Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Masterarbeit Machine Learning

Uncertainty Propagation in Probabilistic Ordinary Differential Equation Solvers

Dingling Yao

30. August 2021

Gutachter

Prof. Dr. Philipp Hennig
Methods of Machine Learning
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Zweitgutachter

Prof. Dr. Robert Bamler
Data Science and Machine Learning
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Yao, Dingling:
Uncertainty Propagation in Probabilistic Ordinary Differential Equation Solvers
Masterarbeit Machine Learning
Eberhard Karls Universität Tübingen
Bearbeitungszeitraum: 02.03.2021 – 01.09.2021

Abstract

In real-world applications, Initial value problems (IVP) for Ordinary differential equations (ODE) often involve uncertainties in the initial values and parameters (e.g., uncertain initial biomass concentration in a bioreactor kinetics problem). Such uncertainties could lead to noticeable variation to the resulted ODE solution, especially in non-linear systems. Therefore, it is crucial to quantify the expected ODE solution and the propagated uncertainty over time. However, ODE solvers on their own fail to propagate the uncertainties properly. Hence, this thesis discusses and compares approaches to address this issue from non-probabilistic and probabilistic perspectives. In particular, we propose a novel fully probabilistic approach to efficiently compute the expected value and variance of the ODE solutions. On top of producing accurate point estimates, our fully probabilistic approach also provides estimated numerical uncertainties, which can be further utilized in a computation pipeline.

Kurzfassung

Ausgangswertprobleme (IVPs) von gewöhnlichen Differentialgleichungen (ODEs) in realen Anwendungen beinhalten oft Unsicherheiten bei Ausgangswerten und Parametern (z.B. unsichere anfängliche Biomasse-Konzentration in einem Bioreaktor-Kinetikproblem). Solche Unsicherheiten können zu merklichen Abweichungen bei der resultierenden ODE-Lösung führen, insbesondere bei nichtlinearen Systemen. Daher ist es von entscheidender Bedeutung, die erwartete ODE-Lösung und die propagierte Unsicherheit über die Zeit, zu quantifizieren. ODE-Löser allein sind jedoch nicht in der Lage, die Unsicherheiten angemessen auszuwerten. In dieser Arbeit werden daher Ansätze zur Lösung dieses Problems aus nicht-probabilistischer und probabilistischer Sicht diskutiert und verglichen. Darüber hinaus wird ein neuer, vollständig probabilistischer Ansatz zur effizienten Berechnung von Erwartungswert und Varianz der ODE-Lösungen vorgestellt. Dieser Ansatz liefert nicht nur genaue Punktschätzungen, sondern auch die numerischen Unsicherheiten, welche in darauf folgenden Berechnungen weiterverwendet werden können.

Contents

Chapter 1

Introduction

A variety of dynamical systems modeled by ordinary differential equations involve uncertainties in initial values and parameters. In practical mechanical and structural systems, uncertainties are inherent in loads, parameters, material properties, fraction tolerance, boundary conditions and geometric dimensions, due to the complexity of real-world problems; such uncertainties could result in noticeable variation in the ODE solutions, especially in non-linear system (Wu *et al.*, 2013). An example of varying ODE solutions of a non-linear system is shown in Figure 1.1. Besides, it is noted that smaller uncertainties in parameters could be propagated in

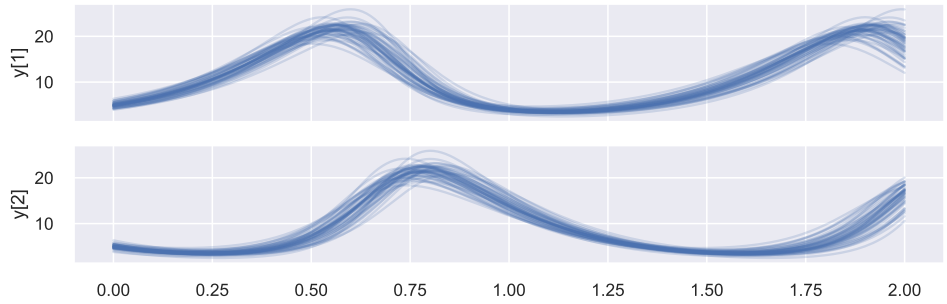


Figure 1.1: *ODE solution samples:* Varying ODE solutions of Lotka-Volterra system w.r.t. different initial values, which are sampled from $\mathcal{N}((5,5)^\intercal, 0.3 \cdot I_2)$ with I_2 denoting the identity matrix.

the dynamical system and lead to comparably large uncertainties in the dynamic response (Chen, 2004; Cheng and Sandu, 2009; Hanss, 2002; Wu *et al.*, 2013). Therefore, of great interest are the expected ODE solution and uncertainty propagation over time. Since the approaches for uncertain initial values are directly applicable to the uncertain parameters, we will only discuss the former. Consider an Initial Value Problem (IVP) over the timespan $\mathbb{T} := [0, \mathcal{T}]$:

$$\dot{y}(t) = \frac{dy}{dt} = f_\theta(y(t), t) \quad t \in [0, \mathcal{T}],\, y(0) = y_0, \tag{1.1}$$

where $f : \mathbb{R}^d \times \mathbb{T} \to \mathbb{R}^d$ denotes the vector field, which is a function of the time $t \in \mathbb{T}$ and the current state $y(t) \in \mathbb{R}^d$. The vector field f is often characterized by

a set of parameter(s) $\theta \in \Theta$. In our setting, the initial value $y_0 \in \mathbb{R}^d$ is not a fixed value but a random variable with the distribution $p(y_0)$:

$$y(0) = y_0 \sim p(y_0),$$

which leads to the objects of interest:

- Expected ODE solution:

$$\mathbb{E}_{y_0}[y(y_0, t)] = \int y(y_0, t) p(y_0) dy_0, \tag{1.2}$$

- Variance of ODE solutions:

$$\mathbb{V}_{y_0}[y(y_0, t)] = \int (y(y_0, t) - \mathbb{E}_{y_0}[y(y_0, t)])^2 p(y_0) dy_0. \tag{1.3}$$

We use $y(y_0, t)$ to explicitly denote the ODE solution regarding the initial value y_0. Also, in this thesis, we only consider the variation of the ODE solutions of each dimension. Therefore, the variance is computed element-wise.

A particular family of probabilistic numerical ODE solvers treats IVP as a Gauss-Markov process regression and utilizes Bayesian Filtering/Smoothing to compute the solution efficiently (Schober *et al.*, 2019; Tronarp *et al.*, 2019, 2021). In such filtering-based solvers, the initial value $y(0)$ is assumed to be Gaussian distributed:

$$y(0) \sim \mathcal{N}(\mu_0, \Sigma_0).$$

Based on this fact, one would expect that such probabilistic ODE solvers could compute $\mathbb{E}_{y_0}[y(y_0, t)]$ and $\mathbb{V}_{y_0}[y(y_0, t)]$. To clarify this, we take a time-invariant linear ODE as an example:

$$\dot{y}(t) = f(y(y_0, t), t) = a \cdot y(y_0, t) + b, \quad y(0) = y_0 \sim \mathcal{N}(m_0, \sigma_0^2), \tag{1.4}$$

where f denotes the vector field, a, b the parameters, and $\mathcal{N}(m_0, \sigma_0^2)$ the Gaussian distribution over the initial value y_0 with mean m_0 and variance σ_0^2. As known, this ODE has an analytical solution in the form:

$$y(y_0, t) = \exp(at) \cdot (y_0 + {}^{b}/_{a}) - {}^{b}/_{a}, \tag{1.5}$$

which is globally linear in y_0. Therefore, the expected ODE solution and the variance are also available in closed-form, which are given below:

$$\mathbb{E}_{y_0}[y(y_0, t)] = \exp(at) \cdot \mathbb{E}[y_0] = \exp(at) \cdot (m_0 + {}^{b}/_{a}) - {}^{b}/_{a}, \tag{1.6a}$$
$$\mathbb{V}_{y_0}[y(y_0, t)] = \exp(at)^2 \cdot \mathbb{V}[y_0] = \exp(2at) \cdot \sigma_0^2. \tag{1.6b}$$

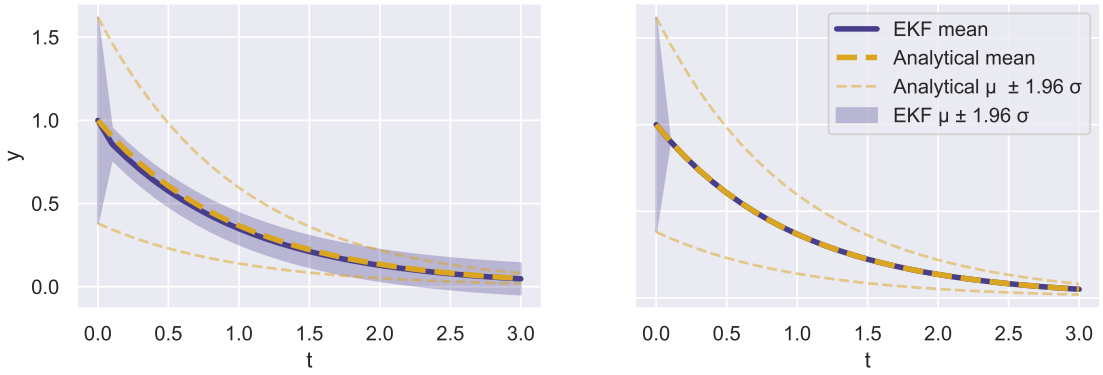


Figure 1.2: *PN posterior for the expected ODE solution and propagated variance.* —— shows the posterior returned by EKF and - - - depicts the analytical solution. Implementation used ProbNum. *Left*: EKF with constant step size 0.05. *Right:* EKF with adaptive step size (Bosch *et al.*, 2021).

Figure 1.2 depicts the posterior returned by *Extended Kalman Filtering* (EKF) on a linear ODE with $a = 1, b = 0, m_0 = 1$, and $\sigma_0^2 = 0.1$. The result is generated using `ODEFilter` from ProbNum[1]. As shown in the right subfigure, when *adaptive step size* is employed, the variance vanishes after several time steps, although the posterior mean coincides with the analytical solution. On the other hand, when we fixed the step size with $h = 0.05$, the uncertainty will not vanish but fail to capture the true variation. Therefore, we need to develop new methods to solve this problem, which loops back to the main purpose of this thesis.

Apart from filtering-based ODE solvers, other works also aimed to solve this problem: Lin and Stadtherr (2006) utilized the interval Taylor Series (ITS) to find the "unique" validated solution (a tight enclosure) of the given initial value problem, where the initial values and parameters are from the corresponding enclosure of uncertainties. Wu *et al.* (2013) proposed a new interval analysis method for non-linear ODE systems with bounded uncertain parameters, where the set of ODEs with uncertain parameters can be transformed into a new set of ODEs with deterministic parameters, which can be directly solved by a classic numerical ODE solver. More recently, Gerlach *et al.* (2020) proposed a systematic approach to deal with ODE problems involved with uncertainties, using non-probabilistic ODE solvers along with classic numerical integration methods. However, these approaches neglected the computational uncertainty around the outputs. Being aware of the uncertainty is crucial in many fields of process engineering, especially regarding safe learning problems. Hence, *probabilistic numerics* (PN) (Hennig *et al.*, 2015; Oates and Sullivan, 2019) aims to generally quantify the uncertainties in numerical algorithms by assigning probability measures to numerical objects. Thus, numerical algorithms

[1]ProbNum: `https://github.com/probabilistic-numerics`

are turned into inference problems, which return a posterior distribution over the quantity of interest.

Since numerical uncertainties are involved in solving ODEs and computing integrals, we aim to not only find the point estimates for the expected ODE solution and the propagated variance, but also efficiently compute the posteriors over theses two quantities:

$$\mathbb{E}_{y_0 \sim p(y_0)}[y(t, y_0)] \sim \mathcal{N}(\mu_{\mathbb{E}}(t), \Sigma_{\mathbb{E}}(t)), \tag{1.7a}$$

$$\mathbb{V}_{y_0 \sim p(y_0)}[y(t, y_0)] \sim \mathcal{N}(\mu_{\mathbb{V}}(t), \Sigma_{\mathbb{V}}(t)), \tag{1.7b}$$

where $\mu_{\mathbb{E}}(t), \mu_{\mathbb{V}}(t)$ denote the posterior means and $\Sigma_{\mathbb{E}}(t), \Sigma_{\mathbb{V}}(t)$ quantify the numerical uncertainties produced during the computation.

In this thesis, we firstly offer a brief overview of the theoretical background related to this problem. In Chapter 3, we discuss existing approaches and propose a novel, fully probabilistic approach to calibrate numerical uncertainties, on top of giving accurate solutions. Afterward, Chapter 4 evaluates and compares the proposed approaches from both perspectives of the accuracy of the returned point estimates and uncertainty calibration, based on various dynamical systems. In Chapter 5, we conclude the contribution and limitation of this thesis and outline future work.

Chapter 2

Foundations

This chapter begins with a brief review of the necessary mathematical background. After that, we introduce the ODE solvers and the integration methods from both non-probabilistic and probabilistic perspectives and discuss the connections between both families of methods based on the current literature.

2.1 Mathematical Prerequisites

This section reviews the mathematical background related to the derivation of the approaches in Chapter 3. We start with the calculation rules of Gaussian distributions, followed by the introduction to the Kronecker product and some of its properties.

2.1.1 Properties of Gaussian distributions

Gaussian distributions are of great importance in probabilistic machine learning because of their convenient mathematical properties: Gaussian distributions are closed under multiplication, linear projection, and conditioning. In summary, Gaussian distributions provide the linear algebra of inference. Let $\mathcal{N}$ denote the Gaussian distribution, the properties are summarized below (Petersen *et al.*, 2008):

- Product of two Gaussian distributions is a scaled Gaussian:

$$\mathcal{N}(x; a, A)\mathcal{N}(x; b, B) = \mathcal{N}(x; c, C)\mathcal{N}(a; b, A + B), \tag{2.1}$$

  where $C = (A^{-1} + B^{-1})^{-1}, c = C(A^{-1}a + B^{-1}b)$.

- Given that $x \sim \mathcal{N}(x; \mu, \Sigma)$, then the linear map of x is also Gaussian distributed:

$$Ax \sim \mathcal{N}(Ax; A\mu, A\Sigma A^{\intercal}), \tag{2.2}$$

  and the marginalisation can be treated as a special case of linear projection with

$$A = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}.$$

- Given that

$$p(x) = \mathcal{N}(x; \mu, \Sigma), \tag{2.3a}$$
$$p(y|x) = \mathcal{N}(y; Ax + b, \Lambda), \tag{2.3b}$$

  then the marginal distribution of y yields

$$p(y) = \mathcal{N}(y; A\mu + b, A\Sigma A^\intercal + \Lambda). \tag{2.4}$$

2.1.2 Kronecker product

The Kronecker product is an operation on two matrices of arbitrary sizes, often denoted by $\otimes$. Kronecker product can be treated as a generalization of the outer-product of vectors onto matrices. Formally, the Kronecker product between two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$ is a $mp \times nq$ block matrix (Horn and Johnson, 1991, Section 4.2):

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} \tag{2.5}$$

Kronecker products possess many convenient properties. The mostly used properties in this thesis are:

- The mixed-product property: If the matrices A, B, C, D are of such shapes that AC and BD are well-defined matrix products, then:

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD). \tag{2.6}$$

- The inverse of a Kronecker product: It follows that $A \otimes B$ is invertible *if and only if* A and B are invertible, in which case the inverse is given by:

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \tag{2.7}$$

2.2 ODE Solvers

Numerical ODE solvers can be categorized into two families: Classic (or non-probabilistic) solvers and probabilistic solvers. While the non-probabilistic numerical ODE solvers compute numerical approximations of an ODE solution without considering the numerical uncertainties, the probabilistic numerical ODE solvers treat solving ODE as an inference problem and assign a probability measure to the output. In this section, we give an elaborate explanation of both solver families.

2.2.1 Non-probabilistic numerical ODE Solvers

Non-probabilistic numerical ODE solvers approximates an ODE solution iteratively. Most non-probabilistic numerical ODE solvers for first-order IVPs fall into one of three categories: Taylor series methods, Runge–Kutta methods, and Linear multistep methods. Most of the content in this subsection is adapted from (Griffiths and Higham, 2010). Hence, we do not cite repeatedly.

Taylor series method

The idea of Taylor series method is to predict the ODE solution in the near future based on the current ODE solution and its derivative information. We use TS(q) to denote the Taylor series method of the order q. Applying TS(q), we can compute the ODE solution as follows:

$$y(t+h) = \sum_{k=0}^{q} \frac{h^k y^{(k)}(t)}{k!} + O(h^{q+1}), \tag{2.8}$$

where h denotes the step size, $y^{(k)}(t)$ represents the k-th derivative of the ODE solution y at the time point t. According to this definition, the Euler's method:

$$y(t+h) = y(t) + hy'(t) \tag{2.9}$$

can be treated as the first-order Taylor series method, i.e., TS(1).

As discussed in Griffiths and Higham (2010), increasing the order of Taylor series method could improve the performance of the solver drastically. However, Taylor series methods with an order larger than two are not widely used in complicated systems due to the requirement of higher-order differentiability.

Runge–Kutta method

Runge–Kutta (RK) methods are one-step multi-stage methods: In each step, RK computes the weighted average of the slopes (evaluations of vector field f) of several nearby points (stages) and then predicts the solution at t_{n+1} using this averaged slope. According to this definition, Euler's method can be seen as the special case of RK methods with only one stage. The general RK method is summarized as follows:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i, \tag{2.10}$$

where the k_i are computed from the function f:

$$k_i = f(t_i + c_i h, x_n + h \sum_{j=1}^{s} a_{i,j} k_j), \quad i = 1:s. \tag{2.11}$$

Linear multietstep method

The Linear multistep methods (LLMs) share the same idea as the Taylor series methods but employing some mathematical tricks to avoid the exact computation of higher-order derivatives. Instead, LLMs approximates the higher-order derivatives based on the "history", with the constraint that the approximation error will not exceed the reminder error $O(h^q)$. Formally, the k-step LLMs can be summarized as:

$$y_{n+k} + \alpha_{k-1} y_{n+k-1} + ... + \alpha_0 y_n = h(\beta_k f_{n+k} + \beta_{k-1} f_{n+k-1} + ... + \beta_0 f_n), \quad (2.12)$$

where f_n denotes the corresponding vector field of y_n. $\{\alpha_i\}$s and $\{\beta_i\}$s are coefficients. The k-step LLM is considered *explicit* if $\beta_k = 0$; otherwise *implicit*. An overview of Linear multistep methods is given in Table 2.1:

Table 2.1: Summary of LLM methods, adapted from Griffiths and Higham (2010).

k	p	Method	Name
1	1	$x_{n+1} - x_n = hf_n$	Euler
1	1	$x_{n+1} - x_n = hf_{n+1}$	Backward Euler
1	2	$x_{n+1} - x_n = {}^1\!/_2 h(f_{n+1} + f_n)$	trapezoidal
2	2	$x_{n+2} - x_{n+1} = {}^1\!/_2 h(3f_{n+1} - f_n)$	two-step Adams-Bashforth
2	2	$x_{n+2} - x_{n+1} = {}^1\!/_{12} h(5f_{n+1} + 8f_{n+1} - f_n)$	twp-step Adams-Moulton
2	4	$x_{n+2} - x_n = {}^1\!/_3 h(f_{n+2} + 4f_{n+1} + f_n)$	Simpson's rule
2	3	$x_{n+2} + 4x_{n+1} - 5x_n = h(4f_{n+1} + 2f_n)$	Dahlquist

2.2.2 Probabilistic numerical ODE solvers

Probabilistic numerical ODE solvers treat IVP as an inference problem. One particular family among the probabilistic solvers applies Gaussian process regression (GPR) to solve the IVP (Hennig and Hauberg, 2014; Schober *et al.*, 2014). Since inference on GP is of cubic complexity, Kersting and Hennig (2016), Schober *et al.* (2019) sped up this process by assigning a Gauss-Markov prior to the state-space model; thus, the GPR can be reformulated as a Bayesian filtering/smoothing problem, which can be solved efficiently in linear time (Särkkä, 2013). Such filtering-based ODE solvers are proven to be able to converge with polynomial order (Kersting *et al.*, 2020; Tronarp *et al.*, 2021).

Another class of probabilistic numerical ODE solvers (see Chkrebtii *et al.*, 2016; Teymur *et al.*, 2016; Conrad *et al.*, 2017; Lie *et al.*, 2019; Teymur *et al.*, 2018; Abdulle and Garegnani, 2020) represents the probability distribution over the ODE solutions with a set of sampling paths. Hence, this class of solvers is capable of representing more expressive, non-Gaussian posteriors. An intersection of these

two lines of work is provided in Tronarp *et al.* (2019), where a particle ODE filter is proposed as a sampling-based filtering method.

While the above-mentioned sampling-based solvers can return more statistically expressive posteriors, the algorithm suffers from high computational cost. By contrast, filtering-based solvers enjoy fast computation of Bayesian filtering/smoothing and scale linearly. Since the primary purpose of this thesis is to compute the posteriors of interest efficiently, we will work with the filtering-based ODE solvers and refer to these solvers as PN ODE solvers. A brief review of such probabilistic methods is given below.

The inference problem

We assume the q-times integrated Wiener Process prior $\boldsymbol{y}(t)$, where $\boldsymbol{y}(t)$ represents a state-space model:

$$\boldsymbol{y}(t) = \left[(y^{(0)}(t))^\intercal, (y^{(1)}(t))^\intercal, ..., (y^{(q)}(t))^\intercal\right]^\intercal. \tag{2.13}$$

The prior $\boldsymbol{y}(t)$ satisfies the following stochastic differential equation:

$$y(0) \sim \mathcal{N}(\mu_0, \Sigma_0), \tag{2.14a}$$

$$dy^{(i)}(t) = y^{(i+1)}dt, \tag{2.14b}$$

$$dy^{(1)}(t) = \Gamma^{1/2}dW(t), \tag{2.14c}$$

where $W(t)$ is a standard wiener process, $\Gamma^{1/2}$ is the diffusion terms, which is the symmetric square root of some positive semi-definite matrix $\Gamma \in \mathbb{R}^{d\times d}$ with d denoting the dimension of $y^{(0)}(t)$ (Bosch *et al.*, 2021).

$$\dot{y}(t) - f(y(t), t) = 0 \tag{2.15}$$

represents the inherent condition that comes with an ODE. Based on this condition, we can construct the observation model as follows:

$$h(\boldsymbol{y}(t), t) := \boldsymbol{y}^{(1)}(t) - f(\boldsymbol{y}^{(0)}(t), t). \tag{2.16}$$

The realization of $h(\boldsymbol{y}(t), t))$ are zeros at the true ODE solution, which is known *a priori*. Under this circumstance, the inference problem can be represented as

$$p(\boldsymbol{y}(t)|\{z_n\}_{n=1}^N), \tag{2.17}$$

where $z_n = 0$ are the realizations of the measurement $h(\boldsymbol{y}(t), t)$ on a given time grid $\{t_n\}_{n=1}^N$ (Tronarp *et al.*, 2019; Bosch *et al.*, 2021).

The inference scheme: Extended Kalman Filter

Since inference problem with a non-linear measurement function is analytically intractable, we consider the approximated Bayesian inference problem using *Extended Kalman Filter* (EKF), which linearizes the measurement model using Taylor-series expansion and assumes Gaussianity of the state $\boldsymbol{y}(t)$ and the observation $\{z_n\}_{n=1}^N$ (see Särkkä, 2013, Section 5.2). This gives rise to Gaussian approximations of the prediction, filtering, and smoothing densities, as well as the likelihood:

$$p(\boldsymbol{y}(t_n)|\{z_i\}_{i=1}^{n-1}) \approx \mathcal{N}(\mu_n^P, \Sigma_n^P), \tag{2.18a}$$
$$p(\boldsymbol{y}(t_n)|\{z_i\}_{i=1}^{n}) \approx \mathcal{N}(\mu_n^F, \Sigma_n^F), \tag{2.18b}$$
$$p(\boldsymbol{y}(t_n)|\{z_i\}_{i=1}^{N}) \approx \mathcal{N}(\mu_n^S, \Sigma_n^S), \tag{2.18c}$$
$$p(z_n|\{z_i\}_{i=1}^{n-1}) \approx \mathcal{N}(\hat{z}_n, S_n). \tag{2.18d}$$

The filtering density can be efficiently computed by processing the *predict* and *update* steps iteratively, as described in Särkkä (2013):

- Prediction:

$$\mu_n^P = A(h_{n-1})\mu_{n-1}^F, \tag{2.19a}$$
$$\Sigma_n^P = A(h_{n-1})\Sigma_{n-1}^F A(h_{n-1})^\intercal + Q(h_{n-1}), \tag{2.19b}$$

- Update:

$$\hat{z}_n = E_1\mu_n^P - f(E_0\mu_n^P, t_n), \tag{2.20a}$$
$$S_n = H_n\Sigma_n^P H_n^\intercal, \tag{2.20b}$$
$$K_n = \Sigma_n^P H_n^\intercal S_n^{-1}, \tag{2.20c}$$
$$\mu_n^F = \mu_n^P + K_n(z_n - \hat{z}_n), \tag{2.20d}$$
$$\Sigma_n^F = \Sigma_n^P - K_n S_n K_n^\intercal, \tag{2.20e}$$

where H_n can be either $H_n := E_1$ for a zero-th order approximation, or $H_n := E_1 - J_f(t_n, E_0\mu_n)E_0$ for a first order approximation, with $E_0 := e_0^\intercal \otimes I$ and $E_1 := e_1^\intercal \otimes I$ (Tronarp *et al.*, 2019).

The approximated full posterior conditioned on all of the measurements (see Equation 2.17) can be computed by the *Extended Kalman Smoother* (EKS), through a backward recursion:

$$G_n = \Sigma_n^F A(h_n)(\Sigma_{n+1}^P)^{-1}, \tag{2.21a}$$
$$\mu_n^S = \mu_n^F + G_n(\mu_{n+1}^S - \mu_{n+1}^P), \tag{2.21b}$$
$$\Sigma_n^S = \Sigma_n^F - G_n(\Sigma_{n+1}^S - \Sigma_{n+1}^P)G_n^\intercal. \tag{2.21c}$$

Connection to non-probabilistic numerical ODE solvers

Although several probabilistic models are proposed to solve the initial value problem (e.g. Skilling, 1992; Chkrebtii *et al.*, 2016), there lacks analytical guarantees about the desirable output of the probabilistic solvers. Since no clear connection was established to the non-probabilistic algorithms, the extensive numerical analysis on the IVP cannot be applied in the probabilistic case. Hence, considerable interest has emerged in the connection and equivalence between the point estimates returned by the non-probabilistic numerical solvers and the posterior mean given by the probabilistic numerical solver. After Hennig and Hauberg (2014) pointed out that the linear extrapolation steps in Runge–Kutta methods can be emulated by a Gaussian process regression, Schober *et al.* (2014) firstly constructed probabilistic ODE solver whose posterior mean exactly matches the Runge–Kutta mean, of the order 1, 2, and 3 respectively. Teymur *et al.* (2016) extended the Adams-Bashforth and Adams-Moulton family of the linear multistep methods to their probabilistic versions, with the probabilistic formulation coinciding with the classical deterministic method in the limit. Later, Schober *et al.* (2019) conducted a more exhaustive study between ODE solvers and the probabilistic regression methods. They presented a new class of probabilistic solvers and showed whose posterior mean can be interpreted as a multistep method in Nordsieck representation. Also, a summary of evaluation on the existing probabilistic solvers according to a list of desiderata can be found in Schober *et al.* (2019, Table 1).

2.3 Integration Methods

Quadrature, also termed integration, is a historical mathematical term that means "finding a square equal in area to a given area" (Merriam-Webster, nd). Integration arises in many fields related to machine learning. Especially in the field of probabilistic inference, where one needs to compute the evidence Z to infer some parameter θ:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{Z},$$

where

$$Z = \int p(\mathcal{D}|\theta)p(\theta)d\theta,$$

and $\mathcal{D}$ denotes the *data* given to the model. Unfortunately, computing the analytical solution by finding the antiderivative is almost impossible due to the high complexity of the integrand function. In most cases, we only know about the function at a finite number of locations, possibly with some noise. Under this circumstance, one has to approximate the integral. There are two families of integral approximation methods: non-probabilistic numerical integration methods or probabilistic numerical integration, also called Bayesian Quadrature or Cubature.

While non-probabilistic numerical methods pay more attention to the point estimate of the desired integral, probabilistic numerical integration methods take the numerical uncertainty into consideration and aim to provide a reliable error bar on top of accurate integral approximations. This section introduces both numerical quadrature rules and the probabilistic numerical integration methods and discusses the connection between both families of approaches.

2.3.1 Non-probabilistic numerical integration

Traditional non-probabilistic numerical integration approximates an integral using numerical techniques. The term "numerical quadrature" (abbrev. Quadrature) was often considered as a synonym of numerical integration until Ueberhuber (1997) explicitly used the word "Quadrature" to refer to the uni-variate integral and "Cubature" as multidimensional integral. In general, the idea of numerical integration is to combine evaluations of the integrand function to approximate the true integral. Since quadrature can be treated as a "one-dimensional" cubature, in the scope of this thesis, we use "cubature" to denote both one-dimensional and multidimensional integration.

Quadrature

The numerical methods for one-dimensional integrals can be roughly separated into two families, distinguished by whether the integration points are equally spaced or not. The choice of the interval space is determined by our prior knowledge about the integrand function. In most cases, if the function is known analytically (which means we can evaluate it at any location), methods with varying space would be a better choice (e.g., Gaussian Quadrature) because they are typically more accurate. Nonetheless, if the integrand function values are only given at equally spaced locations, then the Newton-Coates formulas would be an adequate solution. Most content from this subsection refers (Press *et al.*, 1992), so we do not cite repeatedly.

Newton-Cotes formulas The general idea of the Newton-Coates algorithm is to integrate simple functions piece-wise to approximate the true area. These simple functions are termed interpolation functions. A common choice of such interpolating function is polynomials. The Newton-Coates formulas can be constructed with many different explicit rules, depending on the polynomial degree. Since polynomials with a high degree often lack stability, low degree polynomials like linear functions are more widely used. This section introduces quadrature rules w.r.t. various degrees of polynomials and their convergence rates, respectively.

Assume $f(x)$ is continuous over $[a, b]$. Let $n \in \mathbb{N}$ and $h = {}^{b-a}/_{n}$. Divide the interval $[a, b]$ into n sub-intervals with length h and endpoints $\{x_1, ..., x_n\}$.

Midpoint rule Midpoint rule (or rectangle rule) corresponds to the Newton-Cotes formulas with a constant interpolating function. The integral can then be approximated as:

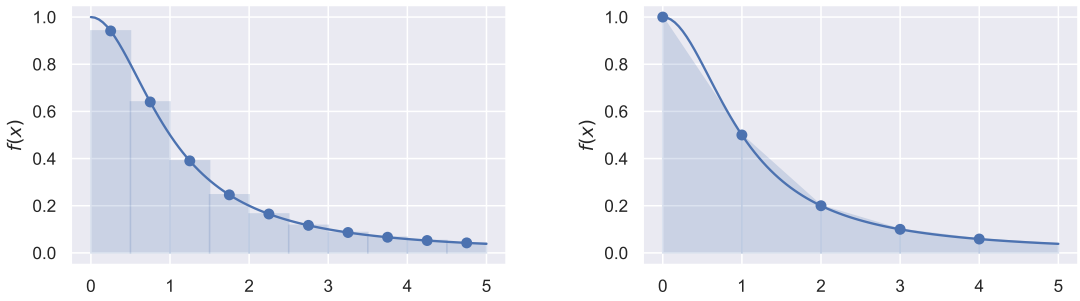
$$\int_{x_1}^{x_2} f(x)dx = hf\left(\frac{x_1 + x_2}{2}\right) + \mathcal{O}(h^3 f''). \tag{2.22}$$

Trapezoidal rule Trapezoidal rule employs the linear function (polynomial with degree 1), the closed formula is shown below:

$$\int_{x_1}^{x_2} f(x)dx = h\left(\frac{1}{2}f(x_1) + \frac{1}{2}f(x_2)\right) + \mathcal{O}(h^3 f''). \tag{2.23}$$

Simpson's rule Simpson's rule is based on a polynomial of order 2, which can be summarized as:

$$\int_{x_1}^{x_3} f(x)dx = h\left(\frac{1}{3}f(x_1) + \frac{4}{3}f(x_2) + \frac{1}{3}f(x_3)\right) + \mathcal{O}(h^5 f^{(4)}). \tag{2.24}$$



(a) Midpoint rule **(b)** Trapezoidal rule

Figure 2.1: Midpoint and Trapezoidal rules for the function $f(x) = 1/x^2+1$ on $[0, 5]$

Gaussian quadrature Gaussian Quadrature provides freedom not only in constructing weight coefficients for the function evaluations but also in choosing the locations to be evaluated. In general, the computation of Gaussian quadrature rules comprises of two distinct phases (Press *et al.*, 1992):

1. Compute the orthogonal polynomials $p_0(x), ..., p_N(x)$.
2. Determine the zeros of $p_N(x)$ and compute the associated weights.

The most commonly used Gaussian quadrature rules are listed in (Press *et al.*, 1992, Section 4.5), including the *Gauss-Legendre, Gauss-chebyshev, Gauss-Lauguerre* and *Gauss-Hermite*.

Cubature

Cubature (or multidimensional integration) could be solved iteratively by applying Fubini's theorem; however, the required number of function evaluations grows exponentially with the dimension (Hinrichs *et al.*, 2014). Several approaches are available to alleviate this issue:

Monte Carlo As mentioned before, Monte Carlo methods are the most simple and robust option. In low dimensional case, MC might not be the best option due to the slow convergence rate. However, when dealing with high dimensional problems, Monte Carlo might be the only remaining choice, since MC converges dimension-independently, with a rate of $\mathcal{O}(N^{-1/2})$. Acceleration of MC can be obtained by using *quasi*-random sequences, this approach is termed *quasi*-Monte Carlo with a convergence rate of $\mathcal{O}(\log(N)^k N^{-1})$ (Caflisch, 1998).

Sparse grid Sparse grids, firstly proposed by Sergey A. Smolyak, approximate multidimensional integrals based on individual quadrature rules and sparse tensor product construction. One of the most commonly used sparse grids methods is Smolyak's rule, which recursively computes the high dimensional integral. A summary of sparse grid methods can be found in (Zenger, 1991; Garcke, 2013).

Adaptive quadrature/cubature

Apart from the basic integration rules mentioned above, some algorithms compute the integrals of interest adaptively (Rice, 1975; de Boor and Rice, 1979). The *h-adaptive* integration recursively refines subdivisions of the region of integration until the estimated error becomes lower than the pre-defined tolerance value. In each subinterval of the region of integration, the same static integration algorithms (e.g., the Simpson's rule) are applied to approximate the integrand function (Genz and Malik, 1980; Berntsen *et al.*, 1991). On the other hand, the *p-adaptive* integration repeatedly doubles the degree of the static quadrature rules until convergence is achieved. This algorithm is based on a tensor product of *Clenshaw–Curtis quadrature*. While *h-adaptive* integration is well-suited for functions that have localized sharp features. *p*-adaptive integration performs better in integrating smooth functions (infinitely differentiable, ideally analytic) in low dimensions (ideally 1 or 2) (Johnson, 2020).

2.3.2 Probabilistic numerical integration

This subsection reviews the probabilistic numerical integration method: Bayesian Cubature, followed by a non-exhaustive discussion about connection between the

non-probabilistic numerical integration methods and the corresponding Bayesian approaches.

Bayesian cubature

As stated before, numerical methods for integration comes with unknown uncertainties. The PN solution to integration is Bayesian Cubature (BC) (Diaconis (1988a), O'Hagan (1991), Kennedy (1998), Rasmussen and Ghahramani (2003), Minka (2000), Briol *et al.* (2019)). Assume our target integral $F = \int_\Omega f(x)\nu(x)$ with $f : \Omega \to \mathbb{R}$ is analytically intractable, BC approximates the integrand function f with a Gaussian process $\mathcal{GP}(f; m, k)$ with a mean function $m : \Omega \to \mathbb{R}$ and a covariance kernel $k : \Omega \times \Omega \to \mathbb{R}$. GP is a stochastic process such that every finite collection of those random variables $f_{GP}(x_1, ..., x_n)$ for any $n \in \mathbb{N}$ is Gaussian distributed.

Mathematically, we can formulate BC as follows: Without loss of generality, we model the function of interest f with a zero-mean Gaussian process, $\mathcal{GP}(f; 0, k)$. Assume a *noise-free* setting where the evaluation of function f is exact. After evaluated the function f at locations $X := \{x_i\}_{i=1}^N$, we obtain $f_X = \{f(x_i)\}_{i=1}^N$, and the joint distribution of f_X and the integral F yields:

$$p\left(\begin{bmatrix} f_X \\ F \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{XX} & I_K \\ I_K^\intercal & I_K^2 \end{bmatrix}\right), \tag{2.25}$$

where

$$\begin{aligned} \text{Kernel mean} \ : \quad & I_K(x) = \int_\Omega k(x, x')d\nu(x'), \\ \text{Initial error} \ : \quad & I_K^2 = \int_\Omega \int_\Omega k(x, x')d\nu(x)d\nu(x'). \end{aligned} \tag{2.26}$$

After applying the Gaussian conditioning rules, the posterior can be summarized as:

$$p(F|f_X) = \mathcal{N}(F; m_F, V_F), \tag{2.27}$$

where

$$m_F = I_K^\intercal K_{XX}^{-1} f_X, \tag{2.28a}$$

$$V_F = I_K^2 - I_K^\intercal K_{XX}^{-1} I_K. \tag{2.28b}$$

BC requires that the kernel mean and the initial errors (see Equation 2.26) are analytically tractable. However, not all kernel-distribution pairs satisfy this condition. Table 2.2 lists some of the kernel-distribution pairs which provides closed-form kernel mean and initial values.

For other distributions which are not listed in Table 2.2, it is possible to apply the importance re-weighting trick: $\frac{\nu}{q} \cdot q$ (where q offers closed-form solution given the pre-defined kernel in a particular problem) such that the integration against any probability measure becomes feasible (Gunter *et al.*, 2014).

Table 2.2: A non-exhaustive list of kernel-distribution pair (k, ν) which provide closed form for the kernel mean and initial errors (see Equation 2.26). "TP" stands for tensor product of one-dimensional kernels. Adapted from Briol *et al.* (2019).

$\mathcal{X}$	ν	k	Reference
$[0,1]^d$	Unif($\mathcal{X}$)	Wendland TP	Oates *et al.* (2016)
$[0,1]^d$	Unif($\mathcal{X}$)	Matérn Weighted TP	Briol *et al.* (2019)
$[0,1]^d$	Unif($\mathcal{X}$)	Exponential Quadratic	Use of error function
$\mathbb{R}^d$	Mixt. of Gauss.	Exponential Quadratic	Kennedy (1998)
$\mathbb{S}^d$	Unif($\mathcal{X}$)	Gegenbauer	Briol *et al.* (2019)
Arbitrary	Unif($\mathcal{X}$)/Mixt. of Gauss.	Trigonometric	Integration by parts
Arbitrary	Unif($\mathcal{X}$)	Splines	Wahba (1990)
Arbitrary	Known moments	Polynomial TP	Briol *et al.* (2015)
Arbitrary	Known $\partial \log \nu(x)$	Gradient-based Kernel	Oates *et al.* (2019, 2017)

Connection to numerical integration methods

The connection between the numerical integration method and the Bayesian integration rules has been discussed for decades (see Larkin, 1970; Diaconis, 1988b; O'Hagan, 1991; Minka, 2000; Särkkä *et al.*, 2016; Prüher and Särkkä, 2016). Diaconis (1988b) showed that some non-probabilistic quadrature rules coincide with the posterior mean from BQ when employing the (integrated) Wiener process kernel. Särkkä *et al.* (2016) provides the relation between the quadrature/cubature rules used in the Kalman Filter in the nonlinear case and Bayesian integration rules with certain GP kernel of polynomial form. More recently, Karvonen and Särkkä (2017) investigated the equivalence of classical polynomial-based Quadrature and Bayesian Quadrature rules with suitable covariance kernels. The authors showed that any classical quadrature rule can be interpreted as a Bayesian quadrature rule if the kernel is an orthogonal polynomial kernel of the form given in Karvonen and Särkkä (2017, Eq 5.) with a suitable degree p. Analogously, correspondence is found between the polynomial interpolants and the kernel interpolant for the increasing flat stationary kernels (in the limit), as summarized in Karvonen (2019, Sec 5.4).

Chapter 3

Approaches

Figure 3.1 outlines the general work flow to compute the expected value and the variation of ODE solutions, w.r.t. an uncertain initial value. It is observable that there are two stages where one can decide whether to take numerical uncertainties into consideration or not: Firstly, when solving the ODEs and secondly, when computing the integrals. As a result, there are four combinations of methods, as given in the Figure 3.1. This chapter discusses and compares these approaches. Furthermore, we provide a detailed derivation of a novel fully probabilistic method, where both sources of uncertainties are quantified.

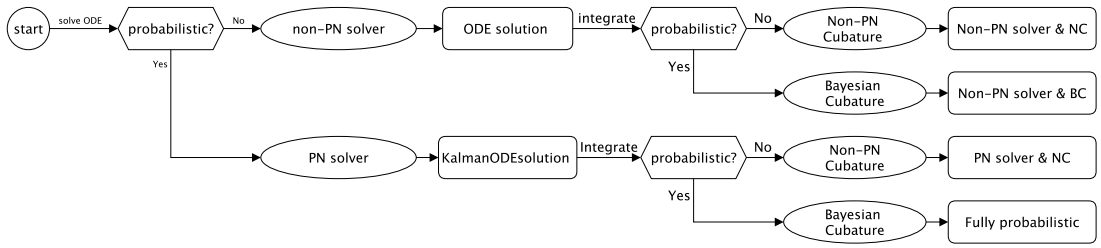


Figure 3.1: Overview of the working pipeline to compute the expected ODE solution and the uncertainty propagation.

3.1 The non-PN approach

Gerlach *et al.* (2020) provided solution for the non-probabilistic numerical approach, which is denoted by *Non-PN solver & NC* in Figure 3.1. This approach is termed *Koopman Expectation* in Gerlach *et al.* (2020). By utilizing the Koopman operator and its adjoint property, Gerlach *et al.* showed that it is feasible to compute the central moments of the ODE solutions with a stacked computation pipeline of non-PN ODE solvers and non-PN numerical integration rules.

Since non-PN solvers output point estimates for an IVP, denoted by $y(y_0, t) \in \mathbb{R}^d,\ \forall t \in [0, \mathcal{T}]$, one can directly integrate the ODE solution against the pre-defined

distribution of $p(y_0)$. Hence, the expected ODE solution yields

$$\mathbb{E}_{y_0 \sim p(y_0)}[y(y_0, t)] = \int y(y_0, t) p(y_0) dy_0. \tag{3.1}$$

Analogously, for the variance of the ODE solutions we have:

$$\mathbb{V}_{y_0 \sim p(y_0)}[y(y_0, t)] = \mathbb{E}_{y_0 \sim p(y_0)}[y^2(y_0, t)] - \mathbb{E}_{y_0 \sim p(y_0)}[y(y_0, t)]^2 \tag{3.2}$$

The non-probabilistic numerical algorithms *discretize* the problem: For example, when computing the integrals, the domain of interest is divided into sub-intervals, and the area of these sub-intervals are approximated (see Figure 2.1). Such discretization gives rise to *discretization errors*. However, since the Koopman Expectation only returns point estimates without uncertainty estimation, such discretization errors are neglected. Therefore, Section 3.2 and 3.3 introduce methods based on the idea of probabilistic numerics to quantify the computational uncertainties.

3.2 Partially probabilistic approach

This section summarizes the approaches which calibrate numerical uncertainties from either solving ODEs or integration. As shown in Figure 3.1, there are two approaches where numerical uncertainties are partially considered. We begin with the combination of PN ODE solvers and non-PN integration approaches, followed by a method that implements vanilla BC on the non-PN ODE solution.

3.2.1 PN ODE solver & non-PN Cubature

Opposite to non-probabilistic numerical ODE solvers, PN ODE solvers assign a posterior measure to the solution.

$$p(y(t)|y_0, \{z_i\}_{i=1}^N) = \mathcal{N}(\mu^S(y_0, t), \Sigma^S(y_0, t)) \tag{3.3}$$

denotes the posterior given by a PN solver, where $p(y(t)|y_0, \{z_i\}_{i=1}^N)$ represents the full posterior conditioned on the synthetic measurements $\{z_i\}_{i=1}^N$. As introduced in Chapter 2, PN ODE solvers return a Gaussian distribution, which is characterized by the posterior mean $\mu^S(y_0, t)$ and the posterior covariance matrix $\Sigma^S(y_0, t)$. According to the law of total expectation, the expected ODE solution yields

$$\mathbb{E}_{y_0 \sim p(y_0)}[y(y_0, t)] = \mathbb{E}_{y_0}(\mathbb{E}_{y(t)}[y(t)|y_0, \{z_i\}_{i=1}^N]) = \mathbb{E}_{y_0}[\mu^S(y_0, t)] \tag{3.4}$$

This solution turns out to be the same as in Equation 3.1. In both cases, we integrate over the point estimates of the ODE solution, which is $y(y_0, t)$ for non-PN solvers and $\mu^S(y_0, t)$ for PN-solvers.

Similarly, the variance of the ODE solutions can be computed using the law of total variance, as summarized below:

$$\begin{aligned}\mathbb{V}_{y_0 \sim p(y_0)}[y(y_0, t)] &= \mathbb{E}_{y_0}(\mathbb{V}_{y(t)}[y(t)|y_0, \{z_i\}_{i=1}^N])] + \mathbb{V}_{y_0}(\mathbb{E}_{y(t)}[y(t)|y_0, \{z_i\}_{i=1}^N]) \\ &= \mathbb{E}_{y_0}[\Sigma^S(y_0, t)] + \mathbb{V}_{y_0}[\mu^S(y_0, t)],\end{aligned} \tag{3.5}$$

where the numerical uncertainty from the ODE solver $\mathbb{E}_{y_0}[\Sigma^S(y_0, t)]$ and the variance originating from the initial values $\mathbb{V}_{y_0}[\mu^S(y_0, t)]$ are additive to each other.

In summary, this approach takes the numerical uncertainty in the ODE solver into account by representing the ODE solution as a posterior distribution. However, the discretization error during the integral computation is neglected.

3.2.2 Non-PN ODE solver & Bayesian Cubature

Bayesian Cubature *infers* the integral based on a finite number of function evaluations. In this scenario, the available information is the point estimates of the ODE solutions, w.r.t. the initial values $Y_0 := \{(y_0)_i\}_{i=1}^M \in \mathbb{R}^{M \times d}$ over the discretized timespan $T := \{t_j\}_{j=1}^N \in \mathbb{T}^N$, where M denotes the number of ODE solves (aka the number of initial values), d the dimensionality of the ODE and N denotes the number of timestamps. Given the ODE solutions returned by a non-PN ODE solver, this problem reduces to applying the vanilla BC with Kronecker product-structured kernel. Mathematical details are presented below.

Within the framework of PN, the following probability distributions are of special interest:

$$p(\mathbb{E}_{y_0}[y(y_0, t)]|\mathcal{D}) = \mathcal{N}(\mu_{\mathbb{E}}(t), \Sigma_{\mathbb{E}}(t)), \tag{3.6a}$$

$$p(\mathbb{V}_{y_0}[y(y_0, t)]|\mathcal{D}) = \mathcal{N}(\mu_{\mathbb{V}}(t), \Sigma_{\mathbb{V}}(t)), \tag{3.6b}$$

where $\mathcal{D}$ denotes *data*. In this context, the data consists of the input grid $X := Y_0 \times T$ and the ODE solutions $y(Y_0, T)$ with $y_{ij}(Y_0, T) = y((y_0)_i, t_j)\ \forall i = 1, ..., M,\ j = 1, ..., N$. We consider the Gaussian process with a zero-mean function:

$$y \sim \mathcal{GP}(0, k(\cdot, \cdot)). \tag{3.7}$$

Let k_t denote the $\alpha + 1$ times integrated Wiener process (IWP) kernel over time and k_s denote the RBF kernel for the spatial dimension. The product kernel is defined as:

$$k((y_0, t), (y_0', t')) = k_s(y_0, y_0') \cdot k_t(t, t'). \tag{3.8}$$

We assume that the output dimensions $(f_1, \ldots, f_d)$ of the GP are mutually independent. For the output dimension f_i, we have the temporal and spatial kernel given as:

$$k_t(t, t') = \sum_{m=0}^{\alpha} \frac{t_1^m}{m!}\frac{t_2^m}{m!} + \sum_{m=0}^{\alpha} \binom{\alpha + m}{m} \frac{|t_1 - t_2|^{\alpha - m}}{(\alpha - m)!} \frac{(t_1 \wedge t_2)^{\alpha + m + 1}}{(\alpha + m + 1)!}, \tag{3.9a}$$

$$k_s(y_0, y_0') = v_i \cdot \exp\left(-1/2 (y_0 - y_0')^\intercal \Lambda_i^{-1} (y_0 - y_0')\right), \tag{3.9b}$$

which means we employ the same IWP kernel for each output dimension, in particular, with the same degree α. For the RBF kernel, varying kernel parameters are allowed, including the outputscale v_i and the kernel lengthscale Λ_i.

As shown in Subsection 2.3.2, the two essential quantities to compute the posterior for the integral are the *kernel mean* and *initial error*. Without loss of generality, we assume the initial values are Gaussian distributed: $y_0 \sim \mathcal{N}(m_0, \Sigma_0)$. Note that other analytical solvable kernel-distribution pairs can be found in Table 2.2.

Kernel mean For the i-th output dimension, we have:

$$\begin{aligned} I_{Ki}(t_\star, t, y_0) &= \int k_s(y_0, y_{0\star}) \cdot k_t(t, t_\star) \cdot \mathcal{N}(y_{0\star}; m_0, \Sigma_0) dy_{0\star} \\ &= \int k_t(t, t_\star) \cdot v_i \exp\left(-1/2 (y_0 - y_{0\star})^\intercal \Lambda_i^{-1} (y_0 - y_{0\star})\right) \cdot \mathcal{N}(y_{0\star}; m_0, \Sigma_0) dy_{0\star} \\ &= v_i (2\pi)^{\frac{d}{2}} |\Lambda_i|^{\frac{1}{2}} k_t(t, t_\star) \cdot \int \mathcal{N}(y_{0\star}; y_0, \Lambda_i) \mathcal{N}(y_{0\star}; m_0, \Sigma_0) dy_0 \\ &= v_i (2\pi)^{\frac{d}{2}} |\Lambda_i|^{\frac{1}{2}} k_t(t, t_\star) \cdot \mathcal{N}(y_0; m_0, \Lambda_i + \Sigma_0). \end{aligned} \tag{3.10}$$

Initial error According to Fubini's theorem, one can compute the initial error as follows:

$$\begin{aligned} & I_{Ki}^2(t_\star, t) \\ &= \int \int k_s(y_0, y_{0\star}) k_t(t, t_\star) \cdot \mathcal{N}(y_0; m_0, \Sigma_0) \mathcal{N}(y_{0\star}; m_0, \Sigma_0) dy_0 y_{0\star} \\ &= \int \int k_t(t, t_\star) \cdot v_i \exp(-1/2 (y_0 - y_{0\star})^\intercal \Lambda_i^{-1} (y_0 - y_{0\star})) \mathcal{N}(y_0; m_0, \Sigma_0) \mathcal{N}(y_{0\star}; m_0, \Sigma_0) dy_0 y_{0\star} \\ &= v_i (2\pi)^{\frac{d}{2}} |\Lambda_i|^{\frac{1}{2}} k_t(t_\star, t_\star) \cdot \int \left(\int \mathcal{N}(y_0; y_{0\star}, \Lambda_i) \mathcal{N}(y_0; m_0, \Sigma_0) dy_0 \right) \mathcal{N}(y_{0\star}; m_0, \Sigma_0) dy_{0\star} \\ &= v_i (2\pi)^{\frac{d}{2}} |\Lambda_i|^{\frac{1}{2}} k_t(t, t_\star) \cdot \int \mathcal{N}(y_{0\star}; m_0, \Sigma_0 + \Lambda_i) \mathcal{N}(y_{0\star}; m_0, \Sigma_0) dy_{0\star} \\ &= v_i (2\pi)^{\frac{d}{2}} |\Lambda_i|^{\frac{1}{2}} k_t(t, t_\star) \cdot \mathcal{N}(m_0; m_0, 2\Sigma_0 + \Lambda_i) \\ &= \frac{v_i |\Lambda_i|^{\frac{1}{2}}}{|2\Sigma_0 + \Lambda_i|^{\frac{1}{2}}} k_t(t, t_\star). \end{aligned} \tag{3.11}$$

The covariance matrix of the input grid X yield

$$K_{XX} = k_s(Y_0, Y_0) \otimes k_t(T, T) \tag{3.12}$$

With the kernel mean and initial error in closed form, we can derive the posterior distribution of interest given in Equation 3.6 as follows:

- Expected ODE solution $\mathbb{E}_{y_0}[y(y_0, t)]$:

$$\mu_{\mathbb{E}}(t) = I^{\intercal}(t, T, Y_0) K_{XX}^{-1}(y(Y_0, T)), \tag{3.13a}$$
$$\Sigma_{\mathbb{E}}(t) = I_{K_i}^2(t, T) - I^{\intercal}(t, T, Y_0) K_{XX}^{-1} I(t, T, Y_0). \tag{3.13b}$$

- Variance of the ODE solutions $\mathbb{V}_{y_0}[y(y_0, t)]$:

$$\mu_{\mathbb{V}}(t) = I^{\intercal}(t, T, Y_0) K_{XX}^{-1}(y(Y_0, T) - \mu_{\mathbb{E}}(T))^2, \tag{3.14a}$$
$$\Sigma_{\mathbb{V}}(t) = I_{K_i}^2(t, T) - I^{\intercal}(t, T, Y_0) K_{XX}^{-1} I(t, T, Y_0), \tag{3.14b}$$

  where we enforce the squared deviation $(y(y_0, t) - \mu_{\mathbb{E}}(t))^2$ to be Gaussian for tractability.

3.3 Fully probabilistic approach

This section provides a fully probabilistic approach that takes uncertainties from both ODE solving and integration into account. We begin with a linear ODE and represent a single probabilistic ODE solve as a Gaussian process regression (GPR) in the temporal dimension. Then we extend this GPR into the spatio-temporal scenario where multi initial values are considered. Thereafter, we generalize the idea onto the non-linear cases and discuss the possibility of utilizing Bayesian filtering/smoothing to preserve the linear complexity in the temporal dimension.

3.3.1 Linear ODE

We consider a linear vector field:

$$f(y(t), t) = a(t) \cdot y(t) + b(t) \tag{3.15}$$

and the state-space vector $\boldsymbol{y}(t) = (y(t), \dot{y}(t))^{\intercal}$. The measurement model introduced in Subsection 2.2.2 is defined as:

$$h(\boldsymbol{y}(t), t) = \dot{y}(t) - f(y(t), t) = L(t)\boldsymbol{y}(t) - b(t), \tag{3.16}$$

where $L(t) = [-a(t), 1]$. As mentioned in Subsection 2.2.2, Bayesian smoothing computes the full posterior, conditioned on all of the synthetic measurements $\{z_i\}_{i=1}^N$ (see Equation 2.17). Since the measurement model $h(\boldsymbol{y}(t), t)$ is a linear map of the state-space vector $\boldsymbol{y}(t)$, one can reformulate the Extended Kalman Smoothing (in this case, the EKS reduces to exact Kalman Smoothing due to the linear vector field) in the PN solver as a Gaussian process regression with a Markovian kernel. In this section, for the sake of simplicity, we only consider one-dimensional ODE; however, under the assumption that the output dimensions of the GP are mutually independent, one could easily generalize the mathematical derivations into the multi-dimensional case.

Temporal GP regression model

In the language of Gaussian process regression, we consider

$$\boldsymbol{y}(t) \sim \mathcal{GP}(m(\cdot), k^{\nabla}(\cdot,\cdot)), \tag{3.17}$$

where $k^{\nabla}(\cdot,\cdot)$ is defined via the IWP kernel given in Equation 3.9, along with its derivatives:

$$k_t^{\nabla}(t_1,t_2) := \begin{bmatrix} k_t(t_1,t_2) & \partial_{t_1} k_t(t_1,t_2) \\ \partial_{t_2} k_t(t_1,t_2) & \partial^2 k_t(t_1,t_2) \end{bmatrix}. \tag{3.18}$$

Given the timestamps $T = \{t_j\}_{j=1}^N \in \mathbb{T}^N$, we define the stacked state-space vectors and the observations on this given time grid as:

$$\boldsymbol{y}(T) := [\boldsymbol{y}(t_j)]_{j=1}^N \in \mathbb{R}^{2N} \tag{3.19a}$$

$$z(T) := [z(t_j)]_{j=1}^N = \mathbf{0} \in \mathbb{R}^N \tag{3.19b}$$

Correspondingly, the measurement function for theses given timestamps is characterized by $L(T)$ and $b(T)$, which are defined as follows:

$$L(T) := \begin{bmatrix} L(t_1) & & & \\ & L(t_2) & & \\ & & \ddots & \\ & & & L(t_N) \end{bmatrix} = \operatorname{diag}\left(L(t_1), \cdots, L(t_N)\right) \in \mathbb{R}^{N \times 2N} \tag{3.20a}$$

$$b(T) := \begin{bmatrix} b(t_1) & \cdots & b(t_N) \end{bmatrix}^{\intercal} \in \mathbb{R}^N \tag{3.20b}$$

To improve readability, in the following sections, we refer to $L(T)$ as L and $b(T)$ as B. For a location within the timespan $t \in [0, \mathcal{T}]$, the posterior distribution yields

$$p(\boldsymbol{y}(t)|z(T)) = \mathcal{N}(\mu(t), \Sigma(t)), \tag{3.21}$$

with

$$\begin{aligned} \mu(t) &= m(t) + K_{tT} L^{\intercal} (L K_{TT} L^{\intercal})^{-1} (\mathbf{0} - (L m(T) - B)) \\ &= m(t) - K_{tT} L^{\intercal} (L K_{TT} L^{\intercal})^{-1} (L m(T) - B), \end{aligned} \tag{3.22a}$$

and

$$\Sigma(t) = K_{tt} - K_{tT} L^{\intercal} (L K_{TT} L^{\intercal})^{-1} L K_{Tt}. \tag{3.22b}$$

Spatio-temporal GP regression model

We extend the temporal GPR to the spatio-temporal variant by incorporating multiple initial values, where the state-space vector $\boldsymbol{y}(y_0, t) = (y(y_0, t), \dot{y}(y_0, t))^{\intercal}$ is modeled as follows:

$$\boldsymbol{y}(y_0, t) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot)). \tag{3.23}$$

We define the input grid as $Y_0 \times T$ where $Y_0 := \{(y_0)_i\}_{i=1}^M$ with M denoting the number of initial values and $T := \{t_j\}_{j=1}^N$ with N denoting the number of timestamps. As the data are given in the form of a grid, the kernel can be factorized as Kronecker-product:

$$k((Y_0, T), (Y_0, T)) = k_s(Y_0, Y_0) \otimes k_t^{\nabla}(T, T)$$

Note that the spatial kernel k_s is given in Equation 3.9b and the differentiated temporal kernel k_t^{∇} is shown in Equation 3.18. In the spatio-temporal setting, we define the spatially stacked state-space vectors and the observations as follows:

$$\boldsymbol{y}(Y_0, T) := [\boldsymbol{y}(t_j)]_{j=1}^N \in \mathbb{R}^{2MN} \tag{3.24a}$$

$$z(Y_0, T) := [z(t_j)]_{j=1}^N = \mathbf{0} \in \mathbb{R}^{MN} \tag{3.24b}$$

Consequently, the measurement model on the grid is also extended with the spatial dimension, we have

$$z(Y_0, T) = \tilde{L}(T)\boldsymbol{y}(Y_0, T) - \tilde{b}(T), \tag{3.25}$$

where

$$\tilde{L}(T) := \begin{bmatrix} L(T) & & & \\ & L(T) & & \\ & & \ddots & \\ & & & L(T) \end{bmatrix} = I_M \otimes L \in \mathbb{R}^{MN \times 2MN} \tag{3.26a}$$

(3.26b)

$$\tilde{b}(T) := \begin{bmatrix} b(t_1) & \dots & b(t_N) \\ \vdots & \ddots & \vdots \\ b(t_1) & \dots & b(t_N) \end{bmatrix} = [B^\intercal]_{i=1}^M \in \mathbb{R}^{M \times N}$$

For the sake of readability, we denote $\tilde{L}(T)$ as $\tilde{L}$ and $\tilde{b}(T)$ as $\tilde{B}$. Analogous as in the temporal GPR, the full posterior for $t \in \mathbb{T}, y_0 \in \mathbb{R}^d$ yields

$$p(\boldsymbol{y}(y_0, t)|z(Y_0, T)) = \mathcal{N}(\mu(y_0, t), \Sigma(y_0, t)), \tag{3.27}$$

where

$$\mu(y_0, t) = m(y_0, t) - (K_{y_0 Y_0} \otimes K_{tT})\tilde{L}^\intercal(\tilde{L}(K_{Y_0 Y_0} \otimes K_{TT})\tilde{L}^\intercal)^{-1}(\tilde{L}m(Y_0, T) - \tilde{B}), \tag{3.28a}$$

$$\Sigma(y_0, t) = (K_{y_0 y_0} \otimes K_{tt}) - (K_{y_0 Y_0} \otimes K_{tT})\tilde{L}^\intercal(\tilde{L}(K_{Y_0 Y_0} \otimes K_{TT})\tilde{L}^\intercal)^{-1}\tilde{L}(K_{Y_0 y_0} \otimes K_{Tt}). \tag{3.28b}$$

Since calculating the inverse $(\tilde{L}(K_{Y_0 Y_0} \otimes K_{TT})\tilde{L}^\intercal)^{-1}$ in this spatio-temporal GP regression model is of complexity $\mathcal{O}(M^3 N^3)$, we seek approaches to solve this GPR

more efficiently. Recall that the linear map $\tilde{L}$ can be factorized as the Kronecker product of $I_M \otimes L$. Using the calculation rules of the Kronecker product, we can further simplify the posterior mean and covariance as follows:

$$\begin{aligned}
\mu(y_0, t) &= m(y_0, t) - (K_{y_0 Y_0} \otimes K_{tT})(I_M \otimes L)^\intercal \\
&\qquad ((I_M \otimes L)(K_{Y_0 Y_0} \otimes K_{TT})(I_M \otimes L)^\intercal)^{-1}(\tilde{L}m(Y_0, T) - \tilde{B}) \\
&= m(y_0, t) - (K_{y_0 Y_0} \otimes K_{tT} L^\intercal)(K_{Y_0 Y_0} \otimes (L K_{TT} L^\intercal))^{-1}(\tilde{L}m(Y_0, T) - \tilde{B}) \\
&= m(y_0, t) - (\underbrace{(K_{y_0 Y_0} K_{Y_0 Y_0}^{-1})}_{=:C_s \in \mathbb{R}^{1 \times M}} \otimes \underbrace{(K_{tT} L^\intercal (L K_{TT} L^\intercal)^{-1})}_{=:C_t \in \mathbb{R}^{1 \times N}}) \underbrace{[Lm((y_0)_i, T) - B]_{i=1}^{M}}_{\in \mathbb{R}^{MN}} \\
&\overset{\star}{=} m(y_0, t) - \sum_{i=1}^{M} (C_s)_i (C_t(Lm((y_0)_i, T) - B)) \\
&= m(y_0, t) - \sum_{i=1}^{M} (C_s)_i (m((y_0)_i, t) - \mu((y_0)_i, t)),
\end{aligned} \tag{3.29}$$

where

$$\star : C_t(Lm((y_0)_i, T) - B) = m((y_0)_i, t) - \mu((y_0)_i, t)$$

according to Equation 3.22a. As for the posterior covariance, we have:

$$\begin{aligned}
\Sigma(y_0, t) &= (K_{y_0 y_0} \otimes K_{tt}) - (K_{y_0 Y_0} \otimes K_{tT})(I_M \otimes L)^\intercal \\
&\qquad ((I_M \otimes L)(K_{Y_0 Y_0} \otimes K_{TT})(I_M \otimes L)^\intercal)^{-1} \\
&\qquad (I_M \otimes L)(K_{Y_0 y_0} \otimes K_{Tt}) \\
&= (K_{y_0 y_0} \otimes K_{tt}) - (K_{y_0 Y_0} \otimes K_{tT} L^\intercal) \\
&\qquad (K_{Y_0 Y_0} \otimes L K_{TT} L^\intercal)^{-1} \\
&\qquad (K_{Y_0 y_0} \otimes L K_{Tt}) \\
&= (K_{y_0 y_0} \otimes K_{tt}) - (K_{y_0 Y_0} \otimes K_{tT} L^\intercal) \\
&\qquad (K_{Y_0 Y_0}^{-1} \otimes (L K_{TT} L^\intercal)^{-1}) \\
&\qquad (K_{Y_0 y_0} \otimes L K_{Tt}) \\
&= (K_{y_0 y_0} \otimes K_{tt}) - (K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \otimes (K_{tT} L^\intercal (L K_{TT} L^\intercal)^{-1} L K_{Tt}) \\
&\overset{\star}{=} (K_{y_0 y_0} \otimes K_{tt}) - (K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \otimes (K_{tt} - \Sigma(t)) \\
&= (K_{y_0 y_0} \otimes K_{tt}) - (K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \otimes K_{tt} + (K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \otimes \Sigma(t) \\
&= (K_{y_0 y_0} - K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \otimes K_{tt} + (K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \otimes \Sigma(t) \\
&= (K_{y_0 y_0} - K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \cdot K_{tt} + (K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \cdot \Sigma(t)
\end{aligned} \tag{3.30}$$

where

$$\star : K_{tT} L^\intercal (L K_{TT} L^\intercal)^{-1} L K_{Tt} = K_{tt} - \Sigma(t)$$

according to Equation 3.22b. We use $\Sigma(t)$ to denote the smoothing variance at time t because $\Sigma((y_0)_i t)$ remains the same for all $(y_0)_i$. Notice that after such simplification, the time complexity for computing posterior reduces from $\mathcal{O}(M^3N^3)$ to $\mathcal{O}(\max(M^3, MN))$, which is significantly cheaper.

3.3.2 Non-linear ODE

Unlike the linear ODE, one cannot directly solve the non-linear ODE using a GPR with the synthetic measurements $\{z\}_{i,j=1}^{M,N}$ since the observation model cannot be characterized in the form of a linear map. In this case, we need to introduce further assumptions to **approximate** the posterior. We consider a two-step procedure:

$$p(y(y_0, t)|z(Y_0, T)) = \int p(y(y_0, t)|y(Y_0, t)) \cdot p(y(Y_0, t)|z(Y_0, T))dy(Y_0, t), \quad (3.31)$$

where the first term $p(y(y_0, t)|y(Y_0, t))$ represents the interpolation in the spatial dimension and the second term $p(y(Y_0, t)|z(Y_0, T))$ denotes interpolation in the temporal dimension, within the spatio-temporal GPR model.

Interpolation in time For a non-linear ODE, the second term $p(y(Y_0, t)|z(Y_0, T))$ becomes intractable; hence, we approximate this term in two steps: Firstly, we omit the correlation in the spatial dimension, which means

$$p(y(Y_0, t)|z(Y_0, T)) \approx \prod_{i=1}^{M} p(y((y_0)_i, t)|z((y_0)_i, T)). \quad (3.32)$$

One may notice that the i-th factorized term $p(y((y_0)_i, t)|z((y_0)_i, T))$ is indeed the full posterior returned by a single ODE solve, regarding the initial value $(y_0)_i$. This fact leads to our second approximation: Computing the individual posteriors using Extended Kalman Smoother (EKS). In this case, the posterior $p(y((y_0)_i, t)|z((y_0)_i, T))$ is approximated by:

$$p(y((y_0)_i, t)|z((y_0)_i, T)) \approx \mathcal{N}(\mu^S((y_0)_i, t), \Sigma^S((y_0)_i, t)), \quad (3.33)$$

where $\mu^S((y_0)_i, t)$ and $\Sigma^S((y_0)_i, t)$ denote the smoothing mean and covariance, respectively. This give rise to the approximation of the $p(y(Y_0, t)|z(Y_0, T))$:

$$p(y(Y_0, t)|z(Y_0, T)) \approx \mathcal{N}(\mu^S(Y_0, t), \Sigma^S(Y_0, t)), \quad (3.34)$$

where $\mu^S(Y_0, t) = [\mu^S((y_0)_i, t)]_{i=1}^M$ and $\Sigma^S(Y_0, t) = \text{diag}(\{\Sigma^S((y_0)_i, t)\}_{i=1}^M)$ are the stacked smoothing means and covariances.

Interpolation in space After resolving the issue in the temporal dimension, we still need to find the closed-form representation of the spatial interpolation:

$$p(y(y_0,t)|y(Y_0,t)) = \mathcal{N}(\mu_s(y_0,t), \Sigma_s(y_0,t)). \tag{3.35}$$

For this, we consider the same GP as given in Subsection 3.2.2. The posterior is characterized by the following mean and covariance:

$$\mu_s(y_0,t) = (K_{y_0Y_0} \otimes K_{tt})(K_{Y_0Y_0} \otimes K_{tt})^{-1} y(Y_0,t), \tag{3.36a}$$
$$\Sigma_s(y_0,t) = (K_{y_0y_0} \otimes K_{tt}) - (K_{y_0Y_0} \otimes K_{tt})(K_{Y_0Y_0} \otimes K_{tt})^{-1}(K_{Y_0y_0} \otimes K_{tt}). \tag{3.36b}$$

Applying the Kronecker-product calculation rules, we have:

$$\begin{aligned}\mu_s(y_0,t) &= (K_{y_0Y_0} \otimes K_{tt})(K_{Y_0Y_0}^{-1} \otimes K_{tt}^{-1}) y(Y_0,t) \\ &= ((K_{y_0Y_0}K_{Y_0Y_0}^{-1}) \otimes \underbrace{K_{tt}K_{tt}^{-1}}_{1}) y(Y_0,t) \\ &= K_{y_0Y_0}K_{Y_0Y_0}^{-1} y(Y_0,t),\end{aligned} \tag{3.37a}$$

$$\begin{aligned}\Sigma_s(y_0,t) &= (K_{y_0y_0} \otimes K_{tt}) - (K_{y_0Y_0} \otimes K_{tt})(K_{Y_0Y_0}^{-1} \otimes K_{tt}^{-1})(K_{Y_0y_0} \otimes K_{tt}) \\ &= (K_{y_0y_0} \otimes K_{tt}) - (K_{y_0Y_0}K_{Y_0Y_0}^{-1}K_{Y_0y_0}) \otimes K_{tt} \\ &= (K_{y_0y_0} - K_{y_0Y_0}K_{Y_0Y_0}^{-1}K_{Y_0y_0}) \otimes K_{tt} \\ &= (K_{y_0y_0} - K_{y_0Y_0}K_{Y_0Y_0}^{-1}K_{Y_0y_0}) \cdot K_{tt}.\end{aligned} \tag{3.37b}$$

Posterior inference In summary, the posterior in Equation 3.31 can be approximated as:

$$\begin{aligned}p(y(y_0,t)|z(Y_0,T)) &\approx \mathcal{N}(\hat{\mu}(y_0,t), \hat{\Sigma}(y_0,t)) \\ &= \int \mathcal{N}(y(y_0,t); \mu_s(y_0,t), \Sigma_s(y_0,t)) \\ &\qquad \cdot \mathcal{N}(y(Y_0,t); \mu^S(Y_0,t), \Sigma^S(Y_0,t)) dy(Y_0,t),\end{aligned} \tag{3.38}$$

where

$$\hat{\mu}(y_0,t) = K_{y_0Y_0}K_{Y_0Y_0}^{-1}\mu^S(Y_0,t), \tag{3.39a}$$
$$\hat{\Sigma}(y_0,t) = (K_{y_0y_0} - K_{y_0Y_0}K_{Y_0Y_0}^{-1}K_{Y_0y_0}) \cdot K_{tt} + K_{y_0Y_0}K_{Y_0Y_0}^{-1}\Sigma^S(Y_0,t)K_{Y_0Y_0}^{-1}K_{Y_0y_0}. \tag{3.39b}$$

It is noteworthy that for the linear ODE, we would arrive at the same expression for the posterior mean if we employ the zero-mean function in Equation 3.29 since the GPR posterior coincides with the smoothing posterior computed by EKS:

$$\mu(y_0,t) = \sum_{i=1}^{M} (C_s)_i \mu((y_0)_i, t) = K_{y_0Y_0}K_{Y_0Y_0}^{-1} \underbrace{\mu(Y_0,t)}_{=\mu^S(Y_0,t)}, \tag{3.40}$$

which means this approximated posterior mean is *loss-free* for linear ODEs. However, the expression for the posterior covariance of the approximate version differs from the exact posterior covariance. Recall that the exact posterior covariance is:

$$\Sigma(y_0, t) = (K_{y_0 y_0} - K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \cdot K_{tt} + (K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \cdot \Sigma(t).$$

While the first term remains the same, the second term is different compared to $\hat{\Sigma}(y_0, t)$ given in Equation 3.39b. This approximation loss and its effect will be further discussed in Chapter 4.

3.3.3 Integral inference

Recall that the objects of interest are

$$p(\mathbb{E}_{y_0}[y(y_0, t)]|\mathcal{D}) = \mathcal{N}(\mu_{\mathbb{E}}(t), \Sigma_{\mathbb{E}}(t)) \quad \text{and} \quad p(\mathbb{V}_{y_0}[y(y_0, t)]|\mathcal{D}) = \mathcal{N}(\mu_{\mathbb{V}}(t), \Sigma_{\mathbb{V}}(t)).$$

With the exact and approximated posterior for $p(y(y_0, t)|z(Y_0, T))$, we present the posteriors for the expected value and variance of the ODE solutions.

Expected ODE solution and its estimated error

We compute the posterior for the expected ODE solution, which is characterized by its posterior mean and covariance.

Exact inference for linear ODE We begin with the exact posterior inference for linear ODEs. For the sake of simplicity, we employ the zero-mean function. Recall that posterior mean and covariance in the linear case are in the form:

$$\mu(y_0, t) = K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} \mu(Y_0, t), \tag{3.41a}$$
$$\Sigma(y_0, t) = (K_{y_0 y_0} - K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \cdot K_{tt} + (K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \cdot \Sigma(t). \tag{3.41b}$$

Integrating the posterior mean and covariance against $p(y_0)$ over $\mathbb{R}^d$, we have:

$$\mu_{\mathbb{E}}(t) = I_k^\intercal(Y_0) K_{Y_0 Y_0}^{-1} \mu(Y_0, t), \tag{3.42a}$$
$$\Sigma_{\mathbb{V}}(t) = (I_k^2 - I_k^\intercal(Y_0) K_{Y_0 Y_0}^{-1} I_k(Y_0)) \cdot K_{tt} + I_k^\intercal(Y_0) K_{Y_0 Y_0}^{-1} I_k(Y_0) \cdot \Sigma(t), \tag{3.42b}$$

where

$$I_k(Y_0) = \int K_{Y_0 y_0} p(y_0) dy_0, \tag{3.43a}$$

and

$$I_k^2 = \int \int k_s(y_0, y_0') p(y_0) p(y_0') dy_0 dy_0' \tag{3.43b}$$

are the kernel mean and initial error, respectively.

Approximated inference for non-linear ODE The approximated posterior is given by:

$$\hat{\mu}(y_0, t) = K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} \hat{\mu}(Y_0, t),$$
$$\hat{\Sigma}(y_0, t) = (K_{y_0 y_0} - K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} K_{Y_0 y_0}) \cdot K_{tt} + K_{y_0 Y_0} K_{Y_0 Y_0}^{-1} \hat{\Sigma}(Y_0, t) K_{Y_0 Y_0}^{-1} K_{Y_0 y_0},$$

which give rise to

$$\mu_{\mathbb{E}}(t) = I_k^\intercal(Y_0) K_{Y_0 Y_0}^{-1} \hat{\mu}(Y_0, t), \tag{3.45a}$$
$$\Sigma_{\mathbb{E}}(t) = (I_k^2 - I_k^\intercal(Y_0) K_{Y_0 Y_0}^{-1} I_k(Y_0)) \cdot K_{tt} + I_k^\intercal(Y_0) K_{Y_0 Y_0}^{-1} \hat{\Sigma}(Y_0, t) K_{Y_0 Y_0}^{-1} I_k(Y_0), \tag{3.45b}$$

Variance of ODE solutions and its estimated error

The probability distribution of the variance (see Equation 3.6b) is computed analogously to the expected ODE solution. Although the posterior will not be a GP, but instead a non-central χ^2-process, we imposed the Gaussianity into the variance for tractability. As shown in Equation 3.5, the resulted variance is composed of two resources: The local smoothing variance for each time point and the global variance among all of the smoothing means. Inspired by this fact, we design our observations as

$$\hat{\mu}_V(Y_0, t) := \hat{\Sigma}(Y_0, t) + (\hat{\mu}(Y_0, t) - \mu_{\mathbb{E}}(t))^2. \tag{3.46}$$

Therefore, the corresponding posterior mean for $\mathbb{V}(t)$ is summarized as:

$$\mu_{\mathbb{V}}(t) = I_k^\intercal(Y_0) K_{Y_0 Y_0}^{-1} \hat{\mu}_V(Y_0, t). \tag{3.47}$$

Since the posterior covariance in BC is invariant to the observations, the estimated error for the variance $\Sigma_{\mathbb{V}}(t)$ has the same form as the posterior variance given in Equation 3.42b and 3.45b, for both linear and non-linear cases.

We name this fully probabilistic spatio-temporal factorized Bayesian Cubature *STFBC*. In this thesis, we refer to the exact approach for linear ODEs as *e-STFBC*. For the non-linear ODEs, we derived an approximate version which is termed *a-STFBC*.

3.3.4 Pre-defined integration points

By default, one chooses the integration locations, in our context, the initial values, randomly. Inspired by the design of classic numerical integration rules, we question whether our STFBC benefits from the pre-defined integration locations (aka sigma points). To answer this question, we apply the design rule of sigma points in spherical cubature integration (Särkkä, 2013, Section 6.5) into our setting:

The unit sigma points are computed as:

$$\xi_i = \begin{cases} \sqrt{d}\boldsymbol{e}_i & i = 1, \ldots, d \\ -\sqrt{d}\boldsymbol{e}_{i-d} & i = d+1, \ldots, 2d \end{cases} \tag{3.48}$$

where e_i denotes the unit vector in the direction of coordinate axis i. The effect of the use of pre-defined sigma points will be shown in Chapter 4.

Chapter 4

Experiments and Analysis

This chapter presents the results and analysis of different experiments. We first evaluate the performance of our fully probabilistic STFBC approaches on various types of ODEs. Next, we compare how the number of ODE solves affects the residual error returned by Koopman Expectation (Gerlach *et al.*, 2020) and STFBC. Afterward, we focus on the internal characteristics of the STFBC algorithm and provide qualitative analysis for the uncertainty calibration.

4.1 Implementation details

As shown in Chapter 3, both non-PN and PN approaches to compute the expected value and variation of the ODE solutions require an ODE solver and integration tools. The ODE solvers used in this thesis are taken from `ProbNum`[1]. By default, we employ the `WrappedScipyRungeKutta` as the non-PN ODE solver and `ExtendedKalmanSmoother`("EK1") with order 2 as our PN-solver. In terms of integration rules, we chose the `Cubature`[2] method for the non-PN case, and build our PN integration methods based on `NumPy`.

For all experiments shown in this Chapter, the root-mean-square error (RMSE) is computed between the point estimates from the algorithms above and the empirical mean from ODE solutions w.r.t. 10^5 Monte Carlo samples of initial values. We refer to these MC results as baselines.

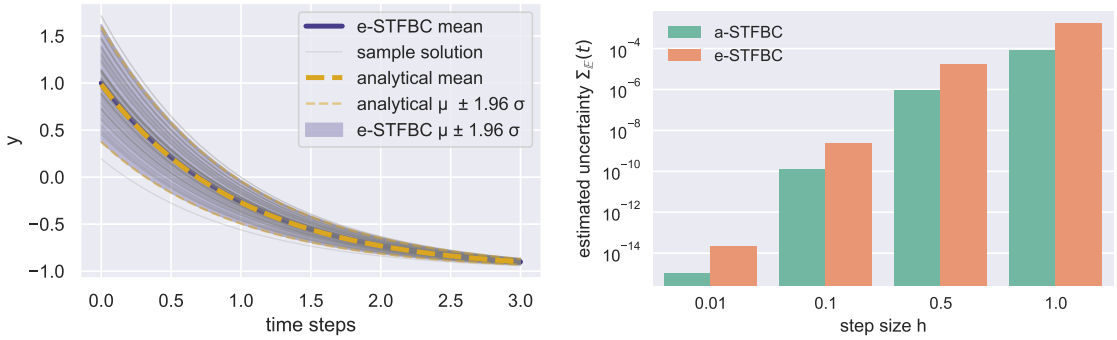
4.2 Performance evaluation

This section evaluates the performance of STFBC based on various ODEs. We first show the results computed by e-STFBC on a time-invariant linear ODE. Then we solve non-linear systems with a-STFBC and present the outputs, compared to a baseline given by Monte Carlo.

[1]ProbNum: https://github.com/probabilistic-numerics
[2]Cubature: https://github.com/saullocastro/cubature

Linear ODE

We begin with a time-invariant linear ODE, as defined in Chapter 1. Figure 4.1a shows the results computed by the e-STFBC, where we choose $a = b = -1, m_0 = 1, \sigma_0^2 = 0.1$. The step size of the PN solver is fixed with $h = 0.05$. Note that both the integral mean and the error bar approximate the analytical solution well.



(a) Performance of e-STFBC

(b) Comparison of integration error

Figure 4.1: *Left*: Results given by e-STFBC on the time-invariant linear ODE. *Right*: Comparison of the numerical uncertainties returned by e-STFBC and a-STFBC, regarding different step sizes.

In addition to the performance, we question what kind of information loss was caused by the approximation in a-STFBC. As discussed in Section 3.3, the posterior mean given by a-STFBC is loss-free, when we employ a zero-mean prior function in e-STFBC. The only difference lies in the estimated numerical uncertainties in the computation, namely the covariance $\Sigma_{\mathbb{E}}(t)$ in the posterior distribution:

$$\mathbb{E}_{y_0}[y(y_0, t)] \sim \mathcal{N}(\mu_{\mathbb{E}}(t), \Sigma_{\mathbb{E}}(t)).$$

Figure 4.1b compares the estimated numerical uncertainties of both exact and approximate approaches, concerning various time steps. As can be seen, the estimated uncertainty given by a-STFBC is lower than that from e-STFBC, for all step sizes $h = 0.01, 0.1, 0.5, 1.0$. Although the magnitude of the difference between the two estimated uncertainties remains similar from a relative angle, the absolute discrepancy could be reduced by selecting smaller step sizes.

Non-linear ODEs

This part presents the performance evaluation of the a-STFBC on different dynamical systems. Table 4.1 summarizes the details of these ODEs, including the corresponding vector fields, the distribution of the initial value, and the parameter setting. The values in the parameter tuples are ordered alphabetically $(a, b, \dots)$.

For the ODE solver, we employed a constant step size $h = 0.05$ for the logistic equation and Lotka-Volterra system, and $h = 0.1$ for Fitzhugh-Nagumo and Van der Pol.

Table 4.1: Details about the ODEs: Including the vector fields, distribution of the initial values and the parameter settings. I_d denotes the identity matrix of dimension d.

ODE	Vector field	$p(y_0)$	parameters
Logistic	$f(t,y) = ay(1 - y/b)$	$\mathcal{N}(0.05, 10^{-4})$	(3, 3)
Fitzhugh-Nagumo	$f(t,y) = \begin{bmatrix} y_1 - 1/3 y_1^3 - y_2 + a \\ 1/d(y_1 + b - cy_2) \end{bmatrix}$	$\mathcal{N}([0.5, 1]^\intercal, 0.1 \cdot I_2)$	(0, 0.08, 0.07, 1.25)
Lokta-Volterra	$f(t,y) = \begin{bmatrix} ay_1 - by_1y_2 \\ -cy_2 + dy_1y_2 \end{bmatrix}$	$\mathcal{N}([5, 5]^\intercal, 0.3 \cdot I_2)$	(5, 0.5, 5, 0.5)
Van der Pol	$f(t,y) = \begin{bmatrix} y_2 \\ a \cdot (1 - y_1^2)y_2 - y_1 \end{bmatrix}$	$\mathcal{N}([5, 5]^\intercal, 2 \cdot I_2)$	(0.05)

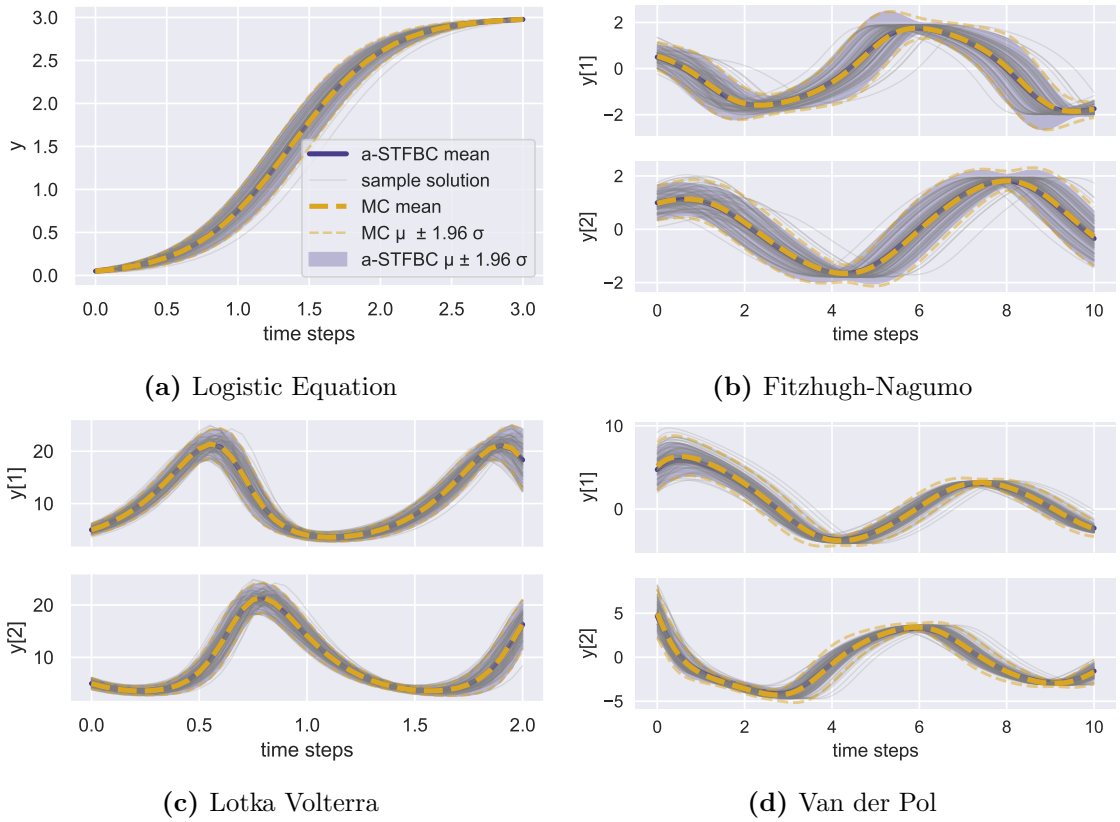
Figure 4.2 depicts the estimated mean of the ODE solutions μ and the 95%-confidence interval ($\pm 1.96\sigma$) over time, given an uncertain initial value. We computed the empirical mean and standard deviation from 10^5 Monte Carlo samples as the baseline (displayed by - - -). The counterpart (denoted by —) represents the output from our a-STFBC. As can be seen, for all ODEs, our approach produces considerably accurate integral mean estimates, and the error bar of a-STFBC also reflects the ground truth variation reasonably well. This fact indicates that the performance of a-STFBC is insensitive to the choice of ODEs.

4.3 Data efficiency

Computing the expected value requires multiple solves regarding various initial values, which may introduce drastic computational overhead. Therefore, efficient integral approximation with possibly fewer data points is desirable. First, we compare the performance of our probabilistic numerical STFBC and the non-PN Koopman Expectation approach concerning different numbers of integration points. After that, we investigate the possible improvement to STFBC introduced by the use of pre-defined sigma points (see Equation 3.48).

Convergence rate concerning the number of integration points

To investigate the data efficiency of STFBC and Koopman Expectation, we provide the convergence rates regarding the number of integration points of both approaches in Figure 4.3. The convergence rate is displayed by the drop of the root mean squared error (RMSE) of the expected ODE solution w.r.t. the number of ODE solves. The setting for step sizes follows that from Section 4.2. Since Koopman



(a) Logistic Equation

(b) Fitzhugh-Nagumo

(c) Lotka Volterra

(d) Van der Pol

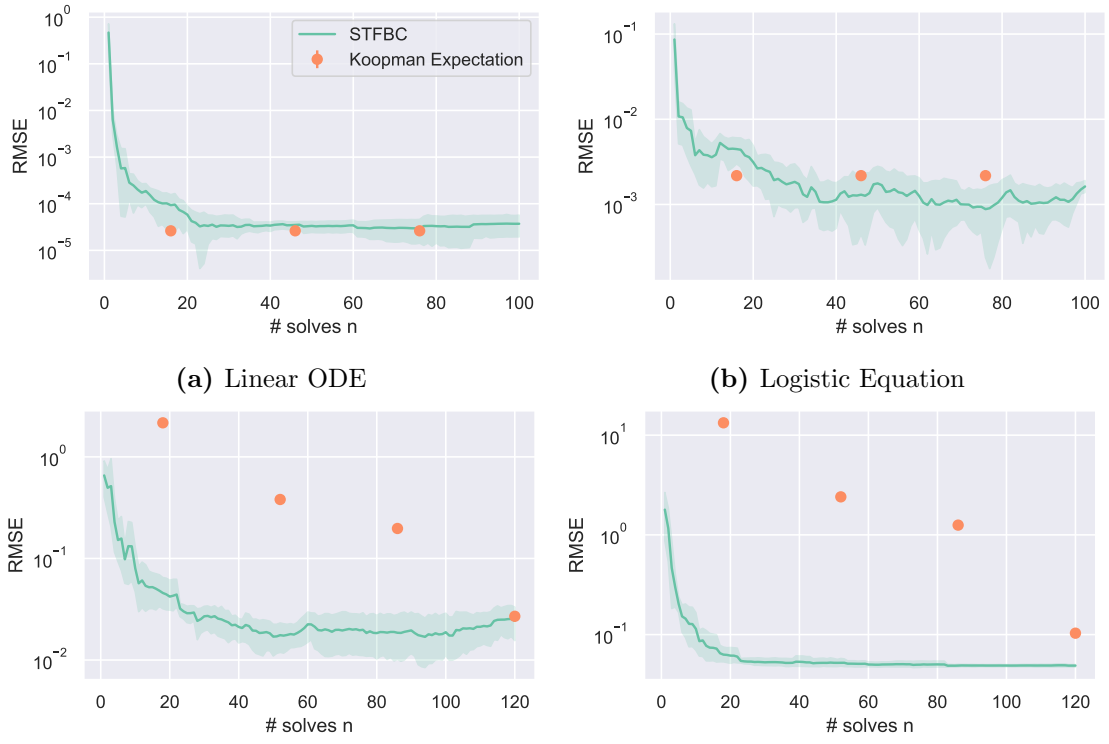
Figure 4.2: *Performance of a-STFBC.* Expected ODE solution and the propagated uncertainty computed by a-STFBC (——), compared to the Monte Carlo baseline (- - -) from 10^5 sample solves

Expectation utilizes the classic integration rule `hcubrature()`[3], where the number of integration points cannot be freely chosen, observations are only available at limited locations. One can see in Figures 4.3a and 4.3b that Koopman Expectation marginally surpassed STFBC in the one-dimensional case. However, when dealing with multi-dimensional ODEs, STFBC outperformed Koopman Expectation, as shown in Figures 4.3c and 4.3d.

Pre-defined integration points

This section investigates the effect of using pre-defined integration points (sigma points) on the performance of STFBC instead of using randomly sampled integration locations. We experiment on both one-dimensional (Logistic Equation) and multi-dimensional (Fitzhugh-Nagumo) ODEs. Figure 4.4 shows the superiority of

[3] http://ab-initio.mit.edu/wiki/index.php/Cubature_(Multi-dimensional_integration)



(a) Linear ODE

(b) Logistic Equation

(c) Fitzhugh-Nagumo

(d) Lotka-Volterra

Figure 4.3: *Comparison of convergence rates of STFBC and Koopman Expectation.* The plot shows the averaged RMSEs over five independent runs with different seeds, with an error bar representing the standard deviation.

the use of pre-defined sigma points compared to the random locations of integration. The RMSE, when integrating over the pre-defined sigma points, turns out to be a lower bound of the RMSEs with the same number of randomly chosen solves (i.e., using $2d$ integration points where d denotes the dimensionality).

4.4 Error trade-off

In the context of PN, numerical uncertainties are represented by posterior covariances. For the expected ODE solution, this is given by:

$$\Sigma_{\mathbb{E}}(t) = (\underbrace{(I_k^2 - I_k^\intercal(Y_0)K_{Y_0Y_0}^{-1}I_k(Y_0)) \cdot K_{tt}}_{\textit{uncertainty from integration}}) + \underbrace{I_k^\intercal(Y_0)K_{Y_0Y_0}^{-1}\hat{\Sigma}(Y_0,t)K_{Y_0Y_0}^{-1}I_k(Y_0)}_{\textit{uncertainty from ODE solver}}, \quad (4.1)$$

where I_k, I_k^2 denotes the kernel mean and initial error in the spatial dimension, respectively (see Equation 3.43). We define the estimated error as the square-root

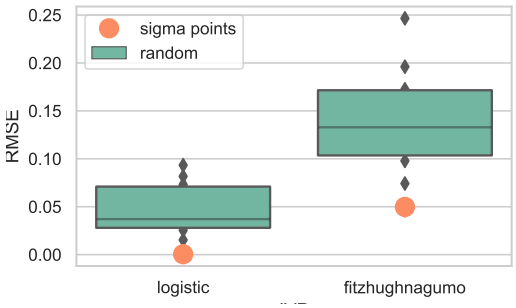


Figure 4.4: *Comparison of qualitative effectiveness of pre-defined sigma points and the same number of randomly chosen integration locations using STFBC.* The boxplots are constructed from ten independent runs.

of the posterior covariance:

$$\sigma_{\mathbb{E}}(t) := \sqrt{\Sigma_{\mathbb{E}}(t)}. \tag{4.2}$$

As can be seen, the estimated error of the expected value is decomposed into two sources of computational errors: Errors from the integration, which is introduced from the Bayesian cubature, and errors from the ODE solver, which originates from the Extended Kalman Smoothing. For the error from integration, the number of ODE solves plays a central role, whereas the step size affects the accuracy of the error from ODE solver directly. This section explores the underlying error trade-off within the STFBC approach, given different numbers of solves n and step sizes h.

The experiments are performed on the Lotka-Volterra system. Figure 4.5 displays the heat map of the RMSE and estimated numerical errors $\sigma_{\mathbb{E}}$ of the expected ODE solution. The number of solves ranges from 5 to 45, and the step sizes are equally distanced between 0.05 and 0.15. Although the a-STFBC tends to underestimate the RMSE globally, similar patterns can be observed from both plots: In general, the error declines with increasing numbers of solves and decreasing step sizes. When the number of solves is not sufficient (e.g., $n = 5, ..., 10$), the error does not decrease further despite smaller step sizes. Similar observations can be made at a large step size $h = 0.15$, where the RMSE arrived at its lowest level with around 17 solves; adding more integration points does not improve the error. To conclude, when there is only a limited number of solves available (this case could happen in solving highly complex ODEs due to limited computational budget), we anticipate the *error from integration* would dominate the *error from ODE solver*. In this case, using extremely small step sizes to solve ODEs might not be helpful to improve the accuracy of the approach but produces more computational overhead.

To further investigate how the previously mentioned error sources constitute the overall error, we visualize the RMSE and the estimated error across a subset of step sizes and numbers of solves. As can be seen in Figure 4.6, for all number of solves n, the RMSE drops linearly w.r.t. the step size when the magnitude of the step

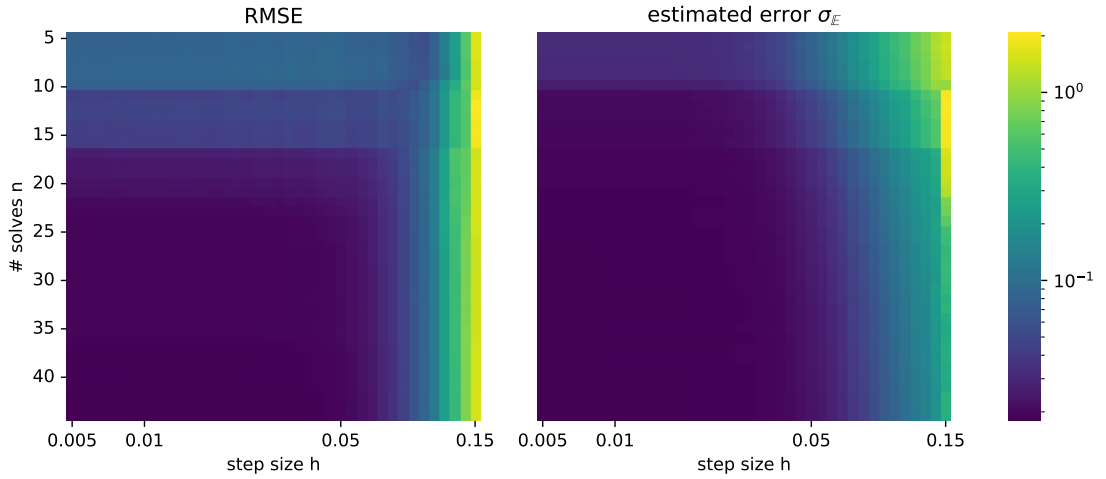


Figure 4.5: *Error trade-off between integration and ODE solving.* Experiment is conducted on Lotka-Volterra. The plot shows the errors across various combinations of numbers of solves and step sizes. *Left:* The RMSE. *Right:* The estimated error.

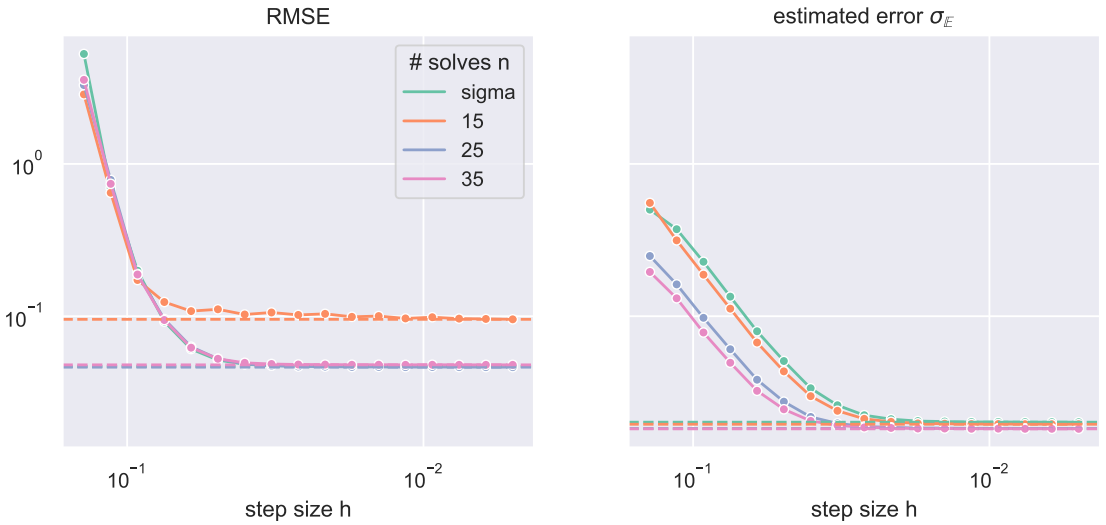


Figure 4.6: *Error source separation.* Both plots show the evolution of the error with step sizes, given varying numbers n. *Left:* The RMSE. *Right:* The estimated error.

size is relatively large. Since the plot is shown in the log-log space, this indicates that the RMSE converges in polynomial rate, which coincides with the property of filtering-based ODE solvers (Kersting *et al.*, 2020; Krämer and Hennig, 2020; Tronarp *et al.*, 2021). However, the RMSE saturates at some point. We assume this is where the error from integration starts to contribute an overwhelming part to the overall error, compared to the error from the ODE solver. A similar effect can also be observed in the estimated errors, as shown in the right subfigure.

4.5 Uncertainty calibration

Under the frame of probabilistic approaches, we are primarily interested in how well the uncertainty is calibrated; in other words, to which extent can the output variance reflect the deviation of the point estimates from the ground truth. In this section, we concentrate on the estimated uncertainty around the expected ODE solution (see Equation 4.1).

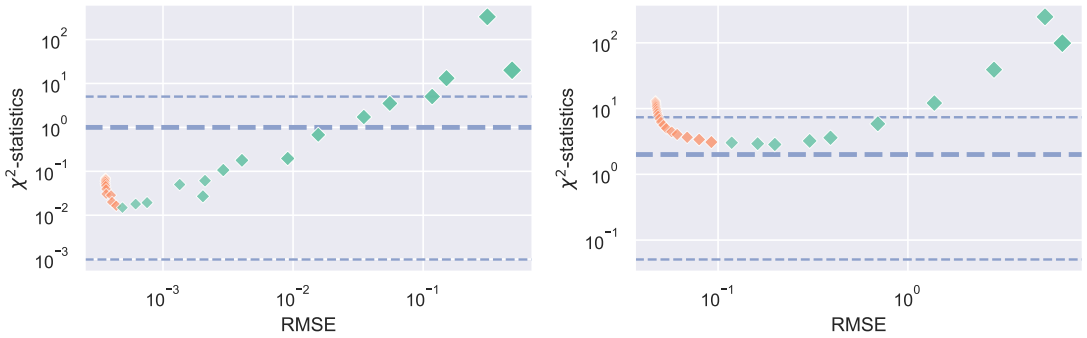
To measure the quality of the calibrated uncertainties, we employ the χ^2-statistics (Bar-Shalom *et al.*, 2004):

$$\chi^2 = \frac{1}{N}(\mathbb{E}_{y_0}(y(y_0,t)) - \mu_{\mathbb{E}}(t))^\intercal \Sigma_{\mathbb{E}}(t)^{-1}(\mathbb{E}_{y_0}(y(y_0,t)) - \mu_{\mathbb{E}}(t)), \tag{4.3}$$

where $\mathbb{E}_{y_0}(y(y_0,t))$ denotes the baseline of the expected ODE solution, $\mu_{\mathbb{E}}(t), \Sigma_{\mathbb{E}}(t)$ the posterior mean and covariance returned by STFBC. N denotes the size of the time grid. For well-calibrated models, it holds that $\chi^2 \approx d$, where d denotes the dimensionality of the ODE. When $\chi^2 \gg d$, the calibration is considered to be *overconfident*, which means the actual error is much higher than the estimated error. $\chi^2 \ll d$ indicates that the model is *underconfident*.

Figure 4.7 visualizes the fitness of the uncertainty calibration and the actual RMSEs across different configurations. These values are computed by the a-STFBC approach with pre-defined sigma points, regarding step sizes ranging from 0.005 to 1.0 for the Logistic Equation and from 0.005 to 0.15 for the Lotka-Volterra system. The step sizes are also reflected by the size of the markers in the plots.

As can be seen, our a-STFBC approach calibrates the uncertainties reasonably well for both one-dimensional (Logistic Equation) and multi-dimensional (Lotka-Volterra) ODEs, when the step sizes are chosen adequately. For both ODEs, the model tends to be overconfident when the step sizes are extremely large. When the error from the ODE solver outweighs the error from integration by a factor of $\times 10$ (depicted as green), our plots show similar behavior as the uncertainty calibration of PN solver (see Bosch *et al.*, 2021, Appendix). However, while Bosch *et al.* (2021) showed that the χ^2-statistics decrease further with smaller step sizes, these values returned by a-STFBC do not seem to reflect this (see the orange dots). As one can observe, the χ^2-statics of the orange dots show a slight tendency to grow when the



(a) Logistic **(b)** Lotka-Volterra

Figure 4.7: *Uncertainty calibration cross configurations.* Experiments are conducted on the Logistic equation and Lotka-Volterra system. Results are computed by a-STFBC using pre-defined sigma points concerning various step sizes, which are reflected by the marker size. Configurations with χ^2 inside the 95% confidence interval are considered well-calibrated (- - -). Orange dots denotes where the estimated error from integration outweighs that from ODE solver and the green dots describes the other case.

step size further decreases. The reason for this phenomenon lies in the trade-off between estimated errors from the ODE solver and integration: When the step size gets smaller, the contribution of estimated error from the ODE solver to the overall estimated error diminishes gradually. However, since the estimated error from integration is invariant to step sizes, this value remains stable even when the step size further decreases. As a result, the overall estimated error drops to a comparable level as the error from integration. At the same time, the actual RMSE does not change a lot due to the limited number of integration points. Therefore, the χ^2-statistics slowly rises.

Chapter 5

Conclusion and Future Work

This thesis dealt with the expected ODE solutions and the uncertainty propagation over time, given uncertainties in initial values or parameters. We begin with an existing approach where only point estimates are considered: Gerlach *et al.* (2020) leverages the Koopman operator and employs non-PN integration algorithms with non-PN ODE solver as the integrand to compute the desired output. However, this approach ignores the discretization error from both solving ODEs and computing integrals. To take the computational uncertainty in the ODE solver into consideration, one could combine the PN ODE solver with the non-PN integration rules. If only a limited number of integration points (solves) are available, it might be a good choice to apply the Bayesian integration tools to quantify the discretization errors. However, none of these approaches captures the total epistemic uncertainties produced during the computation of the expected ODE solution and the uncertainty propagation. Thus, in this thesis, we proposed a novel fully probabilistic approach with the assumption of gaussianity for most of the objects in our computational pipeline.

For the derivation of the fully probabilistic approach, we start with a linear ODE. For such ODEs, we can reformulate a single ODE solve with a fixed initial value as a Gaussian process regression, as remarked by Schober *et al.* (2019); Tronarp *et al.* (2021). Afterward, we extend this temporal GPR into a spatio-temporal case, where multiple initial values are considered simultaneously. Unfortunately, the spatio-temporal GPR introduces prohibitive computational cost because it considers an input grid of M initial values and N time stamps, which requires $\mathcal{O}(M^3N^3)$ for inverting the kernel covariance matrix. Therefore, we explored and discussed possibilities to alleviate such computational overhead. To take advantage of the individual probabilistic ODE solves, which only takes linear time complexity, we exploit the calculation rules from the Kronecker product to reuse the posterior mean and covariance given by the PN ODE solver. Based on the Kronecker product rules, we derived the closed-form solution for linear ODEs. We name this approach *e-STFBC*. However, when confronting a non-linear ODE, one cannot directly solve the ODE with a GPR due to the non-linearity in the measurement model; thus, the e-STFBC does not apply. As a result, we need to approximate the object of interest: Considering a two-step factorization with interpolation in time and space, omitting

the correlation in the given initial value grid, and utilizing the Extended Kalman Smoothing from PN solver, we successfully arrived at an approximated posterior for the expected ODE solution $\mathbb{E}_{y_0}[y(y_0, t)]$. We refer to this approximate approach as *a-STFBC*. As for the posterior of the propagated uncertainty $\mathbb{V}_{y_0}[y(y_0, t)]$, we enforced this quantity to be Gaussian for tractability. There are several ways to improve STFBC, potentially from both perspectives of mathematical explainability and performance. For instance, Gunter *et al.* (2014) offered a square-root transformed BQ approximation using linearization and moment-matching; Prüher and Straka (2017) provided a closed-form solution of moment transform based on the law of total variance. Integrating these methods into the current framework to compute the closed-form posterior for the variance $\mathbb{V}_{y_0}[y(t)]$ might be interesting for future work.

We showed the usefulness of the e-STFBC approach on a linear ODE and discussed the information loss introduced by the approximated version. It was proven that the posterior mean given by a-STFBC coincides with the posterior mean of e-STFBC; both approaches only differ in the estimated numerical uncertainties. We compared the numerical uncertainties returned by e-STFBC and a-STFBC using various step sizes. It is shown that the discrepancy in the estimated error diminishes with decreasing step sizes in an absolute sense.

To evaluate and analyze the STFBC approaches, we compare the returned point estimates to the Monte Carlo baseline, computed using 10^5 sampled ODE solutions. STFBC turns out to be able to produce considerably accurate point estimates for the expected ODE solution and reasonably good fits for the variance over time. Next, we visualize the convergence rates of STFBC and Koopman Expectation regarding the number of integration points. The empirical results showed that STFBC outperformed Koopman Expectation in the multivariate case. In addition, we explored other possibilities of improving the data efficiency of STFBC: Instead of randomly choosing the integration points, one could solve the ODEs at pre-defined integration locations (aka sigma points), which are given in spherical cubature integration (Särkkä, 2013). We showed that the performance with sigma points generally surpasses the method that uses randomly chosen integration points.

Since our computation process consists of ODE solving and integration, we need to consider the numerical uncertainties from both aspects. Of particular interest is the trade-off between errors from ODE solving and errors from integration. Therefore, we visualized the development of the RMSE and the estimated uncertainties returned by a-STFBC regarding a grid of different numbers of solves n and step sizes h. Our results show that RMSE and estimated errors decrease when the step sizes are relatively large but saturate as the step sizes get smaller. This fact hints that when only a limited number of ODE solves are available, it might not be helpful to choose extremely small step sizes of the ODE solver because the error from integration will dominate and cannot be reduced by smaller step sizes.

To access the quality of the uncertainty returned by a-STFBC, we calculated

the χ^2-statistics for various configurations, regrading different step sizes. Experiments are conducted on both one-dimensional (the logistic equation) and multi-dimensional (Lotka-Volterra system) ODEs. In both cases, the uncertainties seem to be well-calibrated for adequate step sizes. Besides, we discovered that when the step sizes are large, the behavior of a-STFBC aligns with the observation about the PN solver in Bosch *et al.* (2021) since the error from the ODE solver dominates over the error from integration. On the other hand, when the step sizes are small, the overall estimated error would gradually fall to the same magnitude as the error from integration, which results in a slight rise in the χ^2-statistics.

Abbreviations

PN	Probabilistic numerics/numerical
ODE	Ordinary Differential Equation
IVP	Initial value problem
GP	Gaussian Process
GPR	Gaussian Process Regression
BQ	Bayesian Quadrature
BC	Bayesian Cubature
NC	Numerical Cubature
EKF	Extended Kalman Filter
EKS	Extended Kalman Smoothing
RK	Runge-Kutta
TS	Taylor series
RMSE	Root-mean-square error

Bibliography

Abdulle, A. and Garegnani, G. (2020). Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration. *Stat. Comput.*, **30**(4), 907–932.

Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. (2004). *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons.

Berntsen, J., Espelid, T. O., and Genz, A. (1991). An adaptive algorithm for the approximate calculation of multiple integrals. *ACM Transactions on Mathematical Software (TOMS)*, **17**(4), 437–451.

Bosch, N., Hennig, P., and Tronarp, F. (2021). Calibrated adaptive probabilistic ode solvers. In *International Conference on Artificial Intelligence and Statistics*, pages 3466–3474. PMLR.

Briol, F.-X., Oates, C., Girolami, M., and Osborne, M. A. (2015). Frank-wolfe bayesian quadrature: Probabilistic integration with theoretical guarantees. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Briol, F.-X., Oates, C., Girolami, M., Osborne, M., and Sejdinovic, D. (2019). Probabilistic integration: A role in statistical computation? *Statistical Science*, **34**, 1–22.

Caflisch, R. E. (1998). Monte Carlo and quasi-Monte Carlo methods. In *Acta numerica, 1998*, volume 7 of *Acta Numer.*, pages 1–49. Cambridge Univ. Press, Cambridge.

Chen, H.-H. (2004). Stability and chaotic dynamics of a rate gyro with feedback control under uncertain vehicle spin and acceleration. *Journal of sound and vibration*, **273**(4-5), 949–968.

Cheng, H. and Sandu, A. (2009). Uncertainty quantification and apportionment in air quality models using the polynomial chaos method. *Environmental Modelling & Software*, **24**(8), 917–925.

Chkrebtii, O. A., Campbell, D. A., Calderhead, B., and Girolami, M. A. (2016). Bayesian solution uncertainty quantification for differential equations. *Bayesian Analysis*, **11**(4), 1239–1267.

Conrad, P. R., Girolami, M., Särkkä, S., Stuart, A., and Zygalakis, K. (2017). Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Stat. Comput.*, **27**(4), 1065–1082.

de Boor, C. and Rice, J. R. (1979). An adaptive algorithm for multivariate approximation giving optimal convergence rates. *J. Approx. Theory*, **25**(4), 337–359.

Diaconis, P. (1988a). Bayesian numerical analysis. In *Statistical decision theory and related topics, IV, Vol. 1 (West Lafayette, Ind., 1986)*, pages 163–175. Springer, New York.

Diaconis, P. (1988b). Bayesian numerical analysis. In *Statistical decision theory and related topics, IV, Vol. 1 (West Lafayette, Ind., 1986)*, pages 163–175. Springer, New York.

Garcke, J. (2013). Sparse grids in a nutshell. In *Sparse grids and applications*, volume 88 of *Lect. Notes Comput. Sci. Eng.*, pages 57–80. Springer, Heidelberg.

Genz, A. C. and Malik, A. A. (1980). Remarks on algorithm 006: An adaptive algorithm for numerical integration over an n-dimensional rectangular region. *Journal of Computational and Applied mathematics*, **6**(4), 295–302.

Gerlach, A. R., Leonard, A., Rogers, J., and Rackauckas, C. (2020). The koopman expectation: An operator theoretic method for efficient analysis and optimization of uncertain hybrid dynamical systems.

Griffiths, D. F. and Higham, D. J. (2010). *Numerical Methods for Ordinary Differential Equations - Initial Value Problems*. Springer undergraduate mathematics series. Springer.

Gunter, T., Osborne, M. A., Garnett, R., Hennig, P., and Roberts, S. J. (2014). Sampling for inference in probabilistic models with fast bayesian quadrature. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Hanss, M. (2002). The transformation method for the simulation and analysis of systems with uncertain parameters. *Fuzzy Sets and systems*, **130**(3), 277–289.

Hennig, P. and Hauberg, S. (2014). Probabilistic Solutions to Differential Equations and their Application to Riemannian Statistics. In S. Kaski and J. Corander,

editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 347–355, Reykjavik, Iceland. PMLR.

Hennig, P., Osborne, M. A., and Girolami, M. (2015). Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, **471**(2179).

Hinrichs, A., Novak, E., Ullrich, M., and Woźniakowski, H. (2014). The curse of dimensionality for numerical integration of smooth functions. *Math. Comp.*, **83**(290), 2853–2863.

Horn, R. A. and Johnson, C. R. (1991). *Topics in matrix analysis*. Cambridge University Press, Cambridge.

Johnson, S. G. (2020). Implementation of cubature. Retrieved 20 Aug. 2021.

Karvonen, T. (2019). *Kernel-Based and Bayesian Methods for Numerical Integration*. Doctoral thesis, School of Electrical Engineering.

Karvonen, T. and Särkkä, S. (2017). Classical quadrature rules via gaussian processes. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6.

Kennedy, M. (1998). Bayesian quadrature with non-normal approximating functions. *Statistics and Computing*, **8**(4), 365–375.

Kersting, H. and Hennig, P. (2016). Active uncertainty calibration in bayesian ode solvers. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 309–318. AUAI Press.

Kersting, H., Sullivan, T. J., and Hennig, P. (2020). Convergence rates of gaussian ode filters. *Statistics and computing*, **30**(6), 1791–1816.

Krämer, N. and Hennig, P. (2020). Stable implementation of probabilistic ode solvers.

Larkin, F. M. (1970). Optimal approximation in Hilbert spaces with reproducing kernel functions. *Math. Comp.*, **24**, 911–921.

Lie, H. C., Stuart, A. M., and Sullivan, T. J. (2019). Strong convergence rates of probabilistic integrators for ordinary differential equations. *Stat. Comput.*, **29**(6), 1265–1283.

Lin, Y. and Stadtherr, M. (2006). Validated solution of odes with parametric uncertainties. *Computer-aided chemical engineering*, **21**, 167–172.

Merriam-Webster (n.d.). Quadrature. In *Merriam-Webster.com dictionary*. Retrieved 9 Aug. 2021.

Minka, T. (2000). Deriving quadrature rules from gaussian processes. Microsoft Research.

Oates, C. J. and Sullivan, T. J. (2019). A modern retrospective on probabilistic numerics. *Statistics and Computing*, **29**(6), 1335–1351.

Oates, C. J., Papamarkou, T., and Girolami, M. (2016). The controlled thermodynamic integral for bayesian model evidence evaluation. *Journal of the American Statistical Association*, **111**(514), 634–645.

Oates, C. J., Girolami, M., and Chopin, N. (2017). Control functionals for Monte Carlo integration. *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, **79**(3), 695–718.

Oates, C. J., Cockayne, J., Briol, F.-X., and Girolami, M. (2019). Convergence rates for a class of estimators based on stein's method. *Bernoulli*, **25**(2), 1141–1159.

O'Hagan, A. (1991). Bayes-Hermite quadrature. *J. Statist. Plann. Inference*, **29**(3), 245–260.

Petersen, K. B., Pedersen, M. S., *et al.* (2008). The matrix cookbook. *Technical University of Denmark*, **7**(15), 510.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical recipes in FORTRAN*. Cambridge University Press, Cambridge, second edition. The art of scientific computing, With a separately available computer disk.

Prüher, J. and Särkkä, S. (2016). On the use of gradient information in gaussian process quadratures. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

Prüher, J. and Straka, O. (2017). Gaussian process quadrature moment transform. *IEEE Transactions on Automatic Control*, **63**(9), 2844–2854.

Rasmussen, C. E. and Ghahramani, Z. (2003). Bayesian monte carlo. *Advances in neural information processing systems*, pages 505–512.

Rice, J. R. (1975). A metalgorithm for adaptive quadrature. *J. Assoc. Comput. Mach.*, **22**, 61–82.

Särkkä, S., Hartikainen, J., Svensson, L., and Sandblom, F. (2016). On the relation between gaussian process quadratures and sigma-point methods. *JOURNAL OF ADVANCES IN INFORMATION FUSION*, **11**(1), 31–46.

Schober, M., Duvenaud, D. K., and Hennig, P. (2014). Probabilistic ode solvers with runge-kutta means. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Schober, M., Särkkä, S., and Hennig, P. (2019). A probabilistic model for the numerical solution of initial value problems. *Stat. Comput.*, **29**(1), 99–122.

Skilling, J. (1992). Bayesian solution of ordinary differential equations. In *Maximum entropy and Bayesian methods*, pages 23–37. Springer.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press.

Teymur, O., Zygalakis, K., and Calderhead, B. (2016). Probabilistic linear multistep methods. In *Advances in Neural Information Processing Systems*, pages 4321–4328.

Teymur, O., Lie, H. C., Sullivan, T., and Calderhead, B. (2018). Implicit probabilistic integrators for odes. In *NeurIPS*.

Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. (2019). Probabilistic solutions to ordinary differential equations as non-linear bayesian filtering: A new perspective. *Statistics and Computing*, **29**(6), 1297–1315.

Tronarp, F., Särkkä, S., and Hennig, P. (2021). Bayesian ODE solvers: the maximum a posteriori estimate. *Statistics and Computing*, **31**(3), Article no. 23.

Ueberhuber, C. W. (1997). *Numerical computation. 1*. Springer-Verlag, Berlin. Methods, software, and analysis, Translated and revised from the 1995 German original.

Wahba, G. (1990). *Spline models for observational data*. SIAM.

Wu, J., Zhang, Y., Chen, L., and Luo, Z. (2013). A chebyshev interval method for nonlinear dynamic systems under uncertainty. *Applied Mathematical Modelling*, **37**(6), 4578–4591.

Zenger, C. (1991). Sparse grids. In *Parallel algorithms for partial differential equations (Kiel, 1990)*, volume 31 of *Notes Numer. Fluid Mech.*, pages 241–251. Friedr. Vieweg, Braunschweig.

Selbständigkeitserklärung

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

Ort, Datum	Unterschrift