Initial Value Problem Uncertainty Propagation

Mathias Van Gompel — *Leuven Gravity Institute, Department of Electrical Engineering (PSI), KU Leuven, Belgium*
Tom Colemont — *Leuven Gravity Institute, Department of Electrical Engineering (STADIUS), KU Leuven, Belgium*
Tjonnie G.-F. Li — *Leuven Gravity Institute, Department of Physics & Astronomy, KU Leuven, Belgium*
Johan A.K. Suykens — *Department of Electrical Engineering (STADIUS), KU Leuven, Belgium*
Frederik De Ceuster — *Leuven Gravity Institute, Department of Physics & Astronomy, KU Leuven, Belgium*

Abstract

Probabilistic ODE solvers quantify numerical uncertainty by returning a posterior distribution over the solution, rather than only a point estimate. However, these methods typically assume that the initial value problem (IVP) itself is deterministic and fully known. When initial conditions or parameters are uncertain, increasing the numerical accuracy during the solve should not reduce uncertainty about the IVP itself. Standard probabilistic ODE solvers do not distinguish clearly between numerical uncertainty and IVP uncertainty, and may therefore contract the latter inappropriately. Recent work addressed this issue by combining filtering-based probabilistic ODE solvers with numerical quadrature to marginalise correctly over the IVP uncertainty. We extend this work by formulating this outer marginalisation problem in a Bayesian quadrature framework, allowing uncertainty from the quadrature approximation itself to be quantified alongside propagated IVP uncertainty and conditional solver uncertainty. In addition, for Gaussian uncertainty in the initial conditions or parameters, we derive closed-form recursions that can be incorporated directly into filtering and smoothing methods, thereby avoiding numerical quadrature altogether.

Proceedings of the 2nd International Conference on Probabilistic Numerics (ProbNum) 2026, Lappeenranta, Finland. PMLR Volume 341.
 Copyright 2026 with the author(s)

1 Introduction

Many models in science and engineering describe how a system changes over time. Examples include population growth, mechanical motion, planetary motion, chemical reactions and biological dynamics (Damour and Deruelle, 1985; Daun et al., 2008; Gillespie, 1977; Kermack and McKendrick, 1927). In such models, the current state of the system determines its future evolution through a differential equation.

In practice, this evolution is rarely known in closed form. Even when the model equations are specified exactly, the solution usually has to be approximated numerically. This introduces numerical discretisation uncertainty: the computed solution only approximates the true solution.

Consider the ordinary differential equation (ODE) initial value problem (IVP)

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t); \boldsymbol{\theta}), \qquad t \in [t_0, T], \tag{1}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0. \tag{2}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ denotes the state trajectory.

Given a vector field $\boldsymbol{f}$, an initial time t_0, parameters $\boldsymbol{\theta}$, and an initial condition $\boldsymbol{x}_0$, we seek a trajectory $\boldsymbol{x}(t)$ that satisfies the differential equation for all $t \in [t_0, T]$, assuming its existence and uniqueness.

Probabilistic ODE solvers enrich this approximation with uncertainty quantification (Schober et al., 2014, 2019; Tronarp et al., 2019). Rather than providing just a single numerical trajectory, they return a probability distribution over trajectories. By choosing a Gauss–Markov process as a prior over the unknown solution, one obtains efficient filtering-based formulations in which inference is performed by conditioning on the information that the ODE residual vanishes along the solution trajectory (Schober et al., 2019; Tronarp et al., 2019).

At this point, it is useful to distinguish two quite different roles that uncertainty can play. On the one hand, there is *numerical uncertainty*: even when the inputs $(\boldsymbol{x}_0, \boldsymbol{\theta})$ for the ODE solver are fixed and known exactly, discretisation and approximation errors remain because the exact flow map is unavailable. On the other hand, there is *IVP uncertainty*: in many applications, neither the initial condition $\boldsymbol{x}_0$ nor the model parameters $\boldsymbol{\theta}$ are known exactly. If these inputs are uncertain, the uncertainty in the final state must reflect both the numerical uncertainty due to the approximate solver and the IVP uncertainty induced by the uncertain IVP.

This distinction is especially important in probabilistic numerics (PN), which aims to represent uncertainty about numerical computations. (Hennig et al., 2022; Oates and Sullivan, 2019). Standard probabilistic ODE solvers quantify uncertainty conditional on a deterministic, fully specified IVP. If the inputs, i.e. the initial condition or parameters, are uncertain, one faces an additional outer problem: propagating the input distribution through the conditional numerical solver.

It is therefore helpful to separate three conceptually distinct uncertainty sources:

1. **conditional solver uncertainty**, which represents numerical error in the inner ODE solve;
2. **propagated IVP uncertainty**, which reflects uncertainty induced by the uncertain initial conditions or parameters; and

3. **quadrature uncertainty**, which represents uncertainty due to the numerical approximation of the marginalisation over uncertain IVP inputs.

The central principle of this paper can hence be stated as follows:

Uncertainty induced by computation should decrease with numerical refinement, while uncertainty inherited from the IVP should be propagated separately.

Improving the inner numerical solve should reduce conditional solver uncertainty, but not the uncertainty of the IVP itself. This is not a limitation of ODE filtering itself, since an ODE filter is designed to solve a single, fixed IVP. The problem arises only if its conditional covariance is mistakenly interpreted as already containing uncertainty about unknown initial conditions or parameters. Those inputs instead require the explicit outer marginalisation developed in (Yao et al., 2025).

This motivates a distinction between two tasks:

1. **state estimation:** infer hidden states from genuinely informative noisy observations (related to numerical uncertainty);
2. **uncertainty propagation:** push uncertainty in initial conditions or parameters through the dynamics (related to IVP uncertainty).

Yao et al. (2025) showed how uncertain IVP inputs can be handled recursively by combining filtering-based probabilistic ODE solvers with deterministic outer quadrature. For each fixed input, the conditional solver is run and its output is averaged over the input distribution. We make two additions, presented in that order. First, we derive exact closed-form moment recursions for jointly linear–Gaussian models and use their local linearisation as a first-order method for nonlinear IVPs. Second, we formulate the outer marginalisation using Bayesian quadrature (BQ), which yields an explicit outer uncertainty term $\mathbf{\Sigma}_k^{\text{out}}$. The former avoids quadrature and is inexpensive when linearisation is adequate; the latter requires multiple conditional solves but better resolves nonlinear input-to-solution maps.

The main idea can be summarised as follows. For every fixed choice of uncertain inputs $\boldsymbol{u} = (\boldsymbol{x}_0, \boldsymbol{\theta})$, we run an ordinary conditional probabilistic ODE solver. This produces a conditional posterior distribution over the solver state. The actual propagation problem then consists of averaging these conditional posteriors over the prior distribution of $\boldsymbol{u}$.

This point is conceptually important: we do *not* alter the Bayesian inference performed by the solver itself. Rather, we separate two levels: (1) On the inner level, the ODE filter solves a *conditional* inference problem: for fixed inputs $\boldsymbol{u}$, it computes the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{u})$. (2) On the outer level, if the inputs are uncertain with distribution $p(\boldsymbol{u})$, one must additionally propagate this uncertainty through the solver, yielding the marginal distribution

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{u})\, p(\boldsymbol{u})\, d\boldsymbol{u}. \tag{3}$$

This hierarchical structure is summarised in Fig. 1.

Section 2 reviews filtering-based probabilistic ODE solvers, distinguishes conditional inference from IVP uncertainty propagation and illustrates this distinction through a linear–Gaussian toy model. The common moment decomposition is introduced in Section 3. We then present the first-order method in Section 4, followed by BQ in Section 5.

2 Background

2.1 Filtering-based probabilistic ODE solvers

Let $t_0 < t_1 < \cdots < t_K$ be a discretisation grid. A filtering-based probabilistic ODE solver introduces a solver state $\boldsymbol{x}_k$ and a solver-internal residual variable $\boldsymbol{y}_k$ such that

$$\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1} \sim p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}), \tag{4}$$
$$\boldsymbol{y}_k \mid \boldsymbol{x}_k \sim p(\boldsymbol{y}_k \mid \boldsymbol{x}_k). \tag{5}$$

Equation (4) describes the Gauss–Markov prior with Markovian kernel, while (5) introduces the solver-internal residual or consistency condition.

For ODE filters, the state $\boldsymbol{x}_k$ typically contains the solution and several of its derivatives at time t_k, while $\boldsymbol{y}_k$ is often a residual that compares a predicted derivative against a vector-field evaluation and is targeted at zero.

Because $\boldsymbol{y}_k$ is solver-internal rather than external data, conditioning on it should reduce only numerical uncertainty for fixed IVP inputs. It must not reduce uncertainty arising from genuinely unknown IVP inputs such as $\boldsymbol{x}_0$ or $\boldsymbol{\theta}$. For a more detailed introduction of probabilistic ODE solvers, we refer to (Schober et al., 2019; Tronarp et al., 2019)

2.2 Conditional inference versus uncertainty propagation

Let

$$\boldsymbol{u} := \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{\theta} \end{bmatrix}$$

collect the uncertain IVP inputs, with prior distribution $p(\boldsymbol{u})$. For every fixed value of $\boldsymbol{u}$, the probabilistic solver returns a conditional posterior

$$p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}, \boldsymbol{u}). \tag{6}$$

In most practical filters, this conditional posterior is approximated as a Gaussian distribution,

$$\begin{aligned} \boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}, \boldsymbol{u} \; &\sim p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}, \boldsymbol{u}) \\ &\approx \mathcal{N}\big(\boldsymbol{m}_k(\boldsymbol{u}), \bar{\boldsymbol{P}}_k(\boldsymbol{u})\big), \end{aligned} \tag{7}$$

so that the conditional solver output is described by a mean map $\boldsymbol{m}_k(\boldsymbol{u})$ and a covariance map $\bar{\boldsymbol{P}}_k(\boldsymbol{u})$.

The physically meaningful propagated distribution is then obtained by averaging these conditional posteriors over the uncertain input distribution:

$$p_{\text{prop}}(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}) := \int p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}, \boldsymbol{u})\, p(\boldsymbol{u})\, d\boldsymbol{u}. \tag{8}$$

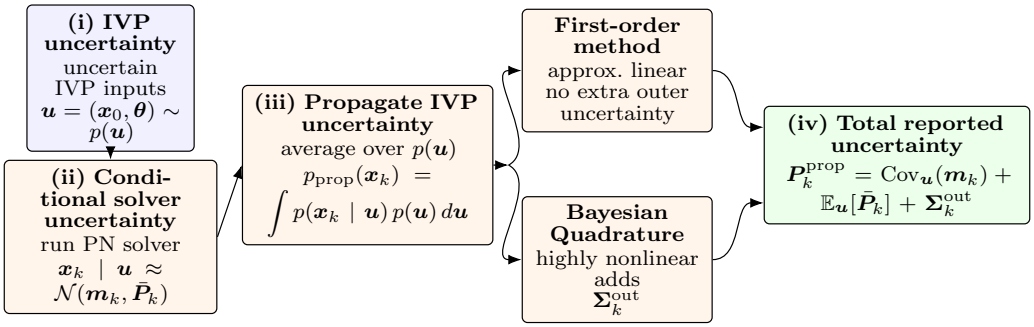


Figure 1: Hierarchical view of uncertainty propagation for IVPs with uncertain inputs. For fixed inputs $u = (x_0, \boldsymbol{\theta})$, the probabilistic ODE solver returns a conditional distribution with conditional solver uncertainty. IVP uncertainty is then propagated by averaging these conditional outputs over the input distribution $p(u)$. In approximately linear regimes, this outer propagation can be handled by Jacobian recursion without introducing additional outer numerical uncertainty. In strongly nonlinear regimes, the outer marginalisation must generally be approximated numerically, for example by quadrature, Monte Carlo, or Bayesian quadrature. BQ additionally reports an outer computational uncertainty term.

This equation is one of the key objects in the paper. It simply states that the uncertainty at time t_k should account for *all plausible inputs*, weighted by the probability assigned to them initially. Put differently, the final propagated distribution is an outer marginalisation of conditional solver outputs.

The integral in Eq. (8) is generally intractable. Even if each conditional posterior is Gaussian, the marginal need not admit a simple analytic form, because the maps $\boldsymbol{u} \mapsto \boldsymbol{m}_k(\boldsymbol{u})$ and $\boldsymbol{u} \mapsto \bar{\boldsymbol{P}}_k(\boldsymbol{u})$ may depend nonlinearly on the uncertain inputs. This is exactly the point at which approximation methods such as Monte Carlo, deterministic quadrature, or Bayesian quadrature become necessary (O'Hagan, 1991; Briol et al., 2019).

2.3 Toy model

In this section, we use a minimal linear–Gaussian toy model to illustrate why ordinary filtering uncertainty does not, in general, provide the correct uncertainty propagation for an IVP with uncertain initial conditions. The example isolates the distinction between solver-induced uncertainty, which should contract under numerical conditioning, and uncertainty inherited from the initial condition, which should remain present in the propagated solution.

Consider a single time step with the scalar linear–Gaussian model

$$\begin{aligned} &x_0 \sim \mathcal{N}(0, \sigma_0^2), \qquad x_1 = x_0 + w, \quad w \sim \mathcal{N}(0, q), \\ &y_1 = x_1 + v, \qquad v \sim \mathcal{N}(0, r), \end{aligned} \tag{9}$$

fixed solver-internal residual value $y_1 \in \mathbb{R}$.

If we interpret this as an ordinary filtering problem, then the Kalman update yields posterior variance (Särkkä and Svensson, 2023)

$$P_{\text{filter}} = \frac{r(\sigma_0^2 + q)}{\sigma_0^2 + q + r}. \tag{10}$$

This is the familiar variance contraction from Bayesian conditioning:

$$\sigma_0^2 \to \infty \quad \Rightarrow \quad P_{\text{filter}} \to r. \tag{11}$$

This limit is inappropriate for IVP uncertainty propagation. Here y_1 is not informative external data about the unknown initial condition; it is a solver-internal consistency condition enforcing that the numerical step approximately satisfies the ODE. Conditioning on y_1 should therefore reduce only numerical uncertainty and must not eliminate genuine uncertainty about the IVP input x_0. Accordingly, the correct propagated behaviour is

$$\sigma_0^2 \to \infty \quad \Rightarrow \quad P_{\text{prop}} \to \infty. \tag{12}$$

The toy model therefore exposes the core tension clearly: ordinary filtering treats y_1 as informative about the latent state itself, whereas IVP uncertainty propagation should treat it only as a numerical consistency condition. We now formalise this propagation principle recursively.

Propagation prediction distribution Assume that at time $k-1$ we have already propagated both numerical uncertainty and IVP uncertainty via $p_{\text{prop}}(\boldsymbol{x}_{k-1} \mid \boldsymbol{y}_{1:k-1})$. The prediction step is the standard Chapman–Kolmogorov update (Särkkä and Svensson, 2023; Yao et al., 2025).

$$p_{\text{prop}}(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k-1}) = \tag{13}$$

$$\int p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})\, p_{\text{prop}}(\boldsymbol{x}_{k-1} \mid \boldsymbol{y}_{1:k-1})\, d\boldsymbol{x}_{k-1}, \tag{14}$$

Propagation filtering distribution The update step is where the distinction from ordinary filtering enters. For a *fixed* previous state $\boldsymbol{x}_{k-1}$, Bayes' rule yields the local conditional update

$$p(\boldsymbol{x}_k \mid \boldsymbol{y}_k, \boldsymbol{x}_{k-1}) = \frac{p(\boldsymbol{y}_k \mid \boldsymbol{x}_k)\, p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})}{p(\boldsymbol{y}_k \mid \boldsymbol{x}_{k-1})}, \tag{15}$$

where

$$p(\boldsymbol{y}_k \mid \boldsymbol{x}_{k-1}) = \int p(\boldsymbol{y}_k \mid \boldsymbol{x}_k)\, p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})\, d\boldsymbol{x}_k. \tag{16}$$

So far this is still ordinary Bayesian conditioning, but crucially only at the local level for fixed $\boldsymbol{x}_{k-1}$. The propagation update is then obtained by averaging these locally updated conditionals over the previously propagated uncertainty:

$$p_{\mathrm{prop}}(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}) =$$

$$\int p(\boldsymbol{x}_k \mid \boldsymbol{y}_k, \boldsymbol{x}_{k-1})\; p_{\mathrm{prop}}(\boldsymbol{x}_{k-1} \mid \boldsymbol{y}_{1:k-1})\; \mathrm{d}\boldsymbol{x}_{k-1} = \tag{17}$$

$$\int \frac{p(\boldsymbol{y}_k \mid \boldsymbol{x}_k)\, p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})}{p(\boldsymbol{y}_k \mid \boldsymbol{x}_{k-1})}\; p_{\mathrm{prop}}(\boldsymbol{x}_{k-1} \mid \boldsymbol{y}_{1:k-1})\; \mathrm{d}\boldsymbol{x}_{k-1}. \tag{18}$$

Remark 1. *For comparison, the standard Bayesian filter associated with the state-space model normalises globally. In contrast, the propagation update (18) normalises* locally *with $p(\boldsymbol{y}_k \mid \boldsymbol{x}_{k-1})$ before averaging over $\boldsymbol{x}_{k-1}$.*

3 Propagated moments

The conditional solver is available for every fixed input $\boldsymbol{u}$, but the outer average in Eq. (8) is generally intractable. Both methods developed below in Section 4 and Section 5 target the same first two moments, so we state those quantities before introducing either approximation.

For this reason, it is useful to separate the computational task into two layers:

1. for fixed $\boldsymbol{u}$, apply the conditional probabilistic ODE solver and obtain $(\boldsymbol{m}_k(\boldsymbol{u}), \bar{\boldsymbol{P}}_k(\boldsymbol{u}))$;
2. evaluate or approximate the outer integrals over $p(\boldsymbol{u})$.

This modular decomposition leaves the inner solver unchanged. The outer layer may be treated analytically, by first-order linearisation, by Monte Carlo, by deterministic quadrature as in (Yao et al., 2025), or by BQ.

3.1 Gaussian-weighted integrals

Assume a Gaussian distribution

$$\boldsymbol{u} := \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{\theta} \end{bmatrix} \in \mathbb{R}^{d_u}, \qquad \boldsymbol{u} \sim \mathcal{N}(\boldsymbol{m}_u, \boldsymbol{P}_u). \tag{19}$$

For each deterministic input $\boldsymbol{u}$, apply the conditional solver to obtain

$$\boldsymbol{x}_k \mid \boldsymbol{u} \approx \mathcal{N}\big(\boldsymbol{m}_k(\boldsymbol{u}), \bar{\boldsymbol{P}}_k(\boldsymbol{u})\big), \tag{20}$$

where $\boldsymbol{m}_k(\boldsymbol{u})$ is the conditional solver mean and $\bar{\boldsymbol{P}}_k(\boldsymbol{u})$ is the corresponding conditional covariance.

The first two moments of the propagation distribution then follow from the law of total expectation and the law of total variance:

$$\boldsymbol{m}_k^{\mathrm{prop}} = \mathbb{E}_{\boldsymbol{u}}\big[\boldsymbol{m}_k(\boldsymbol{u})\big], \tag{21}$$

$$\boldsymbol{P}_k^{\mathrm{prop}} = \mathbb{E}_{\boldsymbol{u}}\big[\bar{\boldsymbol{P}}_k(\boldsymbol{u})\big] + \mathrm{Cov}_{\boldsymbol{u}}\big(\boldsymbol{m}_k(\boldsymbol{u})\big). \tag{22}$$

These two formulas separate the two contributions to the propagated covariance. The first term is the average conditional solver covariance and reflects conditional numerical uncertainty. The second term measures how strongly the conditional mean varies across different plausible inputs and therefore captures the spread induced by uncertain initial conditions and parameters.

If $\boldsymbol{m}_k(\boldsymbol{u})$ hardly changes with $\boldsymbol{u}$, then propagated IVP uncertainty is small. If it varies strongly with $\boldsymbol{u}$, then propagated IVP uncertainty becomes large even if each conditional solver covariance is itself small.

Remark 2 (Consistency under solver refinement)**.** *Suppose the numerical conditional mean and covariance converge pointwise to those induced by the exact flow as the inner solver is refined and suppose they are dominated by integrable bounds under $p(\boldsymbol{u})$. Dominated convergence then implies convergence of both moments in Eqs. (21) and (22) to the moments of the exact pushforward. Refinement should therefore improve the approximation of the propagated distribution without collapsing its intrinsic IVP uncertainty.*

4 First-order propagation

We first treat the setting in which outer marginalisation is exact. The same algebra then yields a local first-order approximation for nonlinear IVPs, making explicit both its computational appeal and its limitations.

4.1 Exact jointly linear–Gaussian recursion

Assume a linear–Gaussian model

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{m}_{\boldsymbol{\theta}}, \boldsymbol{P}_{\boldsymbol{\theta}}), \tag{23}$$

$$\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0), \tag{24}$$

$$\boldsymbol{x}_k = \boldsymbol{A}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{B}_{k-1}\boldsymbol{\theta} + \boldsymbol{q}_{k-1}, \tag{25}$$

$$\boldsymbol{y}_k = \boldsymbol{C}_k\boldsymbol{x}_k + \boldsymbol{D}_k\boldsymbol{\theta} + \boldsymbol{r}_k, \tag{26}$$

where

$$\boldsymbol{q}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}_{k-1}), \qquad \boldsymbol{r}_k \sim \mathcal{N}(0, \boldsymbol{R}_k). \tag{27}$$

Write the joint prior of $(\boldsymbol{x}_0, \boldsymbol{\theta})$ as

$$\begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{\theta} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}_0 \\ \boldsymbol{m}_{\boldsymbol{\theta}} \end{bmatrix}, \boldsymbol{\Sigma}_0\right), \qquad \boldsymbol{\Sigma}_0 := \begin{bmatrix} \boldsymbol{P}_0 & 0 \\ 0 & \boldsymbol{P}_{\boldsymbol{\theta}} \end{bmatrix}. \tag{28}$$

So we assume that $\boldsymbol{x}_0$ and $\boldsymbol{\theta}$ are independent.

For fixed $(\boldsymbol{x}_0, \boldsymbol{\theta})$, the standard Kalman recursions are

$$\boldsymbol{m}_0(\boldsymbol{x}_0, \boldsymbol{\theta}) = \boldsymbol{x}_0, \tag{29}$$

$$\bar{\boldsymbol{P}}_0 = 0, \tag{30}$$

$$\boldsymbol{m}_k^-(\boldsymbol{x}_0, \boldsymbol{\theta}) = \boldsymbol{A}_{k-1}\boldsymbol{m}_{k-1}(\boldsymbol{x}_0, \boldsymbol{\theta}) + \boldsymbol{B}_{k-1}\boldsymbol{\theta}, \tag{31}$$

$$\bar{\boldsymbol{P}}_k^- = \boldsymbol{A}_{k-1}\bar{\boldsymbol{P}}_{k-1}\boldsymbol{A}_{k-1}^\top + \boldsymbol{Q}_{k-1}, \tag{32}$$

$$\boldsymbol{v}_k(\boldsymbol{x}_0, \boldsymbol{\theta}) = \boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{m}_k^-(\boldsymbol{x}_0, \boldsymbol{\theta}) - \boldsymbol{D}_k\boldsymbol{\theta}, \tag{33}$$

$$\boldsymbol{S}_k = \boldsymbol{C}_k\bar{\boldsymbol{P}}_k^-\boldsymbol{C}_k^\top + \boldsymbol{R}_k, \tag{34}$$

$$\boldsymbol{K}_k = \bar{\boldsymbol{P}}_k^-\boldsymbol{C}_k^\top\boldsymbol{S}_k^{-1}, \tag{35}$$

$$\boldsymbol{m}_k(\boldsymbol{x}_0, \boldsymbol{\theta}) = \boldsymbol{m}_k^-(\boldsymbol{x}_0, \boldsymbol{\theta}) + \boldsymbol{K}_k\boldsymbol{v}_k(\boldsymbol{x}_0, \boldsymbol{\theta}), \tag{36}$$

$$\bar{\boldsymbol{P}}_k = \bar{\boldsymbol{P}}_k^- - \boldsymbol{K}_k\boldsymbol{S}_k\boldsymbol{K}_k^\top. \tag{37}$$

such that

$$\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}, \boldsymbol{x}_0, \boldsymbol{\theta} \sim \mathcal{N}\big(\boldsymbol{m}_k(\boldsymbol{x}_0, \boldsymbol{\theta}), \bar{\boldsymbol{P}}_k\big)\,, \tag{38}$$

where $\bar{\boldsymbol{P}}_k$ and the Kalman gain $\boldsymbol{K}_k$ do not depend on the values of $(\boldsymbol{x}_0, \boldsymbol{\theta})$. Consequently, the conditional mean is affine in the uncertain inputs and the outer marginalisation remains Gaussian.

4.2 Closed-form propagated moments

Let $\boldsymbol{m}_k^\star$ be the conditional mean obtained at the prior mean input. Then

$$\boldsymbol{m}_k(\boldsymbol{x}_0, \boldsymbol{\theta}) = \boldsymbol{m}_k^\star + \boldsymbol{J}_k^x(\boldsymbol{x}_0 - \boldsymbol{m}_0) + \boldsymbol{J}_k^{\boldsymbol{\theta}}(\boldsymbol{\theta} - \boldsymbol{m}_{\boldsymbol{\theta}}). \tag{39}$$

These Jacobians satisfy the recursions

$$\begin{aligned}
\boldsymbol{J}_0^x &= \boldsymbol{I}_d, && (40)\\
\boldsymbol{J}_k^{x,-} &= \boldsymbol{A}_{k-1}\boldsymbol{J}_{k-1}^x, && (41)\\
\boldsymbol{J}_k^x &= (\boldsymbol{I}_d - \boldsymbol{K}_k\boldsymbol{C}_k)\,\boldsymbol{J}_k^{x,-}, && (42)\\
\boldsymbol{J}_0^{\boldsymbol{\theta}} &= 0, && (43)\\
\boldsymbol{J}_k^{\boldsymbol{\theta},-} &= \boldsymbol{A}_{k-1}\boldsymbol{J}_{k-1}^{\boldsymbol{\theta}} + \boldsymbol{B}_{k-1}, && (44)\\
\boldsymbol{J}_k^{\boldsymbol{\theta}} &= (\boldsymbol{I}_d - \boldsymbol{K}_k\boldsymbol{C}_k)\,\boldsymbol{J}_k^{\boldsymbol{\theta},-} - \boldsymbol{K}_k\boldsymbol{D}_k. && (45)
\end{aligned}$$

The two Jacobian blocks have distinct meanings. The matrix $\boldsymbol{J}_k^x$ transports uncertainty already present in the initial state, whereas $\boldsymbol{J}_k^{\boldsymbol{\theta}}$ measures how parameter perturbations affect the state throughout the dynamics. Initial-condition and parameter uncertainty therefore share an outer marginalisation but are not assumed to propagate identically. Since the model is linear, this first-order representation is exact rather than approximate. More precisely, it is exact for the linear–Gaussian state-space model to which the Kalman recursions are applied. When this model is obtained by locally linearising a nonlinear ODE filter, the propagated moments inherit the approximation error of that linearisation. Moreover, the Gaussian closure represents uncertainty through a covariance ellipsoid that is symmetric about the mean. A nonlinear input-to-solution map may instead produce asymmetric or otherwise non-Gaussian pushforward distributions, so a first-order Gaussian approximation can underestimate or misorient the propagated uncertainty.

As a consequence,

$$\boldsymbol{m}_k^{\text{prop}} = \boldsymbol{m}_k^\star, \tag{46}$$

and if we write $\boldsymbol{J}_k := [\,\boldsymbol{J}_k^x \;\; \boldsymbol{J}_k^{\boldsymbol{\theta}}\,]$, then

$$\boldsymbol{P}_k^{\text{prop}} = \bar{\boldsymbol{P}}_k + \boldsymbol{J}_k\,\boldsymbol{\Sigma}_0\,\boldsymbol{J}_k^\top. \tag{47}$$

This is the exact outer marginalisation for the approximating linear–Gaussian solver model; no quadrature uncertainty is introduced.

4.3 Extension to nonlinear IVPs

For a nonlinear IVP, the preceding closed-form result is not exact. We instead linearise the conditional mean map around the input mean $\boldsymbol{m}_u$:

$$\begin{aligned}
\boldsymbol{m}_k(\boldsymbol{u}) &\approx \boldsymbol{m}_k(\boldsymbol{m}_u) + \boldsymbol{J}_k(\boldsymbol{u} - \boldsymbol{m}_u),\\
\boldsymbol{J}_k &:= \left.\frac{\partial \boldsymbol{m}_k(\boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}=\boldsymbol{m}_u}.
\end{aligned} \tag{48}$$

The Jacobian is propagated through the locally linearised prediction and update maps of the conditional ODE filter, in direct analogy with an extended Kalman filter (Tronarp et al., 2019). The resulting moment approximation is

$$\boldsymbol{m}_k^{\text{FO}} = \boldsymbol{m}_k(\boldsymbol{m}_u), \qquad \boldsymbol{P}_k^{\text{FO}} = \bar{\boldsymbol{P}}_k(\boldsymbol{m}_u) + \boldsymbol{J}_k\boldsymbol{P}_u\boldsymbol{J}_k^\top. \tag{49}$$

Thus the nonlinear method requires one conditional solve at the mean input and a sensitivity recursion. It is best viewed as an extended-Kalman-style approximation: it is reliable for smooth vector fields, small steps and concentrated input distributions whose input-to-solution map is nearly affine. Curvature and state interactions can make it underestimate or misorient the propagated covariance, motivating BQ.

5 Bayesian quadrature

When the first-order approximation is inadequate, the outer integrals can be evaluated at several input locations. BQ treats the conditional mean map as an unknown function under a Gaussian-process prior and turns its evaluations into a posterior distribution over the required Gaussian-weighted moments.

5.1 Bayesian quadrature moment transform

The goal is to approximate the outer Gaussian-weighted integrals defining the propagated moments while assigning an additional numerical-uncertainty term under the BQ model. Monte Carlo methods target the same moments by drawing $\boldsymbol{u}^{(i)} \sim p(\boldsymbol{u})$, running a solver for each sample and replacing the expectations in Eqs. (21) and (22) by empirical averages. The BQ construction used here instead combines deterministic sigma points with Gaussian-process quadrature weights and returns an additional uncertainty term for the outer numerical integration (Prüher and Šimandl, 2016).

A detailed derivation appears in (Prüher and Šimandl, 2016). Write the uncertain input as

$$\begin{aligned}
&\boldsymbol{u} = \boldsymbol{m}_u + \boldsymbol{L}_u\,\boldsymbol{\xi}, \qquad \boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I}_{d_u}),\\
&\boldsymbol{L}_u\boldsymbol{L}_u^\top = \boldsymbol{P}_u.
\end{aligned} \tag{50}$$

Define the propagated conditional-mean map

$$g_k(\boldsymbol{\xi}) := \boldsymbol{m}_k(\boldsymbol{m}_u + \boldsymbol{L}_u\boldsymbol{\xi}). \tag{51}$$

The required quantities are $\boldsymbol{m}_k^{\text{prop}} = \mathbb{E}_{\boldsymbol{\xi}}[g_k(\boldsymbol{\xi})]$ and $\boldsymbol{\Pi}_k = \mathrm{Cov}_{\boldsymbol{\xi}}(g_k(\boldsymbol{\xi}))$, the propagated IVP covariance.

Choose sigma points $\{\boldsymbol{\xi}^{(i)}\}_{i=1}^N$, set $\boldsymbol{u}^{(i)} = \boldsymbol{m}_u + \boldsymbol{L}_u\boldsymbol{\xi}^{(i)}$ and run the conditional solver at each input. We use tensor-product Gauss–Hermite points where, in classical quadrature, we would use normalised deterministic

weights $\bar{w}_i$ (Stroud et al., 1966). Collect the conditional means row-wise in

$$\boldsymbol{G}_k \in \mathbb{R}^{N \times d_x}, \qquad (\boldsymbol{G}_k)_{i,:} = g_k(\boldsymbol{\xi}^{(i)})^\top. \tag{52}$$

Each output component of g_k is modelled by an independent GP with shared kernel k (Prüher and Šimandl, 2016; Särkkä et al., 2015). Its Gram matrix is

$$\boldsymbol{K}_{ij} := k(\boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}^{(j)}). \tag{53}$$

Define the kernel mean and second-moment matrices by

$$\boldsymbol{\ell}_i := \mathbb{E}_{\boldsymbol{\xi}}\big[k(\boldsymbol{\xi}, \boldsymbol{\xi}^{(i)})\big]. \tag{54}$$

$$\boldsymbol{L}_{ij} := \mathbb{E}_{\boldsymbol{\xi}}\big[k(\boldsymbol{\xi}, \boldsymbol{\xi}^{(i)})k(\boldsymbol{\xi}, \boldsymbol{\xi}^{(j)})\big]. \tag{55}$$

The BQ weights and moment approximations are

$$\boldsymbol{w} := \boldsymbol{K}^{-1}\boldsymbol{\ell}, \tag{56}$$

$$\boldsymbol{W} := \boldsymbol{K}^{-1}\boldsymbol{L}\boldsymbol{K}^{-1}, \tag{57}$$

$$\boldsymbol{\mu}_k^{\mathrm{BQ}} := \boldsymbol{G}_k^\top \boldsymbol{w}, \tag{58}$$

$$\widehat{\boldsymbol{\Pi}}_k^{\mathrm{BQ}} := \boldsymbol{G}_k^\top \boldsymbol{W} \boldsymbol{G}_k - \boldsymbol{\mu}_k^{\mathrm{BQ}}(\boldsymbol{\mu}_k^{\mathrm{BQ}})^\top. \tag{59}$$

The subtraction centres the approximated second moment and therefore separates propagated IVP covariance from the BQ integration term. A more detailed derivation can be found in Section B.

With $\bar{k} := \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi})]$, the BQ model assigns the integrated posterior variance (Prüher and Šimandl, 2016)

$$\sigma^2_{\mathrm{int}} := \bar{k} - \mathrm{tr}(\boldsymbol{K}^{-1}\boldsymbol{L}). \tag{60}$$

The distinct outer quadrature-uncertainty term is

$$\boldsymbol{\Sigma}_k^{\mathrm{out}} := \sigma^2_{\mathrm{int}} \boldsymbol{I}_{d_x}. \tag{61}$$

The average conditional solver covariance uses the BQ weights on the same nodes,

$$\widehat{\mathbb{E}_{\boldsymbol{u}}[\bar{\boldsymbol{P}}_k(\boldsymbol{u})]} := \sum_{i=1}^{N} w_i \, \bar{\boldsymbol{P}}_k(\boldsymbol{u}^{(i)}). \tag{62}$$

and $\boldsymbol{m}_k^{\mathrm{prop,BQ}} := \boldsymbol{\mu}_k^{\mathrm{BQ}}$. The covariance decomposes into three interpretable terms:

$$\boldsymbol{P}_k^{\mathrm{prop,BQ}} = \underbrace{\sum_{i=1}^{N} w_i \bar{\boldsymbol{P}}_k(\boldsymbol{u}^{(i)})}_{\text{conditional solver}} + \underbrace{\widehat{\boldsymbol{\Pi}}_k^{\mathrm{BQ}}}_{\text{propagated IVP}} + \underbrace{\boldsymbol{\Sigma}_k^{\mathrm{out}}}_{\text{outer BQ}}. \tag{63}$$

For the exponentiated-quadratic kernel, all kernel integrals above are available in closed form. A deterministic solver removes the first term; deterministic Gauss–Hermite quadrature uses $\bar{w}_i$ and has $\boldsymbol{\Sigma}_k^{\mathrm{out}} = 0$; exact linear marginalisation needs neither quadrature term. Tensor-product Gauss–Hermite uses $N = n^{d_u}$ nodes, so both quadrature methods are practical mainly for low-dimensional uncertain inputs.

The distinction between BQ and classical Gauss–Hermite quadrature is shown in Section D. In that comparison, both methods use the same sigma points and the same conditional solver evaluations, the difference comes from the outer moment weights. Classical Gauss–Hermite uses the deterministic weights $\bar{w}_i$, whereas BQ uses the kernel-dependent weights $\boldsymbol{w} = \boldsymbol{K}^{-1}\boldsymbol{\ell}$ for the mean and $\boldsymbol{W} = \boldsymbol{K}^{-1}\boldsymbol{L}\boldsymbol{K}^{-1}$ for the second moment. Consequently, BQ also returns the additional outer uncertainty term $\boldsymbol{\Sigma}_k^{\mathrm{out}}$, which is absent in deterministic quadrature.

6 Numerical experiments

This section benchmarks the *first-order* propagation method and BQ against Monte Carlo (MC) on three IVPs. We evaluate the accuracy–runtime tradeoff for the pushforward moments using a high-sample MC reference. The code for the numerical experiments is available online[1].

6.1 Protocol and implementation details

Choice of input distributions and BQ prior. The distributions over $\boldsymbol{x}_0$ and $\boldsymbol{\theta}$ are modelling assumptions external to the solver and should encode domain knowledge, measurement uncertainty, or an inferred posterior. We use Gaussian distributions for tractability. BQ uses an exponentiated-quadratic kernel because its Gaussian kernel integrals are available in closed form; other kernels are possible when the corresponding integrals can be evaluated analytically.

Benchmark problems and input distributions. We consider the three benchmark IVPs summarised in Table 1. They span linear and nonlinear dynamics and include uncertainty in both initial conditions and model parameters.

As shown in Table 1, the linear and Lotka–Volterra benchmarks contain uncertain initial conditions, whereas the logistic benchmark contains uncertainty in the carrying-capacity parameter b. These problems exercise qualitatively different propagation mechanisms. For the linear equation, initial-state variance expands with the flow. In the logistic equation, uncertainty in the carrying capacity changes the attracting level rather than merely perturbing the initial state. Lotka–Volterra combines a two-dimensional uncertain initial condition with strongly nonlinear coupled dynamics. The common outer formulation does not assume that these sources propagate identically; their effects enter through the corresponding input-to-solution sensitivities.

Conditional probabilistic ODE solver. For each deterministic input realisation $\boldsymbol{u}$, we run the same conditional solver: an EKF ODE filter with an integrated Wiener process prior of order $q = 4$ (IWP(4)) and adaptive step-size selection (Schober et al., 2019; Tronarp et al., 2019; Bosch et al., 2021). This yields the conditional mean $\boldsymbol{m}_k(\boldsymbol{u})$ and covariance $\bar{\boldsymbol{P}}_k(\boldsymbol{u})$ at discrete time points. The first-order method uses a single such

[1] ProbNum26_UncertaintyPropagation

Table 1: Benchmark IVPs and uncertain-input distributions used in the numerical experiments.

Problem	Vector field	Time interval	Initial value	Parameters
Linear	$f(t,x) = x$	$t \in [0,3]$	$x(0) \sim \mathcal{N}(1, 10^{-2})$	—
Logistic	$f(t,x) = a\,x\left(1 - \frac{x}{b}\right)$	$t \in [0,3]$	$x(0) = 0.05$	$a = 3, \quad b \sim \mathcal{N}(3, 10^{-2})$
Lotka–Volterra	$\boldsymbol{f}(t,\boldsymbol{x}) = \begin{bmatrix} ax_1 - bx_1x_2 \\ -cx_2 + dx_1x_2 \end{bmatrix}$	$t \in [0,2]$	$\boldsymbol{x}(0) \sim \mathcal{N}\left(\begin{bmatrix}5\\5\end{bmatrix}, 0.3\,\boldsymbol{I}_2\right)$	$(a,b,c,d) = (5, 0.5, 5, 0.5)$

solver run at the nominal input together with Jacobian propagation, whereas BQ evaluates the conditional solver at each sigma point.

Reference solution (high-sample MC). As reference, we use a Monte Carlo estimate with $N_{\text{ref}} = 10^6$: sample $\boldsymbol{u} \sim p(\boldsymbol{u})$, solve the deterministic IVP with `solve_ivp` using `DOP853` and compute the empirical mean and covariance $(\boldsymbol{m}_k^{\text{ref}}, \boldsymbol{P}_k^{\text{ref}})$ at each reported time point.

Methods, baselines and runtime measurement. We compare the first-order method, BQ and MC. The first-order moments use Eq. (49); BQ uses Section 5.1 and Eqs. (59) and (63) and the GH nodes in Table 2. For MC we use

$$N \in \{200, 500, 10^3, 2\cdot10^3, 5\cdot10^3, 10^4, 2\cdot10^4, 5\cdot10^4, 10^5\}.$$

For each N, MC uses N deterministic forward solves and empirical moment estimates. Each experiment is repeated 10 times, except for $N = 10^5$, which is run once. Errors are shown as boxplots. Runtime is measured as wall-clock time for the full method evaluation, including all deterministic forward solves for MC and all conditional solver calls for BQ.

Benchmark metric. At each time point t_k, a method returns a Gaussian approximation $x(t_k) \approx \mathcal{N}(\boldsymbol{m}_k, \boldsymbol{P}_k)$, which we compare to the Gaussian reference $\mathcal{N}(\boldsymbol{m}_k^{\text{ref}}, \boldsymbol{P}_k^{\text{ref}})$ using the closed-form 2-Wasserstein distance (Gelbrich, 1990):

$$W_2^2\Big(\mathcal{N}(\boldsymbol{m}_1, \boldsymbol{C}_1), \mathcal{N}(\boldsymbol{m}_2, \boldsymbol{C}_2)\Big) = \|\boldsymbol{m}_1 - \boldsymbol{m}_2\|^2 + \operatorname{tr}\Big(\boldsymbol{C}_1 + \boldsymbol{C}_2 - 2\big(\boldsymbol{C}_2^{1/2}\boldsymbol{C}_1\boldsymbol{C}_2^{1/2}\big)^{1/2}\Big). \tag{64}$$

Define $w_k := W_2\Big(\mathcal{N}(\boldsymbol{m}_k, \boldsymbol{P}_k), \mathcal{N}(\boldsymbol{m}_k^{\text{ref}}, \boldsymbol{P}_k^{\text{ref}})\Big)$, and aggregate over time by

$$\mathrm{W}_{2\text{rms}} := \left(\frac{1}{T}\sum_{k=1}^{T} w_k^2\right)^{1/2}. \tag{65}$$

Remark 3 (IVP-propagation accuracy)**.** *The runtime–accuracy figures evaluate recovery of the pushforward mean and covariance induced by uncertain inputs, namely* $\mathbb{E}_{\boldsymbol{u}}[\boldsymbol{m}_k(\boldsymbol{u})]$ *and* $\mathrm{Cov}_{\boldsymbol{u}}(\boldsymbol{m}_k(\boldsymbol{u}))$ *in Eqs. (21) and (22). Conditional solver covariance and* $\boldsymbol{\Sigma}_k^{\text{out}}$ *have different meanings and are not the target of this metric.*

6.2 Results

Fig. 2 shows the expected MC convergence trend: error decreases steadily with increasing N at increasing runtime. In this linear setting, the first-order method achieves $\mathrm{W2}_{\text{rms}} \approx 10^{-3}$ at sub-second runtime, outperforming MC even at $N = 10^5$ (which attains errors on the order of 10^{-3}–10^{-2} at two orders of magnitude larger runtime). BQ reaches a comparable error level but at higher cost because it performs multiple conditional solver evaluations. This is consistent with the linear theory: the first-order outer marginalisation is exact for the approximating linear–Gaussian model, while the conditional ODE solve remains numerical.

Logistic growth. In Fig. 3, the first-order method remains highly competitive: it matches or exceeds the accuracy of large-N MC at a fraction of the runtime, which is consistent with the mapping $\boldsymbol{u} \mapsto \boldsymbol{m}_k(\boldsymbol{u})$ being close to affine over the support of the input distribution. BQ further reduces error, down to the 10^{-4} range in these experiments, at a noticeable increase in runtime.

Lotka–Volterra. Fig. 4 highlights the regime where local linearisation becomes unreliable. The first-order method incurs a substantially larger error than MC, consistent with strong nonlinearity and state–state interactions amplifying higher-order effects of IVP uncertainty. BQ, which evaluates the nonlinear conditional mean at multiple input locations, recovers accuracy and improves substantially over MC at comparable or lower runtime

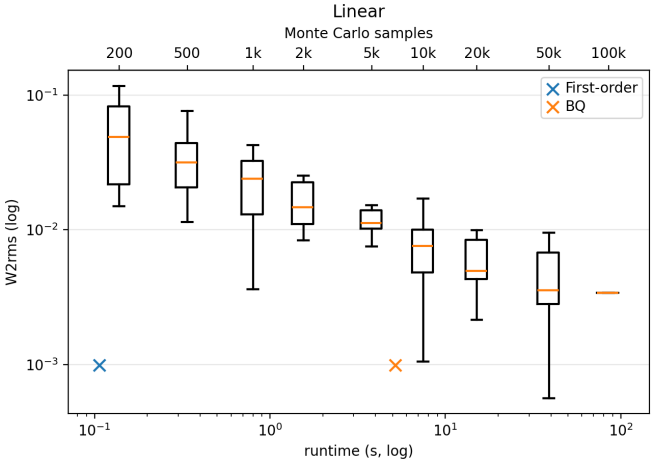


Figure 2: Linear IVP: distribution of MC errors (boxplots) versus runtime, measured by $\mathrm{W2}_{\text{rms}}$ relative to a $N_{\text{ref}} = 10^6$ reference.

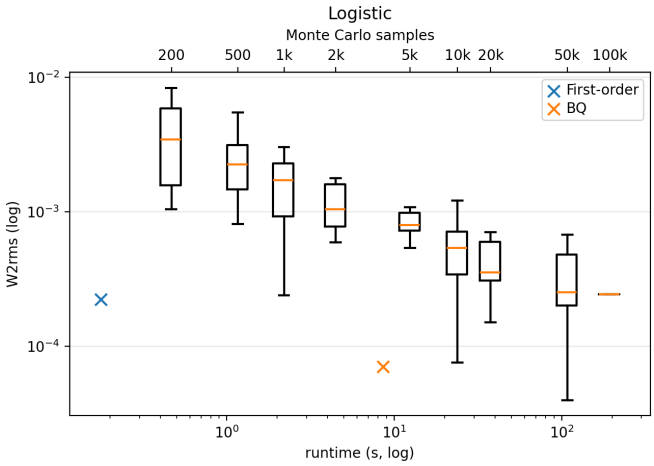


Figure 3: Logistic IVP: distribution of MC errors (boxplots) versus runtime, measured by $\mathrm{W2}_{\mathrm{rms}}$ relative to a $N_{\mathrm{ref}} = 10^6$ reference.

than very large MC sample sizes.

The trajectories of the solved benchmarks and the propagation of IVP uncertainty are given in Section E

7 Discussion

The first-order method is exact for jointly linear–Gaussian models. In that setting, the conditional mean is affine in the uncertain inputs and the conditional covariance does not depend on their realised values, so the outer marginalisation can be evaluated analytically. For nonlinear problems, it becomes an extended-Kalman-style approximation in which the input-to-solution map is linearised around the mean input. It is therefore most suitable when the input distribution is concentrated and the conditional mean map is approximately affine on its high-probability region.

The experiments illustrate this distinction. First-order propagation is accurate and computationally efficient for the linear and logistic problems, but performs worse for

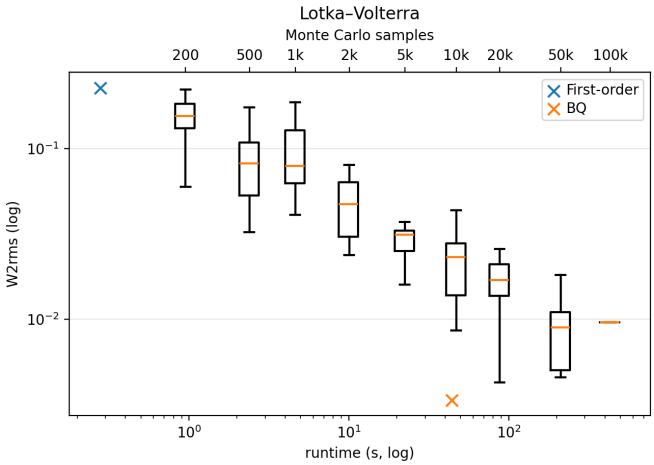


Figure 4: Lotka–Volterra IVP: distribution of MC errors (boxplots) versus runtime, measured by $\mathrm{W2}_{\mathrm{rms}}$ relative to a $N_{\mathrm{ref}} = 10^6$ reference.

Lotka–Volterra, where nonlinear state interactions introduce higher-order dependence on the uncertain initial condition. In such cases, the method may underestimate or misorient the propagated covariance. Moreover, the resulting Gaussian approximation only describes the first two moments and may not capture skewness, heavy tails, or multimodality in the true pushforward distribution.

BQ provides a nonlinear alternative by evaluating the conditional solver at multiple input locations. It also reports the additional outer uncertainty term $\mathbf{\Sigma}_k^{\mathrm{out}}$, which represents uncertainty in the numerical approximation of the outer integral under the GP model. A comparison with a classical quadrature method is given in Section D. Its main limitations are the dependence on the chosen kernel and node set, and the exponential scaling of tensor-product Gauss–Hermite quadrature with the uncertain-input dimension.

Probabilistic methods cannot remove uncertainty in the IVP or prevent loss of predictability in unstable or chaotic systems. Their contribution is to separate uncertainty caused by the dynamics from uncertainty introduced by the numerical solver and the outer integration. This prevents increased numerical accuracy from being incorrectly interpreted as increased certainty about the underlying IVP.

A natural extension is an adaptive method that uses first-order propagation by default and switches to BQ when sigma-point evaluations reveal substantial nonlinear behaviour. Sparse or dimension-adaptive quadrature could improve scalability. The framework may also be useful in Bayesian inverse problems, where the forward IVP solver is repeatedly evaluated inside a likelihood or posterior computation and numerical uncertainty must be distinguished from inferential uncertainty (Poot et al., 2026).

8 Conclusion

We showed that conditional solver uncertainty, propagated IVP uncertainty, and outer quadrature uncertainty should be treated separately. The first-order method provides exact propagated moments for jointly linear–Gaussian models and an efficient approximation when the nonlinear input-to-solution map is close to affine. BQ provides a more flexible nonlinear moment transform and additionally quantifies uncertainty in the outer numerical integration.

The experiments support a practical hierarchy: first-order propagation is preferable in linear or mildly nonlinear regimes, whereas BQ is more appropriate when nonlinear input effects dominate and additional conditional solver evaluations are affordable. More generally, probabilistic methods help by identifying and separating the sources of uncertainty, rather than by eliminating uncertainty intrinsic to the IVP.

Acknowledgments

Tom Colemont, Frederik De Ceuster and Tjonnie G.F. Li acknowledge support by the Research Foundation – Flanders (FWO; Grant No. I000725N, I002123N, and 1178926N). Johan Suykens acknowledges support of Research Council KU Leuven C14/24/103, iBOF/23/064; Flemish Government AI Research Program.

References

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

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

T. Damour and N. Deruelle. General relativistic celestial mechanics of binary systems. i. the post-newtonian motion. In *Annales de l'IHP Physique théorique*, volume 43, pages 107–132, 1985.

S. Daun, J. Rubin, Y. Vodovotz, and G. Clermont. Equation-based models of dynamic biological systems. *Journal of critical care*, 23(4):585–594, 2008.

M. Gelbrich. On a formula for the l2 wasserstein metric between measures on euclidean and hilbert spaces. *Mathematische Nachrichten*, 147(1):185–203, 1990.

D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

P. Hennig, M. A. Osborne, and H. P. Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022.

W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115 (772):700–721, 1927.

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

A. O'Hagan. Bayes–hermite quadrature. *Journal of statistical planning and inference*, 29(3):245–260, 1991.

A. Poot, I. Rocha, P. Kerfriden, and F. van der Meer. The bayesian finite element method in inverse problems: a critical comparison between probabilistic models for discretization error. *SIAM/ASA Journal on Uncertainty Quantification*, 14(2):287–312, 2026.

J. Prüher and M. Šimandl. Bayesian quadrature variance in sigma-point filtering. In *Informatics in Control, Automation and Robotics 12th International Conference, ICINCO 2015 Colmar, France, July 21-23, 2015 Revised Selected Papers*, pages 355–370. Springer, 2016.

S. Särkkä and L. Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

S. Särkkä, J. Hartikainen, L. Svensson, and F. Sandblom. On the relation between gaussian process quadratures and sigma-point methods. *arXiv preprint arXiv:1504.05994*, 2015.

M. Schober, D. Duvenaud, and P. Hennig. Probabilistic ode solvers with runge-kutta means. volume 27, 2014.

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

A. H. Stroud, D. Secrest, et al. *Gaussian quadrature formulas*, volume 374. Prentice-Hall Englewood Cliffs, NJ, 1966.

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.

D. Yao, F. Tronarp, and N. Bosch. Propagating model uncertainty through filtering-based probabilistic numerical ode solvers. 2025.

A Continuous-time jointly linear SDE

Consider the continuous-time jointly linear model

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{F}\,\boldsymbol{x}(t) + \boldsymbol{G}\,\boldsymbol{\theta} + \boldsymbol{L}\,\boldsymbol{w}(t), \qquad t \geq t_0, \tag{66}$$

with Gaussian prior

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{m}_{\boldsymbol{\theta}}, \boldsymbol{P}_{\boldsymbol{\theta}}), \qquad \boldsymbol{x}(t_0) \mid \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{m}_{0|\boldsymbol{\theta}}, \boldsymbol{P}_{0|\boldsymbol{\theta}}). \tag{67}$$

Here $\boldsymbol{w}(t)$ denotes vector-valued white noise with spectral density $\boldsymbol{Q}$, that is, $\mathbb{E}[\boldsymbol{w}(t)\boldsymbol{w}(t')^\top] = \boldsymbol{Q}\,\delta(t-t')$.

A.1 Augmented-state representation

Define the augmented state

$$\boldsymbol{z}(t) := \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{\theta} \end{bmatrix}. \tag{68}$$

Under the assumptions that $\boldsymbol{\theta}$ is static and that the drift is linear in $\boldsymbol{\theta}$, Eq. (66) can be written equivalently as the linear SDE

$$\begin{aligned} &\frac{d}{dt}\boldsymbol{z}(t) = \bar{\boldsymbol{F}}\,\boldsymbol{z}(t) + \bar{\boldsymbol{L}}\,\boldsymbol{w}(t), \qquad \bar{\boldsymbol{F}} := \begin{bmatrix} \boldsymbol{F} & \boldsymbol{G} \\ 0 & 0 \end{bmatrix}, \\ &\bar{\boldsymbol{L}} := \begin{bmatrix} \boldsymbol{L} \\ 0 \end{bmatrix}. \end{aligned} \tag{69}$$

This augmentation is the key step that turns uncertain static parameters into part of a larger linear Gaussian dynamical system (Särkkä and Svensson, 2023).

A.2 Mean and covariance dynamics

For a time-invariant linear SDE of the form

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{F}\,\boldsymbol{x}(t)+\boldsymbol{b}(t)+\boldsymbol{L}\,\boldsymbol{w}(t), \qquad \boldsymbol{x}(t_0) \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0), \tag{70}$$

the mean $\boldsymbol{m}(t) := \mathbb{E}[\boldsymbol{x}(t)]$ and covariance $\boldsymbol{P}(t) := \mathrm{Cov}(\boldsymbol{x}(t))$ satisfy (Särkkä and Svensson, 2023)

$$\frac{d}{dt}\boldsymbol{m}(t) = \boldsymbol{F}\,\boldsymbol{m}(t) + \boldsymbol{b}(t), \tag{71}$$

$$\frac{d}{dt}\boldsymbol{P}(t) = \boldsymbol{F}\,\boldsymbol{P}(t) + \boldsymbol{P}(t)\,\boldsymbol{F}^\top + \boldsymbol{L}\,\boldsymbol{Q}\,\boldsymbol{L}^\top. \tag{72}$$

Specialising these equations to the augmented model Eq. (69), where $\boldsymbol{b}(t) \equiv 0$, the augmented mean $\boldsymbol{m}_z(t) := \mathbb{E}[\boldsymbol{z}(t)]$ satisfies

$$\frac{d}{dt}\boldsymbol{m}_z(t) = \bar{\boldsymbol{F}}\,\boldsymbol{m}_z(t), \qquad \boldsymbol{m}_z(t) = \begin{bmatrix}\boldsymbol{m}_x(t)\\ \boldsymbol{m}_{\boldsymbol{\theta}}(t)\end{bmatrix}, \tag{73}$$

and the augmented covariance $\boldsymbol{P}_z(t) := \mathrm{Cov}(\boldsymbol{z}(t))$ satisfies

$$\begin{aligned}\frac{d}{dt}\boldsymbol{P}_z(t) &= \bar{\boldsymbol{F}}\,\boldsymbol{P}_z(t) + \boldsymbol{P}_z(t)\,\bar{\boldsymbol{F}}^\top + \bar{\boldsymbol{L}}\,\boldsymbol{Q}\,\bar{\boldsymbol{L}}^\top,\\ \boldsymbol{P}_z(t) &= \begin{bmatrix}\boldsymbol{P}_{\boldsymbol{x}}(t) & \boldsymbol{P}_{\boldsymbol{x}\boldsymbol{\theta}}(t)\\ \boldsymbol{P}_{\boldsymbol{x}\boldsymbol{\theta}}(t)^\top & \boldsymbol{P}_{\boldsymbol{\theta}}(t)\end{bmatrix}.\end{aligned} \tag{74}$$

Writing this equation blockwise yields the expected structure. In particular, the parameter covariance is static:

$$\frac{d}{dt}\boldsymbol{P}_x(t) = \boldsymbol{F}\,\boldsymbol{P}_x(t) + \boldsymbol{P}_x(t)\,\boldsymbol{F}^\top + \boldsymbol{G}\,\boldsymbol{P}_{x\theta}(t)^\top \tag{75a}$$

$$+ \boldsymbol{P}_{x\theta}(t)\,\boldsymbol{G}^\top + \boldsymbol{L}\,\boldsymbol{Q}\,\boldsymbol{L}^\top, \tag{75b}$$

$$\frac{d}{dt}\boldsymbol{P}_{x\boldsymbol{\theta}}(t) = \boldsymbol{F}\,\boldsymbol{P}_{x\boldsymbol{\theta}}(t) + \boldsymbol{G}\,\boldsymbol{P}_{\boldsymbol{\theta}}(t), \tag{75c}$$

$$\frac{d}{dt}\boldsymbol{P}_{\boldsymbol{\theta}}(t) = 0. \tag{75d}$$

A.3 Closed-form solutions

Let $\boldsymbol{\Sigma}_0 := \boldsymbol{P}_z(t_0)$ denote the joint prior covariance of $(\boldsymbol{x}(t_0), \boldsymbol{\theta})$, and let $\boldsymbol{m}_z(t_0)$ be the corresponding joint prior mean. Then

$$\boldsymbol{m}_z(t) = \exp\big(\bar{\boldsymbol{F}}(t-t_0)\big)\,\boldsymbol{m}_z(t_0), \tag{76a}$$

$$\boldsymbol{P}_z(t) = \exp\big(\bar{\boldsymbol{F}}(t-t_0)\big)\,\boldsymbol{\Sigma}_0\,\exp\big(\bar{\boldsymbol{F}}(t-t_0)\big)^\top \tag{76b}$$

$$+ \int_0^{t-t_0} \exp(\bar{\boldsymbol{F}}s)\,\bar{\boldsymbol{L}}\,\boldsymbol{Q}\,\bar{\boldsymbol{L}}^\top\,\exp(\bar{\boldsymbol{F}}s)^\top\,ds. \tag{76c}$$

These formulas are standard linear-SDE results (Särkkä and Svensson, 2023), but in the present context they show explicitly how parameter uncertainty is transported into state uncertainty and cross-covariance.

A.4 Discrete jointly linear SDE

The exact discretisation of Eq. (70) on a grid $\{t_k\}_{k\geq 0}$ with step size $\Delta t_k := t_{k+1} - t_k$ can be written as (Särkkä and Svensson, 2023)

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k\,\boldsymbol{x}_k + \boldsymbol{u}_k + \boldsymbol{q}_k, \qquad \boldsymbol{q}_k \sim \mathcal{N}(0, \boldsymbol{Q}_k), \tag{77a}$$

$$\boldsymbol{y}_k = \boldsymbol{C}_k\,\boldsymbol{x}_k + \boldsymbol{r}_k, \qquad \boldsymbol{r}_k \sim \mathcal{N}(0, \boldsymbol{R}_k), \tag{77b}$$

where

$$\boldsymbol{A}_k = \exp(\boldsymbol{F}\,\Delta t_k), \tag{78}$$

$$\boldsymbol{u}_k = \int_{t_k}^{t_{k+1}} \exp\big(\boldsymbol{F}\,(t_{k+1}-\tau)\big)\,\boldsymbol{b}(\tau)\,d\tau, \tag{79}$$

$$\boldsymbol{Q}_k = \int_0^{\Delta t_k} \exp(\boldsymbol{F}s)\,\boldsymbol{L}\,\boldsymbol{Q}\,\boldsymbol{L}^\top\,\exp(\boldsymbol{F}s)^\top\,ds. \tag{80}$$

Specialising to the augmented SDE Eq. (69) with constant step size h yields

$$\boldsymbol{z}_{k+1} \mid \boldsymbol{z}_k \sim \mathcal{N}\big(\bar{\boldsymbol{A}}(h)\,\boldsymbol{z}_k,\ \bar{\boldsymbol{Q}}(h)\big), \tag{81}$$

with

$$\bar{\boldsymbol{A}}(h) = \exp(\bar{\boldsymbol{F}}h), \tag{82a}$$

$$\bar{\boldsymbol{Q}}(h) = \int_0^h \exp(\bar{\boldsymbol{F}}s)\,\bar{\boldsymbol{L}}\,\boldsymbol{Q}\,\bar{\boldsymbol{L}}^\top\,\exp(\bar{\boldsymbol{F}}s)^\top\,ds. \tag{82b}$$

The corresponding observation model from Section 4.1 can be written in augmented form as

$$\boldsymbol{y}_k = \begin{bmatrix}\boldsymbol{C}_k & \boldsymbol{D}_k\end{bmatrix}\boldsymbol{z}_k + \boldsymbol{r}_k, \qquad \boldsymbol{r}_k \sim \mathcal{N}(0, \boldsymbol{R}_k). \tag{83}$$

Lemma 1 (Block structure of $\exp(\bar{\boldsymbol{F}}h)$)**.** *For any $h \geq 0$,*

$$\exp(\bar{\boldsymbol{F}}h) = \begin{bmatrix}\boldsymbol{A}(h) & \boldsymbol{B}(h)\\ 0 & \boldsymbol{I}\end{bmatrix}, \qquad \boldsymbol{A}(h) := \exp(\boldsymbol{F}h),$$
$$\boldsymbol{B}(h) := \left(\int_0^h \exp(\boldsymbol{F}s)\,ds\right)\boldsymbol{G}. \tag{84}$$

Proof. Using the power-series definition of the matrix exponential,

$$\exp(\bar{\boldsymbol{F}}h) = \boldsymbol{I} + \sum_{n=1}^{\infty}\frac{h^n}{n!}\,\bar{\boldsymbol{F}}^{\,n}. \tag{85}$$

A direct induction shows that for $n \geq 1$,

$$\bar{\boldsymbol{F}}^{\,n} = \begin{bmatrix}\boldsymbol{F}^n & \boldsymbol{F}^{n-1}\boldsymbol{G}\\ 0 & 0\end{bmatrix}. \tag{86}$$

Substituting this into the series yields

$$\exp(\bar{\boldsymbol{F}}h) = \begin{bmatrix}\boldsymbol{I} + \sum_{n=1}^{\infty}\frac{(\boldsymbol{F}h)^n}{n!} & \sum_{n=1}^{\infty}\frac{h^n}{n!}\,\boldsymbol{F}^{n-1}\boldsymbol{G}\\ 0 & \boldsymbol{I}\end{bmatrix}. \tag{87}$$

The top-left block equals $\exp(\boldsymbol{F}h) = \boldsymbol{A}(h)$. For the top-right block,

$$\sum_{n=1}^{\infty}\frac{h^n}{n!}\,\boldsymbol{F}^{n-1} = \sum_{n=0}^{\infty}\frac{h^{n+1}}{(n+1)!}\,\boldsymbol{F}^n \tag{88}$$

$$= \int_0^h \sum_{n=0}^{\infty}\frac{s^n}{n!}\,\boldsymbol{F}^n\,ds \tag{89}$$

$$= \int_0^h \exp(\boldsymbol{F}s)\,ds, \tag{90}$$

so the block equals $\boldsymbol{B}(h)$, proving Eq. (84). □

A.5 Noise covariance for the augmented discretisation

By Lemma 1,

$$\exp(\bar{\boldsymbol{F}}s)\,\bar{\boldsymbol{L}} = \begin{bmatrix} \boldsymbol{A}(s)\boldsymbol{L} \\ 0 \end{bmatrix}, \tag{91}$$

and hence Eq. (82) implies

$$\begin{aligned} \bar{\boldsymbol{Q}}(h) &= \begin{bmatrix} \boldsymbol{Q}(h) & 0 \\ 0 & 0 \end{bmatrix}, \\ \boldsymbol{Q}(h) &= \int_0^h \boldsymbol{A}(s)\,\boldsymbol{L}\,\boldsymbol{Q}\,\boldsymbol{L}^\top\,\boldsymbol{A}(s)^\top\,ds. \end{aligned} \tag{92}$$

Consequently, the discrete augmented dynamics Eq. (81) can be written as

$$\begin{aligned} &\boldsymbol{x}_{k+1} = \boldsymbol{A}(h)\,\boldsymbol{x}_k + \boldsymbol{B}(h)\,\boldsymbol{\theta} + \boldsymbol{q}_k, \qquad \boldsymbol{q}_k \sim \mathcal{N}(0, \boldsymbol{Q}(h)), \\ &\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k. \end{aligned} \tag{93}$$

A.6 Example: IWP(1)

For the integrated Wiener process prior of order 1 (IWP(1)) with diffusion intensity q_c,

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \boldsymbol{L} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \boldsymbol{Q} = q_c, \tag{94}$$

and Eqs. (84) and (92) yield the familiar closed forms

$$\boldsymbol{A}(h) = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \qquad \boldsymbol{Q}(h) = q_c \begin{bmatrix} h^3/3 & h^2/2 \\ h^2/2 & h \end{bmatrix}. \tag{95}$$

B Kernel used for Bayesian quadrature

Here, we make explicit which kernel is used in the BQ construction of Section 5.1, and how the quantities $\boldsymbol{K}, \boldsymbol{\ell}, \boldsymbol{L}, \boldsymbol{W}$ and σ^2_{int} are computed in closed form.

The construction in Section 5.1 is written for a generic scalar kernel $k(\cdot,\cdot)$. In the present work, this kernel is instantiated as the exponentiated quadratic (EQ) kernel; see, e.g., (O'Hagan, 1991; Prüher and Šimandl, 2016).

Whitened coordinates. Recall from Eq. (50) that the uncertain input is written as

$$\begin{aligned} &\boldsymbol{u} = \boldsymbol{m}_u + \boldsymbol{L}_u\,\boldsymbol{\xi}, \qquad \boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I}_{d_u}), \\ &\boldsymbol{L}_u \boldsymbol{L}_u^\top = \boldsymbol{P}_u. \end{aligned} \tag{96}$$

Kernel. The scalar GP kernel used is

$$k(\boldsymbol{\xi}, \boldsymbol{\xi}') = \alpha^2 \exp\left(-\frac{\|\boldsymbol{\xi} - \boldsymbol{\xi}'\|^2}{2\lambda^2}\right), \tag{97}$$

with amplitude $\alpha^2 > 0$ and length-scale $\lambda > 0$.

Gauss–Hermite sigma points. As described in Section 5.1, the BQ construction itself does not prescribe a unique sigma-point set. In this paper, the points $\{\boldsymbol{\xi}^{(i)}\}_{i=1}^N$ are chosen as tensor-product Gauss–Hermite points for the Gaussian measure; see also (Stroud et al., 1966).

In one dimension, the Gauss–Hermite rule of order n approximates integrals of the form

$$\int_{-\infty}^{\infty} h(x)\,e^{-x^2}\,dx \;\approx\; \sum_{j=1}^{n} \omega_j\,h(x_j), \tag{98}$$

where x_j are the Hermite nodes and ω_j the corresponding weights. Since in Eq. (50) the reference measure is standard Gaussian, we rewrite

$$\begin{aligned} \mathbb{E}_\xi[\varphi(\xi)] &= \int_{\mathbb{R}} \varphi(\xi)\,\frac{1}{\sqrt{2\pi}} e^{-\xi^2/2}\,d\xi \\ &= \frac{1}{\sqrt{\pi}} \int_{\mathbb{R}} \varphi(\sqrt{2}\,x)\,e^{-x^2}\,dx \\ &\approx \sum_{j=1}^{n} \bar{w}_j\,\varphi(\bar{\xi}_j), \end{aligned} \tag{99}$$

with rescaled nodes and weights

$$\bar{\xi}_j := \sqrt{2}\,x_j, \qquad \bar{w}_j := \frac{\omega_j}{\sqrt{\pi}}. \tag{100}$$

In d_u dimensions, the tensor-product rule is obtained by taking Cartesian products of the one-dimensional nodes and multiplying the corresponding weights. Thus,

$$\boldsymbol{\xi}^{(i_1,\ldots,i_{d_u})} = \sqrt{2} \begin{bmatrix} x_{i_1} \\ \vdots \\ x_{i_{d_u}} \end{bmatrix}, \tag{101}$$

with tensor-product weights

$$\bar{w}_{i_1,\ldots,i_{d_u}} = \prod_{r=1}^{d_u} \frac{\omega_{i_r}}{\sqrt{\pi}}. \tag{102}$$

These are the sigma points used to evaluate the conditional solver mean map g_k from Eq. (51).

B.1 Closed-form kernel quantities

Given the sigma points $\{\boldsymbol{\xi}^{(i)}\}_{i=1}^N$, we now define the matrices and vectors appearing in Eqs. (53), (54), (57) and (60).

Gram matrix. The Gram matrix from Eq. (53) is

$$\boldsymbol{K}_{ij} = k(\boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}^{(j)}) = \alpha^2 \exp\left(-\frac{\|\boldsymbol{\xi}^{(i)} - \boldsymbol{\xi}^{(j)}\|^2}{2\lambda^2}\right). \tag{103}$$

Kernel mean vector. The vector $\boldsymbol{\ell}$ from Eq. (54) has entries

$$\boldsymbol{\ell}_i := \mathbb{E}_{\boldsymbol{\xi}}\big[k(\boldsymbol{\xi}, \boldsymbol{\xi}^{(i)})\big], \qquad \boldsymbol{\xi} \sim \mathcal{N}(0, \boldsymbol{I}_{d_u}). \tag{104}$$

For the EQ kernel Eq. (97), this Gaussian integral is available in closed form:

$$\boldsymbol{\ell}_i = \alpha^2 \left(\frac{\lambda^2}{\lambda^2 + 1}\right)^{d_u/2} \exp\left(-\frac{\|\boldsymbol{\xi}^{(i)}\|^2}{2(\lambda^2 + 1)}\right). \tag{105}$$

Kernel second-moment matrix. The matrix $\boldsymbol{L}$ from Eq. (55) has entries

$$\boldsymbol{L}_{ij} := \mathbb{E}_{\boldsymbol{\xi}}\big[k(\boldsymbol{\xi}, \boldsymbol{\xi}^{(i)})\, k(\boldsymbol{\xi}, \boldsymbol{\xi}^{(j)})\big]. \tag{106}$$

Again, for the EQ kernel this integral can be evaluated in closed form:

$$\boldsymbol{L}_{ij} = \alpha^4 \left(\frac{\lambda^2}{\lambda^2+2}\right)^{d_u/2} \times \exp\left(\frac{2\,(\boldsymbol{\xi}^{(i)})^\top \boldsymbol{\xi}^{(j)} - (\lambda^2+1)\big(\|\boldsymbol{\xi}^{(i)}\|^2 + \|\boldsymbol{\xi}^{(j)}\|^2\big)}{2\lambda^2(\lambda^2+2)}\right). \tag{107}$$

Bayesian quadrature weights. With $\boldsymbol{K}$, $\boldsymbol{\ell}$, and $\boldsymbol{L}$ defined above, the BQ moment transform uses

$$\boldsymbol{w} = \boldsymbol{K}^{-1}\boldsymbol{\ell}, \qquad \boldsymbol{W} = \boldsymbol{K}^{-1}\boldsymbol{L}\boldsymbol{K}^{-1}, \tag{108}$$

exactly as in Eqs. (56) and (57). Therefore, the Bayesian quadrature approximation of the propagated mean is

$$\boldsymbol{\mu}_k^{\mathrm{BQ}} = \boldsymbol{G}_k^\top \boldsymbol{w}, \tag{109}$$

with $\boldsymbol{G}_k$ defined in Section 5.1 by

$$(\boldsymbol{G}_k)_{i,:} = g_k(\boldsymbol{\xi}^{(i)})^\top. \tag{110}$$

Integration-variance term. Finally, the additional scalar variance term from Eq. (60) is

$$\sigma_{\mathrm{int}}^2 = \bar{k} - \mathrm{tr}(\boldsymbol{K}^{-1}\boldsymbol{L}). \tag{111}$$

For the EQ kernel Eq. (97),

$$k(\boldsymbol{\xi}, \boldsymbol{\xi}) = \alpha^2 \qquad \text{for all } \boldsymbol{\xi}, \tag{112}$$

so that

$$\bar{k} = \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi})] = \alpha^2. \tag{113}$$

Hence

$$\sigma_{\mathrm{int}}^2 = \alpha^2 - \mathrm{tr}(\boldsymbol{K}^{-1}\boldsymbol{L}). \tag{114}$$

B.2 Resulting BQ moment approximation

Using the notation of Section 5.1, the covariance contribution induced by uncertain IVP inputs is approximated as

$$\widehat{\boldsymbol{\Pi}}_k^{\mathrm{BQ}} = \boldsymbol{G}_k^\top \boldsymbol{W} \boldsymbol{G}_k - \boldsymbol{\mu}_k^{\mathrm{BQ}} (\boldsymbol{\mu}_k^{\mathrm{BQ}})^\top, \tag{115}$$

which is exactly the structure of Eq. (59). Together with the Gauss–Hermite approximation of the average conditional covariance,

$$\widehat{\mathbb{E}_{\boldsymbol{u}}[\bar{\boldsymbol{P}}_k(\boldsymbol{u})]} = \sum_{i=1}^{N} w_i\, \bar{\boldsymbol{P}}_k(\boldsymbol{u}^{(i)}), \tag{116}$$

this yields the propagated covariance approximation

$$\boldsymbol{P}_k^{\mathrm{prop,BQ}} = \widehat{\mathbb{E}_{\boldsymbol{u}}[\bar{\boldsymbol{P}}_k(\boldsymbol{u})]} + \widehat{\boldsymbol{\Pi}}_k^{\mathrm{BQ}} + \sigma_{\mathrm{int}}^2 \boldsymbol{I}_{d_x}. \tag{117}$$

C Used parameters for benchmark problems

In Table 2, the settings for the benchmark problems are given.

D Matched-node comparison with classical quadrature

To compare the outer integration rules directly, classical Gauss–Hermite quadrature and BQ are evaluated on the same tensor-product Gauss–Hermite node set. At each node, the same conditional probabilistic ODE solver is run. Hence the conditional solver output is fixed across the two methods, and the comparison isolates the effect of the outer moment weights.

Classical Gauss–Hermite quadrature uses deterministic weights $\bar{\boldsymbol{w}}$. With $\boldsymbol{G}_k \in \mathbb{R}^{N \times d_x}$ denoting the matrix of conditional solver means at time t_k, its propagated mean and IVP covariance are

$$\boldsymbol{\mu}_k^{\mathrm{GH}} = \boldsymbol{G}_k^\top \bar{\boldsymbol{w}}, \tag{118}$$

$$\widehat{\boldsymbol{\Pi}}_k^{\mathrm{GH}} = \boldsymbol{G}_k^\top \mathrm{diag}(\bar{\boldsymbol{w}}) \boldsymbol{G}_k - \boldsymbol{\mu}_k^{\mathrm{GH}} (\boldsymbol{\mu}_k^{\mathrm{GH}})^\top. \tag{119}$$

BQ uses the same evaluations $\boldsymbol{G}_k$, but replaces the deterministic Gauss–Hermite weights by kernel-dependent Bayesian quadrature weights,

$$\boldsymbol{w} = \boldsymbol{K}^{-1}\boldsymbol{\ell}, \qquad \boldsymbol{W} = \boldsymbol{K}^{-1}\boldsymbol{L}\boldsymbol{K}^{-1}. \tag{120}$$

The corresponding moments are

$$\boldsymbol{\mu}_k^{\mathrm{BQ}} = \boldsymbol{G}_k^\top \boldsymbol{w}, \tag{121}$$

$$\widehat{\boldsymbol{\Pi}}_k^{\mathrm{BQ}} = \boldsymbol{G}_k^\top \boldsymbol{W} \boldsymbol{G}_k - \boldsymbol{\mu}_k^{\mathrm{BQ}} (\boldsymbol{\mu}_k^{\mathrm{BQ}})^\top. \tag{122}$$

Thus, in Fig. 5, the difference between GH and BQ is not caused by different solver calls or different sigma-point locations, but by the different weights used to approximate the outer Gaussian-weighted moments. Both methods converge to the same result as the number of nodes increases. For GH, the convergence rate depends primarily on how accurately the propagated quantity can be represented by the chosen quadrature order, whereas for BQ it additionally depends on the kernel and its hyperparameters.

The second distinction is that BQ also reports an outer quadrature uncertainty,

$$\boldsymbol{\Sigma}_k^{\mathrm{out}} = \sigma_{\mathrm{int}}^2 \boldsymbol{I}_{d_x}, \qquad \sigma_{\mathrm{int}}^2 = \bar{k} - \mathrm{tr}(\boldsymbol{K}^{-1}\boldsymbol{L}),$$

whereas deterministic Gauss–Hermite quadrature does not quantify an uncertainty estimate. The main benefit of BQ in this comparison is therefore not necessarily a smaller matched-node point estimate, but the availability of an explicit uncertainty estimate for the outer integration step.

E Trajectory-level benchmark diagnostics

Fig. 6 gives trajectories for the three benchmark problems considered in Section 6.

The linear benchmark in Fig. 6a illustrates the expected expansion of initial-condition uncertainty. Since the exact flow is $x(t) = e^t x(0)$, the variance induced by the uncertain initial condition grows like $e^{2t}\mathrm{Var}(x(0))$. The

Table 2: Benchmark IVPs and uncertainty specifications. The Gauss–Hermite (GH) settings are those used to evaluate the BQ moments.

Problem	Uncertain inputs $\boldsymbol{u}$	Prior on $\boldsymbol{u}$	d_u	GH order	# nodes N
Linear	$x(0)$	$\mathcal{N}(1, 10^{-2})$	1	20	20
Logistic	b	$\mathcal{N}(3, 10^{-2})$	1	50	50
Lotka–Volterra	$\boldsymbol{x}(0)$	$\mathcal{N}([5,5]^\top, 0.3\boldsymbol{I}_2)$	2	12	$12^2 = 144$

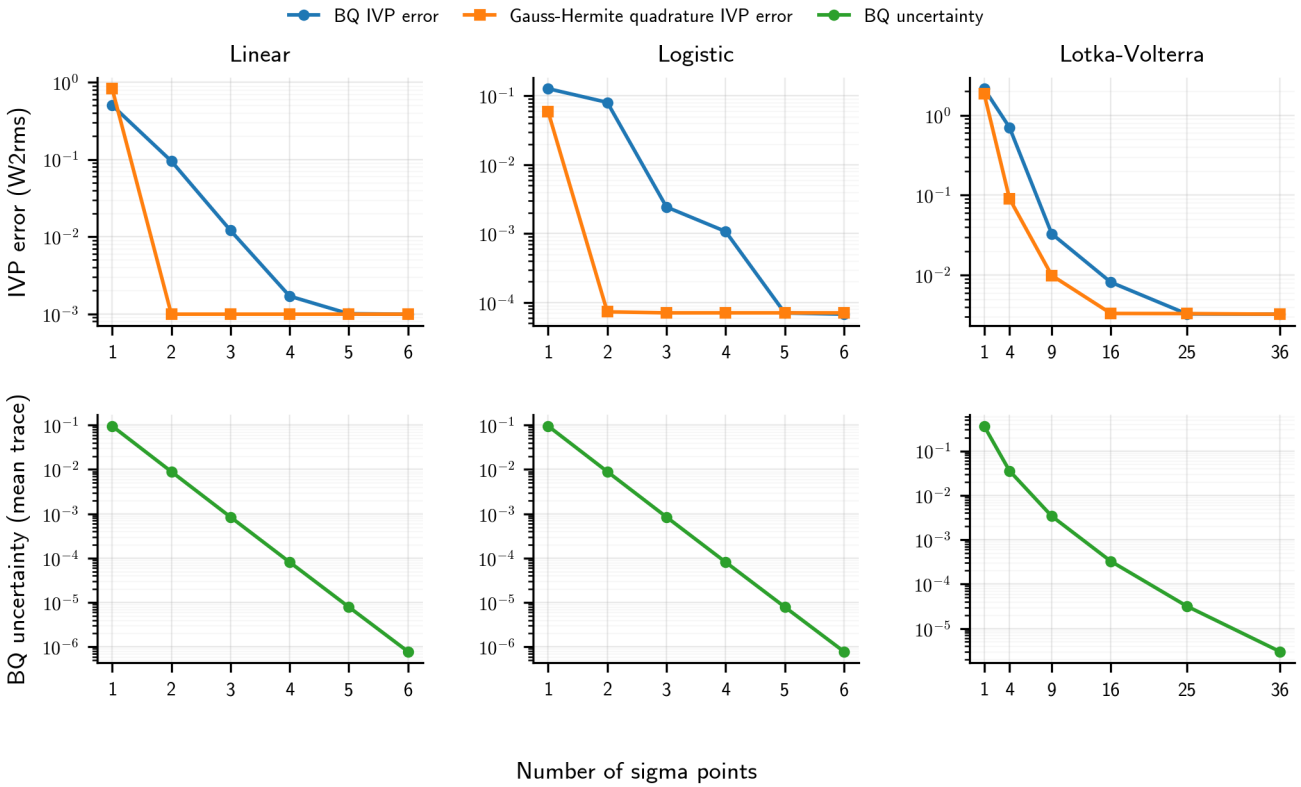


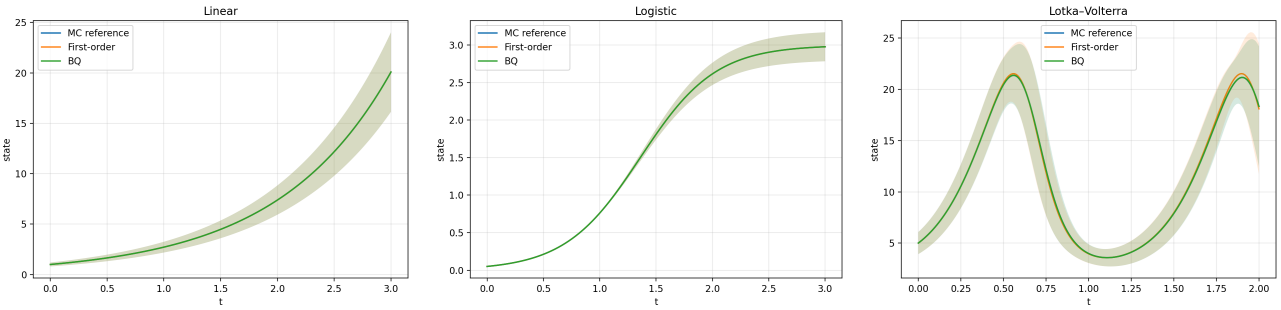
Figure 5: Matched-node comparison of classical Gauss–Hermite quadrature and BQ. Both methods use the same tensor-product Gauss–Hermite nodes and the same conditional probabilistic ODE solver evaluations. The difference is only in the outer integration weights. The top row reports the $W_{2,\mathrm{rms}}$ error of the propagated IVP moments relative to a refined reference. The bottom row reports the mean trace of the BQ outer uncertainty Σ_k^{out}.

propagated uncertainty therefore increases over time and is not reduced by making the numerical solve more accurate. This is the behaviour that the separation between conditional solver uncertainty and IVP uncertainty is intended to preserve.

The logistic benchmark in Fig. 6b shows a different qualitative regime. The dynamics approach an attracting level, and the propagated uncertainty does not grow in the same way as in the unstable linear problem. In this experiment the uncertain input is the carrying-capacity parameter b, rather than the initial condition. The uncertainty therefore affects the asymptotic level of the trajectory. This illustrates that initial-condition and parameter uncertainty are treated within the same outer marginalisation framework, but their propagated effects differ through the corresponding input-to-solution sensitivities.

The Lotka–Volterra benchmark in Fig. 6c illustrates the nonlinear coupled case. Here the uncertain initial condition is propagated through oscillatory dynamics, so the input-to-solution map is no longer close to affine. This explains why the first-order approximation is less accurate in this benchmark, while BQ benefits from evaluating the nonlinear conditional solver at multiple input locations.

Overall, these trajectory plots show that the proposed propagation framework does not impose either expansion or contraction of uncertainty. The qualitative behaviour is determined by the IVP dynamics and by the uncertain input under consideration.



(a) Linear benchmark. (b) Logistic benchmark. (c) Lotka–Volterra benchmark.

Figure 6: Trajectory-level propagated uncertainty bands for the three benchmark IVPs. The blue curve denotes the Monte Carlo reference, the orange curve the first-order approximation, and the green curve the BQ approximation. Shaded regions show the corresponding propagated uncertainty 95% bands.