Probabilistic Ordinary Differential Equation Solvers

Theory and Applications

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Dipl.-Inf. Michael Schober

aus Nürtingen

Tübingen

2018

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

Tag der mündlichen Qualifikation:	26.09.2018
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Philipp Hennig
2. Berichterstatter:	Prof. Dr. Ulrike von Luxburg

PhD Thesis

Practical Probabilistic Ordinary Differential Equation Solvers

—

Theory and Applications

Michael Schober
Eberhard Karls Universität Tübingen

Tübingen, June 2018

Abstract

Ordinary differential equations are ubiquitous in science and engineering, as they provide mathematical models for many physical processes. However, most practical purposes require the temporal evolution of a particular solution. Many relevant ordinary differential equations are known to lack closed-form solutions in terms of simple analytic functions. Thus, users rely on numerical algorithms to compute discrete approximations.

Numerical methods replace the intractable, and thus inaccessible, solution by an approximating model with known computational strategies. This is akin to a process in statistics where an unknown true relationship is modeled with access to instances of said relationship. One branch of statistics, *Bayesian modeling*, expresses degrees of uncertainty with probability distributions. In recent years, this idea has gained traction for the design and study of numerical algorithms which established *probabilistic numerics* as a research field in its own right.

The theory part of this thesis is concerned with bridging the gap between classical numerical methods for ordinary differential equations and probabilistic numerics. To this end, an algorithm is presented based on Gaussian processes, a general and versatile model for Bayesian regression. This algorithm is compared to two standard frameworks for the solution of initial value problems. It is shown that the maximum a-posteriori estimator of certain Gaussian process regressors coincide with certain multistep formulae. Furthermore, a particular initialization scheme based on an improper prior model coincides with a Runge-Kutta method for the first discretization step. This analysis provides *a higher-order probabilistic numerical algorithm for initial value problems*.

Based on the probabilistic description, an estimator of the local integration error is presented, which is used in a *step size adaptation scheme*. The completed algorithm is evaluated on a benchmark on initial value problems, confirming empirically the theoretically predicted error rates and displaying particularly efficient performance on domains with low accuracy requirements.

To establish the practical benefit of the probabilistic solution, a probabilistic boundary value problem solver is applied to a medical imaging problem. In *tractography*, diffusion-weighted magnetic resonance imaging data is used to infer connectivity of neural fibers. The first application of the probabilistic solver shows how the quantification of the discretization error can be used in subsequent estimation of fiber density. The second application additionally incorporates the measurement noise of the imaging data into the tract estimation model. These two extensions of the shortest-path tractography method give more faithful *data, modeling and algorithmic uncertainty* representations in neural connectivity studies.

Zusammenfassung

Gewöhnliche Differentialgleichungen sind allgegenwärtig in Wissenschaft und Technik, da sie die mathematische Beschreibung vieler physikalischen Vorgänge sind. Jedoch benötigt ein Großteil der praktischen Anwendungen die zeitliche Entwicklung einer bestimmten Lösung. Es ist bekannt, dass viele relevante gewöhnliche Differentialgleichungen keine geschlossene Lösung als Ausdrücke einfacher analytischer Funktion besitzen. Daher verlassen sich Anwender auf numerische Algorithmen, um diskrete Annäherungen zu berechnen.

Numerische Methoden ersetzen die unauswertbare, und daher unzugängliche, Lösung durch eine Annäherung mit bekannten Rechenverfahren. Dies ähnelt einem Vorgang in der Statistik, wobei ein unbekanntes wahres Verhältnis mittels Zugang zu Beispielen modeliert wird. Eine Unterdisziplin der Statistik, *Bayes'sche Modellierung*, stellt graduelle Unsicherheit mittels Wahrscheinlichkeitsverteilungen dar. In den letzten Jahren hat diese Idee an Zugkraft für die Konstruktion und Analyse von numerischen Algorithmen gewonnen, was zur Etablierung von *probabilistischer Numerik* als eigenständiges Forschungsgebiet führte.

Der Theorieteil dieser Dissertation schlägt eine Brücke zwischen herkömmlichen numerischen Verfahren zur Lösung gewöhnlicher Differentialgleichungen und probabilistischer Numerik. Ein auf Gauß'schen Prozessen basierender Algorithmus wird vorgestellt, welche ein generelles und vielseitiges Modell der Bayesschen Regression sind. Dieser Algorithmus wird verglichen mit zwei Standardansätzen für die Lösung von Anfangswertproblemen. Es wird gezeigt, dass der Maximum-a-posteriori-Schätzer bestimmter Gaußprozess-Regressoren übereinstimmt mit bestimmten Mehrschrittverfahren. Weiterhin stimmt ein besonderes Initialisierungsverfahren basierend auf einer uneigentlichen A-priori-Wahrscheinlichkeit überein mit einer Runge-Kutta Methode im ersten Rechenschritt. Diese Analyse führt zu einer *probabilistisch-numerischen Methode höherer Ordnung zur Lösung von Anfangswertproblemen*.

Basierend auf der probabilistischen Beschreibung wird ein Schätzer des lokalen Integrationfehlers präsentiert, welcher in einem *Schrittweitensteuerungsverfahren* verwendet wird. Der vollständige Algorithmus wird auf einem Satz standardisierter Anfangswertprobleme ausgewertet, um empirisch den von der Theorie vorhergesagten Fehler zu bestätigen. Der Test weist dem Verfahren einen besonders effizienten Rechenaufwand im Bereich der niedrigen Genauigkeitsanforderungen aus.

Um den praktischen Nutzen der probabilistischen Lösung nachzuweisen, wird ein probabilistischer Löser für Randwertprobleme auf eine Fragestellung der medizinischen Bildgebung angewandt. In der *Traktografie* werden die Daten der diffusionsgewichteten Magnetresonanzbildgebung verwendet, um die Konnektivität neuronaler Fasern zu bestimmen. Die erste Anwendung des probabilistische Lösers demonstriert, wie die Quantifizierung des Diskretisierungsfehlers in einer nachgeschalteten Schätzung der Faserdichte verwendet werden kann. Die zweite Anwendung integriert zusätzlich das Messrauschen der Bildgebungsdaten in das Strangschätzungsmodell. Diese beiden Erweiterun-

gen der Kürzesten-Pfad-Traktografie repräsentieren die *Daten-, Modellierungs- und algorithmische Unsicherheit* abbildungstreuer in neuronalen Konnektivitätsstudien.

Acknowledgments

First and foremost, I thank Philipp Hennig. I have not only learned how to conduct my own research, but also how to navigate the acadamic world from him. More importantly, I am very grateful for him never taking it personally when the discussion got appropriately heated.

I thank Ulrike v. Luxburg for her valuable time and energy invested in my thesis.

I thank my former colleagues Edgar Klenske, Maren Mahsereci, Simon Bartels, Hans Kersting, Lukas Balles, Alexandra Gessner, Emilia Magnani, Filip De Roos and all other PhD students in Tübingen. It's the discussions in the lab and over lunch that ensures steady progress.

I thank my collaborators Søren Hauberg, Aasa Feragen, Niklas Kasenburg, David Duvenaud and Simo Särkkä for their support and patience. In particular, I thank Søren, Aasa and Tom Dela Haije who helped me proof-reading the chapters on tractography.

I thank my peers Franz-Xaver Briol, Jon Cockayne, Toni Karvonen and Onur Teymur for the times we shared. A special thank you goes out to Jon for the very constructive and lively discussions on my journal article.

I thank Sebastian Nowozin and everyone at MSR Cambridge. In particular, I thank the foosball crowd as well as my fellow interns. It was a great summer.

I also thank my new colleagues at the Bosch Center for Artificial Intelligence. In particular, I thank Sebastian Gerwinn, Barbara Rakitsch, Christoph Zimmer and Patrick Engel who helped me proof-reading various chapters of this thesis.

A particular note of thanks I want to extend to Markus Schneller who deserves much credit in the creation of this thesis. Obscure papers published half a century ago (or longer) in journals of countries far away have magically appeared in my inbox. For the most of my time at the MPI, at least one textbook long out of print has been lying on top of my desk. Value your librarians for you will miss them when they're gone!

I thank Bernhard Schölkopf who first welcomed me to the institute and who has created this liberal atmosphere of curiosity.

I thank my family, Eva-Maria Denk, Melanie Schulz, and Rudolf Schober and I thank the boys: Jakob Erne, Wolfgang Preiß, David Sixt, and Rouven Witt.

I cannot thank enough Sonja Tiemann for giving me her support in hard times and giving me some challenges when I am in fear of getting stuck in my own ways. You are the application to my abstraction and together we make a pretty good fixed point combinator $(\lambda f.\,(\lambda x.\,f(x\,x))\,(\lambda x.\,f(x\,x)))\,\heartsuit$.

Listen master, can you answer a question?
Is it the fingers or the brain that you're teaching a lesson?
Oh, I can't tell you how proud I am
I'm writing down things that I don't understand
Well, maybe I'll put my love on ice
And teach myself, maybe that'll be nice

The White Stripes. "Black Math." *Elephant.* V2 Records, 2003.

Contents

List of Figures

List of Tables

List of Algorithms

Disclaimer

Gauß has been attributed to have said "no architect leaves the scaffolding after completing the building" (Kline [118, §5]), writing "elegant, but highly compact, carefully polished papers with no hint of the motivation, meaning or details of the steps" (Ibid.).

While I agree that there is an *ideal level of abstraction* in scientific communication, it depends both on the *purpose* of the communication and the *audience*. This manuscript is intended for fellow researchers of (probabilistic) numerical analysis. The purpose of this work is to find the connections between classical algorithms and probabilistic models and to enable colleagues, new and old, to speak a common language and to understand the motivation and the results of their respective counterparts.

This motivation entails important stylistic decisions. I have decided to use a more colloquial first person plural "we" throughout the thesis, to invite you, *dear reader*, on a common tour of discovery. Together, we will take the leap from classical numerical methods to probabilistic ordinary differential equation solvers, retracing our steps from its audacious beginnings to its first maturity and the establishment of a small, but steady, research community.

This style has been inspired by the excellent textbooks Hairer and Wanner [84] and Hairer, Nørsett, and Wanner [82]. I hope to follow their example, if not in quality, then at least in spirit. I apologize to readers not inclined to this style of presentation and commend my scientific publications to them.

Part I

Prologue

1 Introduction

The expansion of available scientific knowledge has always been a history of the development of richer mathematical models suitable of describing complex natural systems in an accessible language. Ever since the invention of calculus by Gottfried Wilhelm Leibniz[1], ordinary differential equations (ODEs), as well as partial differential equations, have played a fundamental role in describing natural phenomena. The long list of famous equations include:

[1] that settles it

- The linear ordinary differential equation

  $$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = \boldsymbol{K}\boldsymbol{a}.$$

  This equation describes exponential growth (or decay), and appears, e.g., in *rate equations* for chemical reactions.

- Newton's law of planetary motion

  $$m_i \frac{\mathrm{d}^2 \boldsymbol{y}_i}{\mathrm{d}t^2} = G \sum_{\substack{j=1 \\ j \neq i}}^{n} m_i m_j \frac{(\boldsymbol{y}_j - \boldsymbol{y}_i)}{\left|\left|\boldsymbol{y}_j - \boldsymbol{y}_i\right|\right|^3}$$

  for n bodies with masses m_i, positions $\boldsymbol{y}_i$ and the gravitational constant G.

- The law of population growth

  $$\frac{\mathrm{d}P}{\mathrm{d}t} = rP(1 - \frac{P}{K})$$

  describing the size of population P growing at rate r with limited resources K and its generalization to the Lotka-Volterra equations

  $$\frac{\mathrm{d}P_1}{\mathrm{d}t} = \alpha P_1 - \beta P_1 P_2 \quad \frac{\mathrm{d}P_2}{\mathrm{d}t} = \delta P_1 P_2 - \gamma P_2$$

  of a predator-prey relationship.

- Oscillator equations, like the universal oscillator equation

  $$\frac{\mathrm{d}^2 q}{\mathrm{d}t^2} + 2\zeta \frac{\mathrm{d}q}{\mathrm{d}t} + q = 0$$

  or generalizations thereof, e.g., the Van der Pol equation

  $$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} - \mu(1 - x^2)\frac{\mathrm{d}x}{\mathrm{d}t} + x = 0$$

and the FitzHugh-Nagumo model

$$\frac{\mathrm{d}V}{\mathrm{d}t} = V - \frac{V^3}{3} - W + I \quad \frac{\mathrm{d}W}{\mathrm{d}t} = \tau^{-1}(V + a - bW)$$

However, with new tools come new challenges. Setting aside partial differential equations which are a whole different beast altogether, many ordinary differential equations are very hard to solve. While the existence of solutions can already be shown with elementary methods (Piccard iteration), closed-form analytical expressions are hard to find. In some cases, it can even be shown that closed-form solutions don't exist. One example has already been given: for $n = 3$, Newton's equation is the famous *three-body problem* which has historically been focused on in lunar theory. Bruns [26] provided a first proof of the non-existence of a closed-form solution which was completed later by Poincaré [172].

Even before the advent of fast automatic computers, mathematicians have come up with numerical approximate methods to the exact solutions. Early attempts are Runge [179] and Kutta [124]. Another set of now fundamental methods are the family of multistep methods, originally developed by Adams (see Bashforth and Adams [11]) probably predating Runge-Kutta methods almost 50 years ([82, §III.1]). An excellent overview of the historical development is outlined in Butcher [27]. Nowadays, ODEs are solved routinely as part of numerical partial differential equation solvers, in physical, chemical or biological problems, in image processing applications, etc.

In a similar fashion, the advent of the calculus of probability theory has allowed the rigorous analysis of *uncertain* knowledge. Laplace [127] famously asserted an *uncertain* value to the specific weight of the planet Saturn, Reverend Bayes (in Bayes and Price [14]) is credited for coming up with turning prior beliefs and likelihood assumptions into posterior uncertainty, and Gauß has both invented the Gaussian normal distribution as well as the method of least squares.

While both these tools have a well-deserved place in a modern scientist's toolbox, there has not been a lot of attention on bringing both together to "speak a common language". While some problems are determined up to floating point precision, other systems might only be known up to a probabilistic uncertainty description. And yet, numerical analysists have developed a completely independent set of descriptions to articulate the additional uncertainty introduced by the finite description of the infinite dimensional problem. Thus, up to now practitioners have been forced to abandon probability theory once they enter the realms of numerical approximations and have had to awkwardly pick up the pieces again afterwards.

This thesis is concerned with a first attempt of "bridging the gap" between classical numerical analysis and probabilistic modeling of

uncertainty. More specifically, this thesis will develop probabilistic numerical methods which attempt to model the numerical approximation uncertainty in the language of probability theory. To this end, we will study the connection of Gaussian process regression to numerical spline models in order to arrive at probabilistic interpretations of classical numerical methods. Conversely, we will interpret some classical methods in light of their probabilistic assumptions to critique or improve upon them. We will investigate how prior information about dynamical systems can be incorporated into methods and how numerical methods fit into a larger chain of scientific computations.

1.1 Outline

This thesis is structured in three major parts. Part ii provides the necessary background material. In Chapter 2, we introduce the mathematical framework of *probabilistic inference* which serves as the *lingua franca* in this manuscript. Chapter 3 gives a short introduction to the theory of ordinary differential equations and the analysis of algorithms for their numerical solution. Prior work on probabilistic methods applied to the numerical solution of ODEs is presented in Chapter 4.

We will present contributions in both theory (Part iii) and applications (Part iv) of probabilistic ODE solvers. Part iii opens with general considerations about probabilistic ODE solvers and formulates the *open questions* in Chapter 5. Based on these, we will describe and analyze our *probabilistic filtering ODE solver* in Chapter 6. Chapter 7 provides an extension of the basic algorithm and tests our reference implementation on a standard benchmark set.

In Part iv, we discuss how probabilistic ODE solvers can provide *novel functionality*. We start by presenting our area of application: tractography (Chapter 8). We show how the estimated solution uncertainty of a probabilistic ODE solver can be used in subsequent computations (Chapter 9) and how uncertainty in previous computations can be used within the ODE solver itself (Chapter 10).

In the final Part v, we will summarize and reflect upon our work in Chapter 11.2.

1.2 Publications

In partial fulfillment of this thesis, research was conducted and published with colleagues. In particular, the following manuscripts have been published after scientific peer-review:

- M. Schober, N. Kasenburg, A. Feragen, P. Hennig, and S. Hauberg. "Probabilistic shortest path tractography in DTI using Gaussian Process ODE solvers." In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*. Springer, 2014

- M. Schober, D. Duvenaud, and P. Hennig. "Probabilistic ODE Solvers with Runge-Kutta Means." In: *Advances in Neural Information Processing Systems (NIPS)* (2014) This work has been selected for full oral presentation.

- M. Schober, S. Särkkä, and P. Hennig. "A probabilistic model for the numerical solution of initial value problems." In: *Statistics and Computing* (2018). DOI: 10.1007/s11222-017-9798-7

I have also contributed major parts in the following contribution of my colleague Søren Hauberg:

- S. Hauberg, M. Schober, M. Liptrot, P. Hennig, and A. Feragen. "A Random Riemannian Metric for Probabilistic Shortest-Path Tractography." In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*. Vol. 18. Springer, Sept. 2015

During an internship at Microsoft Research, Cambridge, I have been able to conduct research on an unrelated problem in depth imaging. This work culminated in the publication of the following peer-reviewed manuscript which will not be included in this thesis:

- M. Schober, A. Adam, O. Yair, S. Mazor, and S. Nowozin. "Dynamic Time-Of-Flight." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017

Part II

Preliminaries

2 Probabilistic inference

This chapter provides the necessary mathematical background on the statistical models applied within this thesis.

2.1 Bayesian regression

This thesis is concerned with the numerical solution of ODEs from the perspective of probabilistic modeling. In *regression analysis*, the goal is to determine an unknown function $y : \mathbb{R}^D \to \mathbb{R}$ from pairs of (noisy) observations $(\boldsymbol{x}_n, z_n), \boldsymbol{x}_n \in \mathbb{R}^D, z_n \approx y(\boldsymbol{x}_n), n = 1, \dots, N$. The inputs $\boldsymbol{x}_n$ are called the *independent variables*, *regressors* or simply *inputs*. It is assumed that the behavior of the *response variables* z_n can be expressed as a conditional expectation $\mathbb{E}[y(\boldsymbol{x}_n) \mid \boldsymbol{x}_n]$ of $\boldsymbol{x}_n$.

regression analysis

independent variables

response variables

The set $\mathcal{D} = \{(\boldsymbol{x}_n, z_n) \mid n = 1, \dots, N\}$ of all available observations is called *training data*. It is used to make a *prediction* $\hat{y}(\tilde{\boldsymbol{x}})$ at a previously unobserved input $\tilde{\boldsymbol{x}}$ or to get hopefully more accurate *estimation* $\hat{y}(\boldsymbol{x}_n)$ for the corrupted value z_n of input $\boldsymbol{x}_n$.

training data

prediction

estimation

To this end, we will apply *Bayesian inference*, wherein all modeling choices and assumptions are encoded by specifying a joint *probabilistic model* for all unknowns and observables. A *prior distribution* $P_{\mathcal{Y}}(y)$ is placed over a set of possible functions $y \in \mathcal{Y}$. A *likelihood* $P_z(z \mid y(\boldsymbol{x}))$ specifies the probability of obtaining z for the input $\boldsymbol{x}$ of y. Bayes' theorem

Bayesian inference

probabilistic model

prior distribution

likelihood

$$P(y(\boldsymbol{x}) \mid \mathcal{D}) = \frac{P_z(z_1, \dots, z_N \mid y(\boldsymbol{x}_1), \dots, y(\boldsymbol{x}_N))\, P_{\mathcal{Y}}(y)}{\int P_z(z_1, \dots, z_N \mid y(\boldsymbol{x}_1), \dots, y(\boldsymbol{x}_N))\, P_{\mathcal{Y}}(y)\, \mathrm{d}y} \tag{1}$$

is used to obtain the *posterior distribution* $P(y(\boldsymbol{x}) \mid \mathcal{D})$, lending the method its name.

posterior distribution

When modeling temporal phenomena, it is often the case that observations $(\boldsymbol{x}_n, z_n)$ become available sequentially, yet it might be necessary to make predictions *online*, i.e., during the *runtime* of the inference algorithm. Other times, it suffices to perform *offline* inference after all data has become available and no time-critical component is waiting for immediate answers.

online

runtime

offline

In the literature, a distinction is drawn between three different posterior distributions that one might be interested in. Let $\mathcal{D}_{[m]} =$

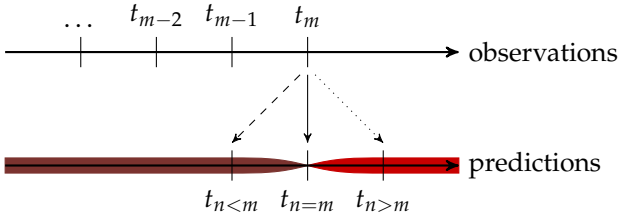

Figure 1: Illustration of the filtering, predictive and smoothing distribution.

$\{(\boldsymbol{x}_n, z_n) \mid n \leq m\}$ be the set of observations up to and including z_m. We are interested in (posterior) conditional distributions of the form

$$P(y(\boldsymbol{x}_n) \mid \mathcal{D}_{[m]}). \tag{2}$$

If $n = m$ in (2), this is called the *filtering distribution* and determining (2) is called the *filtering problem*. The cases $n > m$ and $n < m$ are called the *predictive* and *smoothing distribution*, respectively.

filtering distribution

filtering problem

predictive

smoothing distribution

For the most part, we will be concerned with ODEs obtaining unique, well-defined, deterministic solutions. Yet, we will model the solution with a *random variable (RV)* and we will assign a probability measure over it. The intension of the probability measure here is expressly *not* to describe randomness, but to characterize a lack of complete certainty over its value, akin to the Laplace probability of a die before it's thrown. However, unlike a die, the numerical finite solution will never completely reach the infinite true solution, and as a consequence, some epistemic uncertainty always remains.

random variable (RV)

2.2 Inference in Gaussian probability models

A necessary ingredient to inference are assumptions about the relationships between involved quantities. This thesis builds up on *Gaussian process* regression wherein all variables are related to each other by multivariate normal distributed, or simply Gaussian, RVs. The (multivariate) Gaussian distribution holds a dominant role in statistics:

Gaussian process

1. The Gaussian distribution is a member of the exponential family, and thus, can be completely characterized by its sufficient statistics consisting of a *mean vector* and a *covariance matrix*.

mean vector

covariance matrix

2. Linear and affine transformations of Gaussian distributed RVs are again Gaussian distributed.

3. If two RVs X, Y are jointly Gaussian distributed, the conditional distribution of X given $Y = y$ is normally distributed as well.

Together, these properties lead to efficient algorithms for inference under Gaussian model assumptions.

In the remainder of this section, we present the basic theory of multivariate normal distributions. For details, we refer the reader to Durrett [60, §3] and Tong [211, §3]. A rigorous presentation is given in Bogachev [17].

Denote by $\phi_{\boldsymbol{X}}(\boldsymbol{t}) : \mathbb{R}^D \to \mathbb{R}$, $\phi_{\boldsymbol{X}}(\boldsymbol{t}) = \mathbb{E}[\mathrm{e}^{i\boldsymbol{t}^\intercal \boldsymbol{X}}]$ the *characteristic function* of the q-dimensional random variable $\boldsymbol{X}$. A *random variable (RV)* $\boldsymbol{X} \in \mathbb{R}^D$ is said to have the *multivariate Gaussian distribution* $P(\boldsymbol{X}) = \mathcal{N}(\boldsymbol{X}; \boldsymbol{m}, \boldsymbol{C})$ with mean vector $\boldsymbol{m} \in \mathbb{R}^D$ and positive (semi-) definite covariance matrix $\boldsymbol{C} \in \mathbb{R}^{D\times D}$, if its characteristic function is given by

characteristic function

random variable (RV)

Gaussian distribution

$$\phi_{\boldsymbol{X}}(t) = \exp(i\boldsymbol{t}^\intercal \boldsymbol{m} - \frac{1}{2}\boldsymbol{t}^\intercal \boldsymbol{C}\boldsymbol{t}). \tag{3}$$

If $\boldsymbol{C}$ has full rank, the distribution of $\boldsymbol{X}$ has a *probability density function (PDF)* $p : \mathbb{R}^D \to \mathbb{R}$, i.e., the probability of $\boldsymbol{X}$ taking a value in the set E can be written as

probability density function (PDF)

$$P(\boldsymbol{X} \in E) = \int_E p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x},$$
$$p(\boldsymbol{x}) = (2\pi)^{\frac{q}{2}}|\boldsymbol{C}|^{\frac{1}{2}} \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^\intercal \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{m})).$$

The advantage of working with (3) is that it is also valid for covariance matrices $\boldsymbol{C}$ of rank $D' < D$. In particular, we will consider deterministic variables $\overline{\boldsymbol{x}} = \overline{\boldsymbol{m}}$ as degenerate random variables with *Dirac measure* $\delta_{\overline{\boldsymbol{m}}}$

Dirac measure

$$\delta_{\overline{\boldsymbol{m}}}(\overline{\boldsymbol{x}}) = \mathcal{N}(\overline{\boldsymbol{x}}; \overline{\boldsymbol{m}}, \boldsymbol{0}) = \begin{cases} 1 & \text{if } \overline{\boldsymbol{x}} = \overline{\boldsymbol{m}}, \\ 0 & \text{otherwise,} \end{cases}$$

since $\phi_{\overline{\boldsymbol{x}}}(\boldsymbol{t}) = \mathrm{e}^{i\boldsymbol{t}^\intercal \overline{\boldsymbol{m}}}$.

Let $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{X} + \boldsymbol{b}, \boldsymbol{H} \in \mathbb{R}^{E\times D}, \boldsymbol{b} \in \mathbb{R}^E$ be an affine transformation of the RV $\boldsymbol{X}$ and denote the resulting RV $\boldsymbol{Y}$. The *joint distribution* $P(\boldsymbol{X}, \boldsymbol{Y})$ of the RV $\boldsymbol{Z} = (\boldsymbol{X}, \boldsymbol{Y})^\intercal$ is given by

joint distribution

$$P(\boldsymbol{Z}) = \mathcal{N}\left[\begin{pmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{pmatrix}; \begin{pmatrix}\boldsymbol{m}\\ \boldsymbol{H}\boldsymbol{m}+\boldsymbol{b}\end{pmatrix}, \begin{pmatrix}\boldsymbol{C} & \boldsymbol{C}\boldsymbol{H}^\intercal\\ \boldsymbol{H}\boldsymbol{C} & \boldsymbol{H}\boldsymbol{C}\boldsymbol{H}^\intercal\end{pmatrix}\right]. \tag{4}$$

which follows from inserting the transformation into (3) and reorganization of terms. Conversely, let the joint system $(\boldsymbol{X}, \boldsymbol{Y})^\intercal$ have the joint distribution

$$P(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}\left[\begin{pmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{pmatrix}; \begin{pmatrix}\boldsymbol{m}_x\\ \boldsymbol{m}_y\end{pmatrix}, \begin{pmatrix}\boldsymbol{C}_{xx} & \boldsymbol{C}_{xy}\\ \boldsymbol{C}_{xy}^\intercal & \boldsymbol{C}_{yy}\end{pmatrix}\right].$$

Then the conditional distribution $P(\boldsymbol{X} \mid \boldsymbol{Y} = \boldsymbol{y})$ of $\boldsymbol{X}$ given $\boldsymbol{Y} = \boldsymbol{y}$ is also a multivariate Gaussian

$$P(\boldsymbol{X} \mid \boldsymbol{Y} = \boldsymbol{y}) = \mathcal{N}(\boldsymbol{X}; \boldsymbol{m}_x + \boldsymbol{C}_{xy}\boldsymbol{C}_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{m}_y), \boldsymbol{C}_{xx} - \boldsymbol{C}_{xy}\boldsymbol{C}_{yy}^{-1}\boldsymbol{C}_{xy}^{\intercal}). \quad (5)$$

If $\boldsymbol{C}_{yy}$ is singular, $\boldsymbol{C}_{yy}^{-1}$ denotes the pseudo-inverse of $\boldsymbol{C}_{yy}$ in (5). Eqs. (4), (5) imply that Bayes' theorem (1) has an analytic solution which can be expressed solely in terms of linear algebra, thus fulfilling the promises made at the beginning of this chapter.

2.3 Types of Gaussian assumptions

We have built up the necessary mathematical background to consider various types of model assumptions within the family of Gaussian regression. We will now present basic models of Gaussian regression. For a more detailed introduction, we recommend the books by Rasmussen and Williams [176] and Lifshits [136].

The most basic model is *Gaussian linear regression*

Gaussian linear regression

$$z_n = \boldsymbol{w}^{\intercal}\boldsymbol{x}_n + \nu_n, \quad (6)$$

where $\boldsymbol{w} \in \mathbb{R}^D$ is a parameter vector of weights and $\nu_n \sim \mathcal{N}(0, \sigma^2)$ are observation errors. Positing a prior distribution on the unknown weights parameter $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$, we can use Equations (4) and (5) to obtain the *(predictive) posterior conditional distribution*

(predictive) posterior conditional distribution

$$\begin{aligned} P(\boldsymbol{w} \mid \mathcal{D}) &= \mathcal{N}(\underline{\boldsymbol{m}}_{\mathcal{D}}, \underline{\boldsymbol{C}}_{\mathcal{D}}), \\ \underline{\boldsymbol{m}}_{\mathcal{D}} &= \boldsymbol{m} + \boldsymbol{C}\boldsymbol{\mathcal{X}}^{\intercal}(\boldsymbol{\mathcal{X}}\boldsymbol{C}\boldsymbol{\mathcal{X}}^{\intercal} + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{z} - \boldsymbol{\mathcal{X}}\boldsymbol{m}), \\ \underline{\boldsymbol{C}}_{\mathcal{D}} &= \boldsymbol{C} - \boldsymbol{C}\boldsymbol{\mathcal{X}}^{\intercal}(\boldsymbol{\mathcal{X}}\boldsymbol{C}\boldsymbol{\mathcal{X}}^{\intercal} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{\mathcal{X}}\boldsymbol{C}, \end{aligned} \quad (7)$$

where $\boldsymbol{I}$ denotes the identity matrix. In (7), we have written the training set $\mathcal{D}$ in a $N \times q$ matrix $(\boldsymbol{\mathcal{X}})_{nq}$ and a N-dimensional vector $\boldsymbol{z} = (z_1, \ldots, z_N)^{\intercal}$, accordingly.

The matrix $\boldsymbol{G} = \boldsymbol{\mathcal{X}}\boldsymbol{C}\boldsymbol{\mathcal{X}}^{\intercal} + \sigma^2\boldsymbol{I}$ of covariances $(\boldsymbol{G})_{ij} = \mathrm{cov}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is also called the *Gram matrix* [176, §4.1]. Computing the inverse of the Gram matrix is often the computational bottleneck of Gaussian inference. Occassionally, we will encounter models where the measurement noise ν_n is independent, but non-identically distributed, according to $\nu_n \sim \mathcal{N}(0, \sigma_n^2)$. In these cases, we will write $(\boldsymbol{x}_n, z_n, \sigma_n^2)$ for observations with non-identically distributed noise and the Gram matrix consequently turns into $\boldsymbol{G} = \boldsymbol{\mathcal{X}}\boldsymbol{C}\boldsymbol{\mathcal{X}}^{\intercal} + \mathrm{diag}(\sigma_1^2, \ldots, \sigma_N^2)$, where $\mathrm{diag}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k)$ denotes the (block) diagonal matrix with entries $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k$.

Gram matrix

In many cases, a model in the form of(6) will not be sufficient in practice as most relationships are nonlinear. In these cases, one can try to design a nonlinear transformation of the inputs $\Phi : \mathbb{R}^D \to$

$\mathbb{R}^E$ and perform linear regression on the transformed variables $y = \boldsymbol{w}^\intercal \Phi(\boldsymbol{x})$. The problem with this approach is that one needs to find an appropriate transformation manually which might not always be possible.

Another option is to specify a *nonparametric model* in form of a (nonlinear) *Gaussian process (GP)* $\mathcal{GP}(Y; \mu, k)$ with mean function $\mu : \mathbb{R}^D \to \mathbb{R}$ and covariance kernel $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$. This assumes that the function $Y(\boldsymbol{x})$ lies in the sample space of the GP which is closely related to the *reproducing kernel Hilbert space (RKHS)* $\mathcal{H}_k$ of k (for details, see [143]). Formally, the GP is an infinite collection of RVs $Y_{\boldsymbol{x}}$ such that any finite subset $(Y_{\boldsymbol{x}_1}, \dots, Y_{\boldsymbol{x}_N})^\intercal$ is jointly Gaussian distributed with mean vector $\mu(\boldsymbol{\mathcal{X}}) = (\mu(\boldsymbol{x}_1), \dots, \mu(\boldsymbol{x}_N))^\intercal$ and covariance matrix $k(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}})$ with entries $(k(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}}))_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j = 1, \dots, N$.

nonparametric model

Gaussian process (GP)

reproducing kernel Hilbert space (RKHS)

Valid covariance kernels are *positive (semi-) definite (p.s.d.)* functions, i.e.,

positive (semi-) definite (p.s.d.)

$$\sum_{n,m=1}^{N} a_n a_m k(\boldsymbol{x}, \boldsymbol{x}') \geq 0 \quad \forall N \in \mathbb{N}, a_n, a_m \in \mathbb{R}, \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^D.$$

An example for such a function is the *squared exponential (SE)* kernel

squared exponential (SE)

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\left|\left|\boldsymbol{x} - \boldsymbol{x}'\right|\right|^2 (2\lambda^2)^{-1}), \quad \lambda \in \mathbb{R}_{>0}$$

with lengthscale parameter λ.

As before, denote by $\mathcal{D} = \{(\boldsymbol{x}_n, z_n) \,|\, n = 1, \dots, N\}$ with $z_n = Y(\boldsymbol{x}_n) + \nu_n, \nu_n \sim \mathcal{N}(0, \sigma^2)$ i.i.d., a data set of training points. The posterior process distribution is given as

$$\begin{aligned} P(Y(\boldsymbol{x}) \,|\, \mathcal{D}) &= \mathcal{GP}(Y(\boldsymbol{x}); \underline{\mu}_{\mathcal{D}}(\boldsymbol{x}), \underline{k}_{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{x}')) \\ \underline{\mu}_{\mathcal{D}}(\boldsymbol{x}) &= \mu(\boldsymbol{x}) + k(\boldsymbol{x}, \boldsymbol{\mathcal{X}})(k(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}}) + \sigma^2 \boldsymbol{I})^{-1}(\boldsymbol{z} - \mu(\boldsymbol{\mathcal{X}})), \quad (8) \\ \underline{k}_{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{x}') &= k(\boldsymbol{x}, \boldsymbol{x}') - k(\boldsymbol{x}, \boldsymbol{\mathcal{X}})(k(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}}) + \sigma^2 \boldsymbol{I})^{-1} k(\boldsymbol{\mathcal{X}}, \boldsymbol{x}')), \end{aligned}$$

with the notation as in (7).

Note that in the general setting of an infinite dimensional Hilbert space, the mathematical concepts of a posterior or conditioning are subject to measure-theoretic complexities [205, 71]. In this work, we will only be dealing with finitely many Gaussian RVs of the discretized underlying process and, thus, we do not have to worry about these restrictions in practice.

Analogous to the finite dimensional case, Gaussian processes are closed under linear maps $\mathcal{L} : \mathcal{H} \to \mathcal{H}'$ [163, §10], [17, §2.10]

$$P(Y, \mathcal{L}Y) = \mathcal{GP}\left[\begin{pmatrix} Y \\ \mathcal{L}Y \end{pmatrix}; \begin{pmatrix} \mu \\ \mathcal{L}\mu \end{pmatrix}, \begin{pmatrix} k & k\mathcal{L}^\intercal \\ \mathcal{L}k & \mathcal{L}k\mathcal{L}^\intercal \end{pmatrix}\right]$$

where $\mathcal{L}^\intercal$ is the application of $\mathcal{L}$ to the second argument of k. In particular, if $\mathcal{Y}$ contains the set $\mathcal{C}^q(\mathbb{R}^D, \mathbb{R})$ of q-times continuously differentiable functions $y : \mathbb{R}^D \to \mathbb{R}$, then

$$P(Y(\boldsymbol{x}), \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}} Y(\boldsymbol{x})) = \mathcal{GP}\left[\begin{pmatrix} Y(\boldsymbol{x}) \\ \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}} Y(\boldsymbol{x}) \end{pmatrix}; \begin{pmatrix} \mu(\boldsymbol{x}) \\ \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}} \mu(\boldsymbol{x}) \end{pmatrix}, \begin{pmatrix} k(\boldsymbol{x}, \boldsymbol{x}') & \frac{\partial}{\partial \boldsymbol{x}'} k(\boldsymbol{x}, \boldsymbol{x}') \\ \frac{\partial}{\partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{x}') & \frac{\partial^2}{\partial \boldsymbol{x} \partial \boldsymbol{x}'} k(\boldsymbol{x}, \boldsymbol{x}') \end{pmatrix}\right] \tag{9}$$

is the joint distribution over $Y(\boldsymbol{x})$ and its derivative $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}} Y(\boldsymbol{x})$ and similarly for higher derivatives. Conversely, it can be shown that $\mathcal{Y}$ contains $\mathcal{C}^q(\mathbb{R}^D, \mathbb{R})$, if $\mu \in \mathcal{C}^q(\mathbb{R}^D, \mathbb{R})$ and k is $2q$-times continuously partially differentiable. Eq. (9) can be used in conjunction with (8) to use derivative observations $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}} y$ for inference of function values y. This discussion is a bit overly simplistic, but it suffices for our practial applications. For details, we refer the reader to Adler [2, §3.3].

In some applications, we will require online predictions, i.e., fast access to the filtering and predictive distribution. This is not feasible with the model (8), as the Gram matrix $\boldsymbol{G} \in \mathbb{R}^{N \times N}$ grows with the number of observations and its inversion is of cubic cost $\mathcal{O}(N^3)$.

In the parametric model (6), this issue can be solved by using the *Shermann–Morrison–Woodbury formula* [80], sometimes also called the *matrix inversion lemma*,

Shermann–Morrison–Woodburry formula

matrix inversion lemma

$$(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}$$

to find expressions which are less taxing to compute

$$\begin{aligned} \underline{\boldsymbol{m}}_{\mathcal{D}} &= (\boldsymbol{C}^{-1} + \frac{1}{\sigma^2}\boldsymbol{\mathcal{X}}^\intercal\boldsymbol{\mathcal{X}})^{-1}(\frac{1}{\sigma^2}\boldsymbol{\mathcal{X}}^\intercal\boldsymbol{z} + \boldsymbol{C}^{-1}\boldsymbol{m}) \\ \underline{\boldsymbol{C}}_{\mathcal{D}} &= (\boldsymbol{C}^{-1} + \frac{1}{\sigma^2}\boldsymbol{\mathcal{X}}^\intercal\boldsymbol{\mathcal{X}})^{-1}. \end{aligned} \tag{10}$$

Eq. (10) only requires the inversion of two $D \times D$ matrices. Assuming the number of dimensions D is much less than the number of observations N, we find these expressions much faster to evaluate.

Note that this can only be applied to parametric regression whose fixed number of parameters cannot be fitted to complicated data sets. However, it is still possible to achieve fast inference for some nonparametric models. The key lies in understanding where the computational benefit comes from—the extra knowledge of structure of the involved computations. In the next section, we will describe a nonparametric model for one-dimensional inputs that obtains a fast inference algorithm.

2.4 Stochastic differential equations and Kalman filtering

We still require a model that is both expressive enough for complicated data sets as well as structured enough to yield fast inference algorithms for online prediction. In this section, we will present such a model based on *stochastic differential equations (SDEs)*. However, we will need the additional assumption that the input space $\mathbb{X} \subset \mathbb{R}^D$ is one-dimensional, i.e., $q = 1$. It will turn out that this is no restriction for the application at hand, since we will mostly deal with one-dimensional problems of time. To highlight this restriction, we change the notation of the input variables $\boldsymbol{x}_n$ to $t_n \in \mathbb{T} \subset \mathbb{R}$. More information on this subject is given in Jazwinski [102].

stochastic differential equations (SDEs)

Gaussian processes are a particular class of stochastic processes which are completely specified by their first and second moments. Another special case of stochastic processes are Markov processes.

Let $t > s_N \geq \cdots \geq s_1$ be a finite collection of elements from the ordered set $\mathbb{T}$. A vector-valued stochastic process $(\boldsymbol{X}_t)_{t\in\mathbb{T}}$ is a *Markov process* if it satisfies the Markov property

Markov process

$$P(\boldsymbol{X}_t \mid \boldsymbol{X}_{s_n}, n = 1, \ldots, N) = P(\boldsymbol{X}_t \mid \boldsymbol{X}_{s_N}), \tag{11}$$

that is, the distribution of $\boldsymbol{X}_t$ conditioned on any set of previous RVs $\{\boldsymbol{X}_{s_n}\}$ only depends on the most recent RV $\boldsymbol{X}_{s_N}$. A colloquial description is to say "the future is independent of the past given the present". Technically, Eq. (11) is a simplification of the full measure-theoretic description which can be found, e.g., in Karatzas and Shreve [114, §2.5].

One particular class of models that gives rise to Markov processes are *linear time-invariant (LTI) stochastic differential equations (SDEs)* of the form

linear time-invariant (LTI) stochastic differential equations (SDEs)

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{F}\,\boldsymbol{X}_t\,\mathrm{d}t + \boldsymbol{L}\,\mathrm{d}W_t, \tag{12}$$

where $(\boldsymbol{X}_t)_{t\in\mathbb{T}}$ is a stochastic process mapping to the so-called *state space* $\mathbb{R}^q$. The parameters $\boldsymbol{F} \in \mathbb{R}^{q\times q}$ and $\boldsymbol{L} \in \mathbb{R}^q$ are the *drift (or feedback) matrix* and (continuous) *diffusion matrix*, respectively. $\mathrm{d}W_t$ is the increment of a Wiener process with intensity σ^2, i.e., $\mathrm{d}W_t \sim \mathcal{N}(0, \sigma^2\,\mathrm{d}t)$.

state space

drift (or feedback) matrix

diffusion matrix

Conditioning on (random) initial conditions $\boldsymbol{X}_{t_*}$ at a starting time t_* of the process, the solution of Eq. (12) has the analytic form

$$\boldsymbol{X}_t = \mathrm{e}^{\boldsymbol{F}\Delta^t_{t_*}}\boldsymbol{X}_{t_*} + \int_{t_*}^{t} \mathrm{e}^{\boldsymbol{F}\Delta^t_\tau}\,\boldsymbol{L}\,\mathrm{d}W(\tau),$$

where $\mathrm{e}^{\boldsymbol{F}\Delta^t_{t'}} = \sum_{k=0}^{\infty}[\boldsymbol{F}\Delta^t_{t'}]^k[k!]^{-1}$ is the matrix exponential of $\boldsymbol{F}\Delta^t_{t'}$ and $\Delta^t_s = t - s$.

If $\boldsymbol{X}_{t_*} \sim \mathcal{N}(\boldsymbol{m}_*, \boldsymbol{C}_*)$, then the distribution of $\boldsymbol{X}_t$ remains Gaussian for all t by linearity and its statistics can be computed explicitly [75, 181] via

$$\begin{aligned} \boldsymbol{m}_t =& \mathbb{E}(\boldsymbol{X}_t) = \mathrm{e}^{\boldsymbol{F}\Delta_{t_*}^t} \boldsymbol{m}_* \\ \operatorname{cov}(\boldsymbol{X}_t, \boldsymbol{X}_{t'}) =& \mathrm{e}^{\boldsymbol{F}\Delta_{t_*}^t} \boldsymbol{C}_* (\mathrm{e}^{\boldsymbol{F}\Delta_{t_*}^{t'}})^\intercal \\ &+ \underbrace{\int_{t_*}^{\min(t,t')} \mathrm{e}^{\boldsymbol{F}\Delta_\tau^t} \boldsymbol{L}\sigma^2\boldsymbol{L}^\intercal (\mathrm{e}^{\boldsymbol{F}\Delta_\tau^{\tilde{t}}})^\intercal d\tau}_{=\boldsymbol{Q}_{t_*}(t,t')}. \end{aligned} \tag{13}$$

This implies that linear SDEs with a Gaussian initial distribution $P(\boldsymbol{X}_{t_*}) = \mathcal{N}(\boldsymbol{m}_*, \boldsymbol{C}_*)$ define a *Gauss-Markov process*, i.e., a stochastic process that is both Gaussian and Markovian. Examples of Gauss-Markov processes include the *Wiener process* $\mathcal{GP}(\boldsymbol{0}, \min(t,t'))$ and the (one-dimensional) *Matérn processes*, e.g., $\mathcal{GP}(\boldsymbol{0}, k_{3/2}(t,t'))$ where $k_{3/2}(t,t') = (1+\sqrt{3}\lambda^{-1}\,|t-t'|)\exp(-\sqrt{3}\lambda^{-1}\,|t-t'|)$.

Gauss-Markov process

Wiener process

Matérn processes

Denote by $\boldsymbol{A}(h) = \mathrm{e}^{\boldsymbol{F}\Delta_t^{t+h}}$ the *transition matrix* of step size h and $\boldsymbol{Q}(h) = \boldsymbol{Q}_t(t+h, t+h)$ the (discrete) *diffusion matrix* of step size h, respectively. For LTI SDE systems, $\boldsymbol{A}(h)$ and $\boldsymbol{Q}(h)$ fulfill matrix-valued differential equations which can be solved analytically via matrix fraction decomposition [75, 181]. Define

transition matrix

diffusion matrix

$$\boldsymbol{\Phi}(h) = \begin{pmatrix} \boldsymbol{\Phi}_{11}(h) & \boldsymbol{\Phi}_{12}(h) \\ \boldsymbol{\Phi}_{21}(h) & \boldsymbol{\Phi}_{22}(h) \end{pmatrix} = \exp\left\{ \begin{pmatrix} \boldsymbol{F} & \sigma^2\boldsymbol{L}\boldsymbol{L}^\intercal \\ \boldsymbol{0} & -\boldsymbol{F}^\intercal \end{pmatrix} h \right\}.$$

Then, matrix $\boldsymbol{A}(h)$ and $\boldsymbol{Q}(h)$ are given by

$$\boldsymbol{A}(h) = \exp(\boldsymbol{F}h) \qquad \boldsymbol{Q}(h) = \boldsymbol{\Phi}_{12}(h)\boldsymbol{\Phi}_{22}^{-1}(h). \tag{14}$$

$\boldsymbol{\Phi}_{22}^{-1}(h)$ can be computed efficiently: from the two properties of the matrix exponential, $\exp(\boldsymbol{M})^{-1} = \exp(-\boldsymbol{M})$ and $\exp(\boldsymbol{M}^\intercal) = \exp(\boldsymbol{M})^\intercal$, it follows that $\boldsymbol{\Phi}_{22}^{-1}(h) = \boldsymbol{A}(h)^\intercal$ and therefore $\boldsymbol{Q}(h) = \boldsymbol{\Phi}_{12}(h)\boldsymbol{A}(h)^\intercal$. In the following, it will be beneficial to write $\boldsymbol{Q}(h)$ as $\boldsymbol{Q}(h)^\intercal = \boldsymbol{A}(h)\boldsymbol{\Phi}_{12}(h)^\intercal$, which is valid since $\boldsymbol{Q}(h)$ is a covariance matrix and, therefore, symmetric.

The importance of Markov processes stems from the fact that the joint distribution over a finite subset $\{\boldsymbol{X}_{t_n} \mid n = 0, \ldots, N\}$ factorizes

$$P(\boldsymbol{X}_{t_0}, \boldsymbol{X}_{t_1}, \ldots, \boldsymbol{X}_{t_N}) = P(\boldsymbol{X}_{t_0}) \prod_{n=1}^{N} P(\boldsymbol{X}_{t_n} \mid \boldsymbol{X}_{t_{n-1}})$$

which can be exploited to speed up inference computations.

Assume as in the previous section a data set $(t_n, z_n), n = 0, \ldots, N$ and a linear relationship

$$z_n = \boldsymbol{H}\boldsymbol{X}_{t_n} + \nu_n, \quad \nu_n \sim \mathcal{N}(0, \boldsymbol{R}_n)$$

between the observations z_n and the (hidden) state $\boldsymbol{X}_{t_n}$ at time t_n perturbed with *observation noise* ν_n. The matrix $\boldsymbol{H} \in \mathbb{R}^{1\times q}$ is called the *observation matrix* of the sytem.

observation noise

observation matrix

Let us first consider the filtering problem. We will factorize Eq. (2) appropriately

$$\begin{aligned} P(\boldsymbol{X}_{t_n} \mid z_{[n]}) &= P(z_n \mid \boldsymbol{X}_{t_n}) P(\boldsymbol{X}_{t_n} \mid z_{[n-1]}) \\ &= P(z_n \mid \boldsymbol{X}_{t_n}) P(\boldsymbol{X}_{t_n} \mid \boldsymbol{X}_{t_{n-1}}) P(\boldsymbol{X}_{t_{n-1}} \mid z_{[n-1]}) \end{aligned}$$

to arrive at a recursive scheme known as the Kalman filter [113, 112]:

$$\begin{aligned} P(\boldsymbol{X}_{t_n} \mid z_{[n-1]}) &= \mathcal{N}(\boldsymbol{m}_{t_n}^-, \boldsymbol{C}_{t_n}^-) \\ \boldsymbol{m}_{t_n}^- &= \boldsymbol{A}(h_n)\boldsymbol{m}_{t_{n-1}}, && \text{(15a)} \\ \boldsymbol{C}_{t_n}^- &= \boldsymbol{A}(h_n)\boldsymbol{C}_{t_{n-1}}\boldsymbol{A}(h_n)^\intercal + \boldsymbol{Q}(h_n) && \text{(15b)} \end{aligned}$$

and

$$\begin{aligned} P(\boldsymbol{x}_{t_n} \mid z_{[n]}) &= \mathcal{N}(\boldsymbol{m}_{t_n}, \boldsymbol{C}_{t_n}) \\ \lambda_n &= z_n - \boldsymbol{H}\boldsymbol{m}_{t_n}^-, && \text{(16a)} \\ \boldsymbol{K}_n &= \boldsymbol{C}_{t_n}^-\boldsymbol{H}^\intercal[\boldsymbol{H}\boldsymbol{C}_{t_n}^-\boldsymbol{H}^\intercal + \boldsymbol{R}_n]^{-1}, && \text{(16b)} \\ \boldsymbol{m}_{t_n} &= \boldsymbol{m}_{t_n}^- + \boldsymbol{K}_n\lambda_n, && \text{(16c)} \\ \boldsymbol{C}_{t_n} &= \boldsymbol{C}_{t_n}^- - \boldsymbol{K}_n[\boldsymbol{H}\boldsymbol{C}_{t_n}^-\boldsymbol{H}^\intercal + \boldsymbol{R}_n]\boldsymbol{K}_n^\intercal, && \text{(16d)} \end{aligned}$$

where $h_n = t_n - t_{n-1}$. Equations (15a) and (15b) are referred to as the *prediction step* and Equations (16a) through (16d) are referred to as the *update step*. The newly defined variables λ_n and $\boldsymbol{K}_n$ are the *residual* and the *Kalman gain*, respectively. Equations (15a), (15b) follow directly from (13), whereas Equations (16a)–(16d) can be derived from repeated applications of Eqs. (4) and (5).

The prediction problem $P(\boldsymbol{X}_t \mid z_{[n]})$, $t > t_n$, is particularly simple, once the filtering problem has been solved, using the Markov property (11) and the decomposition $P(\boldsymbol{X}_t \mid z_{[n]}) = P(\boldsymbol{X}_t \mid \boldsymbol{X}_{t_n})P(\boldsymbol{X}_{t_n} \mid z_{[n]})$.

The smoothing problem can be solved by updating the filtered distributions $p(\boldsymbol{X}_{t_n} \mid z_{[n]})$ in a backward sweep with the *Rauch-Tung-Striebel* smoothing equations [177, 181, 182] which will be given below.

Type	Form	# parameters	Cost
Linear parametric	$y = \boldsymbol{w}^\intercal \boldsymbol{x}$	D	$\mathcal{O}(D^3)$
Nonparametric	$y = f(\boldsymbol{x}), f \sim \mathcal{GP}(f; \mu, k)$	$\propto N$	$\mathcal{O}(N^3)$
Markov	$y = \boldsymbol{H}\boldsymbol{X}, \boldsymbol{X} \sim \mathrm{d}\boldsymbol{X} = \boldsymbol{F}\boldsymbol{X}\,\mathrm{d}t + \boldsymbol{L}\,\mathrm{d}W$	$\propto N$	$\mathcal{O}(Nq^3)$

Table 1: Gaussian regression models

Define $\boldsymbol{m}^s_{t_N} = \boldsymbol{m}_{t_N}, \boldsymbol{C}^s_{t_N} = \boldsymbol{C}_{t_N}$. The posterior smoothing distribution can be computed recursively as

$$
\begin{aligned}
P(\boldsymbol{X}_{t_{n-1}} \mid z_{[N]}) &= \mathcal{N}(\boldsymbol{m}^s_{t_{n-1}}, \boldsymbol{C}^s_{t_{n-1}}) \\
\boldsymbol{m}^-_{t_n} &= \boldsymbol{A}(h_n)\boldsymbol{m}_{t_{n-1}} && \text{(17a)} \\
\boldsymbol{C}^-_{t_n} &= \boldsymbol{A}(h_n)\boldsymbol{C}_{t_{n-1}}\boldsymbol{A}(h_n)^\intercal + \boldsymbol{Q}(h_n) && \text{(17b)} \\
\boldsymbol{G}_{t_{n-1}} &= \boldsymbol{C}_{t_{n-1}}\boldsymbol{A}(h_n)^\intercal(\boldsymbol{C}^-_{t_n})^{-1} && \text{(17c)} \\
\boldsymbol{m}^s_{t_{n-1}} &= \boldsymbol{m}_{t_{n-1}} + \boldsymbol{G}_{t_{n-1}}(\boldsymbol{m}^s_{t_n} - \boldsymbol{m}^-_{t_n}) && \text{(17d)} \\
\boldsymbol{C}^s_{t_{n-1}} &= \boldsymbol{C}_{t_{n-1}} + \boldsymbol{G}_{t_{n-1}}(\boldsymbol{C}^s_{t_n} - \boldsymbol{C}^-_{t_n})\boldsymbol{G}^\intercal_{t_{n-1}}. && \text{(17e)}
\end{aligned}
$$

A good description of sampling strategies for this model as well as further relevant references can be found in Doucet [59].

The Kalman filter Equations (16) and RTS smoother Equations (17) reveal that inference in LTI SDEs can be performed at cost linear in the number of observations $\mathcal{O}(Nq^3)$. And yet, these systems also being Gaussian processes, we know that equivalent expressions of the form (8) must exist. In the remainder of this section, we will show how this speed-up can be explained in GP notation.

Define the following block matrices:

$$
\begin{aligned}
(\boldsymbol{A}(\boldsymbol{t}))_{mn} &= \mathbb{1}_{m \geq n}\boldsymbol{A}(\Delta^{t_m}_{t_n}) \\
(\boldsymbol{Q}(\boldsymbol{t}))_{mn} &= \begin{cases} \boldsymbol{A}(\Delta^{t_1}_{t_1 - t_*})\boldsymbol{C}_*\boldsymbol{A}(\Delta^{t_1}_{t_1-t_*})^\intercal + \boldsymbol{Q}(\Delta^{t_1}_{t_1-t_*}) & m = n = 1 \\ \boldsymbol{Q}(\Delta^{t_{n+1}}_{t_n}) & m = n > 1 \\ \boldsymbol{0} & m \neq n. \end{cases}
\end{aligned}
$$

With these definitions, the full covariance matrix is given by (Grigorievskiy, Lawrence, and Särkkä [77])

$$
k(\boldsymbol{X}_{\boldsymbol{t}}, \boldsymbol{X}_{\boldsymbol{t}}) = \boldsymbol{A}(\boldsymbol{t})\boldsymbol{Q}(\boldsymbol{t})\boldsymbol{A}(\boldsymbol{t})^\intercal
$$

and we also find an analytic solution of the inverse matrix since

$$(\boldsymbol{A}(\boldsymbol{t}))^{-1}_{mn} = \begin{cases} \boldsymbol{I} & m = n \\ -\boldsymbol{A}(\Delta^{t_n}_{t_{n-1}}) & m+1 = n \\ \boldsymbol{0} & \text{else} \end{cases}$$

$$(\boldsymbol{Q}(\boldsymbol{t}))^{-1}_{mn} = \begin{cases} [\boldsymbol{A}(\Delta^{t_1}_{t_1-t_*})\boldsymbol{C}_*\boldsymbol{A}(\Delta^{t_1}_{t_1-t_*})^\intercal + \boldsymbol{Q}(\Delta^{t_1}_{t_1-t_*})]^{-1} & m = n = 1 \\ \boldsymbol{Q}(\Delta^{t_{n+1}}_{t_n})^{-1} & m = n > 1 \\ \boldsymbol{0} & m \neq n. \end{cases}$$

In particular, $k(\boldsymbol{X_t}, \boldsymbol{X_t})^{-1} = \boldsymbol{A}(\boldsymbol{t})^{-\intercal}\boldsymbol{Q}(\boldsymbol{t})^{-1}\boldsymbol{A}(\boldsymbol{t})^{-1}$ is a banded matrix.

Since we have an analytic solution of $k(\boldsymbol{X_t}, \boldsymbol{X_t})^{-1}$, we do not require a numerical procedure to compute its inverse and the matrix-vector multiplications can be executed immediately. However, observe that we had to introduce the latent states $(\boldsymbol{X}_{t,i})_{t\in\mathbb{T}}, i = 1, \ldots, q$ to make the algebra truely equivalent.

In particular, recall that the observations $z_n = \boldsymbol{H}\boldsymbol{X}_n$ might be lower-dimensional projections of the full state-space such that $\boldsymbol{H}$ might not be invertible in which case the necessary inversion $[(\boldsymbol{H} \otimes \boldsymbol{I})K(\boldsymbol{H} \otimes \boldsymbol{I})^\intercal]^{-1}$ does not simply factor into $[(\boldsymbol{H} \otimes \boldsymbol{I})^\intercal]^{-1}K^{-1}[\boldsymbol{H} \otimes \boldsymbol{I}]^{-1}$.

2.5 Sampling and Monte Carlo methods

The previous chapters have presented algebraic rules for the efficient manipulation of Gaussian distributions in inference problems. Inference requires *realizations* of the involved random variables to draw conclusions. This section presents the artificial generation of *samples* which can be used as a tool to understanding modeling assumptions or to approximate intractable computations.

realizations

samples

Samples, or *draws*, are realizations of a (generative) probability model. For example, for a six-sided die a sample would be one of the elements $\{1, 2, 3, 4, 5, 6\}$. Samples are expected to appear with a relative frequency in a larger population according to their probability distribution. When throwing two dice, the sum of their eyes should more frequently be 7 than 2 or 3.

A collection of *independent, identically distributed (i.i.d.)* samples $X_n \sim P(X), n = 1, \ldots, N$, are considered to be a good representative for the distribution as a whole, in particular, when N is large. The *empirical distribution* defined by its cumulative distribution function $F_N(x) = N^{-1}\sum_{n=1}^{N} \mathbb{1}_{X_n \leq x}$ converges almost surely to the cumulative

independent, identically distributed (i.i.d.)

empirical distribution

distribution function F of P for all x. There exist various characterizations of the convergence speed.

Although computers are deterministic calculators, they can act to be *(pseudo-) random number generators (RNGs)*. A RNG is a deterministic method of producing a sequence of numbers that pass many tests for randomness. Typically, random numbers u_i are generated from a uniform distribution $u_i \sim (0, 1)$ over the open unit interval. From these, almost any desired target distribution can be derived. For details, we refer the interested reader to Gentle [70].

(pseudo-) random number generators (RNGs)

In Gaussian process models, both samples from the prior as well as the predictive posterior distribution serve an important role as a *visualization of the correlation*. Smoothness in the displayed mean or marginal variance might give a wrong impression of sample-path smoothness or the level of variation within the marginal variance.

In cases of very high-dimensional distributions, there is much more room for variations and thus a bigger sample size is required to capture the full variablitiy. One particular case of a high-dimensional distributions are random functions. However, drawing many samples will not lead to better visualization as more simultaneous realizations will merely clutter diagrams. In this situation, it is helpful to provide animated videos of continuously varying samples that capture the variability in the added dimension of time. Two ways of producing such animations are presented in Hennig [90] and Skilling [199].

Random numbers are also used in *Monte Carlo (MC)* methods. MC methods describe quantities of interest as expectations or other significant statistics of a particularly designed probability distribution. Then the empirical distribution can stand in for the intractable target computation and it can be analyzed with tools from statistics. The most common example is quadrature where an integral $\int_{\mathbb{D}} f(x)\, \mathrm{d}x$ over a closed domain $\mathbb{D}$ is replaced by the empirical expectation $N^{-1} \sum_{n=1}^{N} f(u_n)$ over N samples drawn i.i.d. $u_n \sim \mathcal{U}(\mathbb{D})$. The main area of application for MC methods are situations in which the MC method can be proven to converge faster than a corresponding deterministic method. In the case of high-dimensional quadrature ($\mathbb{D} \subset \mathbb{R}^D, D > 10$), the rate of convergence for the MC method is $\mathcal{O}(N^{-1/2})$ whereas quadrature methods typically converge with a rate $\mathcal{O}(N^{-k/D})$ for some order $k \in \mathbb{N}$. Thus, even for relatively small numbers of D, Monte Carlo integration converges faster than general purpose quadrature codes.

Monte Carlo (MC)

2.6 Hyper-parameter selection

So far, we have seen how to perform necessary inference computations *within* a specific model. Begging the question, *which* model should be selected in the first place?

In a probabilistic model, a criterion that can be used is the *(model) evidence*

(model) evidence

$$P(z_{0:N}) = \int P(z_{0:N} \mid f) P(f) \, \mathrm{d}f, \tag{18}$$

where $P(f)$ needs to be understood as the abstract prior over individual realizations. The higher the evidence, the better the explanatory power of the model, which is a combination of *goodness-of-fit* and *simplicity*.

goodness-of-fit

Typically, the prior $P(f) = P_\theta(f)$ over f will be an algebraic expression relying on another set of parameters θ which are called *hyper-parameters*. Hyper-parameters can take many different forms. Continuous parameters, for instance, are the output variance σ^2 or the length scale λ. The dimensionality q of a state-space is an example of a discrete parameter. Some choices cannot meaningfully be indexed such as the form of explicit feature maps Φ.

hyper-parameters

Of course, a proper Bayesian would put another probability distribution over these decisions. In practice, this is most often computationally prohibitive and we are forced to more practical decisions. A standard procedure is to fix the general form, maybe compare two or three different discrete choices, and use a standard statistical procedure for the remaining continuous parameters. This approach is commonly known as *empirical Bayes*.

empirical Bayes

In this thesis, two empirical Bayes methods are applied: *type-II maximum likelihood (ML)* estimation and *moment matching*. In the former case, one maximizes the marginal likelihood, i.e., Equation (18). Commonly, this requires a numerical optimization algorithm. Similarly, moment matching involves matching the moment of a prior to the empirical moment of the data.

type-II maximum likelihood (ML)

moment matching

3

Ordinary differential equations

This chapter provides a primer in ordinary differential equations, and an introduction to numerical methods and their analysis for the solution of ordinary differential equation problems with modern codes.

3.1 Introduction to ODEs

An *ordinary differential equation (ODE)* is the expression of a function $\boldsymbol{y} : \mathbb{T} \subset \mathbb{R} \to \mathbb{R}^D$ of one variable over a closed interval $\mathbb{T} = [t_0, T]$ in terms of its derivative

ordinary differential equation (ODE)

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{y} = \boldsymbol{f}(t, \boldsymbol{y}). \tag{19}$$

For the most part, we will use Lagrange's notation $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{y} = \boldsymbol{y}'$ to denote the derivative of $\boldsymbol{y}$, but occasionally we will use Newton's notation $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{y} = \dot{\boldsymbol{y}}$ when it is more convenient. A particular $\boldsymbol{y}_* : \mathbb{T} \to \mathbb{R}^D$ that satisfies Eq. (19) for all $t \in \mathbb{T}$ is called a *solution* of (19). We will denote both the equation variable as well as possible solutions by $\boldsymbol{y}$ where possible without confusion.

The graph of $\boldsymbol{f}$ is also called *vector field* or *phase space* of $\boldsymbol{y}$. As an example, consider the logistic growth equation

vector field

phase space

$$y' = ry(1 - \frac{y}{K}) \tag{20}$$

with growth rate $r \in \mathbb{R}_{>0}$ and capacity $K \in \mathbb{R}_{>0}$. The vector field of Eq. (20) with rate $r = 3$ and capacity $K = 1$ is depicted in Figure 2.

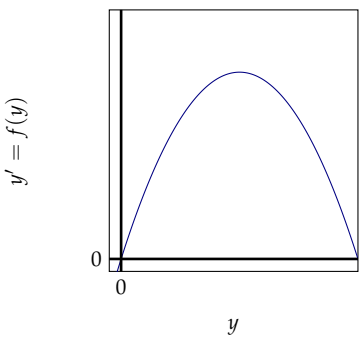

Figure 2: The vector field of the example problem $y' = ry(1 - y/K)$ with rate $r = 3$ and capacity $K = 1$.

If the right-hand side of Eq. (19) does not include the independent variable t, the ODE is called *autonomous*, otherwise it is called non-autonomous. The logistic growth equation (20) is autonomous. A non-autonomous ODE $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$ can be transformed to an autonomous ODE by writing

autonomous

$$\tilde{\boldsymbol{y}}' = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} t \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} 1 \\ \boldsymbol{f}(t, \boldsymbol{y}) \end{pmatrix}.$$

Eq. (19) might include more than just the first derivative

$$\boldsymbol{y}^{(n+1)} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(n)}),$$

in which case the ODE is called of *n-th order*. A D-dimensional n-th

n-th order

order ODE can be transformed to a nD-dimensional first order ODE via the transformation

$$\tilde{\boldsymbol{y}}' = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y}' \\ \vdots \\ \boldsymbol{y}^{(n+1)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}' \\ \boldsymbol{y}^{(2)} \\ \vdots \\ \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{y}', \ldots, \boldsymbol{y}^{(n)}) \end{pmatrix}.$$

To simplify the notation, we will present the theory for one-dimensional first-order problems. Where multi-dimensional problems require special treatment, we will indicate so in the text.

More general types of differential equations exist, such as *delay differential equations*, *implicit ODEs* or general *differential-algebraic equations*. However, we will not be concerned with these more general types of equations here.

We will now present the two most important existence theorems regarding ODEs from Peano [167] (see also Mie [151]) and the theorem of Picard [170] and Lindelöf [137]. The theorems are cited without proofs as they can be found in standard textbooks, e.g., Hairer, Nørsett, and Wanner [82] or Deuflhard and Bornemann [53].

Theorem 1 (Peano)**.** *Let $\Omega \subset \mathbb{T} \times \mathbb{R}$ be an open subset and $f : \Omega \to \mathbb{R}$ be continuous. Then for every $(t_0, y_0) \in \Omega$ exists a solution $y_* : \mathcal{U} \subset \mathbb{T} \to \mathbb{R}$ on a neighborhood $\mathcal{U}$ of t_0 such that $y_*(t_0) = y_0$ and $y_*'(t) = f(t, y_*(t))$ for all $t \in \mathcal{U}$.*

Peano's theorem guarantees existence under very mild conditions, but solutions need not be unique. The next theorem establishes a sufficient condition for uniqueness.

Theorem 2 (Picard-Lindelöf)**.** *Let $f : \mathbb{T} \times \mathbb{R} \to \mathbb{R}$ be Lipschitz in its second argument, i.e.,*

$$|f(t, y) - f(t, \tilde{y})| \leq L\,|y - \tilde{y}|$$

for some $L \in \mathbb{R}_{>0}$ for all $y, \tilde{y} \in \mathbb{R}$ and all $t \in \mathbb{T}$. Then for all $(t_0, y_0) \in \mathbb{T} \times \mathbb{R}$ there exists an $\epsilon > 0$ such that there exists a unique solution $y_*(t) : [t_0 - \epsilon, t_0 + \epsilon] \to \mathbb{R}$ satisfying $y_*(t_0) = y_0$ and $y_*'(t) = f(t, y_*(t))$ for all $t \in [t_0 - \epsilon, t_0 + \epsilon]$.

In the following, we will tacitly assume the existence of a suitable Lipschitz constant L for f on $\mathbb{T}$ which is usually the case for closed integration domains $\mathbb{T}$.

Solutions to D-dimensional n-th order ODEs usually contain nD free parameters [82, §I.1]. Since the solution of an ODE is uniquely defined if its value $\boldsymbol{y}_0 = \boldsymbol{y}(t_0)$ is completely specified at some time t_0,

we can distinguish two classes of problems. If the user can provide a point on the solution graph

$$(t_0, \boldsymbol{y}_0, \ldots, \boldsymbol{y}_0^{(n)}) \in \mathbb{T} \times \underbrace{\mathbb{R}^D \times \cdots \times \mathbb{R}^D}_{n \text{ times}}$$

called the *initial condition*, an approximate solution can be constructed iteratively, expanding locally from t_0. In this case, the problem is called an *initial value problem (IVP)*.

initial condition

initial value problem (IVP)

As a concrete example consider again the logistic growth (20) with initial conditions $y(t_0) = y_0$. The solution of (20) can be found analytically to be

$$y = \frac{K y_0 \exp(r(t - t_0))}{K + y_0(\exp(r(t - t_0)) - 1)}.$$

Figure 3 plots the solution for the particular condition $y(0) = 10^{-1}$, rate $r = 3$ and capacity $K = 1$. Although this IVP can be solved in closed-form, it will serve as a non-linear example for numerical methods throughout this thesis.

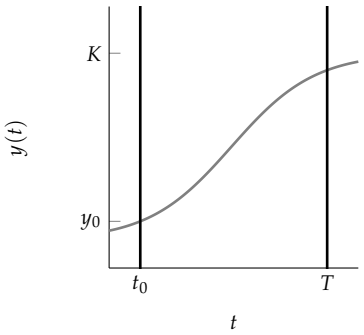

Figure 3: The closed-form solution of the example problem.

Alternatively, the problem may be stated in terms of *boundary conditions (BCs)*

$$g(\boldsymbol{y}(a), \boldsymbol{y}(b)) = 0, \quad a, b \in \mathbb{T} \tag{21}$$

for some function g, and the problem is called a *boundary value problem (BVP)*. Often, (21) takes the form

$$\boldsymbol{y}(a) = \boldsymbol{y}_a, \quad \boldsymbol{y}(b) = \boldsymbol{y}_b$$

for some $\boldsymbol{y}_a, \boldsymbol{y}_b \in \mathbb{R}^D$. The numerical solution of BVPs is more difficult as it typically requires multiple iterations over the entire domain $\mathbb{T}$. It can be argued that IVPs are a special subclass of BVPs that are particularly easy to solve [7, p. xxiii]. BVPs will be discussed in more detail in Section 3.5.

boundary conditions (BCs)

boundary value problem (BVP)

An important special case of an ODE is the linear equation

$$\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y}$$

with the general solution $\boldsymbol{y}(t) = exp(\boldsymbol{A}(t - t_0))\boldsymbol{y}_0$. The linear ODE is also called the *(Dahlquist) test equation* (Dahlquist [49, 50], Hairer and Wanner [83, §IV.2]), in particular for the one-dimensional case

$$y' = \lambda y, \tag{22}$$

(Dahlquist) test equation

$\lambda \in \mathbb{C}$. For values $\lambda \in \mathbb{C}^-$, the solution converges to zero for $t \to \infty$ and the problem is called *stable*. For details, we refer the reader to Deuflhard and Bornemann [53, §3] and Brugnano and Trigiante [25, §1].

stability

The test equation plays a dominant role in the study of numerical ODE solvers. Inserting (22) into the algebraic description of a numerical ODE solvers often leads to a closed form description of the algorithm's output which can be compared to the true solution. Stability analysis can reveal pathological behavior of an algorithm. This will be discussed in more detail in Section 3.3.

3.2 Numerical solution of initial value problems

The numerical solution of an ODE is the approximation $y_{0:N}$ to the true solution $y(t_n)$ on some discrete mesh $\{t_0, \dots, t_N\} = \Delta \subset \mathbb{T}$. The process of numerically solving an ODE is also known as *integration* (in contrast to quadrature) and a particular implementation of an algorithm is called a *code*. Excellent introductions can be found in the textbooks by Hairer, Nørsett, and Wanner [82], Brugnano and Trigiante [25] and Deuflhard and Bornemann [53].

integration

code

Initial value problems are typically solved by *step-by-step methods*. Assume for the moment that we want to compute an approximation to the solution on an equidistant mesh $t_n = t_0 + nh, n = 0, \dots, N$, of *step size* $h = N^{-1}(t_N - t_0)$. Step-by-step methods exploit the special structure of the IVP wherein the solution $y(t_{n+1})$ is entirely defined by its state at $y(t_n)$. As a consequence, the numerical solution y_{n+1} at *(evaluation) knot* t_{n+1} is constructed from its predecessor y_n and the process continues in a step-by-step fashion without revisiting former approximations y_{n-1} at a later stage.

step size

(evaluation) knot

One such method is used in the constructive proof of the Picard-Lindelöf theorem. At knot t_n, define a sequence $i = 0, 1, \dots$ of approximations

$$\begin{aligned} y_{n,0}(t) &= y_n, \\ y_{n,i+1}(t) &= y_n + \int_{t_n}^{t} f(s, y_{n,i}(s)) \, \mathrm{d}s. \end{aligned}$$

The iterative process is called *Picard iteration* and the numerical solution is set to $y_{n+1} = y_{n,i}(t_{n+1})$ for some $i > 0$. The integral typically requires a numerical quadrature method to be solved and every ODE solver can also be understood as an approximation to the quadrature problem.

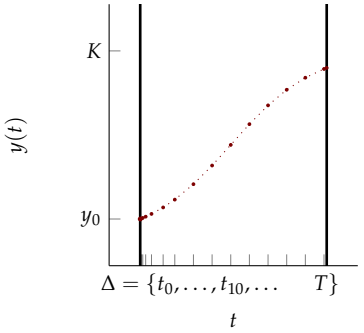

Figure 4: The numerical solution of the example problem $y' = ry(1 - y/K)$. Ticks indicate the mesh, big dots represent the numerical solution. Small dots are plotted as visual aid.

Picard iteration

In this thesis, we will only consider *linear methods* of the form

linear methods

$$\int_{t_n}^{t_{n+1}} f(s, y(s)) \, \mathrm{d}s \approx \sum_{k=1}^{K} w_k f(s_k, y(s_k)) \quad K \in \mathbb{N}$$

with suitable weights $w_k \in \mathbb{R}$ and knots $s_k \in [t_n, t_{n+1}]$. Since we do not have access to the solution y_*, the quadrature method has to be evaluated at approximations $f_{n,k}$ to the true dynamic function evalu-

ations $f(s_k, y_*(s_k))$. The easiest way to obtain the approximations is to set $f_{n,k} = f(s_k, y_n)$ for $s_k \in [t_n, t_{n+1}]$.

Overall, this defines a lower (block) triangular linear system of equations for the unknown variables $y_n, f_{n,k}$ which can be solved by forward substitution in one pass. The methods presented in Sections 3.2.1 and 3.2.2 will be of this type, although the linear system will not be written down explicitly.

But before we look at individual methods, let's start to get an intuition for the classical error analysis. There are three sources of errors for a numerical IVP solvers.

The first error source are *roundoff errors* which are the difference between the ideal real-valued numbers y_n and their finite representations on a digital computer (Goldberg [73]). In this thesis, all computations are carried out using the IEEE 754 floating point standard with double precision (Kahan and Palmer [111]). In double precision, roundoff errors are almost always dominated by the other error sources and we may ignore these in our analysis.

roundoff errors

The second source of error stems from the approximation of the quadrature problem. Let

$$\tau_n = \left| \int_{t_n}^{t_{n+1}} f(s, y_*(s))\,\mathrm{d}s - \sum_{k=1}^{K} w_k f(s_k, y_*(s_k)) \right|$$

denote the error from approximating the integral by a quadrature method whilst assuming access to the true solution y_*. We will call τ_n the *truncation* or *local error* and we will say that a method is of *q-th order*, if $\tau_n \in \mathcal{O}(h^{q+1})$.

local error

q-th order

The overall error is called the *discretization error* and will also be denoted by τ_n where the distinction is not important. This error includes the effects of inserting previous approximations into the linear system leading to propagated and accumulated errors. In most cases, the overall discretization error will be of order $\mathcal{O}(h^q)$ if the local error is of order $\mathcal{O}(h^{q+1})$, i.e., if the method is of order q.

3.2.1 Runge-Kutta methods

Runge-Kutta (RK) methods, named after their inventors Runge [179] and Kutta [124], belong to the class of one-step (cf. Section 3.2.2) shooting methods. At time t_{n+1}, the numerical approximation is defined as

Runge-Kutta (RK)

$$\begin{aligned} k_{i,n} &= f(t_n + hc_i,\, y_{t_n} + h\sum_{j=0}^{s-1} a_{ij} k_{j,n}), \qquad i = 0, \ldots, s-1 \\ y_{t_{n+1}} &= y_{t_n} + h\sum_{i=0}^{s-1} b_i k_{i,n}. \end{aligned} \tag{23}$$

The parameters a_{ij}, c_i and b_i are usually expressed as a matrix $\boldsymbol{A}$ and two vectors $\boldsymbol{b}, \boldsymbol{c}$, written compactly in a so-called Butcher tableau:

c_0	a_{00}	a_{01}	$\dots$	$a_{0,s-2}$	$a_{0,s-1}$
c_1	a_{10}	a_{11}	$\dots$	$a_{1,s-2}$	$a_{1,s-1}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
c_{s-2}	$a_{s-2,0}$	$a_{s-2,1}$	$\dots$	$a_{s-2,s-2}$	$a_{s-1,s-1}$
c_{s-1}	$a_{s-1,0}$	$a_{s-1,1}$	$\dots$	$a_{s-1,s-2}$	$a_{s-1,s-1}$
	b_0	b_1	$\dots$	b_{s-2}	b_{s-1}

If the matrix $\boldsymbol{A}$ is strictly lower triangular and $c_1 = 0$, Equation (23) simplifies to an iterative procedure of explicit equations

$$k_{i,n} = f(t_n + hc_i, y_n + h\sum_{j=0}^{i-1} a_{ij}k_{j,n}), \qquad i = 0, \dots, s-1 \tag{24a}$$

$$y_{n+1} = y_n + h\sum_{j=0}^{s-1} b_j k_{j,n}, \tag{24b}$$

and in this case, the formula is called an *explicit RK method*. Otherwise Eq. (23) defines a system of nonlinear equations that needs to be solved at every step and the method is called an *implicit RK method*. In this thesis, we will be mostly concerned with explicit methods.

For instance, the simplest Runge-Kutta method is the well-known *Euler's method* Euler's method

$$k_{0,n} = f(t_n, y_n), \qquad y_{n+1} = y_n + hk_{0,n} \tag{25}$$

which is a first order method. Other popular examples include the *midpoint rule* midpoint rule

$$k_{0,n} = f(t_n, y_n), \quad k_{1,n} = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_{0,n}),$$
$$y_{n+1} = y_n + hk_{1,n},$$

and the *trapezoidal method* trapezoidal method

$$k_{0,n} = f(t_n, y_n), \qquad k_{1,n} = f(t_n + h, y_n + hk_{0,n}),$$
$$y_{n+1} = y_n + \frac{h}{2}(k_{0,n} + k_{1,n}).$$

The last two methods get their name from their counterparts in numerical quadrature and are second order methods. The Butcher tableaus of the examples are displayed in Eq. (26).

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array} \tag{26}$$

Euler's method | midpoint method | trapezoidal method

We will now analyze the local truncation error of explicit Runge-Kutta methods. To this end, consider the error $\tau_1 = |y(t_1) - y_1|$. If $\tau_1 \in \mathcal{O}(h^{q+1})$ for some $q \in \mathbb{N}$, then it is possible to show that the *global error* $E_n = |y(t_n) - y_n|$ is bounded by $h^q C(exp(L(t_n - t_0)) - 1)$ and, thus, the method converges to the true solution if the step size h decreases. However, note that this is not only a statement about an asymptotic behavior. Even for small *but finite* h, the local error will be a function of h^{q+1}, which decreases faster than a function of h^q. In this sense, a method of high order is more efficient, since it will require less computational effort to achieve the same level of accuracy.

global error

Recall from calculus the total derivative of f

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} f(t, y(t)) &= \frac{\partial}{\partial t} f(t, y(t)) \frac{\mathrm{d}}{\mathrm{d}t} t + \frac{\partial}{\partial y} f(t, y(t)) \frac{\mathrm{d}}{\mathrm{d}t} y(t) \\ &= f_t(t, y) + f_y(t, y) f(t, y). \end{aligned} \tag{27}$$

Using Eq. (27), we can also find the higher derivatives of y with recursive applications. Write $y(t_0 + h)$ in Taylor expansion form and replace higher deriatives of y by f and its partial derivatives:

$$\begin{aligned} y(t_0 + h) = y_0 + h f(t_0, y_0) + \frac{h^2}{2}[f_t + f_y f](t_0, y_0) \\ + \frac{h^3}{6}[f_{tt} + 2 f_{ty} f + f_y f_t + f_{yy} f^2 + f_y^2 f](t_0, y_0) + \mathcal{O}(h^4). \end{aligned} \tag{28}$$

If y_1 is defined by Euler's method (25), it follows from (28) that $\tau_1 \in \mathcal{O}(h^2)$. Similarly, we can compute the Taylor expansion of the trapezoidal method (as a function of h):

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}h} y_1\Big|_{h=0} &= \frac{1}{2} f(t_0, y_0) + \frac{1}{2} f(t_0 + h, y_0 + h f(t_0, y_0)) \\
&\quad + \frac{h}{2} [f_t + f_y f(t_0, y_0)](t_0 + h, y_0 + h f(t_0, y_0))\Big|_{h=0} \\
&= f(t_0, y_0), \\
\frac{\mathrm{d}^2}{\mathrm{d}h^2} y_1\Big|_{h=0} &= 0 + \frac{1}{2} [f_t + f_y f(t_0, y_0)](t_0 + h, y_0 + h f(t_0, y_0)) \\
&\quad + \frac{1}{2} [f_t + f_y f(t_0, y_0)](t_0 + h, y_0 + h f(t_0, y_0)) \\
&\quad + \frac{h}{2} \frac{\mathrm{d}^2}{\mathrm{d}h^2} f(t_0 + h, y_0 + h f(t_0, y_0))\Big|_{h=0} \\
&= [f_t + f_y f(t_0, y_0)](t_0, y_0).
\end{aligned} \tag{29}$$

Substracting (29) from (28), we find that $|y_1 - y(t_0 + h)| = \mathcal{O}(h^3)$, i.e., the trapezoidal method is convergent at order $\mathcal{O}(h^3)$. One important aspect of (29) is that $y|_{h=0} = f|_{t=t_0, y=y_0}$, which is why the evaluation of f can always be considered at (t_0, y_0).

The manual calculation becomes cumbersome very quickly. The general theory for evaluating the order of a specific Runge-Kutta method, represented by its coefficients, is usually derived using labeled trees for representing the combinatorial problem of computing higher (partial) derivatives of y and f [82]. The result is a systematic way for generating (non-linear) algebraic equations for the coefficients which must be satisfied in order to obtain a certain order. For methods of order $q \leq 4$, the algebraic equations can be summarized by a parameterized Butcher tableau which capture all available methods of low order. For $q > 4$, there is no explicit method of order q with $s = q$ stages, because there are more constraints than number of free parameters. However, methods of all orders $q \in \mathbb{N}$ can be achieved by using a higher number $s_q > q$ of stages.

3.2.2 Multistep methods

The Runge-Kutta methods from Section 3.2.1 have the advantages of being easy to both analyze and implement, because their global behavior is completely determined by their local behavior and individual steps are mutually independent.

However, it should be intuitively clear that this mutual independence comes at the cost of a computational overhead. Each evaluation of f contains information about the unknown function y, and ignoring previous evaluations can be thought of as learning from scratch after each step.

Thus, it is natural to consider methods that carry some information from previous steps. Keeping the constraint of linear methods and general approximations, one arrives naturally at methods that use polynomial interpolation over several steps. Methods of this type are called *linear multistep methods (LMMs)* and are defined by

linear multistep methods (LMMs)

$$\sum_{j=0}^{q} \alpha_j y_{n-q+j} = h \sum_{j=0}^{q} \beta_j f_{n-q+j}. \tag{30}$$

The numbers α_j, β_j are the parameters of the method which will be chosen to gain high convergence order. The y_{n-q+j} are the numerical solution of $y(t)$ and $f_{n-q+j} = f(t_{n-q+j}, y_{n-q+j})$. If $\beta_q = 0$, f_{n-q+j} is only required after y_{n-q+j} has been computed and (30) is an explicit computation scheme. If $\beta_q \neq 0$, then y_{n-q+j} and f_{n-q+j} are defined simulateously, and (30) turns into the implicit equation

$$\alpha_q y_n - h\beta_q f(t_n, y_n) = \sum_{j=0}^{q-1} (\alpha_j y_{n-q+j} - h\beta_j f_{n-q+j}) \tag{31}$$

for y_n.

Since LMMs are commonly derived by interpolation or quadrature problems, there are two natural ways to solve (31). The first way is a direct solution of (31) with Newton-type solvers [55] which is always feasible. Alternatively, one can look at the corresponding explicit version of (31) to use this as a *predictor* $y_n^{(0)}$ of y_n. Then, one updates $y_n^{(0)}$ with the iterative procedure

$$y_n^{(m)} = \frac{1}{\alpha_q} \left(h\beta_q f(t_n, y_n^{(m-1)}) + \sum_{j=0}^{q-1} (\alpha_j y_{n-q+j} - h\beta_j f_{n-q+j}) \right) \tag{32}$$

for $m = 1, \ldots, M$, a pre-defined fixed number M of times. Eq. (32) is called the *corrector* step of (31) and the whole procedure is called a *M-steps predictor-corrector P(EC)^M* method. Solving (31) via (32) is also called solving via *function iteration*.

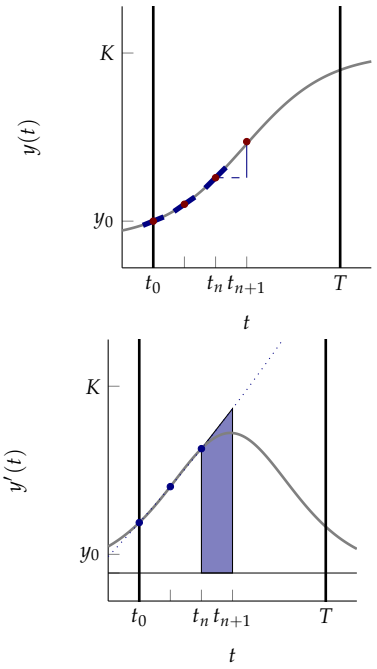

Figure 5: Illustration of an explicit linear multistep method. Red dots represent the numerical solution. Blue dots represent the interpolation of the derivative $y'(t)$. The next point is computed via quadrature (blue area).

predictor

corrector

M-steps predictor-corrector P(EC)^M

function iteration

The analysis of the numerical error can be carried out to some extend similarly as in the Runge-Kutta case. First, we analyze the idealized local truncation error by considering the operater

$$L(y, t, h) = \sum_{j}^{q} \left(\alpha_j y(t - (q+j)h) - h\beta_j y'(t - (q+j)h) \right). \tag{33}$$

By subtracting (33) from (30), we arrive at the local error

$$\tau_q = \left(\alpha_q \boldsymbol{I} - h\beta_q f_y(t_q, \tilde{y}) \right)^{-1} L(y, t_0, h),$$

where the Jacobian f_y is evaluated at appropriately chosen values $\tilde{y}$ whose existence is guaranteed by the mean value theorem. Thus, it

is justified to say that a LMM is of order q, if $L(y,t,h) \in \mathcal{O}(h^{q+1})$ for all sufficiently regular functions y (see Hairer, Nørsett, and Wanner [82, §III.2]).

Closely connected to the theory of multistep methods are the *generating polynomials* of a method. They are defined as

generating polynomials

$$\rho(\zeta) = \alpha_q \zeta^q + \alpha_{q-1}\zeta^{q-1} + \cdots + \alpha_0,$$
$$\sigma(\zeta) = \beta_q \zeta^q + \beta_{q-1}\zeta^{q-1} + \cdots + \beta_0.$$

If $\rho(\zeta)$ and $\sigma(\zeta)$ have no common factor, they are called *irreducible* and there is a one-to-one correspondence between a multistep method and its generating polynomials. We will tacitly assume that the polynomials are already in irreducible form. From the generating polynomials, properties of the corresponding multistep method can be deduced such as order and stability (see Deuflhard and Bornemann [53]).

irreducible

Skeel [198] showed that methods of type (30) are equivalent to multistep methods written in *Nordsieck form*:

Nordsieck form

$$\boldsymbol{x}_n = \left(y_n, hy'_n, \ldots, {}^{h^q y_n^{(q)}}/_{q!}\right)^\mathsf{T}, \tag{34}$$
$$\boldsymbol{x}_{n+1} = \left(\boldsymbol{I} - \boldsymbol{l}\boldsymbol{e}_1^\mathsf{T}\right) \boldsymbol{P}\boldsymbol{x}_n + h\boldsymbol{l} z_{n+1}. \tag{35}$$

The variable $\boldsymbol{x}_n$ now contains the entire information of the local polynomial approximation in terms of its Taylor coefficients. In Eq. (35), $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{P}$ is the upper triangular Pascal matrix with coefficients $(\boldsymbol{P})_{ij} = \mathbb{1}_{i\le j}\binom{i}{j}$, $\boldsymbol{e}_1^\mathsf{T} = (0,1,0,0,\ldots)$ is the second unit vector, and $\boldsymbol{l} = (l_0, l_1, \ldots, l_q)^\mathsf{T}$ is a vector of parameters with $l_1 = 1$. In this case, the Nordsieck representation corresponds to the multistep method with the generating polynomials

$$\rho(\zeta) = \det(\zeta\boldsymbol{I} - \boldsymbol{P})\boldsymbol{e}_1^\mathsf{T}(\zeta\boldsymbol{I} - \boldsymbol{P})^{-1}\boldsymbol{l},$$
$$\sigma(\zeta) = \det(\zeta\boldsymbol{I} - \boldsymbol{P})\boldsymbol{e}_0^\mathsf{T}(\zeta\boldsymbol{I} - \boldsymbol{P})^{-1}\boldsymbol{l}.$$

Similar expressions hold for the reverse case.

The difference in presentation between (30) and (34) can be understood as expressing the interpolation polynomial $\pi(t)$ in either Lagrange notation (Equation (30)) or Taylor expansion notation (Equation (34)) ([53, §7.3.1]). In this case, $\boldsymbol{P}\boldsymbol{x}_n$ yields a prediction of the numerical Taylor expansion at t_{n+1} and the scalar increment z_{n+1} is implicitly defined as the solution to

$$h^{-1}(\boldsymbol{P}\boldsymbol{x}_n)_1 + l_1 z_{n+1} = f(t_n + h, (\boldsymbol{P}\boldsymbol{x}_n)_0 + hl_0 z_{n+1}), \tag{36}$$

which is the correction from $\boldsymbol{x}_n$ to $\boldsymbol{x}_{n+1}$ to the Taylor coefficients. Equation (36) can be solved by iterated function evaluations of the form

$$z_{n+1}^{(1)} = f\left(t_n + h, (\boldsymbol{P}\boldsymbol{x}_n)_0\right), \tag{37}$$
$$z_{n+1}^{(M)} = l_1^{-1}\left[f\left(t_n + h, (\boldsymbol{P}\boldsymbol{x}_n)_0 + h l_0 z_{n+1}^{(M-1)}\right) - h^{-1}(\boldsymbol{P}\boldsymbol{x}_n)_1\right],$$

or by directly solving (36) with some variant of the Newton-Raphson method.

If $z^{(M)}$ is used in the computation of (35), the resulting algorithm is called a P(EC)$^{\mathrm{M}}$ method, which stands for *predict–evaluate–correct*. If Equation (36) is solved up to numerical precision, the method is called a P(EC)$^{\infty}$ method. Generally, these methods are called *predictor-corrector (PEC) methods*. Nordsieck methods with suitable weights $\boldsymbol{l}$ can be shown to have a local truncation error of order q or $q+1$ [197, 198]. More details can also be found in standard textbooks [82, 53].

predictor-corrector (PEC) methods

3.3 Stiff problems and stability of numerical codes

One important concept that is associated with the numerical solution of ODEs is that of *stiffness* [48]. While non-stiff ODEs have been considered to be a solved problem [69], methods for stiff systems are still an active area of research [24].

stiffness

While various attempts have been made to characterize and define *stiffness* [95, 51, 24], an agreed upon precise definition is still missing. Many researchers take the perspective that "stiff equations are equations where certain implicit methods, in particular BDF (*Backwards Differentiation Formulae*, note from the author), perform better, usually tremendously better, than explicit ones". [48, 83].

An intuition for stiffness can be developed from the test equation [50]. For $y' = \lambda y$, the solution of a linear (multistep) method can be expressed as a linear (in y) (multi-dimensional) recurrence equation, usually of the form

$$\boldsymbol{y}_{n+1} = \boldsymbol{R}(z)\boldsymbol{y}_n,$$

where $z = h\lambda \in \mathbb{C}$ is the product of step size $h > 0$ and linearity coefficient $\lambda \in \mathbb{C}$, and $\boldsymbol{R}$ is a polynomial of the complex argument z. The set $\mathcal{S} = \{z \in \mathbb{C} : |\boldsymbol{R}(z)| \leq 1\} \subset \mathbb{C}$ is called the *stability domain* of the method. If $\mathrm{Re}(\lambda) < 0$ and $h\lambda = z \notin \mathcal{S}$, the numerical solution typically contains diverging oscillations whereas the true solution is converging.

stability domain

This thesis will only be concerned with explicit methods which typically have a small associated stability domain. Thus, the admis-

sible step sizes h for linear equations with large negative coefficients will be prohibitively small. In this sense, the methods in this thesis can only handle "non-stiff" systems. However, these methods provide a novel *ansatz* and novel functionality which may provide a foundation for future developments for stiff problems.

3.4 Step size adaptation

The methods in the previous sections have been presented using a constant step size h which generates an equidistant mesh $\Delta_h = \{t_0 + nhN^{-1} \mid n = 0, \dots, N\}$. This section discusses strategies to compute adaptive step sizes h_n in step-by-step methods to minimize the computational effort while guaranteeing certain error criteria. For a more detailed introduction, we recommend Deuflhard and Bornemann [53, §5].

Most step size adaptation schemes are based on *estimates* of the *local discretization error* which is defined to be the error τ_n introduced in the current integration step only, i.e., treating the last approximation y_n as the correct integration value and considering the IVP

$$y' = f(t, y) \quad y(t_n) = y_n.$$

For both linear method classes, Runge-Kutta and multistep methods, the local error will typically satisfy

$$|y(t_{n-1} + h_{n-1}) - y_{t_n}| \leq Ch^{q+1},$$

for some appropriate $q \in \mathbb{N}$. Now take a second method for the same integration step which computes $\hat{y}_{t_n}$ at order $\hat{q}$, usually $\hat{q} = q - 1$, and estimate

$$\hat{\tau}_n = |y_{t_n} - \hat{y}_{t_n}| \approx Ch^q. \tag{38}$$

If $\hat{\tau}_n$ is bigger than the desired (relative) tolerance, h_n is reduced and the step is repeated. If $\hat{\tau}_n$ is sufficiently small, Eq. (38) is used to solve for h_{n+1} which is proposed as tentative step size for the next step.

Thus, the key to effective step size adaptation is the design of methods which can compute two integration values with little overhead. For a Runge-Kutta method (c, A, b), one searches for similar methods $(\hat{c}, \hat{A}, \hat{b})$ with as many common factors as possible which are called *embedded Runge-Kutta (ERK)* methods. Typically, only the final weights $\hat{b}$ need to be adjusted or only one additional function evaluation $\hat{c} = (c, c_{s+1})$ is required to find a suitable pair. In multistep methods, in particular PEC methods, one can use the predictor as a method of order $q - 1$ which makes error estimation particularly

embedded Runge-Kutta (ERK)

easy. The application of this feature of PEC methods for step size selection is also called the *Milne device* (Milne [152]). However, LMMs with varying step sizes may suffer from certain dynamic behavior which effectively reduces the convergence order by one (cf. Kulikov and Shindin [123] and Kulikov [122]).

Milne device

One important thing to note in the context of this thesis is the necessarily *heuristic* nature of step size adaptation schemes. Local errors cannot be computed accurately, but may only be *estimated*. While it has rarely been pointed out in the literature, this estimation can be built upon the concepts presented in Chapter 2. For instance, Deuflhard [54] suggests an algorithm for both step size and order estimation in the context of information theory—a close cousin to probability theory.

Secondly, while the error analyses of linear methods clearly state required conditions for convergence, the step size adaptation schemes rely on assumptions which are much harder to validate during runtime. This needs to be kept in mind for the models which will be presented later in this thesis, as the assumptions there are not necessarily *more restrictive*, but merely *more explicitely* stated than the requirements and assumptions of a regular numerical algorithm.

3.5 Numerical solution of boundary value problems

BVP solvers can broadly be classified into two categories: *shooting methods* and *finite difference methods*. The former class uses a sequence of step-by-step methods to find missing initial values, typically by applying a variant of Newton's method. Finite difference methods are representative for a broader class of *global methods* which are considered to be state-of-the-art (Cash and Mazzia [33] and Mazzia, Cash, and Soetaert [146]). This thesis expands on an algorithm by Hennig and Hauberg [92] to develop medical imaging applications in Chapters 9 and 10. The original algorithm by Hennig and Hauberg [92] was inspired by a collocation method, which is structurally similar to finite difference methods.

shooting methods

finite difference methods

Here, we present the simplest case of a finite difference scheme to gain an intuition of the method by Hennig and Hauberg [92] which will be presented in detail in Section 4.3. The reader is referred to Ascher, Mattheij, and Russell [7] for details.

First, consider the special case of a *linear BVP*. A linear BVP is a problem of the type

linear BVP

$$\boldsymbol{y}' = \boldsymbol{A}(t)\boldsymbol{y} + \boldsymbol{q}(t), \qquad \boldsymbol{A}(t) : \mathbb{T} \to \mathbb{R}^{n\times n},\ \boldsymbol{q}(t) : \mathbb{T} \to \mathbb{R}^n, \tag{39a}$$

$$\boldsymbol{\beta} = \boldsymbol{B}_a\boldsymbol{y}(a) + \boldsymbol{B}_b\boldsymbol{y}(b), \quad \boldsymbol{B}_a, \boldsymbol{B}_b \in \mathbb{R}^{n\times n},\ \boldsymbol{\beta} \in \mathbb{R}^n, \tag{39b}$$

where $\mathbb{T} = [a, b] \subset \mathbb{R}$ is a closed intervall.

Assume that we have been given a step size $h \in \mathbb{R}$. Consider the finite difference approximation to the true derivative

$$\boldsymbol{y}'\left(\frac{t_n + t_{n+1}}{2}\right) \approx \frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{h}. \tag{40}$$

Plugging (40) into (39), we arrive at the linear problem

$$\begin{pmatrix} \boldsymbol{S}_1 & \boldsymbol{R}_1 & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_2 & \boldsymbol{R}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{S}_N & \boldsymbol{R}_N \\ \boldsymbol{B}_a & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{B}_b \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_N \\ \boldsymbol{y}_{N+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}(\frac{t_1+t_2}{2}) \\ \boldsymbol{q}(\frac{t_2+t_3}{2}) \\ \vdots \\ \boldsymbol{q}(\frac{t_N+t_{N+1}}{2}) \\ \boldsymbol{\beta} \end{pmatrix} \tag{41}$$

where $\boldsymbol{S}_n = -h^{-1}\boldsymbol{I} - 2^{-1}\boldsymbol{A}(t_n)$ and $\boldsymbol{R}_n = h^{-1}\boldsymbol{I} - 2^{-1}\boldsymbol{A}(t_{n+1})$. Thus, if the matrix on the left hand side is invertible, the numerical solution exists. If the boundary conditions are separated, i.e., if (39b) can be written as $\boldsymbol{B}_{a'}\boldsymbol{y}(a) = \boldsymbol{\beta}_a$, $\boldsymbol{B}_{b'}\boldsymbol{y}(b) = \boldsymbol{\beta}_b$, the linear system (41) can be turned into a banded matrix which can be solved efficiently. Writing the boundary conditions in separated form is always possible (Ascher and Russell [6]).

For a general non-linear BVP, we can apply the same ansatz of finite difference which yields

$$\frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{h} = \frac{1}{2}(\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})), \quad n = 1, \dots, N \tag{42a}$$

$$0 = g(\boldsymbol{y}_1, \boldsymbol{y}_{N+1}). \tag{42b}$$

These are $N+1$ non-linear equations for $N+1$ unknowns. These can be solved with standard techniques, e.g., by writing (42) as a root-finding problem and using some Newton method to solve it. Finally, Eq. (42) could be solvable by a fixed point iteration scheme, similarly to the PEC-scheme in linear multistep methods (cf. Section 3.2.2).

As in the case for Runge-Kutta methods, we can generalize the finite difference approach to also match higher derivatives or to consider more general approximation models (Mazzia and Sgura [145]), such as B-splines (Mazzia, Sestini, and Trigiante [149]), which will yield linearizations obtaining truncation errors of higher order. An introduction can be found in Brugnano and Trigiante [25].

4 Probabilistic ODE solvers

Sections 2 and 3 layed out the necessary mathematical machinery. Now, we want to discuss the general idea of *probabilistic numerics* with a special focus on probabilistic numerical methods for the solution of ordinary differential equations. We will also highlight two early probabilistic algorithms serving as prototypes and motivation for the developments of Parts iii and iv.

4.1 The probabilistic interpretation of numerical methods and probabilistic numerics

Interpreting numerical computations as probabilistic inference methods is almost as old as the oldest numerical methods for solving differential equations. In Poincaré [171], Poincaré writes:

> "Plaçons-nous à un autre point de vue.
>
> Une question de probabilités ne se pose que par suite de notre ignorance: il n'y aurait place que pour la certitude si nous connaissions toutes les données du problème. D'autre part, notre ignorance ne doit pas être complète, sans quoi nous ne pourrions rien évaluer. Une classification s'opérerait donc suivant le plus ou moins de profondeur de notre ignorance.
>
> Ainsi la probabilité pour que la sixième décimale d'un nombre dans une table de logarithmes soit égale à 6 est *a priori* de $1/10$; en réalité, toutes les données du problème sont bien déterminées, et, si nous voulions nous en donner la peine, nous connaîtrions exactement cette probabilité. De même, dans les interpolations, dans le calcul des intégrales définies par la méthode de Cotes ou celle de Gauss, etc.
>
> [...]
>
> Dans d'autres problèmes, enfin, il peut arriver que notre ignorance soit plus grande encore, que la loi elle-même nous échappe. La définition des probabilités devient alors presque impossible. Si, par exemple, x est une fonction inconnue de t, nous ne savons pas très bien quelle probabilité il faut attribuer, au début, à x_0, pour connaître
>
> $$\int_{t_0}^{t} x \, \mathrm{d}t.$$
>
> On se laissera souvent guider par un sentiment vague qui s'impose avec puissance, qu'on ne saurait pourtant justifier, mais sans lequel, en tout cas, aucune science ne serait possible."

An English translation can be found in Appendix C.

In the above section, Poincaré argues for the necessity of assumptions for approximation theory—a sentiment that is unchallenged these days. However, by suggesting the assignment of probabilities, he also implies to weigh various hypotheses *gradually*. In contrast, modern textbooks typically focus on error bounds which could be interpreted as a proposition of a uniform posterior distribution.

The idea of assigning probabilities to computation outcomes has been revisited in the last century by only few mathematicians. An overview of early work in English is given in Larkin [128]. He points to previous work by Sul'din [206, 207] and Sard [180]. In these works, the idea of applying probability theory to linear approximation problems typically considered in numerical analysis is considered in a way closely resembling modern presentations.

In [56], Diaconis presents a precise equivalence between certain quadrature rules and Bayesian inference on certain models. A similar observation has been repeated by O'Hagan [159]. Today, this methodology is an established area of research in its own right and is referred to as *Bayesian quadrature (BQ)*, an early demonstration of the applicability of probabilistic numerical methods.

Bayesian quadrature (BQ)

In a parallel development, the theory of function interpolation on more general spaces than polynomials has started to get more attention by authors like Schoenberg [188] and later by Ahlberg, Nilson, and Walsh [3] and Schoenberg [189]. When explicit connections are found, the results usually rely upon an equivalence between generalized spline smoothing and Gaussian process regression which was first formulated by Kimeldorf and Wahba [117] in 1970 and later refined by Wahba [217].

However, in the case of the numerical solution of differential equations, the idea did not seem to catch on. The author is only aware of two early references, Skilling [199] and Fierro and Torres [64], of which the latter has not received public awareness until very recently.

4.2 Current understanding of probabilistic numerical methods

Even though the idea of interpreting numerical algorithms as probabilistic inference has been floating around for some time, it has only attracted sustained research interest in recent years [93]. In fact, the research area is so novel that there is no clear unique definition yet of what should be a probabilistic numerical method.

At the time of writing, two sets of authors have presented working definitions, Hennig, Osborne, and Girolami [93] and Cockayne, Oates, Sullivan, and Girolami [41]. For the purpose of this thesis, we will follow the definition of Hennig, Osborne, and Girolami [93, §3.(a)].

probabilistic numerical method

A *probabilistic numerical method* is a *numerical algorithm* for a *mathematical problem* which admits an *interpretation of probabilistic inference*. For this to be the case, the problem needs to be such that it is typically solved by an iterative process where each step provides novel information about the sought solution. In a probabilistic numerical method, the unknown solution is treated as a latent random variable and intermediate quantities are treated as observable random variables. A generative model relates the observable quantities to the unknown solution. Often, this model admits a natural separation into a *prior* over the unknown solution and a *likelihood* of observing the data given the solution. In this setting, the methods from Chapter 2 can be used to find the *posterior distribution* over the solution which is intended as the desired output.

Contrary to the setting in Chapter 2, wherein inference methods have no ability to interact with the environment they try to infer, a probabilistic numerical algorithm additionally consists of an *action rule* which may actively decide on how to generate new observable quantities. This creates a mutual sequential dependence which can increase the complexity of the analysis in certain cases as, for example, in the numerical solution of ODEs.

This situation is similar to ideas in (Bayesian) decision theory (see, e.g., Parmigiani and Inoue [165]) and sequential analysis (Wald [218]). Recently, the connection to Wald's sequential analysis has also been pointed out in Owhadi and Scovel [162]. In particular, this concept of sequential execution is not yet incorporated in the framework proposed by Cockayne, Oates, Sullivan, and Girolami [41] which renders the rigorous theory presented therein unapplicable in our setting.

4.3 The deterministic solver from Hennig & Hauberg

In the remainder of this section, we present two early publications on probabilistic ODE solvers. These will serve as foundations to extend upon or as examples to contrast against. First, we consider the work by Hennig and Hauberg [92]. This algorithm is the basis for the extensions presented in Chapters 9 and 10. In the next section, we will have a closer look at the work of Chkrebtii, Campbell, Calderhead, and Girolami [39].

In Hennig and Hauberg [92], the authors consider the problem of finding curves $\boldsymbol{y} : [0, 1] \to \mathbb{R}^D$ subject to an autonomous, second order differential equation

$$\boldsymbol{y}''(t) = \boldsymbol{f}(\boldsymbol{y}(t), \boldsymbol{y}'(t)). \tag{43}$$

Their algorithm can be applied to initial conditions $(\boldsymbol{y}(0), \boldsymbol{y}'(0))^\intercal = (\boldsymbol{a}, \boldsymbol{v})^\intercal$, and two-point linear boundary conditions $(\boldsymbol{y}(0), \boldsymbol{y}(1))^\intercal = (\boldsymbol{a}, \boldsymbol{b})^\intercal$.

The authors base their numerical algorithm on probabilistic inference with a Gaussian process prior over $\mathcal{GP}(\boldsymbol{y}(t); \boldsymbol{\mu}(t), k(t,s))$ and linear observations with Gaussian observation noise. Their prior mean is chosen as linear functions, i.e., $\boldsymbol{\mu}(t) = \boldsymbol{a} + t\boldsymbol{v}$ and $\boldsymbol{\mu}(t) = \boldsymbol{a} + t(\boldsymbol{b} - \boldsymbol{a})$ for the IVP and BVP, respectively. They choose a factorizing prior covariance function $\mathrm{cov}[(\boldsymbol{y}(t))_i, (\boldsymbol{y}(s))_j] = (\boldsymbol{V})_{ij} k(t,s)$ with positive semi-definite inter-dimensional covariance matrix $\boldsymbol{V} \in \mathbb{R}^{D \times D}$ and square-exponential kernel $k(t,s) = \exp(-|t-s|^2 [2\lambda^2]^{-1})$ for the spatial covariance. This factorization can be written compactly with the Kronecker product $\mathrm{cov}(\boldsymbol{y}(t), \boldsymbol{y}(s)) = \boldsymbol{V} \otimes k(t,s)$.

The initial or boundary conditions are expressed as Gaussian likelihoods

$$P(\boldsymbol{y}(0), \boldsymbol{y}'(0)) = \mathcal{N}(\boldsymbol{y}(0);\, \boldsymbol{a}, \boldsymbol{C}_a)\mathcal{N}(\boldsymbol{y}'(0);\, \boldsymbol{v}, \boldsymbol{C}_v), \tag{44a}$$

or

$$P(\boldsymbol{y}(0), \boldsymbol{y}(1)) = \mathcal{N}(\boldsymbol{y}(0);\, \boldsymbol{a}, \boldsymbol{C}_a)\mathcal{N}(\boldsymbol{y}(1);\, \boldsymbol{b}, \boldsymbol{C}_b), \tag{44b}$$

for IVPs and BVPs, respectively, where $\boldsymbol{C}_x$ is a $D \times D$ positive semi-definite covariance associated with vector $\boldsymbol{x}$. For classical problems, $\boldsymbol{C}_x = \boldsymbol{0} \in \mathbb{R}^{D \times D}$ and (44) are Dirac distributions. These can be used to form (predictive) posterior Gaussian process distributions $\mathcal{GP}(\boldsymbol{y}(t); \underline{\boldsymbol{\mu}}_{\mathcal{D}}(t), \underline{\boldsymbol{k}}_{\mathcal{D}}(t,s))$ with mean and covariance

$$\boldsymbol{G} = \left[\boldsymbol{V} \otimes \begin{pmatrix} k(0,0) & k^{\partial}(0,0) \\ {}^{\partial}k(0,0) & {}^{\partial}k^{\partial}(0,0) \end{pmatrix} + \begin{pmatrix} \boldsymbol{C}_a & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_v \end{pmatrix}\right] \tag{45a}$$

$$\underline{\boldsymbol{\mu}}_{\mathcal{D}}(t) = \boldsymbol{\mu}(t) + \left[\boldsymbol{V} \otimes (k(t,0)\; k^{\partial}(t,0))\right] \boldsymbol{G}^{-1} \begin{bmatrix} \boldsymbol{a} - \boldsymbol{\mu}(0) \\ \boldsymbol{v} - \boldsymbol{\mu}'(0) \end{bmatrix} \tag{45b}$$

$$\underline{\boldsymbol{k}}_{\mathcal{D}}(t,s) = \boldsymbol{V} \otimes k(t,s) - [\boldsymbol{V} \otimes (k(t,0)\; k^{\partial}(t,0))] \boldsymbol{G}^{-1} \left[\boldsymbol{V} \otimes \begin{pmatrix} k(0,s) \\ k^{\partial}(0,s) \end{pmatrix}\right] \tag{45c}$$

in the case of the IVP and similarly for the BVP. We have introduced the short-hand notation $k^{\partial}(t,s) = \frac{\partial}{\partial s} k(t,s)$, ${}^{\partial}k(t,s) = \frac{\partial}{\partial t} k(t,s)$ and so forth in Eq. (45).

The predictive posterior distribution is used to construct new data points which in turn generates a new predictive posterior distribution. This iterative process is then repeated for a pre-selected amount of steps.

Let $\Delta = \{t_1, \dots, t_N\} \in [0,1]$ be a set of mesh points. Use Eq. (45) to compute the "observation" $\boldsymbol{z}_n = \boldsymbol{f}(\underline{\boldsymbol{\mu}}_{\mathcal{D}}(t_n), \underline{\boldsymbol{\mu}}'_{\mathcal{D}}(t_n))$. Since $\boldsymbol{\mu}(t_n) \approx \boldsymbol{y}(t_n)$ and similarly for $\boldsymbol{\mu}'(t_n)$, one might assume that $\boldsymbol{z}_n \approx \boldsymbol{y}''(t_n)$ by (43). Specifically, the model states

$$\begin{aligned} P(\boldsymbol{z}_n \mid \boldsymbol{y}''(t_n)) &= \mathcal{N}(\boldsymbol{z}_n; \boldsymbol{y}''(t_n), \boldsymbol{\Lambda}_n) \\ \boldsymbol{\Lambda}_n &= U^\intercal \boldsymbol{\Sigma}_{yy,n} U + \left| U^\intercal \boldsymbol{\Sigma}_{c\dot{c},n} \dot{U} \right| \\ &\quad + \left| \dot{U}^\intercal \boldsymbol{\Sigma}_{\dot{c}c,n} U \right| + \dot{U}^\intercal \boldsymbol{\Sigma}_{\dot{c}\dot{c},n} \dot{U} \\ \begin{pmatrix} \boldsymbol{\Sigma}_{yy,n} & \boldsymbol{\Sigma}_{yy',n} \\ \boldsymbol{\Sigma}_{y'y,n} & \boldsymbol{\Sigma}_{y'y',n} \end{pmatrix} &= \operatorname{cov}\left(\begin{pmatrix} \boldsymbol{y}(t_n) \\ \boldsymbol{y}'(t_n) \end{pmatrix}, \begin{pmatrix} \boldsymbol{y}(t_n) \\ \boldsymbol{y}'(t_n) \end{pmatrix} \right) \in \mathbb{R}^{2D \times 2D} \end{aligned} \tag{46}$$

and $U, \dot{U} \in \mathbb{R}^{D \times D}$ are bounds $\left| \frac{\partial (f(\boldsymbol{y}, \boldsymbol{y}'))_j}{\partial (\boldsymbol{y}_i)} \right| \leq U_{ij}$ on the partial Jacobians of $\frac{\partial}{\partial \boldsymbol{y}} f(\boldsymbol{y}, \boldsymbol{y}')$ with respect to $\boldsymbol{y}$ and $\boldsymbol{y}'$, respectively, which must be provided by the user. The intuition behind the modeling assumption (46) is that we do not expect $\boldsymbol{z}_n$ to be an accurate evaluation of the second derivative of $\boldsymbol{y}$ at t_n, but we expect the error to be bounded by the first-order term.

Conditioning on this observation yields an updated predictive posterior similar to (45). After all initial derivative observations $\boldsymbol{z}_n^{(0)} = \boldsymbol{z}_n$ have been added, the algorithm iteratively refines each data point $\boldsymbol{z}_n^{(s)} = \boldsymbol{f}(\underline{\boldsymbol{\mu}}_{\mathcal{D}^c_{s,n}}(t_n), \underline{\boldsymbol{\mu}}'_{\mathcal{D}^c_{s,n}}(t_n))$, $s = 1, \dots, S$, where $\mathcal{D}^c_{s,n} = \{\boldsymbol{z}_m^{(s-1)} \mid m = 0, \dots, N, m \neq n\}$ is the data set withholding $\boldsymbol{z}_n^{(s-1)}$. Crucially, the observation noise matrices $\boldsymbol{\Lambda}_n$ are kept fixed during this process as it seems to overfit otherwise. Structurally, this is similar to the function iteration process (32) of implicit numerical methods.

The algorithm relies on a set of hyper-parameters. The authors use a fixed mesh Δ and fixed number of refinements S. The remaining free parameters are the parameters of the prior covariance. In the application of Hennig and Hauberg [92], the vector field f of the problem (43) is defined by a given data set $\mathcal{D}_f = \{\boldsymbol{x}_k \in \mathbb{R}^D \mid k = 1, \dots, K\}$. The inter-dimensional covariance amplitude $\boldsymbol{V} \in \mathbb{R}^{D \times D}$ is chosen a priori in an empirical Bayes fashion $\boldsymbol{V} = (\boldsymbol{a} - \boldsymbol{b})^\intercal \boldsymbol{V}_{\mathcal{D}_f} (\boldsymbol{a} - \boldsymbol{b})$ where $\boldsymbol{V}_{\mathcal{D}_f} = \sum_{k=1}^K (\boldsymbol{x}_k - \mathbb{E}[\mathcal{D}_f])(\boldsymbol{x}_k - \mathbb{E}[\mathcal{D}_f])^\intercal$ is the sample covariance matrix. The kernel length scale λ is optimized during execution with type-II maximum likelihood (see Section 2.6).

Algorithm 1 presents the pseudo-code for the proposed method. We want to draw attention to the outer loop starting in line 2. As data points and uncertainty covariance matrices are depending on the hyper-parameters, it is necessary to restart the algorithm if a change in hyper-parameters was determined. In practice, this can lead to a cyclic sequence of hyper-parameters $(\lambda_0, \lambda_1, \dots, \lambda_k = \lambda_0, \dots)$ which is prevented by stopping the outer loop when a cycle is detected.

Algorithm 1: BVP solver from Hennig and Hauberg [92]

Require: BVP: $f(\boldsymbol{y}, \boldsymbol{y}'), \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}_a, \boldsymbol{C}_b, U, \dot{U}$
Require: Hyper-parameters: $\Delta, S, \boldsymbol{V}, k(s,t), \lambda_0$
$\mu(t) \leftarrow \boldsymbol{a} + t(\boldsymbol{b} - \boldsymbol{a})$
while do $|\lambda_{k+1} - \lambda_k| > \varepsilon$
  $\mathcal{D} \leftarrow \{(0, \boldsymbol{a}, \boldsymbol{C}_a), (1, \boldsymbol{b}, \boldsymbol{C}_b)\}$
  for $s = 0, \ldots, S$ **do**
    for $n = 0, \ldots, N$ **do**
      if $s == 0$ **then**
        $\Lambda_n \leftarrow U^\intercal \underline{k}_{\mathcal{D}}(t_n, t_n) U + \left| U^\intercal \underline{k}^{\partial}_{\mathcal{D}}(t_n, t_n) \dot{U} \right| + \cdots$
          $\cdots + \left| \dot{U}^\intercal \underline{{}^{\partial}k}_{\mathcal{D}}(t_n, t_n) U \right| + \dot{U}^\intercal \underline{{}^{\partial}k^{\partial}}_{\mathcal{D}}(t_n, t_n) \dot{U}$
      else
        $\mathcal{D} \leftarrow \mathcal{D} \setminus \{(t_n, \boldsymbol{z}^{(s-1)}, \boldsymbol{\Lambda}_n)\}$
        $\boldsymbol{\Lambda}_n \leftarrow \boldsymbol{\Lambda}_n$
      end if
      $\boldsymbol{z}_n^{(s)} \leftarrow \boldsymbol{f}(\underline{\mu}_{\mathcal{D}}(t_n), \underline{\mu}'_{\mathcal{D}}(t_n))$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(t_n, \boldsymbol{z}_n^{(s)}, \boldsymbol{\Lambda}_n)\}$
    end for
  end for
  $\lambda_{k+1} \leftarrow \operatorname{argmax}_\lambda P(\boldsymbol{z}_{0:N} \mid \boldsymbol{V}, \lambda)$
end while
return $\mathcal{GP}(\boldsymbol{y}(t); \underline{\mu}_{\mathcal{D}}(t), \underline{k}_{\mathcal{D}}(s,t))$

4.4 The randomized ODE solver from Chkrebtii et al.

In Chkrebtii, Campbell, Calderhead, and Girolami [39] (first communicated in Chkrebtii, Campbell, Calderhead, and Girolami [38]), the authors propose various Monte Carlo algorithms for the solution of (general) differential equations. In this presentation, we will adapt their notation to our standard.

Structurally, the algorithms of Chkrebtii, Campbell, Calderhead, and Girolami [39] and Hennig and Hauberg [92] differ mostly in their *application* and *representation* of the involved probability distributions. In Hennig and Hauberg [92], computations are based *deterministically* on the distribution parameters, and (intermediate) results determine distribution parameters in turn. In Chkrebtii, Campbell, Calderhead, and Girolami [39], probability distributions are typically represented by a (randomized) *set of samples* which can describe a richer set of non-parametric probability distributions and can be used in conjunction with many Monte Carlo algorithms. We will present a discussion on these different philosophies in Section 5.2.2.

These differences can be motivated through the different requirements of the application in Chkrebtii, Campbell, Calderhead, and Girolami [39]. Therein, the authors are interested in the *inverse problem* of determing the posterior distribution $P(\theta, \boldsymbol{y} \mid \boldsymbol{z}, \psi)$ of unknown

inverse problem

system parameters θ in $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}, \theta)$ *given* noisy observations $\boldsymbol{z} = \boldsymbol{y}_* + \nu$ from one realization of the system and hyper-parameters ψ, e.g., the solver configuration and hyper-prior parameters.

Algorithm 2 draws one sample from the conditional distribution $P(\boldsymbol{y} \mid \theta, \psi)$ of the ODE solution given its system parameters and hyper-parameters. Together with the prior distribution $P(\theta \mid \psi)$ and the conditional distribution $P(\boldsymbol{z} \mid \boldsymbol{y})$, a Metropolis-Hastings algorithm can be used to draw samples from $P(\theta, \boldsymbol{y} \mid \boldsymbol{z}, \psi)$ [39, Alg. 2].

Algorithm 2: IVP solver from Chkrebtii, Campbell, Calderhead, and Girolami [39]

Require: IVP: $\boldsymbol{f}(t, \boldsymbol{y}, \theta), \boldsymbol{y}_0$
Require: Hyper-parameters: $\psi = \{\Delta_N, \Delta_S, k_0(s,t), \lambda, \alpha, \dots\}$
1: $\mu(t) \leftarrow \boldsymbol{y}_0$
2: $k(s,t) \leftarrow \alpha^{-1} \int_{\mathbb{R}} \int_{t_0}^{s} k_0(y,x)\,\mathrm{d}y \int_{t_0}^{t} k_0(z,x)\,\mathrm{d}z\,\mathrm{d}x$
3: $\boldsymbol{f}_0 \leftarrow \boldsymbol{f}(t_0, \boldsymbol{y}_0, \theta)$ ▷ First update
4: $\mathcal{D} \leftarrow \{(t_0, \boldsymbol{f}_0, \boldsymbol{0})\}$
5: **for** $n = 1, \dots, N$ **do** ▷ Subsequent updates
6: $\quad \boldsymbol{s}_n(t_n) \leftarrow \boldsymbol{u}_n \sim \mathcal{N}(\boldsymbol{y}(t_n); \underline{\boldsymbol{\mu}}_{\mathcal{D}}(t_n), \underline{k}_{\mathcal{D}}(t_n, t_n))$
7: $\quad \boldsymbol{f}_n \leftarrow \boldsymbol{f}(t_n, \boldsymbol{s}_n(t_n), \theta)$
8: $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(t_n, \boldsymbol{f}_n, \underline{\partial k^{\partial}}_{\mathcal{D}}(t_n, t_n))\}$
9: **end for**
10: **return** $\boldsymbol{s}_{\Delta_S} \leftarrow \boldsymbol{u}_{\Delta_S} \sim \mathcal{GP}(\boldsymbol{y}(\Delta_S); \underline{\boldsymbol{\mu}}_{\mathcal{D}}(\Delta_S), \underline{k}_{\mathcal{D}}(\Delta_S, \Delta_S))$

The main algorithmic differences between Algorithm 1 and 2 are in line 6 (and to some extend in line 10). In Algorithm 2, the vector field $\boldsymbol{f}$ is evaluated at *samples* $\boldsymbol{u}_n$ of the respective predictive posterior. Empirically, this leads to more expressive posteriors at higher computational cost as each sample from the posterior process requires a separate run of the algorithm. The authors also suggest a slightly modified covariance function (Line 2), but the suggestions from both set of authors could be used in the respective other's work without further modification. The common structure will be presented in abstract form in Section 6.2.

Part III

Theory of Probabilistic IVP Solvers

5 Facets of probabilistic ODE solvers

The previous chapters presented the preliminaries of statistical and numerical methods as well as the state of probabilistic ODE solvers at the commencement of this research. In this chapter, we will discuss aspects of probabilistic numerical ODE solvers and derive a list of desiderata for the methods we want to develop. This will serve as a guiding principle for the material presented in Sections 6 and 7.

5.1 A taxonomy of PN ODE solvers

For the purposes of this thesis, we roughly classify the recent wave of publications on probabilistic ODE solvers into three categories which are also depicted in Figure 6:

1. basic probabilistic algorithms which mostly serve the purpose of establishing proof-of-concepts and *showcasing novel functionality*

2. algorithms whose main motivation stems from the study of the numerical error to provide *accurate uncertainty quantification*

3. algorithms whose main motivation stems from the interpretation of classical methods to provide *fast methods with additional probabilistic output*

This categorization is necessarily a bit fuzzy, but it helps in understanding the modeling decisions of publications in the field.

Prior to the work of this PhD thesis, there have only been three publications concerned with the probabilistic numerical solution of ODEs. The works by Skilling [199] and Hennig and Hauberg [92] have to be considered proof-of-concept studies as the algorithms presented therein have only been evaluated empirically. However, these pieces showcased potential applications of probabilistic integrators. Later publications within the same context are Schober, Kasenburg, Feragen, Hennig, and Hauberg [186] and Hauberg, Schober, Liptrot, Hennig, and Feragen [87] which will be presented later in this thesis.

The work by Chkrebtii, Campbell, Calderhead, and Girolami [39] was the first work to also provide a proof of convergence. Their motivation is modeling the *uncertainty* associated with the numerical solution as accurately as possible. Yet, their algorithm is prohibitively slow compared to the standard methods. In earlier work, Fierro and Torres [64] have presented a formal analysis of their algorithm, but

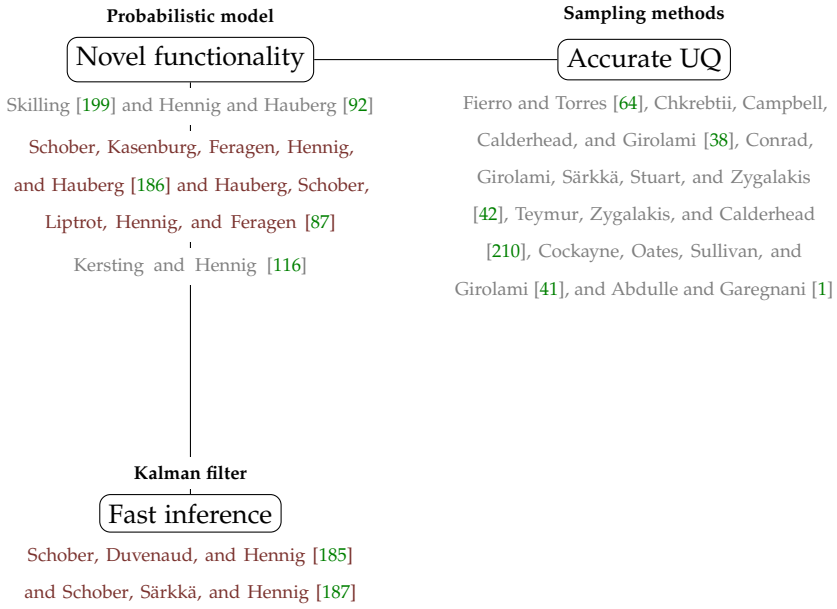

Figure 6: Based on the early works, research focus has split between methods more focused on accurate uncertainty quantification (UQ) and fast methods. Methods by the author are highlighted in red. Within categories, works are sorted by publication date.

their work has only recently attracted attention by Krebs [120] and had not received wide-spread attention before.

Following the work of Diaconis [56], the approach of this thesis focuses on *fast methods* that resemble the classical solvers presented in Chapter 3. Thus, we will try to answer the following questions: *Is it possible to cast certain existing methods as (exact) inference in a Gaussian probability model? What would be the resulting interpretation of said methods and how can this knowledge be used to construct fast (linear) probabilistic numerical ODE solvers?*

At the time of completion of this thesis, not all questions have been fully answered. However, we have made enough progress to present some of the connections between standard methods and their probabilistic counter-parts as well as some consequences thereof for practical numerical methods.

5.2 From classical to probabilistic numerical methods

Currently, there are two different goals of probabilistic numerical methods—accurate uncertainty quantification and fast inference—which require according properties of proposed algorithms. For instance, to provide accurate uncertainty quantification for a general class of problems, one must specify models with a large degree of flexibility and with as few prior assumptions as possible.

We are interested in fast methods that resemble classical solvers. In the following, we will try to make the point that this setting

provides more leeway in its freedom of approximation but requires stronger runtime guarantees. The result will be methods which are not driven by their model assumptions but by their *algorithm desiderata*.

To construct a probabilistic numerical method, we define the following list of desiderata that an algorithm should fulfill. These properties will be defined and motivated in turn below.

Probabilistic inference: required computations should be operations on probability distributions.

Global definition: the probabilistic model should not depend on the discretization mesh.

Analytic guarantees: the algorithm's output should have desirable analytic properties.

Deterministic execution: when run several times on the same problem, the algorithm produces the same output each time.

Speed: the execution time should not be prohibitively slow.

Problem adaptive: the user should be able to control the accuracy/speed trade-off.

5.2.1 The value of a probabilistic model

In this section, we explain and motivate the first three items from our list of desiderata in turn—*probabilistic inference*, *global definition* and *analytic guarantees*.

On a high-level view, numerical algorithms can be described as combinations of *tractable approximating function classes* and *computation strategies* for *informative values*. Analyses of numerical methods show to what level the approximations can converge to the true problem solution and how fast the computation strategies can be carried out.

Accepting the probabilistic approach as a framework for plausible reasoning [104, 44, 93], we require a *probability distribution* P_Y over the numerical solution Y. The computations necessary for the construction of P_Y should be interpretable as (approximate) probabilistic inference. The motivation behind this requirement is that there should not be an analysis gap between statistical and numerical computations. This is particularly beneficial when the differential equation solver is embedded in a longer chain of computations [41]. This allows us to build fine-tuned methods adapting to sources of data uncertainty and computational approximation during runtime. Furthermore, they provide richer feedback of approximation quality which is demonstrated empirically in Sections 9 and 10.

Method	glob. def.?	determ.?	guarantees?
Skilling [199]	✓	×	×
Chkrebtii et al. [39]	✓	×	≈
Schober et al. [185]	≈	✓	≈
Conrad et al. [42]	×	×	≈
Kersting & Hennig [116]	✓	✓	×
Teymur et al. [210]	×	×	≈
PFOS (this thesis)	✓	✓	✓

Table 2: Properties of existing PNM ODE solvers. A ✓ indicates that the method satisfies a given property, a × indicates that a method does not satisfy a given property, and a ≈ indicates that a property holds with some restrictions.

global definition

Furthermore, a probabilistic IVP solver shall be called *globally defined* on its input domain $\mathbb{T}$, if its probabilistic interpretation does not depend on the discretization mesh Δ. PNMs satisfying this property provide two benefits. Users may evaluate the (predictive posterior) distribution P_Y for any $t \in \mathbb{T}$. In particular, users may evaluate $P_Y(Y(t))$ for $t \notin \Delta$. Thus, users may request $P_Y(Y(t_s)), t_s \in \Delta_S$ and the support of a user-defined mesh Δ_S is not a separate requirement. Secondly, this implies that the inference can be paused and continued after every expansion from $\Delta_n \mapsto \Delta_{n+1}$. In principle, this also enables iterative refinement of the solution quality based on its prediction uncertainty.

Table 2 lists PNM ODE solvers that have been proposed in the literature. A ✓ indicates that the method satisfies a given property, a × indicates that a method does not satisfy a given property, and a ≈ indicates that a property holds with some restrictions. The listing shows that almost all methods proposed so far are globally defined. Furthermore, we see that the definition is independent of a method being sampling-based or not. The method proposed by Conrad et al. [42] is a generative process on sub-intervals $[t_n, t_{n+1}] \subset \mathbb{T}$ based on a numerical discretization. It is easy to construct two different meshes Δ_n, Δ_n' that define different distributions for $P_Y(Y(t_n))$ in the case of $y' = \lambda y$ and a general argument can be made from this example. In Teymur et al. [210], the predictive posterior is only defined on the discretization mesh. This defect is not for lack of definition, but a consequence of the underlying numerical method the probabilistic algorithm is built upon. Since the method is defined on a windowed data frame, it is easy to construct a mesh such that the prediction $Y(t)$ at time t will be different depending on the window $[t_{n-i}, \ldots, t_{n+j}] \ni t$ is considered to be part of.

The analysis in Schober et al. [185] proposes two main modes of operation: naive chaining and probabilistic continuation. Naive chaining is not a globally defined method since mesh points t_n are

part of adjacent Runge-Kutta blocks and the corresponding predictive posterior $P_Y(Y(t_n))$ is different for these two blocks. Probabilistic continuation is globally defined, but there no convergence theory has been presented for this case.

This leads to the next item on the list and the last point discussed in this section: *analytic guarantees*. As in any mathematical discipline, the performance of numerical algorithms are subject to rigorous analysis. These typically take the form of limiting behavior for certain properties of a code.

analytic guarantees

For IVPs, where the existence of a solution can be proven under mild conditions, an algorithm should converge under equally mild conditions. The examples of Section 3.2 are shown not only to converge for $\lim h \to 0$, but that the error can be bounded by a known function. This yields a high rate of convergence even in the non-limit case. Another important property are the various classes of stability for IVP solvers which are related to problem stiffness (see Section 3.3).

5.2.2 Requirements for practical algorithms

This section discusses the remaining three desiderata—*deterministic execution*, *speed* and *problem adaptiveness*. We will argue that these three properties are desiderata for *practical algorithms*. We will now look at these properties in turn.

In this thesis, a deterministic algorithm has to be understood in contrast to Monte Carlo methods which rely on (pseudo-) random numbers. In fact, our argument *for* deterministic methods could mainly be thought of as an argument *against* random numbers.

Before we start, we want to highlight that there are cases where the usage of random numbers *is* called for, in particular in the setting of high-dimensional problems where the stochastic convergence is provably faster than its deterministic counter-part (see Section 2.5).

In the context of probabilistic numerics, however, randomized methods are typically applied to guard against model errors (Cockayne, Oates, Sullivan, and Girolami [41] and Rainforth [174]). The sampling mechanism adds independence over steps during runtime execution, the argument being that discretization errors exhibit an independent behavior per step. It has been shown that floating point *rounding errors* can be accurately modeled by independent Gaussian noise [98, 94].

From a more abstract view, modeling *discretization errors* by independent additive Gaussian noise is an additional assumption that is omitted in deterministic schemes. As a consquence, the randomized algorithms should indeed fare better *if* the discretization errors of a

combination of problem and method at hand are of this kind. For instance, one setting where this might come up are systems with chaotic attractors.

In practice, however, most problems do not converge to a chaotic attractor, in which case the independence of the discretization error is doubtful. In these cases, discretization errors typically fulfill an unknown, but systematic drift. This is obvious from the pathological example $y' = c$ for some constant $c \neq 0$. Even the explicit Euler scheme can recover the exact solution $y(t) = ct$ without discretization error, so any non-zero discretization error assumption is wrong. In practically relevant problems, systematic errors can be observed with unstable methods or stiff problems ([82, §III.3]).

Secondly, the finite sample size of the randomized method adds another problem in its own right (O'Hagan [160]). Since sampling methods are only required if the posterior distribution, i.e., its density, cannot be computed analytically, the user has to check the convergence of the empirical density estimate. This problem can sometimes be tackled by analyzing the convergence rates, e.g., as in Cockayne, Oates, Sullivan, and Girolami [41]. While we had to worry about one convergence before, we now have to worry about two.

To make matters worse, this obstacle of the (theoretical) algorithm has multiple facets for any actual implementation. Markov Chain Monte Carlo methods, in particular, can suffer from a *missing mode problem* if the sampling chain cannot escape a local high density area during the finite runtime (Betancourt [16, §2.3]). In any case, the indeterminism makes a program harder to debug as guarantees can only be checked up to probabilistic uncertainty (Gentle [70] and Cook [43]). In deterministic methods, the pre- and postconditions are typically easier to establish. For the methods presented in Chapter 6, these are identical to the requirements for classical numerical solvers. The disadvantage comes in the form of very weak global bounds, but these are known to hold under every condition.

The final two requirements—*speed* and *problem adaptiveness*—can be understood as two different aspects of the common desiderata of low computational budget. The runtime of a numerical code should be as short as possible.

In the context of ODE solvers, speed is a function both of *convergence order* and *computational complexity*. Classical algorithms can achieve higher order convergence at linear computational cost (see Section 3.2). A probabilistic numerical algorithm may be a little more time consuming, but its runtime behavior needs to be on a similar scale. Otherwise, other means of error quantification such as *sensitivity analysis* [134, 30] is to be preferred.

Problem adaptiveness describes the property of a code to allocate an efficient integration strategy automatically for a specific task. This

property usually operates *within* the convergence order and complexity class constraints. In the context of ODE solvers, the most common functionality in this regard is the ability to select the coarseness of the discretization mesh based on accuracy requirements. This is not only important as a convenience tool, but might be a necessity for non-expert users who might not be able to choose an appropriate strategy by themselves.

5.3 A roadmap to probabilistic numerical methods

Chapter 6 will present a model and corresponding algorithm that fulfills the desired properties. The description will follow the progression of its mathematical building blocks. As a consequence, this might obfuscate its chronological discovery making it harder for the uninitiated reader to follow the motivation. Thus, before we start with the mathematical description we will outline a roadmap to the methods presented in the next chapter.

Following the example of O'Hagan [159] and Diaconis [56], we want to find methods that closely resemble existing numerical methods for the solution of initial value problems. Reflecting on the structure of the methods presented in Chapter 3, it is possible to construct numerical methods from *linear combinations* of *function evaluations* and suitable *weights*. More importantly, the appropriate weights can be determined *statically* (at compile time) working for a large class of dynamics functions. Reframing this, one could say that the optimality of a set of weights is a property of the approximation model, *not* of the specific problem at hand.

The key insight now is to observe that Gaussian process regression models fit this description; the posterior process is given by linearly weighted function observations (see Eq. (8)). Additionally, the weights are only determined by the *covariance structure* which is purely a property of the model, *not* of the specific problem at hand.[1]

Given only the location of the function evaluation, but not the evaluation itself, the weights are completely determined.

The rest of the program is almost a natural consequence: search for those Gaussian models which produce the weights of any classical method. A consequence of the *global definition* property will be that this analysis has to be applied twice: once for the initial ramp up phase and once for the stationary phase of the algorithm. In fact, it will turn out that globally defined methods have strong connections to *spline smoothing* via the equivalences found by Kimeldorf and Wahba [117] and Wahba [217]. We conjecture that this severly limits the space of all globally defined methods and it is an open problem to trade off this property against approximation quality.

[1] Hyper-parameters are often considered to be problem specific, but can alter the weights. Thus, this statement *per se* is controversial. However, in the following, we will consider models whose posterior mean is independent of the hyper-parameter choice and the statement is unproblematic.

The Probabilistic Filtering ODE Solver

This chapter introduces a concrete probabilistic model for the numerical solution of IVPs. We will prove that the maximum a posteriori estimator resulting from this model conincides with certain classical numerical methods. As a consequence, our probabilistic numerical method shares analytic guarantees of the classical methods which guarantee convergence at favorable rates.

This chapter mostly follows the presentation given in Schober, Särkkä, and Hennig [187]. Additionally, Section 6.6 presents the theoretical results from Schober, Duvenaud, and Hennig [185] in the notation of [187].

6.1 Gauss-Markov process priors for IVPs

Our approximation model of the true solution $y(t)$ is a vector

$$\boldsymbol{x}(t) = (y^{(0)}(t), \dots, y^{(q)}(t))^\mathsf{T}, \tag{47}$$

where $y^{(i)}(t)$ is the true ith derivative of $y(t)$ at time t. We represent the prior uncertainty about $\boldsymbol{x}(t)$ by the distribution $P(\boldsymbol{X}_t)$ of the random variable $\boldsymbol{X}_t$—or more generally as the law $P_{\boldsymbol{X}}$ of the stochastic process $\boldsymbol{X}$—which is then conditioned on the observed values.

As has been argued in Section 5.3, the goal is to find models that share the underlying building blocks of classical general purpose methods, i.e., linear methods with low (linear) inference cost. It has been shown in Section 2.4 how these models can be constructed from *linear time-invariant (LTI) stochastic differential equations (SDEs)*.

Here, we consider models of the form

$$\mathrm{d}\boldsymbol{X}_t = \underbrace{(\boldsymbol{U}_{q+1} + \boldsymbol{f}^\mathsf{T}\boldsymbol{e}_q)}_{=\boldsymbol{F}} \boldsymbol{X}_t \,\mathrm{d}t + \underbrace{\boldsymbol{e}_q}_{=\boldsymbol{L}} \,\mathrm{d}W_t, \tag{48}$$

where $\boldsymbol{L}$ is the last standard basis vector $\boldsymbol{e}_q$ and $\boldsymbol{F}$ is a (transposed) companion matrix. That is, $\boldsymbol{U}_{q+1}$ denotes the upper shift matrix and the row vector $\boldsymbol{f}^\mathsf{T}$ contains the coefficients in the last row of $\boldsymbol{F}$. In this case, the vector-valued process $\boldsymbol{X}_t = (X_{t,0}, \dots, X_{t,q})^\mathsf{T}$ obtains the interpretation $\boldsymbol{X}_t = (Y_t, Y_t', \dots, Y_t^{(q)})^\mathsf{T}$, because the form of $\boldsymbol{F}$ and $\boldsymbol{L}$ implies that the realizations of Y_t are q-times continuously differentiable on $\mathbb{R}$. Later, we will also consider scaled systems $\tilde{\boldsymbol{X}} = \boldsymbol{B}\boldsymbol{X}$ with an invertible linear transformation $\boldsymbol{B}$. In this case, we denote by $\boldsymbol{H}_i$

the matrix that projects onto the ith derivative $Y_t^{(i)} = \boldsymbol{H}_i \tilde{\boldsymbol{X}}_t = e_i^\intercal \boldsymbol{B} \boldsymbol{X}_t$. Two particular models of this type are the q-times *integrated Wiener process* (IWP(q)) and the continuous auto-regressive processes of order q. Detailed introductions can be found, for example, in [114, 161, 181]. SDEs can also be seen as path-space representations of more general temporal Gaussian processes arising in machine learning models [183].

Models of the form (48) are also related to nonparametric spline regression models [216] which often have a natural interpretation in frequentist analysis [117]. Conceptually, these models are a compromise between globally defined parametric models, which might be too restrictive to achieve convergence, and local parametric models, which might be too expressive to be captured by a globally defined probability distribution. Models of this type have been studied in the literature [141, 5], but the presentation here starts from other principles.

The choice of prior $P(\boldsymbol{X})$ in Eq. (48) can be interpreted as a *prior assumption* or *belief* encoded in the algorithm, in the sense that the algorithm amounts to an autonomous agent. We emphasise that if one adopts this view, then the results reported in later sections amount to an external analysis of the effects of these assumptions. That is, we will show that if the agent is based on this prior measure $P_{\boldsymbol{X}}$ with a likelihood to be defined in Section 6.3, it gives rise to a posterior distribution with certain desirable properties. By contrast, one could also take a more restrictive standpoint internal to the algorithm, and state that the proposed method works well if the true solution $\boldsymbol{x}$ is indeed a sample from $P(\boldsymbol{X})$. This is expressly *not* our viewpoint here, and it would be a flawed argument, too, given that in practice, $\boldsymbol{x}$ is defined through the ODE, thus patently not a sample from any stochastic process.

For the rest of this manuscript, we will focus on the q-times integrated Wiener process IWP(q), which is defined by

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{U}_{q+1} \boldsymbol{X}_t \,\mathrm{d}t + \boldsymbol{e}_q \,\mathrm{d}W_t.$$

In this case, $\boldsymbol{f}^\intercal = (0, \ldots, 0)$ and there is no feedback from higher states $X_{t,i}$ to lower states $X_{t,j}, i < j$. In particular, this process is non-stationary and does not revert to the initial mean $\boldsymbol{m}_{t_*}$. In this system, the (discrete) drift matrix $\boldsymbol{A}(h)$ and the diffusion matrix $\boldsymbol{Q}(h, \sigma^2)$ can be be computed analytically

$$\begin{aligned} (\boldsymbol{A}(h))_{ij} &= \mathbb{1}_{i \leq j} \frac{h^{j-i}}{(j-i)!}, \\ (\boldsymbol{Q}(h))_{ij} &= \sigma^2 \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!}, \end{aligned} \tag{49}$$

which can be derived directly from Eq. (13). Notice how $\boldsymbol{Q}(h)$ linearly depends on the diffusion scale σ^2. It will be convenient to write $\boldsymbol{Q}(h)$ in this explicit multiplicative way $\boldsymbol{Q}(h) = \sigma^2 \overline{\boldsymbol{Q}}(h)$.

6.2 Data generation mechanism

Many problems in statistics assume the existence of an externally produced, thus fixed data set $\{(t_n, z_n) \mid t_n \in \Delta\}$ and develop appropriate solutions from there. An analogous concept in numerical algorithms for solving differential equations would be to pose a global discretization scheme and to obtain a solution with other tools from numerical analysis. Methods of this type are often applied to boundary value problems (BVPs) and partial differential equations (PDEs) where the integration domains need to be specified a priori in any case. Cockayne et al. [41] take this approach by assuming a fixed information operator A. However, there are cases where the final time T cannot be stated beforehand, when the quantity of interest depends on a qualitative behavior of the solution. For example, in modeling of chemical reactions, an user might be interested in the long-term behavior of the compounds and it is unknown when the reaction reaches equilibrium.

In contrast, many numerical IVP solvers proceed in a step-by-step manner. Having computed a numerical approximation $P_{Y \mid z_{[n]}}$ on the mesh Δ_n, a *prediction* y_{n+1}^- of $y(t_{n+1})$ is used to *evaluate* $f(t_{n+1}, y_{n+1}^-)$, and the resulting output z_{n+1} is used to *update* the approximation $P_{Y \mid z_{[n+1]}}$ on the extended mesh Δ_{n+1}. For example, in a deterministic IVP the data (t_0, y_0) can be used to construct the observation $z_0 = f(t_0, y_0)$ which is $y'(t_0) \sim \delta(z_0 - y'(t_0))$ in the probabilistic interpretation. This serves as a corner case for the general situation. Setting $t_{-1} = t_0$ and $z_{-1} = y_0$, it follows that $y(t_0) \sim \delta(z_{-1} - y(t_{-1}))$ and the initial value requires almost no special treatment. The concept is illustrated in Algorithm 3 and can, in principle, be extended indefinitely, at constant cost per step. The term *predict–evaluate–correct (PEC)* or *predictor–corrector* methods have a more technical meaning in classic textbooks [82, 53], but the idea is common to many numerical IVP solvers. Chkrebtii et al. [39] calls the process of evaluating $f(t_n, y_{t_n}^-)$ with tentative $y_{t_n}^-$ to generate z_n a *model interrogation*. From a statistical perspective, this concept of active model interrogation is similar to the sequential analysis of Wald [218, 162].

Algorithm 3 conveys the general idea of a probabilistic ODE solver while omitting parameter tuning aspects like error control and step size selelgction. The exact form of lines 5 and 6 depends on the choice of observation construction and data likelihood model. Without data, the prior induces a probability distribution on the hidden

Algorithm 3: Active probabilistic model

1: $P(X_{t_{-1}}) \leftarrow$ MODEL(hyper-parameters)
2: $P(X_{t_0} \mid z_{[0]}) \leftarrow$ INIT$(f, \mathbb{T}, y_0, P(X_{t_{-1}}))$
3: **for** $n = 0, \ldots, N-1$ **do**
4: $\quad P(X_{t_{n+1}} \mid z_{[n]}) \leftarrow$ PREDICT$(P(X_{t_n} \mid z_{[n]}))$
5: $\quad P(z_{n+1} \mid X_{t_{n+1}}) \leftarrow$ OBSERVE$(f, P(X_{t_{n+1}}))$
6: $\quad P(X_{t_{n+1}} \mid z_{[n+1]}) \leftarrow$ UPDATE$(P(z_{n+1} \mid X_{t_{n+1}}), P(X_{t_{n+1}} \mid z_{[n]}))$
7: **end for**

state $\boldsymbol{X}_{t_n}$. It remains to construct an observation z_n and a likelihood model $P(z_n \mid \boldsymbol{X}_{t_n})$.

6.3 Observation assumptions

Recall from Section 6.1 the prior state-space assumption

$$\boldsymbol{X}_t = (Y_t, Y_t', \ldots, Y_t^{(q)})^\mathsf{T} \sim \mathcal{N}(\boldsymbol{m}_t, \boldsymbol{C}_t). \tag{50}$$

Combining Eqs. (19) and (50) gives

$$\mathcal{N}((\boldsymbol{m}_t)_1, (\boldsymbol{C}_t)_{11}) = P(Y_t') = f_\# \mathcal{N}((\boldsymbol{m}_t)_0, (\boldsymbol{C}_t)_{00})$$

where $f_\# \mu$ is the push-forward measure of μ. The exact form of that push-forward is not usually tractable for general f (one exception is linear ODEs, which of course do not require nontrivial numerical algorithms).

We will show below, however, that using an approximate Gaussian likelihood leads to good analytic properties of the resulting Gaussian posterior. This likelihood will be parametrized as

$$P\left(z_n \mid Y_{t_n}'\right) = \mathcal{N}(Y_{t_n}', R_n^2) \tag{51}$$

where z_n are the observations that have yet to be constructed and R_n^2 can be interpreted as an observation uncertainty. Another way to phrase Eq. (51) is to write

$$z_n = \boldsymbol{H}_1 \boldsymbol{X}_{t_n} + \nu,$$

where the latent variable $\nu = y'(t_n) - f(t_n, \boldsymbol{H}_0 \boldsymbol{X}_{t_n})$ captures the error between the estimated solution $f(t_n, \boldsymbol{H}_0 \boldsymbol{X}_{t_n})$ and the derivative of the true solution $y'(t_n)$. The approximation in Eq. (51) is to assign a centered Gaussian distribution $P(\nu) = \mathcal{N}(0, R_n^2)$ to this latent variable. Purely from a formal perspective, this ν is a "random variable", but we stress again that $P(\nu)$ captures *uncertainty* arising from lack of computational information about a deterministic quantity, not any physical sort of randomness in a frequentist sense. That is, solving the same IVP several times will always produce the exact same ν, be-

cause the algorithm is deterministic. But that same ν will always be just as unknown. Repeated runs will not refine the uncertainty. Figure 7 displays a graphical model corresponding to the construction. All current probabilistic numerical ODE solvers share this particular assumption (51) [199, 39, 185, 42, 116, 210]. The differences between these algorithms lies mainly in the prior on $\boldsymbol{X}$, and how the observation z_n is produced within the algorithm.

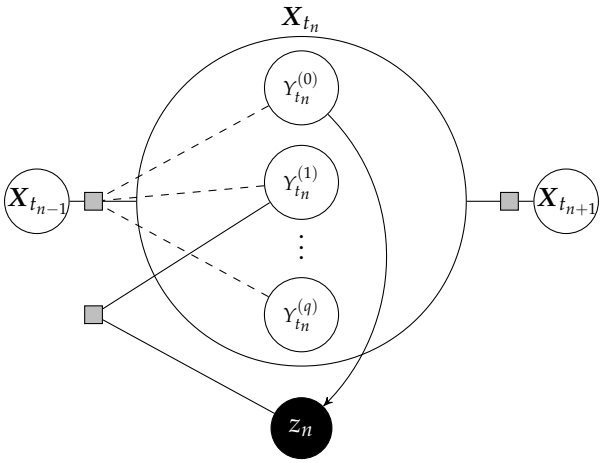

Figure 7: The graphical model corresponding to the proposed construction. White circles represent unobserved hidden states and the black circle represents the observed data. Gray squares represent a jointly normal distribution. The arrow indicates a model interrogation. An implied non-Gaussian factor between $Y^{(0)}(t_n)$ and z_n is ignored to obtain a practical algorithm.

It remains to construct z_n and R_n^2. One possible way to achieve this is to compute the expected value of the vector field f under the prediction for the true solution

$$z_n \leftarrow \mathbb{E}[f] = \int f(t_n, Y_{t_n})\, \mathcal{N}(Y_{t_n}; (\boldsymbol{m}^-_{t_n})_0, (\boldsymbol{C}^-_{t_n})_{00})\, \mathrm{d}Y_{t_n}, \tag{52}$$

where $\mathcal{N}(\boldsymbol{X}_{t_n}; \boldsymbol{m}^-_{t_n}, \boldsymbol{C}^-_{t_n}) = P(\boldsymbol{X}_{t_n} \mid z_{[n-1]})$ is the prediction distribution of $\boldsymbol{X}_{t_n}$ given the data $z_{[n-1]}$ and $\leftarrow$ denotes assignment in code.

With these conventions, two new issues emerge: the evaluation of the intractable Eq. (52) and the determination of R_n^2. Kersting and Hennig [116] propose to put

$$R_n^2 \leftarrow \int f(t_n, Y_{t_n})^2\, \mathcal{N}(Y_{t_n}; (\boldsymbol{m}^-_{t_n})_0, (\boldsymbol{C}^-_{t_n})_{00})\, \mathrm{d}Y_{t_n} - \mathbb{E}[f]^2$$

and to evaluate both integrals by Bayesian quadrature. Chkrebtii et al.'s [39] method draws a sample $u_n \sim \mathcal{N}((\boldsymbol{m}^-_{t_n})_0, (\boldsymbol{C}^-_{t_n})_{00})$, computes $z_n \leftarrow f(t_n, u_n)$ and R_n^2 is set to $(\boldsymbol{C}^-_{t_n})_{11}$. In light of Kersting and Hennig [116], this could be thought of as a form of Monte Carlo scheme to evaluate (52).

As a further restriction to the likelihood (51) more widely used by other probabilistic numerical solvers, we will here focus on models with vanishing $R_n^2 \to 0$. That is

$$\begin{aligned} z_n &\leftarrow f(t_n, (\boldsymbol{m}^-_{t_n})_0), \\ P(z_n \mid Y'_{t_n}) &= \delta(z_n - Y'_{t_n}) = \mathcal{N}(Y'_{t_n}, 0). \end{aligned} \tag{53}$$

This means the estimation node $y_{t_n}^-$ for the evaluation of f is simply the current mean prediction, and the resulting observation is modeled as being correct.

6.4 The complete algorithm

In this section, the algorithm components—the prior 6.1, the action rule 6.2 and the likelihood 6.3—are combined to produce a complete algorithm. The pseudo-code Algorithm 3 is turned into a code blueprint. The resulting Algorithm 4 serves as the groundwork for the main Definition 1 of this thesis.

The prior and the likelihood are linear models in the sense of Section 2.4, which stated that the predictive and filtering distribution can be computed by the linear-time algorithm known as the Kalman filter [113]. The resulting PFOS Algorithm 4 returns the filtering distribution at every mesh point $t_n = t_0 + hn$. For many applications, the posterior distribution at the end of the integration domain $\boldsymbol{m}_T, \boldsymbol{C}_T$ suffices, but the complete posterior can be computed from the filtering distribution.

Algorithm 4: Probabilistic Filtering ODE Solver (PFOS)

Require: IVP: $f, \mathbb{T}, \boldsymbol{y}_0$
Require: hyper-parameters: h, q, σ^2
$\boldsymbol{F} \leftarrow \boldsymbol{U}_{q+1} + \boldsymbol{e}_q^\intercal \boldsymbol{0}, \boldsymbol{L} \leftarrow \boldsymbol{e}_q, \boldsymbol{H}_0 \leftarrow \boldsymbol{e}_0, \boldsymbol{H}_1 \leftarrow \boldsymbol{e}_1^\intercal$ ▷ Create model
$\boldsymbol{A} \leftarrow \boldsymbol{A}(h), \boldsymbol{Q} \leftarrow \sigma^2 \overline{\boldsymbol{Q}}(h)$ ▷ Eq. (49)
$\boldsymbol{m}_{t_1}, \boldsymbol{C}_{t_1} \leftarrow \textsc{init}(f, \mathbb{T}, y_0, \mathit{model})$ ▷ see Section 6.6
for $n = 2, \ldots, N$ **do**
  $\boldsymbol{m}_{t_n}^- \leftarrow \boldsymbol{A}\boldsymbol{m}_{t_{n-1}}$ ▷ Predict (Eqs. (15))
  $\boldsymbol{C}_{t_n}^- \leftarrow \boldsymbol{A}\boldsymbol{C}_{t_{n-1}}\boldsymbol{A}^\intercal + \boldsymbol{Q}$
  $z_n \leftarrow f(t_0 + hn, \boldsymbol{H}_0 \boldsymbol{m}_{t_n}^-)$ ▷ Evaluate (Eq. (53))
  $\lambda_n \leftarrow z_n - \boldsymbol{H}_1 \boldsymbol{m}_{t_n}^-$ ▷ Update (Eqs. (16))
  $\boldsymbol{K}_n \leftarrow \boldsymbol{C}_{t_n}^- \boldsymbol{H}_1^\intercal [\boldsymbol{H}_1 \boldsymbol{C}_{t_n}^- \boldsymbol{H}_1^\intercal]^{-1}$
  $\boldsymbol{m}_{t_n} \leftarrow \boldsymbol{m}_{t_n}^- + \boldsymbol{K}_n \lambda_n$
  $\boldsymbol{C}_{t_n} \leftarrow (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_1) \boldsymbol{C}_{t_n}^-$
end for
return $\{\boldsymbol{m}_{t_N}, \boldsymbol{C}_{t_N},\ n = -1, \ldots, N\}$

Since the first version of the algorithm is not adaptive, the user has to provide a couple of hyper-parameters. We have decided to focus on the IWP(q) process, which has two remaining degrees of freedom: the integration order q and the diffusion scale σ^2. A low order of $q \in \{1, 2\}$ is a robust choice if the problem is expected to be mildly stiff. The diffusion σ^2 could also be adapted after-the-fact via type-II maximum likelihood optimization (see Section 2.6). At this point, no uniquely "right" formal way to fix this uncertainty

has been identified. One possible approach is due to Kersting and Hennig [116]. We will present an alternative later in Section 7.2.

Additionally, the user must provide an appropriate step size h. It is common to choose a number of steps N and set $h \leftarrow (T - t_0)N^{-1}$. Time and computational resources permitting, a standard technique to obtain reasonable step size is to lower the step size until the solution y_T at the final time T does no longer change in a desired number of digits.

Although this thesis focuses on the integrated Wiener process IWP(q), Algorithm 4 can, in principle, be applied to IVPs using any prior of the form (48). This motivates the following definition:

Definition 1. *A* probabilistic filtering ODE solver (PFOS) *is the Kalman filter applied to an initial value problem with an underlying Gauss–Markov linear, time-invariant SDE and Gaussian observation likelihood.*

For sake of completeness, Algorithm 5 presents the explicit form of the *Rauch-Tung-Striebel (RTS) smoother* [177] which can be used to compute the posterior smoothing distribution. The execution of the RTS smoother can also be used to simultaneously draw K samples $\boldsymbol{S}_{t_n,k}$ from the posterior distribution which is helpful to visualize the overall covariance structure. Algorithm 5 can also be extended to compute the posterior and draw samples on a user-defined mesh Δ_S, thus providing dense output functionality. The required changes are the additional logic for merging the output mesh Δ_N with Δ_S and extra book-keeping of drift and diffusion matrices which has been omitted for clarity.

In Algorithm 5, RANDN generates a $q+1$-dimensional standard multivariate normal sample. CHOL($\boldsymbol{C}$) computes the Cholesky factorization $\boldsymbol{L}\boldsymbol{L}^\intercal = \boldsymbol{C}$, $\boldsymbol{L}$ lower triangular. The Cholesky factorization can be circumvented with a bit of extra algebra if necessary [59].

These computations approximately double the required work. But they are only necessary if a dense output, i.e., a solution at a point other than T, is desired, for example, when the solution is required on a pre-defined grid which might not coincide with the mesh selected by an automated step size selection criterion (cf. Section 7.2), or when a large subset of points from $\mathbb{T}$ is selected to get a continuous plot of the numerical solution. Classical algorithms share this overhead.

We conclude this section by considering again the example from Section 3.1, page 25. Recall the IVP definiton

$$\begin{aligned} y' &= f(t,y) = f(y) = ry(1 - y/K), \\ y(t_0) &= y_0 = 1/10, \quad r = 3, K = 1, \end{aligned}$$

to be solved on the interval $[0, 1.5]$.

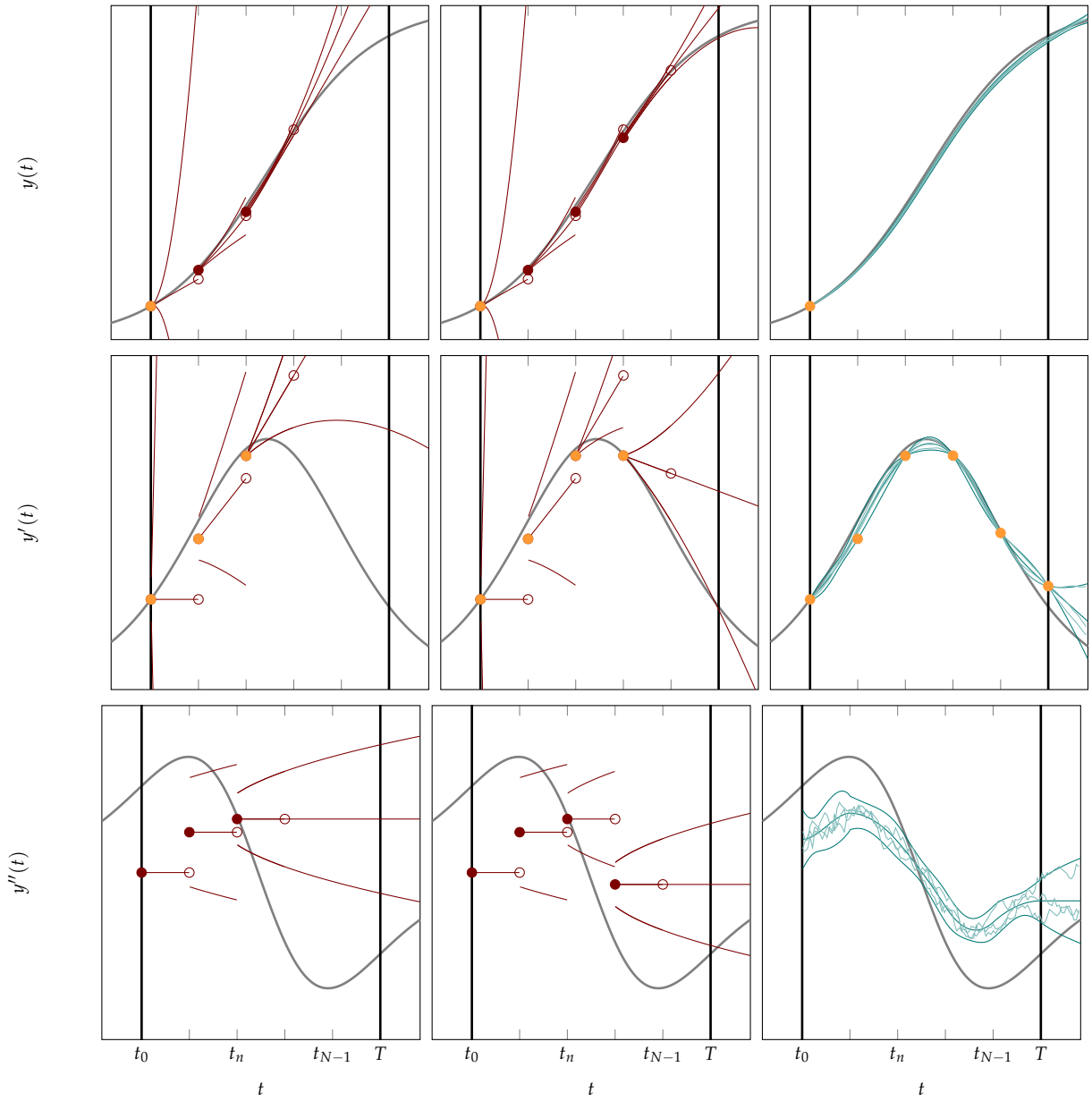

Figure 8: The 2-times integrated Wiener process $\mathrm{d}\boldsymbol{X}_t = \boldsymbol{U}_3\boldsymbol{X}_t\,\mathrm{d}t + \boldsymbol{e}_2\,\mathrm{d}W_t$ applied to the logistic growth problem $y' = ry(1 - y/K)$. The plot shows the true solution (gray) of the function y and its first two derivatives, as well as the numerical solution Y, given by its mean $\boldsymbol{m}_i$ (red line) and covariance $\boldsymbol{C}$, visualized as point-wise plusminus twice the standard deviation $\boldsymbol{m}_i \pm 2\sqrt{\boldsymbol{C}_{ii}}$. Empty circles are predicted values at time t_n, filled circles represent updated values, orange dots are function and derivative observations. The first two columns display two *predict–evalute–update–predict* cycles. The last column shows the smoothed final solution (green, thick lines) and three samples from the predictive posterior (thin lines).

Algorithm 5: Rauch-Tung-Striebel smoothing and sampling

Require: filtering distribution: $\{\boldsymbol{m}_{t_N}, \boldsymbol{C}_{t_N},\ n = -1, \ldots, N\}$
Require: hyper-parameters: h, q, σ^2, K
$\boldsymbol{m}^S_{t_N} \leftarrow \boldsymbol{m}_{t_N}, \boldsymbol{C}^S_{t_N} \leftarrow \boldsymbol{C}_{t_N}$
for $k = 1, \ldots, K$ **do**
    $\boldsymbol{S}_{t_N,k} \leftarrow \boldsymbol{m}^S_{t_N} + \textsc{chol}(\boldsymbol{C}^S_{t_N})\ \textsc{randn}(q+1)$
end for
$\boldsymbol{F} \leftarrow \boldsymbol{U}_{q+1} + \boldsymbol{e}_q^\intercal \boldsymbol{0}, \boldsymbol{L} \leftarrow \boldsymbol{e}_q, \boldsymbol{A} \leftarrow \boldsymbol{A}(h), \boldsymbol{Q} \leftarrow \sigma^2 \overline{\boldsymbol{Q}}(h)$
for $n = N-1, N-2, \ldots, -1$ **do**
    $\boldsymbol{m}^-_{t_{n+1}} \leftarrow \boldsymbol{A}\boldsymbol{m}_{t_n}$
    $\boldsymbol{C}^-_{t_{n+1}} \leftarrow \boldsymbol{A}\boldsymbol{C}_{t_n}\boldsymbol{A}^\intercal + \boldsymbol{Q}$
    $\boldsymbol{G}_{t_n} \leftarrow \boldsymbol{C}_{t_n}\boldsymbol{A}^\intercal(\boldsymbol{C}^-_{t_{n+1}})^\intercal$
    $\boldsymbol{m}^S_{t_n} \leftarrow \boldsymbol{m}_{t_n} + \boldsymbol{G}_{t_n}(\boldsymbol{m}^S_{t_{n+1}} - \boldsymbol{m}^-_{t_{n+1}})$
    $\boldsymbol{C}^S_{t_n} \leftarrow \boldsymbol{C}_{t_n} + \boldsymbol{G}_{t_n}(\boldsymbol{C}^S_{t_{n+1}} - \boldsymbol{C}^-_{t_{n+1}})\boldsymbol{G}^\intercal_{t_n}$
    for $k = 1, \ldots, K$ **do**
        $\boldsymbol{s}_{t_n,k} \leftarrow \textsc{chol}(\boldsymbol{C}_{t_n} - \boldsymbol{G}_{t_n}\boldsymbol{C}^-_{t_{n+1}}\boldsymbol{G}^\intercal_{t_n})\ \textsc{randn}(q+1)$
        $\boldsymbol{S}_{t_n,k} \leftarrow \boldsymbol{m}_{t_n} + \boldsymbol{G}_{t_n}(S_{t_{n+1},k} - \boldsymbol{m}^-_{t_{n+1}}) + \boldsymbol{s}_{t_n,k}$
    end for
end for
return $\{\boldsymbol{m}^S_{t_N}, \boldsymbol{C}^S_{t_N},\ n = -1, \ldots, N\}$

Figure 8 shows the state of the algorithm after 2 steps have been taken. The solution looks discontinuous, because the information of later updates z_n has not been propagated to previous time points $t_m, m < n$. The last column of Figure 8 shows the *(predictive posterior) smoothing distribution* wherein all the information is globally available.

6.5 Interpretation

From the analytical viewpoint external to the algorithm itself, of course, one does not expect that the model assumption of a Gaussian likelihood, much less one with vanishing width, holds in reality. The point of the analysis in Section 6.7 is to demonstrate that this model and evaluation scheme yield a method satisfying sufficient conditions to prove that its point estimate converges at a nontrivial order for some choices of state spaces, while simultaneously keeping computational cost very low (that is very similar to that of classical multistep solvers). That is because the predictive posterior distributions $P(\boldsymbol{X}_{t_n} \mid z_{[n]})$ can be computed by the linear-time algorithm known as *Kalman filtering* [113, 181, 182]. The marginal predictive posterior distributions given all data $P(\boldsymbol{X}_\Delta \mid \Delta, z_{[N]})$ can be computed using the *Rauch-Tung-Striebel smoothing* equations [177, 181, 182]. Simultaneously, one can draw samples from the full joint posterior. These two operations increase the computational cost marginally: They require

additional computations comparable to those used for interpolation in classical solvers, but neither smoothing nor sampling requires additional evaluations of f. The computational complexity stays linear in number of data points collected. If the full joint posterior is also required for some reason, this is also possible to construct [201, 77]. As a second consequence, the computation becomes deterministic which enables unit testing of the resulting code.

As a side remark, we note some obvious restrictions of the combination of Gaussian (process) prior and likelihood used here: Since the posterior is always a Gaussian process, one cannot hope to capture bifurcations, higher-order correlations in the discretization errors or other higher order effects.

6.6 Connection to Runge-Kutta methods

So far, we have presented an algorithm without a convergence analysis. In this section, we will show how the MAP of an IWP(q) coincides with an explicit Runge-Kutta method of order q for $q \leq 4$. This can be used to create a one-step initialization scheme to a multistep method similar to Gear [68].

Let us first recall the general form of explicit Runge-Kutta formulas (24) from Section 3.2.1:

$$
\begin{aligned}
k_{i,n} &= f\left(t_n + hc_i, y_n + h\sum_{j=0}^{i-1} a_{ij}k_{j,n}\right), \qquad i = 0, \ldots, s-1 \\
y_{n+1} &= y_n + h\sum_{j=0}^{s-1} b_j k_{j,n}.
\end{aligned}
$$

Let us rewrite this equation a little bit:

$$
\begin{aligned}
t_{n,-1} &= t_n, \quad t_{n,i} = t_n + hc_i, i = 0, \ldots, s-1 \\
z_{-1,n} &= y_n, \quad z_{i,n} = f\left(t_{n,i}, \sum_{j=-1}^{i-1} ha_{ij}z_{j,n}\right), \qquad i = 0, \ldots, s-1 \\
y_{n+1} &= \sum_{j=-1}^{s-1} hb_j z_{j,n}.
\end{aligned}
$$

Compare this form with the MAP of an iteratively growing predictive posterior

$q=1$	$q=2$	$q=3$
$\begin{array}{c\|c} 0 & 0 \\ & 1 \end{array}$	$\begin{array}{c\|cc} 0 & 0 & \\ u & u & 0 \\ \hline & (1-\frac{1}{2u}) & \frac{1}{2u} \end{array}$	$\begin{array}{c\|ccc} 0 & 0 & & \\ u & u & 0 & \\ v & v-\frac{v(v-u)}{u(2-3u)} & \frac{v(v-u)}{u(2-3u)} & 0 \\ \hline & 1-\frac{2-3v}{6u(u-v)}-\frac{2-3u}{6v(v-u)} & \frac{2-3v}{6u(u-v)} & \frac{2-3u}{6v(v-u)} \end{array}$

Table 3: All explicit Runge-Kutta methods of order $q \leq 3$ and number of stages $s = q$ (see [82]).

$$\underline{\mu}_{\{y_n\}}(t_{n,0}) = \underbrace{\left[k(t_{n,0}, t_{n,-1})\right]\left[k(t_{n,-1}, t_{n,-1})\right]^{-1}}_{=\boldsymbol{\alpha}_{t_{n,0}}}\left[y_n\right], \tag{54a}$$

$$\underline{\mu}_{\{y_n, k_{0,n}\}}(t_{n,1}) = \underbrace{\begin{bmatrix} k(t_{n,1}, t_{n,-1}) & k^{\partial}(t_{n,1}, t_{n,0}) \end{bmatrix}\begin{bmatrix} k(t_{n,-1}, t_{n,-1}) & k^{\partial}(t_{n,-1}, t_{n,0}) \\ \partial k(t_{n,0}, t_{n,-1}) & \partial k^{\partial}(t_{n,0}, t_{n,0}) \end{bmatrix}^{-1}}_{=\boldsymbol{\alpha}_{t_{n,1}}}\begin{bmatrix} y_n \\ z_{0,n} \end{bmatrix}, \tag{54b}$$

and so forth for $z_{i,n}, i > 0$. The first two factors on the RHS of Eq. (54) define a weight-vector $\boldsymbol{\alpha}$ which is determined by the choice of evaluation knots $t_{n,i}$ and the kernel. Thus, if we want to achieve equivalence in the predictive posterior MAP, we have to match evaluation knots and calculate algebraically, whether the weights a_{ij}, b_j and the vectors $(\boldsymbol{\alpha}_{t_n})_j$ match.

Table 3 lists all explicit Runge-Kutta methods of orders $q = 1, 2, 3$ where $s = q$. Explicit methods necessarily have $c_0 = 0$, the remaining knots are free parameters. The table shows how the weights a_{ij}, b_j are algebraic expressions depending on the evaluation knots.

Consider now the once integrated Wiener process IWP(1) and let $\boldsymbol{m}^-_{t_{-1}} = \mathbf{0}$ and $\mathbf{0} \preccurlyeq \boldsymbol{C}^-_{t_{-1}} \in \mathbb{R}^{2\times 2}$ be an arbitrary covariance matrix. At t_0, we have $[y(t_0), y'(t_0)] = [y_0, f(t_0, y_0)]$ which are linear transformations of the model (47), i.e.,

$$\begin{aligned} y_0 &= \boldsymbol{e}_0 \boldsymbol{X}_{t_0} & z_0 &= f(t_0, y_0) = \boldsymbol{e}_1 \boldsymbol{X}_{t_0} \\ \Rightarrow \begin{bmatrix} y_0 \\ z_0 \end{bmatrix} &= \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{=\boldsymbol{H}_{01}} \begin{bmatrix} X_{t_0,0} \\ X_{t_0,1} \end{bmatrix} \end{aligned}$$

where the observation matrix $\boldsymbol{H}_{01}$ is the identity matrix $\boldsymbol{I}$.

These observations lead to the predictive posterior conditional

$$
\begin{aligned}
&P(\boldsymbol{X}_{t_0} \mid (\boldsymbol{X}_{t_0})_0 = y_0, (\boldsymbol{X}_{t_0})_1 = z_0) = \mathcal{N}(\boldsymbol{X}_{t_0}; \boldsymbol{m}, \boldsymbol{C}),\\
\boldsymbol{m} &= \boldsymbol{m}_{t_{-1}}^- + \underbrace{\boldsymbol{C}_{t_{-1}}^- \boldsymbol{H}_{01}^\mathsf{T} (\boldsymbol{H}_{01} \boldsymbol{C}_{t_{-1}}^- \boldsymbol{H}_{01}^\mathsf{T})^{-1}}_{=\boldsymbol{I}} (\boldsymbol{z} - \boldsymbol{m}_{t_{-1}}^-)\\
&= \boldsymbol{z},\\
\boldsymbol{C} &= \boldsymbol{C}_{t_{-1}}^- - \boldsymbol{C}_{t_{-1}}^- \boldsymbol{H}_{01}^\mathsf{T} (\boldsymbol{H}_{01} \boldsymbol{C}_{t_{-1}}^- \boldsymbol{H}_{01}^\mathsf{T})^{-1} (\boldsymbol{H}_{01} \boldsymbol{C}_{t_{-1}}^-)\\
&= \boldsymbol{0}
\end{aligned}
$$

according to Eq. (5). Given this filtering distribution, the prediction at $t_0 + h$, using (15) and the algebraic expressions for $\boldsymbol{A}(h)$ and $\boldsymbol{Q}(h)$ in Eq. (49), yields

$$
\begin{aligned}
P(\boldsymbol{X}_{t_0+h} \mid y_0, z_0) &= \mathcal{N}(\boldsymbol{X}_{t_1}; \boldsymbol{A}(h)\boldsymbol{z}, \boldsymbol{Q}(h))\\
&= \mathcal{N}(\boldsymbol{X}_{t_1}; [y_0 + hz_0, z_0]^\mathsf{T}, \boldsymbol{Q}(h)).
\end{aligned}
\tag{55}
$$

We see that $\boldsymbol{H}_0 \boldsymbol{m}_{t_0+h}^- = \boldsymbol{e}_0^\mathsf{T} \boldsymbol{m}_{t_1}^- = y_0 + hz_0$ is equivalent to the explicit Euler step which is the result from [185] written in state-space form.

However, continuing with the filtering formulation, we will now perform the next Kalman update

$$
\begin{aligned}
z_1 &= f(t_1, y_0 + hz_0)\\
\boldsymbol{K}_1 &= \boldsymbol{Q}(h) \boldsymbol{H}_1^\mathsf{T} (\boldsymbol{H}_1 \boldsymbol{Q}(h) \boldsymbol{H}_1^\mathsf{T})^{-1} = \begin{bmatrix} h/2 & 1 \end{bmatrix}^\mathsf{T}\\
\boldsymbol{m}_{t_1} &= (\boldsymbol{I} - \boldsymbol{K}_1 \boldsymbol{H}_1) \boldsymbol{m}_{t_1}^- + \boldsymbol{K}_1 z_1 = \begin{bmatrix} y_0 + h/2(z_0 + z_1) & z_1 \end{bmatrix}^\mathsf{T}
\end{aligned}
$$

to observe that $\boldsymbol{m}_{t_1}$ does *not* correspond to Euler's method. This implies that the argument above cannot be applied to the operation mode "probabilistic continuation" suggested in [185]. Analogous results hold for the arguments presented in the remainder of this section.

One might assume that this is a strong negative result, since we excluded the operation mode "naive chaining" (see [185]) to obtain globally defined models, but we do not recover Runge-Kutta methods for the alternative. However, we will show in Section 6.7 that we can retain an alternative equivalence still.

Nevertheless, in the remainder of this section, we will show that similar models and arguments can be used to construct Runge-Kutta predictions of orders $q = \{2, 3, 4\}$ as well, which will motivate an initialization procedure. Before we can do that, however, we will analyze the results and the methods which will simplify the derivations significantly.

First, we recall that the notation $t_{n,-1} = t_{n,0}$ was only introduced for convenience, thus, we have $\boldsymbol{\alpha}_{t_{n,0}} = 1$ in all models. When searching for Runge-Kutta methods, we find that Eq. (54a) is always of the right form for $n = 0$.

Secondly, the analysis of Runge-Kutta methods in Section 3.2.1 tells us that a Runge-Kutta method of order q matches the first q derivatives of the true solution at $y(t_n)$ which implies that appropriate models will require a state-space of dimensionality at least $q+1$. The matrix-vector algebra of the Kalman filter will soon become to cumbersome to work with without the help of computer algebra systems. However, we can use the equivalence result from Grigorievskiy, Lawrence, and Särkkä [77] (see Section 2.4) to get sub-expressions of $\boldsymbol{\alpha}_{t_n,j}$ which can be evaluated more easily.

To this end, let $\boldsymbol{H}_{01} = [\boldsymbol{e}_0 \ \boldsymbol{e}_1] \in \mathbb{R}^{2\times(q+1)}$ consider $\boldsymbol{\alpha}_{t_n,1}$ of (54b) written in the notation of Section 2.4:

$$\boldsymbol{\alpha}_{t_0,1} = \boldsymbol{e}_0^\intercal \boldsymbol{A}(hc_1)\boldsymbol{C}_{t_{-1}}^- \boldsymbol{A}(0)^\intercal \boldsymbol{H}_{01}^\intercal [\boldsymbol{H}_{01}\boldsymbol{A}(0)^{-\intercal}\boldsymbol{C}_{t_{-1}}^- \boldsymbol{A}(0)^{-1}\boldsymbol{H}_{01}^\intercal]^{-1}. \quad (56)$$

Recall that $\boldsymbol{A}(0) = \boldsymbol{I}$ from the basic properties of the matrix exponential. In the case of $q = 1$, $\boldsymbol{H}_{01} = \boldsymbol{I}$ and Eq. (56) simplifies to $\boldsymbol{\alpha}_{t_0,1} = \boldsymbol{e}_0^\intercal \boldsymbol{A}(hc_1)$ which is, again, the result from above. However, in the case of $q \geq 2$, we find that the intermediate matrix products in Eq. (56) do not cancel.

So let us look at Eq. (56) for the case of the IWP(2). The equation looks identical, but the involved matrices have an additional row and column. We split the multiplication into three parts:

$$\boldsymbol{\alpha}_{t_0,1} = \underbrace{\boldsymbol{e}_0^\intercal \boldsymbol{A}(hc_1)}_{\boldsymbol{M}_1} \underbrace{\boldsymbol{C}_{t_{-1}}^- \boldsymbol{H}_{01}^\intercal}_{\boldsymbol{M}_2} \underbrace{[\boldsymbol{H}_{01}\boldsymbol{C}_{t_{-1}}^- \boldsymbol{H}_{01}^\intercal]^{-1}}_{\boldsymbol{M}_3}. \quad (57)$$

From Eq. (49) we find $\boldsymbol{M}_1 = [1 \ hc_1 \ 2^{-1}(hc_1)^2]$. Matrix $\boldsymbol{M}_3$ is the inverse of the upper left part of matrix $\boldsymbol{C}_{t_{-1}}^-$ whereas Expression $\boldsymbol{M}_2$ yields the left side of said matrix. Lets put $\boldsymbol{C}_{t_{-1}}^- = \boldsymbol{Q}(\tau)$ to get concrete values

$$\boldsymbol{M}_2\boldsymbol{M}_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \frac{-20}{3\tau^2} & \frac{4}{\tau} \end{bmatrix} \quad (58)$$

and we see that this yields the correct result only in the case of $\lim \tau \to \infty$. Contrasting Equation (58) to Equation (12) from Schober, Duvenaud, and Hennig [185], we can appreciate the easier algebra of the state-space formulation.

We finish the proof of Theorem 2 from [185] by inserting the values from Table 3 into

$$\begin{aligned} \boldsymbol{M}_1 &= \boldsymbol{e}_0^\intercal [\boldsymbol{A}(h) \ \boldsymbol{A}(h(1-u))], \\ \boldsymbol{M}_2 &= \boldsymbol{Q}(\boldsymbol{t})\boldsymbol{A}(\boldsymbol{t})^\intercal \boldsymbol{H}_{01,1}^\intercal, \\ \boldsymbol{M}_3 &= [\boldsymbol{H}_{01,1}\boldsymbol{A}(\boldsymbol{t})\boldsymbol{Q}(\boldsymbol{t})\boldsymbol{A}(\boldsymbol{t})^\intercal \boldsymbol{H}_{01,1}^\intercal]^{-1}, \\ \boldsymbol{\alpha}_{t_0,2} &= \boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{M}_3, \end{aligned} \quad (59)$$

where $\boldsymbol{t} = (t_0, t_0 + hu)^\intercal$ and $\boldsymbol{H}_{01,1} = \operatorname{diag}(\boldsymbol{H}_{01}, \boldsymbol{H}_1)$ is the block-diagonal matrix of size $(2+1) \times 2(q+1)$ with blocks $\boldsymbol{H}_{01}$ and $\boldsymbol{H}_1$. Analogous to Eq. (57), we find

$$
\begin{aligned}
\boldsymbol{M}_1 &= \begin{bmatrix} 1 & h & \frac{h^2}{2} & 1 & h\,(1-u) & \frac{h^2(1-u)^2}{2} \end{bmatrix} \\
\boldsymbol{M}_2 &= \sigma^2 \begin{bmatrix} \frac{\tau^5}{20} & \frac{\tau^4}{8} & \frac{\alpha h}{6}\tau^3 + \frac{\tau^4}{8} \\ \frac{\tau^4}{8} & \frac{\tau^3}{3} & \frac{uh}{2}\tau^2 + \frac{\tau^3}{3} \\ \frac{\tau^3}{6} & \frac{\tau^2}{2} & uh\tau + \frac{\tau^2}{2} \\ 0 & 0 & \frac{u^4h^4}{8} \\ 0 & 0 & \frac{u^3h^3}{3} \\ 0 & 0 & \frac{u^2h^2}{2} \end{bmatrix} \\
\boldsymbol{M}_3 &= \sigma^{-2} \begin{bmatrix} \frac{960uh+720\tau}{\tau^5(3uh+\tau)} & -\frac{360u^2h^2+360uh\tau+60\tau^2}{uh\tau^4(3uh+\tau)} & \frac{60}{uh\tau^2(3uh+\tau)} \\ -\frac{360u^2h^2+360uh\tau+60\tau^2}{uh\tau^4(3uh+\tau)} & \frac{48u^2h^2(3uh+\tau)+9\tau(4uh+\tau)^2}{u^2h^2\tau^3(3uh+\tau)} & -\frac{36uh+9\tau}{u^2h^2\tau(3uh+\tau)} \\ \frac{60}{uh\tau^2(3uh+\tau)} & \frac{36uh+9\tau}{u^2h^2\tau(3uh+\tau)} & \frac{9}{u^2h^2(3uh+\tau)} \end{bmatrix}.
\end{aligned}
$$

Multiplying and taking the limit as $\tau \to \infty$ yields a second-order Runge-Kutta as the mean of the predictive distribution.

We now turn our attention to the (posterior) covariance. Letting $\lim \tau \to \infty$ leads to an improper prior covariance model, similar to Wahba [217] and Koopman [119]. For instance, calculating the posterior $\mathcal{GP}(y; \underline{\mu}, \underline{k} \mid y_0, z_0)$, we find for the predictive posterior covariance

$$
\begin{aligned}
\underline{k}(t,s) &= k(t,s) - k(t,\boldsymbol{t})K(\boldsymbol{t},\boldsymbol{t})^{-1}k(\boldsymbol{t},s) \\
&= \boldsymbol{A}(t,\boldsymbol{t}) \underbrace{\left(\boldsymbol{Q}(\boldsymbol{t}) - \boldsymbol{Q}(\boldsymbol{t})\boldsymbol{A}(\boldsymbol{t})^\intercal \boldsymbol{H}_{01,1}^\intercal \boldsymbol{G}^{-1} \boldsymbol{H}_{01,1} \boldsymbol{A}(\boldsymbol{t})\boldsymbol{Q}(\boldsymbol{t})\right)}_{=\underline{\boldsymbol{Q}}(\boldsymbol{t})} \boldsymbol{A}(s,\boldsymbol{t})^\intercal,
\end{aligned}
$$

where $\boldsymbol{G} = \boldsymbol{H}_{01,1}\boldsymbol{A}(\boldsymbol{t})\boldsymbol{Q}(\boldsymbol{t})\boldsymbol{A}(\boldsymbol{t})^\intercal\boldsymbol{H}_{01,1}^\intercal$. Taking the limit as $\tau \to \infty$ yields a finite value after three observations. It is not immediately obvious that this should be the case. An intuitive explanation is that the IWP(2) is 3-dimensional and each observation provides a rank-1 update to the state space. Similar to the case in (59), it suffices to check $\underline{\boldsymbol{Q}}(\boldsymbol{t})$. This involves algebraically calculating the inverse of $\boldsymbol{G}$, a $(2 \cdot 3) \times (2 \cdot 3)$ matrix of polynomials which yields a matrix of polynomial fractions. Doing this manually is a cumbersome and error-prone process, so we omit the details. An implementation in

SymPy can be found in Appendix A. Here, we only present the final result

$$\underline{\boldsymbol{Q}}(\boldsymbol{t}) = \sigma^2 \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{hu}{3} & -\frac{(hu)^3}{8} & -\frac{(hu)^2}{3} & -\frac{hu}{2} \\ 0 & 0 & -\frac{(hu)^3}{8} & \frac{(hu)^5}{20} & \frac{(hu)^4}{8} & \frac{(hu)^3}{6} \\ 0 & 0 & -\frac{(hu)^2}{3} & \frac{(hu)^4}{8} & \frac{(hu)^3}{3} & \frac{(hu)^2}{2} \\ 0 & 0 & -\frac{hu}{2} & \frac{(hu)^3}{6} & \frac{(hu)^2}{2} & hu \end{bmatrix}.$$

In this case of $q = 2$, the philosophical reservations for chaining such single-step methods together to build a Runge-Kutta solver are even more severe. Equivalence is achieved for $\boldsymbol{C}_{t_n}^- = \lim_{\tau\to\infty} \boldsymbol{Q}(\tau)$ only which can be thought as a "diffuse state" (cf. Wahba [217]) or "infinitely uninformative prior". Matching Runge-Kutta over two consecutive steps would require to brush over the predictive posterior of the first step and reset the the initial believe to the diffuse state. Thus, we would ignore information that we have collected, although our task is to infer the solution with as little effort as possible.

However, the connection to Runge–Kutta can still be harnessed to initialize the algorithm. Just as we did above, we can compute the algebraic expressions for the entries $(\boldsymbol{m}_{t_1}^-)_i, (\boldsymbol{C}_{t_1}^-)_{ij}, i, j = 0, \ldots, q$ of the predictive distribution $P(\boldsymbol{X}_{t_1} \mid \{y_0, z_1, \ldots, z_{q-1}\})$ in state-space formulation with the formal initialization $\boldsymbol{m}_{t_{-1}}^- = \boldsymbol{0}$ and $\boldsymbol{C}_{t_{-1}}^- = \boldsymbol{Q}(\tau)$. From the preceeding analysis, we know that this will yield

$$(\boldsymbol{m}_{t_1}^-)_0 = y_0 + h \sum_{j=0}^{s-1} b_j z_j$$

which coincides with the Runge–Kutta prediction in the limit as $\tau \to \infty$ and will produce finite expressions for the entries of $\boldsymbol{C}_{t_1}^-$. Thus, it is valid to compute the z_j from a regular Runge–Kutta step, and then insert these into the expressions of $\lim_{\tau\to\infty} \boldsymbol{m}_{t_1}^-, \boldsymbol{C}_{t_1}^-$. Algorithm 6 presents a pseudo-code where $\boldsymbol{m}_{t_{-1}}^*$ denotes the mean-vector with symbolic expressions for $z_{0:q-1}$. This approach is structurally similar to an algorithm given by Gear [68] for the case of classical Runge–Kutta and Nordsieck methods.

Algorithm 6: Runge-Kutta initialization

function INIT($f, \mathbb{T}, y_0$, model)
  $\boldsymbol{z}_{0:(q-1)} \leftarrow$ RUNGE-KUTTA(f, t_0, y_0, q)
  $\boldsymbol{m}_{t_1}^-, \boldsymbol{C}_{t_1}^- \leftarrow$ SYMPY($\boldsymbol{m}_{t_{-1}}^* = \boldsymbol{0}, \boldsymbol{C}_{t_{-1}}^- = \boldsymbol{Q}(\tau), q$)
  $\boldsymbol{m}_{t_1}^- \leftarrow$ INSERT($\boldsymbol{m}_{t_1}^*, \boldsymbol{z}_{0:q-1}$)
end function

Note that the expression of $\boldsymbol{C}_{t_1}^-$ is independent of the observations and can be computed directly. While the expressions are *not* inde-

pedent of the evaluation knots hc_i, in practice, we have decided to always use the same evaluation knots since in the case of $q = 3, 4$, not all combinations are feasible (see below).

Analoguous calculations for the case of $q = s = 3$ and the IWP(3) can be made to achieve equivalence in the mean for the corresponding Runge-Kutta methods, except for the case when $v \neq \frac{2}{3}$. Schober et al. [185] noted that this could be remedied by considering a special element in the reproducing kernel Hilbert space of the thrice-integrated Wiener process kernel, and it was conjectured that this special element bears a meaningful interpretation. However, with the more advanced understanding of diffuse priors, this view has to be updated.

Underlying the construction of higher-order Runge-Kutta methods are Taylor expansions of increasing order, i.e., local polynomial fits. To achieve ever higher order, the fitted polynomials need to fulfill an ever bigger set of constraints. These conditions are satisfied by comparing the derivative of the true solution and the numerical approximation viewed as a function of h at the end of the current step [82, §2]. Notably, these are the *only* constraints the coefficients need to satisfy. Conversely, the IWP(q) model assumes noise-free derivative observations at the intermediate knots $t_0 + hc_i$. This can be understood as additional constraints on the coefficients of the numerical solution to satisfy certain derivative values in the middle of the integration step. Thus, it is natural that the set of high order filtering methods is smaller than the set of high order RK methods.

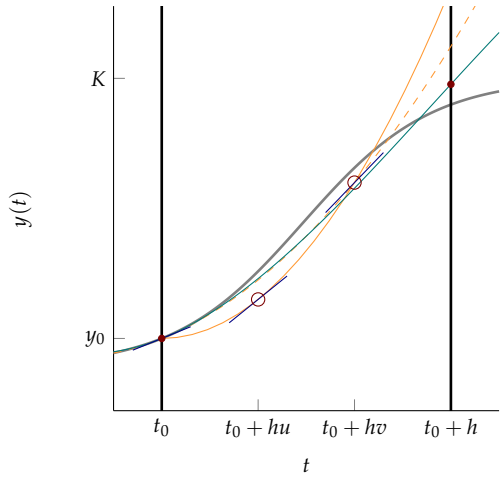

Figure 9: Runge-Kutta mismatch in the case $q = 3$. For a description see text.

The situation is also depicted in Figure 9 for the IWP(3). After obtaining the second derivative z_1, the Runge–Kutta extrapolation $y_{RK}(t)$ is formally depicted by the solid yellow line. Notice, how this curve does not satisfy $y'_{RK}(t_0) = z_0$, i.e., does not satisfy the first derivative observation *known to be correct*. The dashed yellow line represents the predictive posterior mean of the IWP(3). At $hv, v = 2/3$

the two predictions intersect yielding a method probabilistic Runge–Kutta method. In the cases $v \neq 2/3$, this constraint is not enforced by the algebra of the Runge–Kutta analysis.

In [187], a match was reported between a four step Runge–Kutta formula and the IWP(4). This match is obtained for the evaluation knots $t_0 + c_i h$ with the vector $\boldsymbol{c} = (0, 1/3, 1/2, 1)^\mathsf{T}$. Exact expressions are listed in Appendix B.

Finally, it should be pointed out that this is only one feasible initialization. In cases where automatic differentiation (see Griewank and Walther [76]) is available, this can be used to initialize the Nordsieck vector up to numerical precision and set $\boldsymbol{C}_{t_{-1}}$ to $\boldsymbol{0}$. Nordsieck [156] originally proposed to start with an initial vector $\boldsymbol{m}_{t_{-1}} = 0$, followed by $q+1$ steps with positive and $q+1$ with negative direction (that is, integrating backwards to the start). One interpretation is that the method uses $\boldsymbol{m}_{t_{-1}} = \mathbb{E}[X_{t_{-1}} \,|\, \tilde{z}_{-1}, \ldots, \tilde{z}_q]$, with tentative $\tilde{z}_i$ computed out of this process.

6.7 Connection to multistep methods

We will now show how the Kalman filter (15), (16) can be rewritten such that the mean prediction takes the form of (35). This enables to analyze the proposed algorithm in light of classical Nordsieck method results, but can also guide the further development of the probabilistic approach with the experience of existing software.

We rescale the state space and SDE of the IWP(q) by scaling matrix $\boldsymbol{B}$ to define an equivalent notation:

$$\begin{aligned}\tilde{\boldsymbol{X}}_t &= \left(Y_t, \quad hY_t', \quad \tfrac{h^2}{2!}Y_t'', \quad \ldots \quad \tfrac{h^q}{q!}Y_t^{(q)}\right)^\mathsf{T} \\ &= \underbrace{\operatorname{diag}\left(1, \quad h, \quad \tfrac{h^2}{2!}, \quad \ldots \quad \tfrac{h^q}{q!}\right)}_{=\boldsymbol{B}} \boldsymbol{X}_t.\end{aligned}$$

This state vector is the *Nordsieck vector*. The advantage of this notation is that (48) simplifies to

$$\mathrm{d}\tilde{\boldsymbol{X}}_t = \boldsymbol{B}\boldsymbol{U}_{q+1}\boldsymbol{B}^{-1}\tilde{\boldsymbol{X}}_t \,\mathrm{d}t + \boldsymbol{B}\boldsymbol{e}_q \,\mathrm{d}W_t, \tag{60}$$

where $\tilde{\boldsymbol{A}}(h) = \boldsymbol{P}$, the Pascal triangle matrix, and

$$\begin{aligned}(\tilde{\boldsymbol{Q}}(h))_{ij} &= (\boldsymbol{B}\boldsymbol{Q}(h)\boldsymbol{B}^\mathsf{T})_{ij} \\ &= \frac{h^i}{i!}\sigma^2 \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!}\frac{h^j}{j!} \\ &= \frac{\sigma^2 h^{2q+1}}{(2q+1-i-j)(q-i)!(q-j)!i!j!},\end{aligned}$$

which can be seen by inserting (60) into (13) and simplifying. Furthermore, the observation matrices become $\tilde{\boldsymbol{H}}_0 = \boldsymbol{H}_0\boldsymbol{B}^{-1} = \boldsymbol{e}_0$ and $\tilde{\boldsymbol{H}}_1 = \boldsymbol{H}_1\boldsymbol{B}^{-1} = h^{-1}\boldsymbol{e}_1$. Rewriting the filtering equations, we arrive at

$$\begin{aligned}\boldsymbol{C}_{t_n}^- &= \boldsymbol{P}\boldsymbol{C}_{t_{n-1}}\boldsymbol{P}^\intercal + \tilde{\boldsymbol{Q}}(h),\\ \boldsymbol{K}_n &= \boldsymbol{C}_{t_n}^-\tilde{\boldsymbol{H}}_1^\intercal[\tilde{\boldsymbol{H}}_1\boldsymbol{C}_{t_n}^-\tilde{\boldsymbol{H}}_1^\intercal]^{-1}\end{aligned} \tag{61}$$

and

$$\boldsymbol{m}_{t_n} = (\boldsymbol{I} - \boldsymbol{K}_n\tilde{\boldsymbol{H}}_1)\boldsymbol{P}\,\boldsymbol{m}_{t_{n-1}} + \boldsymbol{K}_n z_n^{(1)}, \tag{62}$$

$$\boldsymbol{C}_{t_n} = (\boldsymbol{I} - \boldsymbol{K}_n\tilde{\boldsymbol{H}}_1)\boldsymbol{P}(\boldsymbol{C}_{t_{n-1}}\boldsymbol{P}^\intercal + \tilde{\boldsymbol{\Phi}}_{12}(h)^\intercal). \tag{63}$$

Choosing a prior covariance matrix with entries $(\boldsymbol{C}_{t_{-1}}^-)_{ij} = \sigma^2 h^{2q+1} c_{ij}$, for some $c_{ij} \in \mathbb{R}$ such that $\boldsymbol{C}_{t_{-1}}^-$ is a valid covariance matrix, it can be shown by induction that all entries of $\boldsymbol{C}_{t_n}$ for all n have this structural form. As a by-product, $\boldsymbol{K}_n = h\,(k_{n,0}, 1, k_{n,2}, \ldots, k_{n,q})^\intercal$ for some $k_{n,i} \in \mathbb{R}$ which follows from (61).

Given these invariants, Equation (62) has the structure of a multistep method written in Nordsieck form (35). The only difference is the changing weight vector $\boldsymbol{K}_n$ (62) as compared to the constant weights in (35). Multistep methods with varying weights have been studied in the literature [47, 23]. These works are often in the context of variable step sizes $h_n \neq h$. But variable-coefficient methods have also been studied for other purposes, for example, cyclic methods [4]. These works have in common that the weights are free variables which are not limited through the choice of the model class. As a consequence, determining optimal weights can be algebraically difficult [82, §III.5].

Here, variable step sizes are easily obtained by working with the representation (48) instead of (60) and computing (14) according to h_n. In contrast to classical methods, the weights $\boldsymbol{K}_n$ cannot be chosen freely, but are determined through the choice of model (48), and the evolution of the underlying uncertainty $\boldsymbol{C}_{t_n}$. While Kersting and Hennig [116] provide some preliminary empirical evidence that these adaptive weights $\boldsymbol{K}_n$ might actually improve the estimate, more rigorous analysis is required for theoretical guarantees.

The model is agnostic to the generative model used to create the observation data z_n: We could also choose to solve the implicit equation (31) corresponding to the weight vector $\boldsymbol{K}$. This should be considered a criticsm of the proposed model, since the associated numerical uncertainty $\boldsymbol{C}_{t_n}$ is ignorant to the evaluation strategy, although we know from classical analysis that the P(EC)$^\infty$ yields a more accurate solution than the P(EC)^1 method.

We will now study the long-term behavior of the PFOS. In particular, we will ask what the long-term behavior for the sequence of

Kalman gains $(\boldsymbol{K}_n)_{n=0,\ldots}$ is and how this will influence the solution quality. It can be shown that its properties are linked to properties of the *discrete algebraic Riccati equation*, of which the theory has largely been developed [126]. Denote by $\gamma : \mathbb{R}^{(q+1)\times(q+1)} \to \mathbb{R}^{(q+1)\times(q+1)}$ the function that maps the covariance matrix $\boldsymbol{C}_{t_{n-1}}$ of the previous knot t_{n-1} to the covariance matrix $\boldsymbol{C}_{t_n}$ at the current knot t_n (Equation (63)). If there exists a (unique) fixed point $\boldsymbol{C}^*$ of γ, it is called the *steady state* of the model (48). Associated with a fixed point $\boldsymbol{C}^*$ is also a constant Kalman gain $\boldsymbol{K}^*$ that is obtained at the (numerical) convergence of $\boldsymbol{C}^*$.

We will now show that there is a subset of model (48) that converges to a steady state. This subsystem completely determines a constant Kalman gain $\boldsymbol{K}^*$ at least in the case of the IWP(1) and IWP(2). Thus, like in the equivalence result for the Runge–Kutta methods in Schober et al. [185], the result of the PFOS is equivalent (in the sense of numerically identical) after an initialization period to a corresponding classical Nordsieck method defined by the weight vector $\boldsymbol{K}^*$ and we can apply all the known theory of multistep methods to the mean of the PFOS.

Proposition 1. *The PFOS arising from the once integrated Wiener process IWP(1) is equivalent in its predictive posterior mean to the P(EC)^1 implementation of the trapezoidal rule.*

Proof. The trapezoidal rule, written as implicit linear multistep method of form (30), is given as

$$y_{t_n} = y_{t_n} + \frac{h}{2}(f_{t_{n-1}} + f_{t_n}).$$

We will show that $(\boldsymbol{m}_{t_n})_0 = (\boldsymbol{m}_{t_{n-1}})_0 + h/2[(\boldsymbol{m}_{t_{n-1}})_1 + (\boldsymbol{m}_{t_n})_1]$ for all n by induction. Let $\boldsymbol{m}^-_{t_{-1}} = \boldsymbol{0}$ and $\boldsymbol{C}^-_{t_{-1}} \in \mathbb{R}^{2\times 2}$ be an arbitrary covariance matrix. Applying the first three lines of Algorithm 4 algebraically, the predicted values are

$$\boldsymbol{m}_{t_{-1}} = \begin{pmatrix} y_0 \\ m^-_{t_0,1} \end{pmatrix}, \qquad \boldsymbol{C}_{t_{-1}} = \begin{pmatrix} 0 & 0 \\ 0 & c^-_{t_0,11} \end{pmatrix},$$

for some $m^-_{t_0,1}, c^-_{t_0,11}$. Continuing in this fashion yields $z_0 = f(t_0, y_0)$ and $\boldsymbol{m}_{t_0} = (y_0, z_0)^\intercal, \boldsymbol{C}_{t_0} = \boldsymbol{0}$. Using (15a) and (15b) to compute the predictions at t_1, we arrive at

$$\boldsymbol{m}^-_{t_1} = \begin{pmatrix} y_0 + hz_0 \\ z_0 \end{pmatrix}, \qquad \boldsymbol{C}^-_{t_1} = \boldsymbol{Q}(h),$$

and we see that $\boldsymbol{H}_0 \boldsymbol{m}^-_{t_0+h} = y_0 + hz_0 = (\boldsymbol{P}(y_0, hz_0)^\intercal)_0$. Completing the Kalman step by applying Equations (16a)-(16d) yields

$$\boldsymbol{m}_{t_1} = \begin{pmatrix} y_0 + \frac{h}{2}[z_0 + z_1] \\ z_1 \end{pmatrix}, \qquad \boldsymbol{C}_{t_1} = \sigma^2 \begin{pmatrix} \frac{h^3}{12} & 0 \\ 0 & 0 \end{pmatrix}, \tag{64}$$

where $z_1 = f(t_1, y_0 + hz_0)$. Comparing with (37), we see that z_1 is of the desired form, which completes the start of the induction. Finally, we observe that the second column of $\boldsymbol{C}_{t_1} = \boldsymbol{0} = \boldsymbol{C}_{t_0}$, i.e., this will be invariant and, as a consequence, the second column of $\boldsymbol{C}^-_{t_n}$ is simply the second column of $\boldsymbol{Q}(h)$, and the induction is completed. □

The following Theorem 3 for the IWP(2) requires a bit more algebra, but is based on the same principle.

Theorem 3. *The predictive posterior mean of the IWP(2) with fixed step size h is a third order Nordsieck method, when the predictive distribution has reached the steady state.*

Proof. The proof proceeds in two steps. First, we show that the update equations induce a specific form for the covariance matrix $\boldsymbol{C}_{t_n}$. Then, we will analyze individual entries.

We proof by induction that $\boldsymbol{C}_{t_n}$ is of the form

$$\boldsymbol{C}_{t_n} = \sigma^2 h^5 \begin{pmatrix} c_{t_n,00} & 0 & c_{t_n,02} \\ 0 & 0 & 0 \\ c_{t_n,02} & 0 & c_{t_n,22} \end{pmatrix}, \tag{65}$$

with coefficients $c_{t_n,ij}$ such that $\boldsymbol{C}_{t_n}$ is a valid covariance matrix. The base case is achieved after the first derivative observation $f(t_0, y_0)$ at t_0 which can be checked by algebraic computation. The inductive step can be verified by assuming the form (65) for t_{n-1} and compute one step ahead using Equations (61) and (63) similar to the base case. Next, for the individual entries we find

$$\begin{aligned} c_{t_{n+1},00} &= \sigma^2 h^5 \tfrac{3840c_{00}c_{22} + 320c_{00} - 3840c_{02}^2 + 110c_{02} + 32c_{22} + 1}{320(12c_{22}+1)} \\ c_{t_{n+1},02} &= \sigma^2 h^5 \tfrac{-(48c_{02} + 24c_{22} + 1)}{96(12c_{22}+1)} = (\boldsymbol{C}_{t_{n+1}})_{20} \\ c_{t_{n+1},22} &= \sigma^2 h^5 \tfrac{16c_{22}+1}{16(12c_{22}+1)} \\ c_{t_{n+1},ij} &= 0, \quad i,j = 0,1,2, i \vee j = 1, \end{aligned} \tag{66}$$

where we put $c_{ij} = c_{t_n,ij}$ on the respective right-hand sides of Equation (66) for brevity.

We will now consider the behavior of the coefficients c_{ij}. Consider the dynamical system $\bar{\gamma}_{22}(c) = (16c+1)[16(12c+1)]^{-1}$ which maps the coefficient of the last entry in $\boldsymbol{C}_{t_n}$ to the next. The range and image of $\bar{\gamma}_{22}$ are the nonnegative reals, since variances cannot

be negative. On this domain, $\bar{\gamma}_{22}$ has a continuous and bounded derivative $|\bar{\gamma}'_{22}| \leq \frac{1}{4}$. In particular, $\bar{\gamma}_{22}$ is a contraction with Lipschitz constant $\frac{1}{4}$. Thus, the entries converge to the fixpoint $c^*_{22} = \frac{\sqrt{3}}{24}$ (which can be found with some simple algebra). Similarly, one can either insert c^*_{22} into the respective form of $\bar{\gamma}_{02}$ or one considers the two-dimensional mapping of both entries. In both cases, a similar argument guarantees the convergence to a fixpoint, which is found to be $c^*_{02} = -\frac{\sqrt{3}}{144}$. Inserting these into Equation (61), we find that $\boldsymbol{K}_n = \boldsymbol{K}^* = (\frac{3+\sqrt{3}}{12}, 1, \frac{3-\sqrt{3}}{2})^\intercal$ is the static probabilistic Nordsieck method of the IWP(2) filter. Inserting these weights into [198, Theorem 4.2] yields the result. □

Although Theorem 3 is only valid when the system has reached its steady state, we find that the convergence (visualized in Figure 10) is rapid in practice. In the extreme case of $q = 1$ (not shown) it is instantaneous, in fact, and Proposition 1 is valid from the second step onwards. This limitation could also be circumvented in practice by initializing $\boldsymbol{C}_{t_{-1}}$ at steady state coefficients, but this possibility is not required to achieve high-order convergence on the benchmark problems we considered.

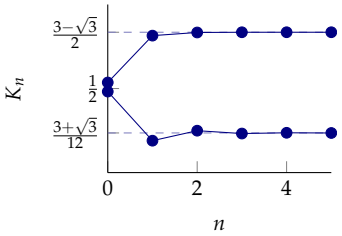

Figure 10: The weights $(K_n)_0$ and $(K_n)_2$ for $n = 0, \ldots, 5$.

Figure 10 shows the situation for a constant value of the diffusion amplitude σ^2. In Section 7.2, we will discuss error estimation and step size adaptation. This process leads to a continuous adaptation of this variable, which in turn means that the convergence shown in the figure continues throughout the run of the algorithm. So the practical algorithm presented here and empirically evaluated in Section 7.3 is not formally identical to Nordsieck methods, merely conceptually closely related.

Inspecting the weights of the IWP(2), we find that this method has not been considered previously in the literature, and, in particular, cannot be related to any of the classical formulas, such as Adams-Moulton or backward differentiation formulae. This is not surprising, since the result of this method has been constructed to be globally twice continuously differentiable, whereas there is no such guarantee for the solution provided by the typical methods. In fact, the probabilistic Nordsieck method is much closer related to spline-based multistep methods such as [141, 140, 29, 5] since Gaussian process regression models have a one-to-one correspondence to spline smoothing in a reproducing kernel Hilbert space of appropriate choice [117, 216]. This also justifies the application of a full-support distribution, even though it is known that the solution will remain in a compact set. In the former case, the interpretation is one of average-case error, whereas in the latter, the bound corresponds to the worst-case error [166].

Additionally, the forms of C_{t_n} found in Equations (64) and (66) show that the standard deviation $\mathrm{std}[Y_{t_n}] = (\boldsymbol{C}_{t_n})_{00}^{1/2}$ can be meaningfully, if weakly, interpreted as an approximation to the *expected error* $|y_{t_n} - y(t_n)|$ of the numerical solution in the following local, asymptotic sense: From our analysis of the IWP(q), $q \in \{1, 2\}$, we have $|y_{t_n} - y(t_n)| \leq Ch^{q+1}$, whereas $(\boldsymbol{C}_{t_n})_{00}^{1/2} \in \mathcal{O}(\sigma h^{q+1/2})$. Estimating the intensity σ of the stochastic process amounts to estimating the unknown constant C.

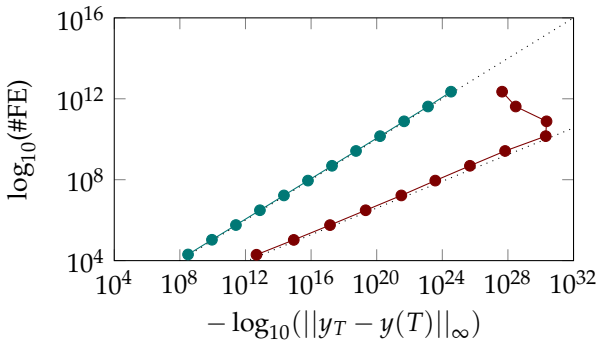

Figure 11: Work-precision diagram for the IWP(1) (green) and IWP(2) (red) applied to the logistic growth problem from Section 6.4. Plotted are the logarithms of the number of function evaluations (#FE) against the logarithmic error at the end of the integration domain. Dotted lines mark ideal convergence rates of orders two and three, respectively.

Figure 11 displays the work-precision diagram for the IWP(1) and IWP(2) applied to the examplary problem of Section 6.4. The plot shows a good agreement between the theoretical rate and the observed rate of convergence.

We conclude this section by considering some implications of the probabilistic interpretation in contrast to other classical multistep methods.

Keeping all hyper-parameters (order q, prior diffusion intensity σ^2, and step size h) fixed, the gain $\boldsymbol{K}_n$ is completely determined, and, as a consequence, solving Equation (36) up to numerical precision would also have been an admissible action rule for the generation of z_n. This can be interpreted as observing the true value of the model (48) at t_n which gives another justification for using $R_n^2 = R^2 = 0$. Since the P(EC)$^\infty$ and the P(EC)$^\mathrm{M}$ have the same order for all M [53], this argument can be extended to the case of the PEC^1 implementation which gives the most natural connection to the Kalman filter.

In fact, a P(EC)^M, $M > 1$, implementation would collect and put aside the values $z_n^{(1)}, \ldots, z_n^{(M-1)}$, which seems unintuitive from an inference point of view, where it is natural to assume that more data should yield a better approximation. A natural question would be whether this is a case of diminishing returns of approximation quality for computational power, but this is beyond the scope of this thesis.

One current limitation of the PFOS is its fixed integration order q over the whole integration domain $\mathbb{T}$. The reason for this limitation is that it is conceptually not straight forward to connect spline

models of different orders at knots t_n. However, the ability to adapt the integration order during runtime has been key in improving the efficiency of modern solvers [28]. Furthermore, the method corresponding to the IWP(2) model has a rather small region of stability which is depicted in Figure 12, specially in comparison with backward differentiation formulas (BDFs) [53]. This makes the method impractical for stiff equations.

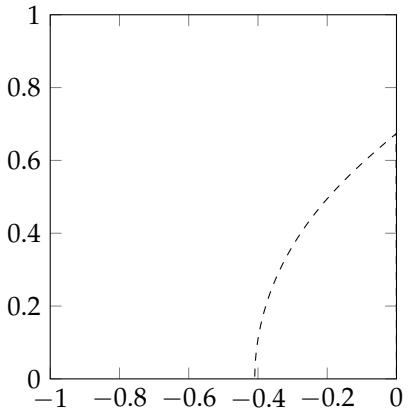

Figure 12: Partial stability domain of the probabilistic filtering ODE solver using the IWP(2) in the negative real, positive imaginary quadrant. The method converges for step sizes h on linear problems $y' = \lambda y$, if $h\lambda = z \in \mathbb{C}$ lies in the region of stability in the lower right corner. See [53] for details.

It is natural to ask what happens in the case of the IWP(q), $q > 2$. Using techniques from the analysis of Kalman filters, one can show that these models also contain a stable subsystem and that the weights $\boldsymbol{K}_n$ will converge to a fixed point $\boldsymbol{K}^*$, even for nonzero, but constant, R^2. However, it remains unclear whether they will be practical. In particular, these methods might even be unstable for most spline models [141]. We have tested the IWP(q), $q \in \{1, \ldots, 4\}$, empirically on the Hull et al. benchmark (see Section 7.3) and have observed that these converge in practice on these non-stiff problems.

6.8 Connection to other related work

Since the first publication of Schober, Duvenaud, and Hennig [185], various other authors have also looked into the construction of probabilistic numerical methods. These methods share a connection to Gaussian (non-) parametric regression or Gaussian processes. The differences lie in their use of randomization, their motivation and their generality.

The connection to Chkrebtii, Campbell, Calderhead, and Girolami [39] has already been discussed in Section 6.3. Similarly, the work of Kersting and Hennig [116] can be considered a generalization of Schober, Särkkä, and Hennig [187] as has been pointed out in Section 6.3.

Similar in spirit is the work by Cockayne, Oates, Sullivan, and Girolami [40]. In this work, the main contribution is a model and al-

gorithm for the solution of linear partial differential equations, which can also be applied to linear IVPs and BVPs. In the case of linear problems, the differential operator can be incorporated into the observation without approximations, yielding a problem that can be solved globally in one step. To observe $y' = \lambda y \Leftrightarrow y' - \lambda y = 0$, they propose to consider the observation matrix $\boldsymbol{H} = [-\lambda, 1, 0, \ldots, 0]$ and the noise-free observation $z_n \equiv 0$ and similar for non-autonomous problems. Recent work of Raissi, Perdikaris, and Karniadakis [175] generalizes this approach to integro-differential equations and operator equations of fractional order. For IVPs, these algorithms are of limited applicability, because the linear problem is only considered as a model problem.

Other authors who are mainly motivated by accurate uncertainty quantification [208, 131] have proposed novel randomized algorithms. Conrad, Girolami, Särkkä, Stuart, and Zygalakis [42] and Teymur, Zygalakis, and Calderhead [210] propose to randomize Runge-Kutta and multistep methods, respectively.

Conrad, Girolami, Särkkä, Stuart, and Zygalakis [42] proposed a sampling-based single step method built on Runge-Kutta methods. Denote by $\boldsymbol{y}_{t_{n+1}} = \Psi_h(\boldsymbol{y}_{t_n}) = \boldsymbol{y}_{t_n} + h\sum_{j=0}^{s-1} b_j k_{j,n}$ a Runge–Kutta one-step integrator of order $\mathcal{O}(h^{s+1})$. The authors propose to model the solution by

$$\boldsymbol{y}(t + t_n) = \Psi_{t-t_n}(\boldsymbol{y}_{t_n}) + \xi_n(t - t_n),$$

where ξ_n is a continuous Gaussian process with covariance of order Ch^{s+1}. That is, the algorithm adds a continuous Gaussian error to a one-step integration scheme. The overall result is a randomized solver with a free-form posterior. Lie, Stuart, and Sullivan [135] have improved the convergence analysis for this method. A special case of this general scheme has been proposed earlier in Fierro and Torres [64] and analyzed in some more detail in Krebs [120].

Analogously, Teymur, Zygalakis, and Calderhead [210] proposed a construction similar in spirit to Conrad, Girolami, Särkkä, Stuart, and Zygalakis [42] for multistep methods. They show that a multistep method of order s perturbed by additive Gaussian noise with variance of order $(h^{s+1})^2$ maintains the convergence order in expectation. Their proof is analogous to Conrad, Girolami, Särkkä, Stuart, and Zygalakis [42], but the multistep method requires less function evaluations for the same order.

Abdulle and Garegnani [1] have presented a symplectic algorithm that randomizes step sizes.

In an orthogonal approach, John and Wu [106] apply Bayesian analysis to the output of non-probabilistic finite difference schemes to provide credible intervals for the numerical solution. Their method

is more of a hybrid method than a pure probabilistic method as the Bayesian analysis is applied only after a numerical solution is found. Nevertheless, the final result is a probability distribution over the numerical solution which justifies its inclusion in this list.

7 A practical PFOS implementation

The previous chapter has introduced a probabilistic method for the solution of IVPs with known guarantees. However, solving IVPs is considered to be a "solved task" [69] and, as a consequence, the novel description and functionality will only gain momentum when users not familiar with the literature are provided with a reliable reference implementation at the same time.

This chapter presents another main contribution of this thesis: An implementation of the probabilistic filtering ODE solver which offers the novel probabilistic description while retaining most of the extra functionality of commonly available implementations.

The work of this chapter is based on Schober, Särkkä, and Hennig [187].

7.1 Requirements of a reference implementation

Practitioners are interested in code that *reliably* solves their problem *as fast as possible*, and they will use available features as they see fit. The theoretical analysis can serve as a convincing justification, but only if the method stands the test of empirical evaluation. If we hope to convince users of applying the probabilistic approach, we must offer a code that can compete on par with existing implementations.

As has already been pointed out (cf. Sections 3.4 and 5.2), a good numerical code will automatically adapt its step size during runtime throughout the integration domain. The first computational scheme proposed with this feature has been reported by Richardson and Gaunt [178] for an extrapolation method. Merson [150] introduced the standard technique of embedded Runge-Kutta pairs. Ceschino [36] and Nordsieck [156] proposed heuristics for multistep methods. In this latter case, the convergence theory of methods with variable step size is more involved due to memory effects in the system and is still an active area of research [123, 122].

Soon after the introduction of step size adaptations, it was considered a standard requirement for numerical integrators as is evident by the first established benchmark [99]. Other useful features, such as order selection [28, 32], stiffness detection [192, 193, 169], and stiffness-aware step size adaptation [78] have not been established as default requirements.

For this work, we have to refrain from these advanced features, as the theory of the probabilistic integrators has not been developed thus far yet. Furthermore, recent trends in the theory of *boundary value methods* [10, 25, 148] may hint that the development of these features might not be the next logical step.

boundary value methods

In the design of our API, we have been guided by the existing codes in the `MATLAB` programming environment [195]. It can be argued that these codes are one of the most applied libraries besides the Sundials suite [96]. `MATLAB` can be considered the *de facto* standard for engineers in many industries. We provide a function interface that is identical to `MATLAB`'s default ODE method in its standard activation. Thus, we expect that future users can easily integrate our implementation into existing projects and can start experimenting with the probabilistic formulation with little overhead.

7.2 Error estimation and step size selection

While the general algorithm described in Section 6.7 can be applied to any IVP at this stage, a modern ODE solver also requires the ability to automatically select sensible values for its hyper-parameters. The filter has three remaining parameters to choose: the dimensionality q of the state space, the diffusion amplitude σ^2 and the step size h.

To obtain a globally consistent probability distribution, we fix $q = 2$ throughout the integration to test the third-order method presented in Section 6.7. For the remaining two parameters, we first note that estimating σ^2 will lend itself naturally to choose the step size. To see this, one needs to make the connection to classical ODE solvers and the interpretation of the state-space model. In classical ODE solvers, h_n is determined based on local error analysis, i.e., h_n is a function of the error τ_{t_n} introduced from step t_{n-1} to step t_n. Then, h_n is computed as a function of the allowed tolerance and the expected error which is assumed to evolve similarly to the current error.

As is common in solving IVPs, we base error estimation on *local errors*. Assume that the predicted solution $\boldsymbol{m}_{t_{n-1}}$ at time t_{n-1} is error-free, i.e., $\boldsymbol{C}_{t_{n-1}} = \boldsymbol{0}$. Then, by Equations (15b) and (16a), we have

$$p(\lambda_n \,|\, \sigma^2) = \mathcal{N}(\lambda_n; z_n - \boldsymbol{H}_1 \boldsymbol{m}^-_{t_n}, \boldsymbol{H}_1 \sigma^2 \bar{\boldsymbol{Q}}(h) \boldsymbol{H}_1^\mathsf{T}). \tag{67}$$

One way to find the optimal σ^2 is to construct the maximum likelihood estimator from Equation (67) which is given by

$$\begin{aligned}\hat{\sigma}^2 &= (z_n - \boldsymbol{H}_1 \boldsymbol{m}_{t_{n-1}}^-)^\intercal (\boldsymbol{H}_1 \bar{\boldsymbol{Q}}(h) \boldsymbol{H}_1^\intercal)^{-1} (z_n - \boldsymbol{H}_1 \boldsymbol{m}_{t_{n-1}}^-) \\ &= \frac{(z_n - \boldsymbol{H}_1 \boldsymbol{m}_{t_{n-1}}^-)^2}{\boldsymbol{H}_1 \bar{\boldsymbol{Q}}(h) \boldsymbol{H}_1^\intercal}.\end{aligned}$$

For the last equation, we used the fact that all the involved quantities are scalars.

To allow for a greater flexibility of the model, we allow the amplitude σ^2 to vary for different steps $\sigma_{t_n}^2$. Note that the mean values are then no longer independent of σ^2, because the factor no longer cancels out in the computation of K_n in Equation (16c). However, this situation is indeed intended: If there was more diffusion in $[t_{n-1}, t_n]$, we want a stronger update to the mean solution as the observed value is more informative. Additionally, Equation (16a) is independent of $\sigma_{t_n}^2$ or any other covariance information $\boldsymbol{C}_{t_n}^-, \boldsymbol{Q}(h)$. Therefore, we can apply Equation (16a) before Equation (15b), update $\sigma_{t_n}^2$ and then continue to compute the rest of the Kalman step. This inference scheme is similar in spirit to [102, §11], but follows the general idea of error estimation in numerical ODE solvers—in particular, the *Milne device* (see Milne [152] and Section 3.4)—, where only local error information is available.

At this point, the inference interpretation of numerical computation comes to bear: once the initial modeling decision—modeling a deterministic object with a probability measure to describe the uncertainty over the solution—is accepted, everything else follows naturally from the probabilistic description. Most importantly, there are no neglected higher-order terms, as they are all incorporated in the diffusion assumption.

This kind of lightweight error estimation is a key ingredient to probabilistic numerical methods: one goal of a probabilistic model is improved decisions under *uncertainty*. This uncertainty is necessarily a crude approximation, since a more accurate error estimator could be used to improve the overall solution quality. However, the reduction in computational efforts up to a tolerated error is exactly what modern numerical solvers try to achieve.

This error estimate can now be used in the conventional way of adapting the step size which we will restate here to give a complete description of the inference algorithm (see also [28]). Given an error weighting vector $\boldsymbol{w}$, the algorithm computes the weighted expected error

$$(\boldsymbol{D}_{t_n})_i = (\boldsymbol{H}_\mathbf{1} \sigma_{t_n}^2 \bar{\boldsymbol{Q}}(h_n) \boldsymbol{H}_\mathbf{1}^\intercal)_i^{1/2} \boldsymbol{w}_i,$$

where $\bar{\boldsymbol{Q}}(h_n) = [\sigma_{t_n}^2]^{-1}\boldsymbol{Q}(h_n)$ is the normalized diffusion matrix. Then, it checks whether some error tolerance with parameter ϵ is met

$$\boldsymbol{D}_{t_n} \leq \bar{\epsilon} = \epsilon \frac{h_n}{S} \tag{68}$$

where h_n is the step length and either $S = 1$ (error per unit step) or $S = h_n$ (error per step) as per user's choice. If Equation (68) holds, the step is accepted and integration continues. Otherwise, the step is rejected as too inaccurate and repeated. In both cases, a new step length is computed which will likely satisfy Equation (68) on the next step attempt. The new step size is computed as

$$h_{n+1} = \rho \left(\frac{\bar{\epsilon}}{\boldsymbol{D}_{t_n}} \right)^{\frac{1}{q+1}},$$

where $\rho \in (0,1)$, $\rho \approx 1$ is a safety factor. Additionally, we also follow established best practices [82] to limit the rate of change $\eta_{\min} < {}^{h_{n+1}}/_{h_n} < \eta_{\max}$. In our code, we set $\rho = 0.95$, $\eta_{\min} = 0.1$ and $\eta_{\max} = 5$.

7.2.1 Global vs. local error estimation

The results presented in preceding sections pertain to the estimation of *local* extrapolation errors. It is a well-known aspect of ODE solvers [82, §III.5] that the *global* error can be exponentially larger than the local error.

More precisely, to scale the stochastic process such that the variance of the resulting posterior measure relates to the square *global* error, the intensity σ_n^2 of the stochastic process must be multiplied by a factor [82, Thm III.5.8] $\exp(L^*(T - t_0))$, where L^* is a constant depending on the problem. Although related, L^* is not the same as the local Lipschitz constant L and harder to estimate in practice (more details in [82, §III.5]).

We stress that this issue does not invalidate the probabilistic interpretation of the posterior measure as such. It is just that the scale of the posterior has to be estimated differently if the posterior is supposed to capture global error instead of local error. In practice, the global error estimate resulting from this re-scaling is often very conservative.

7.3 Experiments

To evaluate the model, we provide two sets of experiments. First, we qualitatively examine the uncertainty quantification by visualiz-

ing the posterior distribution of two example problems. We also compare our proposed observation assumption against the model described by Chkrebtii, Campbell, Calderhead, and Girolami [39]. Second, we more rigorously evaluate the solver on a benchmark and compare it to existing non-probabilistic codes. Our goal in this work is to construct an algorithm that produces meaningful probabilistic output at a computational cost that is comparable to that of classical, non-probabilistic solvers. The experiments will show that this is indeed possible. Other probabilistic methods, in particular that of Chkrebtii, Campbell, Calderhead, and Girolami [39], aim for a more expressive, non-Gaussian posterior. In exchange, the computational cost of these methods is at least a large multiple of that of the method proposed here, or even polynomially larger. These methods and ours differ in their intended use-cases: More elaborate but expensive posteriors are valuable for tasks in which uncertainty quantification is a central goal, while our solver aims to provide a meaningful posterior as additional functionality in settings where fast estimates are the principal objective.

7.3.1 Uncertainty quantification

We apply the probabilistic filtering ODE solver on two problems with attracting orbits: the Brusselator [132] and van der Pol's equation [173]. The filter is applied twice on each problem, once with a fixed step size and once with the adaptive step size algorithm described in Section 7.2. To get a visually interesting plot, the fixed step size and the tolerance threshold were chosen as large as possible without causing instability. Both cases are modeled with a local diffusion parameter σ_n^2 which is estimated using the maximum likelihood estimator of Section 7.2. In the following plots, the samples use the scale σ_n^2 arising from the local error estimate. Because these systems are attractive, the global error correction mentioned in Section 7.2.1 would lead to significantly more conservative uncertainty.

The Brusselator is the idealized and simplified model of an autocatalytic multi-molecular chemical reaction [132]. The rate equations for the oscillating reactants are

$$\begin{aligned} y_1' &= A + y_1^2 y_2 - (B+1) y_1 \\ y_2' &= B y_1 - y_1^2 y_2, \end{aligned} \tag{69}$$

where A and B are positive constants describing the initial concentrations of two reactants. Following [82], we set $A = 1$, $B = 3$ and $(y_1(0), y_2(0))^\intercal = (1.5, 3)^\intercal$. The integration domain $\mathbb{T} = [0, 10]$ has been chosen such that the solution completes one cycle on the attractor after an initial convergence phase.

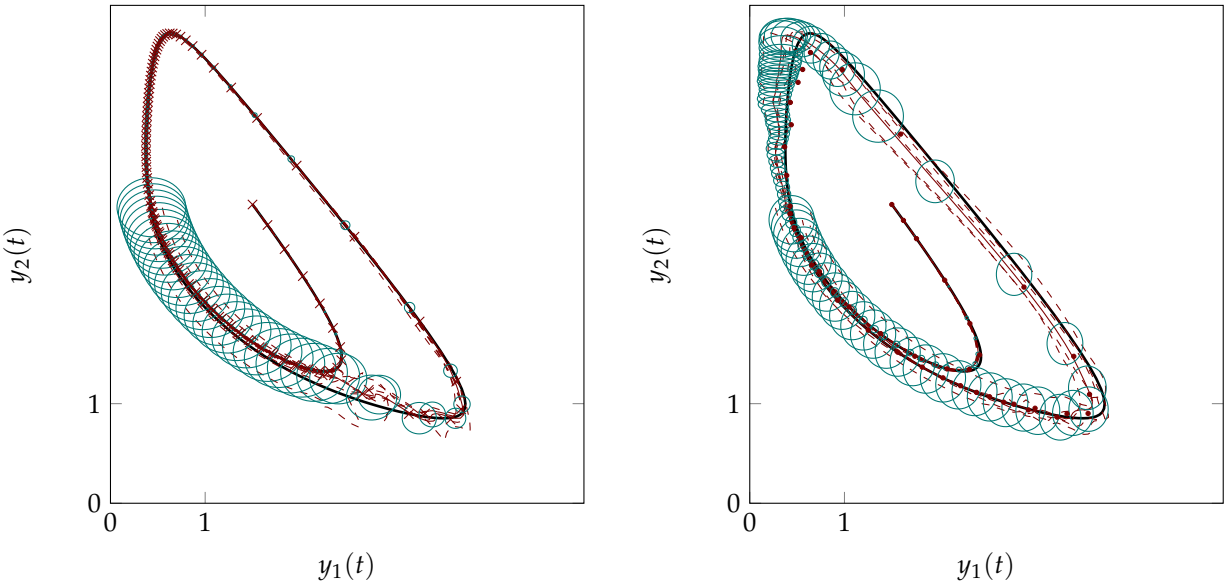

Figure 13: Numerical solution of the Brusselator (69) using the probabilistic filtering ODE solver. The plots show the solution computed by `ode45` using `RelTol` $=$ `AbsTol` $= 1 \times 10^{-13}$ (black, background), the posterior mean (red, thick line), iso-contourlines of twice the posterior standard deviation at a subsample of the knots (green) and samples from the posterior distribution (red, dashed lines). **Left:** Using a fixed step size of $h = h_n = 0.0834$. The computation requires 120 steps. **Right:** Using the adaptive step size selection with error weighting $w_i(y) = (\tau y_i + \tau)^{-1}, \tau = 0.1$. The computation requires 43 steps. See [82, §1.6] for details.

The results in Figure 13 demonstrate the effectiveness of the error estimator. This problem also demonstrates the quality and utility of the step size adaptation algorithm, since on the majority of the solution trajectory the algorithm is not limited by stability constrains. In the right plot, it can be seen how an increase in step size $h_{n+1} > h_n$ can also lead to a reduction in posterior uncertainty. This is a consequence of $\sigma^2_{t_{n+1}} / \sigma^2_{t_n} < 1$.

Figure 14 also displays the solution as a function of time.

Van der Pol's equation [173] describes an oscillation with a nonlinear damping factor α

$$\begin{aligned} 0 &= y'' + \alpha y' + y \\ \alpha &= \mu(y^2 - 1) \end{aligned} \tag{70}$$

with a positive constant $\mu > 0$. Originally, this model has been used to describe vacuum tube circuits. The limit cycle alternates between a non-stiff phase of rapid change and a stiff phase of slow decay. The larger μ the more pronounced both effects are. In our example, we set $\mu = 1$ and integrate over one period with the initial value on the graph of the limit cycle. Exact values can be found in [82, §I.16].

Figure 15 plots the filter results. Figure 16 displays the solution as a function of time. In the case of van der Pol's equation, the benefit of step size adaptation is essentially nil, because conservative adaptation—in particular from a cautious starting step size—consumes the gains on the non-stiff parts. However, the example demonstrates the capability to learn an anisotropic diffusion model for individual components.

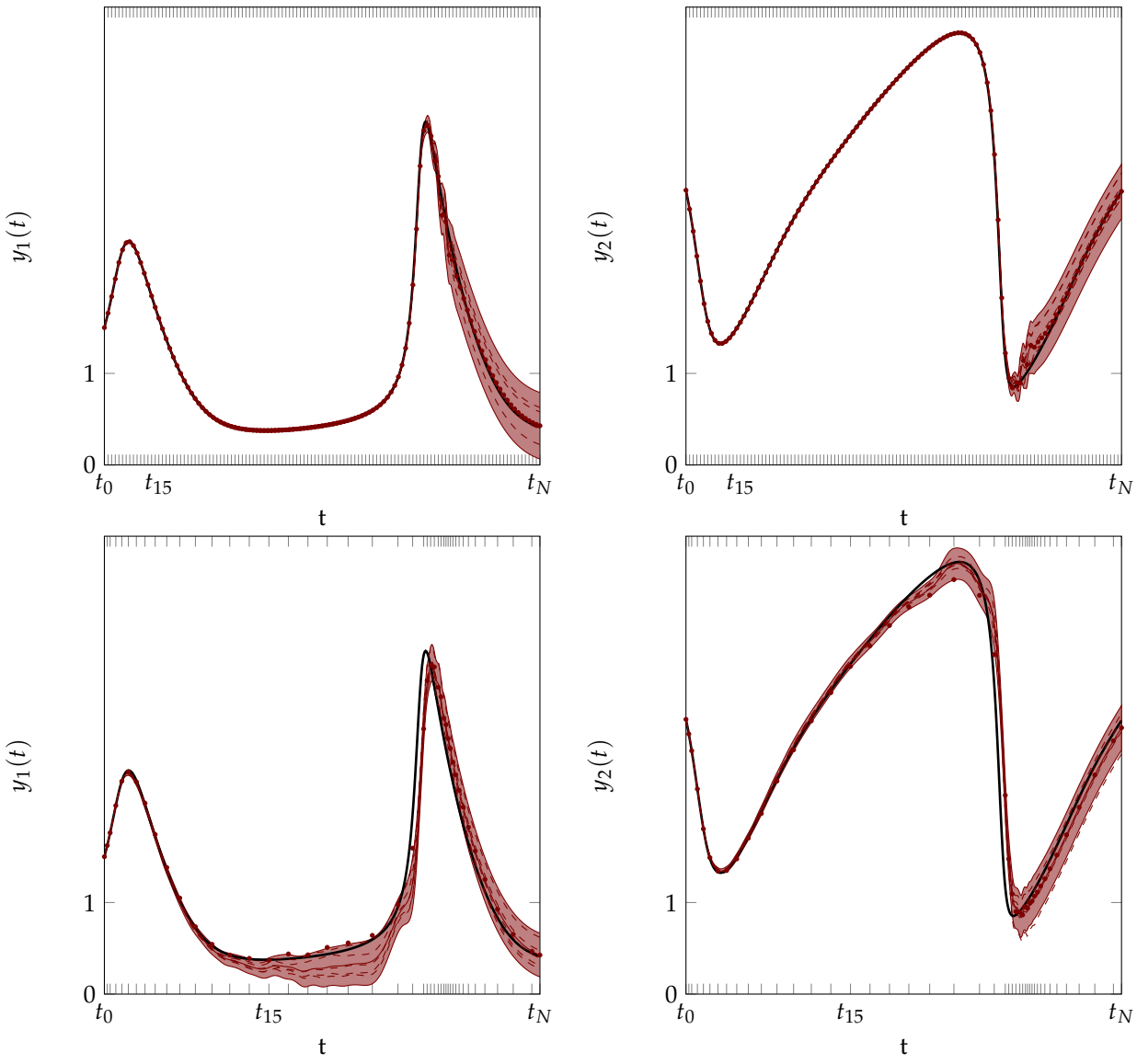

Figure 14: Numerical solution of the Brusselator (69) using the probabilistic filtering ODE solver plotted against time. The plot shows the true solution (black line), the mean of the filtering distribution (red dots), the posterior mean (red, thick line) plusminus two times standard deviation (light red, filled area) and samples from the posterior (red, dashed line). Tickmarks in t indicate mesh points. **Top:** Using a fixed step size. **Bottom:** Using adaptive step size selection.

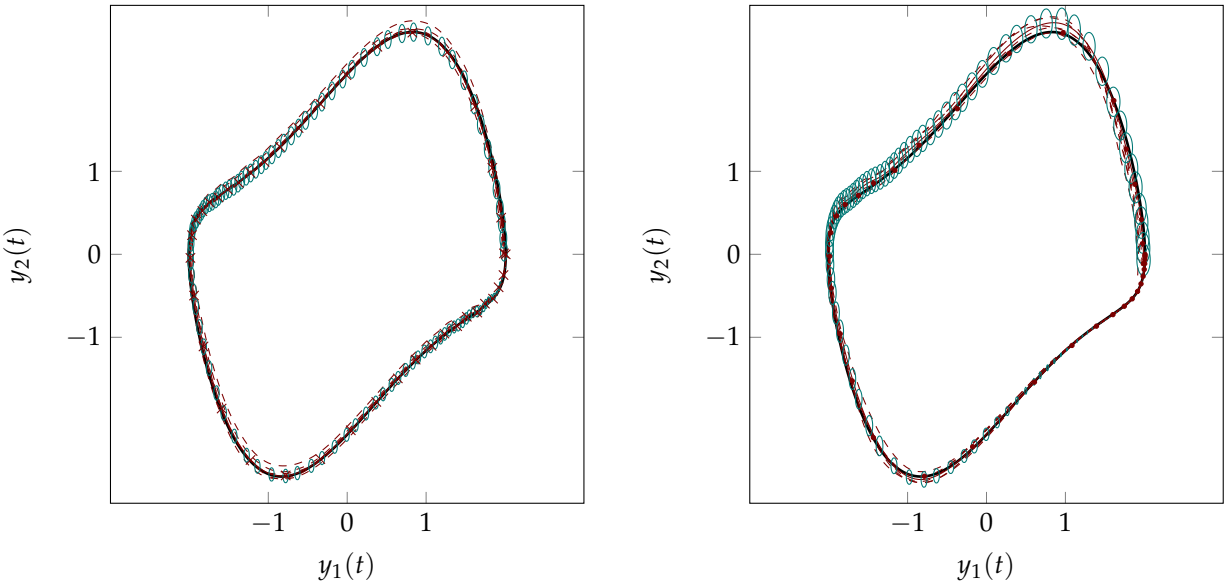

Figure 15: Numerical solution of van der Pol's equation (70) using the probabilistic filtering ODE solver, integrated over one limit cycle period $\mathbb{T} = [0, T]$ with initial value $y(0) = (A, 0)^\intercal$, where $T \approx 6.6633$ and $A \approx 2.0086$. The plots show the solution computed by `ode45` using `RelTol` $=$ `AbsTol` $= 1 \times 10^{-13}$ (black, background), the posterior mean (red, thick line), iso-contourlines of twice the posterior standard deviation at a subsample of the knots (green) and samples from the posterior distribution (red, dashed lines). **Left:** Using a fixed step size of $h = h_n = 0.1667$. The computation requires 40 steps. **Right:** Using the adaptive step size selection with error weighting $w_i(y) = (\tau y_i + \tau)^{-1}, \tau = 0.1$. The computation requires 41 steps. See [82, §1.6] for details.

Finally, we compare two different strategies of quantifying the uncertainty. To this end, we compare our proposed model to the observation model proposed by Chkrebtii et al. [39, §3.1]. In this case, we set $z_n = f(t_n, (u_{t_n})_0), u_{t_n} \sim \mathcal{N}(\boldsymbol{m}^-_{t_n}, \boldsymbol{C}^-_{t_n})$. Figure 17 shows samples of the posterior distribution, computed with two different evaluation schemes. This scheme is not exactly the same as the one proposed by Chkrebtii et al.—their algorithm actually has cubic complexity in the number of f-evaluations; thus, it is limited to a relatively small number of evaluation steps. But our version captures the principal difference between their algorithm and the simpler filter proposed here: Their algorithm draws separate samples involving independent evaluations of f at perturbed locations, while ours draws samples from a single posterior constructed from one single set of f-evaluations. As expected, the model of Chkrebtii et al. provides a richer output structure, e.g., by identifying divergent solutions (right subplot) if the solver leaves the region of attraction. However, to obtain individual samples, the entire algorithm has to run repeatedly, so the cost of producing S samples is S times that of our algorithm, which produces all its samples in one run, without requiring additional evaluations of f.

7.3.2 Benchmark evaluation

As is the case with many modern solvers, the theoretical guarantees do not extend to the full implementation with error estimation and step size control. Therefore, an empirical assessment is necessary to compare against trusted implementations. We compare the proposed Kalman filter to a representative set of standard algorithms on

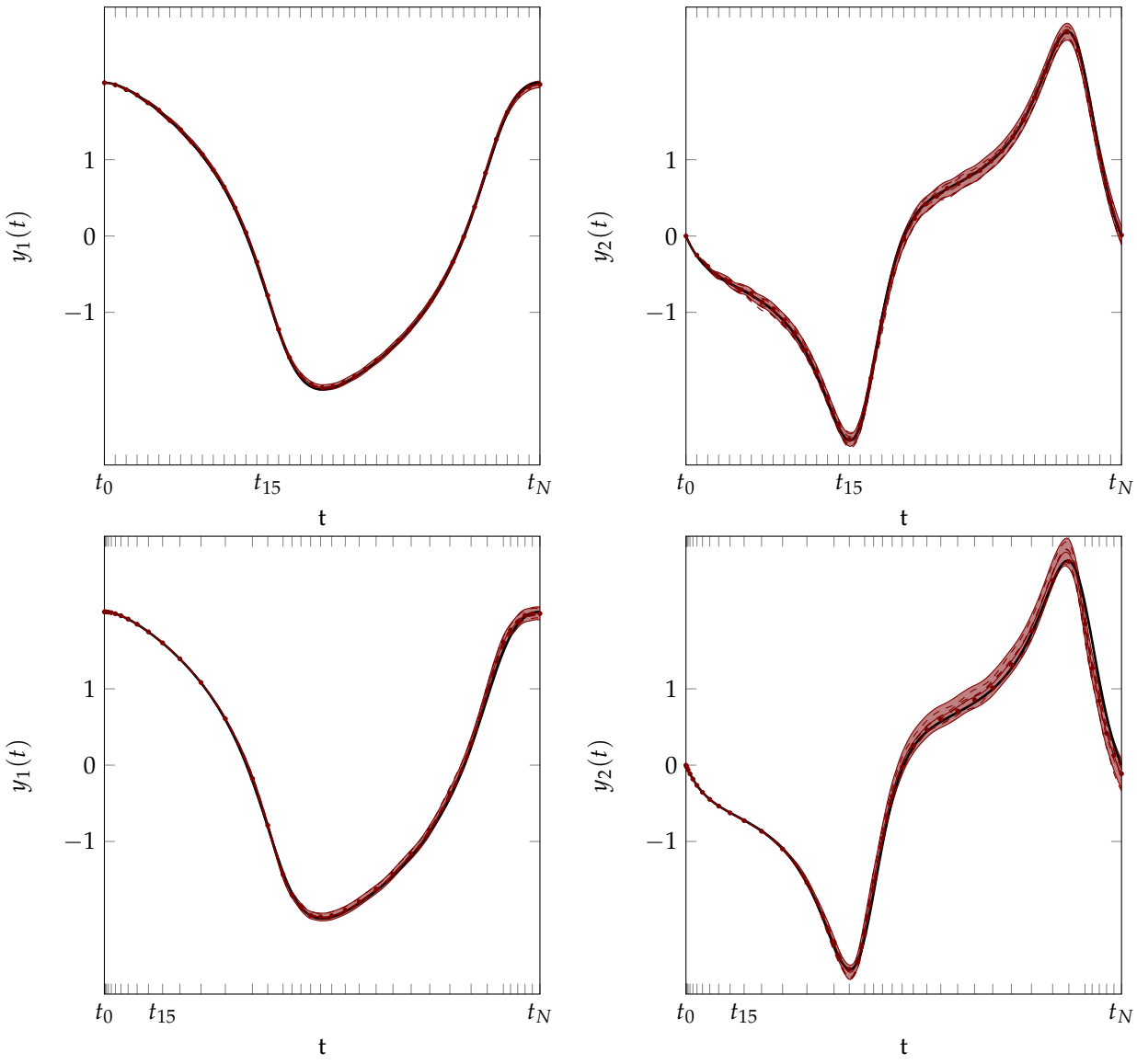

Figure 16: Numerical solution of van der Pol's equation (70) using the probabilistic filtering ODE solver plotted against time. The plot shows the true solution (black line), the mean of the filtering distribution (red dots), the posterior mean (red, thick line) plusminus two times standard deviation (light red, filled area) and samples from the posterior (red, dashed line). Tickmarks in t indicate mesh points. **Top:** Using a fixed step size. **Bottom:** Using adaptive step size selection.

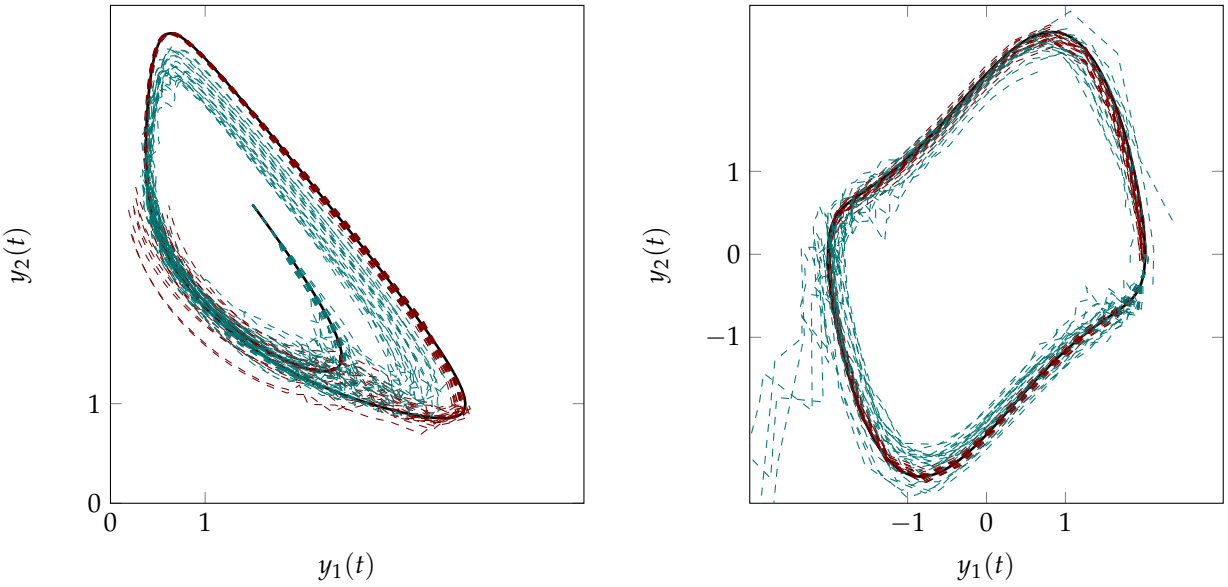

Figure 17: Comparison of two different evaluation strategies on problems (69) and (70). Red: samples from the posterior as in Figs. 13 and 15. Green: Similar, but evaluating at $z_n = f(t_n, (u_{t_n})_0), u_{t_n} \sim \mathcal{N}(\boldsymbol{m}_{t_n}^-, \boldsymbol{C}_{t_n}^-)$. This is similar to [39].

the DETEST benchmark set [99]. While other standardized tests have been proposed [46, 121], DETEST has repeatedly been described as representative [194, 54]. By choosing the same comparison criteria across all test problems and tested implementations, the benchmark provides the necessary data to make predictions on the behavior on a large class of problems.

Two different dimensions of performance are considered in [99]: the *computational cost* and the *solution quality*. Computational cost is reported in execution time (in seconds) and number of function evaluations (abbreviated as #FE). Although the former is more relevant in practice, we only report the latter here as the codes in [99] are outdated and our proof-of-concept code is not yet optimized for speed. Nevertheless, since the execution times are proportional to the #FE, this provides a reliable estimator of computational efficiency. DETEST only considers methods with automatic step size adaptation, and thus measures the solution quality by comparing the local error with the requested tolerance ϵ. A code is considered to produce high quality solutions if the results are within the requested tolerance, but are also not of excessive unrequested higher accuracy. Therefore, errors are reported per unit step. Reported are the maximum error $\max\{\xi_n[h_n\epsilon]^{-1} \,|\, n = 1, \ldots, N\}$ per unit step and the percentage of deceived steps $|\{\xi_n \,|\, \xi_n > h_n\epsilon, n = 1, \ldots, N\}|/N$, where the local errors ξ_n are defined as $\left|\left|y_{t_n} - y(t_n; y(t_{n-1}) = y_{t_{n-1}})\right|\right|_\infty$ and $y(t_n; y(t_{n-1}) = y_{t_{n-1}})$ defines the IVP $y' = f(t, y), y(t_{n-1}) = y_{t_{n-1}}$, $t \in [t_{n-1}, t_n]$.

Here, we report the results from the proposed solver originating from the IWP(2) model as well as the results from the original Hull et al. paper [99]. We haven't been able to obtain a copy of the codes

Method	Total fcn. evals.	Avg. % deceived	Max. error
$\epsilon = 10^{-3}$			
Extrapolation	16553	2.0	7.8
Adams (Krogh)	5394	1.1	5.3
Adams (Gear)	9498	0.9	1.5
RK (4th, Kutta)	8363	5.1	25.9
RK (6th, Butcher)	11105	5.1	1788.1
RK (8th, Shanks)	12355	6.3	1120.6
RK (3th, Shampine)	15085	5.9	2.4
RK (5th, Shampine)	5785	11.2	9.5
Adams (Shampine)	5692	6.5	7.7
PNM	19091	0.2	1.5
$\epsilon = 10^{-6}$			
Extrapolation	26704	0.1	2.3
Adams (Krogh)	11353	1.4	7.3
Adams (Gear)	18155	0.8	2.6
RK (4th, Kutta)	30763	1.8	29.1
RK (6th, Butcher)	23540	1.6	142.5
RK (8th, Shanks)	20493	4.2	4.7
RK (3th, Shampine)	430975	0.0	1.9
RK (5th, Shampine)	19879	0.0	1.1
Adams (Shampine)	10777	3.6	6.3
PNM	405469	0.0	1.4
$\epsilon = 10^{-9}$			
Extrapolation	43054	0.0	0.6
Adams (Krogh)	18984	0.5	4.0
Adams (Gear)	38439	2.3	2.7
RK (4th, Kutta)	146262	0.3	2.9
RK (6th, Butcher)	58634	0.9	443.4
RK (8th, Shanks)	39663	2.1	20.9
RK (3th, Shampine)	13587187	3.1	689.0
RK (5th, Shampine)	103345	0.1	2.4
Adams (Shampine)	18274	2.2	11.5
PNM	12731730	4.5	1938.0

Table 4: Summary of DETEST results

used in Hull et al. and only report their numbers for sake of completeness. We also ran the tests on the solvers provided in MATLAB. Table 4 lists the summary results for all methods and all tolerances. Detailed results on individual problems are depicted in Figures 18–20. For a complete and detailed description of the benchmark, we refer to [99]. Our implementation is publicly available.[1]

[1] https://pn.is.tuebingen.mpg.de/code/pfos

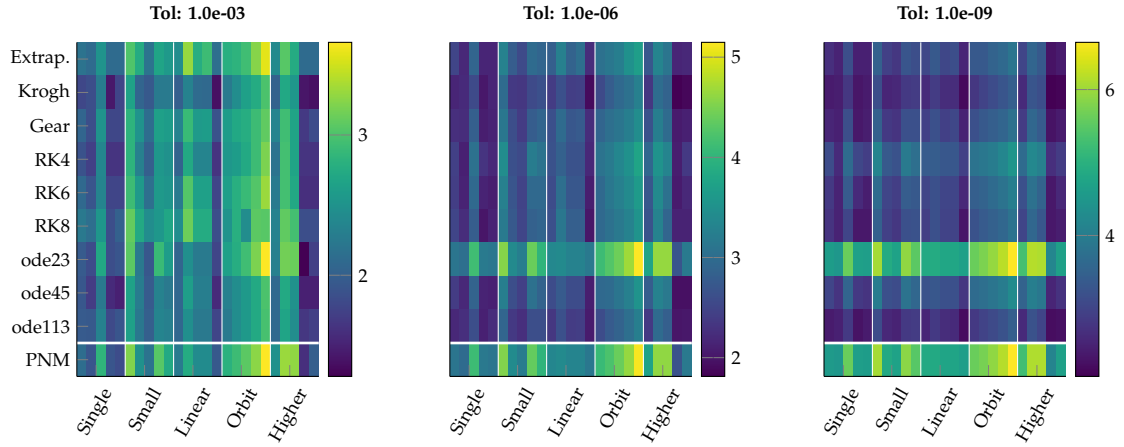

Figure 18: $\log_{10}$(#FE), the number of function evaluations in logarithmic scale, for all tested methods and individual problems.

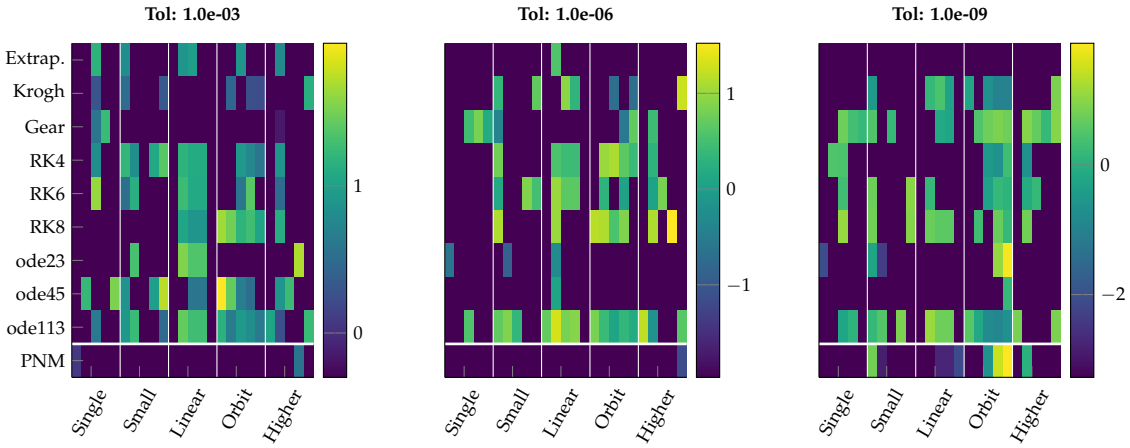

Figure 19: $\log_{10}(|\{\xi_n \,|\, \xi_n > \epsilon, n = 1,\ldots,N\}|N^{-1})$, the percent of deceived steps in logarithmic scale, for all tested methods and individual problems.

In addition to the benchmark results, we analyze the error estimation model from a probabilistic perspective. Figure 21 plots the cumulative distribution function (CDF) of the local error ξ_n, as defined above, divided by the estimated local error $(Q(t_n))_{00}^{1/2} = (\sigma_n^2 \bar{Q}(h_n))_{00}^{1/2}$ for each set of five tasks (different blue colored lines) of each of the five problem classes (figures from left to right). Under the algorithm's internal model, the error is assumed to follow a Gaussian distribution

$$P(y_{t_n} \,|\, \hat{y}_{t_n}) = \mathcal{N}(y_{t_n}; \hat{y}_{t_n}, (\boldsymbol{Q}(h_n))_{00}).$$

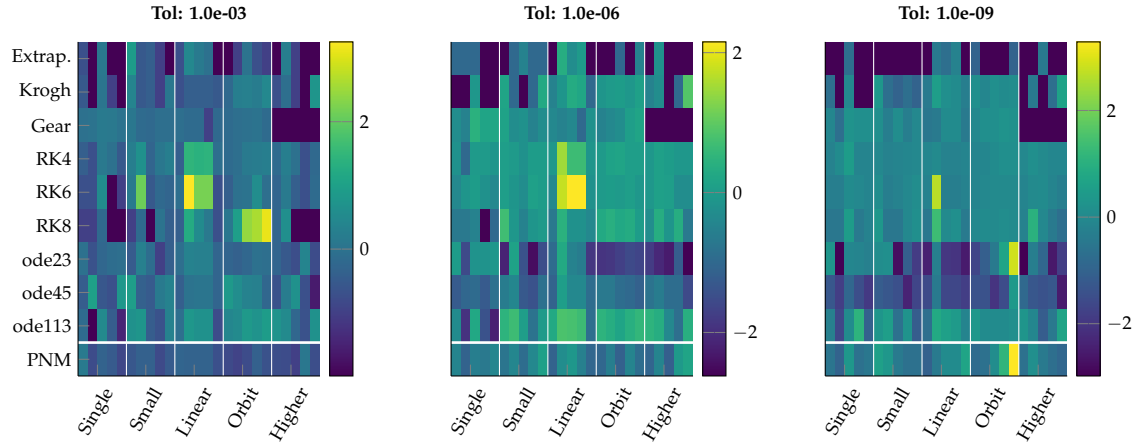

Figure 20: $\log_{10}(\max\{\xi_n[h_n\epsilon]^{-1} \mid n = 1, \ldots, N\})$, the maximum error per unit step in logarithmic scale, for all tested methods and individual problems

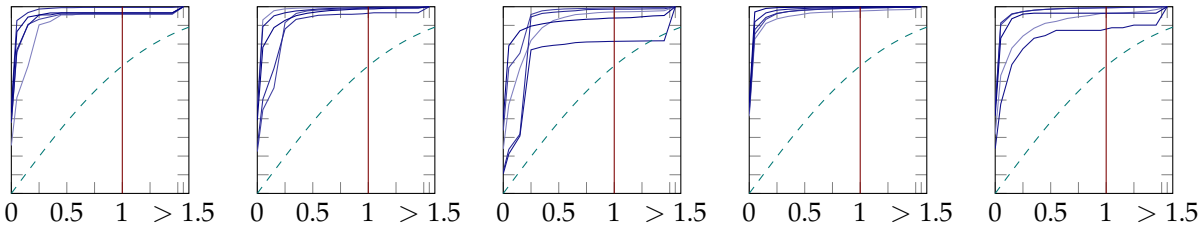

Figure 21: Empirical cumulative distriubtion function (CDF) of true local errors ξ_n divided by the estimated local errors $(Q(t_n))_{00}^{1/2}$. Ticks on the y-axis are spaced at 0.1 intervals from 0 to 1. Values less than 1 (red line) are *over-estimated* leading to a conservative step size adaptation. Green dashed line shows the CDF of the $\chi(1)$-distribution which implies that the empirical distribution has weaker tails. See text for more details.

Hence, if that model were a perfect fit, the scaled absolute error plotted in this figure would be χ-distributed:

$$P(|y_{t_n} - \hat{y}t_n| \, (\boldsymbol{Q}(h_n))_{00}^{-1/2}) = \chi(1).$$

The comparison with the CDF of $\chi(1)$ shows that the true local error has weaker tails than the predicted χ-distribution.

So, as expected, the error estimator is typically a conservative one.

While the probabilistic method does not achieve the same high performance as modern higher-order codes, the performance matches the results of a production Runge–Kutta code of the same order. This is of particular interest since applications in the low-accuracy regime could benefit the most from accurate error indicators [69].

Part IV

Applications of Probabilistic BVP Solvers

8

Introduction to tractography

We will now present an application of probabilistic ODE solvers in the field of neuroscience. *Tractography* is the computational task of finding neural connections based on medical imaging techniques. Tractography has both applications in medicine [209] and neuroscience [8]. However, the difficult physical measurement process as well as our limited understanding of the brain introduce uncertainties to the problem. It is reasonable to assume that the methodology introduces another source of errors and uncertainty in the process. Probabilistic ODE solvers are one way to communicate and propagate these uncertainties through the computational pipeline, thus providing a more complete picture to practitioners.

Tractography

In this chapter, we will provide the necessary background for this application. In Chapters 9 and 10, we will demonstrate how probabilistic ODE solvers can utilized in medical imaging.

8.1 Neural connectivity studies

In the brain, nerve cells—called *neurons*—are the main computational units responsible for signal propagation and processing [19]. Neurons consist of the *soma*, *dendrites* and the *axon*. Axons are long, thin, pipe-like extensions of neurons that forward the electrical signals of the brain where the signals are received by corresponding dendrites of other neurons via *synapses*. The soma, or cell body, contains the nucleus and its function is to maintain and support the signal processing parts of the neuron.

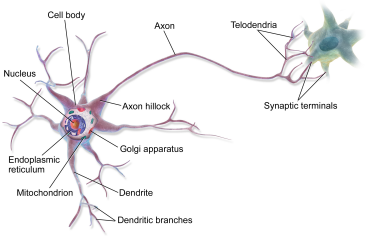

Figure 22: Illustration of a neuron. (Source: Wikipedia CC BY 3.0 by Bruce Blaus https://commons.wikimedia.org/w/index.php?curid=28761830)

In the human brain, axons are often surrounded by myelin layers of glia cells. The myelination isolates parts of the axon which increases the speed of the signal propagation. In *post mortem* dissections, volumes including much myelination appear lighter in color. These areas are called *white matter (WM)* and they contain the bulk of the inter-neural connections. Accordingly, the *gray matter (GM)* contains most of the cell bodies [19].

white matter (WM)

gray matter (GM)

One way to study the brain, is to analyze the various connections in the brain. Since the brain is a complex organ and the available methodology is quite recent, our understanding of the brain is currently fragmented over levels both of physical extension and structural facilities.

Early research on structural facilities has mostly focused on *cortical localization* [139], i.e., mapping specific areas of the brain to behavioral functions such as speech [22]. With the advent of modern imaging technology, research interest has shifted on studying the inter-neural connections [34]. This area has since come to be known as *connectomics* [203, 215].

connectomics

There are three levels of connectivity studies. Structural connectivity studies the anatomical foundations of whether there exists any connection between two *regions-of-interest (ROI)* [74]. Functional connectivity refers to observations of correlations and other patterns in neural activity [66]. The third level—effective connectivity—tries to identify the causal dependencies of the functional level [66]. A good introduction can be found in [97].

regions-of-interest (ROI)

In physical extensions, the analysis reaches from individual intercellular connections [89] on the microscopic scale to cortical tissue of functional brain units [45] on the macroscopic scale. The mesoscopic scale is concerned with the analysis of axon bundles—called *fibers* or *fascicles*—that connect different functional units. A group of fibers that has similar origin and destination is called a *tract*. Reconstruction of these tracts is called *tractography* and has both neuroscientific [154] and clinical [58] applications.

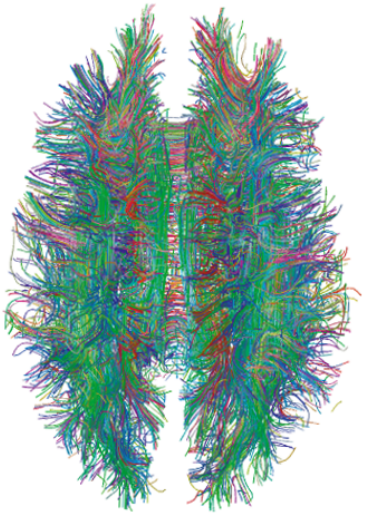

Figure 23: An example of white matter tractography. (Source: Wikipedia CC BY 2.5 by Xavier Gigandet et al. https://commons.wikimedia.org/w/index.php?curid=8134159)

fibers

tract

tractography

Since tractography is a key methodology to building connectomes, it is imperative to provide measures of uncertainty to researchers and physicians such that the results can be interpreted accordingly.

8.2 Diffusion-weighted MR imaging

Diffusion-weighted MR imaging (DWI) is a particular type of *magnetic resonance (MR)* imaging technique to measure molecular diffusivity *in vivo*. The theoretical foundations [212] and first practical implementations [204] have been developed in the middle of the 20th century while the first applications to neuroscience have been published in the late 1980s [222, 129, 155]. Until then, the only way to construct (partial) tractographies was *ex vivo* analysis using fluorescent tracers [79, 88]. However, there is only one generation of research experience and knowledge available yet [130] and best practices are only beginning to be formulated [108] and discussed [219].

Diffusion-weighted MR imaging (DWI)

magnetic resonance (MR)

DWI, like any magnetic resonance imaging technique, is based on the excitation of the spins in hydrogen nuclei. Frequency and phase properties of these spins can be changed by inducing magnetic (gradient) fields with coils. These changes cause echo effects whose magnetic fields can be translated into signals and measured with the same coils. For details, we refer the reader to [81] and references therein.

The hydrogen atoms in water and fat of the body undergo a diffusion process. In a homogeneous medium, this process could adequately be described by Brownian motion. In the brain, this diffusion is anisotropic due to fiber membranes and other inhomogeneities which leads to varying *apparent diffusion coefficients (ADCs)* for different directions g and voxels v. In the special case of DWI, the Torrey-Bloch equations [212] can be used to relate a signal $S_k(v)$ at voxel v using a gradient g_k to the ADC $D(v, g_k)$ as [204]

apparent diffusion coefficients (ADCs)

$$S_k(v) = S_0 \exp(-bD(v, g_k))$$

where S_0 is a diffusion-free reference signal and b are constants that include physical properties and other experiment parameters like the field strength [219] (see Section 8.4).

In *diffusion tensor imaging (DTI)* [12], measurements from various gradient directions are taken successively, such that the main axes of anisotropy can be resolved. The result is a voxel-wise second-order tensor, i.e., a symmetric positive (semi-)definite 3×3 matrix

$$\boldsymbol{D}(v) = \begin{pmatrix} D_{xx}(v) & D_{xy}(v) & D_{xz}(v) \\ D_{xy}(v) & D_{yy}(v) & D_{yz}(v) \\ D_{xz}(v) & D_{yz}(v) & D_{zz}(v) \end{pmatrix},$$

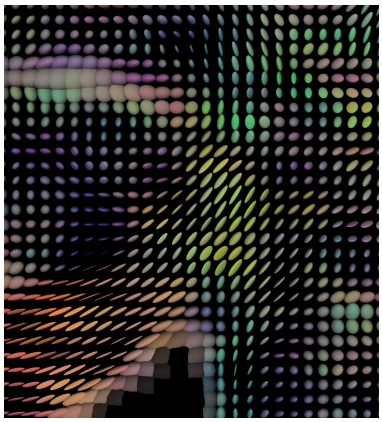

Figure 24: Illustration of DTI voxel data. (Source: Wikipedia CC BY-SA 3.0 by Thomas Schultz https://commons.wikimedia.org/w/index.php?curid=1201206)

where $D_{ij}(v)$ describes the ADC in the ij direction in voxel v. This diffusion tensor describes the second-moment of the local *orientation distribution function (ODF)* for every voxel. The advantages of DTI are the fast acquisition time, the wide availability of respective hardware, and the high *signal-to-noise ratio (SNR)* (as compared to more advanced imaging techniques).

orientation distribution function (ODF)

signal-to-noise ratio (SNR)

However, voxels containing many fiber crossings or only partial tracts, the second-order tensor is an inadequate description of the ODF [81]. To deal with these problematic voxels, researchers have developed imaging techniques with more advanced imaging protocols and models. These are commonly refered to as *high angular resolution diffusion imaging (HARDI)* and examples of these models include *diffusion spectrum imaging (DSI)* [221] and *q-ball imaging* [214]. But these methods require advanced hardware and long imaging time, thus limiting the applicability in larger studies or common clinic applications [81]. These methods also suffer from lower SNRs [213] such that more advanced post-processing is required—which also induces more human bias towards the current understanding of the brain.

high angular resolution diffusion imaging (HARDI)

Conceptually, DTI can be thought of as estimating the second moment of a random process by measuring the Fourier transform of a finite difference scheme. Connecting this to the overall theme of this thesis, it is clear that this process is inherently noisy with many

sources of uncertainty. On top, there are the usual problems of a physical realization. In addition to the common measurement noise of any physical device, sources of uncertainty in DWI include motion artifacts, eddy currents and partial volume effects [15, 108] with SNR sometimes as low as 3 : 1 [108].

This is not to say that the situation is futile, but modeling and communicating the inherent uncertainty is a necessity in order to advance the scientific knowledge. Even more so when it is considered how costly and laborious neurological research is.

8.3 Tractography

Tractography refers to the process of generating 3D models of tracts based on DWI data. The output of tractography algorithms are *streamlines*, or *candidate tracts*, that are consistent with the measured diffusion process. The common interpretation of these streamlines is that they represent the locations of fiber bundles in the white matter. A good introduction to the whole processing pipeline is given in [200]. [213] presents a good overview of methods and [103, 219] put tractography into the wider research perspective.

streamlines

candidate tracts

There exists a variety of models and algorithms to turn DWI data into streamlines. Common to all approaches is the idea that the diffusion information can be turned into a distance or cost function. Locally, streamlines are extended by minimizing the distance or cost.

Algorithms can broadly be classified according to three different features.

The first feature is the modeling domain. Streamlines can either be modeled continuously [153, 133], i.e., as curves $c : \mathbb{R} \to \mathbb{R}^3$ or discretely [157, 100, 115], in which case the output is a path on the weighted graph of neighboring voxels. In principle, this allows to investigate the problem independently of the resolution limitations, but challenges arise when the DWI data needs to be interpolated between voxels [63, 65].

A second design choice is the method of including streamlines. One common approach are tracking methods [13, 164], which define a starting seed region and extend the current streamline until some defined stopping criterion is met. *Shortest-path tractography (SPT)* methods [133, 157, 115] specify two regions-of-interest of which it is known that a connecting tract exist. Of these two choices, tracking methods are more popular at time of writing. Yet, these suffer from two problems:

Shortest-path tractography (SPT)

1. Tracts of tracking methods often terminate in low-connected areas or in areas of crossing fibers because these are typically areas with no clear directional information.

2. Voxels near the starting region are explored more thoroughly than voxels far away. Some parts of the brain can, thus, be under-explored. This not only introduces a bias towards paths close to the starting region, but may also have the effect that the optimal path is never explored.

On the other hand, SPT methods will always return some tract between two input voxels regardless of whether there actually is a tract present in the subject.

A third criterion of separation is the usage of *deterministic* methods (e.g., [13, 142, 37]) on one hand and *randomized* methods (e.g., [15, 164, 196]) on the other hand. Methods from the first category expand the local tract via deterministic rules from the ODF, whereas randomized methods take the probabilistic nature of the diffusion into account.

The fourth and final criterion is the *type of data* that is used in the tractography algorithm. In some cases, this could also be considered as a priori constraints from the imaging device or data preprocessing pipeline of the scientific project the data stems from. But as a general rule, it is usually the case that simpler imaging data types can be constructed from more advanced reconstruction techniques and, therefore, a researcher might opt knowingly for a simpler algorithm.

8.4 Data

For the applications in Sections 9 and 10, data were provided [in part] by the *Human Connectome Project (HCP)*, WU-Minn (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University.

Human Connectome Project (HCP)

Experiments were run on diffusion data of 40 and 20 subjects in Sections 9 and 10, respectively, provided as a subsample from the Q3 release of the HCP [62, 202, 72, 215]. This pre-processed diffusion data contains 270 diffusion directions distributed equally over 3 shells with b-values $b = 1000, 2000, 3000\,\mathrm{s/mm^2}$ [202]. `dtifit` [12] and `FAST` [225] were used to compute DTI tensors and perform segmentation, respectively. Two different tracts have been used in the experiments: the *cortico-spinal tract (CST)* and the *inferior longitudinal fasiculus (ILF)* were obtained from the expert annotated Catani tract atlas [35]. ROI atlases were constructed in "template space" by overlapping the tract atlas with regions from the Harvard-Oxford atlas [52]. For the CST, the ROIs are the overlap with the brainstem, the hippocampus and the amygdala for one region, and the overlap with the superior frontal gyrus, the precentral gyrus and the postcentral

cortico-spinal tract (CST)

inferior longitudinal fasiculus (ILF)

gyrus for the the other region. The ILF ROIs are the two respective overlaps with the temporal pole on one end and the superior occipital cortex, the inferior occipital cortex and the occipital pole on the other end. Warps provided by the HCP were applied to warp the respective ROIs from “template space” to “subject space.”

9 A probabilistic ODE solver for the uncertainty quantification in shortest-paths tractography

We have seen in Section 8 that tractography is an important methodology for physicians and scientists to non-invasively create connectomes of the human brain. However, the output of these tractography algorithms can be misleading since streamlines reconstructions depend on more or less motivated modeling assumptions and algorithmic decisions. In this chapter, we will present a way to use probabilistic ODE solvers to *quantify* and *visualize* this uncertainty.

This chapter is based on the publication Schober, Kasenburg, Feragen, Hennig, and Hauberg [186].

9.1 Introduction

A core difficulty in tractography is the large gap that needs to be bridged from the physical measurements to the algorithm output. In particular, coarse voxel data needs to be transformed into a continuous model to achieve resolutions that are on the same scale as the underlying biological fascicles.

Depending on the sophistication of the applied tractography model, fitting procedures might introduce additional *algorithmic uncertainty*. For instance, wavefront propagation schemes [101] are usually based on initial value problems that have to be solved numerically. Stochastic schemes [196] might suffer from the missing mode problem as discussed in Section 5.2.2.

In order to draw reliable conclusions from tractography algorithms, it is important to communicate the degree of uncertainty associated with the algorithm output. In this work, we present a consistent way to *estimate* and *visualize* the uncertainty introduced by the algorithmic approximation error.

9.1.1 Related work

While some of these methods provide some visualization of uncertainty for fibers, these visualizations typically do not include the *uncertainty in the method*. E.g., wavefront propagation techniques are usually model by IVPs, but usually fixed step sizes are used which adds numerical error in the integration.

Ehricke, Klose, and Grodd [61] color code streamlines either according to a global confidence score or a locally varying confidence score.

The graphical user interface tool PASTA, presented in Jones, Travis, Eden, Pierpaoli, and Basser [110], can display various measures of uncertainty by varying width and color along the the streamline. PASTA can be thought of as an orthogonal work since their main focus is on representation strategies for otherwise obtained measures. These strategies could be, in principle, transfered to our method.

Similarly, Brecheisen, Platel, Haar Romeny, and Vilanova [21] propose a method of visualizing tracts by confidence level sets. If the model evidence of the GP posterior is interpreted as a confidence level, the method of [21] can directly be applied to our model.

Wiens, Schlaffke, Schmidt-Wilcke, and Schultz [224] incorporate the anisotropy of the diffusion tensor as local deformation of the streamtubes. However, no uncertainty in the diffusion tensor data or algorithm uncertainty is tracked.

Details and further references can be found in Brecheisen [20] and Schultz, Vilanova, Brecheisen, and Kindlmann [191].

9.2 Manifold models for continuous tractography

Recall from Section 8.2 that the measured DTI signal corresponds to *apparent* diffusion as the diffusion is anisotropically restricted via the cell membranes. One way to obtain a fiber model is to transform the geometry of the model space such that streamlines correspond to isotropic diffusion in the warped space. Mathematically, this can be modeled with a *Riemannian manifold*. Models of this type have been considered, e.g., in [133, 157, 67].

Riemannian manifold

Riemannian manifolds are manifolds that are endowed with a *Riemannian metric* $\boldsymbol{M}(\boldsymbol{x})$, which is a smoothly changing inner product $\langle \boldsymbol{a}, \boldsymbol{b} \rangle_x = \boldsymbol{a}^\intercal \boldsymbol{M}(\boldsymbol{x}) \boldsymbol{b}$ in the tangent space $T_x\mathcal{M}$ of each point on the manifold $\mathcal{M}$. Typically, the metric $\boldsymbol{M}(\boldsymbol{v})$ in voxel $\boldsymbol{v}$ is defined by $\boldsymbol{M}(\boldsymbol{v}) = \alpha_v \boldsymbol{D}(\boldsymbol{v})^{-1}$, where α_v is a voxel-wise scaling factor. A smoothly-varying inner product is obtained by applying trilinear interpolation to the voxel-wise DTI data similar to Lenglet, Deriche, and Faugeras [133]. The necessity of α_v has been demonstrated by Hao, Whitaker, and Fletcher [85] who also propose a numerical scheme for its computation. Fuster, Tristan-Vega, Haije, Westin, and Florack [67] provides a derivation which demonstrates that $\alpha_v = \det \boldsymbol{D}(\boldsymbol{v})$ can be interpreted as Brownian motion under the thus constructed metric. In this chapter, we will evaluate the choice of α_v empirically.

On a Riemannian manifold, the length of a curve $\boldsymbol{y}(t) : [0,1] \to \mathcal{M}$ is defined by integrating the induced infinitesimal norm

$$\begin{aligned}\text{Length}(\boldsymbol{y}(t)) &= \int_0^1 \left|\left|\boldsymbol{y}'(t)\right|\right|_{M(\boldsymbol{y}(t))} \, \mathrm{d}t \\ &= \int_0^1 \sqrt{\boldsymbol{y}'(t)^\intercal \boldsymbol{M}(\boldsymbol{y}(t))\boldsymbol{y}'(t)} dt.\end{aligned}$$

Let $\mathcal{C}_{a,b} = \{\boldsymbol{y} : [0,1] \to \mathcal{M} \,|\, \boldsymbol{y}(0) = \boldsymbol{a}, \boldsymbol{y}(1) = \boldsymbol{b}\}$ denote the set of piecewise smooth curves connecting points $\boldsymbol{a}$ and $\boldsymbol{b}$, then the distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined to be

$$\text{Dist}(\boldsymbol{a}, \boldsymbol{b}) = \min_{\boldsymbol{y} \in \mathcal{C}_{a,b}} \text{Length}(\boldsymbol{y}(t)). \tag{71}$$

The curve $\hat{\boldsymbol{y}}(t)$ that minimizes (71) is called the *geodesic* between $\boldsymbol{a}$ and $\boldsymbol{b}$. Geodesics satisfy the following ODE [86, 168]:

$$(\boldsymbol{y}''(t))_d = f_d(t, \boldsymbol{y}, \boldsymbol{y}') = -\boldsymbol{\Gamma}_d^\intercal \cdot \left(\boldsymbol{y}'(t) \otimes \boldsymbol{y}'(t)\right), \quad d = 1, \ldots, D = 3, \tag{72a}$$

$$\boldsymbol{\Gamma}_d = \frac{1}{2} \sum_{k=1}^{D} \left[\boldsymbol{M}_{\boldsymbol{y}(t)}^{-1}\right]_{d,k} \left(\frac{\partial \operatorname{vec} \boldsymbol{M}_{\boldsymbol{p}}}{\partial p_k}\right)_{\boldsymbol{p}=\boldsymbol{y}(t)} \in \mathbb{R}^{D^2 \times 1}, \tag{72b}$$

where $\otimes$ denotes the Kronecker product and vec stacks the columns of a matrix. The geodesic between two points $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{M}$ is given by the solution to the BVP:

$$\boldsymbol{y}(0) = \boldsymbol{a}, \quad \boldsymbol{y}(1) = \boldsymbol{b}, \quad \boldsymbol{y}''(t) = f_d(t, \boldsymbol{y}, \boldsymbol{y}').$$

For our work, we have considered an shortest-path algorithm as these algorithms do not suffer from path-length dependency [138]. However, the approach presented in this chapter can be readily applied to wavefront propagation schemes as well.

9.3 Methods

To quantify the model uncertainty, we apply the probabilistic solver proposed by Hennig and Hauberg [92] to the geodesic BVP (72). It was empirically found that convergence was harder to obtain in this setting which required a couple of modifications, specifically in the GP model selection.

In this setting, the prior function $\boldsymbol{\mu}(t)$ needs to be initialized much closer to the true solution $\boldsymbol{y}(t)$. Otherwise, the model will too often be evaluated in known areas of gray matter where the data is meaningless. This was circumvented by running Dijkstra's algorithm [57] on the discrete model $\boldsymbol{v}_0 = \boldsymbol{a}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_N = \boldsymbol{b}$ and fitting a GP regression model to the data $\{(t_n = nN^{-1}, \boldsymbol{v}_n) \,|\, n = 0, \ldots, N\}$.

Furthermore, the hyper-parameter selection strategy of Hennig and Hauberg [92] did not prove to be adequate. Line 16 of Algorithm 1 was replaced by a marginal empirical moment matching method for both the output covariance V and the kernel length scale λ.

9.4 Experiments

In a first experiment, we test the validity of the Riemannian model by comparing the results from the mathematical model with the expert annotated Catani atlas [35].

We subsample 250 pairs of points at random from the ROIs of the CST for all 40 subjects and compute the discrete and continuous shortest paths. For each continuous path, we mark all voxels it passes through. We measure *accuracy* as the percentage of voxels that have been classified as belonging to the CST tract by at least one expert. The results can be found in Figure 25.

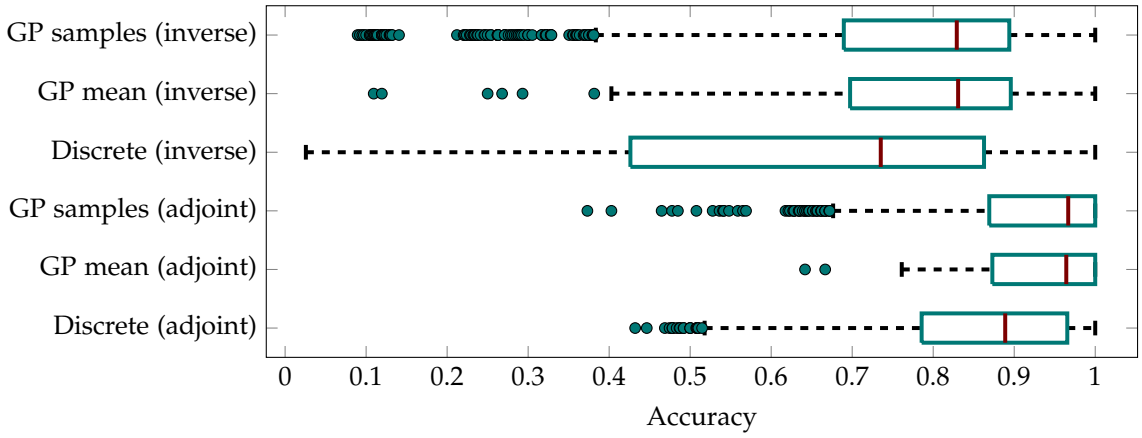

Figure 25: Box plot displaying percentage of voxels that are considered to be part of the CST by at least one expert in the Catani atlas.

Figure 26 shows example paths from both algorithms and metrics computed on a single subject. To visualize the uncertainty associated with each solution, we have created a video displaying animated samples from the GP posterior using a technique described in [90] which is included in the electronic supplements of this thesis. A modified version of this video presented at the MICCAI conference can be found online at URL `https://www.youtube.com/watch?v=VrhulgVaRMg`.

In a second application of the solution uncertainty, we marginalize over the posterior using Monte Carlo quadrature to compute density heat maps. Figure 27 shows a 2D heat map slice of the 3D density. The slice has been chosen to contain maximal variance which has been determined by principal component analysis.

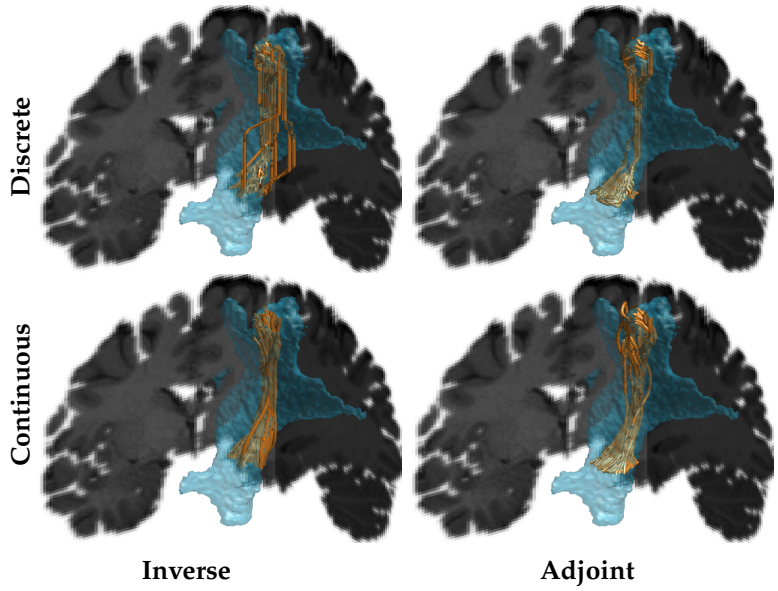

Figure 26: Geodesics in the CST under the inverse and adjoint metric computed in discrete and continuous space. Blue area shows voxels in the CST according to at least one expert in the Catani atlas. From top left clockwise: discrete voxels with inverse metric, discrete voxels with adjoint metric, continuous solution under adjoint metric, continuous solution under inverse metric.

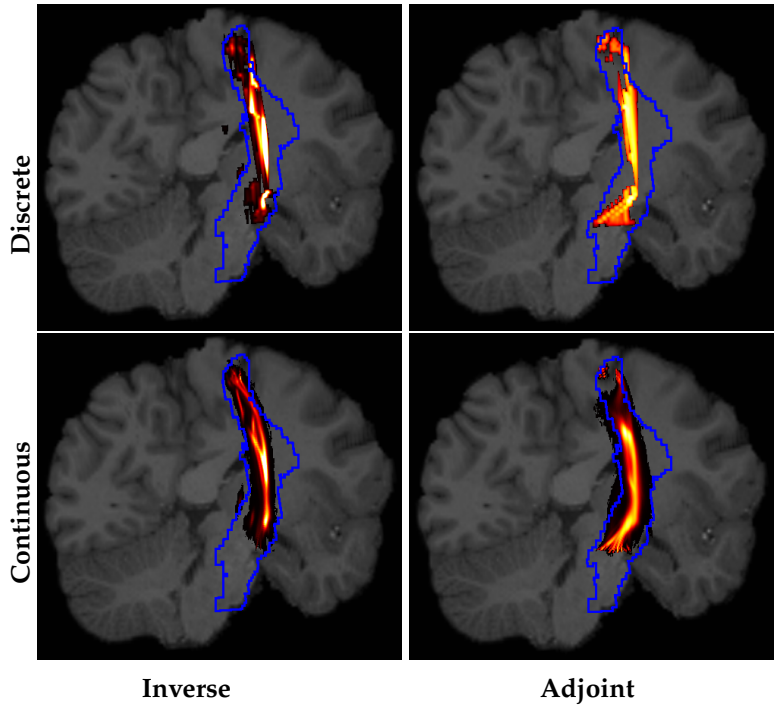

Figure 27: 2D heat map slice displaying candidate tract densities. Blue outline delineates the CST as defined above. From top left clockwise: discrete voxels with inverse metric, discrete voxels with adjoint metric, continuous solution under adjoint metric, continuous solution under inverse metric.

9.5 Discussion

The experiments show that the GP ODE framework can be used to capture uncertainty inherent in the shortest path computation. In both clinical and scientific applications, it is important to communicate this methodological uncertainty. Probability theory can be used to communicate the uncertainty, visualize results accordingly as well as incorporate it in down-stream computations, e.g., in the generation of path densities heat maps.

Compared to discrete methods, these heat maps display an overall larger domain of support with sharper peaks. This can partially be explained by the higher resolution of the continuous paths. Alternatively, this could also indicate that the graph-based paths disproportionately favor subsequences of low-cost compared to the overall model.

Figure 26 shows the classic "spaghetti plot" used for visualizing tractography results. The discrete solutions show tendencies to straight line segments connected by rather sharp turns which cannot solely be explained by the discretization error. In general, the continuous solutions bend less drastically, as we would expect from actual fibers. The figure only shows the mean function of the GP estimates of the geodesics; the supplements contain an animation of the uncertainty. This provides a visualization of the solutions which makes it very clear that individual paths cannot, and *should not*, be interpreted as individual fibers in the brain—a common misinterpretation of "spaghetti plots".

We introduce a quality measure to compare different methods. First, we compare the standard inverse metric [157, 133] to the recently suggested adjoint metric of Fuster, Tristan-Vega, Haije, Westin, and Florack [67]; empirical results show that the theoretically strong adjoint metric is consistently better. Secondly, we find that the GP posteriors agree with experts slightly more often than the discrete solutions.

Samples from the posterior score lower than the mean prediction which is expected as the posterior mean is modeled to be the best approximation. Interestingly, these samples score higher on average than the discrete solutions which suggests that even sub-par solutions of the continuous model agree more than the optimal solution of the discrete model—at least according to our metric.

A general disadvantage of shortest path tractography is that there will always be *some* path connecting any two points, whether it is anatomically there or not. This can be alleviated to some extent by discarding improbable paths. Future work could try to identify a cut-off criterion using the model evidence of the GP posterior. Alter-

natively, the probabilistic IVP solver can be applied to the continuous model for wavefront propagation methods.

Work by Wassermann, Rathi, Bouix, Kubicki, Kikinis, Shenton, and Westin [220] illustrates that GPs form a particularly useful representation of shortest paths for population studies of brain connectivity. Our GP solution to the tractography problem lends itself particularly well to this type of population analysis as it avoids the post-hoc GP fitting used by Wassermann et al. This could be further investigated in future work.

10

Propagating model uncertainty in shortest-path tractography using a probabilistic BVP solver

Probabilistic numerics promises to capture uncertainty across a computational pipeline [41]. In this section, we empirically test a method that can capture both data and algorithmic uncertainty in a tractography model.

This chapter is based on the publication Hauberg, Schober, Liptrot, Hennig, and Feragen [87]. An explanatory video for this publication was created by my co-authors which can be found online at URL https://www.youtube.com/watch?v=xQwoT92B0YU.

10.1 Introduction

The biggest source of uncertainty in tractography remains the noisy measurement process and the associated data uncertainty. While there are established methods to quantify the uncertainty on the model of interest [109, 196, 223], algorithms are either limited to Monte Carlo integration schemes [105] or fit streamlines to the best guess and only visualize the uncertainty post fitting [190].

Using a probabilistic numerical method, model approximation error as well as algorithmic approximation error can be expressed in a common inference scheme. As in Chapter 9, this can be used to visualize the model uncertainty to users. More importantly, this allows to separate the influence of data and algorithmic uncertainty, giving practioners feedback as to where more computational resources might still lead to more accurate models.

10.2 A model for random Riemannian manifolds

Existing SPT algorithms regard the diffusion tensor $\boldsymbol{D}_v$, and hence the corresponding metric, as a deterministic object faithfully representing the data. However, the diffusion tensor $\boldsymbol{D}_v$ at voxel v is estimated from diffusion data. Like any estimation process, the resulting diffusion tensor is a *stochastic* variable, not an exact representative of the true underlying diffusion process [107]. If we interpret the adjugate diffusion tensor field as a Riemannian metric, then this metric should also be considered stochastic. This, however, complicates the geometric interpretation as the resulting object is now a *random Riemannian metric* [125] rather than a deterministic metric.

We approximately solve the geodesic ODE (72) using a probabilistic description of the uncertainty over $f(t, \boldsymbol{y}, \boldsymbol{y}')$, the stochastic variable arising from Eq. 72 with a stochastic metric. We approximate f with a Gaussian distribution (using the shorthand notation $f_{t,\boldsymbol{y},\boldsymbol{y}'} \equiv f(t, \boldsymbol{y}, \boldsymbol{y}')$)

$$P(f_{t,\boldsymbol{y},\boldsymbol{y}'}) = \mathcal{N}\left(f_{t,\boldsymbol{y},\boldsymbol{y}'}; m_{t,\boldsymbol{y},\boldsymbol{y}'}, \boldsymbol{C}_{t,\boldsymbol{y},\boldsymbol{y}'}\right). \tag{73}$$

The mean and covariance that form this approximation are computed from

$$m_d = -\mathbb{E}(\boldsymbol{\Gamma}_d)^\intercal \left(\boldsymbol{y}' \otimes \boldsymbol{y}'\right)$$

and

$$C_d^2 = \left(\boldsymbol{y}' \otimes \boldsymbol{y}'\right)^\intercal \operatorname{cov}(\boldsymbol{\Gamma}_d) \left(\boldsymbol{y}' \otimes \boldsymbol{y}'\right),$$

where $\boldsymbol{m}_{t,\boldsymbol{y},\boldsymbol{y}'} = (m_1, m_2, m_3)^\intercal$ and $\boldsymbol{C}_{t,\boldsymbol{y},\boldsymbol{y}'} = \operatorname{diag}\left(C_1^2, C_2^2, C_3^2\right)$.

We estimate the mean and covariance of $\boldsymbol{\Gamma}_d$ empirically, subsampling the gradient directions used to generate the local diffusion tensors into $S = 20$ batches each containing 80% of the directions. For each subsample we fit a diffusion tensor field and compute $\boldsymbol{\Gamma}_d$ (72b). Finally, we estimate the moments $\mathbb{E}(\boldsymbol{\Gamma}_d) \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{\Gamma}_d^{(s)}$ and $\operatorname{cov}(\boldsymbol{\Gamma}_d) \approx \frac{1}{S-1}\sum_{s=1}^{S}\left(\boldsymbol{\Gamma}_d^{(s)} - \mathbb{E}(\boldsymbol{\Gamma}_d)\right)\left(\boldsymbol{\Gamma}_d^{(s)} - \mathbb{E}(\boldsymbol{\Gamma}_d)\right)^\intercal$.

10.3 Solving uncertain boundary value problems

When the metric is *uncertain*, the geodesic ODE can only be evaluated probabilistically as $P(f)$. To handle this situation, we extend the probabilistic ODE solver to cope with Gaussian uncertain observations.

In the random geometry setting of Eq. 73, the curvature at the extrapolation point, $\boldsymbol{y}_i = f(t_i, \tilde{\boldsymbol{y}}(t_i), \tilde{\boldsymbol{y}}'(t_i))$, cannot be observed directly. Instead, the numerically accessible variables are the parameters $\boldsymbol{m}, \boldsymbol{C}$ of a Gaussian distribution over function values $P(\boldsymbol{y}_i \mid \boldsymbol{m}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}, \boldsymbol{C}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}) = \mathcal{N}(\boldsymbol{y}_i; \boldsymbol{m}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}, \boldsymbol{C}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'})$. As the normal distribution is symmetric around the mean, this is a likelihood for $\boldsymbol{y}_i$,

$$P(\boldsymbol{m}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'} \mid \boldsymbol{y}_i, \boldsymbol{C}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}) = \mathcal{N}(\boldsymbol{m}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}; \boldsymbol{y}_i, \boldsymbol{C}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}). \tag{74}$$

Since both likelihoods (46, 74) are Gaussian, the latent $\boldsymbol{y}_i$ can be marginalized analytically, giving the complete observation likelihood for $\boldsymbol{m}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}$

$$P(\boldsymbol{m}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'} \mid \boldsymbol{y}''(t_i)) = \mathcal{N}\left(\boldsymbol{m}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}; \boldsymbol{y}_i'', \boldsymbol{C}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'} + \boldsymbol{\Lambda}_i\right). \tag{75}$$

Solutions to noisy ODEs are then inferred by replacing Eq. (46) with Eq. (75), using $\boldsymbol{m}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'} = \mathbb{E}[f_{t,\boldsymbol{y},\boldsymbol{y}'}]$ in place of the (inaccessible) function evaluations $\boldsymbol{y}_i = f_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}$, such that the approximation error is modeled by the additive uncertainty $\boldsymbol{C}_{t,\tilde{\boldsymbol{y}},\tilde{\boldsymbol{y}}'}$.

The ODE solver relies on two noise terms: $\boldsymbol{\eta}_i \sim \mathcal{N}(0, \boldsymbol{\Lambda}_i)$ captures the numerical error, whereas $\boldsymbol{C}$ in Eq. 74 describes the uncertainty arising from the data. While the terms are structurally similar, they capture different error sources; e.g., changes in the number of grid points imply a change in $\boldsymbol{\eta}$, while $\boldsymbol{C}$ is unaffected. Both noise terms are approximated as Gaussian to ensure efficient inference.

Hyper-parameter adaptation in this work was done by type-II maximum likelihood optimization for $\boldsymbol{V}, \lambda$ on the positive definite cone using the open source library `Manopt` (Boumal, Mishra, Absil, and Sepulchre [18]).

10.4 Experiments

As a first illustration, Figure 28 shows the density of a *single* geodesic within the CST projected onto a slice. The center column shows that the geodesic density roughly consists of two certain vertical line segments with an uncertain connection between them (green box). The bottom row shows the standard deviation of $\boldsymbol{\Gamma}_d$. The geodesic uncertainty appears related to data noise. This is not attained with other GP ODE solvers [186] as they model constant observation noise.

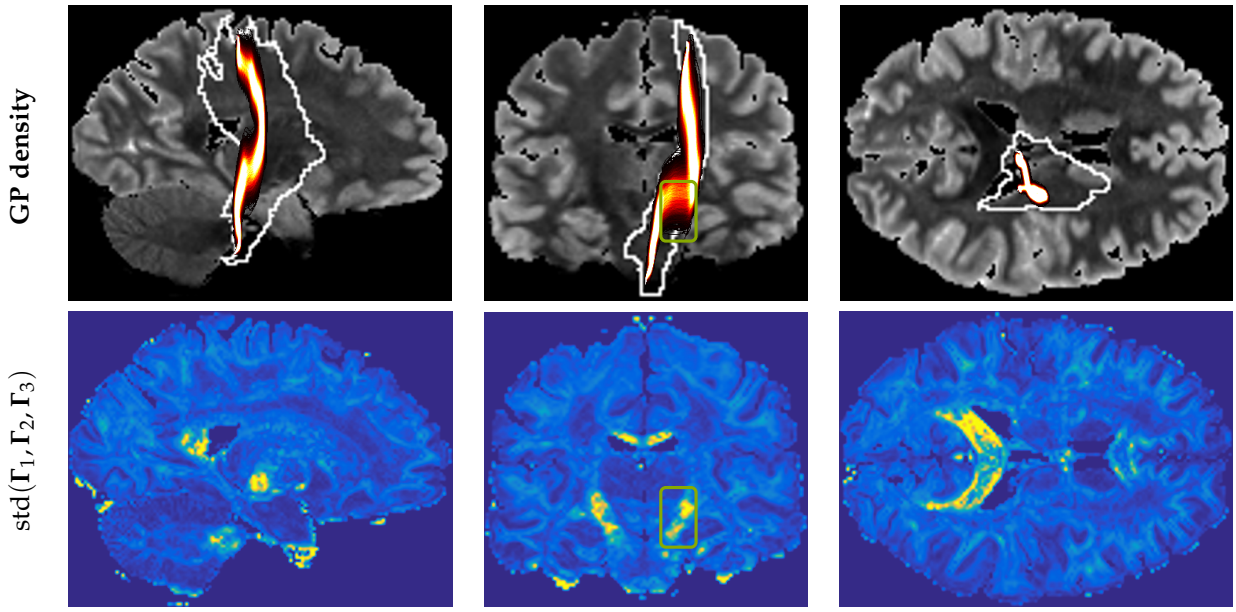

Figure 28: Shortest paths in the CST and ILF.

Next we sample 250 endpoint pairs and compute the corresponding geodesics. Figure 29 shows the resulting heatmaps. The GP

solution provides a more coherent picture of the tract compared to the picture generated with Dijkstra's algorithm.

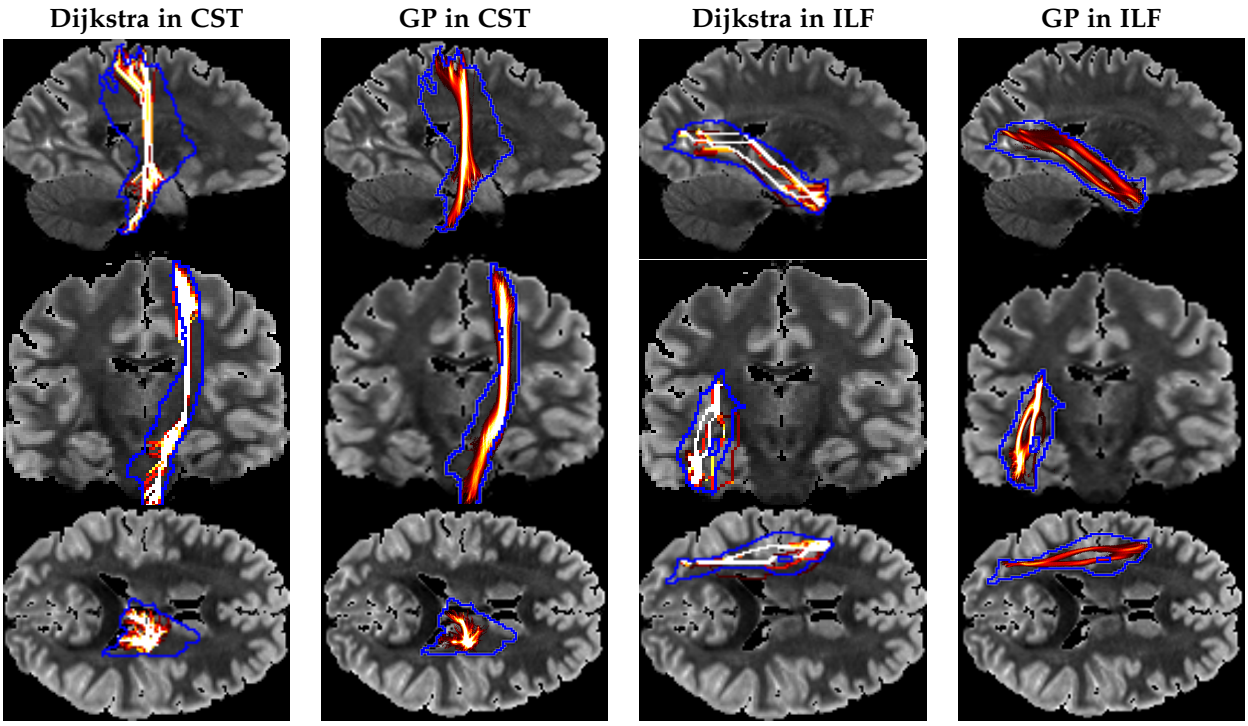

Figure 29: Example shortest paths in the CST and ILF using both Dijkstra's algorithm on a deterministic metric, and a GP solver with a random Riemannian metric.

Solution qualities are computed in *accuracy* as defined in Section 9.4. Figure 30 shows the results for 20 subjects. In ILF the median accuracy is comparable for Dijkstra and GP paths, but the GP error bars are significantly smaller. For the CST, we observe similar quality results for Dijkstra and GP paths. This is expected, as the ILF is generally considered the harder tract to estimate.

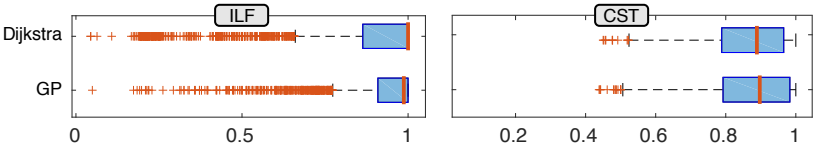

Figure 30: Agreement with at least one expert in the Catani atlas as estimated by Dijkstra's algorithm and the GP solver.

10.5 Discussion

SPT methods are advantageous for tracts that connect two brain regions, e.g. for structural brain networks. However, a long-standing problem is that SPT only finds a single connection without insight to its uncertainty. This is in contrast to walk-based methods, which

are often probabilistic in nature. We provide a fully probabilistic SPT algorithm that models stochastic diffusion tensors and returns a distribution over the shortest path. While uncertainty propagates in probabilistic walk-based methods, the uncertainty of probabilistic SPT only depends on local data uncertainty, not the seed point distance.

Through experiments we visualize the estimated geodesic densities between brain regions, and validate that the estimated geodesic densities are less certain in areas where the estimated diffusion tensor is uncertain. We see that the visualized geodesic densities from the probabilistic SPT yield smoother and more coherent tract than the corresponding Dijkstra solutions.

While Riemannian SPT only applies to second order tensor models, a Finsler geometry framework has emerged enabling continuous SPT with higher order tensors for HARDI data [9]. Our proposed numerical tools also extend to these models, and can provide a probabilistic interpretation of Finsler models. Future work includes shape analysis on the resulting estimated geodesic densities, which are suitable for GP-based tract shape analysis [220].

This work has extended our previous work from Chapter 9 to also include *data uncertainty*. As we have already pointed out, there are several intermediate computational steps from data to the Riemannian model. Feragen and Fuster [63] have recently pointed out that the interpolation algorithm to create the Riemannian manifold also plays a crucial role. Another natural extension would be to model the tensor interpolation using a probabilistic description, for example, with a generalized Wishart process [31].

Part V

Epilogue

11 Discussion

In this last chapter of the thesis, we will reexamine and reflect upon our contribution as well as point towards potential future directions.

11.1 Summary and contribution

The aim of this thesis was to extend probabilistic modeling approaches to the numerical solution of ordinary differential equations. Our contributions can broadly be classified into two categories.

In search of fast and reliable algorithms, we have explored connections between Gaussian inference and linear solvers for initial value problems. We have proposed a specific Gauss-Markov filtering algorithm and have shown links to the theory of both Runge-Kutta and Nordsieck methods. As a consequence, we have established the first probabilistic numerical method for initial value problems of higher-order convergence in the posterior mean with step size adaptation. We have demonstrated its applicability on a standard benchmark problem set and provided an open source reference implementation.

In a parallel investigation, we have demonstrated how the probabilistic model may provide novel functionality to practitioners. To this end, we have applied a Gaussian process boundary value problem solver to the problem of tractography in neuro-imaging. Our experiments have shown how uncertainties in the numerical computation can be both propagated through the modeling pipeline as well as visualized to the domain expert.

11.2 Critical reflection

On a higher level, the work presented in this thesis has demonstrated how the common language of probability theory can help to express the many aspects of a numerical pipeline holistically. In Section 5.1, we have identified two different application scenarios for probabilistic numerical methods, and have derived desiderata which can be used to guide the search of suitable numerical methods. In contrast, the analysis of classical algorithms considers problems individually and independently. Sometimes, algorithms are based on other numerical algorithms, e.g., when implicit methods require a non-linear equation solver. While this allows to match algorithms freely to solve a problem stack, it is much more difficult to consider the combined

behavior of said stack. Where we have developed probabilistic numerical ODE solvers so far, the interaction with different methods almost occurs naturally.

On the search for probabilistic ODE solvers, we have pursued to find equivalences similar to [117, 158, 91]. This approach has the advantage that methods are immediately familiar to a wide audience and one can concentrate on expanding its applications as desired. However, we have also seen in Section 6.6 that not all classical methods can be recovered this way. At their roots, numerical methods are algebraic expressions that try to optimize certain analytic properties. These algebraic expressions may be manipulated in any legal way to obtain the desired result. A probabilistic numerical method is by its very definition constrained to those subsets of algebraic and analytic manipulations that carry a probabilistic interpretation. This has two important consequences.

First, this will severely limit the search for viable algorithms for more restrictive problem classes. In the case of ordinary differential equations, the most urgent next steps are probabilistic solvers for stiff equations and differential algebraic equations. Many practically relevant problems belong to one of these two classes. This is evidenced by a more recent benchmark problem set [147]. At the time of writing, it is an open question whether there exist efficient probabilistic numerical algorithms for these problems.

But even within the domain of applicable problems, our state of knowledge is lagging behind best practices. Many modern solvers additionally provide order selection per step. In our model, this would correspond to modeling the solution by a different stochastic process per step. This, however, would directly violate our desiderata of a globally defined model.

Thus, another important open problem is to formulate criteria under what circumstances which desiderata are beneficial or maybe even potentially harmful. This may be equivalently stated as a *taxonomy of applications* for probabilistic numerical methods. Importantly, this includes the understanding when *not to* apply a probabilistic numerical method altogether. While the study of probabilistic numerical algorithms will deepen our understanding of numerical methods regardless, it has to be critically reflected when the probabilistic interpretation brings additional benefit to the user. In optimization, this has been clearly demonstrated in the novel *probabilistic line search* algorithm by Mahsereci and Hennig [144], and we argue that our contributions (see Chapters 9 and 10) in tractography offer added value. The resonance has not been conclusive so far.

As a second consequence, this suggests that future research should look into more variants of probabilistic numerical methods for ordinary differential equations. Current methods under investigation are

either entirely sampling based or entirely based on Gaussian approximations. These are two ends of a spectrum of approximate inference method which is largely unexplored. The machine learning literature has a large body of approximate inference techniques that could be considered as building blocks in probabilistic numerics. Understanding the theory of approximate inference for numerical solvers might lead to novel applications for probabilistic methods, to algorithms that might be able to scale even in larger chains of numerical methods, and it might even develop new analysis tools for approximate inference itself.

11.3 Conclusion

While we have not been the first to propose probabilistic numerical algorithms for ordinary differential equations, this idea has gained considerable momentum as evidenced by the recent wave of publications on the matter. The first cornerstones have been set, but only sustained progress on its modeling benefits will establish probabilistic numerics as a standard tool in science and engineering.

Part VI

Appendix

Source code listings

This chapter contains the source of the symbolic algebra implemention of Section 6.6 in SymPy. The runtime of the entire program takes about 2 days on my machine.

```
## Setup, Configuration
import sympy as sp
sp.init_printing(use_unicode=True)

## Helper functions
# creates a 1x1 matrix with element sym
def mat(sym):
  return sp.Matrix([[sym]])

# creates e_i in R^(q+1)
def basisVector(i,q):
  return sp.Matrix(q+1, 1,
                   lambda m,n: 1 if m == i else 0)

# applies limit elementwise
def limitElements(A, t, lim):
  return sp.Matrix(A.rows, A.cols,
                   lambda i,j:
                   sp.limit(A[i,j], t, lim))

# returns the finite differences of a list
def diff(l):
  res = [0]
  for i in range(1, len(l)):
    res.append(l[i] - l[i-1])
  return res

# returns the culumative sum of a list
def cumsum(l):
  sum = 0
  res = []
  for e in l:
    sum = sum + e
    res.append(sum)
  return res

# displays a variable during execution
def show(var):
  sp.pprint(var)
  print

## State space functions
```

```
h, sig = sp.symbols('h sigma')

# creates the matrix [F, L*L.T; 0, -F.T], needed for
# the discretization
def Phi(F,L):
  return F.row_join(L*L.T).col_join(
           (0 * F).row_join(-F.T))

# creates the discrete A(h)
def discreteDrift(F, h = h):
  return sp.exp(F*h)

# creates the discrete Q(h)
def discreteDiffusion(F, L, h = h, ssq = sig**2):
  q = F.rows - 1
  P = sp.exp(Phi(F,L)*h)
  A = P[:q+1,:q+1]
  Q = ssq * P[:q+1,q+1:] * A.T
  return Q

# creates the pair (A(h), Q(h)) (convenience wrapper)
def discreteTransfer(F, L, h = h, ssq = sig**2):
  A = discreteDrift(F, h)
  Q = discreteDiffusion(F, L, h, ssq)
  return (A, Q)

# executes one Kalman predict/update step
def kalmanStep(N, F, L, deltaT, ssq, H, z):
  (A, Q) = discreteTransfer(F, L, deltaT, ssq)
  Cp = A * N[1] * A.T + Q
  K  = Cp * H.T * (H * Cp * H.T)**-1

  S = sp.eye(F.rows) - K * H
  m = S * A * N[0] + K * z
  C = S * Cp

  return (m, C)

## State space / GP equivalence computations
# creates the big A matrix to construct
# the GP representation, page 20
def gpA(F, deltaTs):
  q = F.rows - 1
  N = len(deltaTs)

  A = sp.zeros((N+1)*(q+1),0)
  for i in range(N):
    colA = sp.diag([sp.zeros(i*(q+1),q+1),
                    sp.eye(q+1)]
                 + [discreteDrift(F, sp.simplify(e))
                    for e in cumsum(deltaTs[i:])])
    A = A.row_join(colA)
```

```
  A = A.row_join(sp.diag([sp.zeros(N*(q+1),q+1),
                          sp.eye(q+1)]))
  return A

# creates the big Q matrix to construct
# the GP representation, page 20
def gpQ(F, L, P0, deltaTs, ssq = sig**2):
  Qs = [discreteDiffusion(F,L,h,ssq)
        for h in deltaTs]
  return sp.diag(*([P0] + Qs))

## Definitions

# algorithm parameters
a, u, v = sp.symbols('alpha u v')
# problem constants
t0, y0 = sp.symbols('t_0 y_0')

tau = sp.symbols('tau')

z = sp.Matrix(1, 10, lambda i,j:
              sp.symbols('z_%d' % j))

# creates the feedback matrix F for the IWP(q)
def feedbackMatrix(q):
  return sp.Matrix(q+1, q+1,
                   lambda i,j: 1 if j == i+1 else 0)

# creates the diffusion matrix L for the IWP(q)
def diffusionMatrix(q):
  return sp.Matrix(q+1, 1,
                   lambda i,j: 1 if i == q else 0)



## equivalence of IWP(2) and second-order RK methods
print('q=2:')
q  = 2

# basic state-space definitions
B  = sp.eye(q+1)
F  = B * feedbackMatrix(q) * B.inv()
L  = B * diffusionMatrix(q)
H0 = basisVector(0,q).T * B.inv()
H1 = basisVector(1,q).T * B.inv()

# Runge-Kutta connection
# C_{t-1}^-
Cmtm1 = discreteDiffusion(F, L, tau)
H01   = H0.col_join(H1)
```

```
# Uncomment this block for Eqs. (102) and (103)
M3q2   = (H01 * Cmtm1 * H01.T)**-1
M2q2   = Cmtm1 * H01.T
M1q2   = basisVector(0,q).T * discreteDrift(F, h*u)
M2M3q2 = M2q2 * M3q2
print('alpha_{t_0,1}:')
sp.pprint(M2M3q2)
print('limit tau -> oo:')
sp.pprint(limitElements(M2M3q2, tau, sp.oo))
print('limit tau -> oo for M1*M2*M3:')
sp.pprint(limitElements(M1q2 * M2M3q2, tau, sp.oo))

H011 = sp.diag(H01, H1)

Qhu  = discreteDiffusion(F, L, h*u)

bigQt = sp.diag(Cmtm1, Qhu)
bigAt = gpA(F, [h*u])

# Uncomment this block for Eqs. (104) and (105)
M1q2 = basisVector(0,q).T * discreteDrift(F,h).row_join(discreteDrift
    (F,h*(1-u)))
M2q2 = bigQt * bigAt.T * H011.T
M3q2 = sp.simplify((H011 * bigAt * bigQt * bigAt.T * H011.T)**-1)
print('M1:')
sp.pprint(M1q2)
print('M2:')
sp.pprint(M2q2)
print('M3:')
sp.pprint(M3q2)
print('M1*M2*M3, limit tau -> oo:')
sp.pprint(limitElements(M1q2 * M2q2 * M3q2, tau, sp.oo))
limQtq2 = sp.simplify(bigQt - M2q2 * M3q2 * M2q2.T)
print('_Q_(t):')
sp.pprint(limitElements(limQtq2, tau, sp.oo))


# Runge-Kutta initialization

# define grid points
hs = [0, h*u]

# initialization
Nm1 = (sp.zeros(q+1,1), Cmtm1)

Ns = [Nm1]
N0 = kalmanStep(Nm1, F, L, 0, sig**2, H0, mat(y0))
Ns.append(N0)

deltaTs = diff(hs)

for (hi,zi) in zip(deltaTs, z):
```

```
  print(hi)
  Np1 = kalmanStep(N0, F, L, hi, sig**2, H1, mat(zi))
  Ns.append(Np1)
  N0  = Np1

Ah1mu = discreteDrift(F,h*(1-u))
Qh1mu = discreteDiffusion(F, L, h*(1-u))

mt1q2 = sp.simplify(Ah1mu * limitElements(sp.simplify(N0[0]), tau, sp
    .oo))
sp.pprint(mt1q2)
Qt1q2 = sp.simplify(Ah1mu * limitElements(sp.simplify(N0[1]), tau, sp
    .oo) * Ah1mu.T + Qh1mu)
sp.pprint(Qt1q2)




## equivalence of IWP(2) and second-order RK methods
print('q=3:')
q  = 3

# basic state-space definitions
B  = sp.eye(q+1)
F  = B * feedbackMatrix(q) * B.inv()
L  = B * diffusionMatrix(q)
H0 = basisVector(0,q).T * B.inv()
H1 = basisVector(1,q).T * B.inv()

# Runge-Kutta connection
# C_{t-1}^-
Cmtm1 = discreteDiffusion(F, L, tau)
H01   = H0.col_join(H1)

# Uncomment this block for Eqs. (102) and (103)
M3q3   = (H01 * Cmtm1 * H01.T)**-1
M2q3   = Cmtm1 * H01.T
M1q3   = basisVector(0,q).T * discreteDrift(F, h*u)
M2M2q3 = M2q3 * M3q3
print('Eq. (103):')
sp.pprint(M2M2q3)
print('limit tau -> oo:')
sp.pprint(limitElements(M2M2q3, tau, sp.oo))
print('limit tau -> oo for I*II*III:')
sp.pprint(limitElements(M1q3 * M2M2q3, tau, sp.oo))

twothirds = sp.Rational(2,3)

H011 = sp.diag(H01, H1)

Qhu  = discreteDiffusion(F, L, h*u)
```

```
bigQt = sp.diag(Cmtm1, Qhu)
bigAt = gpA(F, [h*u])

# Uncomment this block for Eqs. (104) and (105)
M1q3   = basisVector(0,q).T * discreteDrift(F,h*v).row_join(
    discreteDrift(F,h*(v-u)))
M2q3  = bigQt * bigAt.T * H011.T
M3q3 = sp.simplify((H011 * bigAt * bigQt * bigAt.T * H011.T)**-1)
print('Eq. (104), I:')
sp.pprint(M1q3)
print('Eq. (104), II:')
sp.pprint(M2q3)
print('Eq. (104), III:')
sp.pprint(M3q3)
print('I * II * III, limit tau -> oo:')
sp.pprint(limitElements(M1q3 * M2q3 * M3q3, tau, sp.oo))

H0111 = sp.diag(H01, H1, H1)

Qhv = discreteDiffusion(F, L, h*(v-u))

bigQt = sp.diag(Cmtm1, Qhu, Qhv)
bigAt = gpA(F, [h*u, h*(v-u)])

# Uncomment this block for _Q_(t)
M1q3   = basisVector(0,q).T * discreteDrift(F,h).row_join(
    discreteDrift(F,h*(1-u))).row_join(discreteDrift(F,h*(1-v)))
M2q3  = bigQt * bigAt.T * H0111.T
M3q3 = sp.simplify((H0111 * bigAt * bigQt * bigAt.T * H0111.T)**-1)
print('Eq. (104), I:')
sp.pprint(M1q3)
print('Eq. (104), II:')
sp.pprint(M2q3)
print('Eq. (104), III:')
sp.pprint(M3q3)
print('I * II * III, limit tau -> oo:')
M2M3q3   = sp.simplify(M2q3 * M3q3)
M1M2M3q3 = sp.simplify(M1q3 * limitElements(M2M3q3, tau, sp.oo))
sp.pprint(M1M2M3q3)
print('_Q_(t):')
limQtq3 = limitElements(sp.simplify(bigQt - M2M3q3 * M2q3.T), tau, sp
    .oo)
sp.pprint(limQtq3)


# Runge-Kutta initialization

# define grid points
hs = [0, h*u, h*v]

# initialization
Nm1 = (sp.zeros(q+1,1), Cmtm1)
```

```
Ns = [Nm1]
N0 = kalmanStep(Nm1, F, L, 0, sig**2, H0, mat(y0))
Ns.append(N0)

deltaTs = diff(hs)

for (hi,zi) in zip(deltaTs, z):
  print(hi)
  Np1 = kalmanStep(N0, F, L, hi, sig**2, H1, mat(zi))
  Ns.append(Np1)
  N0  = Np1
  N0  = (sp.simplify(Np1[0]), sp.simplify(Np1[1]))

# N0  = (sp.simplify(Np1[0]), sp.simplify(Np1[1]))
# N0 = (sp.expand(Np1[0]), sp.expand(Np1[1]))
Ah1mu = discreteDrift(F,h*(1-v))
Qh1mu = discreteDiffusion(F, L, h*(1-v))

mt1q3 = sp.simplify(Ah1mu * limitElements(N0[0], tau, sp.oo))
sp.pprint(mt1q3)
Qt1q3 = sp.simplify(Ah1mu * limitElements(N0[1], tau, sp.oo) * Ah1mu.
    T + Qh1mu)
sp.pprint(Qt1q3)

## equivalence of IWP(2) and second-order RK methods
print('q=4:')
q  = 4

# basic state-space definitions
B  = sp.eye(q+1)
F  = B * feedbackMatrix(q) * B.inv()
L  = B * diffusionMatrix(q)
H0 = basisVector(0,q).T * B.inv()
H1 = basisVector(1,q).T * B.inv()

Cmtm1 = discreteDiffusion(F, L, tau)

# Runge-Kutta initialization

# define grid points
hs = [0, h*u, h*v, h]

# initialization
Nm1 = (sp.zeros(q+1,1), Cmtm1)

Ns = [Nm1]
N0 = kalmanStep(Nm1, F, L, 0, sig**2, H0, mat(y0))
Ns.append(N0)

deltaTs = diff(hs)
```

```
for (hi,zi) in zip(deltaTs, z):
  print(hi)
  Np1 = kalmanStep(N0, F, L, hi, sig**2, H1, mat(zi))
  Ns.append(Np1)
  N0 = Np1
  N0  = (sp.simplify(Np1[0]), sp.simplify(Np1[1]))

N0  = (sp.simplify(Np1[0]), sp.simplify(Np1[1]))
mt1q4 = sp.simplify(limitElements(N0[0], tau, sp.oo))
sp.pprint(mt1q4)
Qt1q4 = sp.simplify(limitElements(N0[1], tau, sp.oo))
sp.pprint(Qt1q4)
```

B

Runge-Kutta initialization values

This chapter provides the algebraic expressions for the integrated Wiener process after the improper prior has collapsed to finite value.

The once integrated Wiener process does not require an improper prior model to match a Runge-Kutta initialization scheme.

For $q = 2$, the predictive posterior at $t_0 + h = t_1$ is normally distributed with mean

$$\boldsymbol{m}_{t_1} = \begin{pmatrix} y_0 + h(1 - \frac{1}{2u})z_0 + h\frac{1}{2u}z_1 & z_0 + \frac{1}{u}(z_1 - z_0) & \frac{z_1 - z_0}{hu} \end{pmatrix}^\mathsf{T}$$

and covariance matrix

$$\boldsymbol{C}_{t_1} = \sigma^2 \begin{pmatrix} \frac{h^5(-5u^3+20u^2-20u+6)}{120} & & \text{[symmetric]} \\ \frac{h^4(-u^3+6u^2-8u+3)}{24} & \frac{h^3(u-1)^2}{3} & \\ \frac{h^3(-u^3+4u^2-8u+4)}{24} & \frac{h^2(u-3)(u-1)}{6} & \frac{h(3-2u)}{3} \end{pmatrix}$$

The predictive posterior mean of the IWP(3) is defined by

$$\boldsymbol{m}_{t_2} = \begin{pmatrix} \frac{\left(hu^2vz_0 - \frac{hz_0}{2}u^2 + \frac{hz_2}{2}u^2 - huv^2z_0 + \frac{hu}{3}z_0 - \frac{hu}{3}z_2 + \frac{hz_0}{2}v^2 - \frac{hz_1}{2}v^2 - \frac{hv}{3}z_0 + \frac{hv}{3}z_1 + u^2vy_0 - uv^2y_0\right)}{uv(u-v)} \\ \frac{\left(u^2vz_0 - u^2z_0 + u^2z_2 - uv^2z_0 + uz_0 - uz_2 + v^2z_0 - v^2z_1 - vz_0 + vz_1\right)}{uv(u-v)} \\ \frac{\left(-u^2z_0 + u^2z_2 + 2uz_0 - 2uz_2 + v^2z_0 - v^2z_1 - 2vz_0 + 2vz_1\right)}{huv(u-v)} \\ \frac{(2uz_0 - 2uz_2 - 2vz_0 + 2vz_1)}{h^2uv(u-v)} \end{pmatrix}.$$

The entries of the covariance matrix $\boldsymbol{C}_{t_2}$ are

$$\begin{aligned}
(\boldsymbol{C}_{t_2})_{0,0} = -\frac{h^7\sigma^2}{15120v}\Big(&21u^4v^2 - 14u^4v + 21u^3v^3 - 140u^3v^2 + 147u^3v - 42u^3 \\
&+ 21u^2v^4 - 140u^2v^3 + 210u^2v^2 - 126u^2v + 28u^2 + 21uv^5 - 140uv^4 \\
&+399uv^3 - 378uv^2 + 112uv - 14v^5 + 84v^4 - 210v^3 + 196v^2 - 60v\Big), \\
(\boldsymbol{C}_{t_2})_{1,0} = -\frac{h^6\sigma^2}{720v}\Big(&u^4v^2 - u^4v + u^3v^3 - 10u^3v^2 + 14u^3v - 5u^3 + u^2v^4 \\
&- 10u^2v^3 + 20u^2v^2 - 15u^2v + 4u^2 + uv^5 - 10uv^4 + 38uv^3 - 45uv^2 \\
&+16uv - v^5 + 8v^4 - 25v^3 + 28v^2 - 10v\Big), \\
(\boldsymbol{C}_{t_2})_{2,0} = -\frac{h^5\sigma^2}{720v}\Big(&u^4v^2 - 2u^4v + u^3v^3 - 8u^3v^2 + 18u^3v - 8u^3 + u^2v^4 \\
&- 8u^2v^3 + 24u^2v^2 - 24u^2v + 8u^2 + uv^5 - 8uv^4 + 42uv^3 - 72uv^2 \\
&+32uv - 2v^5 + 12v^4 - 40v^3 + 56v^2 - 24v\Big),
\end{aligned}$$

$$
\begin{aligned}
(\boldsymbol{C}_{t_2})_{3,0} &= \frac{h^4\sigma^2}{360v}\Big(v\Big(u^4-2u^3v+6uv^3-3v^4+15\,(v-1)^4\Big)\\
&\quad +4\,(v-1)^3\Big(u^2+4uv-8v^2\Big)+3\,(v-1)^2\Big(u^3-u^2v-7uv^2+7v^3\Big)\Big),\\
(\boldsymbol{C}_{t_2})_{1,1} &= \frac{h^5\sigma^2}{60v}\Big(u^3v^2-2u^3v+u^3+u^2v^3-3u^2v^2+3u^2v-u^2\\
&\quad +uv^4-6uv^3+9uv^2-4uv-v^4+5v^3-7v^2+3v\Big),\\
(\boldsymbol{C}_{t_2})_{2,1} &= \frac{h^4\sigma^2}{120v}\Big(u^3v^2-4u^3v+3u^3+u^2v^3-6u^2v^2+9u^2v-4u^2+uv^4\\
&\quad -12uv^3+27uv^2-16uv-2v^4+15v^3-28v^2+15v\Big),\\
(\boldsymbol{C}_{t_2})_{3,1} &= \frac{h^3\sigma^2}{60v}\,(v-1)\Big(-u^3+u^2v+7uv^2-7v^3-10v\,(v-1)^2\\
&\quad +2\,(v-1)\Big(-u^2-4uv+8v^2\Big)\Big),\\
(\boldsymbol{C}_{t_2})_{2,2} &= -\frac{h^3\sigma^2}{60v}\Big(u^3v-2u^3+2u^2v^2-6u^2v\\
&\quad +4u^2+5uv^3-18uv^2+16uv-10v^3+28v^2-20v\Big),\\
(\boldsymbol{C}_{t_2})_{3,2} &= \frac{h^2\sigma^2}{60v}\Big(u^3+3u^2v-4u^2+9uv^2-16uv+5v^3-28v^2+30v\Big),\\
(\boldsymbol{C}_{t_2})_{3,3} &= -\frac{h\sigma^2}{15v}\Big(u^2+4uv+7v^2-15v\Big),
\end{aligned}
$$

and the rest are defined implicitely since $\boldsymbol{C}_{t_2}$ is a symmetric matrix.

The predictive posterior mean of the IWP(4) is defined by

$$
\begin{aligned}
(\boldsymbol{m}_{t_3})_0 &= y_0 + h\frac{1-2(u+v)+6uv}{12uv}z_0 + h\frac{2v-1}{12u(u-v)(u-1)}z_1\\
&\quad + h\frac{1-2u}{12v(u-v)(v-1)}z_2 + h\frac{3-4(u+v)+6uv}{12(u-1)(v-1)}z_3\\
(\boldsymbol{m}_{t_3})_1 &= z_3\\
(\boldsymbol{m}_{t_3})_2 &= \frac{1}{h}\frac{u+v-uv-1}{uv}z_0 + \frac{1}{h}\frac{1-v}{u(u-v)(u-1)}z_1\\
&\quad + \frac{1}{h}\frac{u-1}{v(u-v)(v-1)}z_2 + \frac{1}{h}\frac{3-2(u+v)+uv}{(u-1)(v-1)}z_3\\
(\boldsymbol{m}_{t_3})_3 &= \frac{1}{h^2}\frac{2(u+v-2)}{uv}z_0 + \frac{1}{h^2}\frac{2(2-v)}{u(u-v)(u-1)}z_1\\
&\quad + \frac{1}{h^2}\frac{2(u-2)}{v(u-v)(v-1)}z_2 + \frac{1}{h^2}\frac{2(3-u-v)}{(u-1)(v-1)}z_3\\
(\boldsymbol{m}_{t_3})_4 &= \frac{1}{h^3}\frac{-6}{uv}z_0 + \frac{1}{h^3}\frac{6}{u(u-v)(u-1)}z_1\\
&\quad + \frac{1}{h^3}\frac{-6}{v(u-v)(v-1)}z_2 + \frac{1}{h^3}\frac{6}{(u-1)(v-1)}z_3
\end{aligned}
$$

Furthermore, we get the following algebraic equations for the elements of the covariance matrix $\boldsymbol{C}_{t_3}$:

$$
\begin{aligned}
(\boldsymbol{C}_{t_3})_{0,0} = \sigma^2 h^9 [&6u^6v^2 - 3u^6v + 6u^5v^3 - 27u^5v^2 + 20u^5v \\
&- 4u^5 + 6u^4v^4 - 27u^4v^3 + 28u^4v^2 - 12u^4v \\
&+ 2u^4 + 6u^3v^5 - 27u^3v^4 + 28u^3v^3 - 12u^3v^2 \\
&+ 2u^3v + 6u^2v^6 - 27u^2v^5 + 28u^2v^4 + 68u^2v^3 \\
&- 78u^2v^2 + 20u^2v - 9uv^6 + 38uv^5 - 42uv^4 \\
&- 48uv^3 + 70uv^2 - 20uv + 3v^6 - 13v^5 + 17v^4 \\
&+ 5v^3 - 15v^2 + 5v] / [725760v(1-u)]
\end{aligned}
$$

$$
(\boldsymbol{C}_{t_3})_{1,0} = 0
$$

$$
\begin{aligned}
(\boldsymbol{C}_{t_3})_{2,0} = \sigma^2 h^7 [&(v-1)(3u^6v + 3u^5v^2 - 16u^5v + 6u^5 + 3u^4v^3 \\
&- 16u^4v^2 + 14u^4v - 4u^4 + 3u^3v^4 - 16u^3v^3 \\
&+ 14u^3v^2 - 4u^3v + 3u^2v^5 - 16u^2v^4 + 14u^2v^3 \\
&+ 76u^2v^2 - 40u^2v - 6uv^5 + 29uv^4 - 24uv^3 \\
&- 85uv^2 + 50uv + 3v^5 - 14v^4 + 15v^3 + 20v^2 \\
&- 15v)] / [120960v(u-1)]
\end{aligned}
$$

$$
\begin{aligned}
(\boldsymbol{C}_{t_3})_{3,0} = \sigma^2 h^6 [&3u^6v^2 - 6u^6v + 3u^5v^3 - 18u^5v^2 + 32u^5v \\
&- 10u^5 + 3u^4v^4 - 18u^4v^3 + 40u^4v^2 - 30u^4v \\
&+ 8u^4 + 3u^3v^5 - 18u^3v^4 + 40u^3v^3 - 30u^3v^2 \\
&+ 8u^3v + 3u^2v^6 - 18u^2v^5 + 40u^2v^4 + 50u^2v^3 \\
&- 192u^2v^2 + 80u^2v - 9uv^6 + 41uv^5 - 69uv^4 \\
&- 81uv^3 + 271uv^2 - 117uv + 6v^6 - 28v^5 \\
&+ 50v^4 + 2v^3 - 78v^2 + 39v] / [60480v(u-1)]
\end{aligned}
$$

$$
\begin{aligned}
(\boldsymbol{C}_{t_3})_{4,0} = \sigma^2 h^5 [&3u^6v + 3u^5v^2 - 12u^5v + 4u^5 + 3u^4v^3 \\
&- 12u^4v^2 + 12u^4v - 4u^4 + 3u^3v^4 - 12u^3v^3 \\
&+ 12u^3v^2 - 4u^3v + 3u^2v^5 - 12u^2v^4 + 12u^2v^3 \\
&+ 76u^2v^2 - 40u^2v + 3uv^6 - 12uv^5 + 12uv^4 \\
&+ 16uv^3 - 140uv^2 + 72uv - 3v^6 + 13v^5 - 19v^4 \\
&+ 5v^3 + 45v^2 - 27v] / [20160v(1-u)] \\
(\boldsymbol{C}_{t_3})_{1,1} = 0 \\
(\boldsymbol{C}_{t_3})_{2,1} = 0 \\
(\boldsymbol{C}_{t_3})_{3,1} = 0 \\
(\boldsymbol{C}_{t_3})_{4,1} = 0.
\end{aligned}
$$

The last four equations are a consequence of the noise-free observation z_3 at $t_0 + c_4 h = t_0 + h = t_3$. The remaining entries are given by the expressions

$$
\begin{aligned}
(\boldsymbol{C}_{t_3})_{2,2} = \sigma^2 h^5 [&(v-1)^2(u^4 + u^3v + u^2v^2 + uv^3 - 10uv \\
&- 2v^3 + 2v^2 + 6v)] / [2520v] \\
(\boldsymbol{C}_{t_3})_{3,2} = \sigma^2 h^4 [&(v-1)(u^5v - 3u^5 + u^4v^2 - 5u^4v + 4u^4 \\
&+ u^3v^3 - 5u^3v^2 + 4u^3v + u^2v^4 - 5u^2v^3 \\
&- 16u^2v^2 + 40u^2v - 5uv^4 + 15uv^3 + 37uv^2 \\
&- 77uv + 5v^4 - 15v^3 - 11v^2 + 33v)] / [2520v(u-1)] \\
(\boldsymbol{C}_{t_3})_{4,2} = \sigma^2 h^3 [&(v-1)(-u^5 - u^4v + 2u^4 - u^3v^2 + 2u^3v \\
&- u^2v^3 + 2u^2v^2 + 20u^2v - uv^4 + 2uv^3 \\
&+ 5uv^2 - 50uv + 2v^4 - 5v^3 + 25v)] / [840v(u-1)] \\
(\boldsymbol{C}_{t_3})_{3,3} = \sigma^2 h^3 [&u^5v - 2u^5 + 2u^4v^2 - 6u^4v + 4u^4 + 2u^3v^3 \\
&- 6u^3v^2 + 4u^3v + 2u^2v^4 + 4u^2v^3 - 36u^2v^2 \\
&+ 40u^2v + uv^5 - 12uv^4 - 12uv^3 + 104uv^2 - 96uv \\
&- 2v^5 + 16v^4 - 8v^3 - 48v^2 + 48v] / [630v(1-u)] \\
(\boldsymbol{C}_{t_3})_{4,3} = \sigma^2 h^2 [&u^5 + 3u^4v - 4u^4 + 3u^3v^2 - 4u^3v + 3u^2v^3 \\
&+ 16u^2v^2 - 40u^2v + 3uv^4 + uv^3 - 95uv^2 + 135uv \\
&+ v^5 - 10v^4 + 14v^3 + 54v^2 - 81v] / [420v(u-1)] \\
(\boldsymbol{C}_{t_3})_{4,4} = \sigma^2 h [&u^4 + u^3v + u^2v^2 + 10u^2v + uv^3 + 20uv^2 \\
&- 60uv + v^4 - 5v^3 - 15v^2 + 45v] / [70v(1-u)]
\end{aligned}
$$

which defines the entire matrix since $\boldsymbol{C}_{t_3}$ is a symmetric matrix.

For the entries of $\underline{\boldsymbol{Q}}(t)$ in the case $q = 3$, we get the following expressions:

$$
\begin{array}{llll}
(\underline{\boldsymbol{Q}}(t))_{0,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{1,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{2,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{3,0} = 0 \\
(\underline{\boldsymbol{Q}}(t))_{4,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{5,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{6,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{7,0} = 0 \\
(\underline{\boldsymbol{Q}}(t))_{8,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{9,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{10,0} = 0 & (\underline{\boldsymbol{Q}}(t))_{11,0} = 0 \\
 & (\underline{\boldsymbol{Q}}(t))_{1,1} = 0 & (\underline{\boldsymbol{Q}}(t))_{2,1} = 0 & (\underline{\boldsymbol{Q}}(t))_{3,1} = 0 \\
(\underline{\boldsymbol{Q}}(t))_{4,1} = 0 & (\underline{\boldsymbol{Q}}(t))_{5,1} = 0 & (\underline{\boldsymbol{Q}}(t))_{6,1} = 0 & (\underline{\boldsymbol{Q}}(t))_{7,1} = 0 \\
(\underline{\boldsymbol{Q}}(t))_{8,1} = 0 & (\underline{\boldsymbol{Q}}(t))_{9,1} = 0 & (\underline{\boldsymbol{Q}}(t))_{10,1} = 0 & (\underline{\boldsymbol{Q}}(t))_{11,1} = 0
\end{array}
$$

$$
\begin{array}{ll}
(\underline{\boldsymbol{Q}}(t))_{2,2} = -\frac{h^3\sigma^2}{60}u^2\,(u-3v) & (\underline{\boldsymbol{Q}}(t))_{3,2} = -\frac{h^2\sigma^2 u}{60v}\left(-u^2+2uv+6v^2\right) \\
(\underline{\boldsymbol{Q}}(t))_{4,2} = \frac{h^5\sigma^2u^5}{720v}\,(-u+5v) & (\underline{\boldsymbol{Q}}(t))_{5,2} = \frac{h^4\sigma^2u^4}{120v}\,(-u+4v) \\
(\underline{\boldsymbol{Q}}(t))_{6,2} = \frac{h^3\sigma^2u^3}{24v}\,(-u+3v) & (\underline{\boldsymbol{Q}}(t))_{7,2} = \frac{h^2\sigma^2u^2}{6v}\,(-u+2v) \\
(\underline{\boldsymbol{Q}}(t))_{8,2} = -\frac{h^5\sigma^2u}{72v}\,(u-v)^5 & (\underline{\boldsymbol{Q}}(t))_{9,2} = \frac{h^4\sigma^2u}{20v}\,(u-v)^4 \\
(\underline{\boldsymbol{Q}}(t))_{10,2} = -\frac{h^3\sigma^2u}{8v}\,(u-v)^3 & (\underline{\boldsymbol{Q}}(t))_{11,2} = -\frac{h^2\sigma^2u}{6v}\left(-u^2+2uv-v^2\right)
\end{array}
$$

$$
\begin{array}{ll}
(\underline{\boldsymbol{Q}}(t))_{3,3} = -\frac{h\sigma^2}{15v}\left(u^2-6uv-3v^2\right) & (\underline{\boldsymbol{Q}}(t))_{4,3} = \frac{h^4\sigma^2u^4}{360v}\,(u-15v) \\
(\underline{\boldsymbol{Q}}(t))_{5,3} = \frac{h^3\sigma^2u^3}{60v}\,(u-10v) & (\underline{\boldsymbol{Q}}(t))_{6,3} = \frac{h^2\sigma^2u^2}{12v}\,(u-6v) \\
(\underline{\boldsymbol{Q}}(t))_{7,3} = \frac{h\sigma^2u}{3v}\,(u-3v) & (\underline{\boldsymbol{Q}}(t))_{8,3} = \frac{h^4\sigma^2}{36v}\,(u-v)^5 \\
(\underline{\boldsymbol{Q}}(t))_{9,3} = -\frac{h^3\sigma^2}{10v}\,(u-v)^4 & (\underline{\boldsymbol{Q}}(t))_{10,3} = \frac{h^2\sigma^2}{4v}\,(u-v)^3 \\
(\underline{\boldsymbol{Q}}(t))_{11,3} = -\frac{h\sigma^2}{3v}\,(u-v)^2 &
\end{array}
$$

$$
\begin{array}{ll}
(\underline{\boldsymbol{Q}}(t))_{4,4} = \frac{h^7\sigma^2}{252}u^7 & (\underline{\boldsymbol{Q}}(t))_{5,4} = \frac{h^6\sigma^2}{72}u^6 \\
(\underline{\boldsymbol{Q}}(t))_{6,4} = \frac{h^5\sigma^2}{30}u^5 & (\underline{\boldsymbol{Q}}(t))_{7,4} = \frac{h^4\sigma^2}{24}u^4 \\
(\underline{\boldsymbol{Q}}(t))_{8,4} = 0 & (\underline{\boldsymbol{Q}}(t))_{9,4} = 0 \\
(\underline{\boldsymbol{Q}}(t))_{10,4} = 0 & (\underline{\boldsymbol{Q}}(t))_{11,4} = 0
\end{array}
$$

$$
\begin{array}{lll}
(\underline{\boldsymbol{Q}}(t))_{5,5} = \frac{h^5\sigma^2}{20}u^5 & (\underline{\boldsymbol{Q}}(t))_{6,5} = \frac{h^4\sigma^2}{8}u^4 & \\
(\underline{\boldsymbol{Q}}(t))_{7,5} = \frac{h^3\sigma^2}{6}u^3 & (\underline{\boldsymbol{Q}}(t))_{8,5} = 0 & \\
(\underline{\boldsymbol{Q}}(t))_{9,5} = 0 & (\underline{\boldsymbol{Q}}(t))_{10,5} = 0 & (\underline{\boldsymbol{Q}}(t))_{11,5} = 0
\end{array}
$$

$$
\begin{array}{lll}
(\underline{\boldsymbol{Q}}(t))_{6,6} = \frac{h^3\sigma^2}{3}u^3 & (\underline{\boldsymbol{Q}}(t))_{7,6} = \frac{h^2\sigma^2}{2}u^2 & (\underline{\boldsymbol{Q}}(t))_{8,6} = 0 \\
(\underline{\boldsymbol{Q}}(t))_{9,6} = 0 & (\underline{\boldsymbol{Q}}(t))_{10,6} = 0 & (\underline{\boldsymbol{Q}}(t))_{11,6} = 0
\end{array}
$$

$$
\begin{array}{lll}
(\underline{\boldsymbol{Q}}(t))_{7,7} = h\sigma^2u & (\underline{\boldsymbol{Q}}(t))_{8,7} = 0 & (\underline{\boldsymbol{Q}}(t))_{9,7} = 0 \\
(\underline{\boldsymbol{Q}}(t))_{10,7} = 0 & (\underline{\boldsymbol{Q}}(t))_{11,7} = 0 &
\end{array}
$$

$$(\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{8,8} = -\frac{h^7\sigma^2}{252}(u-v)^7 \quad (\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{9,8} = \frac{h^6\sigma^2}{72}(u-v)^6$$

$$(\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{10,8} = -\frac{h^5\sigma^2}{30}(u-v)^5 \quad (\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{11,8} = \frac{h^4\sigma^2}{24}(u-v)^4$$

$$(\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{9,9} = -\frac{h^5\sigma^2}{20}(u-v)^5 \quad (\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{10,9} = \frac{h^4\sigma^2}{8}(u-v)^4$$

$$(\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{11,9} = -\frac{h^3\sigma^2}{6}(u-v)^3$$

$$(\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{10,10} = -\frac{h^3\sigma^2}{3}(u-v)^3 \quad (\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{11,10} = \frac{h^2\sigma^2}{2}(u-v)^2$$

$$(\underline{\boldsymbol{Q}}(\boldsymbol{t}))_{11,11} = -h\sigma^2(u-v)$$

The rest of the entries follow from $\underline{\boldsymbol{Q}}(\boldsymbol{t})$ being a symmetric matirx.

C

Poincaré translation

Dr. Johannes Dahlem and Dr. Michael Perrot helped me to translate the passage from [171], quoted in Section 4.1, into English:

> "Let us take a different point of view.
>
> A question of probability arises as a consequence of our ignorance: there would only be place for certainty, if we would know all the givens of a problem. On the other hand, our ignorance should not be complete or we could not evaluate anything. A classification would, therefore, follow our degree of ignorance.
>
> For instance, the probability of the the sixth decimal of a number in a logarithmic table being equal to 6 is $1/10$ *a priori*; in reality, all the givens of the problem are well determined, and, if we wanted to go to such lengths, we would know this probability exactly. Similarly, the same holds for interpolation, for the computation of definite integrals after the method of Cotes or of Gauß, etc.
>
> [...]
>
> In other problems, finally, it might happen that our ignorance is bigger still, such that the law itself eludes us. Then the definition of probabilities becomes almost impossible. For example, if x is an unknown function of t, then we do not know very well which probability we need to initially assign to x_0 to know
>
> $$\int_{t_0}^{t} x \, \mathrm{d}t.$$
>
> One will often let oneself be guided by a vague feeling which will impose mightyly, which one will not be able to justify, but without which science is not possible in any case."

Bibliography

[1] A. Abdulle and G. Garegnani. "Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration." In: *ArXiv e-prints* (Jan. 2018). arXiv: 1801.01340 [math.NA].

[2] R. Adler. *The Geometry of Random Fields*. Wiley, 1981.

[3] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh. "Fundamental Properties of Generalized Splines." In: *Proceedings of the National Academy of Sciences* 52 (1964), pp. 1412–1419.

[4] P. Albrecht. "Explicit, optimal stability functionals and their application to cyclic discretization methods." In: *Computing* 19.3 (1978), pp. 233–249. DOI: 10.1007/BF02252202.

[5] G. D. Andria, G. D. Byrne, and D. R. Hill. "Integration formulas and schemes based on g-splines." In: *Mathematics of Computation* 27.124 (1973), pp. 831–838.

[6] U. Ascher and R. D. Russell. "Reformulation of boundary value problems into "standard" form." In: *SIAM review* 23.2 (1981), pp. 238–254.

[7] U. M. Ascher, R. M. Mattheij, and R. D. Russell. *Numerical solution of boundary value problems for ordinary differential equations*. Vol. 13. Siam, 1994.

[8] Y. Assaf and O. Pasternak. "Diffusion tensor imaging (DTI)-based white matter mapping in brain research: a review." In: *Journal of molecular neuroscience* 34.1 (2008), pp. 51–61.

[9] L. Astola, A. Jalba, E. Balmashnova, and L. Florack. "Finsler Streamline Tracking with Single Tensor Orientation Distribution Function for High Angular Resolution Diffusion Imaging." In: *Journal of Mathematical Imaging and Vision* 41.3 (2011), pp. 170–181.

[10] A. Axelsson and J. Verwer. "Boundary value techniques for initial value problems in ordinary differential equations." In: *Mathematics of Computation* 45.171 (1985), pp. 153–171.

[11] F. Bashforth and J. C. Adams. *An Attempt to Test the Theories of Capillary Action by Comparing the Theoretical and Measured Forms of Drops of Fluid. With an Explanation of the Method of Integration Employed in Constructing Tables Which Give the Theoretical Form of Such Drops*. Cambridge University Press, 1883.

[12] P. J. Basser, J. Mattiello, and D. LeBihan. "Estimation of the effective self-diffusion tensor from the NMR spin echo." In: *J Magn Reson B* 103.3 (1994), pp. 247–254.

[13] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi. "In vivo fiber tractography using DT-MRI data." In: *Magnetic resonance in medicine* 44.4 (2000), pp. 625–632.

[14] T. Bayes and R. Price. "An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S." In: *Philosophical Transactions (1683-1775)* 53 (1763), pp. 370–418.

[15] T. Behrens, M. Woolrich, M. Jenkinson, H. Johansen-Berg, R. Nunes, S. Clare, P. Matthews, J. Brady, and S. Smith. "Characterization and propagation of uncertainty in diffusion-weighted MR imaging." In: *Magnetic Resonance in Medicine* 50.5 (2003), pp. 1077–1088. DOI: 10.1002/mrm.10609.

[16] M. Betancourt. "A Conceptual Introduction to Hamiltonian Monte Carlo." In: *ArXiv e-prints* (Jan. 2017). arXiv: 1701.02434 [stat.ME].

[17] V. I. Bogachev. *Gaussian measures*. Providence: American Mathematical Society, 1998.

[18] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. "Manopt, a Matlab Toolbox for Optimization on Manifolds." In: *Journal of Machine Learning Research* 15 (2014), pp. 1455–1459. URL: http://www.manopt.org.

[19] V. Braitenberg and A Schüz. "Allgemeine Neuroanatomie." In: *Neuro-und Sinnesphysiologie*. Ed. by R. F. Schmidt and H.-G. Schaible. Springer, 2006. Chap. 2, pp. 2–13.

[20] R Brecheisen. "Visualization of uncertainty in fiber tracking based on diffusion tensor imaging." PhD thesis. Technische Universiteit Eindhoven, 2012.

[21] R. Brecheisen, B. Platel, B. M. ter Haar Romeny, and A. Vilanova. "Illustrative uncertainty visualization of DTI fiber pathways." In: *The Visual Computer* 29.4 (2013), pp. 297–309. DOI: 10.1007/s00371-012-0733-9.

[22] P. Broca. "Perte de la parole, ramollissement chronique et destruction partielle du lobe antérieur gauche du cerveau." In: *Bulletins de la Société d'Anthropologie de Paris* 2.1 (1861), pp. 235–238.

[23] P. Brown, G. Byrne, and A. Hindmarsh. "VODE: A Variable-Coefficient ODE Solver." In: *SIAM Journal on Scientific and Statistical Computing* 10.5 (1989), pp. 1038–1051. DOI: 10.1137/0910062.

[24] L. Brugnano, F. Mazzia, and D. Trigiante. "Fifty years of stiffness." In: *Recent Advances in Computational and Applied Mathematics*. Springer, 2011, pp. 1–21.

[25] L. Brugnano and D. Trigiante. *Solving Differential Equations by Multistep Initial and Boundary Value Methods*. CRC Press, 1998.

[26] H. Bruns. "Über die Integrale des Vielkörper-Problems." In: *Acta Math.* 11 (1887), pp. 25–96. DOI: 10.1007/BF02612319.

[27] J. C. Butcher. "Numerical methods for ordinary differential equations in the 20th century." In: *Journal of Computational and Applied Mathematics* 125.1 (2000), pp. 1–29.

[28] G. D. Byrne and A. C. Hindmarsh. "A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations." In: *ACM Trans. Math. Softw.* 1.1 (Mar. 1975), pp. 71–96. ISSN: 0098-3500. DOI: 10.1145/355626.355636.

[29] G. D. Byrne and D. N. H. Chi. "Linear Multistep Formulas Based on g-Splines." In: *SIAM Journal on Numerical Analysis* 9.2 (1972), pp. 316–324. DOI: 10.1137/0709031.

[30] Y. Cao, S. Li, and L. Petzold. "Adjoint sensitivity analysis for differential-algebraic equations: algorithms and software." In: *Journal of Computational and Applied Mathematics* 149.1 (2002), pp. 171 –191. DOI: 10.1016/S0377-0427(02)00528-9.

[31] H. D. V. Cardona, M. A. Álvarez, and Á. A. Orozco. "Generalized Wishart processes for interpolation over diffusion tensor fields." In: *International Symposium on Visual Computing*. Springer. 2015, pp. 499–508.

[32] J. R. Cash and A. H. Karp. "A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides." In: *ACM Transactions on Mathematical Software (TOMS)* 16.3 (1990), pp. 201–222.

[33] J. R. Cash and F. Mazzia. "Efficient Global Methods for the Numerical Solution of Nonlinear Systems of Two Point Boundary Value Problems." In: *Recent Advances in Computational and Applied Mathematics*. Springer, 2011, pp. 23–39.

[34] M. Catani, F. Dell'Acqua, A. Bizzi, S. J. Forkel, S. C. Williams, A. Simmons, D. G. Murphy, and M. T. de Schotten. "Beyond cortical localization in clinico-anatomical correlation." In: *Cortex* 48.10 (2012), pp. 1262–1287.

[35] M. Catani and M. T. de Schotten. "A diffusion tensor imaging tractography atlas for virtual in vivo dissections." In: *Cortex* 44.8 (2008), pp. 1105–1132.

[36] F. Ceschino. "Modification de la longueur du pas dans l'intégration numérique par les méthodes à pas liés." In: *Chiffres* 2 (1961), pp. 101–106.

[37] P. Cheng, V. Magnotta, D. Wu, P. Nopoulos, D. Moser, J. Paulsen, R. Jorge, and N. Andreasen. "Evaluation of the GTRACT diffusion tensor tractography algorithm: A validation and reliability study." In: *NeuroImage* 31.3 (2006), pp. 1075–1085.

[38] O. A. Chkrebtii, D. A. Campbell, B. Calderhead, and M. A. Girolami. "Bayesian Solution Uncertainty Quantification for Differential Equations." In: *ArXiv e-prints* (June 2013). arXiv: 1306.2365v1 [stat.ME].

[39] O. A. Chkrebtii, D. A. Campbell, B. Calderhead, and M. A. Girolami. "Bayesian Solution Uncertainty Quantification for Differential Equations." In: *Bayesian Anal.* 11.4 (Dec. 2016), pp. 1239–1267. DOI: 10.1214/16-BA1017.

[40] J. Cockayne, C. Oates, T. Sullivan, and M. Girolami. "Probabilistic Numerical Methods for Partial Differential Equations and Bayesian Inverse Problems." In: *ArXiv e-prints* (May 2016). arXiv: 1605.07811 [stat.ME].

[41] J. Cockayne, C. Oates, T. Sullivan, and M. Girolami. "Bayesian Probabilistic Numerical Methods." In: *ArXiv e-prints* (Feb. 2017). arXiv: 1702.03673 [stat.ME].

[42] P. R. Conrad, M. Girolami, S. Särkkä, A. Stuart, and K. Zygalakis. "Statistical analysis of differential equations: introducing probability measures on numerical solutions." In: *Statistics and Computing* 27.4 (July 2017), pp. 1065–1082. DOI: 10.1007/s11222-016-9671-0.

[43] J. D. Cook. “Testing a Random Number Generator.” In: *Beautiful Testing: Leading Professionals Reveal How They Improve Software*. Ed. by A. Goucher and T. Riley. O’Reilly Media, 2009, pp. 129–141.

[44] R. Cox. “Probability, frequency and reasonable expectation.” In: *American Journal of Physics* 14.1 (1946), pp. 1–13.

[45] R. C. Craddock, S. Jbabdi, C.-G. Yan, J. T. Vogelstein, F. X. Castellanos, A. Di Martino, C. Kelly, K. Heberlein, S. Colcombe, and M. P. Milham. “Imaging human connectomes at the macroscale.” In: *Nature methods* 10.6 (2013), pp. 524–539.

[46] P. Crane and P. Fox. *A comparative study of computer programs for integrating differential equations*. Tech. rep. Bell Telephone Laboratories, 1969.

[47] M Crouzeix and F. Lisbona. “The convergence of variable-stepsize, variable-formula, multistep methods.” In: *SIAM journal on numerical analysis* 21.3 (1984), pp. 512–534.

[48] C. Curtiss and J. O. Hirschfelder. “Integration of stiff equations.” In: *Proceedings of the National Academy of Sciences* 38.3 (1952), pp. 235–243.

[49] G. Dahlquist. “Convergence and Stability in the Numerical Integration of Ordinary Differential Equations.” In: *Mathematica Scandinavica* 4.1 (1956), pp. 33–53.

[50] G. Dahlquist. “A special stability problem for linear multistep methods.” In: *BIT Numerical Mathematics* 3.1 (1963), pp. 27–43.

[51] G. Dahlquist. “33 years of numerical instability, Part I.” In: *BIT Numerical Mathematics* 25.1 (1985), pp. 188–204.

[52] R. S. Desikan, F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, M. S. Albert, and R. J. Killiany. “An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest.” In: *NeuroImage* 31.3 (2006), pp. 968–980.

[53] P. Deuflhard and F. Bornemann. *Scientific Computing with Ordinary Differential Equations*. New York: Springer, 2002.

[54] P. Deuflhard. “Order and stepsize control in extrapolation methods.” In: *Numerische Mathematik* 41.3 (1983), pp. 399–422.

[55] P. Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*. Vol. 35. Springer Science & Business Media, 2011.

[56] P. Diaconis. “Bayesian numerical analysis.” In: *Statistical decision theory and related topics* IV.1 (1988), pp. 163–175.

[57] E. Dijkstra. “A note on two problems in connexion with graphs.” In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

[58] Q. Dong, R. C. Welsh, T. L. Chenevert, R. C. Carlos, P. Maly-Sundgren, D. M. Gomez-Hassan, and S. K. Mukherji. “Clinical applications of diffusion tensor imaging.” In: *Journal of Magnetic Resonance Imaging* 19.1 (2004), pp. 6–18. DOI: 10.1002/jmri.10424.

[59] A. Doucet. *A note on efficient conditional simulation of Gaussian distributions*. Tech. rep. Departments of Computer Science and Statistics, University of British Columbia, Apr. 2010.

[60] R. Durrett. *Probability: theory and examples*. Cambridge University Press, 2010.

[61] H.-H. Ehricke, U. Klose, and W. Grodd. "Visualizing MR diffusion tensor fields by dynamic fiber tracking and uncertainty mapping." In: *Computers & Graphics* 30.2 (2006), pp. 255 –264. DOI: 10.1016/j.cag.2006.01.031.

[62] D. C. V. Essen, S. M. Smith, D. M. Barch, T. E. J. Behrens, E. Yacoub, K. Ugurbil, and W. U.-M. H. Consortium. "The WU-Minn Human Connectome Project: an overview." In: *NeuroImage* 80 (2013), pp. 62–79.

[63] A. Feragen and A. Fuster. *Geometries and interpolations for symmetric positive definite matrices*. Tech. rep. Centre for Stochastic Geometry and Advanced Bioimaging, Aarhus University, 2016.

[64] R. Fierro and S. Torres. "A stochastic scheme of approximation for ordinary differential equations." In: *Electronic Communications in Probability* 13 (2008), pp. 1–9.

[65] L. Florack, T. Dela Haije, and A. Fuster. "Direction-Controlled DTI Interpolation." In: *Visualization and Processing of Higher Order Descriptors for Multi-Valued Data*. Ed. by I. Hotz and T. Schultz. Springer International Publishing, 2015, pp. 149–162. DOI: 10.1007/978-3-319-15090-1_8.

[66] K. J. Friston. "Functional and effective connectivity: a review." In: *Brain connectivity* 1.1 (2011), pp. 13–36.

[67] A. Fuster, A. Tristan-Vega, T. Haije, C.-F. Westin, and L. Florack. "A Novel Riemannian Metric for Geodesic Tractography in DTI." In: *Computational Diffusion MRI and Brain Connectivity*. Mathematics and Visualization. Springer, 2014, pp. 97–104.

[68] C. W. Gear. "Runge-Kutta Starters for Multistep Methods." In: *ACM Trans. Math. Softw.* 6.3 (Sept. 1980), pp. 263–279. DOI: 10.1145/355900.355901.

[69] C. W. Gear. "Numerical Solution of Ordinary Differential Equations: Is There Anything Left to Do?" In: *SIAM Review* 23.1 (1981), pp. 10–24.

[70] J. E. Gentle. *Random number generation and Monte Carlo methods*. Second Edition. Springer Science & Business Media, 2003.

[71] E. Giné and R. Nickl. *Mathematical foundations of infinite-dimensional statistical models*. Vol. 40. Cambridge University Press, 2015.

[72] M. Glasser, S. Sotiropoulos, J. Wilson, T. Coalson, B. Fischl, J. Andersson, J. Xu, S. Jbabdi, M. Webster, J. Polimeni, D. Van Essen, M. Jenkinson, and W. U.-M. H. Consortium. "The minimal preprocessing pipelines for the Human Connectome Project." In: *NeuroImage* 80 (2013), pp. 105–124.

[73] D. Goldberg. "What every computer scientist should know about floating-point arithmetic." In: *ACM Computing Surveys (CSUR)* 23.1 (1991), pp. 5–48.

[74] G. Gong, Y. He, L. Concha, C. Lebel, D. W. Gross, A. C. Evans, and C. Beaulieu. "Mapping anatomical connectivity patterns of human cerebral cortex using in vivo diffusion tensor imaging tractography." In: *Cerebral cortex* 19.3 (2008), pp. 524–536.

[75] M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB*. John Wiley & Sons, Inc., 2001.

[76] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. 2nd. Other Titles in Applied Mathematics 105. Philadelphia, PA: SIAM, 2008.

[77] A. Grigorievskiy, N. Lawrence, and S. Särkkä. "Parallelizable sparse inverse formulation Gaussian processes (SpInGP)." In: *ArXiv e-prints* (Oct. 2016). arXiv: 1610.08035 [stat.ML].

[78] K. Gustafsson. "Control Theoretic Techniques for Stepsize Selection in Explicit Runge-Kutta Methods." In: *ACM Trans. Math. Softw.* 17.4 (1991), pp. 533–554.

[79] S. Haber. "Tracing intrinsic fiber connections in postmortem human brain with WGA-HRP." In: *Journal of neuroscience methods* 23.1 (1988), pp. 15–22.

[80] W. Hager. "Updating the Inverse of a Matrix." In: *SIAM Review* 31.2 (1989), pp. 221–239.

[81] P. Hagmann, L. Jonasson, P. Maeder, J.-P. Thiran, V. J. Wedeen, and R. Meuli. "Understanding diffusion MR imaging techniques: from scalar diffusion-weighted imaging to diffusion tensor imaging and beyond." In: *Radiographics* 26.suppl_1 (2006), S205–S223. DOI: 10.1148/rg.26si065510.

[82] E. Hairer, S. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I – Nonstiff Problems*. Springer, 1987.

[83] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*. Springer, 1996.

[84] E. Hairer and G. Wanner. *Analysis by its History*. Springer, 2008.

[85] X. Hao, R. Whitaker, and P. Fletcher. "Adaptive Riemannian Metrics for Improved Geodesic Tracking of White Matter." In: *Information Processing in Medical Imaging*. Vol. 6801. LNCS. Springer, 2011, pp. 13–24.

[86] S. Hauberg, O. Freifeld, and M. Black. "A Geometric Take on Metric Learning." In: *Advances in Neural Information Processing Systems (NIPS) 25*. Ed. by P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger. MIT Press, 2012, pp. 2033–2041.

[87] S. Hauberg, M. Schober, M. Liptrot, P. Hennig, and A. Feragen. "A Random Riemannian Metric for Probabilistic Shortest-Path Tractography." In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*. Vol. 18. Springer, Sept. 2015.

[88] C. Heilingoetter and M. Jensen. "Histological methods for ex vivo axon tracing: A systematic review." In: *Neurological Research* 38.7 (2016), pp. 561–569. DOI: 10.1080/01616412.2016.1153820.

[89] M. Helmstaedter. "Cellular-resolution connectomics: challenges of dense neural circuit reconstruction." In: *Nature methods* 10.6 (2013), pp. 501–507.

[90] P. Hennig. *Animating Samples from Gaussian Distributions*. Tech. rep. 8. Max Planck Institute for Intelligent Systems, Tübingen, Germany, 2013.

[91] P. Hennig. "Probabilistic Interpretation of Linear Solvers." In: *SIAM J on Optimization* 25.1 (2015), pp. 210–233.

[92] P. Hennig and S. Hauberg. "Probabilistic Solutions to Differential Equations and their Application to Riemannian Statistics." In: *Proc. of the 17th int. Conf. on Artificial Intelligence and Statistics (AISTATS)*. Vol. 33. JMLR, W&CP, 2014.

[93] P. Hennig, M. A. Osborne, and M. Girolami. "Probabilistic numerics and uncertainty in computations." In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 471.2179 (2015).

[94] P. Henrici. "Test of Probabilistic Models for the Propagation of Roundoff Errors." In: *Commun. ACM* 9.6 (June 1966), pp. 409–410. DOI: 10.1145/365696.365698.

[95] A. Hindmarsh. "On Numerical Methods for Stiff Differential Equations–Getting the Power to the People." In: *Lawrence Livermore Laboratory report, UCRL-83259* (1979).

[96] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. "SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers." In: *ACM Trans. Math. Softw.* 31.3 (Sept. 2005), pp. 363–396. DOI: 10.1145/1089014.1089020.

[97] M. Hinne. "Bayesian Connectomics: A probabilistic perspective on brain networks." PhD thesis. Radboud University Nijmegen, 2017.

[98] T. E. Hull and J. R. Swenson. "Tests of Probabilistic Models for Propagation of Roundoff Errors." In: *Commun. ACM* 9.2 (Feb. 1966), pp. 108–113. DOI: 10.1145/365170.365212.

[99] T. Hull, W. Enright, B. Fellen, and A. Sedgwick. "Comparing numerical methods for ordinary differential equations." In: *SIAM Journal on Numerical Analysis* 9.4 (1972), pp. 603–637.

[100] Y. Iturria-Medina, E. J. Canales-Rodríguez, L. Melie-García, P. A. Valdés-Hernández, E. Martínez-Montes, Y. Alemán-Gómez, and J. M. Sánchez-Bornot. "Characterizing brain anatomical connections using diffusion weighted MRI and graph theory." In: *NeuroImage* 36.3 (2007), pp. 645–660.

[101] M. Jackowski, C. Y. Kao, M. Qiu, R. T. Constable, and L. H. Staib. "White matter tractography by anisotropic wavefront evolution and diffusion tensor imaging." In: *Medical Image Analysis* 9.5 (2005). Medical Image Computing and Computer-Assisted Intervention - MICCAI 2004, pp. 427 –440. DOI: 10.1016/j.media.2005.05.008.

[102] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[103] S. Jbabdi and H. Johansen-Berg. "Tractography: where do we go from here?" In: *Brain connectivity* 1.3 (2011), pp. 169–183.

[104] H. Jeffreys. *Theory of Probability*. 3rd ed. Oxford University Press, 1969.

[105] B. Jeurissen, A. Leemans, D. K. Jones, J.-D. Tournier, and J. Sijbers. "Probabilistic fiber tracking using the residual bootstrap with constrained spherical deconvolution." In: *Human Brain Mapping* 32.3 (2011), pp. 461–479. DOI: 10.1002/hbm.21032.

[106] M. John and Y. Wu. "Confidence intervals for finite difference solutions." In: *Communications in Statistics - Simulation and Computation* 0.0 (2017), pp. 1–17.

[107] D. K. Jones. "Determining and visualizing uncertainty in estimates of fiber orientation from diffusion tensor MRI." In: *Magnetic Resonance in Medicine* 49.1 (2003), pp. 7–12. DOI: 10.1002/mrm.10331.

[108] D. K. Jones, T. R. Knösche, and R. Turner. "White matter integrity, fiber count, and other fallacies: The do's and don'ts of diffusion MRI." In: *NeuroImage* 73 (2013), pp. 239 –254. DOI: 10.1016/j.neuroimage.2012.06.081.

[109] D. K. Jones and C. Pierpaoli. "Confidence mapping in diffusion tensor magnetic resonance imaging tractography using a bootstrap approach." In: *Magnetic Resonance in Medicine* 53.5 (2005), pp. 1143–1149. DOI: 10.1002/mrm.20466.

[110] D. K. Jones, A. R. Travis, G. Eden, C. Pierpaoli, and P. J. Basser. "PASTA: Pointwise assessment of streamline tractography attributes." In: *Magnetic Resonance in Medicine* 53.6 (2005), pp. 1462–1467. DOI: 10.1002/mrm.20484.

[111] W Kahan and J Palmer. "On a proposed floating-point standard." In: *ACM SIGNum Newsletter* 14.si-2 (1979), pp. 13–21.

[112] R. E. Kalman and R. S. Bucy. "New Results in Linear Filtering and Prediction Theory." In: *Journal of Fluids Engineering* 83.1 (1961), pp. 95–108.

[113] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems." In: *Journal of Fluids Engineering* 82.1 (1960), pp. 35–45.

[114] I. Karatzas and S. E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, 1991.

[115] N. Kasenburg, M. Liptrot, N. L. Reislev, S. N. Ørting, M. Nielsen, E. Garde, and A. Feragen. "Training shortest-path tractography: Automatic learning of spatial priors." In: *NeuroImage* 130 (2016), pp. 63 –76. DOI: 10.1016/j.neuroimage.2016.01.031.

[116] H. P. Kersting and P. Hennig. "Active Uncertainty Calibration in Bayesian ODE Solvers." In: *Uncertainty in Artificial Intelligence (UAI)*. Ed. by Janzing and Ihlers. Vol. 32. 2016.

[117] G. S. Kimeldorf and G. Wahba. "A correspondence between Bayesian estimation on stochastic processes and smoothing by splines." In: *The Annals of Mathematical Statistics* 41.2 (1970), pp. 495–502.

[118] M. Kline. *Mathematics and the Physical World*. John Murray, 1960.

[119] S. J. Koopman. "Exact initial Kalman filtering and smoothing for nonstationary time series models." In: *Journal of the American Statistical Association* 92.440 (1997), pp. 1630–1638.

[120] J. T. Krebs. "Consistency and asymptotic normality of stochastic Euler schemes for ordinary differential equations." In: *Statistics & Probability Letters* 125 (2017), pp. 1–8.

[121] F. T. Krogh. "On Testing a Subroutine for the Numerical Integration of Ordinary Differential Equations." In: *J. ACM* 20.4 (1973), pp. 545–562.

[122] G. Kulikov. "On quasi-consistent integration by Nordsieck methods." In: *Journal of Computational and Applied Mathematics* 225.1 (2009), pp. 268 –287. DOI: 10.1016/j.cam.2008.07.038.

[123] G. Kulikov and S. Shindin. "Nordsieck methods on nonuniform grids: Stability and order reduction phenomenon." In: *Mathematics and Computers in Simulation* 72.1 (2006), pp. 47 –56. DOI: 10.1016/j.matcom.2006.03.005.

[124] W. Kutta. "Beitrag zur näherungsweisen Integration totaler Differentialgleichungen." In: *Zeitschrift für Mathematik und Physik* 46 (1901), pp. 435–453.

[125] T. LaGatta and J. Wehr. "Geodesics of Random Riemannian Metrics." In: *Communications in Mathematical Physics* 327.1 (2014), pp. 181–241.

[126] P. Lancaster and L. Rodman. *Algebraic riccati equations*. Clarendon press, 1995.

[127] P. S. Laplace. "Sur le calcul des probabilités appliqué a la philosophie naturelle." In: *Connaissance des Temps pour l'an 1818* (Sept. 1815), pp. 378–381.

[128] F. M. Larkin. "Gaussian measure in Hilbert space and applications in numerical analysis." In: *Rocky Mountain Journal of Mathematics* 2.3 (1972), pp. 379–422.

[129] D. Le Bihan, E. Breton, D. Lallemand, P. Grenier, E. Cabanis, and M. Laval-Jeantet. "MR Imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders." In: *Radiology* 161.2 (1986), pp. 401–407.

[130] D. Le Bihan and H. Johansen-Berg. "Diffusion MRI at 25: exploring brain tissue structure and function." In: *Neuroimage* 61.2 (2012), pp. 324–341.

[131] S. H. Lee and W. Chen. "A comparative study of uncertainty propagation methods for black-box-type problems." In: *Structural and Multidisciplinary Optimization* 37.3 (May 2008), p. 239. DOI: 10.1007/s00158-008-0234-7.

[132] R. Lefever and G. Nicolis. "Chemical instabilities and sustained oscillations." In: *Journal of Theoretical Biology* 30.2 (1971), pp. 267 –284. DOI: 10.1016/0022-5193(71)90054-3.

[133] C. Lenglet, R. Deriche, and O. Faugeras. "Inferring White Matter Geometry from Diffusion Tensor MRI: Application to Connectivity Mapping." In: *Computer Vision – ECCV*. Vol. 3024. Lecture Notes in Computer Science. 2004, pp. 127–140.

[134] S. Li and L. Petzold. "Software and algorithms for sensitivity analysis of large-scale differential algebraic systems." In: *Journal of Computational and Applied Mathematics* 125.1 (2000), pp. 131 –145. DOI: 10.1016/S0377-0427(00)00464-7.

[135] H. C. Lie, A. M. Stuart, and T. J. Sullivan. "Strong convergence rates of probabilistic integrators for ordinary differential equations." In: *ArXiv e-prints* (Mar. 2017). arXiv: 1703.03680 [math.NA].

[136] M. A. Lifshits. *Gaussian random functions*. New York: Springer, 2013.

[137] E. Lindelöf. "Sur l'application des méthodes d'approximations successives à l'étude des intégrales réelles des équations différentielles ordinaires." In: *Journal de Mathématiques Pures et Appliquées* 10 (1894), pp. 117–128.

[138] M. G. Liptrot, K. Sidaros, and T. B. Dyrby. "Addressing the Path-Length-Dependency Confound in White Matter Tract Segmentation." In: *PLOS ONE* 9.5 (May 2014), pp. 1–11. DOI: 10.1371/journal.pone.0096247.

[139] M. Lorch. "Re-examining Paul Broca's initial presentation of M. Leborgne: Understanding the impetus for brain and language research." In: *Cortex* 47.10 (2011), pp. 1228 –1235. DOI: 10.1016/j.cortex.2011.06.022.

[140] F. R. Loscalzo. "An introduction to the application of spline functions to initial value problems." In: *Theory and Applications of spline functions*. Academic Press New York. 1969, pp. 37–64.

[141] F. R. Loscalzo and T. D. Talbot. "Spline Function Approximations for Solutions of Ordinary Differential Equations." In: *SIAM Journal on Numerical Analysis* 4.3 (1967), pp. 433–445. DOI: 10.1137/0704038.

[142] Y. Lu, A. Aldroubi, J. Gore, A. Anderson, and Z. Ding. "Improved fiber tractography with Bayesian tensor regularization." In: *NeuroImage* 31.3 (2006), pp. 1061–1074.

[143] M. Lukić and J. Beder. "Stochastic processes with sample paths in reproducing kernel Hilbert spaces." In: *Transactions of the American Mathematical Society* 353.10 (2001), pp. 3945–3969.

[144] M. Mahsereci and P. Hennig. "Probabilistic Line Searches for Stochastic Optimization." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 181–189.

[145] F. Mazzia and I. Sgura. "Numerical approximation of nonlinear BVPs by means of BVMs." In: *Applied Numerical Mathematics* 42.1 (2002), pp. 337 –352. DOI: 10.1016/S0168-9274(01)00159-3.

[146] F. Mazzia, J. R. Cash, and K. Soetaert. "Solving boundary value problems in the open source software R: package bvpSolve." In: *Opuscula Math.* 34.2 (2014), pp. 387–403. DOI: 10.7494/OpMath.2014.34.2.387.

[147] F. Mazzia and C. Magherini. *Test set for initial value problem solvers, release 2.4*. Technical Report 4. Available at http://pitagora.dm.uniba.it/~testset. Department of Mathematics, University of Bari, Italy, Feb. 2008.

[148] F. Mazzia, A. Sestini, and D. Trigiante. "B-Spline Linear Multistep Methods and their Continuous Extensions." In: *SIAM Journal on Numerical Analysis* 44.5 (2006), pp. 1954–1973. DOI: 10.1137/040614748.

[149] F. Mazzia, A. Sestini, and D. Trigiante. "The continuous extension of the B-spline linear multistep methods for {BVPs} on non-uniform meshes." In: *Applied Numerical Mathematics* 59.3–4 (2009), pp. 723 –738. DOI: 10.1016/j.apnum.2008.03.036.

[150] R. Merson. "An operational method for the study of integration processes." In: *Proc. Symp. Data Processing*. Weapon Research Establishment Salisbury, S. Australia. 1957, pp. 1–25.

[151] G. Mie. "Beweis der Integrierbarkeit gewöhnlicher Differentialgleichungssysteme nach Peano." In: *Mathematische Annalen* 43.4 (1893), pp. 553–568. DOI: 10.1007/BF01446453.

[152] W. E. Milne. "A note on the numerical integration of differential equations." In: *Journal of Research of the National Bureau of Standards* 43.6 (1949), pp. 537–542.

[153] S. Mori, B. J. Crain, V. P. Chacko, and P. C. M. Van Zijl. "Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging." In: *Annals of Neurology* 45.2 (1999), pp. 265–269. DOI: 10.1002/1531-8249(199902)45:2<265::AID-ANA21>3.0.CO;2-3.

[154] S. Mori and P. C. M. van Zijl. "Fiber tracking: principles and strategies – a technical review." In: *NMR in Biomedicine* 15.7-8 (2002), pp. 468–480. DOI: 10.1002/nbm.781.

[155] M. Moseley, Y Cohen, J Mintorovitch, L Chileuitt, H Shimizu, J Kucharczyk, M. Wendland, and P. Weinstein. "Early detection of regional cerebral ischemia in cats: comparison of diffusion and T2-weighted MRI and spectroscopy." In: *Magnetic resonance in medicine* 14.2 (1990), pp. 330–346.

[156] A. Nordsieck. "On numerical integration of ordinary differential equations." In: *Mathematics of Computation* 16.77 (1962), pp. 22–49.

[157] L. O'Donnell, S. Haker, and C.-F. Westin. "New Approaches to Estimation of White Matter Connectivity in Diffusion Tensor MRI: Elliptic PDEs and Geodesics in a Tensor-Warped Space." In: *MICCAI (1)*. 2002, pp. 459–466.

[158] A. O'Hagan. "Bayes–Hermite quadrature." In: *J of Statistical Planning and Inference* 29.3 (1991), pp. 245–260.

[159] A. O'Hagan. "Some Bayesian Numerical Analysis." In: *Bayesian Statistics* 4 (1992), pp. 345–363.

[160] A. O'Hagan. "Monte Carlo is fundamentally unsound." In: *The Statistician* (1987), pp. 247–249.

[161] B. Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. 6th ed. Springer, 2003.

[162] H. Owhadi and C. Scovel. "Toward Machine Wald." In: *Springer Handbook of Uncertainty Quantification*. Springer, 2016, pp. 1–35.

[163] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. 3rd ed. New York: McGraw-Hill, 1991.

[164] G. J. M. Parker, H. A. Haroon, and C. A. M. Wheeler-Kingshott. "A framework for a streamline-based probabilistic index of connectivity (PICo) using a structural interpretation of MRI diffusion measurements." In: *J. Magn. Reson. Imaging* 18.2 (2003), pp. 242–254.

[165] G. Parmigiani and L. Inoue. *Decision theory: principles and approaches*. Vol. 812. John Wiley & Sons, 2009.

[166] S. H. Paskov. "Average case complexity of multivariate integration for smooth functions." In: *Journal of Complexity* 9.2 (1993), pp. 291–312.

[167] G. Peano. "Démonstration de l'intégrabilité des équations différentielles ordinaires." In: *Mathematische Annalen* 37.2 (1890), pp. 182–228. ISSN: 1432-1807. DOI: 10.1007/BF01200235. URL: https://doi.org/10.1007/BF01200235.

[168] X. Pennec. "Probabilities and statistics on Riemannian manifolds: Basic tools for geometric measurements." In: *Proceedings of Nonlinear Signal and Image Processing*. 1999, pp. 194–198.

[169] L. Petzold. "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations." In: *SIAM journal on scientific and statistical computing* 4.1 (1983), pp. 136–148.

[170] E. Picard. "Mémoire sur la théorie des équations aux dérivées partielles et la méthode des approximations successives." In: *Journal de Mathématiques Pures et Appliquées* 6 (1890), pp. 145–210.

[171] H. Poincaré. *Calcul des probabilités*. Paris: Gauthier-Villars, 1896.

[172] H. Poincaré. "Sur la méthode de Bruns." In: *Comptes rendus hebdomadaires de l'Académie des sciences de Paris* 123 (1896), pp. 1224–1228.

[173] B. van der Pol. "LXXXVIII. On "relaxation-oscillations"." In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1926), pp. 978–992. DOI: 10.1080/14786442608564127.

[174] T. Rainforth. "Automating Inference, Learning, and Design." PhD thesis. Oxford University, 2017.

[175] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Machine learning of linear differential equations using Gaussian processes." In: *Journal of Computational Physics* 348 (Nov. 2017), pp. 683–693. DOI: 10.1016/j.jcp.2017.07.050.

[176] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT, 2006.

[177] H. E. Rauch, C. Striebel, and F Tung. "Maximum likelihood estimates of linear dynamic systems." In: *AIAA journal* 3.8 (1965), pp. 1445–1450.

[178] L. F. Richardson and J. A. Gaunt. "The Deferred Approach to the Limit. Part I. Single Lattice. Part II. Interpenetrating Lattices." English. In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 226 (1927), pp. 299–361.

[179] C. Runge. "Über die numerische Auflösung von Differentialgleichungen." In: *Mathematische Annalen* 46 (1895), pp. 167–178.

[180] A. Sard. *Linear approximation*. Providence, R.I.: American Mathematical Society, 1963.

[181] S. Särkkä. "Recursive Bayesian Inference on Stochastic Differential Equations." PhD thesis. Helsinki University of Technology, 2006.

[182] S. Särkkä. *Bayesian filtering and smoothing*. Cambridge University Press, 2013.

[183] S. Särkkä, A. Solin, and J. Hartikainen. "Spatiotemporal Learning via Infinite-Dimensional Bayesian Filtering and Smoothing." In: *IEEE Signal Processing Magazine* 30.4 (2013), pp. 51–61.

[184] M. Schober, A. Adam, O. Yair, S. Mazor, and S. Nowozin. "Dynamic Time-Of-Flight." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[185] M. Schober, D. Duvenaud, and P. Hennig. "Probabilistic ODE Solvers with Runge-Kutta Means." In: *Advances in Neural Information Processing Systems (NIPS)* (2014).

[186] M. Schober, N. Kasenburg, A. Feragen, P. Hennig, and S. Hauberg. "Probabilistic shortest path tractography in DTI using Gaussian Process ODE solvers." In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*. Springer, 2014.

[187] M. Schober, S. Särkkä, and P. Hennig. "A probabilistic model for the numerical solution of initial value problems." In: *Statistics and Computing* (2018). DOI: 10.1007/s11222-017-9798-7.

[188] I. J. Schoenberg. "Contributions to the Problem of Approximation Of Equidistant Data by Analytic Functions, Part B—On the Problem of Osculatory Interpolation. A Second Class of Analytic Approximation Formulae." In: *Quarterly Applied Mathematics* 4.2 (1946), pp. 112–141.

[189] I. J. Schoenberg. "Spline functions and the problem of graduation." In: *Proceedings of the National Academy of Sciences* 52 (1964), pp. 947–950.

[190] T. Schultz, H. Theisel, and H. P. Seidel. "Topological Visualization of Brain Diffusion MRI Data." In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1496–1503. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.70602.

[191] T. Schultz, A. Vilanova, R. Brecheisen, and G. Kindlmann. "Fuzzy Fibers: Uncertainty in dMRI Tractography." In: *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*. Ed. by C. D. Hansen, M. Chen, C. R. Johnson, A. E. Kaufman, and H. Hagen. London: Springer London, 2014, pp. 79–92. DOI: 10.1007/978-1-4471-6497-5_8.

[192] L. F. Shampine. "Stiffness and Non-Stiff Differential Equation Solvers." In: *Numerische Behandlung von Differentialgleichungen*. Vol. 27. Birkhäuser Basel, 1975, pp. 287–301.

[193] L. F. Shampine and K. L. Hiebert. "Detecting stiffness with the Fehlberg (4, 5) formulas." In: *Computers & Mathematics with Applications* 3.1 (1977), pp. 41 –46. DOI: 10.1016/0898-1221(77)90112-2.

[194] L. F. Shampine, H. A. Watts, and S. M. Davenport. "Solving Nonstiff Ordinary Differential Equations–The State of the Art." In: *SIAM Review* 18 (3 1976), pp. 376–411.

[195] L. F. Shampine and M. W. Reichelt. "The Matlab ODE Suite." In: *SIAM Journal on Scientific Computing* 18.1 (1997), pp. 1–22.

[196] A. J. Sherbondy, R. F. Dougherty, M. Ben-Shachar, S. Napel, and B. A. Wandell. "ConTrack: Finding the most likely pathways between brain regions using diffusion tractography." In: *Journal of Vision* 8.9 (2008), p. 15. DOI: 10.1167/8.9.15.

[197] R. D. Skeel and L. W. Jackson. "Consistency of Nordsieck Methods." In: *SIAM Journal on Numerical Analysis* 14.5 (1977), pp. 910–924.

[198] R. D. Skeel. "Equivalent forms of multistep formulas." In: *Mathematics of Computation* 33.148 (1979), pp. 1229–1250.

[199] J. Skilling. "Bayesian solution of ordinary differential equations." In: *Maximum Entropy and Bayesian Methods, Seattle* (1991).

[200] J. M. Soares, P. Marques, V. Alves, and N. Sousa. "A hitchhiker's guide to diffusion tensor imaging." In: *Frontiers in neuroscience* 7 (2013).

[201] A. Solin. "Stochastic Differential Equation Methods for Spatio-Temporal Gaussian Process Regression." PhD thesis. Helsinki, Finland: Aalto University, 2016.

[202] S. N. Sotiropoulos, S. Moeller, S. Jbabdi, J. Xu, J. L. Andersson, E. J. Auerbach, E. Yacoub, D. Feinberg, K. Setsompop, L. L. Wald, T. E. J. Behrens, K. Ugurbil, and C. Lenglet. "Effects of image reconstruction on fiber orientation mapping from multichannel diffusion MRI: reducing the noise floor using SENSE." In: *Magn Reson Med* 70.6 (2013), pp. 1682–89.

[203] O. Sporns, G. Tononi, and R. Kötter. "The Human Connectome: A Structural Description of the Human Brain." In: *PLOS Computational Biology* 1.4 (Sept. 2005). DOI: 10.1371/journal.pcbi.0010042.

[204] E. O. Stejskal and J. E. Tanner. "Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient." In: *The journal of chemical physics* 42.1 (1965), pp. 288–292.

[205] A. M. Stuart. "Inverse problems: A Bayesian perspective." In: *Acta Numerica* 19 (2010), 451–559. DOI: 10.1017/S0962492910000061.

[206] A. V. Sul'din. "Wiener measure and its applications to approximation methods. I." In: *Izvestiya Vysshikh Uchebnykh Zavedenii. Matematika* 6 (1959), pp. 145–158.

[207] A. V. Sul'din. "Wiener measure and its applications to approximation methods. II." In: *Izvestiya Vysshikh Uchebnykh Zavedenii. Matematika* 5 (1960), pp. 165–179.

[208] T. J. Sullivan. *Introduction to uncertainty quantification*. Vol. 63. Springer, 2015.

[209] P. C. Sundgren, Q. Dong, D. Gómez-Hassan, S. K. Mukherji, P. Maly, and R. Welsh. "Diffusion tensor imaging of the brain: review of clinical applications." In: *Neuroradiology* 46.5 (May 2004), pp. 339–350.

[210] O. Teymur, K. Zygalakis, and B. Calderhead. "Probabilistic Linear Multistep Methods." In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 4314–4321.

[211] Y. L. Tong. *The multivariate normal distribution*. New York: Springer, 1990.

[212] H. C. Torrey. "Bloch equations with diffusion terms." In: *Physical Review* 104.3 (1956), pp. 563–565.

[213] J.-D. Tournier, S. Mori, and A. Leemans. "Diffusion tensor imaging and beyond." In: *Magnetic Resonance in Medicine* 65.6 (2011), pp. 1532–1556. DOI: 10.1002/mrm.22924.

[214] D. S. Tuch, T. G. Reese, M. R. Wiegell, and V. J. Wedeen. "Diffusion MRI of Complex Neural Architecture." In: *Neuron* 40.5 (2003), pp. 885 –895. DOI: 10.1016/S0896-6273(03)00758-X.

[215] D. Van Essen, K. Ugurbil, E. Auerbach, D. Barch, T. Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S. Curtiss, S. Della Penna, D. Feinberg, M. Glasser, N. Harel, A. Heath, L. Larson-Prior, D. Marcus, G. Michalareas, S. Moeller, R. Oostenveld, S. Petersen, F. Prior, B. Schlaggar, S. Smith, A. Snyder, J. Xu, E. Yacoub, and W. U.-M. H. Consortium. "The Human Connectome Project: a data acquisition perspective." In: *NeuroImage* 62.4 (2012), pp. 2222–2231.

[216] G. Wahba. *Spline models for observational data*. CBMS-NSF Regional Conferences series in applied mathematics 59. SIAM, 1990.

[217] G. Wahba. "Improper priors, spline smoothing and the problem of guarding against model errors in regression." In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1978), pp. 364–372.

[218] A. Wald. *Sequential analysis*. Courier Corporation, 1973.

[219] B. A. Wandell. "Clarifying human white matter." In: *Annual review of neuroscience* 39 (2016), pp. 103–128.

[220] D. Wassermann, Y. Rathi, S. Bouix, M. Kubicki, R. Kikinis, M. E. Shenton, and C.-F. Westin. "White Matter Bundle Registration and Population Analysis Based on Gaussian Processes." In: *IPMI*. 2011, pp. 320–332.

[221] V. J. Wedeen, P. Hagmann, W.-Y. I. Tseng, T. G. Reese, and R. M. Weisskoff. "Mapping complex tissue architecture with diffusion spectrum magnetic resonance imaging." In: *Magnetic Resonance in Medicine* 54.6 (2005), pp. 1377–1386. DOI: 10.1002/mrm.20642.

[222] G. E. Wesbey, M. E. Moseley, and R. L. Ehman. "Translational Molecular Self-Diffusion in Magnetic Resonance Imaging: II. Measurement of the Self-Diffusion Coefficient." In: *Investigative Radiology* 19.6 (1984), pp. 491–498.

[223] B. Whitcher, D. S. Tuch, J. J. Wisco, A. G. Sorensen, and L. Wang. "Using the wild bootstrap to quantify uncertainty in diffusion tensor imaging." In: *Human Brain Mapping* 29.3 (2008), pp. 346–362. DOI: 10.1002/hbm.20395.

[224] V. Wiens, L. Schlaffke, T. Schmidt-Wilcke, and T. Schultz. "Visualizing uncertainty in HARDI tractography using superquadric streamtubes." In: *Proc. EG Conf. on Visualization (EuroVis) Short Papers*. 2014.

[225] Y. Zhang, M. Brady, and S. Smith. "Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm." In: *IEEE Trans Med Imaging* 20.1 (2001), pp. 45–57.