A Flexible and Efficient Framework for Probabilistic Numerical Simulation and Inference

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Nathanael Bosch
aus Stuttgart

Tübingen
2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	26.02.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Philipp Hennig
2. Berichterstatter:	Dr. Jon Cockayne

Abstract

Probabilistic numerics has emerged as a promising approach for quantifying and propagating numerical error in computational simulations. Given a numerical problem, probabilistic numerical methods compute not only a point estimate for the solution, but they provide a full posterior distribution over the quantity of interest. This distribution quantifies the numerical approximation error of the method in a structured manner.

For ordinary differential equations (ODEs), probabilistic numerical methods based on Gaussian filtering and smoothing have been introduced as a particularly promising class of methods. These so-called *ODE filters* scale linearly in the number of time steps, they satisfy well-known stability properties, and they converge to the true solution with polynomial rates, all while also providing a posterior distribution over the ODE solution. But despite these properties, ODE filters have not yet reached the same level of computational efficiency and versatility as their non-probabilistic counterparts.

In this thesis, we address these limitations and establish ODE filtering as a flexible, efficient, and feature-rich framework for probabilistic numerical simulation and inference. To achieve this, we examine the different building blocks of these solvers, including the underlying prior model, information operator, and approximate inference scheme, and we show how they can be adjusted to further improve the performance and utility of these methods.

Our contributions broadly fall into four categories. First, we consider particular classes of differential equations and we leverage their structure to develop specialized solvers with improved stability and efficiency for semi-linear ODEs, higher-order ODEs, Hamiltonian dynamical systems, and differential-algebraic equations. Second, we leverage structure of the underlying state-space model and develop efficient ODE filters for high-dimensional problems. Third, we investigate the underlying inference algorithm and its time discretization, and we present a step-size adaptation scheme as well as a *parallel-in-time* ODE filter, both of which can provide significant speed-ups. Fourth, we apply ODE filters to parameter inference problems and we develop new numerical-error-aware algorithms for robust parameter inference in ODEs. In addition, to make these methods accessible to a broader audience and to facilitate their application in practice, we develop ProbNumDiffEq.jl, an efficient, accessible, and feature-rich open-source software library for probabilistic numerical ODE solvers.

In summary, this thesis improves the computational efficiency, stability, and versatility of ODE filters and reduces the gap between classic and probabilistic numerical methods. Our contributions thereby establish ODE filtering as a flexible and efficient framework for probabilistic numerical simulation and inference and pave the way for the broader adoption of probabilistic numerics in scientific computing and engineering applications.

Zusammenfassung

Die *probabilistische Numerik* ist ein vielversprechender Ansatz für die Quantifizierung und Propagierung numerischer Fehler in Rechensimulationen. Probabilistisch-numerische Methoden berechnen nicht nur einen Punktschätzer für die Lösung eines numerischen Problems, sondern sie liefern auch eine A-posteriori-Wahrscheinlichkeitsverteilung über die Lösung, welche den numerischen Approximationsfehler des Verfahrens auf eine strukturierte Weise beschreibt.

Für gewöhnliche Differentialgleichungen (ODEs; engl. "ordinary differential equations") wurden probabilistisch-numerische Methoden auf Basis von Gauß'schen Filtern und Smoothern als besonders vielversprechende Klasse von Verfahren eingeführt. Diese sogenannten *ODE-Filter* skalieren linear in der Anzahl der Zeitschritte, sie erfüllen bestimmte Stabilitätseigenschaften, sie konvergieren mit polynomiellen Raten, und sie liefern eine A-posteriori-Wahrscheinlichkeitsverteilung über die ODE-Lösung. Doch trotz dieser Eigenschaften haben ODE-Filter noch nicht das gleiche Maß an Recheneffizienz und Vielseitigkeit erreicht wie etablierte, nicht-probabilistische Verfahren.

In dieser Dissertation adressieren wir diese Einschränkungen und etablieren ODE-Filter als ein flexibles, effizientes und funktionsreiches Framework für probabilistisch-numerische Simulation und Inferenz. Hierzu untersuchen wir die verschiedenen Bausteine dieser Verfahren—die A-priori-Verteilung, den Informationsoperator und das Inferenzverfahren—und wir zeigen, wie diese angepasst werden können, um die Leistungsfähigkeit dieser Methoden zu verbessern.

Unsere Arbeiten lassen sich grob in vier Kategorien einteilen. Erstens betrachten wir spezielle Klassen von Differentialgleichungen, wie semi-lineare ODEs, ODEs höherer Ordnung, Hamilton'sche dynamische Systeme und differential-algebraische Gleichungen, und entwickeln spezialisierte Verfahren für deren Lösung. Zweitens nutzen wir die Struktur des zugrundeliegenden Zustandsraummodells und entwickeln effiziente ODE-Filter für hochdimensionale Probleme. Drittens untersuchen wir das zugrundeliegende Inferenzverfahren und dessen Zeitdiskretisierung, und entwickeln ein Schema zur automatischen Anpassung der Schrittweite sowie einen zeitlich-parallelierten ODE-Filter, welche beide signifikante Performanzsteigerungen ermöglichen. Viertens wenden wir ODE-Filter auf Parameterinferenzprobleme an und entwickeln neue, numerische Fehler berücksichtigende Algorithmen für robuste Parameterinferenz in ODEs. Außerdem entwickeln wir ProbNumDiffEq.jl, eine effiziente, zugängliche und funktionsreiche Open-Source-Softwarebibliothek für probabilistisch-numerische ODE-Löser, um diese Methoden einem breiteren Publikum zugänglich zu machen und ihre Anwendung in der Praxis zu erleichtern.

Zusammengefasst verbessert die vorliegende Dissertation die Recheneffizienz, Stabilität und Vielseitigkeit von ODE-Filtern und bringt somit probabilistisch-numerische Methoden näher

an die Leistungsfähigkeit klassischer numerischer Verfahren heran. Unsere Erkenntnisse etablieren ODE-Filter als ein flexibles und effizientes Framework für probabilistisch-numerische Simulation und Inferenz und ebnen damit den Weg für eine breitere Nutzung probabilistisch-numerischer Methoden in wissenschaftlichen und technischen Anwendungen.

Acknowledgments

I would like to express my deepest gratitude to Philipp Hennig, whose guidance, support, and mentorship have been invaluable throughout this journey. You have been an inspiration to me both as a scientist and as a person and I am grateful for everything I have learned from you. Thank you for the many opportunities you have given me and for your unwavering support. And perhaps most importantly, thank you for always fostering such a friendly and collaborative atmosphere in our research group.

I am thankful to Jon Cockayne for investing his time and expertise in evaluating this thesis. I also thank Jakob Macke and Georg Martius for forming my examination committee, and for their helpful feedback as members of my thesis advisory committee.

I am immensely grateful to all my colleagues from the MoML group for creating this positive and stimulating atmosphere filled with productivity, mutual support, and good humor. Thank you for the many discussions and the productive collaborations, but also for the long coffee and lunch breaks, the barbecues and retreats, and all the fun moments outside of work.

Particular thanks go to Filip, Nico, and Jonathan. I've learned a lot from all of you—about ODE filters, the process of doing science, and beyond—and I am grateful for the many shared moments, in the office and around the world. Working with you really has been a pleasure and this thesis would have been very different without you.

I am thankful for the inspiring collaborations with Adrien Corenflos, Fatemeh Yaghoobi, Simo Särkkä, Amon Lahr, Melanie Zeilinger, Jonas Beck, Michael Deistler, Kyra Kadhim, Jakob Macke, and Philipp Berens. I am particularly grateful to Simo Särkkä for the opportunity to visit Aalto University and spend time with his research group.

I thank Franziska Weiler for her administrative support and for always being there to help with any questions or issues. I am also grateful to the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for their support.

I would like to thank all the proofreaders who have helped improve the quality of this thesis. Your feedback has been invaluable.

I am deeply grateful to my friends and family, especially to my mother, Roswitha Bosch, for their unwavering support and encouragement throughout this journey.

Finally, I thank Lisa Mörchen for her support, her presence, and for being such a wonderful person.

Nathanael Bosch
Tübingen, 31 März, 2024

Table of Contents

Acronyms

CUDA	Compute Unified Device Architecture; a parallel computing platform used for general-purpose computing on Nvidia GPUs
DAE	Differential-algebraic equation
EK0 / EK1	Probabilistic numerical ODE solver based on extended Kalman filtering and smoothing, with zeroth-order / first-order Taylor linearization of the vector field
EKF	Extended Kalman filter
EKS	Extended Kalman smoother / Extended Rauch–Tung–Striebel smoother
GP	Gaussian process
GPU	Graphical processing unit
GSSM	Gaussian state-space model
IEKS	Iterated extended Kalman smoother
IOUP	Integrated Ornstein–Uhlenbeck process
IVP	Initial value problem
IWP	Integrated Wiener process
KF	Kalman filter
LGSSM	Linear Gaussian state-space model
LL	Log-likelihood
LTI	Linear time-invariant
LTI-SDE	Linear time-invariant stochastic differential equation
MAP	Maximum a-posteriori
MCMC	Markov chain Monte Carlo
MLE	Maximum-likelihood estimate
MSE	Mean square error
NLGSSM	Nonlinear Gaussian state-space model
NLL	Negative log-likelihood
ODE	Ordinary differential equation
PDE	Partial differential equation
RB	Rosenbrock
RMSE	Root mean square error
SDE	Stochastic differential equation
SSM	State-space model

List of Publications

This dissertation is based on the following published papers:

I Nathanael Bosch, Philipp Hennig, and Filip Tronarp. **Calibrated Adaptive Probabilistic ODE Solvers**. *International Conference on Artificial Intelligence and Statistics* (AISTATS), 2021.

Individual contributions: The original idea of the article is due to Philipp Hennig and Filip Tronarp and was expanded on by Nathanael Bosch. Nathanael Bosch produced the code, designed and evaluated the experiments, and wrote the article. Filip Tronarp and Philipp Hennig provided valuable feedback along the way.

II Nathanael Bosch, Filip Tronarp, and Philipp Hennig. **Pick-and-Mix Information Operators for Probabilistic ODE Solvers**. *International Conference on Artificial Intelligence and Statistics* (AISTATS), 2022.

Individual contributions: The original idea is due to Nathanael Bosch and Filip Tronarp. Nathanael Bosch produced the code, designed and evaluated the experiments, and wrote the article, with valuable feedback from Filip Tronarp and Philipp Hennig.

III Nicholas Krämer[*], Nathanael Bosch[*], Jonathan Schmidt[*], and Philipp Hennig. **Probabilistic ODE Solutions in Millions of Dimensions**. *International Conference on Machine Learning* (ICML), 2022.

Individual contributions: The original idea is due to Nicholas Krämer, Nathanael Bosch, and Jonathan Schmidt. Nicholas Krämer, Nathanael Bosch, and Jonathan Schmidt jointly developed the method, wrote the code, designed and evaluated the experiments, and wrote the article. Philipp Hennig provided valuable feedback.

IV Filip Tronarp[*], Nathanael Bosch[*], Philipp Hennig. **Fenrir: Physics-Enhanced Regression for Initial Value Problems**. *International Conference on Machine Learning* (ICML), 2022.

Individual contributions: The original idea is due to Filip Tronarp. Nathanael Bosch implemented the method and designed and evaluated the experiments, with feedback from Filip Tronarp and Philipp Hennig. Filip Tronarp and Nathanael Bosch wrote the article, with valuable feedback from Philipp Hennig.

[*]Equal contribution.

V Nathanael Bosch, Philipp Hennig, and Filip Tronarp. **Probabilistic Exponential Integrators**. *Conference on Neural Information Processing Systems* (NeurIPS), 2023.

Individual contributions: The original idea is due to Filip Tronarp. Nathanael Bosch implemented the method and designed and evaluated the experiments, with feedback from Filip Tronarp and Philipp Hennig. Nathanael Bosch wrote the article, with valuable feedback from Filip Tronarp and Philipp Hennig.

VI Jonas Beck, Nathanael Bosch, Michael Deistler, Kyra L. Kadhim, Jakob H. Macke, Philipp Hennig, and Philipp Berens. **Diffusion Tempering Improves Parameter Estimation with Probabilistic Integrators for Ordinary Differential Equations**. *International Conference on Machine Learning* (ICML), 2024.

Individual contributions: The original project idea is due to Philipp Hennig and Philipp Berens. Jonas Beck and Nathanael Bosch developed the algorithm, with help from Michael Deistler. Jonas Beck implemented the method with Nathanael Bosch. The experiments were designed and evaluated by Jonas Beck, with feedback from Nathanael Bosch and Michael Deistler. Jonas Beck wrote the majority of the article, Nathanael Bosch wrote the background section. All authors were involved in editing and reviewing the manuscript. Philipp Berens, Philipp Hennig, and Jakob Macke provided valuable feedback along the way.

VII Nathanael Bosch, Adrien Corenflos, Fatemeh Yaghoobi, Filip Tronarp, Philipp Hennig, and Simo Särkkä. **Parallel-in-time Probabilistic Numerical ODE Solvers**. *Journal of Machine Learning Research* (JMLR), 2024.

Individual contributions: The original idea for this article came independently from Simo Särkkä and from discussions between Filip Tronarp and Nathanael Bosch. The joint project was initiated and coordinated by Simo Särkkä and Philipp Hennig. The methodology was developed by Nathanael Bosch in collaboration with Adrien Corenflos, Filip Tronarp, Philipp Hennig, and Simo Särkkä. The implementation is primarily due to Nathanael Bosch, with help from Adrien Corenflos. The experimental evaluation was done by Nathanael Bosch with support from Filip Tronarp and Philipp Hennig. The first version of the article was written by Nathanael Bosch, after which all authors reviewed the manuscript.

VIII Nathanael Bosch. **ProbNumDiffEq.jl: Probabilistic Numerical Solvers for Ordinary Differential Equations in Julia**. *Journal of Open Source Software* (JOSS), 2024.

The publications are contained, in full, in the appendix (Part IV).

Introduction

Dynamical systems are ubiquitous throughout science and engineering [118]. In domains such as climate and weather forecasting [30, 91], epidemiology [51, 37], computational chemistry [59], physics and mechanics [2, 18, 26], finance [62, 84, 81], and even within machine learning [24, 132], their accurate and efficient simulation often plays a central role. Mathematically, dynamical systems are typically described with differential equations, such as for example ordinary differential equations of the form

$$\dot{y}(t) = f(y(t), t), \qquad t \in [0, T], \tag{0.1a}$$

with so-called vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial value $y(0) = y_0 \in \mathbb{R}^d$ [44]. *Simulating* a dynamical system amounts to *solving* the corresponding differential equation, that is, finding the function $y(t) : [0, T] \to \mathbb{R}^d$ that satisfies the above. But while differential equations in general have by now been studied for centuries [21], most equations of interest cannot be solved analytically [44]. Therefore, we rely on numerical methods to approximate their solution.

The development of sophisticated numerical methods for solving ordinary differential equations (ODEs) predates even the first electronic computers, with Runge [104] and Kutta [72] developing a class of numerical methods which are now known as "Runge–Kutta" methods, and Bashforth et al. [3] developing another class of now fundamental methods called "multistep" methods. These classical methods have since then been refined, extended, and optimized over many decades [20]. Nowadays, a plethora of efficient, accurate, and stable Runge–Kutta and multistep methods exists, with specialized solvers for many different use cases and problems of interest [44, 45], accessible via modern, feature-rich, and highly optimized software libraries [97, 126, 116]. But despite the sophistication of these classical methods, they come with a notable limitation: they typically compute only a single point-estimate of the solution and do not quantify their inherent numerical approximation error. This omission of numerical error can be significant in practice [53]. It is often up to the practitioners to decide if a specific simulation is trustworthy or if it should be re-computed with a higher computational budget. And when numerical simulations are used in downstream tasks, they are often treated as if they were free of any numerical error, which can lead to misguided conclusions or inefficient allocation of computational resources [89].

Probabilistic numerics has emerged as a promising framework to address exactly this limitation of classic numerical methods [49, 48, 88]. By rephrasing both the numerical problem and its solution algorithm in the language of probability theory, probabilistic numerical methods compute a full posterior distribution over the solution of interest, which offers not only a point estimate but also quantifies the numerical approximation error in a structured way. This idea

can be applied not only to differential equations but also to other numerical problems such as linear algebra, quadrature, and optimization, and by now a variety of probabilistic numerical methods has been developed; refer to the book by Hennig et al. [49] for an overview.

In the context of ordinary differential equations there are two main classes of probabilistic numerical methods: perturbative solvers and filtering-based solvers. Perturbative solvers, as presented by Chkrebtii et al. [25], aim to represent the uncertainty that arises from the numerical discretization by a set of samples. Conrad et al. [28] proposed to compute these samples by adding random perturbations to classical numerical solvers at each time step. This sparked a line of research that has since then developed a variety of perturbative solvers, including perturbation-based multistep methods [120], implicit methods [119], and geometric integrators [1]. Filtering-based solvers on the other hand aim to compute the full posterior distribution over the ODE solution by formulating the ODE problem as Bayesian filtering and smoothing [112, 65, 122]. These so-called "ODE filters" will be the focus of this thesis.

To account for the numerical error which, in the case of ODEs, arises due to discretization of the temporal domain, ODE filters consider inference problems of the form

$$y(t) \sim \mathcal{GP}(m, k), \tag{0.2a}$$

$$y(0) = y_0, \tag{0.2b}$$

$$\dot{y}(t_n) = f(y(t_n), t_n), \qquad n = 1, \ldots, N. \tag{0.2c}$$

The term $y(t) \sim \mathcal{GP}(m, k)$ specifies a (Gaussian process) *prior* over the ODE solution $y(t)$ which, as is common in Bayesian inference, describes our belief over the quantity of interest before doing any measurements or computation [101]. The other two terms represent the *likelihood model* and correspond to the differential equation problem of interest, as given in eq. (0.1a). Crucially, the likelihood model not only encodes the differential equation problem, but also explicitly includes the temporal discretization of the ODE in the inference problem. This enables the probabilistic quantification of the numerical error in ODE filtering: Applying Bayes' rule yields the posterior distribution

$$p\big(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^{N}\big), \tag{0.3}$$

which describes our belief over the ODE solution $y(t)$ given the initial value problem of interest, the temporal discretization, and our prior. In the following, we call this posterior distribution and approximations thereof the *probabilistic numerical ODE solution*. The main question that remains is then how to compute this posterior distribution accurately and efficiently. In ODE filtering, this is done with Bayesian filtering and smoothing.

ODE filters have been a topic of interest in the probabilistic numerics community for several years, and by now they have been shown to satisfy a number of properties that are desirable for numerical ODE solvers. Similarly to classic numerical simulators, ODE filters converge polynomially to the true ODE solution as their step size decreases [65, 123], they satisfy certain well-established numerical stability guarantees [122], and their computational complexity

scales linearly in the number of time points [112]. But despite these promising properties, ODE filters have not yet been applied to the same breadth of problems as classic numerical solvers, as they have not reached the same level of computational efficiency, versatility, and general usability as their non-probabilistic counterparts.

In this thesis, we aim to bridge this gap and enable the application of ODE filters to more complex and computationally demanding applications. We focus on improving their computational efficiency in both theory and practice and we extend their utility to a broader range of ODE problems and downstream tasks. In addition to the theoretical development of new algorithms and methods, we also provide an efficient, feature-rich implementation of these methods in the software library ProbNumDiffEq.jl to make ODE filters more accessible to practitioners and researchers.

Outline

This thesis provides a comprehensive introduction to filtering-based probabilistic numerical ODE solvers and presents a range of new methods and algorithms that improve their efficiency, stability, flexibility, and utility. It is structured into three main parts.

Part I provides the necessary background for developing filtering-based probabilistic numerical ODE solvers. Chapter 1 introduces ordinary differential equations and shows how these are traditionally solved with non-probabilistic numerical methods. Chapter 2 then introduces sequential Bayesian state estimation problems and presents various inference algorithms for these, such as notably the extended Kalman filter and smoother—these will provide the algorithmic backbone of our probabilistic numerical ODE solvers. Then, Chapter 3 broadens the estimation problem to inference over functions and introduces Gauss–Markov processes as a particularly efficient class of models. These will be the probabilistic models that we formulate probabilistic numerical ODE solvers in.

Part II builds on these foundations and introduces the probabilistic numerical ODE solvers that are the main focus of this thesis, together with our main contributions. Chapter 4 introduces the ODE filtering framework, provides an overview over its main components, and discusses some properties of these solvers compared to traditional non-probabilistic ODE solvers. The subsequent chapters then individually present summaries of the included publications: Chapter 5 discusses the cubic computational scaling of ODE filters and presents two state-space factorizations that result in linear scaling in the ODE dimension, and thus enable the application of ODE filters to high-dimensional problems. Chapter 6 investigates the calibration of ODE filters, presents multiple strategies to improve their uncertainty quantification, and proposes an adaptive step-size selection scheme. Chapter 7 extends the ODE filtering framework to higher-order ODEs, conserved quantities, and differential-algebraic equations, by adjusting the likelihood model of the solver. Chapter 8 introduces probabilistic exponential integrators, a new class of ODE filters for stiff semi-linear ODEs which operate by including the linear dynamics in the prior model. Chapter 9 presents a parallel-in-time formulation of ODE filters, which can significantly speed up the computation by leveraging modern compu-

tational hardware such as GPUs. Finally, Chapter 10 develops a new probabilistic numerical algorithm for ODE parameter inference in ODEs based on the ODE filtering framework.

Part III concludes the thesis and provides an outlook on future research directions.

Part IV contains the full publications on which this thesis is based.

Part I

Preliminaries

1 Ordinary Differential Equations and Non-Probabilistic Numerical Simulators

This chapter provides a very brief introduction to ordinary differential equations and to non-probabilistic numerical methods for their solution.

1.1 Ordinary differential equations

Ordinary differential equations (ODEs) describe a function $y : [0, T] \to \mathbb{R}^d$ in terms of an initial condition $y(0) = y_0 \in \mathbb{R}^d$ and a differential equation of the form

$$\dot{y}(t) = f(y(t), t), \tag{1.1}$$

where $f : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ is a given *vector field*, and $[0, T] \subset \mathbb{R}$ is some time domain. The *solution* of an ODE is a function $y(t)$ that satisfies both the differential equation and the initial condition. This function $y(t)$ can also be described in integral form, as

$$y(t) = y_0 + \int_0^t f(y(s), s)\mathrm{d}s. \tag{1.2}$$

The *solution operator* of an ODE is the mapping that maps the initial condition to the solution at time t, i.e., $\Phi_f : (y_0, t) \mapsto y(t)$; this is also known as the *flow map* of the ODE. For some simple ODEs, the solution can be computed in closed form:

Example 1.1 (**Logistic ODE**) The logistic ODE describes a population $y(t)$ that grows proportionally to its size, but is limited by some carrying capacity. It is given by the differential equation

$$\dot{y}(t) = \alpha y(t)\left(1 - \frac{y(t)}{\beta}\right), \tag{1.3}$$

where $\alpha > 0$ is a growth rate parameter and $\beta > 0$ is the carrying capacity, and with an initial condition $y(0) = y_0$. For this specific ODE, the solution is known to be given by the logistic function

$$y(t) = \frac{\beta}{1 + \left(\frac{\beta - y_0}{y_0}\right)e^{-\alpha t}}, \tag{1.4}$$

which can be easily verified by differentiating eq. (1.4). See the left plot in Figure 1.1 for a visualization of the logistic ODE and its analytical solution.

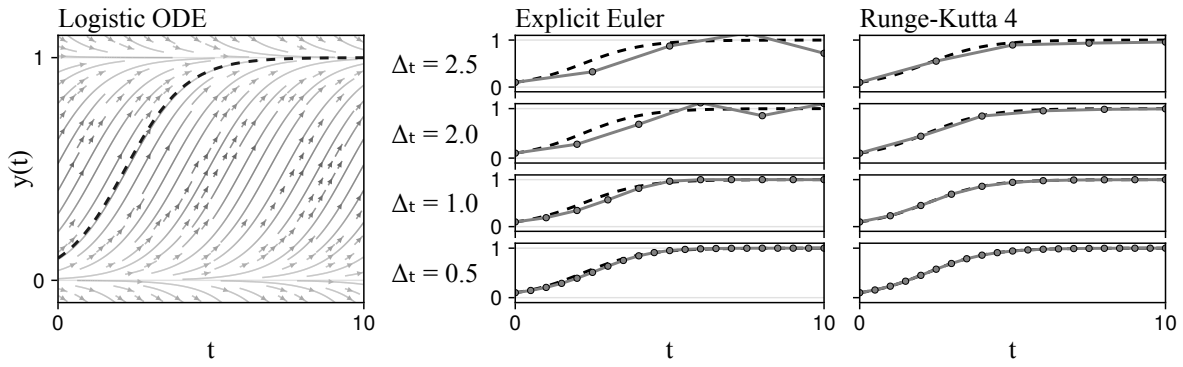


Figure 1.1: **Logistic ODE and numerical solutions.** Left: The logistic ODE from Example 1.1, with the vector field shown in gray in the background and the true analytical solution shown in black. Center and right: Numerical solutions computed with the explicit Euler method and the fourth-order Runge–Kutta method, respectively, for various step sizes. We see that both methods become more accurate as the step size is decreased, and the Runge–Kutta method is generally more accurate than the explicit Euler method for the same step size.

1.2 Non-probabilistic numerical ODE solvers

Non-probabilistic numerical solvers for ODEs aim to compute an approximation $\hat{y}(t)$ to the ODE solution $y(t)$ such that $\hat{y}(t) \approx y(t)$. One very popular class of such solvers are so-called *Runge–Kutta* methods [104, 72, 44]. These methods are based on the idea that the integral in eq. (1.2) can be decomposed into a sum of integrals over small time intervals, and thus the solution satisfies the recursive relation

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(y(s), s)\mathrm{d}s. \tag{1.5}$$

Therefore, the main question in Runge–Kutta methods is how to approximate these integrals.

Arguably the simplest approximation arises from approximating the integrated by its value at the left end of the interval, i.e., $f(y(s), s) \approx f(y(t_n), t_n)$ for $s \in [t_n, t_{n+1}]$. This leads to the *explicit Euler method* [34].

Algorithm 1.1 (Explicit Euler method) Given an ODE initial value problem (f, y_0) and a discrete time grid $\{t_n\}_{n=0}^N \subset [0, T]$, perform the following steps:

1. Set the initial condition $\hat{y}_0 = y_0$.
2. For $n = 1, \ldots, N$, compute the update step

$$\hat{y}_n = \hat{y}_{n-1} + (t_n - t_{n-1}) f(\hat{y}_{n-1}, t_{n-1}). \tag{1.6}$$

Return the discrete approximate solution $\{\hat{y}_n\}_{n=0}^N$.

The explicit Euler method is very simple to implement and the individual update step is computationally very cheap. But, it is also very inaccurate (it has a low order of convergence) and it can be numerically unstable for certain ODEs; see Figure 1.1. Both of these issues can be addressed with more sophisticated Runge–Kutta methods.

Runge–Kutta methods approximate the integral by a weighted sum of function evaluations at different points in the interval. Following Hairer et al. [44, Definition 7.1], the general form of a Runge–Kutta method is

$$y(t_{n+1}) = y(t_n) + \Delta t_n \sum_{i=1}^{s} b_i k_i, \tag{1.7a}$$

$$k_i = f\left(y(t_n) + \Delta t_n \sum_{j=1}^{s} a_{ij} k_j, t_n + c_i h\right), \tag{1.7b}$$

where $\Delta t_n = t_{n+1} - t_n$ is the step size, k_i are intermediate function evaluations, and a_{ij}, b_i, and c_i are the coefficients that fully define the specific Runge–Kutta method.

Algorithm 1.2 (Runge–Kutta method) Given an ODE initial value problem (f, y_0) and a discrete time grid $\{t_n\}_{n=0}^{N} \subset [0, T]$, as well as the Runge–Kutta coefficients $\{a_{ij}\}_{i,j=1}^{s}$, $\{b_i\}_{i=1}^{s}$, and $\{c_i\}_{i=1}^{s}$, perform the following steps:

1. Set the initial condition $\hat{y}_0 = y_0$.
2. For $n = 1, \ldots, N$, recursively compute $\hat{y}_n$ with the RK update step from eq. (1.7). (This may or may not involve solving an implicit system of equations).

Return the discrete approximate solution $\{\hat{y}_n\}_{n=0}^{N}$.

Since the coefficients are the defining feature of a Runge–Kutta method, they are usually shown in a compact representation known as the *Butcher tableau* [19, 44]

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \ldots & b_s \end{array} \tag{1.8}$$

Example 1.2 (Fourth-Order Runge–Kutta method) The fourth-order Runge–Kutta method by Runge [104] can be written compactly with the Butcher tableau [44]

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \tag{1.9}$$

Figure 1.1 also visualizes numerical solutions to the logistic ODE using this fourth order Runge–Kutta method, and we observe that for the same step size, the fourth order Runge–Kutta method produces more accurate solutions than the explicit Euler method.

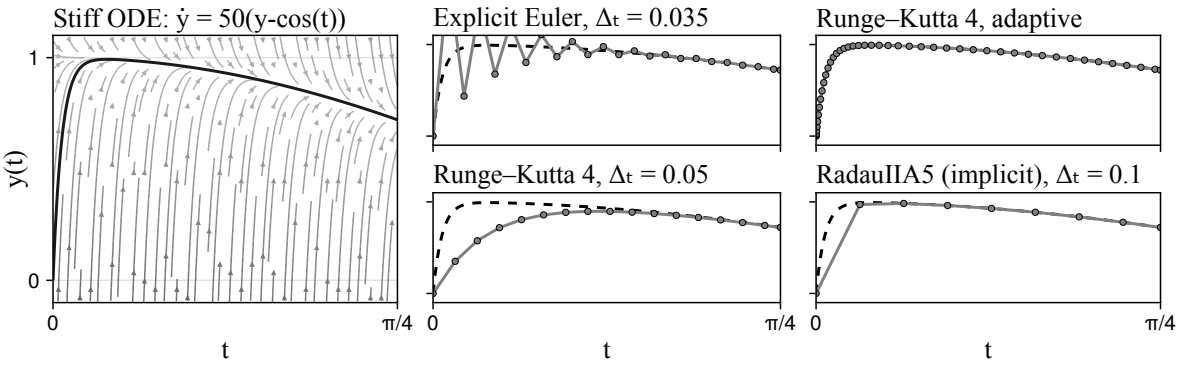


Figure 1.2: **Performance of different Runge–Kutta methods on a stiff differential equation.** Left: Vector field and accurate reference solution of the stiff ODE. Center and right: Various numerical solutions for different solvers, step sizes, and step-size selection mechanisms. We see that for larger step sizes, both the Euler and Runge–Kutta 4 solvers start diverging. By using adaptive step-size selection, the Runge–Kutta 4 method is able to compute an accurate solution, but at the cost of large numbers of steps. The implicit Runge–Kutta method `RadauIIA5` produces a stable solution even for large numbers of steps.

1.3 Properties of numerical ODE solvers

- **Convergence rates:** Numerical ODE solvers should be able to approximate the true ODE solution arbitrarily well, given enough computational resources and patience. Formally, this means that the numerical error should vanish as the step size decreases to zero. The presented Runge–Kutta methods are known to satisfy polynomial convergence guarantees: the *local* error (the error made in a single step) decreases with the step size Δt as $\mathcal{O}\big(\Delta t^{p+1}\big)$, where p is the order of the method; and under some additional assumptions, the *global* error (the error made over the whole interval) decreases with rate $\mathcal{O}(\Delta t^p)$ [44, Section II.3].

- **Continuous solutions:** In addition to returning the solution at the discrete grid points, Runge–Kutta methods can also provide a continuous representation of the solution $y(t)$ on the whole integration interval by constructing a suitable interpolant, at low additional computational cost. This functionality is also known as *dense output* [44, Section II.6] and is implemented in many popular ODE solver packages, such as SciPy [126], DifferentialEquations.jl [97], and Matlab [116].

- **Adaptive step-size selection**: Instead of choosing the time discretization in advance, it can be preferable to select the size of each step automatically in such a way that the resulting numerical error is sufficiently small. This can often improve the accuracy of the solver, save computational resources, and speed-up the simulation process. Most commonly, step-size adaptation requires an estimate of the local error produced by the solver, which in the case of Runge–Kutta methods is typically computed by embedding a second, lower-accuracy solver into the method and then comparing the solution estimate of both methods. This can be done at low additional computational cost [44, Section II.4]. Then, step sizes can be adapted to the local error estimate by embedding the numerical solver step into a control algorithm, such as proportional-integral (PI) control [41].

- **Stability:** Certain differential equations, often referred to as *stiff* differential equations, can cause numerical solutions to oscillate or diverge; see Figure 1.2 for an example. Typically, this particularly affects *explicit* methods, which are methods where the step function can be computed without having to solve a linear or non-linear system of equations (such as those presented in this chapter). *Implicit* methods on the other hand are often more stable and less affected by these issues. Therefore, implicit ODE solvers are particularly well-suited to solve stiff ODEs and are often more efficient on these problems than explicit methods. At the same time, as implicit methods are also more expensive to compute per step, they can be inferior on non-stiff problems. Refer to Hairer et al. [45] for a more thorough overview on stiff ODEs and implicit solvers.
- **Computational complexity:** We can describe the computational complexity of Runge–Kutta methods with respect to three main factors: The number of steps N, the dimension of the ODE d, and the number of stages of the Runge–Kutta method s (which relates to the order of convergence discussed above). All the methods we discussed so far operate sequentially on the time grid, which leads to linear cost in N. At each step, *explicit* Runge–Kutta methods compute sums of s vectors of dimension d, whereas *implicit* Runge–Kutta methods need to solve an $sd \times sd$ linear system. This leads to a computational complexity of $\mathcal{O}(sdN)$ for explicit Runge–Kutta methods and $\mathcal{O}\big(s^3d^3N\big)$ for implicit Runge–Kutta methods (though often iterative solvers are used instead of solving the linear system exactly to make implicit methods more efficient).

1.4 Conclusion

In this chapter, we provided an overview of ordinary differential equations (ODEs) and classical non-probabilistic numerical methods for solving them. We discussed the formulation of ODEs and introduced the concept of the solution operator. While some simple ODEs, like the logistic equation, have closed-form solutions, most require numerical approaches. Among these, Runge–Kutta methods are particularly prominent, offering varying degrees of accuracy and stability. We examined both the explicit Euler method and more sophisticated Runge–Kutta methods, highlighting their computational properties, convergence rates, and the importance of adaptive step-size selection and stability considerations for stiff equations. This foundational understanding sets the stage for exploring probabilistic approaches to numerical ODE solving in later chapters.

2 Bayesian State Estimation

In this chapter, we develop algorithms to do efficient, numerically stable approximate inference in nonlinear Gaussian state-space models (NLGSSMs), of the form

$$x_0 \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{2.1a}$$

$$x_n = f_n(x_{n-1}) + w_n, \qquad w_n \sim \mathcal{N}(0, Q_n), \tag{2.1b}$$

$$z_n = h_n(x_n) + v_n, \qquad v_n \sim \mathcal{N}(0, R_n), \tag{2.1c}$$

where $x_n \in \mathbb{R}^{d_x}$ is the state of the system, $z_n \in \mathbb{R}^{d_z}$ is the observation, $f_n : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ and $h_n : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ are the transition and observation models, and $Q_n \in \mathbb{R}^{d_x \times d_x}$ and $R_n \in \mathbb{R}^{d_z \times d_z}$ are the transition and observation noise covariances, respectively, at time $n = 1, \ldots, N$. Figure 2.1 shows a graphical representation of this model. The goal in Bayesian state estimation is to compute a posterior distribution $p(x_{0:N} \mid z_{1:N})$ over the state at each point in time $x_{0:N} := (x_0, \ldots, x_n)$, given a sequence of observations $z_{1:N} := (z_1, \ldots, z_n)$. It turns out that this posterior, together with other related quantities of interest such as the marginal likelihood $p(z_{1:N})$, can be computed recursively and efficiently using so-called filtering and smoothing algorithms.

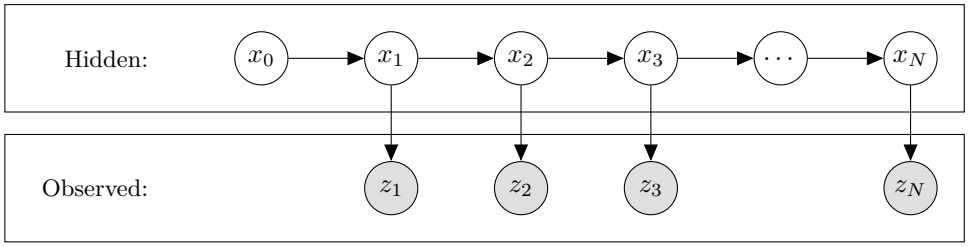


Figure 2.1: **Graphical representation of a Bayesian state-space model.**

In this chapter, we present recursive and numerically stable algorithms for approximate inference in nonlinear Gaussian state-space models. Section 2.1 introduces exact Gaussian inference in affine models, and Section 2.2 shows how to make this inference numerically stable. Section 2.3 then presents the recursive algorithm for exact Gaussian inference in linear Gaussian state-space models, commonly known as the Kalman filter and the Rauch–Tung–Striebel smoother. Then, Section 2.4 shows how to do approximate Gaussian inference in nonlinear models via linearization and develops two recursive algorithms for approximate Gaussian inference in nonlinear Gaussian state-space models. These will be the main workhorse for the probabilistic numerical methods we develop in later chapters.

2.1 Gaussian inference in affine models

Consider a Gaussian distribution $p(x) = \mathcal{N}(x; \mu_x, \Sigma_x)$ and a linear Gaussian conditional distribution $p(z \mid x) = \mathcal{N}(z; Ax + b, Q)$. To build Gaussian filtering and smoothing algorithms, we need to perform three basic operations on these distributions:

1. **Marginalization:** The marginal distribution of z is also Gaussian, of the form

$$p(z) = \int p(z \mid x)p(x)\mathrm{d}x = \mathcal{N}(z; \mu_z, \Sigma_z), \tag{2.2}$$

with mean and covariance

$$\mu_z = A\mu_x + b, \tag{2.3a}$$
$$\Sigma_z = A\Sigma_x A^\top + Q. \tag{2.3b}$$

This is also known as the *prediction* step in Kalman filtering.

2. **Inversion:** The conditional distribution of x given z is linear Gaussian, of the form

$$p(x \mid z) = \frac{p(x)p(z \mid x)}{p(z)} = \mathcal{N}(x; Gz + d, \Lambda), \tag{2.4}$$

with parameters

$$G = \Sigma_x A^\top \Sigma_z^{-1}, \tag{2.5a}$$
$$d = \mu_x - G\mu_z, \tag{2.5b}$$
$$\Lambda = \Sigma_x - G\Sigma_z G^\top. \tag{2.5c}$$

The covariance Λ can also be computed with equivalent, alternative formulas:

$$\Lambda = (I - GA)\Sigma_x, \tag{2.6a}$$
$$\text{or} \qquad \Lambda = (I - GA)\Sigma_x(I - GA)^\top + GQG^\top. \tag{2.6b}$$

The latter is also known as the *Joseph form* of the covariance update.

3. **Update on an observation:** Given a Gaussian prior $p(x)$ and linear Gaussian observation model $p(y \mid x)$ as above, the posterior distribution of x given an observation $z = \mathbf{z}$ can then be computed by first inverting the observation model, and then evaluating it at the observation, that is

$$p(x \mid z = \mathbf{z}) = \mathcal{N}(x; G\mathbf{z} + d, \Lambda), \tag{2.7}$$

with G, d, and Λ as in eq. (2.5). This is also known as the *update* step in Kalman filtering.

2.2 Numerically stable Gaussian inference in affine models

When using the formulas presented above in Section 2.1, the computed covariances can become non-positive semi-definite due to numerical errors. To avoid these issues we can formulate Gaussian inference in square-root form: Instead of working with covariance matrices $\Sigma \in \mathbb{R}^{n\times n}$, we work with their square-roots $\Sigma^{1/2} \in \mathbb{R}^{m\times n}$, defined to satisfy $\Sigma = (\Sigma^{1/2})^\top \Sigma^{1/2}$. For example, for positive definite matrices the Cholesky decomposition $\Sigma = LL^\top$ could provide such a square-root. But for our purposes we do not require the square-root to be unique and we can work with any square-root of the covariance matrix, including non-square ones (i.e., $m \neq n$), as long as they satisfy the relation $\Sigma = (\Sigma^{1/2})^\top \Sigma^{1/2}$.

For numerically stable Gaussian inference, the formulas that need to be re-derived in square-root form are the computation of the marginal covariance and the covariance of the inverted conditional distribution, namely

$$\Sigma_z = A\Sigma_x A^\top + Q, \tag{2.8a}$$

$$\Lambda = (I - GA)\Sigma_x (I - GA)^\top + GQG^\top. \tag{2.8b}$$

We wrote the latter in the Joseph form to show very clearly that two types of operations are performed on covariance matrices: multiplication with a matrix from the left and right, and addition of covariance matrices.

1. **Multiplication with a matrix from the left and right:** Given a positive semi-definite matrix $\Sigma \in \mathbb{R}^{n\times n}$ with square-root $\Sigma^{1/2} \in \mathbb{R}^{m\times n}$ and some matrix $A \in \mathbb{R}^{p\times n}$, the product $A\Sigma A^\top$ satisfies

$$A\Sigma A^\top = A\left(\Sigma^{1/2}\right)^\top \Sigma^{1/2} A^\top = \left(\Sigma^{1/2} A^\top\right)^\top \Sigma^{1/2} A^\top \tag{2.9}$$

   Thus, the product can be computed in square-root form with a single matrix-matrix multiplication $\Sigma^{1/2}A^\top$ without ever forming the full covariance matrix.

2. **Addition of covariance matrices in square-root form:** Given two positive semi-definite matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$, with square-roots $A^{1/2} \in \mathbb{R}^{m\times n}$ and $B^{1/2} \in \mathbb{R}^{p\times n}$, a square-root of their sum $C = A + B$ is given by concatenating the square-roots as

$$C^{1/2} = \begin{bmatrix} A^{1/2} \\ B^{1/2} \end{bmatrix}, \tag{2.10}$$

   since $(C^{1/2})^\top C^{1/2} = A + B = C$. Thus, concatenation of square-roots provides a square-root of the sum with minimal computational cost. But, this approach is impractical when performing many computations on square-root matrices as the resulting matrices would grow in size and become prohibitively large. To avoid this growth in size, we compress square-root matrices using a QR decomposition. Given a (possibly non-square) matrix square-root $A^{1/2} \in \mathbb{R}^{m\times n}$ of some square, positive semi-definite matrix $A \in \mathbb{R}^{n\times n}$, we can compute a square matrix square-root $R \in \mathbb{R}^{n\times n}$ of A by performing a thin QR

decomposition of $A^{1/2}$, that is $QR = A^{1/2}$, where $Q \in \mathbb{R}^{m\times n}$ is an orthogonal matrix and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. Then, R is also a square-root of A, since

$$A = (A^{1/2})^\top A^{1/2} = (QR)^\top (QR) = R^\top Q^\top QR = R^\top R. \tag{2.11}$$

The orthogonal matrix Q can be discarded after the computation. We denote this triangularization operation of computing a QR decomposition and returning only R as $\mathrm{tria} : \mathbb{R}^{m\times n} \to \mathbb{R}^{n\times n}$. Thus in summary, computing the sum $C = A + B$ of two positive semi-definite matrices $A, B \in \mathbb{R}^{n\times n}$ can be done in square-root form with $C^{1/2} = \mathrm{tria}\left(\begin{bmatrix} A^{1/2} \\ B^{1/2} \end{bmatrix}\right)$.

Coming back to the required covariance operations in eq. (2.8), we can now directly compute the marginal covariance $\Sigma_z = A\Sigma_x A^\top + Q$ and the inverted conditional distribution $\Lambda = (I - GA)\Sigma_x (I - GA)^\top + GQG^\top$ in square-root form, with

$$\Sigma_z^{1/2} = \mathrm{tria}\left(\begin{bmatrix} A\Sigma_x^{1/2} \\ Q^{1/2} \end{bmatrix}\right), \qquad \Lambda^{1/2} = \mathrm{tria}\left(\begin{bmatrix} \Sigma_x^{1/2}(I - GA)^\top \\ Q^{1/2}G^\top \end{bmatrix}\right). \tag{2.12}$$

Both terms can be easily verified by multiplying the above expressions with their transposes to recover eq. (2.8). The formulas from eq. (2.12) now enable us to perform marginalization, inversion, and update operations in a numerically stable manner.

Algorithm 2.1 (Square-root marginalization in Gaussian affine models) Given the parameters $(\mu_x, \Sigma_x^{1/2})$ and $(A, b, Q^{1/2})$ of distributions $p(x) = \mathcal{N}(x; \mu_x, \Sigma_x)$ and $p(z \mid x) = \mathcal{N}(z; Ax + b, Q)$, compute the mean and covariance of the Gaussian marginal distribution $p(z) = \mathcal{N}(z; \mu_z, \Sigma_z)$ as

$$\mu_z := A\mu_x + b, \qquad \Sigma_z^{1/2} := \mathrm{tria}\left(\begin{bmatrix} A\Sigma_x^{1/2} \\ Q^{1/2} \end{bmatrix}\right). \tag{2.13}$$

Return $\mu_z, \Sigma_z^{1/2}$.

Algorithm 2.2 (Square-root inversion in Gaussian affine models) Given the parameters $(\mu_x, \Sigma_x^{1/2})$, $(A, b, Q^{1/2})$, and $(\mu_z, \Sigma_z^{1/2})$ of distributions $p(x) = \mathcal{N}(x; \mu_x, \Sigma_x)$, $p(z \mid x) = \mathcal{N}(z; Ax + b, Q)$, and $p(z) = \mathcal{N}(z; \mu_z, \Sigma_z)$, compute the parameters of the conditional distribution $p(x \mid z) = \mathcal{N}(x; Gz + d, \Lambda)$ as

$$G := \Sigma_x A^\top \Sigma_z^{-1}, \tag{2.14a}$$

$$d := \mu_x - G\mu_z, \tag{2.14b}$$

$$\Lambda^{1/2} := \mathrm{tria}\left(\begin{bmatrix} \Sigma_x^{1/2}(I - GA)^\top \\ Q^{1/2}G^\top \end{bmatrix}\right). \tag{2.14c}$$

Return $G, d, \Lambda^{1/2}$.

Algorithm 2.3 (Square-root updating on data in Gaussian affine models) Given the parameters $(\mu_x, \Sigma_x^{1/2})$, $(A, b, Q^{1/2})$, and $(\mu_z, \Sigma_z^{1/2})$ of distributions $p(x) = \mathcal{N}(x; \mu_x, \Sigma_x)$, $p(z \mid x) = \mathcal{N}(z; Ax + b, Q)$, and $p(z) = \mathcal{N}(z; \mu_z, \Sigma_z)$, as well as an observation z, compute the parameters of the posterior distribution of x given the observation in two steps:

1. Compute the parameters $(G, d, \Lambda^{1/2})$ of the backward transition $p(x \mid z) = \mathcal{N}(x; Gz + d, \Lambda)$ with Algorithm 2.2.
2. Evaluate the backward transition at z to get the posterior mean and covariance:

$$\mu_x := Gz + d, \tag{2.15a}$$

$$\Sigma_x^{1/2} := \Lambda^{1/2}. \tag{2.15b}$$

Return $\mu_x, \Sigma_x^{1/2}$.

2.3 Sequential inference in linear Gaussian state-space models

Now that we established the Gaussian inference formulas for affine models, we can derive a general inference algorithm for linear Gaussian state-space models, which corresponds to the Kalman filter [61] and the Rauch–Tung–Striebel smoother [102]. Consider a linear Gaussian state-space model (LGSSM) of the form

$$x_0 \sim \mathcal{N}(x_0; \mu_0, \Sigma_0), \tag{2.16a}$$

$$x_n \mid x_{n-1} \sim \mathcal{N}(x_n; A_n x_{n-1} + b_n, Q_n), \tag{2.16b}$$

$$z_n \mid x_n \sim \mathcal{N}(z_n; H_n x_n + c_n, R_n), \tag{2.16c}$$

where $x_n \in \mathbb{R}^{d_x}$ is the state of the system, $z_n \in \mathbb{R}^{d_z}$ is the observation, $A_n \in \mathbb{R}^{d_x \times d_x}$, $b_n \in \mathbb{R}^{d_x}$, $Q_n \in \mathbb{R}^{d_x \times d_x}$ are the transition model parameters, and $H_n \in \mathbb{R}^{d_z \times d_z}$, $c_n \in \mathbb{R}^{d_z}$, $R_n \in \mathbb{R}^{d_z \times d_z}$ are the observation model parameters at time n. Assume also that the square-roots of the covariances are known, i.e., $\Sigma_0^{1/2}$, $Q_n^{1/2}$, and $R_n^{1/2}$ for all $n = 1, \ldots, N$.

In the field of Bayesian filtering and smoothing, we are typically interested in computing (some of) the following quantities:

- filtering distributions $p(x_n \mid z_{1:n})$ for all n,
- prediction distributions $p(x_n \mid z_{1:n-1})$ for all n,
- smoothing distributions $p(x_n \mid z_{1:N})$ for all n,
- marginal likelihood $p(z_{1:N})$,
- the full posterior $p(x_{0:N} \mid z_{1:N})$; since it satisfies

$$p(x_{0:N} \mid z_{1:N}) = p(x_N \mid z_{1:N}) \prod_{n=0}^{N-1} p(x_n \mid x_{n+1}, z_{1:n}), \tag{2.17}$$

  it is sufficient to compute the filtering distribution $p(x_N \mid z_{1:N})$ and the backwards transitions $\{p(x_n \mid x_{n+1}, z_{1:n})\}_{n=0}^{N-1}$.

It turns out that all of these quantities can be computed recursively by applying the Gaussian inference formulas from Sections 2.1 and 2.2.

Algorithm 2.4 (Inference in affine Gaussian state-space models) Given the parameters of an LGSSM $(\mu_0, \Sigma_0^{1/2}), (A_{1:N}, b_{1:N}, Q_{1:N}^{1/2}), (H_{1:N}, c_{1:N}, R_{1:N}^{1/2})$ (as in Equation 2.16) and a sequence of observations $z_{1:N}$, perform sequential Gaussian inference in the LGSSM:

1. Initialize the log-likelihood, defined as $\mathrm{LL}_n := p(z_{1:n})$, with $\mathrm{LL}_0 = 0$.
2. For $n = 1, \ldots, N$ do
   - **Predict:** Compute $p(x_n \mid z_{1:n-1}) = \mathcal{N}\big(x_n; \mu_n^P, \Sigma_n^P\big)$ with
     $\mu_n^P, \Sigma_n^P := \mathrm{MARGINALIZE}((\mu_{n-1}, \Sigma_{n-1}), (A_n, b_n, Q_n))$. (Algorithm 2.1)
   - **Compute the observation estimate:** $p(z_n \mid z_{1:n-1}) = \mathcal{N}(z_n; \hat{z}_n, S_n)$ with
     $\hat{z}_n, S_n := \mathrm{MARGINALIZE}\big((\mu_n^P, \Sigma_n^P), (H_n, c_n, R_n)\big)$. (Algorithm 2.1)
   - **Increment the log-likelihood**
     $\mathrm{LL}_n := \mathrm{LL}_{n-1} + \log \mathcal{N}(z_n; \hat{z}_n, S_n)$.
   - **Update:** Compute $p(x_n \mid z_{1:n}) = \mathcal{N}(x_n; \mu_n, \Sigma_n)$ with
     $\mu_n, \Sigma_n := \mathrm{UPDATE}\big((\mu_n^P, \Sigma_n^P), (H_n, c_n, R_n), (\hat{z}_n, S_n), z_n\big)$. (Algorithm 2.3)
   - (Optional) **Compute the backward transition** $p(x_{n-1} \mid x_n, z_{1:n-1}) = \mathcal{N}(x_{n-1}; G_n x_n + d_n, \Lambda_n)$ with
     $G_n, d_n, \Lambda_n := \mathrm{INVERT}\big((\mu_{n-1}, \Sigma_{n-1}), (\mu_n^P, \Sigma_n^P), (A_n, b_n, Q_n)\big)$. (Algorithm 2.2)
3. (Optional) **Compute the posterior marginals**
   - Set $\mu_N^S, \Sigma_N^S := \mu_N, \Sigma_N$
   - For $n = N, \ldots, 1$ compute
     $\mu_{n-1}^S, \Sigma_{n-1}^S := \mathrm{MARGINALIZE}\big((\mu_n^S, \Sigma_n^S), (G_n, d_n, \Lambda_n)\big)$ (Algorithm 2.1)

Return the parameters of all desired quantities:

- the filtering distributions: $(\mu_{1:N}, \Sigma_{1:N})$,
- the smoothing distributions: $(\mu_{1:N}^S, \Sigma_{1:N}^S)$,
- the log-likelihood: LL_N,
- the backward transitions of the posterior: $(G_{1:N}, d_{1:N}, \Lambda_{1:N})$.

If we only compute the filtering distributions, Algorithm 2.4 corresponds exactly to a (square-root) Kalman filter. Similarly, the smoothing distributions returned by this algorithm correspond exactly to those returned by a (square-root) Rauch–Tung–Striebel smoother. The sequential likelihood computation is also known as prediction error decomposition [113].

Remark 2.1 (Missing observations) If for certain time steps n the observation z_n is missing, Algorithm 2.4 can still be used for exact inference by simply skipping the observation estimate computation, the log-likelihood increment, and the update step for these steps.

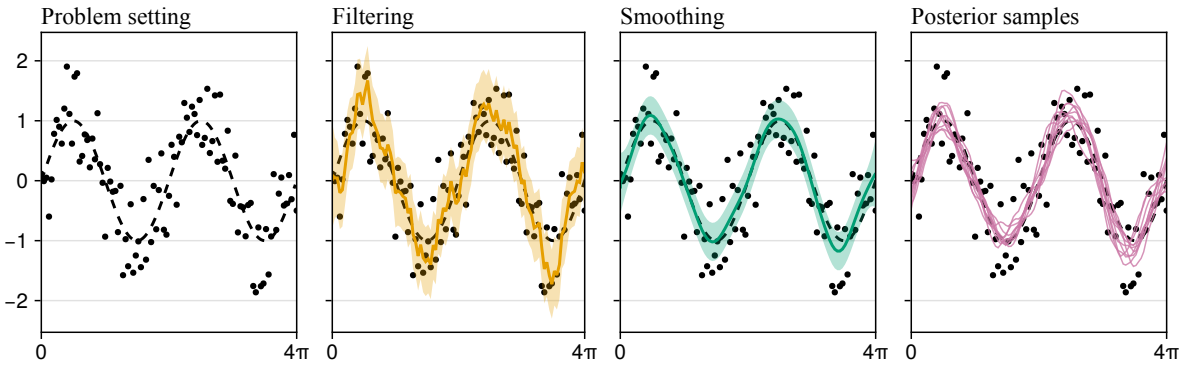


Figure 2.2: **Filtering and smoothing outputs for an LGSSM inference problem.** The filtering distribution appears closer to the observations, but the smoothing distribution is a better estimate for the underlying function of interest (black dashed line). Sampling from the posterior gives the most expressive description as it also visualizes temporal correlations.

Remark 2.2 (On the computational (in)efficiency of Algorithm 2.4) Our presentation of Algorithm 2.4 favors simplicity over efficiency: By sticking to the simple building blocks of Gaussian marginalization, inversion, and updating, we aim to formulate Gaussian filtering and smoothing in a very simple and intuitive manner. This hopefully empowers the reader to approach more complex state-estimation problems and build custom algorithms for these, by simply reducing the required steps to the three building blocks. On the flip side, the strict separation into these building blocks leads to some redundant computations and therefore to suboptimal performance. But this can easily be improved in an actual implementation by merging some of these operations.

2.3.1 Example: Sequential inference in a simple LGSSM

To give a brief example, we apply the LGSSM inference algorithm (2.4) to a simple Bayesian state estimation problem. We consider an LGSSM given by

$$x_0 \sim \mathcal{N}\left(x_0; \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & \\ & 1\end{bmatrix}\right), \tag{2.18a}$$

$$x_n \mid x_{n-1} \sim \mathcal{N}\left(x_n; \begin{bmatrix}1 & \Delta t\\ & 1\end{bmatrix} x_{n-1}, \begin{bmatrix}\frac{\Delta t^3}{3} & \frac{\Delta t^2}{2}\\ \frac{\Delta t^2}{2} & \Delta t\end{bmatrix}\right), \tag{2.18b}$$

$$z_n \mid x_n \sim \mathcal{N}\Big(z_n; \begin{bmatrix}1 & 0\end{bmatrix} x_n, 0.25\Big), \tag{2.18c}$$

with $\Delta t = \frac{4\pi}{100}$; this parameter can be interpreted as the step size at which some underlying continuous model is discretized. We generate synthetic data as noisy draws from a sine function $z_n = \sin(n \cdot \Delta t) + \varepsilon_n$ with $\varepsilon_n \sim \mathcal{N}(\varepsilon_n; 0, 0.25)$ for $n = 1, \ldots, 100$. We then perform inference with the LGSSM inference algorithm (2.4).

Figure 2.2 visualizes the quantities returned by the method, projected into the observation space via multiplication with the observation matrix $H = \begin{bmatrix}1 & 0\end{bmatrix}$. The filtering distribution often appears to be closer to the observations, but it is very non-smooth. The smoothing

distribution is in comparison a much better estimate for the true solution trajectory as it is able to take into account all data, but it also shows only marginal distributions. Sampling from the posterior distribution returned by the method gives the most expressive description here as it also visualizes temporal correlations.

2.4 Sequential approximate inference in nonlinear Gaussian state-space models

The state-space models that we are ultimately interested in are unfortunately not linear. But, we can still build on the LGSSM inference algorithm from the previous section. We consider a nonlinear Gaussian state-space model (NLGSSM) of the form

$$x_0 \sim \mathcal{N}(x_0; \mu_0, \Sigma_0), \tag{2.19a}$$

$$x_n \mid x_{n-1} \sim \mathcal{N}(x_n; f_n(x_{n-1}), Q_n), \tag{2.19b}$$

$$z_n \mid x_n \sim \mathcal{N}(z_n; h_n(x_n), R_n), \tag{2.19c}$$

where $x_n \in \mathbb{R}^{d_x}$ is the state, $z_n \in \mathbb{R}^{d_z}$ is the observation, $f_n : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ is a nonlinear transition function, $h_n : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ is a nonlinear observation function, and $Q_n \in \mathbb{R}^{d_x \times d_x}$, $R_n \in \mathbb{R}^{d_z \times d_z}$ are noise covariances at time $n = 1, \ldots, N$. As before, we assume that the covariances are known in square-root form, i.e., $\Sigma_0^{1/2}$, $Q_n^{1/2}$, and $R_n^{1/2}$.

2.4.1 Linearizing conditional Gaussian distributions

In nonlinear models the marginal distribution, the inverse transition, and the updated distribution all become non-Gaussian, and the three building blocks that we defined in Sections 2.1 and 2.2 are not directly applicable. One conceptually simple approach for developing efficient, approximate inference algorithms is via linearization: By approximating the nonlinear conditional Gaussian distributions into affine models, we can build on the Gaussian inference framework that we established so far to perform efficient, closed-form inference.

The linearization step can be performed in different ways, and many approaches for linearizing conditional Gaussian distributions have been proposed in the filtering and smoothing literature. We will focus on linearization via Taylor-approximation here as it is conceptually simple, very well-known, efficient, and tends to work well in practice.

To this end, consider a Gaussian distribution $p(x) = \mathcal{N}(x; \mu_x, \Sigma_x)$ with mean $\mu_x \in \mathbb{R}^N$ and covariance $\Sigma_x \in \mathbb{R}^{N \times N}$ and a nonlinear Gaussian observation model $p(z \mid x) = \mathcal{N}(z; h(x), R)$, where $h : \mathbb{R}^n \to \mathbb{R}^m$ is a nonlinear function. By doing a first-order Taylor expansion of h around some point $\xi \in \mathbb{R}^n$, we obtain

$$h(x) = \underbrace{h(\xi) + J_h(\xi)(x - \xi)}_{=:\bar{h}_\xi(x)} + \mathcal{O}(\|x - \xi\|), \tag{2.20}$$

where $J_h(\xi) \in \mathbb{R}^{m \times n}$ is the Jacobian of h at ξ. Then, we approximate h with $\bar{h}_\xi$ to obtain a new, *affine* approximate observation model $q(z \mid x) \approx p(z \mid x)$, with which we can then apply all the Gaussian inference algorithms from Sections 2.1 and 2.2 to perform approximate inference. We summarize the linearization approach in the following algorithm.

Algorithm 2.5 (Linearization via Taylor-approximation) Given the parameters (h, R) of a nonlinear conditional Gaussian distribution $p(z \mid x) = \mathcal{N}(z; h(x), R)$, and a linearization point $\xi \in \mathbb{R}^n$, compute the Gaussian approximation $q(z \mid x) = \mathcal{N}(z; Hx + c, R)$ with

$$H := J_h(\xi), \tag{2.21a}$$

$$c := h(\xi) - H\xi, \tag{2.21b}$$

Return the parameters (H, c, R) of the linearized model.

This is exactly the approach taken in the extended Kalman filter and extended Rauch–Tung–Striebel smoother: By linearizing the observation model in this way, the problem becomes affine and inference becomes tractable. We did not yet discuss how to select the linearization point ξ, but we will discuss two common approaches in the next sections which lead to two different algorithms: local linearization at the predicted mean, and iterated posterior linearization.

Remark 2.3 (Alternative linearization approaches) There are many other ways to linearize a nonlinear Gaussian model, for example with higher-order Taylor approximations, quadrature-based methods (e.g. Gauss–Hermite cubature, spherical cubature, the unscented transform, ...), statistical linearization, or statistical linear regression (which can even be applied to non-Gaussian models). For a more detailed discussion of these methods, refer to Särkkä et al. [108, Chapter 7-10].

2.4.2 Local linearization: The extended Kalman filter and smoother

One common choice for the linearization point is the prior mean, that is $\xi = \mu_x$. To apply this to the NLGSSM inference problem from eq. (2.19), this means that we always linearize the transition and observation models around the mean of the "current best" state estimate: at each step, we linearize the transition model around the filtering mean such that we can then predict the next state with the linearized model, and we linearize the observation model around the predicted mean before computing the likelihood and updating the state. The resulting algorithm is also known as the (square-root) extended Kalman filter and (square-root) extended Rauch–Tung–Striebel smoother, and we summarize it in the following algorithm.

Algorithm 2.6 (Approximate inference in nonlinear Gaussian state-space models) Given the parameters of an NLGSSM $(\mu_0, \Sigma_0^{1/2}), (f_{1:N}, Q_{1:N}^{1/2}), (h_{1:N}, R_{1:N}^{1/2})$, as in eq. (2.16), and a sequence of observations $z_{1:N}$, perform sequential Gaussian inference in the LGSSM:

1. Initialize the log-likelihood $\mathrm{LL}_0 = 0$
2. For $n = 1, \ldots, N$ do
   - **Linearize the transition model:**
     $A_n, b_n, Q_n := \mathrm{LINEARIZE}\big(f_n, Q_n^{1/2}, \mu_{n-1}, \big)$, (Algorithm 2.5)
   - **Predict:** Compute $p(x_n \mid z_{1:n-1}) = \mathcal{N}\big(x_n; \mu_n^P, \Sigma_n^P\big)$ with
     $\mu_n^P, \Sigma_n^P := \mathrm{MARGINALIZE}((\mu_{n-1}, \Sigma_{n-1}), (A_n, b_n, Q_n))$. (Algorithm 2.1)
   - **Linearize the observation model:**
     $H_n, c_n, R_n := \mathrm{LINEARIZE}\big(h_n, R_n^{1/2}, \mu_n^P, \big)$, (Algorithm 2.5)
   - **Compute observation estimate:** $p(z_n \mid z_{1:n-1}) = \mathcal{N}(z_n; \hat{z}_n, S_n)$ with
     $\hat{z}_n, S_n := \mathrm{MARGINALIZE}\big((\mu_n^P, \Sigma_n^P), (H_n, c_n, R_n)\big)$. (Algorithm 2.1)
   - **Increment the log-likelihood**
     $\mathrm{LL}_n := \mathrm{LL}_{n-1} + \log \mathcal{N}(z_n; \hat{z}_n, S_n)$.
   - **Update:** Compute $p(x_n \mid z_{1:n}) = \mathcal{N}(x_n; \mu_n, \Sigma_n)$ with
     $\mu_n, \Sigma_n := \mathrm{UPDATE}\big((\mu_n^P, \Sigma_n^P), (H_n, c_n, R_n), (\hat{z}_n, S_n), z_n\big)$. (Algorithm 2.3)
   - (Optional) **Compute the backward transition** $p(x_{n-1} \mid x_n, z_{1:n-1}) = \mathcal{N}(x_{n-1}; G_n x_n + d_n, \Lambda_n)$ with
     $G_n, d_n, \Lambda_n := \mathrm{INVERT}\big((\mu_{n-1}, \Sigma_{n-1}), (\mu_n^P, \Sigma_n^P), (A_n, b_n, Q_n)\big)$ (Algorithm 2.2)
3. (Optional) **Compute the posterior marginals**
   - Set $\mu_N^S, \Sigma_N^S := \mu_N, \Sigma_N$
   - For $n = N, \ldots, 1$ compute
     $\mu_{n-1}^S, \Sigma_{n-1}^S := \mathrm{MARGINALIZE}\big((\mu_n^S, \Sigma_n^S), (G_n, d_n, \Lambda_n)\big)$ (Algorithm 2.1)

Return the parameters of all desired quantities:

- the filtering distributions: $(\mu_{1:N}, \Sigma_{1:N})$,
- the smoothing distributions: $(\mu_{1:N}^S, \Sigma_{1:N}^S)$,
- the log-likelihood: LL_N,
- the backward transitions of the posterior: $(G_{1:N}, d_{1:N}, \Lambda_{1:N})$.

Remark 2.4 (On affine models, Algorithm 2.6 is equivalent to Algorithm 2.4 and performs exact inference) First-order Taylor approximations of affine functions are exact. Therefore, given a model with affine transition and observation functions, the general NLGSSM

inference algorithm 2.6 returns the same result as the LGSSM inference algorithm 2.4 and thus performs exact inference.

2.4.3 Global linearization: The iterated extended Kalman smoother

Instead of linearizing the NLGSSM at each time step separately, we can also linearize the entire model *globally* over some trajectory of linearization points $\xi_{1:N}$. This is done in the *iterated extended Kalman smoother* (IEKS): Given an initial trajectory of linearization points $\xi_{1:N}$, the IEKS linearizes the entire model around these points and then performs a single pass of the Kalman smoother to compute the posterior. Then, the mean of the posterior marginals is used as the new linearization points, and the process is repeated until convergence.

Algorithm 2.7 (Iterated extended Kalman smoother) Given the parameters of an NLGSSM as in eq. (2.16) $(\mu_0, \Sigma_0^{1/2}), (f_{1:N}, Q_{1:N}^{1/2}), (h_{1:N}, R_{1:N}^{1/2})$, a sequence of observations $z_{1:N}$, and an initial trajectory of linearization points $\xi_{0:N}$, repeat the following steps until convergence:

1. Linearize the entire model around the linearization points for all $n = 1, \ldots, N$:

$$A_n, b_n, Q_n := \text{LINEARIZE}\Big(f_n, Q_n^{1/2}, \xi_{n-1}\Big), \qquad \text{(Algorithm 2.5)}$$

$$H_n, c_n, R_n := \text{LINEARIZE}\Big(h_n, R_n^{1/2}, \xi_n\Big). \qquad \text{(Algorithm 2.5)}$$

2. Compute the posterior marginals of the linearized LGSSM with Algorithm 2.4:

$$(\mu_{0:N}^S, \Sigma_{0:N}^S), \ldots := \text{LGSSM\_INFERENCE}\,\big(\\ (\mu_0, \Sigma_0^{1/2}), (A_{1:N}, b_{1:N}, Q_{1:N}), (H_{1:N}, c_{1:N}, R_{1:N}), z_{1:N}\big)$$

3. Set the new linearization points to the posterior means: $\xi_n := \mu_n^S$ for all n.

Return the parameters of all desired quantities: the smoothing distributions $(\mu_{1:N}^S, \Sigma_{1:N}^S)$, the posterior, the log-likelihood, ...

The IEKS is a powerful method which often leads to good results in nonlinear models [108]. It is also known to (locally) converge to the maximum a posteriori (MAP) estimate of the state trajectory. That is, the returned posterior means satisfy (in the limit)

$$\mu_{1:N}^S = \arg\max_{x_{1:N}} p(x_{1:N} \mid z_{1:N}). \tag{2.22}$$

The IEKS algorithm is also known to be equivalent to the Gauss–Newton method for computing this MAP estimate, implemented in a recursive, sequential manner; refer to Bell [5] for a detailed discussion of this connection. Finally, the initial trajectory of linearization points $\xi_{1:N}$ is typically computed with a local-linearization-based extended Kalman smoother (Algorithm 2.6). This can help speed up the convergence of the IEKS if the initial trajectory is already close to the MAP estimate. But overall, the IEKS comes with a larger computational cost than its non-iterated counterpart.

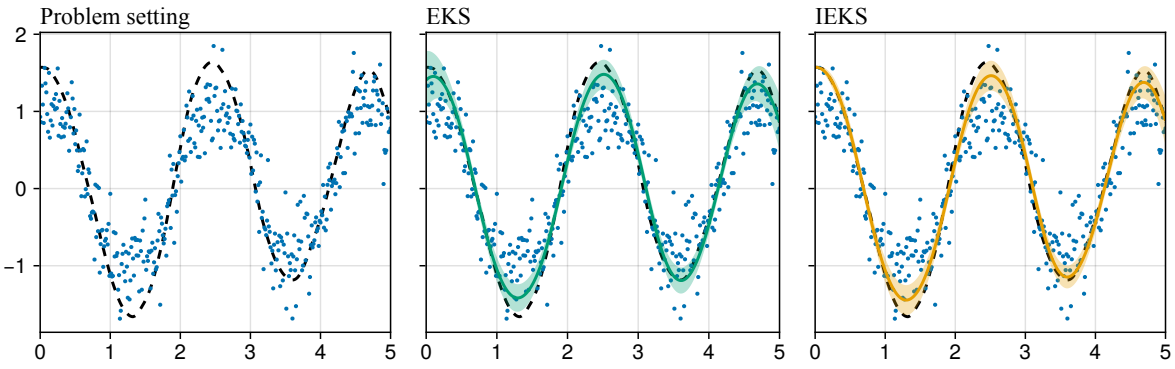


Figure 2.3: **NLGSSM inference results with the local and iterated global linearization.** *Left*: The problem setting, consisting of a true underlying function and nonlinear observations. *Center*: The local-linearization-based `EKS` is able to return an accurate and calibrated estimate. *Right*: The iterated, global linearization-based `IEKS` also returns an accurate and calibrated estimate, but in particular in the beginning of the time interval the estimate is more accurate and more certain than the `EKS`.

2.4.4 Example: Sequential inference in an NLGSSM

We demonstrate both NLGSSM inference algorithms 2.6 and 2.7 on a simple but nonlinear Bayesian state estimation problem. Consider the NLGSSM given by

$$x_0 \sim \mathcal{N}\left(x_0; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}\right), \tag{2.23a}$$

$$x_{n+1} \mid x_n \sim \mathcal{N}\left(x_{n+1}; \begin{bmatrix} (x_n)_1 + (x_n)_2 \Delta t \\ (x_n)_2 - 9.81 \sin((x_n)_1)\Delta t \end{bmatrix}, \begin{bmatrix} \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & \Delta t \end{bmatrix}\right), \tag{2.23b}$$

$$z_n \mid x_n \sim \mathcal{N}\left(z_n; \sin\left(\begin{bmatrix} 1 & 0 \end{bmatrix} x_n\right), 0.3^2\right), \tag{2.23c}$$

with $\Delta t = 1/80$; this parameter can be interpreted as the step size at which some underlying continuous model is discretized. We generate ground-truth states and observations by sampling from this NLGSSM for $n = 0, \ldots, 400$, starting with a known $x_0 = \begin{bmatrix} \pi/2 & 0 \end{bmatrix}^\top$. Then, we infer the unknown states $x_{0:400}$ from the observations $z_{0:400}$ with both the local-linearization-based NLGSSM inference algorithm 2.6 (`EKS`) and with the iterated global linearization-based algorithm 2.7 (`IEKS`).

Figure 2.3 shows the results. Overall we observe that both methods return meaningful posterior estimates for the true underlying state, with both accurate mean estimates and good coverage by the 95% credible intervals. The difference between the `EKS` and the `IEKS` appears most prominently in the very beginning of the trajectory: the `EKS` is calibrated but rather uncertain about the initial value, whereas the `IEKS` is able to estimate the true initial value with both high accuracy and low uncertainty. We can also evaluate the quality of the

posterior mean by computing its root mean square error (RMSE) with respect to the true trajectory, defined as

$$\mathrm{RMSE}(\mu_{1:N}) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} \|\mu_n - x_n^{(\mathrm{true})}\|_2^2}. \tag{2.24}$$

We obtain RMSEs of 0.52 for the `EKS` and 0.48 for the `IEKS`, indicating again that while both methods returned similar results on this simple example, the `IEKS` tends to be a bit more accurate. This difference can be much more pronounced on more complicated problems, and the MAP estimate computed by the `IEKS` is often the more accurate (as well as more interpretable) result. For more examples and a much more thorough discussion, refer to Särkkä et al. [108].

2.5 Conclusion

In this chapter, we introduced linear and nonlinear Gaussian state-space models and developed efficient algorithms for exact and approximate inference. At their core, these methods all build on three building blocks of Gaussian inference: marginalization, inversion, and updating on an observation. From these, we derived the sequential inference algorithm for linear Gaussian state-space models, known as the Kalman filter and Rauch–Tung–Striebel smoother. For nonlinear models, we simply linearize the nonlinearities via Taylor-approximation and then again rely on the Gaussian inference formulas. When done locally we obtain the extended Kalman filter and extended Rauch–Tung–Striebel smoother, and when iterated globally we obtain the iterated extended Kalman smoother. These inference algorithms will be the main workhorse of the probabilistic numerical ODE solvers developed in later chapters.

3 Gauss–Markov Processes

This chapter considers problems of estimating unknown *functions* from observations. Section 3.1 introduces *Gaussian processes* (GPs), a general framework for modeling and inferring unknown functions that is well-known in machine learning. But, their computational cost can be prohibitive for certain applications as it scales cubically with the number of data points—which would be particularly problematic for our application of simulating ODEs as the number of steps performed by numerical solvers can become arbitrarily large. To address this issue, Section 3.2 introduces a different framework for working with certain Gaussian processes, namely linear time-invariant stochastic differential equations, and shows how we can work with the resulting Gauss–Markov processes in a computationally efficient manner with recursive algorithms and with linear cost $\mathcal{O}(N)$. This is explained in Section 3.3. Section 3.5 then extends this framework to nonlinear observation models and provides efficient approximate inference procedures.

3.1 Gaussian process regression

Gaussian processes provide a convenient framework for modeling and inferring unknown functions. We follow Rasmussen et al. [101, Definition 2.1] and define them in the following way.

Definition 3.1 (Gaussian Process) *A* Gaussian process *(GP) is a collection of random variables on a common space $\mathbb{R}^d$, any finite number of which have a joint Gaussian distribution.*

A Gaussian process y is typically described by its mean function m and its covariance function k (or *kernel function*) as

$$m(\xi) = \mathbb{E}[y(\xi)], \tag{3.1a}$$

$$k(\xi, \xi') = \mathbb{E}\Big[\big(y(\xi) - m(\xi)\big)\big(y(\xi') - m(\xi')\big)^\top\Big], \tag{3.1b}$$

and we write $y \sim \mathcal{GP}(m, k)$. Given a finite collection of inputs $\{\xi_n\}_{n=1}^N$, the function values $\{y(\xi_n)\}_{n=1}^N$ are jointly Gaussian distributed, as

$$\begin{pmatrix} y(\xi_1) \\ \vdots \\ y(\xi_N) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(\xi_1) \\ \vdots \\ m(\xi_N) \end{pmatrix}, \begin{pmatrix} k(\xi_1, \xi_1) & \cdots & k(\xi_1, \xi_N) \\ \vdots & \ddots & \vdots \\ k(\xi_N, \xi_1) & \cdots & k(\xi_N, \xi_N) \end{pmatrix}\right). \tag{3.2}$$

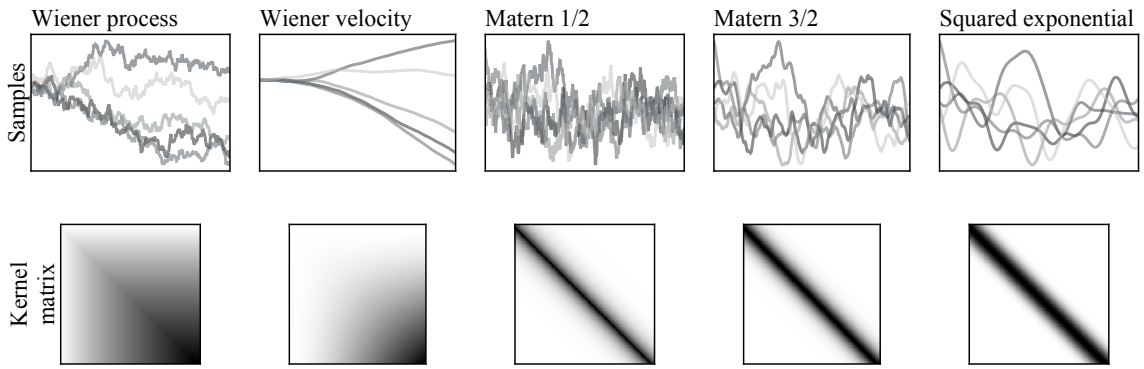


Figure 3.1: **Examples of various Gaussian process priors.** Samples (top) and kernel matrix (bottom) for a range of different priors (columns).

More compactly, we denote collections of inputs using subscripts, i.e. $\xi_{1:N}$, and with a slight abuse of notation we write

$$y(\xi_{1:N}) \sim \mathcal{N}(m(\xi_{1:N}), k(\xi_{1:N}, \xi_{1:N})). \tag{3.3}$$

The covariance matrix $k(\xi_{1:N}, \xi_{1:N})$ is also referred to as the *kernel matrix* or *Gram matrix*. Without loss of generality we only consider zero-mean Gaussian processes with $m \equiv 0$ [101].

Example 3.1 (Popular GP kernels) Popular covariance functions used for GP regression include:

1. *Wiener process*:
$$k_{\text{Wiener}}(\xi, \xi') = \sigma^2 \min(\xi, \xi'), \tag{3.4}$$
for one-dimensional non-negative $\xi \in \mathbb{R}_{\geq 0}$, with a scale hyperparameter σ^2.

2. *Wiener velocity*:
$$k_{\text{WienerVel}}(\xi, \xi') = \sigma^2 \left(\frac{\min^3(\xi, \xi')}{3} + \left|\xi - \xi'\right| \frac{\min^2(\xi, \xi')}{2} \right) \tag{3.5}$$
for one-dimensional non-negative $\xi \in \mathbb{R}_{\geq 0}$, with a scale hyperparameter σ^2.

3. *Matérn*:
$$k_{\text{Matérn}}(\xi, \xi') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} \left\|\xi - \xi'\right\| \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\ell} \left\|\xi - \xi'\right\| \right), \tag{3.6}$$
where σ^2 is a scale hyperparameter, ℓ is a lengthscale hyperparameter, and ν is a smoothness hyperparameter, and where Γ is the gamma function and K_ν is the modified Bessel function of the second kind [101].

4. *Squared exponential*:

$$k_{\mathrm{SE}}(\xi, \xi') = \sigma^2 \exp\left(-\frac{\|\xi - \xi'\|^2}{2\ell}\right), \tag{3.7}$$

where σ^2 is a scale hyperparameter and ℓ is a lengthscale hyperparameter.

See Figure 3.1 for a visualization of these kernel functions and of the resulting Gaussian processes.

Now that we introduced GPs as priors for modeling unknown functions, we can incorporate knowledge from data into these distributions. Let $y \sim \mathcal{GP}(0, k)$ be a Gaussian process and assume that the data $z_{1:N}$ are noisy observations of y, generated by the observation model

$$z_n \sim \mathcal{N}\big(z_n; y(\xi_n), \sigma^2 I\big), \qquad n = 1, \dots, N. \tag{3.8}$$

We then want to compute the posterior distribution over point evaluations of the unknown function y given the observations $z_{1:N}$, that is, $p(y(\cdot) \mid z_{1:N})$. This is known as *Gaussian process regression*; see also Rasmussen et al. [101, Section 2.2].

Proposition 3.1 (Batch GP regression) *Let $y \sim \mathcal{GP}(0, k)$ be a Gaussian process and $z_{1:N}$ be noisy observations of y as defined above with locations $\xi_{1:N}$ Then, the posterior distribution of $y(\cdot)$ given $z_{1:N}$ is again a Gaussian process, with*

$$\begin{aligned} p\big(y(\xi'_{1:M}) \mid z_{1:N}\big) \sim \mathcal{N}\Big(y(\xi'_{1:M}); & K_{MN}\big(K_{NN} + \sigma^2 I\big)^{-1} z_{1:N}, \\ & K_{MM} - K_{MN}\big(K_{NN} + \sigma^2 I\big)^{-1} K_{MN}^\top\Big), \end{aligned} \tag{3.9}$$

where $\xi'_{1:M}$ are the arbitrary locations at which we want to evaluate the posterior, and with $K_{NN} := k(\xi_{1:N}, \xi_{1:N})$, $K_{MN} := k(\xi'_{1:M}, \xi_{1:N})$, and $K_{MM} := k(\xi'_{1:M}, \xi'_{1:M})$.

Sketch of the proof. This theorem follows from the general Gaussian inference formulas from Section 2.1: First, we compute the posterior exactly on the input locations by applying the Gaussian inversion formula. This yields $p(y(\xi_{1:N}) \mid z_{1:N})$. Then, to compute the posterior at arbitrary locations $\xi'_{1:M}$, we use the known prior joint distribution of $y(\xi_{1:N})$ and $y(\xi'_{1:M})$ and apply the marginalization formula. We obtain the posterior $p(y(\xi'_{1:M}) \mid z_{1:N})$. □

Figure 3.2 shows examples of Gaussian process regression posteriors for different kernels, together with the resulting covariance and precision matrices.

Remark 3.1 (Computational cost of batch Gaussian process regression) Implementing Gaussian process regression naively as described in Proposition 3.1 is expensive: Eq. (3.9) requires inverting the $(N \times N)$-dimensional Gram matrix $K_{NN} + \sigma^2 I$. This has cubic computational cost $\mathcal{O}\big(N^3\big)$.

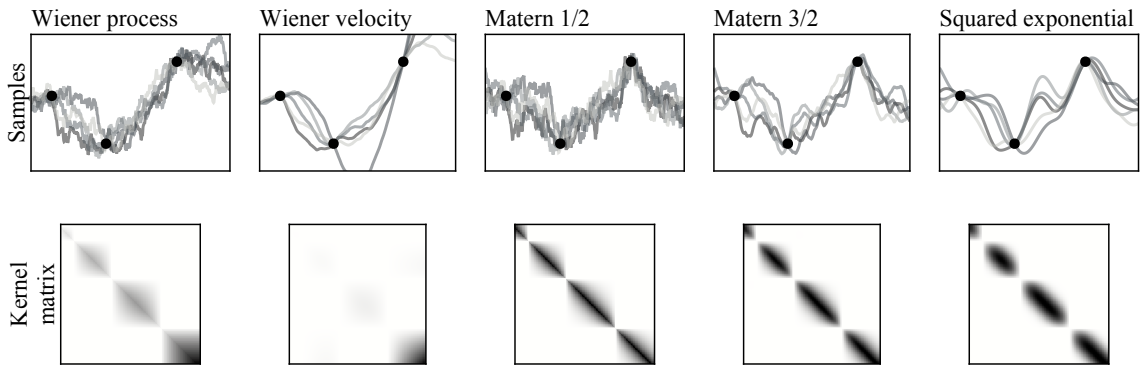


Figure 3.2: **Examples of various Gaussian process regression posteriors.** Samples (top) and kernel matrix (bottom) of Gaussian process posteriors for a range of different priors (columns).

This cubic computational cost sparked a lot of research in the Gaussian process community, and many approximations have been proposed since; we refer to Liu et al. [78] for a review on the topic. But for certain one-dimensional Gaussian processes we do not need any additional approximations, and they can be implemented *exactly* with linear cost $\mathcal{O}(N)$. These will be the topic of interest in the remaining parts of this chapter.

3.2 Gauss–Markov processes as linear time-invariant stochastic differential equations

In this section, we present a different framework for representing and working with Gaussian processes that will lead to a more efficient exact inference procedure: linear time-invariant stochastic differential equations.

We consider linear, time-invariant (LTI) stochastic differential equations (SDEs) [107, 90] of the form

$$x(0) \sim \mathcal{N}(x(0); \mu_0, \Sigma_0), \tag{3.10a}$$

$$\mathrm{d}x(t) = Fx(t)\mathrm{d}t + \Gamma^{1/2}\mathrm{d}w(t), \tag{3.10b}$$

$$y(t) = Ex(t), \tag{3.10c}$$

with *state* $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^D$, *drift* matrix $F \in \mathbb{R}^{D\times D}$, *dispersion* matrix $\Gamma^{1/2} \in \mathbb{R}^{D\times d}$, and $w : t \to \mathbb{R}^d$ a vector of standard Wiener processes. The function $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^d$ is called the *output* of the system, and it is obtained from the state x by a linear transformation with the *output matrix* $E \in \mathbb{R}^{d\times D}$. Note that we write the dispersion matrix $\Gamma^{1/2}$ as a matrix square-root, as it can be interpreted as a transformation of the standard Wiener process into a process with semi-positive definite diffusion $\Gamma \in \mathbb{R}^{D\times D}$.

In the following we will show that solutions of LTI-SDEs can be described analytically, have properties that enable efficient computation, and they are Gaussian processes. Later we will then use these as priors for Gaussian process regression.

Proposition 3.2 (Properties of LTI-SDE solutions) *The solution x of the LTI-SDE eq. (3.10) satisfies the following properties:*

1. Markov property*: x is a Markov process, with*

$$p(x(t) \mid \{x(\tau)\}_{0 \le \tau \le s}) = p(x(t) \mid x(s)). \qquad \forall t \ge s. \tag{3.11}$$

   Or more intuitively, "its future is independent of its past given the present".

2. Gaussian marginals*: The marginal distributions of $x(t)$ are Gaussian, with*

$$x(t) \sim \mathcal{N}(\mu(t), \Sigma(t)), \tag{3.12a}$$
$$\mu(t) = A(t)\mu_0, \tag{3.12b}$$
$$\Sigma(t) = A(t)\Sigma_0 A^\top(t) + Q(t), \tag{3.12c}$$

   where the matrices $A(t), Q(t)$ relate to $F, \Gamma^{1/2}$ as

$$A(t) := \exp(Ft)\mu_0, \tag{3.13a}$$
$$Q(t) := \int_0^t \exp(F(t-\tau))\Gamma^{1/2}(\Gamma^{1/2})^\top \exp(F(t-\tau))^T \mathrm{d}\tau. \tag{3.13b}$$

   These matrices can be computed efficiently with numerical methods, for example with a matrix fraction decomposition [107, Section 6.3], or the doubling-based method by Stillfjord et al. [117] which directly computes square roots of $Q(t)$.

3. Gaussian transition densities*: The conditional distribution of a state $x(t)$ given $x(s)$, with $s < t$, is a conditional linear Gaussian distribution of the form*

$$p(x(t) \mid x(s)) = \mathcal{N}(x(t); A(t-s)x(s), Q(t-s)), \tag{3.14}$$

   with transition matrices $A(t-s), Q(t-s)$ as defined in eq. (3.13).

4. Sequential representation of the process*: Assuming ordered inputs $t_1 < t_2 < \cdots < t_N$, the distribution of the state $x(t_{1:N})$ can be written as*

$$p(x(t_{1:N})) = p(x(t_1)) \prod_{n=2}^{N} p(x(t_n) \mid x(t_{n-1})). \tag{3.15}$$

5. Gaussian process*: x is a Gaussian process, and any finite collection of states $x(t_{1:N})$ have a joint Gaussian distribution.*

Proof. We prove the individual properties as follows:

1. *Markov property*: The state x is defined as the solution to the LTI-SDE eq. (3.10b) with initial distribution eq. (3.10a). Since the SDE is driven by a standard Wiener process,

and Wiener processes are Markov processes, the state x is also a Markov process. See also Särkkä et al. [107].

2. *Gaussian marginals*: Taking the mean and covariance of the LTI-SDE eq. (3.10b) results in ordinary differential equations for the mean and covariance of the state x

$$\frac{\mathrm{d}\mu(t)}{\mathrm{d}t} = F\mu(t), \tag{3.16a}$$
$$\frac{\mathrm{d}\Sigma(t)}{\mathrm{d}t} = F\Sigma(t) + \Sigma(t)F^\top + \Gamma^{1/2}(\Gamma^{1/2})^\top. \tag{3.16b}$$

These are solved exactly by [107, Section 6.2]

$$\mu(t) = \exp(Ft)\mu(0), \tag{3.17a}$$
$$\begin{aligned}\Sigma(t) = \exp(Ft)\Sigma(0)\exp(Ft)^\top \\ + \int_0^t \exp(F(t-\tau))\Gamma^{1/2}(\Gamma^{1/2})^\top \exp(F(t-\tau))^T \mathrm{d}\tau.\end{aligned} \tag{3.17b}$$

Defining $A(t)$ and $Q(t)$ as in eq. (3.13) yields the desired result.

3. *Gaussian transition densities*: The transition densities can be derived by applying the marginal distribution formula to a modified version of the LTI-SDE of eq. (3.10), with its initial distribution set exactly to $x(s)$, i.e. $\mu_0 = x(s)$ and $\Sigma_0 = 0$, and then computing the marginals at time $t - s$ with the formulas above (eq. (3.17a)).

4. *Sequential representation of the process*: Given a sequence of inputs $t_{1:N}$, the distribution of the state $x(t_{1:N})$ can be written as

$$p(x(t_{1:N})) = p(x(t_1)) \prod_{n=2}^{N} p(x(t_n) \mid x(t_{1:i-1})). \tag{3.18}$$

Since x is Markovian (eq. (3.11)) the conditional distributions simplify to

$$p(x(t_{1:N})) = p(x(t_1)) \prod_{n=2}^{N} p(x(t_n) \mid x(t_{i-1})). \tag{3.19}$$

5. *Gaussian process*: Since $p(x(t_1))$ is Gaussian and $p(x(t_n) \mid x(t_{n-1}))$ are linear conditional Gaussian distributions, the joint distribution of $x(t_{1:N})$ is Gaussian for any finite collection of inputs $t_{1:N}$. Therefore, x is a Gaussian process.

□

Corollary 3.3 (The LTI-SDE output is a Gaussian process) *The solution y of the LTI-SDE eq. (3.10) is a Gaussian process, and $y(t)$ has Gaussian marginals*

$$p(y(t)) = \mathcal{N}\Big(E(A(t)\mu_0), E\Big(A(t)\Sigma_0 A^\top(t) + Q(t)\Big)E^\top\Big), \tag{3.20}$$

with $A(t), Q(t)$ as defined in eq. (3.13).

Proof. The output $y(t)$ of the LTI-SDE eq. (3.10) is a linear transformation of the Gaussian process $x(t)$, with $y(t) = Ex(t)$. Therefore: (i) since $x(t)$ is a GP, $y(t)$ is also a GP, and (ii) since $y(t)$ is a linear transformation of $x(t)$, the marginals of $y(t)$ are transformations of the marginals of $x(t)$ as provided in eq. (3.20). □

Corollary 3.3 and Proposition 3.2 link LTI-SDEs to Gaussian processes and establish LTI-SDEs as another way to define Gaussian processes, compared to the traditional kernel-based approach. It turns out that we can write many familiar Gaussian processes as LTI-SDEs.

Example 3.2 (Popular GP kernels as LTI-SDEs) Some of the Gaussian processes defined by the kernels from Example 3.1 can be represented as outputs of LTI-SDEs:

1. *Wiener process*: The Wiener process can be represented as a simple one-dimensional LTI-SDE with

$$F_{\mathrm{WP}} = \begin{bmatrix} 0 \end{bmatrix}, \qquad \Gamma^{1/2}_{\mathrm{WP}} = \begin{bmatrix} 1 \end{bmatrix}, \qquad E_{\mathrm{WP}} = \begin{bmatrix} 1 \end{bmatrix}. \tag{3.21a}$$

2. *Wiener velocity*: The Wiener velocity process can be represented as the output of a two-dimensional LTI-SDE, where the second dimension is just a Wiener process and the first dimension is the integral of the second dimension:

$$F_{\mathrm{WV}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \Gamma^{1/2}_{\mathrm{WV}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad E_{\mathrm{WV}} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{3.22a}$$

   We also refer to this process as the *once-integrated Wiener process*.

3. *Matérn process* with smoothness ν and lengthscale ℓ: The Matérn process with half-integer smoothness ν can be represented as a $(\nu + 1/2)$-dimensional LTI-SDE, with

$$F_{\mathrm{Mat}(\nu)} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ -a_1\lambda^D & -a_2\lambda^{D-1} & \cdots & -a_D\lambda \end{bmatrix}, \qquad \Gamma^{1/2}_{\mathrm{Mat}(\nu)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \tag{3.23a}$$

$$E_{\mathrm{Mat}(\nu)} = \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix}, \tag{3.23b}$$

   $\lambda = \sqrt{2\nu}/\ell$, $D = \nu + 1/2$, $a_i = \binom{D}{i-1}$ are binomial coefficients. See also [47].

Remark 3.2 GPs with squared exponential kernel cannot be represented exactly by a finite-dimensional LTI-SDE as they are infinitely smooth. They can however be approximated; we refer to Sarkka et al. [105].

What now remains to be done is to show how to efficiently work with LTI-SDEs. The answer is already provided in Proposition 3.2, but we summarize the main finding in the following corollary.

Corollary 3.4 (Discretized LTI-SDEs are LGSSMs) *Discretizing the LTI-SDE eq. (3.10) on a grid of time points $\mathbb{T} = \{t_n\}_{n=1}^N$ yields a linear Gaussian state-space model (LGSSM) of the form*

$$x(t_0) \sim \mathcal{N}(x(t_0); \mu_0, \Sigma_0), \tag{3.24a}$$

$$x(t_{n+1}) \mid x(t_n) \sim \mathcal{N}(x(t_{n+1}); A(t_{n+1} - t_n)x(t_n), Q(t_{n+1} - t_n)), \tag{3.24b}$$

$$y(t_n) = Ex(t_n), \tag{3.24c}$$

with $A(t_{n+1} - t_n), Q(t_{n+1} - t_n)$ as defined in eq. (3.13).

Proof. Eqs. (3.24a) and (3.24c) are both specified in the given LTI-SDE eq. (3.10). The discrete transitions eq. (3.24b) are given as part of Proposition 3.2. □

This is the key to enable efficient inference: Since discretizing an LTI-SDE yields an LGSSM, we can then use the filtering and smoothing algorithms from Section 2.1 to efficiently compute the posterior distribution of the state x given noisy observations. This has $\mathcal{O}(N)$ runtime, as opposed to the $\mathcal{O}(N^3)$ runtime of the naive Gaussian process regression algorithm. We can already observe this speed-up for sampling from the prior:

Corollary 3.5 (Sampling from a Gauss–Markov process is efficient) *Sampling the Gauss–Markov process y on a grid evaluation points $\mathbb{T}_{eval} = \{t_n\}_{n=1}^N$ can be done efficiently with cost $\mathcal{O}(ND^3)$, where D is the dimension of the state.*

Proof. This directly follows from Corollary 3.4: To sample from the discretized Gauss–Markov process we can directly sample from the initial distribution and the N conditional transition densities, each of which comes with cost $\mathcal{O}(D^3)$. □

In comparison, sampling naively from a standard Gaussian Process has cubic computational cost in the number of evaluation points. Figure 3.3 shows resulting samples of different Gauss–Markov processes as defined in Example 3.2, and indeed the processes look equivalent to the kernel function-defined GPs from Example 3.1.

3.3 Recursive Gauss–Markov process regression

Now that we have established LTI-SDEs as a way to represent Gauss–Markov processes, we can use them as priors for a more efficient Gaussian process regression. To this end, let x be a Gauss–Markov process, defined as the solution of an LTI-SDE as in eq. (3.10), and let y be

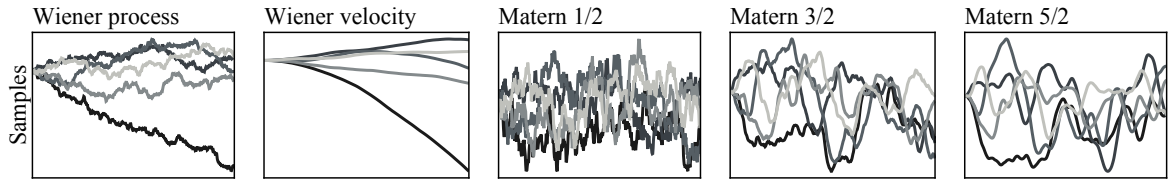


Figure 3.3: **Samples of various Gauss–Markov process priors**: These samples are generated by simulating the LTI-SDEs from Example 3.2 using the known Gaussian transition densities from Proposition 3.2.

the corresponding output process. Assume that we have noisy observations $z_{1:N}$ of y at times $\mathbb{T}_{\text{data}} = t_{1:N}$, modeled by an affine Gaussian observation model

$$z_n \sim \mathcal{N}(H_n x(t_n) + c_n, R_n), \qquad n = 1, \ldots, N, \tag{3.25}$$

with $H_n \in \mathbb{R}^{O \times D}$, $c_n \in \mathbb{R}^O$, and $R_n \in \mathbb{R}^{O \times O}$. Then, the posterior distribution $p(x(\cdot) \mid z_{1:N})$ is again a Gauss–Markov process. We call this task *Gauss–Markov process regression*, and it is also known as *continuous-discrete Bayesian state estimation* in the filtering and smoothing literature [107, Section 10.5].

Remark 3.3 (Relation to Gaussian process regression) The problem/model formulation that is common in GP regression (and that we presented in Section 3.1) slightly differs from the observation model above, as here we consider an affine observation model and heteroscedastic noise. We can recover the simpler observation model from Section 3.1 by setting $H_n = E$, $c_n = 0$, and $R_n = \sigma^2 I$. In addition, from now on the quantity of interest is the state x, but since the output directly depends on the state $y(t) = Ex(t)$ the posterior $p(y(\cdot) \mid z_{1:N})$ is also fully described by $p(x(\cdot) \mid z_{1:N})$.

With everything we have established before, the inference procedure is rather straightforward: First, we discretize the process on the data and evaluation locations to obtain a linear Gaussian state-space model. Then, we compute the posterior of the state using the LGSSM inference algorithm 2.4.

Algorithm 3.1 (Gauss–Markov process regression) Let $(\mu_0, \Sigma_0), (F, \Gamma^{1/2}, E))$ be the parameters of an LTI-SDE describing a Gauss–Markov process x, let $(H_{1:N}, c_{1:N}, R_{1:N})$ be the parameters of an affine observation model and let $z_{1:N}$ be the observations at times $\mathbb{T}_{\text{data}} = t_{1:N}$. Let $\mathbb{T}_{\text{eval}} = t'_{1:M}$ be the set of locations at which we want to compute the posterior distribution of the Gauss–Markov process. Then:

1. Discretize the LTI-SDE on the (sorted) union of the data and evaluation locations $\mathbb{T} = \mathbb{T}_{\text{data}} \cup \mathbb{T}_{\text{eval}}$, to obtain a linear Gaussian state-space model

$$x(t_0) \sim \mathcal{N}(x(t_0); \mu_0, \Sigma_0), \tag{3.26a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A(t_n - t_{n-1})x(t_{i-1}), Q(t_n - t_{n-1})), \qquad \forall t_n \in \mathbb{T}, \tag{3.26b}$$

$$z_n \sim \mathcal{N}(H_n x(t_n) + c_n, R_n), \qquad \forall t_n \in \mathbb{T}_{\text{data}}. \tag{3.26c}$$

2. Use the LGSSM inference algorithm 2.4 to compute the posterior $p(x(\mathbb{T}) \mid z_{1:N})$.

Return all desired quantities, such as the filtering and smoothing distributions, the marginal log-likelihood, the posterior $p(x(\mathbb{T}) \mid z_{1:N})$ (represented via backward transitions), and any corresponding quantities of the output $y(\mathbb{T})$.

Proposition 3.6 (Computational complexity of Gauss–Markov process regression) *The computational cost of the Gauss–Markov process regression algorithm (3.1) is $\mathcal{O}(ND^3)$, where N is the number of data points and D is the dimension of the state.*

Proof. Both the discretization and the inference can be done sequentially in time. Computing the discrete transition densities requires computing matrix exponentials, which come with cost $\mathcal{O}(D^3)$. Each step of the LGSSM inference algorithm 2.4 requires matrix-matrix multiplications and inversions, which also come with cost $\mathcal{O}(D^3)$. Since the algorithm is run for N time steps, the total cost is $\mathcal{O}(ND^3)$. □

3.3.1 Example: Gauss–Markov process regression

We can now formulate the example from Section 2.3.1 in continuous time. Consider a Wiener velocity process, defined as the output of an LTI-SDE

$$x(0) \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right), \tag{3.27a}$$

$$\mathrm{d}x(t) = \begin{bmatrix}0 & 1\\0 & 0\end{bmatrix} x(t)\mathrm{d}t + \begin{bmatrix}0\\1\end{bmatrix} \mathrm{d}w(t), \tag{3.27b}$$

$$y(t) = \begin{bmatrix}1 & 0\end{bmatrix} x(t). \tag{3.27c}$$

We assume noisy observations $\{z_n\}_{n=1}^N$ of this process, modeled by

$$z_n \sim \mathcal{N}(y(t_n), \sigma^2), \qquad n = 1, \ldots, N, \tag{3.28}$$

with $N = 100$ observations at equidistant times in the interval $[0, 4\pi]$, $\mathbb{T}_{\text{data}} = \{t_n\}_{n=1}^N$ with $t_n = n \cdot \frac{4\pi}{N}$ for $n = 1, \ldots, N$, and observation noise level $\sigma^2 = 0.1$. We also consider a set of $M = 250$ equidistant query locations $\mathbb{T}_{\text{eval}}$ to better visualize the posterior distribution, which implies that we discretize the process on the grid $\mathbb{T}_{\text{data}} \cup \mathbb{T}_{\text{eval}}$. We consider

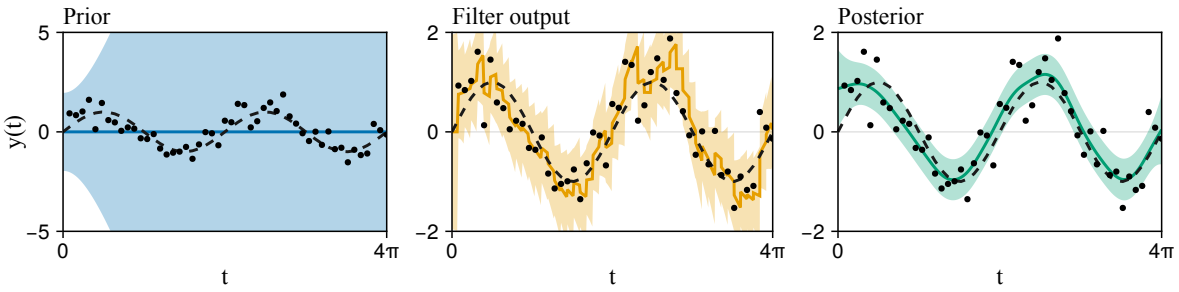


Figure 3.4: **Gauss–Markov process regression.** The Wiener velocity prior is very uninformed and broad. The filter output provides a better estimate for the true underlying function, but it is very non-smooth as at each point in time it only considers past data; in the context of Gauss–Markov process regression, filtering can be seen as an intermediate step. Smoothing returns the marginals of the actual Gauss–Markov process posterior.

a ground-truth function $y_{\text{true}}(t) = \sin(t)$, and generate synthetic data as noisy draws from $z_n \sim \mathcal{N}(y_{\text{true}}(t_n), 0.25)$, for $n = 1, \ldots, 100$.

Figure 3.4 shows the results. Similar to the previous example in Section 2.3.1, we again see that the filtering marginals have sharp edges whereas the posterior marginals are smooth. But since we now computed the estimates on a denser grid than just on the observation points we can also see the behavior of the models in-between the data points. On the filtering distribution, we observe the characteristic "trumpets" of uncertainty which arise from the interplay of extrapolation and updating: In-between the data points the filter extrapolates with the Wiener velocity process prior and thus has always-increasing uncertainty, but on the data locations the data is observed and the uncertainty reduces sharply. On the other hand, the posterior marginals returned by the smoother are aware of all the data points and thus do not show such sharp changes, as one would expect from a Gauss–Markov process posterior.

3.4 Interpolation and extrapolation of Gauss–Markov posteriors

In this section we briefly explain how a Gauss–Markov posterior can be evaluated at arbitrary locations *post-hoc*, that is after running the Gauss–Markov regression Algorithm 3.1.

The discretized posterior of the Gauss–Markov process as computed by the LGSSM inference algorithm (2.4) is represented by backward transitions $p(x(t_k) \mid x(t_{k+1}), z_{1:n})$, as

$$p(x(t_{0:N}) \mid z_{1:N}) = p(x(t_N) \mid z_{1:N}) \prod_{n=0}^{N-1} p(x(t_n) \mid x(t_{n+1}), z_{1:n}). \tag{3.29}$$

To include a new location t located in-between t_n and t_{n+1}, that is $t_n < t < t_{n+1}$, we therefore essentially need to "split" the corresponding backward transition from t_{n+1} to t_n into two separate ones. This can be done as follows. First, starting with the filtering distribution $p(x(t_n) \mid z_{1:n})$, we predict from time t_n to t with Algorithm 2.1 to obtain $p(x(t) \mid z_{1:n})$, and we then compute the backward transition $p(x(t_n) \mid x(t), z_{1:n})$ via inversion with Algorithm 2.2. Then, we repeat both steps from time t to t_{n+1} and obtain the second backward transition

$p(x(t) \mid x(t_{n+1}), z_{1:n})$. By replacing the original backward kernel $p(x(t_n) \mid x(t_{n+1}), z_{1:n})$ with the two newly computed ones, we can then compute marginals or samples of the posterior which include the desired query point t.

Note that if the posterior marginal $p(x(t_{n+1}) \mid z_{1:N})$ (the smoothing distribution) has already been computed and stored, the posterior marginal at time t given by

$$p(x(t) \mid z_{1:N}) = \int p(x(t) \mid z_{1:n}, x(t_{n+1}))p(x(t_{n+1}) \mid z_{1:N})\mathrm{d}x(t_{n+1}) \tag{3.30}$$

can be computed by simply predicting the smoothing distribution backwards with the newly computed backward kernel $p(x(t) \mid z_{1:n}, x(t_{n+1}))$.

3.5 Nonlinear Gauss–Markov process regression

Finally, we consider Gauss–Markov process regression with nonlinear observations. Let x be a Gauss–Markov process, defined as the solution of an LTI-SDE as in eq. (3.10), and let y be the corresponding output process. Assume that we have noisy observations $z_{1:N}$ of y at times $\mathbb{T}_{\text{data}} = t_{1:N}$, modeled by a nonlinear observation model

$$z_n \sim \mathcal{N}(h_n(x(t_n)), R_n), \qquad n = 1, \ldots, N, \tag{3.31}$$

with $h_n : \mathbb{R}^D \to \mathbb{R}^O$ and $R_n \in \mathbb{R}^{O \times O}$. The problem of interest is then to compute the posterior distribution of the Gauss–Markov process given the observations, $p(x(\cdot) \mid z_{1:N})$.

This is a challenging problem as the true posterior distribution is in general not Gaussian and exact inference is intractable. But, we can efficiently perform approximate inference via linearization by building on the inference algorithms for nonlinear Gaussian state-space models that we developed in Section 2.4.

The idea is the same as for Gauss–Markov regression with linear observations from before: First, we discretize the continuous-time model on the desired locations to obtain a Gaussian state-space model, only that this time it is nonlinear (an NLGSSM). Then, we apply one of the NLGSSM inference algorithms from Section 2.4, such as the extended Kalman filtering and smoothing algorithm (2.6) or the iterated extended Kalman smoothing algorithm (2.7) to compute a discretized posterior distribution. We summarize this procedure in the following algorithm.

Algorithm 3.2 (Nonlinear Gauss–Markov process regression) Let $(\mu_0, \Sigma_0), (F, \Gamma^{1/2}, E))$ be the parameters of an LTI-SDE describing a Gauss–Markov process x, let $(h_{1:N}, R_{1:N})$ be the parameters of a nonlinear observation model and let $z_{1:M}$ be the observations at times $\mathbb{T}_{\text{data}} = t_{1:N}$. Let $\mathbb{T}_{\text{eval}} = t'_{1:M}$ be the set of locations at which we want to compute the posterior distribution of the Gauss–Markov process. Then:

1. Discretize the LTI-SDE on the (sorted) union of the data and evaluation locations $\mathbb{T} = \mathbb{T}_{\text{data}} \cup \mathbb{T}_{\text{eval}}$, to obtain a nonlinear Gaussian state-space model

$$x(t_0) \sim \mathcal{N}(x(t_0); \mu_0, \Sigma_0), \tag{3.32a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A(t_n - t_{n-1})x(t_{n-1}), Q(t_n - t_{n-1})), \qquad \forall t_n \in \mathbb{T}, \tag{3.32b}$$

$$z_n \sim \mathcal{N}(h_n(x(t_n)), R_n), \qquad \forall t_n \in \mathbb{T}_{\text{data}}. \tag{3.32c}$$

2. Use one of the NLGSSM inference algorithms (2.6 or 2.7) to approximate the posterior $p(x(\mathbb{T}) \mid z_{1:N})$.

Return all desired quantities, such as the filtering and smoothing distributions, the marginal log-likelihood, and the (approximate) posterior $p(x(\mathbb{T}) \mid z_{1:N})$ represented via backward transitions.

This (approximate) posterior can then also be evaluated post-hoc at arbitrary locations, as described in Section 3.4.

3.6 Conclusion

This chapter established Gauss–Markov processes as a computationally efficient prior for modeling temporal functions. Whereas standard Gaussian processes are typically described by a mean and covariance function, we write Gauss–Markov processes as the output of a linear time-invariant stochastic differential equation. These can be discretized to obtain a linear Gaussian state-space model. By then building on filtering and smoothing algorithms, we can perform exact inference with linear cost in the number of observations. And, we can leverage the nonlinear filters and smoothers, introduced in Chapter 2, to efficiently perform approximate Gauss–Markov regression with nonlinear observations. Gauss–Markov processes will constitute the prior models of the probabilistic numerical ODE solvers which we develop in the next chapter.

Part II

Flexible and Efficient Probabilistic Numerical ODE Solvers

4 Filtering-based Probabilistic Numerical ODE Solvers

This chapter introduces the filtering-based probabilistic numerical ODE solvers that build the basis of all publications in this thesis, known as *ODE filters*. Consider an ODE initial value problem (IVP)

$$\dot{y}(t) = f(y(t), t), \quad t \in [0, T], \qquad y(0) = y_0, \tag{4.1}$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial value $y_0 \in \mathbb{R}^d$. To capture the numerical error that arises from temporal discretization, the quantity of interest in ODE filtering is the *probabilistic numerical ODE solution*, defined as the posterior distribution

$$p\Big(y(t) \;\Big|\; y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^N\Big), \tag{4.2}$$

where $\{t_n\}_{n=1}^N \subset [0, T]$ is a chosen time-discretization [122]. In the following, we pose the probabilistic numerical ODE solution as a problem of Gauss–Markov regression with a nonlinear observation model. We first define the prior, likelihood, and data model that corresponds to this posterior distribution. Then, we then solve this problem with the recursive Bayesian filtering algorithms that we have introduced in Chapter 2.

4.1 Gauss–Markov process prior

A priori, we model the ODE solution $y(t)$ as a Gauss–Markov process, defined as the output of a linear time-invariant stochastic differential equation (LTI-SDE)

$$x(0) \sim \mathcal{N}(x(0); \mu_0, \Sigma_0), \tag{4.3a}$$
$$\mathrm{d}x(t) = Fx(t)\mathrm{d}t + \Gamma^{1/2}\mathrm{d}w(t), \tag{4.3b}$$
$$y(t) = E_0 x(t), \tag{4.3c}$$

where $x(t) \in \mathbb{R}^D$ is the state of the process at time t, $F \in \mathbb{R}^{D\times D}$ is the drift matrix, $\Gamma^{1/2} \in \mathbb{R}^{D\times d}$ is the dispersion matrix, $E_0 \in \mathbb{R}^{d\times D}$ is a projection matrix, and $w : \mathbb{R}_{\geq 0} \to \mathbb{R}^d$ is a standard Wiener process. In addition, we assume that both the solution $y(t)$ and its derivative $\dot{y}(t)$ can be obtained from the state $x(t)$ via multiplication with projection matrices $E_0, E_1 \in \mathbb{R}^{d\times D}$, with $y(t) = E_0 x(t)$ and $\dot{y}(t) = E_1 x(t)$.

In discrete time, the Gauss–Markov process satisfies the transition densities

$$x(t+h) \mid x(t) \sim \mathcal{N}(x(t+h); A(h)x(t), Q(h)), \tag{4.4}$$

where $A(h), Q(h)$ are obtained from $F, \Gamma^{1/2}$ as described in Section 3.2.

Example 4.1 (The q-times integrated Wiener process) Arguably the most popular and most widely used prior for probabilistic numerical ODE solvers is the *q-times integrated Wiener process* $\mathrm{IWP}(q)$. Here, the state $x(t)$ is a stack of the ODE solution and its first q derivatives at time t, that is, $x(t) = \begin{bmatrix} y(t) & \dot{y}(t) & \cdots & y^{(q)}(t) \end{bmatrix}^\top$, and the projection matrices E_i are selection matrices of the form $E_i = e_i^\top \otimes I_d$. By modeling each dimension of the ODE solution independently as a one-dimensional $\mathrm{IWP}(q)$, the transition and dispersion matrices of the d-dimensional $\mathrm{IWP}(q)$ are Kronecker-structured matrices

$$F_{\mathrm{IWP}} := \breve{F}_{\mathrm{IWP}} \otimes I_d, \tag{4.5a}$$
$$\Gamma^{1/2}_{\mathrm{IWP}} := \breve{\Gamma}^{1/2}_{\mathrm{IWP}} \otimes I_d, \tag{4.5b}$$

where the drift and dispersion matrices $\breve{F}_{\mathrm{IWP}} \in \mathbb{R}^{(q+1)\times(q+1)}$ and $\breve{\Gamma}^{1/2}_{\mathrm{IWP}} \in \mathbb{R}^{(q+1)\times 1}$ of the one-dimensional $\mathrm{IWP}(q)$ are given by

$$\breve{F}_{\mathrm{IWP}} := \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \qquad \breve{\Gamma}^{1/2}_{\mathrm{IWP}} := \sigma \cdot \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \tag{4.6}$$

where $\sigma \in \mathbb{R}_{\geq 0}$ is a diffusion parameter. Its treatment will be the topic of Chapter 6, so we will not go into further detail in this chapter and assume a known, fixed σ for simplicity. One particularly convenient property of the $\mathrm{IWP}(q)$ is that its discrete-time transition densities are known in closed form [65]:

$$x(t+h) \mid x(t) = \mathcal{N}\big(x(t+h); A_{\mathrm{IWP}}(h)x(t), \sigma^2 Q_{\mathrm{IWP}}(h)\big), \tag{4.7}$$

with Kronecker-structured upper-triangular drift matrix $A_{\mathrm{IWP}}(h) \in \mathbb{R}^{d(q+1)\times d(q+1)}$ and Kronecker-structured covariance matrix $Q_{\mathrm{IWP}}(h) \in \mathbb{R}^{d(q+1)\times d(q+1)}$ of the form

$$A_{\mathrm{IWP}}(h) := \breve{A}_{\mathrm{IWP}}(h) \otimes I_d, \qquad \Big[\breve{A}_{\mathrm{IWP}}(h)\Big]_{ij} := \mathbb{1}_{i\leq j} \cdot \frac{h^{j-1}}{(j-i)!}, \tag{4.8a}$$
$$Q_{\mathrm{IWP}}(h) := \breve{Q}_{\mathrm{IWP}}(h) \otimes I_d, \qquad \Big[\breve{Q}_{\mathrm{IWP}}(h)\Big]_{ij} := \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!}, \tag{4.8b}$$

where $\mathbb{1}_{i\leq j}$ is the indicator function that is one if $i \leq j$ and zero otherwise, and $i, j = 1, \ldots, q+1$.

4.2 ODE information operator

To relate the introduced Gauss–Markov prior to the ODE of interest, as given in eq. (4.1), we define an observation model in terms of the information operator [27, 123]

$$\mathcal{Z}[y](t) := \dot{y}(t) - f(y(t), t). \tag{4.9}$$

By construction, $\mathcal{Z}$ maps the true solution y *exactly* to the zero function, that is, $\mathcal{Z}[y](t) = 0$ for all $t \in [0, T]$. In terms of the stochastic process x, the information operator can be expressed as

$$\mathcal{Z}[x](t) := E_1 x(t) - f(E_0 x(t), t), \tag{4.10}$$

where E_0 and E_1 are the selection matrices that map the state x to the zeroth and first derivative of the output y, respectively. There again, if x corresponds to the true ODE solution (and its true derivatives), then $\mathcal{Z}[x] \equiv 0$ holds everywhere. Conversely, if $\mathcal{Z}[x] \equiv 0$ holds everywhere and if the initial condition is satisfied, then x corresponds to the true ODE solution. Thus, ideally, we would like constrain the state x to satisfy $\mathcal{Z}[x](t) = 0$ everywhere. But, as in classic ODE solvers, computing the true ODE solution is in general infeasible. Instead, we discretize the time domain and we constrain the state to satisfy $\mathcal{Z}[x](t) = 0$ only on the discrete set of time points $\{t_n\}_{n=1}^N \subset [0, T]$. This leads to the observation and data model used in most probabilistic ODE solvers [122]:

$$z_n \mid x(t_n) \sim \delta(h_n(x(t_n))), \qquad n = 1, \ldots, N, \tag{4.11}$$

where the observation function $h_n \colon \mathbb{R}^{d(q+1)} \to \mathbb{R}^d$ is given by $h_n(x) := E_1 x - f(E_0 x, t_n)$. To enforce this equality constraint, the *data* that we condition the state on under this observation model are then noise-free, zero-valued observations $z_n = 0$ for all $n = 1, \ldots, N$.

4.3 Initialization

To solve the ODE-IVP of eq. (4.1) we also need to ensure that the process satisfies the initial value $y(0) = y_0$. One generic approach to ensure this is to start with a standard normal distributed $x(0) \sim \mathcal{N}(0_D, I_D)$, and then "observe" y_0 via the observation model

$$y_0 \mid x(0) \sim \delta(E_0 x(0)). \tag{4.12}$$

Performing a standard Gaussian update as in Algorithm 2.3 returns a mean μ_0 and covariance Σ_0 which satisfy $E_0 \mu_0 = y_0$ and $E_0 \Sigma_0 E_0^\top = 0_d$, and therefore $E_0 x(0) = y_0$ holds exactly. Alternatively, for those priors in which the state $x(t)$ is a stacked vector of the ODE solution and its first q derivatives $x(t) = \begin{bmatrix} y(t) & \dot{y}(t) & \cdots & y^{(q)}(t) \end{bmatrix}^\top$ (such as the popular q-times integrated

Wiener process presented in Example 4.1), the resulting mean and (degenerate) covariance can also be constructed directly without resorting to a Gaussian update (Algorithm 2.3), as

$$\mu_0 := \begin{bmatrix} y_0 \\ 0_d \\ \vdots \\ 0_d \end{bmatrix}, \qquad \Sigma_0 := \begin{bmatrix} 0_d & & & \\ & I_d & & \\ & & \ddots & \\ & & & I_d \end{bmatrix}. \tag{4.13}$$

This again ensures $E_0 x(0) = y_0$.

We can also include additional information about higher-order derivatives $\frac{\mathrm{d}^i y}{\mathrm{d}t^i}(0)$ into the initial state $x(0)$ to improve the stability and performance of the ODE solver [69]. These quantities can be derived from the ODE via repeated application of the chain rule (also known as Faà di Bruno's formula [103]) and can be efficiently computed via (Taylor-mode) automatic differentiation [40, 8]. Then, to ensure $E_i x(0) = \frac{\mathrm{d}^i y}{\mathrm{d}t^i}(0)$ we either condition on these with a number of "update" operations or we construct the initial mean and covariance accordingly, as described above. We refer to Krämer et al. [69] for a more detailed description of initialization of probabilistic numerical ODE solvers with higher-order information. In this thesis, unless otherwise specified we will use this exact initialization scheme and compute the higher-order derivatives with Taylor-mode automatic differentiation.

4.4 The discrete-time inference problem

The continuous-time SDE prior together with the nonlinear discrete observation model form a continuous-discrete state estimation problem; see also Section 3.5. Given the discrete time grid $\{t_n\}_{n=1}^N \subset [0, T]$, this problem can be discretized to obtain a nonlinear Gaussian state-space model (NLGSSM) of the form

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{4.14a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A(h_n)x(t), Q(h_n)), \qquad n = 1, \ldots, N, \tag{4.14b}$$

$$z_n \mid x(t_n) \sim \delta(E_1 x(t_n) - f(E_0 x(t_n), t_i)), \qquad n = 1, \ldots, N, \tag{4.14c}$$

where $h_n := t_n - t_{n-1}$ is the step-size, and with zero-valued observations $z_n = 0$ for all $n = 1, \ldots, N$. By construction, the posterior distribution $p(x(t) \mid z_{1:N})$ of this NLGSSM satisfies both the initial condition and the ODE at the discrete time points. Since the ODE solution $y(t)$ is contained in the state $x(t)$ via $y(t) = E_0 x(t)$, this posterior distribution thus corresponds to the probabilistic numerical ODE solution of eq. (4.2).

4.5 The probabilistic numerical ODE solver

We can then compute the posterior of the NLGSSM, and thus the probabilistic ODE solution, with the NLGSSM inference algorithms that we have introduced in Section 2.4. Unless oth-

erwise specified, we will use the sequential, local-linearization-based Algorithm 2.6, i.e. the extended Kalman filter and smoother.

Algorithm 4.1 (Filtering-based probabilistic numerical ODE solver) Given

- an ODE-IVP with vector field f, initial value y_0, and time interval $[0, T]$,
- a time discretization $\{t_n\}_{n=1}^N \subset [0, T]$,
- a Gauss–Markov prior with drift, dispersion, and projection matrices $F, \Gamma^{1/2}, E_0, E_1$,

compute the probabilistic numerical ODE solution (eq. (4.2)) with the following steps:

1. Discretize the continuous-time prior to obtain an NLGSSM (as in eq. (4.14)).
2. Solve the NLGSSM with the extended Kalman filter and smoother (Algorithm 2.6).

Return all desired quantities, such as the posterior marginals or the full posterior.

This is *the* probabilistic numerical ODE solver that we use in most parts of this thesis. In the following chapters, we will introduce extensions and modifications to this algorithm that make it more efficient, more flexible, and more accurate, by using different priors, information operators, linearization schemes, hyperparameter choices, and filtering and smoothing algorithms. But at the core, all the presented solvers are based on the same idea: We model the ODE solution as a Gauss–Markov process, we condition the state on the ODE at discrete time points, and we solve the resulting continuous-discrete state estimation problem with Gaussian filtering and smoothing.

4.6 Example: Probabilistic solution of a Lotka–Volterra ODE

Before we dive into more details, let us demonstrate the solver on a simple example problem. Consider the Lotka–Volterra ODE, given by

$$\dot{y}_1(t) = \alpha y_1(t) - \beta y_1(t) y_2(t), \tag{4.15a}$$

$$\dot{y}_2(t) = \delta y_1(t) y_2(t) - \gamma y_2(t), \tag{4.15b}$$

on the time interval $[0, 30]$, with model parameters $\alpha = 2/3$, $\beta = 4/3$, $\delta = 1$, and $\gamma = 1$, and initial value $y_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$. To solve this ODE in a probabilistic numerical manner, we need to decide on a prior and discretization scheme, construct the corresponding NLGSSM, and then perform inference with the extended Kalman filter and smoother.

In this example, we consider a 3-times integrated Wiener process as the prior for the ODE solution and we condition on the ODE at $N = 60$ equidistant points $t_n = n \cdot 0.5$ for $n = 1, \ldots, 60$. This gives us the following NLGSSM:

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{4.16a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A(h_n) x(t), Q(h_n)), \tag{4.16b}$$

$$z_n \mid x(t_n) \sim \delta(E_1 x(t_n) - f(E_0 x(t_n), t_n)), \tag{4.16c}$$

$$z_n = 0, \tag{4.16d}$$

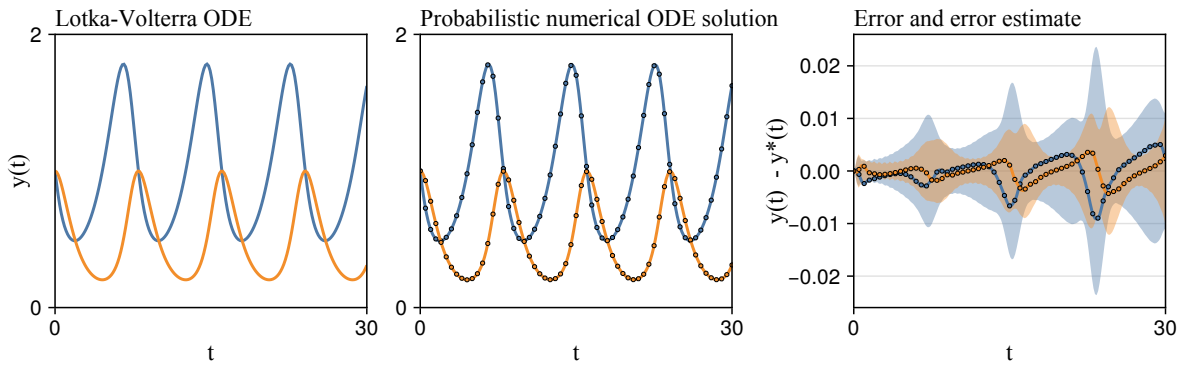


Figure 4.1: **Probabilistic numerical solution to the Lotka–Volterra ODE**: The reference solution (left) and the probabilistic numerical solution (center) to the Lotka–Volterra ODE seem very similar, indicating that the probabilistic solver produced an accurate solution estimate. The right figure visualizes the numerical error (with respect to the reference solution) together with the 95% credible interval of the numerical error estimate produced by the probabilistic solver, and from their similarity we observe that the probabilistic solver produced a meaningful estimate of the numerical error.

where the initial mean and covariance are chosen to match the initial value and derivative exactly, and standard normal for higher derivatives, as

$$\mu_0 := \begin{bmatrix} y_0 \\ f(y_0, 0) \\ 0_2 \\ 0_2 \end{bmatrix}, \qquad \Sigma_0 := \begin{bmatrix} 0_2 & & & \\ & 0_2 & & \\ & & I_2 & \\ & & & I_2 \end{bmatrix}, \tag{4.17}$$

and with the known closed-form expression for the IWP(3) transition matrices

$$A(h) := \begin{bmatrix} 1 & h & \frac{h^2}{2} & \frac{h^3}{6} \\ & 1 & h & \frac{h^2}{2} \\ & & 1 & h \\ & & & 1 \end{bmatrix} \otimes I_2, \qquad Q(h) := \begin{bmatrix} \frac{h^7}{252} & \frac{h^6}{72} & \frac{h^5}{30} & \frac{h^4}{24} \\ \frac{h^6}{72} & \frac{h^5}{20} & \frac{h^4}{8} & \frac{h^3}{6} \\ \frac{h^5}{30} & \frac{h^4}{8} & \frac{h^3}{3} & \frac{h^2}{2} \\ \frac{h^4}{24} & \frac{h^3}{6} & \frac{h^2}{2} & h \end{bmatrix} \otimes I_2. \tag{4.18}$$

The selection matrices are given by

$$E_0 := \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \otimes I_2, \qquad E_1 := \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \otimes I_2. \tag{4.19}$$

As before, the Kronecker product appears since we model each dimension of the ODE solution independently as a one-dimensional IWP(3). Then, we solve the resulting NLGSSM (as in eq. (4.14)) with the extended Kalman filter and smoother (Algorithm 2.6).

Figure 4.1 visualizes the resulting probabilistic numerical ODE solution. We see that the computed solution (center) and the reference solution (left) look very similar, showing that the probabilistic ODE solver produced an accurate solution estimate. In addition, the right figure visualizes the numerical error (with respect to the reference solution), together with the 95% credible interval of the numerical error estimate produced by the probabilistic solver.

We observe that the error and error estimate look structurally similar, the error lies within the credible interval, and the error and error estimate have a similar magnitude. Note that here we additionally *calibrated* the posterior by estimating an additional hyperparameter that was not yet introduced—this will be the topic of Section 6.1.

4.7 Probabilistic numerical solvers with approximate linearization

The Taylor-linearization-based inference algorithms from Section 2.4 approximate the observation function $h_n(x) = E_1 x - f(E_0 x, t_n)$ with

$$h_n(x) \approx J_{h_n}(\xi)x + (h_n(\xi) - J_{h_n}(\xi)\xi), \tag{4.20}$$

where the Jacobian of the observation function h_n of the form

$$J_{h_n}(\xi) = E_1 - J_f(E_0\xi, t_n)E_0, \tag{4.21}$$

and where J_f is the Jacobian of the vector field f. It turns out that a zeroth-order Taylor expansion of the vector field also yields a valid probabilistic ODE solver: By approximating the Jacobian of the vector field with the zero matrix, that is, $J_f(y,t) \approx 0$, we obtain an approximate Jacobian of the observation function $J_{h_n}(x) \approx E_1$ and the linearized observation function

$$h_n(x) \approx E_1 x + (h_n(\xi) - E_1\xi). \tag{4.22}$$

Applying this approximate linearization scheme together with the the extended Kalman filter and smoother Algorithm 2.6 yields the probabilistic ODE solver introduced by Schober et al. [112] and Kersting et al. [65]. We also refer to this extended Kalman filter/smoother-based solver with zeroth-order vector-field linearization as the `EK0` in this thesis. In comparison to the first-order linearization-based solver that we have introduced before, which we also refer to as `EK1` from now on, the `EK0` has the advantage of being computationally more efficient per step [III]; this will be discussed in more detail in Chapter 5. But, it is less stable than the `EK1` and can diverge for stiff ODEs [122]; see Figure 4.2 for an illustration of this behavior.

4.8 Properties of probabilistic numerical ODE solvers

As discussed for non-probabilistic ODE solvers in Section 1.3, probabilistic ODE solvers have a number of properties that are important to consider when using them in practice. We provide a brief overview of these properties here.

- **Convergence rates:** The convergence rates of probabilistic ODE solvers are similar to those of non-probabilistic ODE solvers. Depending on the smoothness of the chosen prior, we obtain polynomial convergence rates for the mean of the solution: For the zeroth-order linearization-based `EK0`, the *local* error of the mean estimate decreases with the step size h as $\mathcal{O}\big(h^{q+1}\big)$, where q is the smoothness of the prior (this essentially establishes smoothness

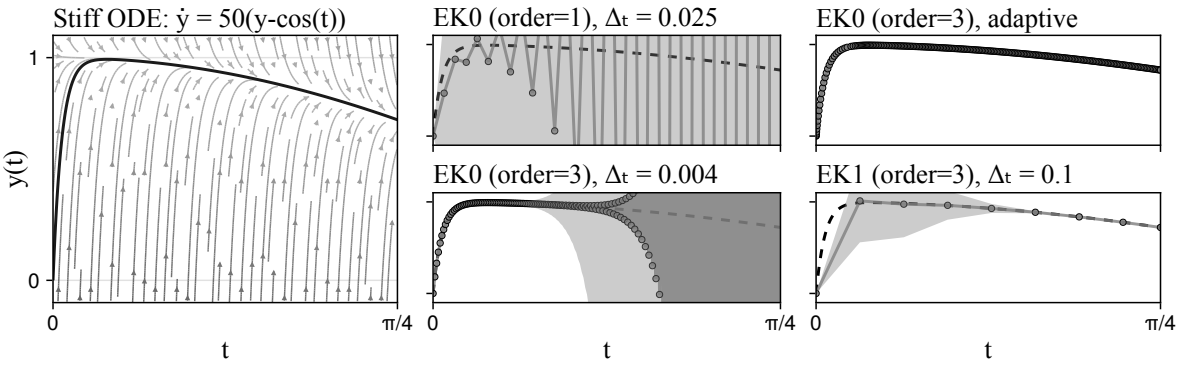


Figure 4.2: **Performance of different ODE filters on a stiff differential equation:** For larger step sizes, the `EK0` method diverges for both considered orders. Adaptive step-size selection can help, but comes with the cost of high numbers of steps. The `EK1` does not diverge even with large steps.

as the "order" of the solver) [65]; and under some additional assumptions, the *global* error of the MAP of the probabilistic ODE solution (the MAP estimate of eq. (4.2), which can be computed with the IEKS (Algorithm 2.7)) decreases with the step size as $\mathcal{O}(h^q)$ [123].

- **Continuous solutions:** The probabilistic numerical ODE solvers return a Gauss–Markov posterior which can be evaluated at arbitrary points $t > 0$ (as described in Section 3.4). This does not require the construction of any additional interpolant, but for efficient interpolation, we need to store the filtering and smoothing distributions.

- **Adaptive step-size selection**: As with non-probabilistic ODE solvers, it can be beneficial to adapt the step-size during the integration process to improve the efficiency of the solver. This can be done in a similar manner as for non-probabilistic ODE solvers. By controlling a local error estimate of the solver with a classic control algorithm, the step size can be selected on-line by the solver to achieve a desired error level. Adaptive step-size selection will be discussed in detail in Chapter 6.

- **Stability:** The probabilistic ODE solver with first-order Taylor linearization (`EK1`) is A-stable [122], which is a desirable stability property for ODE solvers that ensures that the solver does not diverge for decaying ODEs, independently of the chosen step size. In practice, the `EK1` has also been observed to be suitable for many stiff ODEs; see for example Publications I-III and [69]. On the other hand, the zeroth-order linearization-based solver presented in Section 4.7 is not A-stable and can diverge for stiff ODEs, as shown in Figure 4.2. The probabilistic exponential integrator presented Publication V satisfies an even stronger stability property, *L-stability*, which can be advantageous for certain stiff ODEs; this solver will be the topic of Chapter 8.

- **Computational complexity:** We can describe the computational complexity of probabilistic ODE solvers with respect to three main factors: The number of steps N, the dimension of the ODE d, and the smoothness of the Gauss–Markov prior q. The presented method operates sequentially on the time grid, which leads to linear cost in N. At each step,

we multiply system and covariance matrices of size $d(q+1) \times d(q+1)$ with each other. This cubic computational cost leads to a total computational complexity of $\mathcal{O}\big(Nd^3q^3\big)$. But note that this is a preliminary result for the basic formulation introduced so far: In Chapter 5 we will see how certain variations can be implemented linearly in d, and in Chapter 9 we present a parallel-in-time implementation that makes the solvers logarithmic in N.

- **Explicit and semi-implicit solvers:** The presented probabilistic ODE solvers with zeroth-order linearization (`EK0`) and first-order Taylor linearization (`EK1`) can be categorized as *explicit* and *semi-implicit* solvers, respectively: In both the `EK0` and the `EK1`, the next step can be computed without having to solve a non-linear system of equations. But in the `EK1` the update step solves a linear system of equations that involves the Jacobian of the vector field, while the update step of the `EK0` does not involve the Jacobian. Therefore, the `EK1` can be considered *semi-implicit,* or *linearly implicit* [45], while the `EK0` is *explicit.* This classification is also in line with the stability properties and the computational complexity of the solvers, as classic, non-probabilistic semi-implicit solvers are often A-stable but cubic in d, while explicit solvers are not A-stable but linear in d. However, a more formal and detailed discussion and categorization of probabilistic ODE solvers into explicit, semi-implicit, and implicit solvers is still missing in the literature.

4.9 Conclusion

This chapter introduced filtering-based probabilistic numerical ODE solvers. By choosing a Gauss–Markov process prior and transforming the ODE into a discrete observation model, the probabilistic numerical ODE solution turns into a nonlinear Gauss–Markov regression problem, which we can then solve efficiently with nonlinear filtering and smoothing. The resulting algorithms can be treated similarly to classic, non-probabilistic solvers and they have comparable convergence rates, stability properties, and computational complexity.

In the following chapters, we will extend these probabilistic numerical solvers in various ways. We improve their computational complexity (Chapters 5 and 9), their uncertainty quantification and practical computational efficiency (Chapter 6), and their numerical stability (Chapter 8). We generalize them to higher-order ODEs, energy-preserving systems, and differential-algebraic equations (Chapter 7). And, we use these probabilistic numerical solvers to perform parameter inference in ODEs (Chapter 10).

5 Scaling Probabilistic ODE Solvers to High-dimensional Problems

This chapter is based on Publication III:

Nicholas Krämer*, Nathanael Bosch*, Jonathan Schmidt*, and Philipp Hennig. **Probabilistic ODE Solutions in Millions of Dimensions**. *International Conference on Machine Learning* (ICML), 2022.

The computational complexity of probabilistic ODE solvers is linear in the number of time points, but cubic in the dimension of the ODE (as discussed in Section 4.8). This is a limiting factor for high-dimensional problems. In this section, we will discuss how probabilistic ODE solvers can be scaled to high-dimensional problems by exploiting structure in the state-space model, namely block-diagonal structure (Section 5.1) and Kronecker structure (Section 5.2)

5.1 Preserving block-diagonal structure

One way to define multi-dimensional priors is by choosing a separate prior for each dimension of the ODE solution. That is, let $y(t) = \begin{bmatrix} y_1(t) & \cdots & y_d(t) \end{bmatrix}^\top \in \mathbb{R}^d$, and let each $y_i(t) \in \mathbb{R}$ be modeled by a Gauss–Markov process, given by an LTI-SDE of the form

$$x_i(0) \sim \mathcal{N}(\mu_{0,i}, \Sigma_{0,i}), \tag{5.1a}$$
$$\mathrm{d}x_i(t) = F_i x_i(t)\mathrm{d}t + \Gamma_i^{1/2}\mathrm{d}w_i(t), \tag{5.1b}$$
$$y_i^{(j)} = E_{i,j} x_i, \qquad j = 0, \ldots, q_i, \tag{5.1c}$$

where $x_i(t) \in \mathbb{R}^{(q_i+1)}$ is the state of the process for the i-th dimension of the ODE solution, $F_i \in \mathbb{R}^{(q_i+1)\times(q_i+1)}$ is the drift matrix, $\Gamma_i^{1/2} \in \mathbb{R}^{(q_i+1)\times 1}$ is the dispersion matrix, $E_{i,j} \in \mathbb{R}^{1\times(q_i+1)}$ are the projection matrices, and $w_i : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a standard Wiener process. Then, the d-dimensional process is simply a stacked version of the one-dimensional priors:

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{5.2a}$$
$$\mathrm{d}x(t) = F x(t)\mathrm{d}t + \Gamma^{1/2}\mathrm{d}w(t), \tag{5.2b}$$
$$y^{(j)} = E_j x, \qquad j = 0, \ldots, q, \tag{5.2c}$$

*Equal contribution.

with initial mean and covariance

$$\mu_0 := \begin{bmatrix} \mu_{0,1} \\ \vdots \\ \mu_{0,d} \end{bmatrix} \in \mathbb{R}^D, \qquad \Sigma_0 := \begin{bmatrix} \Sigma_{0,1} & & \\ & \ddots & \\ & & \Sigma_{0,d} \end{bmatrix} \in \mathbb{R}^{D \times D}, \tag{5.3}$$

and block-diagonal drift, dispersion, and projection matrices

$$F := \begin{bmatrix} F_1 & & \\ & \ddots & \\ & & F_d \end{bmatrix} \in \mathbb{R}^{D \times D}, \tag{5.4a}$$

$$\Gamma^{1/2} := \begin{bmatrix} \Gamma_1^{1/2} & & \\ & \ddots & \\ & & \Gamma_d^{1/2} \end{bmatrix} \in \mathbb{R}^{D \times d}, \tag{5.4b}$$

$$E_j := \begin{bmatrix} E_{1,j} & & \\ & \ddots & \\ & & E_{d,j} \end{bmatrix} \in \mathbb{R}^{d \times D}, \tag{5.4c}$$

where $D = \sum_{i=1}^{d}(q_i + 1)$ is the total dimension of the stacked state vector. Then, since additions, products, and exponentials of matrices preserve their block-diagonal structure, the discrete-time transition densities of the d-dimensional process are also block-diagonal:

$$x(t+h) \mid x(t) = \mathcal{N}(x(t+h); A(h)x(t), Q(h)), \tag{5.5}$$

with block-diagonal transition matrices

$$A(h) := \begin{bmatrix} A_1(h) & & \\ & \ddots & \\ & & A_d(h) \end{bmatrix} \in \mathbb{R}^{D \times D}, \qquad Q(h) := \begin{bmatrix} Q_1(h) & & \\ & \ddots & \\ & & Q_d(h) \end{bmatrix} \in \mathbb{R}^{D \times D}, \tag{5.6}$$

where $A_i(h)$ and $Q_i(h)$ are the transition matrices of the one-dimensional process for the i-th dimension of the ODE solution.

We can then show that, if the observation model is affine with block-diagonal observation matrix and observation noise covariance with matching block sizes, all matrices computed by the LGSSM inference algorithm (2.4) are also block-diagonal.

Proposition 5.1 (Gaussian inference in block-diagonal LGSSMs preserves block-diagonal structure) *Consider an LGSSM with block-diagonal state-space matrices*

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{5.7a}$$
$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A(h_n)x(t), Q(h_n)), \tag{5.7b}$$
$$z_n \mid x(t_n) \sim \mathcal{N}(H_n x(t_n) + c_n, R_n), \tag{5.7c}$$

for $n = 1, \ldots, N$, where all matrices $\Sigma_0, A(h_n), Q(h_n) \in \mathbb{R}^{D\times D}$, are block-diagonal with matching block sizes (as above), and where the observation matrices $H_n \in \mathbb{R}^{d\times D}$, $R_n \in \mathbb{R}^{d\times d}$, are also block diagonal with matching block sizes. Then, all matrices that are computed by the LGSSM inference algorithm (2.4) are also block-diagonal. In particular this includes not only the covariance matrices of the filtering and smoothing distributions, but also all intermediate matrices that are computed during the inference process.

Proof. Algorithm 2.4 is a composition of the algorithms for marginalization (Algorithm 2.1), updating (Algorithm 2.3), and inversion (Algorithm 2.2). These in-turn require matrix additions, multiplications, and inversions, all of which preserve block-diagonal structure. The only operation that is not immediately clear to preserve block-diagonal structure is the triangularization operation tria, applied to a stack of input matrices, i.e. the computation of $C := \mathrm{tria}\left(\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]\right)$ for some matrices A and B. Therefore, to show that the LGSSM inference algorithm preserves block-diagonal structure, we only need to show that this operation preserves block-diagonal structure.

The operation tria is defined as follows: First, tria computes a (thin) QR decomposition of its input, here $QR := \left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$. Then, tria returns only the computed R factor, that is $C := R$. This way, the output $C := \mathrm{tria}\left(\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]\right)$ satisfies $C^\top C = A^\top A + B^\top B$. Now let A and B be block-diagonal with matching block-sizes. First, observe that re-ordering the rows of a matrix does not change the "R" factor in the QR decomposition: Let $X =: QR$ be a QR decomposition of some matrix X, and let P be a permutation matrix. Then, PQ is also orthogonal, and $(PQ)R$ is a QR decomposition of PX. Therefore, re-ordering the rows of the input matrix does not change the output of tria. We have

$$\mathrm{tria}\left(\begin{bmatrix} \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_d \end{bmatrix} \\ \begin{bmatrix} B_1 & & \\ & \ddots & \\ & & B_d \end{bmatrix} \end{bmatrix}\right) = \mathrm{tria}\left(\begin{bmatrix} \begin{bmatrix} A_1 \\ B_1 \end{bmatrix} & & \\ & \ddots & \\ & & \begin{bmatrix} A_d \\ B_d \end{bmatrix} \end{bmatrix}\right). \tag{5.8}$$

Second, QR decompositions preserve block-diagonal structure, that is

$$\begin{pmatrix} \begin{bmatrix} A_1 \\ B_1 \end{bmatrix} & & \\ & \ddots & \\ & & \begin{bmatrix} A_d \\ B_d \end{bmatrix} \end{pmatrix} = \underbrace{\begin{bmatrix} Q_1 & & \\ & \ddots & \\ & & Q_d \end{bmatrix}}_{Q} \cdot \underbrace{\begin{bmatrix} R_1 & & \\ & \ddots & \\ & & R_d \end{bmatrix}}_{R}. \tag{5.9}$$

where (Q_i, R_i) are the QR-decomposition of $\begin{bmatrix} A_i \\ B_i \end{bmatrix}$. Then, the output $C := R$ is also block-diagonal with block sizes matching those of A and B. This concludes the proof. □

Since all the matrices computed by the LGSSM inference algorithm are block-diagonal, we can exploit this structure to reduce the computational complexity of the inference algorithm.

Corollary 5.2 (Complexity of LGSSM inference in block-diagonal state spaces) *In linear Gaussian state-space models with D-dimensional state spaces and d-dimensional observations, where all matrices are block-diagonal with d-blocks of size $q_1, \ldots, q_d$, $\sum_{i=1}^{d} q_i = D$, the LGSSM inference Algorithm 2.4 has a computational complexity of $\mathcal{O}\left(N \sum_{i=1}^{d} q_i^3\right)$.*

Proof. Algorithm 2.4 requires multiplication, inversion, and QR-decomposition of matrices. Proposition 5.1 states that these matrices are always block-diagonal. Then, instead of applying these operations to the full $D \times D$ matrices, we can apply them separately to each block of size $q_i \times q_i$, with computational cost equal to the sum of the computational costs for each block, which is cubic in q_i. Similarly, a matrix-vector multiplication with a block-diagonal matrix can be computed by splitting the vector into blocks of size q_i and applying the matrix-vector multiplication to each block separately, at cost quadratic in q_i. Thus, the computational complexity of the LGSSM inference algorithm is $\mathcal{O}\left(N \sum_{i=1}^{d} q_i^3\right)$. □

Now we can apply the results from above to the probabilistic ODE solver.

Proposition 5.3 (Linear complexity of the block-diagonal-structured ODE solver) *Let $\dot{y} = f(y, t)$ be an ODE with initial value y_0, and let $p(y(t))$ be a Gauss–Markov process prior with block-diagonal state-space matrices as above, that models each dimension of the ODE solution independently. In addition, let $D(x, t) \approx J_f(x, t)$ be some diagonal approximation of the Jacobian of the vector field (for example the diagonal of the Jacobian, or a zero matrix as in Section 4.7). Then, the probabilistic ODE solver Algorithm 4.1, modified such that it uses the approximate diagonal Jacobian D in the linearization of the observation function, has a computational complexity of $\mathcal{O}\left(N \sum_{i=1}^{d} q_i^3\right)$.*

Proof. The observation function in the probabilistic ODE solver is given by

$$h_n(x) = E_1 x - f(E_0 x, t_n), \tag{5.10}$$

and it is linearized as $h_n(x) \approx H_n x + c_n$, where

$$H_n := E_1 - J_f(E_0 x, t_n) E_0, \tag{5.11a}$$

$$c_n := f(E_0 x, t_n) - E_1 x. \tag{5.11b}$$

The projection matrices $E_0, E_1 \in \mathbb{R}^{d \times D}$ are block-diagonal. If we approximate the Jacobian of the vector field with a diagonal matrix, i.e. if $J_f(E_0 x, t_n) \approx D = \operatorname{diag}(D_{11}, \ldots, D_{dd})$ then the observation matrix $H_n = E_1 - D_n E_0$ is also block-diagonal:

$$H_n = \begin{bmatrix} E_{1,1} - D_{11} E_{0,1} & & \\ & \ddots & \\ & & E_{1,d} - D_{dd} E_{0,d} \end{bmatrix}. \tag{5.12}$$

Thus, after this approximate linearization, all matrices computed in the probabilistic ODE solver are block-diagonal. With Corollary 5.2, it follows that the computational complexity of the solver is $\mathcal{O}\left(N \sum_{i=1}^{d} q_i^3\right)$, where q_i is the block size of the state-space prior for the i-th dimension of the ODE solution. □

Proposition 5.3 is a slight modification of Proposition 3.3 in Publication III, such that it applies to any block-diagonal prior (whereas Proposition 3.3 in Publication III considers the q-times integrated Wiener process specifically). In addition, Proposition 3.3 in Publication III also shows that the memory cost of the probabilistic solver is of order $\mathcal{O}\left(\sum_{i=1}^{d} q_i^2\right)$ per step, and it also discusses *calibration* of the prior—which has not yet been discussed in this thesis but will be in Section 6.1—and shows that it is compatible with the structured state-space models. While the result Proposition 5.3 should still hold for both scalar and diagonal calibration parameters, refer to Publication III for the full statement and proof of the result with calibration for IWP(q) priors.

Proposition 5.3 also provides two practical insights: First, the zeroth-order linearization-based, explicit probabilistic ODE solver as presented in Section 4.7 has a computational complexity that scales linearly in the dimension of the ODE. This holds without any modification to the existing, established algorithm, but its implementation needs to be done carefully to exploit the structure of the state-space matrices. And second, approximating the Jacobian of the vector field not with zero, but with its diagonal, leads to a new probabilistic ODE solver that also scales linearly in the dimension of the ODE. Publication III demonstrates this algorithm in practice and shows that it can be more efficient than the first-order linearization-based solver with exact Jacobian for certain problems, while also being more stable than the zeroth-order linearization-based solver.

5.2 Preserving Kronecker structure

If each dimension of the ODE solution is modeled independently by the same prior, the state-space model is not only block-diagonal, but Kronecker-structured. That is, given a one-dimensional Gauss–Markov process prior

$$x(0) \sim \mathcal{N}\Big(\breve{\mu}_0, \breve{\Sigma}_0\Big), \tag{5.13a}$$

$$\mathrm{d}x(t) = \breve{F}x(t)\mathrm{d}t + \breve{\Gamma}^{1/2}\mathrm{d}w(t), \tag{5.13b}$$

$$y^{(j)} = \breve{E}_j x, \qquad j = 0, \ldots, q, \tag{5.13c}$$

with initial mean $\breve{\mu}_0 \in \mathbb{R}^{q+1}$ and covariance $\breve{\Sigma}_0 \in \mathbb{R}^{(q+1)\times(q+1)}$, and drift matrix $\breve{F} \in \mathbb{R}^{(q+1)\times(q+1)}$, dispersion matrix $\breve{\Gamma}^{1/2} \in \mathbb{R}^{(q+1)\times 1}$, and projection matrices $\breve{E}_j \in \mathbb{R}^{1\times(q+1)}$, we can construct a d-dimensional prior by stacking independent copies of the one-dimensional prior. We obtain a Kronecker-structured prior

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{5.14a}$$

$$\mathrm{d}x(t) = Fx(t)\mathrm{d}t + \Gamma^{1/2}\mathrm{d}w(t), \tag{5.14b}$$

$$y^{(j)} = E_j x, \qquad j = 0, \ldots, q, \tag{5.14c}$$

with initial mean $\mu_0 \in \mathbb{R}^D$, and with Kronecker-structured initial covariance $\Sigma_0 = \breve{\Sigma}_0 \otimes I_d \in \mathbb{R}^{D\times D}$, drift matrix $F = \breve{F} \otimes I_d \in \mathbb{R}^{D\times D}$, dispersion matrix $\Gamma^{1/2} = \breve{\Gamma}^{1/2} \otimes I_d \in \mathbb{R}^{D\times d}$, and projection matrices $E_j = \breve{E}_j \otimes I_d \in \mathbb{R}^{d\times D}$.

Since additions, products, and exponentials of matrices preserve their Kronecker structure, the discrete-time transition densities of the d-dimensional process

$$x(t+h) \mid x(t) = \mathcal{N}(x(t+h); A(h)x(t), Q(h)) \tag{5.15}$$

also have Kronecker-structured transition matrices

$$A(h) := \breve{A}(h) \otimes I_d, \tag{5.16a}$$

$$Q(h) := \breve{Q}(h) \otimes I_d, \tag{5.16b}$$

where $\breve{A}(h)$ and $\breve{Q}(h)$ are the transition matrices of the one-dimensional process.

As in the block-diagonal case, we can show that the probabilistic ODE solver with Kronecker-structured state-space matrices preserves Kronecker structure in the inference algorithm, and thus has a computational complexity that scales linearly in the dimension of the ODE—as long as the linearized observation model is suitably structured. The argument to show this follows the same structure as in Section 5.1, but for brevity we simply state the main result here, following Publication III, Proposition 3.4.

Proposition 5.4 (Complexity of the Kronecker-structured ODE solver) *Let $\dot{y} = f(y, t)$ be an ODE with initial value y_0, and let $p(y(t))$ be a Gauss–Markov process prior with Kronecker-structured state-space matrices as above, that models each dimension of the ODE solution independently. Then, the explicit probabilistic numerical ODE solver with zeroth-order linearization of the vector field has a computational complexity of $\mathcal{O}\big(N\big(dq^2 + q^3\big)\big)$ and a memory complexity of $\mathcal{O}\big(N\big(dq + d^2 + q^2\big)\big)$.*

Sketch of the proof. The proof follows essentially the same structure as the proof for Proposition 5.3: First, we show that if all matrices in the model are Kronecker-structured, then all the the matrices computed in the extended Kalman filter and smoother are also Kronecker structured. Then, we derive the runtime and memory complexity by going through all the operations in the algorithm. And finally, we observe that if we consider the explicit probabilistic numerical ODE solver with zeroth-order linearization of the vector field, then all the system matrices are Kronecker-structured, and the proposition follows. □

The full statement and proof are also given in Publication III. One key take-away from Proposition 5.4 is that the explicit probabilistic ODE solver with zeroth-order linearization can be implemented even more efficiently than the block-diagonal version, provided that the prior is Kronecker-structured, which is the case for the standard d-dimensional IWP(q) prior which is most popularly used. But Kronecker structure also appears naturally in the context of spatio-temporal models, where the ODE solution is modeled with a spatio-temporal prior that is then discretized in space [70]. Proposition 5.4 therefore plays a crucial role in scaling explicit probabilistic ODE solvers to high-dimensional problems such as discretized partial differential equations.

5.3 Example: Performance comparison on the high-dimensional Lorenz96 ODE

We briefly show the influence of the different state-space factorizations on the runtimes of the resulting experiments with a simple experiment. We consider the Lorenz96 problem [79], which is a convenient ODE for this experiment as the dimension can be increased freely, given by a system of $d \geq 4$ ODEs

$$\dot{y}_1 = (y_2 - y_{d-1})y_d - y_1 + \theta_F, \tag{5.17a}$$
$$\dot{y}_2 = (y_3 - y_d)y_1 - y_2 + \theta_F, \tag{5.17b}$$
$$\dot{y}_i = (y_{i+1} - y_{i-2})y_{i-1} - y_i + \theta_F, \qquad i = 3, \ldots, d-1, \tag{5.17c}$$
$$\dot{y}_d = (y_1 - y_{d-2})y_{d-1} - y_d + \theta_F, \tag{5.17d}$$

with forcing term $\theta_F = 8$, initial values $y_1(0) = \theta_F + 0.01$ and $y_{>1}(0) = \theta_F$, and time span $t \in [0, 30]$. We compare EKF-based solvers resulting from three types of vector-field Jacobians: the zero-matrix (`EK0`) a diagonal approximation (`DiagonalEK0`), and the full Jacobian (`EK1`),

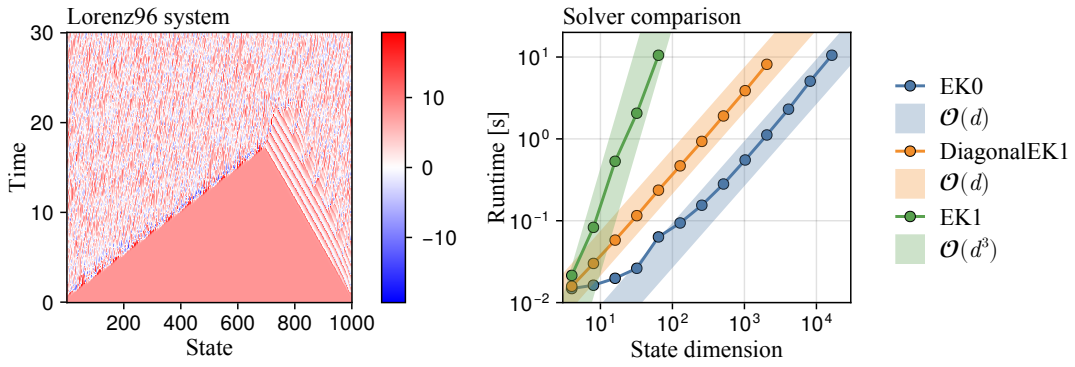


Figure 5.1: **Performance of different ODE filters for increasingly large dynamical systems:** The standard `EK1` scales cubically with the ODE dimension, due to dense covariance matrices. If instead the prior is structured and the likelihood model is (approximated) such that it preserves this structure, the scaling can be decreased to linear. The resulting `DiagonalEK1` with block-diagonal covariances and the `EK0` with Kronecker-factored covariances therefore have much lower runtimes for high-dimensional systems.

all with an independent 3-times integrated Wiener process priors for each ODE dimension. Using the results from this chapter, this implies that `EK0` is implemented with Kronecker-factored matrices, `DiagonalEK1` uses block-diagonal matrices, and `EK1` uses dense matrices, and therefore for a fixed step size we expect the fastest runtimes for the `EK0`, followed by the `DiagonalEK1`, which both scale linearly in the ODE dimension, and the slowest runtimes for the cubically-scaling `EK1`. To analyze the runtime scaling, we use fixed step sizes $\Delta t = 0.01$ for all methods and compute probabilistic solutions for varying ODE dimensions d.

Figure 5.1 shows the results. We observe that the runtime of the factorized methods is much lower for high-dimensional systems than the runtime of the `EK1` with dense covariances. Thus covariance factorizations are key to solve high-dimensional problems. Publication III includes a more thorough experimental evaluation of these methods.

5.4 Conclusion

In this chapter, we have shown how to introduce independence assumptions into the prior and inference algorithm, which we can then leverage to develop probabilistic numerical solvers that scale linearly in the ODE dimension. This enables the probabilistic numerical simulation of very high-dimensional ODEs.

This approach however comes with two potential shortcomings. First, the underlying EKF/EKS inference algorithm needs to discard off-diagonal entries of the vector-field Jacobian, which could potentially hurt its numerical stability on certain stiff systems. And second, by assuming independence between ODE dimensions the posterior uncertainties become less expressive. But filtering and smoothing in high-dimensions is a notoriously challenging problem, and a large variety of methods with different advantages and disadvantaged have been proposed. We believe that developing alternative numerical ODE solvers for high-dimensional ODEs with different trade-offs is an interesting avenue for future research.

6 Uncertainty Calibration and Step-Size Adaptation

This chapter is based on Publication I:

Nathanael Bosch, Philipp Hennig, and Filip Tronarp. **Calibrated Adaptive Probabilistic ODE Solvers**. *International Conference on Artificial Intelligence and Statistics* (AISTATS), 2021.

The probabilistic numerical ODE solvers as introduced in Chapter 4 contain hyperparameters that need to be selected by the user, but which we did not yet elaborate on. First, the *diffusion* is a hyperparameter which appears in all the considered Gauss–Markov prior which relates to the covariance of the corresponding Wiener process, and it greatly influences the scale of the uncertainty estimates computed by the probabilistic numerical solver. And second, the time points $t_1, \ldots, t_N$ at which we want to discretize the ODE to compute the probabilistic numerical solution have so far always been chosen as a uniform grid with some fixed step-size. In this section we discuss both aspects. Sections 6.1 and 6.2 provide multiple methods to estimate the diffusion, and thereby to *calibrate* the posterior computed by the probabilistic solver to obtain more reliable uncertainty estimates. Section 6.4 then discusses the related topic of *local error estimation* for probabilistic ODE solvers, and then presents a step-size adaptation scheme similar to that of non-probabilistic solvers, by building on classic control algorithms.

6.1 Uncertainty calibration

We consider q-times integrated Wiener process priors, similar to those introduced in Example 4.1, but now with an additional hyperparameter to explicitly model the *diffusion coefficient* of the process:

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{6.1a}$$

$$\mathrm{d}x(t) = F_{\mathrm{IWP}} x(t) \mathrm{d}t + \Gamma_{\mathrm{IWP}}^{1/2} \mathrm{d}w(t), \tag{6.1b}$$

$$y^{(j)} = E_j x, \qquad j = 0, \ldots, q, \tag{6.1c}$$

with drift matrix $F_{\mathrm{IWP}} = \breve{F}_{\mathrm{IWP}} \otimes I_d$ and projection matrices $E_j = \breve{E}_j \otimes I_d$ as before to model each ODE dimension independently with an IWP(q) prior, and with dispersion matrix $\Gamma_{\mathrm{IWP}}^{1/2} := \breve{\Gamma}_{\mathrm{IWP}}^{1/2} \otimes \Xi^{1/2}$ for some $\Xi^{1/2} \in \mathbb{R}^{d \times d}$. We call $\Xi^{1/2} \in \mathbb{R}^{d \times d}$ the *diffusion* parameter of

the process, as it can be interpreted as the square-root of the covariance of the Wiener process that enters the system. This parameter directly influences the magnitude of the uncertainty of the prior, and thereby also the posterior. Therefore, to obtain calibrated probabilistic numerical ODE solutions, we need to estimate this diffusion parameter Ξ.

A common way to estimate hyperparameters in Gaussian processes is to maximize the marginal likelihood of the data with respect to the hyperparameters, which is also known as "type-2 maximum likelihood estimation" [101]: Given a sequence of observations $z_{1:n}$, we compute the *maximum likelihood estimate* (MLE) $\hat{\Xi}^{1/2}_{\text{MLE}}$ as

$$\hat{\Xi}^{1/2}_{\text{MLE}} = \arg\max_{\Xi^{1/2}} p\Big(z_{1:n} \mid \Xi^{1/2}\Big) = \arg\max_{\Xi^{1/2}} p\Big(z_1 \mid \Xi^{1/2}\Big) \prod_{n=2}^{n} p\Big(z_n \mid z_{1:n-1}, \Xi^{1/2}\Big), \tag{6.2}$$

which can be computed by numerical optimization. It turns out that in our specific setting of probabilistic numerical ODE solvers, there are some additional model assumptions that we can leverage to compute $\hat{\Xi}^{1/2}_{\text{MLE}}$ more efficiently and without numerical optimization, namely the zero-valued initial prior covariance $\Sigma_0 = 0$ and the noiseless observation model with zero observation noise covariance $R = 0$. and in some cases also additional structure in the observation matrix H. In the following, we show multiple schemes for (approximately) computing the MLE during the forward pass of the ODE solver.

6.1.1 Calibrating a scalar-valued diffusion parameter

Consider a scalar diffusion $\Xi^{1/2} = \sigma \cdot I_d$, with $\sigma \in \mathbb{R}_{>0}$ and identity matrix $I_d \in \mathbb{R}^{d \times d}$. Then, Tronarp et al. [122, Proposition 4] show how to compute a *quasi maximum-likelihood estimate* of the scalar diffusion hyperparameter σ in closed form, where the term "quasi" refers to the fact that the estimate is based on the linearized observation model [76]. Additionally, this quantity can be computed *post-hoc*, that is after running the algorithm with some uninformed choice for the diffusion parameter, and we can then use it to re-scale the resulting covariances. We summarize the result by Tronarp et al. [122, Proposition 4] as follows:

Proposition 6.1 ((Quasi-)MLE for scalar diffusion models) *Let $\dot{y} = f(y,t)$ be an ODE with initial value $y(0) = y_0$, and let $(F, \Gamma^{1/2})$ be the drift and dispersion matrices of an LTI-SDE describing a Gauss–Markov prior, with exact initial state $x(0) = \mu_0$).*
Then, the probabilistic ODE solution computed with the diffusion parameter $\sigma = \bar{\sigma}$ is equal to the probabilistic ODE solution computed with unit diffusion parameter $\sigma = 1$, up to scaling of the computed covariances. More specifically, denote the filter mean and covariance at time t_n that were computed with diffusion parameter $\sigma = \bar{\sigma}$ by (μ_n^F, Σ_n^F) and the filter mean and covariance computed with unit diffusion parameter $\sigma = 1$ by $(\check{\mu}_n^F, \check{\Sigma}_n^F)$. Then, they satisfy $(\mu_n^F, \Sigma_n^F) = (\check{\mu}_n^F, \bar{\sigma}^2 \check{\Sigma}_n^F)$.

Additionally, let $\check{z}_n$ and $\check{S}_n$ be the predicted mean and covariance of the measurement z_n computed while using the unit diffusion parameter $\sigma = 1$. Then the maximum likelihood estimate of σ is given by

$$\hat{\sigma} = \sqrt{\frac{1}{Nd} \sum_{n=1}^{N} \hat{z}_n^\top S_n^{-1} \hat{z}_n}. \tag{6.3}$$

This quantity can be computed on-line during the forward pass of the ODE filter.

This estimate is well-founded in the sense that, if the ODE were linear (and thus the resulting Gaussian state-space model is linear), then the computed $\hat{\sigma}$ corresponds exactly to the maximum likelihood estimate of eq. (6.2). For the full detailed statement and proof, refer to Tronarp et al. [122, Proposition 4].

6.1.2 Calibrating a diagonal-valued diffusion parameter

Now, consider a *diagonal* diffusion parameter

$$\Xi^{1/2} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_d \end{bmatrix}, \tag{6.4}$$

or short $\Xi^{1/2} = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$, with $\sigma_1, \ldots, \sigma_d \in \mathbb{R}_{>0}$. In this case, we cannot obtain the exactly the same result as in Proposition 6.1 since the Kronecker structure of the dispersion $\Gamma^{1/2} = \check{\Gamma}^{1/2} \otimes \Xi^{1/2}$ is generally not guaranteed to be preserved by the ODE filter. But as we have previously seen in Section 5.2, the explicit solver based on zeroth-order linearization of the vector field preserves Kronecker structure. This leads to the following result.

Proposition 6.2 ((Quasi-)MLE for diagonal diffusion models) *Let $\Xi^{1/2} = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$ and $\Sigma_0 = \check{\Sigma}_0 \otimes \Xi$. Then the prediction and filtering covariances computed by the explicit probabilistic ODE solver with IWP prior and with zeroth-order vector-field linearization are of the form $\Sigma_n^P = \check{\Sigma}_n^P \otimes \Xi$, $\Sigma_n^F = \check{\Sigma}_n^F \otimes \Xi$, and the approximated measurement covariances are given by $S_n = \check{s}_n \cdot \Xi^{1/2}$, with $\check{s}_n := \check{E}_1 \check{\Sigma}_n^P \check{E}_1^\top$. The quasi maximum-likelihood estimate of Ξ, denoted by $\hat{\Xi}$, is diagonal and given by*

$$\hat{\Xi}_{ii} = \frac{1}{N} \sum_{n=1}^{N} \frac{(\hat{z}_n)_i^2}{\check{s}_n}, \qquad i \in \{1, \ldots, d\}. \tag{6.5}$$

This quantity can be computed on-line during the forward pass of the ODE filter.

This result is stated as Proposition 1 in Publication I. We refer to the paper for its proof. It enables us to use a multivariate diffusion model and thereby make the uncertainties returned by the explicit solver more expressive. We will visualize these differences in posterior uncertainties later in this chapter.

6.2 Time-varying diffusion models

We now consider a relaxation of the previous model and assume that the diffusion parameter $\Xi^{1/2}$ is not fixed, but instead is allowed to vary for different integration steps; we denote by $\Xi_n^{1/2}$ the diffusion parameter for the time interval $[t_{n-1}, t_n]$. A scalar-valued version of this model has been previously considered by Schober et al. [112], in the context of local error estimation and step-size adaptation. We re-visited this model in Publication I, and extended it to diagonal-valued diffusions. In the following we briefly state the main results; for a more thorough discussion, refer to Publication I.

A main assumption in this context is that we calibrate only *locally*. That is, at each step of the solver we aim to compute a (quasi-)MLE

$$\hat{\Xi}_n^{1/2} = \arg\max_{\Xi_n^{1/2}} p\Big(z_n \mid z_{1:n-1}, \Xi_n^{1/2}\Big). \tag{6.6}$$

In the case of an extended Kalman filter it simplifies to the quasi-MLE

$$\hat{\Xi}_n^{1/2} \approx \arg\max_{\Xi_n^{1/2}} \mathcal{N}(z_n; \hat{z}_n, S_n), \tag{6.7}$$

where $\hat{z}_n, S_n$ are the predicted mean and covariance of the observation at time t_n of the form

$$\hat{z}_n = f\big(E_0 \mu_n^P, t_n\big) - H_n \mu_n^P, \tag{6.8a}$$

$$S_n = H_n \Sigma_n^P H_n, \tag{6.8b}$$

with linearized observation matrix $H_n = E_1 - J_f E_0$. To obtain a tractable approximation of the MLE, we approximate the predicted state covariance $\Sigma_n^P = A\Sigma_{n-1}^F A^\top + \breve{Q}_n \otimes \Xi_n$ by discarding the previous filtering covariance Σ_{n-1}^F, that is $\Sigma_n^P \approx \breve{Q}_n \otimes \Xi_n$. The predicted observation covariance simplifies to

$$S_n \approx H_n\Big(\breve{Q}_n \otimes \Xi_n\Big) H_n^\top. \tag{6.9}$$

Using this approximation, we can then compute the resulting approximated quasi-MLE in closed form for the same settings as before in Section 6.1.

Proposition 6.3 (Local (quasi-)MLE for scalar diffusion models) *Consider a time-varying, scalar-valued diffusion model $\Xi_n^{1/2} = \sigma_n \cdot I$. Then, the local quasi-MLE is*

$$\hat{\sigma}_n^2 = \frac{1}{d} \hat{z}_n^\top \Big(H_n Q_n H_n^\top\Big)^{-1} \hat{z}_n. \tag{6.10}$$

This quantity can be computed on-line during the forward pass of the ODE filter.

Proposition 6.4 (Local (quasi-)MLE for diagonal diffusion models) *Consider a time-varying, diagonal-valued diffusion model* $\Xi_n^{1/2} = \mathrm{diag}(\sigma_{1,n}, \ldots, \sigma_{d,n})$*, used in the explicit probabilistic solver with zeroth-order vector-field linearization. Then, the local quasi-MLE is*

$$\hat{\sigma}_{i,n}^2 = \frac{(\hat{z}_n)_i^2}{(Q_n)_{11}}, \qquad i = 1, \ldots, d, \tag{6.11}$$

where $(\hat{z}_i)_j$ *is the* j*-th component of the predicted mean of the measurement at time* t_i*, and* $(Q_i)_{11}$ *is the first diagonal element of the process noise covariance at time* t_i*.*

The scalar-valued result corresponds exactly to the calibration by Schober et al. [112]; both results are also provided in Publication I.

6.3 Example: Comparison of the different calibration approaches

In the previous sections we introduced various approaches for calibration, with scalar- and diagonal-valued diffusion models, estimated globally and locally. And in addition, the probabilistic ODE solver itself also influences the resulting uncertainty estimates due to the different linearization strategies. Here, we visualize these differences. Consider the Lotka–Volterra ODE, given by

$$\dot{y}_1(t) = \alpha y_1(t) - \beta y_1(t) y_2(t), \tag{6.12a}$$

$$\dot{y}_2(t) = \delta y_1(t) y_2(t) - \gamma y_2(t), \tag{6.12b}$$

on the time interval $[0, 20]$, with model parameters $\alpha = 3/2$, $\beta = 1$, $\delta = 3$, and $\gamma = 1$, and initial value $y_0 = (1, 1)^\top$. We solve this ODE with all the possible combinations of solvers, based on zeroth-order or first-order vector-field linearization (`EK0` and `EK1`), scalar- and diagonal-valued diffusion, and global and local diffusion models. All methods use an IWP(2) prior and fixed steps of size $\Delta t = 10^{-2}$.

Figure 6.1 shows the results. We see that the structure of the resulting uncertainty differs greatly between the approaches. We also observe a number of particular properties of the different models: The uncertainty of the `EK0` always increases as time progresses, for all combinations. For scalar-valued diffusions, the uncertainties of the `EK0` are also the same for all the dimensions of the ODE, but for diagonal-valued models their scale can differ. The `EK0` with global calibration also shows the characteristic uncertainty structure of the IWP(2) prior, but with local calibration the growth of the uncertainty adapts to the vector field. Finally, the `EK1` has the most flexible uncertainties and its structure suits the actual resulting numerical error. In this particular example, the `EK1` with local scalar calibration achieves the lowest errors and also shows very structured uncertainty, but it appears to be underconfident. On the other hand, the `EK1` with global scalar diffusion seems to be the best-calibrated of all the methods. A more quantitative evaluation of the calibration of probabilistic ODE solvers is provided in Publication I.

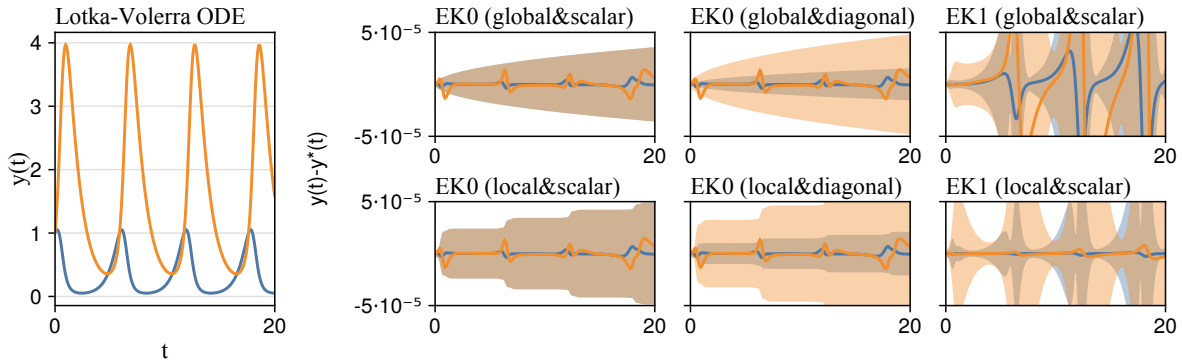


Figure 6.1: **Comparison of the different calibration approaches.** We solve the Lotka–Volterra ODE (left) with a range of combinations of `EK0` and `EK1` as solvers, scalar-valued and diagonal-valued diffusion models, and with local and global calibration. We can see that the structure of the resulting uncertainty differs greatly between the approaches. In particular, the uncertainty of the `EK0` is very limited and it always grows as time progresses. On the other hand, the uncertainty returned by the `EK1` is expressive and follows the structure of the actual numerical error, both for the global and local calibration.

6.4 Step-size adaptation

So far we have always considered the setting in which we manually specify the time steps $t_0, t_1, \ldots, t_N$ at which we want to discretize the ODE, and we have selected this time discretization to be a uniform grid over the integration interval, with $t_i = i \cdot \Delta t$ for some specified step size $\Delta t > 0$. But it is not always clear how the step size should be chosen, and with a uniform grid we cannot focus our computational resources to those parts of the ODE that are particularly challenging to simulate. In this section, we consider an alternative to this approach which allows us to let the solver choose the step sizes automatically, such that a specified numerical error is achieved: *step-size adaptation*.

Step-size adaptation plays an important role in non-probabilistic numerical ODE solvers, and it is implemented in most modern ODE software packages such as SciPy [126], DifferentialEquations.jl [97], and Matlab [116]. Instead of following a specified fixed step size, the ODE solvers compute an estimate of the *local error* at each step, and then continues depending on this error estimate. If the estimated error is too large, the step is rejected, the step-size is decreased, and the attempt is repeated. If the estimated error is sufficiently low, the step is accepted and the next step size is computed. This is typically done by some control algorithm, such as for example proportional control [44, Chapter II.4], PI-control [41], or PID-control [128]. In the following, we first discuss local error estimation in the context of probabilistic ODE solvers, and then briefly describe step-size selection with a proportional controller.

6.4.1 Local error estimation

The local error estimate proposed by Schober et al. [112] and in Publication I works as follows. First, compute the local scalar diffusion estimate σ_n^2 as described in Proposition 6.3. Then, the quantity that we choose to control with the step-size selection algorithm is the marginal standard deviation of the residual $E_1\mu_n - f(E_0\mu_n, t_n)$, also known as the "defect" in classical

numerical analysis [33, 52, 114]. Assuming no error for time t_{n-1} and thus $\Sigma_{n-1} = 0$, the marginal variance of the defect is given by

$$e^2 := \sigma_n^2 \operatorname{diag}\Big(H_n Q_n H_n^\top\Big), \tag{6.13}$$

where H_n are the linearized observation matrices at time t_n and Q_n is the (uncalibrated) transition noise covariance. Note that a similar quantity can also be obtained when using a diagonal diffusion; this is explained in more detail in Publication I.

Remark 6.1 (Other local error estimates for adaptive step-size selection) The defect-based local error estimate from eq. (6.13) is, at the time of writing, the de-facto standard objective for step-size selection in probabilistic ODE solvers. But other quantities might also lend themselves for local error control. Krämer et al. [68] proposed two alternatives in the context of global mesh-refinement for probabilistic boundary-value problem solvers: Instead of controlling the standard deviation of the residual we can also directly control the residual itself, namely

$$e_{\text{res}} := E_1 \mu_n - f(E_0 \mu_n, t_n). \tag{6.14}$$

Or, both ideas can also be combined to a quantity

$$e_{\text{prob}} := e_{\text{res}} + e, \tag{6.15}$$

which relates to a probabilistic upper bound of the probability of the defect being too large; refer to Krämer et al. [68] for a more thorough discussion. In addition, one could also control the estimated error on the solution itself, or compute a local error estimate by using a second method with higher order as is most commonly done in Runge–Kutta methods [44, Chapter II.4]. A thorough exploration of these different local error estimates for probabilistic numerical ODE solvers is still lacking. In the remainder of this thesis and in our implementation (Publication VIII), we therefore use the defect-inspired quantity from eq. (6.13).

6.4.2 Step-size selection with proportional control

Given a local error estimate $e \in \mathbb{R}^d$, the step-size controller aims to select step sizes as large as possible but while ensuring that a specified absolute and relative tolerance level is met. More formally, we want to have

$$e_i \leq \varepsilon_i, \qquad \text{with} \quad \varepsilon_i := \tau_{\text{abs}} + \tau_{\text{rel}} \cdot \max(|(E_0 \mu_n)_i|, |(E_0 \mu_{n-1})_i|), \quad i = 1, \dots d, \tag{6.16}$$

where the parameters τ_{abs} and τ_{rel} specify the absolute and relative tolerance level, respectively, and $E_0 \mu_n$ and $E_0 \mu_{n-1}$ are the estimated mean of the solution at time t_n and t_{n-1} (the

latter is included to improve the stability of the controller). To control that e_i is "close to but smaller than" ε_i, we define the scalar quantity

$$E := \sqrt{\frac{1}{d}\sum_{i=1}^{d}\left(\frac{e_i}{\varepsilon_i}\right)^2}, \tag{6.17}$$

which should then be "close to but smaller than" 1. We can now reason about the acceptance or rejection of the proposed step: If $E \leq 1$ then the proposed step is accepted and the integration continues. Otherwise, the step is rejected as insufficiently accurate and the computation is repeated. In both cases, we use a *proportional controller*, which is a simple control algorithm that is widely used in non-probabilistic ODE solvers [44], to propose a new step size. It operates under the assumption that the error is proportional to the step size, with $E \propto \Delta t^{q+1}$ for some known $q \in \mathbb{N}$; refer to Kersting et al. [65] for such convergence rates for probabilistic ODE solvers. Then, since it observed the error E for the current step size Δt, the proportional controller proposes a new step size Δt_{new} as

$$\Delta t_{\text{new}} = \Delta t \cdot \gamma \cdot \left(\frac{1}{E}\right)^{\frac{1}{q+1}}, \tag{6.18}$$

where the parameter $\gamma \in (0, 1]$ is a safety factor which decreases the step size but thereby increases the probability that the next step will be accepted. In addition, we limit the rate of change to $\eta_{\min} \leq \Delta t_{\text{new}}/\Delta t \leq \eta_{\max}$ to improve the stability of the control algorithm [44]. We follow common default choices for classic solvers, selected for example in the Julia package DifferentialEquations.jl [97], and set default parameters of $\gamma = 0.9$, $\eta_{\min} = 0.2$, $\eta_{\max} = 10$.

Remark 6.2 (Other control algorithms) The quantity E from eq. (6.17) can also be used together with other control algorithms, such as proportional-integral-control (PI-control) [41] or proportional-integral-derivative control (PID-control) [128], which might provide some improvements in the performance of the ODE solver. But, these controllers come with additional hyperparameters, and these different choices have not yet been thoroughly investigated. We therefore use the established proportional controller in this thesis and in our implementation (Publication VIII).

6.4.3 Example: Step-size adaptation on a stiff Van-der-Pol system

To visualize the effect and importance of step-size adaptation we use it to solve a stiff ODE: the Van-der-Pol system [96]. It is given by the ODE

$$\dot{y}_1 = y_2, \qquad \dot{y}_2 = \mu\big((1 - y_1^2) - y_1\big), \tag{6.19}$$

with initial value $y_0 = [2, 0]^\top$, here with a stiffness constant of $\mu = 1e3$, and we integrate the system over the time domain $t \in [0, 3.6]$. We consider a standard probabilistic ODE solver with first-order vector-field linearization (`EK1`), which is known to be A-stable [122], with an

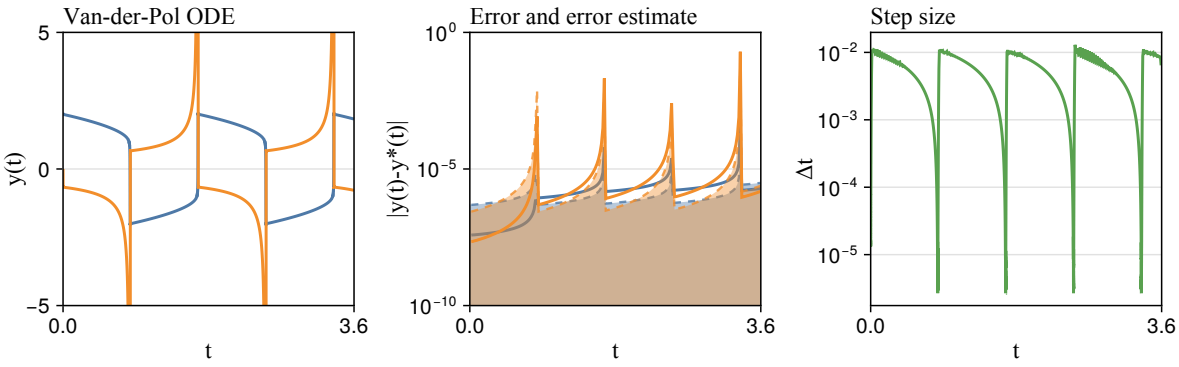


Figure 6.2: **Probabilistic numerical solution with step-size adaptation for a stiff Van-der-Pol system.** We solve the Van-der-Pol ODE (left) with the `EK1` solver with IWP(3) prior, with a locally calibrated diffusion model and with adaptive step-size selection. The resulting numerical error varies greatly (center), but the error estimate (dashed and shaded) follows the structure well and is of similar scale. We also observe that the step size of the solver decreases by multiple orders of magnitude throughout the solution process (right), indicating the necessity for very small steps for the stiffest parts of the system.

IWP(3) prior and with a locally calibrated diffusion model. We use adaptive step-size selection as described in Section 6.4, and choose absolute and relative tolerances of $\tau_{\text{abs}} = \tau_{\text{rel}} = 10^{-6}$.

Figure 6.2 shows the result. We observe that the semi-implicit, A-stable `EK1` is able to solve the system well, achieving low errors $\ll 1$. But in particular, we also see why the adaptive step-size selection is so important for this kind of problem: The numerical error differs greatly throughout the solution, with orders of magnitude larger errors at those locations where the solution has large changes. At the same time, the step-size decreases strongly from around 10^{-2} to around 10^{-5}. If we were to use the smallest selected step size for the whole integration interval, the runtime of the solver would increase by a factor of 200 from around 0.1 seconds to around 20 seconds. And in addition, choosing the step-size such that the solution is stable and does not diverge is difficult to do a-priori for stiff dynamical systems. By instead using an automatic step-size controller, we are able to produce results for various levels of accuracy, without running into numerical instabilities. Overall, this example highlights the importance of adaptive step-size control for solving stiff dynamical systems, and shows how it enables us to use available computational resources more efficiently.

6.5 Conclusion

This chapter covered the calibrated estimation and control of numerical errors. The former relies on estimating the *diffusion* parameter of the Gauss–Markov prior, which directly controls how uncertainty enters the dynamical system and therefore influences the scale of the posterior uncertainties. We have seen how the different models (scalar/diagonal) and estimation approaches (local/global) influence the posterior uncertainties returned by the probabilistic numerical solver. The latter then uses calibrated estimates of the *local* error for step-size adaptation with a proportional control algorithm. This can greatly improve the efficiency of probabilistic numerical ODE solvers in practice, especially for systems with different scales.

7 Adjusting the Information Operator to Specific Problems

This chapter is based on Publication II:

Nathanael Bosch, Filip Tronarp, and Philipp Hennig. **Pick-and-Mix Information Operators for Probabilistic ODE Solvers**. *International Conference on Artificial Intelligence and Statistics* (AISTATS), 2022.

The probabilistic solvers introduced in Chapter 4 all consider first-order ODEs of the form

$$\dot{y}(t) = f(y(t), t), \qquad t \in [0, T]. \tag{7.1}$$

In these solvers, the link between the probabilistic prior and the problem of interest is established by the *information operator* $\mathcal{Z}$, designed in such a way that it maps the true solution to the zero function; this is explained in detail in Section 4.2. For a first-order ODE as given in eq. (7.1), this information operator is defined as

$$\mathcal{Z}[y](t) = \dot{y}(t) - f(y(t), t), \tag{7.2}$$

such that by construction, we have $\mathcal{Z}[y] \equiv 0$ if and only if y corresponds to the true solution of interest. But in principle, the Gaussian filtering algorithm that drives probabilistic ODE solvers does not impose any particular form on the information operator. In this section, show how to construct information operators for other types of related numerical problems and compute the corresponding probabilistic numerical solutions. We formulate probabilistic numerical solvers for higher-order ODEs in Section 7.1, systems with conserved quantities (such as Hamiltonian systems) in Section 7.2, and differential-algebraic equations in Section 7.3, all of which have been proposed in Publication II.

7.1 Higher-order ordinary differential equations

Consider a higher-order ODE

$$\frac{\mathrm{d}^\nu y}{\mathrm{d}t^\nu} = f\left(y(t), \frac{\mathrm{d}y}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^{\nu-1}y}{\mathrm{d}t^{\nu-1}}, t\right), \tag{7.3}$$

with initial values

$$y(0) = y_{0,0}, \quad \frac{\mathrm{d}y}{\mathrm{d}t} = y_{1,0}, \quad \ldots, \quad \frac{\mathrm{d}^{\nu-1}y}{\mathrm{d}t^{\nu-1}} = y_{v-1,0}. \tag{7.4}$$

Similarly to the first-order ODE case, we can construct a suitable information operator from eq. (7.3) that maps the true solution to the zero function, as

$$\mathcal{Z}[y](t) = \frac{\mathrm{d}^\nu y}{\mathrm{d}t^\nu} - f\left(y(t), \frac{\mathrm{d}y}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^{v-1}y}{\mathrm{d}t^{v-1}}, t\right). \tag{7.5}$$

In terms of the stochastic process x that we consider in the probabilistic solvers, the information operator can be equivalently expressed as

$$\mathcal{Z}[x](t) = E_v x(t) - f(E_0 x(t), E_1 x(t), \ldots, E_{v-1} x(t), t). \tag{7.6}$$

As before, $\mathcal{Z}$ maps the true solution to the zero function, and conversely, if $\mathcal{Z}[x] \equiv 0$ holds everywhere, then x corresponds to the true solution. This requires that the prior is sufficiently smooth to evaluate the ODE vector field, for example by using an IWP(q) with $q \geq v$.

With such a suitable prior and the information operator as in eq. (7.6), and given a time grid $\{t_n\}_{n=1}^N$, the resulting discretized state estimation problem is of the form

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{7.7a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A(t_n - t_{n-1})x(t), Q(t_n - t_{n-1})), \tag{7.7b}$$

$$z_n \mid x(t_n) \sim \delta(E_v x(t_n) - f(E_0 x(t_n), \ldots, E_{v-1} x(t_n), t_n)), \tag{7.7c}$$

with zero-data $z_n \triangleq 0$ for all n. The resulting NLGSSM is almost equivalent to the one for first-order ODEs, as introduced in eq. (4.14), only with the adjusted observation model. Inference in this NLGSSM can thus be done with the same algorithms as before, in particular with the extended Kalman filter and smoother (Algorithm 2.6).

Remark 7.1 (Higher-order ODEs should not be solved as first order ODEs) Higher-order ODEs can be transformed into first-order systems by defining a new variable $\tilde{y} := (y, \dot{y}, \ldots, y^{(v-1)})$. In principle, they can therefore be solved by any generic solver for first-order ODEs, including the probabilistic ODE solver introduced in Chapter 4. But this approach has two downsides: First, the resulting ODE has a v-times larger dimension which leads to a significant increase in runtime and memory cost. This also holds for classic, non-probabilistic ODE solvers, and it motivated the development of specialized

methods for higher-order ODEs, such as Runge–Kutta–Nyström methods [87, 44]. And second, if we simply treat the transformed ODE as any other first-order ODE and use a generic q-times integrated Wiener process prior, we ignore the structure in $\tilde{y}$ and model each dimension independently, even though they are clearly related. In particular, entries that should be exactly equal will be modeled independently, as for example $\dot{y}$ appears both in $\tilde{y}$ and $\mathrm{d}\tilde{y}/\mathrm{d}t$. This leads to inconsistent results. By instead solving the higher-order ODE directly, we avoid these issues and obtain a more accurate solution, with a lower computational cost.

7.1.1 Example: Solving the Pleiades ODE

The Pleiades system describes the motion of seven stars in a plane, with coordinates (x_i, y_i) and masses $m_i = i$, $i = 1, \ldots, 7$ [44, p. II.10]. It is given by a second-order ODE

$$\ddot{x}_i = \sum_{j \neq i} m_j (x_j - x_i)/r_{ij}, \qquad \ddot{y}_i = \sum_{j \neq i} m_j (y_j - y_i)/r_{ij}, \tag{7.8}$$

where $r_{ij} = \left((x_i - x_j)^2 + (y_i - y_j)^2\right)^{3/2}$, for $i, j = 1, \ldots, 7$, on the time span $t \in [0, 3]$, with initial locations

$$x(0) = [3, 3, -1, -3, 2, -2, 2], \tag{7.9a}$$
$$y(0) = [3, -3, 2, 0, 0, -4, 4], \tag{7.9b}$$

and initial velocities

$$\dot{x}(0) = [0, 0, 0, 0, 0, 1.75, -1.5], \tag{7.9c}$$
$$\dot{y}(0) = [0, 0, 0, -1.25, 1, 0, 0]. \tag{7.9d}$$

Figure 7.1 compares the naive solvers, applied to the ODE by first transforming it to a first-order ODE, with the native second-order ODE version which uses the suitable information operator from eq. (7.6). We include both the explicit solver with zeroth-order vector-field linearization (`EK0`) and the semi-implicit method with first-order linearization (`EK1`). All methods use integrated Wiener process priors and adaptive step-size selection; the full experimental setup is explained in detail in Publication II, and the experiment code is available online[1]. We see that the methods that solve the second-order ODE directly outperform the first-order solvers, both regarding the resulting numerical error and the runtime.

[1] https://github.com/nathanaelbosch/pick-and-mix

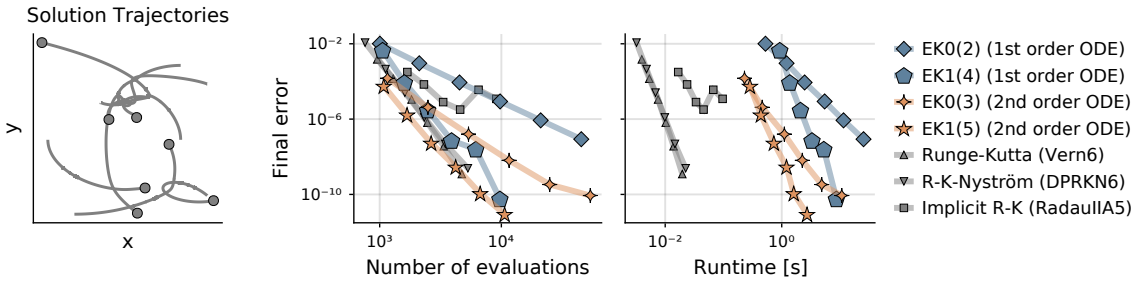


Figure 7.1: **Benchmarking probabilistic numerical solvers on the Pleiades ODE.** The Pleiades system describes the motion of seven stars in a plane (left). Solving this problem directly in second order, compared to solving the equivalent first-order ODE, improves accuracy and efficiency, both in the number of function evaluations (center) and runtime (right). Figure copied from Publication II.

7.2 Systems with conserved quantities

In this section, we demonstrate how additional knowledge about conserved quantities of the dynamical system can be provided to the probabilistic solver. Consider a first-order ODE

$$\dot{y}(t) = f(y(t), t), \tag{7.10}$$

with initial value $y(0) = y_0$, for which we know that some function $g(y(t), \dot{y}(t))$ is conserved along the solution trajectory, that is,

$$g(y(t), \dot{y}(t)) = g(y(0), \dot{y}(0)), \qquad \forall t \in [0, T]. \tag{7.11}$$

This motivates two information operators. The first information operator $\mathcal{Z}_f$ encodes the ODE vector field as before, with

$$\mathcal{Z}_f[x](t) := E_1 x(t) - f(E_0 x(t), t). \tag{7.12}$$

The second information operator $\mathcal{Z}_g$ encodes the conserved quantity with

$$\mathcal{Z}_g[x](t) := g(E_0 x(t), E_1 x(t)) - g(y_0, f(y_0, 0)). \tag{7.13}$$

We can then combine both into a single partitioned information operator

$$\mathcal{Z}[x](t) := \begin{bmatrix} \mathcal{Z}_f[x](t) \\ \mathcal{Z}_g[x](t) \end{bmatrix} = \begin{bmatrix} E_1 x(t) - f(E_0 x(t), t) \\ g(E_0 x(t), E_1 x(t)) - g(y_0, f(y_0, 0)) \end{bmatrix}. \tag{7.14}$$

Once again, $\mathcal{Z}[x] \equiv 0$ holds if and only if x corresponds to the true solution. Discretizing this information operator on a time grid $\{t_n\}_{n=1}^N$ yields the NLGSSM

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{7.15a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A(t_n - t_{n-1})x(t), Q(t_n - t_{n-1})), \tag{7.15b}$$

$$z_n \mid x(t_n) \sim \delta\left(\begin{bmatrix} E_1 x(t_n) - f(E_0 x(t_n), t_n) \\ g(E_0 x(t_n), E_1 x(t_n)) - g(y_0, f(y_0, 0)) \end{bmatrix}\right), \tag{7.15c}$$

with data $z_n \triangleq 0$ for all $n = 1, \ldots, N$. This can again be solved with the extended Kalman filter and smoother (Algorithm 2.6) to obtain a probabilistic numerical solution of the ODE.

Remark 7.2 (Hamiltonian dynamical systems) *Hamiltonian systems* are a particular class of dynamical systems of the form

$$\dot{p} = -\frac{\partial H}{\partial q}(p, q), \qquad \dot{q} = \frac{\partial H}{\partial p}(p, q), \tag{7.16}$$

where the so-called *Hamiltonian* $H : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ describes the total energy in the dynamical system. Hamiltonian systems form an important class of ODEs in the context of geometric numerical integration [43] since their solution trajectories *preserve the Hamiltonian*. That is, for a solution $(p(t), q(t))$ of such problems, the Hamiltonian $H(p(t), q(t))$ is constant, and it holds

$$H(p(t), q(t)) - H(p(0), q(0)) \equiv 0. \tag{7.17}$$

Thus, Hamiltonian systems can be treated as a special case of the conserved quantity problems discussed here and can be solved with the presented probabilistic solver.

Remark 7.3 (Sequential updating on partitioned observation models) In extended Kalman filtering, when observation models are of the form $h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \end{bmatrix}$ it can be favorable to perform two separate linearizations and updates on the two parts of the observation model instead of updating jointly on the full observation model in one step, as this can improve the linearization point and thus the accuracy of the approximate inference step; see for instance Raitoharju et al. [99, 100, 98]. In the context of probabilistic ODE solvers with conserved quantities, updating first on the ODE information and then on the conserved quantity information also relates the algorithm to "projection methods", which are a well-established class of non-probabilistic methods for solving ODEs with conserved quantities where after each step the numerical solution is projected onto the manifold of solutions that satisfy the conserved quantity, using some numerical solver for nonlinear problems [43, Section IV.4]. In Publication II, in this thesis, and in our code, we use such a sequential update scheme.

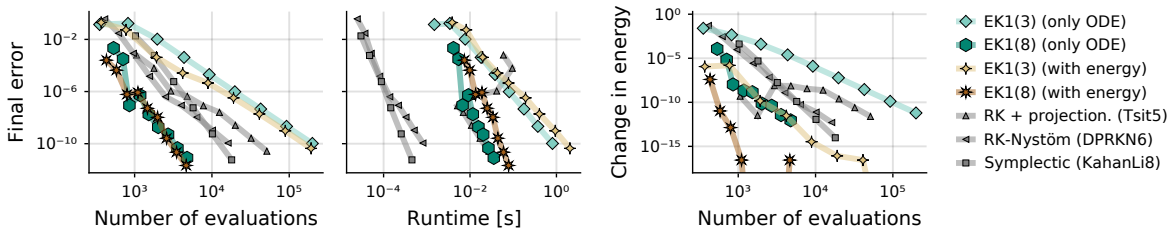


Figure 7.2: **Work-precision diagrams of numerical solvers with and without energy conservation.** Additional information about the total energy in the dynamical Hénon–Heiles system can improve the accuracy of the solution (left). This comes with additional computational cost and increases the runtime (center). But as a result, the total energy is conserved more strictly and solutions become more physically meaningful (right). Figure copied from Publication II.

7.2.1 Example: Simulating the Hénon–Heiles dynamical system

To evaluate the influence of energy conservation on probabilistic numerical solvers, we consider the Hénon–Heiles model which describes a star moving around a galactic center, with its motion restricted to a plane [50]. It is defined by a Hamiltonian

$$H(p,q) = \left[\frac{1}{2}\left(p_1^2 + p_2^2\right)\right] + \left[\frac{1}{2}\left(q_1^2 + q_2^2\right) + q_1^2 q_2 - \frac{1}{3}q_2^3\right], \tag{7.18}$$

which describes the kinetic and potential energy of the star with velocity p and location q. With $y(t) := q(t)$ and $\dot{y}(t) := p(t)$, we can write the problem as a second-order ODE

$$\ddot{y}_1(t) = -y_1(t) - 2y_1(t)y_2(t), \tag{7.19a}$$
$$\ddot{y}_2(t) = y_2^2(t) - y_2(t) - y_1^2(t), \tag{7.19b}$$

and by conservation of the Hamiltonian it further holds [43]

$$g(\dot{y}(t), y(t)) := H(\dot{y}(t), y(t)) - H(\dot{y}_0(t), y_0(t)) = 0. \tag{7.20}$$

We consider initial values $y(0) = [0, 0.1]$, $\dot{y}(0) = [0.5, 0]$, and integrate the system over the time domain $t \in [0, 1000]$.

We solve the system with the semi-implicit ODE filter with first-order linearization (`EK1`), both with and without the additional conservation of energy encoded into the information operator. We evaluate the solvers with both an IWP(3) and IWP(8) prior, with adaptive steps, and for a range of tolerances. We also compare the probabilistic solvers to multiple non-probabilistic alternatives, namely a projection method combined with the `Tsit5` Runge–Kutta solver [43, 124], the `DPRKN6` Runge–Kutta Nyström method [32], and the symplectic `KahanLi8` solver [60], all provided by the DifferentialEquations.jl package [97]. The full experimental setup is explained in detail in Publication II; the experiment code is available online[2].

[2] https://github.com/nathanaelbosch/pick-and-mix

Figure 7.2 shows the results in work-precision diagrams. We observe that probabilistic solvers with energy conservation compute, in some configurations, a more accurate solution than regular probabilistic solvers, but the additional update step also leads to an increase in absolute runtime. However, the probabilistic solutions with energy conservation enforce this conservation very strictly, even in comparison to the specialized non-probabilistic approaches, and achieved the smallest changes in energy in this experiment. Thus, for problems in which conserved quantities are known and their conservation is important, probabilistic solvers can provide a valuable tool to enforce these constraints accurately to obtain physically meaningful solutions. For a more thorough experimental evaluation, refer to Publication II.

7.3 Differential-algebraic equations

Differential-algebraic equations (DAEs) are a generalization of ODEs in which equations can not only describe the evolution of the state, but they can also contain algebraic constraints. DAEs arise naturally in many dynamical systems, such as multi-body dynamics, chemical kinetics, or optimal control [17], In comparison to the systems with conserved quantities discussed in Section 7.2, these algebraic constraints do not over-specify the problem but they are a required part of the dynamical system. This makes their numerical simulation notoriously challenging, and thus DAEs often require specialized methods [94]. Here we show how the probabilistic solver can be naturally adapted to handle DAEs, by defining an appropriate information operator that maps the true solution to the zero function.

We consider DAEs in so-called *mass-matrix form*, given by

$$M\dot{y}(t) = f(y(t)), \qquad \forall t \in [0, T], \tag{7.21}$$

with vector field $f : \mathbb{R}^d \to \mathbb{R}^d$, initial values $y(0) = y_0$, and mass matrix $M \in \mathbb{R}^{d\times d}$. If M is invertible, this system can be rewritten as a regular ODE by multiplying the equation with M^{-1} from the left. But if M is singular, the system is a DAE.

To solve this system with the probabilistic solver, we simply define a suitable information operator

$$\mathcal{Z}[x](t) = ME_1x(t) - f(E_0x(t)). \tag{7.22}$$

As before, $\mathcal{Z}[x] \equiv 0$ if and only if x corresponds to the true solution. Discretizing it on a time grid $\{t_n\}_{n=1}^N$ yields an NLGSSM

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{7.23a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A(t_n - t_{n-1})x(t), Q(t_n - t_{n-1})), \tag{7.23b}$$

$$z_n \mid x(t_n) \sim \delta(ME_1x(t_n) - f(E_0x(t_n))), \tag{7.23c}$$

with data $z_n \triangleq 0$ for all $n = 1, \ldots, N$. This can again be solved with the extended Kalman filter and smoother (Algorithm 2.6) to obtain a probabilistic numerical solution to the DAE.

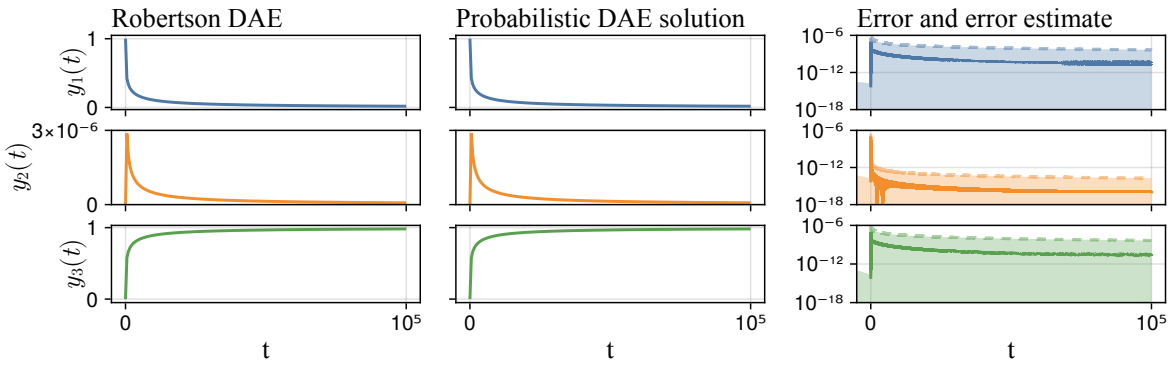


Figure 7.3: **Probabilistic numerical solution of the Robertson DAE.** The reference solution (left) and the probabilistic numerical solution (center) to the Robertson DAE seem very similar, indicating that the probabilistic solver produced an accurate solution estimate. The right figure visualizes the absolute numerical error (with respect to the reference solution) together with the 95% credible interval of the numerical error estimate produced by the probabilistic solver (shaded area). We observe that the error estimate decays similarly to the actual numerical error. But, the error estimate is orders of magnitude larger, indicating that the probabilistic numerical solver is underconfident in this experiment.

7.3.1 Example: Solving the Robertson DAE

The Robertson DAE describes a system of chemical reactions and is commonly used to evaluate stiff ODE and DAE solvers [45]. As a DAE, it is given by the equations

$$y_1(t) = -0.04y_1(t) + 10^4 y_2(t)y_3(t), \tag{7.24a}$$
$$y_2(t) = 0.04y_1(t) + 10^4 y_2(t)y_3(t) - (3 \cdot 10^7)y_2(t)^2, \tag{7.24b}$$
$$0 = y_1(t) + y_2(t) + y_3(t) - 1, \tag{7.24c}$$

and it therefore has a singular mass matrix of the form

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

We consider an initial value $y(0) = [1, 0, 0]$, and time span $t \in [0, 10^5]$. We solve the DAE with the semi-implicit ODE filter with first-order linearization (`EK1`), with its information operator as specified in eq. (7.22), an IWP(1) prior, and adaptive step-size selection with absolute and relative tolerances of $\tau_{\text{abs}} = 10^{-6}$ and $\tau_{\text{rel}} = 10^{-3}$.

Figure 7.3 shows the result. We see that the probabilistic numerical solution looks very similar to the numerical solution, computed with the implicit Runge–Kutta method `RadauIIA5`, provided by the DifferentialEquations.jl Julia package [97, 45]. Upon closer inspection, we see that the errors for the individual dimensions are in the range of 10^{-18} to 10^{-6}, and the corresponding error estimates here are some orders of magnitudes larger. The solvers are therefore underconfident in this experiment. But, they are able to solve the DAE of interest

with high accuracy. For a more thorough evaluation of the probabilistic solvers on DAEs, see Publication II.

7.4 Conclusion

This chapter extended the filtering-based probabilistic numerical ODE solver to higher-order ODEs, systems with conserved quantities, and differential-algebraic equations, by simply adjusting the information operator. Together with recent developments on probabilistic numerics for boundary-value problems [68] and partial differential equations [70], this establishes ODE filtering as a practical framework for solving a wide range of numerical problems related to differential equations.

The presented adjustments of the information operator are, in principle, also fully compatible with most other contributions of this thesis: For all of these problem types, we can consider and estimate various diffusion models and adaptively select step-sizes (Chapter 6), we can use structured priors and approximate linearization to obtain linear scaling with the ODE dimension (Chapter 5), we can solve these problems parallel-in-time (Chapter 9), and we can perform numerical-error-aware ODE parameter inference (Chapter 10). We can also add conserved quantities to the probabilistic exponential integrators which we introduce next in Chapter 8.

8 Probabilistic Exponential Integrators

This chapter is based on Publication V:

Nathanael Bosch, Philipp Hennig, and Filip Tronarp. **Probabilistic Exponential Integrators**. *Conference on Neural Information Processing Systems* (NeurIPS), 2023.

The probabilistic numerical ODE solvers established in Chapter 4 include both explicit and (semi-)implicit methods, and the latter have been shown to be suitable for stiff problems. This is due to their *A-stability*: A-stability essentially guarantees that the numerical solution of a decaying ODE will also decay, independently of the chosen step size. This prevents the numerical solution from blowing up, and thereby enables these methods to use larger step sizes than explicit methods that are not A-stable, which often leads to a significant speed-up on stiff problems in practice. But other stronger stability notions exist, one example of which is *L-stability*: It guarantees that the numerical solution not only decays, but it decays *fast*, i.e. it goes to zero as the step size goes to infinity. This is a strictly stronger stability property, and L-stable methods can be even more effective on stiff problems than A-stable methods.

In non-probabilistic numerical analysis, *exponential integrators* are a class of L-stable numerical methods for efficiently solving large stiff ODEs [125, 54, 29, 56]. They are based on the observation that, if the ODE has a semi-linear structure

$$\dot{y}(t) = f(y(t), t) = Ly(t) + N(y(t), t), \tag{8.1}$$

then the solution at time $t + h$ can be expressed as

$$y(t+h) = \exp(Lh)y(t) + \sum_{k=0}^{\infty} h^{k+1}\varphi_{k+1}(Lh)\frac{\mathrm{d}^k}{\mathrm{d}t^k}N(y(t), t)t, \tag{8.2}$$

where $\varphi_k(z) = \int_0^1 \exp(z(1-\tau))\frac{\tau^{k-1}}{(k-1)!}\mathrm{d}\tau$ are the so-called φ-functions [56]. That is, the solution can be decomposed into the exact solution of the linear part of the ODE, and a series of higher-order terms that depend on the nonlinear part of the ODE. And in particular, only the latter needs to be numerically approximated as the linear part can be solved exactly (with the matrix exponential). This is the main idea behind exponential integrators.

Probabilistic exponential integrators follow the same idea. In ODE filtering, "solving the linear part of the ODE exactly" corresponds to choosing a suitable prior model, designed in such a way that it models the linear part of the ODE exactly. In the following, we briefly review the *probabilistic exponential integrators* proposed in Publication V.

8.1 The integrated Ornstein–Uhlenbeck process

The q-times integrated Wiener process prior that we typically use in the probabilistic ODE solvers (see Example 4.1) models the q-th derivative of the ODE solution $y(t)$ with a Wiener process. Here, we follow a similar motivation but only for the non-linear part N of the ODE.

First, differentiating both sides of the semi-linear ODE given in eq. (8.1) $q-1$ times with respect to t yields

$$\frac{\mathrm{d}^q}{\mathrm{d}t^q} y(t) = L \frac{\mathrm{d}^{q-1}}{\mathrm{d}t^{q-1}} y(t) + \frac{\mathrm{d}^{q-1}}{\mathrm{d}t^{q-1}} N(y(t), t). \tag{8.3}$$

Then, modeling $\frac{\mathrm{d}^{q-1}}{\mathrm{d}t^{q-1}} N(y(t), t)$ as a Wiener process and relating the result to $y(t)$ gives

$$\mathrm{d}y^{(i)}(t) = y^{(i+1)}(t)\mathrm{d}t, \tag{8.4a}$$
$$\mathrm{d}y^{(q)}(t) = Ly^{(q)}(t)\mathrm{d}t + \sigma I_d \mathrm{d}w^{(q)}(t). \tag{8.4b}$$

This process is also known as the q-times integrated Ornstein–Uhlenbeck process (IOUP), with rate parameter L and diffusion parameter σ. It can be equivalently stated with the previously introduced notation from eq. (3.10) as the output of an LTI-SDE

$$\mathrm{d}x(t) = F_{\mathrm{IOUP}(d,q)} x(t)\mathrm{d}t + \Gamma^{1/2}_{\mathrm{IOUP}(d,q)} \mathrm{d}w(t), \tag{8.5a}$$
$$y^{(i)}(t) = E_i x(t), \qquad i = 0, \ldots, q, \tag{8.5b}$$

where, as before, the state $x(t)$ is of the form $x(t) = \begin{bmatrix} y(t) & y^{(1)}(t) & \ldots & y^{(q)}(t) \end{bmatrix}^\top$ and E_i are the corresponding selection matrices. The system matrices of the IOUP are given by

$$F_{\mathrm{IOUP}(d,q)} = \begin{bmatrix} 0 & I_d & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_d \\ 0 & 0 & \cdots & L \end{bmatrix}, \qquad \Gamma^{1/2}_{\mathrm{IOUP}(d,q)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I_d \end{bmatrix}. \tag{8.6}$$

The initial distribution $x(0) \sim \mathcal{N}(\mu_0, \Sigma_0)$ is chosen as usual such that it models the initial value and the initial derivatives correctly, i.e. $\mu_0 = \begin{bmatrix} y_0 & \dot{y}_0 & \ldots & y_0^{(q-1)} & y_0^{(q)} \end{bmatrix}^\top$ and $\Sigma_0 = 0$.

Remark 8.1 (The mean of the IOUP process solves the linear part of the ODE exactly) By taking the expectation of eq. (8.4b) and by linearity of integration, we can see that the mean of the IOUP satisfies

$$\dot{\mu}^{(0)}(t) = L\mu^{(0)}(t), \qquad \mu^{(0)}(0) = y_0. \tag{8.7}$$

This is in line with the motivation of exponential integrators: the linear part of the ODE is solved exactly, and we only need to approximate the nonlinear part. Figure 8.1 visualizes this idea.

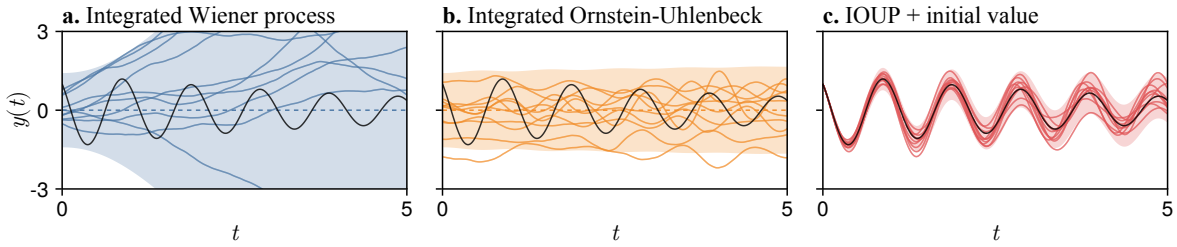


Figure 8.1: **Damped oscillator dynamics and priors with different degrees of encoded information in the prior**: *Left*: Once-integrated Wiener process, a popular prior for probabilistic ODE solvers. *Center*: Once-integrated Ornstein–Uhlenbeck process (IOUP) with rate parameter chosen to encode the known linearity of the ODE. *Right*: IOUP with both the ODE information and a specified initial value and derivative. This is the kind of prior used in the probabilistic exponential integrator. Figure copied from Publication V.

8.2 The information operator and how to linearize it

The information operator of the probabilistic exponential integrator is defined exactly as for generic probabilistic ODE solvers, presented in Section 4.2, as

$$\mathcal{Z}[x](t) = E_1 x(t) - \underbrace{(LE_0 x(t) + N(E_0 x(t), t))}_{f(E_0 x(t), t)} \tag{8.8}$$

where we have used the semi-linear structure of the ODE vector field. The linearization of this information operator can then be done in the same way as for the other probabilistic ODE solvers, with either a first- or zeroth-order Taylor approximation of the vector field, leading to a semi-implicit or explicit solver, respectively (`EK1` and `EK0`). But in this specific case we also have an additional option: Since the linear part of the ODE is known, we only need to linearize the nonlinear part. And if we perform a zeroth-order Taylor approximation of N, we essentially obtain a new approximate Jacobian of the vector field $J_f \approx L$. This leads to a third linearization option for the probabilistic exponential integrator, which we refer to here as the `EKL`; this is also the version used in Publication V.

8.3 The resulting inference problem

Summarizing the above and discretizing the information operator on a time grid $\{t_n\}_{n=1}^N$, we obtain the following NLGSSM for the probabilistic exponential integrator:

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{8.9a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}(A_{\text{IOUP}}(h_n)x(t), Q_{\text{IOUP}}(h_n)), \tag{8.9b}$$

$$z_n \mid x(t_n) \sim \delta(E_1 x(t_n) - LE_0 x(t_n) - N(E_0 x(t_n), t_n)), \tag{8.9c}$$

with data $z_n \triangleq 0$ for all n. This NLGSSM can be solved with the extended Kalman filter and smoother Algorithm 2.6 to obtain a probabilistic numerical solution to the ODE. This yields the *probabilistic exponential integrator* proposed in Publication V.

We conclude the chapter with a few remarks on the properties of this solver, an extension to exponential Rosenbrock-type methods, and an example.

8.4 Properties of the probabilistic exponential integrator

Proposition 8.1 (L-stability) *The probabilistic exponential integrator is L-stable.*

The property essentially follows from Remark 8.1 which stated that the IOUP solves linear ODEs exactly, which directly implies fast decay and this L-stability. The full proof and more discussion can be found in Publication V.

Proposition 8.2 (Equivalence to the PEC exponential trapezoidal rule) *The mean estimate of the probabilistic exponential integrator with once-integrated Ornstein–Uhlenbeck prior with rate parameter L is equivalent to the exponential trapezoidal rule in predict-evaluate-correct (PEC) mode, with the predictor being the exponential Euler method [44, 57].*

This result provides a link between the probabilistic exponential integrator and an established classic exponential integrator; a similar statement for the non-probabilistic case is also provided by Schober et al. [112, Proposition 1]. The full proposition and its proof can be found as Proposition 2 in Publication V.

8.5 Probabilistic exponential Rosenbrock-type methods

Finally, we apply the idea of probabilistic exponential integrators to problems that are not semi-linear, by performing local linearization in a Rosenbrock-type approach [55, 57, 80]. Given a nonlinear ODE of the form $\dot{y}(t) = f(y(t), t)$, (non-probabilistic) exponential Rosenbrock methods perform a continuous linearization of the right-hand side f around the numerical ODE solution and essentially solve a sequence of IVPs

$$\dot{y}(t) = J_n y(t) + (f(y(t), t) - J_n y(t)), \qquad t \in [t_n, t_{n+1}], \tag{8.10a}$$

$$y(t_n) = y_n, \tag{8.10b}$$

where J_n is the Jacobian of f at the numerical solution estimate $\hat{y}(t_n)$. Then, since each subproblem is semi-linear, it can be solved with an exponential integrator. The benefits of this approach are that it allows the use of L-stable exponential integrators even if the ODE is not semi-linear, but in particular it leads to a more accurate linearization of the vector field, and can thus result in more efficient solvers than globally linearized counterparts [57].

This idea can also be applied in a probabilistic setting: By linearizing the ODE right-hand side f at each step of the solver around the filtering mean $E_0\mu_n$, we (locally) obtain a semi-linear problem for which we can choose a suitable IOUP prior and which we can solve with the probabilistic exponential integrator. This results in *probabilistic exponential Rosenbrock-type*

methods. As before, the linearization of the information operator could in principle be done with various approximate Jacobians, but as the local linearization of the ODE relies on exact vector-field Jacobians, we suggest in Publication V to linearize the information operator with the exact vector-field Jacobian, too.

8.6 Example: Solving the discretized Burgers' PDE

We briefly show the utility of the developed methods on a semi-linear stiff dynamical system, namely the Burgers' partial differential equation (PDE). It is given by a PDE

$$\partial_t u(x,t) = D\partial_x^2 u(x,t) - u(x,t)\partial_x u(x,t), \qquad x \in [0,1], \quad t \in [0,1], \tag{8.11}$$

here with diffusion coefficient $D = 0.075$. We consider zero-Dirichlet boundary conditions, that is, $u(0,t) = u(1,t) = 0$, and an initial condition $u(x,0) = \sin(3\pi x)^3(1-x)^{3/2}$. We transform the problem into a semi-linear ODE with the method of lines [82, 109]: By discretizing the spatial domain on $N = 250$ equidistant points and approximating the differential operators with finite differences, we obtain a semi-linear ODE of the form

$$\dot{y}(t) = D \cdot L \cdot y(t) + F(y(t)), \qquad t \in [0,T], \tag{8.12}$$

with N-dimensional $y(t) \in \mathbb{R}^N$, $L \in \mathbb{R}^{N\times N}$ the finite difference approximation of the Laplace operator ∂_x^2 of the form

$$[L]_{ij} = \frac{1}{\Delta x^2} \cdot \begin{cases} -2 & \text{if } i = j, \\ 1 & \text{if } i = j \pm 1, \\ 0 & \text{otherwise,} \end{cases} \tag{8.13}$$

where $\Delta x = 1/N$, and nonlinear part F resulting from another finite-difference approximation of the term $u \cdot \partial_x u$, as

$$[F(y)]_i = \frac{1}{4\Delta x} \begin{cases} y_2^2 & \text{if } i = 1, \\ y_{d-1}^2 & \text{if } i = d, \\ y_{i+1}^2 - y_{i-1}^2 & \text{else.} \end{cases} \tag{8.14}$$

We compare the proposed probabilistic exponential integrator with approximate vector-field Jacobian consisting only of the linear part of the ODE $J_f \approx DL$ (`EKL & IOUP(2)`), the exponential Rosenbrock-type method with local re-linearization (`EK1 & IOUP(2) (RB)`), and the standard probabilistic ODE solvers with integrated Wiener process prior, both in the explicit version with zero Jacobian $J_f \approx 0$ (`EK0 & IWP(2)`), the semi-implicit version with exact Jacobian J_f (`EK1 & IWP(2)`), and a version with using the linear part of the ODE as Jacobian $J_f \approx DL$ (`EKL & IWP(2)`). Figure 8.2 shows the results in work-precision diagrams. We see that for the same step sizes, both exponential methods achieve a lower final error than the non-exponential solvers. And in particular for high step sizes, the exponential solvers still compute accurate solutions with errors $\ll 1$, whereas the solution of the non-exponential

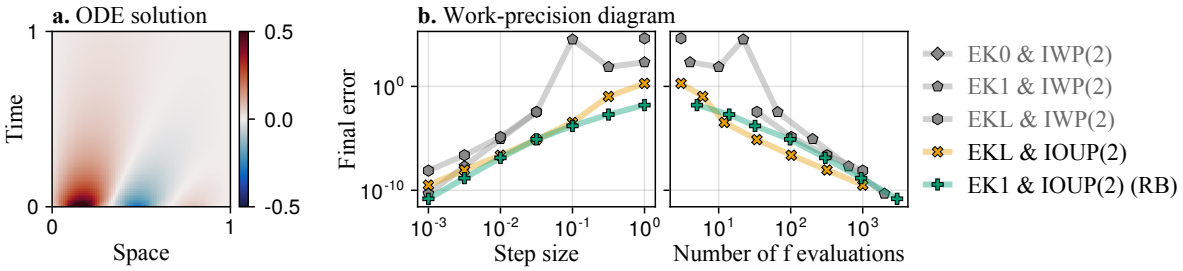


Figure 8.2: **Benchmarking probabilistic exponential integrators on Burgers' equation.** Both exponential integrators with IOUP prior achieve lower errors than the existing IWP-based solvers, in particular for large steps. This indicates their stronger stability properties. Figure copied from Publication V.

solvers diverges strongly from the true solution. We also see that the re-linearization of the exponential Rosenbrock-type method comes with increased numbers of evaluations of the vector-field f. Thus, for expensive-to-evaluate vector fields, the standard probabilistic exponential integrator might be preferable. A more thorough experimental evaluation can be found in Publication V.

8.7 Conclusion

This chapter presented probabilistic exponential integrators, a modification of filtering-based probabilistic numerical ODE solvers for stiff semi-linear ODEs. By incorporating the fast, linear dynamics directly into the prior of the solver, the method essentially solves the linear part exactly, in a similar manner as classic exponential integrators. We also extended the proposed method to general non-linear systems via iterative re-linearization and presented probabilistic exponential Rosenbrock-type methods. Both methods have been shown both theoretically and empirically to be more stable than their non-exponential probabilistic counterparts.

The probabilistic exponential integrators presented in this chapter are conceptually compatible with some, but not all other contributions of this thesis: We can encode additional information such as conserved quantities by adding additional information operators (Chapter 7), and we can use the integrated Ornstein–Uhlenbeck process priors within the parallel-in-time method (Chapter 9). But, adaptive step-size selection can be costly as we need to re-compute the matrix exponential at each step (Chapter 6), and since the linear part of the ODE is often not sufficiently structured we cannot formulate the probabilistic exponential integrators in linear time with the ODE dimension (Chapter 5). Nevertheless, probabilistic exponential integrators provide a powerful tool for stiff ODEs and are a valuable addition to the probabilistic numerics toolbox.

9 Parallel-in-time Probabilistic Numerical ODE Solvers

This chapter is based on Publication VII:

Nathanael Bosch, Adrien Corenflos, Fatemeh Yaghoobi, Filip Tronarp, Philipp Hennig, and Simo Särkkä. **Parallel-in-time Probabilistic Numerical ODE Solvers**. *Journal of Machine Learning Research* (JMLR), 2024.

The probabilistic numerical ODE solvers that we have established in the previous chapters all operate sequentially in time. Therefore, their computational complexity scales linearly with the number of steps taken by the solver. This is also the case for all non-probabilistic methods mentioned so far, such as Runge–Kutta methods. But alternative approaches with better scaling have been proposed: So-called *parallel-in-time* methods leverage the ever-increasing parallelization capabilities of modern computer hardware to achieve sub-linear scaling in the number of time steps [38]. To give one well-established example, `Parareal` [77] achieves this speed-up by splitting the integration interval into multiple smaller sub-problems, and then iteratively simulating and aggregating the sub-problems in a predictor-corrector loop. This development has sparked a number of subsequent research, including also probabilistic extensions of `Parareal` [92, 93, 39]. In Publication VII, we proposed a novel, probabilistic numerical ODE solver with sub-linear scaling that operates parallel-in-time, based on the ODE filtering framework.

ODE filters as introduced in Chapter 4 operate in the following way: First, we formulate a continuous prior model and a discrete observation model. Then, we discretize the model to obtain a nonlinear Gaussian state-space model (NLGSSM). Finally, we perform inference in the NLGSSM with extended Kalman filtering and smoothing (Algorithm 2.6). The main ingredient to develop the parallel-in-time probabilistic ODE solver proposed in Publication VII is to use a different inference algorithm for NLGSSMs, which operates in a time-parallel manner and which has a logarithmic complexity in the number of time steps: the *time-parallel iterated extended Kalman smoother* [130, 131].

9.1 Time-parallel iterated extended Kalman smoothing

The iterated extended Kalman smoother, which we introduced in Algorithm 2.7 but did not yet use to solve ODEs in this thesis, operates by iterating two steps: (i) linearizing the whole nonlinear Gaussian state-space model along a sequence of linearization points, and (ii) solving

the linearized model with a standard Rauch–Tung–Striebel smoother (Algorithm 2.4). The first step can be done fully in parallel, as the linearization at each step only depends on the local linearization point. But the second part has previously been considered to be sequential.

Recently, Särkkä et al. [106] introduced time-parallel Bayesian filters and smoothers which solve LGSSMs in logarithmic time. In a nutshell, it can be shown that by constructing appropriate elements $a_{1:N}$ and an associative operator $\otimes_f$, the filtering distributions can be formulated as a cumulative sum

$$a_1 \otimes_f a_2 \otimes_f \cdots \otimes_f a_n = \begin{bmatrix} p(x_n \mid z_{1:n}) \\ p(z_{1:n}) \end{bmatrix}, \tag{9.1}$$

and similarly with suitable elements $b_{1:N}$ and an associative operator $\otimes_s$, the smoothing distributions can also be formulated (backwards) as a cumulative sum

$$b_n \otimes_s b_{n+1} \otimes_s \cdots \otimes_s b_N = p(x_n \mid z_{1:N}). \tag{9.2}$$

See Särkkä et al. [106, Theorems 3 and 6]. These quantities can then be computed *in parallel* with so-called prefix-sum algorithms, such as the parallel scan algorithm by Blelloch [10].

Remark 9.1 (On Prefix-Sums) Prefix sums, also known as cumulative sums or inclusive scans, play an important role in parallel computing as their computation can be efficiently parallelized. If enough parallel resources are available, their runtime can be reduced from *linear* to *logarithmic* in the number of elements. One such algorithm is the well-known parallel scan algorithm by Blelloch [10], which, given N elements and $N/2$ processors, computes the prefix sum in $2\lceil \log_2 N \rceil$ sequential steps with $2N - 2$ invocations of the binary operation. This algorithm is implemented in popular machine learning frameworks such as tensorflow [83] and JAX [16] and is also available in Julia via CUDA.jl [7, 6].

For linear Gaussian state-space models, these filtering and smoothing elements and operators can be analytically derived and implemented in practice. We refer to Särkkä et al. [106] for a detailed description of the time-parallel linear Gaussian filter and smoother, and to Yaghoobi et al. [131] for the corresponding numerically stable formulation with square-root parameterized covariance matrices; all details can also be found in Publication VII. By then plugging the time-parallel method into the iterated extended Kalman smoother (Algorithm 2.7) we obtain a *time-parallel* iterated extended Kalman smoother.

What remains is to specify how the initial trajectory of linearization points is chosen. In a standard, sequential IEKS, the initial trajectory is typically computed with a sequential EKS (Algorithm 2.6), but here this would break the time-parallel nature of the resulting method. Instead, we choose a constant initial trajectory at the initial mean of the state-space model, that is, $\xi_i = \mu_0$ for all i. This does not require any sequential processing and we found it to be a sufficiently good initialization in practice; see Publication VII.

9.2 The parallel-in-time probabilistic numerical ODE solver

To obtain the parallel-in-time probabilistic numerical ODE solver, we simply replace the EKS in the probabilistic ODE solver (Algorithm 4.1) with the time-parallel IEKS.

Algorithm 9.1 (Parallel-in-time probabilistic numerical ODE solver) Given

- an ODE-IVP with vector field f, initial value y_0, and time interval $[0, T]$,
- a time discretization $\{t_n\}_{n=1}^N \subset [0, T]$,
- a Gauss–Markov prior with drift, dispersion, and projection matrices $F, \Gamma^{1/2}, E_0, E_1$,

compute the probabilistic numerical ODE solution with the following steps:

1. Discretize the continuous-time prior to obtain an NLGSSM.
2. Solve the NLGSSM with the *time-parallel iterated extended Kalman smoother*

Return all desired quantities, such as the posterior marginals or the full posterior.

This algorithm is conceptually very similar to the probabilistic solver presented in Algorithm 4.1, with the only difference being that the filtering and smoothing method used here is the time-parallel iterated extended Kalman smoother. Therefore the algorithm is also compatible most of the extensions presented in this thesis, with the notable exception of the adaptive step-size selection method from Section 6.4 which relies on a sequential solver formulation. For a more detailed description of the parallel-in-time probabilistic numerical ODE solver algorithm, we refer to Publication VII.

9.3 Computational complexity and parallel speed-up

To evaluate the computational complexity of the resulting algorithm and compare it to the cost of the sequential method that we used in previous chapters, let $C^s_{\text{KS-step}}$ be the summed costs of a predict, update, and smoothing step in the sequential Kalman smoother, and let $C^p_{\text{KS-step}}$ be the corresponding cost in the time-parallel formulation (which differs by a constant factor, that is $C^p_{\text{KS-step}} \propto C^s_{\text{KS-step}}$). Let $C_{\text{linearize}}$ be the cost of linearizing the vector field at a single time point. Then, on a grid with N time points, the cost of a standard sequential extended Kalman smoother is linear in N, with

$$C^s_{\text{EKS}} = N \cdot C_{\text{linearize}} + N \cdot C^s_{\text{KS-step}}, \tag{9.3}$$

whereas the cost of the time-parallel IEKS is logarithmic in N, with

$$C^p_{\text{IEKS}} = k \cdot \Big(C_{\text{linearize}} + 2 \log_2(N) \cdot \Big(C^p_{\text{KS-step}} \Big) \Big), \tag{9.4}$$

where k is the number of IEKS iterations (assuming a sufficient number of available cores larger than N to fully parallelize the algorithm). Comparing C^s_{EKS} and C^p_{IEKS}, we clearly see the improved computational complexity in the number of time points of the parallel-in-time probabilistic ODE solver.

The *speedup* of the time-parallel IEKS over the sequential EKS, defined as $S = C^s_{\text{EKS}}/C^p_{\text{IEKS}}$ can be shown to be bounded by

$$S \leq \min\left\{ \frac{N}{k} \cdot \frac{C_{\text{linearize}} + C^s_{\text{KS-step}}}{C_{\text{linearize}}}, \frac{N}{2k\log_2(N)} \cdot \frac{C_{\text{linearize}} + C^s_{\text{KS-step}}}{C^p_{\text{KS-step}}} \right\}. \tag{9.5}$$

This shows two different regimes: If the cost of linearizing the vector field is large compared to the cost of the Kalman smoother, then the benefit of parallelizing the linearization step dominates and the speedup is bounded by $S \lesssim \frac{N}{k}$. On the other hand, if the cost of linearizing the vector field is small compared to the cost of the Kalman smoother, then the speedup is approximately bounded by $S \lesssim \frac{N}{2k\log_2 N}$ (since $C^s_{\text{KS-step}} \approx C^p_{\text{KS-step}}$). In both cases, the speedup increases with the number of time points N and decreases with the number of iterations k. And in particular, for the speedup to be significant in both scenarios, we need a low number of iterations $k \ll \frac{N}{\log_2 N}$. We experimentally evaluate these results in the next section.

9.4 Example: Runtimes and parallel scaling of the parallel-in-time solver

To evaluate the runtimes and parallel scaling of the parallel-in-time probabilistic numerical ODE solver, we consider a simple logistic ODE

$$\dot{y}(t) = y(t)(1 - y(t)), \qquad t \in [0, 10], \tag{9.6}$$

with initial value $y(0) = 0.01$. We solve the problem for a range of solvers. We compare the parallel-in-time method based on the time-parallel IEKS (`ParaIEKS`) to the established probabilistic solver based on sequential filtering and smoothing (`EKS`), as well as to a classic explicit Runge–Kutta method (`Dopri5`) [31, 115] and an implicit Runge–Kutta method (`Kvaerno5`) [73]. We evaluate the methods on a range of grid sizes, resulting from time discretizations with step sizes $\Delta t = 2^0, 2^{-1}, \ldots, 2^{-14}$, as well as for multiple GPUs with varying numbers of CUDA cores.

Figure 9.1 shows the results. As expected, we observe the linear scaling of the sequential methods and the logarithmic scaling of the parallel-in-time method for grid sizes up to around $5 \cdot 10^3$. For grid sizes larger than the available number of CUDA cores, the runtime grows linearly. We also observe that the runtimes of all the sequential method does not noticeably change with the number of available CUDA cores and stays relatively constant throughout different GPU generations. On the other hand, the parallel-in-time method benefits from improved GPU hardware, its runtime decreases as the number of CUDA cores increases, and we observe speed-ups of up to an order of magnitude.

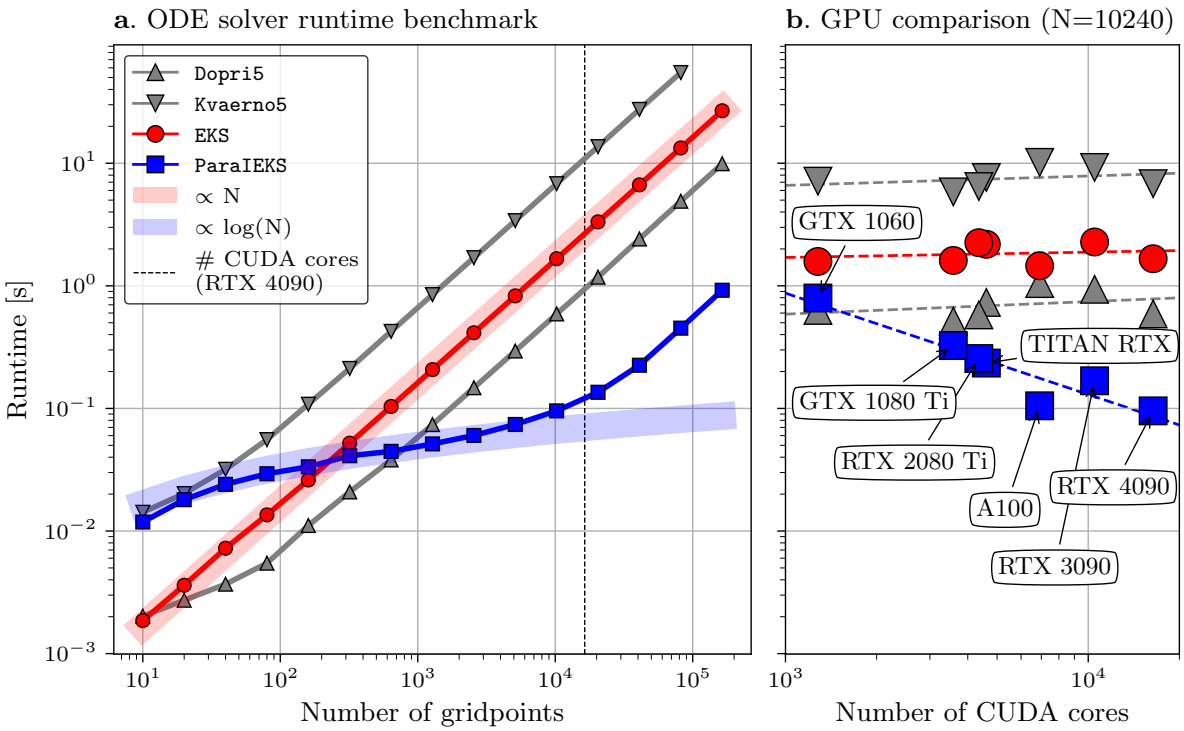


Figure 9.1: **Evolution of ODE solver runtimes with respect to problem size and hardware.** The time-parallel probabilistic ODE solver (`ParaIEKS`) shows logarithmic scaling in the grid size, whereas the sequential ODE filter (`EKS`) and the classic `Dopri5` and `Kvaerno5` solvers show the expected linear runtime complexity (left). The sequential methods also do not show relevant changes in runtime for GPUs with more CUDA cores, whereas the runtime of `ParaIEKS` improves (right). Figure copied from Publication VII.

9.5 Conclusion

In this chapter, we have developed a parallel-in-time probabilistic numerical ODE solver based on the ODE filtering framework presented in previous chapters. The solver uses the time-parallel iterated extended Kalman smoother to solve the inference problem in a logarithmic number of time steps, and it thereby achieves significant speed-ups over the sequential solver.

The parallel-in-time probabilistic ODE solver is conceptually compatible with most of the extensions presented in this thesis and it can be used to solve a wide range of ODE problems. One notable exception is the adaptive step-size selection method from Section 6.4 which relies on a sequential solver formulation. Instead a global step-size selection method may be more appropriate, such as for example the mesh-refinement method proposed by Krämer et al. [68]. Overall, the parallel-in-time probabilistic ODE solver is another valuable addition to the probabilistic numerics toolbox and it opens up a range of new possibilities for efficient ODE solving on modern parallel computer hardware.

10 Parameter Inference with ODE Filters

This chapter is based on Publication IV and Publication VI:

Filip Tronarp*, Nathanael Bosch*, Philipp Hennig. **Fenrir: Physics-Enhanced Regression for Initial Value Problems**. *International Conference on Machine Learning* (ICML), 2022.

Jonas Beck, Nathanael Bosch, Michael Deistler, Kyra L. Kadhim, Jakob H. Macke, Philipp Hennig, and Philipp Berens. **Diffusion Tempering Improves Parameter Estimation with Probabilistic Integrators for Ordinary Differential Equations**. *International Conference on Machine Learning* (ICML), 2024.

Up until now we were always concerned with the simulation of a known dynamical system. In this chapter, we instead consider the so-called *inverse* problem and we aim to estimate (parameters of) the dynamical system itself from observations of the system trajectory.

Consider an initial value problem, given by an ODE

$$\dot{y}_\theta(t) = f_\theta(y_\theta(t), t), \qquad t \in [0, T], \tag{10.1}$$

with vector field $f_\theta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial value $y_\theta(0) = y_{\theta,0} \in \mathbb{R}^d$, both of which are parameterized by some parameter $\theta \in \mathbb{R}^K$. In addition, we observe the solution trajectory y_θ on a set of discrete points in time $\{t_i'\}_{i=1}^M \subset [0, T]$ through a linear Gaussian observation model

$$u_i \mid y_\theta(t_i') \sim \mathcal{N}\big(H_i y_\theta(t_i'), R_i\big), \qquad i \in 1, \ldots, M, \tag{10.2}$$

with observation matrices H_i and observation noise covariances R_i for $i \in 1, \ldots, M$. The problem of interest is then to estimate the unknown parameters θ from the data $\{u_i\}_{i=1}^M$, by computing a *maximum-likelihood estimate* (MLE)

$$\hat{\theta}_{\text{MLE}} := \arg\max_\theta \mathcal{M}(\theta), \tag{10.3}$$

with marginal likelihood $\mathcal{M}(\theta) := p(u_{1:M} \mid \theta)$ given by

$$\mathcal{M}(\theta) = \prod_{i=1}^M \mathcal{N}\big(u_i; H_i y_\theta(t_i'), R_i\big). \tag{10.4}$$

*Equal contribution.

But even though the formula for the marginal likelihood $\mathcal{M}(\theta)$ is known, it is not possible to evaluate it directly as it depends on the true ODE solution y_θ, which is intractable. Thus, we approximate it.

First, note that we can write the marginal likelihood as an integral

$$\mathcal{M}(\theta) = \int \left(\prod_{i=1}^{M} \mathcal{N}\big(u_i; H_i y(t'_i), R_i\big) \right) \delta\big(y(t'_{1:M}) - y_\theta(t'_{1:M})\big) \mathrm{d}y(t'_{1:M}). \tag{10.5}$$

The term $\prod_{i=1}^{M} \mathcal{N}(u_i; H_i y(t'_i), R_i)$ corresponds to the likelihood model and is tractable to evaluate given a solution y. The Dirac distribution $\delta(y(t'_{1:M}) - y_\theta(t'_{1:M}))$ however lies on the true ODE solution y_θ. Thus, one approach to approximate the marginal likelihood is therefore to replace this Dirac distribution with a tractable approximation $\hat{p}(y(t'_{1:M}) \mid \theta)$, and then compute an approximated marginal likelihood $\widehat{\mathcal{M}}$. In the following, we show how the probabilistic numerical ODE solution provides such an approximation and how the ODE filtering framework can be used to efficiently compute it.

Remark 10.1 (Other similar quantities of interest also rely on the marginal likelihood) As an alternative to the MLE, the maximum a-posteriori estimate (MAP) can be more stable point-estimate as it includes additional prior information over the parameters. Or, one could sample from the full posterior $p(\theta \mid u_{1:M})$ via Markov chain Monte Carlo (MCMC). While we focus on the MLE in the following, our main contribution in Publication IV lies in the PN-approximated marginal likelihood. It could also be used for MAP estimation or MCMC in ODE parameter inference problems, but this is left for future work.

10.1 Probabilistic ODE solutions as physics-enhanced priors

The probabilistic ODE solver presented in the previous chapters aims to approximate a posterior distribution over the ODE solution y of the form

$$p\Big(y(t_{1:N}) \;\Big|\; y(0) = y_{\theta,0}, \{\dot{y}(t_n) = f_\theta(y(t_n), t_n)\}_{n=1}^{N}\Big). \tag{10.6}$$

Or short, we denote this posterior by $p(y(t_{1:N}) \mid \mathcal{D}_{\mathrm{PN}}, \theta)$, where $\mathcal{D}_{\mathrm{PN}}$ denotes the "PN-data" (i.e. the equality constraints that are conditioned on), and with θ to show that this posterior explicitly depends on a certain choice of parameters. In ODE filtering, this posterior is computed by discretizing a suitable Gauss–Markov prior and then performing inference with the EKF-based NLGSSM algorithm (2.6); see also Chapter 4 for a more detailed introduction to ODE filtering.

The full posterior distribution over the ODE solution computed by the ODE filter is returned as a backward-in-time LGSSM

$$x(t_N) \mid \mathcal{D}_{\mathrm{PN}}, \theta \sim \mathcal{N}\Big(x(t_N); \mu_N^\theta, \Sigma_N^\theta\Big), \tag{10.7a}$$

$$x(t_{n-1}) \mid x(t_n), \mathcal{D}_{\mathrm{PN}}, \theta \sim \mathcal{N}\Big(x(t_{n-1}); G_n^\theta x(t_n) + d_n^\theta, \Lambda_n^\theta\Big), \tag{10.7b}$$

$$y(t_n) = E_0 x(t_n), \tag{10.7c}$$

where the final distribution $\mathcal{N}\big(x(t_N); \mu_N^\theta, \Sigma_N^\theta\big)$ corresponds to the marginal filtering distribution at the final time point t_N and the affine backward transitions $\mathcal{N}\big(x(t_{n-1}); G_n^\theta x(t_n) + d_n^\theta, \Lambda_n^\theta\big)$ are obtained by inverting the forward transition kernels (as described in Algorithm 2.6). The solution $y(t_n)$ can be extracted from the state $x(t_n)$ via projection with E_0 as before.

In summary, the probabilistic numerical ODE solution is a posterior distribution over the solution $y(t_{1:N})$ and it can be represented recursively as a Gauss–Markov process, as shown in eq. (10.7). In the next step, we will use this posterior $p(y(t_{1:N}) \mid \mathcal{D}_{\mathrm{PN}}, \theta)$ as a *prior* to perform inference on the observations $u_{1:M}$, and compute a PN-approximated marginal likelihood

$$\mathcal{M}(\theta) = \int \left(\prod_{i=1}^{M} \mathcal{N}\big(u_i; H_i y(t_i'), R_i\big) \right) p\big(y(t_{1:M}') \mid \mathcal{D}_{\mathrm{PN}}, \theta\big) \mathrm{d}y(t_{1:M}'). \tag{10.8}$$

10.2 Physics-enhanced Gauss–Markov regression

To compute the marginal likelihood of the ODE parameters θ, we now combine the ODE posterior from eq. (10.7) with the observation model from eq. (10.2). We obtain an LGSSM

$$x(t_N) \mid \mathcal{D}_{\mathrm{PN}}, \theta \sim \mathcal{N}\Big(x(t_N); \mu_N^\theta, \Sigma_N^\theta\Big), \tag{10.9a}$$

$$x(t_{n-1}) \mid x(t_n), \mathcal{D}_{\mathrm{PN}}, \theta \sim \mathcal{N}\Big(x(t_{n-1}); G_n^\theta x(t_n) + d_n^\theta, \Lambda_n^\theta\Big), \qquad n = 1, \ldots, N, \tag{10.9b}$$

$$u_i \mid x(t_i') \sim \mathcal{N}\big(H_i E_0 x(t_i'), R_i\big), \qquad i = 1, \ldots M, \tag{10.9c}$$

Note that here we assume that ODE discretization is chosen such that the solver exactly steps through the observation times, that is $t_{1:M}' \subset t_{1:N}$; for all steps outside of the observation time points $t_n \notin t_{1:M}'$, there is no observation and thus no need to update. Then, since the above eq. (10.9) is a standard LGSSM, we can apply the Kalman filtering algorithm 2.4 to compute the corresponding marginal likelihood

$$p(u_{1:M} \mid \mathcal{D}_{\mathrm{PN}}, \theta) = \int p\big(u_{1:M} \mid y(t_{1:M}')\big) p\big(y(t_{1:M}') \mid \mathcal{D}_{\mathrm{PN}}, \theta\big) \mathrm{d}y(t_{1:M}') =: \widehat{\mathcal{M}}_{\mathrm{PN}}(\theta). \tag{10.10}$$

This quantity then serves as the PN-approximated marginal likelihood $\widehat{\mathcal{M}}_{\mathrm{PN}}(\theta) \approx \mathcal{M}(\theta)$. We briefly summarize the computation of $\widehat{\mathcal{M}}_{\mathrm{PN}}(\theta)$ in the following algorithm; a more detailed description and more derivations can be found in Publication IV.

Algorithm 10.1 (PN-approximated marginal likelihood for ODE parameter inference) Given

- an ODE-IVP with vector-field f_θ, initial value $y_{\theta,0}$, and parameter θ,
- an observation model with parameters $\{H_i, R_i\}_{i=1}^M$, observation times $\{t'_i\}_{i=1}^M$, and observations $\{u_i\}_{i=1}^M$,
- a fully-specified probabilistic numerical ODE solver with prior, linearization strategy, step sizes, etc.,

compute the PN-approximated marginal likelihood in the following two steps:

1. Compute the full posterior of the probabilistic numerical ODE solution using Algorithm 4.1. Make sure that the solver steps exactly through the observation times $\{t_i\}_{i=1}^M$.
2. Compute the marginal likelihood on the resulting LGSSM eq. (10.9) with the standard LGSSM inference algorithm (2.4).

Return the marginal likelihood.

10.3 Optimizing the PN-approximated marginal likelihood

The PN-approximated marginal likelihood $\widehat{\mathcal{M}}_{\mathrm{PN}}(\theta)$ can then be used to compute a maximum-likelihood estimate $\hat{\theta}_{\mathrm{MLE}}$. This can be done with any generic numerical optimizer, such as for example gradient descent, Newton's method, or the (limited-memory) Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS or LBFGS); see for instance Nocedal et al. [86] for more details on numerical optimization.

Remark 10.2 (Jointly optimizing ODE parameters and Gauss–Markov prior hyperparameters) The Gauss–Markov priors underlying probabilistic ODE solvers also contain hyperparameters, which are often not precisely known a-priori. For example, these can include the diffusion hyperparameter (which we discussed in detail in Section 6.1 where we also describe schemes to select it online during solution of the ODE), or the lengthscale parameter of a Matérn process. One approach is to learn these hyperparameters jointly with the ODE parameters, by optimizing the same objective $\widehat{\mathcal{M}}_{\mathrm{PN}}(\theta_{\mathrm{ODE}}, \theta_{\mathrm{GM}})$ where θ_{ODE} denote the ODE parameters and θ_{GM} are the hyperparameters of the prior; this is the approach we proposed in Publication IV.

10.4 Example: Parameter inference in a FitzHugh–Nagumo ODE

We briefly demonstrate the resulting algorithm for MLE parameter inference in ODEs. Consider the FitzHugh–Nagumo dynamical system, given by an ODE

$$\dot{y}_1(t) = \gamma\left(y_1(t) - \frac{1}{3}y_1(t)^3 + y_2(t)\right), \tag{10.11a}$$

$$\dot{y}_2(t) = -\frac{1}{c}(y_1(t) - \alpha - \beta y_2(t)), \tag{10.11b}$$

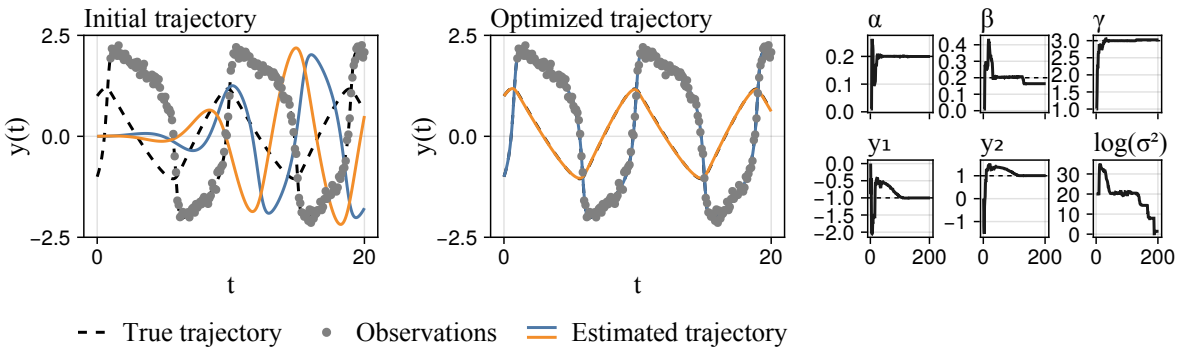


Figure 10.1: **Parameter inference with probabilistic numerical ODE solvers.** Given a FitzHugh–Nagumo ODE with ground-truth ODE parameters $\alpha, \beta, \gamma, y_1, y_2$ and true ODE solution we estimate the parameters from noisy observations with the PN-approximated marginal likelihood. The trajectory resulting from the initial parameter guess is a poor fit (left). After optimization, the trajectory is close to the ground truth and fits the data well (center). The optimization trajectories for each individual parameter, initial value, and for the diffusion hyperparameter σ of the probabilistic solver are shown on the right.

with initial value $y_0 = [-1, 1]$ and time interval $t \in [0, 20]$, and assume ground-truth parameters $\alpha = 0.2$, $\beta = 0.2$ and $\gamma = 3$. We generate observations by solving the ODE with a probabilistic solver with a very low step size and then sampling observations $u_i = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot y(t_i) + v_i$, with observation noise $v_i \sim \mathcal{N}(0, 0.01)$, on a uniform observation time grid $t_i \in \{1 + 0.1 \cdot i\}_{i=0}^{190} \subset [0, 20]$. Then, the problem of interest is to compute an MLE for the vector-field parameters and the initial value, i.e. $\theta = [\alpha, \beta, \gamma, (y_0)_1, (y_0)_2]$, given the data $\{u_i\}_{i=1}^{190}$.

We compute $\hat{\theta}_{\text{MLE}}$ by optimizing the PN-approximated marginal likelihood described above. We use an `EK1` solver with IWP(3) prior and a fixed step-size of $\Delta t = 0.01$ as the ODE solver. In addition to the ODE parameters θ, we also optimize for the scalar global diffusion hyperparameter σ. For optimization, we use the `LBFGS` algorithm with Hager–Zhang line search [42], provided by the Optim.jl Julia package [85]. We initialize the optimizer with ODE parameters $\alpha^{(0)} = 0.01$, $\beta^{(0)} = 0.01$, $\gamma^{(0)} = 1$, initial values $y_0^{(0)} = [0, 0]$, and diffusion $\sigma^2 = 10^{20}$.

Figure 10.1 shows the result. The maximum-likelihood estimates computed by the method are $\hat{\alpha} = 0.21$, $\hat{\beta} = 0.24$, $\hat{\gamma} = 2.98$, $\hat{y_0} = [-0.98, 0.97]$, and $\log(\hat{\sigma}^2) = 3.50$ (all rounded to the first two decimals), which correspond to a relative error of $e_\alpha^{\text{rel}} = 5\%$, $e_\beta^{\text{rel}} = 22\%$, $e_\gamma^{\text{rel}} = 1\%$, and $e_{y_0}^{\text{rel}} = 2\%$, respectively. We also observe that the trajectory resulting from these estimates is close to ground truth solution. A more thorough experimental evaluation of the proposed parameter inference method can be found in Publication IV.

10.5 More reliable ODE parameter inference with diffusion tempering

The optimization procedure presented above has been shown to work well on low-dimensional ODEs (see also Publication IV) but can become unreliable for more challenging problems. One example for such problems is the Hodgkin–Huxley model for neural membrane biophysics [58],

in which parameter inference with gradient-based optimization is known to be challenging as the loss landscape contains many local optima. Hence, to make the optimization more robust to local minima, practitioners often run the optimizer multiple times starting from different random initial parameter guesses, and then keep the best result. But this random-search-like approach also requires a large number of marginal likelihood evaluations, and thus has a larger computational cost. With the goal to develop a more reliable, gradient-based parameter inference procedure for ODEs, we modified our previous approach and presented an improved parameter inference method based on "diffusion tempering" in Publication VI.

The main idea in our proposed method is that instead of optimizing the diffusion hyper-parameter jointly with the ODE parameters, we decrease the diffusion over the course of the optimization procedure along a fixed, chosen schedule. The motivation is as follows. First, we noticed that for very large values of the diffusion parameter σ, the probabilistic ODE solution (and thus the physics-enhanced prior) becomes extremely uncertain. Thereby the subsequent regression task computes a posterior distribution $p(y \mid u_{1:M}, \mathcal{D}_{\mathrm{PN}}, \theta, \sigma)$ which fits the data very well, but has little signal in-between or outside the data. On the other hand, if the diffusion σ is extremely small, then the uncertainty of the probabilistic ODE solution disappears—a zero-diffusion would even lead to a Dirac posterior—and there is no degree of freedom left to infer over. The posterior trajectory therefore stays mostly in place, with $p(y \mid u_{1:M}, \mathcal{D}_{\mathrm{PN}}, \theta, \sigma) \approx p(y \mid \mathcal{D}_{\mathrm{PN}}, \theta, \sigma)$. In principle, the latter case is the desirable one: if for a given parameter we know how to solve the ODE very precisely, then we do not have to adjust our ODE solution based on the data. But on the other hand, the ability of the probabilistic solution to interpolate the data has appeared to be a promising property of the probabilistic solver that could be particularly beneficial for ODE parameter inference; see for instance Experiment 4 in Publication IV, in which ODE filtering recovers the parameters of a pendulum much more reliably than a simple Runge–Kutta least-squares baseline. Therefore, to have the best of both worlds, we propose to start with high diffusions to leverage interpolation, and then programmatically decrease it throughout the optimization to ensure that we eventually have to rely on the actual probabilistic numerical ODE solution.

More formally, consider a diffusion schedule $\mathcal{T} = [\sigma_1, \sigma_2, ..., \sigma_n]$, typically monotonically decreasing with large initial σ_1 and small final σ_n. We subsequently solve n related MLE optimization problems

$$\theta_i = \arg\max_{\theta} \widehat{\mathcal{M}}_{\mathrm{PN}}(\theta_{i-1}, \sigma_i), \qquad i = 1, \dots n, \tag{10.12}$$

where we initialize each optimization problem with the result of the previous one while decreasing the diffusion parameter. This way, we leverage both the interpolation capabilities of the very uncertain probabilistic numerical ODE solution for σ_1, but we also have to rely on the ODE in later stages for σ_n. We briefly summarize the approach in the following algorithm.

Algorithm 10.2 (ODE parameter inference with diffusion tempering) Given

- the requires inputs for the PN-approximated marginal likelihood algorithm 10.1, including the ODE-IVP, observation model, probabilistic ODE solver, and initial parameter θ_0,
- a diffusion tempering schedule $\mathcal{T} = [\sigma_1, \sigma_2, ..., \sigma_n]$,
- an optimizer.

Iteratively solve the following optimization problem for $i = 1, \dots n$:

$$\theta_i = \arg\max_{\theta} \widehat{\mathcal{M}}_{\mathrm{PN}}(\theta_{i-1}, \sigma_i). \tag{10.13}$$

Return the MLE parameter estimate $\hat{\theta} = \theta_n$.

10.6 Example: Improved parameter inference with diffusion tempering

To demonstrate the utility of the tempering algorithm we compare it to the previously presented joint optimization approach on the FitzHugh–Nagumo ODE from Section 10.4. To highlight the different behaviors more clearly, we consider a longer observation time frame with times $t_i \in \{1 + 0.1 \cdot i\}_{i=0}^{390} \subset [0, 40]$, as well as a worse initial parameter guess from which we initialize the optimizer, with initial values $y_0^{(0)} = [0, 0]$, and ODE parameters $\alpha^{(0)} = 10^{-4}$, $\beta^{(0)} = 10^{-4}$, $\gamma^{(0)} = 19$. We again use an `EK1` solver with IWP(3) prior and fixed step-size $\Delta t = 0.01$ as the ODE solver to compute the approximate marginal likelihood. For optimization, we use the `LBFGS` algorithm with backtracking line search [86] provided by the Optim.jl Julia package [85]. In the tempering approach, we consider a diffusion schedule of $\sigma_i = e^{40-i}$ for $i = 0, \dots, 40$.

Figure 10.2 shows the result. First, we see that the trajectory resulting from jointly optimizing the parameters and the diffusion hyperparameter is both a poor fit to the data and to the ground truth ODE solution, as it essentially explains the data with noise by inferring a very high diffusion parameter. On the other hand, the tempering approach does not have this option as the diffusion is decreased over the course of the optimization. As a result, the inferred trajectory is close to the ground truth ODE solution, it fits the data well, and the inferred parameters are close to the ground truth. This shows the effectiveness of the proposed method and the importance of the diffusion tempering approach for challenging ODEs. A much more thorough experimental evaluation of the diffusion tempering approach, which also includes comparisons to non-probabilistic approaches and more challenging dynamical systems, can be found in Publication VI.

10.7 Conclusion

In this chapter, we have presented a method for parameter inference in ODEs which builds on the ODE filtering framework developed in this thesis. We have shown how ODE filters provide an efficient and convenient way to approximate the marginal likelihood of the ODE

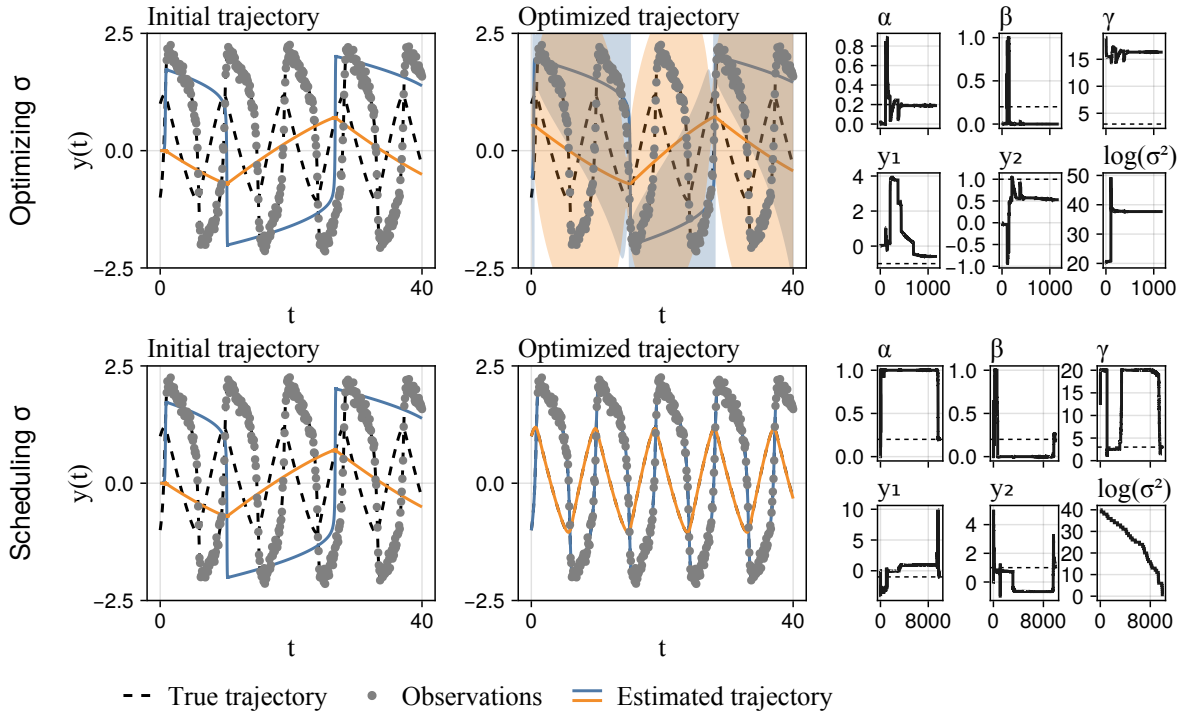


Figure 10.2: **Improved parameter inference with diffusion tempering.** Given a FitzHugh–Nagumo ODE with ground-truth ODE parameters $\alpha, \beta, \gamma, y_1, y_2$ and true ODE solution, we estimate the parameters from noisy observations with both the joint optimization (top) and the tempering approach (bottom). The joint optimization approach results in a poor fit to the data and the ground truth ODE solution and requires a high diffusion parameter to explain the observations. The tempering approach instead sets the diffusion to a fixed, decreasing schedule (shown in the bottom right). It obtains a trajectory close to the ground truth ODE solution and accurate parameter estimates.

parameters given noisy observations and enable the computation of maximum-likelihood estimates in a numerical-error-aware manner. We have also shown how the joint optimization of ODE and diffusion parameters can be improved by using a diffusion tempering approach, which decreases the diffusion over the course of the optimization to prevent the method from fitting the data with only noise, and we have demonstrated the effectiveness of this approach empirically.

Both variants of the proposed method are conceptually compatible with the other extensions presented in this thesis, such as parallel-in-time solvers or the flexible information operator framework, and their combination could lead to even more reliable and efficient parameter inference in ODEs—but this is left for future work. Overall, the presented methods provide a reliable and efficient way to perform parameter inference in ODEs in a numerical-error-aware way.

Part III

Discussion & Conclusion

11 Discussion & Conclusion

This thesis presented a set of tools and methods to improve the stability, computational efficiency, and versatility of filtering-based probabilistic numerical ODE solvers. It thereby established them as a practical alternative to classical numerical methods, all while also providing a probabilistic quantification of the numerical error. We addressed key limitations in the current landscape, introduced novel methods that improve on or extend existing ODE filters, and developed accompanying software to facilitate their adoption. This chapter highlights common themes across the thesis, outlines potential avenues for future research, and discusses the broader implications of our contributions.

11.1 Common themes and future research

Our contributions have been guided by several shared ideas and goals.

11.1.1 Numerical simulation as Bayesian state estimation

The main idea underlying all of the work in this thesis is the conceptual alignment of ODE solving with Bayesian state estimation. Instead of attempting to solve any given differential equation problem directly, we always first formulate the corresponding state estimation problem, consisting of a stochastic process prior and a likelihood model, and only then solve the ODE by computing the posterior distribution of the solution, via Bayesian filtering and smoothing. In the earlier work by Schober et al. [112], Kersting et al. [65], and Tronarp et al. [122], this approach has been mainly motivated by the probabilistic quantification of numerical uncertainty. In this thesis, we have shown how Bayesian state estimation is also a remarkably flexible and versatile framework to formulate ODE filtering in.

The likelihood model of ODE filters in its most basic form contains mostly the given ODE vector field, but it can be easily adjusted to develop probabilistic numerical methods for other numerical problems. This enabled our proposed solvers for higher-order ODEs, systems with conserved quantities, and differential-algebraic equations [II]. Additionally, including actual measurements of the system in the likelihood model enables numerical-error-aware ODE parameter inference methods [IV, VI] and efficient latent force inference [111].

The prior model is another aspect of ODE filters that can be customized. In the past, it has been mostly used to select the desired order of the solver by specifying the smoothness of the q-times integrated Wiener process prior. With the probabilistic exponential integrators developed in this thesis, we have presented a first example for the benefits of choosing a

different prior, resulting in improved stability and allowing for larger step sizes [V]. Other possible motivations for priors could be positivity, boundedness, or known periodicity of the ODE solution. A thorough investigation of priors and their impact on the solution quality is still lacking, but would certainly be a valuable contribution to the field.

The inference algorithm can also be adjusted to suit the given problem, requirements, and computational budget. In most of our work we relied on extended Kalman filtering and smoothing for inference, but we have also used the time-parallel IEKS to develop a probabilistic numerical parallel-in-time method [VII]. In the related literature, unscented Kalman filtering, iterated extended Kalman smoothing, and particle filtering have also been explored for ODE filtering [63, 122, 123, 123, 66]. One particularly promising direction consist in using filters for high-dimensional problems, such as the ensemble Kalman filter [35], the rank-reduced Kalman filter [110], or the computation-aware Kalman filter [95], to better tackle high-dimensional ODEs and discretized PDEs. But there are many more inference algorithms that could be interesting due to computational efficiency or their more expressive uncertainty quantification, that have yet to be explored in the context of ODE filtering.

11.1.2 Improving computational efficiency in theory and practice

Improving the computational efficiency of numerical methods is a common goal throughout many research efforts, both in the context of non-probabilistic methods and in probabilistic numerics. It also motivated many of the contributions developed in this thesis.

On the theoretical side, we provably reduced the computational complexity of probabilistic numerical solvers with respect to the ODE dimension, the order of differentiation of the ODE, and the number of time steps, by leveraging state space model structure, adjusting the information operator, and using time-parallel filtering algorithms, respectively [III, II, VII]. Exploring these ideas jointly could be a promising direction for future research. The improved stability of the probabilistic exponential integrators [V] also provides computational efficiency benefits, as it enables simulations with much coarser step sizes. But stable probabilistic numerical solvers for high-dimensional ODEs with sub-cubic scaling are still lacking, and their development would be a valuable contribution to the field.

On a more practical side, the adaptive step-size selection significantly improves solver runtimes by reducing the number of required steps [I]. A more thorough evaluation of control algorithms, their parameters, and of other local error estimates could provide further improvements. The development of a robust ODE parameter inference method was also driven by the goal of reducing the time required for ODE parameter inference [IV, VI]. Recently, Wu et al. [129] have proposed a similar ODE-filtering-based approach that is particularly suitable for chaotic systems. Further improvements to ODE parameter inference could be obtained by combining these algorithms with some of the other methods developed in this thesis, such as step-size adaptation, linearly-scaling solvers, or the parallel-in-time method.

The implementation of numerical methods in code also plays a substantial role in the actual time required to solve a given numerical problem in practice. With the development of the ProbNumDiffEq.jl package [VIII] in the Julia programming language [9], we have made a

significant contribution to providing an efficient and accessible implementation of probabilistic numerical ODE solvers to the wider public. Efficient ODE filters are also available in the Python JAX ecosystem [16] with the ProbDiffEq package [66]. But non-probabilistic ODE solvers available in highly optimized software libraries such as SciPy [126], MATLAB [116], or DifferentialEquations.jl [97] have been developed and refined over decades, so there is certainly room left for improvement.

11.1.3 Expanding the scope of application

The work presented in this thesis also aims to broaden the range of problems to which probabilistic numerical ODE solvers can be applied, both regarding the kind of numerical problem and the potential downstream application.

Early work in the field focused on rather simple ODEs [112, 65, 122, 123]. We have extended the framework to better handle more complex problems such as stiff ODEs, high-dimensional ODEs and semi-linear ODEs (which both often arise as discretizations of PDEs), and dynamical systems with algebraic constraints [II, V, III]. The development of new probabilistic numerical solvers for other differential equation problems also drove concurrent research in the field, and methods have been proposed for boundary value problems [68], partial differential equations [70], and stochastic differential equations [36]. But many other problem types, such as random ODEs [46], delay differential equations [71], or integro-differential equations [75], have not yet been explored.

The utility of probabilistic numerical ODE solvers for downstream tasks has been previously demonstrated with ODE parameter inference [64] and latent force inference [111]. We have followed these research efforts and have developed new, robust, numerical-error-aware parameter inference algorithms [IV, VI]. ODE filtering has also been shown to be a promising approach for parameter inference in chaotic systems [129]. Future work could explore ODE filtering for robust ODE parameter inference on more challenging problems, for example by combining the likelihood model by Wu et al. [129] with diffusion tempering and with other ODE filtering methods developed in this thesis. ODE-filtering-based methods for the implicit computation of likelihood gradients could be another promising direction for future research, which could be particularly relevant to tackle parameter inference problems related to neural ordinary differential equations [24].

Outside of parameter inference problems, numerical optimal control is another potential application of probabilistic numerics [22, 23]. In a recent work, we have presented how ODE filtering can be used for integration error-aware numerical optimal control [74]. Future research could further explore the utility of ODE filtering for control problems and develop new methods that leverage the probabilistic uncertainty quantification inside the control loop.

Finally, we believe that in order to truly extend ODE filtering to other fields and to appeal to a wider audience we need accessible, documented, feature-rich and efficient software. This has been an ongoing effort in the community and ODE filters are by now implemented in multiple accessible and documented software libraries, namely ProbNum in Python [127], ProbDiffEq in JAX [66], and our contributed ProbNumDiffEq.jl in Julia [VIII]. But in comparison to the

vast amount of software available for non-probabilistic ODE solvers, there is still much work to be done to make probabilistic numerical ODE solvers as accessible and widely usable as their non-probabilistic counterparts.

11.2 Conclusion

This thesis has taken significant steps towards establishing probabilistic numerical solvers as a powerful and flexible framework for solving ODEs. We have developed a range of new methods that improve the efficiency, stability, and flexibility of these solvers, we have shown how they can be applied to a wider range of problems than previously possible, and we have developed efficient and accessible software to make these methods more widely usable. These contributions further narrow the gap between probabilistic and non-probabilistic numerical methods for simulation, facilitate future research in this field and pave the way for the broader adoption of probabilistic numerics in scientific computing and engineering applications.

Bibliography

[1] Assyr Abdulle and Giacomo Garegnani. “Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration”. In: *Statistics and Computing* 30.4 (2020), pp. 907–932. ISSN: 1573-1375.

[2] Vladimir Igorevich Arnol’d. *Mathematical methods of classical mechanics*. Vol. 60. Springer Science & Business Media, 2013.

[3] Francis Bashforth and John Couch Adams. *An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid*. University Press, 1883.

[4] Jonas Beck, Nathanael Bosch, Michael Deistler, Kyra L. Kadhim, Jakob H. Macke, Philipp Hennig, and Philipp Berens. “Diffusion Tempering Improves Parameter Estimation with Probabilistic Integrators for Ordinary Differential Equations”. In: *Forty-first International Conference on Machine Learning*. 2024.

[5] B. M. Bell. “The iterated Kalman smoother as a Gauss–Newton method”. In: *SIAM Journal on Optimization* 4.3 (1994), pp. 626–636.

[6] Tim Besard, Valentin Churavy, Alan Edelman, and Bjorn De Sutter. “Rapid software prototyping for heterogeneous and distributed platforms”. In: *Advances in Engineering Software* 132 (2019), pp. 29–46.

[7] Tim Besard, Christophe Foket, and Bjorn De Sutter. “Effective Extensible Programming: Unleashing Julia on GPUs”. In: *IEEE Transactions on Parallel and Distributed Systems* (2018). ISSN: 1045-9219. arXiv: 1712.03112 [cs.PL].

[8] Jesse Bettencourt, Matthew J. Johnson, and David Duvenaud. “Taylor-Mode Automatic Differentiation for Higher-Order Derivatives in JAX”. In: *Program Transformations for ML Workshop at NeurIPS 2019*. 2019.

[9] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. “Julia: A fresh approach to numerical computing”. In: *SIAM review* 59.1 (2017), pp. 65–98.

[10] G.E. Blelloch. “Scans as primitive parallel operations”. In: *IEEE Transactions on Computers* 38.11 (1989), pp. 1526–1538.

[11] Nathanael Bosch. “ProbNumDiffEq.jl: Probabilistic Numerical Solvers for Ordinary Differential Equations in Julia”. In: *Journal of Open Source Software* 9.101 (2024), p. 7048.

[12] Nathanael Bosch, Adrien Corenflos, Fatemeh Yaghoobi, Filip Tronarp, Philipp Hennig, and Simo Särkkä. "Parallel-in-Time Probabilistic Numerical ODE Solvers". In: *Journal of Machine Learning Research* 25.206 (2024), pp. 1–27.

[13] Nathanael Bosch, Philipp Hennig, and Filip Tronarp. "Calibrated Adaptive Probabilistic ODE Solvers". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 3466–3474.

[14] Nathanael Bosch, Philipp Hennig, and Filip Tronarp. "Probabilistic Exponential Integrators". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[15] Nathanael Bosch, Filip Tronarp, and Philipp Hennig. "Pick-and-Mix Information Operators for Probabilistic ODE Solvers". In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Ed. by Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera. Vol. 151. Proceedings of Machine Learning Research. PMLR, 2022, pp. 10015–10027.

[16] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018.

[17] K.E. Brenan, S.L. Campbell, S.L.V. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1996.

[18] Dirk Brouwer and Gerald M Clemence. *Methods of celestial mechanics*. Elsevier, 2013.

[19] J. C. Butcher. "On Runge-Kutta processes of high order". In: *Journal of the Australian Mathematical Society* 4.2 (1964), pp. 179–194.

[20] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2016. ISBN: 9781119121503.

[21] John Charles Butcher. "A history of Runge-Kutta methods". In: *Applied numerical mathematics* 20.3 (1996), pp. 247–260.

[22] Wenhan Cao, Alexandre Capone, Rishabh Yadav, Sandra Hirche, and Wei Pan. *Computation-Aware Learning for Stable Control with Gaussian Process*. 2024. arXiv: 2406.02272 [eess.SY].

[23] Wenhan Cao and Wei Pan. "Impact of Computation in Integral Reinforcement Learning for Continuous-Time Control". In: *The Twelfth International Conference on Learning Representations*. 2024.

[24] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[25] Oksana A. Chkrebtii, David A. Campbell, Ben Calderhead, and Mark A. Girolami. "Bayesian Solution Uncertainty Quantification for Differential Equations". In: *Bayesian Analysis* 11.4 (2016), pp. 1239–1267.

[26] IP Christiansen. "Numerical simulation of hydrodynamics by the method of point vortices". In: *Journal of Computational Physics* 13.3 (1973), pp. 363–379.

[27] Jon Cockayne, C. Oates, T. Sullivan, and M. Girolami. "Bayesian Probabilistic Numerical Methods". In: *SIAM Rev.* 61 (2019), pp. 756–789.

[28] Patrick R Conrad, Mark Girolami, Simo Särkkä, Andrew Stuart, and Konstantinos Zygalakis. "Statistical analysis of differential equations: introducing probability measures on numerical solutions". In: *Statistics and Computing* 27 (2017), pp. 1065–1082.

[29] Steven M Cox and Paul C Matthews. "Exponential time differencing for stiff systems". In: *Journal of Computational Physics* 176.2 (2002), pp. 430–455.

[30] Michael JP Cullen. *A mathematical theory of large-scale atmosphere/ocean flow*. Imperial College Press, 2006.

[31] J. R. Dormand and P. J. Prince. "A family of embedded Runge–Kutta formulae". In: *Journal of Computational and Applied Mathematics* 6 (1980), pp. 19–26.

[32] JR Dormand and PJ Prince. "Runge-Kutta-Nystrom triples". In: *Computers & Mathematics with Applications* 13.12 (1987), pp. 937–949.

[33] W.H. Enright. "Continuous numerical methods for ODEs with defect control". In: *Journal of Computational and Applied Mathematics* 125.1 (2000). Numerical Analysis 2000. Vol. VI: Ordinary Differential Equations and Integral Equations, pp. 159–170. ISSN: 0377-0427.

[34] L. Euler. *Institutionum calculi integralis*. Institutionum calculi integralis Bd. 1. imp. Acad. imp. Saènt., 1768.

[35] Geir Evensen. "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics". In: *Journal of Geophysical Research: Oceans* 99.C5 (1994), pp. 10143–10162.

[36] Yvann Le Fay, Simo Särkkä, and Adrien Corenflos. *Modelling pathwise uncertainty of Stochastic Differential Equations samplers via Probabilistic Numerics*. 2023. arXiv: 2401.03338 [math.NA].

[37] Bärbel F Finkenstädt and Bryan T Grenfell. "Time series modelling of childhood diseases: a dynamical systems approach". In: *Journal of the Royal Statistical Society Series C: Applied Statistics* 49.2 (2000), pp. 187–205.

[38] Martin J. Gander. "50 Years of Time Parallel Time Integration". In: *Multiple Shooting and Time Domain Decomposition Methods*. Ed. by Thomas Carraro, Michael Geiger, Stefan Körkel, and Rolf Rannacher. Cham: Springer International Publishing, 2015, pp. 69–113. ISBN: 978-3-319-23321-5.

[39] Guglielmo Gattiglio, Lyudmila Grigoryeva, and Massimiliano Tamborrino. "Nearest Neighbors GParareal: Improving Scalability of Gaussian Processes for Parallel-in-Time Solvers". In: *arXiv preprint arXiv:2405.12182* (2024).

[40] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in applied mathematics. Society for Industrial and Applied Mathematics, 2000. ISBN: 9780898714517.

[41] Kjell Gustafsson, Michael Lundh, and Gustaf Söderlind. "A PI stepsize control for the numerical solution of ordinary differential equations". In: *BIT Numerical Mathematics* 28.2 (1988), pp. 270–287. ISSN: 1572-9125.

[42] William W. Hager and Hongchao Zhang. "A New Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search". In: *SIAM Journal on Optimization* 16.1 (2005), pp. 170–192. eprint: https://doi.org/10.1137/030601880.

[43] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2006. ISBN: 9783540306665.

[44] Ernst Hairer, Syvert Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Vol. 8. Springer-Verlag, 1993. ISBN: 978-3-540-56670-0.

[45] Ernst Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Vol. 14. Springer-Verlag, 1996.

[46] Xiaoying Han and Peter E. Kloeden. *Random Ordinary Differential Equations and Their Numerical Solution*. 1st. Springer Publishing Company, Incorporated, 2017. ISBN: 9789811062643.

[47] Jouni Hartikainen and Simo Särkkä. "Kalman filtering and smoothing solutions to temporal Gaussian process regression models". In: *2010 IEEE International Workshop on Machine Learning for Signal Processing*. 2010, pp. 379–384.

[48] Philipp Hennig, Michael A. Osborne, and Mark Girolami. "Probabilistic numerics and uncertainty in computations". In: *Proceedings. Mathematical, physical, and engineering sciences* 471.2179 (2015), pp. 20150142–20150142.

[49] Philipp Hennig, Michael A. Osborne, and Hans P. Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022.

[50] Michel Henon and Carl Heiles. "The applicability of the third integral of motion: Some numerical experiments". In: *The Astronomical Journal* 69 (1964), p. 73.

[51] Herbert W Hethcote. "The mathematics of infectious diseases". In: *SIAM review* 42.4 (2000), pp. 599–653.

[52] Desmond J. Higham. “Robust Defect Control with RungeKutta Schemes”. In: *SIAM Journal on Numerical Analysis* 26.5 (1989), pp. 1175–1183.

[53] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[54] Marlis Hochbruck, Christian Lubich, and Hubert Selhofer. “Exponential integrators for large systems of differential equations”. In: *SIAM Journal on Scientific Computing* 19.5 (1998), pp. 1552–1574.

[55] Marlis Hochbruck and Alexander Ostermann. “Explicit integrators of Rosenbrock-type”. In: *Oberwolfach Reports* 3.2 (2006), pp. 1107–1110.

[56] Marlis Hochbruck and Alexander Ostermann. “Exponential integrators”. In: *Acta Numerica* 19 (2010), pp. 209–286. ISSN: 1474-0508.

[57] Marlis Hochbruck, Alexander Ostermann, and Julia Schweitzer. “Exponential Rosenbrock-type methods”. In: *SIAM Journal on Numerical Analysis* 47.1 (2009), pp. 786–803.

[58] Alan L Hodgkin and Andrew F Huxley. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of physiology* 117.4 (1952), p. 500.

[59] Frank Jensen. *Introduction to computational chemistry*. John wiley & sons, 2017.

[60] William Kahan and Ren-Cang Li. “Composition constants for raising the orders of unconventional schemes for ordinary differential equations”. In: *Mathematics of computation* 66.219 (1997), pp. 1089–1099.

[61] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45. ISSN: 0021-9223.

[62] Takeaki Kariya, Regina Y Liu, Takeaki Kariya, and Regina Y Liu. “Options, futures and other derivatives”. In: *Asset Pricing: -Discrete Time Approach-* (2003), pp. 9–26.

[63] H. Kersting and P. Hennig. “Active Uncertainty Calibration in Bayesian ODE Solvers”. In: *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI)*. 2016, pp. 309–318.

[64] Hans Kersting, Nicholas Krämer, Martin Schiegg, Christian Daniel, Michael Tiemann, and Philipp Hennig. “Differentiable Likelihoods for Fast Inversion of Likelihood-Free Dynamical Systems”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 5198–5208.

[65] Hans Kersting, T. J. Sullivan, and Philipp Hennig. “Convergence rates of Gaussian ODE filters”. In: *Statistics and Computing* 30.6 (2020), pp. 1791–1816. ISSN: 1573-1375.

[66] Nicholas Krämer. “Implementing probabilistic numerical solvers for differential equations”. PhD thesis. Universität Tübingen, 2024.

[67] Nicholas Krämer, Nathanael Bosch, Jonathan Schmidt, and Philipp Hennig. "Probabilistic ODE Solutions in Millions of Dimensions". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 11634–11649.

[68] Nicholas Krämer and Philipp Hennig. "Linear-Time Probabilistic Solution of Boundary Value Problems". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 11160–11171.

[69] Nicholas Krämer and Philipp Hennig. "Stable Implementation of Probabilistic ODE Solvers". In: *Journal of Machine Learning Research* 25.111 (2024), pp. 1–29.

[70] Nicholas Krämer, Jonathan Schmidt, and Philipp Hennig. "Probabilistic Numerical Method of Lines for Time-Dependent Partial Differential Equations". In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Ed. by Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera. Vol. 151. Proceedings of Machine Learning Research. PMLR, 2022, pp. 625–639.

[71] Yang Kuang. *Delay differential equations*. Academic press New York, 1993.

[72] W. Kutta. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*. Teubner, 1901.

[73] Anne Kværnø. "Singly diagonally implicit Runge–Kutta methods with an explicit first stage". In: *BIT Numerical Mathematics* 44.3 (2004), pp. 489–502.

[74] Amon Lahr, Filip Tronarp, Nathanael Bosch, Jonathan Schmidt, Philipp Hennig, and Melanie N. Zeilinger. "Probabilistic ODE solvers for integration error-aware numerical optimal control". In: *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*. Ed. by Alessandro Abate, Mark Cannon, Kostas Margellos, and Antonis Papachristodoulou. Vol. 242. Proceedings of Machine Learning Research. PMLR, 2024, pp. 1018–1032.

[75] Vangipuram Lakshmikantham. *Theory of integro-differential equations*. Vol. 1. CRC press, 1995.

[76] Erik Lindström, Henrik Madsen, and Jan Nygaard Nielsen. *Statistics for finance*. Chapman and Hall/CRC, 2018.

[77] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. "A parareal in time discretization of PDEs". In: *Comptes Rendus de lAcadémie des Sciences. Série I. Mathématique* 332 (2001).

[78] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. "When Gaussian Process Meets Big Data: A Review of Scalable GPs". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11 (2020), pp. 4405–4423.

[79] Edward N Lorenz. "Predictability: A problem partly solved". In: *Proc. Seminar on predictability*. Vol. 1. 1. Reading. 1996.

[80] Vu Thai Luan and Alexander Ostermann. "Exponential Rosenbrock methods of order five construction, analysis and numerical comparisons". In: *Journal of Computational and Applied Mathematics* 255 (2014), pp. 417–431.

[81] James D MacBeth and Larry J Merville. "An empirical examination of the Black-Scholes call option pricing model". In: *The journal of finance* 34.5 (1979), pp. 1173–1186.

[82] Niel K. Madsen. "The method of lines for the numerical solution of partial differential equations". In: *Proceedings of the SIGNUM meeting on Software for partial differential equations* (1975).

[83] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[84] Robert C Merton. "Theory of rational option pricing". In: *The Bell Journal of economics and management science* (1973), pp. 141–183.

[85] Patrick Kofod Mogensen and Asbjørn Nilsen Riseth. "Optim: A mathematical optimization package for Julia". In: *Journal of Open Source Software* 3.24 (2018), p. 615.

[86] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[87] E.J. Nyström. *Über die numerische integration von differentialgleichungen*. Acta Societatis Scientiarum Fennicae. Finn. Literaturges., 1925.

[88] Chris J Oates and Timothy John Sullivan. "A modern retrospective on probabilistic numerics". In: *Statistics and computing* 29.6 (2019), pp. 1335–1351.

[89] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge university press, 2010.

[90] B. Oksendal. *Stochastic Differential Equations: An Introduction with Applications*. Universitext. Springer Berlin Heidelberg, 2013. ISBN: 9783662031858.

[91] Tim Palmer. "The ECMWF ensemble prediction system: Looking back (more than) 25 years and projecting forward 25 years". In: *Quarterly Journal of the Royal Meteorological Society* 145 (2019), pp. 12–24.

[92] Kamran Pentland, Massimiliano Tamborrino, Debasmita Samaddar, and Lynton C. Appel. “Stochastic Parareal: An Application of Probabilistic Methods to Time-Parallelization”. In: *SIAM Journal on Scientific Computing* 0.0 (2021), S82–S102.

[93] Kamran Pentland, Massimiliano Tamborrino, T. J. Sullivan, James Buchanan, and L. C. Appel. “GParareal: a time-parallel ODE solver using Gaussian process emulation”. In: *Statistics and Computing* 33.1 (2022), p. 23. ISSN: 1573-1375.

[94] Linda Petzold. “Differential/Algebraic Equations are not ODE’s”. In: *Siam Journal on Scientific and Statistical Computing* 3 (1982).

[95] Marvin Pförtner, Jonathan Wenger, Jon Cockayne, and Philipp Hennig. *Computation-Aware Kalman Filtering and Smoothing*. 2024. arXiv: 2405.08971 [cs.LG].

[96] Balthasar van der Pol. “On "relaxation-oscillations"”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1926), pp. 978–992. eprint: https://doi.org/10.1080/14786442608564127.

[97] Christopher Rackauckas and Qing Nie. “DifferentialEquations.jl A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia”. In: *Journal of Open Research Software* 5.1 (2017).

[98] Matti Raitoharju, Ángel F. García-Fernández, and Robert Piché. “KullbackLeibler divergence approach to partitioned update Kalman filter”. In: *Signal Processing* 130 (2017), pp. 289–298. ISSN: 0165-1684.

[99] Matti Raitoharju and Robert Piché. “On Computational Complexity Reduction Methods for Kalman Filter Extensions”. In: *IEEE Aerospace and Electronic Systems Magazine* 34.10 (2019), pp. 2–19.

[100] Matti Raitoharju, Robert Piché, Juha Ala-Luhtala, and Simo Ali-Löytty. “Partitioned Update Kalman Filter”. In: *Journal of Advances in Information Fusion* 11 (2016), pp. 3–14.

[101] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. ISBN: 9780262256834.

[102] H. E. Rauch, F. Tung, and C. T. Striebel. “Maximum likelihood estimates of linear dynamic systems”. In: *AIAA Journal* 3.8 (1965), pp. 1445–1450.

[103] Steven Roman. “The formula of Faa di Bruno”. In: *The American Mathematical Monthly* 87.10 (1980), pp. 805–809.

[104] C. Runge. “Ueber die numerische Auflösung von Differentialgleichungen”. In: *Mathematische Annalen* 46.2 (1895), pp. 167–178. ISSN: 1432-1807.

[105] Simo Sarkka, Arno Solin, and Jouni Hartikainen. “Spatiotemporal Learning via Infinite-Dimensional Bayesian Filtering and Smoothing: A Look at Gaussian Process Regression Through Kalman Filtering”. In: *IEEE Signal Processing Magazine* 30.4 (2013), pp. 51–61.

[106] Simo Särkkä and Angel F. García-Fernández. "Temporal Parallelization of Bayesian Smoothers". In: *IEEE Transactions on Automatic Control* 66.1 (2021), pp. 299–306.

[107] Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2019.

[108] Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*. 2nd ed. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2023.

[109] William E Schiesser. *The numerical method of lines: integration of partial differential equations*. Elsevier, 2012.

[110] Jonathan Schmidt, Philipp Hennig, Jörg Nick, and Filip Tronarp. *The Rank-Reduced Kalman Filter: Approximate Dynamical-Low-Rank Filtering In High Dimensions*. 2023. arXiv: 2306.07774 [stat.ML].

[111] Jonathan Schmidt, Nicholas Krämer, and Philipp Hennig. "A Probabilistic State Space Model for Joint Inference from Differential Equations and Data". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 12374–12385.

[112] Michael Schober, Simo Särkkä, and Philipp Hennig. "A probabilistic model for the numerical solution of initial value problems". In: *Statistics and Computing* 29.1 (2019), pp. 99–122. ISSN: 1573-1375.

[113] Fred Schweppe. "Evaluation of likelihood functions for Gaussian signals". In: *IEEE transactions on Information Theory* 11.1 (1965), pp. 61–70.

[114] L. F. Shampine. "Error estimation and control for ODEs". In: *Journal of Scientific Computing* 25.1 (2005), pp. 3–16. ISSN: 1573-7691.

[115] Lawrence F. Shampine. "Some Practical Runge–Kutta Formulas". In: *Mathematics of Computation* 46.173 (1986), pp. 135–150.

[116] Lawrence F. Shampine and Mark W. Reichelt. "The MATLAB ODE Suite". In: *SIAM Journal on Scientific Computing* 18.1 (1997), pp. 1–22.

[117] Tony Stillfjord and Filip Tronarp. *Computing the matrix exponential and the Cholesky factor of a related finite horizon Gramian*. 2023. arXiv: 2310.13462 [math.NA].

[118] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[119] Onur Teymur, Han Cheng Lie, Tim Sullivan, and Ben Calderhead. "Implicit Probabilistic Integrators for ODEs". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[120] Onur Teymur, Kostas Zygalakis, and Ben Calderhead. "Probabilistic Linear Multistep Methods". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[121] Filip Tronarp, Nathanael Bosch, and Philipp Hennig. "Fenrir: Physics-Enhanced Regression for Initial Value Problems". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 21776–21794.

[122] Filip Tronarp, Hans Kersting, Simo Särkkä, and Philipp Hennig. "Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective". In: *Statistics and Computing* 29.6 (2019), pp. 1297–1315.

[123] Filip Tronarp, Simo Särkkä, and Philipp Hennig. "Bayesian ODE solvers: the maximum a posteriori estimate". In: *Statistics and Computing* 31.3 (2021), p. 23. ISSN: 1573-1375.

[124] Ch Tsitouras. "RungeKutta pairs of order 5 (4) satisfying only the first column simplifying assumption". In: *Computers & Mathematics with Applications* 62.2 (2011), pp. 770–775.

[125] C. Van Loan. "Computing integrals involving the matrix exponential". In: *IEEE Transactions on Automatic Control* 23.3 (1978), pp. 395–404.

[126] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, lhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272.

[127] Jonathan Wenger, Nicholas Krämer, Marvin Pförtner, Jonathan Schmidt, Nathanael Bosch, Nina Effenberger, Johannes Zenn, Alexandra Gessner, Toni Karvonen, François-Xavier Briol, Maren Mahsereci, and Philipp Hennig. *ProbNum: Probabilistic Numerics in Python*. 2021. arXiv: 2112.02100 [cs.MS].

[128] Mark J Willis. "Proportional-integral-derivative control". In: *Dept. of Chemical and Process Engineering University of Newcastle* 6 (1999).

[129] Mohan Wu and Martin Lysy. "Data-Adaptive Probabilistic Likelihood Approximation for Ordinary Differential Equations". In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. Ed. by Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li. Vol. 238. Proceedings of Machine Learning Research. PMLR, 2024, pp. 1018–1026.

[130] Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, and Simo Särkkä. "Parallel Iterated Extended and Sigma-Point Kalman Smoothers". In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 5350–5354.

[131] Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, and Simo Särkkä. "Parallel square-root statistical linear regression for inference in nonlinear state space models". In: *arXiv:2207.00426* (2023). arXiv: 2207.00426 [stat.CO].

[132] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. "Diffusion models: A comprehensive survey of methods and applications". In: *ACM Computing Surveys* 56.4 (2023), pp. 1–39.

Part IV

Appendix

Publications

Contents

Calibrated Adaptive Probabilistic ODE Solvers

Nathanael Bosch[1] **Philipp Hennig**[1,2] **Filip Tronarp**[1]
[1]University of Tübingen
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
{nathanael.bosch, philipp.hennig, filip.tronarp}@uni-tuebingen.de

Abstract

Probabilistic solvers for ordinary differential equations assign a posterior measure to the solution of an initial value problem. The joint covariance of this distribution provides an estimate of the (global) approximation error. The contraction rate of this error estimate as a function of the solver's step size identifies it as a well-calibrated worst-case error, but its explicit numerical value for a certain step size is not automatically a good estimate of the explicit error. Addressing this issue, we introduce, discuss, and assess several probabilistically motivated ways to calibrate the uncertainty estimate. Numerical experiments demonstrate that these calibration methods interact efficiently with adaptive step-size selection, resulting in descriptive, and efficiently computable posteriors. We demonstrate the efficiency of the methodology by benchmarking against the classic, widely used Dormand–Prince 4/5 Runge–Kutta method.

1 INTRODUCTION

Ordinary differential equations (ODEs) arise in almost all areas of science and engineering. In the field of machine learning, recent work on normalizing flows (Rezende and Mohamed, 2015) and neural ODEs (Chen et al., 2018) lead to a particular surge of interest. In this paper we consider initial value problems (IVPs), defined by an ODE

$$\dot{y}(t) = f\left(y(t), t\right), \qquad \forall t \in [t_0, T], \tag{1}$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial value $y(t_0) = y_0 \in \mathbb{R}^d$.

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

Limited by finite computational resources, the numerical solution of an IVP is inevitably only an approximation. Though, most classic numerical solvers do not return an estimate of their own numerical error, leaving it to the practitioner to evaluate the reliability of the result. The field of *probabilistic numerics* (PN) (Hennig et al., 2015; Oates and Sullivan, 2019) seeks to overcome this ignorance of numerical uncertainty. By treating numerical algorithms as problems of statistical inference, the numerical error can be quantified probabilistically.

One particular class of probabilstic numerical solvers for ODEs treats IVPs as a Gauss–Markov regression problem (Tronarp et al., 2019, 2020), the solution of which can be efficiently approximated with Bayesian filtering and smoothing (Särkkä, 2013). These so-called ODE filters relate to classic multistep methods (Schober et al., 2018), and have been shown to converge to the true solution of the IVP with high polynomial rates while providing (asymptotically) well-calibrated confidence intervals (Kersting et al., 2020). In practice, however, there remain two gaps: First, efficient implementation of ODE solvers require adaptive step-size selection, which has not received much attention in the past; second, the calibration of the posterior uncertainty estimates depends on the choice of specific diffusion hyperparameters. Both are addressed in this paper.

The contributions of this paper are the following: We introduce and discuss uncertainty calibration methods for models with both constant and time-varying diffusion, and extend existing approaches with multivariate parameter estimates. After calibration, the probabilistic observation model provides an objective for local error control and adaptive step-size selection, enabling the solvers to make efficient use of their computational budget. The resulting probabilistic numerical ODE solvers are evaluated and compared for a large range of configurations and tolerance levels, demonstrating descriptive posteriors and computational efficiency comparable to the classic Dormand–Prince 4/5 Runge–Kutta method.

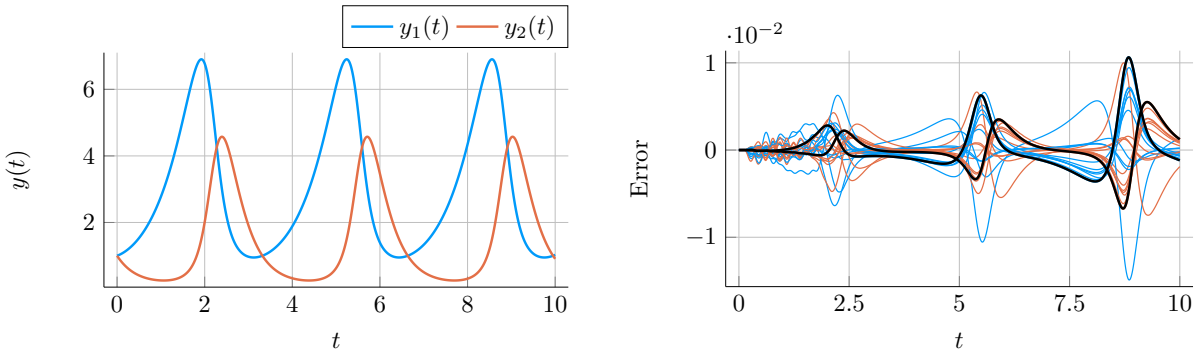


Figure 1: *Probabilistic solution of the Lotka-Volterra equations.* Left: Posterior mean returned by the probabilistic solver. For the chosen tolerance levels the returned uncertainties are too small to be visually separated from the mean. Right: True error trajectory (black) and sampled error trajectories (colored). Both the true solution and the samples exhibit similar patterns, indicating a well-structured and calibrated uncertainty estimate of the provided posterior distribution.

2 PROBABILISTIC ODE SOLVERS

In this paper, the solution of the IVP is posed as a Bayesian inference problem (Cockayne et al., 2019). By considering Gauss–Markov priors on the state-space, the problem is reduced to a case of Bayesian state estimation (Tronarp et al., 2019). In the following, we introduce the state estimation problem and describe the approximate inference procedure.

2.1 ODE Solutions as State Estimation

A priori, we model the solution together with $q \in \mathbb{N}$ of its derivatives as a Gauss–Markov process $X(t) = \left[\left(X^{(0)}(t)\right)^\intercal, \left(X^{(1)}(t)\right)^\intercal, \ldots, \left(X^{(q)}(t)\right)^\intercal\right]^\intercal$, where $X^{(i)}(t)$ models the i-th derivative $y^{(i)}(t)$. More precisely, $X(t)$ is the q-times integrated Wiener process (IWP), which solves the stochastic differential equations

$$\mathrm{d}X^{(i)}(t) = X^{(i+1)}(t)\,\mathrm{d}t, \quad i = 1, \ldots, q-1 \tag{2a}$$
$$\mathrm{d}X^{(q)}(t) = \Gamma^{1/2}\,\mathrm{d}B(t), \tag{2b}$$
$$X(t_0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{2c}$$

where $\Gamma^{1/2}$ is the symmetric square root of some positive semi-definite matrix $\Gamma \in \mathbb{R}^{d\times d}$.

The continuous-time model can be described by transition densities (Särkkä and Solin, 2019, Section 6.2)

$$X(t+h)\,|\,X(t) \sim \mathcal{N}\left(A(h)X(t), Q(h)\right). \tag{3}$$

For the IWP prior, $A(h) \in \mathbb{R}^{d(q+1)\times d(q+1)}$ and $Q(h) \in \mathbb{R}^{d(q+1)\times d(q+1)}$ are of the form

$$A(h) = \breve{A}(h) \otimes I_d, \tag{4a}$$
$$Q(h) = \breve{Q}(h) \otimes \Gamma, \tag{4b}$$

with $\breve{A}(h), \breve{Q}(h)$ given by Kersting et al. (2020, Appendix A)

$$\breve{A}_{ij}(h) = \mathbb{I}_{i\le j}\frac{h^{j-1}}{(j-i)!}, \tag{5a}$$
$$\breve{Q}_{ij}(h) = \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!}. \tag{5b}$$

To relate the prior to the solution of the IVP, define the measurement process

$$Z(t) = X^{(1)}(t) - f\left(X^{(0)}(t)\right). \tag{6}$$

The probabilistic numerical solution of the ODE is computed by conditioning $X(t)$ on the event that the realisation $z(t)$ of $Z(t)$ is zero on the grid $\{t_n\}_{n=1}^N$ (Tronarp et al., 2019)

$$z_n := z(t_n) = 0, \quad n = 1, \ldots, N. \tag{7}$$

The resulting inference problem of computing

$$p\left(X(t)\,|\,\{z_n\}_{n=1}^N\right) \tag{8}$$

is, for non-linear vector fields f, known as a non-linear Gauss–Markov regression problem and is in general intractable, but it is possible to efficiently compute approximations (Särkkä, 2013).

2.2 Approximate Gaussian Inference

We consider approximate Bayesian inference based on linearization by Taylor-series expansion, known as the extended Kalman filter (EKF) in statistical signal processing (Särkkä, 2013, Section 5.2). By linearizing the

measurement likelihood we can efficiently and iteratively compute approximations

$$p\left(X(t_n) \,|\, \{z_i\}_{i=1}^{n-1}\right) \approx \mathcal{N}\left(\mu_n^P, \Sigma_n^P\right), \tag{9a}$$
$$p\left(X(t_n) \,|\, \{z_i\}_{i=1}^{n}\right) \approx \mathcal{N}\left(\mu_n^F, \Sigma_n^F\right), \tag{9b}$$
$$p\left(z_n \,|\, \{z_i\}_{i=1}^{n-1}\right) \approx \mathcal{N}\left(\hat{z}_n, S_n\right), \tag{9c}$$

through the following *prediction* and *update* steps (Särkkä, 2013, Section 5.2):

Prediction:

$$\mu_n^P = A(h_{n-1})\mu_{n-1}^F, \tag{10a}$$
$$\Sigma_n^P = A(h_{n-1})\Sigma_{n-1}^F A(h_{n-1})^\intercal + Q(h_{n-1}). \tag{10b}$$

Update:

$$\hat{z}_n = E_1\mu_n^P - f\left(E_0\mu_n^P, t_n\right), \tag{11a}$$
$$S_n = H_n\Sigma_n^P H_n^\intercal, \tag{11b}$$
$$K_n = \Sigma_n^P H_n^\intercal S_n^{-1}, \tag{11c}$$
$$\mu_n^F = \mu_n^P + K_n(z_n - \hat{z}_n), \tag{11d}$$
$$\Sigma_n^F = \Sigma_n^P - K_n S_n K_n^\intercal, \tag{11e}$$

where H_n can be either $H_n := E_1$ for a zeroth order approximation, or $H_n := E_1 - J_f(E_0\mu_n, t_n)E_0$ for a first order approximation of the vector field f, with $E_0 := e_0^\intercal \otimes I$, $E_1 := e_1^\intercal \otimes I$.

The zeroth and first order linearizations correspond to the updates by Schober et al. (2018) and Tronarp et al. (2019), respectively. In the sequel, we refer to the algorithm with zeroth and first order linearization as EKF0 and EKF1, respectively.

Remark 1. *While most classic ODE solvers do not use the Jacobians of the vector field f, they play a central role in Rosenbrock methods (Rosenbrock, 1963; Hochbruck et al., 2008), a class of semi-implicit solvers for stiff ODEs (Hairer and Wanner, 1996, Chapter IV.7). In probabilistic solvers, the Jacobian was used in a probabilistic multistep method (Teymur et al., 2016) and, more recently, with extended Kalman filtering and smoothing (Tronarp et al., 2019, 2020).*

The Bayesian *filtering* posterior for $X(t)$ is conditioned only on the measurements obtained before and at the time step t, but does not include future measurements. Computing the (approximate) full marginal posterior $p\left(X(t_n) \,|\, \{z_n\}_{n=1}^N\right)$ can be done with Bayesian *smoothing*. The extended Rauch–Tung–Striebel smoother, also called the extended Kalman smoother (EKS), describes an algorithm to efficiently compute Gaussian approximations

$$p\left(X(t_n) \,|\, \{z_i\}_{i=1}^N\right) \approx \mathcal{N}\left(\mu_n^S, \Sigma_n^S\right) \tag{12}$$

with a backwards recursion, given by the *smoothing* step

$$G_n = \Sigma_n^F A(h_n)(\Sigma_{n+1}^P)^{-1}, \tag{13a}$$
$$\mu_n^S = \mu_n^F + G_n(\mu_{n+1}^S - \mu_{n+1}^P), \tag{13b}$$
$$\Sigma_n^S = \Sigma_n^F + G_n(\Sigma_{n+1}^S - \Sigma_{n+1}^P)G_n^\intercal. \tag{13c}$$

See also Särkkä (2013, Chapter 9). The approximate posterior for off-the-grid time steps $t \in [t_0, T]$, relating to *dense output* in classic numerical solvers (Hairer et al., 1993, Chapter II.6), can be straight-forwardly computed by using the Gauss–Markov property. In a similar manner as above, we refer to the resulting algorithms with linearization of order zero and one as EKS0 and EKS1, respectively.

The question remains how to set the initial state $X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$. This problem can also be observed in classic multistep methods: In addition to the multistep formula, they specify a starting procedure using, for example, Taylor series expansion (Bashforth and Adams, 1883), one-step methods (Hairer et al., 1993, Chapter III.1), or other iterative procedures (Nordsieck, 1962). In a similar effort, Schober et al. (2018) describe an initialization for a probabilistic ODE solver via Runge-Kutta methods. Since the probabilistic formulation enables us to explicitly quantify uncertainty over the initial values, it is also possible to set the initial state to a zero mean and unit variance Gaussian distribution and to condition on the correct initial values $X^{(0)}(t_0) = y_0$ and $X^{(1)}(t_0) = f(y_0, t_0)$ (Tronarp et al., 2019, 2020). For our experiments, where orders $q \leq 5$ are used, we found it feasible to compute all derivatives of the correct initial value via automatic differentiation, and we explicitly set $\mu_0 = (y(t_0), \dot{y}(t_0), ..., y^{(q)}(t_0))$ and zero covariance.

Remark 2. *The exact initial derivatives can be computed efficiently with Tylor-mode automatic differentiation (Griewank and Walther, 2000; Bettencourt et al., 2019). A more extensive description of an initialization procedure, together with further considerations for a numerically stable implementation, is provided by Krämer and Hennig (2020).*

3 UNCERTAINTY CALIBRATION

The probabilistic solver with IWP prior presented in Section 2.1 contains a free parameter Γ, which is of particular importance for the posterior uncertainty, as it determines the gain of the Wiener process entering the system in Eq. (2). In this section, we present different diffusion models and discuss approaches for estimating this parameter and thereby calibrating uncertainties.

3.1 Time-Fixed Diffusion Model

A common approach for Bayesian model selection and parameter estimation is to maximize the marginal likelihood, or *evidence*, of the observed data $z_{1:n}$, given by the prediction error decomposition (Schweppe, 1965)

$$p(z_{1:n}) = p(z_1) \prod_{i=2}^{n} p(z_i \,|\, z_{1:i-1}). \tag{14}$$

For affine vector fields, the Kalman filter computes the marginals $p(z_i \,|\, z_{1:i-1})$ exactly, but for non-affine vector fields we solve the Bayesian filtering problem only approximately. Nevertheless, it is a natural choice to approximate the marginal likelihoods in the same way as the filtering solution is approximated, i.e. with extended Kalman filtering, as

$$p(z_{1:n}) \approx \prod_{i=1}^{n} \mathcal{N}\left(z_i; \hat{z}_i, S_i\right), \tag{15a}$$

$$\hat{z}_i = E_1 \mu_i^P - f\left(E_0 \mu_i^P, t_i\right), \tag{15b}$$

$$S_i = H_i \Sigma_i^P H_i^\intercal. \tag{15c}$$

Maximizing Eq. (15a) is referred to as *quasi maximum likelihood estimation* in signal processing (Lindström et al., 2018).

For the case of scalar matrices $\Gamma = \sigma^2 I_d$, Tronarp et al. (2019, Proposition 4) provide a (quasi) maximum-likelihood estimate (quasi-MLE) of σ^2, denoted by $\hat{\sigma}_N^2$. Assuming an initial covariance of the form $\Sigma_0 = \sigma^2 \breve{\Sigma}_0$, $\hat{\sigma}_N^2$ is given by

$$\hat{\sigma}_N^2 = \frac{1}{Nd} \sum_{n=1}^{N} (z_n - \hat{z}_n)^\intercal S_n^{-1} (z_n - \hat{z}_n). \tag{16}$$

This estimation can be performed on-line in order to provide calibrated uncertainty estimates during the solve, which are required for step-size adaptation.

The IWP prior with scalar diffusion $\Gamma = \sigma^2 I_d$ describes the same model for each dimension. Furthermore, as the measurement matrix H_n associated with the EKF0 does not depend on the vector field, the estimated uncertainties of each dimension will be the same. To fix this shortcoming of the EKF0 we propose a model with diagonal $\Gamma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$. Its quasi-MLE is provided in Proposition 1 below.

Proposition 1. *Let $\Gamma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ and $\Sigma_0 = \breve{\Sigma}_0 \otimes \Gamma$. Then the prediction and filtering covariances computed by the EKF0 with IWP prior are of the form $\Sigma_n^P = \breve{\Sigma}_n^P \otimes \Gamma$, $\Sigma_n^F = \breve{\Sigma}_n^F \otimes \Gamma$, and the approximated measurement covariances are given by $S_n = \check{s}_n \cdot \Gamma$, where $\check{s}_n$ is $\check{s}_n := e_2^\intercal \breve{\Sigma}_n^P e_2$. The quasi maximum-likelihood estimate of Γ, denoted by $\hat{\Gamma}$, is diagonal and given by*

$$\hat{\Gamma}_{ii} = \frac{1}{N} \sum_{n=1}^{N} \frac{(\hat{z}_n)_i^2}{\check{s}_n}, \qquad i \in \{1, \ldots, d\}. \tag{17}$$

The proof follows the idea of Tronarp et al. (2019, Proposition 4). A more detailed derivation can be found in Appendix A.

3.2 Time-Varying Diffusion Model

To allow for greater flexibility, Schober et al. (2018) propose a model in which $\Gamma = \Gamma_n$ is allowed to vary for different integration steps t_n. In such a model, all measurements $\{z_i\}_{i=1}^{n-1}$ taken before time t_n are independent of the parameter Γ_n. We obtain

$$\arg\max_{\Gamma_n} p\left(z_{1:n}\right) = \arg\max_{\Gamma_n} p\left(z_n \,|\, z_{1:n}\right) \tag{18a}$$

$$\approx \arg\max_{\Gamma_n} \mathcal{N}\left(z_n; \hat{z}_n, S_n\right), \tag{18b}$$

with $S_n = H_n \left[A_n \Sigma_{n-1}^F A_n^\intercal + (\breve{Q}_n \otimes \Gamma_n)\right] H_n^\intercal$.

To approximately estimate Γ_n, Schober et al. (2018) propose an estimation based on "local" errors, a common procedure for error control and step-size adaptation in classic numerical methods (Hairer et al., 1993, Chapter II.4). Assuming an error-free predicted solution μ_{n-1}^F at time t_{n-1}, that is, $\Sigma_{n-1}^F = 0$, yields

$$S_n = H_n(\breve{Q}_n \otimes \Gamma_n) H_n^\intercal. \tag{19}$$

For scalar matrices $\Gamma_n = \sigma_n^2 I_d$ this implies

$$S_n = \sigma_n^2 \cdot H_n(\breve{Q}_n \otimes I_d) H_n^\intercal. \tag{20}$$

Computing the quasi-MLE by solving Eq. (18) yields the parameter estimate by Schober et al. (2018):

$$\hat{\sigma}_n^2 = \frac{1}{d} (z_n - \hat{z}_n)^\intercal \left(H_n(\breve{Q}_n \otimes I_d) H_n^\intercal\right)^{-1} (z_n - \hat{z}_n). \tag{21}$$

As for the fixed diffusion model, we can improve the expressiveness of the EKF0 by considering a multivariate model with diagonal $\Gamma_n = \mathrm{diag}(\sigma_{n1}^2, \ldots, \sigma_{nd}^2)$. With the local error based estimation, and using $H_n = e_1 \otimes I_d$, we obtain

$$S_n = (\breve{Q}_n)_{11} \cdot \Gamma_n. \tag{22}$$

With a covariance of this form, Eq. (18) can be solved and we obtain the quasi-MLE

$$(\hat{\Gamma}_n)_{ii} = (z_n - \hat{z}_n)_i^2 / (\breve{Q}_n)_{11}, \qquad i \in \{1, \ldots, d\}, \tag{23}$$

as parameter estimate for models with the time-varying, diagonal diffusion.

4 STEP-SIZE ADAPTATION

While the algorithm described in Sections 2 and 3 is able to compute calibrated posterior distributions over

solutions to any IVP, it still lacks a common tool to make efficient use of its computations: *step-size adaptation*. Indeed, most modern, non-probabilistic ODE solvers perform local error control with adaptive step-size selection, allowing them to compute the result up to a desired precision while avoiding unnecessary computational work. In the following, we first review error estimation and step-size control in classic numerical solvers, then provide a principled objective for error control of probabilistic ODE solvers and describe the full step-size adaptation algorithm.

4.1 Error Control in Classic Solvers

An important aspect of numerical analysis is to monitor and control the error of a method. Commonly, a distinction is made between two kinds of errors: The *local* error describes the error that the algorithm introduces after a single step, whereas the *global* error is the cumulative error of the computed solution caused by multiple iterations. The global error is typically not of practical interest for error monitoring and control (Hairer et al., 1993, Chapter II.4), and instead the local error is estimated, for example through Richardson extrapolation (Hairer et al., 1993, Theorem 4.1), or more commonly via embedded Runge-Kutta methods (Hairer et al., 1993, Chapter II.4) or the Milne device (Byrne and Hindmarsh, 1975). These estimates are then used in the step-size control algorithm, which ensures that the chosen step sizes are sufficiently small to yield the desired precision of the computed result, while being sufficiently large to avoid unnecessary computational work. Common control algorithms for step-size selection include proportional control (Hairer et al., 1993, Chapter II.4) and proportional-integral (PI) control (Gustafsson et al., 1988).

4.2 Error Control in Probabilistic Solvers

In Gaussian filtering, the natural object to consider for error estimation and control are the residuals $(z_n - \hat{z}_n)$. Schober et al. (2018) show how to use this quantity for both uncertainty calibration (as presented in Section 3.2) and local error control. We generalize their error control objective to be applicable to all presented algorithms and uncertainty calibration methods.

After calibration, the extended Kalman filtering algorithm approximates (see Eq. (11))

$$p(z_n \mid z_{1:n-1}) \approx \mathcal{N}\left(z_n; \hat{z}_n, S_n\right), \tag{24}$$

with

$$\hat{z}_n = E_1 \mu_n^P - f\left(E_0 \mu_n^P, t_n\right), \tag{25a}$$

$$S_n = H_n \left(A_n \Sigma_{n-1}^F A_n^\intercal + \left(\breve{Q}_n \otimes \hat{\Gamma}_n\right)\right) H_n^\intercal, \tag{25b}$$

where H_n can correspond to either the zeroth or the first order linearization, and $\hat{\Gamma}_n$ has been estimated through one of the approaches of Section 3.

For step-size adaptation we want to control *local* errors, and therefore assume an error-free solution estimate at time t_{n-1}. With $\Sigma_{n-1}^F = 0$ we obtain the approximation

$$p((z_n - \hat{z}_n) \mid z_{1:n-1}) \approx \mathcal{N}\left(z_n - \hat{z}_n; 0, H_n \left(\breve{Q}_n \otimes \hat{\Gamma}_n\right) H_n^\intercal\right). \tag{26}$$

Finally, we define the objective for local error control $D_n \in \mathbb{R}^d$ as the (local) standard deviations of the residual vector $(z_n - \hat{z}_n)$, given by

$$(D_n)_i := \left(H_n \left(\breve{Q}_n \otimes \hat{\Gamma}_n\right) H_n^\intercal\right)_{ii}^{1/2}, \quad i \in \{1, \ldots, d\}. \tag{27}$$

For the EKF0 algorithm and time-varying, scalar $\Gamma = \sigma^2 I_d$, we recover the expected error used by Schober et al. (2018) for step-size control (see also Byrne and Hindmarsh (1975)), but the general formulation in Eq. (27) can also be used with on-line quasi-MLE for time-fixed Γ (Section 3.1) and in combination with the EKF1.

4.3 Step-Size Selection

Following Hairer et al. (1993, Chapter II.4), the step-size controller aims to select step sizes, as large as possible, but while satisfying componentwise, for $i \in \{1, \ldots, d\}$,

$$(D_n)_i \leq \varepsilon_i, \quad \varepsilon_i := \tau_{\text{abs}} + \tau_{\text{rel}} \cdot \max\left(\left|(\hat{y}_{n-1})_i\right|, \left|(\hat{y}_n)_i\right|\right), \tag{28}$$

where τ_{abs} and τ_{rel} are the prescribed absolute and relative tolerances, respectively, and $\hat{y}_{n-1} := E_0 \mu_{n-1}^F$, $\hat{y}_n := E_0 \mu_n^F$ are solution estimates of the numerical solver. To do so, we define the following measure of error as control objective,

$$E := \sqrt{\frac{1}{d} \sum_{i=1}^{d} \left(\frac{(D_n)_i}{\varepsilon_i}\right)^2}. \tag{29}$$

The proportional control algorithm compares the control objective E to 1 to find the optimal step size. If $E \leq 1$ holds, the computed step is accepted and the integration continues. Otherwise, the step is rejected as too inaccurate and is repeated. In both cases, a new step size which will likely satisfy Eq. (28) is computed as

$$h_{\text{new}} = h \cdot \rho \left(\frac{1}{E}\right)^{\frac{1}{q+1}}, \tag{30}$$

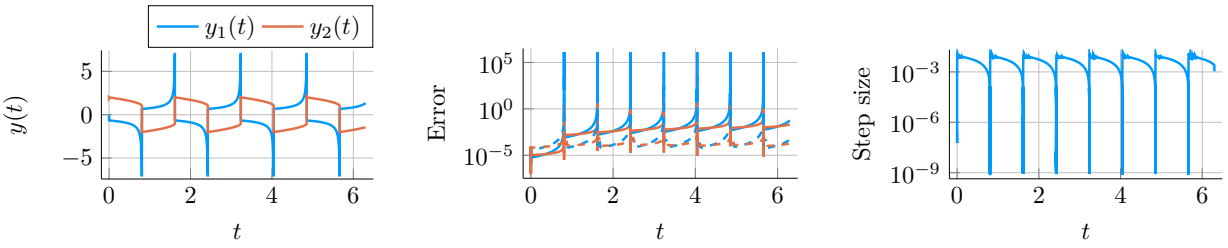


Figure 2: *Probabilistic solution with step-size adaptation of the Van der Pol equations.* Left: Mean of the posterior distribution over solutions, showing only values in the $(-7, 7)$ interval. Middle: Absolute errors (solid lines), and standard deviations of the posterior marginals as error estimates (dashed lines), shown in log-scale. Right: Step sizes of accepted steps throughout the solve. During the stiff phases, the step sizes get drastically decreased by the step-size controller.

making use of the local convergence rate of $q + 1$, as used by Schober et al. (2018) and shown by Kersting et al. (2020). The parameter $\rho \in (0, 1]$ is a safety factor to increase the probability that the next step will be acceptable, and we additionally limit the rate of change $\eta_{\min} \leq h_{n+1}/h_n \leq \eta_{\max}$ (Hairer et al., 1993). In our experiments, we set $\rho := 0.9$, $\eta_{min} := 0.2$, $\eta_{max} := 10$.

The formulation of the local error control objective in Eq. (27) also lends itself to other control algorithms. Notably, PI control (Gustafsson et al., 1988) can be an interesting alternative to proportional control when applied to mildly stiff problems, and has been successfully applied for the related class of Nordsieck methods by Bras et al. (2013).

5 RELATED WORK

Our contribution fits in the formulation of IVPs as problems of Bayesian state estimation (Tronarp et al., 2019, 2020). By using Gaussian filtering methods, these solvers are able to efficiently compute posterior distributions over solutions (Kersting and Hennig, 2016), which converge to the true solution at high polynomial rates (Kersting et al., 2020). In practice, the calibration of the posterior uncertainties depends on specific model hyperparameters. This paper reviews and extends previously proposed global and local calibration methods (Tronarp et al., 2019; Schober et al., 2018). The presented step-size controller builds on the algorithm suggested by Schober et al. (2018).

A different line of work on probabilistic numerical solvers for ODEs aims to represent the distribution over solution with a set of sample paths (Conrad et al., 2017; Abdulle and Garegnani, 2020; Lie et al., 2019; Teymur et al., 2018, 2016; Chkrebtii et al., 2016; Tronarp et al., 2019). While these methods are able to capture arbitrary, non-Gaussian distributions, they come at an increased computational cost.

6 EXPERIMENTS

To evaluate the presented methodology, we provide three sets of experiments. First, we highlight the practical necessity of step-size adaptation by solving a stiff version of the Van der Pol model. Next, we compare the different uncertainty calibration methods presented in Section 3. Finally, we assess the practical performance of the probabilistic ODE solvers by comparing to a classic Runge-Kutta 4/5 method. The code for the implementation and experiments is publicly available on gihub[1].

6.1 Stiff Van der Pol

The Van der Pol model (van der Pol, 1926) describes a non-conservative oscillator with non-linear damping, and can be written in the two-dimensional form:

$$\begin{aligned} \dot{y_1} &= y_2, \\ \dot{y_2} &= \mu \left((1 - y_1^2) y_2 - y_1 \right), \end{aligned} \tag{31}$$

with a positive stiffness constant $\mu > 0$.

To highlight the importance of step-size adaptation, and to demonstrate the A-stability of the EKS1 (Tronarp et al., 2020), we consider a very stiff version of the Van der Pol model and set $\mu = 10^6$. We solve the IVP on the time interval $[0, 6.3]$ with initial value $y(t_0) = [0, \sqrt{3}]^\intercal$ using the EKS1 algorithm with an IWP3 prior, for absolute and relative tolerance specified as 10^{-6} and 10^{-3}, respectively. We were not able to solve this same IVP with the EKS0 (which does not possess this stability property). The reference solution has been computed with the A-stable 5th order implicit Runge-Kutta method Radau IIA (Hairer and Wanner, 1996), implemented as `RadauIIA5` in the Julia DifferentialEquations.jl suite (Rackauckas and Nie, 2017), for absolute and relative tolerances set to 10^{-14}.

[1] `https://github.com/nathanaelbosch/capos`

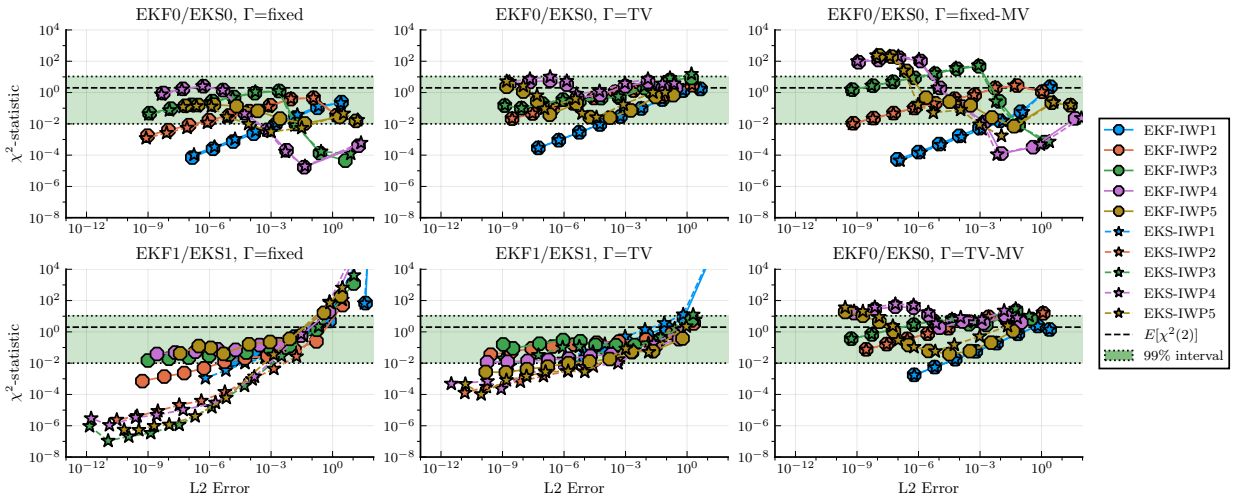


Figure 3: *Uncertainty calibration across configurations.* In each subfigure, a specific combination of filtering algorithm (EKF0/EKS0 or EKS1/EKF1) and calibration method is evaluated, the latter including fixed and time-varying (TV) diffusion models, as well as their multivariate versions (fixed-MV, TV-MV). A well-calibrated solver should provide χ^2-statistics inside the 99% credible interval (green).

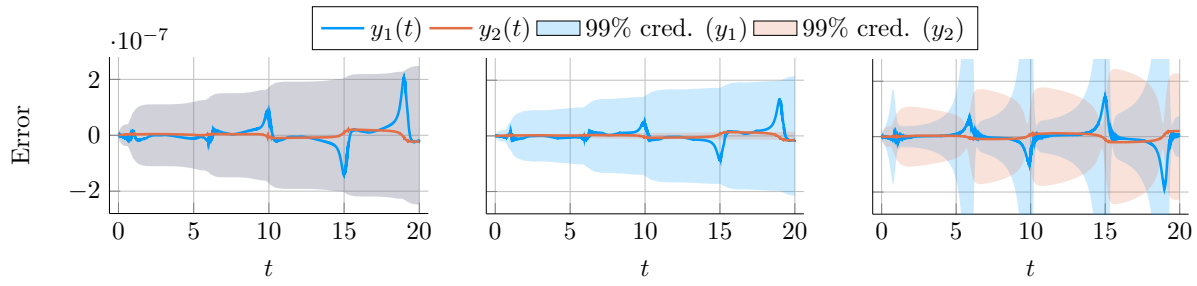


Figure 4: *Qualitative comparison of the different time-varying uncertainty models.* The EKS0 with scalar diffusion (left) estimates a single credible band shared accross for both dimensions, but the multivariate model (middle) is able to attribute different uncertainties to each dimension. By including information on the derivatives of the ODE, the EKS1 with scalar diffusion (right) returns structured and dynamic uncertainty estimate.

Figure 2 shows the results, including the solution posterior, errors and error estimates, and the step sizes during the solve. The posterior mean returned by the solver achieved a final error of $\|\hat{y}(T) - y^*(T)\| = 6.17 \times 10^{-2}$, where y^* denotes the reference solution. The step sizes shown in Fig. 2 (right) change drastically during the solve, and decrease from values $h \sim 10^{-2}$ down to $h < 10^{-8}$ during the stiff phases. If one were to solve this IVP without step-size control, that is, with EKS1 and an IWP3 prior but with fixed steps of size 10^{-8}, one would perform 6.3×10^8 solver iterations. Compared to the ~ 1 second runtime the adaptive step method required for its 23824 iterations (out of which 6977 steps were rejected), the fixed-step solver would require more than 7 hours. This demonstrates the importance of adaptive step-size control for efficient use of computational resources.

6.2 Comparison of Calibration Methods

We evaluate the various algorithms and calibration methods on the FitzHugh-Nagumo model, given by the ODE

$$\begin{aligned} \dot{y}_1 &= c\left(y_1 - \frac{y_1^3}{3} + y_2\right), \\ \dot{y}_2 &= -\frac{1}{c}\left(y_1 - a - b y_2\right), \end{aligned} \tag{32}$$

with parameters $(a = 0.2,\ b = 0.2,\ c = 3.0)$, initial value $y(0) = [-1, 1]^\intercal$, and time span $[0, 20]$.

In the first part of this experiment, the uncertainty calibration is evaluated across a large range of solver configurations and tolerances ($\tau_{\text{abs}} = 10^{-4}, \ldots, 10^{-13}$, $\tau_{\text{rel}} = 10^{-1}, \ldots, 10^{-10}$). To assess the quality of the uncertainty calibration, we use the χ^2-statistics (Bar-

Shalom et al., 2004) defined by

$$\chi^2 = \frac{1}{N}\sum_{i=1}^{N} r(t_i)^\intercal \operatorname{Cov}(y(t_i))^{-1} r(t_i), \qquad (33)$$

where $r(t_i) := (y^*(t_i) - \mathbb{E}[y(t_i)])$ are the residuals and $\mathbb{E}[y(t_i)]$ and $\operatorname{Cov}(y(t_i))$ are computed on the posterior distribution returned by the probabilistic solver. A well calibrated model achieves $\chi^2 \approx d$. If $\chi^2 < d$ or $\chi^2 > d$ we refer to the solution as *underconfident* or *overconfident*, respectively.

Figure 3 visualizes the full comparison of the uncertainty calibration methods and suggests some empirical findings. For most configurations the time-varying uncertainty models seem to be better calibrated than the fixed models. For a fixed choice of algorithm (e.g. EKF0) and order (e.g. IWP5) their calibration varies less across the different tolerance levels. The multi-variate models seem to not have a large impact on the χ^2-statistics, both for the fixed and time-varying models. The first-order linearization approaches EKF1/EKS1 tend to become underconfident, but achieve the lowest errors.

To complement this summarized evaluation based on the χ^2 statistics, we visualize the qualitative behaviour of the different time-varying uncertainty models in Fig. 4. To be comparable, all models share an IWP3 prior and tolerances $\tau_{\text{abs}} = 10^{-10}$, $\tau_{\text{rel}} = 10^{-7}$. The scalar, time-varying (TV) approach attributes the same credible bands to both dimensions (shown in grey), whereas the multivariate, time-varying (TV-MV) model is able to lift this restriction and estimates a large uncertainty for the first dimension (blue) and barely visible uncertainties for the second dimension (orange). However, only the first order linearization of the EKS1 seems to properly describe the structural properties of the true solution in its posterior estimate.

For experiments on additional problems, including classic work-precision diagrams for all methods, see Appendix B.1

6.3 Comparison with Dormand–Prince 4/5

This experiment assesses the performance of the developed methodology and compares the probabilistic solvers of 5th order to the classic, widely used Runge-Kutta 4/5 method by Dormand and Prince (1980), implemented as `DP5` in the Julia DifferentialEquations.jl suite (Rackauckas and Nie, 2017). The comparison was made on the Lotka-Volterra equations (Lotka, 1925; Volterra, 1928), which describe the dynamics of biological systems in which two species interact, one as a predator and the other as prey. The IVP is given by

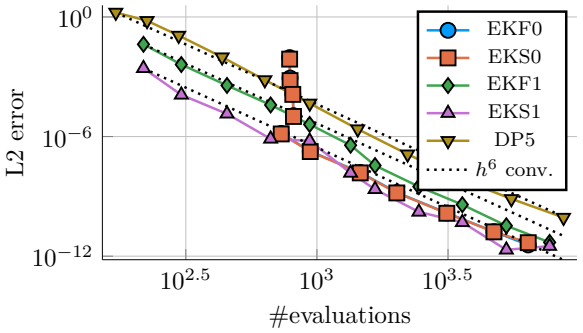


Figure 5: *Comparison to Dormand–Prince 4/5 (DP5).* All probabilistic ODE solvers use an IWP-5 prior and a scalar, time-varying diffusion model. The comparison is made on the Lotka-Volterra equations. The EKF0 and EKS0 performed similarly and are therefore difficult to separate visually in the figure.

the ODE

$$\begin{aligned} \dot{y_1} &= \alpha y_1 - \beta y_1 y_2, \\ \dot{y_2} &= -\gamma y_1 + \delta y_1 y_2, \end{aligned} \qquad (34)$$

with initial value $y(0) = [1, 1]^\intercal$ and parameters ($\alpha = 1.5$, $\beta = 1$, $\gamma = 3$, $\delta = 1$), on the time span $[0, 10]$.

Figure 5 shows the results in a work-precision diagram. We observe convergence rates of order 6 for all methods, one order higher than the expected global convergence rate of order 5 for the DP5 algorithm (Hairer et al., 1993) and for Gaussian ODE filters with IWP-5 prior (Kersting et al., 2020; Tronarp et al., 2020). It can also be seen that the EKF0 requires more function evaluations than expected for high-tolerance settings. Out of all compared methods, the EKS1 seems to require the least number of evaluations of the function and its Jacobian to achieve a specified error, matching the performance of the EKF0 and EKS0 while demonstrating a stable behaviour for high tolerances.

Similar work-precision diagrams for these methods on additional problems are provided in Appendix B.2.

7 CONCLUSION

In this paper, we introduced and discussed various models and methods for uncertainty calibration in Gaussian ODE filters, and presented parameter estimates for both fixed and time-varying, as well as scalar and multivariate diffusion models. The probabilistic observation model of these methods provides a calibrated objective for local error control, enabling the implementation of classic step-size selection algorithms.

The resulting, efficiently computable posteriors have

been empirically evaluated for a wide range of tolerance levels, demonstrating decent error calibration in particular for the time-varying diffusion models. Of all compared methods, the first-order linearization of the EKS1 seems to provide the most expressive posterior covariances, while also efficiently computing accurate solutions – requiring, in our benchmarks, less evaluations than the well-known Dormand–Prince 4/5 method to reach a specified tolerance level.

Acknowledgements

The authors gratefully acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) through Project ADIMEM (FKZ 01IS18052B), and financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting N. Bosch.

The authors are grateful to Nicholas Krämer for many valuable discussions. They further thank Hans Kersting, Jonathan Wenger and Agustinus Kristiadi for helpful feedback on the manuscript.

References

Abdulle, A. and Garegnani, G. (2020). Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration. *Statistics and Computing*, 30(4):907–932.

Bar-Shalom, Y., Li, X., and Kirubarajan, T. (2004). *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley.

Bashforth, F. and Adams, J. C. (1883). *An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid*. University Press.

Bettencourt, J., Johnson, M. J., and Duvenaud, D. (2019). Taylor-mode automatic differentiation for higher-order derivatives in jax.

Bras, M., Cardone, A., and D'Ambrosio, R. (2013). Implementation of explicit Nordsieck methods with inherent quadratic stability. *Mathematical Modelling and Analysis*, 18(2):289–307.

Byrne, G. D. and Hindmarsh, A. C. (1975). A polyalgorithm for the numerical solution of ordinary differential equations. *ACM Trans. Math. Softw.*

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc.

Chkrebtii, O. A., Campbell, D. A., Calderhead, B., and Girolami, M. A. (2016). Bayesian solution uncertainty quantification for differential equations. *Bayesian Anal.*, 11(4):1239–1267.

Cockayne, J., Oates, C., Sullivan, T., and Girolami, M. (2019). Bayesian probabilistic numerical methods. *SIAM Rev.*, 61:756–789.

Conrad, P. R., Girolami, M., Särkkä, S., Stuart, A., and Zygalakis, K. (2017). Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Statistics and Computing*, 27(4):1065–1082.

Dormand, J. R. and Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26.

Griewank, A. and Walther, A. (2000). Evaluating derivatives - principles and techniques of algorithmic differentiation, second edition. In *Frontiers in applied mathematics*.

Gustafsson, K., Lundh, M., and Söderlind, G. (1988). A PI stepsize control for the numerical solution of ordinary differential equations. *BIT Numerical Mathematics*, 28(2):270–287.

Hairer, E., Norsett, S., and Wanner, G. (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8. Springer-Verlag.

Hairer, E. and Wanner, G. (1996). *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, volume 14. Springer-Verlag.

Hennig, P., Osborne, M. A., and Girolami, M. (2015). Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179).

Hochbruck, M., Ostermann, A., and Schweitzer, J. (2008). Exponential Rosenbrock-type methods. *SIAM J. Numer. Anal.*, 47(1):786–803.

Kersting, H. and Hennig, P. (2016). Active uncertainty calibration in bayesian ode solvers. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, page 309–318, Arlington, Virginia, USA. AUAI Press.

Kersting, H., Sullivan, T. J., and Hennig, P. (2020). Convergence rates of gaussian ode filters. *Statistics and Computing*, 30(6):1791–1816.

Krämer, N. and Hennig, P. (2020). Stable implementation of probabilistic ode solvers.

Lie, H. C., Stuart, A. M., and Sullivan, T. J. (2019). Strong convergence rates of probabilistic integrators for ordinary differential equations. *Statistics and Computing*, 29(6):1265–1283.

Lindström, E., Madsen, H., and Nielsen, J. (2018). *Statistics for Finance: Texts in Statistical Science*. Chapman and Hall/CRC.

Lotka, A. (1925). *Elements of Physical Biology*. Williams & Wilkins.

Nordsieck, A. (1962). On numerical integration of ordinary differential equations. *Mathematics of Computation*, 16:22–49.

Oates, C. J. and Sullivan, T. J. (2019). A modern retrospective on probabilistic numerics. *Statistics and Computing*, 29(6):1335–1351.

Rackauckas, C. and Nie, Q. (2017). DifferentialEquations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1).

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France. PMLR.

Rosenbrock, H. H. (1963). Some general implicit processes for the numerical solution of differential equations. *Comput. J.*, 5(4):329–330.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*, volume 3 of *Institute of Mathematical Statistics textbooks*. Cambridge University Press.

Schober, M., Särkkä, S., and Hennig, P. (2018). A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*.

Schweppe, F. (1965). Evaluation of likelihood functions for Gaussian signals. *IEEE transactions on Information Theory*, 11(1):61–70.

Särkkä, S. and Solin, A. (2019). *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press.

Teymur, O., Lie, H. C., Sullivan, T., and Calderhead, B. (2018). Implicit probabilistic integrators for odes. In *Advances in Neural Information Processing Systems 31*, pages 7244–7253. Curran Associates, Inc.

Teymur, O., Zygalakis, K., and Calderhead, B. (2016). Probabilistic linear multistep methods. In *Advances in Neural Information Processing Systems 29*, pages 4321–4328. Curran Associates, Inc.

Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. (2019). Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29(6):1297–1315.

Tronarp, F., Sarkka, S., and Hennig, P. (2020). Bayesian ode solvers: the maximum a posteriori estimate. *CoRR*.

van der Pol, B. (1926). On "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):978–992.

Volterra, V. (1928). Variations and Fluctuations of the Number of Individuals in Animal Species living together. *ICES Journal of Marine Science*, 3(1):3–51.

Nathanael Bosch, Philipp Hennig, Filip Tronarp

Calibrated Adaptive Probabilistic ODE Solvers Supplementary Materials

A Proof of Proposition 1

Proof. The proof is structured as follows. First, we show by induction that an initial covariance $\Sigma_0 = \breve{\Sigma}_0 \otimes \Gamma$ implies covariances $\Sigma_n^P = \breve{\Sigma}_n^p \otimes \Gamma$, $\Sigma_n^F = \breve{\Sigma}_n^F \otimes \Gamma$, and $S_n = \breve{S}_n \cdot \Gamma$, for all n. Then, for measurement covariances S_n of such form, we can compute the (quasi) maximum likelihood estimate $\hat{\Gamma}$.

Assume $\Sigma_{n-1}^F = \breve{\Sigma}_{n-1}^F \otimes \Gamma$. Using the mixed product property and the associativity of the Kronecker product, the covariance of the prediction Σ_n^P can be written as

$$\Sigma_n^P = A_n \Sigma_{n-1}^F A_n^\intercal + Q_n = \left(\breve{A}_n \otimes I_d\right)\left(\breve{\Sigma}_{n-1}^F \otimes \Gamma\right)\left(\breve{A}_n \otimes I_d\right)^\intercal + \left(\breve{Q}_n \otimes \Gamma\right) = \left(\breve{A}_n \breve{\Sigma}_{n-1}^F \breve{A}_n^\intercal + \breve{Q}_n\right) \otimes \Gamma = \breve{\Sigma}_n^P \otimes \Gamma,$$

where $\breve{\Sigma}_n^P := \breve{A}_n \breve{\Sigma}_{n-1}^F \breve{A}_n^\intercal + \breve{Q}_n$. Next, using $H_n = e_1^\intercal \otimes I_d$ (EKF0), the measurement covariance S_n is given by

$$S_n = H_n \Sigma_n^P H_n^\intercal = (e_1^\intercal \otimes I_d)\left(\breve{\Sigma}_n^P \otimes \Gamma\right)(e_1^\intercal \otimes I_d)^\intercal = \left(e_1^\intercal \breve{\Sigma}_n^P e_1\right) \otimes \Gamma = \breve{S}_n \cdot \Gamma.$$

Finally, the filtering covariance can be computed with the update step Eq. (11) as

$$\begin{aligned}
\Sigma_n^F &= \Sigma_n^P - K_n S_n K_n \\
&= \Sigma_n^P - \Sigma_n^P H_n^\intercal S_n^{-1} H_n \Sigma_n^P \\
&= \left(\breve{\Sigma}_n^P \otimes \Gamma\right) - \left(\breve{\Sigma}_n^P \otimes \Gamma\right)(e_1^\intercal \otimes I_d)^\intercal \left(\breve{S}_n \otimes \Gamma\right)^{-1} (e_1^\intercal \otimes I_d)\left(\breve{\Sigma}_n^P \otimes \Gamma\right) \\
&= \left(\breve{\Sigma}_n^P - \breve{\Sigma}_n^P e_1 \breve{S}_n^{-1} e_1^\intercal \breve{\Sigma}_n^P\right) \otimes \Gamma \\
&= \breve{\Sigma}_n^F \otimes \Gamma,
\end{aligned}$$

with $\breve{\Sigma}_n^P := \breve{\Sigma}_n^P - \breve{\Sigma}_n^P e_1 \breve{S}_n^{-1} e_1^\intercal \breve{\Sigma}_n^P$. This concludes the first part of the proof.

It is left to compute the (quasi) MLE by maximizing the log-likelihood $\log p(z_{1:N}) = \log \prod_{n=1}^N \mathcal{N}(z_n; \hat{z}_n, S_n)$. For diagonal Γ and zero measurements $z_n = 0$, we obtain

$$\begin{aligned}
\hat{\Gamma} &= \arg\max_\Gamma \log p(z_{1:N}) \\
&= \arg\max_\Gamma \sum_{n=1}^N \log \mathcal{N}\left(0; \hat{z}_n, \breve{S}_n \cdot \Gamma\right) \\
&= \arg\max_\Gamma \sum_{n=1}^N \left(-\frac{1}{2}\log|\breve{S}_n \cdot \Gamma| - \frac{1}{2}\hat{z}_n^\intercal \left(\breve{S}_n \cdot \Gamma\right)^{-1} \hat{z}_n\right) \\
&= \arg\max_\Gamma \sum_{n=1}^N \left(-\frac{\log \breve{S}_n^d + \sum_{i=1}^d \log \Gamma_{ii}}{2} - \sum_{i=1}^d \frac{(\hat{z}_n)_i^2}{2\breve{S}_n \Gamma_{ii}}\right) \\
&= \arg\max_\Gamma \sum_{i=1}^d \left(-\frac{N \log \Gamma_{ii}}{2} - \sum_{n=1}^N \frac{(\hat{z}_n)_d^2}{2\breve{S}_n \Gamma_{ii}}\right).
\end{aligned}$$

Each diagonal element $\hat{\Gamma}_{ii}$ can be computed independently by taking the derivative setting it to zero:

$$0 = -\frac{N}{2\Gamma_{ii}} + \sum_{n=1}^{N} \frac{(\hat{z}_n)_d^2}{2\breve{S}_n \Gamma_{ii}^2}, \qquad \forall i \in \{1, \ldots, d\}.$$

The solution to this equation provides the (quasi) maximum likelihood estimate for $\hat{\Gamma}$

$$\hat{\Gamma}_{ii} = \frac{1}{N} \sum_{n=1}^{N} \frac{(\hat{z}_n)_i^2}{\breve{S}_n}, \qquad \forall i \in \{1, \ldots, d\}.$$

□

B Additional Experiments

These experiments include two additional IVPs.

Logistic Equation We consider an initial value problem, given by the logistic equation

$$\dot{y}(t) = ry(t)(1 - y(t)), \tag{35}$$

with parameter $r = 3$, integration interval $[0, 2.5]$, and initial value $y(0) = 0.1$. Its exact solution is given by

$$y^*(t) = \frac{\exp(rt)}{1/y_0 - 1 + \exp(rt)}. \tag{36}$$

Brusselator The Brusselator is a model for multi-molecular chemical reactions, given by the ODEs

$$\begin{aligned} \dot{y_1} &= 1 + y_1^2 y_2 - 4y_1, \\ \dot{y_2} &= 3y_1 - y_1^2 y_2. \end{aligned} \tag{37}$$

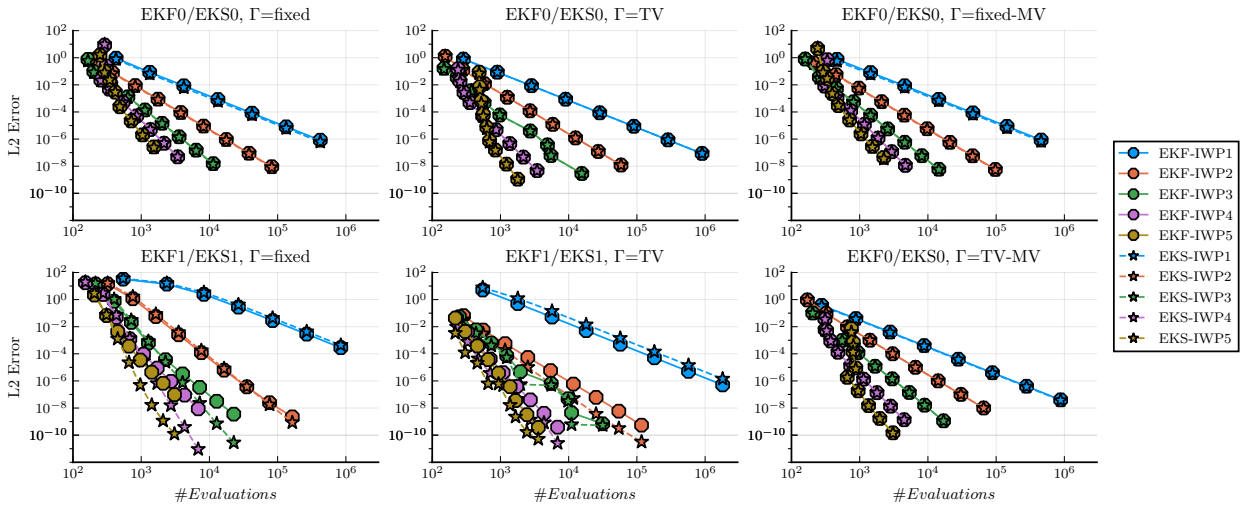
We consider an IVP with initial value $y(0) = [1.5, 3]$ on the time span $[0, 10]$.

B.1 Performance and Calibration on Additional Problems

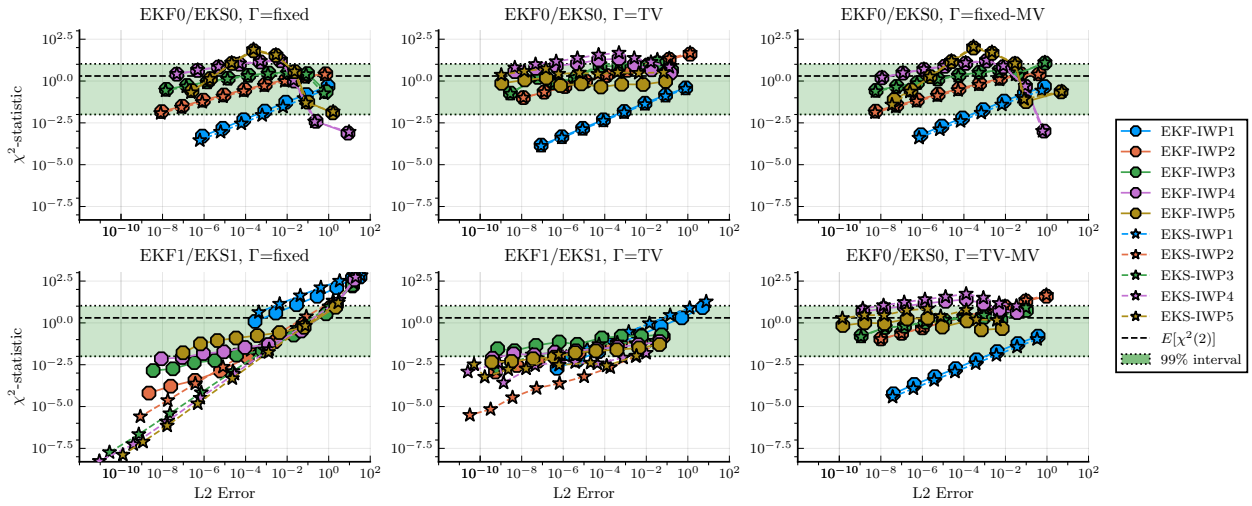
This section extends the results of Section 6.2 with evaluations on additional problems: Lotka-Volterra (Fig. 6), logistic equation (Fig. 7), FitzHugh-Nagumo (Fig. 8), and Brusselator (Fig. 9). In addition, all figures contain classic work-precision diagrams which visualize the relation of achieved error and number of evaluations, where both the evaluations of the vector field and of its Jacobian are counted.

B.2 Comparison to Dormand-Prince 4/5

Comparison of the probabilistic solvers to the classic Dormand-Prince 4/5 method on additional problems. Figure 10 presents the resulting work-precision diagrams. This extends the results of Section 6.3.

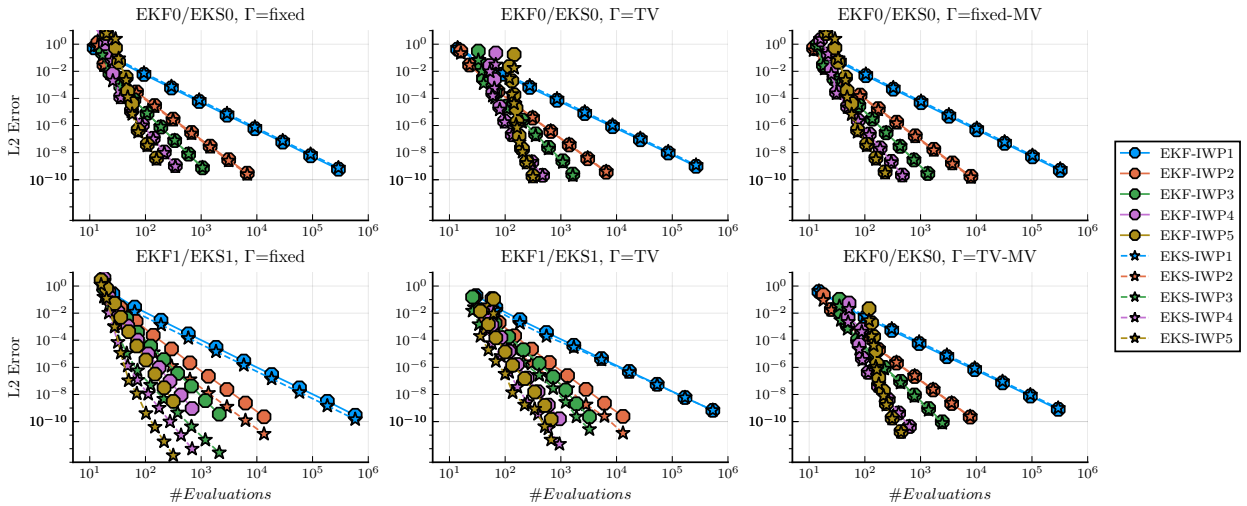


(a) Work-precision diagrams.

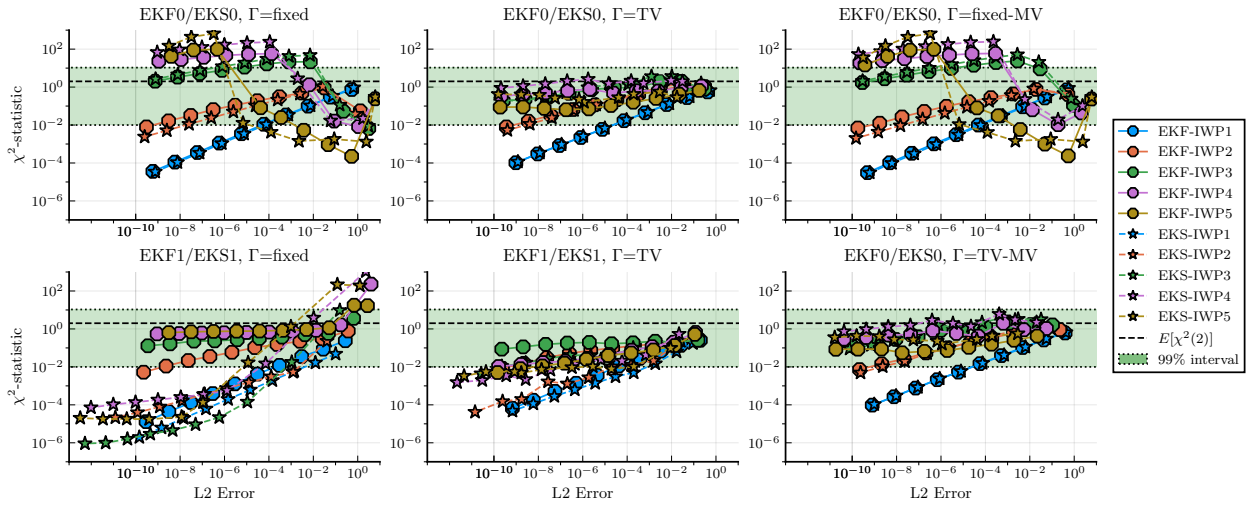


(b) Uncertainty Calibration.

Figure 6: *Accuracy and uncertainty calibration across configurations on the Lotka-Volterra equations.* In each subfigure, a specific combination of filtering algorithm (EKF0/EKS0 or EKS1/EKF1) and calibration method is evaluated, the latter including fixed and time-varying (TV) diffusion models, as well as their multivariate versions (fixed-MV, TV-MV).

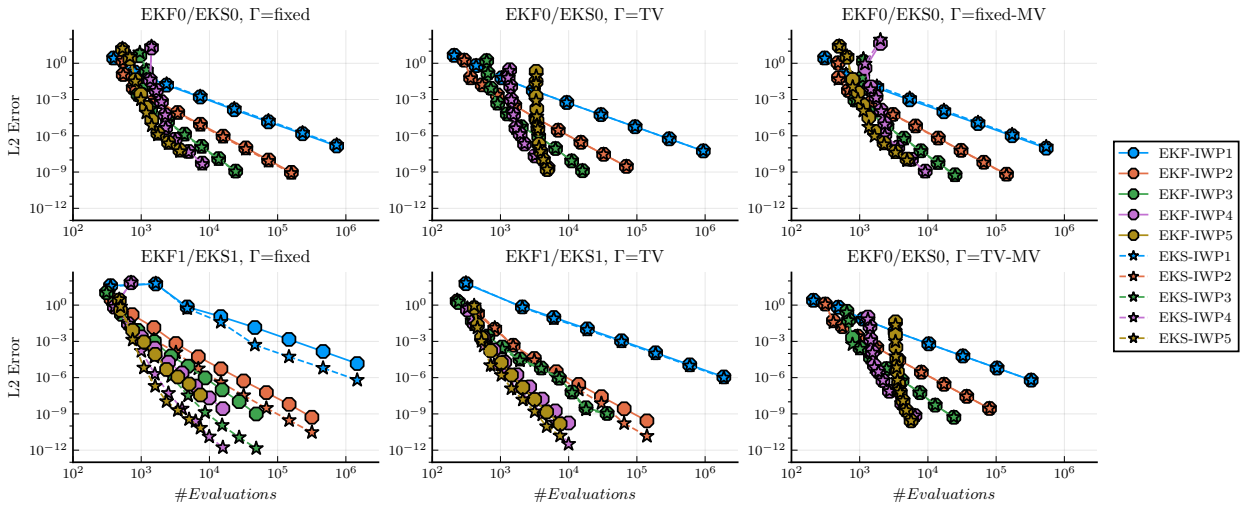


(a) Work-precision diagrams.

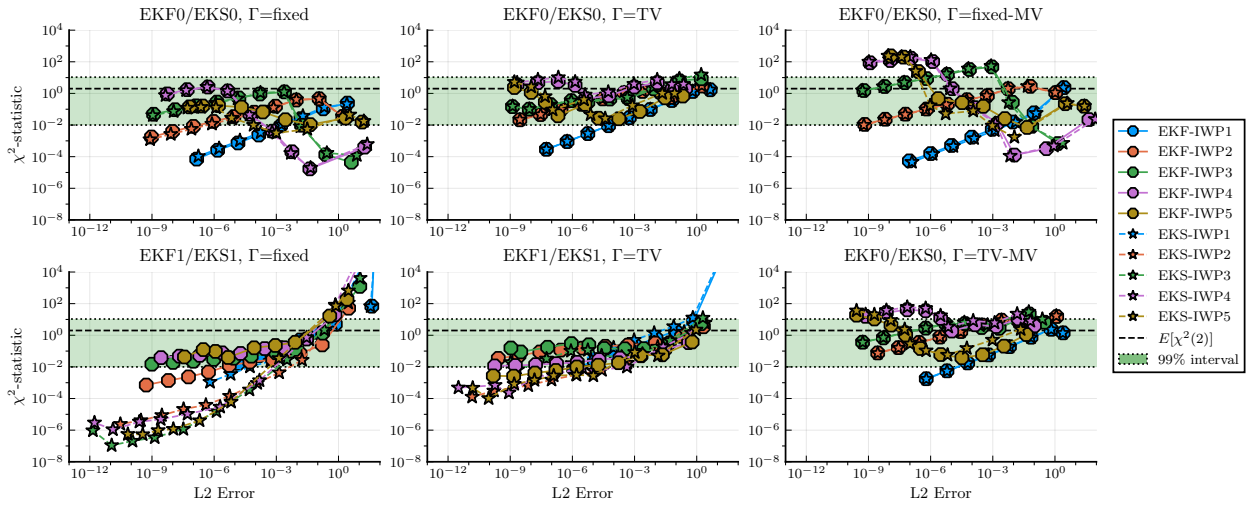


(b) Uncertainty Calibration.

Figure 7: *Accuracy and uncertainty calibration across configurations on the logistic equation.* In each subfigure, a specific combination of filtering algorithm (EKF0/EKS0 or EKS1/EKF1) and calibration method is evaluated, the latter including fixed and time-varying (TV) diffusion models, as well as their multivariate versions (fixed-MV, TV-MV).

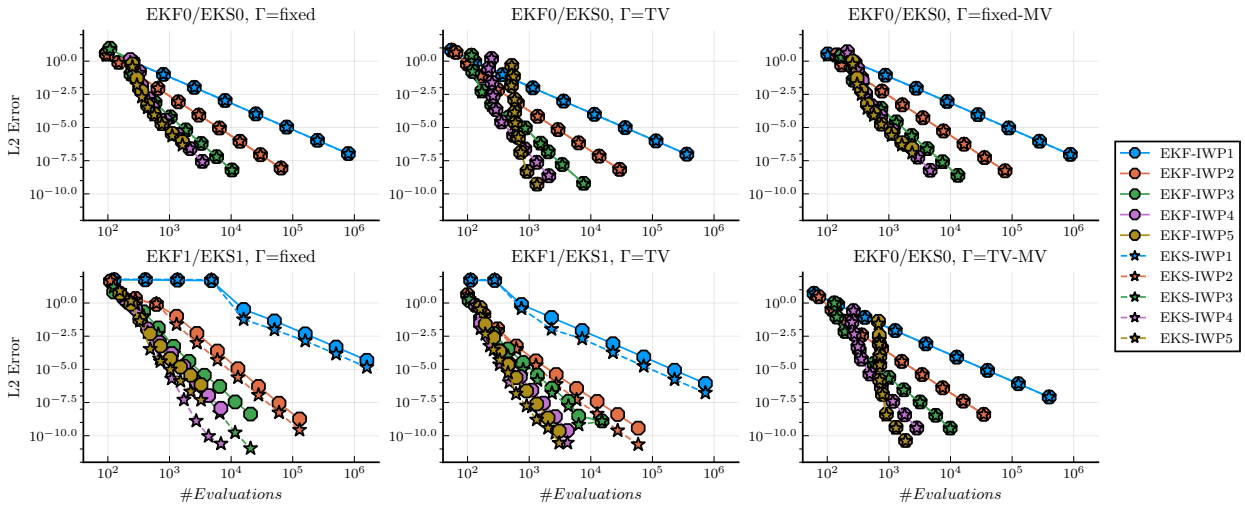


(a) Work-precision diagrams.

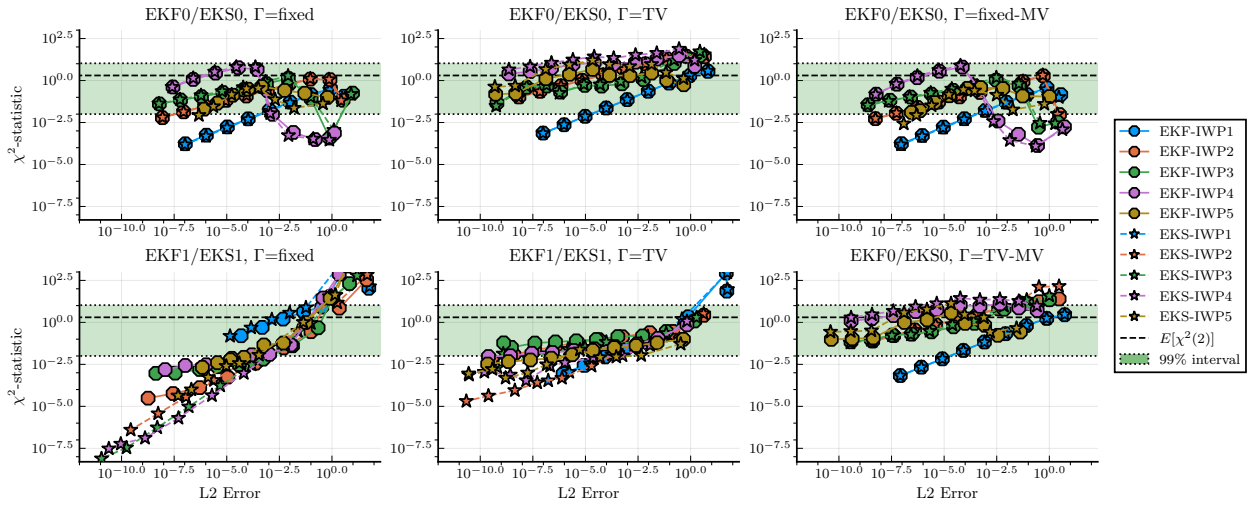


(b) Uncertainty Calibration.

Figure 8: *Accuracy and uncertainty calibration across configurations on the FitzHugh-Nagumo equations.* In each subfigure, a specific combination of filtering algorithm (EKF0/EKS0 or EKS1/EKF1) and calibration method is evaluated, the latter including fixed and time-varying (TV) diffusion models, as well as their multivariate versions (fixed-MV, TV-MV).

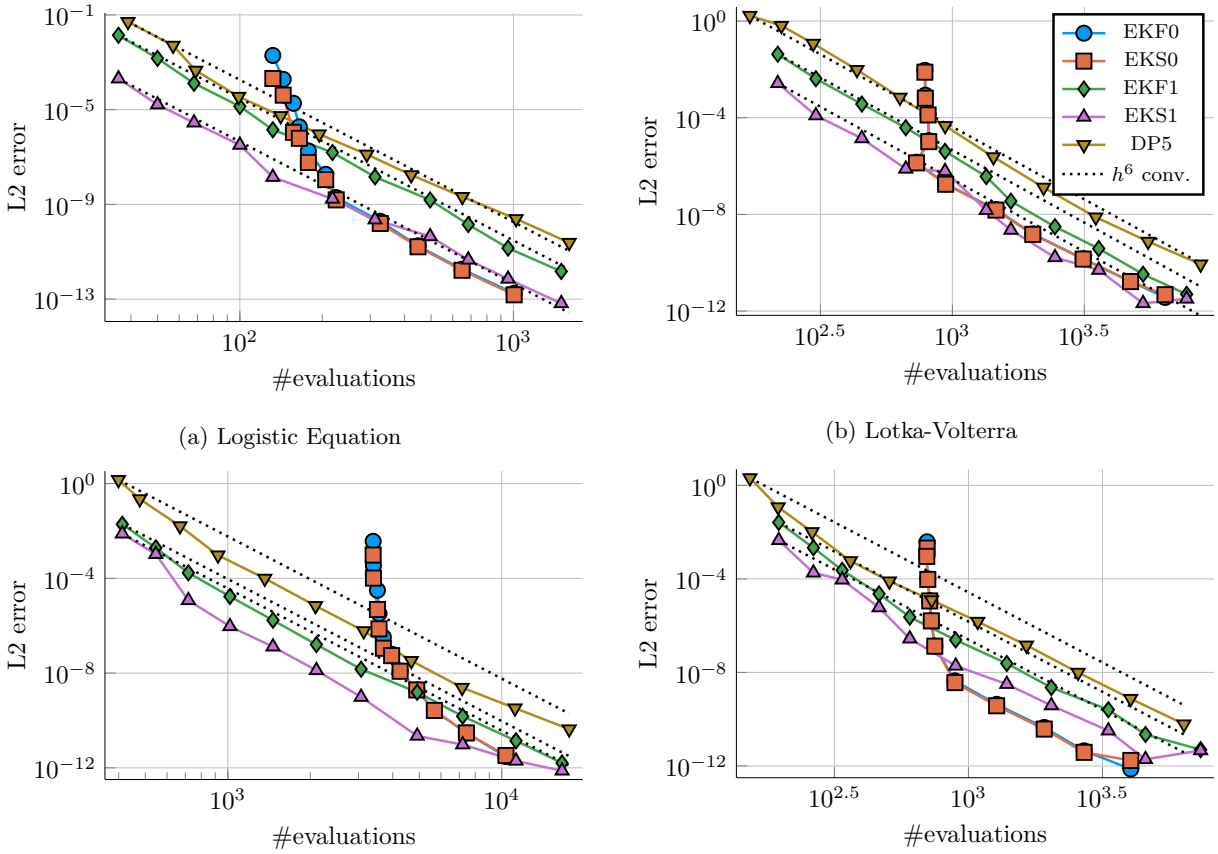


(a) Work-precision diagrams.



(b) Uncertainty Calibration.

Figure 9: *Accuracy and uncertainty calibration across configurations on the Brusselator equations.* In each subfigure, a specific combination of filtering algorithm (EKF0/EKS0 or EKS1/EKF1) and calibration method is evaluated, the latter including fixed and time-varying (TV) diffusion models, as well as their multivariate versions (fixed-MV, TV-MV).



(a) Logistic Equation

(b) Lotka-Volterra

(c) FitzHugh-Nagumo

(d) Brusselator

Figure 10: *Comparison to Dormand-Prince 4/5 (DP5).* All probabilistic ODE solvers use an IWP-5 prior and a scalar, time-varying diffusion model.

Pick-and-Mix Information Operators for Probabilistic ODE Solvers

Nathanael Bosch[1] **Filip Tronarp**[1] **Philipp Hennig**[1,2]
[1]University of Tübingen, [2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
{nathanael.bosch, filip.tronarp, philipp.hennig}@uni-tuebingen.de

Abstract

Probabilistic numerical solvers for ordinary differential equations compute posterior distributions over the solution of an initial value problem via Bayesian inference. In this paper, we leverage their probabilistic formulation to seamlessly include additional information as general likelihood terms. We show that second-order differential equations should be directly provided to the solver, instead of transforming the problem to first order. Additionally, by including higher-order information or physical conservation laws in the model, solutions become more accurate and more physically meaningful. Lastly, we demonstrate the utility of flexible information operators by solving differential-algebraic equations. In conclusion, the probabilistic formulation of numerical solvers offers a flexible way to incorporate various types of information, thus improving the resulting solutions.

1 INTRODUCTION

Throughout science and engineering, dynamical systems are frequently described with ordinary differential equations (ODEs). But in many systems of interest, the differential equation harbors additional information not directly accessible to the numerical algorithm used to solve it. For example, physical systems often follow high-order dynamics and preserve quantities such as energy, mass, or angular momentum. To efficiently compute meaningful solutions, practitioners have to carefully choose from a wide range of numerical solvers, such as Runge–Kutta methods (Hairer et al., 1993), Nyström methods for second-order ODEs (Nyström, 1925), structure-preserving integrators (Hairer et al., 2006), and many more. Each of these methods can be seen as a laboriously custom-designed way to encode specific kinds of information. In this paper, we present a more flexible, unified approach to include additional knowledge into numerical ODE solutions, by leveraging the framework of *probabilistic numerics*.

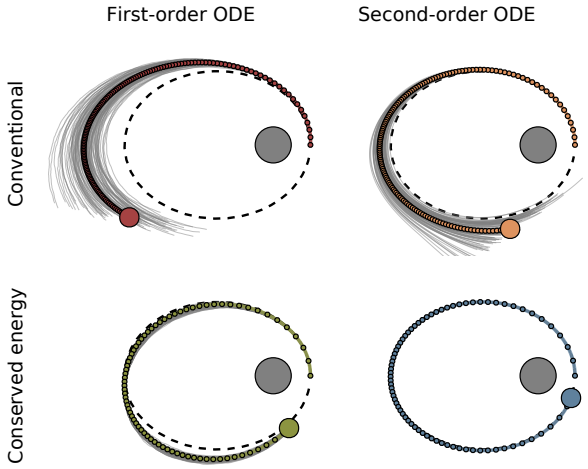


Figure 1: *Faithful modeling of ODE information improves probabilistic solutions.* Informing the solver about the second-order structure of the Kepler problem (●) increases the accuracy over its first-order counterpart (●). Adding physical information about the conservation of energy and angular momentum greatly improves the solution, *even for increased step sizes* (●). Full information leads to the best results (●). The dynamical system is described in Supplement A.6.

In probabilistic numerics (Hennig et al., 2015; Oates and Sullivan, 2019), numerical problems are formulated as problems of probabilistic inference. Probabilistic numerical methods return *distributions over solutions*. Such methods can quantify their own approximation error through samples and other structured quantities – a functionality typically not provided by classic nu-

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

merical methods. This paper builds on probabilistic numerical ODE solvers based on Bayesian filtering and smoothing (Schober et al., 2019; Tronarp et al., 2019). These "ODE filters" have been shown to converge with polynomial rates (Kersting et al., 2020; Tronarp et al., 2021) and their efficiency has been demonstrated on a range of both non-stiff and stiff problems (Krämer and Hennig, 2020; Bosch et al., 2021).

Contributions Probabilistic ODE solvers are defined by two parts: the prior and the likelihood. In the basic version of such solvers, the likelihood is completely defined by the vector field. But, as we show in this work, their formulation is sufficiently flexible to allow for a much richer language. By formulating the likelihood in terms of flexible information operators, information about higher-order derivatives and conserved quantities can be represented with the same semantics as the ODE information itself. We demonstrate the utility of the proposed framework in four case studies:

1. *Second-order differential equations:* Solving second-order ODEs directly, instead of transforming them to first order, greatly improves the efficiency of probabilistic solvers.
2. *Additional second-derivative information:* Information about higher-order derivatives can be additionally included in the joint inference process to increase the solution accuracy.
3. *Systems with conserved quantities:* By including conservation laws into the model, probabilistic solutions become not only more accurate but also more physically meaningful.
4. *Differential-algebraic equations (DAEs):* With the corresponding information operator, probabilistic solvers can be extended to DAEs.

2 PROBABILISTIC ODE SOLVERS

This section introduces filtering-based probabilistic ODE solvers. Consider an initial value problem (IVP)

$$\dot{y} = f(y(t), t), \qquad \forall t \in [0, T], \tag{1}$$

with vector field $f : \mathbb{R}^{d+1} \to \mathbb{R}^d$ and initial value $y(0) = y_0 \in \mathbb{R}^d$. Instead of computing a single point estimate (as done by classic numerical algorithms), ODE filters compute *probabilistic* ODE solutions. That is, they approximate posterior distributions of the form

$$p\left(y(t) \mid y(t_0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=0}^{N}\right), \tag{2}$$

for a chosen time-discretization $\{t_n\}_{n=1}^{N}$. Thereby, they estimate not only the ODE solution, but also the unavoidable, global approximation error that arises due to discretization.

In the following, we pose the probabilistic numerical ODE solution as a problem of Bayesian state estimation, the solution of which can be efficiently approximated with extended Kalman filtering. For a more thorough introduction we refer to Tronarp et al. (2019).

2.1 Numerical ODE Solutions As Inference

Integrated Wiener Process Priors *A priori*, we model the unknown ODE solution $y(t)$ by a q-times integrated Wiener process (IWP). More precisely, define

$$Y(t) = \left[Y^{(0)}(t), Y^{(1)}(t), \ldots, Y^{(q)}(t)\right] \tag{3}$$

as the solution of a linear, time-invariant stochastic differential equation of the form

$$\mathrm{d}Y^{(i)}(t) = Y^{(i+1)}(t)\,\mathrm{d}t, \qquad i = 0, \ldots q-1, \tag{4a}$$

$$\mathrm{d}Y^{(q)}(t) = \Gamma^{1/2}\,\mathrm{d}W(t), \tag{4b}$$

$$Y(0) \sim \mathcal{N}\left(\mu_0, \Sigma_0\right), \tag{4c}$$

driven by a d-dimensional Wiener process W. The matrix $\Gamma^{1/2}$ is the symmetric square-root of some positive semi-definite matrix $\Gamma \in \mathbb{R}^{d\times d}$ and $\mu_0 \in \mathbb{R}^{d(q+1)}$, $\Sigma_0 \in \mathbb{R}^{d(q+1)\times d(q+1)}$ are the initial mean and covariance. Then, $Y^{(i)}$ models the i-th derivative of unknown ODE solution y and we write $y \sim \mathrm{IWP}(q)$.

Discrete-Time Transitions The process $Y(t)$ satisfies transition densities (Särkkä and Solin, 2019)

$$Y(t+h)\,|\,Y(t) \sim \mathcal{N}\left(A(h)Y(t), Q(h)\right). \tag{5}$$

The matrices $A(h), Q(h) \in \mathbb{R}^{d(q+1)\times d(q+1)}$ denote the transition matrix and the process noise covariance. For the chosen IWP(q) prior, it holds

$$A(h) = \breve{A}(h) \otimes I_d, \qquad Q(h) = \breve{Q}(h) \otimes \Gamma, \tag{6}$$

and the matrices $\breve{A}(h), \breve{Q}(h) \in \mathbb{R}^{(q+1)\times(q+1)}$ are known in closed form (Kersting et al., 2020):

$$\breve{A}_{ij}(h) = \mathbb{I}_{i\le j}\frac{h^{j-1}}{(j-i)!}, \tag{7a}$$

$$\breve{Q}_{ij}(h) = \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!}. \tag{7b}$$

Measurement Process To relate the prior process to the ODE solution, we define a measurement model in terms of an *information operator* (Cockayne et al., 2019; Tronarp et al., 2019), similar to the likelihood models used in gradient matching (Calderhead et al., 2009; Wenk et al., 2020). Define

$$\mathcal{Z}[y](t) := \dot{y}(t) - f(y(t), t). \tag{8}$$

The operator $\mathcal{Z}$ maps the true ODE solution (see Eq. (1)) to a known quantity, namely the zero function; $\mathcal{Z}[y] \equiv 0$. On the other hand, the action of the information operator on the process Y can be expressed in terms of the following non-linear function

$$z\left(t, Y\right) := \mathcal{Z}[Y^{(0)}](t) = Y^{(1)}(t) - f\left(Y^{(0)}(t), t\right). \quad (9)$$

Once again, if $Y^{(0)}$ solves the ODE (Eq. (1)) exactly, we have $z(t, Y) \equiv 0$. Consequently, *inferring* the true ODE solution y reduces to conditioning the prior Y on $z(t, Y) = 0$. This inference problem is the subject of the next section.

2.2 Approximate Gaussian Inference

To enable tractable inference, we discretize time to a grid $\{t_n\}_{n=1}^N \subset [0, T]$ and condition the process $Y(t)$ only on discrete observations $z_n := z(t_n, Y(t_n)) = 0$. The resulting non-linear Gauss–Markov regression problem is well-known in the Bayesian filtering and smoothing literature (Särkkä, 2013). Its solution can be efficiently approximated with the extended Kalman filter (EKF), as Gaussian distributions

$$p\left(Y(t_n) \,|\, z_{1:n}\right) \approx \mathcal{N}\left(\mu_n, \Sigma_n\right). \quad (10)$$

In a nutshell, the EKF algorithm proceeds by iterating the following steps (Särkkä, 2013, Section 5.2):

- `PREDICT`: Given $Y(t_n) \mid z_{1:n} \sim \mathcal{N}\left(\mu_n, \Sigma_n\right)$ and the Gaussian transitions of Eq. (5), we can extrapolate to $Y(t_{n+1}) \mid z_{1:n} \sim \mathcal{N}\left(\mu_{n+1}, \Sigma_{n+1}\right)$, with

$$\mu^-_{n+1} = A(h_n)\mu_n, \quad (11a)$$
$$\Sigma^-_{n+1} = A(h_n)\Sigma_n A(h_n)^\intercal + Q(h_n), \quad (11b)$$

  where $h_n := t_{n+1} - t_n$.

- `UPDATE`: To include information about the new measurement z_{n+1} into $Y(t_{n+1})$ approximate $Y(t_{n+1}) \mid z_{1:n+1} \sim \mathcal{N}\left(\mu_{n+1}, \Sigma_{n+1}\right)$, with

$$\hat{z}_{n+1} = z\left(t_{n+1}, \mu^-_{n+1}\right) \quad (12a)$$
$$S_{n+1} = H_{n+1}\Sigma^-_{n+1}H^\intercal_{n+1}, \quad (12b)$$
$$K_{n+1} = \Sigma^-_{n+1}H^\intercal_{n+1}S^{-1}_{n+1}, \quad (12c)$$
$$\mu_{n+1} = \mu^-_{n+1} + K_{n+1}(z_{n+1} - \hat{z}_{n+1}), \quad (12d)$$
$$\Sigma_{n+1} = \Sigma^-_{n+1} - K_{n+1}S_{n+1}K^\intercal_{n+1}. \quad (12e)$$

  In a standard EKF, the matrix H_{n+1} denotes the Jacobian of the measurement model z, evaluated at μ^-_{n+1}. For z as defined in Eq. (9), we have $H_n := E_1 - J_f(E_0\mu_n, t_n)E_0$, where the matrices $E_i \in \mathbb{R}^{d \times d(q+1)}$ denote projection matrices to the i-th component of the state Y, that is $E_i Y = Y^{(i)}$.

  We call the resulting ODE solver `EK1` (Tronarp et al., 2019). Alternatively, Schober et al. (2019) use a zeroth-order approximation of the vector field, i.e. $H_n := E_1$. We refer to this solver as `EK0`.

Remark 1 (Smoothing)**.** *A Rauch–Tung–Striebel backward pass turns the filtering distribution into a smoothing posterior (Särkkä, 2013). At the final time point, the filtering and smoothing posteriors coincide.*

Remark 2 (Alternative inference schemes)**.** *The unscented Kalman filter (Julier and Uhlmann, 2004) can be used for Gaussian filtering, but requires multiple evaluations of the vector field at each time step. Particle filtering (Särkkä, 2013) can provide more descriptive, non-Gaussian ODE posterior estimates (Tronarp et al., 2019). But, similarly to sampling-based approaches to probabilistic ODE solutions (Chkrebtii et al., 2016; Conrad et al., 2017; Abdulle and Garegnani, 2020; Teymur et al., 2018), this expressivity comes at increased computational cost. In comparison, the EKF provides computationally efficient approximate inference.*

2.3 Practical Considerations

Calibration The posterior covariances returned by the solver depend on the choice of diffusion parameter Γ (recall Eq. (4)). Good uncertainty quantification therefore requires the estimation of Γ. In ODE filters, this is usually done by approximately maximizing the marginal likelihood of the observed data $p(z_{1:N})$ (Tronarp et al., 2019). This procedure also extends to more general, time-varying diffusion models Γ_n which have been proposed for greater flexibility (and for step-size adaptation; see below) (Schober et al., 2019). Refer to Bosch et al. (2021) for more detail.

Step-Size Adaptation In practice, computationally efficient ODE solvers typically rely on adaptive step-size selection (Hairer et al., 1993, Chapter II.4). We follow the presentation of Bosch et al. (2021) and control a local error estimate, derived from the measurement z, with a PI control algorithm (Gustafsson et al., 1988).

3 INFORMATION OPERATORS

The previous section established ODE filters as efficient algorithms for computing probabilistic numerical solutions of first-order ODEs. In the following, we extend their formulation to a broader class of problems and include additional types of information, by generalizing the underlying information operators.

The vector-field information enters the inference problem through the specified measurement model: f (recall Eq. (1)) only appears in the information operator $\mathcal{Z}$ (respectively z; see Eqs. (8) and (9)). However,

Table 1: Common problem settings and corresponding information operators.

Description	Equation	Information operator
First-order ODE	$\dot{y}(t) = f\left(y(t), t\right)$	$z(t, Y) := Y^{(1)} - f\left(Y^{(0)}, t\right)$
Second-order ODE	$\ddot{y}(t) = f\left(\dot{y}(t), y(t), t\right)$	$z(t, Y) := Y^{(2)} - f\left(Y^{(1)}, Y^{(0)}, t\right)$
Mass matrix DAE	$M\dot{y}(t) = f\left(y(t), t\right)$	$z(t, Y) := MY^{(1)} - f\left(Y^{(0)}, t\right)$
Invariances	$g\left(y(t), \dot{y}(t)\right) = 0$	$z(t, Y) := g\left(Y^{(0)}, Y^{(1)}\right)$
Chain rule	$\ddot{y}(t) = J_f\left(y(t)\right) \cdot \dot{y}(t)$	$z(t, Y) := Y^{(2)} - J_f\left(Y^{(0)}\right) \cdot Y^{(1)}$

the approximate inference algorithm itself (the EKF; see Section 2.2) does not rely on the specific form of the measurements; except for calibration and step-size adaptation, which we separately discuss below. To extend the ODE filter framework, we consider more general information operators, of the form

$$\mathcal{Z} \in \mathcal{I}_y := \{\mathcal{Z} : \mathcal{Z}[y] \equiv 0\}\,. \tag{13}$$

As before, they map some unknown function of interest y to the known zero function. But, this general form is not restricted to first-order ODEs. For example, given an energy-preserving system with second-order dynamics, we can formulate a corresponding operator to define its probabilistic solution (as will be shown in Section 4.3). Table 1 provides a summary of the problem settings and the corresponding operators considered in this paper, written in the functional form $z(t, Y) := \mathcal{Z}[Y^{(0)}](t)$. Before moving to our case studies, where each model will be explained in more detail, we discuss practical details and implementation.

Inference with Multiple Information Operators Some problems of interest provide multiple types of information about the true solution, for example as additional derivatives (Section 4.2) or physical conservation laws (Section 4.3). Formally, this amounts to an information operator $\mathcal{Z} \in \mathcal{I}_y$ that can be partitioned as $\mathcal{Z}[y] = [\mathcal{Z}_1[y]^\intercal, \mathcal{Z}_2[y]^\intercal]^\intercal$, with $\mathcal{Z}_1, \mathcal{Z}_2 \in \mathcal{I}_y$, and corresponding functional representation

$$z(t, Y) = [z_1(t, Y)^\intercal,\ z_2(t, Y)^\intercal]^\intercal\,. \tag{14}$$

It is still possible to update jointly on both measurement models in a single EKF update step on z; this strategy is chosen in Section 4.2. However, performing two separate update steps can sometimes be preferable (Raitoharju and Piché, 2019; Raitoharju et al., 2016, 2017). In this case, each measurement model is linearized separately in the partially updated state. This strategy is chosen in Section 4.3.

Calibration and Step-Size Adaptation The approaches for calibration and adaptive step-size selection discussed in Section 2.3 do not strictly depend on the specific information operator, but they were developed in the context of first-order ODEs (Bosch et al., 2021). There, the information operator is d-dimensional, i.e. $z(t, Y) \in \mathbb{R}^d$, and describes the *local defect*. We found that this formulation can be extended to settings with a different problem structure (in this work, second-order ODEs and DAEs), but for settings with multiple sources of information (here, additional derivatives or invariances) special care has to be taken. To conveniently consider user-specified relative tolerance levels, the local error should be of the same dimension as the ODE solution. Thus, in Sections 4.2 and 4.3, only the part of the measurement model that relates to the given differential equation is considered for calibration and step-size adaptation.

4 CASE STUDIES

We evaluate the presented framework in four case studies. First, we apply the probabilistic solver to second-order ODEs. We investigate the difference between solving such problems directly, by selecting the correct information operator, and solving the algebraically (but not numerically) equivalent first-order ODEs. Second, we augment the probabilistic numerical solver for first-order ODEs with second-derivative information, which can be computed from the ODE via the chain rule. Third, we consider Hamiltonian systems in which the total energy is conserved over time, and we evaluate the influence of this information on the probabilistic numerical solution. Fourth, we demonstrate how probabilistic solvers can be extended to solve semi-explicit differential-algebraic equations.

Implementation The implementation follows the practices suggested by Krämer and Hennig (2020) and includes exact initialization, preconditioned state transitions, and a square-root implementation. All experiments are implemented in the Julia programming language (Bezanson et al., 2017). Reference solutions are computed with DifferentialEquations.jl (Rackauckas and Nie, 2017). All experiments run on a single, consumer-level CPU. Code for the implementation and experiments is publicly available on GitHub.[1]

[1] `github.com/nathanaelbosch/pick-and-mix`

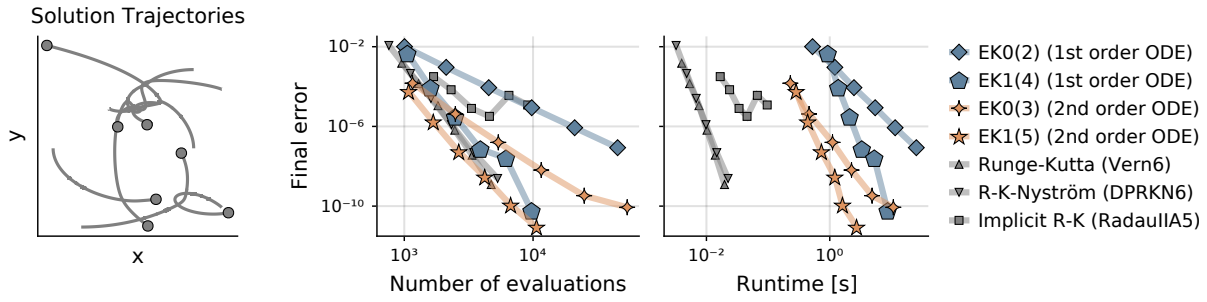


Figure 2: *Second-order ODEs should be solved directly.* The Pleiades system describes the motion of seven stars in a plane (left). Solving this problem directly in second order, compared to solving the equivalent first-order ODE, improves accuracy and efficiency, both in the number of function evaluations (center) and runtime (right).

4.1 Second-Order Differential Equations

This first case study demonstrates how information about the problem structure, such as the order of the ODE, can improve probabilistic solutions. To this end, consider an autonomous, *second*-order ODE

$$\ddot{y}(t) = f\left(\dot{y}(t), y(t)\right), \qquad \forall t \in [0, T], \tag{15}$$

with vector field $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ and initial values $y(0) = y_0$, $\dot{y}(0) = \dot{y}_0$.

Second-order ODEs can be transformed to first order by defining a new variable $\widetilde{y} := (\dot{y}, y)$. They can therefore, in principle, be solved by any generic solver. However, doubling the dimension of the ODE can increase both the solver runtime and memory cost. Specialized non-probabilistic solvers such as Nyström methods have been specifically developed to circumvent this issue (Nyström, 1925; Hairer et al., 1993). In this section, we follow a similar (but much simpler) approach and present a direct application of probabilistic solvers to second-order ODEs.

The motivation is twofold. First, a duplication of the ODE dimension leads to a 4x increase in memory cost and 8x runtime, since the EKF algorithm relies on matrix-matrix operations on the state covariances. Second, the structure of the transformed problem is not a good fit for the integrated Wiener process prior. After transformation, the first derivative $\dot{y}$ appears both in $\widetilde{y}$ and $\frac{\mathrm{d}\widetilde{y}}{\mathrm{d}t}$. It is therefore modeled with both an IWP(q) and IWP$(q-1)$ prior *at the same time* (recall Section 2.1). Both of these shortcomings can be circumvented by solving the second-order problem directly.

Solver Setup The second-order ODE (Eq. (15)) induces an information operator of the form

$$z(t, Y) = Y^{(2)} - f\left(Y^{(1)}, Y^{(0)}\right). \tag{16}$$

We consider two linearizations:

$$H := \begin{cases} E_2, & \texttt{(EK0)} \\ E_2 - \frac{\partial f}{\partial y} \cdot E_0 - \frac{\partial f}{\partial \dot{y}} \cdot E_1, & \texttt{(EK1)} \end{cases} \tag{17}$$

named in correspondence to the existing probabilistic solvers for first-order problems presented in Section 2.2.

Experiment Setup We evaluate the solvers on the Pleiades problem (Hairer et al., 1993, Chapter II.10), a system of 14 second-order ODEs (full problem definition in Supplement A.1). All solvers use adaptive steps and a time-varying diffusion model (Bosch et al., 2021). We compare the resulting mean absolute errors at final time T, referred to as "final error". Thus, the solutions were not smoothed. For a fair comparison, the orders of the first-order solvers are lowered by one compared to their second-order counterparts, such that their highest modeled derivatives coincide.

Results The work-precision diagrams in Fig. 2 show that second-order ODEs are solved both more efficiently and more accurately than their first-order counterparts. We observe not only an improvement in absolute runtime, but also a reduced error even for comparable numbers of vector-field evaluations. Figure 2 also compares the solvers to well-established non-probabilistic methods, including an explicit Runge–Kutta solver (`Vern6`; Verner, 2010), a Nyström method (`DPRKN6`; Dormand and Prince, 1987), and an implicit solver (`RadauIIA5`; Hairer and Wanner, 1999). While these classic solvers require a comparable number of vector-field evaluations, they exhibit a reduced absolute runtime. Since probabilistic solvers have the same cubic complexity as the classic, implicit `RadauIIA5`, we suspect that this discrepancy is partly due to the well-optimized implementation of the DifferentialEquations.jl library (Rackauckas and Nie, 2017). On the other hand, probabilistic ODE solvers provide strictly more functionality than non-probabilistic methods, thus a certain increase

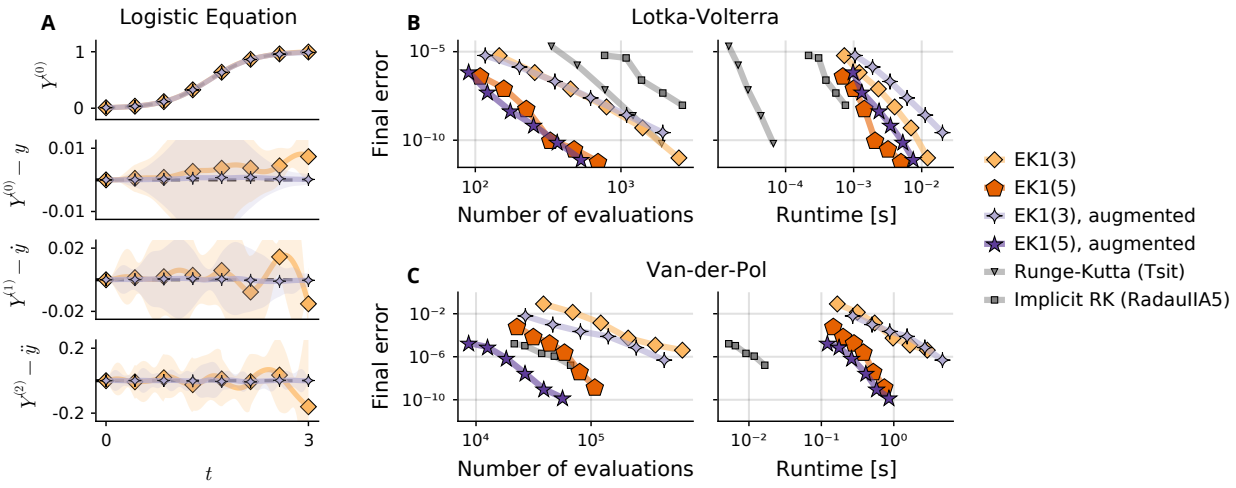


Figure 3: *Additional second-derivative information can improve probabilistic solutions.* On a fixed time discretization, additional information about second-derivatives reduces the approximation error (A). For adaptive-step solvers, it depends on the specific problem. On the non-stiff Lotka–Volterra problem, the utility of the additional information seems limited (B). However, the benefit of second-derivative information on the stiff Van–der–Pol problem outweighs the additional computational cost and leads to reduced runtimes (C).

in runtime is expected. As demonstrated by this case study, this paper further reduces the gap between probabilistic and non-probabilistic methods by providing ODE filters for second-order differential equations.

4.2 First-Order ODEs with Additional Second-Derivative Information

In this section, we augment the probabilistic solver with additional second-derivative information, that can be derived from a standard, first-order problem. For this, consider an autonomous, explicit, first-order ODE

$$\dot{y}(t) = f\left(y(t)\right), \qquad \forall t \in [0, T], \tag{18}$$

with vector field $f : \mathbb{R}^d \to \mathbb{R}^d$ and initial value $y(0) = y_0 \in \mathbb{R}^d$. Second derivatives of the true solution can be derived from Eq. (18) by differentiating both sides and applying the chain rule. We obtain

$$\ddot{y}(t) = J_f(y(t)) \cdot f\left(y(t)\right), \tag{19}$$

where J_f is the Jacobian of f.

Solver Setup Equations (18) and (19) motivate a measurement model $z(t, Y) := [z_1(t, Y)^\intercal, z_2(t, Y)^\intercal]^\intercal$,

$$z_1(t, Y) := Y^{(1)} - f\left(Y^{(0)}\right), \tag{20a}$$

$$z_2(t, Y) := Y^{(2)} - J_f(Y^{(0)}) f\left(Y^{(0)}\right). \tag{20b}$$

In this evaluation, we consider exact linearizations of both z_1 and z_2 (computed with automatic differentiation). Furthermore, the solvers update on both measurement models in a single, joint update step.

Fixed-Step Results We first evaluate the proposed method in a simplified setting to visualize the effect of additional second-derivative information. To this end, consider the logistic ODE (defined in Supplement A.2), and fixed-step solvers with $\Delta t = 3/7$. Figure 3 (A) shows the results. Both solvers approximate the true solution, but the more informed solver achieves lower approximation errors and has reduced uncertainties.

Adaptive-Step Results Next, we evaluate the proposed method on the non-stiff Lotka–Volterra problem (Supplement A.3) and the stiff Van–der–Pol model (Supplement A.4), in conjunction with adaptive step-size selection and a dynamic diffusion model (Bosch et al., 2021). Figure 3 shows the resulting work-precision diagrams. On the Lotka–Volterra problem (B), we observe that additional information does not strictly lead to improvements. Here, the original `EK1` solvers seem preferable. On the other hand, the additional second-derivative information leads to increased accuracy and even to a reduction in the number of vector-field evaluations on the stiff Van–der–Pol problem (C).

4.3 Systems with Conserved Quantities

In this case study, we demonstrate how additional knowledge about conserved quantities of the modeled dynamical system can be provided to the probabilistic solver. To this end, we consider *Hamiltonian problems*, a particular class of dynamical systems of the form

$$\dot{p} = -\frac{\partial H}{\partial q}(p, q), \qquad \dot{q} = \frac{\partial H}{\partial p}(p, q), \tag{21}$$

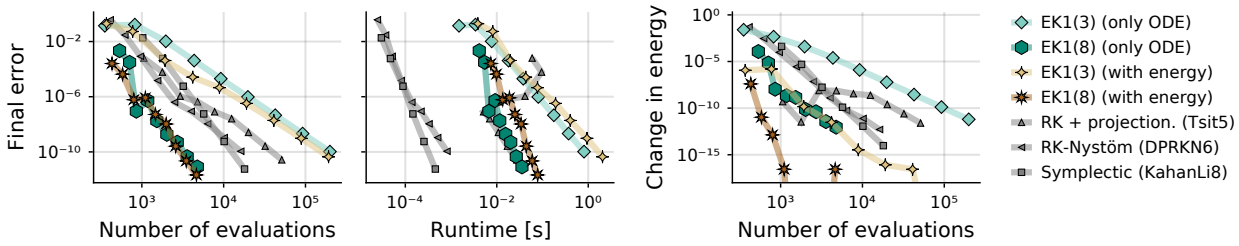


Figure 4: *Work-precision diagram of numerical solvers with and without energy conservation.* Additional information about the total energy in the dynamical Hénon–Heiles system can improve the accuracy of the solution (left). This comes with additional computational cost and increases the runtime (center). But as a result, the total energy is conserved more strictly and solutions become more physically meaningful (right).

where the Hamiltonian $H : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ describes the total energy in the dynamical system. Hamiltonian problems form an important class of ODEs in the context of geometric numerical integration (Hairer et al., 2006) since their trajectories *preserve the Hamiltonian*. That is, for a solution $(p(t), q(t))$ of such problems, the Hamiltonian $H(p(t), q(t))$ is constant, and it holds

$$g\left(p(t), q(t)\right) := H(p(t), q(t)) - H(p(0), q(0)) \equiv 0. \tag{22}$$

Geometric integrators aim to preserve this structure in their numerical approximation. In the following, we present a probabilistic solver for Hamiltonian problems that includes this additional information into its inference process to improve its solution estimates.

Solver Setup The problems considered in this section can all be written as second-order ODEs, with $(\dot{y}, y) := (p, q)$. Together with the conservation law of Eq. (22), this motivates a partitioned measurement model $z(t, Y) := [z_1(t, Y)^\intercal, z_2(t, Y)^\intercal]^\intercal$, with

$$z_1(t, Y) := Y^{(2)} - f\left(Y^{(0)}\right), \tag{23a}$$

$$z_2(t, Y) := g(Y^{(1)}, Y^{(0)}), \tag{23b}$$

where f denotes the vector field of the corresponding ODE. As in the previous section, all considered methods rely on exact linearizations of the measurement models. In addition, the solvers perform a *partitioned* EKF update. That is, they separately linearize and update first on the ODE information z_1 and then on the conserved quantity z_2 – a procedure that parallels established "projection methods" used with non-probabilistic ODE solvers (Hairer et al., 2006, Section IV.4).

Problem Setting We mainly consider the Hénon–Heiles model which describes a star moving around a galactic center (Henon and Heiles, 1964). The full problem definition is given in Supplement A.5. We compare probabilistic solvers with and without additional information about the conservation of energy, for various orders ($q \in \{3, 8\}$). All solvers use adaptive steps and dynamic diffusion models. Since we evaluate the error at the final time point, smoothing is not required.

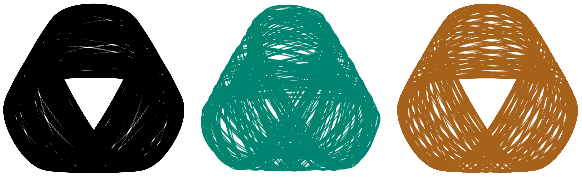


Figure 5: *Conservation stabilizes long simulations.* Probabilistic numerical simulations of the Hénon–Heiles problem over long time horizons, computed with adaptive steps and low precision, deteriorate over time (middle) and deviate strongly from the true trajectory (left). By including energy-preservation into the solver, long-term simulations become more accurate (right).

Results Figure 4 shows the results in multiple work-precision diagrams. We observe that the additional information leads, in some configurations, to improved accuracies, but comes with an increase in absolute runtime. However, the probabilistic solvers enforce the conservation of energy very strictly – even in comparison to non-probabilistic approaches that are particularly well suited for this problem setting, including a Runge–Kutta solver (`Tsit5`; Tsitouras, 2011) combined with a projection method (Hairer et al., 2006, Section IV.4), a Runge–Kutta–Nyström solver (`DPRKN6`; Dormand and Prince, 1987), and a symplectic integrator (`KahanLi8`; Kahan and Li, 1997). This structural preservation is of major concern to obtain physically meaningful solutions and stable long-term simulations of Hamiltonian systems (Hairer et al., 2006). The conservation of energy is therefore often of higher importance than a sole reduction in the (Euclidean) error. Following this motivation, Fig. 5 shows how energy preservation stabilizes long-term simulations with probabilistic solvers. Finally, Fig. 6 demonstrates on the Kepler problem

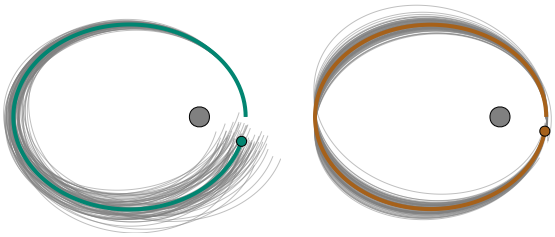


Figure 6: *Energy preservation affects the covariances.* (For a clearer visualization of the samples, covariances are inflated by a factor of 3 (left) and 300 (right).) The samples of a standard probabilistic solution of the Kepler problem appear physically implausible (left). By informing the solver about the conservation of energy and angular momentum, the posterior distribution becomes more physically meaningful (right).

(defined in Supplement A.6) how physical information influences not only the mean of the solution estimate, but also its covariances.

4.4 Differential-Algebraic Equations

In our final case study, we demonstrate how flexible information operators can be used to extend probabilistic solvers to completely new problem classes. To this end, we consider systems of the form

$$M\dot{y}(t) = f\left(y(t)\right), \qquad \forall t \in [0, T], \tag{24}$$

with vector field $f : \mathbb{R}^d \to \mathbb{R}^d$, initial values $y(0) = y_0$, and *mass matrix* $M \in \mathbb{R}^{d\times d}$. If M is singular, the system can not be rewritten as a regular ODE and we call Eq. (24) a differential-algebraic equation (DAE). For instance, in the Robertson DAE considered in this case study, we have $M = \mathrm{diag}\left([1, 1, 0]\right)$. The system thus describes two ODEs and one algebraic equation.

DAEs arise naturally in many dynamical systems, such as multi-body dynamics, chemical kinetics, or optimal control (Brenan et al., 1996). Their numerical simulation is notoriously challenging and often requires specialized methods; only a specific subset of classic ODE solvers is able to solve the problem given in Eq. (24) (Petzold, 1982). To the best of our knowledge, this work presents the first *probabilistic* DAE solver.

Solver Setup To encode the DAE information of Eq. (24), we define a measurement model

$$z(t, Y) := MY^{(1)} - f\left(Y^{(0)}\right). \tag{25}$$

In our experiments, we consider exact linearizations

$$H := M \cdot E_1 - J_f\left(Y^{(0)}\right) \cdot E_0, \tag{26}$$

together with adaptive step-size selection and dynamically calibrated diffusions. Smoothing is not required.

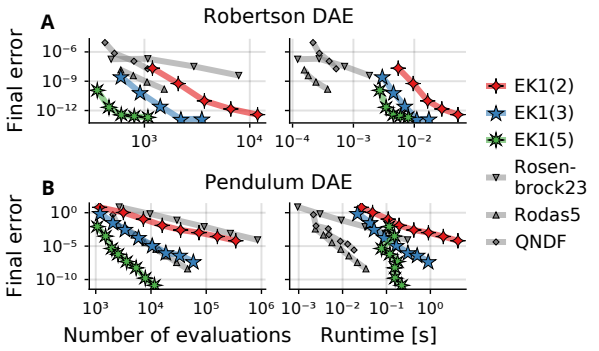


Figure 7: *With the correct information operator, probabilistic ODE solvers can solve semi-explicit DAEs.*

Experiment and Results To investigate the utility of the proposed methods, we compare probabilistic solvers with various orders ($q \in \{2, 3, 5\}$) to three non-probabilistic DAE solvers: Rosenbrock methods of order 2 and 5 (`Rosenbrock23` & `Rodas5`; Hairer and Wanner, 1996) and an adaptive-order multistep method (`QNDF`; Shampine and Reichelt, 1997). All methods are evaluated on the stiff Robertson DAE and on a non-stiff pendulum DAE (defined in Supplements A.7 and A.8). Figure 7 shows the resulting work-precision diagrams. As one would expect, increasing the number of steps leads to reduced error. In addition, we observe higher convergence rates for solvers of higher order. While the proposed solvers display higher runtimes than their classic counterparts, the differences are comparable to our results in the other case studies. In the number of vector-field evaluations, probabilistic and non-probabilistic solvers appear comparable. In summary, the proposed probabilistic solvers demonstrate good performance on the considered DAEs.

5 CONCLUSION

We have shown how to improve ODE solvers by drawing on various sources of information, within the framework of probabilistic numerics. The proposed algorithm performs efficient inference with extended Kalman filtering and can leverage existing methods for uncertainty calibration and step-size adaptation. In four case studies, we demonstrated how information about problem structure, additional derivatives, and conserved quantities can be used to improve the solver performance and the quality of the posterior distributions. By providing a flexible and efficient means to encode mechanistic knowledge beyond the ODE itself, our proposed framework further reduces the gap between probabilistic and non-probabilistic methods and thereby enriches the interface of mechanistic inference and simulation.

Acknowledgements

The authors gratefully acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) through Project ADIMEM (FKZ 01IS18052B), and financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting N. Bosch. The authors are grateful to Nicholas Krämer for many valuable discussions and thank Jonathan Wenger and Jonathan Schmidt for helpful feedback on the manuscript.

References

Abdulle, A. and Garegnani, G. (2020). Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration. *Statistics and Computing*, 30.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59.

Bosch, N., Hennig, P., and Tronarp, F. (2021). Calibrated adaptive probabilistic ODE solvers. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*. PMLR.

Brenan, K., Campbell, S., Campbell, S., and Petzold, L. (1996). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics.

Calderhead, B., Girolami, M., and Lawrence, N. (2009). Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

Chkrebtii, O. A., Campbell, D. A., Calderhead, B., and Girolami, M. A. (2016). Bayesian Solution Uncertainty Quantification for Differential Equations. *Bayesian Analysis*, 11(4).

Cockayne, J., Oates, C., Sullivan, T., and Girolami, M. (2019). Bayesian probabilistic numerical methods. *SIAM review*, 61.

Conrad, P. R., Girolami, M., Särkkä, S., Stuart, A., and Zygalakis, K. (2017). Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Statistics and Computing*, 27.

Dormand, J. and Prince, P. (1987). Runge-Kutta-Nystrom triples. *Computers & Mathematics with Applications*, 13.

Gustafsson, K., Lundh, M., and Söderlind, G. (1988). A PI stepsize control for the numerical solution of ordinary differential equations. *BIT Numerical Mathematics*, 28.

Hairer, E., Lubich, C., and Wanner, G. (2006). *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg.

Hairer, E., Norsett, S., and Wanner, G. (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8. Springer-Verlag.

Hairer, E. and Wanner, G. (1996). *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, volume 14. Springer-Verlag.

Hairer, E. and Wanner, G. (1999). Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics*, 111.

Hennig, P., Osborne, M. A., and Girolami, M. (2015). Probabilistic numerics and uncertainty in computations. *Proceedings. Mathematical, physical, and engineering sciences*, 471.

Henon, M. and Heiles, C. (1964). The applicability of the third integral of motion: Some numerical experiments. *The Astronomical Journal*, 69.

Julier, S. and Uhlmann, J. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92.

Kahan, W. and Li, R.-C. (1997). Composition constants for raising the orders of unconventional schemes for ordinary differential equations. *Mathematics of computation*, 66.

Kersting, H., Sullivan, T. J., and Hennig, P. (2020). Convergence rates of Gaussian ODE filters. *Statistics and Computing*, 30.

Krämer, N. and Hennig, P. (2020). Stable implementation of probabilistic ODE solvers. *arXiv:2012.10106 [stat.ML]*.

Nyström, E. (1925). *Über die numerische integration von differentialgleichungen*. Acta Societatis Scientiarum Fennicae. Finn. Literaturges.

Oates, C. J. and Sullivan, T. J. (2019). A modern retrospective on probabilistic numerics. *Statistics and Computing*, 29.

Petzold, L. (1982). Differential/algebraic equations are not ODE's. *Siam Journal on Scientific and Statistical Computing*, 3.

Rackauckas, C. and Nie, Q. (2017). DifferentialEquations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5.

Raitoharju, M. and Piché, R. (2019). On computational complexity reduction methods for Kalman filter extensions. *IEEE Aerospace and Electronic Systems Magazine*, 34.

Raitoharju, M., Piché, R., Ala-Luhtala, J., and Ali-Löytty, S. (2016). Partitioned update Kalman filter. *Journal of Advances in Information Fusion*, 11.

Raitoharju, M., Ángel F. García-Fernández, and Piché, R. (2017). Kullback–Leibler divergence approach to partitioned update Kalman filter. *Signal Processing*, 130.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*, volume 3 of *Institute of Mathematical Statistics textbooks*. Cambridge University Press.

Schober, M., Särkkä, S., and Hennig, P. (2019). A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29.

Shampine, L. F. and Reichelt, M. W. (1997). The Matlab ODE suite. *SIAM journal on scientific computing*, 18.

Särkkä, S. and Solin, A. (2019). *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press.

Teymur, O., Lie, H. C., Sullivan, T., and Calderhead, B. (2018). Implicit probabilistic integrators for ODEs. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. (2019). Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29.

Tronarp, F., Särkkä, S., and Hennig, P. (2021). Bayesian ODE solvers: the maximum a posteriori estimate. *Statistics and Computing*, 31.

Tsitouras, C. (2011). Runge–Kutta pairs of order 5 (4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications*, 62.

van der Pol, B. (1926). On "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2.

Verner, J. H. (2010). Numerically optimal Runge–Kutta pairs with interpolants. *Numerical Algorithms*, 53.

Wenk, P., Abbati, G., Osborne, M. A., Schölkopf, B., Krause, A., and Bauer, S. (2020). ODIN: ODE-informed regression for parameter and state inference in time-continuous dynamical systems. In *Proceedings of the 34th Conference on Artificial Intelligence (AAAI)*, volume 34. AAAI Press.

Supplementary Material: Pick-and-Mix Information Operators for Probabilistic ODE Solvers

A PROBLEM DEFINITIONS

A.1 Pleiades

The Pleiades system describes the motion of seven stars in a plane, with coordinates (x_i, y_i) and masses $m_i = i$, $i = 1, \ldots, 7$ (Hairer et al., 1993, II.10). It is given by a second-order ODE

$$\ddot{x}_i = \sum_{j \neq i} m_j (x_j - x_i)/r_{ij}, \qquad \ddot{y}_i = \sum_{j \neq i} m_j (y_j - y_i)/r_{ij}, \tag{27}$$

where $r_{ij} = \big((x_i - x_j)^2 + (y_i - y_j)^2\big)^{3/2}$, for $i, j = 1, \ldots, 7$, on the time span $t \in [0, 3]$, with initial locations

$$x(0) = [3, 3, -1, -3, 2, -2, 2], \tag{28a}$$
$$y(0) = [3, -3, 2, 0, 0, -4, 4], \tag{28b}$$

and initial velocities

$$\dot{x}(0) = [0, 0, 0, 0, 0, 1.75, -1.5], \tag{28c}$$
$$\dot{y}(0) = [0, 0, 0, -1.25, 1, 0, 0]. \tag{28d}$$

A.2 Logistic Equation

The logistic equation is a simple IVP problem, given as

$$\dot{y}(t) = 3y(t)(1 - y(t)), \qquad t \in [0, 3], \qquad y(0) = 100, \tag{29}$$

for which the analytical solution is known to be

$$y(t) = \frac{\exp(3t)}{100 - 1 + \exp(3t)}. \tag{30}$$

A.3 Lotka–Volterra

The Lotka–Volterra model describes the dynamics of biological systems in which two species interact, one as a predator and the other as prey. The IVP is given by the ODE

$$\dot{x} = 1.5x - xy, \qquad \dot{y} = xy - 3y. \tag{31}$$

In our experiments, we consider initial values $x(0) = 1$, $y(0) = 1$ and a time span $t \in [0, 7]$.

A.4 Van–der–Pol

The Van der Pol model (van der Pol, 1926) describes a non-conservative oscillator with non-linear damping. In our experiment, we consider a notoriously stiff version of the model, given as

$$\dot{y_1}(t) = y_2(t), \qquad \dot{y_2}(t) = 10^6 \left(\left(1 - y_1^2(t)\right) y_2(t) - y_1(t)\right), \tag{32a}$$

on the time span $t \in [0, 10]$, with initial value $y(0) = [0, \sqrt{3}]^\intercal$.

A.5 Hénon–Heiles

The Hénon-Heiles model describes a star moving around a galactic center, with its motion restricted to a plane (Henon and Heiles, 1964). It is defined by a Hamiltonian

$$H(p,q) = \left[\frac{1}{2}\left(p_1^2 + p_2^2\right)\right] + \left[\frac{1}{2}\left(q_1^2 + q_2^2\right) + q_1^2 q_2 - \frac{1}{3}q_2^3\right], \tag{33}$$

which describes the kinetic and potential energy of the star with velocity p and location q. With $y(t) := q(t)$, we write the Hénon-Heiles problem as an IVP with second-order ODE, as

$$\ddot{y}_1(t) = -y_1(t) - 2y_1(t)y_2(t), \tag{34a}$$
$$\ddot{y}_2(t) = y_2^2(t) - y_2(t) - y_1^2(t), \tag{34b}$$

on the time span $t \in [0, 1000]$, with initial values $y(0) = (0, 0.1)$, $\dot{y}(0) = (0.5, 0)$. It further holds

$$g(\dot{y}(t), y(t)) := H(\dot{y}(t), y(t)) - H(\dot{y}_0(t), y_0(t)) = 0, \tag{35}$$

by conservation of the Hamiltonian (Hairer et al., 2006).

A.6 Kepler Problem

The Kepler problem is a special case of the two-body problem in celestial mechanics, and can be used to describe the movement of a planet around a star. It is given by a Hamiltonian $H : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$, with

$$H\left(p(t), q(t)\right) = \frac{\|p(t)\|^2}{2} - \frac{1}{\|q(t)\|}. \tag{36}$$

With $y(t) := q(t)$ and $\dot{y}(t) := p(t)$, it induces the second-order ODE

$$\ddot{y}(t) = -\frac{y(t)}{\|y(t)\|^3}. \tag{37}$$

In our experiments, the Kepler problem is solved on the time span $t \in \left[0, \frac{99}{100} \cdot 2\pi\right]$, with initial values $y(0) = [0.4, 0]$, $\dot{y}(0) = [0, 2]$. In addition to conserving the Hamiltonian, the Kepler system conserves angular momentum:

$$L\left(p(t), q(t)\right) = q_1(t)p_2(t) - q_2(t)p_1(t). \tag{38}$$

Thus, it holds

$$g(\dot{y}(t), y(t)) := \begin{bmatrix} H(\dot{y}(t), y(t)) - H(\dot{y}(0), y(0)) \\ L(\dot{y}(t), y(t)) - L(\dot{y}(0), y(0)) \end{bmatrix} = 0. \tag{39}$$

A.7 Robertson DAE

The Robertson DAE describes a system of chemical reactions and is a very popular problem to evaluate stiff ODE and DAE solvers (Hairer and Wanner, 1996). As a DAE, it is given by the equations

$$y_1(t) = -0.04y_1(t) + 10^4 y_2(t)y_3(t), \tag{40a}$$
$$y_2(t) = 0.04y_1(t) + 10^4 y_2(t)y_3(t) - (3 \cdot 10^7)y_2(t)^2, \tag{40b}$$
$$0 = y_1(t) + y_2(t) + y_3(t) - 1, \tag{40c}$$

and it therefore has a singular mass matrix of the form

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

We consider an initial value $y(0) = [1, 0, 0]$, and while the system is most often simulated on the time span $t \in [0, 10^5]$, we solve it on $t \in [0, 10^2]$ since we found the final error to be more informative in this setting since the values did then not saturate yet.

A.8 Pendulum DAE

A pendulum can be described in Cartesian coordinates with the following, index-reduced DAE (Hairer and Wanner, 1996)

$$\dot{x}(t) = v_x, \tag{41a}$$
$$\dot{v_x}(t) = xT, \tag{41b}$$
$$\dot{y}(t) = v_y, \tag{41c}$$
$$\dot{v_y}(t) = yT - g, \tag{41d}$$
$$0 = 2(v_x^2 + v_y^2 + y(yT - g) + Tx^2). \tag{41e}$$

In our experiments, we consider initial values $x(0) = 1$, $v_x(0) = 0$, $y(0) = 0$, $v_y(0) = 0$, $T(0) = 0$, and the gravitational acceleration $g = 9.81$. We simulate the system on the time span $t \in [0, 10]$.

Probabilistic ODE Solutions in Millions of Dimensions

Nicholas Krämer [*1] **Nathanael Bosch** [*1] **Jonathan Schmidt** [*1] **Philipp Hennig** [1 2]

Abstract

Probabilistic solvers for ordinary differential equations (ODEs) have emerged as an efficient framework for uncertainty quantification and inference on dynamical systems. In this work, we explain the mathematical assumptions and detailed implementation schemes behind solving high-dimensional ODEs with a probabilistic numerical algorithm. This has not been possible before due to matrix-matrix operations in each solver step, but is crucial for scientifically relevant problems—most importantly, the solution of discretised partial differential equations. In a nutshell, efficient high-dimensional probabilistic ODE solutions build either on independence assumptions or on Kronecker structure in the prior model. We evaluate the resulting efficiency on a range of problems, including the probabilistic numerical simulation of a differential equation with millions of dimensions.

1. Introduction

Problem Statement This paper discusses a class of algorithms that computes the solution of initial value problems based on ordinary differential equations (ODEs), i.e. finding a function y that satisfies

$$\dot{y}(t) = f(y(t), t), \tag{1}$$

for all $t \in [t_0, t_{\max}]$, as well as the initial condition $y(t_0) = y_0 \in \mathbb{R}^d$. Usually, f is non-linear, in which case the solution of Equation (1) cannot generally be derived in closed form and has to be approximated numerically. We continue the work of *probabilistic* numerical algorithms for ODEs (Schober et al., 2019; Tronarp et al., 2019; Kersting et al., 2020b; Tronarp et al., 2021; Bosch et al., 2021; Krämer & Hennig, 2020). Like other filtering-based ODE solvers ("ODE filters"), the algorithm used herein translates the numerical approximation of ODE solutions to a problem of probabilistic inference. The resulting (approximate) posterior distribution quantifies the uncertainty associated with the unavoidable discretisation error (Bosch et al., 2021) and provides a language that integrates well with other data inference schemes (Kersting et al., 2020a; Schmidt et al., 2021). The main difference to prior work is that we focus on the setting where the dimension d of the ODE is high, that is, say, $d \gg 100$. (It is not clearly defined at which point an ODE counts as high-dimensional, but $d \approx 100$ is already a scale of problems in which previous state-of-the-art probabilistic ODE solvers faced computational challenges.)

Motivation and Impact High-dimensional ODEs describe the interaction of large networks of dynamical systems and appear in many disciplines in the natural sciences. The perhaps most prominent example is the simulation of discretised partial differential equations. There, the dimension of the ODE equals the number of grid points used to discretise the problem (with e.g. finite differences; Schiesser, 2012). More recently, ODEs gained popularity in machine learning through the advent of neural ODEs (Chen et al., 2018) or physics-informed neural networks (Raissi et al., 2019). With the growing complexity of the model, each of the above can quickly become high-dimensional. If such use cases shall gain from probabilistic solvers, fast algorithms for large ODE systems are crucial.

Prior Work and State-of-the-Art Many non-probabilistic ODE solvers, for example, explicit Runge–Kutta methods, have a computational complexity linear in the ODE dimension d (Hairer et al., 1993). Explicit Runge–Kutta methods are often the default choices in ODE solver software packages. Compared to the efficiency of the methods provided by DifferentialEquations.jl (Rackauckas & Nie, 2017), SciPy (Virtanen et al., 2020), or Matlab (Shampine & Reichelt, 1997), probabilistic methods have lacked behind so far. Intuitively, ODE filters are a fusion of ODE solvers and Gaussian process models—two classes of algorithms that suffer from high dimensionality. More precisely, the problem is that probabilistic solvers require matrix-matrix operations at

[*]Equal contribution [1]University of Tübingen, Tübingen, Germany [2]Max Planck Institute for Intelligent Systems, Tübingen, Germany. Correspondence to: Nicholas Krämer <nicholas.kraemer@uni-tuebingen.de>, Nathanael Bosch <nathanael.bosch@uni-tuebingen.de>, Jonathan Schmidt <jonathan.schmidt@uni-tuebingen.de>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

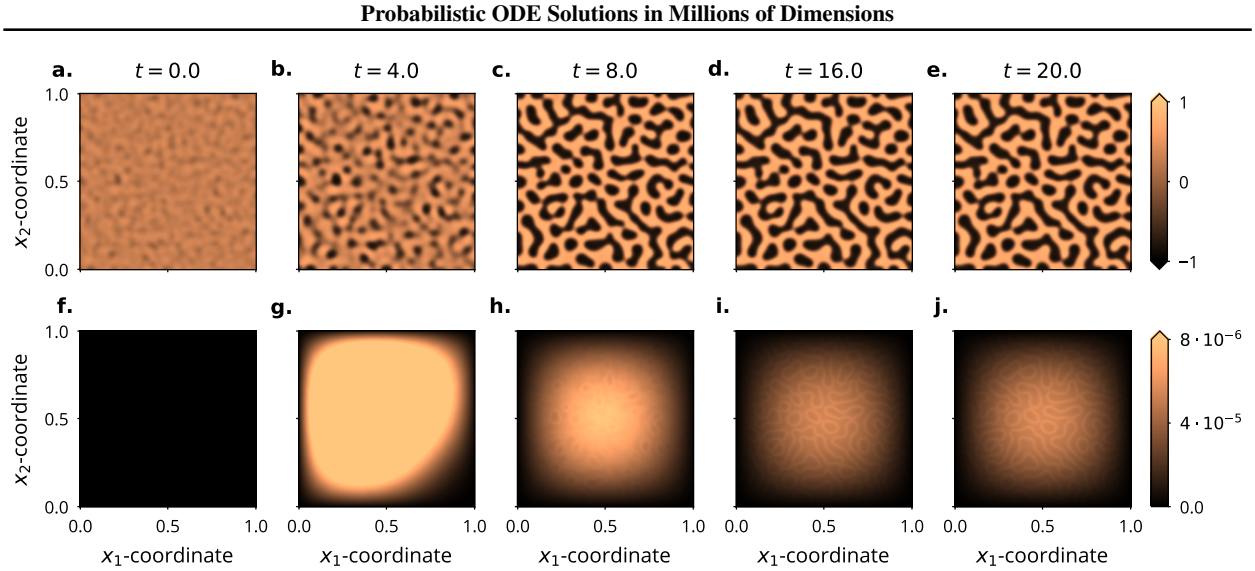


Figure 1. Simulating a high-dimensional ODE: Probabilistic solution of a discretised FitzHugh–Nagumo PDE model (Ambrosio & Françoise, 2009). Means (a-e) and standard-deviations (f-j), $t_0 = 0$ (left) to $t_{\max} = 20$ (right). The patterns in the uncertainties match those in the solution. The simulated ODE is 125k-dimensional. A detailed description of the experiment is provided in Appendix E.1.

each step. The matrices have $O(d^2)$ entries, which leads to $O(d^3)$ complexity for a single solver step and has made the solution of high-dimensional ODEs impossible. ODE filters are essentially nonlinear, approximate Gaussian process inference schemes (with a lot of structure). As in the GP community (e.g. Quiñonero-Candela & Rasmussen, 2005), the path to low computational cost in these models is via factorisation assumptions. But, perhaps surprisingly, the necessary assumptions are minimal, in the sense that they are already fulfilled by existing ODE filtering literature.

Contributions Our main contribution is to prove in which settings ODE filters admit an implementation in $O(d)$ complexity. Thereby, they become a class of algorithms comparable to explicit Runge–Kutta methods not only in estimation performance (error contraction as a function of evaluations of f; Kersting et al., 2020b; Tronarp et al., 2021) but also in computational complexity (cost per evaluation of f). The resulting algorithms deliver uncertainty quantification and other benefits of probabilistic ODE solvers on high-dimensional ODEs (see Figure 1; the ODE from this figure will be explored in more detail in Section 5). The key novelties of the present work are threefold:

1. *Acceleration via independence:* A-priori, ODE filters commonly assume independent ODE dimensions (e.g. Kersting et al., 2020b). We single out those inference schemes that naturally preserve independence. Identification of independence-preserving ODE solvers is helpful because each ODE dimension can be updated separately. The performance implications are that a single matrix-matrix operation with $O(d^2)$ entries is replaced with d matrix-matrix operations with $O(1)$ entries. In other words, $O(d)$ instead of $O(d^3)$ complexity for a single solver step (Proposition 3.3).

2. *Calibration of multivariate output-scales:* A single ODE system often models the interaction between states that occur on different scales. It is useful to acknowledge differing output scales in the "diffusivity" of the prior (details below). We generalise the calibration result by Bosch et al. (2021) to the class of solvers that preserve the independence of the dimensions (Proposition 3.2).

3. *Acceleration via Kronecker structure:* Sometimes, prior independence assumptions may be too restrictive. For instance, one might have prior knowledge of correlations between ODE dimensions (Example 4.1 in Section 4). Fortunately, a subset of probabilistic ODE solvers can exploit *and preserve* Kronecker structure in the system matrices of the state space. Preserving the Kronecker structure brings over the performance gains from above to dependent priors (Proposition 4.3).

To demonstrate how the independence and the Kronecker structure imply extreme scalability of the resulting algorithm, the experiments in Section 5 showcase simulations of ODEs with dimension $d \sim 10^7$.

2. ODE Filter Setup

The following section details the technical setup of an ODE filter, including the prior (Section 2.1), information model (Section 2.2), and practical considerations (Section 2.3).

2.1. Prior Model

The following is standard for probabilistic ODE solvers, and therefore essentially identical to the presentation by e.g. Schober et al. (2019). Herein, however, we place a stronger emphasis on Kronecker- and independence-structures in the system matrices compared to prior work. Both are important for the theoretical statements below. Särkkä (2013) or Särkkä and Solin (2019) provide a comprehensive explanation of the mathematical concepts regarding inference in state-space models.

Stochastic Process Prior on the ODE Solution Let $Y := (Y^i)_{i=1}^d = \left(Y_0^i, \dots, Y_\nu^i\right)_{i=1}^d$ solve the linear, time-invariant stochastic differential equation (SDE)

$$\mathrm{d}Y(t) = AY(t)\,\mathrm{d}t + B\,\mathrm{d}W(t), \tag{2}$$

subject to a Gaussian initial condition

$$Y(t_0) \sim \mathcal{N}(m_0, C_0) \tag{3}$$

for some m_0 and $C_0 := \Gamma \otimes \breve{C}_0$; e.g., $m_0 = 0$, $\breve{C}_0 = I$. The SDE is driven by a d-dimensional Wiener process W with diffusion $\Gamma \in \mathbb{R}^{d\times d}$, and governed by the system matrices

$$A := I_d \otimes \breve{A}, \quad \breve{A} := \sum_{q=0}^{\nu-1} e_q e_{q+1}^\top, \quad B := I_d \otimes e_\nu, \tag{4}$$

where $e_q \in \mathbb{R}^{\nu+1}$ is the q-th basis vector. The zeroth component of Y, $(Y_0^i)_{i=1}^d$, is an integrated Wiener process. With such A and B, the q-th component $(Y_q^i)_{i=1}^d$ models the q-th derivative of the integrated Wiener process. Similar SDEs can be written down for e.g. the integrated Ornstein-Uhlenbeck process or the Matérn process (the only differences would be additional non-zero entries in $\breve{A}$). If Γ were diagonal, the Kronecker structure in A and B would imply prior pairwise independence between Y^i and Y^j, $i \neq j$. Section 3 uses the diagonality assumption to reveal the efficient implementation of a class of ODE filters. Section 4 allows Γ to be any symmetric, positive definite matrix, which is why we do not make strong assumptions on Γ yet.

Discretisation Let $\mathbb{T} = (t_0, ..., t_N)$ be some time-grid with step-size $h_n := t_{n+1} - t_n$. While for the presentation, we assume a fixed grid, practical implementations choose t_n adaptively. Reduced to $\mathbb{T}$, due to the Markov property, the process Y becomes

$$Y(t_{n+1}) \mid Y(t_n) \sim \mathcal{N}(\Phi(h_n)Y_n, \Sigma(h_n)) \tag{5}$$

for matrices

$$\Phi(h_n) := \exp(Ah_n), \tag{6a}$$

$$\Sigma(h_n) := \int_0^{h_n} \Phi(h_n - \tau) B\Gamma B^\top \Phi(h_n - \tau)^\top \,\mathrm{d}\tau. \tag{6b}$$

The definition of $\Phi(h_n)$ uses the matrix exponential. $\Phi(h_n)$ inherits the block diagonal structure from A,

$$\Phi = I_d \otimes \breve{\Phi}(h_n), \quad \breve{\Phi}(h_n) := \exp(\breve{A}h_n), \tag{7}$$

and Σ has a Kronecker factorisation similar to C_0,

$$\Sigma(h_n) = \Gamma \otimes \breve{\Sigma}(h_n), \tag{8a}$$

$$\breve{\Sigma}(h_n) := \int_0^{h_n} \breve{\Phi}(h_n - \tau) e_\nu e_\nu^\top \breve{\Phi}(h_n - \tau)^\top \,\mathrm{d}\tau. \tag{8b}$$

The discretisation allows efficient extrapolation from t_n to t_{n+1}. Let $Y(t_n) \sim \mathcal{N}(m_n, C_n)$. Then,

$$Y(t_{n+1}) \sim \mathcal{N}(m_{n+1}^-, C_{n+1}^-) \tag{9}$$

with mean and covariance

$$m_{n+1}^- = \Phi(h_n) m_n, \tag{10a}$$

$$C_{n+1}^- = \Phi(h_n) C_n \Phi(h_n)^\top + \Sigma(h_n). \tag{10b}$$

For improved numerical stability, probabilistic ODE solvers compute this prediction in square root form, which means that only square root matrices of C_n and C_{n+1}^- are propagated without ever forming full covariance matrices (Krämer & Hennig, 2020; Grewal & Andrews, 2014). Appendix A recalls square root implementations of ODE filters.

2.2. Information Model

Information Operator The information operator

$$\mathcal{I}(Y)(t) := Y_1(t) - f(Y_0(t), t) \tag{11}$$

(recall $Y_q \approx y^{(q)}$), known as the local defect (Gustafsson, 1992), captures "how well (a sample from) Y_0 solves the ODE". If this value is large, the current state is an inaccurate approximation, and if it is small, Y_0 provides a good estimate of the truth. Loosely speaking, the goal is to make the defect as small as possible over the entire time domain.

Artificial Data The local defect $\mathcal{I}$ can be kept small by conditioning Y on $\mathcal{I}(Y)(t) \overset{!}{=} 0$ on "many" grid-points. Due to the regular prior and the regularity-preserving information operator $\mathcal{I}$, conditioning the prior on a zero-defect leads to an accurate ODE solution (Tronarp et al., 2021). Altogether, the probabilistic ODE solver targets

$$p\left(Y \,\middle|\, \{\mathcal{I}(Y)(t_n) = 0\}_{n=0}^N, Y_0(t_0) = y_0\right). \tag{12}$$

(Recall from Equation (2) that lower indices in Y refer to the derivative, i.e. Y_0 is the integrated Wiener process, and Y_q its q-th derivative.) We call the posterior in Equation (12) the *probabilistic ODE solution*. Unfortunately, a nonlinear vector field f implies a nonlinear information operator $\mathcal{I}$. Thus, the exact posterior is intractable.

Linearisation A tractable approximation of the probabilistic ODE solution is available through linearisation. Linearising f indirectly linearises $\mathcal{I}$, and the corresponding probabilistic ODE solution arises via Gaussian inference. Let F_y be (an approximation of) the Jacobian of f with respect to y. One can approximate the ODE vector field with a Taylor series

$$f(y) \approx \hat{f}_\xi(y) := f(\xi) + F_y(\xi)(y - \xi) \tag{13}$$

at some $\xi \in \mathbb{R}^d$. Let $E_q := I_d \otimes e_q$ be the projection matrix that extracts the q-th derivative from the full state Y. In other words, $\dot{Y}_0 = E_1 Y$. Equation (13) implies a linearisation of $\mathcal{I}$ at some $\eta \in \mathbb{R}^{d(\nu+1)}$,

$$\mathcal{I}(Y)(t) \approx \hat{\mathcal{I}}_\eta(Y)(t) := H(t)Y(t) + b(t), \tag{14}$$

with linearisation matrices

$$H(t) := E_1 - F_y(E_0\eta, t)E_0, \tag{15a}$$
$$b(t) := F_y(E_0\eta, t)E_0\eta - f(E_0\eta, t). \tag{15b}$$

$\hat{\mathcal{I}}_\eta$ is linear in $Y(t)$. Therefore, the approximate probabilistic ODE solution becomes tractable with Gaussian filtering and smoothing once $\hat{\mathcal{I}}_\eta$ is plugged into Equation (12) (Särkkä, 2013; Tronarp et al., 2019). At time t_n, the linearisation point η is usually chosen as the predicted mean m_n^-, which yields the extended Kalman filter (Särkkä, 2013). For ODE filters, there are three relevant versions of F_y: EK0, EK1, and the diagonal EK1.

EK0 Approximate the Jacobian as $F_y \equiv 0$, which has been a common choice since early work on ODE filters (Schober et al., 2019; Kersting et al., 2020b), and implies a zeroth-order approximation of f (Tronarp et al., 2019).

EK1 Use the full Jacobian $F_y = \nabla_y f$, which amounts to a first-order Taylor approximation of the ODE vector field (Tronarp et al., 2019). It is more stable than the EK0 (Tronarp et al., 2019), but in its general form, the EK1 does not fit the assumptions made below and thus does not immediately scale to high dimensions. Instead, we introduce the diagonal EK1.

Diagonal EK1 Use only the diagonal of the full Jacobian, $F_y = \mathrm{diag}(\nabla_y f)$. This choice conserves the efficiency of the EK0 to a solver that uses Jacobian information. The diagonal EK1 is another minor contribution of the present work. Section 5 empirically investigates how much stability using only the diagonal of the Jacobian provides.

Measurement and Correction A probabilistic ODE solver step consists of an extrapolation, measurement, and correction phase. Extrapolation has been explained in Equations (9) and (10) above. Denote

$$Y_{n+1}^- := Y(t_{n+1}) \sim \mathcal{N}(m_{n+1}^-, C_{n+1}^-). \tag{16}$$

The measurement phase approximates

$$Z_{n+1} := \hat{\mathcal{I}}_{m_{n+1}^-}(Y_{n+1}^-)(t_{n+1}) \approx \mathcal{I}(Y_{n+1}^-)(t_{n+1}) \tag{17}$$

by exploiting the linearisation matrices H and b,

$$Z_{n+1} \sim \mathcal{N}(z_{n+1}, S_{n+1}), \tag{18a}$$
$$z_{n+1} := H(t_{n+1})m_{n+1}^- + b(t_{n+1}), \tag{18b}$$
$$S_{n+1} := H(t_{n+1})C_{n+1}^- H(t_{n+1})^\top. \tag{18c}$$

Z_{n+1} will be used for calibration (details below). The extrapolated random variable is then corrected as

$$Y_{n+1} \sim \mathcal{N}(m_{n+1}, C_{n+1}), \tag{19a}$$
$$m_{n+1} := m_{n+1}^- - C_{n+1}^- H(t_{n+1})^\top S_{n+1}^{-1} z_{n+1}, \tag{19b}$$
$$C_{n+1} := \Xi C_{n+1}^- \Xi^\top, \tag{19c}$$
$$\Xi := I - C_{n+1}^- H(t_{n+1})^\top S_{n+1}^{-1} H(t_{n+1}). \tag{19d}$$

The update in Equations (19c) and (19d) is the Joseph update (Bar-Shalom et al., 2004). In practice, we never form the full C_{n+1} but compute only the square root matrix by applying Ξ to the square root matrix of C_{n+1}^-. It is not a Cholesky factor (because it is not lower triangular), but generic square root matrices suffice for numerically stable implementation of probabilistic ODE solvers (Krämer & Hennig, 2020).

2.3. Practical Considerations

Let us conclude with brief pointers to further practical considerations that are important for probabilistic ODE solvers.

Initialisation The state space models a stack of a y and the first ν derivatives. The stability of the ODE filter depends on the accurate initialisation of all derivatives (Krämer & Hennig, 2020). We initialise the solver by inferring

$$p(Y(t_0) \mid Y_0(\tau_m) = \hat{y}(\tau_m), m = 0, ..., \nu) \tag{20}$$

on $\nu+1$ small steps $\tau_0, ..., \tau_\nu$ where the $\hat{y}(\tau_m)$ are computed with e.g. a Runge–Kutta method. This is a slight generalisation of the strategy used by Schober et al. (2019) (also refer to Schober et al. (2014); Gear (1980)), in the sense that we formulate this initialisation as probabilistic inference instead of setting the first few means manually. An alternative strategy would be (Taylor-mode) automatic differentiation (Krämer & Hennig, 2020; Griewank & Walther, 2008).

Error Estimation Comprehensive explanation of error estimation and step-size adaptation, and its effect on the estimation quality and calibration of probabilistic ODE solvers, is out of scope for the present work; we refer the reader to Schober et al. (2019) and Bosch et al. (2021).

3. Independence Accelerates ODE Filters

This section establishes the main idea of the present work: *filtering-based probabilistic ODE solvers are fast and efficient when the prior models each dimension independently.*

3.1. Assumptions

Independent dimensions stem from a diagonal Γ.

Assumption 3.1. Assume that the diffusion Γ of the Wiener process in Equation (2) is a diagonal matrix.

Assumption 3.1 implies that the initial covariance C_0 (Equation (3)) is the Kronecker product of a diagonal matrix with another matrix, thus block diagonal. Assumption 3.1 is not very restrictive; in prior work on ODE filters, Γ was always either $\Gamma = \gamma^2 I$ for some $\gamma > 0$ (Schober et al., 2019; Tronarp et al., 2019; Kersting et al., 2020b; Bosch et al., 2021; Tronarp et al., 2021; Krämer & Hennig, 2020), or diagonal (Bosch et al., 2021).

3.2. Calibration

Tuning the diffusion Γ is crucial to obtain accurate posterior uncertainties. As announced in Section 2, the mathematical assumptions for calibrating Γ coincide with those that lead to an efficient ODE filter. Thus, we discuss Γ before proving the linear complexity of ODE filters under Assumption 3.1.

Four Approaches Recall the observed random variable Z_n (Equation (17)). ODE filters calibrate Γ with quasi-maximum-likelihood-estimation (quasi-MLE): Consider the prediction error decomposition (Schweppe, 1965),

$$p(\{Z_n\}_{n=0}^N) = p(Z_0) \prod_{n=0}^{N-1} p(Z_{n+1} \mid Z_n) \tag{21a}$$

$$\approx \mathcal{N}(z_0, S_0) \prod_{n=0}^{N-1} \mathcal{N}(z_{n+1}, S_{n+1}). \tag{21b}$$

Γ is a quasi-MLE if it maximises Equation (21b). The specific choice of calibration depends on the respective model for Γ, and reduces to one of four approaches: on the one hand, fixing and calibrating a *time-constant* Γ versus allowing a *time-varying* Γ; on the other hand, choosing a *scalar* diffusion $\Gamma = \gamma^2 I$ versus choosing a *vector-valued* diffusion $\Gamma = \mathrm{diag}(\gamma^1, ..., \gamma^d)$. Roughly speaking, a time-varying, vector-valued diffusion allows for the greatest flexibility in the probabilistic model. One contribution of the present work is to extend the vector-valued diffusion results by Bosch et al. (2021) to a slightly broader class of solvers (Proposition 3.2 below). Scalar diffusion will reappear in Section 4. Appendix B addresses time-constant diffusion.

Time-Varying Diffusion Allowing Γ to change over time, all measurements before time t_n are independent of Γ_n. Under the assumption of an error-free previous state (which is common for hyperparameter calibration in ODE solvers), a local quasi-MLE for $\Gamma_n = \gamma_n^2 I$ is (Schober et al., 2019)

$$\hat{\gamma}_n^2 := \frac{1}{d} z_n \left[H(t_n) \Sigma(h_n) H(t_n)^\top \right]^{-1} z_n. \tag{22}$$

This can be extended to a quasi-MLE for the EK0 with vector-valued $\Gamma_n = \mathrm{diag}(\gamma_n^1, ..., \gamma_n^d)$ (Bosch et al., 2021)

$$(\hat{\gamma}_n^i)^2 := (z_n^i)^2 / [H(t_n) \Sigma(h_n) H(t_n)^\top]_{ii}, \tag{23}$$

for all $i = 1, ..., d$. In this work, we generalise the EK0 requirement to Assumption 3.1 and a diagonal Jacobian.

Proposition 3.2. *Under Assumption 3.1 and for diagonal F_y, the estimators $(\hat{\gamma}_n^i)_i$ in Equation (23) are quasi-MLEs.*

Sketch of the proof. Two ideas are relevant: (i) a diagonal Jacobian implies a block diagonal $H(t_n)$ and a diagonal $H(t_n)\Sigma(h_n)H(t_n)^\top$ (which will be proved formally in Proposition 3.3 below); (ii) the local evidence, i.e. the probability of $\mathcal{N}(z_n, S_n)$ being zero, decomposes into a sum over the coordinates. Maximising each summand with respect to γ_n^i yields the claim. □

A very similar case can be made for time-constant diffusion (see Appendix B). Proposition 3.2 is a generalisation of the results by Bosch et al. (2021) in the sense that Proposition 3.2 is not restricted to the EK0.

3.3. Complexity

Now, with calibration in place, we can discuss the computational complexity of ODE filters under Assumption 3.1. The following proposition establishes that for diagonal Jacobians, a single solver step costs $O(d)$.

Proposition 3.3. *Suppose that Assumption 3.1 is in place. If the Jacobian of the ODE is (approximated as) a diagonal matrix, then a single step with a filtering-based probabilistic ODE solver costs $O(d\nu^3)$ in floating-point operations, and $O(d\nu^2)$ in memory.*

Proof. Let $Y_n \sim \mathcal{N}(m_n, C_n)$. Assume that C_n is block diagonal. We show that block diagonality is preserved through a step, and since by Assumption 3.1, C_0 is block diagonal, we do not lose generality. Recall $\Phi(h_n)$ and $\Sigma(h_n)$ from Equations (5) and (6). $\Phi(h_n)$ is block diagonal, and since Γ_n is diagonal, $\Sigma(h_n)$ is block diagonal.

(i) Extrapolate mean: The mean is extrapolated according to Equation (10a), which costs $O(d\nu^2)$ due to the block diagonal $\Phi(h_n)$. Each dimension is extrapolated independently.

(ii) Evaluate ODE: Next, $H = H(t_{n+1})$ and $b = b(t_{n+1})$ from Equation (15) are assembled, which involves evaluating f and F_y at $\xi := E_0 m^-_{n+1}$ (E_0 is a projection matrix and can be implemented as array indexing, so ξ comes at negligible cost). $F_y = \mathrm{diag}(F_y^1, ...F_y^d)$ is diagonal, thus

$$H = \mathrm{blockdiag}(H^1, ..., H^d) \tag{24}$$

is block diagonal with blocks

$$H^i := e_1 - e_0 F_y^i, \quad i = 1, ..., d \tag{25}$$

(recall the basis vectors e_q from Equation (4)). The block diagonal H has been pre-empted in Proposition 3.2 above.

(iii) Calibrate Γ*:* The cost of assembling the quasi-MLE for Γ_{n+1} according to Equation (22) or Equation (23) is $O(d)$, because the matrix to be inverted is diagonal.

(iv) Extrapolate the covariance: The covariance can be extrapolated dimension-by-dimension as well, because C_n, $\Phi(h_n)$, and $\Sigma(h_n)$ are all block diagonal with the same block structure: d square blocks with $\nu + 1$ rows and columns; recall Equation (10b). In reality, the matrix-matrix multiplication is replaced by a QR decomposition; we refer to Appendix A.1 for details on square root implementation. Using either strategy—square root or traditional implementation—extrapolating the covariance costs $O(d\nu^3)$ and C^-_{n+1} is block diagonal.

(v) Measure: Computing the mean of Z_{n+1} (recall Equations (17) and (18)) costs $O(d)$. The covariance S_{n+1} of Z_{n+1} is diagonal, since H and C^-_{n+1} are block diagonal. Thus, assembling *and inverting* S_{n+1} costs $O(d)$.

(vi) Correct mean and covariance: The mean is corrected according to Equation (19b), which—since S_{n+1} is diagonal—costs $O(d\nu)$. The covariance is corrected according to Equations (19c) and (19d), the complexity of which hinges on the structure of Ξ (Equation (19d)): due to the block diagonal C^-_{n+1}, H, and S_{n+1}, Ξ is block diagonal again, and correcting the covariance costs $O(d\nu^3)$. The square root of C_{n+1} arises by multiplying Ξ with the "left" square root matrix of C^-_{n+1}. The complexity does not change (asymptotically; QR decompositions cost more than matrix multiplications).

All in all, ODE filter steps preserve block-diagonal structure in the covariances. The expensive phases are the covariance extrapolation and correction in $O(d\nu^3)$ floating-point operations. The maximum memory demand is $O(d\nu^2)$ for the block diagonal covariances. □

While it may seem restrictive to use only the diagonal of the Jacobian, Proposition 3.3 includes the EK0, one of the central ODE filters. The $O(d)$ complexity puts the EK0 and the diagonal EK1 into the complexity class of explicit Runge–Kutta methods. Usually, $\nu < 12$ holds, so ν can be treated as some constant (Krämer & Hennig, 2020).

4. EK0 Preserves Kronecker Structure

As hinted in Section 1, scalar or diagonal diffusion may be too restrictive in certain situations (Krämer et al., 2022).

Example 4.1. Consider a spatio-temporal Gaussian process model $u(t, x) \sim \mathrm{GP}(0, \gamma^2 k_t \otimes k_x)$, where k_t is the covariance kernel that directly corresponds to an integrated Wiener process prior (Särkkä & Solin, 2019). Such a spatio-temporal model could be a useful prior distribution for applying an ODE solver to problems that are discretised PDEs, because k_x encodes spatial dependency structures. Restricted to a spatial grid $\mathcal{X} := \{x_1, ..., x_G\}$, $y := u(t, \mathcal{X})$ satisfies the prior model in Equations (2) to (3)[1], but with $\Gamma = \gamma^2 k_x(\mathcal{X}, \mathcal{X})$ (Solin, 2016), which is usually dense.

4.1. Assumptions

Despite the lack of independence in Example 4.1, fast ODE solutions remain possible with the EK0. In the remainder of this section, let $\Gamma = \gamma^2 \breve{\Gamma}$ for some matrix $\breve{\Gamma}$ and some scalar γ. Calibrating the scalar γ allows preserving Kronecker structure in the system matrices that appear in an ODE filter step (Proposition 4.3 below). Tronarp et al. (2019) show how for $\Gamma = \gamma^2 \breve{\Gamma}$, a time-constant quasi-MLE $\hat{\gamma}$ arises in closed form and also, that the posterior covariances all look like $C_n = \gamma^2 \breve{C}_n$: calibration can happen entirely post-hoc.

Constraints The following statement about linear complexity of ODE filters is only valid under two constraints: one can ignore (i) the quadratic costs of multiplying the posterior covariances with the quasi-MLE, and (ii) the cubic costs of solving a linear system involving Γ. The system matrices Φ, Σ, C_0 are all Kronecker products of a $\mathbb{R}^{d\times d}$ ("left") and a $\mathbb{R}^{\nu\times\nu}$ factor ("right"). The first constraint is thus avoided by scaling the "right" Kronecker factor of the covariances with γ^2 in $O(\nu^2)$. (ODE filters preserve Kronecker structure; see below.) The second one becomes the following assumption.

Assumption 4.2. Assume that the inverse of Γ is readily available and cheap to apply; that is, the quantity $x^\top \Gamma^{-1} x$ can be computed in $O(d)$.

Naturally, Assumption 4.2 holds for diagonal or at least sufficiently sparse matrices Γ. There are also settings in which Assumption 4.2 holds even if Γ is dense. For instance, if Γ is the covariance of a Gauss–Markov random field, the sparsity structure in Γ^{-1} implies adjacency of grid nodes (Lindgren et al., 2011; Sidén & Lindsten, 2020). In Example 4.1 with

[1] Technically, the stack of y and its ν derivatives does.

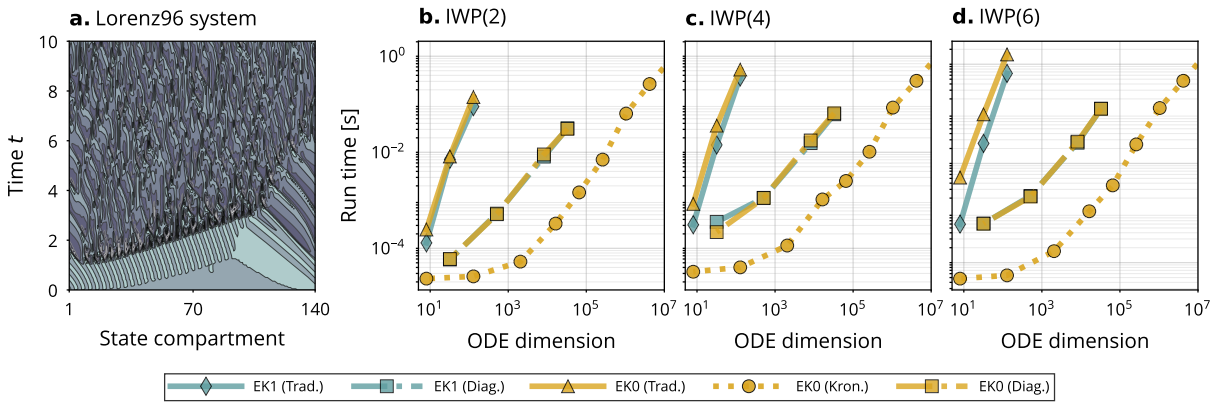


Figure 2. Runtime of a single ODE filter step: Run time (wall-clock) of a single step of ODE filter variations on the Lorenz96 problem (a) for increasing ODE dimension and $\nu = 2, 4, 6$ (b-d). The traditional implementations cost $O(d^3)$ per step, the diagonal EK1 and diagonal EK0 are $O(d)$ per step, just like the Kronecker version of the EK0. The Kronecker EK0 is faster than the diagonal solvers, because covariance operations involve a single $(\nu + 1) \times (\nu + 1)$ matrix instead of a batch of d such matrices (further details: Appendix C).

a spatial Matern kernel, for example, inverse Gram matrices can be approximated efficiently using the stochastic partial differential equation formulation (Lindgren et al., 2011).

4.2. Computational Complexity

Under Assumption 4.2, a single EK0 step costs $O(d)$:

Proposition 4.3. *Under Assumption 4.2, and if a time-constant diffusion model $\Gamma = \gamma^2 \breve{\Gamma}$ is calibrated via γ, a single step of the EK0 costs $O(\nu^3 + d\nu^2)$ floating point operations, and $O(d\nu + d^2 + \nu^2)$ memory.*

The proof parallels that of Proposition 3.3 (details are in Appendix C). It hinges on computing everything only in the "right" factor of each Kronecker matrix. The proposition can be extended to time-varying diffusion if one tracks γ in the "right" Kronecker factor instead of the "left" one. Since this obfuscates the notation, we prove the claim in Appendix D. The $O(d^2)$ memory requirement is entirely due to the cost of storing Γ. If Γ or Γ^{-1} are sparse, it reduces to $O(d\nu + \nu^2)$.

5. Empirical Evaluation

The remainder evaluates the efficiency of the proposals. Next to everything detailed above, our implementation uses the preconditioner suggested by Krämer & Hennig (2020). Its complexity is neglibible in light of Propositions 3.3 and 4.3, because all preconditioners are diagonal matrices. The full code for the solver implementation and experiments is publicly available on GitHub.[2]

[2] https://github.com/pnkraemer/million-dimension-prob-ode-solver-experiments

Single ODE Filter Step We begin by evaluating the cost of a single step of the ODE filter variations on the Lorenz96 problem (Lorenz, 1996). This is a chaotic dynamical system and recommends itself for the first experiment, as its dimension can be increased freely. Appendix E.2 contains a more detailed description of the ODE model. We time a single ODE filter step for increasing ODE dimension d and different solver orders $\nu \in \{2, 4, 6\}$. The results are depicted in Figure 2. The traditional EK0 and EK1 become infeasible due to their cubic complexity in the dimension. The diagonal EK1 and the diagonal EK0 exhibit their $O(d)$ cost. The Kronecker EK0 is cheaper than the independence-based solvers. A step with the Kronecker EK0 takes $\sim$1 second for a 16 million-dimensional ODE on a consumer-level CPU. Figure 2 confirms Propositions 3.3 and 4.3.

Full Simulation Next, we evaluate whether the performance gains for a single ODE filter step translate into a reduced overall runtime (including initialisation, step-size adaptation and calibration) on a medium-dimensional problem: the Pleiades problem (Hairer et al., 1993). It describes the motion of seven stars in a plane and is commonly solved as a system of 28 first-order ODEs. Appendix E.3 describes the ODE dynamics and the experimental setup in more detail. The results are in Figure 3. Pleiades reveals the increased efficiency of the ODE filters. The probabilistic solvers are as fast as Radau, only by a factor $\sim$10 slower than SciPy's RK45 (Virtanen et al., 2020), but 100 times faster than their reference implementations. (It should be noted that the ODE filters use just-in-time compilation for some components, whereas SciPy does not.) These findings confirm how the practical considerations (initialisation, step-size adaptation, etc.) scale sufficiently well to higher-dimensional settings.

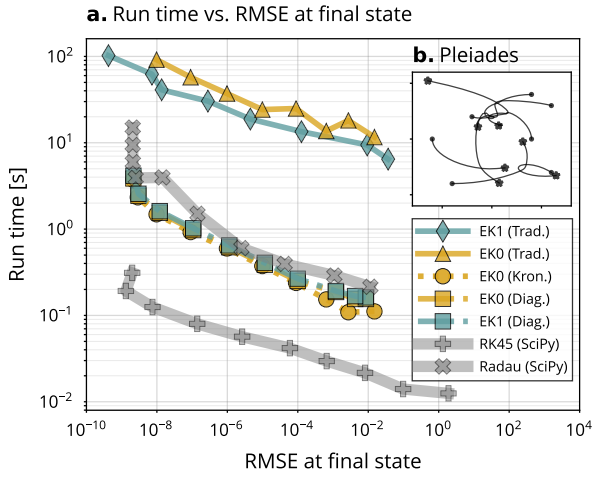


Figure 3. Runtime efficiency of fast ODE filters: Run time per root mean-square error of the ODE filters (a) on the Pleiades problem (b). The figure also shows two reference ODE filters, EK0 and EK1 in the traditional implementation, and Scipy's RK45 (explicit) and Radau (implicit). On the 28-dimensional Pleiades problem, the improved implementation accelerates the ODE filter implementations significantly.

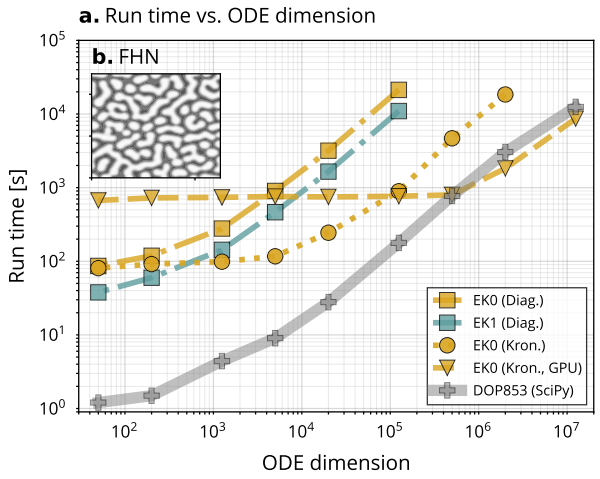


Figure 4. High-dimensional PDE discretisation: Run-time of ODE filters on the discretised FitzHugh–Nagumo model for increasing ODE dimension (i.e. increasing spatial resolution) including calibration and adaptive time-steps. SciPy's DOP853 for reference. Simulating $\gg 10^6$-dimensional ODEs takes $\approx 3h$.

High-Dimensional Setting To evaluate how well the improved efficiency translates to extremely high dimensions, we solve the discretised FitzHugh–Nagumo PDE model on high spatial resolution (which translates to high-dimensional ODEs); details on the experiment setup can be found in Appendix E.4. The results are in Figure 4. The main takeaway

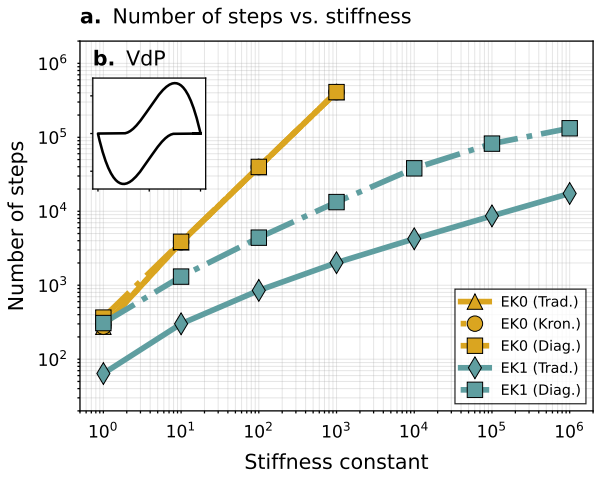


Figure 5. Stability: Number of steps taken by an ODE filter (a) for an increasingly stiff Van der Pol system (b). The diagonal EK1 is more stable than the EK0, but less stable than the EK1 (which is expected because it uses strictly less Jacobian information).

is that ODEs with millions of dimensions can now be solved *probabilistically* within a realistic time frame (hours), which has not been possible before. GPUs improve the runtime for extremely high-dimensional problems ($d \gg 10^5$).

Stability of the Diagonal EK1 How much do we lose by ignoring off-diagonal elements in the Jacobian? To evaluate the loss (or preservation) of stability against the A-stable EK1 (Tronarp et al., 2019), we solve the Van der Pol system (Guckenheimer, 1980). (The reference solver requires a low-dimensional ODE.) It includes a free parameter $\mu > 0$, whose magnitude governs the stiffness of the problem: the larger μ, the stiffer the problem, and for e.g. $\mu = 10^6$, Van der Pol is a famously stiff equation. For further details see Appendix E.5. The results are in Figure 5. We observe how the diagonal EK1 is less stable than the traditional EK1 for an increasing stiffness constant, but also that it is significantly more stable than the EK0. It is a success that the diagonal EK1 solves the Van der Pol equation for large μ.

Uncertainty Calibration of the Diagonal EK1 It has previously been shown by Bosch et al. (2021) that the EK0, diagonal EK0 and EK1 are similarly-well calibrated, with the EK1 having a tendency to be underconfident. Here, we investigate the calibration of the diagonal EK1. We evaluate the chi-squared statistic

$$\chi^2(t) := \|m(t) - y(t)\|^2_{C(t)^{-1}}, \tag{26}$$

which measures the ratio of the error in the mean $m \in \mathbb{R}^d$ of Y_0, which approximates the ODE solution y, normalised

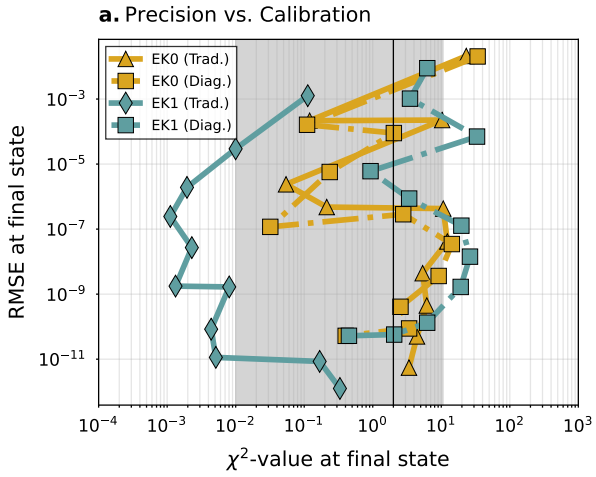


Figure 6. Uncertainty calibration: Chi-squared statistics and root-mean-square errors (RMSEs) over a range of tolerance levels. The gray area shows the 99% confidence intervals for the chi-squared statistic and the black horizontal line denotes perfect calibration ($\chi^2 = d$). While the traditional EK1 shows underconfidence ($\chi^2 \ll d$), the diagonal EK1 is more similar in calibration to the EK0 variants.

by the covariance $C \in \mathbb{R}^{d\times d}$ of Y_0, for a range of error-tolerance levels, on a non-stiff Van der Pol system. The results are in Figure 6. While the traditional EK1 tends to be underconfident, the calibration of the diagonal EK1 is more comparable to that of the EK0 variants. In particular, the diagonal approximation of the Jacobian does not seem to have a negative impact on the uncertainty calibration in terms of what is measured by the chi-squared statistic.

6. Conclusion

For probabilistic ODE solvers to capitalize on their theoretical advantages, their computational cost has to come close to that of their non-probabilistic, point-estimate counterparts (which benefit from decades of optimization). High-dimensional problems are one obstacle on this path, which we cleared here. We showed that independence assumptions in the underlying state-space model, or preservation of Kronecker structures, can bring the computational complexity of a large subset of known ODE filters close to non-probabilistic, explicit Runge–Kutta methods. As a result, probabilistic simulation of extremely large systems of ODEs is now possible, opening up opportunities to exploit the advantages of probabilistic ODE solvers on challenging real-world problems.

Acknowledgements

The authors gratefully acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) through Project ADIMEM (FKZ 01IS18052B), and financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence “Machine Learning - New Perspectives for Science”, EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Nicholas Krämer, Nathanael Bosch, and Jonathan Schmidt. The authors thank Katharina Ott for helpful feedback on the manuscript.

References

Ambrosio, B. and Françoise, J.-P. Propagation of bursting oscillations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1908):4863–4875, 2009.

Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. *Estimation With Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley & Sons, 2004.

Bosch, N., Hennig, P., and Tronarp, F. Calibrated adaptive probabilistic ODE solvers. In *AISTATS 2021*, 2021.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In *NeurIPS 2018*, 2018.

Gear, C. W. Runge–Kutta starters for multistep methods. *ACM Transactions on Mathematical Software (TOMS)*, 6 (3):263–279, 1980.

Grewal, M. S. and Andrews, A. P. *Kalman Filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.

Griewank, A. and Walther, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2008.

Guckenheimer, J. Dynamics of the Van der Pol equation. *IEEE Transactions on Circuits and Systems*, 27(11):983–989, 1980.

Gustafsson, K. *Control of Error and Convergence in ODE solvers*. PhD thesis, Lund University, 1992.

Hairer, E., Nørsett, S. P., and Wanner, G. *Solving Ordinary Differential Equations I, Nonstiff Problems*. Springer, 1993.

Kersting, H., Krämer, N., Schiegg, M., Daniel, C., Tiemann, M., and Hennig, P. Differentiable likelihoods for fast inversion of "likelihood-free" dynamical systems. In *ICML 2020*, 2020a.

Kersting, H., Sullivan, T. J., and Hennig, P. Convergence rates of Gaussian ODE filters. *Statistics and Computing*, 30(6):1791–1816, 2020b.

Krämer, N. and Hennig, P. Stable implementation of probabilistic ODE solvers. *arXiv:2012.10106*, 2020.

Krämer, N., Schmidt, J., and Hennig, P. Probabilistic numerical method of lines for time-dependent partial differential equations. In *AISTATS 2022*, 2022.

Lindgren, F., Rue, H., and Lindström, J. An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011.

Lorenz, E. N. Predictability: A problem partly solved. In *Proceedings of the Seminar on Predictability*, volume 1, 1996.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005.

Rackauckas, C. and Nie, Q. DifferentialEquations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5(1), 2017.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Särkkä, S. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

Särkkä, S. and Solin, A. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

Schiesser, W. E. *The Numerical Method of Lines: Integration of Partial Differential Equations*. Elsevier, 2012.

Schmidt, J., Krämer, N., and Hennig, P. A probabilistic state space model for joint inference from differential equations and data. *NeurIPS 2021*, 2021.

Schober, M., Duvenaud, D. K., and Hennig, P. Probabilistic ODE solvers with Runge–Kutta means. *NeurIPS 2014*, 2014.

Schober, M., Särkkä, S., and Hennig, P. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, 2019.

Schweppe, F. Evaluation of likelihood functions for Gaussian signals. *IEEE Transactions on Information Theory*, 11(1):61–70, 1965.

Shampine, L. F. and Reichelt, M. W. The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.

Sidén, P. and Lindsten, F. Deep Gaussian Markov random fields. In *ICML 2020*, 2020.

Solin, A. *Stochastic Differential Equation Methods for Spatio-Temporal Gaussian Process Regression*. PhD thesis, 2016.

Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29(6):1297–1315, 2019.

Tronarp, F., Särkkä, S., and Hennig, P. Bayesian ODE solvers: The maximum a posteriori estimate. *Statistics and Computing*, 31(3):1–18, 2021.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

Wanner, G. and Hairer, E. *Solving Ordinary Differential Equations II, Stiff Problems*, volume 375. Springer, 1996.

A. Square-Root Implementation of Probabilistic ODE Solvers

The following two sections detail the square-root implementation of the transitions underlying the probabilistic ODE solver. The whole section is a synopsis of the explanations by Krämer & Hennig (2020). See also the monograph by Grewal & Andrews (2014) for additional details.

A.1. Extrapolation

The extrapolation step

$$C_{n+1}^- = \Phi(h_n) C_n \Phi(h_n)^\top + \Sigma(h_n) \tag{27}$$

does not lead to stability issues further down the line (i.e. in calibration/correction/smoothing steps) if carried out in square root form. Square-root form means that instead of tracking and propagating covariance matrices C, only square root matrices $C = \sqrt{C}\sqrt{C}^\top$ are used for extrapolation and correction steps without forming the full covariance.

This is possible by means of QR decompositions. The matrix square root $\sqrt{C_{n+1}^-}$ arises from $\sqrt{C_n}$ through the QR decomposition of

$$Q \begin{pmatrix} R \\ 0 \end{pmatrix} \leftarrow \begin{pmatrix} \sqrt{C_n}\Phi(h_n)^\top \\ \sqrt{\Sigma(h_n)}^\top \end{pmatrix}, \quad \sqrt{C_{n+1}^-} \leftarrow R^\top \tag{28}$$

because

$$R^\top R = \begin{pmatrix} R \\ 0 \end{pmatrix}^\top Q^\top Q \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} \sqrt{C_n}\Phi(h_n)^\top \\ \sqrt{\Sigma(h_n)}^\top \end{pmatrix}^\top \begin{pmatrix} \sqrt{C_n}\Phi(h_n)^\top \\ \sqrt{\Sigma(h_n)}^\top \end{pmatrix} = \Phi(h_n) C_n \Phi(h_n)^\top + \Sigma(h_n). \tag{29}$$

QR decompositions of a rectangular matrix $M \in \mathbb{R}^{m \times n}$, $m > n$ costs $O(mn^2)$, which implies that the covariance square root correction costs $O(d^3\nu^3)$. The QR decomposition is unique up to orthogonal row-operations (e.g. multiplying with ± 1). Probabilistic ODE solvers require only *any* square root matrix, so this equivalence relation can be safely ignored – they all imply the same covariance.

A.2. Correction

The correction follows a similar pattern. Recall the linearised observation model

$$\mathcal{I}(y) \approx \hat{\mathcal{I}}(y) = Hy + b \tag{30}$$

where H contains the vector field information and the Jacobian information (potentially, depending on linearisation style). There are two ways of performing square root corrections: the conventional way, and the way that is tailored to probabilistic ODE solvers, which builds on Joseph form corrections.

Conventional Way Let $\sqrt{C^-}$ be a matrix square root of the current extrapolated covariance (we drop the $n+1$ index for improved readability). Let 0_n be the $n \times n$ zero matrix, and $0_{n \times m}$ the $n \times m$ zero matrix. The heart of the square root correction is another QR decomposition of the matrix

$$Q \begin{pmatrix} R_{11} & R_{12} \\ 0_{d(\nu+1)\times d} & R_{22} \end{pmatrix} \leftarrow \begin{pmatrix} \sqrt{C^-}^\top H^\top & \sqrt{C^-}^\top \\ 0_d & 0_{d \times d(\nu+1)} \end{pmatrix} \tag{31}$$

with $R_{11} \in \mathbb{R}^{d \times d}$, $R_{12} \in \mathbb{R}^{d \times d(\nu+1)}$, and $R_{22} \in \mathbb{R}^{d(\nu+1) \times d(\nu+1)}$. The R_{ij} matrices contain the relevant information about the covariance matrices involved in the correction:

- $\sqrt{S} := R_{11}^\top$ is the matrix square root of the innovation covariance
- $\sqrt{C} := R_{22}^\top$ is the matrix square root of the posterior covariance
- $K := R_{12}(R_{11})^{-1}$ is the Kalman gain and can be used to correct the mean

This QR decomposition costs $O(d^3\nu^3)$ again, but the matrix involved is larger than the stack of matrices in the extrapolation step (it has d more columns), so for high-dimensional problems, the increased overhead becomes significant. However, if only *any* square root matrix is desired, this step can be circumvented.

Joseph Way Again, let $\sqrt{C^-}$ be the square root matrix of the current extrapolated covariance which results from the extrapolation step in square root form. Next, the full covariance is assembled (which goes against the usual grain of avoiding full covariance matrices, but works well in the present case) as $C^- = \sqrt{C^-}\sqrt{C^-}^\top$. Since in the probabilistic solver, C^- is either a Kronecker matrix $I_d \otimes \breve{C}^-$ or a block diagonal matrix $\mathrm{blockdiag}((C^-)^1, ..., (C^-)^d)$, this is sufficiently cheap. The innovation covariance itself then becomes

$$S = HC^-H^\top \tag{32a}$$
$$= (E_1 - F_xE_0)C^-(E_1 - F_xE_0)^\top \tag{32b}$$
$$= E_1C^-E_1^\top - F_xE_0C^-E_1^\top - E_1C^-E_0^\top F_x^\top + F_xE_0C^-E_0^\top F_x^\top \tag{32c}$$

which can be computed rather efficiently because $E_iC^-E_j^\top$ only involves accessing elements, not matrix multiplication. The only non-negligible cost here is multiplication with the Jacobian of the ODE vector field (which is often sparse in high-dimensional problems). Then, the Kalman gain

$$K = C^-H^\top S^{-1} \tag{33}$$

can be computed from S which implies that the covariance correction reduces to

$$\sqrt{C} = (I - KH)\sqrt{C^-} \tag{34}$$

which is the "left half" of the Joseph correction. The resulting matrix is square, and a matrix square root of the posterior covariance, but not triangular thus no valid Cholesky factor. If the sole purpose of the square root matrices is improved numerical stability, generic square root matrices suffice.

B. Independence and Fast ODE Filters for Time-Constant Diffusion

A similar result to Proposition 3.2 can be formulated for vector-valued, time-constant diffusion models.

Proposition B.1. *Under Assumption 3.1 and for diagonal F_y, a quasi-maximum likelihood estimate (MLE) for a vector-valued, time-constant diffusion model $\Gamma = \mathrm{diag}((\gamma^1)^2, ..., (\gamma^d)^2)$ is given by the estimator*

$$(\hat{\gamma}^i)^2 := \frac{1}{N}\sum_{i=1}^{N} \frac{(z_n^i)^2}{[S_n]_{ii}}, \qquad i = 1, \ldots, d, \tag{35}$$

where $S_n := H(t_n)\Sigma(h_n)H(t_n)^\top$ is the diagonal covariance matrix of the measurement Z_n (recall Section 2.2).

Proof. The proof is structured as follows. First, we show that an initial covariance

$$C_0 = \mathrm{blockdiag}((\gamma^1)^2\breve{C}_0, \ldots, (\gamma^d)^2\breve{C}_0) \tag{36}$$

implies covariances

$$C_n^- = \mathrm{blockdiag}\left((\gamma^1)^2(C_n^1)^-, \ldots, (\gamma^d)^2(C_n^d)^-\right), \tag{37a}$$
$$C_n = \mathrm{blockdiag}\left((\gamma^1)^2C_n^1, \ldots, (\gamma^d)^2C_n^d\right), \tag{37b}$$
$$S_n = \mathrm{diag}\left((\gamma^1)^2s_n^1, \ldots, (\gamma^d)^2s_n^d\right). \tag{37c}$$

Then, for measurement covariances S_n of such form, we can compute the (quasi) maximum likelihood estimate $\hat{\Gamma}$. Because every covariance depends multiplicatively on γ, calibration can happen entirely post-hoc.

Block-Wise Scalar Diffusion Recall from Section 2.1 that the transition matrix and the process noise covariance are of the form $\Phi(h_n) = I_d \otimes \breve{\Phi}(h_n)$ and $\Sigma(h_n) = \Gamma \otimes \breve{\Sigma}(h_n)$. Thus, for a diagonal diffusion $\Gamma = \mathrm{diag}((\gamma^1)^2, ..., (\gamma^d)^2)$, both $\Phi(h_n)$ and $\Sigma(h_n)$ are block diagonal. Assuming a block diagonal covariance matrix that depends multiplicatively on γ,

$$C_{n-1} = \mathrm{blockdiag}\left((\gamma^1)^2C_{n-1}^1, \ldots, (\gamma^d)^2C_{n-1}^d\right), \tag{38}$$

the extrapolated covariance is also of the form

$$C_n^- = \text{blockdiag}\left((\gamma^1)^2(C_n^1)^-, \ldots, (\gamma^d)^2(C_n^d)^-\right), \tag{39a}$$
$$(C_n^i)^- := \breve{\Phi}(h_n) C_{n-1}^i \breve{\Phi}(h_n)^\top + \breve{\Sigma}(h_n), \qquad i = 1, \ldots, d. \tag{39b}$$

The diagonal Jacobian F_y implies a block diagonal linearisation matrix

$$H_n = E_1 - F_y E_0 = \text{blockdiag}\left(H_n^1, \ldots, H_n^d\right), \tag{40a}$$
$$H_n^i := e_1 - [F_y]_{i,i} e_0, \qquad i = 1, \ldots, d. \tag{40b}$$

The measurement covariance S_n is therefore given by a *diagonal* matrix and depends multiplicatively on γ, as

$$S_n = H_n C_n^- H_n^\top = \text{diag}\left((\gamma^1)^2 s_n^1, \ldots, (\gamma^d)^2 s_n^d\right), \tag{41a}$$
$$s_n^i := H_n^i (C_n^i)^- (H_n^i)^\top, \qquad i = 1, \ldots, d. \tag{41b}$$

This implies a block diagonal Kalman gain

$$\Xi_n = I - C_n^- H(t_n)^\top S_n^{-1} H(t_n) = \text{blockdiag}\left(\Xi_n^1, \ldots \Xi_n^d\right), \tag{42a}$$
$$\Xi_n^i := I_{\nu+1} - (C_n^-)^i (H_n^i)^\top H_n^i / s_n^i \qquad i = 1, \ldots, d. \tag{42b}$$

Finally, we obtain the corrected covariance

$$C_n = \text{blockdiag}\left((\gamma^1)^2 C_n^1, \ldots, (\gamma^d)^2 C_n^d\right), \tag{43a}$$
$$C_n^i := \Xi_n^i (C_n^i)^- (\Xi_n^i)^\top \qquad i = 1, \ldots, d. \tag{43b}$$

This concludes the first part of the proof.

Computing The Quasi-MLE It is left to compute the (quasi) MLE $\hat{\Gamma} = \text{diag}((\hat{\gamma}^1)^2, ..., (\hat{\gamma}^d)^2)$ by maximizing the log-likelihood $\log p(z_{1:N}) = \log \prod_{n=1}^N \mathcal{N}(0; z_n, S_n)$. Since $S_n = \text{diag}\left((\gamma^1)^2 s_n^1, \ldots, (\gamma^d)^2 s_n^d\right)$ is a diagonal matrix, we obtain

$$\hat{\Gamma} = \arg\max_{\Gamma} \sum_{n=1}^N \log \mathcal{N}(0; z_n, S_n) \tag{44a}$$
$$= \arg\max_{\Gamma} \sum_{i=1}^d \left(-\frac{N \log(\hat{\gamma}^i)^2}{2} - \sum_{n=1}^N \frac{(z_n)_d^2}{2 s_n^i (\hat{\gamma}^i)^2} \right). \tag{44b}$$

By taking the derivative and setting it to zero, we obtain the quasi-MLE from Equation (35).

□

C. Proof of Proposition 4.3

Proof. Let $Y_n \sim \mathcal{N}(m_n, C_n)$. Assume $C_n = \Gamma \otimes \breve{C}_n$ which is no loss of generality, because such a Kronecker structure is preserved through the ODE filter step as shown below.

(i) Extrapolate mean: The mean extrapolation costs $O(d\nu^2)$ like in the proof of Proposition 3.3.

(ii) Evaluate the ODE: Evaluation of H and b is essentially free—recall that we only consider the EK0 in this setting, which uses the projection $H(t_n) = E_1$. Matrix multiplication with H consists of a projection, which costs $O(1)$.

(iii) Calibrate: Calibration of a time-constant γ^2 costs $O(d)$ under Assumption 4.2.

(iv) Extrapolate covariance: In the time-constant diffusion model, $\Sigma(h_n)$ and C_n are both Kronecker matrices and share the left Kronecker factor: Γ. Thus, the extrapolation of the covariance can be carried out "in the right Kronecker factor", which costs $O(\nu^3)$ in traditional as well as square root implementation. Denote the extrapolated covariance by $C_{n+1}^- := \Gamma \otimes \breve{C}_{n+1}^-$.

(v) Measure: Recall $H(t_n) = E_1 = I \otimes e_1$. The mean of the measured random variable $Z_n \sim \mathcal{N}(z_n, S_n)$ comes at negligible cost. The covariance

$$S_{n+1} = H(t_{n+1})C^-_{n+1}H(t_{n+1})^\top = \Gamma \otimes \left[e_1 \breve{C}^-_{n+1} e_1^\top\right] \tag{45}$$

requires a single element in $\breve{C}^-_{n+1}$. The Kalman gain

$$K := C^-_{n+1}H(t_{n+1})^\top S^\top_{n+1} = I \otimes \breve{K}, \tag{46}$$

with $\breve{K} := e_1\breve{C}^-_{n+1} / \left[e_1\breve{C}^-_{n+1}e_1^\top\right]$ involves dividing the first row of $\breve{C}^-_{n+1}$ by a scalar. Its cost is $O(\nu+1)$.

(v) Correct mean and covariance: The mean is corrected in $O(d\nu^2)$ as in the proof of Proposition 3.3. Due to the Kronecker structure in K, the "left" Kronecker factor of C_{n+1} must be Γ again. Therefore, we need to correct only the "right" Kronecker factor in $O(\nu^3)$.

All in all, under the assumption of cheap calibration, a single step with the EK0 costs $O(d\nu^2)$ and the expensive steps are (as before) the covariance extrapolation and the covariance correction. The total memory costs are the requirements of storing Γ, the mean(s) in $O(\nu d)$, and the "right" Kronecker factor(s) in $O(\nu^2)$. □

D. Kronecker Structure and Fast ODE Filters for Time-Varying Diffusion

In the following we extend the results of Proposition 4.3 to time-varying diffusion models.

Proposition D.1. *Under Assumption 4.2, and if a time-varying diffusion model $\Gamma_n = \gamma_n^2\breve{\Gamma}$ is calibrated via γ_n, a single step of the EK0 costs $O(\nu^3 + d\nu^2)$ floating point operations, and $O(d\nu + d^2 + \nu^2)$ memory.*

Proof. The proof of Proposition 4.3 shown in Appendix C depends on the specific time-fixed diffusion model only in the calibration (iii) and the extrapolation of the covariance (iv). In the following, we discuss these two steps for a time-varying diffusion $\Gamma_n = \gamma_n^2\breve{\Gamma}$. We show that Kronecker structure is preserved and we obtain the same complexities as in Proposition 4.3.

(iii) Calibrate: Calibration of a time-varying γ^2_{n+1} is done with

$$\begin{aligned}
\hat{\gamma}^2_{n+1} :&= \frac{1}{d} z_{n+1}^\top \left[H(t_{n+1})\Sigma(h_{n+1})H(t_{n+1})^\top\right]^{-1} z_{n+1} && \text{(47a)}\\
&= \frac{1}{d} z_{n+1}^\top \left(\breve{\Gamma}_{n+1} \cdot \left[\breve{\Sigma}^-_{n+1}\right]_{11}\right)^{-1} z_{n+1} && \text{(47b)}\\
&= \frac{1}{d} z_{n+1}^\top \left(\breve{\Gamma}_{n+1}\right)^{-1} z_{n+1} \Big/ \left[\breve{\Sigma}^-_{n+1}\right]_{11}, && \text{(47c)}
\end{aligned}$$

where we used that $H(t_n) = I \otimes e_1$. With Assumption 4.2, this computation costs $O(d)$.

(iv) Extrapolate covariance: Assume a covariance of the form $C_n = \left(\gamma_n^2\breve{\Gamma}\right) \otimes \breve{C}_n$. Since scalars can be moved between the Kronecker factors, the covariance matrix can be written with the diffusion matrix $\Gamma_{n+1} = \gamma^2_{n+1}\breve{\Gamma}$, as

$$C_n = \left(\gamma^2_{n+1}\breve{\Gamma}\right) \otimes \left(\frac{\gamma_n^2}{\gamma^2_{n+1}}\breve{C}_n\right). \tag{48}$$

Then, since $\Sigma_{n+1} = \left(\gamma^2_{n+1}\breve{\Gamma}\right) \otimes \breve{\Sigma}_{n+1}$, the prediction step can be written as

$$\begin{aligned}
C^-_{n+1} &= \Phi(h_{n+1})C_n\Phi(h_{n+1})^\top + \Sigma(h_{n+1}) && \text{(49a)}\\
&= \left(I_d \otimes \breve{\Phi}(h_{n+1})\right)\left(\left(\gamma^2_{n+1}\breve{\Gamma}\right) \otimes \left(\frac{\gamma_n^2}{\gamma^2_{n+1}}\breve{C}_n\right)\right)\left(I_d \otimes \breve{\Phi}(h_{n+1})\right)^\top + \left(\left(\gamma^2_{n+1}\breve{\Gamma}\right) \otimes \breve{\Sigma}_{n+1}\right) && \text{(49b)}\\
&= \left(\gamma^2_{n+1}\breve{\Gamma}\right) \otimes \left(\frac{\gamma_n^2}{\gamma^2_{n+1}}\breve{\Phi}(h_{n+1})\breve{C}_n\breve{\Phi}(h_{n+1})^\top + \breve{\Sigma}_{n+1}\right) && \text{(49c)}\\
&=: \left(\gamma^2_{n+1}\breve{\Gamma}\right) \otimes \breve{C}^-_{n+1}. && \text{(49d)}
\end{aligned}$$

With a Kalman gain of the form $K = I \otimes \breve{K}$, the corrected covariance can be written as $C_{n+1} = (\gamma_{n+1}^2 \breve{\Gamma}) \otimes \breve{C}_{n+1}$, thus confirming our assumption on the Kronecker structure of covariance matrices. Since all matrix multiplications happen only "in the right Kronecker factor", extrapolating the covariance costs $O(\nu^3)$.

All other parts of the proof can be reproduced as in Appendix C to obtain the specified complexities. □

E. Additional Details on the Empirical Evaluation

E.1. Figure 1: Simulating a high-dimensional ODE

The considered FitzHugh–Nagumo PDE is provided in Appendix E.4. For this first visualization, the PDE solved on the time span $t \in [0, 20]$ and in a spatial domain $x \in [-1.25, 1.25]^2$, discretized in intervals $\Delta_x = 0.01$. The probabilistic numerical solution is computed with a diagonal EK1 solver with a 2-times integrated Wiener process (IWP(2)) prior and a time-varying scalar diffusion model. Adaptive steps are chosen with tolerance levels $\tau_{\text{abs}} = 10^{-3}, \tau_{\text{rel}} = 10^{-1}$.

E.2. Single ODE Filter Step

Lorenz96 The Lorenz96 model describes a chaotic dynamical system for which the dimension can be chosen freely (Lorenz, 1996). It is given by a system of $N \geq 4$ ODEs

$$\dot{y}_1 = (y_2 - y_{N-1})y_N - y_1 + F, \tag{50a}$$
$$\dot{y}_2 = (y_3 - y_N)y_1 - y_2 + F, \tag{50b}$$
$$\dot{y}_i = (y_{i+1} - y_{i-2})y_{i-1} - y_i + F \qquad i = 3, \ldots, N-1, \tag{50c}$$
$$\dot{y}_N = (y_1 - y_{N-2})y_{N-1} - y_N + F, \tag{50d}$$

with forcing term $F = 8$, initial values $y_1(0) = F + 0.01$ and $y_{>1}(0) = F$, and time span $t \in [0, 30]$.

E.3. Full Simulation

Pleiades The Pleiades system describes the motion of seven stars in a plane, with coordinates (x_i, y_i) and masses $m_i = i$, $i = 1, \ldots, 7$ (Hairer et al., 1993, Section II.10). It can be described with a system of 28 ODEs

$$\dot{x}_i = v_i \tag{51a}$$
$$\dot{y}_i = w_i \tag{51b}$$
$$\dot{v}_i = \sum_{j \neq i} m_j (x_j - x_i)/r_{ij}, \tag{51c}$$
$$\dot{w}_i = \sum_{j \neq i} m_j (y_j - y_i)/r_{ij}, \tag{51d}$$

where $r_{ij} = \left((x_i - x_j)^2 + (y_i - y_j)^2\right)^{3/2}$, for $i, j = 1, \ldots, 7$. It is commonly solved on the time span $t \in [0, 3]$ and with initial locations

$$x(0) = [3, 3, -1, -3, 2, -2, 2], \tag{52a}$$
$$y(0) = [3, -3, 2, 0, 0, -4, 4], \tag{52b}$$
$$v(0) = [0, 0, 0, 0, 0, 1.75, -1.5], \tag{52c}$$
$$w(0) = [0, 0, 0, -1.25, 1, 0, 0]. \tag{52d}$$

Further Details All considered probabilistic numerical solvers use a 4-times integrated Wiener process (IWP(4)) prior, as well as a time-varying scalar diffusion. The SciPy solutions and the fast PN solutions shown in Figure 3 correspond to absolute and relative tolerance levels $\tau_{\text{abs}}, \tau_{\text{rel}} \in \{10^{-i}\}_{i=3}^{12}$; PN solutions in their traditional implementation are only computed for tolerances $\tau_{\text{abs}}, \tau_{\text{rel}} \in \{10^{-i}\}_{i=3}^{10}$. The references solution is computed with the LSODA solver and with absolute and relative tolerances of $\tau_{\text{abs}} = 10^{-12}, \tau_{\text{rel}} = 10^{-12}$.

E.4. High-Dimensional Setting

FitzHugh–Nagumo PDE Let $\Delta = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}$ be the Laplacian. The FitzHugh–Nagumo partial differential equation (PDE) is (Ambrosio & Françoise, 2009)

$$\frac{\partial}{\partial t} u(t,x) = a\Delta u(t,x) + u(t,x) - u(t,x)^3 - v(t,x) + k, \tag{53a}$$

$$\frac{\partial}{\partial t} v(t,x) = \frac{1}{\tau}(b\Delta v(t,x) + u(t,x) - v(t,x)), \tag{53b}$$

for some parameters a, b, k, τ, and initial values $u(t_0, x) = h_0(x)$, $v(t_0, x) = h_1(x)$. In our experiments, we chose $a = 208 \cdot 10^{-4}$, $b = 5 \cdot 10^{-3}$, $k = -5 \cdot 10^{-3}$, $\tau = 0.1$. As initial values, we used random samples from the uniform distribution on $(0, 1)$. We solve the PDE from $t_0 = 0$ to $t_{\text{max}} = 20$ on a range of spatial domains $x \in [0, W] \times [0, W] \subseteq \mathbb{R}^2$, with $W \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50\}$. To turn the PDE into a system of ODEs, we discretised the Laplacian with central, second-order finite differences schemes on a uniform grid. The mesh size of the grid determines the number of grid points, which controls the dimension of the ODE problem.

Further Details PN solutions are computed with a 3-times integrated Wiener process (IWP(3)) prior, a time-varying scalar diffusion, and with step-size adaptation for chosen tolerances $\tau_{\text{abs}} = 10^{-3}, \tau_{\text{rel}} = 10^{-1}$. The DOP853 solutions are computed with tolerance levels $\tau_{\text{abs}} = 10^{-6}, \tau_{\text{rel}} = 10^{-3}$.

E.5. Stability of the Diagonal EK1

Van der Pol The Van der Pol system is often employed to evaluate the stability of stiff ODE solvers (Wanner & Hairer, 1996). It is given by a system of ODEs

$$\dot{y_1}(t) = y_2(t), \qquad \dot{y_2}(t) = \mu \left(\left(1 - y_1^2(t)\right) y_2(t) - y_1(t) \right), \tag{54}$$

with stiffness constant $\mu > 0$, time span $t \in [0, 6.3]$, and initial value $y(0) = [2, 0]$.

Further Details All considered probabilistic numerical solutions are computed with a 5-times integrated Wiener process (IWP(5)) prior, a time-varying scalar diffusion model, and with step-size adaptation for tolerances $\tau_{\text{abs}} = 10^{-6}, \tau_{\text{rel}} = 10^{-3}$.

Fenrir: Physics-Enhanced Regression for Initial Value Problems

Filip Tronarp [*1] **Nathanael Bosch** [*1] **Philipp Hennig** [1 2]

Abstract

We show how probabilistic numerics can be used to convert an initial value problem into a Gauss–Markov process parametrised by the dynamics of the initial value problem. Consequently, the often difficult problem of parameter estimation in ordinary differential equations is reduced to hyperparameter estimation in Gauss–Markov regression, which tends to be considerably easier. The method's relation and benefits in comparison to classical numerical integration and gradient matching approaches is elucidated. In particular, the method can, in contrast to gradient matching, handle partial observations, and has certain routes for escaping local optima not available to classical numerical integration. Experimental results demonstrate that the method is on par or moderately better than competing approaches.

1. Introduction

Consider the following initial value problem (IVP)

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi_\theta(t) = f_\theta\left(t, \varphi_\theta(t)\right), \qquad t \in [0, T], \tag{1}$$

where the vector field $f_\theta \colon [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ and the initial condition $\varphi_\theta(0) = y_0(\theta)$ are both parametrised by θ. In this article, the concern lies in estimating θ from noisy measurements of the following form

$$u(t) = H^\mathsf{T}\varphi_\theta(t) + v(t), \quad v(t) \sim \mathcal{N}(0, R_\theta), \tag{2}$$

where $t \in \mathbb{T}_\mathsf{D} \subset [0,T]$ is the finite set of measurement nodes and H is a measurement matrix of appropriate dimension. This is a ubiquitous problem in science and engineering. Examples include ecology (Benson, 1979), pharmacokinetics (Gelman et al., 1996), process engineering (Åström & Eykhoff, 1971), and brain imaging (Friston, 2002).

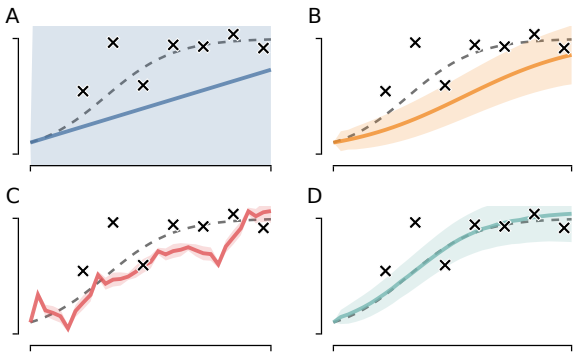


Figure 1. **From an uninformed prior to a calibrated posterior.** Starting from a standard Gauss–Markov prior (A), Fenrir first computes a physics-enhanced prior with probabilistic numerics (B), and then a posterior via Gauss–Markov regression (C). By maximizing the marginal likelihood, we obtain a calibrated posterior and parameter estimates for the underlying dynamical system (D). The data generating model is the logistic equation, which corresponds to the vector field $f(t, y) = ry(1-y)$.

The likelihood functional $\mathcal{L}_\mathsf{D}$, evaluated at some function y, is given by

$$\mathcal{L}_\mathsf{D}(R_\theta, y) = \prod_{t \in \mathbb{T}_\mathsf{D}} \mathcal{N}\left(u(t); H^\mathsf{T} y(t), R_\theta\right),$$

and the marginal likelihood $\mathcal{M}$ of some parameter θ can be expressed by evaluating $\mathcal{L}_\mathsf{D}$ at the corresponding solution φ_θ according to

$$\mathcal{M}(\theta) = \mathcal{L}_\mathsf{D}\left(R_\theta, \varphi_\theta\right).$$

The parameter θ may be estimated by maximising $\mathcal{M}$. A persistent challenge in likelihood-based inference in initial value problems is the fact that φ_θ, and therefore the likelihood, are intractable (Bard, 1974).

A standard appproach to approximating the likelihood is based on solving the IVP numerically (Hairer et al., 1987). However, in optimisation-based inference it has been observed that this leads to many local optima (Cao et al., 2011), and can lead to divergence of the optimiser (Dass et al., 2017). On the other hand, slow convergence and poor mixing has been observed for Monte Carlo-based inference

[*]Equal contribution [1]Department of Computer Science, University of Tübingen, Tübingen, Germany [2]Max–Planck Institute for Intelligent Systems, Tübingen, Germany. Correspondence to: Filip Tronarp <filip.tronarp@uni-tuebingen.de>, Nathanael Bosch <nathanael.bosch@uni-tuebingen.de>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

(Alahmadi et al., 2020), which have led some authors to favour likelihood-free methods (Toni et al., 2009). Another alternative is gradient matching (Voit, 2000) with splines (Varah, 1982; Gugushvili & Klaassen, 2012) or Gaussian processes (Calderhead et al., 2009; Dondelinger et al., 2013; Gorbach et al., 2017; Wenk et al., 2020).

1.1. Contribution

In the present work, a probabilistic numerics approach is developed for computing the marginal likelihood. Probabilistic numerics aims at producing probability measures for solutions of numerical problems, thus giving a probabilistic description of the numerical error (Hennig et al., 2015; Oates & Sullivan, 2019).

The marginal likelihood may be viewed as $\mathcal{L}_{\mathsf{D}}$ integrated against a Dirac measure located at φ_θ according to

$$\mathcal{M}(\theta) = \int \mathcal{L}_{\mathsf{D}}(R_\theta, y)\delta(y - \varphi_\theta)\,\mathrm{d}y. \tag{3}$$

While this representation is not immediately advantageous, it is instructive for understanding the probabilistic numerics approach. Namely, it produces an approximation to the Dirac measure, giving the following approximate marginal likelihood

$$\widehat{\mathcal{M}}_N(\theta, \kappa) = \int \mathcal{L}_{\mathsf{D}}(R_\theta, y)\widehat{\delta}_N(y \mid \theta, \kappa)\,\mathrm{d}y, \tag{4}$$

where $\widehat{\delta}_N$ is the output of a suitably chosen probabilistic numerical method, which is parametrised by κ. It should be noted that $\mathcal{L}_{\mathsf{D}}$ only depends on point evaluations of y on the grid $\mathbb{T}_{\mathsf{D}}$. Therefore, it is sufficient to operate on the finite dimensional distributions of $\widehat{\delta}_N$ to compute $\widehat{\mathcal{M}}_N$.

Kersting et al. (2020a) has previously used the representation (4) and approximated its gradients in combination with low order explicit solvers, at a cost of $O(N^3)$.

The aim of this article is to show how both $\widehat{\delta}_N$ and $\widehat{\mathcal{M}}_N$ can be computed efficiently for general probabilistic solvers, at a cost of $O(N)$. The method consists of two parts:

1. Efficiently construct a Gauss–Markov representation of $\widehat{\delta}_N(y \mid \theta, \kappa)$ using probabilistic numerics.
2. Compute $\widehat{\mathcal{M}}_N(\theta, \kappa)$ and its derivatives via Gauss–Markov regression and automatic differentiation.

The first step essentially takes the initial value problem and produces a *physics-enhanced* Gauss–Markov prior. The second step utilises this prior in standard Gauss–Markov regression to estimate parameters and reconstruct the trajectory (Särkkä & Solin, 2019). Therefore, the method is called Physics-enhanced regression in initial value problems, or *Fenrir* for short. Here physics is used to refer to any mechanistic information pertaining to the dynamics of the data generating process. The method is illustrated in Figure 1.

The rest of the article is organised as follows. Probabilistic numerical solvers are reviewed in Section 2. In Section 3 it is shown how to use probablistic numerics to construct a physics-enhanced Gauss–Markov prior for initial value problems, thus reducing the marginal likelihood to Gauss–Markov regression. Related work is discussed in Section 4, which is followed by experimental results in Section 5. Finally, concluding remarks are given in Section 6.

2. Probabilistic Numerical IVP Solvers

In the Bayesian formulation, an IVP solver is completely specified by a prior and the definition of the data, on which it is conditioned. The latter is obtained by means of an information operator (Cockayne et al., 2019). For constructing a probabilistic numerical solver, we follow the account of Tronarp et al. (2019b; 2021).

2.1. Prior Specification

The probabilistic numerics prior is defined as the output of the following stochastic state-space model

$$\mathrm{d}x(t) = Ax(t)\,\mathrm{d}t + \sqrt{\kappa}B\,\mathrm{d}w(t), \quad x(0) = x_\theta^\dagger, \tag{5a}$$
$$y^{(m)}(t) = \mathrm{E}_m^{\mathsf{T}} x(t), \quad m = 0, 1, \ldots, \nu, \tag{5b}$$

where $x \in \mathbb{R}^{d(\nu+1)}$ models the solution and its ν first derivatives and E_m are selection matrices for the mth derivative of the prior model for the solution of (1), which is denoted by y. Furthermore, $x_\theta^\dagger$ denotes the initial condition of x, $A \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$ and $B \in \mathbb{R}^{d(\nu+1)}$ are model matrices and w is a standard Wiener process in $\mathbb{R}^d$ (Øksendal, 2003).

The state x is a Markov process by construction, with transition density given by (Särkkä & Solin, 2019)

$$\Phi(h) = e^{Ah},$$
$$Q(h) = \int_0^h \Phi(h-\tau)BB^{\mathsf{T}}\Phi^{\mathsf{T}}(h-\tau)\,\mathrm{d}\tau,$$
$$x(t+h) \mid x(t) \sim \mathcal{N}\big(x(t+h); \Phi(h)x(t), \kappa Q(h)\big),$$

which facilitates fast computation for the probabilistic solver and our subsequent marginal likelihood approximation. Additional details on priors for probabilistic solutions of initial value problems can be found in Appendix A.1.

2.2. Data Model

In order to define a data model for probabilistic numerical solvers, a grid

$$\mathbb{T}_{\mathsf{PN}} = \{t_n\}_{n=1}^N \subset [0, T],$$

needs to be coupled with an information operator. The canonical information operator for initial value problems is given by (Tronarp et al., 2021)

$$\mathcal{Z}_\theta[x](t) = \mathrm{E}_1^\mathsf{T} x(t) - f_\theta(t, \mathrm{E}_0^\mathsf{T} x(t)), \tag{6}$$

but there are alternatives that, for instance, also take geometric invariants into account (Bosch et al., 2022).

Note that $\mathcal{Z}$ map solutions of the initial value problem to the zero function, which is a known value. In fact, the set of functions starting at $y_0(\theta)$ which are mapped to the zero function by $\mathcal{Z}$ constitutes the set of solutions to the initial value problem (Arnol'd, 1992).[1] An appropriate data model for a probabilistic numerical solver is thus given by

$$z(t) = \mathcal{Z}_\theta[x](t) = 0, \quad t \in \mathbb{T}_{\mathsf{PN}}, \tag{7}$$

where $z(t) = 0$ is enforced only on the chosen grid as to arrive at a practical algorithm.

It should be noted that the grid $\mathbb{T}_{\mathsf{PN}}$ does not need to be specified a priori but can be constructed adaptively to control the solution error (Schober et al., 2019; Bosch et al., 2021).

2.3. Initial Value Problem Solvers as Non-linear Gauss–Markov Regression

The prior (5), data model (6), and data definition (7) define a non-linear Gauss–Markov regression problem according to Tronarp et al. (2019b)

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}\big(\Phi(\Delta_n) x(t_{n-1}), \kappa Q(\Delta_n)\big), \tag{8a}$$
$$z(t_n) \mid x(t_n) \sim \mathcal{N}\big(\mathrm{E}_1^\mathsf{T} x(t) - f_\theta(t, \mathrm{E}_0^\mathsf{T} x(t)), 0\big), \tag{8b}$$
$$z(t_n) \coloneqq 0, \tag{8c}$$

where $\Delta_n = t_n - t_{n-1}$ is the step-size of the nth step, $x(t_0) = x_\theta^\dagger$ by convention, and $\mathcal{N}(\cdot\,, 0)$ denotes the Dirac distribution. The probablistic numerical solver for (1) associated with the prior (5) and the data (7) is on the grid $\mathbb{T}_{\mathsf{PN}}$ given by

$$\begin{aligned} \gamma_N(t_{1:N}, x_{1:N} \mid \theta, \kappa) &= c^{-1}(\theta, \kappa) \\ &\times \prod_{n=1}^{N} \mathcal{N}\big(x_n; \Phi(\Delta_n) x_{n-1}, \kappa Q(\Delta_n)\big) \\ &\times \prod_{n=1}^{N} \delta\big(\mathrm{E}_1^\mathsf{T} x_n - f_\theta(t_n, \mathrm{E}_0^\mathsf{T} x_n)\big), \end{aligned} \tag{9}$$

where $c(\theta, \kappa)$ is a norming constant. Due to the potential non-linearity of the vector field, this object is generally intractable. However, when the vector field is linear, say

$$f_\theta(t, y) = L_\theta(t) y + b_\theta(t), \tag{10}$$

then the densities of the time marginals can be computed efficiently via Kalman filtering and Rauch–Tung–Striebel smoothing (Kalman, 1960; Rauch et al., 1965).

This fact is exploited for approximate inference when the vector field is non-linear as well. Indeed several linearisation approaches have been employed (Schober et al., 2019; Tronarp et al., 2019b; 2021), which have been demonstrated to yield accurate solvers both empirically (Schober et al., 2019; Bosch et al., 2021; Krämer & Hennig, 2020) and theoretically (Kersting et al., 2020b; Tronarp et al., 2021).

2.4. Initial Value Problem Solvers as Kalman Filtering

The Kalman filtering recursion for (8) when the vector field is affine as in (10), recursively computes the densities

$$\pi(x(t_n) \mid z(t_{1:n})) = \mathcal{N}(\mu_\theta(t_n), \Sigma_\theta(t_n)), \tag{11}$$

which are the time marginals conditioned on all past data up to the present. The recursion is initialised by setting $\mu_\theta(t_0) = x_\theta^\dagger, \Sigma_\theta(t_0) = 0$, and then alternates between prediction and update.

- Prediction:

$$\begin{aligned} \mu_\theta(t_n^-) &= \Phi(\Delta_n)\mu_\theta(t_{n-1}), \\ \Sigma_\theta(t_n^-) &= \Phi(\Delta_n)\Sigma_\theta(t_{n-1})\Phi^\mathsf{T}(\Delta_n) + Q(\Delta_n). \end{aligned}$$

- Update:

$$\begin{aligned} C_\theta(t_n) &= \mathrm{E}_1 - \mathrm{E}_0 L_\theta^\mathsf{T}(t_n), \\ S_\theta(t_n) &= C_\theta^\mathsf{T}(t_n)\Sigma_\theta(t_n^-)C_\theta(t_n), \\ K_\theta(t_n) &= \Sigma_\theta(t_n^-)C_\theta^\mathsf{T}(t_n)S_\theta^{-1}(t_n), \\ e_\theta(t_n) &= b_\theta(t_n) - C_\theta^\mathsf{T}(t_n)\mu_\theta(t_n^-), \\ \mu_\theta(t_n) &= \mu_\theta(t_n^-) + K_\theta(t_n)e(t_n), \\ \Sigma_\theta(t_n) &= \Sigma_\theta(t_n^-) - K_\theta(t_n)S_\theta(t_n)K_\theta^\mathsf{T}(t_n). \end{aligned}$$

The following parameters can be computed from the outputs of the Kalman filter

$$G_\theta(t_{n-1}) = \Sigma_\theta(t_{n-1})\Phi^\mathsf{T}(\Delta_n)\Sigma_\theta^{-1}(t_n^-), \tag{12a}$$
$$P_\theta(t_{n-1}) = \Sigma_\theta(t_{n-1}) - G_\theta(t_{n-1})\Sigma_\theta(t_n^-)G_\theta^\mathsf{T}(t_{n-1}). \tag{12b}$$

They are used for the smoothing recursion and the representation of the probabilistic numerics posterior.

3. Fenrir

In this section, it is shown that the probabilistic numerical solver yields a Gauss–Markov process approximation to (1). Consequently, inference given measurements (2) reduces to a Gauss–Markov regression problem with a *physics-enhanced prior* as determined by the probabilistic solver.

[1] Typically, we assume that the vector field is regular enough for there to be a unique solution of the initial value problem.

3.1. Probabilistic Numerical IVP Solutions as Gauss–Markov Processes

Linearising the vector field allows for approximate computation of the time marginal densities via the Rauch–Tung–Striebel smoother. These linearisations imply a Gauss–Markov representation of the approximate posterior, which in fact is used in the Bayesian derivation of the smoothing algorithm (Särkkä, 2013, c.f. proof of theorem 8.2). The following result lie at the heart of our method.

Proposition 3.1 (Gauss–Markov representation of the probabilistic solver)**.** *The restriction of the probabilistic numerics posteriors to the grid $\mathbb{T}_{\mathsf{PN}}$ admit the following representation*

$$\begin{aligned}\widehat{\gamma}_N(t_{1:N}, x_{1:N} \mid \theta, \kappa) &= \mathcal{N}\big(x_N; \xi_\theta(t_N), \kappa\Lambda_\theta(t_N)\big) \\ &\quad \prod_{n=N-1}^{1} \mathcal{N}\big(x_n; G_\theta(t_n)x_{n+1} + \zeta_\theta(t_n), \kappa P_\theta(t_n)\big),\end{aligned} \tag{13}$$

where $\xi_\theta(t_N) = \mu_\theta(t_N)$, $\Lambda_\theta(t_N) = \Sigma_\theta(t_N)$,

$$\zeta_\theta(t_n) = \mu_\theta(t_n) - G_\theta(t_n)\mu_\theta(t_{n+1}^-),$$

and (G, P) are given by (12).

For completeness, a detailed derivation of proposition 3.1 is given in Appendix A.2. Note that $\widehat{\gamma}_N$ is represented as a Gauss–Markov process running backwards in time. It represents a probabilistic approximation to the solution of the IVP and its derivatives, in terms of a conditional distribution given numerical data (7) and the parameter θ.

3.2. Inference in IVPs as Gauss–Markov Regression

In the previous section, the approximate Dirac $\widehat{\delta}_N(y \mid \theta)$ was implicitly defined through γ_N in (9). The purpose here is to turn this into an implementable algorithm for approximating the marginal likelihood. For ease of notation it is assumed that $\mathbb{T}_{\mathsf{D}} \subset \mathbb{T}_{\mathsf{PN}}$, in which case,

$$\begin{aligned}\widehat{\mathcal{M}}_N(\theta, \kappa) &= \int \mathcal{L}_{\mathsf{D}}(\theta, y)\widehat{\delta}_N(y \mid \theta, \kappa)\,\mathrm{d}y \\ &= \int \mathcal{L}_{\mathsf{D}}(R_\theta, \mathrm{E}_0^{\mathsf{T}}x)\gamma_N(t_{1:N}, x_{1:N} \mid \theta, \kappa)\,\mathrm{d}x_{1:N}.\end{aligned}$$

Additionally, the calibration parameter κ is also included in the marginal likelihood approximation. In practice, γ_N is replaced by its approximation $\widehat{\gamma}_N$ in (13). This results in the following approximation to the marginal likelihood

$$\begin{aligned}\widehat{\mathcal{M}}_N(\theta, \kappa) = \int \prod_{t_n \in \mathbb{T}_{\mathsf{D}}} \mathcal{N}(u(t_n); H^{\mathsf{T}}\mathrm{E}_0^{\mathsf{T}}x_n, R_\theta) \\ \times \widehat{\gamma}_N(t_{1:N}, x_{1:N} \mid \theta, \kappa)\,\mathrm{d}x_{1:N}.\end{aligned} \tag{14}$$

Consequently, the problem of computing the marginal likelihood and trajectory estimates is reduced to inference in the following linear state-space model

$$x(t_N) \sim \mathcal{N}(\xi(t_N), \kappa\Lambda(t_N)), \tag{15a}$$

$$x(t_n) \mid x(t_{n+1}) \sim \widehat{\gamma}_N(x(t_n) \mid x(t_{n+1}), \theta, \kappa), \tag{15b}$$

$$u(t) \mid x(t) \sim \mathcal{N}(H^{\mathsf{T}}\mathrm{E}_0^{\mathsf{T}}x(t), R_\theta), \quad t \in \mathbb{T}_{\mathsf{D}}, \tag{15c}$$

where the backwards transition densities can be read from (13). Therefore, estimating the trajectory of the solution (1) can also be done via Kalman filtering and smoothing. Furthermore, the marginal likelihood approximation can be computed via the Kalman filter through the prediction error decomposition (Schweppe, 1965). Complete details on how to compute trajectory estimates and marginal likelihoods in (15) are given in Appendix B.

Computational complexity The computation of $\widehat{\gamma}_N$ and $\widehat{\mathcal{M}}_N$ can be implemented with Gauss–Markov regression with a state dimension of $d(\nu + 1)$. Therefore, assuming the measurement dimension is smaller, the computational complexity of the method is $O(Nd^3(\nu+1)^3)$. That is, it is linear in the number of data points, in contrast to cubic complexity for standard Gaussian process regresison. Further speed-ups may be obtainable by exploiting structural simplifications for certain probabilistic solvers (Krämer et al., 2021).

Hyperparameter estimation The present method provides a marginal likelihood (14); its derivatives can be computed with automatic differentiation. Consequently, Fenrir interacts with various inference methods, such as gradient-based optimisation or Markov Chain Monte Carlo, in a plug-and-play fashion. In this paper, the maximum likelihood approach is examined.

Model selection The marginal likelihood approximation (14) confers other benefits than providing a cost function for parameter inference. Namely, the possibility for a probabilistically motivated model comparisons, such as likelihood ratio testing for nested models (King, 1998), or via various information criteria (Akaike, 1974; Stoica & Selen, 2004).

4. Related Work: A Tale of Three Approaches

Three different approaches to parameter estimation in initial value problems can be discerned, namely (a) numerical integration, (b) gradient matching, and (c) probabilistic numerics. In order to get a comprehensive lay of the land of parameter estimation in ordinary differnetial equations, these approaches are reviewed in this section. Particular care is taken to highlighting similarities and differences.

4.1. Classical Numerical Integration

The traditional approach is to estimate the parameters via non-linear regression (Biegler et al., 1986), where the correct solution to (1) is replaced by a numerical approximation,

say Runge–Kutta (Hairer et al., 1987). Thus the marginal likelihood approximation reads

$$\widehat{\mathcal{M}}_N(\theta) = \int \prod_{t_n \in \mathbb{T}_\mathsf{D}} \mathcal{N}(u(t_n); H^\mathsf{T} y_n, R_\theta) \times \prod_{t_n \in \mathbb{T}_\mathsf{D}} \delta(y_n - \hat{\varphi}_\theta(t_n)) \, \mathrm{d}y_{1:N}. \tag{16}$$

That is, likelihood computation via numerical integration computes the Dirac approximation $\widehat{\delta}_N$ in (4) by approximating the location of the Dirac in (3).

4.2. Gradient Matching

The main idea of gradient matching is to decompose the inference procedure into two steps:

1. Fit a curve $\hat{y}(t)$ to the data $u(t), t \in \mathbb{T}_\mathsf{D}$.
2. Estimate the parameter θ by minimising the deviation from the differential equation: $\dot{\hat{y}}(t) - f_\theta(t, \hat{y}(t))$.

This procedure is vaguely formulated, purposely so. Indeed, different alternatives for these steps have surfaced throughout the years.

Spline smoothing The first approach was to implement the curve fitting step with splines (Varah, 1982) or kernel regression (Gugushvili & Klaassen, 2012), whereafter the gradient matching step is posed as a non-linear least squares problem. Another variant is to couple the curve fitting step with the gradient matching step, resulting both in higher accuracy and higher computational cost (Ramsay et al., 2007).

Gaussian process regression The effort to formulate gradient matching probabilistically was spear-headed by Calderhead et al. (2009), where Gaussian process regresion is combined with a product of experts approach. This method was improved upon by Dondelinger et al. (2013) via joint sampling for GP and ODE parameters. It was subsequently shown that a mean-field formulation can offer computational speed-ups (Gorbach et al., 2017).

In search for a generative model There has been effort put to formulating Gaussian process-based gradient matching as inference in a generative model. First by Barber & Wang (2014), who instead formulate a model directly linking state derivatives to measurements. However, their approach suffers from identifiability problems, as demonstrated by Macdonald et al. (2015). It was later demonstrated by Wenk et al. (2019) that identifiability issues are also present for the product of experts approach. They propose to resolve this issue by formulating an alternative model; this approach was pursued further by Wenk et al. (2020).

4.3. Probabilistic Numerics

Relation to gradient matching It might be tempting to interpret the probabilistic numerics approach as a variant of gradient matching. But gradient matching fits a curve to the data and then the differential operator to the curve, while for probabilistic numerics the order of operation is reversed:

1. Fit a curve by attempting to satisfy the differential equation at a finite set of points.
2. Fit the parameters of the differential operator by using the aforementioned curve and the data likelihood.

The first step is implemented by probabilistic numerics, resulting in a *physics-enhanced* Gaussian process prior, whereas the second step reduces to Gauss–Markov regression. By directly incorporating the physics of the problem into the prior, it is ensured that inference is done in a well-posed probability model. Consequently, issues regarding model specification and identifiability (Macdonald et al., 2015; Wenk et al., 2019), that have been recurring in gradient matching, are avoided.

Relation to numerical integration The difference between probabilistic numerics and numerical integration for computing the likelihood comes down to the Dirac approximation $\widehat{\delta}_N$. As can be seen in (16), numerical integration does so by simply approximating the locations of the Dirac. On the other hand, probabilistic numerics approximates the Dirac with a distribution of non-zero width, often Gaussian in practice. This has a smoothing effect on the likelihood and parallells can be drawn with the smoothing method in non-convex optimisation (Mobahi & Ma, 2012). But the present method is not equivalent. For example, the smoothing is not with respect to the variable of interest θ, but rather with respect to the function φ_θ.

Previous probabilistic numerics approaches The probabilistic numerics approach to approximate the marginal likelihood has been explored to some extent by Kersting et al. (2020a). However, the present approach confers certain advantages over the former, the most notable being that the Gauss–Markov representation of the probabilistic solvers ensures all computations cost at most $\mathrm{O}(N)$.

A probabilistic numerics approach has also been developed for estimating time varying parameters in the context of latent force modelling (Schmidt et al., 2021). However, for the constant parameter problem, using linearised models can cause divergence in certain situations (Ljung, 1979).

An alternative to the inference-based methods hitherto discussed is to model the error by stochastic perturbation of numerical integrators (Chkrebtii et al., 2016; Conrad et al., 2017; Matsuda & Miyatake, 2021; Teymur et al., 2018).

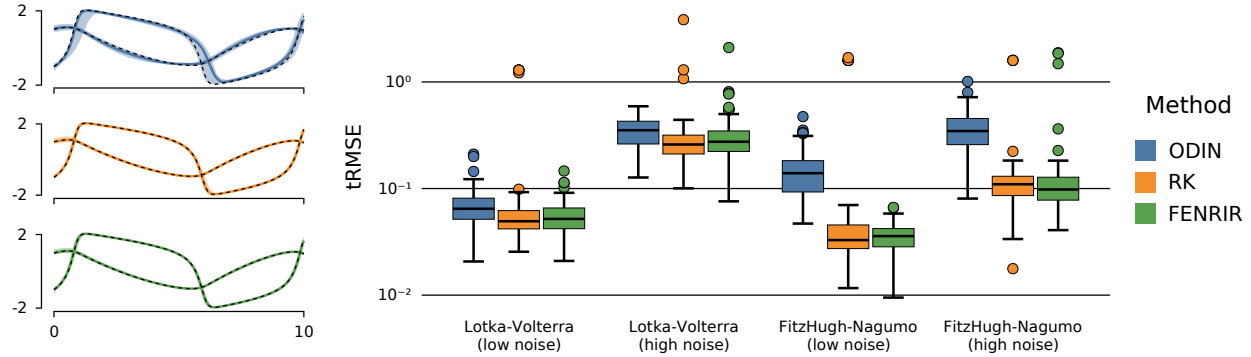


Figure 2. **Benchmarking estimation accuracy.** *Left:* Trajectory summaries of 100 experiments, obtained by numerically integrating the inferred parameters of the FitzHugh–Nagumo system from noisy observations with high noise. The solid lines show the median trajectory, the shaded areas visualize the 10% and 90% quantiles, and the black dashed line shows the ground truth. *Right:* Trajectory RMSEs (tRMSEs) on four benchmarks problems. Fenrir demonstrates performance that is competitive to ODIN and RK.

5. Experimental Results

This section investigates the utility and performance of Fenrir in a range of numerical experiments. It is structured as follows. Section 5.1 evaluates Fenrir on two standard benchmark problems. Section 5.2 demonstrates the utility of the proposed marginal likelihood for model selection. Section 5.3 considers systems with only partially observable states and shows that Fenrir, unlike most gradient matching methods, is still applicable. Finally, Section 5.4 investigates highly oscillatory systems which present a particular challenge for numerical integration-based methods.

Implementation The implementation of the probabilistic numerical IVP solvers follows a number of practices for numerically stable implementation established by Krämer & Hennig (2020). All experiments are implemented in the Julia programming language (Bezanson et al., 2017). Runge–Kutta reference solutions are computed with DifferentialEquations.jl (Rackauckas & Nie, 2017), and numerical optimizers are provided by Optim.jl (Mogensen & Riseth, 2018). All experiments run on a single, consumer-level CPU. Code is publicly available on GitHub.[2]

5.1. Parameter Inference from Fully Observed States

This experiment evaluates Fenrir on two benchmark problems that have been extensively studied in the both the gradient matching and the numerical integration literature (Calderhead et al., 2009; Wenk et al., 2020), namely the Lotka–Volterra predator-prey model and the FitzHugh–Nagumo neuronal model. Detailed system descriptions, along with the ground-truth parameters, initial values, and the chosen observation noise levels, are provided in Appendix C.2. We perform 100 experiments for each experimental setup, in which noisy observations are drawn from the numerically computed, true system trajectories. The inference task then consists in estimating initial values and parameters from noisy state observations. The quality of the resulting parameter estimates is evaluated using the trajectory RMSE (tRMSE) metric as defined in Definition C.1.

We compare Fenrir to the probabilistic gradient matching method ODIN (Wenk et al., 2020) and to a non-linear least squares regression using a Runge–Kutta solver, referred to as RK (Bard, 1974). ODIN results are computed using the code published by Wenk et al. (2020); RK is described in more detail in Appendix C.1. All methods optimise their respective objectives with the L-BFGS algorithm (Nocedal & Wright, 2006). More details are provided in Appendix C.2.

Results of the experiment are shown in Figure 2. In the median, Fenrir performs on par with ODIN and RK on Lotka–Volterra, but both RK and Fenrir outperform ODIN on FitzHugh–Nagumo and achieve more accurate state estimates as well as lower trajectory RMSEs. Both RK and Fenrir suffer from outliers, but this issue appears to be less severe for Fenrir; see also Figure 9 in Appendix C.2.

5.2. Model Selection

For a given set of noisy observations, the true parametric form of the underlying system is often not known exactly. Instead, a set of plausible models has to be evaluated against the observed data in order to find the most fitting candidate. It has been previously shown that probabilistic gradient matching can be used for model selection, by comparing estimated noise parameters which are supposed to account for model mismatch (Wenk et al., 2020). However, as Fenrir operates on a physics-informed probability model, model selection can be accomplished by statistically rigorous methods such as likelihood ratio testing (King, 1998).

[2] https://github.com/nathanaelbosch/fenrir-experiments

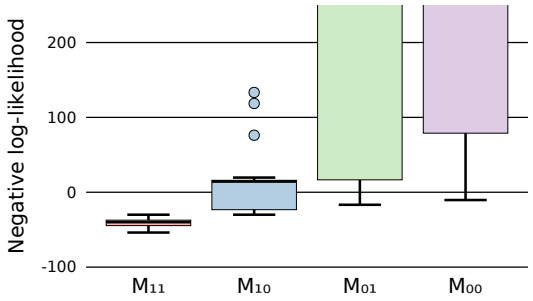


Figure 3. **Model selection results.** Fenrir correctly attributes the lowest negative log-likelihood (i.e. the highest probability) to the true M_{11} model. The figure is restricted to y-values up to 250 to show a clearer visualization, since the results vary largely in scale.

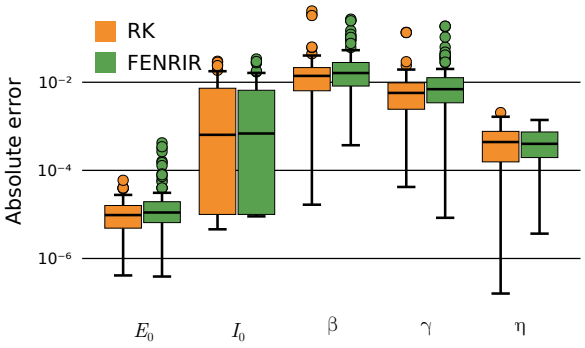


Figure 5. **Absolute parameter errors in the SEIR experiment.** Fenrir performs on par with the non-probabilistic Runge–Kutta baseline (RK) and is able to infer accurate parameter estimates from only partial observations of the SEIR system.

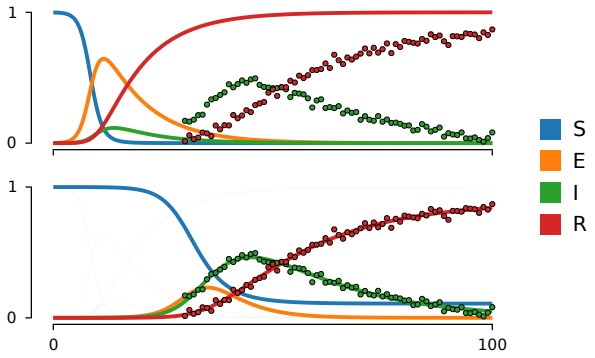


Figure 4. **Parameter inference in a SEIR model.** *Top:* Trajectory resulting from the initial, randomly chosen parameters and initial values. *Bottom:* Fenrir posterior after parameter optimization.

The experiment follows the setup proposed by Wenk et al. (2020). We consider the Lotka–Volterra system as ground truth from which we numerically simulate experimental data, and generate a set of four candidate models by combining the true ODEs with two additional, incorrect equations – all equations and parameters are provided in Appendix C.3. We obtain four models, $\{M_{11}, M_{10}, M_{01}, M_{00}\}$, where M_{11} corresponds to the true Lotka–Volterra dynamics, M_{10} and M_{01} contain one correct and one wrong equation, and M_{00} contains only incorrect equations. Thus, to succeed in this experiment, Fenrir should identify the correct model M_{11}.

We perform 100 individual model selection experiments to evaluate Fenrir's robustness regarding the observation noise. The resulting marginal likelihoods are shown in Figure 3. We observe that Fenrir consistently attributes the lowest negative log-likelihood to the correct model M_{11}, and is thus able to accurately identify the true model.

5.3. Partially Observed System States

Here, we evaluate Fenrir on an epidemeological model in which the system state can only be partially observed. We consider a compartmental SEIR model that describes the fractions of a population that are susceptible (S), exposed (E), infected (I; i.e. diagnosed with a positive test), and recovered (R) over time (Hethcote, 2000). Such compartmental models are commonly used to model the development of infectious diseases, and variants of the SEIR model have been used to explain COVID-19 outbreaks (Menda et al., 2021). The definition of the dynamics, ground-truth initial values, and parameters are provided in Appendix C.4.

At each point in time, only the infected and recovered population can be (approximately) observed, but the exposed and susceptible population is unknown. Since Fenrir's "dynamics-first" approach only requires the observation to be linearly dependent on the system states (see Equations (2) and (15c)), no particular adjustments are needed for this experiment. Similarly, the Runge–Kutta-based approach considered in Section 5.1 is also applicable and will be used for comparison. However, most gradient matching methods require all dimensions of the system states to be measurable in order to construct an interpolant, and are therefore not applicable to problems with partial observability.

Figure 4 visualizes an individual experiment: The initial values, parameters, and true system trajectories have to be estimated from noisy case counts of the infected and recovered population, which are furthermore given only from day 30 onwards. The results of 100 experiments are shown in Figure 5. Fenrir is able to consistently infer accurate parameter and trajectory estimates from noisy, partial observations of the dynamical system.

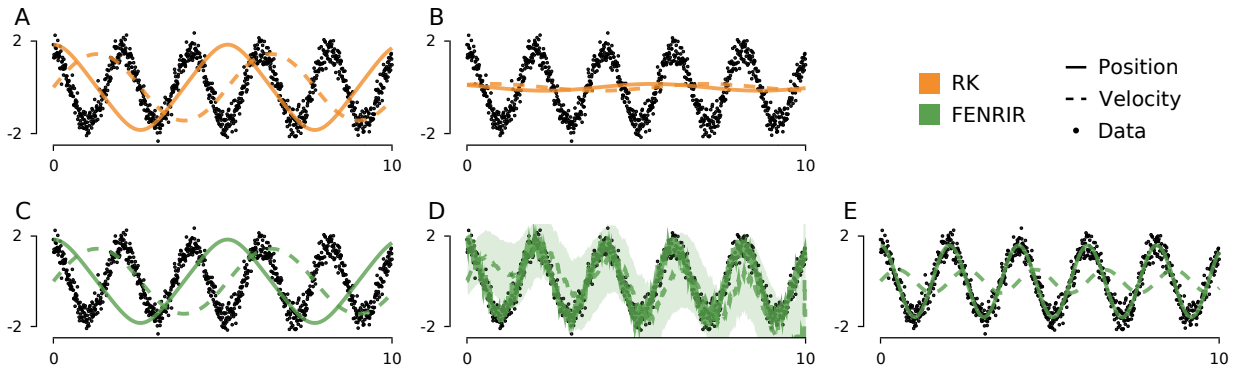


Figure 6. **Parameter inference in oscillatory systems.** Both RK and Fenrir start with an initial guess $L_0 = 5.0$ for the pendulum length parameter [A,C]. After optimization, the Runge–Kutta least-squares method RK ends up in a local minimum and is not able to recover the true parameter $L^* = 1.0$ [B]. On the other hand, Fenrir first increases its diffusion hyperparameter to interpolate the data [D] (c.f. Figure 7 below), and is then able to accurately recover the system parameter via optimization and provides accurate trajectory estimates [E].

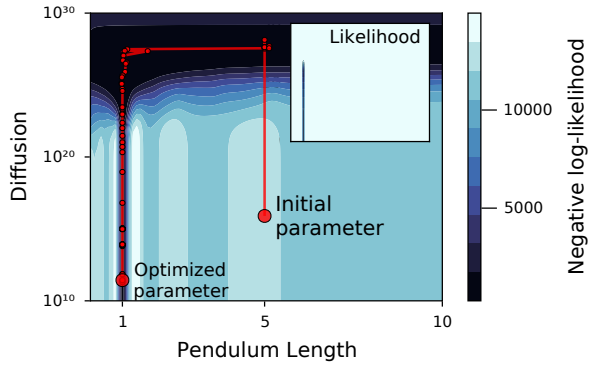


Figure 7. **Negative log-likelihood and optimization trajectory.** By first increasing its diffusion parameter, Fenrir is able to recover the true pendulum length parameter $L = 1$ by minimising the negative log-likelihood using L-BFGS. The likelihood (i.e. the negative exponential of the main plot) is shown in the inset figure.

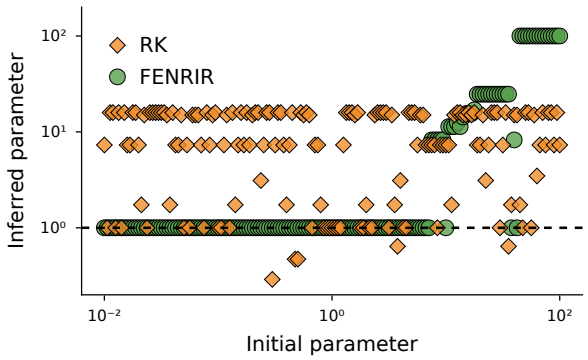


Figure 8. **Inferred parameters for various starting values.** Both RK and Fenrir are evaluated on a wide range of initial parameter estimates, from which they attempt to recover the true parameter $L = 1$ (dashed line) by optimization via L-BFGS. RK is often unable to approximate the true parameter, whereas Fenrir accurately recovers the true parameter for a wide range of starting points.

5.4. Dynamical Systems with Fast Oscillations

Finally, we evaluate Fenrir on a partially observable pendulum system that exhibits fast oscillations. Problems of this form are known to be challenging for simulation-based methods such as the previously considered Runge–Kutta least-squares approach which, with poor initialization, often fail to capture the high frequencies (Benson, 1979). While gradient-matching methods are expected to be more robust to such problems, they require fully observable states and are therefore not applicable in the present setting. Thus, we investigate Fenrir's capabilities of performing trajectory, parameter, and initial value inference under these challenges.

Figure 6 visualizes the problem setup and a single experiment; a detailed description of the dynamics and the chosen hyperparameters is provided in Appendix C.5. In the shown example, the non-linear least squares regression converges towards the constant zero function and is unable to capture the high frequencies of the data. On the other hand, by first optimizing the diffusion and observation noise parameters separately, Fenrir interpolates the experimental data and is then able to accurately approximate the true system parameters. The chosen optimization trajectory is visualised with the corresponding loss landscape in Figure 7. Figure 8 shows inferred parameters for a wider range of starting values; for simplicity, the initial value y_0 is assumed to be known here. RK often fails to converge towards the ground-truth, whereas Fenrir is able to recover the true parameter for a wide range of starting values.

6. Conclusion

It has been demonstrated that the solution of an initial value problem can be approximated by a Gauss–Markov process, reducing the inference problem to Gauss–Markov regression. The method offers advantages such as $\mathrm{O}(N)$ cost for inference, operability in the face of partial observations, regularised likelihoods, and moderate improvements in terms of estimation accuracy. But, perhaps more importantly, it has been shown that probabilistic numerics is a promising method for rigorously incorporating physics in Gaussian process regression.

Acknowledgements

The authors gratefully acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) through Project ADIMEM (FKZ 01IS18052B), and financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence Machine Learning - New Perspectives for Science, EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tubingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting N. Bosch.

References

Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

Alahmadi, A. A., Flegg, J. A., Cochrane, D. G., Drovandi, C. C., and Keith, J. M. A comparison of approximate versus exact techniques for Bayesian parameter inference in nonlinear ordinary differential equation models. *Royal Society open science*, 7(3), 2020.

Arnol'd, V. I. *Ordinary Differential Equations*. Springer-Verlag Berlin Heidelberg, 1992.

Åström, K. J. and Eykhoff, P. System identification – a survey. *Automatica*, 7(2):123–162, 1971.

Barber, D. and Wang, Y. Gaussian processes for Bayesian estimation in ordinary differential equations. In *International Conference on Machine Learning*, pp. 1485–1493. PMLR, 2014.

Bard, Y. *Nonlinear parameter estimation*. Academic Press, 1974.

Bell, B. M. The iterated Kalman smoother as a Gauss–Newton method. *SIAM Journal on Optimization*, 4(3): 626–636, 1994.

Benson, M. Parameter fitting in dynamic models. *Ecological Modelling*, 6(2):97–115, 1979.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 9 2017. doi: 10.1137/141000671.

Bi, Q., Wu, Y., Mei, S., Ye, C., Zou, X., Zhang, Z., Liu, X., Wei, L., Truelove, S. A., Zhang, T., Gao, W., Cheng, C., Tang, X., Wu, X., Wu, Y., Sun, B., Huang, S., Sun, Y., Zhang, J., Ma, T., Lessler, J., and Feng, T. Epidemiology and transmission of COVID-19 in 391 cases and 1286 of their close contacts in Shenzhen, China: a retrospective cohort study. *The Lancet. Infectious diseases*, 20(8):911–919, Aug 2020.

Biegler, L. T., Damiano, J. J., and Blau, G. E. Nonlinear parameter estimation: a case study comparison. *AIChE Journal*, 32(1):29–45, 1986.

Bosch, N., Hennig, P., and Tronarp, F. Calibrated adaptive probabilistic ODE solvers. In *International Conference on Artificial Intelligence and Statistics*, pp. 3466–3474. PMLR, 2021.

Bosch, N., Tronarp, F., and Hennig, P. Pick-and-mix information operators for probabilistic ODE solvers. In *International Conference on Artificial Intelligence and Statistics*, pp. 10015–10027. PMLR, 2022.

Calderhead, B., Girolami, M., and Lawrence, N. D. Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 217–224, 2009.

Cao, J., Wang, L., and Xu, J. Robust estimation for ordinary differential equation models. *Biometrics*, 67(4):1305–1313, 2011.

Chkrebtii, O. A., Campbell, D. A., Calderhead, B., and Girolami, M. A. Bayesian solution uncertainty quantification for differential equations. *Bayesian Analysis*, 11 (4):1239–1267, 12 2016.

Cockayne, J., Oates, C. J., Sullivan, T. J., and Girolami, M. Bayesian probabilistic numerical methods. *SIAM Review*, 61(4):756–789, 2019.

Conrad, P. R., Girolami, M., Särkkä, S., Stuart, A., and Zygalakis, K. Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Statistics and Computing*, 27(4):1065–1082, Jul 2017.

Dass, S. C., Lee, J., Lee, K., and Park, J. Laplace based approximate posterior inference for differential equation models. *Statistics and Computing*, 27(3):679–698, 2017.

Dondelinger, F., Husmeier, D., Rogers, S., and Filippone, M. ODE parameter inference using adaptive gradient matching with Gaussian processes. In *Artificial intelligence and Statistics*, pp. 216–228, 2013.

FitzHugh, R. Mathematical models of threshold phenomena in the nerve membrane. *The bulletin of mathematical biophysics*, 17(4):257–278, 1955.

Friston, K. J. Bayesian estimation of dynamical systems: an application to fMRI. *NeuroImage*, 16(2):513–530, 2002.

Gelman, A., Bois, F., and Jiang, J. Physiological pharmacokinetic analysis using population modeling and informative prior distributions. *Journal of the American Statistical Association*, 91(436):1400–1412, 1996.

Gorbach, N. S., Bauer, S., and Buhmann, J. M. Scalable variational inference for dynamical systems. In *Advances in Neural Information Processing Systems*, pp. 4806–4815, 2017.

Gugushvili, S. and Klaassen, C. A. J. $\sqrt{n}$-consistent parameter estimation for systems of ordinary differential equations: bypassing numerical integration via smoothing. *Bernoulli*, 18(3):1061–1098, 2012.

Hairer, E. and Wanner, G. Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics*, 111, 1999.

Hairer, E., Nørsett, S., and Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 1987.

Hennig, P., Osborne, M. A., and Girolami, M. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179):20150142, 2015.

Hethcote, H. W. The mathematics of infectious diseases. *SIAM review*, 42(4):599–653, 2000.

Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1): 35–45, 1960.

Kersting, H., Krämer, N., Schiegg, M., Daniel, C., Tiemann, M., and Hennig, P. Differentiable likelihoods for fast inversion of likelihood-free dynamical systems. In *International Conference on Machine Learning*, pp. 5198–5208. PMLR, 2020a.

Kersting, H., Sullivan, T. J., and Hennig, P. Convergence rates of Gaussian ODE filters. *Statistics and computing*, 30(6):1791–1816, 2020b.

King, G. *Unifying political methodology: The likelihood theory of statistical inference*. University of Michigan Press, 1998.

Krämer, N. and Hennig, P. Stable implementation of probabilistic ODE solvers. *arXiv preprint arXiv:2012.10106*, 2020.

Krämer, N., Bosch, N., Schmidt, J., and Hennig, P. Probabilistic ODE solutions in millions of dimensions. *arXiv preprint arXiv:2110.11812*, 2021.

Lauer, S. A., Grantz, K. H., Bi, Q., Jones, F. K., Zheng, Q., Meredith, H. R., Azman, A. S., Reich, N. G., and Lessler, J. The incubation period of coronavirus disease 2019 (COVID-19) from publicly reported confirmed cases: estimation and application. *Annals of internal medicine*, 172(9):577–582, 2020.

Ljung, L. Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems. *IEEE Transactions on Automatic Control*, 24(1):36–50, 1979.

Lotka, A. *Elements of Physical Biology*. Williams & Wilkins, 1925.

Macdonald, B., Higham, C., and Husmeier, D. Controversy in mechanistic modelling with Gaussian processes. *Proceedings of Machine Learning Research*, 37:1539–1547, 2015.

Magnani, E., Kersting, H., Schober, M., and Hennig, P. Bayesian Filtering for ODEs with Bounded Derivatives. *arXiv:1709.08471 [cs.NA]*, September 2017.

Matsuda, T. and Miyatake, Y. Estimation of ordinary differential equation models with discretization error quantification. *SIAM/ASA Journal on Uncertainty Quantification*, 9(1):302–331, 2021.

Menda, K., Laird, L., Kochenderfer, M. J., and Caceres, R. S. Explaining COVID-19 outbreaks with reactive SEIRD models. *Scientific Reports*, 11(1):17905, Sep 2021.

Mobahi, H. and Ma, Y. Gaussian smoothing and asymptotic convexity. *Coordinated Science Laboratory Report no. UILU-ENG-12-2201, DC-254*, 2012.

Mogensen, P. K. and Riseth, A. N. Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24):615, 2018. doi: 10.21105/joss.00615.

Nagumo, J., Arimoto, S., and Yoshizawa, S. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, New York, NY, USA, 2e edition, 2006.

Oates, C. J. and Sullivan, T. J. A modern retrospective on probabilistic numerics. *Statistics and Computing*, 29(6): 1335–1351, 2019.

Øksendal, B. *Stochastic Differential Equations - An Introduction with Applications*. Springer, 2003.

Rackauckas, C. and Nie, Q. DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

Ramsay, J. O., Hooker, G., Campbell, D., and Cao, J. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

Rauch, H. E., Tung, F., and Striebel, C. T. Maximum likelihood estimates of linear dynamic system. *AIAA Journal*, 3(8):1445–1450, Aug 1965.

Särkkä, S. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

Särkkä, S. and Solin, A. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

Schmidt, J., Krämer, N., and Hennig, P. A probabilistic state space model for joint inference from differential equations and data. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021.

Schober, M., Särkkä, S., and Hennig, P. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, January 2019.

Schweppe, F. Evaluation of likelihood functions for Gaussian signals. *IEEE Transactions on Information Theory*, 11(1):61–70, 1965.

Stoica, P. and Selen, Y. Model-order selection: a review of information criterion rules. *IEEE Signal Processing Magazine*, 21(4):36–47, 2004.

Teymur, O., Lie, H. C., Sullivan, T., and Calderhead, B. Implicit probabilistic integrators for ODEs. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

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

Tronarp, F., Karvonen, T., and Särkkä, S. Student's t-filters for noise scale estimation. *IEEE Signal Processing Letters*, 26(2):352–356, 2019a.

Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29(6):1297–1315, 2019b.

Tronarp, F., Särkkä, S., and Hennig, P. Bayesian ODE solvers: The maximum a posteriori estimate. *Statistics and Computing*, 31(3):1–18, 2021.

Tsitouras, C. Runge–Kutta pairs of order 5 (4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications*, 62, 2011.

Varah, J. M. A spline least squares method for numerical parameter estimation in differential equations. *SIAM Journal on Scientific and Statistical Computing*, 3(1):28–46, 1982.

Voit, E. O. *Computational Analysis of Biochemical Systems: A Practical Guide for Biochemists and Molecular Biologists*. Cambridge University Press, 2000.

Volterra, V. Variations and Fluctuations of the Number of Individuals in Animal Species living together. *ICES Journal of Marine Science*, 3(1):3–51, 1928.

Wenk, P., Gotovos, A., Bauer, S., Gorbach, N. S., Krause, A., and Buhmann, J. M. Fast Gaussian process based gradient matching for parameter identification in systems of nonlinear ODEs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1351–1360. PMLR, 2019.

Wenk, P., Abbati, G., Osborne, M. A., Schölkopf, B., Krause, A., and Bauer, S. ODIN: ODE-informed regression for parameter and state inference in time-continuous dynamical systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 6364–6371, 2020.

A. Additional Details on Probabilistic Numerics

In this appendix, the probabilistic solver is described in detail. Further details on the prior are given in Appendix A.1. In Appendix A.2, it is explained how to compute the marginal moments and the parameters of the backward Markov representation of the posterior when the vector field is linear (affine). In Appendix A.3 some linearisation methods for approximate inference when the vector field is non-linear are reviewed.

A.1. More details on priors

Recall that the prior in state-space form is given by

$$\mathrm{d}x(t) = Ax(t)\,\mathrm{d}t + \sqrt{\kappa}B\,\mathrm{d}w(t), \quad x(0) = x_\theta^\dagger, \tag{17a}$$

$$y^{(m)}(t) = \mathrm{E}_m^\mathsf{T} x(t), \quad m = 0, 1, \ldots, \nu, \tag{17b}$$

where $y^{(m)}$ models the mth derivative of the solution. By Itô's formula this implies that

$$\mathrm{d}\mathrm{E}_m^\mathsf{T} x(t) = \mathrm{E}_m^\mathsf{T} A x(t)\,\mathrm{d}t + \sqrt{\kappa}\mathrm{E}_m^\mathsf{T} B\,\mathrm{d}w(t), \tag{18}$$

and for this to be consistent with the asserted derivative relations it must hold that

$$\mathrm{E}_m^\mathsf{T} A x(t)\,\mathrm{d}t + \sqrt{\kappa}\mathrm{E}_m^\mathsf{T} B\,\mathrm{d}w(t) = \mathrm{E}_{m+1}^\mathsf{T} x(t)\,\mathrm{d}t, \quad m = 0, 1, \ldots, \nu - 1. \tag{19}$$

This in turn implies that it must hold that

$$\mathrm{E}_m^\mathsf{T} A = \mathrm{E}_{m+1}^\mathsf{T}, \quad m = 0, 1, \ldots, \nu - 1, \tag{20a}$$

$$\mathrm{E}_m^\mathsf{T} B = 0, \quad m = 0, 1, \ldots, \nu - 1, \tag{20b}$$

while $\mathrm{E}_\nu^\mathsf{T} A$ and $\mathrm{E}_\nu^\mathsf{T} B$ are free parameters. Letting e_m be the mth canonical basis vector in $\mathbb{R}^{\nu+1}$, I_d be the d by d identity matrix, and fixing $\mathrm{E}_m = \mathrm{e}_m \otimes \mathrm{I}_d$ then gives the model

$$\mathrm{d}y^{(\nu)}(t) = \sum_{m=0}^{\nu} A_{\nu,m} y^{(m)}\,\mathrm{d}t + \sqrt{\kappa}B_\nu\,\mathrm{d}w(t), \tag{21}$$

where $A_{\nu,m} = \mathrm{E}_\nu^\mathsf{T} A \mathrm{E}_m$ and $B_\nu = \mathrm{E}_\nu^\mathsf{T} B$. Any other state-space model of dimension $d(\nu+1)$ modelling a vector valued function of dimension d and its ν first derivatives must be related to this via similarity transform. The canonical model in probabilistic numerics is the ν-times integrated Wiener process (Schober et al., 2019; Tronarp et al., 2019b; Krämer & Hennig, 2020; Bosch et al., 2021; Kersting et al., 2020b), where the parameters are given by

$$A_{\nu,m} = 0, \quad m = 0, 1, \ldots, \nu - 1. \tag{22}$$

Though other priors are of course possible (Magnani et al., 2017; Tronarp et al., 2021; Kersting et al., 2020b). Usually, the diffusion matrix B_ν is set to identity as well, yielding the following prior

$$\mathrm{d}y^{(\nu)}(t) = \sqrt{\kappa}\,\mathrm{d}w(t), \tag{23}$$

which is used throughout the article.

A.2. Posterior for linear vector fields

Suppose the vector field is linear:

$$f_\theta(t, y) = L_\theta(t) y + b_\theta(t), \tag{24}$$

then the probabilistic IVP solver reduces to inference in the following model:

$$\mathrm{d}x(t) = Ax(t)\,\mathrm{d}t + \sqrt{\kappa}B\,\mathrm{d}w(t), \quad x(0) = x_\theta^\dagger, \tag{25}$$

subject to the data

$$C_\theta(t) = \mathrm{E}_1 - \mathrm{E}_0 L_\theta^\mathsf{T}(t), \tag{26a}$$

$$z(t) = 0 = \mathrm{E}_1^\mathsf{T} x(t) - L_\theta(t)\mathrm{E}_0^\mathsf{T} x(t) - b_\theta(t), \quad t \in \mathbb{T}_{\mathsf{PN}}. \tag{26b}$$

The posterior is Gaussian because the prior is Gaussian and the measurement functionals are linear, and it can be computed with the well-known forward / backward recursions (Kalman, 1960; Rauch et al., 1965).

More specifically, denote the numerics data up to time t by

$$\mathscr{Z}_{[0,t]} = \big\{z(t) = 0 \colon t \in \mathbb{T}_{\mathsf{PN}} \cap [0, t]\big\} \tag{27}$$

and up to *just before* time t by

$$\mathscr{Z}_{[0,t)} = \big\{z(t) = 0 \colon t \in \mathbb{T}_{\mathsf{PN}} \cap [0, t)\big\}. \tag{28}$$

The forward recursion then computes the filtering densities

$$p(t, x \mid \mathscr{Z}_{[0,t]}) = \mathcal{N}(x; \mu_\theta(t), \kappa\Sigma_\theta(t)), \tag{29}$$

which agree with the prediction densities

$$p(t, x \mid \mathscr{Z}_{[0,t)}) = \mathcal{N}(x; \mu_\theta(t^-), \kappa\Sigma_\theta(t^-)), \tag{30}$$

unless $t \in \mathbb{T}_{\mathsf{PN}}$. The filtering moments are then post-processed in the backwards recursion to produce the smoothing densities (time marginals of the posterior)

$$p(t, x \mid \mathscr{Z}(t_N)) = \mathcal{N}(x; \xi_\theta(t), \kappa\Lambda_\theta(t)). \tag{31}$$

For a more thorough exposition on filtering and smoothing refer to Särkkä (2013); Särkkä & Solin (2019). Furthermore, the fact that the scaling κ is retained throughout the recursion follows from the fact that the initial covariance and all transition covariances are scaled by κ (Tronarp et al., 2019a).

Forward recursion The forward recursion starts by initialising the filter mean and covariance according to

$$\mu_\theta(t_0) = x_\theta^\dagger, \tag{32a}$$
$$\Sigma_\theta(t_0) = 0, \tag{32b}$$

whereafter the algorithm alternates between prediction and update. The prediction equations are given by

$$\mu_\theta(t_n^-) = \Phi(\Delta_n)\mu_\theta(t_{n-1}), \tag{33a}$$
$$\Sigma_\theta(t_n^-) = \Phi(\Delta_n)\Sigma_\theta(t_{n-1})\Phi^\mathsf{T}(\Delta_n) + Q(\Delta_n), \tag{33b}$$
$$G_\theta(t_{n-1}) = \Sigma_\theta(t_{n-1})\Phi^\mathsf{T}(\Delta_n)\Sigma_\theta^{-1}(t_n^-), \tag{33c}$$
$$P_\theta(t_{n-1}) = \Sigma_\theta(t_{n-1}) - G_\theta(t_{n-1})\Sigma_\theta(t_n^-)G_\theta^\mathsf{T}(t_{n-1}), \tag{33d}$$

where G_θ and P_θ are parameters associated with the subsequent backward recursion. The update relations are given by

$$C_\theta(t_n) = \mathrm{E}_1 - \mathrm{E}_0 L_\theta^\mathsf{T}(t_n), \tag{34a}$$
$$S_\theta(t_n) = C_\theta^\mathsf{T}(t_n)\Sigma_\theta(t_n^-)C_\theta(t_n), \tag{34b}$$
$$K_\theta(t_n) = \Sigma_\theta(t_n^-)C_\theta^\mathsf{T}(t_n)S_\theta^{-1}(t_n), \tag{34c}$$
$$\mu_\theta(t_n) = \mu_\theta(t_n^-) + K_\theta(t_n)\big(b_\theta(t_n) - C_\theta^\mathsf{T}(t_n)\mu_\theta(t_n^-)\big), \tag{34d}$$
$$\Sigma_\theta(t_n) = \Sigma_\theta(t_n^-) - K_\theta(t_n)S_\theta(t_n)K_\theta^\mathsf{T}(t_n). \tag{34e}$$

Backward recursion The backwards recursion starts by setting the smoother mean and covariance to the filter mean and covariance at the terminal point according to

$$\xi_\theta(t_N) = \mu_\theta(t_N), \tag{35a}$$
$$\Lambda_\theta(t_N) = \Sigma_\theta(t_N). \tag{35b}$$

The backwards recursion is then given by

$$\xi_\theta(t_n) = \mu_\theta(t_n) + G_\theta(t_n)\big(\xi_\theta(t_{n+1}) - \Phi(\Delta_{n+1})\mu_\theta(t_n)\big), \tag{36a}$$
$$\Lambda_\theta(t_n) = G_\theta(t_n)\Lambda_\theta(t_{n+1})G_\theta^\mathsf{T}(t_n) + P_\theta(t_n). \tag{36b}$$

Backward Markov process representation Lastly, the posterior may be represented, on the grid, by the following backwards Markov process

$$\begin{aligned}\gamma_N(x(t_{1:N}) \mid \theta, \kappa) &= \mathcal{N}\big(x(t_N); \xi_\theta(t_N), \kappa\Lambda_\theta(t_N)\big) \\ &\prod_{n=N-1}^{1} \mathcal{N}\big(x(t_n); \mu_\theta(t_n) + G_\theta(t_n)\big(x(t_{n+1}) - \mu_\theta(t_{n+1}^-)\big), \kappa P_\theta(t_n)\big).\end{aligned} \tag{37}$$

This follows from the fact that

$$p(t, x \mid s, x', \mathscr{Z}_{[0,T]}) = p(t, x \mid s, x', \mathscr{Z}_{[0,t]}), \quad t_{n+1} \geq s > t \geq t_n,\ n = 1, \ldots, N. \tag{38}$$

That is, by total probability

$$\begin{aligned}p(t_n, x_n \mid \mathscr{Z}_{[0,T]}) &= \int p(t_n, x_n \mid t_{n+1}, x_{n+1}, \mathscr{Z}_{[0,T]}) p(t_{n+1}, x_{n+1} \mid \mathscr{Z}_{[0,T]})\, \mathrm{d}x_{n+1} \\ &= \int p(t_n, x_n \mid t_{n+1}, x_{n+1}, \mathscr{Z}_{[0,t_n]}) p(t_{n+1}, x_{n+1} \mid \mathscr{Z}_{[0,T]})\, \mathrm{d}x_{n+1},\end{aligned} \tag{39}$$

and by Bayes' rule

$$\begin{aligned}p(t_n, x_n \mid t_{n+1}, x_{n+1}, \mathscr{Z}_{[0,t_n]}) &\propto p(t_n, x_n \mid \mathscr{Z}_{0,t_n}) p(t_{n+1}, x_{n+1} \mid t_n, x_n, \mathscr{Z}_{[0,t_n]}) \\ &= \mathcal{N}(x_n; \mu_\theta(t_n), \kappa\Sigma_\theta(t_n)) \mathcal{N}(x_{n+1}; \Phi(\Delta_{n+1})x_n, \kappa Q(\Delta_{n+1})) \\ &= \mathcal{N}(x_{n+1}; \mu_\theta(t_{n+1}^-), \Sigma_\theta(t_{n+1}^-)) \mathcal{N}(x_n; \mu_\theta(t_n) + G_\theta(t_n)(x_{n+1} - \mu_\theta(t_{n+1}^-)), \kappa P_\theta(t_n)) \\ &\propto \mathcal{N}(x_n; \mu_\theta(t_n) + G_\theta(t_n)(x_{n+1} - \mu_\theta(t_{n+1}^-)), \kappa P_\theta(t_n)),\end{aligned} \tag{40}$$

where the last equality follows from ordinary Gaussian conditioning and the proportionality signs are with respect to x_n. This proves the recursive structure of the posterior as asserted by proposition 3.1, and the complete result follows from the fact that the marginal filtering and smoothing densities coincide at the terminal point. That is,

$$p(t_N, x_N, \mid \mathscr{Z}_{[0,T]}) = \mathcal{N}(x_N; \mu_\theta(t_N), \kappa\Sigma_\theta(t_N)) = \mathcal{N}(x_N; \xi_\theta(t_N), \kappa\Lambda_\theta(t_N)). \tag{41}$$

A.3. Approximate posteriors via linearisation

When the vector field is non-linear, the posterior is in most cases intractable. However, approximate posteriors may be obtained by linearising the data relation in (7). Due to the structure of the information operator, there are multiple choices for doing this, namely

1. Zeroth order linearisation (Schober et al., 2019):

$$\hat{L}_\theta(t) = 0, \tag{42a}$$
$$\hat{b}_\theta(t) = f_\theta(t, \tilde{y}(t)) \tag{42b}$$

2. First order linearisation (Tronarp et al., 2019b):

$$\hat{L}_\theta(t) = J_{f_\theta}(t, \tilde{y}(t)), \tag{43a}$$
$$\hat{b}_\theta(t) = f_\theta(t, \tilde{y}(t)) - J_{f_\theta}(t, \tilde{y}(t))\tilde{y}(t) \tag{43b}$$

The linearisation point is typicaly chosen as the predictive mean:

$$\tilde{y}(t) = \mathrm{E}_0^\mathsf{T} \mu_\theta(t^-). \tag{44}$$

However, other choices are possible as well, such as the smoothing mean (Tronarp et al., 2021)

$$\tilde{y}(t) = \mathrm{E}_0^\mathsf{T} \xi_\theta(t), \tag{45}$$

which leads to the fixed-point equations for the Gauss–Newton algorithm (Bell, 1994).

B. Inference in IVPs as Gauss–Markov regression

Using the probabilistic numerics posterior as a surrogate for the solution of the initial value problem leads to the following inference problem

$$x(t_N) \sim \mathcal{N}(\xi_\theta(t_N), \kappa\Lambda_\theta(t_N)), \tag{46a}$$
$$x(t_n) \mid x(t_{n+1}) \sim \widehat{\gamma}_N(x(t_n) \mid x(t_{n+1}) \mid \theta, \kappa), \tag{46b}$$
$$u(t) \mid x(t) \sim \mathcal{N}(H^\mathsf{T}\mathrm{E}_0^\mathsf{T} x(t_n), R_\theta), \quad t \in \mathbb{T}_\mathsf{D}. \tag{46c}$$

This is again, a problem of Gauss–Markov regresion and can be solved by the usual forward / backward recursions. What is unusual is that the latent process is specified in terms of a terminal distribution and backward transition densities. Therefore, the equations required for implementation are given in detail.

B.1. The forward (but backward in time) recursion and the marginal likelihood

The backward recursion is implemented by a forward recursion with flipped time axis. That is, start by initialising the filter moments:

$$\breve{\mu}_\theta(t_N^+) = \xi_\theta(t_N), \tag{47a}$$
$$\breve{\Sigma}_\theta(t_N^+) = \kappa\Lambda_\theta(t_N), \tag{47b}$$

whereafter the algorithm alternates between a backward prediction and update. If $t_n \in \mathbb{T}_\mathsf{D}$, then an update is performed according to

$$\breve{H} = \mathrm{E}_0 H, \tag{48a}$$
$$\breve{S}(t_n) = \breve{H}^\mathsf{T}\breve{\Sigma}_\theta(t_n)\breve{H} + R_\theta, \tag{48b}$$
$$\breve{K}_\theta(t_n) = \breve{\Sigma}_\theta(t_n)\breve{H}\breve{S}_\theta^{-1}(t_n), \tag{48c}$$
$$\breve{\mu}_\theta(t_n) = \breve{\mu}_\theta(t_n^+) + \breve{K}_\theta(t_n)\big(u(t_n) - \breve{H}^\mathsf{T}\breve{\mu}_\theta(t_n^+)\big), \tag{48d}$$
$$\breve{\Sigma}_\theta(t_n) = \breve{\Sigma}_\theta(t_n^+) - \breve{K}_\theta(t_n)\breve{S}_\theta(t_n)\breve{K}_\theta^\mathsf{T}(t_n). \tag{48e}$$

The prediction step is given by

$$\breve{\mu}_\theta(t_{n-1}^+) = \mu_\theta(t_{n-1}) + G_\theta(t_{n-1})\big(\breve{\mu}_\theta(t_{n+1}) - \mu_\theta(t_{n+1}^-)\big), \tag{49a}$$
$$\breve{\Sigma}_\theta(t_{n-1}^+) = G_\theta(t_{n-1})\breve{\Sigma}_\theta(t_n)G_\theta^\mathsf{T}(t_{n-1}) + \kappa P_\theta(t_{n-1}). \tag{49b}$$

Finally, the marginal likelihood approximation is given by the prediction error decomposition (Schweppe, 1965)

$$\widehat{\mathcal{M}}_N(\theta, \kappa) = \prod_{t \in \mathbb{T}_\mathsf{D}} \mathcal{N}\big(u(t); \breve{H}^\mathsf{T}\breve{\mu}_\theta(t^+), \breve{S}_\theta(t)\big). \tag{50}$$

B.2. The backward (but forward in time) recursion and trajectory estimates

The smoothing parameters for the forward recursion are given by

$$\breve{G}_\theta(t_n) = \breve{\Sigma}_\theta(t_n)G_\theta^\mathsf{T}(t_{n-1})\breve{\Sigma}_\theta^{-1}(t_{n-1}^+), \tag{51a}$$
$$\breve{P}_\theta(t_n) = \breve{\Sigma}_\theta(t_n) - \breve{G}_\theta(t_n)\breve{\Sigma}_\theta(t_{n-1}^+)\breve{G}_\theta^\mathsf{T}(t_n), \tag{51b}$$

and the forward smoothing recursion is given by

$$\breve{\xi}_\theta(t_n) = \breve{\mu}_\theta(t_n) + \breve{G}_\theta(t_n)\big(\breve{\xi}_\theta(t_{n-1}) - G_\theta(t_{n-1})\breve{\mu}_\theta(t_n)\big), \tag{52a}$$
$$\breve{\Lambda}_\theta(t_n) = \breve{G}_\theta(t_n)\breve{\Lambda}_\theta(t_{n-1})\breve{G}_\theta^\mathsf{T}(t_n) + \breve{P}_\theta(t_n). \tag{52b}$$

C. Additional Details on the Experimental Evaluation

In all experiments, Fenrir uses a 5-times integrated Wiener process prior and a first-order linearisation of the vector field during the probabilistic numerical ODE solve when computing its physics-enhanced prior.

Optimization Throughout all experiments, the L-BFGS method has been used for optimization with both Fenrir and RK (Nocedal & Wright, 2006); L-BFGS is also the optimizer of choice in the official ODIN code by Wenk et al. (2020). The specific L-BFGS implementation is provided by the Optim.jl software package (Mogensen & Riseth, 2018). In all experiment, the observation noise σ^2 and the diffusion κ are optimised in log-space.

Parameter Initialization As done in ODIN, ODE parameters are initialised with a folded normal distribution, i.e. as the absolute value of a sample from standard normal Gaussian, and initial values are initialised with their noisy observation $u(t_0)$, unless specified otherwise. Observation noise is always initialised as $\sigma^2 = 1$.

C.1. Baseline: Non-linear Least Squares Regression using a Runge–Kutta Solver

Given data $\mathcal{D} = \{u(t)\}$ on the grid $t \in \mathbb{T}_\mathsf{D}$, the considered "RK" baseline method minimizes the loss

$$L := \sum_{t \in \mathbb{T}_\mathsf{D}} \left\| H \cdot \hat{y}(t) - u(t) \right\|_2^2, \tag{53}$$

where $\hat{y}(t)$ is computed with a classical Runge–Kutta initial value solver and H is the measurement matrix as introduced in Equation (2). In most experiments, the `Tsit5` (Tsitouras, 2011) solver is used, with adaptive step-size selection for absolute and relative tolerances $\tau_\text{abs} = 10^{-8}$, $\tau_\text{rel} = 10^{-6}$. Only on the FitzHugh-Nagumo system we use the implicit `RadauIIA5` (Hairer & Wanner, 1999) method, since we observed it to be more robust as some parameter settings can lead to stiff dynamics. Both solvers are provided by DifferentialEquations.jl (Rackauckas & Nie, 2017).

C.2. Additional Details on Section 5.1: "Parameter Inference from Fully Observed States"

Definition C.1 (Trajectory RMSE)**.** Let $\hat{\theta}$ be the parameters estimated by an inference algorithm, and let $\mathbb{T}_\mathsf{D}$ be the set of measurement nodes. Then, let $\hat{y}(t)$, $t \in \mathbb{T}_\mathsf{D}$, be the estimated system trajectory, computed by numerically integrating the ODE with initial values and parameters as given by the estimated $\hat{\theta}$. The trajectory RMSE (tRMSE) is then defined as

$$\text{tRMSE} := \sqrt{\frac{1}{|\mathbb{T}_\mathsf{D}|} \sum_{t \in \mathbb{T}_\mathsf{D}} \left\| \hat{y}(t) - y(t) \right\|_2^2}. \tag{54}$$

Lotka–Volterra The Lotka–Volterra model describes the dynamics of biological systems in which two species interact, one as a predator and the other as prey (Lotka, 1925; Volterra, 1928). It is described by the ODEs

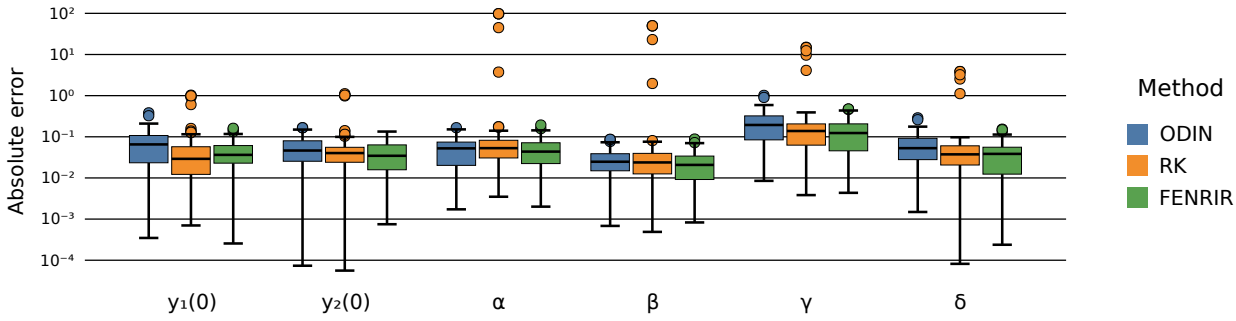
$$\dot{y_1} = \alpha y_1 - \beta y_1 y_2, \tag{55a}$$
$$\dot{y_2} = -\gamma y_1 + \delta y_1 y_2. \tag{55b}$$

As ground truth, we assume an initial value $y_0 = [5, 3]^\mathsf{T}$ and parameters $\alpha = 2$, $\beta = 1$, $\gamma = 4$, $\delta = 1$. The experimental data is generated on the equi-spaced time grid $t_i \in \mathbb{T}_\mathsf{D} = \{0.0, 0.1, \ldots, 2.0\}$, as $u(t_i) = \hat{y}(t_i) + v(t_i)$, where $\hat{y}(t_i)$ is computed via accurate, numerical simulation, and with noise $v(t) \sim \mathcal{N}(0, \sigma^2 \cdot I)$. We further consider two different noise levels $\sigma^2_\text{low} = 0.01$ and $\sigma^2_\text{high} = 0.25$. Thus, the full set of parameters to be estimated is $\theta = \{y_0, \alpha, \beta, \gamma, \delta, \sigma\}$, as well as the diffusion κ. In this system, we found it helpful to first optimize the noise and diffusion parameters σ, κ individually until convergence, and only then optimize all parameters jointly; such an approach is also chosen by the gradient matching method ODIN (Wenk et al., 2020). Furthermore, as in the original experimental setup by Wenk et al. (2020), we consider bounds $y_0 \in [0, 100]^2$, $\alpha, \beta, \gamma, \delta \in [0, 100]$, $\sigma^2 \in [10^{-6}, 10^2]$, and additionally $\kappa \in [10^{-20}, 10^{50}]$. Finally, a step-size of $\Delta = 5 \cdot 10^{-3}$ is chosen for Fenrir's probabilistic numerical integration.

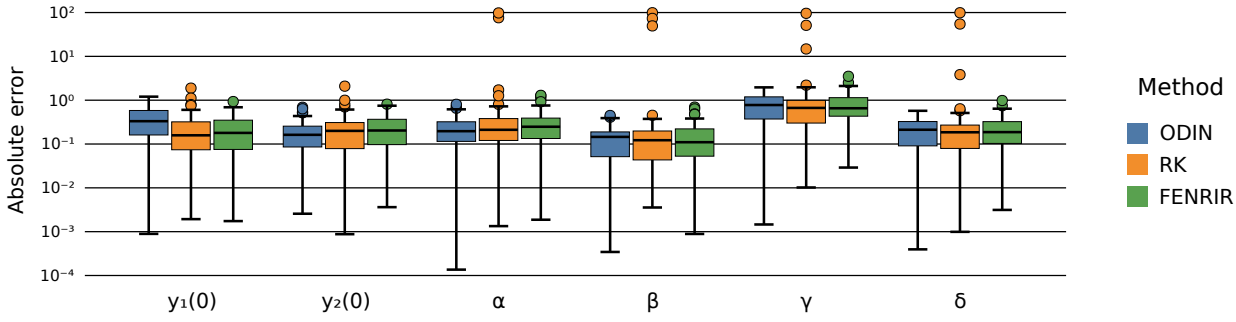
FitzHugh–Nagumo The FitzHugh–Nagumo neuronal model (FitzHugh, 1955; Nagumo et al., 1962) is given by the ODE

$$\dot{y_1} = c \left(y_1 - \frac{y_1^3}{3} + y_2 \right), \tag{56a}$$
$$\dot{y_2} = -\frac{1}{c} \left(y_1 - a - b y_2 \right). \tag{56b}$$

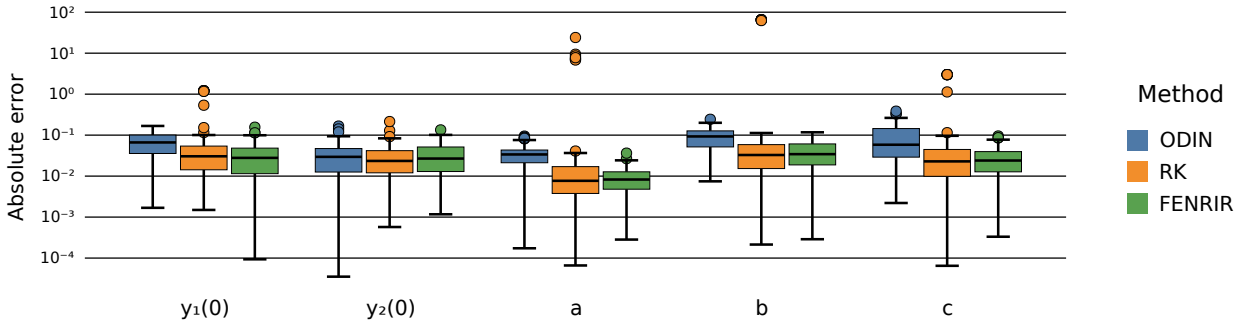
We consider ground-truth parameters $a = 0.2$, $b = 0.2$, $c = 3.0$, and a true initial value $y_0 = [-1, 1]^\mathsf{T}$. The experimental data is generated on the grid $t_i \in \mathbb{T}_\mathsf{D} = \{0.0, 0.5, \ldots, 10.0\}$, by disturbing a high-confidence numerical simulation of



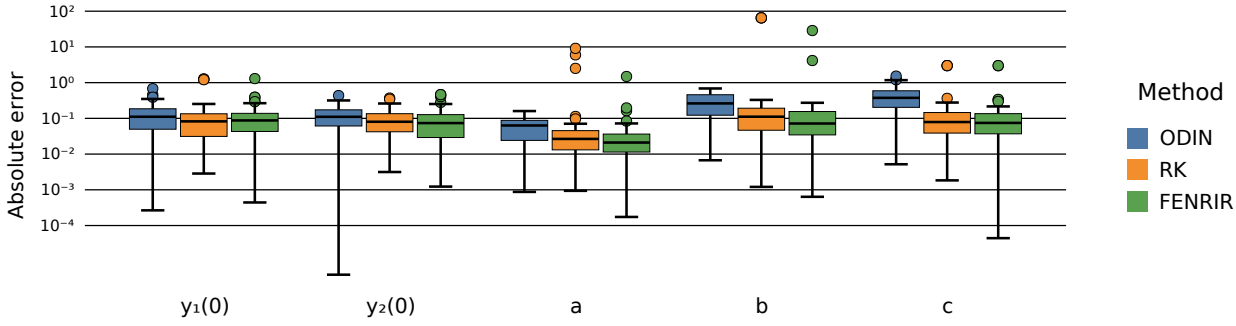
(a) Lotka–Volterra with low observation noise.



(b) Lotka–Volterra with high observation noise.



(c) FitzHugh–Nagumo with low observation noise.



(d) FitzHugh–Nagumo with high observation noise.

Figure 9. **Absolute parameter errors.**

the true trajectory with Gaussian noise $v(t) \sim \mathcal{N}(0, \sigma^2 \cdot I)$, for two noise levels $\sigma^2_{\text{low}} = 0.005$ and $\sigma^2_{\text{high}} = 0.05$. The full set of (hyper)parameters to be estimated by Fenrir is then $\theta = \{y_0, \alpha, \beta, \gamma, \delta, \sigma\}$, as well as the diffusion κ. All of which are jointly optimised via L-BFGS, while assuming bounds $y_0 \in [-100, 100]^2$, $a, b, c \in [0, 100]$, $\sigma^2 \in [10^{-6}, 10^2]$, and $\kappa \in [10^{-20}, 10^{50}]$. Fenrir's physics-enhanced prior is computed with a step size $\Delta = 10^{-2}$.

C.3. Additional Details on Section 5.2: "Model Selection"

The Lotka–Volterra model with ground-truth parameters as described in Appendix C.2 is extended to a set of four candidate models, via the following additional ODEs:

$$\dot{y_1} = \alpha y_1^2 - \beta y_2, \tag{57a}$$

$$\dot{y_2} = -\gamma y_2. \tag{57b}$$

By combining these two wrong equations with the true ODEs, we obtain for models M_{ij}, with $i, j \in \{0, 1\}$ indicating if the correct (1) or incorrect equation (0) has been used; for instance, M_{01} contains Equation (57a) and Equation (55b). The experimental data is generated as described in Appendix C.2, with a "low" noise setting of $\sigma^2_{\text{low}} = 0.01$. All parameters are optimised jointly by Fenrir via L-BFGS, with bounds for parameters and initial values chosen as in Appendix C.2.

C.4. Additional Details on Section 5.3: "Partially Observed System States"

The compartmental SEIR model (Hethcote, 2000) describes the fractions of a population that are susceptible (S), exposed (E), infected (I; i.e. diagnosed by a positive test), and recovered (R). It is given in as differential equations

$$\dot{S} = -(\beta_E \cdot S \cdot E + \beta_I \cdot S \cdot I), \tag{58a}$$

$$\dot{E} = \beta_E \cdot S \cdot E + \beta_I \cdot S \cdot I - \gamma \cdot E, \tag{58b}$$

$$\dot{I} = \gamma \cdot E - \lambda \cdot I, \tag{58c}$$

$$\dot{R} = \lambda \cdot I. \tag{58d}$$

with infection rates β_E and β_I, transition rate γ from exposure to infection, and recovery / death rate λ. Following Menda et al. (2021), which used an extension of the SEIR model to explain COVID-19 outbreaks, we consider ground-truth parameters $\beta_I = 0$, $\beta_E = 0.5$, $\gamma = 1/5$, and $\lambda = 1/21$ (the latter two correspond to realistic estimates of transition and recovery rate in COVID-19, given by Lauer et al. (2020); Bi et al. (2020)). Furthermore, we generate data on the time grid $\mathbb{T}_{\mathsf{D}} = \{30, 31, \ldots, 100\}$ from initial values $E_0 = 10^{-4}$, $I_0 = 10^{-5}$, $R_0 = 0$, and $S_0 = 1 - E_0 - I_0$ at time $t_0 = 0$, as $u(t_i) = H \cdot \hat{y}(t_i) + v(t_i)$, with a measurement matrix

$$H = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{59}$$

such that only the infected and recovered population is measured, and disturbed by Gaussian noise $v(t) \sim \mathcal{N}(0, \sigma^2 \cdot I)$ with $\sigma^2 = 5 \cdot 10^{-4}$.

Instead of estimating the full initial state, we parameterize it by the initial exposed and infected population count:

$$y_0(E_0, I_0) = \begin{bmatrix} 1 - E_0 - I_0, & E_0, & I_0, & 0 \end{bmatrix}^{\mathsf{T}}. \tag{60}$$

Thus, the parameters to be estimated by Fenrir in this experiment are $\theta = \{E_0, I_0, \beta_E, \gamma, \lambda, \sigma\}$, as well as the diffusion κ. All parameters are jointly optimised via L-BFGS, with bounds $E_0, I_0, \beta_E, \gamma, \lambda \in [0, 1]$, $\sigma^2 \in [10^{-6}, 10^2]$, and $\kappa \in [10^{-20}, 10^{20}]$. In each experiment, ODE parameters β_E, γ, λ are initialised as uniformly random; the starting values for E_0, I_0 are initialised as absolute values of samples from a Gaussian $\mathcal{N}(0, 10^{-2})$. Fenrir's probabilistic numerical integration is performed with a step size $\Delta = 0.2$.

C.5. Additional Details on Section 5.4: "Dynamical Systems with Fast Oscillations"

The considered pendulum system is given by a second-order ODE $\ddot{y} = -\frac{g}{L}\sin(y)$, which can be transformed to the following first-order equations

$$\dot{y_1} = y_2, \tag{61a}$$

$$\dot{y_2} = -\frac{g}{L}\sin(y_1), \tag{61b}$$

with the gravity constant $g = 9.81$. We assume a ground-truth parameter $L = 1$ and an initial value $y_0 = [0, \pi/2]$. The observation data is generated as $u(t_i) = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \hat{y}(t_i) + v_i$, with observation noise $v(t_i) \sim \mathcal{N}(0, \sigma^2)$, $\sigma^2 = 0.1$, on the grid $t_i \in \mathbb{T}_{\mathsf{D}} = \{0.01 \cdot i\}_{i=0}^{1000}$. In the corresponding experiment, we found it to be beneficial to first optimize the noise σ and diffusion parameter κ, before jointly optimizing all model parameters $\theta = \{y_0, L, \sigma\}$ and the diffusion κ. while assuming bounds $y_0 \in [-100, 100]^2$, $L \in [0, 100]$, $\sigma^2 \in [10^{-8}, 10^4]$, and $\kappa \in [10^{-20}, 10^{50}]$. Finally, Fenrir's physics-enhanced prior is computed with a fixed step size $\Delta = 0.1$.

Probabilistic Exponential Integrators

Nathanael Bosch
Tübingen AI Center, University of Tübingen
nathanael.bosch@uni-tuebingen.de

Philipp Hennig
Tübingen AI Center, University of Tübingen
philipp.hennig@uni-tuebingen.de

Filip Tronarp
Lund University
filip.tronarp@matstat.lu.se

Abstract

Probabilistic solvers provide a flexible and efficient framework for simulation, uncertainty quantification, and inference in dynamical systems. However, like standard solvers, they suffer performance penalties for certain *stiff* systems, where small steps are required not for reasons of numerical accuracy but for the sake of stability. This issue is greatly alleviated in semi-linear problems by the *probabilistic exponential integrators* developed in this paper. By including the fast, linear dynamics in the prior, we arrive at a class of probabilistic integrators with favorable properties. Namely, they are proven to be L-stable, and in a certain case reduce to a classic exponential integrator—with the added benefit of providing a probabilistic account of the numerical error. The method is also generalized to arbitrary non-linear systems by imposing piece-wise semi-linearity on the prior via Jacobians of the vector field at the previous estimates, resulting in *probabilistic exponential Rosenbrock methods*. We evaluate the proposed methods on multiple stiff differential equations and demonstrate their improved stability and efficiency over established probabilistic solvers. The present contribution thus expands the range of problems that can be effectively tackled within probabilistic numerics.

1 Introduction

Dynamical systems appear throughout science and engineering, and their accurate and efficient simulation is a key component in many scientific problems. There has also been a surge of interest in the intersection with machine learning, both regarding the usage of machine learning methods to model and solve differential equations [36, 18, 35], and in a dynamical systems perspective on machine learning methods themselves [8, 5]. This paper focuses on the numerical simulation of dynamical systems within the framework of *probabilistic numerics*, which treats the numerical solvers themselves as probabilistic inference methods [11, 12, 33]. In particular, we expand the range of problems that can be tackled within this framework and introduce a new class of stable probabilistic numerical methods for stiff ordinary differential equations (ODEs).

Stiff equations are problems for which certain implicit methods perform much better than explicit ones [10]. But implicit methods come with increased computational complexity per step, as they typically require solving a system of equations. *Exponential integrators* are an alternative class of methods for efficiently solving large stiff problems [48, 16, 7, 15]. They are based on the observation that, if the ODE has a semi-linear structure, the linear part can be solved exactly and only the non-linear part needs to be numerically approximated. The resulting methods are formulated in an explicit manner and do not require solving a system of equations, while achieving similar or better stability than implicit methods. However, such methods have not yet been formulated probabilistically.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

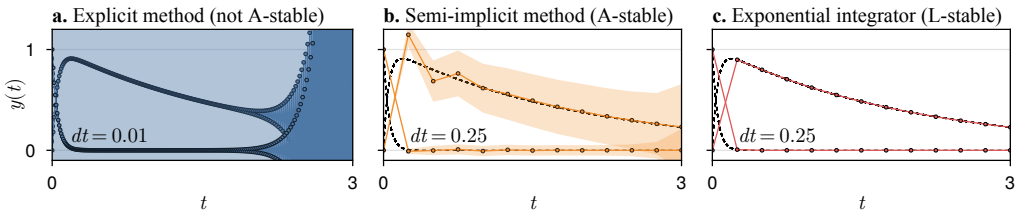


Figure 1: *Probabilistic numerical ODE solvers with different stability properties*. *Left*: The explicit `EK0` solver with a 3-times integrated Wiener process prior is unstable and diverges from the true solution. *Center*: The semi-implicit `EK1` with the same prior does not diverge even though it uses a larger step size, due to it being A-stable, but it exhibits oscillations in the initial phase of the solution. *Right*: The proposed exponential integrator is L-stable and thus does not exhibit any oscillations.

In this paper we develop *probabilistic exponential integrators*, a new class of probabilistic numerical solvers for stiff semi-linear ODEs. We build on the *ODE filters* which have emerged as an efficient and flexible class of probabilistic numerical methods for general ODEs [40, 21, 45]. They have known convergence rates [21, 46], which have also been demonstrated empirically [2, 26, 24], they are applicable to a wide range of numerical differential equation problems [23, 25, 3], their probabilistic output can be integrated into larger inference problems [20, 39, 47], and they can be formulated parallel-in-time [4]. But while it has been shown that the choice of underlying Gauss–Markov prior does influence the resulting ODE solver [30, 45, 2], there has not yet been strong evidence for the utility of priors other than the well-established integrated Wiener process. Probabilistic exponential integrators provide this evidence: in the probabilistic numerics framework, "solving the linear part of the ODE exactly" corresponds to an appropriate choice of prior.

Contributions Our main contribution is the development of probabilistic exponential integrators, a new class of stable probabilistic solvers for stiff semi-linear ODEs. We demonstrate the close link of these methods to classic exponential integrators in Proposition 1, provide an equivalence result to a classic exponential integrator in Proposition 2, and prove their L-stability in Proposition 3. To enable a numerically stable implementation, we present a quadrature-based approach to directly compute square-roots of the process noise covariance in Section 3.2. Finally, in Section 3.6 we also propose probabilistic exponential Rosenbrock methods for problems in which semi-linearity is not known a priori. We evaluate all proposed methods on multiple stiff problems and demonstrate the improved stability and efficiency of the probabilistic exponential integrators over existing probabilistic solvers.

2 Numerical ODE solutions as Bayesian state estimation

Let us first consider an initial value problem with some general non-linear ODE, of the form

$$\dot{y}(t) = f(y(t), t), \qquad t \in [0, T], \tag{1a}$$
$$y(0) = y_0, \tag{1b}$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value $y_0 \in \mathbb{R}^d$, and time span $[0, T]$. Probabilistic numerical ODE solvers aim to compute a posterior distribution over the ODE solution $y(t)$ such that it satisfies the ODE on a discrete set of points $\mathbb{T} = \{t_n\}_{n=0}^N \subset [0, T]$, that is

$$p\left(y(t) \,\middle|\, y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=0}^N\right). \tag{2}$$

We call this quantity, and approximations thereof, a *probabilistic numerical ODE solution*. Probabilistic numerical ODE solvers thus compute not just a single point estimate of the ODE solution, but a posterior distribution which provides a structured estimate of the numerical approximation error.

In the following, we briefly recapitulate the probabilistic ODE filter framework of Schober et al. [40] and Tronarp et al. [45] and define the prior, data model, and approximate inference scheme. In Section 3 we build on these foundations to derive the proposed probabilistic exponential integrator.

2.1 Gauss–Markov prior

A priori, we model $y(t)$ with a Gauss–Markov process, defined by a stochastic differential equation

$$\mathrm{d}Y(t) = AY(t)\,\mathrm{d}t + \kappa B\,\mathrm{d}W(t), \qquad Y(0) = Y_0, \tag{3}$$

with state $Y(t) \in \mathbb{R}^{d(q+1)}$, model matrices $A \in \mathbb{R}^{d(q+1)\times d(q+1)}$, $B \in \mathbb{R}^{d(q+1)\times d}$, diffusion scaling $\kappa \in \mathbb{R}$, and smoothness $q \in \mathbb{N}$. More precisely, A and B are chosen such that the state is structured as $Y(t) = [Y^{(0)}(t), \ldots, Y^{(q)}(t)]$, and then $Y^{(i)}(t)$ models the i-th derivative of $y(t)$. The initial value $Y_0 \in \mathbb{R}^{d(q+1)}$ must be chosen such that it enforces the initial condition, that is, $Y^{(0)}(0) = y_0$.

One concrete example of such a Gauss–Markov process that is commonly used in the context of probabilistic numerical ODE solvers is the q-times Integrated Wiener process, with model matrices

$$A_{\mathrm{IWP}(d,q)} = \begin{bmatrix} 0 & I_d & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_d \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \qquad B_{\mathrm{IWP}(d,q)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I_d \end{bmatrix}. \tag{4}$$

Alternatives include the class of Matérn processes and the integrated Ornstein–Uhlenbeck process [46]—the latter plays a central role in this paper and will be discussed in detail later.

$Y(t)$ satisfies linear Gaussian transition densities of the form [44]

$$Y(t+h) \mid Y(t) \sim \mathcal{N}\left(\Phi(h)Y(t), \kappa^2 Q(h)\right), \tag{5}$$

with transition matrices $\Phi(h)$ and $Q(h)$ given by

$$\Phi(h) = \exp\left(Ah\right), \qquad Q(h) = \int_0^h \Phi(h-\tau)BB^\top\Phi^\top(h-\tau)\,\mathrm{d}\tau. \tag{6}$$

These quantities can be computed with a matrix fraction decomposition [44]. For q-times integrated Wiener process priors, closed-form expressions for the transition matrices are available [21].

2.2 Information operator

The likelihood, or data model, of a probabilistic ODE solver relates the uninformed prior to the actual ODE solution of interest with an *information operator* $\mathcal{I}$ [6], defined as

$$\mathcal{I}[Y](t) \coloneqq E_1 Y(t) - f\left(E_0 Y(t), t\right), \tag{7}$$

where $E_i \in \mathbb{R}^{d\times d(q+1)}$ are selection matrices such that $E_i Y(t) = Y^{(i)}(t)$. $\mathcal{I}[Y]$ then captures how well Y solves the given ODE problem. In particular, $\mathcal{I}$ maps the true ODE solution y to the zero function, i.e. $\mathcal{I}[y] \equiv 0$. Conversely, if $\mathcal{I}[y](t) = 0$ holds for all $t \in [0, T]$ then y solves the given ODE. Unfortunately, it is in general infeasible to solve an ODE exactly and enforce $\mathcal{I}[Y](t) = 0$ everywhere, which is why numerical ODE solvers typically discretize the time interval and take discrete steps. This leads to the data model used in most probabilistic ODE solvers [45]:

$$\mathcal{I}[Y](t_n) = E_1 Y(t_n) - f\left(E_0 Y(t_n), t_n\right) = 0, \qquad t_n \in \mathbb{T} \subset [0, T]. \tag{8}$$

Note that this specific information operator is closely linked to the IVP considered in Eq. (1a). By defining a (slightly) different data model we can also formulate probabilistic numerical IVP solvers for higher-order ODEs or differential-algebraic equations, or encode additional information such as conservation laws or noisy trajectory observations [3, 39].

2.3 Approximate Gaussian inference

The resulting inference problem is described by a Gauss–Markov prior and a Dirac likelihood

$$Y(t_{n+1}) \mid Y(t_n) \sim \mathcal{N}\left(\Phi_n Y(t_n), \kappa^2 Q_n\right), \tag{9a}$$

$$Z_n \mid Y(t_n) \sim \delta\left(E_1 Y(t_n) - f\left(E_0 Y(t_n), t_n\right)\right), \tag{9b}$$

with $\Phi_n \coloneqq \Phi(t_{n+1} - t_n)$, $Q_n \coloneqq Q(t_{n+1} - t_n)$, initial value $Y(0) = Y_0$, discrete time grid $\{t_n\}_{n=0}^N$, and zero-valued data $Z_n = 0$ for all n. The solution of the resulting non-linear Gauss–Markov regression problem can then be efficiently approximated with Bayesian filtering and smoothing techniques [37]. Notable examples that have been used to construct probabilistic numerical ODE solvers include quadrature filters, the unscented Kalman filter, the iterated extended Kalman smoother, or particle filters [19, 45, 46]. Here, we focus on the well-established extended Kalman filter (EKF). We briefly discuss the EKF for the given state estimation problem in the following.

Prediction Given a Gaussian state estimate $Y(t_{n-1}) \sim \mathcal{N}(\mu_{n-1}, \Sigma_{n-1})$ and the linear conditional distribution as in Eq. (9a), the marginal distribution $Y(t_n) \sim \mathcal{N}(\mu_n^-, \Sigma_n^-)$ is also Gaussian, with

$$\mu_n^- = \Phi_{n-1}\mu_{n-1}, \tag{10a}$$
$$\Sigma_n^- = \Phi_{n-1}\Sigma_{n-1}\Phi_{n-1}^\top + \kappa^2 Q_{n-1}. \tag{10b}$$

Linearization To efficiently compute a tractable approximation of the true posterior, the EKF linearizes the information operator $\mathcal{I}$ around the predicted mean μ_n^-, i.e. $\mathcal{I}[Y](t_n) \approx H_n Y(t_n) + b_n$,

$$H_n = E_1 - F_y E_0, \tag{11a}$$
$$b_n = F_y E_0 \mu_n^- - f(E_0\mu_n^-, t_n). \tag{11b}$$

An exact linearization with Jacobian $F_y = \partial_y f(E_0\mu_n^-, t_n)$ leads to a semi-implicit probabilistic ODE solver, which we call the `EK1` [45]. Other choices include the zero matrix $F_y = 0$, which results in the explicit `EK0` solver [40, 21], or a diagonal Jacobian approximation (the `DiagonalEK1`) which combines some stability benefits of the `EK1` with the lower computational cost of the `EK0` [24].

Correction step In the linearized observation model, the posterior distribution of $Y(t_n)$ given the datum Z_n is again Gaussian. Its posterior mean and covariance (μ_n, Σ_n) are given by

$$S_n = H_n \Sigma_n^- H_n^\top, \tag{12a}$$
$$K_n = \Sigma_n^- H_n^\top S_n^{-1}, \tag{12b}$$
$$\mu_n = \mu_n^- - K_n\left(E_1\mu_n^- - f(E_0\mu_n^-, t_n)\right), \tag{12c}$$
$$\Sigma_n = (I - K_n H_n)\,\Sigma_n^-. \tag{12d}$$

This is also known as the *update* step of the EKF.

Smoothing To condition the state estimates on all data, the EKF can be followed by a smoothing pass. Starting with $\mu_N^S := \mu_N$ and $\Sigma_N^S := \Sigma_n$, it consists of the following backwards recursion:

$$G_n = \Sigma_n \Phi_n^\top \left(\Sigma_{n+1}^-\right)^{-1}, \tag{13a}$$
$$\mu_n^S = \mu_n + G_n(\mu_{n+1}^S - \mu_{n+1}^-), \tag{13b}$$
$$\Sigma_n^S = \Sigma_n + G_n(\Sigma_{n+1}^S - \Sigma_{n+1}^-)G_n^\top. \tag{13c}$$

Result The above computations result in a *probabilistic numerical ODE solution* with marginals

$$p\left(Y(t_i) \,\middle|\, \{E_1 Y(t_n) - f\left(E_0 Y(t_n), t_n\right) = 0\}_{n=0}^N\right) \approx \mathcal{N}\left(\mu_i^S, \Sigma_i^S\right), \tag{14}$$

which, by construction of the state Y, also contains estimates for the ODE solution as $y(t) = E_0 Y(t)$. Since the EKF-based probabilistic solver does not compute only the marginals in Eq. (14), but a full posterior distribution for the continuous object $y(t)$, it can be evaluated for times $t \notin \mathbb{T}$ (also known as "dense output" in the context of ODE solvers); it can produce joint samples from this posterior; and it can be used as a Gauss–Markov prior for subsequent inference tasks [40, 2, 47].

2.4 Practical considerations and implementation details

To improve numerical stability and preserve positive-semidefiniteness of the computed covariance matrices, probabilistic ODE solvers typically operate on square-roots of covariance matrices, defined by a matrix decomposition of the form $M = \sqrt{M}\sqrt{M}^\top$ [26]. For example, the Cholesky factor is one possible square-root of a positive definite matrix. But in general, the algorithm does not require the square-roots to be upper- or lower-triangular, or even square. Additionally, we compute the exact initial state Y_0 from the IVP using Taylor-mode automatic differentiation [9, 26], we compute smoothing estimates with preconditioning [26], and we calibrate uncertainties globally with a quasi-maximum likelihood approach [45, 2].

3 Probabilistic exponential integrators

In the remainder of the paper, unless otherwise stated, we focus on IVPs with a semi-linear vector-field

$$\dot{y}(t) = f(y(t), t) = Ly(t) + N(y(t), t). \tag{15}$$

Assuming N admits a Taylor series expansion around t, the variation of constants formula provides a formal expression of the solution at time $t + h$:

$$y(t+h) = \exp(Lh)y(t) + \sum_{k=0}^{\infty} h^{k+1} \left(\int_0^1 \exp(Lh(1-\tau)) \frac{\tau^k}{k!} \, \mathrm{d}\tau \right) \frac{\mathrm{d}^k}{\mathrm{d}t^k} N(y(t), t). \tag{16}$$

This observation is the starting point for the development of *exponential integrators* [31, 15]. By further defining the so-called φ-functions

$$\varphi_k(z) = \int_0^1 \exp(z(1-\tau)) \frac{\tau^{k-1}}{(k-1)!} \, \mathrm{d}\tau, \tag{17}$$

the above identity of the ODE solution simplifies to

$$y(t+h) = \exp(Lh)y(t) + \sum_{k=0}^{\infty} h^{k+1} \varphi_{k+1}(Lh) \frac{\mathrm{d}^k}{\mathrm{d}t^k} N(y(t), t). \tag{18}$$

In this section we develop a class of *probabilistic exponential integrators*. This is achieved by defining an appropriate class of priors that absorbs the partial linearity, which leads to the integrated Ornstein–Uhlenbeck processes. Proposition 1 below directly relates this choice of prior to the classical exponential integrators. Proposition 2 demonstrates a direct equivalence between the predictor-corrector form exponential trapezoidal rule and the once integrated Ornstein–Uhlenbeck process. Furthermore, the favorable stability properties of classical exponential integrators is retained for the probabilistic counterparts as shown in Proposition 3.

3.1 The integrated Ornstein–Uhlenbeck process

In Section 2.1 we highlighted the choice of the q-times integrated Wiener process prior, which essentially corresponds to modeling the $(q-1)$-th derivative of the right-hand side f with a Wiener process. Here we follow a similar motivation, but only for the non-linear part N. Differentiating both sides of Eq. (15) $q-1$ times with respect to t yields

$$\frac{\mathrm{d}^{q-1}}{\mathrm{d}t^{q-1}} \dot{y}(t) = L \frac{\mathrm{d}^{q-1}}{\mathrm{d}t^{q-1}} y(t) + \frac{\mathrm{d}^{q-1}}{\mathrm{d}t^{q-1}} N(y(t), t). \tag{19}$$

Then, modeling $\frac{\mathrm{d}^{q-1}}{\mathrm{d}t^{q-1}} N(y(t), t)$ as a Wiener process and relating the result to $y(t)$ gives

$$\mathrm{d}y^{(i)}(t) = y^{(i+1)}(t) \, \mathrm{d}t, \tag{20a}$$

$$\mathrm{d}y^{(q)}(t) = Ly^{(q)}(t) \, \mathrm{d}t + \kappa I_d \, \mathrm{d}W^{(q)}(t). \tag{20b}$$

This process is also known as the q-times integrated Ornstein–Uhlenbeck process (IOUP), with rate parameter L and diffusion parameter κ. It can be equivalently stated with the previously introduced notation (Section 2.1), by defining a state $Y(t)$, as the solution of a linear time-invariant (LTI) SDE as in Eq. (3), with system matrices

$$A_{\mathrm{IOUP}(d,q)} = \begin{bmatrix} 0 & I_d & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_d \\ 0 & 0 & \cdots & L \end{bmatrix}, \qquad B_{\mathrm{IOUP}(d,q)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I_d \end{bmatrix}. \tag{21}$$

Remark 1 (The mean of the IOUP process solves the linear part of the ODE exactly). By taking the expectation of Eq. (20b) and by linearity of integration, we can see that the mean of the IOUP satisfies

$$\dot{\mu}^{(0)}(t) = L\mu^{(0)}(t), \qquad \mu^{(0)}(0) = y_0. \tag{22}$$

This is in line with the motivation of exponential integrators: the linear part of the ODE is solved exactly, and we only need to approximate the non-linear part. Figure 2 visualizes this idea.

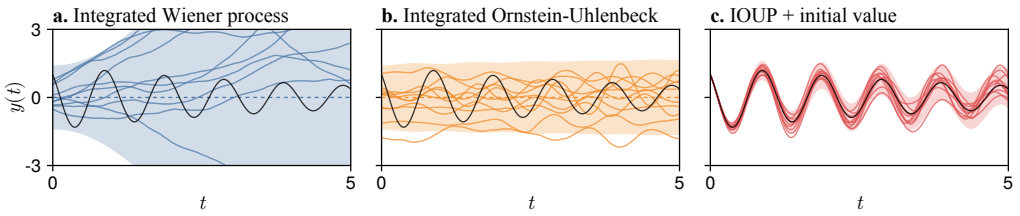


Figure 2: *Damped oscillator dynamics and priors with different degrees of encoded information. Left*: Once-integrated Wiener process, a popular prior for probabilistic ODE solvers. *Center*: Once-integrated Ornstein–Uhlenbeck process (IOUP) with rate parameter chosen to encode the known linearity of the ODE. *Right*: IOUP with both the ODE information and a specified initial value and derivative. This is the kind of prior used in the probabilistic exponential integrator.

3.2 The transition parameters of the integrated Ornstein–Uhlenbeck process

Since the process $Y(t)$ is defined as the solution of a linear time-invariant SDE, it satisfies discrete transition densities $p(Y(t+h) \mid Y(t)) = \mathcal{N}\left(\Phi(h)Y(t), \kappa^2 Q(h)\right)$. The following result shows that the transition parameters are intimately connected with the φ-functions defined in Eq. (17).

Proposition 1. *The transition matrix of a q-times integrated Ornstein–Uhlenbeck process satisfies*

$$\Phi(h) = \begin{bmatrix} \exp\left(A_{\mathrm{IWP}(d,q-1)}h\right) & \Phi_{12}(h) \\ 0 & \exp(Lh) \end{bmatrix}, \qquad \textit{with} \qquad \Phi_{12}(h) := \begin{bmatrix} h^q\varphi_q(Lh) \\ h^{q-1}\varphi_{q-1}(Lh) \\ \vdots \\ h\varphi_1(Lh) \end{bmatrix}. \tag{23}$$

Proof in Appendix A. Although Proposition 1 indicates that $\Phi(h)$ may be computed more efficiently than by calling a matrix-exponential on a $d(q+1) \times d(q+1)$ matrix, this is known to be numerically less stable [41]. We therefore compute $\Phi(h)$ with the standard matrix-exponential formulation.

Directly computing square-roots of the process noise covariance Numerically stable probabilistic ODE solvers require a square-root, $\sqrt{Q(h)}$, of the process noise covariance rather than the full matrix, $Q(h)$. For IWP priors this can be computed from the closed-form representation of $Q(h)$ via an appropriately preconditioned Cholesky factorization [26]. However, for IOUP priors we have not found an analogous method that works reliably. Therefore, we compute $\sqrt{Q(h)}$ directly with numerical quadrature. More specifically, given a quadrature rule with nodes $\tau_i \in [0, h]$ and positive weights $w_i > 0$, the integral for $Q(h)$ given in Eq. (6) is approximated by

$$Q(h) \approx \sum_{i=1}^{m} w_i \exp(A(h-\tau_i))BB^\top \exp(A^\top(h-\tau_i)) =: \sum_{i=1}^{m} M_i, \tag{24}$$

with square-roots $\sqrt{M_i} = \sqrt{w_i}\exp(A(h-\tau_i))B$ of the summands, which is well-defined since $w_i > 0$. We can thus compute a square-root representation of the sum with a QR-decomposition

$$X \cdot R = \mathrm{QR}\left(\begin{bmatrix}\sqrt{M_1} & \cdots & \sqrt{M_m}\end{bmatrix}^\top\right). \tag{25}$$

We obtain $Q(h) \approx R^\top R$, and therefore an approximate square-root factor is given by $\sqrt{Q(h)} \approx R^\top$. Similar ideas have previously been employed for time integration of Riccati equations [42, 43]. We use this quadrature-trick for all IOUP methods, with Gauss–Legendre quadrature on $m = q$ nodes.

3.3 Linearization and correction

The information operator of the probabilistic exponential integrator is defined exactly as in Section 2.2. But since we now assume a semi-linear vector-field f, we have an additional option for the linearization: instead of choosing the exact $F_y = \partial_y f$ (`EK1`) or the zero-matrix $F_y = 0$ (`EK0`), a cheap approximate Jacobian is given by the linear part $F_y = L$. We denote this approach by `EKL`. This is chosen as the default for the probabilistic exponential integrator. Note that the `EKL` approach can also be combined with an IWP prior, which will be serve as an additional baseline in the Section 4.

3.4 Equivalence to the classic exponential trapezoidal rule in predict-evaluate-correct mode

Now that the probabilistic exponential integrator has been defined, we can establish an equivalence result to a classic exponential integrator, similarly to the closely-related equivalence statement by Schober et al. [40, Proposition 1] for the non-exponential case.

Proposition 2 (Equivalence to the PEC exponential trapezoidal rule)**.** *The mean estimate of the probabilistic exponential integrator with a once-integrated Ornstein–Uhlenbeck prior with rate parameter L is equivalent to the classic exponential trapezoidal rule in predict-evaluate-correct mode, with the predictor being the exponential Euler method. That is, it is equivalent to the scheme*

$$\tilde{y}_{n+1} = \varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n), \tag{26a}$$

$$y_{n+1} = \varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n) + h^2\varphi_2(Lh)\frac{N(\tilde{y}_{n+1}) - N(\tilde{y}_n)}{h}, \tag{26b}$$

where Eq. (26a) corresponds to a prediction step with the exponential Euler method, and Eq. (26b) corresponds to a correction step with the exponential trapezoidal rule.

The proof is given in Appendix B. This equivalence result provides another theoretical justification for the proposed probabilistic exponential integrator. But note that the result only holds for the mean, while the probabilistic solver computes additional quantities in order to track the solution uncertainty, namely covariances. These are not provided by a classic exponential integrator.

3.5 L-stability of the probabilistic exponential integrator

When solving stiff ODEs, the actual efficiency of a numerical method often depends on its stability. One such property is *A-stability*: It guarantees that the numerical solution of a decaying ODE will also decay, independently of the chosen step size. In contrast, explicit methods typically only decay for sufficiently small steps. In the context of probabilistic ODE solvers, the `EK0` is considered to be explicit, but the `EK1` with IWP prior has been shown to be A-stable [45]. Here, we show that the probabilistic exponential integrator satisfies the stronger *L-stability*: the numerical solution not only decays, but it decays *fast*, i.e. it goes to zero as the step size goes to infinity. Figure 1 visualizes the different probabilistic solver stabilities. For formal definitions, see for example [27, Section 8.6].

Proposition 3 (L-stability)**.** *The probabilistic exponential integrator is L-stable.*

The full proof is given in Appendix C. The property essentially follows from Remark 1 which stated that the IOUP solves linear ODEs exactly. This implies fast decay and gives L-stability.

3.6 Probabilistic exponential Rosenbrock-type methods

We conclude with a short excursion into exponential Rosenbrock methods [14, 17, 28]: Given a non-linear ODE $\dot{y}(t) = f(y(t), t)$, exponential Rosenbrock methods perform a continuous linearization of the right-hand side f around the numerical ODE solution and essentially solve a sequence of IVPs

$$\dot{y}(t) = J_n y(t) + \left(f(y(t), t) - J_n y(t)\right), \qquad t \in [t_n, t_{n+1}], \tag{27a}$$

$$y(t_n) = y_n, \tag{27b}$$

where J_n is the Jacobian of f at the numerical solution estimate $\hat{y}(t_n)$. This approach enables exponential integrators for problems where the right-hand side f is not semi-linear. Furthermore, by automatically linearizing along the numerical solution the linearization can be more accurate, the Lipschitz-constant of the non-linear remainder becomes smaller, and the resulting solvers can thus be more efficient than their globally linearized counterparts [17].

This can also be done in the probabilistic setting: By linearizing the ODE right-hand side f at each step of the solver around the filtering mean $E_0\mu_n$, we (locally) obtain a semi-linear problem. Then, updating the rate parameter of the integrated Ornstein–Uhlenbeck process at each step of the numerical solver results in *probabilistic exponential Rosenbrock-type methods*. As before, the linearization of the information operator can be done with any of the `EK0`, `EK1`, or `EKL`. But since here the prediction relies on exact local linearization, we will by default also use an exact `EK1` linearization. The resulting solver and its stability and efficiency will be evaluated in the following experiments.

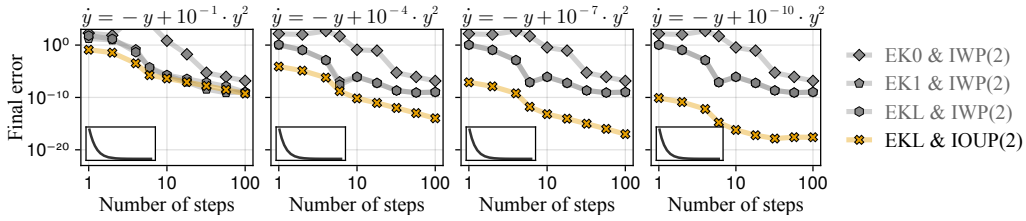


Figure 3: *The IOUP prior is more beneficial with increasing linearity of the ODE.* In all three examples, the IOUP-based exponential integrator achieves lower error while requiring fewer steps than the IWP-based solvers. This effect is more pronounced for the more linear ODEs.

4 Experiments

In this section we investigate the utility and performance of the proposed probabilistic exponential integrators and compare it to standard non-exponential probabilistic solvers on multiple ODEs. All methods are implemented in the Julia programming language [1], with special care being taken to implement the solvers in a numerically stable manner, that is, with exact state initialization, preconditioned state transitions, and a square-root implementation [26]. Reference solutions are computed with the DifferentialEquations.jl package [34]. All experiments run on a single, consumer-level CPU. Code for the implementation and experiments is publicly available on GitHub.[1]

4.1 Logistic equation with varying degrees of non-linearity

We start with a simple one-dimensional initial value problem: a logistic model with negative growth rate parameter $r = -1$ and carrying capacity $K \in \mathbb{R}_+$, of the form

$$\dot{y}(t) = -y(t) + \frac{1}{K} y(t)^2, \qquad t \in [0, 10], \tag{28a}$$

$$y(0) = 1. \tag{28b}$$

The non-linearity of this problem can be directly controlled through the parameter K. Therefore, this test problem lets us investigate the IOUP's capability to leverage known linearity in the ODE.

We compare the proposed exponential integrator to all introduced IWP-based solvers, with different linearization strategies: `EK0` approximates $\partial_y f \approx 0$ (and is thus explicit), `EKL` approximates $\partial_y f \approx -1$, and `EK1` linearizes with the correct Jacobian $\partial_y f$. The results for four different values of K are shown in Fig. 3. The explicit solver shows the largest error of all compared solvers, likely due to its lacking stability. On the other hand, the proposed exponential integrator behaves as expected: the IOUP prior is most beneficial for larger values of K, and as the non-linearity becomes more pronounced the performance of the IOUP approaches that of the IWP-based solver. Though for large step sizes, the IOUP outperforms the IWP prior even for the most non-linear case with $K = 10$.

4.2 Burger's equation

Here, we consider Burger's equation, which is a semi-linear partial differential equation (PDE)

$$\partial_t u(x,t) = D\partial_x^2 u(x,t) - u(x,t)\partial_x u(x,t), \qquad x \in [0,1], \quad t \in [0,1], \tag{29}$$

with diffusion coefficient $D \in \mathbb{R}_+$. We transform the problem into a semi-linear ODE with the method of lines [29, 38], and discretize the spatial domain on 250 equidistant points and approximate the differential operators with finite differences. The full IVP specification, including all domains, initial and boundary conditions, and additional information on the discretization, is given in Appendix D.

The results shown in Fig. 4 demonstrate the different stability properties of the solvers: The explicit EK0 with IWP prior is unable to solve the IVP for any of the step sizes due to its insufficient stability, and even the A-stable EK1 and the more approximate EKL require small enough steps $\Delta t < 10^{-1}$. On the other hand, both exponential integrators are able to compute meaningful solutions for a larger range of step sizes. They both achieve lower errors for most settings than their non-exponential

[1] https://github.com/nathanaelbosch/probabilistic-exponential-integrators

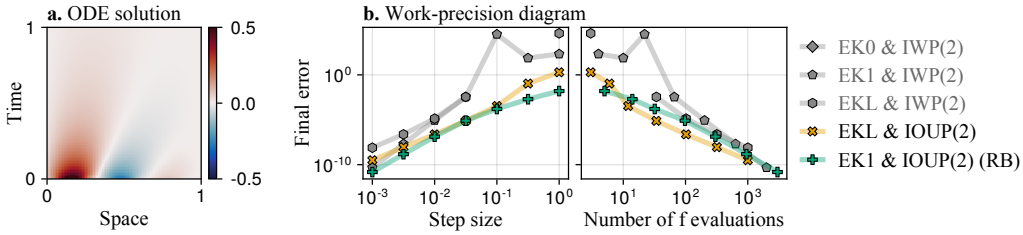


Figure 4: *Benchmarking probabilistic ODE solvers on Burger's equation.* Exponential and non-exponential probabilistic solvers are compared on Burger's equation (a) in two work-precision diagrams (b). Both exponential integrators with IOUP prior achieve lower errors than the existing IWP-based solvers, in particular for large steps. This indicates their stronger stability properties.

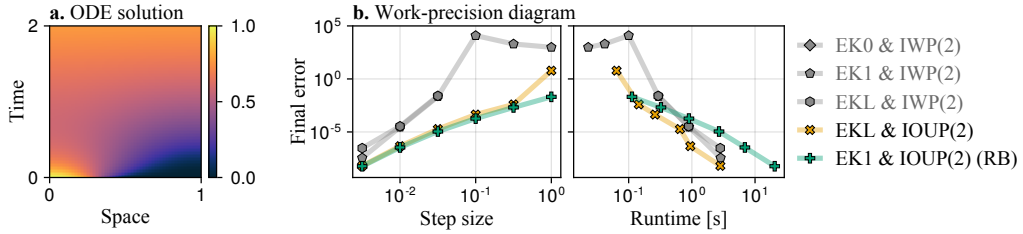


Figure 5: *Benchmarking probabilistic ODE solvers on a reaction-diffusion model.* Exponential and non-exponential probabilistic solvers are compared on a reaction-diffusion model (a) in two work-precision diagrams (b). The proposed exponential integrators with IOUP prior achieve lower errors per step size than the existing IWP-based methods. The runtime comparison shows the increased cost of the Rosenbrock-type (RB) method, while the non-Rosenbrock probabilistic exponential integrator performs best in this comparison.

counterparts. The second diagram in Fig. 4 compares the achieved error to the number of vector-field evaluations and points out a trade-off between both exponential methods: Since the Rosenbrock method additionally computes two Jacobians (with automatic differentiation) per step, it needs to evaluate the vector-field more often than the non-Rosenbrock method. Thus, for expensive-to-evaluate vector fields the standard probabilistic exponential integrator might be preferable.

4.3 Reaction-diffusion model

Finally, we consider a discretized reaction-diffusion model given by a semi-linear PDE

$$\partial_t u(x,t) = D\partial_x^2 u(x,t) + R(u(x,t)), \qquad x \in [0,1], \quad t \in [0,T], \tag{30}$$

where $D \in \mathbb{R}_+$ is the diffusion coefficient and $R(u) = u(1-u)$ is a logistic reaction term [22]. A finite-difference discretization of the spatial domain transforms this PDE into an IVP with semi-linear ODE. The full problem specification is provided in Appendix D.

Figure 5 shows the results. We again observe the improved stability of the exponential integrator variants by their lower error for large step sizes, and they outperform the IWP-based methods on all settings. The runtime-evaluation in Fig. 5 also visualizes another drawback of the Rosenbrock-type method: Since the problem is re-linearized at each step, the IOUP also needs to be re-discretized and thus a matrix exponential needs to be computed. In comparison, the non-Rosenbrock method only discretizes the IOUP prior once at the start of the solve. This advantage makes the non-Rosenbrock probabilistic exponential integrator the most performant solver in this experiment.

5 Limitations

The probabilistic exponential integrator shares many properties of both classic exponential integrators and of other filtering-based probabilistic solvers. This also brings some challenges.

Cost of computing matrix exponentials The IOUP prior is more expensive to discretize than the IWP as it requires computing a matrix exponential. This trade-off is well-known also in the context of classic exponential integrators. One approach to reduce computational cost is to compute the matrix exponential only approximately [32], for example with Krylov-subspace methods [13, 17]. Extending these techniques to the probabilistic solver setting thus poses an interesting direction for future work.

Cubic scaling in the ODE dimension The probabilistic exponential integrator shares the complexity most (semi-)implicit ODE solvers: while being linear in the number of time steps, it scales cubically in the ODE dimension. By exploiting structure in the Jacobian and in the prior, some filtering-based ODE solvers have been formulated with linear scaling in the ODE dimension [24]. But this approach does not directly extend to the IOUP-prior. Nevertheless, exploiting known structure could be particularly relevant to construct solvers for specific ODEs, such as certain discretized PDEs.

6 Conclusion

We have presented probabilistic exponential integrators, a new class of probabilistic solvers for stiff semi-linear ODEs. By incorporating the fast, linear dynamics directly into the prior of the solver, the method essentially solves the linear part exactly, in a similar manner as classic exponential integrators. We also extended the proposed method to general non-linear systems via iterative re-linearization and presented probabilistic exponential Rosenbrock-type methods. Both methods have been shown both theoretically and empirically to be more stable than their non-exponential probabilistic counterparts. This work further expands the toolbox of probabilistic numerics and opens up new possibilities for accurate and efficient probabilistic simulation and inference in stiff dynamical systems.

Acknowledgments and Disclosure of Funding

The authors gratefully acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) through Project ADIMEM (FKZ 01IS18052B), and financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence Machine Learning - New Perspectives for Science, EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. Filip Tronarp was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Nathanael Bosch. The authors also thank Jonathan Schmidt for many valuable discussions and for helpful feedback on the manuscript.

References

[1] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[2] N. Bosch, P. Hennig, and F. Tronarp. Calibrated adaptive probabilistic ODE solvers. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021.

[3] N. Bosch, F. Tronarp, and P. Hennig. Pick-and-mix information operators for probabilistic ODE solvers. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022.

[4] N. Bosch, A. Corenflos, F. Yaghoobi, F. Tronarp, P. Hennig, and S. Särkkä. Parallel-in-time probabilistic numerical ODE solvers, 2023.

[5] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[6] J. Cockayne, C. Oates, T. Sullivan, and M. Girolami. Bayesian probabilistic numerical methods. *SIAM Review*, 61:756–789, 2019.

[7] S. M. Cox and P. C. Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

[8] W. E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):111, Mar 2017.

[9] A. Griewank and A. Walther. *Evaluating Derivatives*. Society for Industrial and Applied Mathematics, second edition, 2008.

[10] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer series in computational mathematics. Springer-Verlag, 1991.

[11] P. Hennig, M. A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings. Mathematical, physical, and engineering sciences*, 471(2179): 20150142–20150142, Jul 2015.

[12] P. Hennig, M. A. Osborne, and H. P. Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022.

[13] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 34(5):19111925, Oct 1997.

[14] M. Hochbruck and A. Ostermann. Explicit integrators of Rosenbrock-type. *Oberwolfach Reports*, 3(2):1107–1110, 2006.

[15] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19:209286, 2010.

[16] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM Journal on Scientific Computing*, 19(5):1552–1574, 1998.

[17] M. Hochbruck, A. Ostermann, and J. Schweitzer. Exponential Rosenbrock-type methods. *SIAM Journal on Numerical Analysis*, 47(1):786–803, 2009.

[18] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422440, May 2021.

[19] H. Kersting and P. Hennig. Active uncertainty calibration in Bayesian ode solvers. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 309–318, June 2016.

[20] H. Kersting, N. Krämer, M. Schiegg, C. Daniel, M. Tiemann, and P. Hennig. Differentiable likelihoods for fast inversion of Likelihood-free dynamical systems. In *International Conference on Machine Learning*. PMLR, 2020.

[21] H. Kersting, T. J. Sullivan, and P. Hennig. Convergence rates of Gaussian ODE filters. *Statistics and computing*, 30(6):1791–1816, 2020.

[22] A. N. Kolmogorov. A study of the equation of diffusion with increase in the quantity of matter, and its application to a biological problem. *Moscow University Bulletin of Mathematics*, 1:1–25, 1937.

[23] N. Krämer and P. Hennig. Linear-time probabilistic solution of boundary value problems. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021.

[24] N. Krämer, N. Bosch, J. Schmidt, and P. Hennig. Probabilistic ODE solutions in millions of dimensions. In *International Conference on Machine Learning*. PMLR, 2022.

[25] N. Krämer, J. Schmidt, and P. Hennig. Probabilistic numerical method of lines for time-dependent partial differential equations. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022.

[26] N. Krämer and P. Hennig. Stable implementation of probabilistic ode solvers, 2020.

[27] J. D. Lambert. *Computational Methods in Ordinary Differential Equations*. Introductory Mathematics for Scientists And Engineers. Wiley, 1973.

[28] V. T. Luan and A. Ostermann. Exponential Rosenbrock methods of order five construction, analysis and numerical comparisons. *Journal of Computational and Applied Mathematics*, 255: 417–431, 2014.

[29] N. K. Madsen. The method of lines for the numerical solution of partial differential equations. *Proceedings of the SIGNUM meeting on Software for partial differential equations*, 1975.

[30] E. Magnani, H. Kersting, M. Schober, and P. Hennig. Bayesian filtering for ODEs with bounded derivatives, 2017.

[31] B. V. Minchev and W. M. Wright. A review of exponential integrators for first order semi-linear problems. Technical report, Norges Teknisk-Naturvetenskaplige Universitet, 2005.

[32] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):349, Jan 2003.

[33] C. J. Oates and T. J. Sullivan. A modern retrospective on probabilistic numerics. *Statistics and Computing*, 29(6):1335–1351, 2019.

[34] C. Rackauckas and Q. Nie. DifferentialEquations.jl a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

[35] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman. Universal differential equations for scientific machine learning, 2021.

[36] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686707, Feb 2019.

[37] S. Särkkä. *Bayesian Filtering and Smoothing*, volume 3 of *Institute of Mathematical Statistics textbooks*. Cambridge University Press, 2013.

[38] W. E. Schiesser. *The numerical method of lines: integration of partial differential equations*. Elsevier, 2012.

[39] J. Schmidt, N. Krämer, and P. Hennig. A probabilistic state space model for joint inference from differential equations and data. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021.

[40] M. Schober, S. Särkkä, and P. Hennig. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, Jan 2019.

[41] R. B. Sidje. Expokit: A software package for computing matrix exponentials. *ACM Transactions on Mathematical Software*, 24(1):130156, mar 1998.

[42] T. Stillfjord. Low-rank second-order splitting of large-scale differential Riccati equations. *IEEE Transactions on Automatic Control*, 60(10):2791–2796, 2015.

[43] T. Stillfjord. Adaptive high-order splitting schemes for large-scale differential Riccati equations. *Numerical Algorithms*, 78(4):11291151, Sep 2017.

[44] S. Särkkä and A. Solin. *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2019.

[45] F. Tronarp, H. Kersting, S. Särkkä, and P. Hennig. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29(6): 1297–1315, 2019.

[46] F. Tronarp, S. Särkkä, and P. Hennig. Bayesian ODE solvers: The maximum a posteriori estimate. *Statistics and Computing*, 31(3):1–18, 2021.

[47] F. Tronarp, N. Bosch, and P. Hennig. Fenrir: Physics-enhanced regression for initial value problems. In *International Conference on Machine Learning*. PMLR, 2022.

[48] C. Van Loan. Computing integrals involving the matrix exponential. *IEEE Transactions on Automatic Control*, 23(3):395–404, 1978.

Probabilistic Exponential Integrators — Appendix

A Proof of Proposition 1: Structure of the transition matrix

Proof of Proposition 1. The drift-matrix $A_{\text{IOUP}(d,q)}$ as given in Eq. (21) has block structure

$$A_{\text{IOUP}(d,q)} = \begin{bmatrix} A_{\text{IWP}(d,q-1)} & E_{q-1} \\ 0 & L \end{bmatrix}, \tag{31}$$

where $E_{q-1} \coloneqq [0 \quad \dots \quad 0 \quad I_d]^\top \in \mathbb{R}^{dq \times d}$. From Van Loan [48, Theorem 1], it follows

$$\Phi(h) = \begin{bmatrix} \exp\big(A_{\text{IWP}(d,q-1)}h\big) & \Phi_{12}(h) \\ 0 & \exp(Lh) \end{bmatrix}, \tag{32}$$

which is precisely Eq. (23). The same theorem also gives $\Phi_{12}(h)$ as

$$\Phi_{12}(h) = \int_0^h \exp(A_{\text{IWP}(d,q-1)}(h-\tau)) E_{q-1}^{(d-1)} \exp(L\tau)\,\mathrm{d}\tau. \tag{33}$$

Its ith $d \times d$ block is readily given by

$$\begin{aligned} (\Phi_{12}(h))_i &= \int_0^h E_i^\top \exp(A_{\text{IWP}(d,q-1)}(h-\tau)) E_{q-1} \exp(L\tau)\,\mathrm{d}\tau \\ &= \int_0^h \frac{(h-\tau)^{q-1-i}}{(q-1-i)!} \exp(L\tau)\,\mathrm{d}\tau \\ &= h^{q-i} \int_0^1 \frac{\tau^{q-1-i}}{(q-1-i)!} \exp(Lh(1-\tau))\,\mathrm{d}\tau \\ &= h^{q-i} \varphi_{q-i}(Lh), \end{aligned} \tag{34}$$

where the second last equality used the change of variables $\tau = h(1-u)$, and the last line follows by definition. □

B Proof of Proposition 2: Equivalence to a classic exponential integrator

We first briefly recapitulate the probabilistic exponential integrator setup for the case of the once integrated Ornstein–Uhlenbeck process, and then provide some auxiliary results. Then, we prove Proposition 2 in Appendix B.3.

B.1 The probabilistic exponential integrator with once-integrated Ornstein–Uhlenbeck prior

The integrated Ornstein–Uhlenbeck process prior with rate parameter L results in transition densities $Y(t+h) \mid Y(t) \sim \mathcal{N}(Y(t+h); \Phi(h)Y(t), Q(h))$, with transition matrices (from Proposition 1)

$$\Phi(h) = \exp(Ah) = \begin{bmatrix} I & h\varphi_1(Lh) \\ 0 & \varphi_0(Lh) \end{bmatrix}, \tag{35}$$

$$Q(h) = \int_0^h \exp(A\tau) BB^\top \exp(A^\top\tau)\,\mathrm{d}\tau \tag{36}$$

$$= \int_0^h \begin{bmatrix} I & \tau\varphi_1(L\tau) \\ 0 & \varphi_0(L\tau) \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \tau\varphi_1(L\tau) \\ 0 & \varphi_0(L\tau) \end{bmatrix}^\top \mathrm{d}\tau \tag{37}$$

$$= \int_0^h \begin{bmatrix} \tau^2\varphi_1(L\tau)\varphi_1(L\tau)^\top & \tau\varphi_1(L\tau)\varphi_0(L\tau)^\top \\ \tau\varphi_0(L\tau)\varphi_1(L\tau)^\top & \varphi_0(L\tau)\varphi_0(L\tau)^\top \end{bmatrix} \mathrm{d}\tau, \tag{38}$$

where we assume a unit diffusion $\sigma^2 = 1$. To simplify notation, we assume an equidistant time grid $\mathbb{T} = \{t_n\}_{n=0}^N$ with $t_n = n \cdot h$ for some step size h, and we denote the constant transition matrices simply by Φ and Q and write $Y_n = Y(t_n)$.

Before getting to the actual proof, let us also briefly recapitulate the filtering formulas that are computed at each solver step. Given a Gaussian distribution $Y_n \sim \mathcal{N}(Y_n; \mu_n, \Sigma_n)$, the prediction step computes

$$\mu_{n+1}^- = \Phi \mu_n, \tag{39}$$
$$\Sigma_{n+1}^- = \Phi(h) \Sigma_n \Phi(h)^\top + Q(h). \tag{40}$$

Then, the combined linearization and correction step compute

$$\hat{z}_{n+1} = E_1 \mu_{n+1}^- - f(E_0 \mu_{n+1}^-), \tag{41}$$
$$S_{n+1} = H \Sigma_{n+1}^- H^\top, \tag{42}$$
$$K_{n+1} = \Sigma_{n+1}^- H^\top S_{n+1}^{-1}, \tag{43}$$
$$\mu_{n+1} = \mu_{n+1}^- - K_{n+1} \hat{z}_{n+1}, \tag{44}$$
$$\Sigma_{n+1} = \Sigma_{n+1}^- - K_{n+1} S_{n+1} K_{n+1}^\top, \tag{45}$$

with observation matrix $H = E_1 - L E_0 = [-L \quad I]$, since we perform the proposed `EKL` linearization.

B.2 Auxiliary results

In the following, we show some properties of the transition matrices and the covariances that will be needed in the proof of Proposition 2 later.

First, note that by defining $\varphi_0(z) = \exp z$, the φ-functions satisfy the following recurrence formula:

$$z \varphi_k(z) = \varphi_{k-1}(z) - \frac{1}{(k-1)!}. \tag{46}$$

See e.g. Hochbruck and Ostermann [15]. This property will be used throughout the remainder of the section.

Lemma B.1. *The transition matrices $\Phi(h), Q(h)$ of the once integrated Ornstein–Uhlenbeck process with rate parameter L satisfy*

$$H \Phi(h) = [-L \quad I], \tag{47}$$
$$Q(h) H^\top = \begin{bmatrix} h^2 \varphi_2(Lh) \\ h \varphi_1(Lh) \end{bmatrix}, \tag{48}$$
$$H Q(h) H^\top = hI, \tag{49}$$

Proof.

$$H\Phi(h) = (E_1 - LE_0) \begin{bmatrix} I & h\varphi_1(Lh) \\ 0 & \varphi_0(Lh) \end{bmatrix} = [0 \quad \varphi_0(Lh)] - L [I \quad h\varphi_1(Lh)] = [-L \quad I]. \tag{50}$$

$$Q(h)H^\top = \int_0^h \begin{bmatrix} \tau^2 \varphi_1(L\tau)\varphi_1(L\tau)^\top & \tau \varphi_1(L\tau)\varphi_0(L\tau)^\top \\ \tau \varphi_0(L\tau)\varphi_1(L\tau)^\top & \varphi_0(L\tau)\varphi_0(L\tau)^\top \end{bmatrix} H^\top \, \mathrm{d}\tau \tag{51}$$
$$= \int_0^h \begin{bmatrix} \tau \varphi_1(L\tau)\varphi_0(L\tau)^\top - L\tau^2 \varphi_1(L\tau)\varphi_1(L\tau)^\top \\ \varphi_0(L\tau)\varphi_0(L\tau)^\top - L\tau \varphi_0(L\tau)\varphi_1(L\tau)^\top \end{bmatrix} \mathrm{d}\tau \tag{52}$$
$$= \int_0^h \begin{bmatrix} \tau \varphi_1(L\tau) \left(\varphi_0(L\tau)^\top - L\tau \varphi_1(L\tau)^\top\right) \\ \varphi_0(L\tau) \left(\varphi_0(L\tau)^\top - L\tau \varphi_1(L\tau)^\top\right) \end{bmatrix} \mathrm{d}\tau \tag{53}$$
$$= \int_0^h \begin{bmatrix} \tau \varphi_1(L\tau) \\ \varphi_0(L\tau) \end{bmatrix} \mathrm{d}\tau \tag{54}$$
$$= \begin{bmatrix} h^2 \varphi_2(Lh) \\ h \varphi_1(Lh) \end{bmatrix} \tag{55}$$

where we used $L\tau\varphi_1(L\tau) = \varphi_0(L\tau) - I$, and $\partial_\tau \left[\tau^k \varphi_k(L\tau)\right] = \tau^{k-1}\varphi_{k-1}(L\tau)$. It follows that

$$HQ(h)H^\top = H \begin{bmatrix} h^2\varphi_2(Lh) \\ h\varphi_1(Lh) \end{bmatrix} = h\left(\varphi_1(Lh) - Lh\varphi_2(Lh)\right) = hI, \tag{56}$$

where we used $L\tau\varphi_2(L\tau) = \varphi_1(L\tau) - I$. □

Lemma B.2. *The prediction covariance Σ^-_{n+1} satisfies*

$$\Sigma^-_{n+1} H^\top = Q(h)H^\top. \tag{57}$$

Proof. First, since the observation model is noiseless, the filtering covariance Σ_n satisfies

$$H\Sigma_n = \begin{bmatrix} 0 & 0 \end{bmatrix}. \tag{58}$$

This can be shown directly from the correction step formula:

$$\begin{aligned}
H\Sigma_n &= H\Sigma^-_n - HK_nS_nK_n^\top && (59)\\
&= H\Sigma^-_n - H\left(\Sigma^-_n H^\top S_n^{-1}\right) S_n K_n^\top && (60)\\
&= H\Sigma^-_n - H\Sigma^-_n H^\top \left(H\Sigma^-_n H^\top\right)^{-1} S_n K_n^\top && (61)\\
&= H\Sigma^-_n - IS_nK_n^\top && (62)\\
&= H\Sigma^-_n - S_n\left(\Sigma^-_n H^\top S_n^{-1}\right)^\top && (63)\\
&= H\Sigma^-_n - S_nS_n^{-1}H\Sigma^-_n && (64)\\
&= \begin{bmatrix} 0 & 0 \end{bmatrix}. && (65)
\end{aligned}$$

Next, since the observation matrix is $H = \begin{bmatrix} -L & I \end{bmatrix}$, the filtering covariance Σ_n is structured as

$$\Sigma_n = \begin{bmatrix} I \\ L \end{bmatrix} [\Sigma_n]_{00} \begin{bmatrix} I & L^\top \end{bmatrix}. \tag{66}$$

This can be shown directly from Eq. (58):

$$\begin{bmatrix} 0 & 0 \end{bmatrix} = H\Sigma = \begin{bmatrix} -L & I \end{bmatrix} \begin{bmatrix} \Sigma_{00} & \Sigma_{01} \\ \Sigma_{10} & \Sigma_{11} \end{bmatrix} = \begin{bmatrix} \Sigma_{10} - L\Sigma_{00} & \Sigma_{11} - L\Sigma_{01} \end{bmatrix}, \tag{67}$$

and thus

$$\begin{aligned}
\Sigma_{10} &= L\Sigma_{00}, && (68)\\
\Sigma_{11} &= L\Sigma_{01} = L\Sigma_{10}^\top = L\Sigma_{00}L^\top. && (69)
\end{aligned}$$

It follows

$$\Sigma = \begin{bmatrix} \Sigma_{00} & L\Sigma_{00} \\ \Sigma_{00}L^\top & L\Sigma_{00}L^\top \end{bmatrix} = \begin{bmatrix} I \\ L \end{bmatrix} \Sigma_{00} \begin{bmatrix} I & L^\top \end{bmatrix}. \tag{70}$$

Finally, together with Lemma B.1 we can derive the result:

$$\begin{aligned}
\Sigma^-_{n+1} H^\top &= \Phi(h)\Sigma_n\Phi(h)^\top H^\top + Q(h)H^\top && (71)\\
&= \Phi(h) \begin{bmatrix} I \\ L \end{bmatrix} \bar{\Sigma}_n \begin{bmatrix} I & L^\top \end{bmatrix} \begin{bmatrix} -L^\top \\ I \end{bmatrix} + Q(h)H^\top && (72)\\
&= \Phi(h) \begin{bmatrix} I \\ L \end{bmatrix} \bar{\Sigma}_n \cdot 0 + Q(h)H^\top && (73)\\
&= Q(h)H^\top. && (74)
\end{aligned}$$

□

B.3 Proof of Proposition 2

With these results, we can now prove Proposition 2.

Proof of Proposition 2. We prove the proposition by induction, showing that the filtering means are all of the form

$$\mu_n := \begin{bmatrix} y_n \\ Ly_n + N(\tilde{y}_n) \end{bmatrix}, \tag{75}$$

where $y_n, \tilde{y}_n$ are defined as

$$\begin{aligned} \tilde{y}_0 &:= y_0, && (76)\\ \tilde{y}_{n+1} &:= \varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n), && (77)\\ y_{n+1} &:= \varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n) - h\varphi_2(Lh)\left(N(\tilde{y}_n) - N(\tilde{y}_{n+1})\right). && (78) \end{aligned}$$

This result includes the statement of Proposition 2.

Base case $n = 0$ The initial distribution of the probabilistic solver is chosen as

$$\mu_0 = \begin{bmatrix} y_0 \\ Ly_0 + N(\tilde{y}_0) \end{bmatrix}, \Sigma_0 = 0. \tag{79}$$

This proves the base case $n = 0$.

Induction step $n \to n+1$ Now, let

$$\mu_n = \begin{bmatrix} y_n \\ Ly_n + N(\tilde{y}_n) \end{bmatrix} \tag{80}$$

be the filtering mean at step n and Σ_n be the filtering covariance. The prediction mean is of the form

$$\mu_{n+1}^- = \Phi(h)\mu_n = \begin{bmatrix} y_n + h\varphi_1(Lh)(Ly_n + N(\tilde{y}_n)) \\ \varphi_0(Lh)(Ly_n + N(\tilde{y}_n)) \end{bmatrix} = \begin{bmatrix} \varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n) \\ \varphi_0(Lh)(Ly_n + N(\tilde{y}_n)) \end{bmatrix}. \tag{81}$$

The residual $\hat{z}_{n+1}$ is then of the form

$$\begin{aligned} \hat{z}_{n+1} &= E_1\mu_{n+1}^- - f(E_0\mu_{n+1}^-) && (82)\\ &= \varphi_0(Lh)(Ly_n + N(\tilde{y}_n)) - f\left(\varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n)\right) && (83)\\ &= \varphi_0(Lh)(Ly_n + N(\tilde{y}_n)) - L\left(\varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n)\right) - N\left(\tilde{y}_{n+1}\right) && (84)\\ &= \varphi_0(Lh)Ly_n + \varphi_0(Lh)N(\tilde{y}_n) - L\varphi_0(Lh)y_n - Lh\varphi_1(Lh)N(\tilde{y}_n) - N\left(\tilde{y}_{n+1}\right) && (85)\\ &= \left(\varphi_0(Lh) - Lh\varphi_1(Lh)\right)N(\tilde{y}_n) - N\left(\tilde{y}_{n+1}\right) && (86)\\ &= N(\tilde{y}_n) - N\left(\tilde{y}_{n+1}\right), && (87)\\ & && (88) \end{aligned}$$

where we used properties of the φ-functions, namely $Lh\varphi_1(Lh) = \varphi_0(Lh)$ and the commutativity $\varphi_0(Lh)L = L\varphi_0(Lh)$. With Lemma B.2, the residual covariance S_{n+1} and Kalman gain K_{n+1} are then of the form

$$\begin{aligned} S_{n+1} &= H\Sigma_{n+1}^- H^\top = HQ(h)H^\top = hI, && (89)\\ K_{n+1} &= \Sigma_{n+1}^- H^\top S_{n+1}^{-1} = Q(h)H^\top (hI)^{-1} = \begin{bmatrix} h\varphi_2(Lh) \\ \varphi_1(Lh) \end{bmatrix}. && (90) \end{aligned}$$

This gives the updated mean

$$\begin{aligned} \mu_{n+1} &= \mu_{n+1}^- - K_{n+1}\hat{z}_{n+1} && (91)\\ &= \begin{bmatrix} \varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n) \\ \varphi_0(Lh)(Ly_n + N(\tilde{y}_n)) \end{bmatrix} - \begin{bmatrix} h\varphi_2(Lh) \\ \varphi_1(Lh) \end{bmatrix}\left(N(\tilde{y}_n) - N(\tilde{y}_{n+1})\right) && (92)\\ &= \begin{bmatrix} \varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n) - h\varphi_2(Lh)\left(N(\tilde{y}_n) - N(\tilde{y}_{n+1})\right) \\ \varphi_0(Lh)(Ly_n + N(\tilde{y}_n)) - \varphi_1(Lh)\left(N(\tilde{y}_n) - N(\tilde{y}_{n+1})\right) \end{bmatrix}. && (93) \end{aligned}$$

This proves the first half of the mean recursion:

$$E_0\mu_{n+1} = \varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n) - h\varphi_2(Lh)\left(N(\tilde{y}_n) - N(\tilde{y}_{n+1})\right) = y_{n+1}. \tag{94}$$

It is left to show that

$$E_1\mu_{n+1} = Ly_{n+1} - N(\tilde{y}_{n+1}). \tag{95}$$

Starting from the right-hand side, we have

$$\begin{aligned}
&Ly_{n+1} + N(\tilde{y}_{n+1}) && (96)\\
&= L\left(\varphi_0(Lh)y_n + h\varphi_1(Lh)N(\tilde{y}_n) - h\varphi_2(Lh)\left(N(\tilde{y}_n) - N(\tilde{y}_{n+1})\right)\right) + N(\tilde{y}_{n+1}) && (97)\\
&= \varphi_0(Lh)Ly_n + Lh\varphi_1(Lh)N(\tilde{y}_n) - Lh\varphi_2(Lh)\left(N(\tilde{y}_n) - N(\tilde{y}_{n+1})\right)N(\tilde{y}_{n+1}) && (98)\\
&= \varphi_0(Lh)Ly_n + (\varphi_0(Lh) - I)N(\tilde{y}_n) - (\varphi_1(Lh) - I)\left(N(\tilde{y}_n) - N(\tilde{y}_{n+1})\right)N(\tilde{y}_{n+1}) && (99)\\
&= \varphi_0(Lh)(Ly_n + N(\tilde{y}_n)) - \varphi_1(Lh)(N(\tilde{y}_n) - N(\tilde{y}_{n+1})) && (100)\\
&= E_1\mu_{n+1}. && (101)
\end{aligned}$$

This concludes the proof of the mean recursion and thus shows the equivalence of the two recursions. □

C Proof of Proposition 3: L-stability

We first provide definitions of L-stability and A-stability, following [27, Section 8.6].

Definition 1 (L-stability)**.** *A one-step method is said to be L-stable if it is A-stable and, in addition, when applied to the scalar test-equation $\dot{y}(t) = \lambda y(t)$, $\lambda \in \mathbb{C}$ a complex constant with $\mathrm{Re}(\lambda) < 0$, it yields $y_{n+1} = R(h\lambda)y_n$, and $R(h\lambda) \to 0$ as $\mathrm{Re}(h\lambda) \to -\infty$.*

Definition 2 (A-stability)**.** *A one-step method is said to be A-stable if its region of absolute stability contains the whole of the left complex half-plane. That is, when applied to the scalar test-equation $\dot{y}(t) = \lambda y(t)$ with $\lambda \in \mathbb{C}$ a complex constant with $\mathrm{Re}(\lambda) < 0$, the method yields $y_{n+1} = R(h\lambda)y_n$, and $\{z \in \mathbb{C} : \mathrm{Re}(z) < 0\} \subset \{z \in \mathbb{C} : R(z) < 1\}$.*

Proof of Proposition 3. Both L-stability and A-stability directly follow from Remark 1: Since the probabilistic exponential integrator solves linear ODEs exactly its stability function is the exponential function, i.e. $R(z) = \exp(z)$. A-stability and L-stability then follow: Since $\mathbb{C}^- \subset \{z : |R(z)| \leq 1\}$ holds the method is A-stable. And since $|R(z)| \to 0$ as $\mathrm{Re}(z) \to -\infty$ the method is L-stable. □

D Experiment details

D.1 Burger's equation

Burger's equation is a semi-linear partial differential equation (PDE) of the form

$$\partial_t u(x,t) = -u(x,t)\partial_x u(x,t) + D\partial_x^2 u(x,t), \qquad x \in \Omega, \quad t \in [0,T], \tag{102}$$

with diffusion coefficient $D \in \mathbb{R}_+$. We discretize the spatial domain Ω on a finite grid and approximate the spatial derivatives with finite differences to obtain a semi-linear ODE of the form

$$\dot{y}(t) = D \cdot L \cdot y(t) + F(y(t)), \qquad t \in [0,T], \tag{103}$$

with N-dimensional $y(t) \in \mathbb{R}^N$, $L \in \mathbb{R}^{N\times N}$ the finite difference approximation of the Laplace operator ∂_x^2, and a non-linear part F.

More specifically, we consider a domain $\Omega = (0,1)$, which we discretize with a grid of $N = 250$ equidistant locations, thus we have $\Delta x = 1/N$. We consider zero-Dirichlet boundary conditions, that is, $u(0,t) = u(1,t) = 0$. The discrete Laplacian is then

$$[L]_{ij} = \frac{1}{\Delta x^2} \cdot \begin{cases} -2 & \text{if } i = j, \\ 1 & \text{if } i = j \pm 1, \\ 0 & \text{otherwise.} \end{cases} \tag{104}$$

The non-linear part of the discretized Burger's equation results from another finite-difference approximation of the term $u \cdot \partial_x u$, and is chosen as

$$[F(y)]_i = \frac{1}{4\Delta x} \begin{cases} y_2^2 & \text{if } i = 1, \\ y_{d-1}^2 & \text{if } i = d, \\ y_{i+1}^2 - y_{i-1}^2 & \text{else.} \end{cases} \tag{105}$$

The initial condition is chosen as

$$u(x, 0) = \sin(3\pi x)^3 (1 - x)^{3/2}. \tag{106}$$

We consider an integration time-span $t \in [0, 1]$, and choose a diffusion coefficient $D = 0.075$.

D.2 Reaction-diffusion model

The reaction-diffusion model presented in the paper, with logistic reaction term, has been used to describe the growth and spread of biological populations [22]. It is given by a semi-linear PDE

$$\partial_t u(x, t) = D\partial_x^2 u(x, t) + R(u(x, t)), \qquad x \in \Omega, \quad t \in [0, T], \tag{107}$$

where $D \in \mathbb{R}_+$ is the diffusion coefficient and $R(u) = u(1 - u)$ is a logistic reaction term. We discretize the spatial domain Ω on a finite grid and approximate the spatial derivatives with finite differences, and obtain a semi-linear ODE of the form

$$\dot{y}(t) = D \cdot L \cdot y(t) + R(y(t)), \qquad t \in [0, T], \tag{108}$$

with N-dimensional $y(t) \in \mathbb{R}^N$, $L \in \mathbb{R}^{N \times N}$ the finite difference approximation of the Laplace operator, and the reaction term R is as before but applied element-wise.

We again consider a domain $\Omega = (0, 1)$, which we discretize on a grid of $N = 100$ points. This time we consider zero-Neumann conditions, that is, $\partial_x u(0, t) = \partial_x u(1, t) = 0$. Including these directly into the finite-difference discretization, the discrete Laplacian is then

$$[L]_{ij} = \frac{1}{\Delta x^2} \cdot \begin{cases} -1 & \text{if } i = j = 1 \text{ or } i = j = d, \\ -2 & \text{if } i = j, \\ 1 & \text{if } i = j \pm 1, \\ 0 & \text{otherwise.} \end{cases} \tag{109}$$

The initial condition is chosen as

$$u(x, 0) = \frac{1}{1 + e^{30x - 10}}. \tag{110}$$

The discrete ODE is then solved on a time-span $t \in [0, 2]$, and we choose a diffusion coefficient $D = 0.25$.

Diffusion Tempering Improves Parameter Estimation with Probabilistic Integrators for Ordinary Differential Equations

Jonas Beck [1 2] **Nathanael Bosch** [2] **Michael Deistler** [2] **Kyra L. Kadhim** [1 2] **Jakob H. Macke** [2] **Philipp Hennig** [2] **Philipp Berens** [1 2]

Abstract

Ordinary differential equations (ODEs) are widely used to describe dynamical systems in science, but identifying parameters that explain experimental measurements is challenging. In particular, although ODEs are differentiable and would allow for gradient-based parameter optimization, the nonlinear dynamics of ODEs often lead to many local minima and extreme sensitivity to initial conditions. We therefore propose diffusion tempering, a novel regularization technique for probabilistic numerical methods which improves convergence of gradient-based parameter optimization in ODEs. By iteratively reducing a noise parameter of the probabilistic integrator, the proposed method converges more reliably to the true parameters. We demonstrate that our method is effective for dynamical systems of different complexity and show that it obtains reliable parameter estimates for a Hodgkin–Huxley model with a practically relevant number of parameters.

1. Introduction

Ordinary differential equations (ODEs) are ubiquitous across science, as they often provide accurate models for physical processes and mechanisms underlying dynamical systems. ODEs can, for example, be used to model the motion of a pendulum, the dynamics of predator and prey populations (Hethcote, 2000) or the action potentials of neurons (Hodgkin & Huxley, 1952). Numerical algorithms for the (approximate) solution of ODEs are well established (Hairer et al., 1987), but identifying parameters of the ODE such that it matches observations can be challenging (Prinz et al., 2003; Kravtsov, 2020). To overcome this, multiple methods have been developed, such as grid (Prinz et al., 2003) or random searches (Taylor et al., 2009), simulated annealing (Kirkpatrick et al., 1983), genetic algorithms (Ben-Shalom et al., 2012), Bayesian methods (e.g., approximate Bayesian computation (Marjoram et al., 2003), Markov-Chain Monte-Carlo (Neal, 1993), or simulation-based inference (Cranmer et al., 2020)). But, these methods can often require large simulation budgets (Cranmer et al., 2020).

By incorporating gradient information, gradient descent provides the potential to be vastly more simulation efficient and, as demonstrated by Neural ODEs (NODEs), has the potential to scale to millions of parameters (Chen et al., 2018). Unfortunately, for many real-world ODEs, gradient-based optimization is challenging: ODEs often have highly nonlinear dynamics, several local minima, and are sensitive to initial conditions (Cao et al., 2011; Dass et al., 2017). In fact, even for models with few parameters, gradient-based optimization can get stuck in local minima, leading to poor performance compared to competing gradient-free methods such as genetic algorithms (Hazelden et al., 2023).

Probabilistic numerical integrators have also been proposed for parameter inference in ODEs (Tronarp et al., 2022). Unlike 'traditional' non-probabilistic ODE solvers which usually return only a single (potentially very crude) ODE solution, probabilistic numerical methods return a posterior distribution over the ODE solution, accounting for numerical uncertainty (Hennig et al., 2022). Parameters can then be estimated by maximizing their likelihood, which thanks to automatic differentiation is amenable to first order methods. This approach has been termed "Physics-Enhanced Regression for Initial Value Problems", or Fenrir for short.

Here, we develop *diffusion tempering* for probabilistic integrators to improve gradient-based parameter inference in ODEs. The technique is based on previous observations that Fenrir can effectively 'smooth out' the loss surface of ODEs (Tronarp et al., 2022). We therefore propose to solve consecutive optimization problems using the Fenrir likelihood. We start with a very smooth loss surface, which yields poor fits to data, but lets the optimizer avoid local minima. By successively solving less and less smooth problems informed by

[1]Hertie Institute for AI in Brain Health, University of Tübingen, Tübingen, Germany [2]AI Center, University of Tübingen, Tübingen, Germany. Correspondence to: Jonas Beck <jonas.beck@uni-tuebingen.de>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

previous parameter estimates, we more reliably converge in the global optimum. This enables gradient-based maximum likelihood estimation for models with a practically relevant number of parameters such as Pospischil et al. (2008).

We first demonstrate that our method is robust to local minima for a simple pendulum. We then show that it produces more reliable parameter estimates than both classical least-squares regression and the original Fenrir method by Tronarp et al. (2022) for several models of growing complexity, even in regimes in which gradient-based parameter inference is challenging.

2. Parameter inference in ODEs

Consider an initial value problem (IVP), given by an ODE

$$\dot{y}_\theta(t) = f_\theta\big(y_\theta(t), t\big)\,, \qquad t \in [0, T], \tag{1}$$

with vector field f_θ with parameters θ, and initial value $y_\theta(0) = y_0$. In this paper, we are concerned with estimating the true ODE parameters θ from a set of noisy observations of the solution $y_\theta(t)$, of the form

$$u_i = u(t_i) = H y_\theta(t_i) + \epsilon_i, \qquad i = 1, \ldots, N, \tag{2}$$

where H is a measurement matrix and $\epsilon_i \sim \mathcal{N}(0, R)$ is Gaussian noise with covariance R. We denote the set of observations by $\mathcal{D} = \{u_i\}_{i=1}^N$, and the set of observation times by $\mathbb{T}_\mathcal{D} = \{t_i\}_{i=1}^N$.

For a given parameter θ, the true ODE solution $y_\theta(t)$ is uniquely defined and thus the true marginal likelihood is

$$\mathcal{M}(\theta) = \mathcal{L}_\mathcal{D}(y_\theta) = \prod_{i=1}^N \mathcal{N}(u_i; H y_\theta(t_i), R), \tag{3}$$

where $\mathcal{L}_\mathcal{D}$ denotes the likelihood functional. Then, the parameter θ can be estimated from the data $\mathcal{D}$ by maximizing the marginal likelihood $\mathcal{M}(\theta)$, that is $\hat{\theta}_{\text{MLE}} = \arg\max_\theta \mathcal{M}(\theta)$. Unfortunately, the true solution y_θ is not generally known and cannot be computed analytically, and thus the true marginal likelihood is intractable.

In the following, we discuss two approaches to make inference tractable by approximating the marginal likelihood. First, note that we can rewrite $\mathcal{M}(\theta)$ as an integral

$$\mathcal{M}(\theta) = \int \mathcal{L}_\mathcal{D}(y)\delta(y - y_\theta)\,\mathrm{d}y. \tag{4}$$

This clarifies that the intractable part of the marginal likelihood is the Dirac measure $\delta(y - y_\theta)$. Thus, a natural approach to approximate the marginal likelihood is to replace the true solution distribution by a suitable approximation.

2.1. Classic Numerical Integration

A standard approach to parameter inference in IVPs approximates the true IVP solution $y_\theta(t)$ with a numerical solution $\hat{y}_\theta(t)$, obtained from a classic numerical IVP solver such as the well-known Runge–Kutta (RK) method (Hairer et al., 1987). The marginal likelihood then becomes

$$\mathcal{M}(\theta) \approx \widehat{\mathcal{M}}_{\text{RK}}(\theta) := \int \mathcal{L}_\mathcal{D}(y)\delta(y - \hat{y}_\theta)\,\mathrm{d}y, \tag{5a}$$

$$= \prod_{i=1}^N \mathcal{N}(u_i; H\hat{y}_\theta(t_i), R). \tag{5b}$$

Maximizing $\widehat{\mathcal{M}}_{\text{RK}}(\theta)$ is equivalent to minimizing the mean squared error

$$L_{\text{RK}}(\theta) = \frac{1}{N}\sum_{i=1}^N \big\|H\hat{y}_\theta(t_i) - u_i\big\|_2^2. \tag{6}$$

This approach is also known as non-linear least-squares regression (Bard, 1974).

2.2. Probabilistic Numerical Integration

Probabilistic numerical methods reformulate numerical problems as probabilistic inference (Hennig et al., 2022). In the context of IVPs, probabilistic numerical ODE solvers make the time-discretization of numerical methods explicitly part of their problem statement, and aim to compute a posterior distribution of the form

$$p\left(y_\theta(t) \;\middle|\; y_\theta(0) = y_0, \{\dot{y}_\theta(t) = f_\theta\big(y_\theta(t), t\big)\}_{t\in\mathbb{T}_{\text{PN}}}\right), \tag{7}$$

where $\mathbb{T}_{\text{PN}}$ is the chosen time-discretization. Assuming fixed f, y_0, and $\mathbb{T}_{\text{PN}}$, we denote this posterior more compactly with $p_{\text{PN}}(y(t) \mid \theta)$. We call this object, and approximations thereof, a *probabilistic numerical ODE solution*

This object then presents itself as a natural candidate to approximate the true solution distribution $\delta(y - y_\theta)$. We obtain the PN-approximated marginal likelihood

$$\widehat{\mathcal{M}}_{\text{PN}}(\theta) = \int \mathcal{L}_\mathcal{D}(y)\; p_{\text{PN}}(y(t) \mid \theta)\,\mathrm{d}y. \tag{8}$$

The remaining question is how to compute this quantity. In the following, we consider probabilistic ODE solvers based on Bayesian filtering and smoothing, which have been shown to be a particularly efficient class of methods for probabilistic numerical simulation (Tronarp et al., 2021; 2019; Kersting et al., 2020; Schober et al., 2019), and we review the PN-approximated marginal likelihood by Tronarp et al. (2022) which builds the base of our proposed approach.

2.2.1. Probabilistic Numerical IVP Solvers

Filtering-based probabilistic numerical ODE solvers formulate the probabilistic numerical ODE solution (Equa-

tion (7)) as a Bayesian state estimation problem, described by a Gauss–Markov prior given by

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0)\,, \tag{9a}$$

$$x(t_n) \mid x(t_{n-1}) \sim \mathcal{N}\Big(\Phi_n x(t_{n-1}), \kappa^2 Q_n\Big)\,, \tag{9b}$$

$$y^{(m)}(t_n) = E_m^\intercal x(t_n), \quad m = 0, 1, \ldots, q, \tag{9c}$$

and a likelihood and data model, or *information operator*, z (Tronarp et al., 2021; Cockayne et al., 2019), of the form

$$z(t_n) \mid x(t_n) \sim \delta\Big(E_1 x(t_n) - f_\theta\big(E_0 x(t_n), t\big)\Big)\,, \tag{9d}$$

$$z(t_n) \coloneqq 0, \tag{9e}$$

which maps solutions of the initial value problem to the zero function on the chosen grid $\mathbb{T}_\mathcal{D}$.

Here, the state $x(t)$ models the solution $y(t)$ of the IVP together with its q first derivatives, and in the following we consider q-times integrated Wiener process priors for which the transition matrices Φ_n and Q_n are known in closed form (Kersting et al., 2020). E_m are selection matrices for the mth derivative. The initial mean μ_0 is chosen to match the initial condition of the given IVP exactly, and the initial covariance Σ_0 is set to zero (Krämer & Hennig, 2020). Finally, κ is a prior hyperparameter which controls the uncertainty of the prior distribution, known as the *diffusion*.

Equation (9) is a well known problem in Bayesian filtering and smoothing, and its solution can be approximated efficiently with extended Kalman filtering and smoothing (Särkkä, 2013). For a given ODE parameter θ and diffusion hyperparameter κ, we obtain a Gauss–Markov posterior distribution $\hat{p}_{\mathrm{PN}}(y(t) \mid \theta, \kappa)$.

2.2.2. PN-approximated Marginal Likelihood

By inserting the PN posterior $\hat{p}_{\mathrm{PN}}(y(t) \mid \theta, \kappa)$, into the marginal likelihood, we obtain the marginal likelihood model by Tronarp et al. (2022) of the form

$$\widehat{\mathcal{M}}_{\mathrm{FN}}(\theta, \kappa) = \int \mathcal{L}_\mathcal{D}(y)\, \hat{p}_{\mathrm{PN}}(y(t) \mid \theta, \kappa)\, \mathrm{d}y. \tag{10}$$

This quantity can again be computed efficiently with Kalman filtering: Since the posterior distribution $\hat{p}_{\mathrm{PN}}$ has known backward transition densities of the form (Tronarp et al., 2022, Proposition 3.1)

$$\hat{p}_{\mathrm{PN}}\big(x(t_{n-1}) \mid x(t_n)\big) = \mathcal{N}\big(G_n^\theta x(t_n) + \zeta_n^\theta, \kappa^2 P_n^\theta\big), \tag{11}$$

$\widehat{\mathcal{M}}_{\mathrm{FN}}(\theta, \kappa)$ can be computed by running a Kalman filter backwards in time on the state-space model

$$x(t_N) \sim \mathcal{N}\big(x(t_N); \xi_\theta(t_N), \kappa^2 \Lambda_\theta(t_N)\big), \tag{12a}$$

$$x(t_{n-1}) \mid x(t_n) \sim \mathcal{N}\big(G_n^\theta x(t_n) + \zeta_n^\theta, \kappa^2 P_n^\theta\big), \tag{12b}$$

$$u(t_n) \mid x(t_n) \sim \mathcal{N}\big(H E_0 x(t_n), R_\theta\big), \quad t_n \in \mathbb{T}_\mathrm{D}, \tag{12c}$$

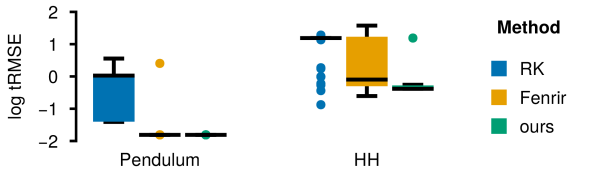


Figure 1. For a pendulum model, Fenrir produces parameter estimates with much lower trajectory mean squared errors (Definition A3.1) than RK least-squares regression. However, for the more complex HH model, the tRMSE is very similar for both RK and Fenrir. Our proposed method is able to produce much better estimates for both problems.

via the prediction error decomposition (Schweppe, 1965). The quantities $G_n^\theta, \zeta_n^\theta, P_n^\theta$ can all be computed during the forward-pass of the probabilistic ODE solver; for the full details refer to Tronarp et al. (2022).

Finally, Tronarp et al. (2022) propose to jointly optimize $\widehat{\mathcal{M}}_{\mathrm{FN}}(\theta, \kappa)$ for both the ODE parameters of interest θ and the diffusion hyperparameter κ, that is

$$\hat{\theta}, \hat{\kappa} = \arg\max_{\theta, \kappa} \widehat{\mathcal{M}}_{\mathrm{FN}}(\theta, \kappa)\,. \tag{13}$$

Then $\hat{\theta}$ is returned as the maximum likelihood estimate. We refer to this ODE parameter inference method as Fenrir.

2.3. Shortcomings

In order to ensure that an optimizer has converged at the global as opposed to a local optimum, in practice, the optimization has to be run multiple times with different initial conditions. A reliable optimizer is therefore highly desirable since a higher reliability also means less restarts to obtain a good parameter estimate. This in turn can make optimizing more complex models feasible, since we can afford many more restarts at the same cost.

Tronarp et al. (2022) show that learning θ and κ jointly leads to more reliable parameter estimates for a pendulum compared to RK least-squares regression (Figure 1) for which the optimization often converges in local minima, i.e. the constant zero function. However, a pendulum is a fairly low dimensional problem. When we evaluated Fenrir for more complex ODEs such as the HH model, we found that this gain in reliability was vastly smaller in systems with more challenging likelihood functions. In these cases, many restarts are needed to reach the global optimum, which makes Fenrir equally unfeasible as RK-based least squares for gradient descent-based parameter estimation in HH models with a practically relevant number of parameters.

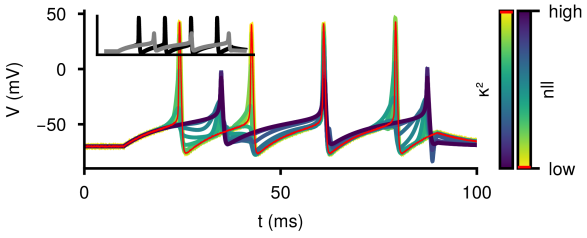


Figure 2. PN posterior means for a range of κ for a HH model. The parameters that generated the observation (inset, black) are different from the parameters of the ODE (inset, grey). For low κ the mean closely adheres to the ODE solution, while for high values, it fits the observation much better. The maximum likelihood κ is highlighted in red.

Algorithm 1 Diffusion tempering

Input: initial parameter θ_0, tempering schedule $\mathcal{T}$, number of iterations m, Optimizer OPT, Fenrir likelihood $\widehat{\mathcal{M}}(\theta, \kappa)$
for $i = 0$ **to** $m - 1$ **do**
  $\kappa_i = \mathcal{T}(i)$
  $\theta_{i+1} = \mathrm{OPT}\left(\widehat{\mathcal{M}}_{\mathrm{PN}}, (\theta_i, \kappa_i)\right)$
end for
$\theta_{est} = \theta_m$
Return: θ_{est}

3. Method

In the following, we attribute the observed shortcomings of Fenrir to the way the diffusion hyperparameter κ was treated and we propose an alternative method which enables effective parameter inference for more complicated loss functions and higher dimensional parameter spaces (Figure 1).

3.1. Diffusion as Regularization

To gain a better understanding of the mechanisms that lead to better optimization performance in Fenrir, we studied the effect of the diffusion hyperparameter in more detail.

We noticed that for a given set of model parameters θ, the loss was lower for high values of κ than for low values (Figure 2). Furthermore, when κ was set high, the mean of the PN posterior interpolated the observed data points, while if set low, it approximated the IVP solution much better. This implies for this particular parameter set, that the maximum likelihood estimate of the PN posterior (Equation (13)) "favors" a good interpolation of the data over a good approximation of the IVP solution. This is an issue for the success of parameter estimation, since we are trying to identify an IVP that reproduces the observed data well, as opposed to finding just a good interpolant of the data.

Why does this happen? Since κ scales the gain of the diffusion, it also determines the uncertainty of the ODE solution for a given θ. A high diffusion leads to a less concentrated prior for the subsequent Gauss–Markov regression. For a broad prior, many observations have a similar likelihood, hence a PN posterior whose mean interpolates the data scores favorably (Figure 2). For a low diffusion, the variance of the prior is very tightly concentrated around the ODE solution, meaning only data that closely matches the trajectory will score a high likelihood, in turn leading to interpolation of the ODE solution (Figure 2).

Based on these observations, we propose to re-interpret the diffusion parameter in Fenrir as a optimization hyperparameter which determines how the data should be "weighted" with respect to the solution of the ODE: Fixing κ regularizes the solution of the IVP on the observed data. In particular, we show in Figure 1 and argue in the following that tempering the diffusion leads to improved convergence.

3.2. Diffusion tempering

Above we argued why κ should not be optimized jointly with θ, but treated as a hyperparameter during optimization, since it can trade-off confidence in the data vs. the IVP solution (Fig. 2). Accordingly, we propose to put κ on a decreasing schedule. Starting with a high κ, Fenrir does not have much confidence in the accuracy of its physics-informed prior and therefore most ODE parameters yield a decent likelihood for the observed data. By steadily lowering the diffusion, we decrease the uncertainty of the prior, which means there are less ways for an ODE solution to fit the data. Finally, when the prior has minimal uncertainty presumably only the "true" ODE fits the data.

We call this procedure diffusion tempering (Algorithm 1). We start by defining the number of iterations m and a tempering schedule $\mathcal{T}$, which can be a list of κ or any function that takes a positive integer and returns a value for the diffusion hyperparameter. We set an initial parameter θ_0, i.e. by drawing it from a pre-specified distribution and set $\widehat{\mathcal{M}}_{\mathrm{PN}}(\theta_0, \kappa = \mathcal{T}(0))$. We run an optimizer of choice until convergence to obtain a new parameter estimate. Then, we obtain the next κ from $\mathcal{T}$ and repeat the optimization with the updated estimate of θ, returning the final θ as our parameter estimate. In our case, we initialized the parameters uniformly distributed and opted for a tempering schedule of $\mathcal{T}(i, \kappa_0) = 10^{(\kappa_0 - i)}$, $\kappa_0 = 20$ with a schedule length of 21. This is equivalent to linearly decreasing $\log_{10}(\kappa)$ from 20 to 0.

4. Experiments

We first investigate the effect of diffusion tempering for a pendulum with a single parameter, and show even in this

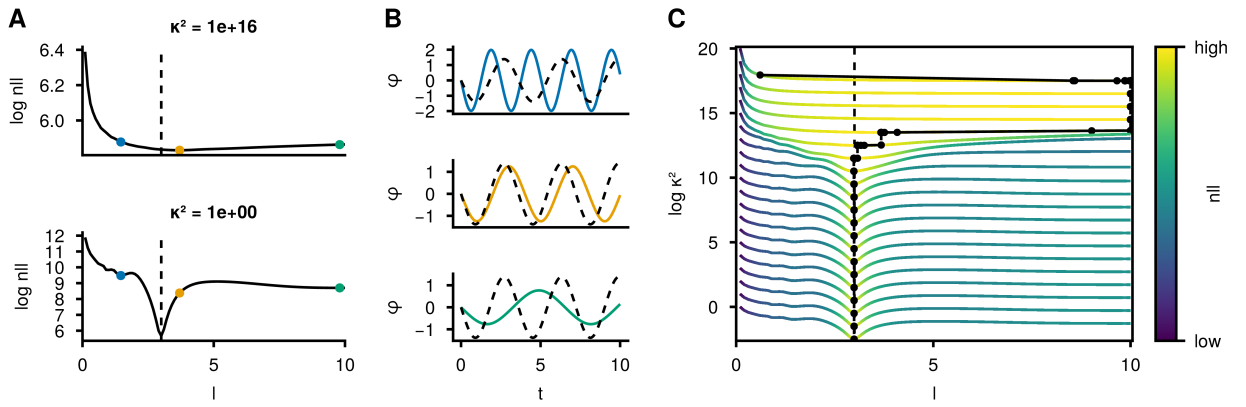


Figure 3. The effect of diffusion tempering on the marginal likelihood $\mathcal{M}(\theta, \kappa)$ of a pendulum with parameter l. **A** The negative log likelihood (nll) for a high κ is very smooth with only one shallow global optimum (yellow). The likelihood for low κ has one sharp global minimum at the true parameters (dashed), but also other local minima (blue, green). **B** IVP solutions for the different local optima marked in A and the true solution (dashed). **C** Diffusion tempering for an exemplary optimization run (black): Optimization starts with a high κ (top) on a very smooth loss landscape. κ is then progressively lowered, revealing a shallow global optimum near the true parameters ($\kappa = 10^{16}$). Further lowering κ finally reveals a sharp global optimum and the optimization converges correctly. By starting with the parameters of the previous optimization, diffusion tempering ensures that the optimization closes in on the correct global optimum.

comparatively simple case that it converges more reliably around the true parameters. We then demonstrate that diffusion tempering also improves Fenrir's convergence for complex ODE models such as the HH model (Hodgkin & Huxley, 1952). Finally, we show that diffusion tempering enables parameter estimation for HH models in parameter regimes where learned diffusions and least-squares methods stop working altogether.

4.1. Exploration of a simple case: The 1D Pendulum

We first demonstrate our algorithm for a pendulum. While this is a relatively simple problem by construction, it is already challenging for classical simulation-based methods such as Runge–Kutta least-squares approaches (Bard, 1974), which tend to fail for high frequencies and are sensitive to initialization (Benson, 1979). Furthermore, with only a single free parameter, the pendulum length l, the effect of tempered optimization can be illustrated well (Figure 3). Details on the full dynamics and the parameterisation are provided in Appendix A5.1.

We evaluated the Fenrir marginal likelihood at different pendulum lengths for a high diffusion constant. This confirmed our theory from Section 3.1 that a high κ indeed leads to similar likelihoods for most parameters (Figure 3A). Additionally, with increasing length l the loss also increased, which discourages the zero-function ($l \to \infty$) which would otherwise be a local optimum. However, while the global optimum was roughly in the right place, it was not yet located at the true l. This was in stark contrast to when we evaluated the marginal likelihood for a low diffusion. Here the global optimum correctly identified the true l, but the loss landscape had additional local optima (Figure 3B). When we repeated this process for every κ that our tempering schedule visited, we observed that solving separate optimization problems at subsequently lower values of κ exploited the smooth nature of the loss landscape to skip over later forming local optima. While earlier minima might not have been accurate, they provided a better initialization for following optimizations at lower κ (Figure 3C). Diffusion tempering first interpolated the observed data, but later put more and more weight on a good ODE solution. This avoided local minima that provided a likelihood that explained neither the ODE or the data well. For the pendulum, diffusion tempering converged as often as Fenrir but three times as often as RK least-squares regression in a ball around the true parameters (Table 1, PD).

4.2. Exploration of a complex case: The Hodgkin–Huxley Model

We next turned to a more complex system of ODEs, the Hodgkin–Huxley model for neural membrane biophysics (Hodgkin & Huxley, 1952). It models the change in voltage across a cell's membrane $V(t)$ in response to an external input $I_{\text{in}}(t)$ as different ionic currents flowing through a circuit (Figure 4A). For the detailed equations and parameterisation see Appendix A5.3.

This model is an interesting case study because there is considerable scientific interest in identifying the conductances of these ionic currents from voltage recordings to inform neuron simulations. All-or-none action potentials (spikes)

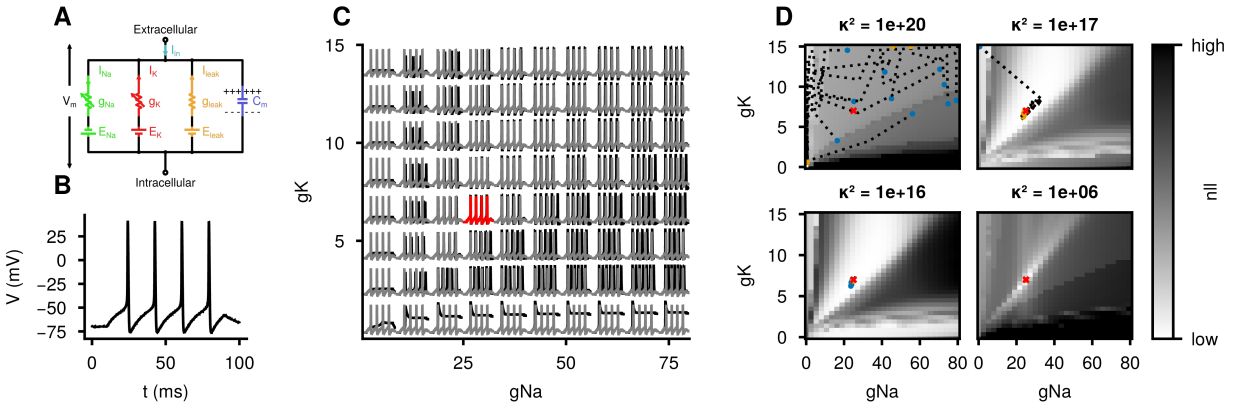


Figure 4. Parameter estimation for the sodium g_{Na} and potassium g_K conductance of a HH model. **A** Circuit modeling components of a neuronal cell as electrical elements. The lipid bilayer acts as a capacitor (C_m). Ion channels are represented by resistors. Sodium and potassium conductances are voltage dependent (g_{Na},g_K), and the leak conductance is constant (g_{leak}). The electrochemical gradients driving the flow of ions can be represented as voltage sources (E_{Na},E_K,E_{leak}). **B** Noisy observation of the membrane voltage. **C** Solutions of the ODE for different combinations of parameters (black) compared to the true parameters (grey). The true parameters are highlighted in red. **D** Parameter optimization during four different stages of diffusion tempering for a random subset of initializations. Optimization trajectories (dotted) in each likelihood landscape are shown from start (blue) to convergence (orange). Very high loss values were clipped for better visual clarity. Plots for the full schedule are provided in Figure A1, with corresponding solutions in Figure A2.

and steep voltage gradients make this challenging, which is why grid (Prinz et al., 2003) or random searches (Taylor et al., 2009), genetic algorithms (Druckmann et al., 2007; Ben-Shalom et al., 2012; Achard & De Schutter, 2006) and simulation-based Inference (Gonçalves et al., 2020; Lueckmann et al., 2017) are common tools.

For our experiments, we generated observations from the HH model with a square wave depolarizing current of 210 pA between 10ms and 90 ms as stimulus. In the initial experiments, we only considered two free parameters, g_{Na} and g_{K}, for which we generated data at 25 mS/cm^2 and 7 mS/cm^2, respectively. The initial voltage was chosen as $V(0) = -70$ mV and the membrane voltage was simulated for a time interval of 100 ms, leading to a spiking trace with four action potentials (Figure 4B). See Appendix A5.3 for details on how we initialized the gating variables.

Even with just two free parameters, the HH model displayed highly complex patterns of activity ranging from non-spiking solutions in regimes of low sodium or potassium conductance to spike trains of different amplitudes, frequencies or phase (Figure 4C). Additionally, small changes in the parameter space could sometimes lead to dramatic changes in the solution, like additional spikes appearing.

To explore this and the effect on the inference more systematically, we computed the loss landscape of the model with respect to observed data for different diffusion parameters (Figure 4D). We observed a similar behavior as in Section 4.1 for the pendulum. For a high diffusion parameter, the loss was very smooth and sloped only in one direction, leading the optimization runs to converge at the non-spiking response simply interpolating the baseline of the action potentials (Figure 4D top, left). This likely happened because the action potentials were narrow, consisting only of a few measurements each. Hence, their effect on the loss was not strong enough to make these solutions prohibitively costly, as the high diffusion parameter made the inference focus on data interpolation. For lower diffusion parameters, a large basin around the true parameters appeared, which the parameter estimates started to converge to (Figure 4D top, right). For very low diffusion values, the basin then morphed into a much sharper global optimum around the true parameters which attracted the estimates that were previously collecting in the basin (Figure 4D bottom, left).

4.3. Systematic performance benchmarking against alternative methods

To demonstrate the effect of tempering the diffusion more systematically, we compared to both learning and fixing κ in the following. More specifically we studied: (i) non-linear least-squares regression with Runge–Kutta (RK), (ii) Fenrir with learned diffusion as proposed by Tronarp et al. (2022), and (iii) Fenrir with a range of fixed diffusion parameters: We explore a low, a high and the best κ according to a grid search over all values visited during tempering (for details see Figure A3) as well as the maximum likelihood κ at the true parameters (optimal).

We considered an optimization run as correctly converged if the final parameter estimate deviated at most 5% from the

Table 1. Comparison of different methods, models and model sizes for parameter estimation. LV: Lottka-Volterra, PD: Pendulum, HH_x Hodgkin–Huxley with x compartments, D_θ: Number of parameters, N_θ: The number of correctly identified parameters, CONV: correctly converged. *In only a few cases the optimization diverged, which lead to NaNs in the loss and tRMSEs orders of magnitude above the mean. We excluded these runs from the statistics marked with a *.*

ODE	D_θ	ALG	ITER	pRMSE	CONV	tRMSE	N_θ
PD	1	FENRIR	74.12 ± 11.55	0.01 ± 0.08	0.99	0.04 ± 0.26	0.99 ± 0.10
PD	1	RK	$\mathbf{34.11 \pm 18.45}$	1.42 ± 1.09	0.32	0.98 ± 0.72	0.32 ± 0.47
PD	1	OURS	303.89 ± 12.43	$\mathbf{0.00 \pm 0.00}$	**1.00**	$\mathbf{0.02 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
PD	1	OURS+	92.21 ± 0.95	$\mathbf{0.00 \pm 0.00}$	**1.00**	$\mathbf{0.02 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
LV	2	FENRIR	$\mathbf{60.68 \pm 36.73}$	1.41 ± 1.01	0.20	2.84 ± 2.71	0.40 ± 0.80
LV	2	RK	97.15 ± 146.95	1.35 ± 1.11	0.24	2.35 ± 1.32	0.48 ± 0.86
LV	2	OURS	411.18 ± 151.25	0.41 ± 0.82	0.77	$^*0.54 \pm 1.14$	1.54 ± 0.85
LV	2	OURS+	132.43 ± 31.01	$\mathbf{0.35 \pm 0.77}$	**0.79**	$^*\mathbf{0.43 \pm 1.03}$	$\mathbf{1.59 \pm 0.81}$
LV	4	FENRIR	$\mathbf{112.45 \pm 47.24}$	1.31 ± 0.94	0.23	$\mathbf{4.18 \pm 13.65}$	1.07 ± 1.70
LV	4	RK	271.49 ± 151.05	1.09 ± 0.76	0.24	6.57 ± 34.45	0.97 ± 1.71
LV	4	OURS	727.15 ± 482.93	0.81 ± 0.88	0.43	$^*10.41 \pm 69.37$	1.76 ± 1.98
LV	4	OURS+	292.05 ± 139.98	$\mathbf{0.74 \pm 0.82}$	**0.44**	$^*13.77 \pm 79.08$	$\mathbf{1.79 \pm 1.98}$
HH_1	1	FENRIR	46.96 ± 19.08	0.38 ± 0.67	0.68	7.85 ± 10.70	0.68 ± 0.47
HH_1	1	RK	$\mathbf{43.30 \pm 43.45}$	0.42 ± 0.48	0.57	7.54 ± 8.26	0.57 ± 0.50
HH_1	1	OURS	382.08 ± 32.19	$\mathbf{0.00 \pm 0.00}$	**1.00**	$\mathbf{0.43 \pm 0.02}$	$\mathbf{1.00 \pm 0.00}$
HH_1	1	OURS+	116.73 ± 7.11	$\mathbf{0.00 \pm 0.00}$	**1.00**	0.43 ± 0.11	$\mathbf{1.00 \pm 0.00}$
HH_1	2	FENRIR	110.04 ± 61.70	0.20 ± 0.37	0.75	5.89 ± 10.15	1.53 ± 0.83
HH_1	2	RK	$\mathbf{54.02 \pm 62.60}$	0.28 ± 0.45	0.72	4.88 ± 7.58	1.44 ± 0.90
HH_1	2	OURS	696.22 ± 78.10	$\mathbf{0.00 \pm 0.00}$	**1.00**	$\mathbf{0.42 \pm 0.04}$	$\mathbf{2.00 \pm 0.00}$
HH_1	2	OURS+	197.81 ± 40.38	0.04 ± 0.19	0.96	1.08 ± 3.20	1.92 ± 0.39
HH_1	3	FENRIR	$\mathbf{122.15 \pm 49.74}$	0.59 ± 0.65	0.51	9.31 ± 9.63	1.53 ± 1.51
HH_1	3	RK	223.55 ± 117.31	0.90 ± 0.25	0.03	14.33 ± 4.01	0.13 ± 0.56
HH_1	3	OURS	676.33 ± 150.72	$\mathbf{0.01 \pm 0.10}$	**0.99**	$\mathbf{0.60 \pm 1.51}$	$\mathbf{2.97 \pm 0.30}$
HH_1	6	FENRIR	$\mathbf{108.06 \pm 108.49}$	13.36 ± 6.97	0.00	26.21 ± 7.38	1.05 ± 0.22
HH_1	6	RK	210.26 ± 120.86	12.27 ± 6.87	0.00	16.80 ± 3.81	1.18 ± 0.39
HH_1	6	OURS	2159.60 ± 532.55	$\mathbf{10.36 \pm 7.72}$	0.00	$^*\mathbf{15.20 \pm 5.41}$	$\mathbf{1.21 \pm 0.46}$
HH_2	4	FENRIR	286.54 ± 205.37	0.28 ± 0.44	0.68	12.00 ± 17.37	2.80 ± 1.78
HH_2	4	RK	$\mathbf{136.50 \pm 200.20}$	0.43 ± 0.56	0.50	7.98 ± 9.86	2.08 ± 1.81
HH_2	4	OURS	1492.03 ± 335.17	$\mathbf{0.00 \pm 0.00}$	**1.00**	$\mathbf{0.60 \pm 0.01}$	$\mathbf{4.00 \pm 0.00}$
HH_2	6	FENRIR	$\mathbf{221.28 \pm 144.56}$	0.62 ± 0.72	0.50	13.01 ± 13.10	3.06 ± 2.96
HH_2	6	RK	390.34 ± 195.85	0.88 ± 0.22	0.00	19.36 ± 5.54	0.31 ± 0.75
HH_2	6	OURS	1525.57 ± 448.56	$\mathbf{0.12 \pm 0.32}$	**0.88**	$\mathbf{3.01 \pm 6.70}$	$\mathbf{5.28 \pm 1.96}$

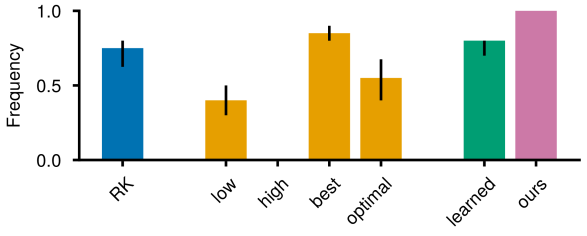


Figure 5. Convergence for a two parameter HH model: Diffusion tempering converges more reliably than than non-linear least-squares regression using a RK solver and Fenrir with a learned and the best single κ. Histogram shows the medians with black bars indicating quartiles for 100 runs split into groups of 10.

true parameters (for details see Appendix A3).

We observed that for a two parameter HH model, learning the diffusion hyperparameter performed on par with the classical RK method, suggesting Tronarp et al. (2022)'s approach did not provide any benefit in this setting (Figure 5). Moreover, the same performance could be achieved if the diffusion parameter was just fixed at a well selected value. Interestingly, the κ that maximized the likelihood at the true parameters performed worse than when the diffusion was just fixed at a low value (Figure 5). Nonetheless, our method still yielded the best convergence, implying that active adaptation of κ during optimization is imperative.

To show that diffusion tempering worked for a variety of different model sizes and complexities, we benchmarked our algorithm against RK least-squares regression and the original method of Tronarp et al. (2022) for the 1D pendulum (Appendix A5.1), the Lotka-Volterra equations (Appendix A5.2) with two and four free parameters as well as multiple HH models with one (HH_1) and two compartments (HH_2) and one to six free different parameters (Appendix A5.3). For 100 different initializations, we evaluated the algorithms based on trajectory mean squared error (tRMSE, Definition A3.1), relative parameter mean squared error (pRMSE,

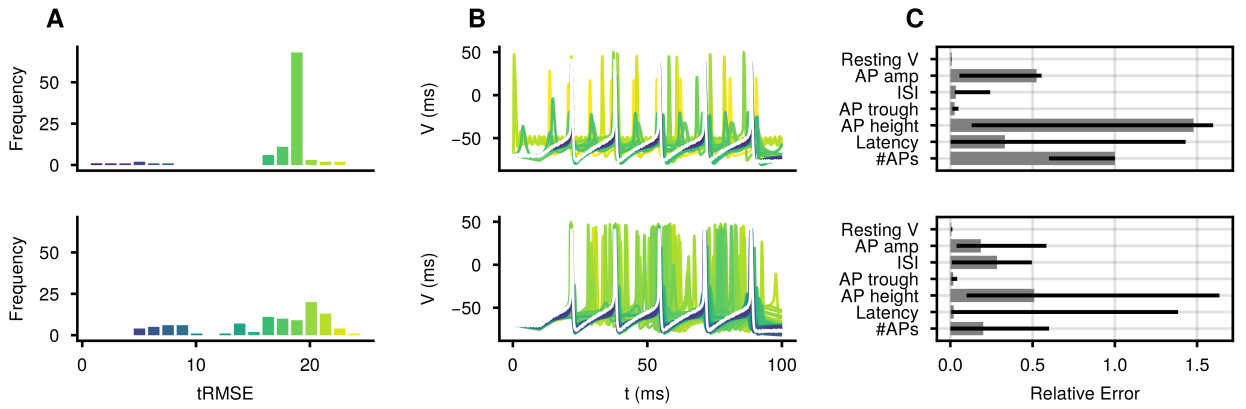


Figure 6. Parameter inference for a six parameter HH model for 100 initializations of RK (top) and tempered Fenrir (bottom). **A** Distribution of tRMSEs. The average tRMSE for Fenrir is lower than for RK. **B** Voltage traces for the inferred parameters, colored by their tRMSE. The observation is shown in white. **C** Median errors for seven commonly used electrophysiological features (as defined in Appendix A3). Quartiles are marked as black bars. Fenrir is able to reproduce qualitative aspects of the data much better than RK.

Definition A3.2), fraction of successfully converged runs (CONV), the number of iteration steps (ITER) and the number of correctly identified parameters (N_θ). While the focus of our work was on the reliability of convergence we also show that the inclusion of a simple early stopping criterion (see Appendix A2 for details) makes our algorithm competitive on cost.

We observed that diffusion tempering (OURS) yielded the best results across all of our experiments in terms of tRMSE, pRMSE and the number of successfully converged initializations (Table 1). Moreover, with our method, we recovered the true parameters from about twice to four times as often as the default Fenrir depending on the model. Even in settings where the RK baseline was not able to converge correctly at all, as was the case for the two compartment model with six parameters, our method was correct 88% of the time. Additionally, in the case of the three parameter HH_1 model, diffusion tempering doubled the performance of Fenrir. As expected, using a schedule came at the expense of additional optimization steps. For our selected schedule, this meant about an order of magnitude more steps compared to the original method. However, the inclusion of a simple early stopping criterion (OURS+) made diffusion tempering cost competitive without loss of performance for the subset of problems we tested. This can be attributed to the fact that convergence with a high precision is unnecessary during early tempering stages and that most trajectories converged far before the end of the schedule (Figure A1). Combined with steeper schedules (Appendix A8.1 this would bring the costs in line with the original Fenrir method while retaining superior performance.

While for the HH_1 model with 6 parameters all methods seemingly performed on par with respect to mean pRMSE and tRMSE, no algorithm manages to converged to the true parameters reliably. However, this was a very complicated model, which is why we looked at the parameter estimates in detail. First we looked at the distributions of tRMSEs.

We noticed that the tRMSEs obtained via diffusion tempering were much more uniformly distributed than those from RK, while being in the same range (Figure 6A). However, not only was the fraction of parameter estimates which yielded steady-state solutions for RK extremely high (Figure 6B) – for many applications these are discarded immediately (Prinz et al., 2003; Rossant et al., 2011) – but they often displayed very different spike frequencies from the observed data. This was not the case for our method for which the solutions come qualitatively much closer to that of the observed data. For practitioners interested in parameter estimation for this model, it mainly matters whether a set of commonly used eletrophysiological summary statistics is reproduced by the fitted model (for their definitions see Appendix A3.1). Here we found that our method deviated much less from the observed behavior, suggesting that the parameters capture essential aspects of the data much better than the classical baseline (Figure 6C). Both methods also performed much better than when the diffusion was learned (Figure A4).

5. Discussion

Gradient descent has the potential to efficiently fit ODEs to data, but non-linear differential equations can exhibit many local minima, leading to poor convergence and extreme sensitivity to initial conditions. Here, we used probabilistic integrators to improve the convergence of gradient descent based parameter fitting of ODEs. We developed diffusion

tempering, a method that successively solves several parameter search tasks. We demonstrated that our method dramatically improves the reliability of parameter optimization across several popular ODEs, and we showed that our method enables gradient descent based parameter estimation on a six-parameter Hodgkin–Huxley model, which had been unattainable with existing gradient-based methods.

5.1. Limitations

Of the compared methods diffusion tempering was the most expensive in terms of runtime, which we attribute to two aspects: the runtime of the ODE solver, and the sequential solving of multiple optimization problems. The probabilistic ODE solvers used by our method are semi-implicit and A-stable (Tronarp et al., 2019). While this allows them to perform well on stiff ODEs, this comes at the cost of cubic scaling with the ODE dimension, shared by traditional implicit Runge–Kutta methods (Hairer & Wanner, 1999). Using explicit probabilistic ODE solvers, which scale linearly with the ODE dimension (Krämer et al., 2022), would improve the runtime of diffusion tempering on larger systems. Additionally, the overall runtime of diffusion tempering is strongly dependent on the iterative optimization procedure that is used. This includes the tempering curve as well as the optimization itself. However, we did show that early stopping and steeper schedules, together, can bring down the computational cost to much closer to that of the other methods, and we suspect better tempering schemes can improve this even further. We also note that in our work we assumed the magnitude of the observation noise R to be known, which in practice is often not the case and which can be taken into account by Fenrir as well.

As the scope of this paper was gradient descent based parameter estimation in ODEs, we did not include other commonly used approaches such as genetic algorithms (Hazelden et al., 2023) in our experimental evaluation. Furthermore, we only looked at recovering parameters for a known form of dynamics. While optimizing parameters of NODEs (Chen et al., 2018) presents an interesting problem set, we see a few potential challenges before our method can be applied here. Firstly, computing gradients efficiently would require the use of adjoints (Chen et al., 2018), which, to our knowledge, has not yet been derived in the probabilistic numerical framework. Additionally, the $O(d^3)$ scaling of our method in the ODE dimension would limit the number of latent dimensions that can be modeled.

5.2. Future Work

While we already presented ways to reduce the cost of diffusion tempering using simple early stopping criteria or different schedules, more elaborate and potentially problem aware tempering schemes could further increase the efficiency and effectiveness of our method. Additionally, in our work we assumed the observation noise to be known, hence further work is needed to study the effect of also learning or scheduling this parameter, which could have a similarly regularizing effect on the optimization.

Authorship Contribution

Conceptualization: PB, PH, JHM, JB, NB; Methodology: JB, NB; Software: JB, NB; Investigation: JB, NB, MD; Formal Analysis: JB; Visualization: JB; Supervision: PB, PH, JHM; Writing – Original Draft: JB, NB; Writing – Review & Editing: PB, NB, MD, PH, JHM, KLK; Funding Acquisition: PB, PH, JHM

Acknowledgments

We thank the members of the Probabilistic Inference in Mechanistic Models (PIMMS) Network project of the Cluster of Excellence “Machine Learning — New Perspectives for Science” for discussion. The work was funded by the German Science Foundation (EXC 2064 “Machine Learning — New Perspectives for Science”, 2064/1 – 390727645), the Hertie Foundation, the European Union (ERC, “NextMechMod”, ref. 101039115, “ANUBIS”, ref. 101123955, “DeepCoMechTome”, ref. 101089288), as well as the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (01IS18039A). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. PH and NB are supported by funds of the Ministry of Science, Research and Arts of the State of Baden-Württemberg. We also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting JB, NB, MD and KLK.

Impact Statement

This paper contributes to the advancement of the field of machine learning. It aligns with the ongoing progress in the broader landscape of machine learning research. Compared to other methods in the field our work does not introduce unique and distinct societal challenges that warrant explicit discussion.

References

Achard, P. and De Schutter, E. Complex parameter landscape for a complex neuron model. *PLoS computational biology*, 2(7):e94, 2006. doi: 10.1371/journal.pcbi.0020094.

Armijo, L. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, January 1966. ISSN 0030-8730. doi: 10.2140/pjm.1966.16.1.

Bard, Y. *Nonlinear parameter estimation*. Academic Press, New York, 1974. ISBN 978-0-12-078250-5.

Ben-Shalom, R., Aviv, A., Razon, B., and Korngreen, A. Optimizing ion channel models using a parallel genetic algorithm on graphical processors. *Journal of Neuroscience Methods*, 206(2):183–194, 2012. doi: 10.1016/j.jneumeth.2012.02.024.

Benson, M. Parameter fitting in dynamic models. *Ecological Modelling*, 6(2):97–115, February 1979. ISSN 03043800. doi: 10.1016/0304-3800(79)90029-2.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, January 2017. ISSN 0036-1445, 1095-7200. doi: 10.1137/141000671.

Cao, J., Wang, L., and Xu, J. Robust estimation for ordinary differential equation models. *Biometrics*, 67(4): 1305–1313, December 2011. ISSN 0006341X. doi: 10.1111/j.1541-0420.2011.01577.x.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in neural information processing systems 31: Annual conference on neural information processing systems, NeurIPS 2018*, pp. 6572–6583, 2018.

Cockayne, J., Oates, C., Sullivan, T., and Girolami, M. Bayesian probabilistic numerical methods. *SIAM Review*, 61(3):756–789, 2019. ISSN 0036-1445, 1095-7200. doi: 10.1137/17M1139357.

Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences of the United States of America*, 117 (48):30055–30062, December 2020. ISSN 1091-6490. doi: 10.1073/pnas.1912789117.

Dass, S. C., Lee, J., Lee, K., and Park, J. Laplace based approximate posterior inference for differential equation models. *Statistics and Computing*, 27(3):679–698, May 2017. ISSN 0960-3174, 1573-1375. doi: 10.1007/s11222-016-9647-0.

Dayan, P. and Abbott, L. F. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, 2001. ISBN 978-0-262-54185-5.

Druckmann, S., Banitt, Y., Gidon, A., Schürmann, F., Markram, H., and Segev, I. A novel multiple objective optimization framework for constraining conductance-based neuron models by experimental data. *Frontiers in Neuroscience*, 1:1, 2007. ISSN 1662-453X. doi: 10.3389/neuro.01.1.1.001.2007.

Gonçalves, P. J., Lueckmann, J.-M., Deistler, M., Nonnenmacher, M., Öcal, K., Bassetto, G., Chintaluri, C., Podlaski, W. F., Haddad, S. A., Vogels, T. P., Greenberg, D. S., and Macke, J. H. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, 9:e56261, 2020. ISSN 2050-084X. doi: 10.7554/eLife.56261.

Hairer, E. and Wanner, G. Stiff differential equations solved by radau methods. *Journal of Computational and Applied Mathematics*, 111(1):93–111, November 1999. ISSN 0377-0427. doi: 10.1016/S0377-0427(99)00134-X.

Hairer, E., Nørsett, S. P., and Wanner, G. *Solving Ordinary Differential Equations I*, volume 8 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987. ISBN 978-3-662-12609-7. doi: 10.1007/978-3-662-12607-3.

Hazelden, J., Liu, Y. H., Shlizerman, E., and Shea-Brown, E. Evolutionary algorithms as an alternative to backpropagation for supervised training of biophysical neural networks and neural odes. (arXiv:2311.10869), November 2023. doi: 10.48550/arXiv.2311.10869.

Hennig, P., Osborne, M. A., and Kersting, H. P. *Probabilistic Numerics*. Cambridge University Press, June 2022.

Hethcote, H. W. The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653, January 2000. ISSN 0036-1445, 1095-7200. doi: 10.1137/S0036144500371907.

Hodgkin, A. L. and Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117 (4):500–544, August 1952. ISSN 0022-3751, 1469-7793. doi: 10.1113/jphysiol.1952.sp004764.

Kersting, H., Sullivan, T. J., and Hennig, P. Convergence rates of gaussian ode filters. *Statistics and Computing*, 30(6):1791–1816, Nov 2020. ISSN 1573-1375. doi: 10.1007/s11222-020-09972-4.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220(4598): 671–680, May 1983. doi: 10.1126/science.220.4598.671.

Krämer, N. and Hennig, P. Stable implementation of probabilistic ode solvers. *CoRR*, 2020.

Kravtsov, S. Dynamics and predictability of hemispheric-scale multidecadal climate variability in an observationally constrained mechanistic model. *Journal of Climate*, 33(11):4599–4620, June 2020. ISSN 0894-8755, 1520-0442. doi: 10.1175/JCLI-D-19-0778.1.

Krämer, N., Bosch, N., Schmidt, J., and Hennig, P. Probabilistic ode solutions in millions of dimensions. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 11634–11649. PMLR, June 2022.

Lueckmann, J.-M., Gonçalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., and Macke, J. H. Flexible statistical inference for mechanistic models of neural dynamics. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in neural information processing systems 30: Annual conference on neural information processing systems*, pp. 1289–1299, 2017.

Marjoram, P., Molitor, J., Plagnol, V., and Tavaré, S. Markov chain monte carlo without likelihoods. 100:15324–15328, December 2003. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0306899100.

Mogensen, P. K. and Riseth, A. N. Optim: A mathematical optimization package for julia. *Journal of Open Source Software*, 3:615, April 2018. ISSN 2475-9066. doi: 10.21105/joss.00615.

Neal, R. M. Probabilistic inference using markov chain monte carlo methods. pp. 144, 1993.

Neelakantan, A., Vilnis, L., Le, Q. V., Kaiser, L., Kurach, K., Sutskever, I., and Martens, J. Adding gradient noise improves learning for very deep networks. 2017.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN 978-0-387-30303-1. doi: 10.1007/978-0-387-40065-5.

Pospischil, M., Toledo-Rodriguez, M., Monier, C., Piwkowska, Z., Bal, T., Frégnac, Y., Markram, H., and Destexhe, A. Minimal hodgkin–huxley type models for different classes of cortical and thalamic neurons. *Biological Cybernetics*, 99(4–5):427–441, November 2008. ISSN 0340-1200, 1432-0770. doi: 10.1007/s00422-008-0263-8.

Prinz, A. A., Billimoria, C. P., and Marder, E. Alternative to hand-tuning conductance-based models: Construction and analysis of databases of model neurons. *Journal of Neurophysiology*, 90(6):3998–4015, December 2003. ISSN 0022-3077, 1522-1598. doi: 10.1152/jn.00641.2003.

Rackauckas, C. and Nie, Q. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(11):15, May 2017. doi: 10.5334/jors.151.

Rossant, C., Goodman, D. F., Fontaine, B., Platkiewicz, J., Magnusson, A., and Brette, R. Fitting neuron models to spike trains. *Frontiers in Neuroscience*, 5:9, 2011. ISSN 1662-453X. doi: 10.3389/fnins.2011.00009.

Scala, F., Kobak, D., Bernabucci, M., Bernaerts, Y., Cadwell, C. R., Castro, J. R., Hartmanis, L., Jiang, X., Laturnus, S., Miranda, E., Mulherkar, S., Tan, Z. H., Yao, Z., Zeng, H., Sandberg, R., Berens, P., and Tolias, A. S. Phenotypic variation of transcriptomic cell types in mouse motor cortex. *Nature*, November 2020. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-020-2907-3.

Schober, M., Särkkä, S., and Hennig, P. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, Jan 2019. ISSN 1573-1375. doi: 10.1007/s11222-017-9798-7.

Schweppe, F. Evaluation of likelihood functions for gaussian signals. *IEEE Transactions on Information Theory*, 11(1):61–70, January 1965. ISSN 0018-9448. doi: 10.1109/TIT.1965.1053737.

Särkkä, S. *Bayesian filtering and smoothing*. Cambridge University Press, 2013.

Taylor, A. L., Goaillard, J.-M., and Marder, E. How multiple conductances determine electrophysiological properties in a multicompartment model. *Journal of Neuroscience*, 29(17):5573–5586, April 2009. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.4438-08.2009.

Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29(6):1297–1315, 2019. doi: 10.1007/s11222-019-09900-1.

Tronarp, F., Särkkä, S., and Hennig, P. Bayesian ode solvers: the maximum a posteriori estimate. *Statistics and Computing*, 31(3):23, March 2021. ISSN 1573-1375. doi: 10.1007/s11222-021-09993-7.

Tronarp, F., Bosch, N., and Hennig, P. Fenrir: Physics-enhanced regression for initial value problems. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 21776–21794. PMLR, June 2022.

Appendix

A1. Implementation

We wrote all of our experiments in the Julia programming language (Bezanson et al., 2017). For the implementation of probabilistic ODE filters we relied on the original code for Fenrir (Tronarp et al., 2022) and ProbNumDiffEq.jl. To compute the trajectories for the observations as well as for least-squares regression with RK, the `RadauIIA5` (Hairer & Wanner, 1999) solver is used, with adaptive step-size selection. The solver was provided by DifferentialEquations.jl (Rackauckas & Nie, 2017). For the optimization we used the implementation of L-BFGS from Optim.jl (Mogensen & Riseth, 2018). This is a comparable setup to that of Tronarp et al. (2022).

All computations were done on an internal cluster running Intel(R) Xeon(R) Gold 6226R CPUs @ 2.90GHz. All code and data are publicly available on GitHub and Zenodo.

A2. Additional experimental details

To generate the data used as the observations we solved the IVP with the fully-implicit Runge–Kutta `RadauIIA5` (Hairer & Wanner, 1999) method with both absolute and relative tolerances of 10^{-14} on an equi-spaced time grid $t_i \in \mathbb{T} = \{0.0, 0.01, ..., T\}$ with $dt = 0.01$. The Ts differed for each model and can be found in Appendix A5. Uniform Gaussian noise of variance $\sigma^2 = 0.1$ was then added to the data.

To evaluate the different methods we conducted all experiments for 100 different initializations.

As the probabilistic ODE solver for the IVP we used a first order linearization of the vector field and a 3-times integrated integrated Wiener process prior.

During diffusion tempering the optimizer often converged very close to the parameter bounds. This meant that the initial values for the next optimization problem would lie right on top of the bounding box, which would throw an error. We mitigate this, by giving a tiny nudge to any initial values that end up right on top of the parameter bounds before starting the optimization.

In the experiments marked with OURS+ in Table 1 we added a simple early stopping criterion to Algorithm 1 to reduce the computational cost of our algorithm. This could prematurely stop the optimization (OPT) during all but the last scheduled value of κ, if the absolute change in the Fenrir log likelihood $\widehat{\mathcal{M}}_{PN}$ across 3 consecutive updates was below 0.1. Since last tempering step was never interrupted, this ensured the final optimization was performed to the same tolerances as the other methods.

A3. Metrics

To compare the quality of the parameter estimates we use several metrics. Firstly, we compare using the trajectory root mean squared error (RMSE)

Definition A3.1 (Trajectory RMSE)**.** Let $\hat{\theta}$ be the parameters estimated by an inference algorithm, and let $\mathbb{T}_\mathsf{D}$ be the set of measurement nodes. Then, let $\hat{y}(t)$, $t \in \mathbb{T}_\mathsf{D}$, be the estimated system trajectory, computed by numerically integrating the ODE with initial values and parameters as given by the estimated $\hat{\theta}$. The trajectory RMSE (tRMSE) is then defined as

$$\text{tRMSE} := \sqrt{\frac{1}{|\mathbb{T}_\mathsf{D}|} \sum_{t \in \mathbb{T}_\mathsf{D}} \left\| \hat{y}(t) - y(t) \right\|_2^2}. \tag{14}$$

which evaluates the quality of the results in the trajectory space and which is also used in the original work(Tronarp et al., 2022). However, while this is useful to get an idea of how closely the observed data was reproduced it does not necessarily allow to deduce how well the parameters match those that have generated the original data. This is especially a problem for oscillatory systems for which the tRMSE of the zero function is often better than for a solution which is slightly phase shifted, which arguably is often a better fit. We therefore evaluate the quality of the inferred trajectories also in parameter space.

Definition A3.2 (Relative Parameter RMSE)**.** Let $\hat{\theta}$ be the parameters estimated by an inference algorithm, and let θ^* be the known, true parameters which are to be inferred. Let D be the dimensionality of the parameter space. Then the relative

parameter RMSE (pRMSE) d_θ is defined as

$$\text{pRMSE} := \sqrt{\frac{1}{D}\sum_{i=1}^{D}\left\|(\hat{\theta}_i - \theta_i^*)/\theta_i^*\right\|_2^2}. \tag{15}$$

This metric is also used to determine whether an optimization converged successfully or not. If the final parameter estimate $\hat{\theta}$ lies within a 5% ball of the true parameters, i.e. $pRMSE < 0.05$, then we consider it to have correctly converged.

A3.1. Electrophysiological Summary Features

It is difficult to evaluate the quality of parameter estimates for higher parameterised HH models, since a good tRMSE does not necessarily mean characteristic features of the data are being reproduced. Hence to score the quality of parameter estimates, we also define seven commonly used electrophysiological summary features. These reflect measures that electrophysiologists regularly use to characterize and quantify the qualitative behavior of recorded neurons (Scala et al., 2020). This reduces the high dimensional model output to a meaningful set of numbers that emphasize specific aspects of the data and are easier to interpret.

Definition A3.3 (Summary feature)**.** Let θ parameterize an IVP. and let $y(t)$, be the system trajectory, computed. Then a summary feature is defined as a function

$$x := s(y(t)). \tag{16}$$

which reduces the system trajectory to a single number x.

The following summary statistics are computed from the trajectory.

Table 2. Summary features to quantify the behavior of different voltage traces.

Feature	Definition
AP peak	Maximum voltage of an action potential (AP).
AP trough	Minimum voltage of the trough / hyperpolarization that follows the AP.
AP amp	Difference of peak V_{peak} and trough voltage V_{trough}.
ISI	The time between two consecutive APs.
Resting V	Average of $V(9\,ms < t < 10\,ms)$, i.e. 1 ms before the stimulus.
Latency	Time between stimulus onset and reaching the first peak voltage.
#APs	Number of APs.

For AP peak, AP trough, AP amp and ISI we compute the mean. To compare to the observation we then take the absolute value of the relative differences between the observed x^* and the estimated $\hat{x}$ summary features. If no spikes are detected, spike dependent features will set to NaN.

A4. Optimization

For optimization we use L-BFGS for both Fenrir and RK (Nocedal & Wright, 2006). This is also the optimizer used in the original Fenrir paper (Tronarp et al., 2022). To speed up convergence we also use backtracking line search (Armijo, 1966). To determine convergence we used an absolute tolerance of 10^{-9} and a relative tolerance of 10^{-6}, which would give an accuracy of about 10^{-4} to 10^{-5} since the total solution accuracy is roughly 1-2 digits less than the relative tolerances (see SciML Docs).

The optimization was performed with box-constraints, such that the optimization would stay within the limits provided by the uniform prior distribution with upper and lower bounds θ_{ub}, θ_{lb}. All initial parameters were also drawn from this distribution, with initial values set to $u(t_0)$. Observation noise is always fixed to the true $\sigma^2 = 0.1$.

All optimized parameters θ were transformed to ranges of 0 to 1 according to:

$$\tilde{\theta} = \frac{\theta - \theta_{lb}}{\theta_{ub} - \theta_{lb}}, \tag{17}$$

where θ_{lb} and θ_{ub} are the lower and upper bounds on the parameter. Additionally κ was first transformed to log-space, before putting it through Equation (17). This ensures that the gradients for each parameter are roughly comparable.

A5. Models

A5.1. Pendulum

The first ODE we consider is that of a pendulum (Equation (18)). Although this is a fairly simple system, parameter estimation can be challenging due to local optima. The ODE is a 2nd order equation $\frac{d^2\phi}{dt^2} = -\frac{g}{l} sin(\phi)$, which we reformulate as a system of first order equations.

$$\frac{d\phi_1}{dt} = \phi_2 \tag{18a}$$

$$\frac{d\phi_2}{dt} = -\frac{g}{l} sin(\phi_1) \tag{18b}$$

where $g = 9.81\, m/s^2$ and $\varphi(0) = \frac{\pi}{4}$. In our experiments we chose the bounds and values for the single parameter l as in Appendix A5.1.

Table 3. Parameter bounds and values that were used for our experiments.

Parameter	Lower Bound	Upper bound	observation
l	0.1	10.0	3

Hence $\theta = \{l\}$. The system was integrated for $t \in [0, 10]$ and only partially observed with a measurement matrix (see Equation (2)) of $H = [1\ 0]$.

A5.2. Lotka-Volterra

The Lotka–Volterra or predator and prey model, can be described by a pair of first-order nonlinear differential equations (Equation (19)). It describes how the populations of two species (predator and prey) evolve when they coexist in an environment.

$$\frac{dx}{dt} = \alpha x - \beta xy \tag{19a}$$

$$\frac{dy}{dt} = \delta xy - \gamma y \tag{19b}$$

We set the initial populations to $x(0) = 1.0$, $y(0) = 1.0$ and use the bounds and values to parameterize the model as shown in Appendix A5.2. Hence $\theta = \{\alpha, \beta\}$ for the 2 parameter and $\theta = \{\alpha, \beta, \gamma, \delta\}$ for the 4 parameter case. The system was

Table 4. Parameter bounds and values that were used for our experiments.

Parameter	Lower Bound	Upper bound	observation
α	$1 \cdot 10^{-3}$	5.0	1.5
β	$1 \cdot 10^{-3}$	5.0	1.0
γ	$1 \cdot 10^{-3}$	5.0	3.0
δ	$1 \cdot 10^{-3}$	5.0	1.0

integrated for $t \in [0, 20]$ and only partially observed with a measurement matrix (Equation (2)) of $H = [0\ 1]$, such that only the prey species is observed.

A5.3. The Hodgkin Huxley Model

Here we outline the specific versions of the HH model that were used in our experiments. We consider sodium, potassium and leak currents to generate spiking, a slow non-inactivating K^+ current to enable spike-frequency adaptation, and a high-threshold Ca^{2+} current to generate bursting (Pospischil et al., 2008):

$$\begin{aligned} C\,A\frac{dV_t}{dt} &= I_t + \bar{g}_{\text{Na}} m^3 h (E_{\text{Na}} - V_t) \\ &\quad + \bar{g}_{\text{K}} n^4 (E_{\text{K}} - V_t) + \bar{g}_{\text{leak}} (E_{\text{leak}} - V_t) \\ &\quad + \bar{g}_{\text{M}} p (E_{\text{K}} - V_t) + \bar{g}_{\text{L}} q^2 r (E_{\text{Ca}} - V_t) \end{aligned} \tag{20}$$

$$\tag{21}$$

$\bar{g}_i,\ i \in \{\text{Na}, \text{K}, \text{leak}, \text{M}, \text{L}\}$ are the maximum conductances of the sodium, potassium, leak, adaptive potassium and calcium ion channels, respectively. E_i are the associated reversal potentials. I_t denotes the current per unit area, C the membrane capacitance, A the compartment area and n, m, h, q, r and p represent the fraction of independent gates in the open state, based on Hodgkin & Huxley (1952).

The dynamics of the gates can be expressed in general form as Equations (22a) and (22b), where $\alpha_z(V_t)$ and $\beta_z(V_t)$ are rate constants for each of the gating variables $z \in \{m, n, h, q, r\}$ and p.

$$\frac{dz_t}{dt} = (\alpha_z(V_t)\,(1 - z_t) - \beta_z(V_t) z_t) \tag{22a}$$

$$\frac{dp_t}{dt} = ((p_\infty(V_t) - p_t)/\tau_p(V_t)) \tag{22b}$$

They model the kinetics of different channel proteins. The parameters for $\alpha_z(V_t)$, $\beta_z(V_t)$, $p_\infty(V_t)$ and $\tau_p(V_t, \tau_{max})$ are taken from Pospischil et al. (2008). The detailed equations are also provided in Equation (23).

$$\alpha_m(V) = -0.32 \frac{V - V_T - 13}{e^{-(V - V_T - 13)/4} - 1}, \quad \beta_m(V) = 0.28 \frac{V - V_T - 40}{e^{(V - V_T - 40)/5} - 1} \tag{23a}$$

$$\alpha_n(V) = -0.032 \frac{(V - V_T - 15)}{e^{-(V - V_T - 15)/5} - 1}, \quad \beta_n(V) = 0.5 e^{-(V - V_T - 10)/40} \tag{23b}$$

$$\alpha_h(V) = 0.128 e^{-(V - V_T - 17)/18}, \quad \beta_h(V) = \frac{4}{e^{-(V - V_T - 40)/5} + 1} \tag{23c}$$

$$\alpha_q(V) = 0.055 \frac{-27 - V}{e^{(-27 - V)/3.8} - 1}, \quad \beta_q(V) = 0.94 e^{(-75 - V)/17} \tag{23d}$$

$$\alpha_r(V) = 0.000457 e^{(-13 - V)/50}, \quad \beta_r(V) = 0.0065 / (e^{(-15 - V)/28} + 1) \tag{23e}$$

$$\tau_p(V) = \frac{\tau_{\text{max}}}{3.3 e^{(V + 35)/20} + e^{-(V + 35)/20}} \tag{23f}$$

The parameter values and bounds we used in our experiments for the HH model can be found in Table 5.

$A = \frac{\tau}{C R_{in}}$, C the membrane capacitance, R_{in} the input resistance. τ is the membrane time constant, τ_{max} the time constant of the slow K^+ current and V_T the threshold voltage.

Table 5. Parameter bounds and values that were used for our experiments.

Parameter	Lower bound	Upper bound	Observation
C ($\mu F/cm^2$)	0.4	3	1
A (cm^2)	$1.9 \cdot 10^{-5}$	$30.2 \cdot 10^{-5}$	$8.3 \cdot 10^{-5}$
g_{Na} (mS)	0.5	80	25
g_{K} (mS)	$1 \cdot 10^{-4}$	15	7
E_{Na} (mV)	50	100	53
E_{K} (mV)	110	-70	-107
g_{leak} (mS)	$1 \cdot 10^{-4}$	0.8	0.1
E_{leak} (mV)	-110	-50	-70
V_T (mV)	90	-40	-60
g_{M} (mS)	$1 \cdot 10^{-5}$	0.6	0.01
E_{Ca} (mV)	100	150	120
g_{L} (mS)	$1 \cdot 10^{-4}$	0.6	0.01
τ_{max} (s)	50	$3 \cdot 10^3$	$4 \cdot 10^3$

The default parameters are chosen according to `http://help.brain-map.org/download/attachments/8323525/BiophysModelPeri.pdf`. With the parameters for the cell area A being taken from in vitro recordings from the mouse cortex from the Allen Cell Type Database the cell ID (509881736). This closely follows the experiments from (Gonçalves et al., 2020).

For the 1, 2 and 3 parameter models we removed the equations for I_M, and I_L to make it easier to simulate. This is the same as considering g_M and g_L to be 0. Furthermore, for the 6 and 8 parameter model the parameters where adjusted to $gleak = 0.05\,mS$ and $\tau_{max} = 1000\,s$. The initial voltage was set to $V(0) = -70\,mV$, from which the initial gating variables where computed according to Equation (24), where $z(0) = z_\infty(V(0))$ for $z \in \{m, n, h, p, q, r\}$. The system was integrated for $t \in [0, 100]$ms and only the voltage was observed, which yields a measurement matrix of $H = [1\ 0\ 0\ 0\ 0\ 0\ 0]$. For the full system and $H = [1\ 0\ 0\ 0]$ with I_M and I_L removed.

$$m_\infty(V) = 1/(1 + \beta_m(V)/\alpha_m(V)) \tag{24a}$$
$$n_\infty(V) = 1/(1 + \beta_n(V)/\alpha_n(V)) \tag{24b}$$
$$h_\infty(V) = 1/(1 + \beta_h(V)/\alpha_h(V)) \tag{24c}$$
$$p_\infty(V) = 1/(1 + e^{-(V+35)/10}) \tag{24d}$$
$$q_\infty(V) = 1/(1 + \beta_q(V)/\alpha_q(V)) \tag{24e}$$
$$r_\infty(V) = 1/(1 + \beta_r(V)/\alpha_r(V)) \tag{24f}$$
$$\tag{24g}$$

Depending on the model, the following parameter sets were optimized.

Table 6. Parameter sets for the different single compartment models, which we denote with HH_1.

#parameters	θ
1	$\{g_{\text{Na}}\}$
2	$\{g_{\text{Na}}, g_{\text{K}}\}$
3	$\{g_{\text{Na}}, g_{\text{K}}, g_{\text{leak}}\}$
6	$\{g_{\text{Na}}, g_{\text{K}}, g_{\text{Na}}, g_{\text{M}}, V_T, g_{\text{L}}\}$

A5.3.1. MULTICOMPARTMENT MODELS

To model neurons with a complex morphological structure the neuron is split into discrete compartments that are small enough that the variation of the membrane potential across it is negligible. Then the continuous membrane potential V can be approximated as by a sum of local compartment potentials. For a non-branching cable, this can be expressed as

$$CA_i \frac{dV_i}{dt} = -I_{ionic,i} + I_i + g_{i,i+1}\left(V_{i+1} - V_i\right) + g_{i,i-1}\left(V_{i-1} - V_i\right), \tag{25}$$

with subscript i indexing the compartments, the sum of the ionic currents across the cells membrane I_{ionic}, the external current I and coupling coefficients g. Each membrane potential V_i satisfies an equation similar to Equation (20) except that each compartments couples to two neighbors (unless at the ends of the cable). For more context see (Dayan & Abbott, 2001) Chapter 6. In our experiments we used coupling coefficients of $g = 1$ and compartment areas of $A_i = \frac{A}{N}, \quad i \in \{1, ..., N\}$, for N compartments, such that the total area would sum to the same A as in Table 5. Furthermore, we stimulated only the first compartment and considered all compartments to be observed (voltage only). The parameters used for the 4 and 6 parameter models where modified from Table 5. For the 4 parameter model the 2nd compartment was adjusted to $g_{Na} = 20\,mS$ and $g_K = 10\,mS$. For the 6 parameter model, the first $g_{leak} = 0.09\,mS$ and the 2nd compartment was changed to $g_{Na} = 20\,mS$, $g_K = 10\,mS$ and $g_{leak} = 0.11\,mS$.

Depending on the number of compartments N this yields parameter sets of the form $\theta = \{\theta_1, ..., \theta_N\}$, where θ_i are analogous to the parameters of the single compartments in Table 6. The measurements are taken with

$$H = \begin{bmatrix} H_1 & 0 & \dots \\ \vdots & \ddots & \\ 0 & & H_N, \end{bmatrix} \tag{26}$$

where H_i are the measurement matrices for a single compartment. We denote the number of compartments that were used in an experiment with a subscript HH_i.

A6. Additional Details on Section 4.2

Here we show all the trajectories and likelihoods for which a subset was presented in Figure 4. Additionally, we plotted the solutions at each of the initial and final parameters for each diffusion parameter.

While the likelihood is initially very flat and the uniformly initialized parameters collect on the top left edge, a basin around the global optimum appears at around $\kappa = 10^{18}$ (Figure A1). At $\kappa = 10^{17}$ the ridge that separates the local optimum from the basin is small enough for the optimizer to cross into the basin in which the global optimum resides. With decreasing κ the basin starts to taper more and more towards the true parameters, which becomes very distinct down to about $\kappa = 10^6$, after which point hardly any changes to the likelihood landscape are visible.

We observed that first all runs converge to the steady state solution at $\kappa = 10^{18}$, before suddenly developing spikes and converging correctly (Figure A2). After reaching a diffusion parameter of $\kappa = 10^{15}$, the subset of initial values has fully converged and hardly any changes are visible.

A7. Additional Details on Section 4.3

Once the diffusion hyperparameter is set to $\log \kappa \leq 17$ most values perform similarly well in terms of recovering the true parameters about half of the time, with the exception of $\log \kappa = 16$, which does so with about 80% reliability (Figure A3). Additionally, the average distance from the true parameters starts decreasing between $\log \kappa = 9$ and $\log \kappa = 17$, before sharply jumping again. The fact that the fraction of parameters that converge correctly does not change significantly until hitting $\log \kappa = 1e15$, suggests that the same set of, favorable, initial guesses converge no matter the loss. However, since the mean pRMSE starts decreasing overall, suggests that initializations further away also end up closer to the global optimum.

Comparison of Fenrir and Diffusion tempering for a 6 parameter HH model. Diffusion tempering yields electrophysiological summary statistics much closer to those of the true observation compared to Fenrir.

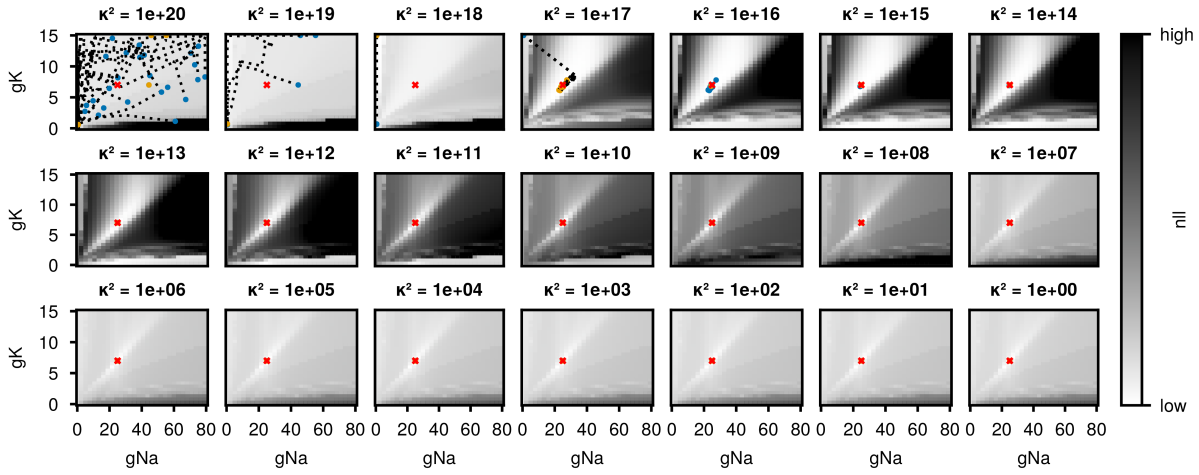


Figure A1. Parameter estimation for the sodium g_{Na} and potassium g_K conductance of a HH model. Parameter optimization during all stages of diffusion tempering for a random subset of 25 initializations. Optimization trajectories (dotted) in each likelihood landscape are shown from start (blue) to convergence (orange). Very high loss values were clipped for better visual clarity.

A8. Additional Experiments

A8.1. Effectiveness of different tempering rates

In our work we mainly focused on the convergence reliability of diffusion tempering. For this reason we picked a tempering schedule that was very simple and reliable during our testing – linearly reducing the $\log \kappa$ from 20 and 0 with a step size of 1. Despite its simplicity this already performed much better than our baselines without any further tuning. Since this schedule is very costly in the number of iterations needed (Table 1), we also investigated the effect of different schedules, i.e. with much steeper tempering rates or exponentially decreasing $\log \kappa$ on the cost of diffusion tempering for the HH_1 model with 2 parameters.

We found that much steeper, hence shorter schedules that decrease $\log \kappa$ between 20 and 0 can save about 20% iterations, while affecting the performance not at all or very little (Figure A5). In particular we found that taking more than 16 tempering steps yields no additional benefits for the HH_1 model with 2 parameters. Additionally, similar to Section 4.2 most of the trajectories started to converge to the true parameters around $\kappa = 10^{15}$, for many of the schedules.

We also tested an exponentially decaying the $\log_{10}(\kappa)$ where at each tempering step κ was set according to:

$$\mathcal{T}(i, \kappa_0) = 10^{\kappa_0 \exp(-i/\tau)}, \tag{27}$$

with initial diffusion $\kappa_0 = 20$, decay constant $\tau = 5$ and tempering steps $i \in \{0, 1, ..., 20\}$. We found that this exponential tempering schedule can save around 30% of the computation cost, while only taking a small hit to performance. Combined with the early stopping criterion we implemented (Appendix A2), this brings the cost of diffusion tempering down to that of the original Fenrir method, while retaining the superior convergence.

While we only considered a small number of possible tempering schemes, our experiments show that this can already reduce the cost of our algorithm by a third. This suggests future work on more efficient tempering schemes will reduce the cost of diffusion tempering even further.

A8.2. Injecting gradient noise during optimization

It is known that injecting noise into the gradient updates can effectively regularize the optimization procedure in challenging loss landscapes (Neelakantan et al., 2017). While Fenrir and hence diffusion tempering are are noise-free methods, we also

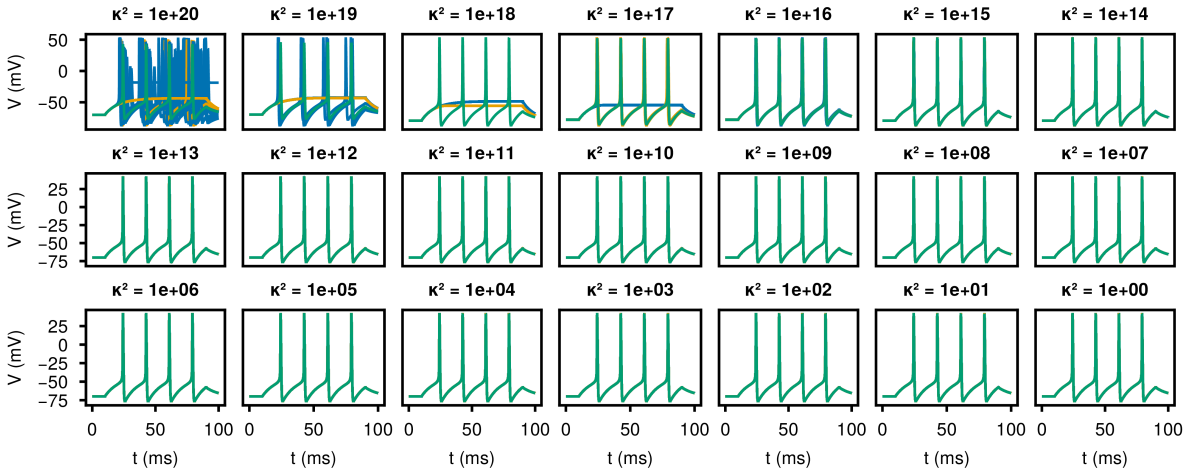


Figure A2. Parameter estimation for the sodium g_{Na} and potassium g_K conductance of a HH model. ODE solutions for the start (blue) to converged (orange) parameters at all stages of diffusion tempering for a random subset of 25 initializations.

investigated how adding gradient noise to RK compares to diffusion tempering for the pendulum.

For this purpose we optimized the pendulum as described in (Appendix A4), but Gaussian noise ϵ_n of the form $\epsilon_n \sim \mathcal{N}(0, \sigma^2 e^{-n/\tau})$, with decay rate $1/\tau$ and initial variance σ^2, was added to the gradients of the log likelihood at every parameter update step n.

The addition of gradient noise with varying scales and decay rates improved the performance of the noise-free RK variant almost across the board, even doubling the convergence rates in some instances (Table 7). Adding noise with a variance at around 5 performed the best reaching a convergence rate of 0.64 for non decaying noise levels. While this improves upon the noise-free RK baseline, this still falls short of the convergence rates that both Fenrir and diffusion tempering achieve however, with 0.75 and 1.00 respectively.

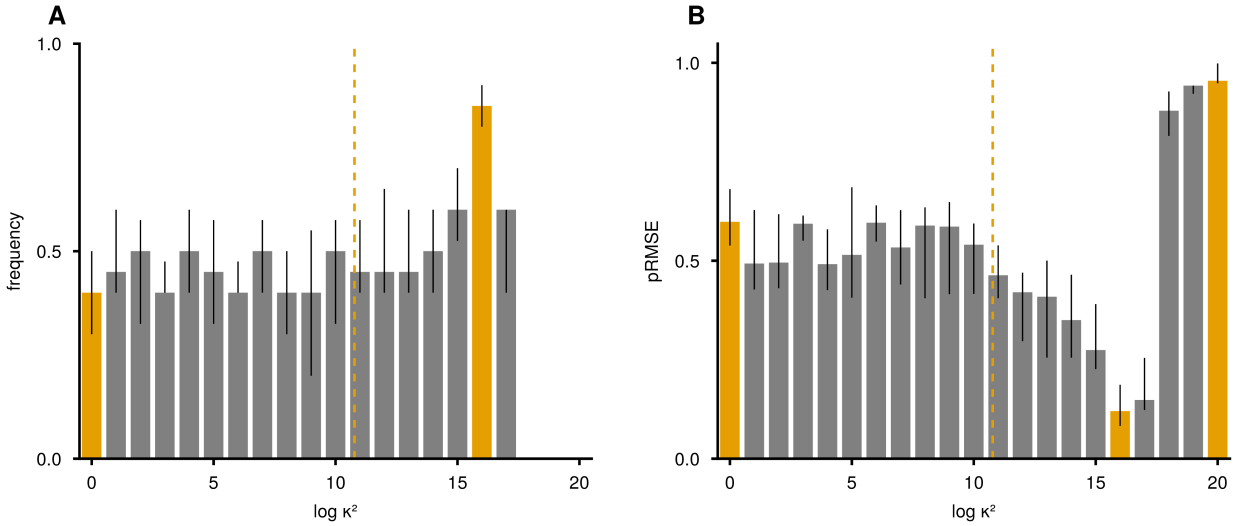


Figure A3. Convergence for a 2 parameter HH model for a grid of different fixed κs. The parameters selected for Figure 5 are highlighted in orange and the optimal diffusion parameter $\log \kappa = 11.6$ (dashed). Histogram shows the medians, with black bars indicating quartiles for 100 runs split into groups of 10. **A** Fraction of runs with $pRMSE < 5\%$. **B** pRMSEs for each diffusion.

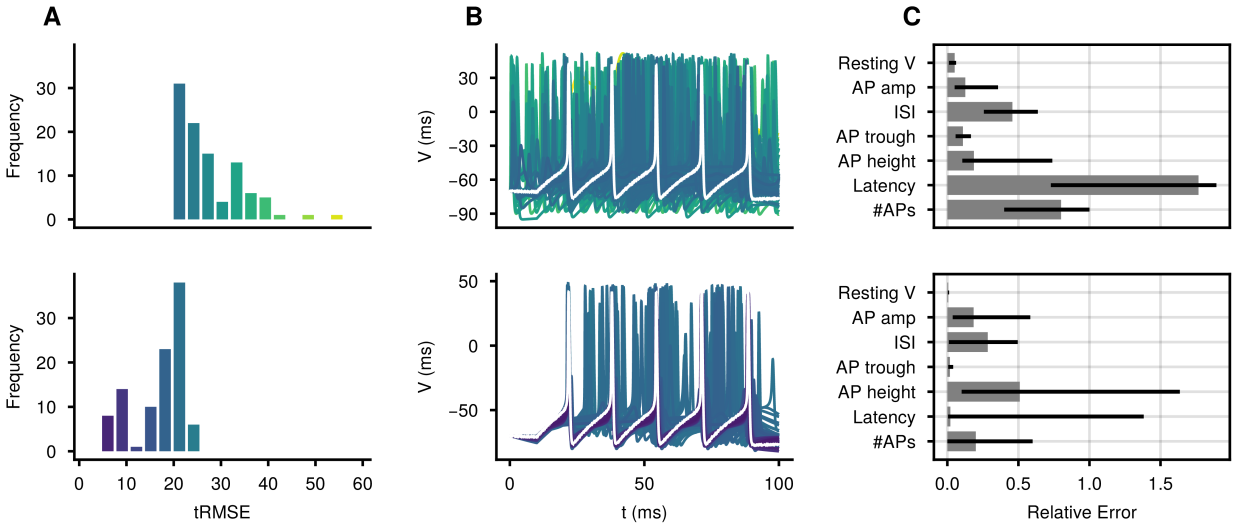


Figure A4. Parameter inference for a six parameter HH model for 100 initializations of Fenrir (top) and our method (bottom). **A** Distribution of tRMSEs. Almost all parameter estimates returned by our method have better tRMSEs than Fenrir. **B** Voltage traces for the inferred parameters, colored by their tRMSE. The observation is shown in white. The traces recovered by Fenrir seem almost random. **C** Median errors for seven commonly used electrophysiological features (as defined in Appendix A3). Quartiles are highlighted with error bars. Our method is able to reproduce qualitative aspects of the data better than Fenrir.

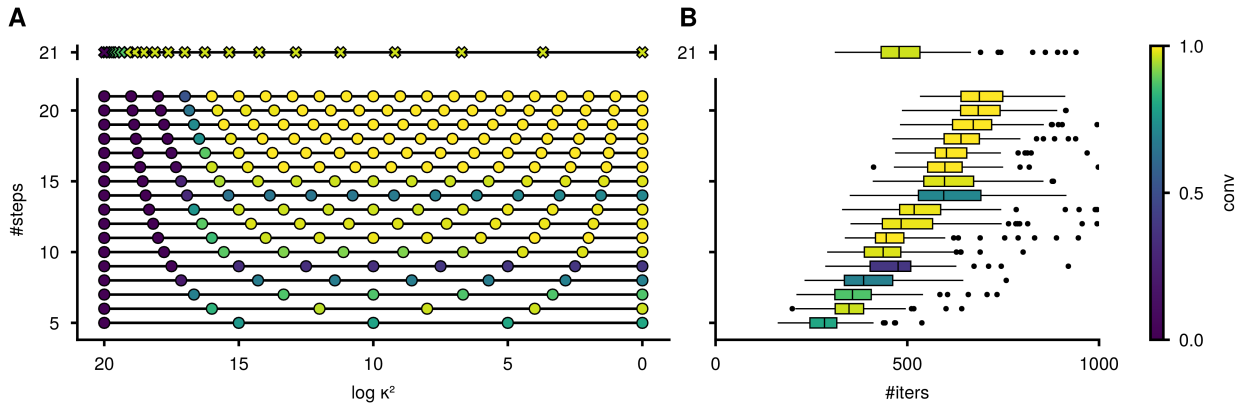


Figure A5. Comparing tempering schedules with different numbers of tempering steps between $\log \kappa = 20$ and $\log \kappa = 0$ for the HH_1 model with 2 parameters. **A** Percentage of correctly inferred parameters at each stage of tempering during optimization. Linear decrease of $\log \kappa$ (dots) vs. exponential decrease (crosses). There is a clear trade-off between the number of tempering steps and convergence, but taking more than 16 steps yields no additional benefits. **B** Cost to performance trade-off for different tempering schedules. Exponential schedules are more efficient at the same number of tempering steps.

Table 7. Effect of injecting gradient noise with different scale, σ^2, and decay rates, τ, during optimization of a pendulum on the convergence. While injecting gradient noise improves convergence compared to the noise-free RK baseline (first row), both Fenrir and diffusion tempering still converge more reliably.

σ^2	τ	CONV	PRMSE	ITER
0.00	–	0.32	1.42 ± 1.09	33.47 ± 18.11
0.01	–	0.35	1.40 ± 1.11	33.79 ± 12.84
0.05	–	0.34	1.41 ± 1.10	29.89 ± 10.77
0.10	–	0.37	1.40 ± 1.13	27.30 ± 11.84
0.50	–	0.49	1.14 ± 1.13	15.19 ± 10.72
1.00	–	0.63	0.77 ± 1.05	15.52 ± 14.12
5.00	–	**0.64**	0.51 ± 0.88	13.01 ± 8.87
10.00	–	0.39	0.64 ± 0.93	10.43 ± 7.19
50.00	–	0.45	0.49 ± 0.83	11.66 ± 8.26
100.00	–	0.28	$\mathbf{0.44 \pm 0.74}$	$\mathbf{10.26 \pm 7.42}$
0.00	5	0.32	1.42 ± 1.09	33.47 ± 18.11
0.01	5	0.31	1.45 ± 1.08	35.94 ± 18.58
0.05	5	0.33	1.45 ± 1.10	32.58 ± 13.39
0.10	5	0.35	1.40 ± 1.11	33.65 ± 12.85
0.50	5	0.43	1.24 ± 1.13	22.23 ± 14.93
1.00	5	0.53	0.98 ± 1.08	22.68 ± 18.51
5.00	5	**0.59**	0.67 ± 0.99	15.23 ± 13.33
10.00	5	0.45	0.51 ± 0.84	13.11 ± 13.49
50.00	5	0.40	$\mathbf{0.51 \pm 0.79}$	$\mathbf{11.36 \pm 11.60}$
100.00	5	0.41	0.58 ± 0.86	12.54 ± 14.20
0.00	10	0.32	1.42 ± 1.09	33.47 ± 18.11
0.01	10	0.33	1.43 ± 1.10	33.42 ± 12.90
0.05	10	0.32	1.44 ± 1.09	34.21 ± 11.00
0.10	10	0.34	1.39 ± 1.11	33.40 ± 12.70
0.50	10	0.46	1.22 ± 1.14	18.89 ± 13.20
1.00	10	0.50	1.08 ± 1.10	18.58 ± 14.95
5.00	10	**0.61**	0.58 ± 0.94	14.28 ± 12.63
10.00	10	0.58	0.55 ± 0.90	12.95 ± 10.90
50.00	10	0.32	0.41 ± 0.71	$\mathbf{9.96 \pm 8.40}$
100.00	10	0.37	$\mathbf{0.40 \pm 0.72}$	10.92 ± 8.47
FENRIR		0.75	0.20 ± 0.37	110.04 ± 61.70
OURS		**1.00**	$\mathbf{0.00 \pm 0.00}$	696.22 ± 78.10

Parallel-in-Time Probabilistic Numerical ODE Solvers

Nathanael Bosch NATHANAEL.BOSCH@UNI-TUEBINGEN.DE
University of Tübingen

Adrien Corenflos ADRIEN.CORENFLOS@AALTO.FI
Department of Electrical Engineering and Automation, Aalto University

Fatemeh Yaghoobi FATEMEH.YAGHOOBI@AALTO.FI
Department of Electrical Engineering and Automation, Aalto University

Filip Tronarp FILIP.TRONARP@MATSTAT.LU.SE
Center for Mathemtaical Sciences, Lund University

Philipp Hennig PHILIPP.HENNIG@UNI-TUEBINGEN.DE
Tübingen AI Center, University of Tübingen

Simo Särkkä SIMO.SARKKA@AALTO.FI
Department of Electrical Engineering and Automation, Aalto University

Abstract

Probabilistic numerical solvers for ordinary differential equations (ODEs) treat the numerical simulation of dynamical systems as problems of Bayesian state estimation. Aside from producing posterior distributions over ODE solutions and thereby quantifying the numerical approximation error of the method itself, one less-often noted advantage of this formalism is the algorithmic flexibility gained by formulating numerical simulation in the framework of Bayesian filtering and smoothing. In this paper, we leverage this flexibility and build on the time-parallel formulation of iterated extended Kalman smoothers to formulate a *parallel-in-time* probabilistic numerical ODE solver. Instead of simulating the dynamical system sequentially in time, as done by current probabilistic solvers, the proposed method processes all time steps in parallel and thereby reduces the span cost from *linear* to *logarithmic* in the number of time steps. We demonstrate the effectiveness of our approach on a variety of ODEs and compare it to a range of both classic and probabilistic numerical ODE solvers.

Keywords: probabilistic numerics, ordinary differential equations, numerical analysis, parallel-in-time methods, Bayesian filtering and smoothing.

1. Introduction

Ordinary differential equations (ODEs) are used throughout the sciences to describe the evolution of dynamical systems over time. In machine learning, ODEs provide a continuous description of certain neural networks (Chen et al., 2018) and optimization procedures (Helmke et al., 2012; Su et al., 2016), and are used in generative modeling with normalizing flows (Papamakarios et al., 2021) and diffusion models (Song et al., 2021), among others. Unfortunately, all but the simplest ODEs are too complex to be solved analytically. Therefore, numerical methods are required to obtain a solution. While a multitude of numerical solvers has been developed over the last century (Hairer et al., 1993; Deuflhard and Bornemann, 2012; Butcher, 2016), most commonly-used methods do not provide a quantification of their own inevitable numerical approximation error.

Probabilistic numerics provides a framework for treating classic numerical problems as problems of probabilistic inference (Hennig et al., 2015; Oates and Sullivan, 2019; Hennig et al., 2022). In the context of ODEs, methods based on Gaussian process regression (Skilling, 1992; Hennig and Hauberg, 2014) and in particular Gauss–Markov regression (Schober et al., 2019; Kersting et al., 2020; Tronarp et al., 2019) provide an efficient and flexible approach to compute posterior distributions over the solution of ODEs (Bosch et al., 2021; Krämer and Hennig, 2020), and even partial differential equations (Krämer et al., 2022) and differential-algebraic equations (Bosch et al., 2022). These so-called *ODE filters* typically scale cubically in the ODE dimension (as do most *implicit* ODE solvers) and specific approximations enable linear scaling (shared by most *explicit* solvers) (Krämer et al., 2022). But to date, their linear scaling with the number of time steps remains.

For very large-scale simulations with very long time horizons, the sequential processing in time of most ODE solvers can become a bottleneck. This motivates the development of *parallel-in-time* methods: By leveraging the ever-increasing parallelization capabilities of modern computer hardware, parallel-in-time methods can achieve *sub-linear* scaling in the number of time steps (Gander, 2015). One well-known method of this kind is Parareal (Lions et al., 2001). It achieves temporal parallelism by combining an expensive, accurate solver with a cheap, coarse solver, in such a way that the fine solver is only ever applied to individual time slices in a parallel manner, leading to a square-root scaling (in ideal conditions). But, due to its sequential coarse-grid solve, Parareal still has only limited concurrency (Gander and Vandewalle, 2007), and while it has recently been extended probabilistically by Pentland et al. (2021, 2022) to improve its performance and convergence, these methods do not provide probabilistic solutions to ODEs per se.

In this paper, we leverage the time-parallel formulation of Gaussian filters and smoothers (Särkkä and García-Fernández, 2021; Yaghoobi et al., 2021, 2023) to formulate a parallel-in-time probabilistic numerical ODE solver. The paper is structured as follows. Section 2 formulates numerical ODE solutions as Bayesian state estimation problems and presents the established, sequential, filtering-based probabilistic ODE solvers. Section 3 then presents our proposed parallel-in-time probabilistic ODE solver; first as exact inference for affine ODEs, then as an iterative, approximate algorithm for general nonlinear ODEs. Section 4 then presents experiments on a variety of ODEs and compares the performance of our proposed method to that of existing, both probabilistic and non-probabilistic, ODE solvers. Finally, Section 5 concludes with a discussion of our results and an outlook on future work.

2. Numerical ODE Solutions as Bayesian State Estimation

Consider an initial value problem (IVP) of the form

$$\dot{y}(t) = f(y(t), t), \quad t \in [0, T], \qquad y(0) = y_0, \tag{1}$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial value $y_0 \in \mathbb{R}^d$. To capture the numerical error that arises from temporal discretization, the quantity of interest in probabilistic numerics for ODEs is the *probabilistic numerical ODE solution*, defined as

$$p\left(y(t) \;\middle|\; y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^N\right), \tag{2}$$

for some prior $p\left(y(t)\right)$ and with $\{t_n\}_{n=1}^N \subset [0, T]$ the chosen time-discretisation.

In the following, we pose the probabilistic numerical ODE solution as a problem of Bayesian state estimation, and we define the prior, likelihood, data, and approximate inference scheme. For a more detailed description of the transformation of an IVP into a Gauss–Markov regression problem, refer to Tronarp et al. (2019).

2.1 Gauss–Markov Process Prior

We model the solution y of the IVP with a ν-times integrated Wiener process prior (IWP(ν)). More precisely, let $Y(t) = \left[Y^{(0)}(t), Y^{(1)}(t), \ldots, Y^{(\nu)}(t)\right]$ be the solution of the following linear, time-invariant stochastic differential equation with Gaussian initial condition

$$\mathrm{d}Y^{(i)}(t) = Y^{(i+1)}(t)\,\mathrm{d}t, \qquad i = 0, \ldots, \nu - 1, \tag{3a}$$

$$\mathrm{d}Y^{(\nu)}(t) = \Gamma\,\mathrm{d}W(t), \tag{3b}$$

$$Y(0) \sim \mathcal{N}\left(\mu_0, \Sigma_0\right), \tag{3c}$$

with initial mean and covariance $\mu_0 \in \mathbb{R}^{d(\nu+1)}$, $\Sigma_0 \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$, diffusion $\Gamma \in \mathbb{R}^{d\times d}$, and d-dimensional Wiener process $W : \mathbb{R} \to \mathbb{R}^d$. Then, $Y^{(i)}$ is chosen to model the i-th derivative of the IVP solution y. By construction, accessing the i-th derivative can be done by multiplying the state Y with a projection matrix $E_i \coloneqq I_d \otimes e_i$, that is, $Y^{(i)}(t) = E_i Y(t)$.

This continuous-time prior satisfies discrete transition densities (Särkkä and Solin, 2019)

$$Y(t+h) \mid Y(t) \sim \mathcal{N}\left(\Phi(h)Y(t), Q(h)\right), \tag{4}$$

with transition matrix and process noise covariance $\Phi(h), Q(h) \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$ and step $h \in \mathbb{R}_+$. For the IWP(ν) these can be computed in closed form (Kersting et al., 2020), as

$$\Phi(h) = I_d \otimes \breve{\Phi}(h), \qquad \left[\breve{\Phi}(h)\right]_{ij} = \mathbb{1}_{i=j}\frac{h^{i-j}}{(j-i)!}, \tag{5a}$$

$$Q(h) = I_d \otimes \breve{Q}(h), \qquad \left[\breve{Q}(h)\right]_{ij} = \frac{h^{2\eta+1-i-j}}{(2\nu+1-i-j)(\nu-i)!(\nu-j)!}. \tag{5b}$$

Remark 1 (Alternative Gauss–Markov priors)**.** *While ν-times integrated Wiener process priors have been the most common choice for filtering-based probabilistic ODE solvers in recent years, the methodology is not limited to this choice. Alternatives include the ν-times integrated Ornstein–Uhlenbeck process and the class of Matérn processes, both of which have a similar continuous-time SDE representation as well as Gaussian transition densities in discrete time. Refer to Tronarp et al. (2021) and Särkkä and Solin (2019).*

The initial distribution $\mathcal{N}(\mu_0, \Sigma_0)$ is chosen such that it encodes the initial condition $y(0) = y_0$. Furthermore, to improve the numerical stability and the quality of the posterior, we initialize not only on the function value $Y^{(0)}(0) = y_0$, but also the higher order derivatives, that is, $Y^{(i)}(0) = \frac{\mathrm{d}^i y}{\mathrm{d}t^i}(0)$ for all $i \leq \nu$ (Krämer and Hennig, 2020). These terms can be efficiently computed via Taylor-mode automatic differentiation (Griewank, 2000; Bettencourt et al., 2019). As a result, we obtain an initial distribution with mean

$$\mu_0 = \left[y_0, \frac{\mathrm{d}y}{\mathrm{d}t}(0), \ldots, \frac{\mathrm{d}^\nu y}{\mathrm{d}t^\nu}(0)\right]^T, \tag{6}$$

and zero covariance $\Sigma_0 = 0$, since the initial condition has to hold exactly.

2.2 Observation Model and Data

To relate the introduced Gauss–Markov prior to the IVP problem from Equation (1), we define an observation model in terms of the information operator

$$\mathcal{Z}[y](t) \coloneqq \dot{y}(t) - f\left(y(t), t\right). \tag{7}$$

By construction, $\mathcal{Z}$ maps the true IVP solution y *exactly* to the zero function, that is, $\mathcal{Z}[y] \equiv 0$. In terms of the continuous process Y, the information operator can be expressed as

$$\mathcal{Z}[Y](t) = E_1 Y(t) - f\left(E_0 Y(t), t\right), \tag{8}$$

where E_0 and E_1 are the projection matrices introduced in Section 2.1 which select the zeroth and first derivative from the process Y, respectively. There again, if Y corresponds to the true IVP solution (and its true derivatives), then $\mathcal{Z}[Y] \equiv 0$.

Conversely, inferring the true IVP solution requires conditioning the process $Y(t)$ on $Z(t) = 0$ over the whole continuous interval $t \in [0, T]$. Since this is in general intractable, we instead condition $Y(t)$ only on discrete observations $Z(t_n) = 0$ on a grid $\mathbb{T} = \{t_n\}_{n=1}^N$. This leads to the Dirac likelihood model commonly used in ODE filtering (Tronarp et al., 2019):

$$Z(t_n) \mid Y(t_n) \sim \delta\left(Y^{(1)}(t_n) - f\left(Y^{(0)}(t_n), t_n\right)\right), \tag{9}$$

with zero-valued data $Z(t_n) = 0$ for all $t_n \in \mathbb{T}$.

Remark 2 (Information operators for other differential equation problems)**.** *Similar information operators can be defined for other types of differential equations that are not exactly of the first-order form as given in Equation* (1)*, such as higher-order differential equations, Hamiltonian dynamics, or differential-algebraic equations (Bosch et al., 2022).*

2.3 Discrete-Time Inference Problem

The combination of prior, likelihood, and data results in a Bayesian state estimation problem

$$Y(0) \sim \mathcal{N}\left(\mu_o, \Sigma_0\right), \tag{10a}$$

$$Y(t_{n+1}) \mid Y(t_n) \sim \mathcal{N}\left(\Phi(t_{n+1} - t_n) Y(t_n), Q(t_{n+1} - t_n)\right), \tag{10b}$$

$$Z(t_n) \mid Y(t_n) \sim \delta\left(Y^{(1)}(t_n) - f\left(Y^{(0)}(t_n), t_n\right)\right), \tag{10c}$$

with zero data $Z(t_n) = 0$ for all $t_n \in \mathbb{T}$. The posterior distribution over $Y^{(0)}(t)$ then provides a probabilistic numerical ODE solution to the given IVP, as formulated in Equation (2).

This is a standard nonlinear Gauss–Markov regression problem, for which many approximate inference algorithms have previously been studied (Särkkä and Svensson, 2023). In the context of probabilistic ODE solvers, a popular approach for efficient approximate inference is Gaussian filtering and smoothing, where the solution is approximated with Gaussian distributions

$$p\left(Y(t) \mid \{Z(t_n) = 0\}_{n=1}^N\right) \approx \mathcal{N}\left(\mu(t), \Sigma(t)\right). \tag{11}$$

This is most commonly performed with extended Kalman filtering (EKF) and smoothing (EKS) (Schober et al., 2019; Tronarp et al., 2019; Kersting et al., 2020); though other methods

have been proposed, for example based on numerical quadrature (Kersting and Hennig, 2016) or particle filtering (Tronarp et al., 2019). *Iterated* extended Kalman smoothing (e.g. Bell, 1994; Särkkä and Svensson, 2023) computes the "maximum a posteriori" estimate of the probabilistic numerical ODE solution (Tronarp et al., 2021). This will be the basis for the parallel-in-time ODE filter proposed in this work, explained in detail in Section 3.

2.4 Practical Considerations for Probabilistic Numerical ODE Solvers

While Bayesian state estimation methods such as the extended Kalman filter and smoother can, in principle, be directly applied to the formulated state estimation problem, there are a number of modifications and practical considerations that should be taken into account:

- *Square-root formulation:* Gaussian filters often suffer from numerical stability issues when applied to the ODE inference problem defined in Equation (10), in particular when using high orders and small steps. To alleviate these issues, probabilistic numerical ODE solvers are typically formulated in square-root form (Krämer and Hennig, 2020); this is also the case for the proposed parallel-in-time method.

- *Preconditioned state transitions:* Krämer and Hennig (2020) suggest a coordinate change preconditioner to make the state transition matrices step-size independent and thereby improve the numerical stability of EKF-based probabilistic ODE solvers. This preconditioner is also used in this work.

- *Uncertainty calibration:* The Gauss–Markov prior as introduced in Section 2.1 has a free parameter, the diffusion Γ, which directly influences the uncertainty estimates returned by the ODE filter. In this paper, we consider scalar diffusions $\Gamma = \sigma \cdot I$ and compute a quasi-maximum likelihood estimate for the parameter σ post-hoc, as suggested by Tronarp et al. (2019).

- *Approximate linearization:* Variants of the standard EKF/EKS-based inference have been proposed in which the linearization of the vector-field is done only approximately. Approximating the Jacobian of the ODE vector field with zero enables inference with a complexity which scales only linearly with the ODE dimension (Krämer et al., 2022), while still providing polynomial convergence rates (Kersting et al., 2020). A diagonal approximation of the Jacobian preserves the linear complexity, but improves the stability properties of the solver (Krämer et al., 2022). In this work, we only consider the exact first-order Taylor linearization.

- *Local error estimation and step-size adaptation:* Rather than predefining the time discretization grid, certain solvers employ an adaptive approach where the solver dynamically constructs the grid while controlling an internal estimate of the numerical error. Step-size adaptation based on *local* error estimates have been proposed for both classic (Hairer et al., 1993, Chapter II.4) and probabilistic ODE solvers (Schober et al., 2019; Bosch et al., 2021). On the other hand, *global* step-size selection is often employed in numerical boundary value problem (BVP) solvers (Ascher et al., 1995, Chapter 9), and has been extended to filtering-based probabilistic BVP solvers (Krämer and Hennig, 2021). For our purposes, we will focus on fixed grids.

3. Parallel-in-Time Probabilistic Numerical ODE Solvers

This section develops the main method proposed in this paper: a parallel-in-time probabilistic numerical ODE solver.

3.1 Parallel-Time Exact Inference in Affine Vector Fields

Let us first consider the simple case: An initial value problem with affine vector field

$$\dot{y}(t) = L(t)y(t) + d(t), \quad t \in [0, T], \qquad y(0) = y_0. \tag{12}$$

The corresponding information model of the probabilistic solver is then also affine, with

$$Z(t) \mid Y(t) \sim \delta\left(H(t)Y(t) - d(t)\right), \tag{13a}$$
$$H(t) \coloneqq E_1 - L(t)E_0. \tag{13b}$$

Let $\mathbb{T} = \{t_n\}_{n=1}^N \subset [0, T]$ be a discrete time grid. To simplify the notation in the following, we will denote a function evaluated at time t_n by a subscript n, that is $Y(t_n) =: Y_n$, except for the transition matrices where we will use $\Phi_n \coloneqq \Phi(t_{n+1} - t_n)$ and $Q_n \coloneqq Q(t_{n+1} - t_n)$. Then, the Bayesian state estimation problem from Equation (10) reduces to inference of $Y(t)$ in the model

$$Y_0 \sim \mathcal{N}\left(\mu_o, \Sigma_0\right), \tag{14a}$$
$$Y_{n+1} \mid Y_n \sim \mathcal{N}\left(\Phi_n Y_n, Q_n\right), \tag{14b}$$
$$Z_n \mid Y_n \sim \delta\left(H_n Y_n - d_n\right), \tag{14c}$$

with zero data $Z_n = 0$ for all $n = 1, \ldots, N$. Since this is an affine Gaussian state estimation problem, it can be solved exactly with Gaussian filtering and smoothing (Kalman, 1960; Rauch et al., 1965; Särkkä and Svensson, 2023); see also (Tronarp et al., 2019, 2021) for explicit discussions of probabilistic numerical solvers for affine ODEs.

Recently, Särkkä and García-Fernández (2021) presented a parallel-time formulation of Bayesian filtering and smoothing, as well as a concrete algorithm for exact linear Gaussian filtering and smoothing—which could be directly applied to the problem formulation in Equation (14). But as mentioned in Section 2.4, the resulting ODE solver might suffer from numerical instabilities. Therefore, we use the square-root formulation of the parallel-time linear Gaussian filter and smoother by Yaghoobi et al. (2023). In the following, we review the details of the algorithm.

3.1.1 Parallel-Time General Bayesian Filtering and Smoothing

First, we follow the presentation of Särkkä and García-Fernández (2021) and formulate Bayesian filtering and smoothing as prefix sums. We define elements $a_n = (f_n, g_n)$ with

$$f_n(Y_n \mid Y_{n-1}) = p(Y_n \mid Z_n, Y_{n-1}), \tag{15a}$$
$$g_n(Y_{n-1}) = p(Z_n \mid Y_{n-1}), \tag{15b}$$

where for $n = 1$ we have $p(Y_1 \mid Z_1, Y_0) = p(Y_1 \mid Z_1)$ and $p(Z_1 \mid Y_0) = p(Z_1)$, together with a binary operator $\otimes_f : (f_i, g_i) \otimes_f (f_j, g_j) \mapsto (f_{ij}, g_{ij})$ defined by

$$f_{ij}(x \mid z) \coloneqq \frac{\int g_j(y) f_j(x \mid y) f_i(y \mid z) \,\mathrm{d}y}{\int g_j(y) f_i(y \mid z) \,\mathrm{d}y}, \tag{16a}$$

$$g_{ij}(z) \coloneqq g_i(z) \int g_j(y) f_i(y \mid z) \,\mathrm{d}y. \tag{16b}$$

Then, Särkkä and García-Fernández (2021, Theorem 3) show that $\otimes_f$ is associative and that

$$a_1 \otimes_f \cdots \otimes_f a_n = \begin{bmatrix} p(Y_n \mid Z_{1:n}) \\ p(Z_{1:n}) \end{bmatrix}, \tag{17}$$

that is, the filtering marginals and the marginal likelihood of the observations at step n are the results of a cumulative sum of the elements $a_{1:n}$ under $\otimes_f$. Since the operator $\otimes_f$ is associative, this quantity can be computed in parallel with prefix-sum algorithms, such as the parallel scan algorithm by Blelloch (1989).

Remark 3 (On Prefix-Sums)**.** *Prefix sums, also known as cumulative sums or inclusive scans, play an important role in parallel computing. Their computation can be efficiently parallelized and, if enough parallel resources are available, their (span) computational cost can be reduced from* linear *to* logarithmic *in the number of elements. One such algorithm is the well-known parallel scan algorithm by Blelloch (1989) which, given N elements and $N/2$ processors, computes the prefix sum in $2\lceil \log_2 N \rceil$ sequential steps with $2N - 2$ invocations of the binary operation. This algorithm is implemented in both tensorflow (Abadi et al., 2015) and JAX (Bradbury et al., 2018); the latter is used in this work.*

The time-parallel smoothing step can be constructed similarly: We define elements $b_n = p(Y_n \mid Z_{1:n}, Y_{n+1})$, with $b_N = p(Y_N \mid Z_{1:N})$, and a binary operator $b_i \otimes_s b_j = b_{ij}$, with

$$b_{ij}(x \mid z) = \int b_i(x \mid y) b_j(y \mid z) \,\mathrm{d}y. \tag{18}$$

Then, $\otimes_s$ is associative and the smoothing marginal at time step n is the result of a reverse cumulative sum of the elements $b_{n:N}$ under $\otimes_s$ (Särkkä and García-Fernández, 2021):

$$b_n \otimes_s \cdots \otimes_s b_N = p(Y_n \mid Z_{1:N}). \tag{19}$$

Again, since the smoothing operator $\otimes_s$ is associative, this cumulative sum can be computed in parallel with a prefix-sum algorithm (Blelloch, 1989).

3.1.2 Parallel-Time Linear Gaussian Filtering in Square-Root Form

In the linear Gaussian case, the filtering elements $a_n = (f_n, g_n)$ can be parameterized by a set of parameters $\{A_n, b_n, C_n, \eta_n, J_n\}$ as follows:

$$f_n(Y_n \mid Y_{n-1}) = p(Y_n \mid Z_n, Y_{n-1}) = \mathcal{N}\left(Y_n; A_n Y_{n-1} + b_n, C_n\right), \tag{20a}$$

$$g_n(Y_{n-1}) = p(Z_n \mid Y_{n-1}) \propto \mathcal{N}_I\left(Y_{n-1}; \eta_n, J_n\right), \tag{20b}$$

where $\mathcal{N}_I$ denotes a Gaussian density parameterized in information form, that is, $\mathcal{N}_I(x; \eta, J) = \mathcal{N}(x; J^{-1}\eta, J^{-1})$. The parameters $\{A_n, b_n, C_n, \eta_n, J_n\}$ can be computed explicitly from the given state-space model (Särkkä and García-Fernández, 2021, Lemma 7). But since probabilistic numerical ODE solvers require a numerically stable implementation of the underlying filtering and smoothing algorithm (Krämer and Hennig, 2020), we formulate the parallel-time linear Gaussian filtering algorithm in square-root form, following Yaghoobi et al. (2023).

To this end, let $\sqrt{M}$ denote a left square-root of a positive semi-definite matrix M, that is, $\sqrt{M}\sqrt{M}^T = M$; the matrix $\sqrt{M}$ is sometimes also called a "generalised Cholesky factor" of M (S. Grewal and P. Andrews, 2014). To operate on square-root matrices, we also define the *triangularization* operator: Given a wide matrix $M \in \mathbb{R}^{n\times m}$, $m \geq n$, the triangularization operator $\text{tria}(M)$ first computes the QR decomposition of M^T, that is, $M^\mathsf{T} = QR$, with wide orthonormal $Q \in \mathbb{R}^{m\times n}$ and square upper-triangular $R \in \mathbb{R}^{n\times n}$, and then returns R^T. This operator plays a central role in square-root filtering algorithms as it enables the numerically stable addition of covariance matrices, provided square-roots are available: Given two positive semi-definite matrices $A, B \in \mathbb{R}^{n\times n}$ with square-roots $\sqrt{A}, \sqrt{B}$, a square-root of the sum $A + B$ can be computed as

$$\sqrt{A+B} = \text{tria}\left(\begin{bmatrix}\sqrt{A} & \sqrt{B}\end{bmatrix}\right). \tag{21}$$

With these definitions in place, we briefly review the parallel-time linear Gaussian filtering algorithm in square-root form as provided by Yaghoobi et al. (2023) in the following.

Parameterization of the filtering elements. Let $m_0 = \mu_0$, $P_0 = \Sigma_0$, and $m_n = 0$, $P_n = 0$ for all $n \geq 1$, and define

$$m_n^- = \Phi_{n-1} m_{n-1}, \tag{22a}$$

$$\sqrt{P_n^-} = \text{tria}\left(\begin{bmatrix}\Phi_{n-1}\sqrt{P_{n-1}} & \sqrt{Q_{n-1}}\end{bmatrix}\right). \tag{22b}$$

Then, the square-root parameterization of the filtering elements a_n is given by

$$A_n = (I - K_n H_n)\Phi_{n-1}, \tag{23a}$$

$$b_n = m_n^- - K_n\left(H_n m_n^- - d_n\right), \tag{23b}$$

$$\sqrt{C_n} = \Psi_{22}, \tag{23c}$$

$$\eta_n = \sqrt{J_n}\sqrt{S_n}^{-1} d_n, \tag{23d}$$

$$\sqrt{J_n} = \Phi_{n-1}^\mathsf{T} H_n^\mathsf{T} \sqrt{S_n}^{-\mathsf{T}}, \tag{23e}$$

where I is the identity matrix and Ψ_{22}, $\sqrt{S_n}$ and K_n are defined via

$$\begin{bmatrix}\Psi_{11} & 0\\ \Psi_{21} & \Psi_{22}\end{bmatrix} = \text{tria}\left(\begin{bmatrix}H_n\sqrt{P_n^-} & \sqrt{R_n}\\ \sqrt{P_n^-} & 0\end{bmatrix}\right), \tag{24a}$$

$$\sqrt{S_n} = \Psi_{11}, \tag{24b}$$

$$K_n = \Psi_{21}\Psi_{11}^{-1}. \tag{24c}$$

For generality the formula includes an observation noise covariance R_n, but note that in the context of probabilistic ODE solvers we have a noiseless measurement model with $\sqrt{R_n} = 0$.

Associative filtering operator. Let a_i, a_j be two filtering elements, parameterized in square-root form by $a_i = \{A_i, b_i, \sqrt{C_i}, \eta_i, \sqrt{J_i}\}$ and $a_j = \{A_j, b_j, \sqrt{C_j}, \eta_j, \sqrt{J_j}\}$. Then, the associative filtering operator $\otimes_f$ computes the filtering element $a_{ij} = a_i \otimes_f a_j$ as

$$A_{ij} = A_j A_i - A_j \sqrt{C_i} \Xi_{11}^{-\mathsf{T}} \Xi_{21}^{\mathsf{T}} A_i, \tag{25a}$$

$$b_{ij} = A_j \left(I - \sqrt{C_i} \Xi_{11}^{-\mathsf{T}} \Xi_{21}^{\mathsf{T}} \right) (b_i + \sqrt{C_i} \sqrt{C_i}^{\mathsf{T}} \eta_j) + b_j, \tag{25b}$$

$$\sqrt{C_{ij}} = \operatorname{tria}\left(\begin{bmatrix} A_j \sqrt{C_i} \Xi_{11}^{-\mathsf{T}} & \sqrt{C_j} \end{bmatrix} \right), \tag{25c}$$

$$\eta_{ij} = A_i^{\mathsf{T}} \left(I - \Xi_{21} \Xi_{11}^{-1} \sqrt{C_i}^{\mathsf{T}} \right) \left(\eta_j - \sqrt{J_j} \sqrt{J_j}^{\mathsf{T}} b_i \right) + \eta_i, \tag{25d}$$

$$\sqrt{J_{ij}} = \operatorname{tria}\left(\begin{bmatrix} A_i^{\mathsf{T}} \Xi_{22} & \sqrt{J_i} \end{bmatrix} \right), \tag{25e}$$

where Ξ_{11}, Ξ_{21}, Ξ_{22} are defined via

$$\begin{bmatrix} \Xi_{11} & 0 \\ \Xi_{21} & \Xi_{22} \end{bmatrix} = \operatorname{tria}\left(\begin{bmatrix} \sqrt{C_i}^{\mathsf{T}} \sqrt{J_j} & I \\ \sqrt{J_j} & 0 \end{bmatrix} \right). \tag{26}$$

See Yaghoobi et al. (2023) for the detailed derivation.

The filtering marginals. The filtering marginals are then given by

$$p(Y_n | Z_{1:n}) = \mathcal{N}\left(Y_n; m_n^f, P_n^f \right), \qquad \text{with} \qquad m_n^f \coloneqq b_{1:n}, \quad \sqrt{P_n^f} \coloneqq \sqrt{C_{1:n}}. \tag{27}$$

This concludes the parallel-time linear Gaussian square-root filter.

3.1.3 Parallel-Time Linear Gaussian Smoothing in Square-Root Form

Similarly to the filtering equations, the linear Gaussian smoothing can also be formulated in terms of smoothing elements b_n and an associative operator $\otimes_s$, and the smoothing marginals can also be computed with a parallel prefix-sum algorithm.

Parameterization of the smoothing elements. The smoothing elements b_n can be described by a set of parameters $\{E_n, g_n, \sqrt{L_n}\}$, as

$$b_n = p(Y_n \mid Z_{1:n}, Y_{n+1}) = \mathcal{N}\left(Y_n; E_n Y_{n+1} + g_n, \sqrt{L_n} \sqrt{L_n}^{\mathsf{T}} \right). \tag{28}$$

The smoothing element parameters can be computed as

$$E_n = \Pi_{21} \Pi_{11}^{-1}, \tag{29a}$$

$$g_n = m_n^f - E_n \Phi_n m_n^f, \tag{29b}$$

$$\sqrt{L_n} = \Pi_{22}, \tag{29c}$$

$$\tag{29d}$$

where I is the identity matrix and the matrices Π_{11}, Π_{21}, Π_{22} are defined via

$$\begin{bmatrix} \Pi_{11} & 0 \\ \Pi_{21} & \Pi_{22} \end{bmatrix} = \operatorname{tria}\left(\begin{bmatrix} \Phi_n \sqrt{P_n^f} & \sqrt{Q_n} \\ \sqrt{P_n^f} & 0 \end{bmatrix} \right). \tag{30}$$

Associative smoothing operator. Given two smoothing elements b_i, b_j be two filtering elements, parameterized in square-root form by $b_i = \{E_i, g_i, \sqrt{L_i}\}$ and $b_j = \{E_j, g_j, \sqrt{L_j}\}$, the associative smoothing operator $\otimes_s$ computes the smoothing element $b_{ij} = b_i \otimes_s b_j$ as

$$E_{ij} = E_i E_j, \tag{31a}$$
$$g_{ij} = E_i g_j + g_i, \tag{31b}$$
$$\sqrt{L_{ij}} = \operatorname{tria}\left(\begin{bmatrix} E_i \sqrt{L_j} & \sqrt{L_i} \end{bmatrix}\right). \tag{31c}$$

The smoothing marginals. The smoothing marginals can then be retrieved from the reverse cumulative sum of the smoothing elements as

$$p(Y_n \mid Z_{1:N}) = \mathcal{N}\left(Y_n; m_n^s, P_n^s\right), \tag{32a}$$
$$m_n^s = g_{n:N}, \tag{32b}$$
$$\sqrt{P_n^s} = \sqrt{L_{n:N}}. \tag{32c}$$

Refer to Yaghoobi et al. (2023) for a thorough derivation. The full parallel-time Rauch–Tung–Striebel smoother is summarized in Algorithm 1.

Algorithm 1 Parallel-time Rauch–Tung–Striebel Smoother (`ParRTS`)

Input: Initial distribution (μ_0, Σ_0), linear transition models $\{(\Phi_n, Q_n)\}_{n=1}^N$, affine observation models $\{(H_n, d_n)\}_{n=1}^N$, data $Z_{1:N}$.

1: *Compute the filtering elements:*
$a_n = (A_n, b_n, \sqrt{C_n}, \eta_n, \sqrt{J_n})$ for all $n = 1, \ldots, N$ ▷ Eq. (23)

2: *Run the time-parallel Kalman filter:*
$\left\{\left(A_n^f, b_n^f, \sqrt{C_n^f}, \eta_n^f, \sqrt{J_n^f}\right)\right\}_{n=1}^N \leftarrow \texttt{AssociativeScan}\left(\otimes_f, (a_n)_{n=1}^N\right)$ ▷ Eq. (25)
$p\left(Y_n \mid Z_{1:N}\right) = \mathcal{N}\left(Y_n; \mu_n^f, \Sigma_n^f\right) \leftarrow \mathcal{N}\left(Y_n; b_n^f, C_n^f\right)$ ▷ Filtering marginals

3: *Compute the smoothing elements:*
$b_n = (E_n, g_n, \sqrt{L_n})$ for all $n = 0, \ldots, N$ ▷ Eq. (28)

4: *Run the time-parallel Rauch–Tung–Striebel smoother:*
$\left\{\left(E_n^s, g_n^s, \sqrt{L_n^s}\right)\right\}_{n=1}^N \leftarrow \texttt{ReverseAssociativeScan}\left(\otimes_s, (b_n)_{n=1}^N\right)$ ▷ Eq. (31)

Output: Smoothing marginals $p(Y_n \mid Z_{1:N}) = \mathcal{N}\left(Y_n; g_n^s, L_n^s\right)$

This concludes the parallel-in-time probabilistic numerical ODE solver for affine ODEs: since affine ODEs result in state-estimation problems with affine state-space models, as discussed in the beginning of this section, the parallel-time Rauch–Tung–Striebel smoother presented here can be used to solve affine ODEs in parallel time.

3.2 Parallel-Time Approximate Inference in Nonlinear Vector Fields

Let us now consider the general case: An IVP with nonlinear vector field

$$\dot{y}(t) = f(y(t), t), \quad t \in [0, T], \qquad y(0) = y_0. \tag{33}$$

As established in Section 2, the corresponding state estimation problem is

$$Y_0 \sim \mathcal{N}(\mu_o, \Sigma_0), \tag{34a}$$
$$Y_{n+1} \mid Y_n \sim \mathcal{N}(\Phi_n Y_n, Q_n), \tag{34b}$$
$$Z_n \mid Y_n \sim \delta(E_1 Y_n - f(E_0 Y_n, t_n)), \tag{34c}$$

with temporal discretization $\mathbb{T} = \{t_n\}_{n=1}^N \subset [0, T]$ and zero data $Z_n = 0$ for all $n = 1, \ldots, N$. In this section, we describe a parallel-in-time algorithm for solving this state estimation problem: the *iterated extended Kalman smoother* (IEKS).

3.2.1 Globally Linearizing the State-Space Model

To make inference tractable, we will linearize the whole state-space model along a reference trajectory. And since the observation model (specified in Equation (34c)) is the only nonlinear part of the state-space model, it is the only part that requires linearization. In this paper, we only consider linearization with a first-order Taylor expansion, but other methods are possible; see Remarks 4 and 5.

For any time-point $t_n \in \mathbb{T}$, we approximate the nonlinear observation model

$$Z_n \mid Y_n \sim \delta(E_1 Y_n - f(E_0 Y_n, t_n)) \tag{35}$$

with an affine observation model by performing a first-order Taylor series expansion around a linearization point $\eta_n \in \mathbb{R}^{d(\nu+1)}$. We obtain the affine model

$$Z_n \mid Y_n \sim \delta(H_n Y_n - d_n), \tag{36}$$

with H_n and d_n defined as

$$H_n \coloneqq E_1 - F_y(E_0\eta_n, t_n)E_0, \tag{37a}$$
$$d_n \coloneqq f(E_0\eta_n, t_n) - F_y(E_0\eta_n, t_n)E_0\eta_n, \tag{37b}$$

where F_y denotes the Jacobian of f with respect to y.

In the IEKS, this linearization is performed *globally* on all time steps simultaneously along a trajectory of linearization points $\{\eta_n\}_{n=1}^N \subset \mathbb{R}^{d(\nu+1)}$. We obtain the following linearized inference problem:

$$Y_0 \sim \mathcal{N}(\mu_o, \Sigma_0), \tag{38a}$$
$$Y_{n+1} \mid Y_n \sim \mathcal{N}(\Phi_n Y_n, Q_n), \tag{38b}$$
$$Z_n \mid Y_n \sim \delta(H_n Y_n - d_n), \tag{38c}$$

with zero data $Z_n = 0$ for all $n = 1, \ldots, N$. This is now a linear state-space model with linear Gaussian observations. It can therefore be solved exactly with the numerically stable, time-parallel Kalman filter and smoother presented in Section 3.1.

Remark 4 (Linearizing with approximate Jacobians (`EK0` & `DiagonalEK1`))**.** *To reduce the computational complexity with respect to the state dimension of the ODE, the vector field can also be linearized with an approximate Jacobian. Established choices include $F_y \approx 0$ and $F_y \approx \mathrm{diag}(\nabla_y f)$, which result in probabilistic ODE solvers known as the* `EK0` *and* `DiagonalEK1`*, respectively. See Krämer et al. (2022) for more details.*

Remark 5 (Statistical linear regression)**.** *Statistical linear regression (SLR) is a more general framework for approximating conditional distributions with affine Gaussian distributions, and many well-established filters can be understood as special cases of SLR. This includes notably the Taylor series expansion used in the EKF/EKS, but also sigma-point methods such as the unscented Kalman filter and smoother (Julier et al., 2000; Julier and Uhlmann, 2004; Särkkä, 2008), and more. For more information on SLR-based filters and smoothers refer to Särkkä and Svensson (2023, Chapter 9).*

3.2.2 Iterated Extended Kalman Smoothing

The IEKS (Bell, 1994; Särkkä and Svensson, 2023) is an approximate Gaussian inference method for nonlinear state-space models, which iterates between linearizing the state-space model along the current best-guess trajectory and computing a new state trajectory estimate by solving the linearized model exactly. It can equivalently also be seen as an efficient implementation of the Gauss–Newton method, applied to maximizing the posterior density of the state trajectory (Bell, 1994). This also implies that the IEKS computes not just some Gaussian estimate, but the *maximum a posteriori* (MAP) estimate of the state trajectory. In the context of probabilistic numerical ODE solvers, the IEKS has been previously explored by Tronarp et al. (2021), and the resulting MAP estimate has been shown to satisfy polynomial convergence rates to the true ODE solution. Here, we formulate an IEKS-based probabilistic ODE solver in a parallel-in-time manner, by exploiting the time-parallel formulation of the Kalman filter and smoother from Section 3.1.

The IEKS is an iterative algorithm, which starts with an initial guess of the state trajectory and then iterates between the following two steps:

1. *Linearization step:* Linearize the state-space model along the current best-guess trajectory. This can be done independently for each time step and is therefore fully parallelizable.

2. *Linear smoothing step:* Solve the resulting linear state-space model exactly with the time-parallel Kalman filter and smoother from Section 3.1.

The algorithm terminates when a stopping criterion is met, for example when the change in the MAP estimate between two iterations is sufficiently small. A pseudo-code summary of the method is provided in Algorithm 2.

As with the sequential filtering-based probabilistic ODE solvers as presented in Section 2, the mean and covariance of the initial distribution $Y(0) \sim \mathcal{N}(\mu_0, \Sigma_0)$ are chosen such that μ_0 corresponds to the exact solution of the ODE and its derivatives and Σ_0 is set to zero; see also Krämer and Hennig (2020). The initial state trajectory estimate $\{\eta_n\}_{n=0}^N$ is chosen to be constant, that is, $\eta_n = \mu_0$ for all $n = 0, \ldots, N$. Note that since only $E_0\eta_n$ is required to perform the linearization, it could equivalently be set to $\eta_n = [y_0, 0, \ldots, 0]$ for all n.

Finally, the stopping criterion should be chosen such that the algorithm terminates when the MAP estimate of the state trajectory has converged. In our experiments, we chose a combination of two criteria: (i) the change in the state trajectory estimate between two iterations is sufficiently small, or (ii) the change in the *objective value* between two iterations is sufficiently small, where the objective value is defined as the negative log-density of the

Algorithm 2 IEKS-based Parallel-in-Time Probabilistic ODE Solver

Input: ODE-IVP (f, y_0), prior transition model $\{(\Phi_n, Q_n)\}_{n=1}^N$, time grid $\{t_n\}_{n=0}^N \subset [0, T]$.

$\mu_0 \leftarrow$ `ComputeExactInitialState`(f, y_0) ▷ With automatic differentiation
$\Sigma_0 \leftarrow 0$
$\eta_n \leftarrow \mu_0$ for all $n = 0, \ldots, N$ ▷ Constant initial guess of state trajectory
while stopping criterion not met **do**
  for $n = 1, \ldots, N$ **do** ▷ Can be done fully in parallel
    $H_n, d_n \leftarrow$ `LinearizeObservationModel`(f, η_n, t_n) ▷ As in Sec. 3.2.1
  end for
  $\{\mu_n, \Sigma_n\}_{n=1}^N \leftarrow$ `ParRTS` $\left((\mu_o, \Sigma_0), \{(\Phi_n, Q_n)\}_{n=1}^N, \{(H_n, d_n)\}_{n=1}^N\right)$ ▷ As in Sec. 3.1
  $\eta_n \leftarrow \mu_n$ for all $n = 1, \ldots, N$
end while

Output: $y(t_n) \sim \mathcal{N}\left(E_0\mu_n, E_0\Sigma_n E_0^\mathsf{T}\right)$ for $n = 1, \ldots, N$.

state trajectory:

$$\mathcal{V}(\eta_{0:N}) = \frac{1}{2}\sum_{n=1}^{N} \|\eta_n - \Phi(h_n)\eta_{n-1}\|^2_{Q^{-1}(h)} \,. \tag{39}$$

In our experiments, we use a relative tolerance of 10^{-13} for the first criterion and absolute and relative tolerances of 10^{-9} and 10^{-6} for the second criterion, respectively.

3.3 Computational Complexity of the Time-Parallel Probabilistic ODE Solver

The standard, sequential formulation of a Kalman smoother has a computational cost that scales linearly in the number of data points N, of the form

$$C^s_{\text{KS}} = N \cdot \left(C^s_{\text{predict}} + C^s_{\text{update}} + C^s_{\text{smooth}}\right), \tag{40}$$

where $C^s_{\text{predict}}, C^s_{\text{update}}, C^s_{\text{smooth}}$ are the costs of the sequential formulation of the predict, update, and smoothing steps, respectively. For nonlinear models, the extended Kalman filter/smoother linearizes the observation model sequentially at each prediction mean. With $C_{\text{linearize}}$ the cost of linearization, which requires evaluating the vector field and computing its Jacobian, the cost for a sequential extended Kalman smoother becomes

$$C^s_{\text{EKS}} = N \cdot \left(C^s_{\text{predict}} + C_{\text{linearize}} + C^s_{\text{update}} + C^s_{\text{smooth}}\right). \tag{41}$$

The proposed IEKS differs in two ways: (i) the prefix-sum formulation of the Kalman smoother enables a time-parallel inference with logarithmic complexity, and (ii) the linearization is not done locally in a sequential manner but can be performed globally, fully in parallel. Assuming a large enough number of processors / threads, the span cost of a single parallelized IEKS iteration becomes

$$C^p_{\text{EKS}} = C_{\text{linearize}} + \log(N) \cdot \left(C^p_{\text{filter}} + C^p_{\text{smooth}}\right), \tag{42}$$

where $C^p_{\text{filter}}, C^p_{\text{smooth}}$ are the costs of the associative filtering and smoothing operation as used in the *parallel* Kalman filter formulation, respectively. They differ from the costs of the sequential formulation in a constant manner.

4. Experiments

This section investigates the utility and performance of the proposed parallel IEKS-based ODE filter on a range of experiments. It is structured as follows: First, Section 4.1 investigates the runtime of a single IEKS step in its sequential and parallel formulation, over a range of grid sizes and for different GPUs. Section 4.2 then compares the performance of both ODE solver implementations on multiple test problems. Finally, Section 4.3 benchmarks the proposed method against other well-established ODE solvers, including both classic and probabilistic numerical methods.

Implementation All experiments are implemented in the Python programming language with the JAX software framework (Bradbury et al., 2018). Reference solutions are computed with SciPy (Virtanen et al., 2020) and Diffrax (Kidger, 2021). Unless specified otherwise, experiments are run on an NVIDIA V100 GPU. Code for the implementation and experiments is publicly available on GitHub.[1]

4.1 Runtime of a Single Extended Kalman Smoother Step

We first evaluate the runtime of the proposed method for only a single IEKS iteration, which consists of one linearization of the model along a trajectory and one extended Kalman smoother step. To this end, we consider the logistic ordinary differential equation

$$\dot{y}(t) = y(t)\left(1 - y(t)\right), \qquad t \in [0, 10], \qquad y(0) = 0.01; \tag{43}$$

though, since here we only investigate the runtime of a single IEKS iteration and thus do not actually solve the problem by iteratively re-linearizing, the precise choice of ODE is not very important. We then compare the runtime of the sequential and parallel EKS formulation for different grid sizes, resulting from time discretizations with step sizes $h = 2^0, 2^1, \ldots, 2^{14}$, and for multiple GPUs with varying numbers of CUDA cores. Figure 1 shows the results.

First, we observe the expected logarithmic scaling of the parallel EKS with respect to the grid size, for grids of size up to around $\sim 5 \cdot 10^3$ (Figure 1a). For larger grid sizes the runtime of the parallel EKS starts to grow linearly. This behaviour is expected: The NVIDIA V100 GPU used in this experiment has only 5120 CUDA cores, so for larger grids the filter and smoother pass can not be fully parallelized anymore and additional grid points need to be processed sequentially. But, the overall runtime of the parallel EKS is still significantly lower than the runtime of the sequential EKS throughout all grid sizes.

Figure 1b. shows runtimes for different GPUs with varying numbers of CUDA cores for a grid of size $N = 81920$. We observe that both the sequential EKS, as well as the classic Dopri5 and Kvaerno5 solvers (Dormand and Prince, 1980; Shampine, 1986; Kværnø, 2004), do not show a benefit from the improved GPU hardware. This is expected as these methods do not explicitly aim to leverage parallelizaton. On the other hand, the runtime of the parallel EKS decreases as the number of CUDA cores increases, and we observe speed-ups of up to an order of magnitude by using a different GPU. Be reminded once more that these evaluations only considered a single IEKS step, so they do not show the runtimes for computing the actual probabilistic numerical ODE solutions—these will be the subject of interest in the next sections.

1. `https://github.com/nathanaelbosch/parallel-in-time-ode-filters`

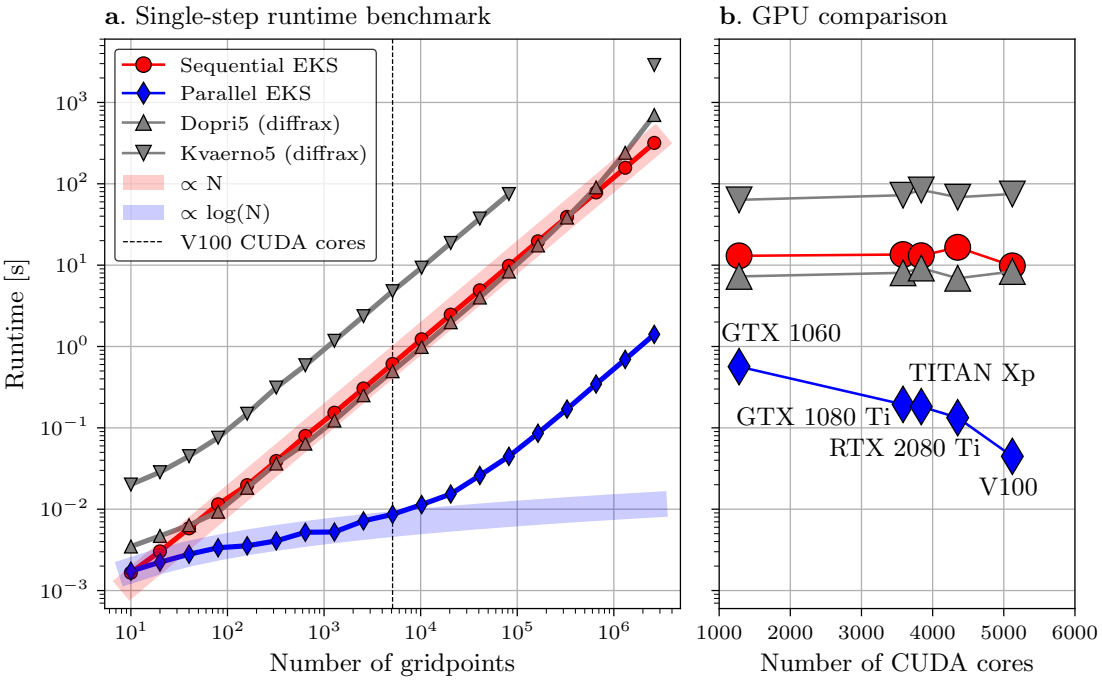


Figure 1: *The parallel EKS shows logarithmic scaling and benefits from GPU improvements.* In comparison, the sequential EKS and the classic Dopri5 and Kvaerno5 solvers show the expected linear runtime complexity (left). They also do not show relevant changes in runtime for GPUs with higher numbers of CUDA cores (right).

4.2 The Parallel-IEKS ODE Filter Compared to its Sequential Version

In this experiment we compare the proposed parallel-in-time ODE solver to a probabilistic solver based on the sequential implementation of the IEKS. In addition to the logistic ODE as introduced in Equation (43), we consider two more problems: An initial value problem based on the rigid body dynamics (Hairer et al., 1993)

$$\dot{y}(t) = \begin{bmatrix} -2y_2(t)y_3(t) \\ 1.25y_1(t)y_3(t) \\ -0.5y_1(t)y_2(t) \end{bmatrix}, \qquad t \in [0, 20], \qquad y(0) = \begin{bmatrix} 1 \\ 0 \\ 0.9 \end{bmatrix}, \tag{44}$$

and the Van der Pol oscillator (Van der Pol, 1920)

$$\dot{y}(t) = \begin{bmatrix} y_2(t) \\ \mu\left(\left(1 - y_1(t)^2\right) y_2(t) - y_1(t)\right) \end{bmatrix}, \qquad t \in [0, 6.3], \qquad y(0) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \tag{45}$$

here in a non-stiff version with parameter $\mu = 1$.

We first solve the three problems with the parallel IEKS on grids of sizes 30, 200, and 100, respectively for the logistic, rigid body, and Van der Pol problem, with a two-times integrated Wiener process prior. Reference solutions are computed with diffrax's Kvaerno5

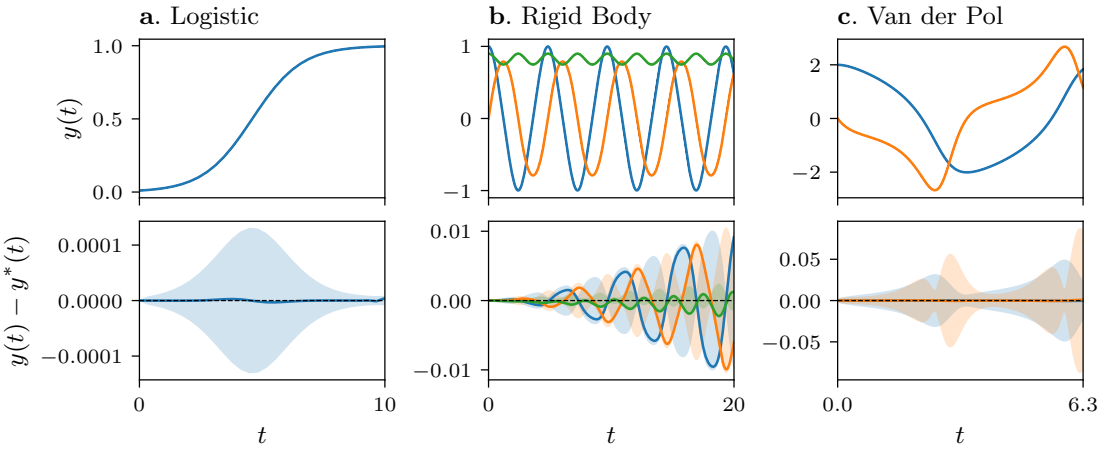


Figure 2: *Trajectories, errors, and error estimates computed by the parallel-in-time solver.* Top row: ODE solution trajectories. Visually, all three test problems seem to be solved accurately. Bottom row: Numerical errors (lines) and error estimates (shaded area). Ideally, for good calibration, the error should be of similar magnitude than the error estimate. The posterior appears underconfident on the logistic and Van der Pol ODE, and reasonably confident for the rigid body problem.

solver using adaptive steps and very low tolerances $\tau_{\{\text{abs,rel}\}} = 10^{-12}$ (Kidger, 2021; Kværnø, 2004). Figure 2 shows the resulting solution trajectories, together with numerical errors and error estimates. For these grid sizes, the parallel IEKS computes accurate solutions on all three problems. Regarding calibration, the posterior appears underconfident for the logistic and Van der Pol problems as it overestimates the numerical error by more than one order of magnitude. This is likely not due to the proposed method itself as underconfidence of ODE filters in low-error regimes has been previously observed (Bosch et al., 2021). For the rigid body problem, the posterior appears reasonably confident and the error estimate is of similar magnitude as the numerical error.

Next, we investigate the performance of the parallel IEKS and compare it to its sequential implementation. We solve the three problems with the parallel and sequential IEKS on a range of grid sizes, with both a one- and two-times integrated Wiener process prior. Reference solutions are computed with diffrax's Kvaerno5 solver using adaptive steps and very low tolerances ($\tau_{\text{abs}} = 10^{-16}, \tau_{\text{rel}} = 10^{-13}$). Figure 3 shows the achieved root-mean-square errors (RMSE) for different grid sizes in a work-precision diagram. As expected, both the parallel and the sequential IEKS always achieve the same error for each problem and grid size, as both versions compute the same quantities and only differ in their implementation. However, the methods differ significantly in actual runtime, as shown in Figure 4. In our experiments on an NVIDIA V100 GPU, the parallel IEKS is always strictly faster than the sequential implementation across all problems, grid sizes, and priors, and we observe speed-ups of multiple orders of magnitude. Thus, when working with a GPU, the parallel IEKS appears to be strictly superior to the sequential IEKS.

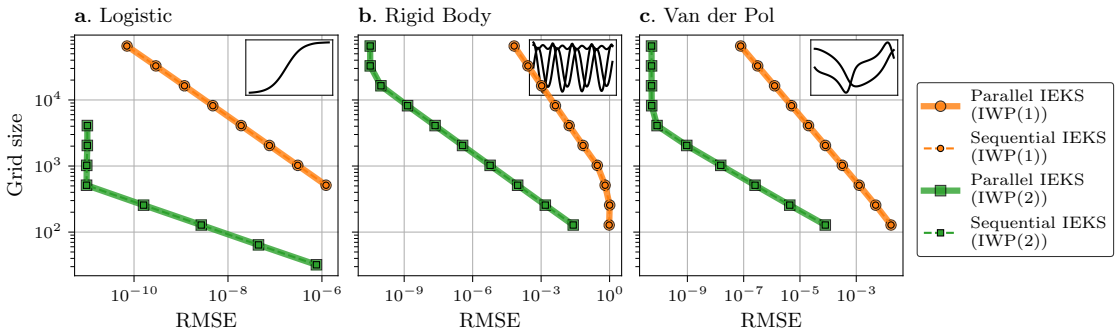


Figure 3: *The sequential and parallel IEKS compute numerically identical solutions.* For all three problems and all considered grid sizes, the sequential and parallel IEKS achieve (numerically) identical errors. This is expected, as both versions compute the same quantities and only differ in their implementation.

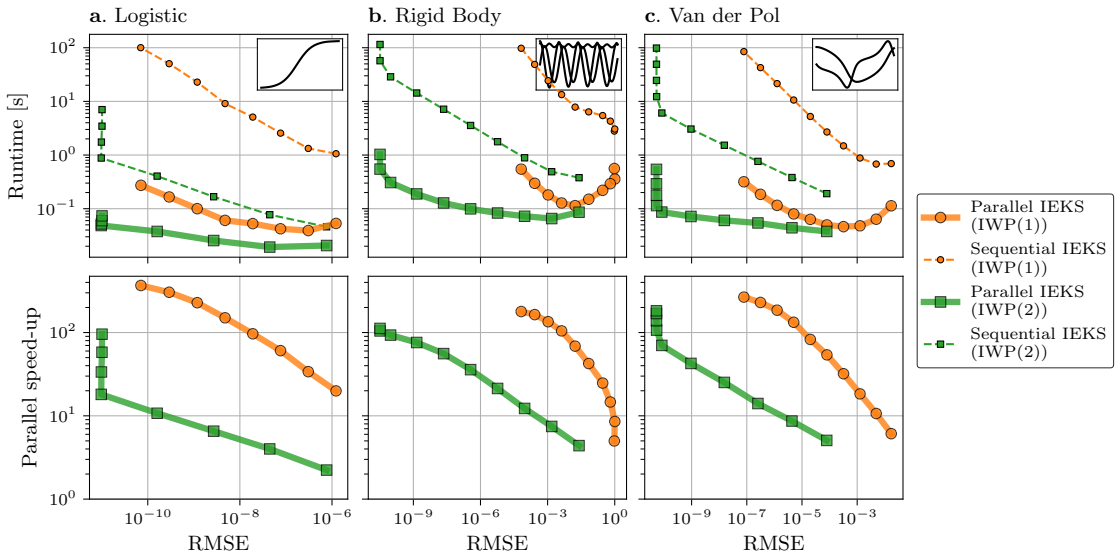


Figure 4: *Work-precision diagrams for the sequential and parallel IEKS-based ODE solver.* Top row: Runtime in seconds per error (lower-left is better). Bottom row: Speed-up of the parallel over the sequential IEKS (higher is better). Accross all problems, grid sizes, and priors, the parallel IEKS outperforms the sequential IEKS.

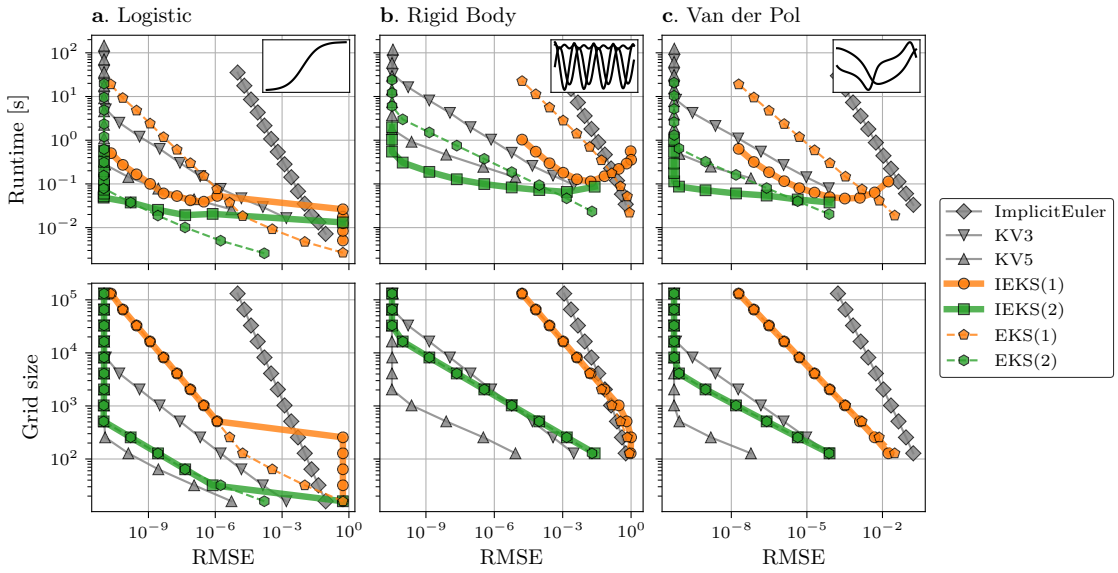


Figure 5: *Benchmarking the parallel IEKS against other common numerical ODE solvers.* Top row: Work-precision diagrams showing runtimes per error for a range of different ODE solvers (lower-left is better). Bottom row: Errors per specified grid size (lower-left is better). Per grid size, the closely related EKS and IEKS solvers often coincide; KV5 achieves the lowest error per step as it has the highest order. In terms of runtime, the IEKS outperforms both the EKS and KV5 on medium-to-high accuracy settings due to its logarithmic time complexity.

4.3 Benchmarking the Parallel-IEKS ODE Filter

Finally, we compare the proposed method to a range of well-established ODE solvers, including both classic and probabilistic numerical methods: we compare against the implicit Euler method, the `Kvaerno3` (KV3) and `Kvaerno5` (KV5) solvers (Kværnø, 2004) provided by Diffrax (Kidger, 2021), as well as the sequential EKS with local linearization, which is one of the currently most popular probabilistic ODE solvers. Note that since the IEKS is considered to be an implicit solver (Tronarp et al., 2021), we only compare to other implicit and semi-implicit methods, and therefore neither include explicit Runge–Kutta methods nor the EKS with zeroth order linearization (also known as EK0) in our comparison. Reference solutions are computed with diffrax's `Kvaerno5` solver with adaptive steps and a very low error tolerance setting ($\tau_{\text{abs}} = 10^{-16}, \tau_{\text{rel}} = 10^{-13}$).

Figure 5 shows the results as work-precision diagrams. For small grid sizes (low accuracy), the logarithmic time complexity of the parallel IEKS seems to not be very relevant and the IEKS is outperformed by the non-iterated EKS. In the particular case of the logistic ODE, it further seems that the MAP estimate differs significantly from ODE solution and thus the error on coarse grids is high (lower left figure). However, for larger grid sizes (medium-to-high accuracy), the parallel IEKS outperforms both its sequential, non-iterated counterpart, as

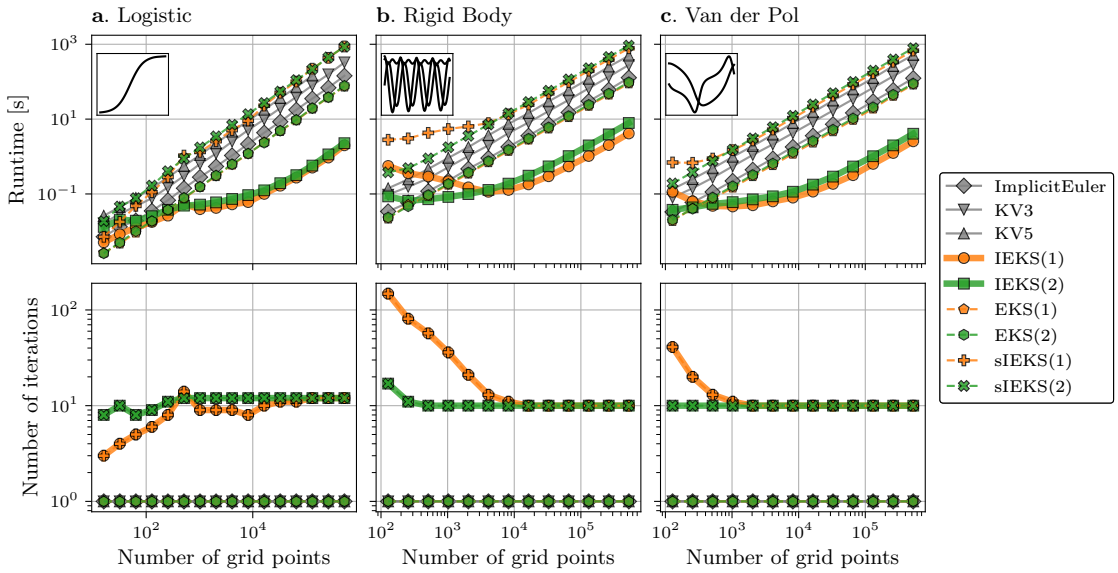


Figure 6: *Runtimes of the ODE solvers for each grid size, and number of IEKS iterations.* While all sequential solvers demonstrate linear scaling with the number of grid points, the parallel IEKS shows sub-linear scaling up to a certain grid size (top). The number of IEKS iterations until convergence can vary with the grid size and the problem, but it seems that in many cases ˜10 iterations suffice (bottom). The sequential methods solve the ODE in one sweep.

well as the classic methods. In particular, the parallel IEKS with IWP(2) prior often shows runtimes lower than those of the classic KV5 method, even though it has a lower order of convergence and is an iterative method; see also Figure 6 for runtimes per grid size and for the number of iterations performed by the IEKS. Overall, the logarithmic time complexity of the proposed parallel IEKS appears to be very beneficial for high accuracy settings on GPUs and makes the parallel IEKS a very competitive ODE solver in this comparison.

5. Conclusion

In this work, we have developed a *parallel-in-time* probabilistic numerical ODE solver. The method builds on iterated extended Kalman smoothing to compute the maximum a posteriori estimate of the probabilistic ODE solution, and by using the time-parallel formulation of the IEKS it is able to efficiently leverage modern parallel computer hardware such as GPUs to parallelize its computations. Given enough processors or cores, the proposed algorithm shares the logarithmic cost per time step of the parallel IEKS and the underlying parallel prefix-sum algorithm, as opposed to the linear time complexity of standard, sequentially-operating ODE solvers. We evaluated the performance of the proposed method in a number of experiments, and have seen that the proposed parallel-in-time solver can provide speed-ups of multiple

orders of magnitude over the sequential IEKS-based solver. We also compared the proposed method to a range of well-established, both probabilistic and classical ODE solvers, and we have shown that the proposed parallel-in-time method is competitive with respect to the state-of-the-art in both accuracy and runtime.

This work opens up a number of interesting avenues for future research in the intersection of probabilistic numerics and parallel-in-time methods. Potential opportunities for improvement include the investigation of other optimization algorithms, such as Levenberg–Marquart or ADMM, or the usage of line search, all of which have been previously proposed for the sequential IEKS. Furthermore, combining the solver with adaptive grid refinement approaches could also significantly improve its performance in practice. A different avenue would be to extend the proposed method to other related differential equation problems for which sequentially-operating probabilistic numerical methods already exist, such as higher-order ODEs, differential-algebraic equations, or boundary value problems. Finally, the improved utilization of GPUs by our parallel-in-time method could be particularly beneficial to applications in the field of machine learning, where GPUs are often required to accelerate the computations of deep neural networks. In summary, the proposed parallel-in-time probabilistic numerical ODE solver not only advances the efficiency of probabilistic numerical ODE solvers, but also paves the way for a range of future research on parallel-in-time probabilistic numerical methods and their application across various scientific domains.

Acknowledgments

The authors gratefully acknowledge financial support by the German Federal Ministry of Education and Research (BMBF) through Project ADIMEM (FKZ 01IS18052B), and financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. The authors would like to thank Research Council of Finland for funding. Filip Tronarp was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Nathanael Bosch. The authors are grateful to Nicholas Krämer for many valuable discussion and to Jonathan Schmidt for feedback on the manuscript.

Individual Contributions

The original idea for this article came independently from SS and from discussions between FT and NB. The joint project was initiated and coordinated by SS and PH. The methodology was developed by NB in collaboration with AC, FT, PH, and SS. The implementation is primarily due to NB, with help from AC. The experimental evaluation was done by NB with support from FT and PH. The first version of the article was written by NB, after which all authors reviewed the manuscript.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

U. M. Ascher, R. M. M. Mattheij, and R. D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. Society for Industrial and Applied Mathematics, 1995. doi: 10.1137/1.9781611971231.

B. M. Bell. The iterated Kalman smoother as a Gauss–Newton method. *SIAM Journal on Optimization*, 4(3):626–636, 1994.

J. Bettencourt, M. J. Johnson, and D. Duvenaud. Taylor-mode automatic differentiation for higher-order derivatives in JAX. In *Program Transformations for ML Workshop at NeurIPS 2019*, 2019.

G. Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38 (11):1526–1538, 1989. doi: 10.1109/12.42122.

N. Bosch, P. Hennig, and F. Tronarp. Calibrated adaptive probabilistic ODE solvers. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3466–3474. PMLR, 2021.

N. Bosch, F. Tronarp, and P. Hennig. Pick-and-mix information operators for probabilistic ODE solvers. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 10015–10027. PMLR, 2022.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

J. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2016. ISBN 9781119121503.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

P. Deuflhard and F. Bornemann. *Scientific computing with ordinary differential equations*, volume 42. Springer Science & Business Media, 2012.

J. R. Dormand and P. J. Prince. A family of embedded Runge–Kutta formulae. *Journal of Computational and Applied Mathematics*, 6:19–26, 1980.

M. J. Gander. 50 years of time parallel time integration. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113, Cham, 2015. Springer International Publishing. ISBN 978-3-319-23321-5.

M. J. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29(2):556–578, 2007. doi: 10.1137/05064607X.

A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in applied mathematics. Society for Industrial and Applied Mathematics, 2000. ISBN 9780898714517.

E. Hairer, S. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8. Springer-Verlag, 1993. ISBN 978-3-540-56670-0. doi: 10.1007/978-3-540-78862-1.

U. Helmke, R. Brockett, and J. Moore. *Optimization and Dynamical Systems*. Communications and Control Engineering. Springer London, 2012. ISBN 9781447134671.

P. Hennig and S. Hauberg. Probabilistic solutions to differential equations and their application to riemannian statistics. In S. Kaski and J. Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 347–355. PMLR, 2014.

P. Hennig, M. A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings. Mathematical, physical, and engineering sciences*, 471, 2015.

P. Hennig, M. A. Osborne, and H. P. Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022. doi: 10.1017/9781316681411.

S. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. doi: 10.1109/JPROC.2003.823141.

S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000. doi: 10.1109/9.847726.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. ISSN 0021-9223. doi: 10.1115/1.3662552.

H. Kersting and P. Hennig. Active uncertainty calibration in Bayesian ODE solvers. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 309–318, 2016.

H. Kersting, T. J. Sullivan, and P. Hennig. Convergence rates of Gaussian ODE filters. *Statistics and computing*, 30(6):1791–1816, 2020.

P. Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

N. Krämer and P. Hennig. Stable implementation of probabilistic ODE solvers. *arXiv:2012.10106*, 2020.

N. Krämer and P. Hennig. Linear-time probabilistic solution of boundary value problems. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

N. Krämer, N. Bosch, J. Schmidt, and P. Hennig. Probabilistic ODE solutions in millions of dimensions. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11634–11649. PMLR, 2022.

N. Krämer, J. Schmidt, and P. Hennig. Probabilistic numerical method of lines for time-dependent partial differential equations. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 625–639. PMLR, 2022.

A. Kværnø. Singly diagonally implicit Runge–Kutta methods with an explicit first stage. *BIT Numerical Mathematics*, 44(3):489–502, 2004.

J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps "pararéel". *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332(7):661–668, 2001. ISSN 0764-4442.

C. J. Oates and T. J. Sullivan. A modern retrospective on probabilistic numerics. *Statistics and Computing*, 29, 2019.

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(1), 2021. ISSN 1532-4435.

K. Pentland, M. Tamborrino, D. Samaddar, and L. C. Appel. Stochastic parareal: An application of probabilistic methods to time-parallelization. *SIAM Journal on Scientific Computing*, 0(0):S82–S102, 2021. doi: 10.1137/21M1414231.

K. Pentland, M. Tamborrino, T. J. Sullivan, J. Buchanan, and L. C. Appel. GParareal: a time-parallel ODE solver using Gaussian process emulation. *Statistics and Computing*, 33 (1):23, 2022. ISSN 1573-1375. doi: 10.1007/s11222-022-10195-y.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965. ISSN 1533-385X. doi: 10.2514/3.3166.

M. S. Grewal and A. P. Andrews. Kalman filtering. 2014. doi: 10.1002/9781118984987.

S. Särkkä and A. F. García-Fernández. Temporal parallelization of Bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, 2021. doi: 10.1109/TAC.2020.2976316.

S. Särkkä and L. Svensson. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2023. ISBN 9781108912303.

M. Schober, S. Särkkä, and P. Hennig. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, 2019. ISSN 1573-1375. doi: 10.1007/s11222-017-9798-7.

L. F. Shampine. Some practical Runge–Kutta formulas. *Mathematics of Computation*, 46 (173):135–150, 1986. doi: https://doi.org/10.2307/2008219.

J. Skilling. *Bayesian Solution of Ordinary Differential Equations*, pages 23–37. Springer, 1992. ISBN 978-94-017-2219-3. doi: 10.1007/978-94-017-2219-3_2.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

W. Su, S. Boyd, and E. J. Candès. A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. *Journal of Machine Learning Research*, 17(153): 1–43, 2016.

S. Särkkä. Unscented Rauch–Tung–Striebel smoother. *IEEE Transactions on Automatic Control*, 53(3):845–849, 2008. doi: 10.1109/TAC.2008.919531.

S. Särkkä and A. Solin. *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2019. doi: 10.1017/9781108186735.

F. Tronarp, H. Kersting, S. Särkkä, and P. Hennig. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29(6):1297–1315, 2019.

F. Tronarp, S. Särkkä, and P. Hennig. Bayesian ODE solvers: The maximum a posteriori estimate. *Statistics and Computing*, 31(3):1–18, 2021.

B. Van der Pol. Theory of the amplitude of free and forced triode vibrations. *Radio Review*, 1:701–710, 1920.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

F. Yaghoobi, A. Corenflos, S. Hassan, and S. Särkkä. Parallel iterated extended and sigma-point Kalman smoothers. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5350–5354, 2021. doi: 10.1109/ICASSP39728.2021.9413364.

Parallel-in-Time Probabilistic Numerical ODE Solvers

F. Yaghoobi, A. Corenflos, S. Hassan, and S. Särkkä. Parallel square-root statistical linear regression for inference in nonlinear state space models. *arXiv:2207.00426*, 2023.

The Journal of Open Source Software

ProbNumDiffEq.jl: Probabilistic Numerical Solvers for Ordinary Differential Equations in Julia

Nathanael Bosch [1]

1 Tübingen AI Center, University of Tübingen, Germany

DOI: 10.21105/joss.07048

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @PieterjanRobbe
- @ranocha

Submitted: 19 July 2024
Published: 30 September 2024

Summary

Probabilistic numerical solvers have emerged as an efficient framework for simulation, uncertainty quantification, and inference in dynamical systems. In comparison to traditional numerical methods, which approximate the true trajectory of a system only by a single point estimate, probabilistic numerical solvers compute a *distribution* over the true unknown solution of the given differential equation and thereby provide information about the numerical error incurred during the computation. ProbNumDiffEq.jl implements such probabilistic numerical solvers for ordinary differential equations (ODEs) and differential-algebraic equations (DAEs) in the Julia programming language (Bezanson et al., 2017) within the DifferentialEquations.jl ecosystem (Rackauckas & Nie, 2017).

More concretely, ProbNumDiffEq.jl provides a range of probabilistic numerical solvers for ordinary differential equations based on Bayesian filtering and smoothing, which have emerged as a particularly efficient and flexible class of methods for solving ODEs (Kersting, Sullivan, et al., 2020; Schober et al., 2019; Tronarp et al., 2019). These so-called "ODE filters" have known polynomial convergence rates (Kersting, Sullivan, et al., 2020; Tronarp et al., 2021) and numerical stability properties (such as A-stability or L-stability) (Bosch, Hennig, et al., 2023; Tronarp et al., 2019), their computational complexity is comparable to traditional numerical methods (Krämer, Bosch, et al., 2022), they are applicable to a range of numerical differential equation problems (Bosch et al., 2022; Krämer, Schmidt, et al., 2022; Krämer & Hennig, 2021), and they can be formulated parallel-in-time (Bosch, Corenflos, et al., 2023). ODE filters also provide a natural framework for ODE parameter inference (Beck et al., 2024; Kersting, Krämer, et al., 2020; Schmidt et al., 2021; Tronarp et al., 2022). ProbNumDiffEq.jl implements many of the methods referenced above and packages them in a software library with the aim to be easy-to-use, feature-rich, well-documented, and efficiently implemented.

Statement of need

Filtering-based probabilistic numerical ODE solvers have been an active field of research for the past decade, but their application in practical simulation and inference problems has been limited. ProbNumDiffEq.jl aims to bridge this gap. ProbNumDiffEq.jl implements probabilistic numerical methods as performant, documented, and easy-to-use ODE solvers inside the well-established DifferentialEquations.jl ecosystem (Rackauckas & Nie, 2017). Thereby, the package benefits from the extensive testing, documentation, performance optimization, and functionality that DifferentialEquations.jl provides. Users can easily find help and examples regarding many features that are not particular to ProbNumDiffEq.jl in the DifferentialEquations.jl documentation, and we provide additional examples and tutorials specific to the probabilistic solvers in the ProbNumDiffEq.jl documentation. We believe that this deep integration within DifferentialEquations.jl is a key feature to attract users to probabilistic numerics by enabling the use of probabilistic ODE solvers as drop-in replacements for traditional ODE solvers.

The Journal of Open Source Software

On the other hand, ProbNumDiffEq.jl also aims to accelerate the development of new probabilistic numerical ODE solvers by providing a solid foundation to both build on and compare against. Several publications have been developed with ProbNumDiffEq.jl, including contributions on step-size adaptation and calibration of these solvers (Bosch et al., 2021), energy-preserving solvers and DAE solvers (Bosch et al., 2022), probabilistic exponential integrators (Bosch, Hennig, et al., 2023), and novel parameter inference algorithms (Beck et al., 2024; Tronarp et al., 2022). We also hope that by providing documented and performant implementations of published algorithms, we facilitate researchers to use these methods as baselines when developing new numerical solvers.

ProbNumDiffEq.jl is also the only software package in Julia, at the time of writing, that provides a comprehensive set of probabilistic numerical ODE solvers. Outside of Julia, two other software packages provide a similar functionality. ProbNum (Wenger et al., 2021) is a Python package that implements probabilistic numerical methods for various numerical problems, including linear systems, quadrature, and ODEs. ProbNum particularly aims to facilitate rapid experimentation and accelerate the development of new methods (Wenger et al., 2021). It is therefore broader in scope and provides functionality not covered by ProbNumDiffEq.jl, but it also lacks some of the specialized ODE solvers available in ProbNumDiffEq.jl. In addition, with its reliance on Python and NumPy (Harris et al., 2020) and the lack of just-in-time compilation, it is also generally less performant. ProbDiffEq (Krämer, 2023) is a probabilistic numerical ODE solver package built on JAX. At the time of writing, it provides a very similar set of ODE solvers as ProbNumDiffEq.jl with the addition of certain filtering and smoothing methods and the lack of certain specialized ODE solvers—but as both ProbDiffEq and ProbNumDiffEq.jl are under active development, this might change in the future. By building on JAX and leveraging its just-in-time compilation capabilities, ProbDiffEq provides ODE solvers with similar performance as those implemented in ProbNumDiffEq.jl (shown through benchmarks in both packages comparing to SciPy (Virtanen et al., 2020)). In summary, ProbNumDiffEq.jl provides one of the most feature-rich and performant probabilistic numerical ODE solver packages currently available and is the only one built on the Julia programming language.

Acknowledgements

The author gratefully acknowledges co-funding by the European Union (ERC, ANUBIS, 101123955. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them). The author thanks the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for their support.

I am grateful to Philipp Hennig for his support throughout the development of this package. I thank Nicholas Krämer, Filip Tronarp, and Jonathan Schmidt for many valuable discussions about probabilistic numerical ODE solvers and their correct, efficient, and elegant implementation. I also thank Christopher Rackauckas for support and feedback on how to integrate ProbNumDiffEq.jl with the DifferentialEquations.jl ecosystem and for including ProbNumDiffEq.jl into the testing pipeline of OrdinaryDiffEq.jl. I acknowledge contributions from Pietro Monticone (@pitmonticone), Vedant Puri (@vpuri3), Tim Holy (@timholy), Daniel González Arribas (@DaniGlez), David Widmann (@devmotion), Christopher Rackauckas (@ChrisRackauckas), Qingyu Qu (@ErikQQY), Cornelius Roemer (@corneliusroemer), and Jose Storopoli (@storopoli).

References

Beck, J., Bosch, N., Deistler, M., Kadhim, K. L., Macke, J. H., Hennig, P., & Berens, P. (2024). Diffusion tempering improves parameter estimation with probabilistic integrators for

ordinary differential equations. *Forty-First International Conference on Machine Learning*. https://openreview.net/forum?id=43HZG9zwaj

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Bosch, N., Corenflos, A., Yaghoobi, F., Tronarp, F., Hennig, P., & Särkkä, S. (2023). *Parallel-in-time probabilistic numerical ODE solvers*. https://doi.org/10.48550/arXiv.2310.01145

Bosch, N., Hennig, P., & Tronarp, F. (2021). Calibrated adaptive probabilistic ODE solvers. In A. Banerjee & K. Fukumizu (Eds.), *Proceedings of the 24th international conference on artificial intelligence and statistics* (Vol. 130, pp. 3466–3474). PMLR. http://proceedings.mlr.press/v130/bosch21a.html

Bosch, N., Hennig, P., & Tronarp, F. (2023). Probabilistic exponential integrators. *Thirty-Seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=2dx5MNs2Ip

Bosch, N., Tronarp, F., & Hennig, P. (2022). Pick-and-mix information operators for probabilistic ODE solvers. In G. Camps-Valls, F. J. R. Ruiz, & I. Valera (Eds.), *Proceedings of the 25th international conference on artificial intelligence and statistics* (Vol. 151, pp. 10015–10027). PMLR. https://proceedings.mlr.press/v151/bosch22a.html

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Kersting, H., Krämer, N., Schiegg, M., Daniel, C., Tiemann, M., & Hennig, P. (2020). Differentiable likelihoods for fast inversion of 'Likelihood-free' dynamical systems. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (Vol. 119, pp. 5198–5208). PMLR. http://proceedings.mlr.press/v119/kersting20a.html

Kersting, H., Sullivan, T. J., & Hennig, P. (2020). Convergence rates of Gaussian ODE filters. *Statistics and Computing*, *30*(6), 1791–1816. https://doi.org/10.1007/s11222-020-09972-4

Krämer, N. (2023). Probdiffeq: Probabilistic solvers for differential equations in JAX. In *GitHub repository*. GitHub. https://github.com/pnkraemer/probdiffeq

Krämer, N., Bosch, N., Schmidt, J., & Hennig, P. (2022). Probabilistic ODE solutions in millions of dimensions. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th international conference on machine learning* (Vol. 162, pp. 11634–11649). PMLR. https://proceedings.mlr.press/v162/kramer22b.html

Krämer, N., & Hennig, P. (2021). Linear-time probabilistic solution of boundary value problems. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (Vol. 34, pp. 11160–11171). Curran Associates, Inc. https://papers.nips.cc/paper/2021/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

Krämer, N., Schmidt, J., & Hennig, P. (2022). Probabilistic numerical method of lines for time-dependent partial differential equations. In G. Camps-Valls, F. J. R. Ruiz, & I. Valera (Eds.), *Proceedings of the 25th international conference on artificial intelligence and statistics* (Vol. 151, pp. 625–639). PMLR. https://proceedings.mlr.press/v151/kramer22a.html

Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl – A performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, *5*(1). https://doi.org/10.5334/jors.151

Schmidt, J., Krämer, N., & Hennig, P. (2021). A probabilistic state space model for joint

inference from differential equations and data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (Vol. 34, pp. 12374–12385). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2021/file/6734fa703f6633ab896eecbdfad8953a-Paper.pdf

Schober, M., Särkkä, S., & Hennig, P. (2019). A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, *29*(1), 99–122. https://doi.org/10.1007/s11222-017-9798-7

Tronarp, F., Bosch, N., & Hennig, P. (2022). Fenrir: Physics-enhanced regression for initial value problems. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th international conference on machine learning* (Vol. 162, pp. 21776–21794). PMLR. https://proceedings.mlr.press/v162/tronarp22a.html

Tronarp, F., Kersting, H., Särkkä, S., & Hennig, P. (2019). Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: A new perspective. *Statistics and Computing*, *29*(6), 1297–1315. https://doi.org/10.1007/s11222-019-09900-1

Tronarp, F., Särkkä, S., & Hennig, P. (2021). Bayesian ODE solvers: The maximum a posteriori estimate. *Statistics and Computing*, *31*(3), 23. https://doi.org/10.1007/s11222-021-09993-7

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wenger, J., Krämer, N., Pförtner, M., Schmidt, J., Bosch, N., Effenberger, N., Zenn, J., Gessner, A., Karvonen, T., Briol, F.-X., Mahsereci, M., & Hennig, P. (2021). *ProbNum: Probabilistic numerics in Python*. https://doi.org/10.48550/arXiv.2112.02100