Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Master Thesis Computer Science

Benchmarking Probabilistic ODE Solvers

Nina Sophie Effenberger

01.05.2021

Reviewers

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

Prof. Dr. Philipp Berens
(Data Science for Vision Research)
Institute for Ophthalmic Research
Universität Tübingen

Effenberger, Nina Sophie:
Benchmarking Probabilistic ODE Solvers
Master Thesis Computer Science
Eberhard Karls Universität Tübingen
Thesis period: 01.11.2020-01.05.2021

Abstract

Ordinary differential equations (ODEs) play an important role in modeling dynamical systems in various scientific fields. Most ODEs cannot be solved in closed-form but various approximate algorithms exist, for instance Runge-Kutta methods. Usually, those methods neglect the fact that approximation necessarily introduces errors and instead, the approximated solution is used as if it was the true solution. Probabilistic numerical ODE solvers fix this shortcoming by augmenting the output of the solvers with appropriate uncertainty estimates. One of these classes of algorithms randomly perturbs the behaviour of non-probabilistic algorithms by a small amount. This results in a non-deterministic approximate solution. By repeatedly solving the ODE with such a randomized algorithm, the uncertainty associated with numerical approximation can be quantified. We ask two questions: 1) *How well does the range of samples quantify the numerical uncertainty?* 2) *How many repetitions are required to reach asymptotic behavior?* This work develops several metrics that benchmark the calibration of a probabilistic ODE solver. These metrics are then used to analyse the behaviour of two representative choices of such algorithms. It turns out that – perhaps surprisingly – few samples (≤ 16) quantify the numerical error as well as many samples do. Unfortunately, the quality of the quantification depends crucially on specific parameters which need to be chosen manually. Therefore, the descriptiveness of the range of samples is ultimately problem- and user-dependent. These, and other phenomena are studied in the present thesis.

Zusammenfassung

Mit gewöhnlichen Differentialgleichungen können dynamische Systeme modelliert werden – Differentialgleichungen spielen deshalb in vielen naturwissenschaftlichen Bereichen eine wichtige Rolle. Die meisten Differentialgleichungen haben keine geschlossen darstellbare Lösung, aber es existieren klassische Verfahren, wie Runge-Kutta Methoden, die ihre Lösung approximieren. Die meisten dieser Algorithmen beziehen dabei die numerische Unsicherheit, die durch Approximationen entstehen, nicht mit ein. Die approximierte Lösung wird verwendet, als wäre sie die richtige Lösung. Probabilistische Methoden versuchen diese Fehler durch Abschätzungen der Unsicherheit sichtbar zu machen. In dieser Arbeit liegt der Fokus auf einer Klasse solcher probabilistischer Differentialgleichungslöser, die während des Lösungsvorgangs Rauschen induziert. Durch mehrfaches Lösen der Differentialgleichung kann dann die numerische Unsicherheit, die durch approximative Methoden entsteht, quantifiziert werden. Im Fokus dieser Arbeit stehen zwei Fragen: 1) *Wie gut kann die Verteilung der Samples die numerische Unsicherheit quantifizieren?* 2) *Wie viele Samples sind nötig, um asymptotisches Verhalten zu erreichen?* Es werden mehrere Metriken verwendet, um das Verhalten dieser probabilistischen Methoden zu quantifizieren. Repräsentativ werden Löser, die auf Methoden von Conrad et al. (2017) und Abdulle und Garegnani (2020) basieren, evaluiert. In der von Abdulle und Garegnani (2020) vorgeschlagenen Methode werden die deterministischen Schrittweiten durch Zufallsvariablen ersetzt und in Conrad et al. (2017) werden Zwischenergebnisse mit dem skalierten Fehlerschätzer verrauscht. Die verrauschten Löser sind nur nützlich, wenn bereits wenige Durchläufe die Varianz aller möglichen Ergebnisse widerspiegeln. Anhand verschiedener Benchmarks und Vergleiche kann festgestellt werden, dass bereits sehr wenige (≤ 16) Samples aussagekräftig sind. Da sampling-basierte Löser jede beliebige Verteilung über Ergebnisse darstellen können, sind im Gegensatz zu filterbasierten Lösern auch bimodale oder andere Verteilungen über die Lösung möglich. Das ermöglicht einen interessanten und besonderen Einblick in chaotische Systeme. Allerdings hängt die Aussagekraft der Verteilung der Samples stark vom Problem selbst und der Parameterwahl ab. Diese und andere Besonderheiten sind Teil der vorliegenden Masterarbeit.

Acknowledgements

I want to thank Philipp Hennig for his continuous support during the process of working on this thesis.
Many thanks to Nico Krämer! Thanks for your patience when I pushed to master (again). Thanks for the weekly meetings and all the spontaneous meetings in-between, after which I always had the feeling of knowing what I'm doing. I've learned so much from you during the last months and I could not have asked for a better supervisor. Your support was a real jim-dandy!
And lastly, thanks to Stephan Effenberger for always supporting my decisions and having confidence in my skills.

Contents

List of Figures

List of Abbreviations

GP	Gaussian Process
IVP	Initial Value Problem
KDE	Kernel Density Estimation
LSODA	Livermore Solver for Ordinary Differential Equations
ODE	Ordinary Differential Equation
RK	Runge-Kutta
RK45	Fourth-order Runge-Kutta method
SRD	Sample-reference distance
SSD	Sample-sample distance

Chapter 1

Introduction

This thesis is structured as follows: In the first Chapter the work is motivated and background information on ODE solvers and ODEs is given. In Chapter 2 the used methods as well as their implementation are described in detail. The benchmarking results are given in Chapter 3. A discussion and short outlook in Chapter 4 conclude this thesis.

Motivation

In our world many things are in motion. Having knowledge about some scenario at one specific time point does therefore not imply that we have knowledge about the same scenario at any point in the future or past. To model and understand (continuous) processes that change over time, ordinary differential equations (ODEs) can be used. ODEs play an important role in describing scientific processes, for example in astrophysics to describe the joint movements of celestial bodies or in molecular biology to forecast bacteria growth. While most ODEs are not solvable in closed-form, various numerical solvers that approximate a solution iteratively exist. Especially for *hard or chaotic* systems of ODEs the approximation of the solution is not always equal or even close to the solution. In this thesis probabilistic solvers that try to tackle this issue are explored. These solvers try to represent the uncertainty of the solution by introducing random perturbations into the solving process.

1.1 Ordinary Differential Equations

A differential equation is an equation that involves an unknown function and at least one of its derivatives (Särkkä and Solin (2019), Section 2.1). In ODEs, the term ordinary refers to the fact that derivatives with respect to only one dependent, in many cases time-dependent, variable appear in the differential equation (Särkkä and Solin (2019), Section 2.1). An ODE that includes up to the n-th derivative is an equation of order n (Teschl (2012), Section 3.5).

Initial Value Problems

One class of examples of ODE problems are initial value problems (IVPs). Given initial conditions at one time point and information about how the system changes (over time), future states can be approximated deterministically in many cases (Chicone (2006), Section 1.1). Such problems are defined by a differential equation (Teschl (2012), Section 2.2)

$$\frac{dy}{dt} = f(y)$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz continuous and the initial conditions are given by:

$$y(0) = y_0 \in \mathbb{R}^d.$$

To solve such an IVP on a time span $[t_0, t_{max}]$ various methods exist. In the following, all methods that are deterministic and do not provide a probabilistic interpretation are referred to as *classic solvers*. Usually, the goal when solving an IVP is one of the following:

- evaluate the solution of the ODE at one or multiple time points $t^* \in [t_0, t_{max}]$, i.e. evaluate the continuous trajectory of the solution
- evaluate the solution at the final time point t_{max}

Those two goals are not disjoint, but their computational complexity is different. The discrete solution of an IVP consists of state evaluations $\{y(t_0), y(t_1), ..., y(t_{max})\}$ and their corresponding locations/time points $T = \{t_0, t_1, ..., t_{max}\}$ (Butcher and Goodwin (2008), Section 212). The output of a classic solver is a finitely large set of discrete evaluations that are directly accessible. To access the solution at intermediate locations $t^* \in [t_0, t_{max}] \setminus \{t_0, t_1, ..., t_{max}\}$ an additional computation has to be performed. The interpolation between the discrete evaluations yields the dense output of the solution (Butcher (1996)). In this work, the focus lies on the evaluation at the time point t_{max}.

1.2 Classic ODE Solvers

The easiest way of solving an ODE is integrating both sides of the equation analytically (Teschl (2012), Section 1.1). However, for most ODEs the closed-form solution is intractable (Teschl (2012), Section 1.1) and approximate methods are used to solve the ODE iteratively on a pre-defined time span $[t_0, t_{max}]$. Such methods are called *ODE solvers* and the most basic ones are explicit and iterative, the solution $y(t)$ at a time point t depends on previous function evaluations and information on how the function changes over time. While explicit methods only use the solution at previously evaluated time points, implicit methods can also include later states of the system (Butcher and Goodwin (2008), Section 204). One important family of such classic methods are Runge-Kutta-methods (RK-methods) (Butcher (1996)). Those classic RK-methods build the base of the probabilistic solvers that are used in this work.

Understand Stepsize Selection and Error Estimates

Most classic methods come with some form of error estimate. Those error estimates give information about the *local truncation error* and by assuming that the approximation $y(t-1)$ at the previous position is correct, i.e. $y(t-1) = y_{true}(t-1)$, the local error estimate at position t is computed (Butcher and Goodwin (2008), Section 211). This does not imply that the true global error is covered by the local error tolerance of the solver (Shampine (2005)). One of the most prominent RK-methods is RK45, an ODE solver of order 4 with a local truncation error (the error that is caused by one iteration) of $O(h^5)$ where h is the stepsize of the solver (Butcher (1996)). The total accumulated error of an RK-method of order n is in $O(h^n)$ (Butcher and Goodwin (2008), Section 318), the total error of RK45 is therefore in $O(h^4)$.
This implies that, when using classic iterative methods, there must be a trade-off between the number of state evaluations, which increases computational complexity, and the accuracy of the solution. To decrease the local and total accumulated error, the size of the steps has to be decreased. The stepsize can also be reduced indirectly by decreasing the relative and absolute error tolerances of the adaptive steprule (e.g. Dormand and Prince (1980)). To use iterative ODE solvers a steprule has to be defined that suggests and, when certain conditions are fulfilled, accepts a step. There are two classic ways of defining such a steprule. The easiest way of performing steps on the time span $[t_0, t_{max}]$ is to divide the time span into an equidistant grid of n steps by choosing a constant stepsize h. The evaluation times T are then defined by

$$T = \{t_0, t_0 + h, ..., t_0 + h(n-1), t_{max}\} \tag{1.1}$$

where $t_{max} = t_0 + hn$ or $t_{max} = t_0 + h(n-1) + h^*, h^* < h$ if $\frac{t_{max}}{n} \neq h$, i.e. a smaller last step is performed if the evaluation grid is not alignable with the stepsize. When using constant steps every proposed step is accepted. If the stepsize is chosen too large, this can cause problems and it might be the case that no solution can be found (Butcher and Goodwin (2008), Section 201). To tackle this problem, adaptive steps can be used. With an adaptive steprule the solver sets each stepsize such that the error estimates are small enough and absolute and relative error tolerances are not exceeded (Dormand and Prince (1980)). By defining thresholds for those error estimates the evaluation grid is no longer equidistant and the stepsize $h_k, k \in \{1, .., n\}$ is unique for each performed step (Butcher (1996)). The evaluation grid is defined by

$$T = \{t_0, t_0 + h_1, ..., t_0 + h_{n-1}, t_{max}\}$$

and the number of performed steps n depends on the ODE and the chosen absolute and relative error tolerances. In adaptive steprules a step is accepted if the error estimate is within a relative and absolute tolerance, otherwise, the stepsize is reduced proportionally to the expected tolerances until the step can be accepted (Dormand and Prince (1980)). Additionally, a safetyscale α that scales the proposed stepsize $\widetilde{h}$ down, such that the actual stepsize is $h = \alpha \widetilde{h}$, can be included (Butcher and Goodwin (2008), Section 203). A common choice is $\alpha = 0.9$ (Butcher and Goodwin (2008), Section 203).

1.3 Probabilistic ODE Solvers

The classic methods described in the previous sub-chapter introduce numerical errors: By approximating the solution $y(t+1)$ at position $t+1$ with the approximation $y(t)$ at time point t, local approximation errors are propagated through the solver which leads to imprecision in the solution. The output of a classic solver is usually an approximation $y(t)$ and an error estimate $\xi(t)$ that aims to represent the estimate of the current local error (Butcher and Goodwin (2008), Dormand and Prince (1980), sci).

One way of improving the solution of an ODE is to increase the number n of state evaluations. This improves the approximations, i.e decreases the (estimated) errors and comes with increased runtime. Solving the ODE with an accuracy that is comparable to a closed-form solution is impossible with classic iterative methods as it would involve approximations at infinitely many locations (Butcher and Goodwin (2008), Section 213). So instead of trying the impossible, i.e. coming up with accurate point estimates such that $y(t) = y_{true}(t)$, the solution of an ODE can be interpreted probabilistically, including information about the uncertainty of the solver (Abdulle and Garegnani (2020), Conrad et al. (2017), Tronarp et al. (2019)).

1.3.1 Sampling-based ODE Solvers

Approximations are based on a number n of discrete evaluations $y(t)$ at timesteps $t \in T = \{t_1, .., t_n\}$ which introduces errors into the solution (Butcher and Goodwin (2008), Section 201). Classic methods take this information of the error approximation only indirectly into account. They allow keeping track of those error estimates, but their output is deterministic and does not include any probabilistic interpretation or uncertainty quantification (Butcher and Goodwin (2008), Chicone (2006), Teschl (2012)). In classic methods stepsize adaptation is often based on those error estimates and the error estimates are used to improve the solution of an ODE by increasing the number n of evaluations without interpreting the actual meaning of those error estimates (Butcher and Goodwin (2008), Section 201).
The in this work evaluated non-deterministic ODE solvers tackle this problem and use the fact that what is often called the solution of classic methods is actually just an approximation of the true solution (Teschl (2012), Chicone (2006)). There are generally two types of variables in ODEs (Särkkä and Solin (2019), Section 2.1.), the independent variables which are often referred to as locations or time points $t \in T$ and the dependant variables which depend on t and are often called states $y(t)$. Both of those variable types influence the solution and by introducing noise ξ into one of them, the deterministic solver becomes probabilistic (Conrad et al. (2017), Abdulle and Garegnani (2020)). In classic solvers $y(t)$ is interpreted as the solution of the ODE and $\xi(t)$ is the error that the solver estimates after having performed the step to t. Wherever the error estimate is high, the solver is less certain about the correctness of the solution. Given that the order of the error depends on the stepsize of the solver, it can be inferred that the solution of an ODE does also depend on the evaluation positions (Butcher and Goodwin (2008), Section 201). Another way of interpreting the solution of a classic solver probabilistically is therefore to evaluate the ODE at perturbed time points (Abdulle and Garegnani (2020)). Both solvers are sampling-based probabilistic versions of classic methods that aim to quantify the uncertainty of approximate methods and will be called *Noisy Solvers*. Those Noisy Solvers are described in detail in Chapter 2. In sampling-based solvers, the ODE has to be solved multiple times, once for each sample (Abdulle and Garegnani (2020), Conrad et al. (2017)). To draw the connection to samples of filtering-based methods and to make the non-determinism of the Noisy Solvers explicit, the term *solution* will not be used for the output of one solver run but for the distribution over several samples.

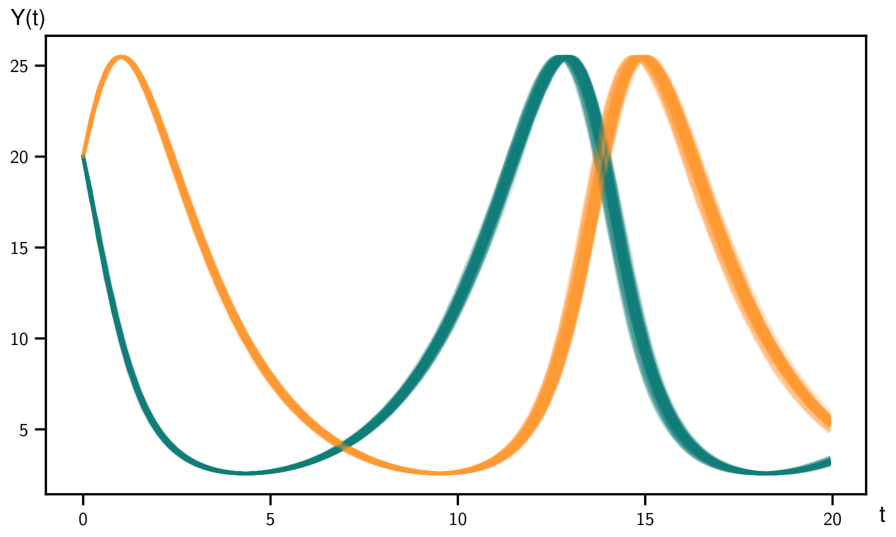

Figure 1.1: 100 Samples of the Noisy-State Solver of the Lotka-Volterra system (Lotka (2002)). The output of the solver is non-deterministic and the width of the solution visualizes the uncertainty of the solver indirectly.

1.3.2 Filtering-based ODE Solvers

Another way of interpreting iterative ODE solvers probabilistically is to use Gaussian Process (GP) regression to solve the ODE (Schober et al. (2014), Skilling (1992), Hennig and Hauberg (2014). By defining the measurement sequence in a specific way, Bayesian filtering and smoothing techniques, such as Kalman filtering or Rauch-Tung-Striebel smoothing, become applicable (Tronarp et al. (2019)). GP regression infers a probability distribution over the possible solutions of the ODE and allows to sample from the posterior Schober et al. (2014). The estimate of the solution is the mean of the solution and its variance, which quantifies the corresponding uncertainty. In Gaussian filtering methods, this uncertainty is always Gaussian-distributed (Särkkä (2013)).

Chapter 2

Methods and Material

2.1 Overview

In this chapter sampling-based ODE solvers as proposed by Conrad et al. (2017) and Abdulle and Garegnani (2020) and their implementation are described. Using such methods leads to a non-deterministic output of classic methods by perturbing either the state or evaluation location while solving the IVP. The samples and the stepsizes of the solvers still fulfill some convergence criteria and the solvers are therefore useful to describe the uncertainties of the underlying deterministic solution.

2.2 Sampling-Based Solvers

Sampling-based ODE solvers are non-deterministic, probabilistic ODE solvers. Non-determinism is achieved by introducing noise, either by using the error estimate of the solver (Conrad et al. (2017)) or by varying the evaluation position (Abdulle and Garegnani (2020)), into the system. The solver that is based on Conrad et al. (2017) will be called *Noisy-State Solver* as it introduces noise on the states. The solver based on Abdulle and Garegnani (2020) introduces noise on the stepsize and is therefore named *Noisy-Step Solver*.
Instead of solving the ODE once it is then solved multiple times and the solution is a distribution over the samples. In contrast to filtering-based ODE solvers (Schober et al. (2014)), sampling-based solvers allow for a non-Gaussian output distribution and can for example detect bifurcations or chaotic behavior of the ODE. The quality and runtime of the solvers depend on the number of samples that have to be drawn to get a solution that covers most of the sample variance. Both solvers have at least one additional parameter compared to the underlying classic method. This parameter scales the noise and will therefore

be called *noise-scale*.

2.2.1 Scale the Perturbation

Sampling-based methods are non-deterministic methods. Their non-determinism arises from a noise-scale that affects the extent of the perturbations. Setting that noise-scale to 0 results in a solver that is equivalent to the underlying deterministic solver. In the Noisy-State Solver the incorporated noise-scale is called σ, in the Noisy-Step Solver it is called τ. Both parameters scale the perturbation similarly, i.e. both noise-scales correlate positively with the extent of perturbation but intervene differently. In the following, it is always assumed $\tau, \sigma > 0$. The stepsize plays, independently of the noise-scale, an important role to the extent of perturbation that is introduced into the system. In the Noisy-Step Solver this dependency arises directly with the perturbed stepsizes (Abdulle and Garegnani (2020)), in the Noisy-State Solver this dependency arises indirectly due to higher error estimates for larger steps (see 1.2). In the following $\hat{y}(t)$ is the noisy, non-deterministic output of one solver run at location t.

2.3 Noisy-State Solver

Make Error Estimates Explicit

One possibility of improving the solution of an ODE is to not only care about improving the overall error but to think about how the error estimate can *improve our understanding* of the ODE solution. Most classic solvers provide a solution to the ODE and additionally, without further computation, some form of error estimate. In classic iterative methods, this error estimate is often used to adapt the stepsize of the solver and therefore *indirectly* influences the solution. The deterministic solution $y(t)$ of the classic method does not take the corresponding error estimate $\epsilon(t)$, which most ODE solvers provide without further computation (e.g. SciPy Integrate Documentation), into account. Conrad et al. (2017) propose to include a scaled version of that error estimate *directly* into the solution as Gaussian-distributed noise $\xi_\sigma(t) \sim \mathcal{N}(0, \sigma\epsilon(t))$ scaled by the noise-scale σ. Each approximated discrete evaluation of the solver at a location $t \in \{t_0, t_1, ..., t_n\}$ can then be described by

$$\hat{y}(t) = y(t) + \xi_\sigma(t) \tag{2.1}$$

i.e. the solver takes the proposed deterministic step, followed by a non-deterministic random draw of noise. For the continuous case at location

$t^* \in [t_k, t_{k+1})$ between two grid points t_k and t_{k+1} it holds that

$$\hat{y}(t^*) = y(t_k) + \int_{t_k}^{t*} f(\hat{y}(s))ds \tag{2.2}$$

where f(ŷ(s)) is an unknown function of time (Conrad et al. (2017)). To include the knowledge about the uncertainty of f(ŷ(s)) in the continuous dense output for $s \in [t_k, t_{k+1})$, f(ŷ(s)) can be approximated stochastically, including Gaussian-distributed noise:

$$y(t^*) = y(t_k) + \xi_k(t^* - t_k) \tag{2.3}$$

with

$$\xi_k(t^* - t_k) = \int_0^{t^*-t_k} \chi_k(\tau)d\tau \tag{2.4}$$

The functions $\{\chi_k\}$ represent the uncertainty about $f(\hat{y}(s))$ and are described by

$$\chi_k \sim \mathcal{N}(0, C^h) \tag{2.5}$$

They denote GPs with zero-mean and a covariance kernel that models the uncertainty. C^h shrinks to zero for small stepsizes h (Conrad et al. (2017)). For the implementation, this means that interpolation between the discrete evaluation locations can be performed with GP Regression. One way of implementing this interpolation according to Conrad et al. (2017) includes fitting one GP to every two consecutive discrete approximations. For a set of evaluations $\{y(t_1), y(t_2), ..., y(t_n)\}$ at discrete time points $\{t_1, t_2, ..., t_n\}$ the continuous solution can then be described as a concatenation of $n-1$ independent GPs. The ODE solution at a time point $t^* \in [t_k, t_{k+1})$ can then be approximated stochastically as

$$y(t^*) = y(t_k) + \chi_k(t^* - t_k) \tag{2.6}$$

where $\chi_k \sim \mathcal{N}(0, \xi(t))$, i.e. each χ_k is defined on $[0, h_k]$ where h_k is the size of the step that is performed by the solver, that is $h_k = t_{k+1} - t_k$ and the n-th evaluation $y(t_n^*)$ with $t_n^* \in [t_k, t_{k+1})$ has to be conditioned on all previous evaluations $\{y(t_1^*), ..., y(t_{n-1}^*)\} \in [t_k, t_{k+1})$.

Interpolation - Alternative

The discrete states are computed using a classic RK-method, the dense output is computed using Kalman filtering and smoothing. The proposal by Conrad et al. (2017) and current implementation of the interpolation of the Noisy-State Solver includes n independent Kalman posteriors for n steps of the solver.

The result of this is a dense output that is not differentiable at the discrete evaluation points, the final solution is a concatenation of independent Kalman posteriors and does not *look smooth*. It might therefore be more efficient and straightforward to evaluate the discrete solution using a classic method and fit one GP to the solution with a stepwise variance function. This adaptation affects only the dense output and is currently not evaluated.

2.4 Noisy-Step Solver

Change the Output by Perturbing the Evaluation Position

Another sampling-based approach is to not take the error estimates into account but to evaluate the function at slightly different positions with each solver run. Instead of perturbing the states, perturbations along the t-axis are introduced. The output is therefore, like the one of the Noisy-State Solver, no longer deterministic. The Noisy-Step Solver is based on a method by Abdulle and Garegnani (2020). The idea is to not only shift the evaluation time points but also project the evaluations back to the originally proposed time points. There are different ways of perturbing the steps s.t. the distribution of the i.i.d. random variables H_k that describe the stepsize of the k'th step satisfy the following assumptions:

1. $H_k > 0$
2. $\exists h > 0 : \mathbb{E}(H_k) = h$
3. $\exists p \geq 1/2$ and $C > 0$ independent of k s.t. $\mathbb{E}(H_k - h)^2 = Ch^{2p+1}$

In Abdulle and Garegnani (2020) two such distributions are proposed. Without further restrictions lognormally distributed i.i.d random variables can be used:

$$H_k \sim \log\mathcal{N}(\log(h) - \log\sqrt{1 + \tau h^{2p}}, \log(1 + \tau h^{2p})) \tag{2.7}$$

It can be shown that all of the above described criteria are fulfilled for $H_K \sim \log\mathcal{N}(\mu, \sigma^2)$, where μ is the mean of the Log-normal distribution and σ^2 the variance as given in equation 2.7:

1. $H_k > 0$ is fulfilled due to the properties of the Log-normal distribution (Crow and Shimizu (1987))

2.

$$\begin{aligned}\mathbb{E}(H_K \sim \log\mathcal{N}(\mu,\sigma^2)) &= exp(\mu + \frac{\sigma^2}{2}) \\ &= \exp(\log(h) - \log\sqrt{1+\tau h^{2p}} + \frac{\log(1+\tau h^{2p})}{2}) \\ &= \exp(\log(h) - \frac{1}{2}\log(1+\tau h^{2p}) + \frac{\log(1+\tau h^{2p})}{2}) \\ &= \exp(\log(h)) \\ &= h\end{aligned}$$

3.

$$\begin{aligned}\mathbb{E}((H_k - h)^2) &= \exp(\mu + \frac{\sigma^2}{2})\sqrt{\exp(\sigma^2) - 1} \\ &= h\sqrt{\exp(\log(1+\tau h^{2p}) - 1} \\ &= h(\tau h^{2p}) \\ &= \tau h^{2p+1}\end{aligned}$$

The assumption is therefore fulfilled for $C = \tau > 0$.

Another choice of i.i.d. random variables is

$$H_k \sim \mathcal{U}(h - \tau h^{p+1/2}, h + \tau h^{p+1/2}) \tag{2.8}$$

where we have to make sure that $0 < h < 1$ and $p \geq 1/2$. This steprule will be called uniform noisy steprule. Given a uniform distribution $\mathcal{U}(a,b)$ the assumptions are again fulfilled:

1. $H_k > 0$ is fulfilled due to the restrictment of the stepsize $0 < h < 1$

2.

$$\begin{aligned}\mathbb{E}(H_K \sim \mathcal{U}(a,b)) &= \frac{a+b}{2} \\ &= \frac{h - \tau h^{p+1/2} + h + \tau h^{p+1/2}}{2} \\ &= \frac{2h}{2} \\ &= h\end{aligned}$$

3.

$$\begin{aligned}\mathbb{E}((H_k - h)^2) &= \frac{1}{12}(b-a)^2\\ &= \frac{1}{12}(2\tau h^{p+1/2})^2\\ &= \frac{1}{12}(4\tau h^{2(p+1/2)})\\ &= \frac{1}{3}\tau h^{2p+1}\end{aligned}$$

The assumption is therefore fulfilled for $C = \frac{1}{3}\tau$.

Both distributions were used and evaluated and as suggested by the authors $p = q$, where q is the order of the solver. Analogously to the previously defined Noisy-State Solver the discrete solution of the Noisy-Step Solver at the time points $t \in \{t_0, t_1, ..., t_n\}$ can be defined as

$$y(t) := y(t + \xi_{q,\tau}(h)) \tag{2.9}$$

where $\xi_{q,\tau}(h)$ are perturbations that depend on the order of the solver q, the chosen noise-scale τ and the stepsize h. Interpolation is performed with a quartic interpolation polynomial after the k'th step between the grid-points $[\hat{t}_{k-1}, \hat{t}_k]$ for $k \in \{1, .., n\}$ where n is the number of performed steps. To evaluate the continuous solution all interpolants have to be projected to their corresponding original time span $[t_{k-1}, t_k]$, i.e. to the time points that were proposed and accepted by the steprule. As the state y which is evaluated at the perturbed time point $\hat{t}$ is projected to the original time point t, the interpolants between $[\hat{t}_{k-1}, \hat{t}_k]$ for $k \in \{1, .., n\}$ have to be rescaled to match the original time span $[t_{k-1}, t_k]$. To access a time point $t^* \in [t_{k-1}, t_k]$ the relative position within the interval $[t_{k-1}, t_k]$ has to be calculated which leads to

$$t^*_{rel} = \frac{t^*}{t_k - t_{k-1}} \tag{2.10}$$

The time point $\hat{t}$ at which the computed interpolant has to be evaluated is then defined by

$$\hat{t}^* = t^*_{rel}(\hat{t}_k - \hat{t}_{k-1}) + t_k \tag{2.11}$$

2.4.1 Noisy Steprule

Only Perturb the Stepsize

Abdulle and Garegnani (2020) provide suggestions for a class of random variables H_k (see 2.4) to perturb the stepsizes of the samples. Instead of projecting the state evaluation at time point $t + \xi_{q,\tau}$, where $\xi_{q,\tau}$ is chosen such that the described criteria are fulfilled, back to the originally proposed position t, we can also only use the perturbed step $h \sim H_k$ and solve the ODE multiple times. This noisy steprule can be applied to the original deterministic solver and leads to a probabilistic output, which is similar to the Noisy-Step Solver but does not need the introduction of a new solver. Interpolation can then be performed without further post-processing.

2.5 Programming

In this sub-chapter, the implementation, accessibility, and the underlying code base are explained. The implementation combines classic methods, which are mainly based on SciPy (SciPy Integrate Documentation), with a probabilistic interpretation. This probabilistic interpretation is highlighted by using ProbNum as the framework for the solution of the IVP. Like this, the solution is analogously defined to the output of filtering-based methods and the discrete states (and also the states at evaluated intermediate time points) are *constant random variables.* This interpretation does not change the solution but makes the probabilistic manner of sampling-based methods and their analogy to filtering-based methods clear.

2.5.1 Code Base

ProbNum

ProbNum is a python package that implements methods from probabilistic numerics (ProbNum Documentation). Among others, it contains a module for differential equations (*probnum.diffeq*) which includes probabilistic solvers that are filtering-based, such as extended Kalman filtering. Within the ProbNum-framework an IVP of dimension d is defined by a time span $[t_0, t_{max}]$, an initial value that is a d-dimensional random variable and the right-hand side function

$$f : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d \tag{2.12}$$

of the ODE system. When using filtering-based solvers the Jacobian and Hessian can also be provided to the IVP-object. To solve the IVP an *ODESolver* object, which could be for example a Noisy-Step Solver, a Noisy-State Solver or a filtering-based solver, has to be instantiated which includes the IVP and the order of the solver. To solve the IVP with the selected solver, a steprule has to be specified. The solution of an IVP is an instance of the *ODESolution* class. It provides dense output, makes the solution be callable, and collects the discrete time grid with its corresponding solution over the states. In both solvers the interpolation between the discrete evaluations is non-trivial.

SciPy

SciPy is a python library that contains modules for classic mathematical subfields, among others it implements various explicit and implicit methods to solve ODEs (SciPy Integrate Documentation). To use a Noisy-State or Noisy-Step Solver an underlying classic solver has to be defined, for this purpose, the Runge-Kutta solvers which are implemented in SciPy (sci) are wrapped. The discrete output of RK23 and RK45 are used by both, the Noisy-State and the Noisy-Step Solver, and a scaled version of the dense output is used for the interpolation of the continuous solution of the Noisy-Step Solver.

2.5.2 Implementation and Accessibility

All code is written in python. Everything is implemented in the ProbNum-Style and the solvers will be integrated into the ProbNum-repository. Both methods, the Noisy-State Solver and the Noisy-Step Solver will be accessible and usable within the ProbNum-framework. The Noisy-Step Solver is independent of the rest of the ProbNum-code, while the Noisy-State Solver uses the implementation of Kalman filtering and smoothing for dense output interpolation. The underlying classic solvers are based on Runge-Kutta methods implemented in SciPy. Both sampling-based methods are expandable to different kinds of classic ODE solvers.

2.5.3 Solve an IVP

Set up the Classic Solver

Both solvers can be set up analogously. First of all, an IVP has to be specified. This IVP consists of the right-hand side of an ODE, initial values, and a time span. With that information, a SciPy solver, for example, based on an RK45-method can be specified. This builds the base of the wrapped SciPy solver. In the ProbNum-framework the state of an ODE is described by a random variable $\in \mathbb{R}^d$ where d is the dimensionality of the ODE. In the case of those classic solvers, the state can simply be defined as a constant that equals the state approximation. The size of the first step has to be defined as well. The adaptive steprule of ProbNum chooses a stepsize that minimizes absolute and relative errors and is only applicable after the first error estimate is accessible, i.e. after the first step was performed. The Noisy-State Solver as well as the Noisy-Step Solver are both based on a noise-free method that wraps a classic SciPy solver.

Introduce Noise

Both Noisy Solvers get as an input a classic noise-free solver and a noise-scale σ or τ that specifies the *amount of noise*, i.e. a solver with $\sigma, \tau = 0$ is a deterministic solver without introduced noise. As there are different ways of introducing noise into the Noisy-Step Solver, implemented are uniform or lognormally distributed stepsizes, the Noisy-Step Solver additionally takes a parameter that describes this distribution.

Solve the IVP

To solve the IVP a steprule has to be specified. As constant steps are quite restrictive, the preferable choice are adaptive steps and all experiments were therefore performed with adaptive steps and various tolerances. Solving the IVP is performed within the ProbNum-Framework. For a time span from $[t_0, t_{max}]$ the solver starts at t_0 and iteratively performs steps until t_{max} is reached. The stepsize is chosen according to the specified step rule, in the case of adaptive steps a step(size) is accepted if it fulfills the relative and absolute tolerance criteria – otherwise, the stepsize is reduced. The solution of the IVP consists of the evaluated time points $\{t_0, ..., t_n\}$, the corresponding states $\{y_0, ..., y_n\}$ and interpolants that allow evaluating the continuous dense output in between the discrete evaluations. Computing the discrete states is quite similar in both sampling-based solvers but the interpolation is performed differently. Details on the mathematical background of the interpolation can be found in 2.3 and 2.4. The final ODE Solution has an intuitive interface: the discrete time points and states can be accessed directly by a getter-method and the dense output for every intermediate result can be called after the ODE is solved. During the evaluation of the ODE at time point $t^* \notin \{t_0, ..., t_n\}$ the Kalman posteriors of the Noisy-State Solver are updated automatically by conditioning the posterior on the current evaluation.

2.5.4 Filtering-based Solvers

Filtering-based solvers as described in 1.3.2 build the counterpart to the solvers described in the previous chapters and are accessible in ProbNum. Their output is probabilistic but deterministic and can be compared to the sampling-based solvers. Their solution is Gaussian-distributed and samples from the posterior can be drawn.

Chapter 3

Benchmarking the Solvers

3.1 The Lorenz System

The Lorenz system is a chaotic, three-dimensional system of ODEs and a mathematical model for atmospheric convection. It is a relatively simple chaotic system; Lorenz himself defines it in the related paper (Lorenz (1963)) as "the simplest example of deterministic nonperiodic flow" of which he was aware. It is defined through:

$$f(t, y) = \begin{pmatrix} a(y_2 - y_1) \\ y_1(b - y_3) - y_2 \\ y_1 y_2 - c y_3 \end{pmatrix} \tag{3.1}$$

for some parameters (a, b, c) where y_1, y_2, y_3 are time dependant variables in a three-dimensional space. The parameters (a, b, c) are proportional to physical properties and for most choices, the system becomes unstable and diverges quickly (James Hateley). For $(a, b, c) = (10.0, 28.0, 8/3)$, proper initial conditions, a sufficient small stepsize (or sufficient small absolute and relative tolerances when using adaptive steps) and a time span $[t_0, t_{max}] = [0.0, 20.0]$ the Lorenz system is *solvable*, i.e. a classic deterministic solver finds a value $y(t = 20.0)$ that fulfills the given criteria. To compare the three different solvers the Lorenz system is chosen as an exemplary ODE system. It is of special interest due to its chaotic behaviour, i.e. its exponential sensitivity with respect to the initial conditions. In chaos theory, this high sensitivity to the initial conditions is often called *butterfly effect* (Lorenz (2000)).

3.2 Settings of the Lorenz System

The parameters are set to $(a, b, c) = (10.0, 28.0, 8/3)$, the initial conditions are $(y_1(0), y_2(0), y_3(0)) = (0.0, 1.0, 1.05)$ and the time span is $[t_0, t_{max}] = [0.0, 20.0]$. The time span is chosen such that it is long enough to show the variance of the samples. The same time span is chosen in Conrad et al. (2017) and is therefore considered to be representative. All experiments were performed using adaptive steps, the stepsize of the first step is $h = 0.01$ setting $atol = rtol \in \{10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}\}$ and $\sigma, \tau \in \{10^{-1}, 10^{0}, ..., 10^{6}\}$.
σ is the noise-scale of the Noisy-State Solver and τ is the noise-scale of the noisy steprule and therefore the corresponding noise-scale for the Noisy-Step Solver. Both scales correlate positively with the *amount of noise and uncertainty* that is introduced but are not comparable in a numerical sense. For some settings, the solvers are not able to find a solution. There are no solutions for the Noisy-State Solver with $\sigma \geq 10^4$ and for the Noisy-Step Solver with $\tau \geq 10^4$ only solvers with small error tolerances were applicable.
The safetyscale is used to adapt the stepsize further: For the Noisy-State Solver the safetyscale is set to 0.8, for the Noisy-Step Solver and the noisy steprule it is set to 0.6. As the adaptive step is in the best case the maximum stepsize that still fulfills relative and absolute error tolerances and on average almost every second noisy step is larger than the suggested step, this makes sure to not lose too many samples. For each setting 512 samples were evaluated. It can be seen in 3.6 that this, for a Monte-Carlo method low number of samples, seems to be more than sufficient to cover the solver variance in all settings. As a reference solution the LSODA solution with an absolute tolerance of $atol = 10^{-15}$ and the in SciPy smallest relative tolerance of $rtol = 2.220446049250313 \cdot 10^{-14}$ is chosen, which equals

$$y_{ref}(20) = (y_1(20), y_2(20), y_3(20)) \approx (8.53507, 11.70335, 22.54430) \tag{3.2}$$

The adaptive steprule which is used, always uses a relative and absolute error tolerance which is in all cases set to the same value. *Tolerance* does therefore always include the relative and absolute error tolerance.

3.3 Benchmarks

Both solvers have several input parameters. Finding solvers, parameters, and settings with which the solutions are useful and bringing out the best of the solvers, is of huge interest when using those probabilistic solvers in practice. There are several questions that one might ask when searching for a probabilistic sampling-based method that is considered to be useful. How many samples are needed to represent the largest part of the sample variance? As the computational complexity increases linearly with the number of solver runs, i.e.

samples, answering this question yields an insight into the **efficiency** of these methods.
Filtering-based methods also provide a probabilistic interpretation of the states, their output is always Gaussian-distributed. The output of the Noisy Solvers does not need to be Gaussian. A density estimate over the samples of one state evaluation at a single location can potentially be any distribution. But maybe they are also Gaussian? Given the Log-normal rule (equation 2.7) in the Noisy-Step Solver and Gaussian-distributed noise in the Noisy-State Solver it is interesting to evaluate the **Gaussianity** of the solution of the solvers.
Regardless of the type of distribution, the solution of the probabilistic solvers should, in the best case, have a meaningful structure that represents the uncertainty of the underlying mathematical method. But how can we measure whether the probabilistic solution can **cover the uncertainty** of the deterministic solution? This question is the hardest one to answer and there is not one metric that allows quantifying this ability of a probabilistic solver. While the theoretic evaluation of the methods can give interesting insights into them, their possible practical benefits should not be forgotten. From this viewpoint, the main goal is to use as few samples as possible while still covering *most of the possible outcomes* and to find parameters for which the solvers work best.

3.3.1 Density estimate

Evaluate One Dimension of a Single State

To evaluate the output of the three solvers a set of benchmarks was developed that allows comparing the different solvers to each other but also the different settings among each other. First of all the densities over the first dimension of the 512 Samples for each setting were evaluated. Two samples of the Noisy-State Solver with $\sigma = 100$ and $rtol = atol = 10^{-6}$ and the LSODA reference solution of the Lorenz attractor are visualized in Fig. 3.1. As described in the previous chapters we use n functions as an approximation of the distribution over the solution instead of taking *one* value as *the* solution of the IVP. Multiple samples allow determining whether there are patterns within the distribution.
In Fig. 3.2 the kernel density estimates $KDE(y(t_{max}))$ with a Gaussian kernel over the set of final states $y(t_{max}) := \{y(t_{max_0}), y(t_{max_1}), ..., y(t_{max_{511}})\}$, where $y(t_{max_i})$ is the the final state of the first dimensions of the i'th sample, are plotted. Each of the kernel density estimates (KDEs) is plotted on an equidistant grid within the time span $[min(y(t_{max})), max(y(t_{max}))]$ respectively, i.e. the leftmost point of the plotted distribution equals the smallest and the rightmost point equals the largest $y(t_{max_i}) \forall i$. To make the results better comparable, the density estimates are normalized such that the

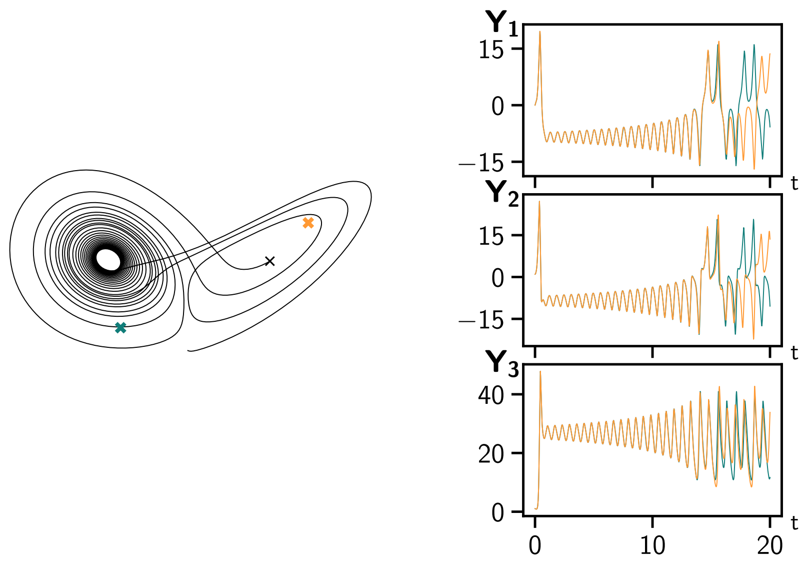

Figure 3.1: LSODA solution and two samples of the Noisy-State Solver. In black on the left, the trajectory proposed by LSODA is visualized. On the right the three dimensions Y_1, Y_2, Y_3 of two Noisy-State samples with $tol = 10^{-6}$ and $\sigma = 100$ are plotted against the time t. The colored crosses on the left mark the evaluation at the endpoint $y(20)$ respectively.

largest KDE $max(KDE(y(t_{max}))) := 1$. E.g. in the setting with $\sigma = 100$ and $rtol = atol = 10^{-6}$ all of the 512 sample states $y(20)$ are in the range $(-18.96, 16.59)$. For better interpretability the high tolerance LSODA solution as well as the solution of the underlying deterministic RK45 solver with the same tolerance are plotted. In contrast to filtering-based methods, the sampling-based solution does not need to be Gaussian-distributed and can therefore show bifurcations or other non-Gaussian distributions over the states. This can for example be seen in the Noisy-State Solver for $\sigma \in \{1, 10\}$ (Fig 3.2 (a),(b)). For $\sigma = 100$ and $tol = 10^{-4}$ (Fig 3.2 (c)) the solution is neither unimodal nor bimodal but shows almost uniformly distributed samples on a broad range of values.
The density of the Noisy-Step Solver solution does also show bimodal endresults for $\tau = 100$ and $\tau = 10^3$ and a tolerance of 10^{-4} (Fig. 3.2 (a) and Fig A.1). Overall the range of samples of the Noisy-Step Solver is, especially for small tolerances $tol < 10^{-4}$ smaller than that of the Noisy-State Solver (Fig. 3.2 (a),(b),(c) and Fig A.1). For small $\sigma = 0.1$ all of the sample solutions of $y_1(20)$ are too large for all of the tolerances, with increasing σ the solver solution becomes capable of covering the reference solution and for $\sigma \geq 10$ the Noisy-State Solver is, independent of the tolerance, capable of covering the reference solution (Fig. 3.2 (b),(c),(e),(f),(h),(i)). Due to the small variance

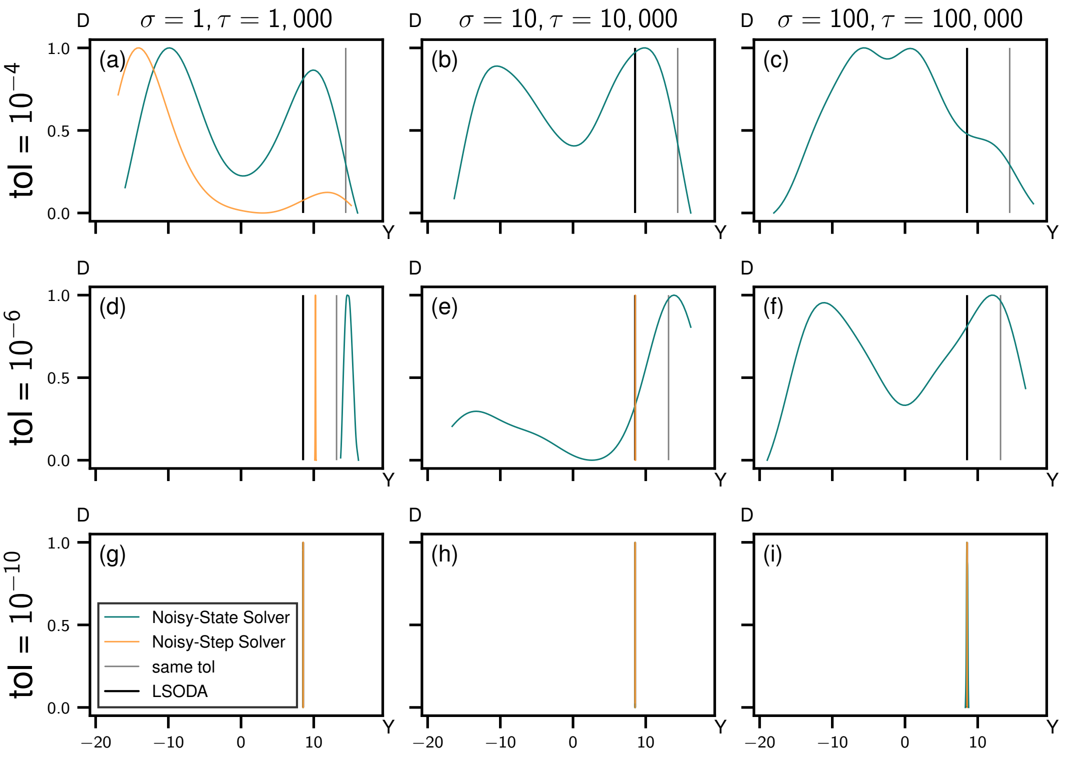

Figure 3.2: Kernel density estimates over the first dimension at the endpoint $y(20)$. Y is the sample result at the endpoint $t = 20$ and D the respective normalized density over the samples. For high tolerances the introduced noise is larger, the variance of the samples is higher. For low tolerances the introduced noise is really small and the corresponding variance is small, too. The density over the final state of the Noisy Solvers, a deterministic solution with the on the left indicated error tolerance, and the LSODA reference solution are plotted.

of the Noisy-Step Solver with small $\tau < 10^3$ and $tol < 10^{-4}$ none of the solutions covers the reference solution (Fig. A.1). With decreasing tolerance the range of the samples, i.e. their variance decreases. For tolerances $< 10^{-4}$ the variance increases for increasing noise-scales.
For some settings in Fig. 3.2 (e.g. (h),(i)), the deterministic and probabilistic solutions are so similar that they are no longer visually separable. This is the case for the Noisy-Step Solver, where the solutions in (b) and (i), For $tol \leq 10^{-6}$ and $\tau = 10^5$ and for $tol = 10^{-4}$ and $\tau = 10^4$ the Noisy-Step Solver fails and there are no results to be visualized in Fig. 3.2 (c) and (f). The above-described density estimates give a good feeling for the influence of the noise-scale and the error tolerances. While the noise-scales cannot be compared numerically directly, the results suggest that there are better and worse settings for both solvers respectively. To use the solvers efficiently, if possible, the capability of covering the reference solution, the number of needed samples for an expressive solution and the advantages or disadvantages compared

to other probabilistic solvers have to be evaluated. The focus of the following sections lies on determining whether and how the solver's uncertainties can and should be taken into account to interpret and use the classic solvers probabilistically and usefully.

3.3.2 Sample-Sample vs. Sample-Reference Distance

Quantify the Errors of the Solver

Ideally, the sample-sample distance of several samples represents the mean deviance of those samples to the reference solution at a single evaluation (or interpolation) location t. A useful probabilistic solver should therefore cover the true solution S_{true}. As the Lorenz system is not solvable in closed-form (Hashim et al. (2006)), S_{true} can only be approximated by a reference solution S_{ref} that is given by a solver with low error tolerances and is therefore assumed to be close to S_{true}. For all samples $S = \{s_1, ..., s_n\}$ the sample-sample distance (SSD) is the normalized distance of the solution $y(t)$ of one sample s_k and every other sample in the subset $S \setminus s_k$:

$$SSD(y_k(t)) = \frac{1}{n-1} \sum_{s \in S \setminus s_k} \|y_k(t) - y_s(t)\|$$

where $\| \cdot \|$ is the Euclidean norm. With this definition, the sample-sample distance is equivalent to the intersample root-mean-square error of one sample to every other sample. By averaging the $SSD(y_k(t)) \forall k \in S$ we get the mean SSD. As the sample-sample distance between two samples i and j is included in two SSDs, i.e. in $SSD(y_i(t))$ and $SSD(y_j(t))$, it has to be divided by two to get the mean sample-sample distance.

$$MSSD(y(t)) = \frac{1}{2n} \sum_{s \in S} SSD(y_s(t))$$

The sample-reference distance (SRD) is the deviation of one sample to the solution S_{ref} which we assume to be close to the underlying truth. As described in 3.2 we get S_{ref} by using SciPy's LSODA method with low error tolerance. The mean SRD is then defined as

$$MSRD(y(t)) = \frac{1}{n} \sum_{s \in S} \|y_{ref}(t) - y_s(t)\|$$

To make the sample-reference distance and the sample-sample distance comparable we again choose $\| \cdot \|$ to be the the Euclidean norm. In fact there are several options of choosing $\| \cdot \|$ in both metrics. As our problem is three-dimensional, the Euclidean norm is an intuitive choice of making the measures interpretable as a distance in the three-dimensional space. When only using non-deterministic sampling-based solvers, the reference solution and, therefore the SRD, are unknown. In the best case, the SSD should be able to represent it, i.e. the variance of the samples should be able to cover the error (estimate) of the solver. To measure this property we can compare the SRD to the SSD in different ways.

Density Estimate of the SSD and SRD

For each sample $s_k \in S$ we can evaluate the endresult $y_k(t)$ and compute the sample-sample distance $SSD(y_k(t))$ to all of the other samples $S \setminus s_k$ and the sample-reference distance $SRD(y_k(t))$ to the reference solution $y_{ref}(t)$. Using those metrics as individual results for each sample a kernel density estimate over them can be computed. There are several possibilities of comparing two density distributions taking the position and width of the corresponding values, their overlap, and the shape of the distributions into account.
The position of a function represents the order of the error estimates, for samples with a large average error or a large intersample variance, the mean of the distribution is higher and centered on the right when looking at the plot. Overlapping areas indicate that the SSD can represent the SRD while non-overlapping areas and density estimates that are far away from each other indicate that the sample-sample distance is not able to cover the mean error of the samples. The shape of the SRD distribution gives information on how the endresults of the samples are distributed and represents the broadness of the density estimate of the states at the corresponding location. A broad distribution of the individual SRDs indicates that at least some samples are far away from the reference solution.
The shape of the SSD can give, for example, information about outliers within the samples. A broad range of the individual SSDs indicates that the samples are not equally distributed, i.e. a small number of samples is not able to represent the variety of the samples that the solver produces. The density estimates of the SSDs and SRDs of both solvers are shown in Fig. 3.4 and Fig. 3.3 respectively. The densities are normalized such that the largest density equals 1, the range of the plotted density is from the largest to the smallest value of the distances.
For error tolerances $\leq 10^{-6}$ and $\sigma \geq 1$ of the Noisy-State Solver, the density estimates of the SSDs and the SRDs are centered around 1 (Fig. 3.3 (a),(b),(c),(d),(e),(f)). For lower error tolerances both distributions are shifted to the left (Fig. 3.3 (g),(h),(i), indicating a lower mean SSD and a lower mean SRD for results with higher accuracy. In those solvers, the distributions over the SSD and SRD are shifted to the right with increasing noise-scale σ, i.e. introducing more noise yields higher SSDs and SRDs. This indicates that for small tolerances both, the mean SSD and the mean SRD, become larger

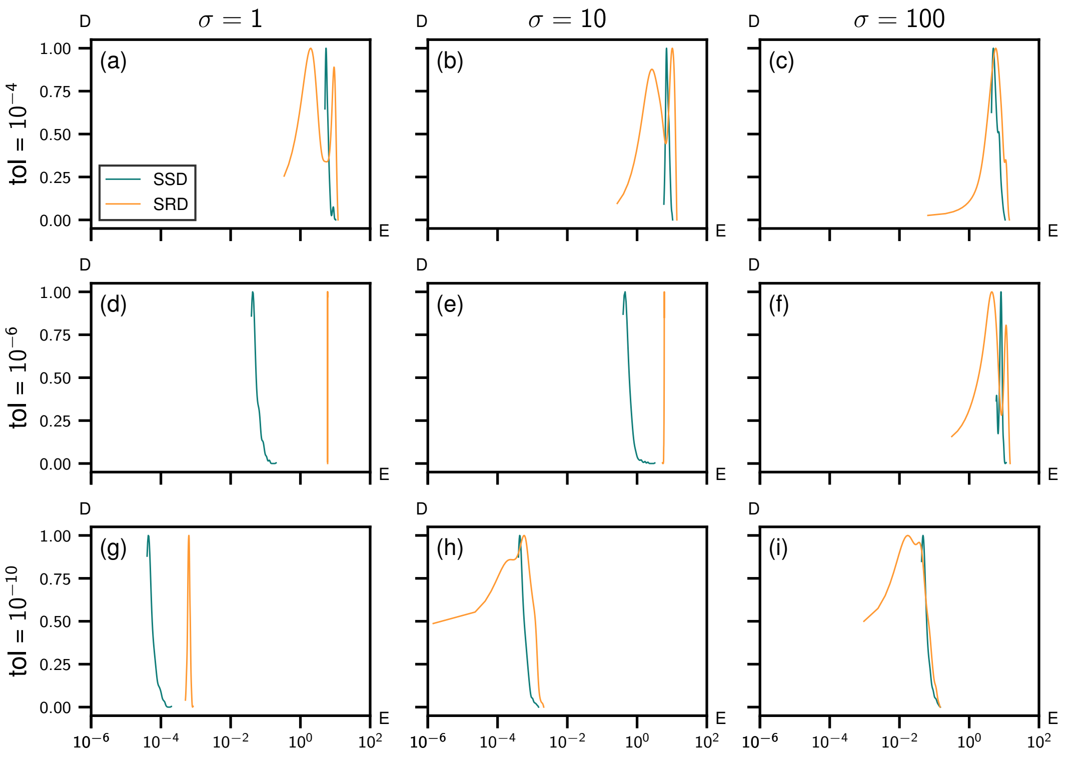

Figure 3.3: Density over the sample-sample (SSD) and sample-reference distances (SRD) of the Noisy-State Solver. E is the respective error estimate and D its normalized density estimation. With increasing noise-scale σ, the average distances within the samples and the distances to the reference solution get larger. With decreasing tolerance the average errors get smaller. The density estimates show overlapping areas in most of the settings.

with increasing noise-scale and decreasing tolerance. For smaller $\sigma \leq 100$ and $\geq 10^{-6}$ decreasing the tolerance by factor 10^2 decreases the SSD and the SRD also by a factor $\approx 10^2$. The overlap of the densities is high for most combinations of tolerances and noise-scales, indicating that most of the error is covered by the variance of the samples. Only for small $\sigma = 0.1$ and $tol \leq 10^{-4}$ the density estimates do not overlap.

The Noisy-Step Solver shows similar results (Fig. 3.4), with lower error tolerances the SRDs and the SSDs get smaller and for larger noise-scales τ, the mean SRD and SSD get larger. While the mean values of the densities deviate, independent of the tolerances, for small $\tau \leq 10^3$, the densities overlap in all settings where $\tau \geq 10^4$ at least partially (Fig. 3.4 (b),(e),(f),(h),(i)). With increasing noise-scale $\tau \geq 10^4$ the SSD and SRD highly overlap, for $\tau = 10^5$ and $tol = 10^{-6}$ the solver fails (Fig. 3.4 (c)). The density distributions of the Noisy-State Solver are of the same order in many settings. For fixed noise-scales (or tolerances) a corresponding tolerance (or noise-scale) such that the sample-sample distances can cover the error can be found in all cases. The density distributions of the Noisy-Step Solver indicate that the noise-scale has to be large to get that property and the solver fails for large noise-scales and low error tolerances. The lowest order of the distances where the densities overlap is the same for both solvers. The mode of the SSD and the SRD density estimates of the Noisy-Step Solver with the lowest tolerance is $\approx 10^{-4}$ for $\tau = 10^4$ (Fig. 3.4 (h)). The same result can be achieved with the Noisy-State Solver for the lowest tolerance and a noise-scale $\sigma = 1$ (Fig. 3.3 (g)).

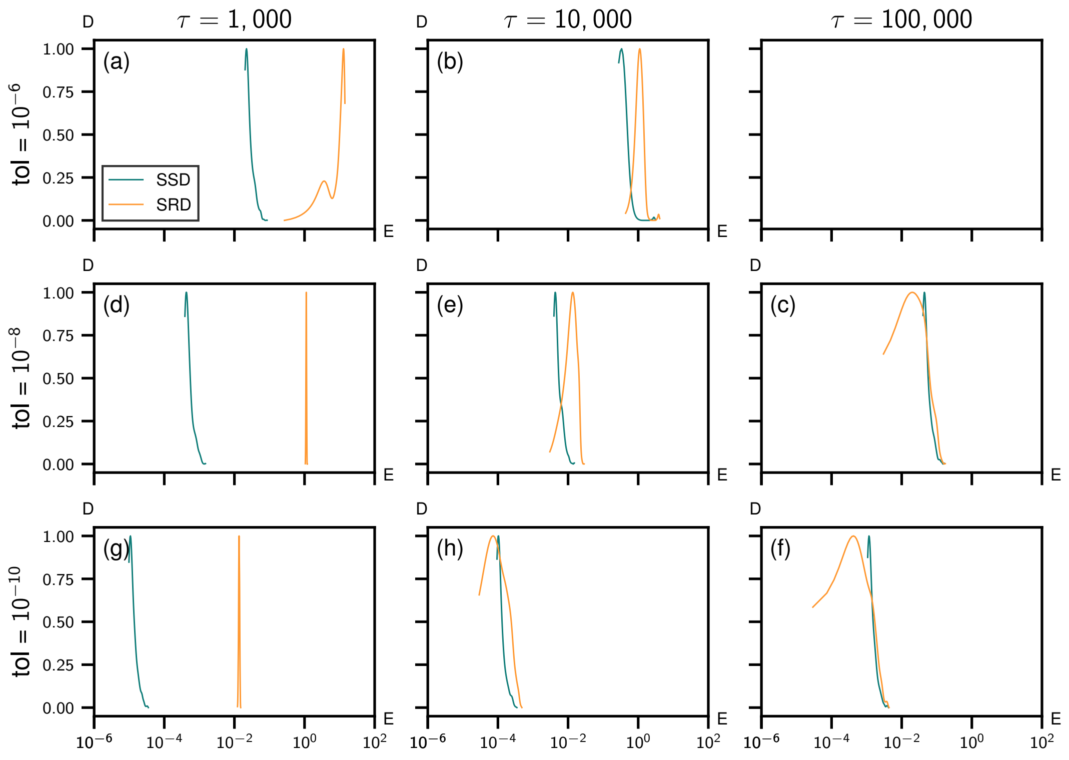

Figure 3.4: Density over the sample-sample (SSD) and sample-reference (SRD) distances of the Noisy-Step Solver. With increasing noise-scale σ the average distances within the samples and the distances to the reference solution get larger. With decreasing tolerance the average errors get smaller. The density estimates show overlapping areas in only a few settings, indicating that the solution is sensitive to the parameters.

Quotient of the SSD and SRD

If the mean SSD, i.e. the mean deviation between every sample to every other sample and the mean SRD, i.e. the mean deviation of every sample to S_{ref} are similar, the SSD is able to cover the reference solution and we call the solution *well-calibrated*. To interpret the SSD and the SRD with one metric, we can test the calibration of the solver by evaluating the quotient

$$Q = \frac{MSSD(y(t))}{MSRD(y(t))} \tag{3.3}$$

of both metrics. If this ratio is in $\mathcal{O}(1)$ the result is said to be well-calibrated, i.e. the mean sample distance is in the order of the error. A solver with a quotient Q that is not in $\mathcal{O}(1)$ but still closer to being in $\mathcal{O}(1)$ is *better-calibrated* than a solver with a much larger or smaller Q. For values < 1 the solver is overconfident: The variance of the samples is too small to cover the reference solution. A ratio > 1 represents underconfidence of the solver, the probabilistic solver induces too much uncertainty. The quotients of the mean SSD and the mean SRD are visualized in Fig. 3.5 for different tolerances and noise-scales. The SSD and SRD are evaluated for sample subsets of size $n \in \{2, 2^2, ..., 2^9\}$ and in all settings, the order of the quotient did not change for subsets of size $n \geq 4$.
Both solvers, the Noisy-State Solver as well as the Noisy-Step Solver, are well-calibrated for large noise-scales, i.e. by introducing a lot of noise, we can be almost sure that the reference solution is covered by the solver (Fig. 3.5 (c),(f)). The Noisy-State Solver is well-calibrated for $\sigma \in \{10, 10^2, 10^3\}$ for all of the tested tolerances, for smaller values of $\sigma \in \{0.1, 1\}$ the solver is well-calibrated for the largest tolerance $tol = 10^{-4}$ (Fig. 3.5 and Fig. A.4). While the calibration of the solver does not show any trend when comparing the large choices of σ, the best-calibrated solvers for smaller choices of σ are the solvers with the largest and the smallest tolerance.
This trend can also be seen in the Noisy-Step Solver. When increasing the noise-scale, the solver with the largest error tolerance is for all σ and τ always the first one to be well-calibrated. This can be explained by the fact that more errors are tolerated by the steprule which leads to a broader range of samples and can also be seen in Fig. 3.2 where setting the tolerance to $tol = 10^{-4}$ results in negative values $y_1(20) < 0$ in both solvers. While the Noisy-State Solver solution is already well-calibrated for small $\sigma \geq 10$ and the solver fails for $\sigma \geq 10^4$ (Fig. 3.5 (b)), the first setting where the solution of the Noisy-Step Solver is well-calibrated, independent of the error tolerance, is $\tau = 10^4$ (Fig. 3.5 (e)). This suggests that the noise-scale of the Noisy-Step Solver must be set higher compared to the noise-scale of the Noisy-State Solver. When increasing the noise-scale $\tau \geq 10^4$, the error tolerance should be decreased. To fulfill the convergence criteria of the uniform noisy steprule (equation 2.8), the stepsize has to be ≤ 1, this is not given when using large tolerances and the solver fails. Within the given framework for $\tau \geq 10^4$ increasing τ by factor 10 could be compensated by decreasing tol by factor 2, for lower tolerances the solver fails in all cases except for $\tau = 10^6$ and $tol = 10^{-8}$ where the solver fails in 155 of 512 cases. For better comparability, only settings where all solver runs were successful are being taken into account. It can be seen that the Noisy-Step

Solver can be well-calibrated, but there is no setting that works for all of the error tolerances and it is not straightforward to determine whether there is an intermediate solution that allows calibrating each Noisy-Step Solver without limiting possible values for the error tolerance.

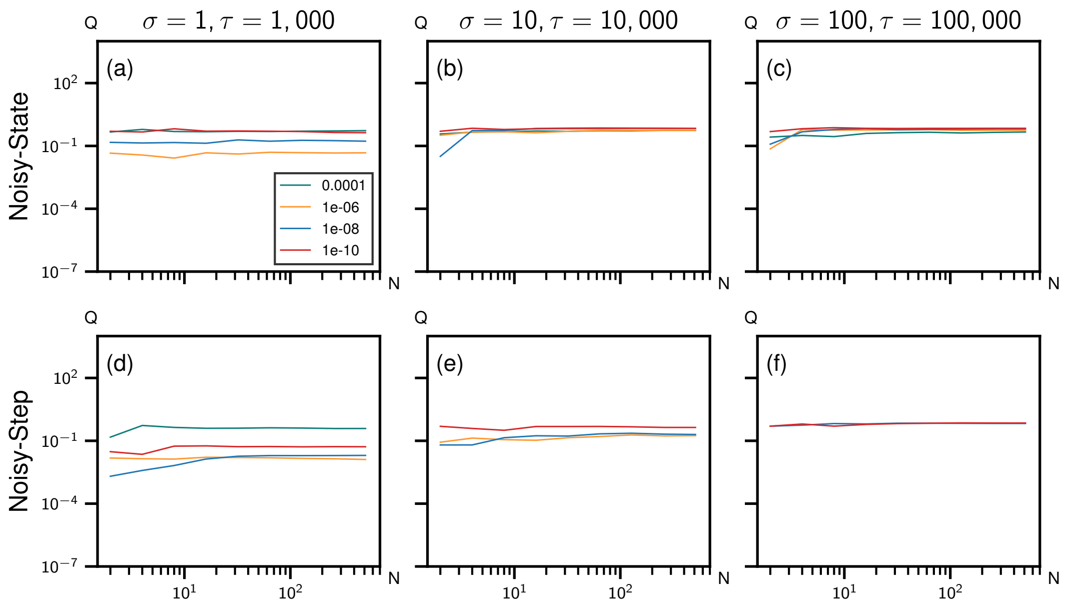

Figure 3.5: Quotient of sample-sample distance (SSD) and sample-reference distance (SRD). Q is the quotient of the SSD and the SRD, N is the number of samples, both are plotted on a log-scale. For $Q \approx 1$ (shaded in green) the solver is said to be well-calibrated. For larger noise-scales both solvers are well-calibrated, i.e. when increasing the overall error, the reference solution is captured by the samples – ignoring the absolute error that might be introduced into the system.

Choose the Best Calibrated Solver

The idea is to use the above-described metrics to calibrate the solvers by choosing the noise-scales σ and τ such that the SSD is able to represent the SRD. This can be achieved by combining the information from the quotient Q of the SSD and the SRD and their density estimate. As described above all of the solvers where $Q \approx 1$ are considered to be well-calibrated. If several solvers are well-calibrated, the solver with the smallest SRD is the solver that is still able to cover the reference solution without introducing too much uncertainty. This means that a well-calibrated solver is not necessarily a useful solver. While Q allows claims about whether the reference solution is covered by the solver, it neglects the fact that a broad range of samples results in a large SSD where many samples have a large SRD. The quotient of those metrics might therefore be ≈ 1 but mainly due to the fact that the SRD, i.e. the average error, is large. The best of several well-calibrated solvers is in the previously described settings always the solver with the lowest noise-scale and an SSD that covers the SRD. For the Noisy-State Solver this means that the solver in Fig. 3.5 (b) is considered to be more useful than 3.5 (c). For the Noisy-Step Solver the choice of τ is more dependent on the chosen error tolerance. Regarding calibration for $tol = 10^{-4}$ the best choice for τ is 100 (see Fig. A.4) and for lower tolerances, the noise-scale has to be increased.

3.3.3 Sample-Sample vs. Asymptotic Sample-Sample Distance

How Many Samples Do You Need?

The efficiency and usefulness of sampling-based methods highly depend on the number of samples that have to be drawn to get a representative range of samples (Marshall (1996)). The process of choosing the number of samples is a trade-off between minimizing the number of samples, i.e. the computation time, and keeping the variety of of the samples as large as possible. One way of tackling this issue is to think about how large a subset of samples has to be to show most of the variability of a large number of possible samples.
This can be evaluated by comparing the sample-sample distance (SSD) of a small subset to the asymptotic SSD – as soon as the SSD of a small sample subset is converged to the asymptotic SSD of a large sample subset if it covers the largest part of the intersample variance. We say that the SSD of k samples is converged to the asymptotic SSD if it is in the order of the asymptotic SSD of n samples, i.e. the quotient $Q_{SSD}(k, n)$ of the SSD and the asymptotic SSD is ≈ 1. We define the mean SSD of k samples $\{y_1(t), ...y_k(t)\}$ at location $t = 20$ as $SSD(k)$. The corresponding quotient of the sample-sample distance and the asymptotic sample-sample distance is then defined by:

$$Q_{SSD}(k, n) = \frac{SSD(k)}{SSD(n)} \tag{3.4}$$

When drawing i.i.d. samples it can be expected that

$$Q_{SSD}(k, n) < Q_{SSD}(n, n) = 1 \forall k < n \tag{3.5}$$

in most cases including more samples should increase the intersample variance. As first evaluations with 1024 samples suggested that a really small amount of samples already covers the intersample distances sufficiently, the following experiments were performed with 512 samples. In Fig. 3.6 the quotients $Q_{SSD}(k, 2^9)$ are visualized for sample subsets of size $k \in \{2^1, 2^2, ..., 2^9\}$. As the results are really similar for all tolerances and noise-scales the settings with

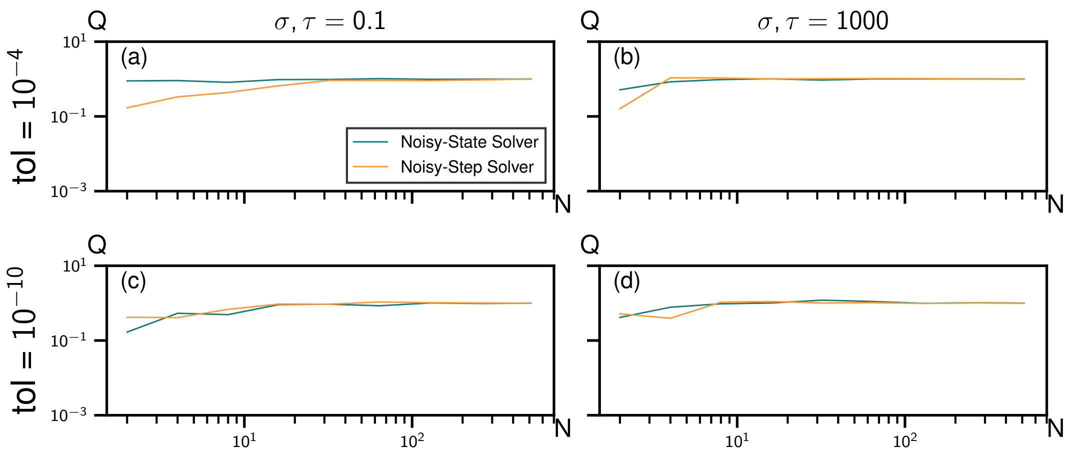

Figure 3.6: Quotient sample-sample distance (SSD) and asymptotic SSD. After at most 16 samples, the SSD captures most of the variance of all samples. Q is the quotient of SSD and asymptotic SSD, $N \leq 512$ is the number of samples, both axes are log-scaled. $Q = [0.5, 1.5]$ is shaded in green and marks the area where the SSD is said to be converged to the asymptotic SSD.

the lowest and highest noise-scales, where samples of all solvers are available, are shown, see A.4 for full results. All of the results show, on average, the expected trend of a slightly increasing $Q_{SSD}(k, n)$ for increasing k. For really small $k \leq 8$, the largest slope can be observed, i.e. the first samples are the most informative ones. $k \geq 4$ samples were in all cases sufficient to achieve $Q_{SSD}(k, n) \in \mathcal{O}(Q_{SSD}(n, n))$ for $n = 512$. With $k \geq 16$ all $Q_{SSD}(k, n)$ are in the range (0.5, 1.5), indicating that more samples are providing almost no new information.

3.3.4 Normaltests

Are the Samples Gaussian Distributed?

In general sampling-based solvers can represent every distribution over the different $y_s(t), s \in \{S\}$. Their main advantage, compared to filtering-based methods, is their possibility of modeling non-Gaussian distributed solutions. Therefore it is interesting to test the distribution over the states $y_s(t)$ for the different samples $\{s_1, ..., s_n\}$ at the one time point t for Gaussianity. This is performed for the first dimension $y_1(t)$ using the in *scipy.stats* implemented *normaltest* which is based on D'AGOSTINO and Pearson (1973) and includes skew and kurtosis of the given distribution. It performs a chi-squared test as proposed in d'Agostino (1971). $X_1, ..., X_n$ are a set of random samples of size n and $X_{1,n} < ... < X_{n,n}$ the ordered observations from it. The test statistic D

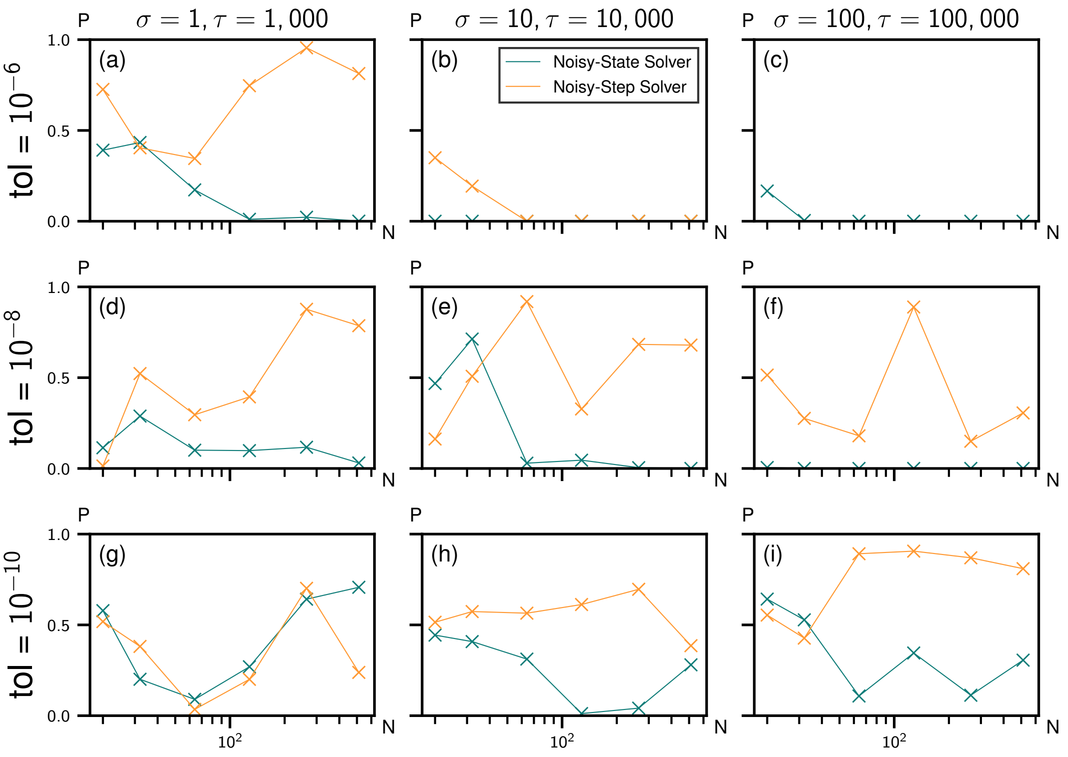

Figure 3.7: Normaltests at the final state. The crosses mark the result of the different subset sizes $N \in \{20, 2^5, 2^6, ..., 2^9\}$. The y-axis P describes the p-values of the normaltest, the grey background marks the range $[0, 0.05]$ where the Gaussianity assumption can be rejected.

is then given by

$$D = \frac{T}{n^2 S} \tag{3.6}$$

where

$$T = \sum_{i=1}^{n} (i - \frac{1}{2}(n+1)) X_{i,n} \tag{3.7}$$

and

$$S^2 = \frac{\sum (X_i - mean(X))^2}{n} \tag{3.8}$$

with $mean(X)$ being the sample mean. The expected value of D for samples that are drawn from a normal distribution is approximately

$$\mathbb{E}(D) = (2\sqrt{\pi})^{-1} \tag{3.9}$$

As the kurtosistest is only valid for $n \geq 20$, the smallest subset that is tested is of size $n = 20$. For each configuration, a normaltest is performed for sample subsets of size $n \in \{20, 2^5, 2^6, ..., 2^9\}$. Distributions for which the p-value p is high are *more Gaussian*, for $p \leq 0.05$ the null hypothesis is rejected and we can be almost sure that the distribution is non-Gaussian. While the p-value cannot be interpreted as the probability of being Gaussian, a higher p-value still indicates that the null hypothesis is not rejected with more confidence. It is therefore legitimate to interpret a test with a higher p-value as belonging to a distribution that is more Gaussian in the sense that its properties are closer to a Gaussian distribution. P-Values for subsets of size $n \geq 50$ are more relevant and expressive as the test is designed for larger sets of samples with $n \geq 50$ (d'Agostino (1971)). Finally, the most important p is that of the last sample subset, taking all of the samples into account.
In Fig. 3.2 it can be seen that the Noisy-State Solver with high absolute and relative error tolerances shows bifurcations and should therefore be considered to be non-Gaussian, in these cases the normaltests (Fig. 3.7) allow to reject the Gaussianity assumption. While the null hypothesis cannot be rejected during the first tests, p decreases, and once $p < 0.3$ it constantly sinks. Similar results can be seen for the Noisy-Step Solver. By decreasing the error tolerance to $tol \leq 10^{-6}$, almost all settings of the Noisy-Step Solver come with results where the distribution is such that the null hypothesis cannot be rejected. There are only two settings where this does not hold: Relative and absolute tolerances of 10^{-6} and a noise-scale of 10^4 and $tol = 10^{-10}$ and $\tau = 0.1$.
Comparing that result to 3.2 it can be seen that in the first setting the distribution shows a slight bifurcation. Overall it can be said that the Gaussianity assumption is rejected for high tolerances, i.e. many allowed errors, for all of the noise-scales and both solvers (Fig. A.6). The Noisy-State Solver shows a trend – for lower tolerances the p-values get higher and the distributions get *more Gaussian*. Only a small subset of the settings of the Noisy-State Solver allows to keep the null hypothesis, therefore most of the results are probably not Gaussian-distributed. For high $\sigma \geq 10^3$ and all of the tolerances none of the results are assumed to be Gaussian-distributed (Fig. A.6). For tolerances $tol \leq 10^{-8}$ and noise-scales $\sigma \geq 10$ the hypothesis test yields similar results and none of the distributions over the solution is Gaussian according to the chi-squared test (Fig. 3.7 (b),(c),(e),(f)).
For smaller perturbations and lower error tolerances, the results of the Noisy-State Solver are almost always assumed to be Gaussian-distributed, according to the chi-squared test. The same holds for the Noisy-Step Solver, with one exception for $tol = 10^{-10}$ and $\tau = 0.1$. Taking those results into account it can therefore be said that the results of the Noisy-Step Solver are overall more likely to be Gaussian-distributed than those of the Noisy-State Solver. The Gaussianity of the Noisy-Step Solver is almost independent of the noise-scale τ and the sample solutions of the Noisy-State Solver are more likely to follow

a Gaussian distribution when the error tolerances and the noise-scales σ are low. It is necessary to add that even a distribution that is non-Gaussian according to that test might still be close to a Gaussian distribution in a sense that its main properties can be described by a mean value that is close to the median of the distribution and almost symmetric tails around that mean. An example for such a setting is the density at the endresult of the Noisy-State Solver for $tol = 10^{-8}$ or $tol = 10^{-6}$ and $\sigma = 1$ (Fig. 3.7 (a),(d)), in both cases the normaltests reject the Gaussianity assumption while both output densities *look Gaussian* (see Fig. 3.2) in the above-described sense.

3.3.5 Filtering-based Solvers

The focus of this work lies on sampling-based ODE solvers. Their output is non-deterministic and able to represent any distribution over the solution of the ODE. But sampling-based solvers need to solve the ODE multiple times to achieve a probabilistic output. Filtering-based methods, as provided in ProbNum, give a probabilistic, deterministic solution by solving the ODE only once. ProbNum is still under construction and the applied benchmarks are therefore just a glimpse into preliminary results of the filtering-based solvers. To make the results comparable, the previously described benchmarks were applied to 512 samples of a filtering solution. The solution can be computed using ProbNums *probsolve_ivp()* (pro) with the following setting for error_tol $\in \{10^{-4}, 10^{-6}, ..., 10^{-14}\}$:

```python
import numpy as np
from probnum import diffeq

ivp = diffeq.ode.ivp_examples.lorenz([0.0,20.0], initrv = np.
    array([0.0, 1.0, 1.05]))

diffeq.probsolve_ivp(f=ivp.rhs, t0=0.0, tmax=20.0, y0=ivp.
    initrv, df=ivp.jacobian, method="EK1", algo_order=4, atol=
    error_tol, rtol=error_tol)
```

Listing 3.1: Filter settings. Using ProbNum and the extended Kalman filter, solving the ODE is basically one line of code.

The parameters are set to the same values as those of the sampling-based solvers which makes the results comparable. The results of the filtering-based solutions are summarized in Fig. 3.8. As the samples are drawn from a Gaussian-distributed random variable, the results of the normaltest in Fig. 3.8 (c) indicate that 512 samples are enough to show this property of the posterior samples. The quotient of the SSD and the asymptotic SSD in Fig. 3.8

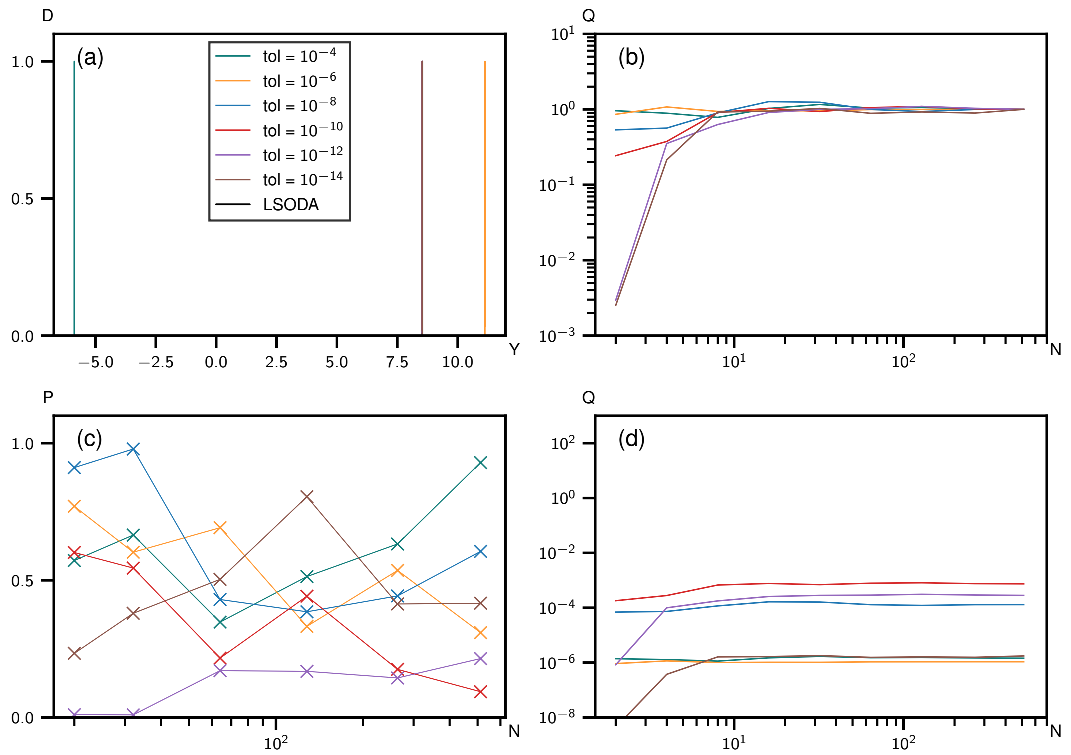

Figure 3.8: Benchmarks for the filtering-based method. (a) corresponds to 3.2, (b) to 3.6, (c) to 3.5 and (d) to 3.7. All of the sample subsets are Gaussian-distributed for subset sizes $N \geq 2^6$ and their SSD covers the asymptotic SSD for small subsets. For lower error tolerances the solution of the filtering-based method is close to the reference solution. In the chosen settings the SSD can not cover the SRD.

(b) indicate that, similar to the sampling-based solvers, around 8 samples are sufficient to cover the SSD. In the density estimate over the final states in Fig. 3.8 (a) it can be seen that higher error tolerances lead to results that are further away from the reference solution, for tolerances $\leq 10^{-8}$ the solver solution is close to the reference solution. While the density distributions are not visually separable, the quotient of the SSD and SRD in 3.8 (d)indicates that the most well-calibrated solvers are those with high error tolerances. As first experiments with the same error tolerances that were used for the sampling-based solvers tended to get more well-calibrated for high tolerances, tolerances up to *tol* $= 10^{-14}$ were evaluated. It can be seen that the best-calibrated solvers are those with error tolerances in $\{10^{-8}, 10^{-10}, 10^{-12}\}$. Those results of the filtering-based methods cannot be seen as final benchmarking results but are only intermediate results.

Chapter 4

Discussion

The main goal of this part is to bring the previous benchmarking results together. The aim is to find settings where the solvers are the most informative in a probabilistic sense and to figure out whether they are useful. By comparing the solvers among each other the individual benefits of each of the probabilistic solvers can be evaluated.

4.1 Bring out the Best of the Solvers

Noisy-State Solver

The Noisy-State Solver perturbs the states $y(t)$ at time point t with Gaussian-distributed noise. The estimated error ξ(t) defines the variance of the Gaussian-distributed noise with zero-mean. Intuitively it makes sense to choose the noise-scale such that the final result $\hat{y}(t) \in [y(t) - \xi(t), y(t) + \xi(t)]$, i.e. setting $\sigma = 1$. It can be seen that this intuition is at least partially correct. While a solver with noise-scale $\sigma = 0.1$ only covers the true solution for the highest error tolerance with a, compared to the lower tolerances, high sample-reference distance, and a high sample-sample distance, the solver with noise-scale $\sigma = 1$ covers the true solution for most of the settings. This can be seen when combining the results from Fig. 3.5 and Fig. 3.3 indicating that $\sigma = 1$ is a better choice for the noise-scale than $\sigma = 0.1$.
The solutions S_{tol} of the underlying deterministic solvers with the same tolerances as corresponding Noisy Solvers (see Fig. 3.2) are for all chosen tolerances $tol \leq 10^{-10}$ not capable to cover the reference solution S_{ref}, i.e. $|S_{tol} - S_{ref}| \geq 10^{-3}$. It can be seen that by choosing $\sigma \geq 1$ all of the Noisy-State Solvers with $tol \geq 10^{-8}$ can cover the solution. While the Noisy-State Solver is well-calibrated for all $\sigma \geq 10$ and close to being well-calibrated for $\sigma < 10$ the calibration for higher noise-scales comes mostly due to an overall high variance, see Fig. 3.3. Comparing the reference solution to the density

over the final states (Fig. 3.2) shows that the calibration of those solvers with high noise-scales can be traced back to a bimodal endresult with one local maximum for $\hat{y}(t)$ being slightly larger than the reference solution and a second smaller local maximum $\hat{y}(t) < y(t)$. This bimodality can also be found in Fig. 3.3 for the densities over the sample-reference solutions, indicating that the reference solution is covered but many samples are far away from it. Given all benchmarking results, it can be concluded that the choice of σ is less important for small tolerances, for large tolerances setting $\sigma \geq 10$ is sufficient to cover the reference solution. The best choice for the noise-scale is therefore, given the benchmarking results, $\sigma \in [10, 100]$.
Overall the Noisy-State Solver shows not too bad performance in all of the settings. Comparing the samples of the Noisy-State Solver and the deterministic solution of the same tolerance solution to the reference solution, both are on average larger than the reference solution. The Noisy-State Solution is therefore in no setting recognisably worse than the deterministic solution with the same tolerance.

Noisy-Step Solver

The Noisy-Step Solver showed overall similar results to the Noisy-State Solver. The higher the noise-scale, the more likely it is that the sample-sample distance can cover the reference solution. This capacity of covering the reference solution comes with higher variance and a higher sample-sample distance. While the sample-sample distance is in all cases smaller compared to the solutions of the Noisy-State Solvers with similar performance, i.e. solvers that are equally calibrated, there are overall less settings where the solvers are well-calibrated. This makes it harder to find a suitable choice for the noise-scale τ that works for all error tolerances. While higher noise-scales lead to more-calibrated results, they are not suitable for high tolerances due to stepsizes ≥ 1 that are not allowed using the uniform noisy steprule. When using only the uniform noisy steprule (equation 2.8), without projecting the result $\hat{y}(t)$ that is evaluated at $t+\xi(t)$ back to position t, no solver is well-calibrated, i.e. there is no noise-scale that covers the reference solution (see A.4). When comparing the Noisy-Step Solver results to the reference solution with the same error tolerances, it can be seen that the modes of the Noisy-Step Solver are in most cases closer to the reference solution than the same tolerance solution (Fig. 3.2 (a),(d),(e)).

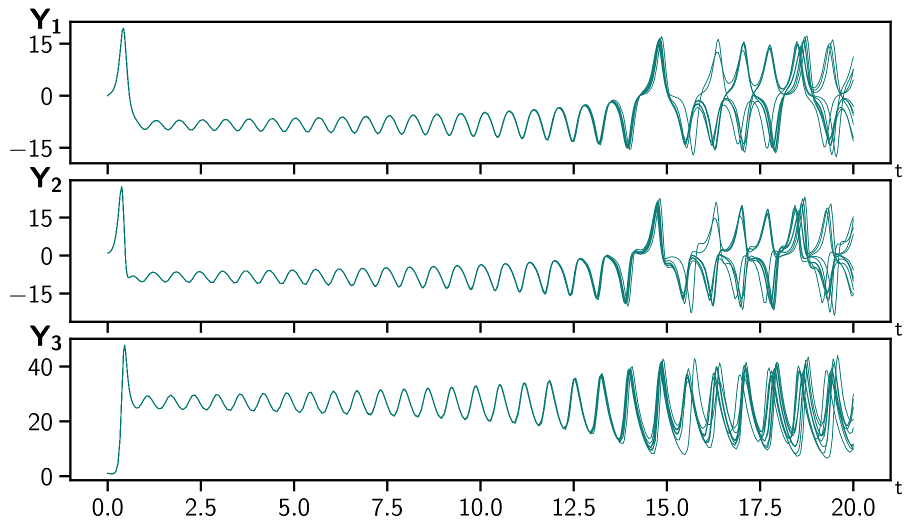

Figure 4.1: Trajectories split dimensionwise. From bottom to top the three dimensions of a Noisy-State Solver with $tol = 10^{-4}$ and $\sigma = 1$ are plotted for 1 samples. As the discrete output at the endresults indicates, most samples can be assigned to one of two groups of trajectories.

4.2 Choose a Useful Solver

Compared to the deterministic solution all of the Noisy Solvers allow to gain further insight into the chaotic behaviour of the system. Small perturbations are sufficient to yield samples that are, at least for parts of the time span, not even close to each other. Especially for high error tolerances where a deterministic solver with the same tolerance as the corresponding Noisy Solver is further away from the reference solution, the probabilistic solvers show a high variance. As the variance of the Noisy-State samples depends directly on the error estimate, this high sample variance indicates that the error estimate is high enough to lead to completely different outputs and should therefore not be neglected. The bimodalities represent a bifurcation at at least one evaluation point. In the continuous case in Fig. 4.1 it can be seen that the interpretation of the bimodality in the plot over the endstate as a bifurcation of the trajectories is plausible. Taking into account that a maximum amount of 16 samples is able to cover enough intersample variance in all Noisy Solvers with all noise-scales and tolerances, both solvers can be seen as useful in the sense that they allow making assumptions about the stability of the solver without too much computational cost.
In many cases, especially with high tolerances or large noise-scales, the solver output shows bimodal outputs, indicating that a small change in the input of the solver at a previous point leads to completely different solutions. These so-

lutions are also interesting when comparing them to the filtering-based solvers as these only allow for a Gaussian-distributed variance and are deterministic. Wherever the distribution over the samples is unimodal, the mode is centered around a positive value $y_1(20) > 0$, except for the Noisy-Step Solver and a high error tolerance $tol = 10^{-4}$. This does hold for the Noisy Solvers and for the filtering-based solver.
While the distributions over the samples are really similar for small error tolerances, seeming to converge to the reference solution, the samples for lower tolerances are in both sampling-based solvers diverse. But which outcome is the one that gives the most insight into to system?

4.3 Why the Lorenz System?

The Lorenz system is an interesting but hard problem, mostly due to its chaotic behaviour. While the system itself involves no random stochasticity, it is known for the butterfly effect (3.1) and is therefore deterministic but unpredictable in practice. For the chosen parameters, i.e. for $b = 28$ the Lorenz system is *symmetric* in (y_1, y_2), i.e. if $(y_1(t), y_2(t), y_3(t))$ is a solution, then $(-y_1(t), -y_2(t), y_3(t))$ is a solution as well (James Hateley). For the solution at the final state this means that the LSODA reference solution y_{ref} is only a part of the solutions that satisfy the ODE.
It can be seen in Fig. 3.2 that for both Noisy Solvers noise-scales and error tolerances exist, such that both, positive and negative y_1 values are reached. This is never the case in the deterministic solvers based on filtering methods or classic methods as RK45 or LSODA. In the continuous solution Fig. 4.1 all of the trajectories are close to each other until a certain time point $t_{div} > 14$ at which the trajectories over the first and second dimension start to diverge. While the solution over the third dimension does also show more perturbance after t_{div} the solution over the first two dimensions indicates the symmetry property of the Lorenz system. The Noisy Solvers for low error tolerances allow therefore to guess a property of the Lorenz system, which the reference solution does not show at all, but come with a huge variance that decreases the information about the actual numerical value of $y_1(t_{max})$. The positive mode of the Noisy Solvers does in no case correspond directly to the LSODA solution y_{ref}. The bimodal noisy solutions show a high variance with a slightly shifted mode compared to the LSODA solution.
Taking into account that the perturbations of the Noisy Solvers fulfill convergence criteria and are mathematical useful, one might ask whether a perturbation that does not fulfill any criteria, i.e. simple random noise, would yield similar results.

4.4 Limitations of the Solvers

Especially the Noisy-Step Solver shows a bias in the solution and only works well for small ranges of noise-scales and tolerances. The Noisy-State Solver works on a broader range of noise-scales and seems to be less sensitive to the noise-scale in general. In a true sampling-based and noisy setting where the reference solution is unknown, the order of the global error should be covered by the order of the local error estimate. The SSD should therefore cover the SRD and the noise-scale has to be chosen without knowing the SRD. In ODEs there are mainly two sources of errors, truncation and rounding errors (Shampine (2005)). Rounding errors occur due to finite precision of floating-point arithmetic and can become a problem in solving ODEs for very small stepsizes or error tolerances (Shampine (2005)). The truncation error is the error that is caused by the approximation, i.e. the local discretization error from time point t to $t + 1$. The local errors are approximated by the solver itself and in classic methods, the global error is not estimated and controlled by the solver (Shampine (2005)). There exist various methods to determine upper bounds of the global error given local truncation errors (Constantinescu (2015), Zadunaisky (1976)).
The global error is the error that is propagated along the numerical integration process. Making this error a part of the solution is one of the most important properties of probabilistic ODE solvers, independent of the underlying method. The Noisy Solvers are both sensitive to the noise-scale and it can be seen that their calibration is a trade-off between accuracy and covering the reference solution.
The filtering-based solvers do not have an additional noise-scale but their uncertainty depends on the initialization of the solver, the introduced covariance and other settings. Therefore the chosen setting probably does not bring the best out of the filtering-based solvers.

4.5 Simplify the Current Solvers

Use Only the Noisy Steprule

One idea of expanding the Noisy-Step Solver is to reduce it to the noisy steprule. The results are, in all settings, uncalibrated solvers (see Fig. A.4). Especially for high error tolerances the results with the noisy steprule are worse compared to all of the other solvers, incapable of covering the reference solution. As the intersample variance of those high tolerance solvers is still the highest, the results are better-calibrated compared to lower error tolerances, see Fig. A.4. For lower tolerances, the solution of the uniform noisy steprule is really close to the solution of the Noisy-Step Solver with a much lower sample variance and therefore worse calibration. Using only the noisy steprule without projecting the solution back to the original time point is therefore considered to be not useful.

Interpolate the Noisy-State Solution Differently

The proposal by Conrad et al. (2017) and current implementation of the interpolation of the Noisy-State Solver includes n independent Kalman posteriors for n steps of the solver. This is computationally expensive and the final dense output is a kinky, non-smooth function. It might therefore be interesting to interpolate the dense output as described in the alternative approach in 2.3.

4.6 Outlook

There are various ways of expanding the current work. Some possibilities that are easy to realize by expanding the current implementation are described in the following section. The current benchmarks can be applied to other ODEs. One interesting problem might be the *three-body problem* which can be described in terms of the Newtonian equations and describes the motion of three bodies that solely depends on their mutual gravitation (Musielak and Quarles (2014)). In general, with the current setting, the benchmarks can be applied to every system of ODEs that can be solved by a classic explicit method, i.e.every non-stiff ODE.
Currently, all evaluations were performed given the solution at the endpoint of the time span. This does not take the interpolation between the approximations of the discrete states into account. Especially in the case of the Noisy-State Solver where the dense output is given by a concatenation of i.i.d Kalman filtering solutions, it might be interesting to consider not only the solution at one time point but the continuous trajectory.
Possibilities of expanding the given probabilistic sampling-based solvers were described in 4.5 and can be easily implemented. This adaptation only changes the interpolation of the dense output of the Noisy-State Solver by fitting one Kalman posterior to the discrete state solution. In that case noise is introduced into the Kalman filter as a stepwise function that depends on the error estimate of the next state evaluation.
In equation 2.7 the Log-normal noisy steprule is given. The data of that steprule has not been taken into account as the first evaluations seemed to be slightly worse than those of the uniform noisy steprule (equation 2.8). As the Log-normal rule and the Uniform-rule are only two steprule suggestions, there exist various other rules that fulfill the given, too.
In the Noisy-Step Solver both steprules showed results that were better-calibrated when increasing the noise-scales, this dependency can be further

investigated. The chosen benchmarks are just a subset of possible benchmarks. Adding more benchmarks, either by including the previously described setting into the current framework or by finding new interesting metrics increases the expressiveness of the current investigations.

4.7 Noisy or Not?

One question that still needs to be answered is whether the Noisy Solvers are considered being useful and advantageous compared to classic deterministic and filtering-based methods. Thousands of samples, several metrics and comparisons later it can clearly be said: It depends. While classic methods neglect uncertainty in computation and the fact that approximations are approximations to the solution and not *the* solution, probabilistic methods make this uncertainty explicit. Especially in problems such as the Lorenz system this can be extremely useful to exploit the system itself, see where chaotic behavior can be observed and how slight changes can change the whole output. But introducing noise comes with the disadvantage that the solution is a distribution over potentially really noisy samples. Without a reference solution it is hard to discover biases in that distribution and, e.g. for the symmetric Lorenz system, even harder to come up with a numerical solution that describes the system. This is achieved much easier with filtering-based methods, the solution can be described by a mean and covariance function, but those filtering-based methods always yield a Gaussian-distributed outcome and are not able to show bifurcations or a non-Gaussian variance. None of the solvers is perfect, all of them come with advantages and disadvantages. To gain the most insight into the behaviour of a system that is described by an ODE, it is therefore recommendable to not only trust the approximation of one solver. Edward Lorenz has put it in a nutshell many years ago:

Chaos – when the present determines the future, but the approximate present does not approximately determine the future. - **Edward Lorenz**

It should always be remembered that computation comes with approximation and therefore always influences the solution.

Appendix A

Further Tables and Figures

Plots for all parameter settings, including the solver which uses the noisy steprule from equation 2.9 without projecting the evaluation locations (see 2.4.1).

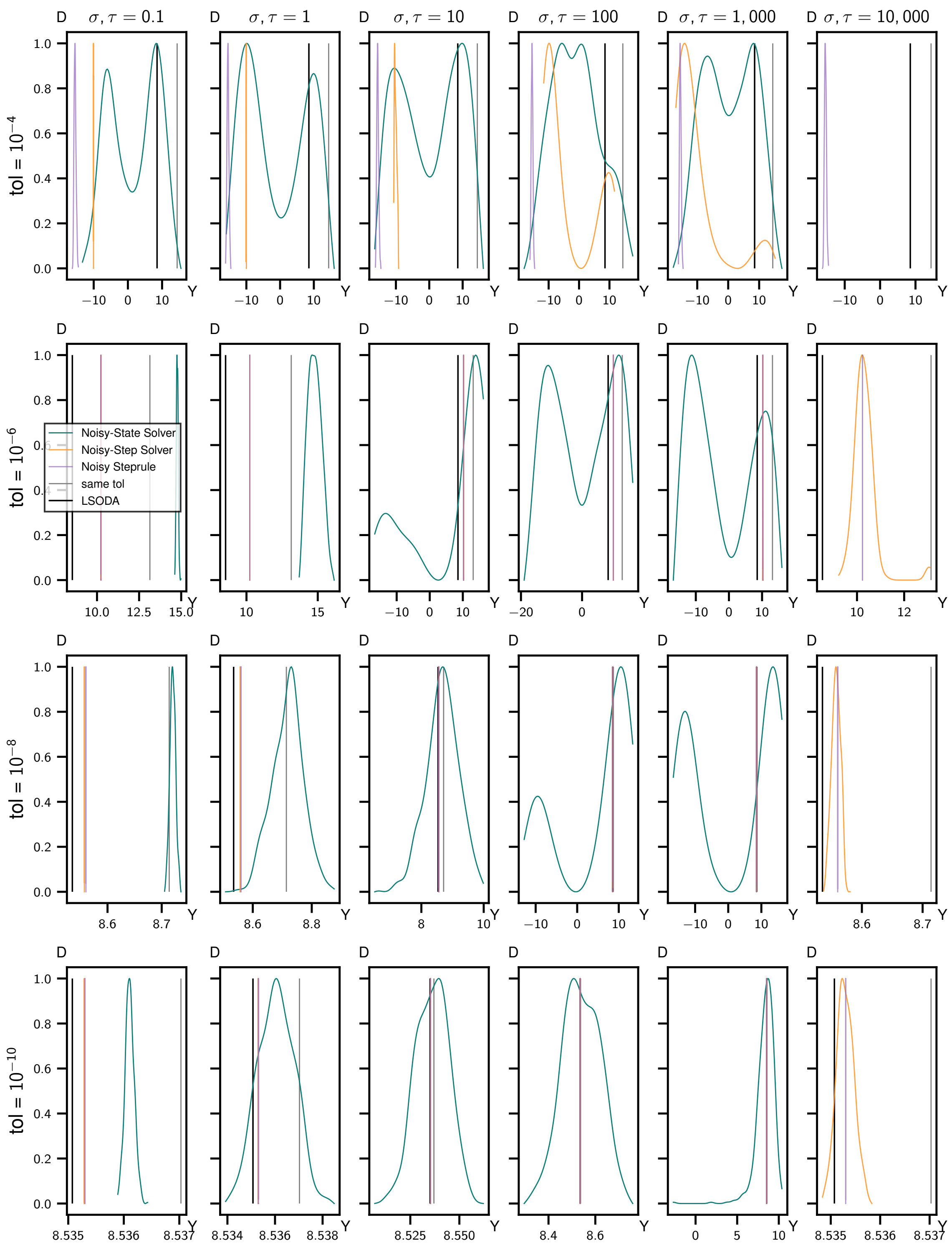

Figure A.1: Density at the final state $Y_1(20)$ for various noise-scale and tolerance combinations. Expanded version of Fig. 3.2.

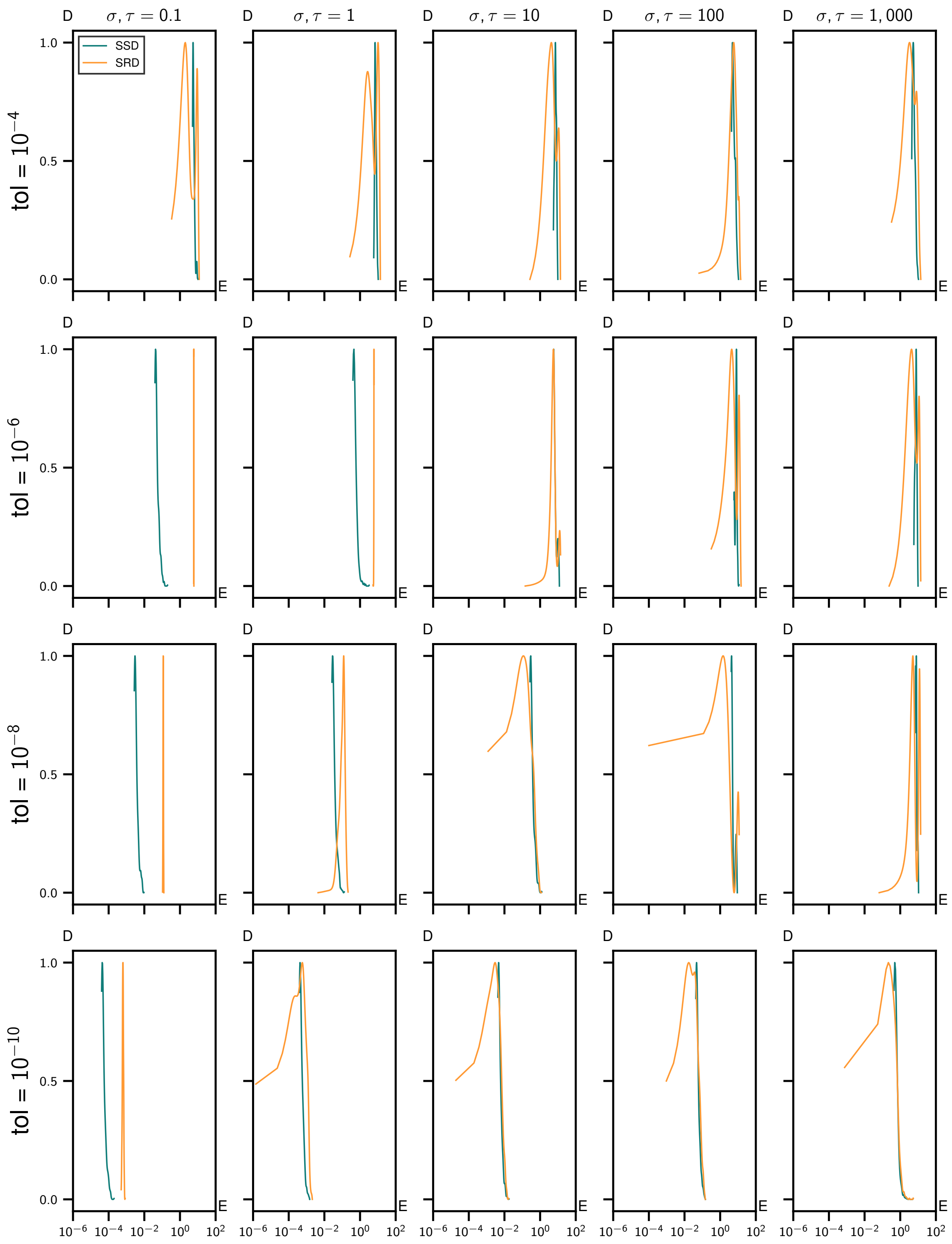

Figure A.2: Density estimation of the SSD and SRD of the Noisy-State Solver for various noise-scale and tolerance combinations. Expanded version of Fig. 3.3.

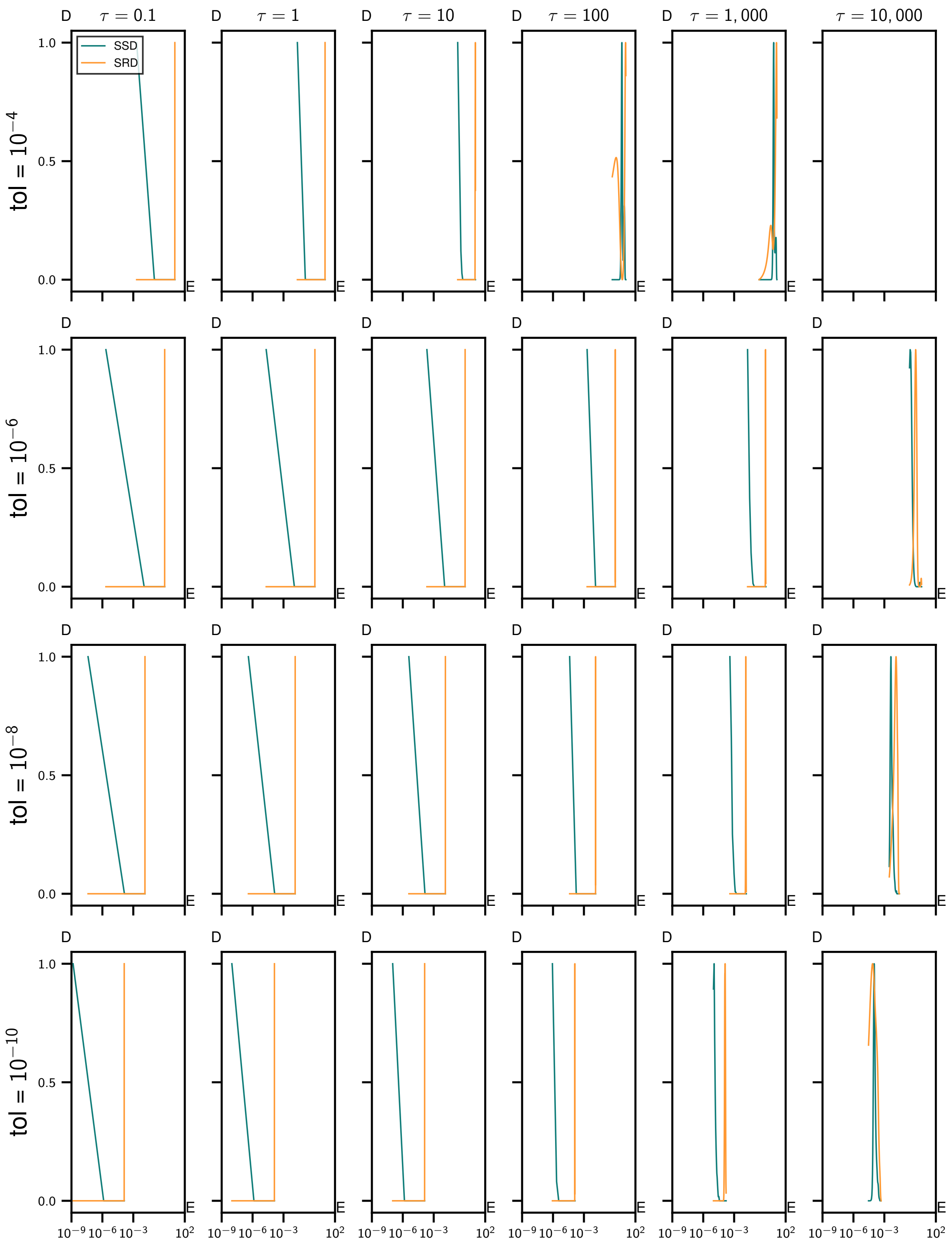

Figure A.3: Density estimation of the SSD and SRD of the Noisy-Step Solver for various noise-scale and tolerance combinations. Expanded version of Fig. 3.4.

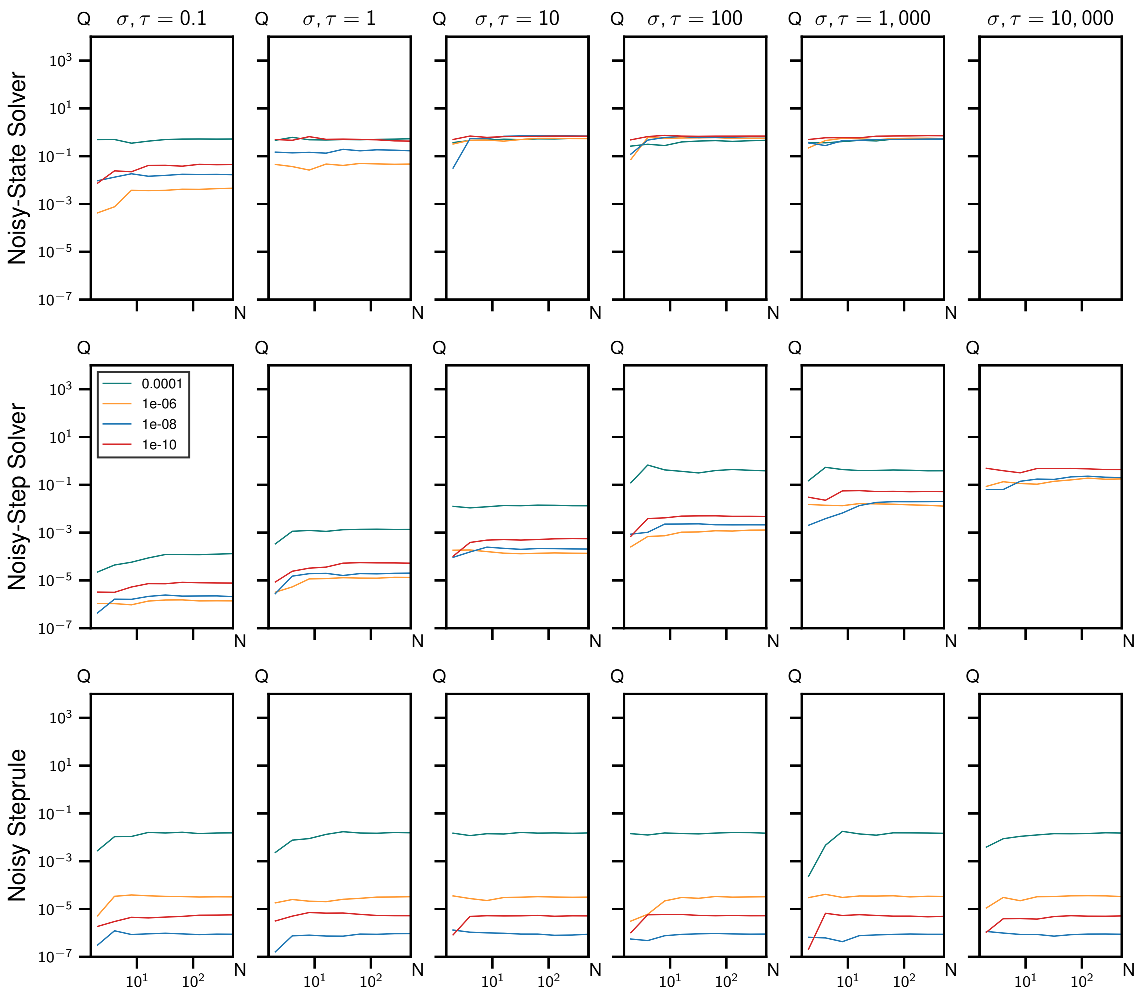

Figure A.4: Quotient sample-sample distance and sample-reference distance for various noise-scale and tolerance combinations. Expanded version of Fig. 3.5.

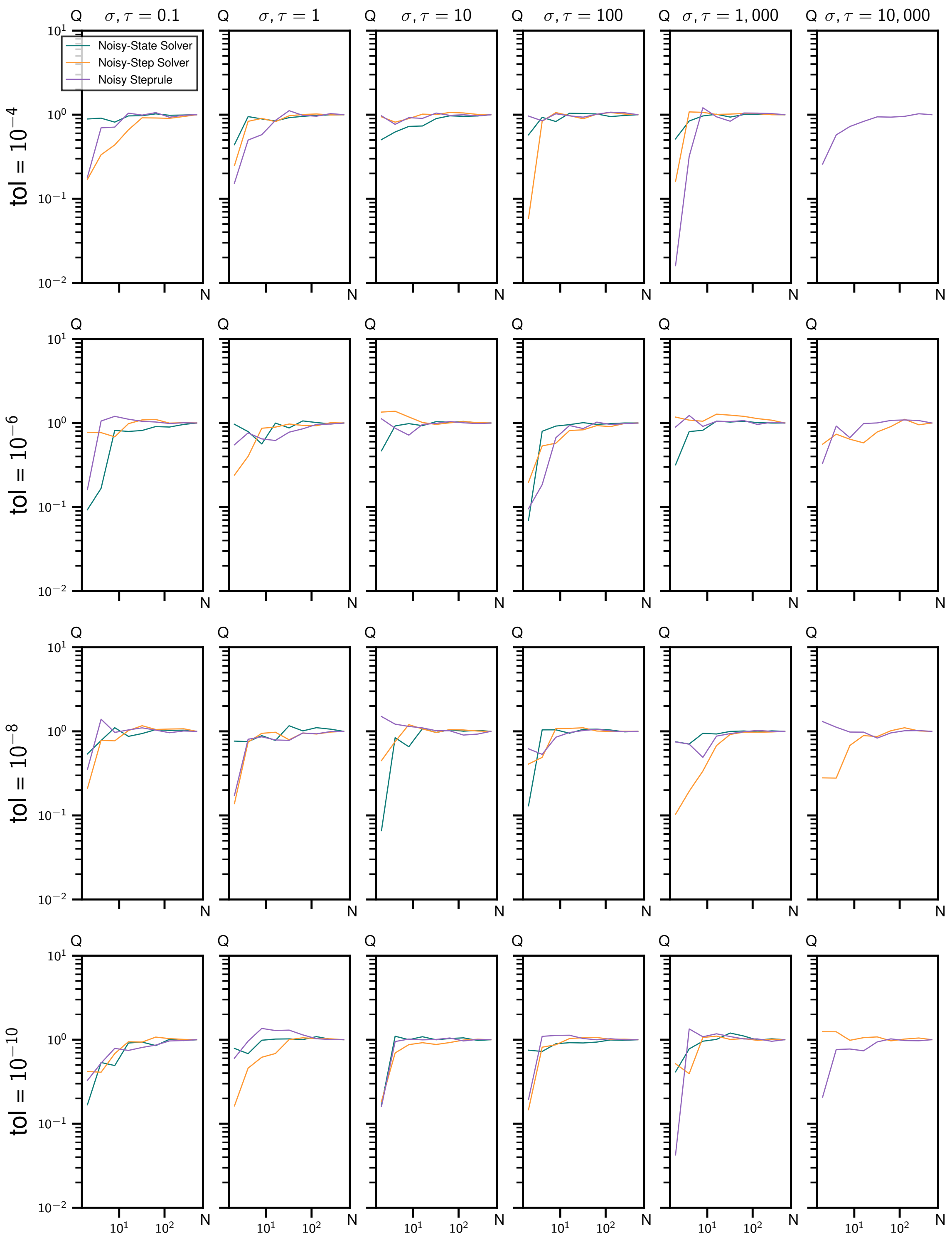

Figure A.5: Quotient of sample-sample distance and asymptotic sample-sample distance for various noise-scale and tolerance combinations. Expanded version of Fig. 3.6.

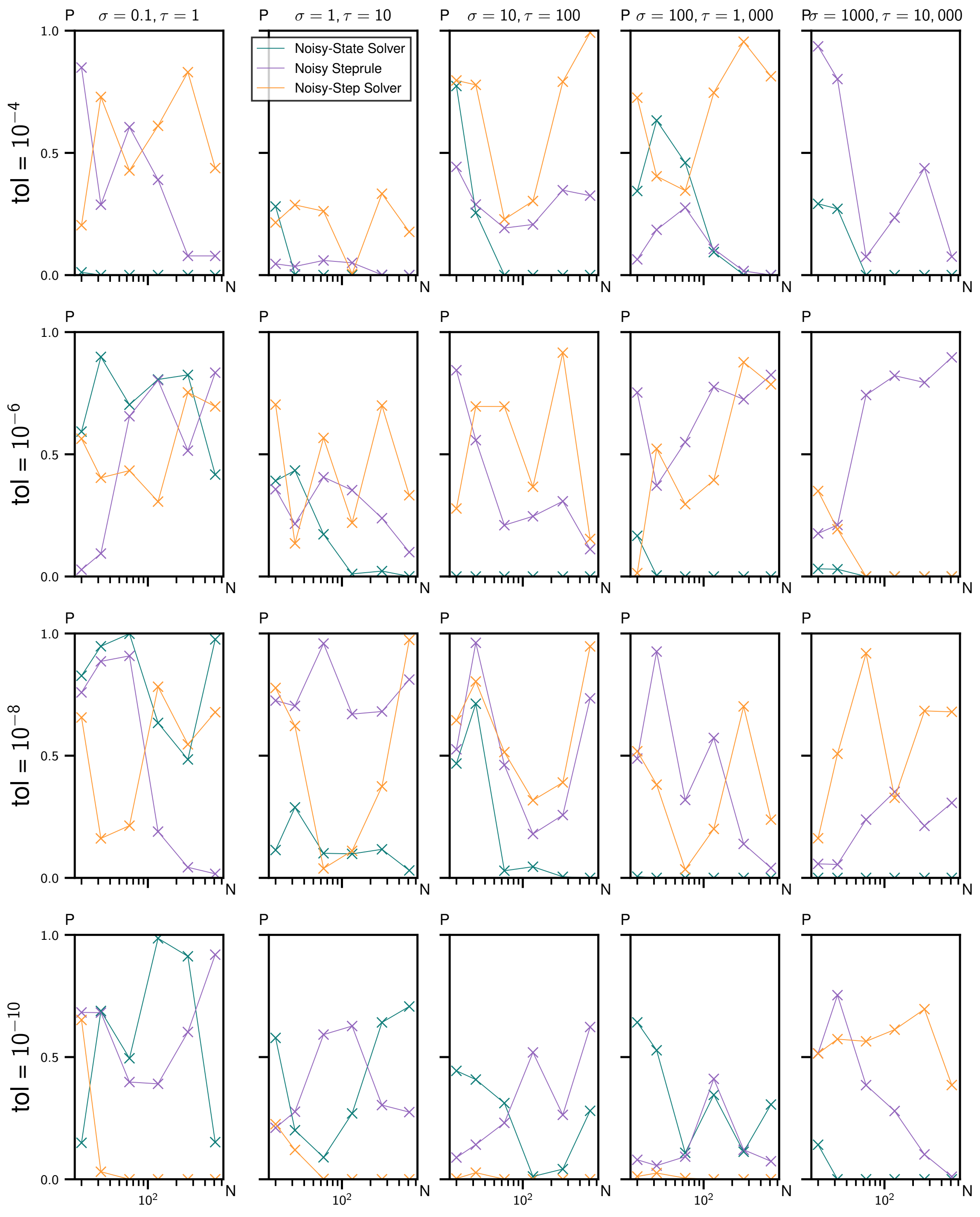

Figure A.6: Normaltests for various noise-scale and tolerance combinations. Expanded version of Fig. 3.7.

Bibliography

ProbNum IVP Documentation. `https://probnum.readthedocs.io/en/latest/automod/probnum.diffeq.probsolve_ivp.html`. Accessed: 2021-04-17.

SciPy RK-methods Documentation. `https://github.com/scipy/scipy/blob/master/scipy/integrate/_ivp/rk.py`. Accessed: 2021-03-22.

A. Abdulle and G. Garegnani. Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration. *Statistics and Computing*, pages 1–26, 2020.

J. C. Butcher. A history of runge-kutta methods. *Applied numerical mathematics*, 20(3):247–260, 1996.

J. C. Butcher and N. Goodwin. *Numerical methods for ordinary differential equations*, volume 2. Wiley Online Library, 2008.

C. Chicone. *Ordinary differential equations with applications*, volume 34. Springer Science & Business Media, 2006.

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

E. Constantinescu. Estimating global errors in time stepping. *arXiv preprint arXiv:1503.05166*, 2015.

E. L. Crow and K. Shimizu. *Lognormal distributions.* Marcel Dekker New York, 1987.

R. D'AGOSTINO and E. S. Pearson. Tests for departure from normality. empirical results for the distributions of b 2 and b. *Biometrika*, 60(3):613–622, 1973.

R. B. d'Agostino. An omnibus test of normality for moderate and large size samples. *Biometrika*, 58(2):341–348, 1971.

J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

I. Hashim, M. Noorani, R. Ahmad, S. Bakar, E. Ismail, and A. Zakaria. Accuracy of the adomian decomposition method applied to the lorenz system. *Chaos, Solitons & Fractals*, 28(5):1149–1158, 2006.

P. Hennig and S. Hauberg. Probabilistic solutions to differential equations and their application to riemannian statistics. In *Artificial Intelligence and Statistics*, pages 347–355. PMLR, 2014.

James Hateley. `https://web.math.ucsb.edu/~jhateley/paper/lorenz.pdf`. Accessed: 2021-04-08.

E. Lorenz. The butterfly effect. *World Scientific Series on Nonlinear Science Series A*, 39:91–94, 2000.

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

A. J. Lotka. Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, 14(3):271–274, 2002.

M. N. Marshall. Sampling for qualitative research. *Family practice*, 13(6): 522–526, 1996.

Z. Musielak and B. Quarles. The three-body problem. *Reports on Progress in Physics*, 77(6):065901, 2014.

ProbNum Documentation. `https://probnum.readthedocs.io/en/latest/`. Accessed: 2021-03-12.

S. Särkkä. *Bayesian filtering and smoothing.* Number 3. Cambridge University Press, 2013.

S. Särkkä and A. Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

M. Schober, D. Duvenaud, and P. Hennig. Probabilistic ode solvers with runge-kutta means. *arXiv preprint arXiv:1406.2582*, 2014.

SciPy Integrate Documentation. `https://docs.scipy.org/doc/scipy/reference/integrate.html`. Accessed: 2021-03-12.

L. F. Shampine. Error estimation and control for odes. *Journal of Scientific Computing*, 25(1):3–16, 2005.

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

G. Teschl. *Ordinary differential equations and dynamical systems*, volume 140. American Mathematical Soc., 2012.

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

P. E. Zadunaisky. On the estimation of errors propagated in the numerical integration of ordinary differential equations. *Numerische Mathematik*, 27 (1):21–39, 1976.

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