Bayesian Filtering for Black Box Simulators

Master's Thesis

in partial fulfilment of the requirements for the degree

Master of Science in **Computer Science**

submitted by

Fabrice von der Lehr

December 2024

First Reviewer: Prof. Dr. Philipp Hennig

Second Reviewer: Prof. Dr. Jakob Macke

Wilhelm-Schickard-Institut für Informatik

Mathematisch-Naturwissenschaftliche Fakultät

Eberhard Karls Universität Tübingen

von der Lehr, Fabrice
6219374
Bayesian Filtering for Black Box Simulators
Master's Thesis *Computer Science*
University of Tübingen
Thesis period: *1 October 2024 - 31 March 2025*

Abstract

Ordinary differential equations (ODEs) are essential for modeling the behavior of complex dynamical systems in a variety of disciplines. In the absence of an analytical solution, however, the use of numerical integration methods, which are always subject to error, is often unavoidable. Thus, there is an interest in probabilistic solutions that take the uncertainty in numerical simulations into account. Another challenging problem is determining the model parameters that best explain experimental measurements of the real system. In the event of such needs, it might be difficult to replace a simulator that has previously been optimized for a specific practical application.

In this work, methods for both problems are developed that allow the usage of existing ODE solvers as a black box, provided that they are differentiable. For probabilistic solutions, an estimator for the local error is also required, which is usually available, e.g., in commonly used embedded Runge-Kutta methods. The two proposed methods then apply Bayesian filtering techniques, primarily the extended Kalman Filter, to perform inference in a probabilistic model over the ODE solution. While the first method models the uncertainty about the solution through estimates of the local error, the second method, which we call *process noise tempering*, gradually reduces the added noise during gradient-based optimization of the parameters, facilitating convergence to the global optimum.

An experimental evaluation shows that the produced probabilistic ODE solutions capture the structure of the uncertainty on a qualitative level, but are not always calibrated ideally. The use of overly large step sizes for simulation, however, can lead to catastrophic failure in the form of mode collapses. Process noise tempering, on the other hand, proves to estimate parameters reliably even for complex Hodgkin-Huxley models with more than ten parameters.

Zusammenfassung

Gewöhnliche Differentialgleichungen (ODEs) sind für die Modellierung des Verhaltens komplexer dynamischer Systeme in einer Vielzahl von Wissenschaften elementar. In Ermangelung einer analytischen Lösung ist die Verwendung numerischer Integrationsmethoden, die immer fehlerbehaftet ist, oft jedoch unvermeidlich. Daher besteht ein Interesse an probabilistischen Lösungen, die die Unsicherheit in numerischen Simulationen berücksichtigen. Eine weitere Herausforderung besteht im Bestimmen der Modellparameter, die experimentelle Messungen des realen Systems am besten erklären. Im Falle solcher Bedarfe kann es allerdings schwierig sein einen Simulator auszutauschen, wenn er zuvor für eine spezifische Praxisanwendung optimiert wurde.

Im Rahmen dieser Arbeit werden Methoden für beide Probleme entwickelt, die es erlauben, bestehende ODE-Löser als Black Box zu verwenden, vorausgesetzt, sie sind differenzierbar. Für probabilistische Lösungen wird zusätzlich ein Schätzer für den lokalen Fehler benötigt, der üblicherweise verfügbar ist, z.B. in weit verbreiteten eingebetteten Runge-Kutta-Methoden. Beide Methoden wenden dann Bayes'sche Filtertechniken an, vordergründig den erweiterten Kalman-Filter, um Inferenz in einem probabilistischen Modell über die ODE-Lösung durchzuführen. Während die erste Methode die Unsicherheit durch Schätzungen des lokalen Fehlers modelliert, verringert die zweite Methode, die wir *Prozessrauschtemperierung* nennen, schrittweise das hinzugefügte Rauschen während der gradientenbasierten Optimierung der Parameter, um die Konvergenz zum globalen Optimum zu begünstigen.

Eine experimentelle Auswertung zeigt, dass die erzeugten probabilistischen ODE-Lösungen die Struktur der Unsicherheit auf qualitativer Ebene erfassen, aber nicht immer ideal kalibriert sind. Die Verwendung von zu hohen Schrittweiten bei der Simulation kann jedoch zu katastrophalem Versagen in Form von kollabierenden Modi führen. Die Prozessrauschtemperierung hingegen erweist sich auch für komplexe Hodgkin-Huxley-Modelle mit mehr als zehn Parametern als zuverlässig in der Parameterschätzung.

Acknowledgements

First and foremost, I am deeply grateful to Tobias Weber for his supervision of this thesis, involving countless invaluable discussions and the willingness to always take time for my questions. Without your encouragement, patience, and support, this work would not have been possible. I would like to extend my gratitude to Prof. Philipp Hennig, who gave me the opportunity to conduct my thesis in his research group, and provided extremely helpful feedback throughout different phases of this project. Special thanks go to Nathanael Bosch for bringing in a multitude of ideas and helping me to look at various concepts from a different perspective. I am also grateful to Jens Beißwenger for his constructive comments during many joint lunches and coffee breaks as well as for final proofreading of this thesis. Next, I would like to acknowledge Jonas Beck and Michael Deistler for the insightful exchange of ideas on ODE parameter estimation and the Hodgkin-Huxley model. Last, my thanks go to Prof. Jakob Macke as the second reviewer of my thesis, and to Franziska Weiler for her assistance in administrative matters.

Besides, I am deeply indebted to my family and friends for their constant emotional support and for lending a sympathetic ear, in particular during difficult times. Here, I would like to explicitly express my gratitude to Stefan Altendorfer, Jens Beißwenger, Vanessa Meyer, as well as my flatmates Francesco Carnazza, Anabel Dickemann, and Moritz Malmede. Finally, many thanks go to my parents Heike and Holger von der Lehr as well as my sister Samira von der Lehr for their unconditional support and love. You always believe in me and enable me to get the best out of myself.

Contents

Introduction | 1

In many cases, it is straightforward to express knowledge about a dynamical system by the rate of change its state undergoes subject to the current state, yielding the formulation of ordinary differential equations (ODEs). A few examples, among many others, are the behavior of mechanical systems like celestial bodies (Musielak & Quarles, 2015) or the time evolution of chemical reactions (D. J. Higham, 2008). Only recently, another example of such a system – the spread of infectious diseases – received global attention, as modeling, simulating, and forecasting its dynamics formed one of the pillars for taking political measures and ultimately saving lives (Cooper et al., 2020).

Predicting a system's state over time requires integration of the corresponding ODE. However, this is only analytically tractable and therefore exact for a few simple cases. Most systems of interest exhibit rather complex nonlinear dynamics and necessitate numerical integration, which inevitably causes an error in the solution. Another problem is that ODE parameters are often unknown. Estimating them from observations of the state is difficult, as the objective function to be optimized is generally non-convex.

In recent years, the idea of rephrasing numerical algorithms as Bayesian inference problems has emerged, forming a new computational paradigm called *probabilistic numerics (PN)* (Hennig et al., 2022). The probabilistic framework allows modeling uncertainty over various quantities, both data and latent variables, in the formulation of a problem. This results in a probability measure over the space of possible solutions, in contrast to mere point estimates of the solution and its error, as provided by many classical numerical algorithms, e.g, ODE solvers.

However, there might be applications where problem-specific and highly optimized, but non-probabilistic ODE solvers are already in use. As those solvers cannot be easily exchanged, an additional demand of uncertainty quantification poses a challenge, motivating this work's approach: We treat the numerical ODE solver as a black box with only few requirements and build a probabilistic model around it by adding stochastic noise. Given the model, we apply Bayesian filtering techniques to solve the corresponding inference problem, and evaluate the calibration of the probabilistic solutions. Last, we investigate whether a related probabilistic model can also be used to reliably solve inverse problems, i.e., estimate parameters of ODEs.

1.1 | Notation

Unless otherwise indicated, mathematical notation in this work is defined as follows:

- The *natural numbers* $\mathbb{N}$ are defined as the *set of all positive integers*, i.e., $\mathbb{N} \coloneqq \{1, 2, 3, \ldots\}$.
- For a general set of numbers $\mathbb{X}$, notations $\mathbb{X}^+ \coloneqq \{x \in \mathbb{X} \mid x > 0\}$ and $\mathbb{X}^- \coloneqq \{x \in \mathbb{X} \mid x < 0\}$ denote *restrictions to positive* and *negative numbers*, respectively. Furthermore, $\mathbb{X}_0 \coloneqq \mathbb{X} \cup \{0\}$ refers to the *extension* of $\mathbb{X}$ *by zero*. The latter may be combined with one of the formers; in this case, the extension is performed after the restriction.
- *Scalars* are written in italics, either upper or lower case (e.g., α or N).
- *Vectors* are written in boldface and lowercase (e.g., $\mathbf{a}$). In general, they are defined as *column vectors*. The same applies to vector-valued functions.
- *Matrices* are written in boldface and uppercase (e.g., $\mathbf{A}$). The same applies to matrix-valued functions.
- Square brackets indicate elements of a vector, i.e., $[\mathbf{x}]_i \coloneqq x_i$ for a vector $\mathbf{x} = [x_1, \ldots, x_N]^\intercal \in \mathbb{R}^N$. Moreover, $[\mathbf{x}]_{i:j} \coloneqq [x_i, \ldots, x_j]^\intercal$ refers to a *slice* of the vector $\mathbf{x}$ from its i-th to the j-th component, including the latter. Analogous notations apply to matrices, with slices in different dimensions being separated by a comma.
- The vector or matrix consisting only of zeros and ones is denoted by $\mathbf{0}$ and $\mathbf{1}$, respectively. Its concrete shape is context-dependent. The identity matrix is written as $\mathbf{I}$.
- The function $\operatorname{diag}(\cdot)$ transforms a vector into a square matrix with the elements of the vector on the diagonal.
- The *matrix square root* $\mathbf{A}^{1/2}$ of a square matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is defined as any matrix that satisfies the factorization $\mathbf{A} = \left(\mathbf{A}^{1/2}\right)^\intercal \mathbf{A}^{1/2}$.
- Standard functions (e.g., $\exp(\cdot)$) and operators (e.g., $\frac{\mathrm{d}}{\mathrm{d}t}$) defined for scalar quantities are applied *element-wise* when applied to a vector or matrix.
- For a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, its *Jacobian* evaluated at $\mathbf{x} \in \mathbb{R}^n$, i.e., $\frac{\mathrm{d}\mathbf{f}(\mathbf{x})}{\mathrm{d}\mathbf{x}^\intercal} \in \mathbb{R}^{m \times n}$, is denoted by $\mathbf{J}\mathbf{f}(\mathbf{x})$. For functions with several arguments, the one differentiated against is included in subscript, e.g., $\mathbf{J}_\mathbf{x}\mathbf{f}(\mathbf{x})$.

1.2 | Outline

The remainder of this work is structured as follows:

Chapter 2 | Preliminaries: Introduction to ODEs, numerical ODE solvers, and Bayesian inference in state-space models. These are the fundamental theoretical building blocks the chosen approaches are based on.

Chapter 3 | Approach: Presentation of the approaches for probabilistic ODE solving and reliable ODE parameter estimation. They draw on black box simulators and Bayesian filtering methods.

Chapter 4 | Evaluation: Experimental evaluation of the methods developed in this work. The experiments are conducted with the aim to answer four research questions that are formulated at the beginning of this chapter.

Chapter 5 | Discussion: Critical assessment of the used methods, as well as an outlook to future work.

Chapter 6 | Related Work: Overview of recent research on probabilistic ODE solvers and ODE parameter estimation.

Chapter 7 | Conclusion: Summary of the present work.

Appendix A | Implementation and Computing: Details on the implementation and computing environment used for experiments.

Appendix B | The Hodgkin-Huxley Model: Details on the Hodgkin-Huxley model used in experiments on ODE parameter estimation.

Preliminaries | 2

The present chapter serves as a summary of the fundamental theory upon which this work is built. In particular, it establishes notations used throughout subsequent chapters.

As outlined in Chapter 1, the methods proposed in this work deal with black box ODE solvers. Consequently, Section 2.1 starts by introducing ordinary differential equations and initial value problems. Section 2.2 then presents an exemplary class of numerical ODE solvers, the celebrated Runge-Kutta methods. It also elaborates on ways to theoretically quantify and practically estimate errors of the numerical solutions. For the most part, Sections 2.1 and 2.2 are based on Hairer et al. (1993) and Iserles (2008).

Finally, since the probabilistic model used in this work fits into the broader class of state-space models (SSMs), Section 2.3 covers Bayesian inference in the context of SSMs. Here, the well-known (extended) Kalman filter is presented as an efficient method to perform inference in these models, provided that further assumptions are made about the problem setting. Section 2.3 is mainly based on Gharamani (2001) and Särkkä and Svensson (2023).

2.1 | Ordinary Differential Equations

Definition 2.1. An *ordinary differential equation (ODE) of order N* is an equation of the form

$$\frac{\mathrm{d}^N \mathbf{z}(t)}{\mathrm{d}t^N} = \mathbf{u}_\theta\left(t, \mathbf{z}(t), \frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t}, \frac{\mathrm{d}^2\mathbf{z}(t)}{\mathrm{d}t^2}, \ldots, \frac{\mathrm{d}^{N-1}\mathbf{z}(t)}{\mathrm{d}t^{N-1}}\right) \tag{2.1}$$

with a vector field $\mathbf{u}_\theta : \mathbb{T} \times \mathbb{R}^D \times \overset{N}{\cdots} \times \mathbb{R}^D \to \mathbb{R}^D$, parametrized by $\theta \in \mathbb{R}^W$. Here, $\mathbf{z}(t) := [z_1(t), z_2(t), \ldots, z_D(t)] \in \mathbb{R}^D$ denotes the D-dimensional *state* of the corresponding dynamical system, dependent on the *time* $t \in \mathbb{T}$.[1]

An ODE relates a system's state to its rate of change and possibly other, higher-order time derivatives. Therefore, it encodes the dynamics of the system. By known laws of nature, e.g., conservation laws, it is often straightforward to formulate ODEs that describe a system's behavior.

[1] In the following, we will assume $\mathbb{T} \subseteq \mathbb{R}_0^+$.

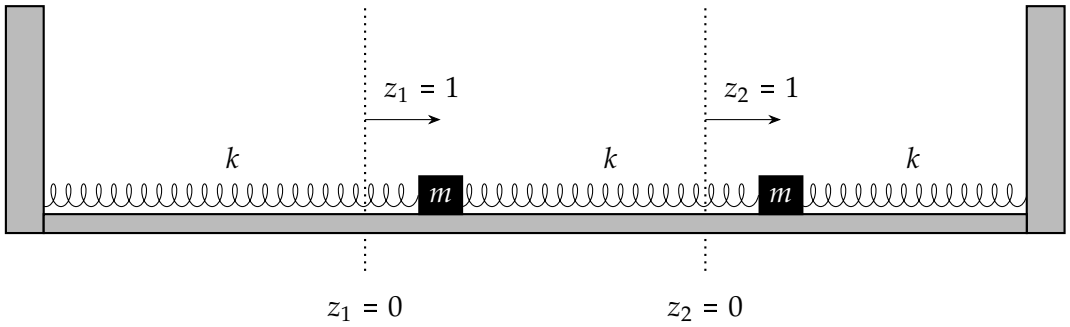


Figure 2.1: Two-degree-of-freedom mass-spring-system. Two identical masses m (■), attached to each other and two stationary walls via three identical horizontal springs (∿∿∿) of spring constant k. Frictionless system. Displacements z_1 and z_2 of the masses from equilibrium indicated by arrows (→).

Example 2.1 (Fitzpatrick (2018), Chapter 3)**.** Consider the following mechanical system, visualized in Fig. 2.1: Two identical masses m are attached to each other as well as to two stationary walls via three identical horizontal springs of spring constant k. The entire system is frictionless. Its state at time t is described by the displacements $z_1(t)$, $z_2(t)$ of the left and right mass. With $z_1 = z_2 = 0$ specifying the equilibrium state in which all springs are unexpended, their extensions at time t (from left to right) are $z_1(t)$, $z_2(t) - z_1(t)$, and $-z_2(t)$, respectively. The behavior of the system is then described by the two scalar second-order ODEs

$$m\frac{\mathrm{d}^2 z_1(t)}{\mathrm{d}t^2} = -kz_1(t) + k(z_2(t) - z_1(t)), \tag{2.2a}$$

$$m\frac{\mathrm{d}^2 z_2(t)}{\mathrm{d}t^2} = -k(z_2(t) - z_1(t)) + k(-z_2(t)). \tag{2.2b}$$

Equations (2.2a) and (2.2b) can be summarized as

$$\begin{bmatrix} \frac{\mathrm{d}^2 z_1(t)}{\mathrm{d}t^2} \\ \frac{\mathrm{d}^2 z_2(t)}{\mathrm{d}t^2} \end{bmatrix} = \begin{bmatrix} -2\frac{k}{m} & \frac{k}{m} \\ \frac{k}{m} & -2\frac{k}{m} \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix}, \tag{2.3}$$

which follows the form of Eq. (2.1) with parameters $\theta = \begin{bmatrix} k & m \end{bmatrix}^\mathsf{T}$.

In subsequent parts of this work, it will be necessary to view any ODE from a purely first-order perspective. To do that, we first establish stacked vector forms for the state and the vector field of an arbitrary N-th-order ODE. Then, we show that the latter can be rewritten as a system of N first-order ODEs, represented by a first-order ODE in stacked vector form.

Definition 2.2. Considering an ODE of order N in accordance to Definition 2.1, let

$$\left\{ \mathbf{z}^{(i)}(t) \coloneqq \frac{\mathrm{d}^i \mathbf{z}(t)}{\mathrm{d}t^i} \right\}_{i=0}^{N-1} \tag{2.4}$$

be a set of N auxiliary states, corresponding to the state $\mathbf{x}(t)$ and its time derivatives up to order $N-1$. Then, we define *stacked vector forms of the state and vector field* as

$$\mathbf{x}(t) \coloneqq \begin{bmatrix} \mathbf{z}^{(0)}(t) \\ \vdots \\ \mathbf{z}^{(N-1)}(t) \end{bmatrix} \in \mathbb{R}^{ND} \tag{2.5}$$

and

$$\mathbf{f}_\theta\left(t, \mathbf{x}(t)\right) \coloneqq \begin{bmatrix} \mathbf{z}^{(1)}(t) \\ \vdots \\ \mathbf{z}^{(N-1)}(t) \\ \mathbf{u}_\theta\left(t, \mathbf{z}^{(0)}(t), \ldots, \mathbf{z}^{(N-1)}(t)\right) \end{bmatrix} \in \mathbb{R}^{ND}, \tag{2.6}$$

respectively.

Proposition 2.1. *Every ODE of order N can be reformulated as a system of N first-order ODEs, expressed by a first-order ODE in stacked vector form*

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}_\theta\left(t, \mathbf{x}(t)\right). \tag{2.7}$$

Proof. We successively apply Definitions 2.1 and 2.2:

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} \overset{(2.5)}{=} \begin{bmatrix} \frac{\mathrm{d}\mathbf{z}^{(0)}(t)}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}\mathbf{z}^{(N-2)}(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}\mathbf{z}^{(N-1)}(t)}{\mathrm{d}t} \end{bmatrix} \overset{(2.4)}{=} \begin{bmatrix} \frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}^{N-1}\mathbf{z}(t)}{\mathrm{d}t^{N-1}} \\ \frac{\mathrm{d}^{N}\mathbf{z}(t)}{\mathrm{d}t^{N}} \end{bmatrix} \overset{(2.1)}{=} \begin{bmatrix} \frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}^{N-1}\mathbf{z}(t)}{\mathrm{d}t^{N-1}} \\ \mathbf{u}_\theta\left(t, \mathbf{z}(t), \frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t}, \frac{\mathrm{d}^2\mathbf{z}(t)}{\mathrm{d}t^2}, \ldots, \frac{\mathrm{d}^{N-1}\mathbf{z}(t)}{\mathrm{d}t^{N-1}}\right) \end{bmatrix}$$

$$\overset{(2.4)}{=} \begin{bmatrix} \mathbf{z}^{(1)}(t) \\ \vdots \\ \mathbf{z}^{(N-1)}(t) \\ \mathbf{u}_\theta\left(t, \mathbf{z}^{(0)}(t), \ldots, \mathbf{z}^{(N-1)}(t)\right) \end{bmatrix} \overset{(2.6)}{=} \mathbf{f}_\theta\left(t, \mathbf{x}(t)\right).$$

□

Example 2.1 (continuing from p. 5)**.** According to Proposition 2.1, we can rewrite Eq. (2.3) as a first-order ODE in stacked vector form. Applying Definition 2.2, we obtain the state

$$\mathbf{x}(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \\ \frac{\mathrm{d}z_1(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}z_2(t)}{\mathrm{d}t} \end{bmatrix}$$

and the vector field

$$\mathbf{f}_\theta\left(t, \mathbf{x}(t)\right) = \begin{bmatrix} [\mathbf{x}(t)]_3 \\ [\mathbf{x}(t)]_4 \\ -2\frac{k}{m}[\mathbf{x}(t)]_1 + \frac{k}{m}[\mathbf{x}(t)]_2 \\ \frac{k}{m}[\mathbf{x}(t)]_1 - 2\frac{k}{m}[\mathbf{x}(t)]_2 \end{bmatrix}. \tag{2.8}$$

in stacked vector form. Note how only the last two elements of the right-hand side vector in Eq. (2.8) convey new information, while the rest is just copied from the state $\mathbf{x}(t)$ that is argument to $\mathbf{f}_\theta$. This is characteristic of how we defined the vector field in stacked vector form in Eq. (2.6).

From here onwards, the term ODE will be used as a synonym for first-order ODE. Consequently, unless otherwise indicated, we will always work with states $\mathbf{x}$ and vector fields $\mathbf{f}_\theta$ in stacked vector form. By the fundamental theorem of calculus, integration of an ODE over a time interval $[t_0, t_k] \subseteq \mathbb{T}$ yields the state $\mathbf{x}$ at time t_k. However, without specification of the state at the initial time t_0, there are infinite possible solutions for $\mathbf{x}(t_k)$ just by varying $\mathbf{x}(t_0)$.

Definition 2.3. An *initial value problem (IVP)* is a tuple $(\mathbf{f}_\theta, \mathbf{x}_0)$ that consists of the vector field $\mathbf{f}_\theta$ corresponding to an ODE, and an *initial condition*

$$\mathbf{x}_0 := \mathbf{x}(t_0).$$

Applying the fundamental theorem of calculus, an IVP is solved at time t_k by

$$\mathbf{x}(t_k) = \mathbf{x}_0 + \int_{t_0}^{t_k} \mathbf{f}_\theta\left(t, \mathbf{x}(t)\right) \mathrm{d}t. \tag{2.9}$$

In order to be the unique solution, the *Picard-Lindelöf theorem* states that the recurrence

$$\begin{aligned} \boldsymbol{\phi}_0(t_k) &:= \mathbf{x}_0, \\ \boldsymbol{\phi}_{n+1}(t_k) &:= \mathbf{x}_0 + \int_{t_0}^{t_k} \mathbf{f}_\theta\left(t, \boldsymbol{\phi}_n(t)\right) \mathrm{d}t \end{aligned}$$

needs to converge to a fixed value – the solution $\mathbf{x}(t_k)$ – in the limit of $n \to \infty$. It can be shown that this is the case if the vector field is continuous in the time domain and Lipschitz-continuous in the state domain (Hairer et al., 1993). In the remainder of this work, we will only consider ODEs that fulfill these conditions.

Example 2.1 (continuing from p. 7)**.** We specify an exemplary IVP with the initial condition

$$\mathbf{x}_0 = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^\mathsf{T}.$$

It corresponds to the situation where both masses initially have zero velocity and are displaced by one unit of length in the direction of the right wall, as shown in Fig. 2.1.

Alternative to an IVP is to consider the ODE on the closed time interval $[t_0, t_n]$ and specify constraints on both extremes of the restricted time domain. This gives rise to a boundary value problem (BVP). However, since solving BVPs is different from solving IVPs and generally more complex, it will not be discussed in this work.

2.2 | Numerical ODE Solvers

The integral in Eq. (2.9) has closed-form solutions only for a few, rather simple cases.[2] In practice, one often has to rely on numerical methods instead. These operate by discretizing the problem along the time axis and iteratively extrapolating from the initial state. As a consequence, they always introduce an approximation error to the true solution (Section 2.2.3). By choosing a finer discretization, the error can be reduced, posing a trade-off between computational cost and accuracy of the simulation. In the following, we will first establish general definitions and subsequently discuss some numerical algorithms for ODE solving.

Definition 2.4. Let $[t_0, t_K] \subseteq \mathbb{T}$ be a time interval and let $h \in \mathbb{R}^+$ denote a *step size*, where

$$K := \left\lfloor \frac{t_K - t_0}{h} \right\rfloor$$

denotes the number of steps. Then, the *equidistant time-discretization* of the above interval by h is defined as

$$\widehat{\mathbb{T}} := \{t_0, t_0 + h, \ldots, t_0 + Kh\}. \tag{2.10}$$

[2] The presented Example 2.1 is one of these cases for which a closed-form solution exists. For the sake of brevity, we refrain from discussing it here and refer to Fitzpatrick (2018) for a derivation.

In the context of the time discretization given by Eq. (2.10), we also introduce shorthand notations

$$t_k \coloneqq t_0 + kh,$$
$$\mathbf{x}_k \coloneqq \mathbf{x}(t_k)$$

for $k \in \{0, 1, \ldots, K\}$.

Definition 2.5. Let $\mathcal{F} \coloneqq \{\mathbb{T} \times \mathbb{R}^{ND} \to \mathbb{R}^{ND}\}$ denote the function space of vector fields $\mathbf{f}_\theta$. Moreover, according to Definition 2.4, we assume an equidistant time discretization $\widehat{\mathbb{T}}$ with step size $h \in \mathbb{R}^+$. A *numerical ODE solver* is a function $\Xi : \mathcal{F} \times \mathbb{T} \times \mathbb{R}^{ND} \times \mathbb{R}^+ \to \mathbb{R}^{ND}$ that takes a vector field $\mathbf{f}_\theta \in \mathcal{F}$, the current time $t_k \in \widehat{\mathbb{T}}$, the corresponding state $\mathbf{x}_k \in \mathbb{R}^{ND}$ as well as the step size $h \in \mathbb{R}^+$ and yields the approximate state $\hat{\mathbf{x}}_{k+1} \approx \mathbf{x}_{k+1}$ at the next point in time t_{k+1},

$$\hat{\mathbf{x}}_{k+1} \coloneqq \Xi\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right). \tag{2.11}$$

Remark 2.1. In the process of computing a state's trajectory numerically, the step size h does not necessarily have to be constant, which then leads to an inequidistant time discretization $\widehat{\mathbb{T}}$. In fact, there are algorithms that adaptively change the step size based on local error estimates (Section 2.2.4, see Hairer et al. (1993) for an extensive overview). However, using an equidistant discretization simplifies subsequent parts of this work substantially, which is why we restrict ourselves to them for now. We will return to adaptive step sizes in Chapter 5.

2.2.1 | Euler's Method

The most basic representative of a numerical ODE solver is called the *explicit Euler method* (Euler, 1768). It simplifies the integration problem by making the assumption of stepwise constant dynamics, allowing for steps taken by single evaluations of the vector field $\mathbf{f}_\theta$.

Assumption 2.1. Given a vector field $\mathbf{f}_\theta$ as well as two consecutive points in time $t_k, t_{k+1} \in \widehat{\mathbb{T}}$, assume approximately constant dynamics within the step interval $[t_k, t_{k+1}]$ equal to the dynamics at t_k, i.e.,

$$\mathbf{f}_\theta\left(t, \mathbf{x}(t)\right) \approx \mathbf{f}_\theta\left(t_k, \mathbf{x}_k\right)$$

for $t \in [t_k, t_{k+1}]$.

Definition 2.6. The *explicit Euler method* is defined as

$$\Xi\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right) \coloneqq \mathbf{x}_k + h\mathbf{f}_\theta\left(t_k, \mathbf{x}_k\right). \tag{2.12}$$

Intuitively, Assumption 2.1 is only valid in combination with a rather small step size h, depending on how much the dynamics change over time. For stiff ODEs, i.e., equations that exhibit rapid changes over several orders of magnitude, it becomes highly inaccurate, if the step size is not extremely small. This inaccuracy can cause severely unstable behavior of the explicit Euler method, e.g., erroneously oscillating or exploding trajectories. In such cases, the following adaptation of Assumption 2.1 is more suitable, giving rise to the *implicit Euler method*.

Assumption 2.2. Given a vector field $\mathbf{f}_\theta$ as well as two consecutive points in time $t_k, t_{k+1} \in \widehat{\mathbb{T}}$, assume approximately constant dynamics within the step interval $[t_k, t_{k+1}]$ equal to the dynamics at t_{k+1}, i.e.,

$$\mathbf{f}_\theta\left(t, \mathbf{x}(t)\right) \approx \mathbf{f}_\theta\left(t_{k+1}, \mathbf{x}_{k+1}\right)$$

for $t \in [t_k, t_{k+1}]$.

Definition 2.7. The *implicit Euler method* is defined as

$$\Xi\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right) \coloneqq \mathbf{x}_k + h\mathbf{f}_\theta\left(t_k + h, \mathbf{x}(t_k + h)\right). \tag{2.13}$$

Its advantage in terms of stability comes at a price: The state $\mathbf{x}(t_k + h)$ on the right-hand side of Eq. (2.13) is unknown at compute time. Since the left-hand side is an estimate $\widehat{\mathbf{x}}(t_k + h)$ for $\mathbf{x}(t_k + h)$ (cf. Eq. (2.11)), this yields the root-finding problem

$$\mathbf{x}(t_k + h) \approx \mathbf{x}_k + h\mathbf{f}_\theta\left(t_k + h, \mathbf{x}(t_k + h)\right)$$

w.r.t. $\mathbf{x}(t_k + h)$, which generally has no closed-form solution. There are several algorithms to solve it numerically, e.g., *Newton's method* (Kelley, 2003), but they inevitably increase the computational cost and introduce an additional approximation error in every step of the implicit Euler method.

2.2.2 | Runge-Kutta Methods

A broader, more sophisticated class of solvers is called *Runge-Kutta methods* after works of Runge (1895) and Kutta (1901). The basic idea is to evaluate the vector field $\mathbf{f}_\theta$ not only once per step at a boundary of the step interval $[t_k, t_{k+1}]$, but also additional times in between.[3] This sequence of evaluations is aggregated to a weighted sum, which is then used as a better, more informed approximation for the assumption of constant dynamics within the step interval.

[3] Evaluations are usually and most intuitively at times within the interval $[t_k, t_{k+1}]$. However, this is not a necessary condition. See, e.g., Kraaijevanger and Spijker (1989) for a contrary example.

Assumption 2.3. Let $\{\tilde{t}_1, \ldots, \tilde{t}_s\} \subset \mathbb{T}$ and $\mathbf{b} \in \mathbb{R}^s$ denote a set of times and a vector of weights, respectively. Given a vector field $\mathbf{f}_\theta$ as well as two consecutive points in time $t_k, t_{k+1} \in \widehat{\mathbb{T}}$, assume approximately constant dynamics within the step interval $[t_k, t_{k+1}]$ equal to the $\mathbf{b}$-weighted sum of dynamics at $\tilde{t}_1, \ldots, \tilde{t}_s$, i.e.,

$$\mathbf{f}_\theta\left(t, \mathbf{x}(t)\right) \approx \sum_{i=1}^{s} [\mathbf{b}]_i \mathbf{f}_\theta\left(\tilde{t}_i, \mathbf{x}(\tilde{t}_i)\right)$$

for $t \in [t_k, t_{k+1}]$.

The states $\mathbf{x}(\tilde{t}_1), \ldots, \mathbf{x}(\tilde{t}_s)$ are usually unknown at the beginning of a step. So-called explicit Runge-Kutta (RK) methods iteratively estimate them by evaluating the vector field at weighted sums of already estimated states $\mathbf{x}(\tilde{t}_i)$, with the first state $\mathbf{x}(\tilde{t}_1)$ being estimated by the explicit Euler method (Definition 2.6). In contrast, implicit RK methods generally need to solve a system of s non-linear equations, e.g., again by applying Newton's method (Kelley, 2003).

Definition 2.8. A *Runge-Kutta (RK) method* is defined as

$$\begin{aligned} \Xi\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right) &\coloneqq \mathbf{x}_k + h \sum_{i=1}^{s} [\mathbf{b}]_i \mathbf{k}_i \\ \text{with} \quad \mathbf{k}_i &\coloneqq \mathbf{f}_\theta\left(t_k + h[\mathbf{c}]_i, \mathbf{x}_k + h \sum_{j=1}^{s} [\mathbf{A}]_{ij} \mathbf{k}_j\right), \end{aligned} \tag{2.14}$$

where $s \in \mathbb{N}$ denotes the *stage* of the method, and $\mathbf{A} \in \mathbb{R}^{s \times s}$, $\mathbf{b} \in \mathbb{R}^s$, and $\mathbf{c} \in \mathbb{R}^s$ are called the *RK matrix*, *weights*, and *nodes*, respectively. If $[\mathbf{A}]_{ij} = 0$ for $i \leq j$, the method is called *explicit*, otherwise *implicit*. An RK method is compactly notated by means of a so-called *Butcher tableau*

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^\intercal \end{array}.$$

For any RK method, the stage s represents the number of vector field evaluations, the RK matrix $\mathbf{A}$ as well as the nodes $\mathbf{c}$ determine where it is evaluated, and the weights $\mathbf{b}$ are used for the final aggregation of evaluations. Specific choices of these parameters give rise to concrete RK methods within the general class defined by Eq. (2.14). For a certain stage s, the parameters $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{c}$ are typically set such that the local error is minimized, upper bounded by a power of the step size h (cf. Section 2.2.3).

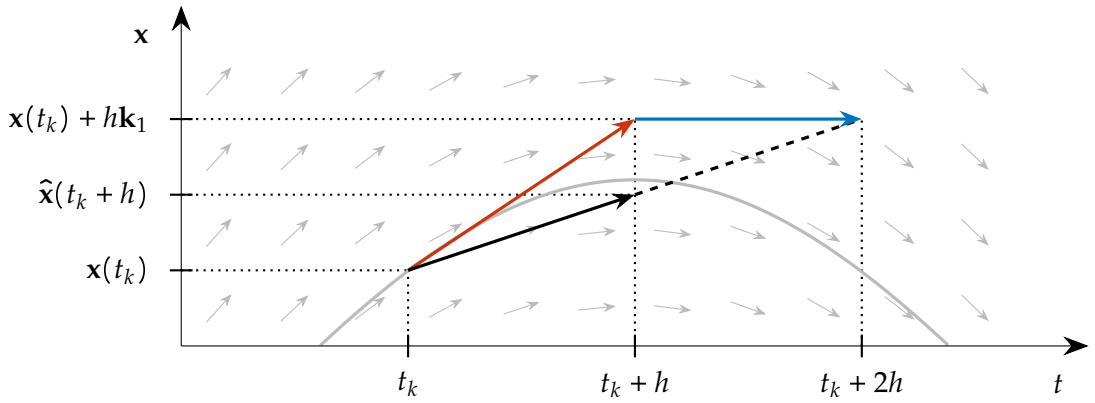


Figure 2.2: Heun's method. Background: True solution (—) and slopes produced by one-dimensional vector field (→). Foreground: Integration step from $(t_k, \mathbf{x}(t_k))$ to $(t_k + h, \widehat{\mathbf{x}}(t_k + h))$. First, evaluation of the vector field at $(t_k, \mathbf{x}(t_k))$, yielding the slope $\mathbf{k}_1$ (→). Next, explicit Euler step from $(t_k, \mathbf{x}(t_k))$ to $(t_k + h, \mathbf{x}(t_k) + h\mathbf{k}_1)$ and evaluation of the vector field at $(t_k + h, \mathbf{x}(t_k) + h\mathbf{k}_1)$, providing the slope $\mathbf{k}_2$ (→). Last, integration step from $(t_k, \mathbf{x}(t_k))$ to $(t_k + h, \widehat{\mathbf{x}}(t_k + h))$ using the average of slopes $\mathbf{k}_1$ and $\mathbf{k}_2$ (→).

Example 2.2. Both the explicit and implicit Euler method presented in Section 2.2.1 are stage one RK methods with Butcher tableaux

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \text{and} \qquad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array},$$

respectively. To see this, insert parameters into Eq. (2.14) according to Definition 2.8, and compare with Eqs. (2.12) and (2.13).

Example 2.3. *Heun's method* is an explicit stage two RK method defined as

$$\Xi(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h) := \mathbf{x}_k + h\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2\right) \tag{2.15a}$$

$$\text{with} \quad \mathbf{k}_1 := \mathbf{f}_\theta(t_k, \mathbf{x}_k), \tag{2.15b}$$

$$\mathbf{k}_2 := \mathbf{f}_\theta(t_k + h, \mathbf{x}_k + h\mathbf{k}_1), \tag{2.15c}$$

or, equivalently, by the Butcher tableau

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array}.$$

Intuitively, for a one-dimensional vector field $\mathbf{f}_\theta$, the method computes the slope of the trajectory at points $(t_k, \mathbf{x}(t_k))$ and $(t_k + h, \mathbf{x}(t_k + h))$. Because the state $\mathbf{x}(t_k + h)$ is not

known, it is estimated using a step of the explicit Euler method (cf. Definition 2.6). Then, the equally-weighted average of both slopes is used to perform the actual integration step. The whole procedure is also visualized in Fig. 2.2.

2.2.3 | Local and Global Error

Due to time discretization (Definition 2.4) and the assumption of stepwise constant dynamics (Assumption 2.3), solving ODEs numerically inevitably leads to an approximation error compared to the true solution. We distinguish between local and global errors.

Definition 2.9. Let $\mathbf{x}_k$, $\mathbf{x}_{k+1}$, $\mathbf{x}_{k+n}$ be true states at times t_k, t_{k+1}, $t_{k+n} \in \widehat{\mathbb{T}}$ with $n \in \mathbb{N}$. The *local error* $\boldsymbol{\delta}\mathbf{x}(t_{k+1})$ induced by a numerical ODE solver Ξ when extrapolating from t_k to t_{k+1} is defined as

$$\begin{aligned} \boldsymbol{\delta}\mathbf{x}(t_{k+1}) &:= |\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}| \\ \text{with} \quad \hat{\mathbf{x}}_{k+1} &:= \Xi\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right). \end{aligned} \tag{2.16}$$

In addition, the *global error* $\boldsymbol{\Delta}\mathbf{x}(t_{k+n})$ induced when extrapolating from t_k to t_{k+n} is defined as

$$\begin{aligned} \boldsymbol{\Delta}\mathbf{x}(t_{k+n}) &:= |\mathbf{x}_{k+n} - \hat{\mathbf{x}}_{k+n}| \\ \text{with} \quad \hat{\mathbf{x}}_{k+i} &:= \begin{cases} \Xi\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right) & i = 1, \\ \Xi\left(\mathbf{f}_\theta, t_{k+i-1}, \hat{\mathbf{x}}_{k+i-1}, h\right) & 1 < i \leq n. \end{cases} \end{aligned} \tag{2.17}$$

As before, we also introduce the shorthand notations

$$\begin{aligned} \boldsymbol{\delta}\mathbf{x}_{k+1} &:= \boldsymbol{\delta}\mathbf{x}(t_{k+1}), \\ \boldsymbol{\Delta}\mathbf{x}_{k+1} &:= \boldsymbol{\Delta}\mathbf{x}(t_{k+1}). \end{aligned}$$

Definition 2.8 (continuing from p. 11)**.** An RK method is called *of order* p if the local error induced by it is in $\mathcal{O}(h^{p+1})$, i.e.,

$$\forall\, t_k \in \widehat{\mathbb{T}} : \exists C \in \mathbb{R}^+ : \boldsymbol{\delta}\mathbf{x}_{t_{k+1}} \leq C h^{p+1}.$$

Moreover, it can be shown that the global error induced by a p-th-order RK method is in $\mathcal{O}(h^p)$ (Hairer et al., 1993), i.e.,

$$\forall\, t_k \in \widehat{\mathbb{T}} : \forall\, n \in \{1, \ldots, K-k\} : \exists C \in \mathbb{R}^+ : \boldsymbol{\Delta}\mathbf{x}_{t_{k+n}} \leq C h^p.$$

To check if a given RK method is of order p or to derive a new RK method of order p, a straight-forward approach is to construct Taylor series expansions of both the true solution $\mathbf{x}(t_k + h)$ and the approximation $\widehat{\mathbf{x}}(t_k + h)$ about the point t_k (or equivalently, for $h = 0$). Next, one compares arising terms of equal order in h and deduces conditions on the RK method's parameters such that terms up to order p cancel out on subtraction of the two Taylor series.

Example 2.3 (continuing from p. 13)**.** As an example, we will show that Heun's method is a second-order RK method. First, we consider Eq. (2.15c) as a function $\boldsymbol{\phi}$ of h, i.e.,

$$\boldsymbol{\phi}(h) \coloneqq \mathbf{k}_2 \overset{(2.15c)}{=} \mathbf{f}_\theta\left(t_k + h, \mathbf{x}_k + h\mathbf{k}_1\right).$$

Its Taylor series expansion about $h = 0$ up to first order is given by

$$\begin{aligned}\boldsymbol{\phi}(h) &= \boldsymbol{\phi}(0) + h\frac{\mathrm{d}\boldsymbol{\phi}(0)}{\mathrm{d}h} + \mathcal{O}(h^2)\\ &= \mathbf{f}_\theta\left(t_k, \mathbf{x}_k\right) + h\frac{\partial \mathbf{f}_\theta(t_k, \mathbf{x}_k)}{\partial t} + h\mathbf{k}_1\frac{\partial \mathbf{f}_\theta(t_k, \mathbf{x}_k)}{\partial \mathbf{x}} + \mathcal{O}(h^2).\end{aligned} \tag{2.18}$$

Inserting Eqs. (2.15b) and (2.18) into Eq. (2.15a) and then simplifying yields

$$\widehat{\mathbf{x}}(t_k + h) = \mathbf{x}_k + h\mathbf{f}_\theta\left(t_k, \mathbf{x}_k\right) + \frac{h^2}{2}\left(\frac{\partial \mathbf{f}_\theta(t_k, \mathbf{x}_k)}{\partial t} + \mathbf{f}_\theta\left(t_k, \mathbf{x}_k\right)\frac{\partial \mathbf{f}_\theta(t_k, \mathbf{x}_k)}{\partial \mathbf{x}}\right) + \mathcal{O}(h^3). \tag{2.19}$$

Second, the Taylor series expansion of $\mathbf{x}(t_k + h)$ at the point $t = t_k$ up to second order evaluates to

$$\begin{aligned}\mathbf{x}(t_k + h) &= \mathbf{x}_k + h\frac{\mathrm{d}\mathbf{x}(t_k)}{\mathrm{d}t} + \frac{h^2}{2}\frac{\mathrm{d}^2\mathbf{x}(t_k)}{\mathrm{d}t^2} + \mathcal{O}(h^3)\\ &= \mathbf{x}_k + h\mathbf{f}_\theta\left(t_k, \mathbf{x}_k\right) + \frac{h^2}{2}\left(\frac{\partial \mathbf{f}_\theta(t_k, \mathbf{x}_k)}{\partial t} + \mathbf{f}_\theta\left(t_k, \mathbf{x}_k\right)\frac{\partial \mathbf{f}_\theta(t_k, \mathbf{x}_k)}{\partial \mathbf{x}}\right) + \mathcal{O}(h^3).\end{aligned} \tag{2.20}$$

Finally, by comparing Eqs. (2.19) and (2.20), we see that they are identical and all terms up to order two cancel out on subtraction, confirming that Heun's method is of order two.

Unfortunately, the above derivation quickly becomes cumbersome for higher-order conditions. Using a graph theoretical description of the problem, it is possible to analyze its structural properties in a more organized way. Ultimately, this provides a general formalism to deduce the set of order conditions for any p. While its detailed introduction would exceed the scope of this work, one particular result is that

$$\mathbf{b}^\top\mathbf{A}^{p-1}\mathbf{1} = \frac{1}{p!} \tag{2.21}$$

is an omnipresent constraint for RK methods of all orders p (Zhang, 2019). It gives rise to a lower bound of an explicit RK method's number of stages s, if the method should be of order p.

Proposition 2.2 (Zhang (2019), Theorem 1.26)**.** *For all $p \in \mathbb{N}$, any explicit RK method of order p must have at least $s \geq p$ stages.*

Proof. Proof by contradiction: For all $p \in \mathbb{N}$, suppose there is an explicit RK method of order p with $s < p$ stages. By definition of being an explicit method, its RK matrix $\mathbf{A} \in \mathbb{R}^{s \times s}$ is lower-triangular. As a consequence, it is nilpotent of index s, and any power $\mathbf{A}^q$ with $q \geq s$ is the zero matrix. Last, the initial assumption $s < p$ is equivalent to $p - 1 \geq s$, thus the power $\mathbf{A}^{p-1}$ is the zero matrix and Eq. (2.21) is contradicted:

$$\forall \mathbf{b} \in \mathbb{R}^s : \mathbf{b}^\intercal \mathbf{A}^{p-1} \mathbf{1} \overset{(2.21)}{=} \frac{1}{p!} \overset{\lightning}{=} 0.$$

□

While Proposition 2.2 reveals that any explicit RK method must have at least $s \geq p$ stages if it is of order p, the converse (*"If an explicit RK method has $s \geq p$ stages, it is of order p."*) is only true up to order $p = 4$ (Hairer et al., 1993). This is also known as the *Butcher barrier* and boils down to the fact that the number of order conditions grows asymptotically exponentially in p (Zhang, 2019), whereas the number of degrees of freedom by the RK method only grows polynomially in s. Satisfaction of all constraints then typically requires more degrees of freedom than given by an RK method with $s = p$ stages.

2.2.4 | Embedded Runge-Kutta Methods

Knowledge of the approximation error is essential to decide where the numerical solution of an ODE can be trusted and where more accuracy (usually in the form of step size reduction) is necessary. Without access to the true solution, however, it is generally not possible to determine the exact local or global error, as expressed by Eqs. (2.16) and (2.17), respectively.

Embedded RK methods provide an elegant way to estimate the local error with only little additional computational effort. In essence, they combine two RK methods with different orders in one by letting the higher-order method share parts of its RK matrix $\mathbf{A}$ and nodes $\mathbf{c}$ with the lower-order method. They differ in the weights $\mathbf{b}$, necessitating only the additional computation of a weighted sum for the lower-order approximation when compared to the higher-order RK method alone.

Definition 2.10. A *numerical ODE solver with local error estimator* is a function $\Xi_\delta : \mathcal{F} \times \mathbb{T} \times \mathbb{R}^{ND} \times \mathbb{R}^+ \to \mathbb{R}^{2\times ND}$ that takes a vector field $\mathbf{f}_\theta \in \mathcal{F}$, the current time $t_k \in \widehat{\mathbb{T}}$, the corresponding state $\mathbf{x}_k \in \mathbb{R}^{ND}$ as well as the step size $h \in \mathbb{R}^+$ and yields the approximate state $\widehat{\mathbf{x}}_{k+1} \approx \mathbf{x}_{k+1}$ at the next point in time t_{k+1} as well as an estimate $\widehat{\delta\mathbf{x}}_{k+1} \approx \delta\mathbf{x}_{k+1}$ for the local error that arises when extrapolating from t_k to t_{k+1},

$$\begin{bmatrix} \widehat{\mathbf{x}}_{k+1} & \widehat{\delta\mathbf{x}}_{k+1} \end{bmatrix}^\intercal := \Xi_\delta\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right).$$

Definition 2.11. An *embedded RK method of order $p_1(p_2)$* is a numerical ODE solver with local error estimator Ξ_δ according to Definition 2.10, where

$$\widehat{\mathbf{x}}_{k+1} := \mathbf{x}_k + h\sum_{i=1}^{s}\left[\mathbf{b}^{(1)}\right]_i \mathbf{k}_i, \tag{2.22a}$$

$$\widehat{\delta\mathbf{x}}_{k+1} := \left| h\sum_{i=1}^{s}\left[\mathbf{b}^{(1)}\right]_i \mathbf{k}_i - h\sum_{i=1}^{s}\left[\mathbf{b}^{(2)}\right]_i \mathbf{k}_i \right| \tag{2.22b}$$

$$\text{with} \quad \mathbf{k}_i := \mathbf{f}_\theta\left(t_k + h[\mathbf{c}]_i, \mathbf{x}_k + h\sum_{j=1}^{s}[\mathbf{A}]_{ij}\mathbf{k}_j\right). \tag{2.22c}$$

As before, $s \in \mathbb{N}$ denotes the stage of the method in Eqs. (2.22a) to (2.22c). Moreover, $\mathbf{b}^{(1)}, \mathbf{b}^{(2)} \in \mathbb{R}^s$ are the weights that together with the RK matrix $\mathbf{A} \in \mathbb{R}^{s\times s}$ and the nodes $\mathbf{c} \in \mathbb{R}^s$ parametrize individual RK methods of order p_1 and p_2, respectively. An embedded RK method is summarized by a Butcher tableau of the form

$\mathbf{c}$	$\mathbf{A}$
	$\left(\mathbf{b}^{(1)}\right)^\intercal$
	$\left(\mathbf{b}^{(2)}\right)^\intercal$

In order to keep the additional computational effort as well as the accuracy of the numerical solution and the error estimate in reasonable balance, orders p_1 and p_2 usually differ by one. If $p_1 < p_2$, $\widehat{\delta\mathbf{x}}$ can be considered asymptotically as an error estimator for $\delta\mathbf{x}$ (Hairer et al., 1993). Otherwise, if $p_1 > p_2$, $\widehat{\delta\mathbf{x}}$ tends to overestimate the true local error, but using the higher-order approximation as solution is often beneficial in terms of accuracy and stability (Shampine & Watts, 1976). This mode is called *local extrapolation* and most modern embedded RK methods are developed in its favor, i.e., their parameters are chosen to minimize the error of the higher-order solution (Bogacki & Shampine, 1989; Dormand & Prince, 1980; Prince & Dormand, 1981).

Example 2.3 (continuing from p. 14)**.** Heun's method already exploits the explicit Euler method to estimate the first intermediate state (cf. Eqs. (2.15b) and (2.15c)). By Proposition 2.2, the explicit Euler method is of order one. Consequently, Heun's method can readily be transformed into an embedded RK method with the explicit Euler method as lower-order method. If local extrapolation is performed, the resulting embedded RK method, subsequently referred to as *Heun-Euler method*, is of order $2(1)$ and corresponds to the Butcher tableau

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \\ & 1 & 0 \end{array}.$$

2.3 | Bayesian Inference in State-Space Models

In probability theory, *Bayesian inference* describes the process of updating the probability distribution $p(\mathbf{x})$ of a random variable $\mathbf{x}$ using new information $\mathbf{y}$. At the center of this framework is *Bayes' theorem*, which states that the *posterior* distribution on $\mathbf{x}$ after conditioning on $\mathbf{y}$ is given by

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} = \frac{p(\mathbf{x}, \mathbf{y})}{\int p(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x}} = \frac{p(\mathbf{y} \mid \mathbf{x}) \, p(\mathbf{x})}{\int p(\mathbf{y} \mid \mathbf{x}) \, p(\mathbf{x}) \, \mathrm{d}\mathbf{x}}, \tag{2.23}$$

where $p(\mathbf{y} \mid \mathbf{x})$, $p(\mathbf{x})$ and $p(\mathbf{y})$ are called *likelihood*, *prior*, and *evidence*, respectively. Unfortunately, (naive) Bayesian inference quickly becomes intractable for higher-dimensional problems due to its exponential complexity. Partial remedy for the curse of dimensionality lies in the exploitation of conditional independence between random variables.

Example 2.4. Consider the case of component-wise binary random variables $\mathbf{x} \in \{0,1\}^M$ and $\mathbf{y} \in \{0,1\}^L$. Their joint probability distribution $p(\mathbf{x}, \mathbf{y})$ has $2^{M+L} - 1$ parameters and the integral in the denominator of Eq. (2.23) turns into a sum over 2^M terms, being infeasible to store and compute, respectively, for large M and L.

For the sake of clarity, we suppose $M = L = 2$ and define $x_i := [\mathbf{x}]_i$ and $y_j := [\mathbf{y}]_j$ for $i, j \in \{1, 2\}$. By the chain rule, every n-variate joint probability distribution can be factorized into a product of n univariate conditional and marginal distributions. Thus, we can rewrite the joint probability distribution of $\mathbf{x}$ and $\mathbf{y}$ as, e.g.,

$$p(\mathbf{x}, \mathbf{y}) = p(y_1 \mid y_2, x_1, x_2) \, p(y_2 \mid x_1, x_2) \, p(x_1 \mid x_2) \, p(x_2) \, . \tag{2.24}$$

Under the exemplary assumption that the conditional independence statements

$$y_1 \perp\!\!\!\perp x_2, y_2 \mid x_1,$$
$$y_2 \perp\!\!\!\perp x_1, y_1 \mid x_2,$$

hold, Eq. (2.24) can be simplified to

$$p(\mathbf{x}, \mathbf{y}) = p(y_1 \mid x_1)\, p(y_2 \mid x_2)\, p(x_1 \mid x_2)\, p(x_2)\,. \tag{2.25}$$

When comparing Eqs. (2.24) and (2.25), we see that the second, simplified representation only requires 7 parameters instead of the 15 parameters needed by the first one.

2.3.1 | State-Space Models

In the following, we will focus on a special class of probabilistic models called *state-space models*. Ultimately, they will form the framework for constructing a probabilistic model around generic ODE solvers. State-space models are composed of *stochastic processes*.

Definition 2.12. A *discrete-time stochastic process* $\mathcal{X}$ is defined as a collection of random variables $\{\mathbf{x}_k\}_{k=0}^{K} \subset \mathbb{R}^M$, indexed by non-negative integers that refer to discrete points in time $\{t_k\}_{k=0}^{K}$.

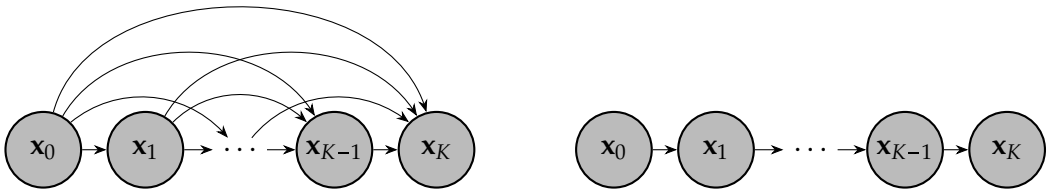
In accordance with Sections 2.1 and 2.2, we also refer to a random variable $\mathbf{x}_k$ in the context of a stochastic process as *state* and use notations $\mathbf{x}_k = \mathbf{x}(t_k)$ interchangeably for a previously specified set of corresponding times $\widehat{\mathbb{T}} := \{t_k\}_{k=0}^{K}$. In addition, we abbreviate collections of states consecutive in time by $\mathbf{x}_{k:k+j} := \{\mathbf{x}_k, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+j}\}$ and define $\mathbf{x}_{-j:k} := \mathbf{x}_{0:k}$ for any $j \geq 0$, respectively.

Definition 2.13. A discrete-time stochastic process $\mathcal{X} := \{\mathbf{x}_k\}_{k=0}^{K}$ is called *n-th-order Markov*, if the *Markov condition*

$$p(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-n:k-1}) \tag{2.26}$$

holds, i.e., a state $\mathbf{x}_k$ is conditionally independent w.r.t. states $\mathbf{x}_{0:k-n-1}$ in the far past, given the states $\mathbf{x}_{k-n:k-1}$ in the immediate past.

Remark 2.2. In an analogous way to how any n-th-order ODE can be transformed into a first-order ODE (cf. Proposition 2.1), any n-th-order Markov process can be transformed into a first-order Markov process by stacking n states consecutive in time into a single state. Consequently, we will use the term Markov as a synonym for first-order Markov below.



(a) Non-Markov process. **(b)** Markov process.

Figure 2.3: Dynamic Bayesian networks of a stochastic process $\{\mathbf{x}_k\}_{k=0}^K$. Left: Without Markov assumption. Right: With Markov assumption.

For a Markov process, the joint probability distribution factorizes into

$$\begin{aligned} p(\mathbf{x}_{0:K}) &= p(\mathbf{x}_0) \prod_{k=1}^{K} p(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}) \\ &\overset{(2.26)}{=} p(\mathbf{x}_0) \prod_{k=1}^{K} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}). \end{aligned} \tag{2.27}$$

Using graphical representations, the independence structure of a probabilistic model can be visualized. Figures 2.3a and 2.3b show so-called *dynamic Bayesian networks* of a non-Markov and a Markov process, respectively, where arrows point from variables being conditioned on to variables that condition on these. In particular, they visualize how the factorization is simplified in Eq. (2.27) due to the Markov property.

Definition 2.14. A *probabilistic SSM*, also known as *hidden Markov model*, is a pair of two interdependent Markov processes $(\mathcal{X}, \mathcal{Y})$ with $\mathcal{X} = \{\mathbf{x}_k\}_{k=0}^K \subset \mathbb{R}^M$, $\mathcal{Y} = \{\mathbf{y}_k\}_{k=1}^K \subset \mathbb{R}^L$, where states $\mathbf{x}_k$ are unobserved. Consequently, we refer to states $\mathbf{y}_k$ as *observations*. Every state $\mathbf{x}_k$ is independent of past states $\mathbf{x}_{0:k-2}$ and past observations $\mathbf{y}_{1:k-1}$, given its previous state $\mathbf{x}_{k-1}$. Moreover, every observation $\mathbf{y}_k$ is conditionally independent of past states $\mathbf{x}_{0:k-1}$ and past observations $\mathbf{y}_{1:k-1}$, yielding the Markov conditions

$$p(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \tag{2.28a}$$

$$p(\mathbf{y}_k \mid \mathbf{x}_{0:k}, \mathbf{y}_{1:k-1}) = p(\mathbf{y}_k \mid \mathbf{x}_k). \tag{2.28b}$$

The conditional distributions $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ and $p(\mathbf{y}_k \mid \mathbf{x}_k)$ are called *transition* and *observation models*, respectively.

Figure 2.4 shows a probabilistic SSM, represented by a dynamic Bayesian network. Next, we will discuss how to efficiently perform inference tasks in such models.

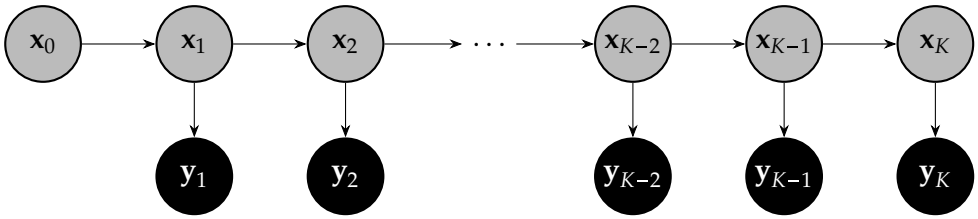


Figure 2.4: Dynamic Bayesian network of a probabilistic SSM: Unobserved states $\mathbf{x}_{0:K}$ (○), observations $\mathbf{y}_{1:K}$ (●).

2.3.2 | Bayesian Filtering

Depending on the quantity being inferred, Bayesian inference tasks have different names in the context of SSMs:

1. *Prediction*: $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$,
2. *Filtering*: $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$,
3. *Smoothing*: $p(\mathbf{x}_k \mid \mathbf{y}_{1:K})$.

In the remainder of this work, we will only consider prediction and filtering tasks. As a consequence of the Markov dependence structure, there is a linear-in-time recursive algorithm that solves these two tasks at once, called *Bayesian filtering*. Every recursive step in the algorithm consists of two substeps, referred to as *prediction* and *update step*.

In essence, the k-th prediction step estimates the distribution of the state $\mathbf{x}_k$ conditioned on previous observations $\mathbf{y}_{1:k-1}$, using the distribution of the previous state $\mathbf{x}_{k-1}$, itself conditioned on previous observations $\mathbf{y}_{1:k-1}$. It is computed by

$$\begin{aligned} p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}\right) &= \int p\left(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}\right) \mathrm{d}\mathbf{x}_{k-1} \\ &= \int p\left(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_{1:k-1}\right) p\left(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}\right) \mathrm{d}\mathbf{x}_{k-1} \\ &\overset{(2.28a)}{=} \int \underbrace{p\left(\mathbf{x}_k \mid \mathbf{x}_{k-1}\right)}_{\substack{\text{transition} \\ \text{model}}} \underbrace{p\left(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}\right)}_{\substack{\text{previous filtering} \\ \text{distribution}}} \mathrm{d}\mathbf{x}_{k-1}, \end{aligned} \tag{2.29}$$

also known as *Chapman-Kolmogorov equation* in the literature (Särkkä & Svensson, 2023). In the first prediction step, the previous filtering distribution is provided as initial state distribution $p(\mathbf{x}_0)$.

The update step then corrects that estimate using the corresponding observation $\mathbf{y}_k$, yielding the distribution of the state $\mathbf{x}_k$ conditioned on up-to-date observations $\mathbf{y}_{1:k}$. It is

derived as follows:

$$\begin{aligned} p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k}\right) &= p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \mathbf{y}_k\right) \\ &= \frac{p\left(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1}\right) p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}\right)}{\int p\left(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1}\right) p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}\right) \mathrm{d}\mathbf{x}_k} \\ &\overset{(2.28b)}{=} \frac{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}\right)}{\int \underbrace{p\left(\mathbf{y}_k \mid \mathbf{x}_k\right)}_{\substack{\text{observation} \\ \text{model}}} \underbrace{p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}\right)}_{\substack{\text{prediction} \\ \text{distribution}}} \mathrm{d}\mathbf{x}_k}. \end{aligned} \tag{2.30}$$

2.3.3 | The (Extended) Kalman Filter

Although Markovianity enables linear-in-time inference, Eqs. (2.29) and (2.30) might still be difficult to apply in practice, as they contain integrals of general probability distributions over potentially high-dimensional states. In most cases, there exist no closed-form solutions to these integrals, necessitating the usage of expensive Monte-Carlo based methods (sometimes called *particle filters* (Gordon et al., 1993)) with rather bad convergence rates (Crisan & Doucet, 2002). However, if one is able to make further assumptions on the used probability distributions as well as transition and observation models, computational complexity can be reduced drastically.

Definition 2.15. An SSM $(\mathcal{X}, \mathcal{Y})$ with $\mathcal{X} = \{\mathbf{x}_k\}_{k=0}^K \subset \mathbb{R}^N$, $\mathcal{Y} = \{\mathbf{y}_k\}_{k=1}^K \subset \mathbb{R}^L$ is called *Gaussian SSM*, if its transition and observation models are Gaussian, i.e.,

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}\left(\mathbf{x}_k; \mathbf{f}(\mathbf{x}_{k-1}), \mathbf{Q}_{k-1}\right), \tag{2.31a}$$

$$p(\mathbf{y}_k \mid \mathbf{x}_k) = \mathcal{N}\left(\mathbf{y}_k; \mathbf{h}(\mathbf{x}_k), \mathbf{R}_k\right) \tag{2.31b}$$

for a *transition function* $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$, a *observation function* $\mathbf{h} : \mathbb{R}^N \to \mathbb{R}^L$, a *process noise covariance matrix* $\mathbf{Q}_{k-1} \in \mathbb{R}^{N \times N}$ at time t_{k-1}, and a *observation noise covariance matrix* $\mathbf{R}_k \in \mathbb{R}^{L \times L}$ at time t_k.[4] If both the transition function $\mathbf{f}$ and the observation function $\mathbf{h}$ are also affine, i.e.,

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}\left(\mathbf{x}_k; \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{b}_{k-1}, \mathbf{Q}_{k-1}\right), \tag{2.32a}$$

$$p(\mathbf{y}_k \mid \mathbf{x}_k) = \mathcal{N}\left(\mathbf{y}_k; \mathbf{H}_k\mathbf{x}_k + \mathbf{c}_{k-1}, \mathbf{R}_k\right), \tag{2.32b}$$

the SSM is called *affine Gaussian SSM*. Here, $\mathbf{A}_{k-1} \in \mathbb{R}^{N \times N}$ and $\mathbf{b}_{k-1} \in \mathbb{R}^N$ are the *transition matrix* and *transition offset* at time t_{k-1}, whereas $\mathbf{H}_k \in \mathbb{R}^{L \times N}$ and $\mathbf{c}_k \in \mathbb{R}^L$ denote the *observation matrix* and *observation offset* at time t_k, respectively.

[4] In general, the transition function $\mathbf{f}$ may also depend on the time t_{k-1}, and the covariance matrix $\mathbf{Q}_{k-1}$ may also depend on the state $\mathbf{x}_{k-1}$, but we omit the explicit notation of these dependencies here, following Särkkä and Svensson (2023).

Assumption 2.4. Assume an affine Gaussian SSM according to Definition 2.15. In addition, assume Gaussian prediction, filtering, and initial state distributions, i.e.,

$$\begin{aligned} p(\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_0; \mathbf{m}_0, \mathbf{P}_0), \\ p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) &= \mathcal{N}\left(\mathbf{x}_k; \widehat{\mathbf{m}}_k, \widehat{\mathbf{P}}_k\right), \\ p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) &= \mathcal{N}(\mathbf{x}_k; \mathbf{m}_k, \mathbf{P}_k). \end{aligned}$$

With Assumption 2.4, there is a cubic-in-state Bayesian filtering algorithm called the *Kalman filter*. Its efficiency is gained by the facts that Gaussians are closed under affine transformations, multiplication, marginalization as well as conditioning, and that these operations effectively reduce to simple linear algebra operations in this case (Särkkä & Svensson, 2023).

While the favorable complexity properties make the Kalman filter appealing in theory, its restrictive assumptions on affinity and Gaussianity ultimately limit applicability in many practical situations. In particular, the affinity assumption expressed by Eqs. (2.32a) and (2.32b) poses a problem: Probability distributions can often be approximated reasonably well by Gaussians, but non-linear transition and/or observation functions are hard exclusion criteria for the Kalman filter.

Assumption 2.5. Assume a locally approximately affine Gaussian SSM, i.e.,

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) &\approx \mathcal{N}(\mathbf{x}_k; \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{b}_{k-1}, \mathbf{Q}_{k-1}), \\ p(\mathbf{y}_k \mid \mathbf{x}_k) &\approx \mathcal{N}\left(\mathbf{y}_k; \mathbf{H}_k\mathbf{x}_k + \mathbf{c}_{k-1}, \mathbf{R}_k\right). \end{aligned}$$

The *extended Kalman filter (EKF)* is an adaptation of the Kalman filter that tries to overcome the problem of non-linear functions $\mathbf{f}$ and $\mathbf{h}$ by linearizing them locally using a first-order Taylor expansion. Under Assumption 2.5, this is a reasonable approximation. However, since the linearization becomes increasingly inaccurate at greater distances from the point around which it is formed, it needs to be recomputed in every step. First-order Taylor expansions of $\mathbf{f}$ and $\mathbf{h}$ around $\mathbf{m}_{k-1}$ and $\widehat{\mathbf{m}}_k$ are

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &\approx \mathbf{f}(\mathbf{m}_{k-1}) + \mathbf{J}\mathbf{f}(\mathbf{m}_{k-1})(\mathbf{x} - \mathbf{m}_{k-1}) \\ &= \underbrace{\mathbf{J}\mathbf{f}(\mathbf{m}_{k-1})}_{\mathbf{A}_{k-1}}\mathbf{x} + \underbrace{\mathbf{f}(\mathbf{m}_{k-1}) - \mathbf{J}\mathbf{f}(\mathbf{m}_{k-1})\mathbf{m}_{k-1}}_{\mathbf{b}_{k-1}} \end{aligned} \tag{2.33}$$

and

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &\approx \mathbf{h}(\widehat{\mathbf{m}}_k) + \mathbf{J}\mathbf{h}(\widehat{\mathbf{m}}_k)(\mathbf{x} - \widehat{\mathbf{m}}_k) \\ &= \underbrace{\mathbf{J}\mathbf{h}(\widehat{\mathbf{m}}_k)}_{\mathbf{H}_k}\mathbf{x} + \underbrace{\mathbf{h}(\widehat{\mathbf{m}}_k) - \mathbf{J}\mathbf{h}(\widehat{\mathbf{m}}_k)\widehat{\mathbf{m}}_k}_{\mathbf{c}_k}, \end{aligned} \tag{2.34}$$

respectively. Being affine functions, Eqs. (2.33) and (2.34) then allow usage of the basic Kalman filter.

All computational steps of the EKF are summarized by Algorithm 1, where Eqs. (2.35) and (2.36) correspond to the prediction step and Eqs. (2.37) to (2.41) to the update step. A detailed derivation of these equations from Eqs. (2.29) and (2.30) is omitted at this point, the interested reader is referred to Särkkä and Svensson (2023).

Algorithm 1: Extended Kalman filter

Input:

$\mathbf{m}_0$, $\mathbf{P}_0$	Initial filtering mean and covariance matrix
$\mathbf{y}_{1:K}$	Observations
$\mathbf{f}$	Transition function
$\mathbf{h}$	Observation function
$\{\mathbf{Q}_k\}_{k=0}^{K-1}$	Process noise covariance matrices
$\{\mathbf{R}_k\}_{k=1}^{K}$	Observation noise covariance matrices

Output:

$\{\widehat{\mathbf{m}}_k\}_{k=1}^{K}$	Prediction means
$\{\hat{\mathbf{P}}_k\}_{k=1}^{K}$	Prediction covariance matrices
$\{\mathbf{m}_k\}_{k=1}^{K}$	Filtering means
$\{\mathbf{P}_k\}_{k=1}^{K}$	Filtering covariance matrices
$\{\hat{\mathbf{y}}_k\}_{k=1}^{K}$	Data likelihood means
$\{\mathbf{S}_k\}_{k=1}^{K}$	Data likelihood covariance matrices

for $k = 1, 2, \ldots, K$ **do**

Prediction step

$$\widehat{\mathbf{m}}_k = \mathbf{f}(\mathbf{m}_{k-1}) \tag{2.35}$$

$$\hat{\mathbf{P}}_k = \mathbf{J}\mathbf{f}(\mathbf{m}_{k-1})\mathbf{P}_{k-1}\mathbf{J}\mathbf{f}(\mathbf{m}_{k-1})^\mathsf{T} + \mathbf{Q}_{k-1} \tag{2.36}$$

Update step

$$\hat{\mathbf{y}}_k = \mathbf{h}(\widehat{\mathbf{m}}_k) \tag{2.37}$$

$$\mathbf{S}_k = \mathbf{J}\mathbf{h}(\widehat{\mathbf{m}}_k)\hat{\mathbf{P}}_k\mathbf{J}\mathbf{h}(\widehat{\mathbf{m}}_k)^\mathsf{T} + \mathbf{R}_k \tag{2.38}$$

$$\mathbf{K}_k = \hat{\mathbf{P}}_k\mathbf{J}\mathbf{h}(\widehat{\mathbf{m}}_k)^\mathsf{T}\mathbf{S}_k^{-1} \tag{2.39}$$

$$\mathbf{m}_k = \widehat{\mathbf{m}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right) \tag{2.40}$$

$$\mathbf{P}_k = \hat{\mathbf{P}}_k - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\mathsf{T} \tag{2.41}$$

end

Approach | 3

The following two sections describe the methods developed in this work. First, Section 3.1 deals with the definition of a probabilistic model that is based on black box ODE solvers and an estimator for their local error. Two ways to perform inference in this model are presented, one of which is sequential sampling and the other using the EKF. Furthermore, the requirements for the applicability of a solver, as well as additional assumptions are set out. In Section 3.2, process noise tempering is developed as a method to estimate ODE parameters more reliably than with the standard approach of non-linear least-squares-regression, which often converges only to local minima. Process noise tempering is based on another probabilistic model that indirectly accounts for the uncertainty about parameters during simulation. In the course of optimization, this uncertainty is gradually decreased to facilitate convergence to the global optimum.

3.1 | Black Box Probabilistic ODE Solvers

This work's approach of extending existing ODE solvers by probabilistic output is based on a work by Conrad et al. (2017). There, the authors perturbed predictions of a classic RK solver step-wise using zero-mean Gaussian noise. By sampling from the noise distribution in every step, they produced a collection of sample trajectories that represents a probabilistic ODE solution. However, in order to achieve a good calibration of the uncertainty in these samples, they had to determine the noise covariance specific for every new simulation setting, e.g., different initial values or parameter choices, by solving a non-linear optimization problem. Below, we will define a similar probabilistic model, but define the Gaussian noise using an estimator for the solver's local error, circumventing manual calibration.

3.1.1 | Probabilistic Model

Definition 3.1. Let $\mathbf{f}_\theta$ be a vector field, let $\widehat{\mathbb{T}}$ be an equidistant time discretization with corresponding step size h, and let Ξ be a numerical ODE solver. The probabilistic model

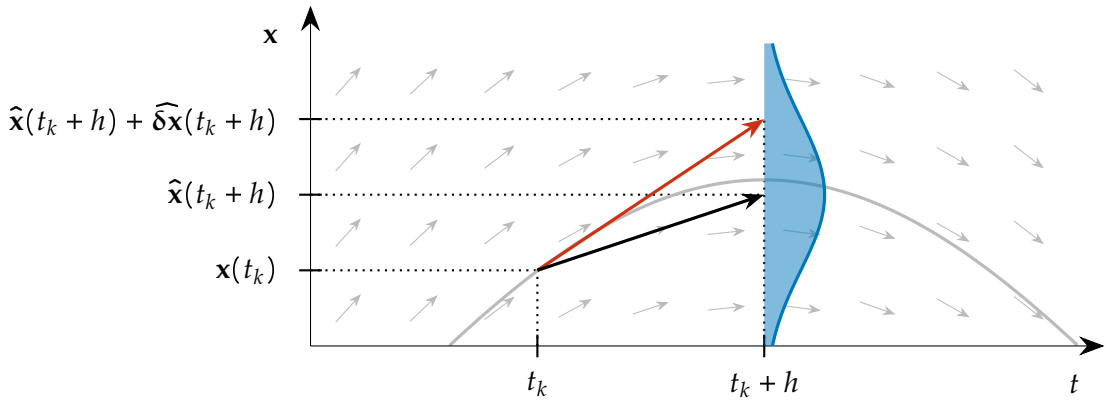


Figure 3.1: Probabilistic model $p_{\text{num.}}$. Background: True solution (—) and slopes produced by one-dimensional vector field (→). Foreground: Integration step from $(t_k, \mathbf{x}(t_k))$ to $(t_k + h, \hat{\mathbf{x}}(t_k + h))$ using Heun's method (→). An estimate for the local error $\widehat{\delta\mathbf{x}}(t_k + h)$ is provided by the explicit Euler method (→). The two predicted quantities $\hat{\mathbf{x}}(t_k + h)$ and $\widehat{\delta\mathbf{x}}(t_k + h)$ together define a Gaussian distribution on the state $\mathbf{x}(t_k + h)$ (—), according to Eq. (3.3).

by Conrad et al. (2017) over the state $\mathbf{x}_{k+1}$, given the previous state $\mathbf{x}_k$ is defined as

$$p_{\text{Conrad}}\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k\right) := \mathcal{N}\left(\mathbf{x}_{k+1}; \Xi\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right), \sigma^2 \mathbf{I}\right), \tag{3.1}$$

where $\sigma^2 \in \mathbb{R}_0^+$ denotes the constant scalar variance applied to all state components.

Definition 3.2. Let $\mathbf{f}_\theta$ be a vector field, let $\widehat{\mathbb{T}}$ be an equidistant time discretization with corresponding step size h, and let Ξ_δ be a numerical ODE solver with local error estimator. Moreover, according to Definition 2.10, let

$$\begin{bmatrix} \hat{\mathbf{x}}_{k+1} & \widehat{\delta\mathbf{x}}_{k+1} \end{bmatrix}^\intercal := \Xi_\delta\left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h\right) \tag{3.2}$$

denote the predicted state and the estimated local error at time $t_{k+1} \in \widehat{\mathbb{T}}$, given the previous time $t_k \in \widehat{\mathbb{T}}$ and corresponding state $\mathbf{x}_k \in \mathbb{R}^{ND}$. We define the following probabilistic model over the state $\mathbf{x}_{k+1}$, given the previous state $\mathbf{x}_k$:[1]

$$p_{\text{num.}}\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k\right) := \mathcal{N}\left(\mathbf{x}_{k+1}; \hat{\mathbf{x}}_{k+1}, \operatorname{diag}\left(\widehat{\delta\mathbf{x}}_{k+1}^2\right)\right). \tag{3.3}$$

Example 2.3 (continuing from p. 17)**.** We use the Heun-Euler method for Ξ_δ and assume a one-dimensional vector field $\mathbf{f}_\theta$ as shown in Fig. 2.2. For a given state $\mathbf{x}_k$, the probabilistic

[1] Note that both $\hat{\mathbf{x}}_{k+1}$ and $\widehat{\delta\mathbf{x}}_{k+1}$ depend on $\mathbf{x}_k$ via Eq. (3.2).

model defined by Eq. (3.3) is a Gaussian distribution centered in the approximate next state $\hat{\mathbf{x}}_{k+1}$ that is predicted by Heun's method. About 68 percent of its probability mass is contained in the interval $[\hat{\mathbf{x}}_{k+1} - \widehat{\delta\mathbf{x}}_{k+1}, \hat{\mathbf{x}}_{k+1} + \widehat{\delta\mathbf{x}}_{k+1}]$, where the local error estimate $\widehat{\delta\mathbf{x}}_{k+1}$ arises from the difference of predictions by Heun's method and the explicit Euler method. The described setting is also visualized in Fig. 3.1.

3.1.2 | Inference

The probabilistic model $p_{\text{num.}}$ introduced in Section 3.1.1 can be used to create probabilistic solutions over an ODE. Following Conrad et al. (2017), one way to perform inference in this model is to sample from it in every time step. Another is to linearize the numerical ODE solver and apply the EKF without observation model, i.e., only conducting the predict step. While we focus on these two methods in the present work, there are others reported for non-linear filtering problems in the literature, each of which has its own advantages and disadvantages. Examples are the unscented Kalman filter (Julier & Uhlmann, 1997) and Gaussian mixture filters (Alspach & Sorenson, 1972; Huber, 2011). In Chapter 4, we will experimentally evaluate how well probabilistic solutions created with this work's model are calibrated in comparison to a baseline provided by the model of Conrad et al. (2017).

Sequential Sampling Starting with the initial value $\mathbf{x}_0$, the first integration step is performed by the numerical ODE solver Ξ_δ, producing the predicted next state $\hat{\mathbf{x}}_1$ as well as the corresponding local error estimate $\widehat{\delta\mathbf{x}}_1$. Next, a fixed number of M samples $\tilde{\mathbf{x}}_1^{(1)}, \ldots, \tilde{\mathbf{x}}_1^{(M)}$ is drawn from the Gaussian distribution defined by Eq. (3.3). In the subsequent and all following integration steps, each of these samples represents a distinct starting point, yielding M different predicted states $\hat{\mathbf{x}}_k^{(1)}, \ldots, \hat{\mathbf{x}}_k^{(M)}$ and local error estimates $\widehat{\delta\mathbf{x}}_k^{(1)}, \ldots, \widehat{\delta\mathbf{x}}_k^{(M)}$. They give rise to a collection of M individual Gaussian distributions, each of which is sampled once, resulting in the sample states $\tilde{\mathbf{x}}_k^{(1)}, \ldots, \tilde{\mathbf{x}}_k^{(M)}$ for the next integration step. Ultimately, the obtained set of sample trajectories $\{\{\tilde{\mathbf{x}}_k^{(i)}\}_{k=1}^K\}_{i=1}^M$ represents an empirical approximation to the true stochastic process that satisfies Eq. (3.3). The whole procedure is also summarized by Algorithm 2.

Compared to the regular case of solving ODEs deterministically, sequential sampling increases the computational effort in terms of calls of the numerical ODE solver by a factor of M. However, as all samples at a time t_k are independent of each other, these additional calls can be effectively parallelized, if sufficient compute hardware is available.

Remark 3.1. Sequential sampling greatly benefits from the fact that the used time discretization is equidistant, as both sampling and calls of the numerical ODE solver can be

Algorithm 2: Sequential sampling

Input:

Ξ_δ	Numerical ODE solver with local error estimator
$\mathbf{f}_\theta$	Vector field
$\widehat{\mathbb{T}}$	Equidistant time discretization
h	Step size
$\mathbf{x}_0$	Initial value
M	Number of samples

Output:

$\{\{\tilde{\mathbf{x}}_k^{(i)}\}_{k=1}^K\}_{i=1}^M$	Sample trajectories

$\left[\hat{\mathbf{x}}_1 \quad \widehat{\delta\mathbf{x}}_1\right]^\intercal = \Xi_\delta\left(\mathbf{f}_\theta, t_0, \mathbf{x}_0, h\right)$
for $i = 1, 2, \ldots, M$ **do**
  $\tilde{\mathbf{x}}_1^{(i)} \sim \mathcal{N}\left(\mathbf{x}_1; \hat{\mathbf{x}}_1, \operatorname{diag}\left(\widehat{\delta\mathbf{x}}_1^2\right)\right)$
end
for $k = 2, 3, \ldots, K$ **do**
  for $i = 1, 2, \ldots, M$ **do**
    $\left[\hat{\mathbf{x}}_k \quad \widehat{\delta\mathbf{x}}_k\right]^\intercal = \Xi_\delta\left(\mathbf{f}_\theta, t_{k-1}, \tilde{\mathbf{x}}_{k-1}^{(i)}, h\right)$
    $\tilde{\mathbf{x}}_k^{(i)} \sim \mathcal{N}\left(\mathbf{x}_k; \hat{\mathbf{x}}_k^{(i)}, \operatorname{diag}\left(\left(\widehat{\delta\mathbf{x}}_k^{(i)}\right)^2\right)\right)$
  end
end

parallelized perfectly. Besides, it leads to collections of sample states $\{\tilde{\mathbf{x}}_k^{(i)}\}_{i=1}^M$ corresponding to the same point in time t_k, enabling post-hoc computation of statistics like empirical moments. With adaptive step size selection, on the other hand, sampled states are in general not time-coherent. This complicates the computation of meaningful statistics and is less efficient if individual trajectories take more integration steps than others.

Extended Kalman Filter Since a numerical ODE solver is in general a non-linear function w.r.t. its input state, the probabilistic model defined by Eq. (3.3) specifies a non-linear Gaussian transition model in the form of Eq. (2.31a). Hence, approximate Gaussian inference can be conducted using the predict step of the EKF (cf. Eqs. (2.35) and (2.36)). This results in approximations for the first two moments of the true underlying stochastic process.

Due to finite precision arithmetic, the EKF as introduced in Section 2.3.3 is prone to

numerical instabilities. In practical applications where the matrix-multiplied Jacobians or the added noise matrices may contain extremely small numbers, this can lead to round-off errors that accumulate considerably over long time horizons. By using an adapted version of the algorithm that only operates on the square roots of covariance matrices, this problem can be mitigated (Grewal & Andrews, 2014).

Proposition 3.1. *Let* $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n \in \mathbb{R}^{N \times N}$ *be n square matrices with matrix square roots* $\mathbf{A}_1^{1/2}, \mathbf{A}_2^{1/2}, \ldots, \mathbf{A}_n^{1/2}$*, respectively. Moreover, let*

$$\mathfrak{Q}\mathfrak{R} = \begin{bmatrix} \mathbf{A}_1^{1/2} \\ \mathbf{A}_2^{1/2} \\ \vdots \\ \mathbf{A}_n^{1/2} \end{bmatrix} \tag{3.4}$$

denote the reduced QR decomposition (Trefethen & Bau, 2022) of the matrix that results from stacking $\mathbf{A}_1^{1/2}, \mathbf{A}_2^{1/2}, \ldots, \mathbf{A}_n^{1/2}$ *vertically, with* $\mathfrak{Q} \in \mathbb{R}^{M \times N}$ *and* $\mathfrak{R} \in \mathbb{R}^{N \times N}$ *being orthogonal and upper-triangular matrices, respectively. Then,* $\mathfrak{R}$ *is a matrix square root of the sum of matrices* $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n$*, i.e.,*

$$\mathfrak{R} = \left(\sum_{i=1}^{n} \mathbf{A}_i \right)^{1/2}.$$

Proof. We first rewrite the sum of matrices as a product of stacked matrix square roots, applying the definition of the matrix square root (cf. Section 1.1). Subsequently, we insert Eq. (3.4) and exploit the fact that the matrix $\mathfrak{Q}$ is orthogonal, hence $\mathfrak{Q}^\top \mathfrak{Q} = \mathbf{I}$. Last, we apply the definition of the matrix square root again.

$$\begin{aligned}
\left(\sum_{i=1}^{n} \mathbf{A}_i \right)^{1/2} &= \left(\begin{bmatrix} \left(\mathbf{A}_1^{1/2}\right)^\top & \left(\mathbf{A}_2^{1/2}\right)^\top & \cdots & \left(\mathbf{A}_n^{1/2}\right)^\top \end{bmatrix} \begin{bmatrix} \mathbf{A}_1^{1/2} \\ \mathbf{A}_2^{1/2} \\ \vdots \\ \mathbf{A}_n^{1/2} \end{bmatrix} \right)^{1/2} \\
&\overset{(3.4)}{=} \left(\mathfrak{R}^\top \mathfrak{Q}^\top \mathfrak{Q} \mathfrak{R} \right)^{1/2} \\
&= \left(\mathfrak{R}^\top \mathfrak{R} \right)^{1/2} \\
&= \mathfrak{R}
\end{aligned}$$

□

Definition 3.3. Let

$$\mathfrak{QR} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_n \end{bmatrix}$$

denote the reduced QR decomposition of the matrix that results from stacking n matrices $\mathbf{B}_1 \in \mathbb{R}^{M_1 \times N}, \mathbf{B}_2 \in \mathbb{R}^{M_2 \times N}, \ldots, \mathbf{B}_n \in \mathbb{R}^{M_n \times N}$ vertically. For this case, we define the shorthand notation

$$\mathrm{qr}_\mathrm{r}(\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_n) \coloneqq \mathfrak{R}.$$

When performing inference, we assume that the initial value $\mathbf{x}_0$ is known, so the initial covariance matrix square root is the zero matrix. For the both-sided matrix product between the Jacobian $\mathbf{Jf}(\mathbf{m}_{k-1})$ and the previous filtering covariance matrix $\mathbf{P}_{k-1}$ in Eq. (2.36), its square root evaluates to $\mathbf{P}_{k-1}^{1/2}\mathbf{Jf}(\mathbf{m}_{k-1})^\mathsf{T}$, since

$$\begin{aligned} \underbrace{\mathbf{Jf}(\mathbf{m}_{k-1})\mathbf{P}_{k-1}\mathbf{Jf}(\mathbf{m}_{k-1})^\mathsf{T}}_{\mathbf{A}} &= \mathbf{Jf}(\mathbf{m}_{k-1}) \left(\mathbf{P}_{k-1}^{1/2}\right)^\mathsf{T} \mathbf{P}_{k-1}^{1/2}\mathbf{Jf}(\mathbf{m}_{k-1})^\mathsf{T} \\ &= \underbrace{\left(\mathbf{P}_{k-1}^{1/2}\mathbf{Jf}(\mathbf{m}_{k-1})^\mathsf{T}\right)^\mathsf{T}}_{\left(\mathbf{A}^{1/2}\right)^\mathsf{T}} \underbrace{\mathbf{P}_{k-1}^{1/2}\mathbf{Jf}(\mathbf{m}_{k-1})^\mathsf{T}}_{\mathbf{A}^{1/2}}. \end{aligned}$$

Finally, by Proposition 3.1, the addition of the process noise matrix square root can be realized using a QR decomposition. Algorithm 3 lists the computational steps of the EKF in square-root form, applied to the probabilistic model defined by Eq. (3.3) as transition model and without observation model.

Compared to sequential sampling, the EKF requires only one call of the numerical ODE solver per integration step, but still increases the effort over a pure run of the solver. Due to additional matrix multiplication and QR decomposition, the computational complexity is $\mathcal{O}(N^3D^3)$ (N. J. Higham, 2008).

3.1.3 | Assumptions and Requirements

The idea behind the presented approach is to extend existing black-box simulators by probabilistic output, avoiding their replacement in otherwise well-working settings. Applying the approach, however, is tied to a few assumptions and requirements, which are set out below.

Local Error Estimator Trivially, a numerical ODE solver has to provide a local error estimator in order to be used in the context of the probabilistic model defined by Eq. (3.3).

Algorithm 3: Extended Kalman filter (square-root form, no update step)

Input:

Ξ_δ	Numerical ODE solver with local error estimator
$\mathbf{f}_\theta$	Vector field
$\widehat{\mathbb{T}}$	Equidistant time discretization
h	Step size
$\mathbf{x}_0$	Initial value

Output:

$\{\widehat{\mathbf{m}}_k\}_{k=1}^K$	Prediction means
$\{\widehat{\mathbf{P}}_k^{1/2}\}_{k=1}^K$	Prediction covariance matrix square roots

$\widehat{\mathbf{m}}_0 = \mathbf{x}_0$

$\widehat{\mathbf{P}}_0^{1/2} = \mathbf{0}$

for $k = 1, 2, \ldots, K$ **do**

$$\left[\widehat{\mathbf{m}}_k \quad \widehat{\delta\mathbf{x}}_k\right]^\intercal = \Xi_\delta\left(\mathbf{f}_\theta, t_{k-1}, \widehat{\mathbf{m}}_{k-1}, h\right)$$

$$\widehat{\mathbf{P}}_k^{1/2} = \mathrm{qr}_\mathrm{r}\left(\widehat{\mathbf{P}}_{k-1}^{1/2}\left(\mathbf{J}_{\widehat{\mathbf{m}}_{k-1}}\left[\Xi_\delta\left(\mathbf{f}_\theta, t_{k-1}, \widehat{\mathbf{m}}_{k-1}, h\right)\right]_1\right)^\intercal, \mathrm{diag}\left(\widehat{\delta\mathbf{x}}_k\right)\right)$$

end

The estimator should not be too computationally demanding compared to the solver itself, which would render the approach unattractive. A good and efficient example are embedded RK methods as introduced in Section 2.2.4, which compute the error estimator with minimal overhead. Another way would be to use a solver without error estimator, but running additional solves with halved step size in parallel, whose difference to the main solve then represents an estimate for the numerical error. In this case, however, one could also half the step size for the main solve, increasing the solution's accuracy and potentially making a probabilistic solution obsolete.

Furthermore, the estimator should be of sufficiently high accuracy. While underestimating the true error would, in the worst case, lead to a highly overconfident solution, overestimation poses a different issue: If sequential sampling is applied to create a collection of sample trajectories, overly large error estimates could lead to perturbations far away from the true solution with non-zero probability and hence cause divergent behavior. Conrad et al. (2017) conducted an extensive theoretical analysis on convergence properties of their probabilistic model. They showed that the unknown stochastic process which underlies computed sample trajectories converges to the true solution with the same polynomial rate in the step size h as a used RK method, if the model's covariance is

upper bounded with at least the doubled polynomial order. Applied to the probabilistic model in the present work, this means that such convergence is guaranteed, if the local error estimator is of the same or higher order as/than the numerical ODE solver. Popular embedded RK methods that perform local extrapolation, for example, do not meet this assumption, as the order of their error estimator is at least one less than the order of the main solver itself. In Chapter 4, we will examine whether this affects the applicability of the probabilistic model to those methods.

Time Discretization If the chosen time discretization is too coarse, corresponding to a too large step size h, the accuracy of a numerical ODE solution will generally suffer. In addition, this will also violate the inherent assumption of the probabilistic model defined by Eq. (3.3), i.e., the true next state being normally distributed around the solver's prediction: If the solver's prediction is a bad estimate for the true next state, centering the normal distribution at it will introduce a systematic bias, as already identified by Conrad et al. (2017). Moreover, overly large step sizes are likely to decrease the accuracy of the local error estimator as well, which then leads to the same problems as outlined in the preceding paragraph. If one performs inference using the EKF, this might also violate the EKF's local affinity assumption (cf. Assumption 2.5). As a consequence, the propagated uncertainty could be distorted substantially when consecutive states $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ are far apart.

Ultimately, choosing an appropriate step size is both dependent on the given simulation problem and the used numerical solver. We will investigate the influence of different, both more fine- and coarse-grained time discretizations on the quality of the probabilistic solutions in Chapter 4.

Differentiability The EKF requires the computation of Jacobian matrices of the numerical ODE solver w.r.t. its input state (cf. Algorithm 3). Thus, in order to apply the EKF to the probabilistic model proposed in Definition 3.2, the used numerical ODE solver needs to be at least once differentiable. Theoretically, this is the case with the vast majority of common ODE solvers, in particular with RK methods. In practice, however, it might be a problem for legacy software, which often does not offer the automatic differentiation capabilities of modern scientific computing libraries like JAX (Bradbury et al., 2018) or PyTorch (Ansel et al., 2024).

3.2 | ODE Parameter Estimation

Another goal of this work is to estimate unknown parameters of ODEs reliably, given their functional form and a collection of noisy measurements of the state. Building upon the probabilistic model defined in Section 3.1, the approach adopted here in general inherits the assumptions and requirements listed in Section 3.1.3. Moreover, it is fundamentally based on gradient-based maximum likelihood estimation, which is described below.

3.2.1 | Gradient-based Maximum Likelihood Estimation

Assumption 3.1. For an ODE parametrized by θ, assume partially observed measurements $\mathbf{y}_1, \ldots, \mathbf{y}_K \in \mathbb{R}^L$ of a true trajectory of states $\mathbf{x}_1, \ldots, \mathbf{x}_K \in \mathbb{R}^{ND}$, with $\mathbf{H} \in \mathbb{R}^{L \times ND}$ denoting the matrix that maps a state $\mathbf{x}_k$ to its partially observed counterpart $\mathbf{y}_k$. In addition, assume that observations are distorted by zero-mean Gaussian noise of covariance $\mathbf{R} \in \mathbb{R}^{L \times L}$, i.e.,

$$\mathbf{y}_k \sim p\left(\mathbf{y}_k \mid \mathbf{x}_k\right) \coloneqq \mathcal{N}\left(\mathbf{y}_k; \mathbf{H}\mathbf{x}_k, \mathbf{R}\right). \tag{3.5}$$

Assumption 3.2. Assume that observations $\mathbf{y}_1, \ldots, \mathbf{y}_K$ are conditionally independent, given parameters θ, i.e.,

$$p\left(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \mathbf{y}_{k+1:K}, \theta\right) = p\left(\mathbf{y}_k \mid \theta\right).$$

Assumption 3.3. Assume that a trajectory of states $\mathbf{x}_1, \ldots, \mathbf{x}_K$ is uniquely defined by parameters θ of the ODE, and vice versa, i.e., there are no two different parametrizations $\theta^{(1)} \neq \theta^2$ that induce the same trajectory of states $\mathbf{x}_1, \ldots, \mathbf{x}_K$, and there are no two different trajectories of states at equal points in time $\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_K^{(1)}$ and $\mathbf{x}_1^{(2)}, \ldots, \mathbf{x}_K^{(2)}$ that correspond to the same parameters θ, provided that they share the same initial value $\mathbf{x}_0$.

Definition 3.4. For a conditional distribution $p(\mathbf{y}_{1:K} \mid \theta)$ of observations $\mathbf{y}_{1:K}$, given parameters θ, the *likelihood function* $\mathcal{L}$ of θ is defined as

$$\mathcal{L}(\theta; \mathbf{y}_{1:K}) \coloneqq p(\mathbf{y}_{1:K} \mid \theta).$$

Furthermore, the *maximum likelihood estimate* $\hat{\theta}_{\text{MLE}}$ of θ is defined as

$$\hat{\theta}_{\text{MLE}} \coloneqq \arg\max_{\theta} \mathcal{L}(\theta; \mathbf{y}_{1:K}). \tag{3.6}$$

Under Assumptions 3.1 to 3.3, the likelihood function of parameters θ evaluates to

$$\mathcal{L}(\theta; \mathbf{y}_{1:K}) = \prod_{k=1}^{K} p\left(\mathbf{y}_k \mid \theta\right) = \prod_{k=1}^{K} p\left(\mathbf{y}_k \mid \mathbf{x}_k\right) = \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{y}_k; \mathbf{H}\mathbf{x}_k, \mathbf{R}\right). \tag{3.7}$$

Equation (3.7) is usually intractable in practice, as the solution of the ODE is unknown and the true states $\mathbf{x}_1, \ldots, \mathbf{x}_K$ are therefore not available. Through numerical integration of the ODE, however, the true states can be approximated by the predicted states $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_K$, giving rise to a tractable approximation of Eq. (3.7).

Definition 3.5. Let $\mathbf{f}_\theta$ be a vector field parametrized by $\theta \in \mathbb{R}^W$, let $\widehat{\mathbb{T}}$ be an equidistant time discretization with corresponding step size h, and let Ξ be a numerical ODE solver. Moreover, let $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_K$ denote the trajectory of states predicted by Ξ, starting at an initial value $\mathbf{x}_0$, and let $\mathbf{y}_1, \ldots, \mathbf{y}_K$ denote a collection of partially observed, noisy measurements of the true states $\mathbf{x}_1, \ldots, \mathbf{x}_K$, according to Assumption 3.1. Then, a tractable approximation $\mathcal{L}_{\text{NLSR}}$ to the exact likelihood function $\mathcal{L}$ is defined by

$$\mathcal{L}(\theta; \mathbf{y}_{1:K}) \approx \mathcal{L}_{\text{NLSR}}(\theta; \mathbf{y}_{1:K}) \coloneqq \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{y}_k; \mathbf{H}\hat{\mathbf{x}}_k, \mathbf{R}\right). \tag{3.8}$$

Since maximization of the likelihood $\mathcal{L}_{\text{NLSR}}(\theta; \mathbf{y}_{1:K})$ is equivalent to minimizing the negative log-likelihood (NLL), the optimization problem in Eq. (3.6) can be rewritten as

$$\begin{aligned} \hat{\theta}_{\text{MLE}} &= \arg\max_{\theta} \mathcal{L}(\theta; \mathbf{y}_{1:K}) \\ &= \arg\min_{\theta} -\log \mathcal{L}(\theta; \mathbf{y}_{1:K}) \\ &\overset{(3.8)}{\approx} \arg\min_{\theta} \frac{1}{K} \sum_{k=1}^{K} \|\mathbf{H}\hat{\mathbf{x}}_k - \mathbf{y}_k\|_2^2. \end{aligned} \tag{3.9}$$

This is also known as *non-linear least-squares-regression (NLSR)* (Bard, 1974). If the used numerical solver Ξ is differentiable, gradient-based methods can be applied to solve the optimization problem in Eq. (3.9), e.g., the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm (Liu & Nocedal, 1989).

Unfortunately, the above NLL function is generally non-convex in θ and often contains a multitude of local minima in case of non-linear dynamics (Cao et al., 2011). In order to converge to the global minimum that corresponds to the desired true parameters, the optimization must therefore be restarted for several initializations of θ, which quickly becomes infeasible in high-dimensional parameter spaces.

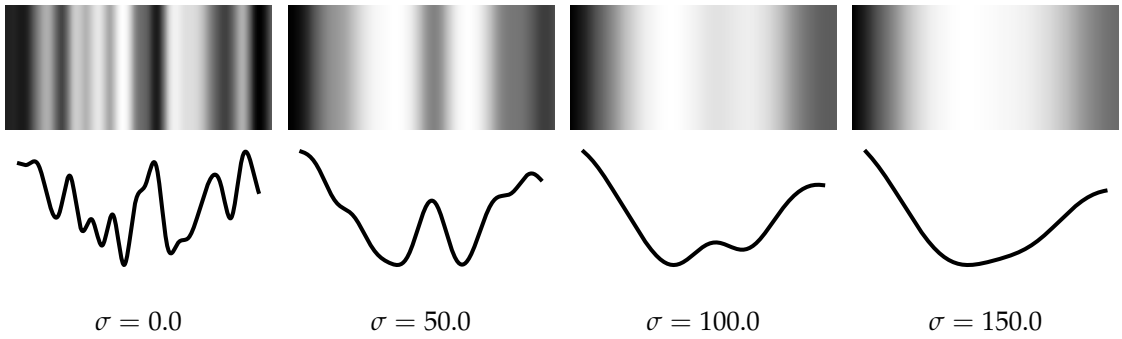


Figure 3.2: Idea behind process noise tempering. An analogy: By increasing the variance σ^2 of the Gaussian filter applied to a one-dimensional image (top), the corresponding intensity profile (bottom) becomes more convex.

3.2.2 | Process Noise Tempering

This work's approach to estimate the true parameters more reliably, i.e., determining the global optimum for various initializations in the majority of cases, can be outlined as follows: First, we observe that the likelihood function $\mathcal{L}_{\text{NLSR}}$ (cf. Definition 3.5) assumes no uncertainty about parameters in the process of numerical integration. However, we can construct a Gaussian transition model between subsequent states that indirectly accounts for the uncertainty about parameters in a controllable manner. Together with available observations and the known observation model, this defines a probabilistic SSM that can be inferred approximately using the EKF, leading to a tractable likelihood on parameters that incorporates the uncertainty about them. Finally, we can gradually decrease the uncertainty in the transition model and optimize the resulting likelihood function. In the following, this method is referred to as *process noise tempering (PNT)*.

The idea behind PNT is to smooth out the NLL function and therefore make it convex initially, enabling convergence to the global minimum of this manipulated objective function. In general, the global minimum of the smoothed NLL is not identical to the global optimum of the original NLL, but it is often located in the vicinity of the latter. By iteratively restarting optimization at the previous point of convergence and reducing the magnitude of smoothing, the true global minimum may be reached ultimately. Related techniques to solve non-convex optimization problems have been reported in the literature under various names, e.g., *graduated optimization* (Blake & Zisserman, 1987) in computer vision and *deterministic annealing* (Rose, 1998) in the context of clustering. Process noise tempering has been inspired in particular by the work of Beck et al. (2024), where the authors gradually lowered the diffusion parameter of a PN ODE solver, which controls the uncertainty in the prior model over solutions.

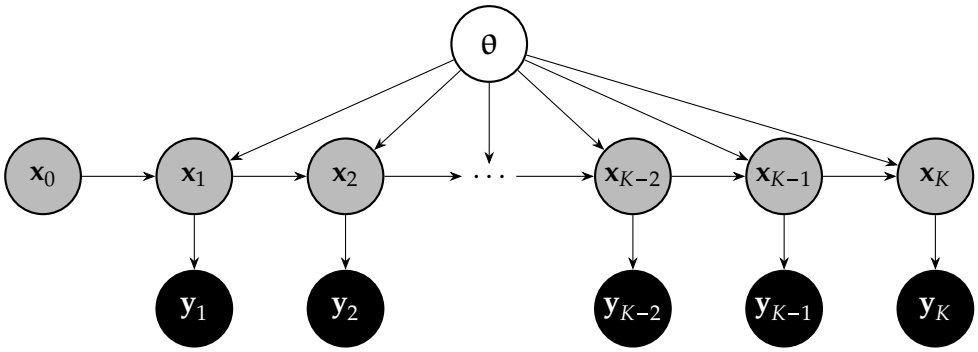


Figure 3.3: Dynamic Bayesian network of the probabilistic SSM used for parameter estimation: Parameters θ (○), unobserved states $\mathbf{x}_{0:K}$ (●), observations $\mathbf{y}_{1:K}$ (●).

Example 3.1. The basic principle of PNT can be exemplified by the following analogy, illustrated in Fig. 3.2: Consider a high-frequent one-dimensional grayscale image, similar to a barcode, but with smooth transitions and differently bright and dark patches. The intensity profile of this image is non-convex and contains several local minima. By applying a low-pass Gaussian filter to the image, high frequencies that induce non-convexity are eliminated and the intensity profile of the resulting blurred image becomes more convex. The strength of this smoothing effect depends on the variance σ^2 of the Gaussian filter. By starting with a large variance and then lowering it, an initially convex approximation to the true intensity profile can be gradually transformed into the latter.

Assumption 3.1 gives rise to a probabilistic SSM on states $\mathbf{x}_k$ and observations $\mathbf{y}_k$, where the states are conditioned on parameters θ through the solution of the ODE. Here, we assume that the initial value $\mathbf{x}_0$ is known beforehand. This SSM is visualized by Fig. 3.3. While its observation model $p(\mathbf{y}_k \mid \mathbf{x}_k)$ is defined by Eq. (3.5), the transition model $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta)$ has not been specified yet. Therefore, it can be used to encode the uncertainty about parameters θ.

Remark 3.2. A priori knowledge of the initial value $\mathbf{x}_0$ is a simplifying assumption made in the present model. In the scope of this work, we generally consider only the case of fully known initial values. If unknown, however, the initial value could in principle also be treated as part of the parameters to be estimated.

Definition 3.6. Let $\mathbf{f}_\theta$ be a vector field parametrized by θ, let $\widehat{\mathbb{T}}$ be an equidistant time discretization with corresponding step size h, and let Ξ_δ be a numerical ODE solver with local error estimator. Moreover, according to Definition 2.10, let

$$\begin{bmatrix} \widehat{\mathbf{x}}_{k+1} & \widehat{\delta \mathbf{x}}_{k+1} \end{bmatrix}^\intercal := \Xi_\delta \left(\mathbf{f}_\theta, t_k, \mathbf{x}_k, h \right)$$

denote the predicted state and the estimated local error at time $t_{k+1} \in \widehat{\mathbb{T}}$, given the previous time $t_k \in \widehat{\mathbb{T}}$ and corresponding state $\mathbf{x}_k \in \mathbb{R}^{ND}$. Finally, let $\gamma \in \mathbb{R}_0^+$ denote a scaling factor that controls the uncertainty about parameters θ, called *process noise temperature*. We define the following probabilistic model over the state $\mathbf{x}_{k+1}$, given the previous state $\mathbf{x}_k$ and parameters θ:

$$p_{\text{PNT}}\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \theta\right) \coloneqq \mathcal{N}\left(\mathbf{x}_{k+1}; \hat{\mathbf{x}}_{k+1}, \gamma \mathbf{I}\right). \tag{3.10}$$

Combining with the computational uncertainty of the solver (cf. Definition 3.2), we also define

$$p_{\text{PNT+num.}}\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \theta\right) \coloneqq \mathcal{N}\left(\mathbf{x}_{k+1}; \hat{\mathbf{x}}_{k+1}, \gamma \mathbf{I} + \operatorname{diag}\left(\widehat{\delta \mathbf{x}}_k^2\right)\right). \tag{3.11}$$

The transition models introduced in Definition 3.6 indirectly account for the uncertainty about parameters, as higher values of γ lead to a lower confidence in the solution of the integration steps. Since these models are non-linear Gaussian and the observation model defined by Eq. (3.5) is linear Gaussian, the EKF can be applied again to perform approximate inference in the SSM. As in Section 3.1, we use a square-root form to increase numerical stability. This becomes even more important here with the usage of the EKF update step because Eq. (2.41) harbors the risk that the covariance matrix becomes indefinite after the subtraction of matrices.

Overall, the predict step is the same as shown in Algorithm 3, except for the different process noise matrix, dependent on the used transition model. Computing the data likelihood covariance matrix square root $\mathbf{S}_k^{1/2}$ is analogous to the prediction covariance matrix square root $\hat{\mathbf{P}}_k$. Since $\mathbf{S}_k^{1/2}$ is upper-triangular after QR decomposition, the Kalman gain $\mathbf{K}_k$ can be determined efficiently using forward and backward substitution. Last, for the computation of the filtering covariance matrix square root $\mathbf{P}_k^{1/2}$, we observe that Eq. (2.41) can be rewritten as

$$\begin{aligned}
\mathbf{P}_k &= \hat{\mathbf{P}}_k - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\top \\
&\overset{(2.38)}{=} \hat{\mathbf{P}}_k - \mathbf{K}_k \left(\mathbf{H} \hat{\mathbf{P}}_k \mathbf{H}^\top + \mathbf{R}\right) \mathbf{K}_k^\top \\
&= \hat{\mathbf{P}}_k - \mathbf{K}_k \mathbf{H} \hat{\mathbf{P}}_k \mathbf{H}^\top \mathbf{K}_k^\top + \mathbf{K} \mathbf{R} \mathbf{K}^\top \\
&= \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}\right) \hat{\mathbf{P}}_k \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}\right)^\top + \mathbf{K} \mathbf{R} \mathbf{K}^\top,
\end{aligned} \tag{3.12}$$

where we already inserted $\mathbf{J}\mathbf{h}(\widehat{\mathbf{m}}_k) = \mathbf{H}$. This factorization is also known as *Joseph form* (Bucy & Joseph, 2005). With the square roots of both summands being easily obtainable, Eq. (3.12) then enables computing $\mathbf{P}_k^{1/2}$ by means of another QR decomposition. All computational steps are also listed in Algorithm 4.

Algorithm 4: Extended Kalman filter (square-root form)

Input:

Ξ_δ	Numerical ODE solver with local error estimator
$\mathbf{f}_\theta$	Vector field
$\widehat{\mathbb{T}}$	Equidistant time discretization
h	Step size
$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta)$	Transition model
$\mathbf{x}_0$	Initial value
$\mathbf{y}_{1:K}$	Observations
$\mathbf{H}$	Observation matrix
$\mathbf{R}^{1/2}$	Observation noise covariance matrix square root

Output:

$\{\widehat{\mathbf{m}}_k\}_{k=1}^K$	Prediction means
$\{\hat{\mathbf{P}}_k^{1/2}\}_{k=1}^K$	Prediction covariance matrix square roots
$\{\mathbf{m}_k\}_{k=1}^K$	Filtering means
$\{\mathbf{P}_k^{1/2}\}_{k=1}^K$	Filtering covariance matrix square roots
$\{\hat{\mathbf{y}}_k\}_{k=1}^K$	Data likelihood means
$\{\mathbf{S}_k^{1/2}\}_{k=1}^K$	Data likelihood covariance matrix square roots

$\mathbf{m}_0 = \mathbf{x}_0$

$\mathbf{P}_0^{1/2} = \mathbf{0}$

for $k = 1, 2, \ldots, K$ **do** ▷ $\mathbf{Q}_{k-1} \coloneqq \mathrm{Cov}(p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta))$

Prediction step:

$$\begin{bmatrix} \widehat{\mathbf{m}}_k & \widehat{\delta \mathbf{x}}_k \end{bmatrix}^\intercal = \Xi_\delta\left(\mathbf{f}_\theta, t_{k-1}, \mathbf{m}_{k-1}, h\right)$$

$$\hat{\mathbf{P}}_k^{1/2} = \mathrm{qr}_\mathrm{r}\left(\mathbf{P}_{k-1}^{1/2}\left(\mathbf{J}_{\mathbf{m}_{k-1}}\left[\Xi_\delta\left(\mathbf{f}_\theta, t_{k-1}, \mathbf{m}_{k-1}, h\right)\right]_1\right)^\intercal, \mathbf{Q}_{k-1}^{1/2}\right)$$

Update step:

$$\hat{\mathbf{y}}_k = \mathbf{H}\widehat{\mathbf{m}}_k$$

$$\mathbf{S}_k^{1/2} = \mathrm{qr}_\mathrm{r}\left(\hat{\mathbf{P}}_k^{1/2}\mathbf{H}^\intercal, \mathbf{R}^{1/2}\right)$$

$$\mathbf{K}_k = \hat{\mathbf{P}}_k\mathbf{H}^\intercal\mathbf{S}_k^{-1}$$

$$\mathbf{m}_k = \widehat{\mathbf{m}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right)$$

$$\mathbf{P}_k^{1/2} = \mathrm{qr}_\mathrm{r}\left(\hat{\mathbf{P}}_k^{1/2}\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)^\intercal, \mathbf{R}^{1/2}\mathbf{K}_k^\intercal\right)$$

end

Proposition 3.2. *The conditional distribution $p(\mathbf{y}_k \mid \theta)$ is approximately[2] equal to the data likelihood $\mathcal{N}\left(\mathbf{y}_k; \hat{\mathbf{y}}_k, \mathbf{S}_k\right)$ computed in the update step of the EKF.*

Proof. We exploit the Markov independence structure of the SSM and apply basic rules of probability theory. Under Assumption 2.5, the prediction distribution can be approximately replaced by a Gaussian that is specified by the EKF's prediction mean and prediction covariance. Applying rules for the product of two Gaussians and the marginalization of a Gaussian (cf. Särkkä and Svensson (2023)), we ultimately show approximate equality to the data likelihood computed in the EKF.

$$
\begin{aligned}
p\left(\mathbf{y}_k \mid \theta\right) &= p\left(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \theta\right) \\
&= \int p\left(\mathbf{y}_k, \mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta\right) \mathrm{d}\mathbf{x}_k \\
&= \int p\left(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1}, \theta\right) p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta\right) \mathrm{d}\mathbf{x}_k \\
&= \int p\left(\mathbf{y}_k \mid \mathbf{x}_k\right) p\left(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}, \theta\right) \mathrm{d}\mathbf{x}_k \\
&\approx \int \mathcal{N}\left(\mathbf{y}_k; \mathbf{H}\mathbf{x}_k, \mathbf{R}\right) \mathcal{N}\left(\mathbf{x}_k; \widehat{\mathbf{m}}_k, \widehat{\mathbf{P}}_k\right) \mathrm{d}\mathbf{x}_k \\
&= \mathcal{N}\left(\mathbf{y}_k; \mathbf{H}\widehat{\mathbf{m}}_k, \mathbf{H}\widehat{\mathbf{P}}_k\mathbf{H}^T + \mathbf{R}\right) \\
&\overset{(2.38)}{=} \mathcal{N}\left(\mathbf{y}_k; \hat{\mathbf{y}}_k, \mathbf{S}_k\right)
\end{aligned}
$$

□

With Proposition 3.2 and the Markov independence structure of the SSM, we therefore obtain a tractable approximation to the likelihood function $\mathcal{L}(\theta; \mathbf{y}_{1:K})$ that can incorporate various magnitudes of uncertainty about the parameters θ, controlled by the value of γ in the transition model (cf. Definition 3.6).

Definition 3.7. Let $\mathbf{f}_\theta$ be a vector field parametrized by $\theta \in \mathbb{R}^W$, let $\widehat{\mathbb{T}}$ be an equidistant time discretization with corresponding step size h, and let Ξ be a numerical ODE solver. Moreover, let $\mathbf{x}_0$ be an initial value, let $\mathbf{y}_1, \ldots, \mathbf{y}_K$ denote a collection of partially observed, noisy measurements of the true states $\mathbf{x}_1, \ldots, \mathbf{x}_K$, according to Assumption 3.1, and let $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \theta)$ denote a transition model between consecutive states, whose covariance is scaled by the process noise temperature γ. Finally, let $\{\hat{\mathbf{y}}_k\}_{k=1}^K$ and $\{\mathbf{S}_k\}_{k=1}^K$ denote the data likelihood means and covariance matrices obtained when applying the EKF as presented in Algorithm 4 to the above configuration. Then, a tractable approximation $\mathcal{L}_{\mathrm{PNT}}$ to the

[2] Up to linearization of the transition model.

exact likelihood function $\mathcal{L}$ is defined by

$$\mathcal{L}(\theta; \mathbf{y}_{1:K}) \approx \mathcal{L}_{\text{PNT}}(\theta; \mathbf{y}_{1:K}, \gamma) \coloneqq \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{y}_k; \hat{\mathbf{y}}_k, \mathbf{S}_k\right). \tag{3.13}$$

Process noise tempering starts with an initial parametrization θ_0, e.g., an educated guess or just a random initialization. For a fixed number of iterations $m \in \mathbb{N}$, in every iteration i, first a process noise temperature γ_i is obtained through a pre-defined *temperature schedule* $\Gamma : \mathbb{N} \to \mathbb{R}_0^+$. An optimization algorithm OPT then optimizes the process noise-tempered likelihood $\mathcal{L}_{\text{PNT}}$ stated in Eq. (3.13) w.r.t. parameters θ for γ_i and the current parameter values θ_i. Finally, the parameters θ_m determined in the last iteration are returned as estimate $\hat{\theta}_{\text{PNT}}$. This procedure is related to Beck et al. (2024) and summarized in Algorithm 5.

Algorithm 5: Process noise tempering

Input:

θ_0	Initial parameters
$\mathbf{y}_{1:K}$	Observations
Γ	Tempering schedule
$\mathcal{L}_{\text{PNT}}(\theta; \mathbf{y}_{1:K}, \gamma)$	Process-noise-tempered likelihood
OPT	Optimization algorithm
m	Number of iterations

Output:

$\hat{\theta}_{\text{PNT}}$	Estimated parameters

for $i = 1, 2, \ldots, m$ **do**
  $\gamma_i = \Gamma(i)$
  $\theta_i = \text{OPT}\left(\mathcal{L}_{\text{PNT}}(\,\cdot\,; \mathbf{y}_{1:K}, \gamma_i), \theta_{i-1}\right)$
end

$\hat{\theta}_{\text{PNT}} = \theta_m$

The effect of different tempering schedules Γ on efficiency and quality of the results will be examined in Chapter 4. Moreover, we will investigate how well PNT works for increasingly complex ODEs and a growing number of parameters to be optimized.

Evaluation | 4

In this chapter, the methods presented in Chapter 3 are evaluated experimentally. The experiments are designed and conducted with the aim to answer the following research questions:

Q1.1. How well are probabilistic solutions created with the model $p_{\text{num.}}$ calibrated, i.e., reflect the numerical error appropriately, for different ODE solvers with local error estimator Ξ_δ, compared to a manually calibrated baseline provided by Conrad et al. (2017)?

Q1.2. Does inference using the extended Kalman filter yield similar probabilistic solutions as with sequential sampling, and how does the coarseness of time discretizations $\widehat{\mathbb{T}}$ affect these solutions?

Q2.1. How reliable and efficient is the estimation of ODE parameters using process noise tempering for different tempering schedules Γ and with/without addition of local error estimates, compared to the baseline of non-linear least-squares-regression and the method by Beck et al. (2024)?

Q2.2. For how complex and highly parametrized ODEs does process noise tempering still reliably provide correct parameter estimates?

The remainder of this chapter is structured as follows: First, Section 4.1 contains the experiments regarding black box probabilistic ODE solvers, addressing Q1.1 and Q1.2. Subsequently, Section 4.2 deals with experiments on ODE parameter estimation that approach Q2.1 and Q2.2. For details on the implementation and computing environment used to conduct them, see Appendix A.

4.1 | Black Box Probabilistic ODE Solvers

We conduct the following three experiments on black box probabilistic ODE solvers:

Experiment 1.1. Quantitative assessment of uncertainty calibration, addressing Q1.1.

Experiment 1.2. Qualitative assessment of uncertainty calibration, addressing Q1.1 and Q1.2.

Experiment 1.3. Effects of overly coarse time discretizations, addressing Q1.2.

4.1.1 | Experiment 1.1: Quantitative Assessment of Uncertainty Calibration

In the first experiment, the calibration of probabilistic ODEs solutions is compared quantitatively by evaluating the marginal likelihood $p(\mathbf{y}_{1:K})$. More specifically, we consider the NLL, which is minimal for an optimal model fit to observations $\mathbf{y}_{1:K}$. It is again approximated by the data likelihood of the EKF, here without conditioning on parameters θ (cf. Section 3.2). We use the probabilistic model p_{Conrad} defined by Eq. (3.1) as a baseline for comparison with this work's model $p_{\text{num.}}$ (cf. Eq. (3.3)). The NLL is computed for a single simulation using $p_{\text{num.}}$ and for 500 simulations using p_{Conrad}, where the model parameter σ varies with a logarithmically equidistant spacing between 1×10^{-16} and 1. These simulations are repeated for various IVPs and corresponding vector fields $\mathbf{f}_\theta$, different ODE solvers Ξ as well as several time discretizations $\widehat{\mathbb{T}}$, being specified below.

Initial value problems. We consider four different ODEs, each defining an IVP:

First, the *Lotka-Volterra equations* (Lotka, 1925; Volterra, 1928) are a system of two first-order ODEs

$$\frac{\mathrm{d}z_1(t)}{\mathrm{d}t} = \theta_1 z_1(t) - \theta_2 z_1(t) z_2(t), \tag{4.1a}$$

$$\frac{\mathrm{d}z_2(t)}{\mathrm{d}t} = -\theta_3 z_1(t) + \theta_4 z_1(t) z_2(t), \tag{4.1b}$$

describing the interaction of two biological populations, one as a pray and the other as a predator. We specify an IVP with initial value $z_1(0) = z_2(0) = 1$ and parameters $\theta = \begin{bmatrix} 1.5 & 1 & 3 & 1 \end{bmatrix}^\intercal$ for the time interval $\mathbb{T} = [0, 20]$.

Second, the *van der Pol equation* (Guckenheimer, 1980) is a second-order ODE

$$\frac{\mathrm{d}^2 z(t)}{\mathrm{d}t^2} = \theta \left(1 - z^2(t)\right) \frac{\mathrm{d}z(t)}{\mathrm{d}t} - z(t),$$

corresponding to an oscillating system with non-linear damping. We specify an IVP with initial conditions $z(10) = 2$, $\frac{\mathrm{d}z}{\mathrm{d}t}(10) = 10$ and the parameter $\theta = 5$ for the time interval $\mathbb{T} = [10, 80]$.

Third, the *Lorenz'63 system* (Lorenz, 1963) consists of three first-order ODEs

$$\begin{aligned}\frac{\mathrm{d}z_1(t)}{\mathrm{d}t} &= \theta_1(z_2(t) - z_1(t)),\\ \frac{\mathrm{d}z_2(t)}{\mathrm{d}t} &= z_1(t)(\theta_2 - z_3(t)) - z_2(t),\\ \frac{\mathrm{d}z_3(t)}{\mathrm{d}t} &= z_1(t)z_2(t) - \theta_3 z_3(t),\end{aligned}$$

representing a model for atmospheric convection. We specify an IVP with initial value $z_1(0) = z_2(0) = z_3(0) = 1$ and canonical parameters $\boldsymbol{\theta} = \left[10, 28, \frac{8}{3}\right]^\intercal$ for the time interval $\mathbb{T} = [0, 50]$.

Finally, the dynamics of two *linearly coupled anharmonic oscillators* (Steeb et al., 1987) suffice the following system of second-order ODEs

$$\begin{aligned}\frac{\mathrm{d}z_1(t)}{\mathrm{d}t} &= -\theta_1 z_1(t) - \theta_2 z_1^3(t) - \theta_3 z_2(t),\\ \frac{\mathrm{d}z_2(t)}{\mathrm{d}t} &= -\theta_1 z_2(t) - \theta_2 z_2^3(t) - \theta_3 z_1(t).\end{aligned}$$

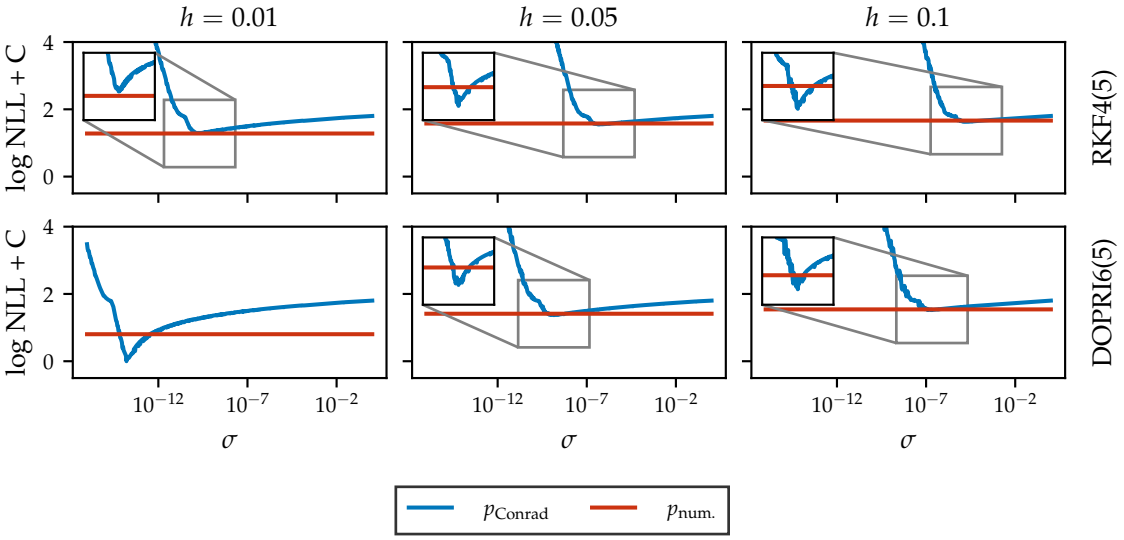
We consider the parametrization $\boldsymbol{\theta} = \left[1, 2, 0.5\right]^\intercal$, and specify an IVP with initial conditions $z_1(0) = 1$, $z_2(0) = -2$, $\frac{\mathrm{d}z_1}{\mathrm{d}t}(0) = -1$, $\frac{\mathrm{d}z_2}{\mathrm{d}t}(0) = 0.5$ for the time interval $\mathbb{T} = [0, 80]$.

Numerical ODE solvers. We use two different explicit embedded RK methods:

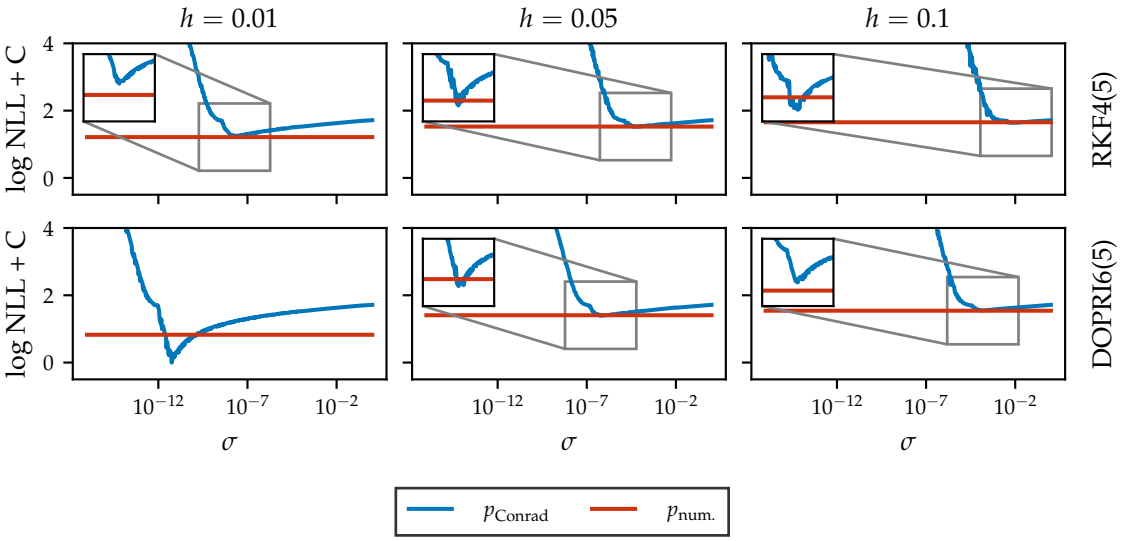
First, the *Runge-Kutta-Fehlberg-4(5) (RKF4(5)) method* (Fehlberg, 1970) is an explicit embedded RK method of order four with a local error estimator of order five. It has six stages.

Second, the *Dormand-Prince-6(5) (DOPRI6(5)) method* (Prince & Dormand, 1981) is an explicit embedded RK method of order six with a local error estimator of order five. It comes with eight stages and, opposite to the RKF4(5) method, performs local extrapolation.

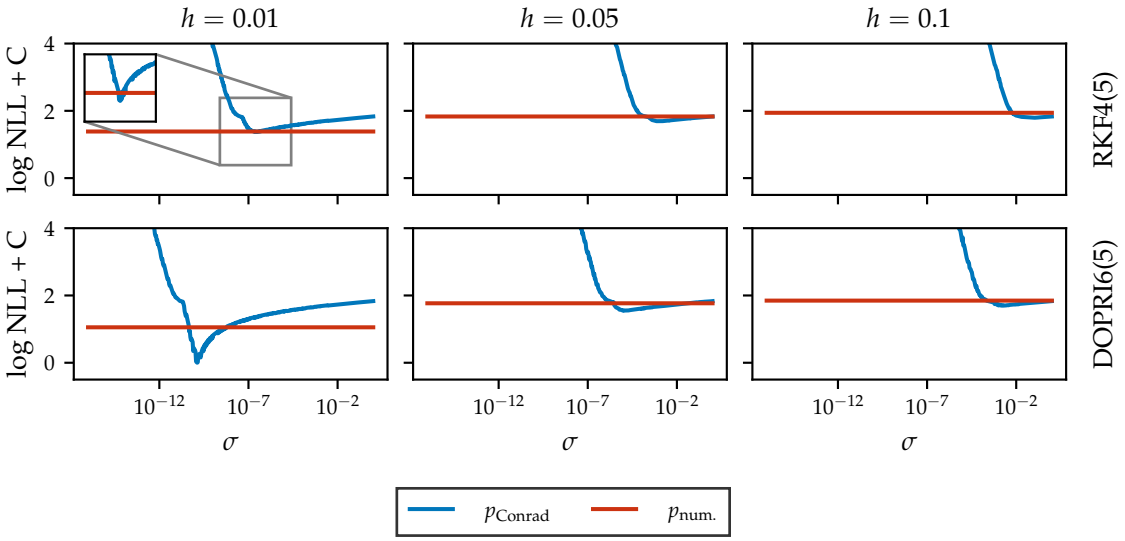
Time discretizations. We use three different constant step sizes h for simulations, being $h = 0.01$, $h = 0.05$, and $h = 0.1$. Together with the time intervals specified for the above IVPs, they give rise to equidistant time discretizations $\widehat{\mathbb{T}}$. In order to make the NLL comparable between time discretizations of varying granularity, it is rescaled by multiplication with h.



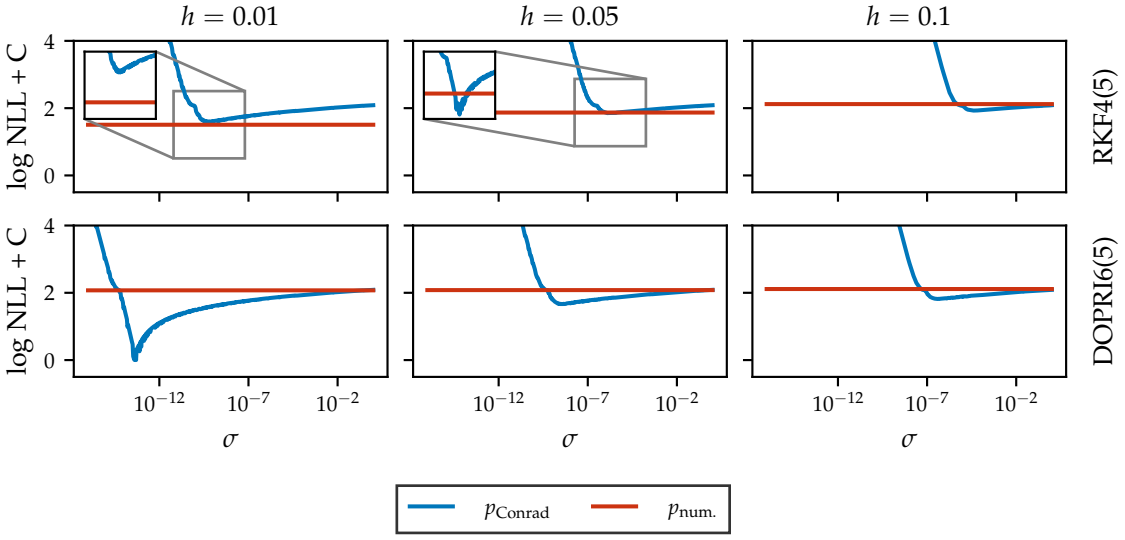
(a) Lotka-Volterra.



(b) Van der Pol.



(c) Lorenz'63.



(d) Linearly coupled anharmonic oscillators.

Figure 4.1: Quantitative evaluation of uncertainty calibration for transition models $p_{\text{num.}}$ and p_{Conrad}. Plots show the log NLL over scaling factors σ of p_{Conrad} for different ODEs (4.1a-4.1d), numerical ODE solvers (rows), and step sizes h (columns). Log NLL curves are shifted to positive values by the same value C, allowing their comparison with each other. A more detailed, zoomed-in view is provided in case of indiscernible intersection between the two curves.

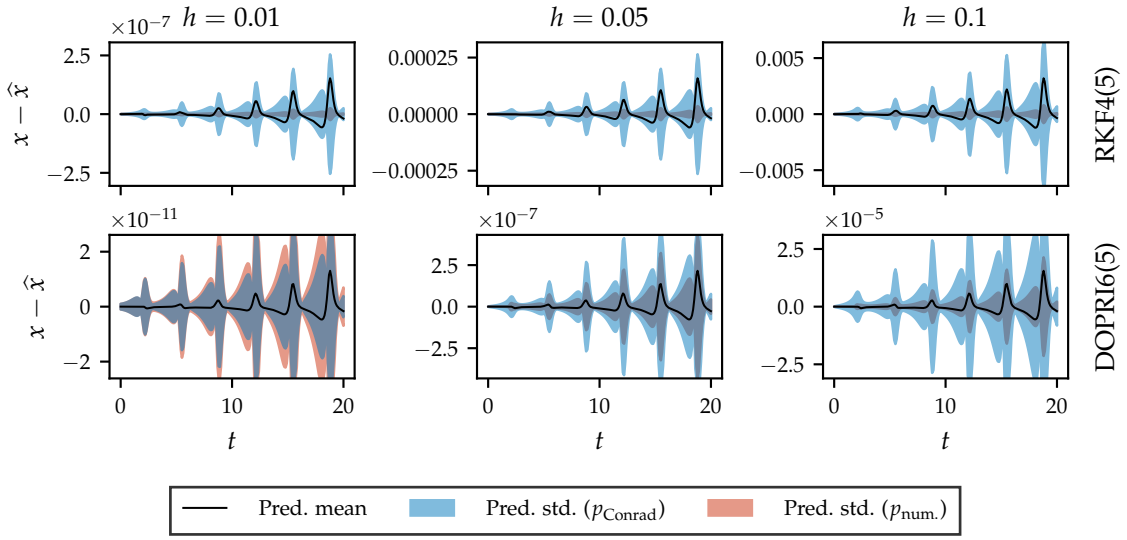
Last, observations $\mathbf{y}_{1:K}$ are produced with a reduced step size $h = 1 \times 10^{-4}$ and the higher-order DOPRI6(5) solver, ensuring that they are close to the exact solutions of the given IVPs. To this end, the measurements are considered fully observable as well as noise-free, i.e., $\mathbf{H} = \mathbf{I}$ and $\mathbf{R} = \mathbf{0}$.

Results Figures 4.1a to 4.1d show the results for ODEs Lotka-Volterra, van der Pol, Lorenz'63, and linearly coupled anharmonic oscillators, respectively. As reported by the authors (Conrad et al., 2017), the model p_{Conrad} is best calibrated for a single value of σ, indicated by sharp minima of the NLL. For larger step sizes h, the minima are located at larger values of σ, supporting the intuition that a coarser time discretization requires an increased uncertainty about numerical integration steps. In most cases, this work's model $p_{\text{num.}}$ induces values of the NLL that are larger than p_{Conrad} at its minima, suggesting that the manual calibration of the latter is better. A recurrent pattern across different ODE, however, is that $p_{\text{num.}}$ produces comparable or lower NLL values than p_{Conrad} for RKF4(5) and $h = 0.01$. According to the NLL, the calibration of $p_{\text{num.}}$ is sometimes better with RKF4(5) (e.g., Fig. 4.1b) and sometimes with DOPRI6(5) (e.g., Fig. 4.1d). In general, it becomes worse for increased step sizes h.

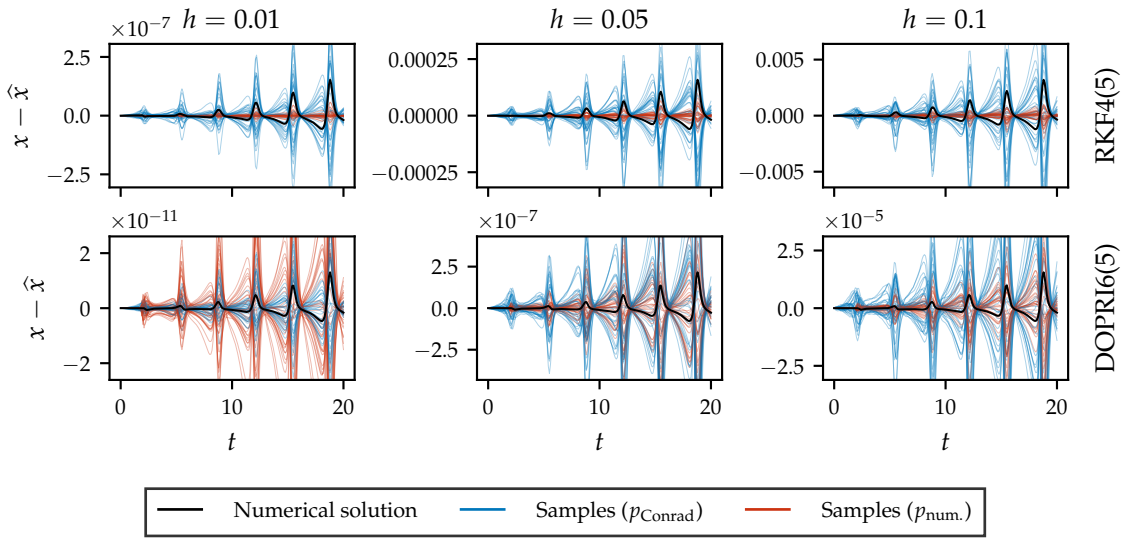
4.1.2 | Experiment 1.2: Qualitative Assessment of Uncertainty Calibration

The second experiment deals with the qualitative evaluation of probabilistic ODE solutions and how well they are calibrated. Analogous to Section 4.1.1, we compare $p_{\text{num.}}$ with p_{Conrad} as a baseline. But this time, we set the scalar variance σ of the latter to the corresponding minimum of the NLL found in Section 4.1.1. To this end, simulations are conducted for the same IVPs, ODE solvers and time discretizations as in Section 4.1.1. Here, however, we assume the more practical situation that observations of the exact solution are not available during simulation, and therefore consider a pure prediction task. We apply both sequential sampling and the EKF to infer probabilistic solutions for the trajectory of states $\mathbf{x}_{1:K}$. With sequential sampling, $m = 100$ sample trajectories are created. In the end, we compare solutions from the two algorithms both with each other and with the almost-exact trajectory of withheld observations $\mathbf{y}_{1:K}$. Observations are produced under the same conditions as in Section 4.1.1.

Results Figures 4.2 to 4.5 depict again the results for ODEs Lotka-Volterra, van der Pol, Lorenz'63, and linearly coupled anharmonic oscillators, respectively. There, Figs. 4.2a to 4.5a arise from EKF-based inference and Figs. 4.2b to 4.5b are produced using sequen-

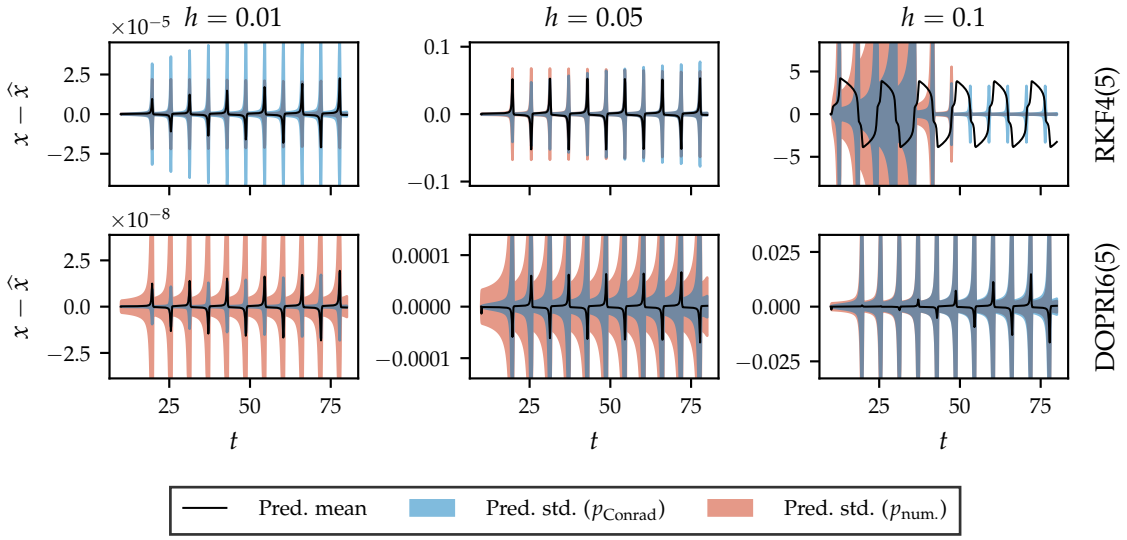


(a) EKF.

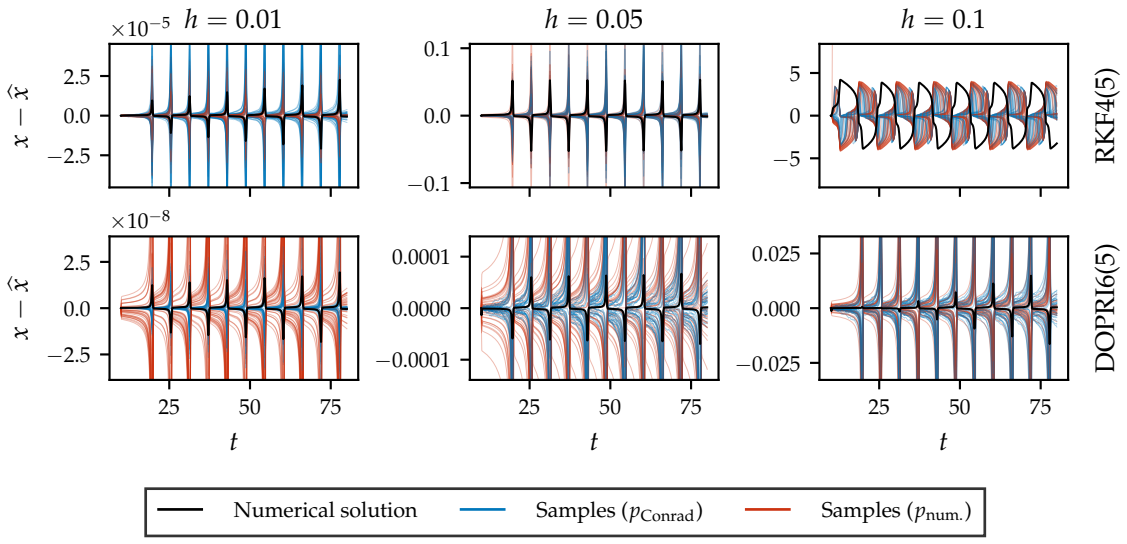


(b) Sequential sampling.

Figure 4.2: Qualitative evaluation of uncertainty calibration for the Lotka-Volterra ODE and transition models $p_{\text{num.}}$ and p_{Conrad}. Plots show the residuals between predicted and true solutions of the first state component for the EKF (4.2a) and sequential sampling (4.2b) over different numerical solvers (rows) and step sizes h (columns). Single standard deviations (EKF) are indicated by zero-centered error bands, sample trajectories (sequential sampling) are plotted in difference to the unperturbed numerical solution, enabling direct comparison with the former.

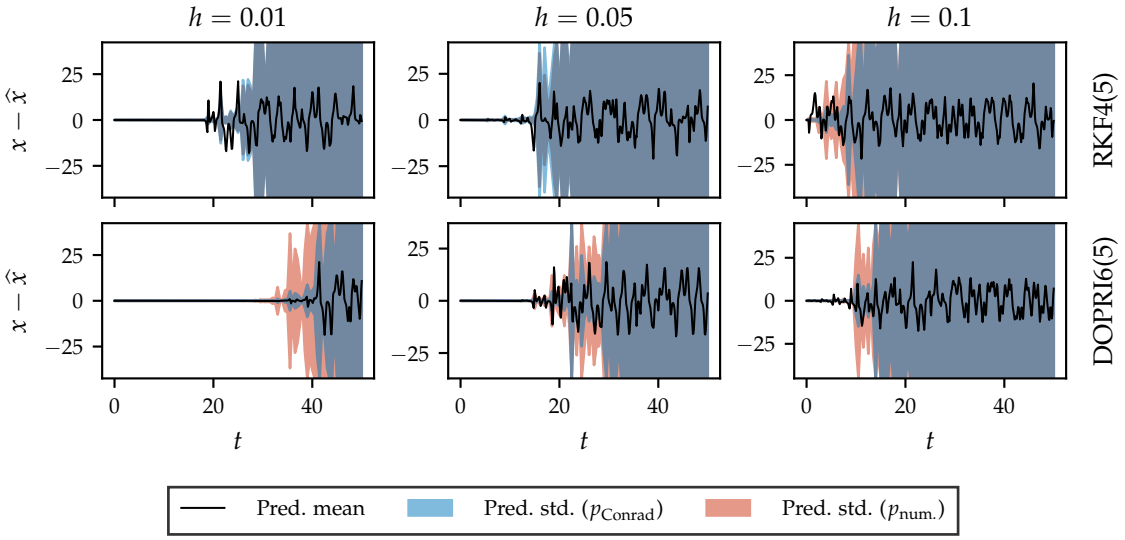


(a) EKF.

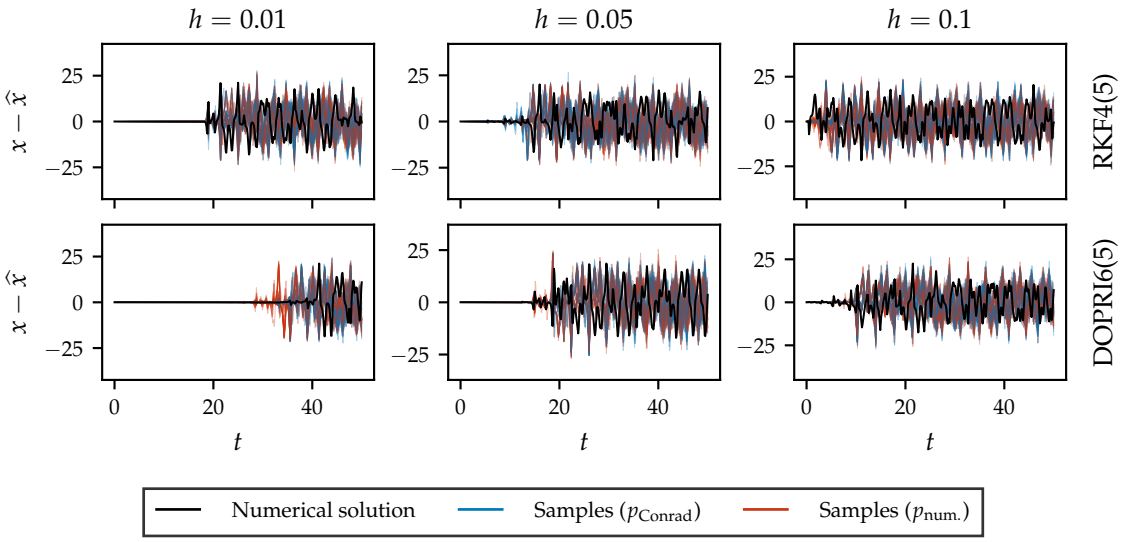


(b) Sequential sampling.

Figure 4.3: Qualitative evaluation of uncertainty calibration for the van der Pol ODE and transition models $p_{\text{num.}}$ and p_{Conrad}. Plots show the residuals between predicted and true solutions of the first state component for the EKF (4.2a) and sequential sampling (4.2b) over different numerical solvers (rows) and step sizes h (columns). Single standard deviations (EKF) are indicated by zero-centered error bands, sample trajectories (sequential sampling) are plotted in difference to the unperturbed numerical solution, enabling direct comparison with the former.

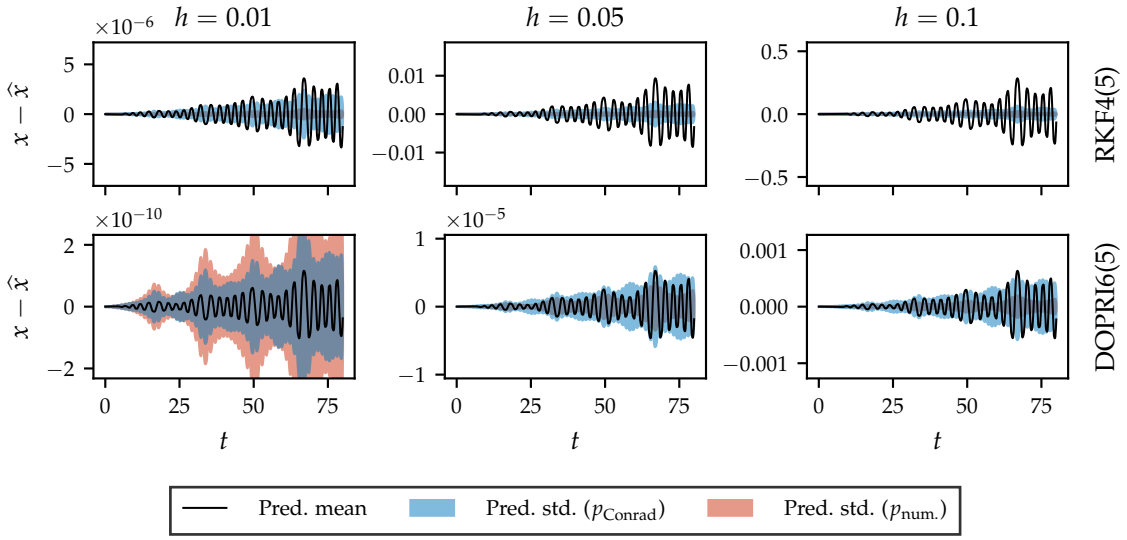


(a) EKF.

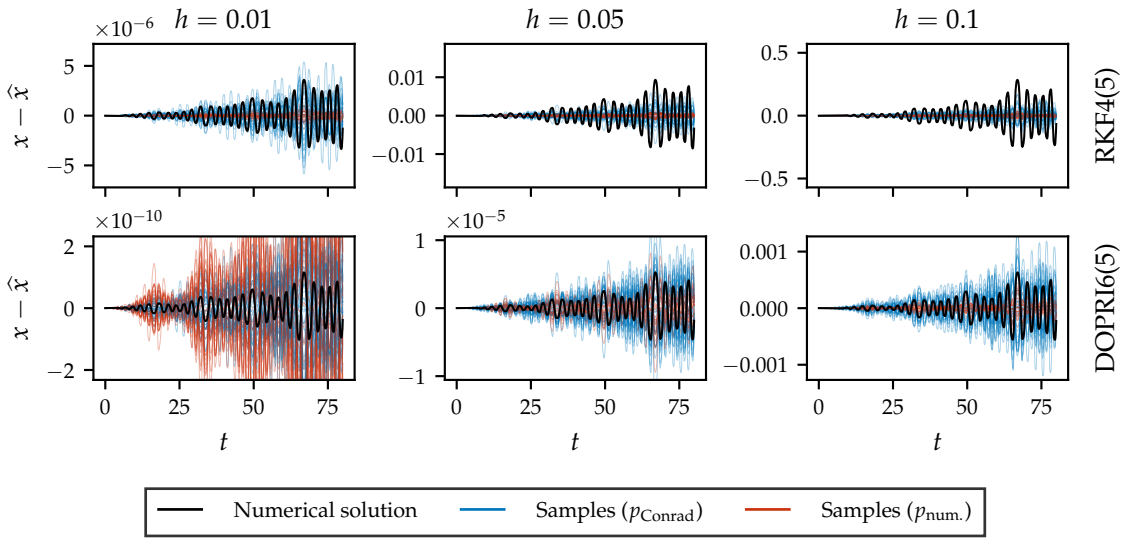


(b) Sequential sampling.

Figure 4.4: Qualitative evaluation of uncertainty calibration for the Lorenz'63 ODE and transition models $p_{\text{num.}}$ and p_{Conrad}. Plots show the residuals between predicted and true solutions of the first state component for the EKF (4.2a) and sequential sampling (4.2b) over different numerical solvers (rows) and step sizes h (columns). Single standard deviations (EKF) are indicated by zero-centered error bands, sample trajectories (sequential sampling) are plotted in difference to the unperturbed numerical solution, enabling direct comparison with the former.



(a) EKF.



(b) Sequential sampling.

Figure 4.5: Qualitative evaluation of uncertainty calibration for the linearly coupled anharmonic oscillator ODE and transition models $p_{\text{num.}}$ and p_{Conrad}. Plots show the residuals between predicted and true solutions of the first state component for the EKF (4.2a) and sequential sampling (4.2b) over different numerical solvers (rows) and step sizes h (columns). Single standard deviations (EKF) are indicated by zero-centered error bands, sample trajectories (sequential sampling) are plotted in difference to the unperturbed numerical solution, enabling direct comparison with the former.

tial sampling. Due to mostly invisibly small errors, in particular for the smallest tested step size $h = 0.01$, the residuals between the predicted and true solutions are displayed.

As indicated by the results in Section 4.1.1, there are many cases where manual calibration of the model p_{Conrad} leads to better calibrated results than the automatic calibration of $p_{\text{num.}}$ using the local error estimator. For example, using $p_{\text{num.}}$, the uncertainty to solutions of ODEs Lotka-Volterra and the linearly coupled anharmonic oscillators is slightly underestimated with RKF4(5) (cf. Fig. 4.2), while substantially overestimated for the van der Pol ODE and DOPRI6(5) (cf. Fig. 4.3). Compared to RKF4(5), DOPRI6(5) generally induces a larger uncertainty, which could be expected due to the lower-order characteristic of the error estimator. Interestingly, this seems to compensate for the otherwise too weak signal of the local error estimator for the Lotka-Volterra ODE and larger step sizes, which is then better calibrated than with p_{Conrad}. Something similar happens for the lineary coupled anharmonic oscillators (cf. Fig. 4.5). For smaller step sizes and RKF4(5), probabilistic solutions to the van der Pol ODE are also slightly better calibrated when using $p_{\text{num.}}$, compared to p_{Conrad}.

Hence, the overall results are mixed. While there are cases where results produced by $p_{\text{num.}}$ are better calibrated, the majority of tested situations are in favor of p_{Conrad}. Still, the probabilistic solutions produced by $p_{\text{num.}}$ are in general well-structured and match the occurred numerical errors on a qualitative level, i.e., the predicted uncertainty is larger at points of large errors and smaller at points of small errors.

Last, Gaussian solutions computed with the EKF are overall close to the true, in general non-Gaussian stochastic process that is approximated by sequential sampling. Two exceptions are the van der Pol ODE for RKF4(5) and the largest tested step size $h = 0.1$ (cf. Fig. 4.3), as well as the Lorenz'63 system for all solvers and step sizes (cf. Fig. 4.4). The former is clearly caused by too large global errors, while the latter can be attributed to the chaotic characteristics of the Lorenz'63 system. Both exhibit non-symmetric sample trajectories, which can not be captured well by the Gaussian approximation of the EKF. While the uncertainty here simply explodes over the remaining time horizon for Lorenz'63, it does so first too in the case of van der Pol, but then becomes minimal, losing any structure. Interestingly, this effect is less prominent with the manually calibrated model p_{Conrad}, although still observable.

4.1.3 | Experiment 1.3: Effects of Overly Coarse Time Discretizations

The final experiment of this section investigates the effect of overly large step sizes on probabilistic solutions created with this work's model $p_{\text{num.}}$. In Section 4.1.2, we noted that the step size $h = 0.1$ specified above is too small for the RKF4(5) method to

solve the IVP specified for the van der Pol ODE. In order to inspect in depth how the probabilistic solutions are affected by such misconfiguration, we compare probabilistic state predictions $\widehat{\mathbf{x}}_k$ visually at specific points in time t_k. At these times, the Gaussian approximation $\mathcal{N}\left(\widehat{\mathbf{x}}_k; \widehat{\mathbf{m}}_k, \widehat{\mathbf{P}}_k\right)$ provided by the EKF is contrasted with a kernel density estimation (KDE) of the sample states $\tilde{\mathbf{x}}_k^{(1)}, \ldots, \tilde{\mathbf{x}}_k^{(m)}$ obtained in sequential sampling, and the Dirac measure of the exact state $\mathbf{x}_k$, being approximated by the numerically computed observation $\mathbf{y}_k$. For the KDE, we use a Gaussian kernel and a diagonal smoothing matrix $\frac{1}{10}\mathbf{I}$, i.e., we approximate the probability density function $p(\widehat{\mathbf{x}}_k)$ that underlies samples $\tilde{\mathbf{x}}_k^{(1)}, \ldots, \tilde{\mathbf{x}}_k^{(m)}$ by an equally-weighted mixture of Gaussians

$$p\left(\widehat{\mathbf{x}}_k\right) \approx \frac{1}{m} \sum_{i=1}^{m} \mathcal{N}\left(\widehat{\mathbf{x}}_k; \tilde{\mathbf{x}}_k^{(i)}, \frac{1}{10}\mathbf{I}\right).$$

In this experiment, we consider the IVP on the van der Pol ODE being specified in Section 4.1.2. Moreover, we compare probabilistic solutions created with RKF4(5) and $h = 0.1$ as well as DOPRI6(5) and $h = 0.15$, respectively. These probabilistic solutions are then inspected as described above at times $t = 10.3$, $t = 12.5$, $t = 48.7$, and $t = 55.0$.

Results The results of this experiment are displayed in Fig. 4.6. At the start of simulations ($t = 10.3$), both RKF4(5) and DOPRI6(5) give rise to sharp distributions around the exact solution. Only two time units later, however, the Gaussian of the EKF is overly wide and the KDE of the sample trajectories exhibits a bimodal structure for RKF4(5). However, the results are unchanged with DOPRI6(5). At $t = 48.7$, the KDEs of both simulations show similar bimodal distributions, although DOPRI6(5) leads to more probability mass in between the two modes, where the exact solution is located. On the contrary, the Gaussian distributions produced by the EKF are collapsed at one mode for RKF4(5) and roughly correcly located and scaled for DOPRI6(5). Last, at $t = 55.0$, the unchanged bimodal KDEs of RKF4(5) captures the exact solution at one mode, but the Gaussian approximation is collapsed at the other, wrong mode. With DOPRI6(5), by contrast, unimodality around the exact solution is restored both for solutions provided by the EKF and sequential sampling.

In summary, the results indicate that overly coarse time discretizations in cases of multimodality can lead to failure modes such as mode collapses, which may not be recoverable in subsequent prediction steps.

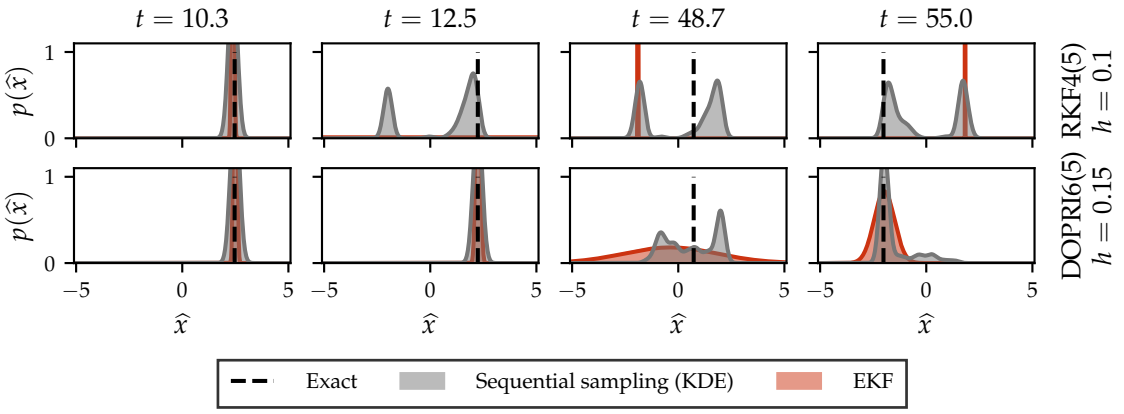


Figure 4.6: Effects of overly coarse time discretizations using the example of the van der Pol ODE. Plots show the estimated probability density function of the predicted state's first component for the EKF and sequential sampling as well as for different numerical ODE solvers (rows) at different points in time (rows). The distribution for sequential sampling is computed from samples using kernel density estimation with a Gaussian kernel.

4.2 | ODE Parameter Estimation

In the context of ODE parameter estimation, the following three experiments are carried out:

Experiment 2.1. Influence of the error estimator and tempering schedule on process noise tempering, addressing Q2.1.

Experiment 2.2. Applying process noise tempering to the Hodgkin-Huxley model, addressing Q2.2.

Experiment 2.3. Improving process noise tempering using prior knowledge, addressing Q2.2.

4.2.1 | Experiment 2.1: Influence of the Error Estimator and Tempering Schedule on Process Noise Tempering

In the first experiment on ODE parameter estimation, reliability and efficiency are compared between the two transition models p_{PNT} and $p_{\text{PNT+num.}}$ in conjunction with different tempering schedules Γ for PNT. As a baseline, we also perform NLSR.

Definition 4.1. Let $\hat{\theta}, \theta^* \in \mathbb{R}^W$ denote estimated and true parameters of an ODE, respectively. Then, the *relative parameter root mean square error (rpRMSE)* between $\hat{\theta}$ and θ^* is

defined as

$$\text{rpRMSE} := \sqrt{\frac{1}{W}\sum_{i=1}^{W}\left\|\frac{[\hat{\theta}]_i - [\theta^*]_i}{[\hat{\theta}]_i}\right\|_2^2}. \tag{4.2}$$

Furthermore,
time t_k, arising from the estimated and true parameters $\hat{\theta}$ and θ^*, respectively. Then, the *trajectory root mean square error (tRMSE)* between trajectories of states $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_k^*$ is defined as

$$\text{tRMSE} := \sqrt{\frac{1}{K}\sum_{k=1}^{K}\|\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^*\|_2^2}. \tag{4.3}$$

In general, the experimental setup closely follows Beck et al. (2024). The reliability is measured in terms of the *convergence success rate*. It is defined as the fraction of differently initialized optimization runs producing an estimate $\hat{\theta}$ whose rpRMSE (cf. Eq. (4.2)) is below 0.05. Additionally, we also consider the deviation between state trajectories that originate from different parametrizations, measured by the tRMSE according to Eq. (4.3). The efficiency, on the other hand, is rated by the number of iterations the optimization algorithm needs until convergence. In case of PNT, the total number of optimization iterations accumulated over tempering iterations is considered. We use the algorithm *limited-memory Broyden-Fletcher-Goldfarb-Shanno with bounds (L-BFGS-B)* (Byrd et al., 1995; Zhu et al., 1997) for gradient-based optimization with box constraints on individual parameters. In the following, the considered parameter estimation problems and used tempering schedules Γ are specified.

Parameter estimation problems. We perform parameter estimation for two problems, both based on the Lotka-Volterra equations given by Eqs. (4.1a) and (4.1b), and use the identical parametrization, time interval, and initial values as in Section 4.1.1. For the first problem *LV-2*, we assume that only the subset of parameters θ_1, θ_2 is unknown and the remaining θ_3, θ_4 are known. The second problem *LV-4* then considers all four parameters to be unknown. During parameter estimation, all parameters are bounded by the interval $[1 \times 10^{-3}, 5.0]$.

In both problems, measurements of the trajectory corresponding to the true parameters are only partly observed through the first component of the state, referring to the prey population. Moreover, they are subject to zero-mean Gaussian noise of variance 0.1, i.e., $\mathbf{H} = \begin{bmatrix}1 & 0\end{bmatrix}$ and $\mathbf{R} = \begin{bmatrix}0.1\end{bmatrix}$. Observations are again created using DOPRI6(5) and a step size of $h = 1 \times 10^{-4}$. Simulations being run in the context of parameter estimation to

Table 4.1: Comparison of different Lotka-Volterra-type problems and methods for parameter estimation. CONV: Convergence success rate. OptIter: Total number of iterations needed by the optimization algorithm.

PROB.	METHOD	RPRMSE ↓	CONV ↑	TRMSE ↓	OPTITER ↓
LV-2	NLSR	1.43 ± 1.10	0.20	2.49 ± 1.25	**11.37 ± 4.09**
LV-2	FENRIR+DT	0.35 ± 0.77	0.79	0.43 ± 1.03	132.43 ± 31.01
LV-2	PNT ($\Gamma_{\text{high+coarse}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	44.61 ± 5.01
LV-2	PNT ($\Gamma_{\text{high+coarse}}$ + $\widehat{\delta \mathbf{x}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	44.61 ± 5.01
LV-2	PNT ($\Gamma_{\text{low+coarse}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	33.45 ± 1.82
LV-2	PNT ($\Gamma_{\text{low+coarse}}$ + $\widehat{\delta \mathbf{x}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	33.45 ± 1.82
LV-2	PNT ($\Gamma_{\text{high+fine}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	84.61 ± 5.01
LV-2	PNT ($\Gamma_{\text{high+fine}}$ + $\widehat{\delta \mathbf{x}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	84.61 ± 5.01
LV-2	PNT ($\Gamma_{\text{low+fine}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	54.45 ± 1.82
LV-2	PNT ($\Gamma_{\text{low+fine}}$ + $\widehat{\delta \mathbf{x}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	54.45 ± 1.82
LV-4	NLSR	0.95 ± 0.58	0.17	13.30 ± 56.91	42.35 ± 23.90
LV-4	FENRIR+DT	0.74 ± 0.82	0.44	13.77 ± 79.08	292.05 ± 139.98
LV-4	PNT ($\Gamma_{\text{high+coarse}}$)	0.77 ± 0.32	0.15	2.64 ± 1.10	**33.03 ± 29.27**
LV-4	PNT ($\Gamma_{\text{high+coarse}}$ + $\widehat{\delta \mathbf{x}}$)	0.77 ± 0.32	0.15	2.64 ± 1.10	**33.02 ± 29.27**
LV-4	PNT ($\Gamma_{\text{low+coarse}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	83.45 ± 7.53
LV-4	PNT ($\Gamma_{\text{low+coarse}}$ + $\widehat{\delta \mathbf{x}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	83.45 ± 7.53
LV-4	PNT ($\Gamma_{\text{high+fine}}$)	0.02 ± 0.09	0.99	0.06 ± 0.31	169.31 ± 15.97
LV-4	PNT ($\Gamma_{\text{high+fine}}$ + $\widehat{\delta \mathbf{x}}$)	0.02 ± 0.09	0.99	0.06 ± 0.31	169.31 ± 16.00
LV-4	PNT ($\Gamma_{\text{low+fine}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	146.43 ± 7.54
LV-4	PNT ($\Gamma_{\text{low+fine}}$ + $\widehat{\delta \mathbf{x}}$)	**0.01 ± 0.00**	**1.00**	**0.03 ± 0.00**	146.41 ± 7.58

compute the NLL, by contrast, use RKF4(5) and a step size of $h = 0.01$, since the method's local error estimator is then interpretable as such.

Tempering schedules. We consider four different tempering schedules, defined by the following relations:

$$\begin{aligned}
\Gamma_{\text{high+coarse}} &:= \left\{(1, 10.0), (2, 1 \times 10^{-2}), (3, 1 \times 10^{-5}), (4, 1 \times 10^{-8}), (5, 0.0)\right\}, \\
\Gamma_{\text{low+coarse}} &:= \left\{(1, 1 \times 10^{-2}), (2, 1 \times 10^{-5}), (3, 1 \times 10^{-8}), (4, 0.0)\right\}, \\
\Gamma_{\text{high+fine}} &:= \left\{(1, 10.0), (2, 1.0), (3, 1 \times 10^{-1}), \ldots, (10, 1 \times 10^{-8}), (11, 0.0)\right\}, \\
\Gamma_{\text{low+fine}} &:= \left\{(1, 1 \times 10^{-2}), (2, 1 \times 10^{-3}), (3, 1 \times 10^{-4}), \ldots, (7, 1 \times 10^{-8}), (8, 0.0)\right\}.
\end{aligned}$$

When applying one of these, the number of tempering iterations m is automatically determined by $|\Gamma|$.

For each possible combination between transition models (p_{PNT}, $p_{\text{PNT+num.}}$), estimation problems (LV-2, LV-4), and tempering schedules ($\Gamma_{\text{high+coarse}}$, $\Gamma_{\text{low+coarse}}$, $\Gamma_{\text{high+fine}}$,

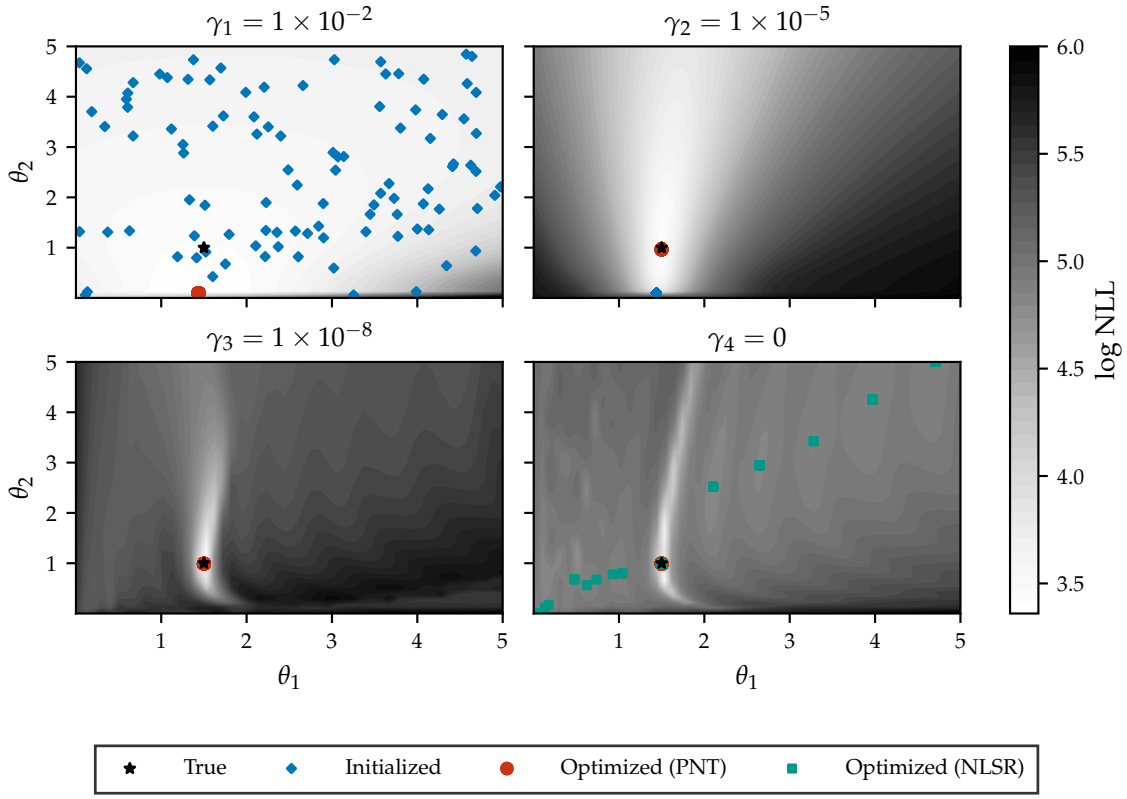


Figure 4.7: LV-2 parameter estimation using PNT and NLSR. Plots show the NLL across the two-dimensional space of optimized parameters in different tempering iterations. PNT uses the optimization results of a tempering iteration as initial points for optimization in the next tempering iteration. NLSR performs optimization only on the zero-noise NLL function.

$\Gamma_{\text{low+fine}}$), 100 differently initialized runs of the PNT algorithm are performed. In addition, 100 differently initialized runs are performed for every estimation problem, applying NLSR. All these initializations are random but kept equal across different transition models, tempering schedules as well as between PNT and NLSR. Besides NLSR, we also compare our results to the method *Fenrir with diffusion tempering (DT)* by Beck et al. (2024), which was the inspiration for PNT. However, we do not repeat their experiments but simply compare to the results reported in the corresponding publication.

Results Table 4.1 lists the results obtained in the estimation of parameters for the Lotka-Volterra ODE using different algorithms. Regarding problem LV-2, PNT reaches 100 percent convergence success rate and therefore surpasses both NLSR and Fenrir with DT in terms of the correctness of estimated parameters. Moreover, the latter is also outperformed with regard to the number of L-BFGS-B iterations needed until convergence.

While NLSR requires the lowest number of iterations here, its efficiency can still be valuated worse than that of PNT, considering the convergence success rate of only 20 percent. For PNT, differences across different tempering schedules are only observable in the number of optimization steps, where $\Gamma_{\text{low+coarse}}$ with the smallest number of tempering iterations performs best, as expected. The addition of local error estimates, i.e., the usage of the transition model $p_{\text{PNT+num.}}$ instead of p_{PNT}, makes no difference at all. Figure 4.7 exemplarily illustrates the change of the NLL across tempering iterations and how NLSR and PNT with $\Gamma_{\text{low+coarse}}$ converge to different minima.

Concerning the second problem LV-4, PNT is still able to achieve the maximum convergence success rate, but not with an arbitrary choice of the tempering schedule Γ. In fact, only tempering schedules that start with a lower initial process noise temperature γ attain 100 percent. With a more fine-grained temperature scheduling, however, still 99 out of 100 randomly initialized runs converge to the global minimum. This is opposite to the coarse-grained case, which only performs on a level comparable to NLSR. Last, there is again no significant difference when adding the local error estimator.

4.2.2 | Experiment 2.2: Applying Process Noise Tempering to the Hodgkin-Huxley Model

The next experiment investigates how reliable PNT can estimate parameters correctly for increasingly complex and highly parametrized ODEs. Here, the method is applied to the *Hodgkin-Huxley model* (Hodgkin & Huxley, 1952), which describes the initiation and propagation of action potentials in neurons by the dynamics of an electrical circuit model with capacitive elements. Following Beck et al. (2024), we use the concrete model formulation of Pospischil et al. (2008), defining a system of eight first-order ODEs with 15 parameters. It contains several exponential and polynomial terms and therefore exhibits highly non-linear behavior. For the details of the model, see Appendix B.1.

As described in Appendix B.2, the Hodgkin-Huxley model can also be refined to a so-called *multi-compartment model*. In essence, the Hodgkin-Huxley model is then replicated there several times and these individual submodels are coupled to form a larger model that approximates the characteristics of a neuron more accurately. In the following, the single-compartment model and the multi-compartment model with c compartments are referred to by *HH* and *HH-Cc*, respectively. Here, we consider only cases where the same parameters are to be estimated in all compartments.

Similar to Section 4.2.1, we perform parameter estimation for varying numbers of known and unknown parameters of the model and measure the rpRMSE and tRMSE afterwards. Since individual parameters only contribute to specific components of the

Table 4.2: Parameter estimation problems for the Hodgkin-Huxley model and corresponding parameters to be estimated. As before, the last number in a problem's name indicates the number of optimized parameters. In multi-compartment models, the same parameter types are estimated in every compartment. See Appendix B.1 for an explanation of the parameters.

PROBLEM	ESTIMATED PARAMETERS
HH-R4-1	g_{Na}
HH-R4-2	$g_{\mathrm{Na}}, g_{\mathrm{K}}$
HH-R1-6	$g_{\mathrm{Na}}, g_{\mathrm{K}}, g_{\mathrm{leak}}, g_{\mathrm{M}}, g_{\mathrm{L}}, V_T$
HH-7	$g_{\mathrm{Na}}, g_{\mathrm{K}}, g_{\mathrm{leak}}, g_{\mathrm{M}}, g_{\mathrm{L}}, g_{\mathrm{T}}, V_T$
HH-R1-9	$g_{\mathrm{Na}}, g_{\mathrm{K}}, g_{\mathrm{leak}}, g_{\mathrm{M}}, g_{\mathrm{L}}, E_{\mathrm{Na}}, E_{\mathrm{K}}, E_{\mathrm{leak}}, V_T$
HH-11	$g_{\mathrm{Na}}, g_{\mathrm{K}}, g_{\mathrm{leak}}, g_{\mathrm{M}}, g_{\mathrm{L}}, g_{\mathrm{T}}, E_{\mathrm{Na}}, E_{\mathrm{K}}, E_{\mathrm{leak}}, E_{\mathrm{Ca}}, V_T$
HH-C2-R4-4	$g_{\mathrm{Na}}, g_{\mathrm{K}}$
HH-C2-R4-6	$g_{\mathrm{Na}}, g_{\mathrm{K}}, g_{\mathrm{leak}}$
HH-C2-R1-12	$g_{\mathrm{Na}}, g_{\mathrm{K}}, g_{\mathrm{leak}}, g_{\mathrm{M}}, g_{\mathrm{L}}, V_T$

state, we omit simulation of certain components of the state for which all parameters are considered known in order to reduce the computational load, as done by Beck et al. (2024). A therefore simplified Hodgkin-Huxley model that disregards the last r components of the state is subsequently denoted by *HH-Rr*.

As parameters in the Hodgkin-Huxley model reside on substantially different scales, leading to potentially ill-conditioned optimization problems, they are normalized before optimization. With $\theta_{\mathrm{lb}}, \theta_{\mathrm{ub}} \in \mathbb{R}^W$ denoting the lower and upper bounds on parameters θ, these are transformed to the $[0,1]^W$ hypercube by

$$\frac{\theta - \theta_{\mathrm{lb}}}{\theta_{\mathrm{ub}} - \theta_{\mathrm{lb}}}.$$

Parameter estimation problems. We perform parameter estimation for nine problems based on the Hodgkin-Huxley model, being comprised in Table 4.2. Both the true parameters and parameter ranges are listed in Tables B.1 and B.2 of the appendix. These as well as the considered time interval $\mathbb{T} = [0, 100]$ ms and the initial value are identical for each problem. The initial value of the first state component corresponding to the voltage is set to -70 mV, while the remaining components are computed as described in Appendix B.1. Numerical integration is performed using the implicit *Kvaerno3(2)* method (Kværnø, 2004) due to the stiffness of the ODE. As in Section 4.2.1, simulations during parameter estimation are conducted with the step size $h = 0.01$, while measurements are produced with a reduced step size of $h = 1 \times 10^{-4}$. The latter are only partly observed through the voltage and also perturbed by zero-mean Gaussian noise

Table 4.3: Comparison of different Hodgkin-Huxley-type problems and methods for parameter estimation. CONV: Convergence success rate. OptIter: Total number of iterations needed by the optimization algorithm.

PROBLEM	METHOD	RPRMSE ↓	CONV ↑	TRMSE ↓	OPTITER ↓
HH-R4-1	NLSR	0.77 ± 0.40	0.21	13.47 ± 6.92	**1.81 ± 1.61**
HH-R4-1	FENRIR+DT	**0.00 ± 0.00**	**1.00**	0.43 ± 0.02	382.08 ± 32.19
HH-R4-1	PNT	**0.00 ± 0.00**	**1.00**	**0.11 ± 0.00**	21.25 ± 1.62
HH-R4-2	NLSR	1.00 ± 0.20	0.00	13.41 ± 7.60	**4.15 ± 5.50**
HH-R4-2	FENRIR+DT	**0.00 ± 0.00**	**1.00**	0.42 ± 0.04	696.22 ± 78.10
HH-R4-2	PNT	**0.00 ± 0.00**	**1.00**	**0.12 ± 0.00**	30.86 ± 4.38
HH-R1-6	NLSR	13.91 ± 9.66	0.09	13.96 ± 5.08	**66.85 ± 106.72**
HH-R1-6	FENRIR+DT	10.36 ± 7.72	0.00	15.20 ± 5.41	2159.60 ± 532.55
HH-R1-6	PNT	**0.28 ± 2.41**	**0.99**	**0.27 ± 1.53**	130.23 ± 19.88
HH-7	NLSR	22.96 ± 8.41	0.06	13.81 ± 4.88	**78.07 ± 119.76**
HH-7	PNT	**0.02 ± 0.0**	**1.00**	**0.12 ± 0.00**	137.24 ± 40.17
HH-R1-9	NLSR	16.96 ± 7.29	0.00	13.03 ± 5.61	**124.16 ± 143.04**
HH-R1-9	PNT	**0.54 ± 3.62**	**0.98**	**0.41 ± 2.06**	260.19 ± 47.73
HH-11	NLSR	21.68 ± 7.13	0.00	13.35 ± 4.61	**123.95 ± 129.96**
HH-11	PNT	**0.57 ± 3.14**	**0.01**	**0.43 ± 2.15**	265.80 ± 53.00
HH-C2-R4-4	NLSR	0.70 ± 0.40	0.24	24.47 ± 13.73	**21.11 ± 24.05**
HH-C2-R4-4	FENRIR+DT	**0.00 ± 0.00**	**1.00**	0.60 ± 0.01	1492.03 ± 335.17
HH-C2-R4-4	PNT	**0.00 ± 0.00**	**1.00**	**0.24 ± 0.00**	72.01 ± 7.66
HH-C2-R4-6	NLSR	1.38 ± 0.51	0.11	26.11 ± 9.64	**61.95 ± 51.25**
HH-C2-R4-6	FENRIR+DT	0.12 ± 0.32	0.88	3.01 ± 6.70	1525.57 ± 448.56
HH-C2-R4-6	PNT	**0.00 ± 0.00**	**1.00**	**0.24 ± 0.00**	123.92 ± 18.67
HH-C2-R1-12	NLSR	10.58 ± 6.37	0.00	27.30 ± 9.72	**160.83 ± 148.74**
HH-C2-R1-12	PNT	**0.80 ± 4.13**	**0.94**	**2.33 ± 8.30**	412.92 ± 85.17

of variance 0.1, i.e., $\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ and $\mathbf{R} = \begin{bmatrix} 0.1 \end{bmatrix}$.[1] For all problems, we perform PNT with the transition model p_{PNT} and the tempering schedule

$$\Gamma = \left\{(1, 1 \times 10^{-2}), (2, 1 \times 10^{-5}), (3, 1 \times 10^{-8}), (4, 0.0)\right\},$$

which achieved the best results in experiment 2.1. Furthermore, as in experiment 2.1, we compare to a baseline provided by NLSR. Finally, both algorithms are run with the same 100 random initializations of the parameters. The described setting is again comparable to Beck et al. (2024), whose results are also taken from their manuscript for comparison.

[1] For reduced models *HH-Rr* and multi-compartment models *HH-Cc*, the observation matrix $\mathbf{H}$ is accordingly reduced or extended (cf. Appendix B.2), respectively.

Results The measurements taken in this experiment are summarized in Table 4.3. Overall, PNT surpasses both Fenrir with DT and NLSR in every considered problem, often by a wide margin. For the less complex problems HH-R4-1, HH-R4-2, and HH-C2-R4-4, PNT achieves 100 percent convergence success rate just like the method of Beck et al. (2024), but reaches lower values with regard to tRMSE and the needed number of optimization steps. NLSR, by comparison, is below 25 percent. Figures 4.8a and 4.8b show the change of the NLL as well as the results of optimization across tempering iterations for problems HH-R4-1 and HH-R4-2, respectively.

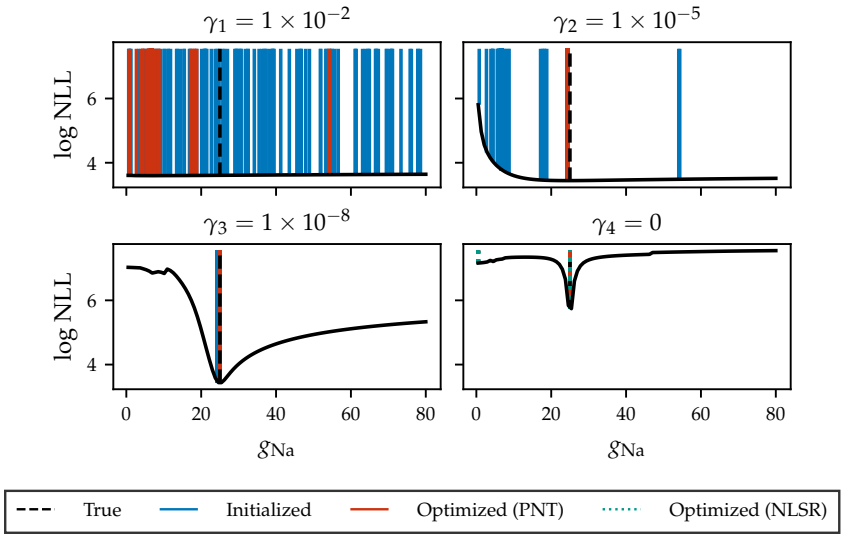
In the more complex problems HH-R1-6 and HH-C2-R4-6, PNT still reaches the maximum convergence success rate, while Fenrir with DT is only able to converge to the global minimum for the multi-compartment problem. When increasing the complexity of the model and the number of parameters to be optimized even further, PNT shows the ability to estimate parameters with high reliability up to problems HH-R1-9 in the single-compartment case and HH-C2-R1-12 for two compartments. Concerning problem HH-11, the convergence success rate is only 1 percent, but the small values of the rpRMSE and tRMSE, being only slightly worse than for problem HH-R1-9, indicate that parameter estimates are still close to optimum.

Although efficiency was not the focus of this experiment, as opposed to accuracy of estimations, PNT needs considerably fewer iterations of the optimization algorithm than Fenrir with DT. As in experiment 2.1, NLSR performs best in this regard, but does not estimate the correct parameter values reliably and might require impracticably many restarts.

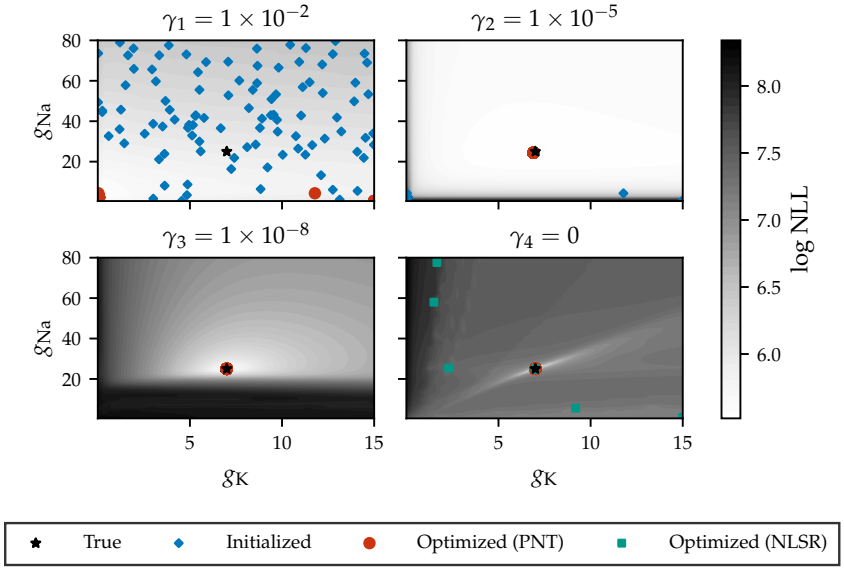
4.2.3 | Experiment 2.3: Improving Process Noise Tempering using Prior Knowledge

In the last experiment of this work, we attempt to improve the process noise tempering algorithm and the results shown in the previous section by the use of prior knowledge. In the basic form that was presented in Section 3.2.2, the transition model p_{PNT} constructs an isotropic Gaussian, implicitly assuming that the uncertainty about parameters has the same impact on the uncertainty of every component of the state. This assumption, however, might be violated in practice depending on the functional form of the ODE whose parameters are to be estimated.

Here, we will alter the covariance matrix of the transition model p_{PNT} by the use of prior knowledge about the ODE of interest. More precisely, we exploit the simple binary criterion of whether the time derivatives of state components, as provided by the ODE, depend on at least one parameter that should be estimated. For components of the state



(a) HH-R4-1.



(b) HH-R4-2.

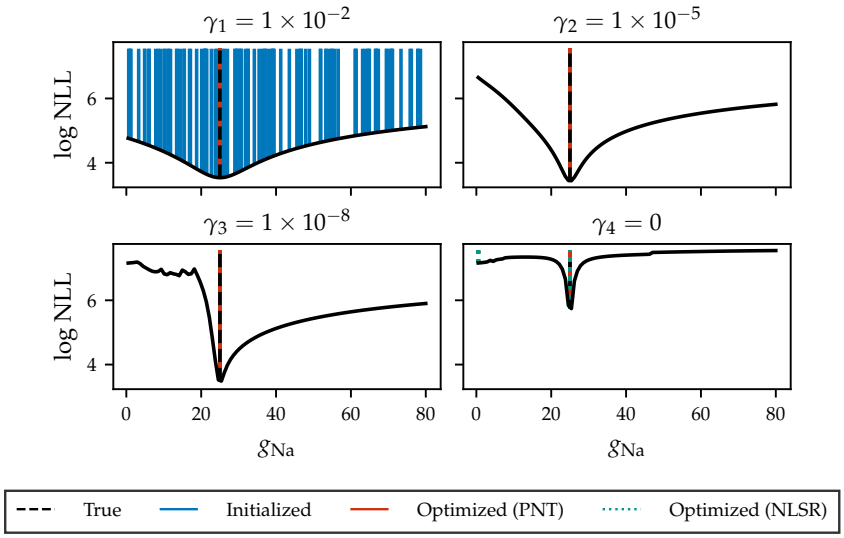
Figure 4.8: Parameter estimation on problems HH-R4-1 (4.8a) and HH-R4-2 (4.8b) using PNT and NLSR. Plots show the NLL across the one and two-dimensional space of optimized parameters in different tempering iterations, respectively. PNT uses the optimization results of a tempering iteration as initial points for optimization in the next tempering iteration. NLSR performs optimization only on the zero-noise NLL function.

Table 4.4: Comparison of different Hodgkin-Huxley-type problems and the PNT algorithm with (PNT+)/without (PNT) the heuristic on parameter dependence. CONV: Convergence success rate. OptIter: Total number of iterations needed by the optimization algorithm.

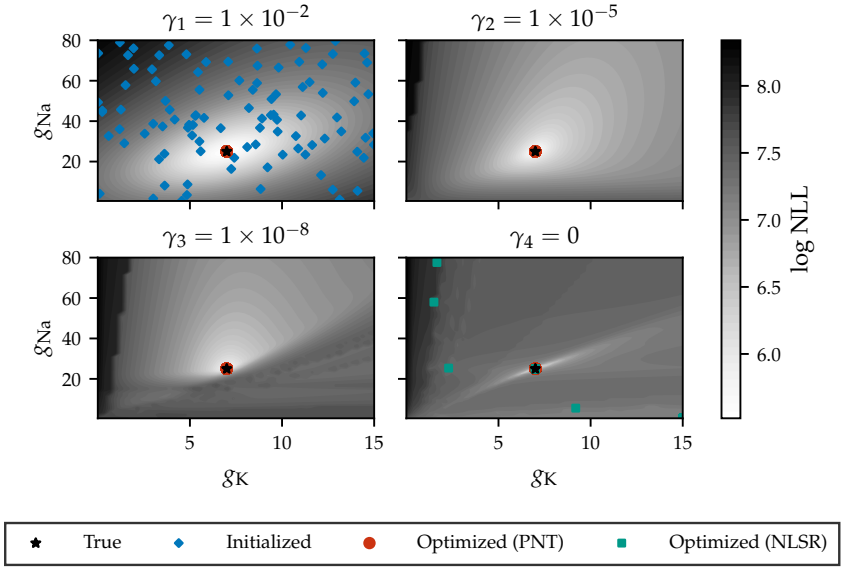
PROBLEM	METHOD	RPRMSE ↓	CONV ↑	TRMSE ↓	OPTITER ↓
HH-R4-1	PNT	**0.00 ± 0.00**	**1.00**	**0.11 ± 0.00**	21.25 ± 1.62
HH-R4-1	PNT+	**0.00 ± 0.00**	**1.00**	**0.11 ± 0.00**	**13.17 ± 0.70**
HH-R4-2	PNT	**0.00 ± 0.00**	**1.00**	**0.12 ± 0.00**	30.86 ± 4.38
HH-R4-2	PNT+	**0.00 ± 0.00**	**1.00**	**0.12 ± 0.00**	**22.21 ± 1.13**
HH-R1-6	PNT	0.28 ± 2.41	0.99	0.27 ± 1.53	**130.23 ± 19.88**
HH-R1-6	PNT+	**0.04 ± 0.00**	**1.00**	**0.12 ± 0.00**	160.88 ± 21.99
HH-7	PNT	0.02 ± 0.0	**1.00**	**0.12 ± 0.00**	**137.24 ± 40.17**
HH-7	PNT+	**0.01 ± 0.0**	**1.00**	**0.12 ± 0.00**	163.04 ± 28.27
HH-R1-9	PNT	0.54 ± 3.62	0.98	0.41 ± 2.06	**260.19 ± 47.73**
HH-R1-9	PNT+	**0.03 ± 0.00**	**1.00**	**0.12 ± 0.00**	314.68 ± 53.98
HH-11	PNT	0.57 ± 3.14	**0.01**	0.43 ± 2.15	**265.80 ± 53.00**
HH-11	PNT+	**0.28 ± 1.78**	0.00	**0.26 ± 1.33**	349.04 ± 57.17
HH-C2-R4-4	PNT	**0.00 ± 0.00**	**1.00**	**0.24 ± 0.00**	72.01 ± 7.66
HH-C2-R4-4	PNT+	**0.00 ± 0.00**	**1.00**	**0.24 ± 0.00**	**42.31 ± 1.12**
HH-C2-R4-6	PNT	**0.00 ± 0.00**	**1.00**	**0.24 ± 0.00**	123.92 ± 18.67
HH-C2-R4-6	PNT+	**0.00 ± 0.00**	**1.00**	**0.24 ± 0.00**	**96.30 ± 2.81**
HH-C2-R1-12	PNT	**0.80 ± 4.13**	**0.94**	**2.33 ± 8.30**	412.92 ± 85.17
HH-C2-R1-12	PNT+	1.12 ± 0.45	0.01	26.65 ± 11.14	**380.74 ± 70.12**

whose time derivative does not depend on any optimized parameter, we set the corresponding variance in the transition model to zero. With this simple heuristic, we repeat parameter estimation for the Hodgkin-Huxley type problems specified in Section 4.2.2.

Results Table 4.4 lists the corresponding results. In general, the altered transition model surpasses the regular one in all considered problems, either in terms of the accuracy of parameter estimates or by fewer optimization steps, except for the problem HH-C2-R1-12. More specifically, the number of iterations needed by the optimization algorithm is reduced for "-R4" problems, whereas parameter estimates become more accurate in the remaining problems. Figures 4.9a and 4.9b visualize the results of the optimization procedure for problems HH-R4-1 and HH-R4-2, respectively. When compared to Figs. 4.8a and 4.8b, which are based on the regular transition model, it becomes apparent that the NLL function is, in particular initially, more well-behaved, with the



(a) HH-R4-1.



(b) HH-R4-2.

Figure 4.9: Parameter estimation on problems HH-R4-1 (4.8a) and HH-R4-2 (4.8b) using PNT with the heuristic on parameter dependence, and NLSR. Plots show the NLL across the one and two-dimensional space of optimized parameters in different tempering iterations, respectively. PNT uses the optimization results of a tempering iteration as initial points for optimization in the next tempering iteration. NLSR performs optimization only on the zero-noise NLL function.

global minimum of the smoothed function visibly close to the true global minimum. While this explains the lower optimization effort in those cases, it remains unknown how exactly the results of the more complex problems arise. One can hypothesize that the isotropic noise forces the optimization into the boundary regions of the parameter space, as observable in Fig. 4.8b, and that the coarse tempering scheme subsequently leads to local minima arising quickly in these regions. Figure 4.9b suggests that this is mitigated by the heuristical choice of noise. Ultimately, however, it is unclear why the heuristic fails for the problem HH-C2-R1-12.

Discussion | 5

In this chapter, the methods used in the present work are discussed in a broader context. Their limitations are reviewed critically, and promising ideas for future work in these directions are outlined. Section 5.1 is about the probabilistic models used in this work. Subsequently, Section 5.2 deals with the aspects of scalability and efficiency, and Section 5.3 concludes with a discussion on process noise tempering.

5.1 | Probabilistic Models

In this work, we investigated how well probabilistic solutions created with the model $p_{\text{num.}}$ are calibrated in comparison to the manually calibrated p_{Conrad} by Conrad et al. (2017). The primary goal of our approach was to circumvent the additional effort of manual calibration and to provide a cheap method for ad-hoc uncertainty quantification of black box simulators that have access to a local error estimator. Section 4.2 has shown that the probabilistic solutions created with $p_{\text{num.}}$ are reasonable if the step size is sufficiently small, albeit often less well calibrated than with p_{Conrad}. Therefore, if one seeks to obtain a better calibration irrespective of the additional tuning effort, a promising next step would be to extend $p_{\text{num.}}$ by tunable parameters, either globally or per state dimension. Another option would be to change the way in which the error estimate goes into the covariance. While some preliminary tests were done with an outer product of the error estimate that performed worse than the diagonal covariance of $p_{\text{num.}}$, there are many other possible choices, e.g., a combination of the manually calibrated p_{Conrad} and such an outer product.

The used probabilistic models $p_{\text{num.}}$ and p_{PNT} are all Gaussian, as is the model p_{Conrad} they are based on. On one hand, this simplifies computations, enabling, e.g., approximate inference using the EKF. But on the other hand, the symmetry of Gaussians might pose a problem, in particular when modeling the numerical error of solutions. Since the error is signed, the model $p_{\text{num.}}$ assigns the same probability mass that covers the actual error also to the opposite part of the state space. Therefore, and due to potentially long tails of the distribution, sampling from the model could lead to states far off from the true trajectory of states with non-zero probability, and eventually cause divergence. While

this problem was not prominent in this work's experiments on sequential sampling, other types of distributions that are more informed about the dynamics of a particular ODE, e.g., by limited support of the state space, could further improve the calibration of uncertainty. However, this would prevent usage of the EKF, and limit inference to sequential sampling for now.

In the context of parameter estimation using PNT, the Gaussian prior is less problematic and even necessary to apply the EKF. However, as indicated in Section 4.2.3, incorporating prior knowledge into the model's covariance has the potential to boost its performance. Here, we have shown that a simple heuristic on the parameter dependence improves the results in the vast majority of the considered estimation problems. Another more sophisticated option would be to measure the sensitivity of individual state components to changes in the parameters. While preliminary tests using the Jacobian of the ODE or the numerical ODE solver with respect to parameters turned out less successful than with the simple heuristic, there may be other ways to include more prior knowledge into the model that ultimately improves performance further.

Since the transition model p_{PNT} only indirectly accounts for the uncertainty about parameters through the uncertainty about state predictions, a diametral option would be to extend the whole probabilistic SSM. More precisely, one could directly model parameters in a probabilistic fashion using a prior probability distribution on parameters. This would not only allow to incorporate prior knowledge on parameters through specification of the prior, but also facilitate the computation of probabilistic parameter estimates represented by their posterior distribution, as opposed to the maximum likelihood estimates computed by the approach adopted here. This idea goes into the direction of *latent force inference*, which was subject of a work by Schmidt et al. (2021). However, the authors there used a fully probabilistic ODE solver, whereas the computational tractability of the approach in case of non-probabilistic black box ODE solvers would have to be investigated first.

Last, as outlined in 3.1, inequidistant time discretizations would in principle be tolerated by the used probabilistic models, but render inference more difficult. With sequential sampling, the parallelization of simulations and the subsequent computation of meaningful statistics for the sample solutions are hindered. While pure prediction tasks using the EKF are not influenced negatively, filtering also becomes more complicated, as observations might not be available at the exact points in time. Hence, e.g., parameter estimation with the PNT algorithm could potentially be made more efficient through adaptive step size selection of the numerical ODE solver, but one would have to provide observations on a more fine-grained time axis and/or deal with inaccuracies introduced by observations and predictions not being time-coherent.

5.2 | Scalability and Efficiency

Although various ODEs are covered in this work, they can be considered fairly low-dimensional. Even the largest Hodgkin-Huxley model with two compartments that was used in Sections 4.2.2 and 4.2.3 only has a 14-dimensional state. By contrast, spatial discretization of partial differential equations often leads to ODEs whose states contain several thousands or even millions of dimensions. Since the complexity of the EKF is cubic with respect to the state dimensionality, the methods presented in this work quickly become computationally infeasible when applied to such problems. This does not hold for sequential sampling, but its applicability is generally limited due to the need for massive parallelization and the lack of a cheap approximation to the marginal likelihood of observations. However, there are ways to approximate the EKF and make computations tractable in high-dimensional state spaces, e.g., via low-rank approximation of covariance matrices (Schmidt et al., 2023). Moreover, a recent work by Pförtner et al. (2024) presented a novel PN-based approximation to the EKF. It appropriately accounts for the additional uncertainty introduced by the approximation and provides a tunable trade-off between computational cost and predictive uncertainty.

In the performance evaluation of PNT against other methods for parameter estimation (cf. Section 4.2), the total number of optimization steps needed until convergence was used to assess the efficiency of the methods. This was mainly done to compare directly to the results of Beck et al. (2024), who also used the metric for that purpose. However, in order to compare the methods' efficiency in a truly fair manner, one would have to perform detailed runtime benchmarks for each method. For reasons of time, extensive benchmarking was not done in the scope of this work.

5.3 | Process Noise Tempering

In this work's experiments on ODE parameter estimation using PNT, several assumptions were made:

First, we assumed that the initial value is fully known, which is often not the case in practical applications. As hinted in Remark 3.2, the initial value could in principle also be considered part of the parameters to be estimated. The question of how reliably PNT can estimate unknown initial values is, however, left open for future work.

Second, the results in Section 4.2.1 have shown that the addition of local error estimates to the covariance of the transition model does not bring an advantage. By assuming the rather small step size of $h = 0.01$, though, the numerical error introduced by the RKF4(5) solver can be considered negligible (cf. Section 4.1.2). It would therefore be

interesting to investigate whether the combined model $p_{\text{PNT+num.}}$ is beneficial in cases of larger step sizes and larger numerical errors. But since the error estimator does not provide a more informative signal on the true location of parameters, one can hypothesize that it would not improve maximum likelihood estimation.

Last, we assumed that observations are available at every point in time and that the observation noise $\mathbf{R}$ is known beforehand. It can be hypothesized that only irregularly available observations with an unknown magnitude of noise would worsen the reliability of parameter estimation using PNT. To what extent, however, remains to be answered.

Concluding this section, there are other open research directions by which the performance of the PNT algorithm could be increased further. Besides the usage of more informed covariances in the transition model that was already mentioned in Section 5.1, one could develop more advanced, problem-dependent tempering schedules. Finally, another promising approach would be to simulate the ODE during parameter estimation not in a single sweep, but partition the time horizon and simulate them separately, also known as *multiple shooting*. The interested reader is referred to Peifer and Timmer (2007) for a comprehensive introduction.

Related Work | 6

This chapter presents more recent research results in the fields of probabilistic ODE solvers (Section 6.1) and ODE parameter estimation (Section 6.2). They are compared with the approaches taken in this work, and the main similarities and differences are briefly outlined.

6.1 | Probabilistic ODE Solvers

First described by Diaconis (1988) and Skilling (1992), the idea of rephrasing the numerical task of solving ODEs as a Bayesian inference problem gained increasing interest during the last decade. Corresponding works fall into the broader field of PN (Hennig et al., 2022) and aim for probability distributions over solution spaces rather than mere point estimates provided by predominant numerical algorithms. While this new paradigm enables, e.g., comprehensive uncertainty quantification using confidence intervals, it typically also comes at the price of increased computational cost and potentially less strict convergence rates (Hennig & Hauberg, 2014), two aspects classical algorithms like RK solvers excel at.

An early work by Schober et al. (2014) tried to bridge the gap by designing a probabilistic ODE solver that computes solutions as a Gaussian process whose posterior mean is identical to the solution of RK methods of first, second, or third order. Thus, it automatically inherits error bounds from the latter. Extending the approach to higher-order RK methods, however, turned out difficult, limiting its practical applicability. From a result-oriented perspective, the method is similar to this work when inference is conducted using the EKF, as the resulting mean trajectory then corresponds to the deterministic solution of the black box solver, which can be an RK method as well. The covariance, however, differs in general. More conceptual differences are the restriction to low orders, but also the more rigorous theoretical foundation in Schober et al. (2014). Their framing of the problem as Gaussian process regression enables cheap, dense sampling from the posterior after its one-time computation. This work, by contrast, requires a full ODE solve per sample trajectory on a discretized time interval.

Building upon the work of Schober et al. (2014), Kersting and Hennig (2016) improved

the calibration of the posterior uncertainty by obtaining observations not through single, but multiple evaluations of the vector field, which are then approximately integrated using Bayesian quadrature. As a consequence, they lost the strict theoretical RK error bounds but provided some empirical evidence for smaller derivations of the posterior mean from the true trajectory. Last, their method exhibits increased computational cost that is cubic w.r.t. the state dimensionality, a property shared with EKF-based inference in this work.

In the following years, above methods were refined and generalized by Schober et al. (2019) and Tronarp et al. (2019), who reframed them as stochastic filtering problems. Thus, both allow usage of efficient Bayesian filtering techniques to perform Gaussian process regression, similar to this work, where the EKF can be used for inference as well. While the formulation of Schober et al. (2019) coupled the prior and likelihood, Tronarp et al. (2019) came up with a decoupled likelihood model, being more rigorous with regard to the Bayesian inference framework and defining the basis for most subsequent research on so-called *probabilistic ODE filters*. For instance, Kersting, Sullivan, and Hennig (2020) derived polynomial convergence rates of the local error, competitive to those of classic RK solvers. More recently, works by Bosch et al. (2021) and Krämer et al. (2022) extended probabilistic ODE filters by adaptive step size selection, further improved calibration of the uncertainty, and scaled them to extremely high-dimensional state spaces under additional approximative assumptions. While those properties are mostly not matched by the current method presented in this work, ideas in some of the mentioned directions have been outlined in Chapter 5.

Conceptionally most similar to this work's approach is the method proposed by Conrad et al. (2017). They perturbed predictions of a classic RK solver step-wise using zero-mean Gaussian noise, yielding a collection of sample trajectories that represents a probabilistic ODE solution. For specific choices of the noise covariance, they proved that the stochastic process that underlies the sample trajectories converges to the true solution with a polynomial rate, in accordance with RK solvers. However, they had to determine the magnitude of the noise covariance anew for every ODE (or different parametrizations of the same ODE) by solving a non-linear optimization problem. This work investigated whether the local error estimate provided by many ODE solvers, e.g., embedded RK methods, can be exploited to directly inform about the noise covariance, circumventing manual calibration. Whereas above probabilistic ODE filters represent a distinct class of probabilistic solvers with specific properties, an alternative has been proposed for how any existing black box simulator that provides a suitable error estimate can be equipped with ad hoc uncertainty quantification.

6.2 | ODE Parameter Estimation

The task of estimating unknown parameters of ODEs from noisy or partial measurements of the state, also known as inverse problems, is notoriously difficult. Most relevant dynamical systems exhibit highly non-linear dynamics, and closed-form solutions to corresponding ODEs are usually not available. In consequence, one has to conduct numerous numerical simulations to obtain state trajectories for different parameter choices, making the estimation of the true parameters computationally demanding. Moreover, NLSR (Bard, 1974) often does not converge to the global minimum, but only to one of many local minima (Cao et al., 2011), requiring multiple restarts at different initial parametrizations.

In order to overcome the computational burden of repeated simulations, Varah (1982) fitted splines to the measured data and then minimized the least-squares deviation between evaluations of the vector field and time derivatives of the fitted splines. This was an early example of a *collocation method* (Ramsay et al., 2007), which attempts to construct a differentiable surrogate model comprised of simple basis functions for the given data. Works by Ramsay et al. (2007) and Cao et al. (2011) then developed the method further, allowing also for partial state measurements and being more robust with respect to outliers in the data. In general, the accuracy of the estimated parameters depends strongly on the quality of the surrogate model. This work's approach is fundamentally different from such collocation methods, as it does not involve fitting a surrogate model, but is based on regular simulations using a generic ODE solver.

Evolutionary algorithms are another class of methods regularly applied to solve the problem of unknown parameters. For example, Nyarko and Scitovski (2004) and Akman et al. (2018) used a genetic algorithm and particle swarm optimization (Kennedy & Eberhart, 1995), respectively, in order to estimate the parameters of various ODEs. Furthermore, Gonzalez et al. (2007) determined parameters of a common biochemical model using simulated annealing (Kirkpatrick et al., 1983). All these algorithms operate gradient-free but become typically less efficient for higher-dimensional problems. By contrast, gradient information in the form of quasi-Newton optimization was used in this work.

Closely related to simulated annealing are Markov chain Monte Carlo (MCMC) algorithms, which can be used to compute a posterior distribution of the sought parameters, given the measurements (Gelman et al., 1996). In a later work, Dass et al. (2017) approximated the posterior of the parameters by the Laplace approximation, reducing the computational effort. However, while the prospect of a probability measure over parameter estimations is appealing, these and related methods for approximate Bayesian

computation have so far been shown to model the true posterior insufficiently in practice (Alahmadi et al., 2020). The method proposed in this work does not yield a full posterior, but only maximum-likelihood estimates of parameters.

Last, several methods have been developed for parameter estimation using PN ODE solvers (cf. Section 6.1). In this regard, Kersting, Krämer, et al. (2020) derived a differentiable approximation to the parameter likelihood through linearization, enabling both gradient-based and sampling-based optimization. A subsequent work by Tronarp et al. (2022) built upon this, reducing the computational complexity from cubic-in-time to linear-in-time by exploiting modern probabilistic ODE filters. Their method, called *Fenrir* (physics-enhanced regression in initial value problems), was only recently improved by Beck et al. (2024), who gradually decreased the diffusion parameter of the underlying probabilistic model during optimization. The PNT algorithm presented in this work has been inspired by the work of Beck et al. (2024). While it does not require a PN ODE solver, but is applicable to black box simulators, a similar smoothing effect on the objective function is achieved by successively lowering the process noise in the probabilistic model.

Conclusion | 7

This work explored how Bayesian filtering methods, in particular the extended Kalman filter (EKF), can be used to (a) create probabilistic solutions to ODEs and (b) reliably estimate their parameters, given a black box simulator.

First, in Section 3.1, we specified a simple Gaussian model that is wrapped around predictions of a black box ODE solver. Sequential sampling and the EKF were stated as methods to perform inference in this model and therefore create probabilistic solutions of an ODE. Furthermore, the model's assumptions as well as requirements for the black box solver were elaborated. Most importantly, the solver has to provide an estimator for the numerical error and needs to be differentiable in order to apply the EKF.

Section 3.2 then introduced the process noise tempering (PNT) algorithm, being developed to compute accurate maximum-likelihood estimates of ODEs parameters. This is achieved by gradually smoothing the likelihood function, which is then optimized using a gradient-based optimization method. PNT also applies the EKF to obtain a tractable approximation of the likelihood and is based on a similar probabilistic model as used to produce probabilistic ODE solutions.

Finally, we conducted an experimental evaluation on the proposed methods in Chapter 4. Here, the probabilistic solutions that were created for various simulation problems have been shown to capture the uncertainty about the numerical error qualitatively, but not always ideally calibrated. Overly large step sizes, however, occasionally led to catastrophic failure in the form of mode collapses. In the following, the PNT algorithm proved to estimate the correct parameter values reliably and efficiently for a variety of test problems, including the complex Hodgkin-Huxley model with over ten parameters. It has been shown that the schedule by which smoothing is performed is crucial and that the addition of the solver's error estimator can be neglected.

To conclude, this work provides methods to produce probabilistic ODE solutions that are not ideally calibrated, but ad-hoc computable, and to estimate unknown parameters reliably, both of which are applicable to black box simulators. We hope that the presented methods as well as the additional ideas outlined in Chapter 5 inspire further research in this direction.

Implementation and Computing | A

Here, detailed information on the implementation and the computing environment used for this work's experiments (cf. Chapter 4) is provided.

A.1 | Implementation Details

For all experiments, a self-developed Python codebase was used. It utilizes *Jax* (Bradbury et al., 2018) for automatic differentiation, linear algebra, and numerical computations. While the explicit embedded RK solvers RKF4(5) and DOPRI6(5) were self-implemented, the implicit Kvaerno3(2) solver used for simulations of the Hodgkin-Huxley model was provided by the Jax library *Diffrax* (Kidger, 2022). For gradient-based optimization in the context of parameter estimation, the "ScipyBoundedMinimize" L-BFGS-B wrapper of *Jaxopt* (Blondel et al., 2022) was used. Other essential components like the extended Kalman filter in square-root form, sequential sampling, non-linear least-squares-regression, as well as the process noise tempering algorithm were also self-implemented. The code is publicly available at `https://github.com/f-lair/ode-uncertainty`.

A.2 | Computing Environment

Experiments 1.1 to 1.3 and 2.1 were conducted on a consumer-grade CPU with 16 GB RAM. Due to the increased computational load of simulating the more complex Hodgkin-Huxley model with an implicit ODE solver, experiments 2.2 and 2.3 were carried out on a computing cluster with Intel Xeon Platinum 8380 CPU. In principle, however, they can also be run on the aforementioned consumer-level hardware.

The Hodgkin-Huxley Model | B

In the following, the Hodgkin-Huxley model used in this work's experiments about ODE parameter estimation is presented. It describes the initiation and propagation of action potentials in neurons by the dynamics of an electrical circuit. Here, we use the formulation of Pospischil et al. (2008). Appendices B.1 and B.2 specify the basic single-compartment model and the extension to a multi-compartment model, respectively.

B.1 | Single-Compartment Model

The non-reduced single-compartment model illustrated by Fig. B.1 considers sodium, potassium, and leak currents to generate spiking, an additional slow potassium current for spike-frequency adaptation, as well as high- and low-threshold calcium currents to

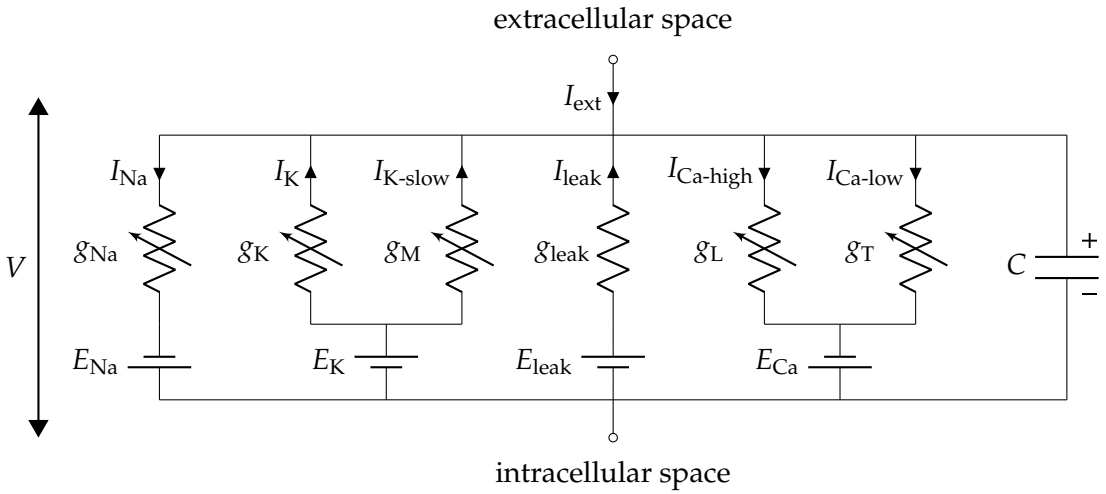


Figure B.1: Electrical circuit representing the used Hodgkin-Huxley model (cf. Pospischil et al. (2008)). The lipid bilayer acts as a capacitor (C) and ion channels are represented by resistors. Sodium, potassium and calcium conductances (g_{Na}, g_K, g_M, g_L, g_T) are voltage dependent, whereas the leak conductance (g_{Na}) is constant. The electrochemical gradients driving the flow of ions are modeled by voltage sources (E_{Na}, E_K, E_{leak}, E_{Ca}).

generate bursting. It describes the change of voltage by the first-order ODE

$$\begin{aligned}\frac{\mathrm{d}V(t)}{\mathrm{d}t} = \frac{1}{CA}\Bigg(& I_{\text{ext}}(t) + \underbrace{g_{\text{Na}} m^3(t) h(t)\,(E_{\text{Na}} - V(t))}_{-I_{\text{Na}}} + \underbrace{g_{\text{K}} n^4(t)\,(E_{\text{K}} - V(t))}_{-I_{\text{K}}} + \\ & \underbrace{g_{\text{leak}}\,(E_{\text{leak}} - V(t))}_{-I_{\text{leak}}} + \underbrace{g_{\text{M}} p(t)\,(E_{\text{K}} - V(t))}_{-I_{\text{K-slow}}} + \\ & \underbrace{g_{\text{L}} q^2(t) r(t)\,(E_{\text{Ca}} - V(t))}_{-I_{\text{Ca-high}}} + \underbrace{g_{\text{T}} s_\infty^2(V(t), V_X) u(t)\,(E_{\text{Ca}} - V(t))}_{-I_{\text{Ca-low}}} \Bigg),\end{aligned} \tag{B.1}$$

where $I_{\text{ext}}(t)$ represents an external current, while $h(t)$, $m(t)$, $n(t)$, $p(t)$, $q(t)$, $r(t)$, and $u(t)$ denote the fraction of independent gates in the open state, following the convention of Hodgkin and Huxley (1952). In this work's experiments, the external current is specified by

$$I_{\text{ext}}(t) = \begin{cases} 2.1 \times 10^{-4} & 10 \le t \le 90, \\ 0 & \text{else}, \end{cases}$$

in accordance with Beck et al. (2024). The dynamics of the gates are modeled by additional ODEs

$$\frac{\mathrm{d}h(t)}{\mathrm{d}t} = \alpha_h(V(t); V_T)(1 - h(t)) - \beta_h(V(t); V_T) h(t), \tag{B.2}$$

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = \alpha_m(V(t); V_T)(1 - m(t)) - \beta_m(V(t); V_T) m(t), \tag{B.3}$$

$$\frac{\mathrm{d}n(t)}{\mathrm{d}t} = \alpha_n(V(t); V_T)(1 - n(t)) - \beta_n(V(t); V_T) n(t), \tag{B.4}$$

$$\frac{\mathrm{d}p(t)}{\mathrm{d}t} = \frac{p_\infty(V(t)) - p(t)}{\tau_p(V(t); \tau_{\max})}, \tag{B.5}$$

$$\frac{\mathrm{d}q(t)}{\mathrm{d}t} = \alpha_q(V(t))(1 - q(t)) - \beta_q(V(t)) q(t), \tag{B.6}$$

$$\frac{\mathrm{d}r(t)}{\mathrm{d}t} = \alpha_r(V(t))(1 - r(t)) - \beta_r(V(t)) r(t), \tag{B.7}$$

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = \frac{u_\infty(V(t), V_X) - u(t)}{\tau_u(V(t); V_X)}. \tag{B.8}$$

Here, gating functions α_v, β_v and τ_w for $v \in \{m, h, n, q, r\}$ and $w \in \{p, u\}$ are given by

$$
\begin{aligned}
\alpha_h(V; V_T) &= 0.128 \exp\left((V_T + 17 - V)/18\right), \\
\alpha_m(V; V_T) &= \frac{-0.32(V - V_T - 13)}{\exp\left((V_T + 13 - V)/4\right) - 1}, \\
\alpha_n(V; V_T) &= \frac{-0.032(V - V_T - 15)}{\exp\left((V_T + 15 - V)/5\right) - 1}, \\
\alpha_q(V) &= \frac{0.055(-27 - V)}{\exp\left((-27 - V)/3.8\right) - 1}, \\
\alpha_r(V) &= 0.000457 \exp\left((-13 - V)/50\right), \\
\beta_h(V; V_T) &= \frac{4}{1 + \exp\left((V_T + 40 - V)/5\right)}, \\
\beta_m(V; V_T) &= \frac{0.28(V - V_T - 40)}{\exp\left((V_T + 40 - V)/5\right) - 1}, \\
\beta_n(V; V_T) &= 0.5 \exp\left((V_T + 10 - V)/40\right), \\
\beta_q(V) &= 0.94 \exp\left((-75 - V)/17\right), \\
\beta_r(V) &= \frac{0.0065}{\exp\left((-15 - V)/28\right) + 1}, \\
\tau_p(V; \tau_{\max}) &= \frac{\tau_{\max}}{3.3 \exp\left((V + 35)/20\right) + \exp\left((-35 - V)/20\right)}, \\
\tau_u(V; V_X) &= \frac{30.8(211.4 + \exp\left((V + V_X + 113.2)/5\right))}{3.7(\exp\left((V + V_X + 84)/3.2\right) + 1)}.
\end{aligned}
$$

Finally, gate activations at steady-state are computed by

$$h_\infty(V; V_T) = \left(1 + \frac{\beta_h(V; V_T)}{\alpha_h(V; V_T)}\right)^{-1}, \tag{B.9}$$

$$m_\infty(V; V_T) = \left(1 + \frac{\beta_m(V; V_T)}{\alpha_m(V; V_T)}\right)^{-1}, \tag{B.10}$$

$$n_\infty(V; V_T) = \left(1 + \frac{\beta_n(V; V_T)}{\alpha_n(V; V_T)}\right)^{-1}, \tag{B.11}$$

$$p_\infty(V) = \left(1 + \exp -\frac{V + 35}{10}\right)^{-1}, \tag{B.12}$$

$$q_\infty(V) = \left(1 + \frac{\beta_q(V)}{\alpha_q(V)}\right)^{-1}, \tag{B.13}$$

$$r_\infty(V) = \left(1 + \frac{\beta_r(V)}{\alpha_r(V)}\right)^{-1}, \tag{B.14}$$

$$s_\infty(V; V_X) = \left(1 + \exp -\frac{V + V_X + 57}{6.2}\right)^{-1}, \tag{B.15}$$

Table B.1: Parameters of the Hodgkin-Huxley model and their interpretation, together with bounds and true values used for experiments.

PARAMETER (UNIT)	INTERPRETATION	BOUNDS (LOW, HIGH)	TRUE VALUE
C ($\mu\mathrm{F\,cm^{-2}}$)	Membrane capatitance	$(0.4, 3.0)$	1.0
A (cm^2)	Compartment area	$(1.9 \times 10^{-5}, 3.2 \times 10^{-4})$	8.3×10^{-5}
g_{Na} (mS)	Na^+ conductance	$(0.5, 80.0)$	25.0
g_{K} (mS)	K^+ conductance	$(1 \times 10^{-4}, 15.0)$	7.0
g_{leak} (mS)	Leak conductance	$(1 \times 10^{-4}, 0.6)$	0.1
g_{M} (mS)	Slow K^+ conductance	$(1 \times 10^{-5}, 0.6)$	0.01
g_{L} (mS)	High-T Ca^{2+} conductance	$(-1 \times 10^{-4}, 0.6)$	0.01
g_{T} (mS)	Low-T Ca^{2+} conductance	$(-1 \times 10^{-4}, 0.6)$	0.01
E_{Na} (mV)	Na^+ reversal potential	$(50.0, 100.0)$	53.0
E_{K} (mV)	K^+ reversal potential	$(-110.0, -70.0)$	-107.0
E_{leak} (mV)	Leak reversal potential	$(-100.0, -35.0)$	-70.0
E_{Ca} (mV)	Ca^{2+} reversal potential	$(100.0, 150.0)$	120.0
$\tau_{\max}$ (s)	Slow K^+ time constant	$(50.0, 5 \times 10^3)$	4×10^3
V_T (mV)	Threshold voltage	$(-90.0, -40.0)$	-60.0
V_X (mV)	Voltage dependence shift	$(0.0, 4.0)$	2.0

$$u_\infty(V; V_X) = \left(1 + \exp \frac{V + V_X + 81}{4}\right)^{-1}. \tag{B.16}$$

With the definition of the state $\mathbf{x}(t)$ and parameters θ as

$$\mathbf{x}(t) \coloneqq \begin{bmatrix} V(t) & h(t) & m(t) & n(t) & p(t) & q(t) & r(t) & u(t) \end{bmatrix}^\mathsf{T}$$

and

$$\theta \coloneqq \begin{bmatrix} C & A & g_{\mathrm{Na}} & g_{\mathrm{K}} & g_{\mathrm{leak}} & g_{\mathrm{M}} & g_{\mathrm{L}} & g_{\mathrm{T}} & E_{\mathrm{Na}} & E_{\mathrm{K}} & E_{\mathrm{leak}} & E_{\mathrm{Ca}} & \tau_{\max} & V_T & V_X \end{bmatrix}^\mathsf{T},$$

Eqs. (B.1) to (B.8) give rise to a system of eight ODE with 15 parameters in the form of Eq. (2.7). Its initial value $\mathbf{x}_0$ is provided by an initial voltage V_0 and corresponding gate activations at steady-state (cf. Eqs. (B.9) to (B.14) and (B.16)), i.e.,

$$\mathbf{x}(0) \coloneqq \begin{bmatrix} V_0 \\ h_\infty(V_0; V_T) \\ m_\infty(V_0; V_T) \\ n_\infty(V_0; V_T) \\ p_\infty(V_0) \\ q_\infty(V_0) \\ r_\infty(V_0) \\ u_\infty(V_0; V_X) \end{bmatrix}.$$

Table B.1 gives a summary on the model's parameters, together with the bounds and true values used for experiments.

B.2 | Multi-Compartment Model

In the above single-compartment model, a neuron is modeled by a single electrical circuit. To model neurons with a complex morphological structure appropriately, a *multi-compartment model* splits the neuron into N discrete compartments. If these are small enough so that the variation of the membrane potential across compartments is negligible, the continuous membrane potential V can be approximated by a sum of local compartment potentials. In case of a non-branching cable, this can be expressed as

$$\frac{\mathrm{d}V_i(t)}{\mathrm{d}t} = \frac{1}{C}\left(\frac{1}{A_i}I_{\text{ext},i}(t) - I_{\text{ionic},i} + g_{i,i+1}(V_{i+1}(t) - V_i(t)) + g_{i,i-1}(V_{i-1}(t) - V_i(t))\right), \tag{B.17}$$

where $i \in \{1, 2, \ldots, N\}$ indexes the compartments, $I_{\text{ext},i}$ and $I_{\text{ionic},i}$ refer to the external current to and the sum of ionic currents in the i-th compartment, respectively, and $g_{i,i+1} = g_{i+1,i}$ denotes the coupling coefficients of adjacent compartments i and $i+1$ (Beck et al., 2024).

For this work's experiments acting on Eq. (B.17), we again follow Beck et al. (2024) and consider identical coupling coefficients of $g_{i,i+1} = 1$ and equally distributed compartment areas of $A_i = \frac{A}{N}$, leading to the same total area as with a single-compartment model. Furthermore, the capacitance C is the same as in the single-compartment model. Observations are taken individually per compartment, such that the observation matrix $\mathbf{H}$ for the multi-compartment model is composed as a block-diagonal matrix from observations matrices $\mathbf{H}_i$ for the individual compartments, i.e.,

$$\mathbf{H} \coloneqq \begin{bmatrix} \mathbf{H}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{H}_N \end{bmatrix}.$$

Finally, parameters θ_i of individual compartments are stacked to form the parameter vector θ of the multi-compartment model, i.e.,

$$\theta \coloneqq \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_N \end{bmatrix}.$$

Table B.2: True parameter values used for experiments with a two-compartment Hodgkin-Huxley model.

PARAMETER (UNIT)	TRUE VALUE (1ST COMP.)	TRUE VALUE (2ND COMP.)
g_{Na} (mS)	25.0	20.0
g_{K} (mS)	7.0	10.0
g_{leak} (mS)	0.09	0.11
g_{M} (mS)	0.01	0.1
g_{L} (mS)	0.1	0.01
g_{T} (mS)	0.01	0.01
E_{Na} (mV)	53.0	53.0
E_{K} (mV)	-107.0	-107.0
E_{leak} (mV)	-70.0	-70.0
E_{Ca} (mV)	120.0	120.0
τ_{max} (s)	4×10^3	4×10^3
V_T (mV)	-70.0	-50.0
V_X (mV)	2.0	2.0

Adapted true parameter values used for experiments with a two-compartment Hodgkin-Huxley model are listed in Table B.2.

Bibliography

Akman, D., Akman, O., & Schaefer, E. (2018). Parameter Estimation in Ordinary Differential Equations Modeling via Particle Swarm Optimization. *Journal of Applied Mathematics*, *2018*(1), 9160793. https://doi.org/10.1155/2018/9160793

Alahmadi, A. A., Flegg, J. A., Cochrane, D. G., Drovandi, C. C., & Keith, J. M. (2020). A comparison of approximate versus exact techniques for Bayesian parameter inference in nonlinear ordinary differential equation models. *Royal Society Open Science*, *7*(3), 191315. https://doi.org/10.1098/rsos.191315

Alspach, D., & Sorenson, H. (1972). Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE Transactions on Automatic Control*, *17*(4), 439–448. https://doi.org/10.1109/TAC.1972.1100034

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., … Chintala, S. (2024). PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. https://doi.org/10.1145/3620665.3640366

Bard, Y. (1974). *Nonlinear parameter estimation* (Vol. 1209). Academic press New York.

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. *Proceedings of the 41st International Conference on Machine Learning*, *235*, 3305–3326. https://proceedings.mlr.press/v235/beck24a.html

Blake, A., & Zisserman, A. (1987). *Visual Reconstruction*. The MIT Press. https://doi.org/10.7551/mitpress/7132.001.0001

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., & Vert, J.-P. (2022). *Efficient and Modular Implicit Differentiation*. arXiv: 2105.15183. https://doi.org/10.48550/arXiv.2105.15183

Bogacki, P., & Shampine, L. F. (1989). A 3(2) pair of Runge - Kutta formulas. *Applied Mathematics Letters*, *2*(4), 321–325. https://doi.org/10.1016/0893-9659(89)90079-7

Bosch, N., Hennig, P., & Tronarp, F. (2021). Calibrated adaptive probabilistic ODE solvers. *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, *130*, 3466–3474. https://proceedings.mlr.press/v130/bosch21a.html

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/jax-ml/jax

Bucy, R. S., & Joseph, P. D. (2005). *Filtering for Stochastic Processes With Applications to Guidance*. American Mathematical Society.

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, *16*(5), 1190–1208. https://doi.org/10.1137/0916069

Cao, J., Wang, L., & Xu, J. (2011). Robust estimation for ordinary differential equation models. *Biometrics. Journal of the International Biometric Society*, *67*(4), 1305–1313. https://doi.org/10.1111/j.1541-0420.2011.01577.x

Conrad, P. R., Girolami, M., Särkkä, S., Stuart, A., & Zygalakis, K. (2017). Statistical analysis of differential equations: Introducing probability measures on numerical solutions. *Statistics and Computing*, *27*(4), 1065–1082. https://doi.org/10.1007/s11222-016-9671-0

Cooper, I., Mondal, A., & Antonopoulos, C. G. (2020). A SIR model assumption for the spread of COVID-19 in different communities. *Chaos, Solitons & Fractals*, *139*, 110057. https://doi.org/10.1016/j.chaos.2020.110057

Crisan, D., & Doucet, A. (2002). A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, *50*(3), 736–746. https://doi.org/10.1109/78.984773

Dass, S. C., Lee, J., Lee, K., & Park, J. (2017). Laplace based approximate posterior inference for differential equation models. *Statistics and Computing*, *27*(3), 679–698. https://doi.org/10.1007/S11222-016-9647-0

Diaconis, P. (1988). Bayesian numerical analysis. *Statistical decision theory and related topics IV*, *1*, 163–175.

Dormand, J. R., & Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, *6*(1), 19–26. https://doi.org/10.1016/0771-050X(80)90013-3

Euler, L. (1768). *Institutiones calculi integralis* (Vol. 1). Academia Imperialis Scientiarum.

Fehlberg, E. (1970). Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme. *Computing*, *6*(1), 61–71. https://doi.org/10.1007/BF02241732

Fitzpatrick, R. (2018). *Oscillations and Waves: An Introduction, Second Edition* (2nd ed.). CRC Press.

Gelman, A., Bois, F., & Jiang, J. (1996). Physiological Pharmacokinetic Analysis Using Population Modeling and Informative Prior Distributions. *Journal of the American Statistical Association*, *91*(436), 1400–1412. https://doi.org/10.1080/01621459.1996.10476708

Gharamani, Z. (2001). An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, *15*(01), 9–42. https://doi.org/10.1142/S0218001401000836

Gonzalez, O. R., Küper, C., Jung, K., Naval, P. C., Jr, & Mendoza, E. (2007). Parameter estimation using Simulated Annealing for S-system models of biochemical networks. *Bioinformatics*, *23*(4), 480–486. https://doi.org/10.1093/bioinformatics/btl522

Gordon, N., Salmond, D., & Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, *140*(2), 107–113. https://doi.org/10.1049/ip-f-2.1993.0015

Grewal, M. S., & Andrews, A. P. (2014). *Kalman filtering: Theory and practice with MATLAB*. John Wiley & Sons.

Guckenheimer, J. (1980). Dynamics of the Van der Pol equation. *IEEE Transactions on Circuits and Systems*, *27*(11), 983–989. https://doi.org/10.1109/TCS.1980.1084738

Hairer, E., Wanner, G., & Nørsett, S. P. (1993). *Solving Ordinary Differential Equations I* (2nd ed.). Springer. https://doi.org/10.1007/978-3-540-78862-1

Hennig, P., & Hauberg, S. (2014). Probabilistic solutions to differential equations and their application to riemannian statistics. *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, *33*, 347–355.

Hennig, P., Osborne, M. A., & Kersting, H. P. (2022). *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press. https://doi.org/10.1017/9781316681411

Higham, D. J. (2008). Modeling and simulating chemical reactions. *SIAM Review*, *50*(2), 347–368. https://doi.org/10.1137/060666457

Higham, N. J. (2008). *Functions of matrices*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898717778

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, *117*(4), 500–544. https://doi.org/10.1113/jphysiol.1952.sp004764

Huber, M. F. (2011). Adaptive Gaussian mixture filter based on statistical linearization. *14th International Conference on Information Fusion*, 1–8.

Iserles, A. (2008). *A First Course in the Numerical Analysis of Differential Equations* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511995569

Julier, S. J., & Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. *3068*, 182–193. https://doi.org/10.1117/12.280797

Kelley, C. T. (2003). *Solving nonlinear equations with newton's method*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718898

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Kersting, H., & Hennig, P. (2016). Active uncertainty calibration in bayesian ODE solvers. *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI)*, 309–318. http://www.auai.org/uai2016/proceedings/papers/163.pdf

Kersting, H., Krämer, N., Schiegg, M., Daniel, C., Tiemann, M., & Hennig, P. (2020). Differentiable likelihoods for fast inversion of 'likelihood-free' dynamical systems. *Proceedings of the 37th International Conference on Machine Learning (ICML)*, *119*, 5154–5164. http://proceedings.mlr.press/v119/kersting20a/kersting20a.pdf

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

Kidger, P. (2022). *On Neural Differential Equations*. arXiv: 2202.02435.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kraaijevanger, J., & Spijker, M. (1989). Algebraic stability and error propagation in Runge-Kutta methods. *Applied Numerical Mathematics*, *5*(1), 71–87. https://doi.org/10.1016/0168-9274(89)90025-1

Krämer, N., Bosch, N., Schmidt, J., & Hennig, P. (2022). Probabilistic ODE solutions in millions of dimensions. *Proceedings of the 39th International Conference on Machine Learning*, *162*, 11634–11649. https://proceedings.mlr.press/v162/kramer22b.html

Kutta, W. (1901). Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Zeit. Math. Phys.*, *46*, 435–53.

Kværnø, A. (2004). Singly Diagonally Implicit Runge–Kutta Methods with an Explicit First Stage. *BIT Numerical Mathematics*, *44*(3), 489–502. https://doi.org/10.1023/B:BITN.0000046811.70614.38

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*(1), 503–528. https://doi.org/10.1007/BF01589116

Lorenz, E. N. (1963). Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*, *20*(2), 130–141. https://doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2

Lotka, A. J. (1925). Elements of Physical Biology. *Nature*, *116*(2917), 461–461. https://doi.org/10.1038/116461b0

Musielak, Z. E., & Quarles, B. (2015). *The three-body problem*. arXiv: 1508.02312.

Nyarko, E. K., & Scitovski, R. (2004). Solving the parameter identification problem of mathematical models using genetic algorithms. *Applied Mathematics and Computation*, *153*(3), 651–658. https://doi.org/10.1016/S0096-3003(03)00661-1

Peifer, M., & Timmer, J. (2007). Parameter estimation in ordinary differential equations for biochemical processes using the method of multiple shooting. *IET Systems Biology*, *1*(2), 78–88. https://doi.org/10.1049/iet-syb:20060067

Pförtner, M., Wenger, J., Cockayne, J., & Hennig, P. (2024). *Computation-Aware Kalman Filtering and Smoothing*. arXiv: 2405.08971.

Pospischil, M., Toledo-Rodriguez, M., Monier, C., Piwkowska, Z., Bal, T., Frégnac, Y., Markram, H., & Destexhe, A. (2008). Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. *Biological Cybernetics*, *99*(4), 427–441. https://doi.org/10.1007/s00422-008-0263-8

Prince, P., & Dormand, J. (1981). High order embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, *7*(1), 67–75. https://doi.org/10.1016/0771-050X(81)90010-3

Ramsay, J. O., Hooker, G., Campbell, D., & Cao, J. (2007). Parameter estimation for differential equations: A generalized smoothing approach. *Journal of the Royal*

Statistical Society Series B: Statistical Methodology, *69*(5), 741–796. https://doi.org/10.1111/j.1467-9868.2007.00610.x

Rose, K. (1998). Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, *86*(11), 2210–2239. https://doi.org/10.1109/5.726788

Runge, C. (1895). Über die numerische Auflösung von Differentialgleichungen. *Mathematische Annalen*, *46*(2), 167–178. https://doi.org/10.1007/BF01446807

Särkkä, S., & Svensson, L. (2023). *Bayesian Filtering and Smoothing* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/9781108917407

Schmidt, J., Hennig, P., Nick, J., & Tronarp, F. (2023). The rank-reduced kalman filter: Approximate dynamical-low-rank filtering in high dimensions. *Thirty-Seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=dnB71DMyDD

Schmidt, J., Krämer, N., & Hennig, P. (2021). A probabilistic state space model for joint inference from differential equations and data. *Advances in Neural Information Processing Systems*, *34*, 12374–12385. https://proceedings.neurips.cc/paper_files/paper/2021/file/6734fa703f6633ab896eecbdfad8953a-Paper.pdf

Schober, M., Duvenaud, D. K., & Hennig, P. (2014). Probabilistic ODE Solvers with Runge-Kutta Means. *Advances in Neural Information Processing Systems*, *27*.

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

Shampine, L. F., & Watts, H. A. (1976). Global Error Estimates for Ordinary Differential Equations. *ACM Trans. Math. Softw.*, *2*(2), 172–186. https://doi.org/10.1145/355681.355687

Skilling, J. (1992). Bayesian Solution of Ordinary Differential Equations. In *Maximum Entropy and Bayesian Methods: Seattle, 1991* (pp. 23–37). Springer Netherlands. https://doi.org/10.1007/978-94-017-2219-3_2

Steeb, W.-H., Louw, J., & Villet, C. (1987). Linearly Coupled Anharmonic Oscillators and Integrability. *Australian Journal of Physics*, *40*(5), 587–592. https://doi.org/10.1071/PH870587

Trefethen, L. N., & Bau, D. (2022). *Numerical linear algebra* (Twenty-fifth anniversary edition). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611977165

Tronarp, F., Bosch, N., & Hennig, P. (2022). Fenrir: Physics-enhanced regression for initial value problems. *Proceedings of the 39th International Conference on Machine Learning*, *162*, 21776–21794. 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

Varah, J. M. (1982). A spline least squares method for numerical parameter estimation in differential equations. *SIAM Journal on Scientific and Statistical Computing*, *3*(1), 28–46. https://doi.org/10.1137/0903003

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. https://doi.org/10.1093/icesjms/3.1.3

Zhang, D. K. (2019). *Discovering new runge-kutta methods using unstructured numerical search*. arXiv: 1911.00318.

Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, *23*(4), 550–560. https://doi.org/10.1145/279232.279236

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift