in Stan, MCMC sampling takes around $10^{2.15} \approx 140$ s. On the other hand, we can sample from the same distribution by first MCMC sampling with $M = \text{RK4}5(0.001)$ and then performing importance sampling with $M^* = \text{RK4}5(10^{-6})$ in just $10^{1.7} \approx 50$ s.

We performed MCMC inference and importance sampling using also the RK4 and midpoint methods for solving the ODE (Appendix S1). We use notation RK4($K$) and MIDPOINT($K$) where $K$ is the number of steps taken between subsequent output time points and thus implicitly determines the step size $h$. The results in Figure 5b (purple and orange lines) show that for both methods, the runtime increases consistently with $K$.

Furthermore, we find that using $M = \text{MIDPOINT}(3)$ is faster than any tested tolerance values for RK45 and $M = \text{RK4}(2)$ is a bit slower than RK45.
with $\epsilon = 0.001$ or $\epsilon = 10^{-4}$. Smaller $K$ for either MIDPOINT or RK4 in this case caused the solver to break at initial parameter values and not be able to produce MCMC draws.

Quantities in Figure 6 validate that our importance sampling workflow can be trusted. The $\text{MAE}^{M^{*}}$ and other metrics converge as $M^{*}$ is made more accurate, and $\hat{k}$ values converge to $<0.5$ in all cases with close to 1 relative efficiency. The convergence is especially smooth for the nonadaptive solvers, as $K^{*}$ is increased for $M^{*}$. Additional metrics are provided in Figure S2.
DISCUSSION

We have described a useful approach for efficient and reliable Bayesian inference of general ODE models. We have demonstrated our workflow in ODE problems, but it is generally applicable also to various other types of models that require approximate numerical solvers. We have shown that by using a less accurate numerical method during MCMC and importance sampling with increasingly more accurate versions of the method can be of an order of magnitude faster than plain MCMC sampling with a software default method. Moreover, after MCMC sampling, users can easily and without further computational effort check what the posterior of the numerical solutions would look like with any given other solver.

Efficiently selecting the initial method $M$ to be used in our workflow still has some challenges if $M$ has to belong to the class of commonly used adaptive ODE solvers. Inference with these methods can become slower if $M$ is made too inaccurate. These methods have not originally been designed to be used in Bayesian inference but have been widely adopted into even gradient-based MCMC and optimization software despite their discontinuous adaptation rules, and the fact that their local error control itself is not enough for reliable statistical inference. In our experience, tolerances on the order of $10^{-4}$ or $10^{-3}$ are a good starting point for the initial method $M$ in many ODE applications.

Conveniently, our framework enables also rapid testing of different types of new and old numerical solvers, which do not need to have error control. The only requirement is that they are convergent numerical methods whose accuracy can be controlled. We have shown a large potential and obtained promising results using explicit non-adaptive solvers whose gradient is computed using direct AD of the solver formula.

ACKNOWLEDGMENTS

We would like to acknowledge the computational resources provided by Aalto Science-IT, Finland. This work was supported by the Academy of Finland Flagship program: Finnish Center for Artificial Intelligence, and the Academy of Finland projects 340721, 311584, and 328401.

CONFLICT OF INTEREST STATEMENT

The authors declare no potential conflict of interests.

ORCID

Juho Timonen
https://orcid.org/0000-0003-2341-6765

Aki Vehtari
https://orcid.org/0000-0003-2164-9469

FIGURE 6 Convergence of some metrics during our workflow in the Lotka–Volterra experiment, as the reference method $M^*$ is made more and more accurate by increasing either the tolerance $\epsilon^*$ or number of steps $K^*$. The columns correspond to different choices of the initial ODE solver $M$ used during MCMC.
ENDNOTES

1 We only consider problems where these conditions are satisfied.

2 We use $|x|_{\text{max}} = \max_i |x|$ to denote the maximum norm for vectors.

3 For example negative values when positive ones are required.

4 This refers to the step size of the MCMC sampler, and should not be confused with the step size of the ODE solver used to solve the modeled system. The lower initial value of 0.1 was observed to reduce some warmup problems compared to default 1.0.

REFERENCES


SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.