Propagating Model Uncertainty through Filtering-based Probabilistic Numerical ODE Solvers

Dingling Yao, Filip Tronarp, Nathanael Bosch

2. September 2025

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

imprs-is

erc some of the presented work is supported by the European Research Council.

What are PN methods and what capabilities do they provide?

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

https://www.probabilistic-numerics.org/

Value of a Probabilistic Approach

As well as offering an enriched reinterpretation of classical methods, the PN approach has several concrete practical points of value. The probabilistic interpretation of computation

- allows to build customized methods for specific problems with bespoke priors
- formalizes the design of adaptive methods using tools from decision theory
- provides a way of setting parameters of numerical methods via the Bayesian formalism
- expedites the solution of mutually related problems of similar type
- naturally incorporates sources of stochasticity in the computation
- can give structural uncertainty via a probability measure compared to an error estimate

and finally it offers a principled approach of including numerical error in the propagation of uncertainty through chains of computations.

What are PN methods and what capabilities do they provide?

https://www.probabilistic-numerics.org/

Value of a Probabilistic Approach

As well as offering an enriched reinterpretation of classical methods, the PN approach has several concrete practical points of value. The probabilistic interpretation of computation

- allows to build customized methods for specific problems with bespoke priors
- formalizes the design of adaptive methods using tools from decision theory
- provides a way of setting parameters of numerical methods via the Bayesian formalism
- expedites the solution of mutually related problems of similar type
- naturally incorporates sources of stochasticity in the computation
- can give structural uncertainty via a probability measure compared to an error estimate

and finally it offers a principled approach of including numerical error in the propagation of uncertainty through chains of computations.

https://en.wikipedia.org/wiki/Probabilistic_numerics

- Because all probabilistic numerical methods use essentially the same data type – probability measures – to quantify uncertainty over *both inputs and outputs* they can be chained together to propagate uncertainty across large-scale, composite computations

A common feature request for probabilistic ODE solvers

TheFibonacciEffect opened on Jul 13, 2024 · edited by TheFibonacciEffect

Edits

Is it possible to use a Gaussian as an intial condition?

For example something like this?

```
using LinearAlgebra, Statistics, Distributions
using DiffEqDevTools, ParameterizedFunctions, SciMLBase, OrdinaryDiffEq, Sundials, Plots, ODEInterfaceDiffEq
using ModelingToolkit
using ProbNumDiffEq

function osscilator!(ddu, du, u, p,t)
    ddu .= - p .* u
end

# osscilator
u0 = [1]
du0 = [1]
p = 100
T = 3
t = 0:0.1:T
prob = SecondOrderODEProblem(osscilator!, du0, u0, (0,T),p)
@time sol = solve(prob, EK0(;smooth=true), abstol=1e-1, reltol=1e-1)
plot(sol)
```

Where I would like to use
`u0 = [Gaussian(1,1)]` or something similar as an intial condition instead.

Currently it does not seem to be suported: `ERROR: MethodError: no method matching zero(::Type{Any})` but it would be very useful. For example when the initial condition is a result of a measurement and is not known with infinite precision.

Create sub-issue

Filtering-based probabilistic numerical ODE solvers

- Ordinary differential equation (ODE):

$$\dot{y}(t) = f(y(t), t), \qquad y(0) = y_0,$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial value y_0.

- Ordinary differential equation (ODE):

$$\dot{y}(t) = f(y(t), t), \qquad y(0) = y_0,$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial value y_0.

- Probabilistic numerical ODE solution:

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^{N}\right)$$

for a chosen time discretization $\{t_n\}_{n=1}^{N}$.

Filtering-based probabilistic numerical ODE solvers

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f\left(y(t_n), t_n\right)\}_{n=1}^{N}\right)$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value y_0, and time discretization $\{t_n\}_{n=1}^{N}$.

Filtering-based probabilistic numerical ODE solvers

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^{N}\right)$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value y_0, and time discretization $\{t_n\}_{n=1}^{N}$.

- **Prior:** $y(t) \sim \mathcal{GP}$

Filtering-based probabilistic numerical ODE solvers

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^{N}\right)$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value y_0, and time discretization $\{t_n\}_{n=1}^{N}$.

- **Prior:** $y(t) \sim \mathcal{GP}$ a Gauss–Markov process

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^{N}\right)$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value y_0, and time discretization $\{t_n\}_{n=1}^{N}$.

- **Prior:** $y(t) \sim \mathcal{GP}$ a Gauss–Markov process with state-space representation $x(t)$:

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^{N}\right)$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value y_0, and time discretization $\{t_n\}_{n=1}^{N}$.

- **Prior:** $y(t) \sim \mathcal{GP}$ a Gauss–Markov process with state-space representation $x(t)$:

$$x(0) \sim \mathcal{N}\left(\mu_0^-, \Sigma_0^-\right),$$

$$x(t+h) \mid x(t) \sim \mathcal{N}\left(A(h)x(t), \sigma^2 Q(h)\right),$$

$$y(t) = E_0 x(t), \qquad \dot{y}(t) = E_1 x(t),$$

where A, Q define the Gauss–Markov prior (e.g. an *integrated Wiener process*).

Filtering-based probabilistic numerical ODE solvers

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^{N}\right)$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value y_0, and time discretization $\{t_n\}_{n=1}^N$.

- **Prior:** $y(t) \sim \mathcal{GP}$ a Gauss–Markov process with state-space representation $x(t)$:

$$x(0) \sim \mathcal{N}\left(\mu_0^-, \Sigma_0^-\right),$$

$$x(t+h) \mid x(t) \sim \mathcal{N}\left(A(h)x(t), \sigma^2 Q(h)\right),$$

$$y(t) = E_0 x(t), \qquad \dot{y}(t) = E_1 x(t),$$

where A, Q define the Gauss–Markov prior (e.g. an *integrated Wiener process*).
To satisfy the initial condition, set μ_0^- to match y_0 and set $\Sigma_0^- = 0$.

Filtering-based probabilistic numerical ODE solvers

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^N\right)$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value y_0, and time discretization $\{t_n\}_{n=1}^N$.

- **Prior:** $y(t) \sim \mathcal{GP}$ a Gauss–Markov process with state-space representation $x(t)$:

$$x(0) \sim \mathcal{N}\left(\mu_0^-, \Sigma_0^-\right),$$

$$x(t+h) \mid x(t) \sim \mathcal{N}\left(A(h)x(t), \sigma^2 Q(h)\right),$$

$$y(t) = E_0 x(t), \qquad \dot{y}(t) = E_1 x(t),$$

  where A, Q define the Gauss–Markov prior (e.g. an *integrated Wiener process*).
  To satisfy the initial condition, set μ_0^- to match y_0 and set $\Sigma_0^- = 0$.
- **Likelihood:** (aka "observation model" or "information operator")

$$z(t_n) = E_1 x(t_n) - f(E_0 x(t_n), t_n) \equiv 0$$

Filtering-based probabilistic numerical ODE solvers

$$p\left(y(t) \mid y(0) = y_0, \{\dot{y}(t_n) = f(y(t_n), t_n)\}_{n=1}^N\right)$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, initial value y_0, and time discretization $\{t_n\}_{n=1}^N$.

- **Prior:** $y(t) \sim \mathcal{GP}$ a Gauss–Markov process with state-space representation $x(t)$:

$$x(0) \sim \mathcal{N}\left(\mu_0^-, \Sigma_0^-\right),$$

$$x(t+h) \mid x(t) \sim \mathcal{N}\left(A(h)x(t), \sigma^2 Q(h)\right),$$

$$y(t) = E_0 x(t), \qquad \dot{y}(t) = E_1 x(t),$$

  where A, Q define the Gauss–Markov prior (e.g. an *integrated Wiener process*).
  To satisfy the initial condition, set μ_0^- to match y_0 and set $\Sigma_0^- = 0$.
- **Likelihood:** (aka “observation model” or “information operator”)

$$z(t_n) = E_1 x(t_n) - f(E_0 x(t_n), t_n) \equiv 0$$

- **Inference:** Extended Kalman filter / smoother (or other Bayesian filtering / smoothing methods).

The extended Kalman filter

Given a state-space model:

$$
\begin{aligned}
x(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \\
x(t+h) \mid x(t) &\sim \mathcal{N}(Ax(t), Q), \\
z(t_n) &= \underbrace{E_1 x(t_n) - f(E_0 x(t_n), t_n)}_{=: h(x(t_n))} \equiv 0
\end{aligned}
$$

Given a state-space model:

$$\begin{aligned} x(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \\ x(t+h) \mid x(t) &\sim \mathcal{N}(Ax(t), Q), \\ z(t_n) &= \underbrace{E_1 x(t_n) - f(E_0 x(t_n), t_n)}_{=:h(x(t_n))} \equiv 0 \end{aligned}$$

The EKF computes:

$$\begin{aligned} p(x(t_n) \mid z(t_{1:n-1})) &\approx \mathcal{N}\left(\mu^P, \Sigma^P\right) \\ p(x(t_n) \mid z(t_{1:n})) &\approx \mathcal{N}\left(\mu^F, \Sigma^F\right) \end{aligned}$$

by iterating *prediction* and *update* steps.

The extended Kalman filter

Given a state-space model:

$$
\begin{aligned}
x(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \\
x(t+h) \mid x(t) &\sim \mathcal{N}(Ax(t), Q), \\
z(t_n) &= \underbrace{E_1 x(t_n) - f(E_0 x(t_n), t_n)}_{=: h(x(t_n))} \equiv 0
\end{aligned}
$$

The EKF computes:

$$
\begin{aligned}
p(x(t_n) \mid z(t_{1:n-1})) &\approx \mathcal{N}\left(\mu^P, \Sigma^P\right) \\
p(x(t_n) \mid z(t_{1:n})) &\approx \mathcal{N}\left(\mu^F, \Sigma^F\right)
\end{aligned}
$$

by iterating *prediction* and *update* steps.

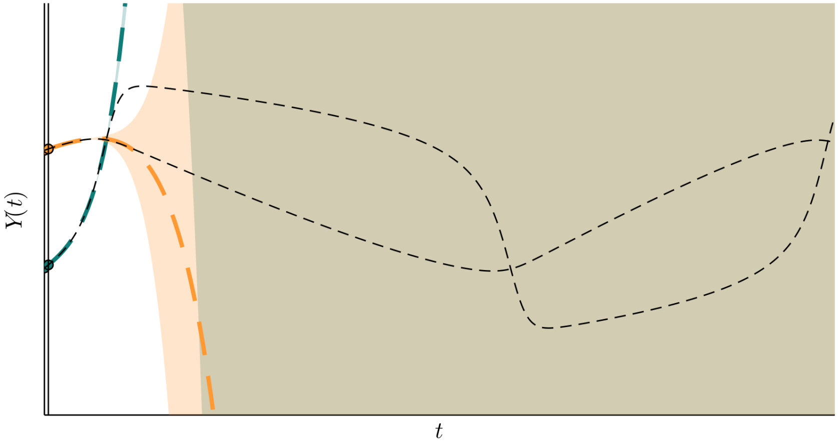
Algorithm Kalman filter prediction

```
procedure KF_PREDICT(μ, Σ, A, Q)
    μ^P ← Aμ                          // Predict mean
    Σ^P ← AΣA^⊤ + Q                   // Predict covariance
    return μ^P, Σ^P
end procedure
```

Algorithm Extended Kalman filter update

```
procedure EKF_UPDATE(μ, Σ, h)
    ẑ ← h(μ)                          // evaluate the observation model
    H ← J_h(μ)                        // Jacobian of the observation model
    S ← HΣH^⊤                         // Measurement covariance
    K ← ΣH^⊤S^{-1}                    // Kalman gain
    μ^F ← μ + K(0 − ẑ)                // update mean
    Σ^F ← Σ − KSK^⊤                   // update covariance
    return μ^F, Σ^F
end procedure
```

Probabilistic numerical ODE solvers in action



Uncertainty propagation in ordinary differential equations

- **Ordinary differential equation *with uncertain initial condition*:**

$$\dot{y}(t) = f\left(y(t), t\right), \qquad y(0) \sim \mathcal{N}(\mu_0, \Sigma_0),$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial *distribution* $\mathcal{N}(\mu_0, \Sigma_0)$.

- **Ordinary differential equation *with uncertain initial condition*:**

$$\dot{y}(t) = f(y(t), t), \qquad y(0) \sim \mathcal{N}(\mu_0, \Sigma_0),$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial *distribution* $\mathcal{N}(\mu_0, \Sigma_0)$.

- **An attempt at solving this with ODE filters:** Remember the ODE filter definition

$$x(0) \sim \mathcal{N}(\mu_0^-, \Sigma_0^-),$$
$$x(t+h) \mid x(t) \sim \mathcal{N}\left(A(h)x(t), \sigma^2 Q(h)\right),$$
$$z(t_n) = E_1 x(t_n) - f(E_0 x(t_n), t_n) \equiv 0.$$

ODEs with uncertain initial values and how to solve them

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Ordinary differential equation *with uncertain initial condition*:**

$$\dot{y}(t) = f(y(t), t), \qquad y(0) \sim \mathcal{N}\left(\boxed{\mu_0}, \boxed{\Sigma_0}\right),$$

with vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and initial *distribution* $\mathcal{N}(\mu_0, \Sigma_0)$.

- **An attempt at solving this with ODE filters:** Remember the ODE filter definition

$$\begin{aligned} x(0) &\sim \mathcal{N}\left(\boxed{\mu_0^-}, \boxed{\Sigma_0^-}\right), \\ x(t+h) \mid x(t) &\sim \mathcal{N}\left(A(h)x(t), \sigma^2 Q(h)\right), \\ z(t_n) &= E_1 x(t_n) - f(E_0 x(t_n), t_n) \equiv 0. \end{aligned}$$

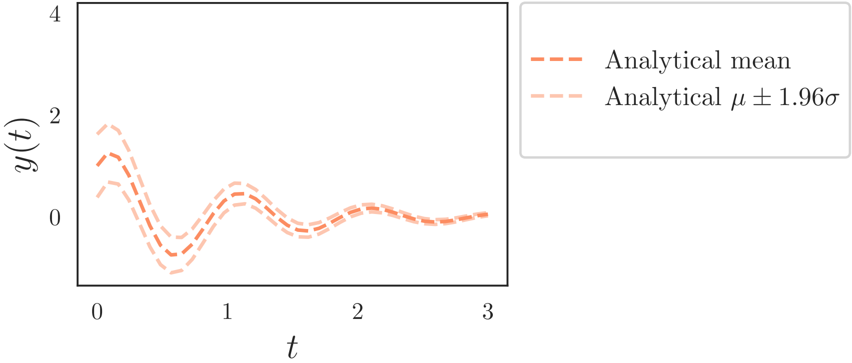
$\Rightarrow$ **Idea:** Just set (μ_0^-, Σ_0^-) to match the true initial distribution $\mathcal{N}(\mu_0, \Sigma_0)$!

Applying this “solution” to the github example

- **ODE:** A simple linear damped oscillator

$$\dot{y}(t) = Ly(t), \qquad y(0) \sim \mathcal{N}(\mu_0, \Sigma_0).$$

- **Result:**

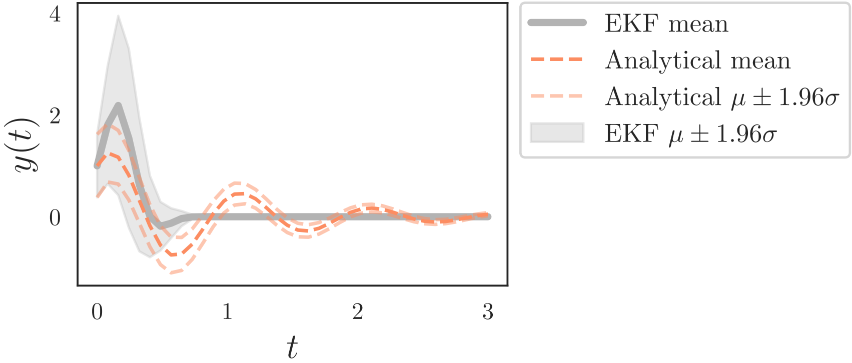


Applying this “solution” to the github example

- **ODE:** A simple linear damped oscillator

$$\dot{y}(t) = Ly(t), \qquad y(0) \sim \mathcal{N}(\mu_0, \Sigma_0).$$

- **Result:**

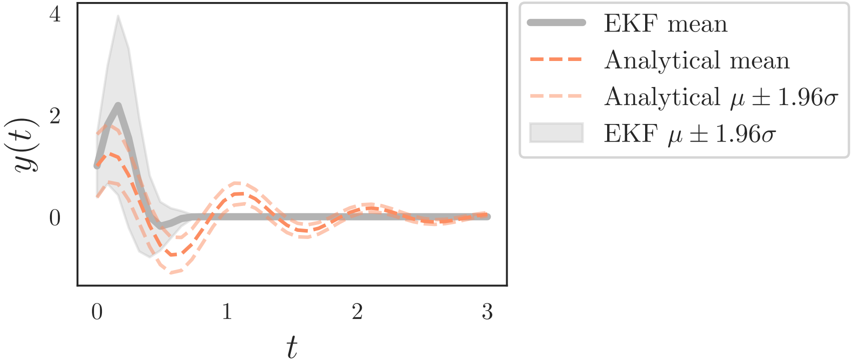


Applying this “solution” to the github example

- **ODE:** A simple linear damped oscillator

$$\dot{y}(t) = Ly(t), \qquad y(0) \sim \mathcal{N}(\mu_0, \Sigma_0).$$

- **Result:**



What’s going on???

Let’s simplify the problem

Simplifying the problem to just a single step

- **Simplified problem:** A generic state-space model with only a single time step:

Unknown initial state:	$p(x_0)$
Transition model:	$p(x_1 \mid x_0)$
Observation model:	$p(z_1 \mid x_1)$

Simplifying the problem to just a single step

- **Simplified problem:** A generic state-space model with only a single time step:

Unknown initial state:	$p(x_0)$
Transition model:	$p(x_1 \mid x_0)$
Observation model:	$p(z_1 \mid x_1)$

- **Goal:** Learn from z_1 and marginalize out x_0

$$p_{\mathrm{UP}}(x_1 \mid z_1) = \int p(x_1 \mid z_1, x_0) p(x_0) \, \mathrm{d}x_0$$

Simplifying the problem to just a single step

- **Simplified problem:** A generic state-space model with only a single time step:

Unknown initial state:	$p(x_0)$
Transition model:	$p(x_1 \mid x_0)$
Observation model:	$p(z_1 \mid x_1)$

- **Goal:** Learn from z_1 and marginalize out x_0

$$p_{\mathrm{UP}}(x_1 \mid z_1) = \int p(x_1 \mid z_1, x_0) p(x_0) \, \mathrm{d}x_0$$

- What **Bayesian filtering** computes: **Predict:** Marginalize out x_0

$$p_{\mathrm{predict}}(x_1) = \int p(x_1 \mid x_0) p(x_0) \, \mathrm{d}x_0$$

Simplifying the problem to just a single step

- **Simplified problem:** A generic state-space model with only a single time step:

Unknown initial state:	$p(x_0)$
Transition model:	$p(x_1 \mid x_0)$
Observation model:	$p(z_1 \mid x_1)$

- **Goal:** Learn from z_1 and marginalize out x_0

$$p_{\text{UP}}(x_1 \mid z_1) = \int p(x_1 \mid z_1, x_0) p(x_0) \, \mathrm{d}x_0$$

- What **Bayesian filtering** computes: **Predict:** Marginalize out x_0

$$p_{\text{predict}}(x_1) = \int p(x_1 \mid x_0) p(x_0) \, \mathrm{d}x_0$$

Update: Learn from z_1 with Bayes' rule

$$p_{\text{filter}}(x_1 \mid z_1) = \frac{p(z_1 \mid x_1) \, p_{\text{predict}}(x_1)}{p(z_1)}$$

Simplifying the problem to just a single step

- **Simplified problem:** A generic state-space model with only a single time step:

Unknown initial state:	$p(x_0)$
Transition model:	$p(x_1 \mid x_0)$
Observation model:	$p(z_1 \mid x_1)$

- **Goal:** Learn from z_1 and marginalize out x_0

$$p_{\mathrm{UP}}(x_1 \mid z_1) = \int p(x_1 \mid z_1, x_0) p(x_0) \, \mathrm{d}x_0$$

- What **Bayesian filtering** computes: **Predict:** Marginalize out x_0

$$p_{\mathrm{predict}}(x_1) = \int p(x_1 \mid x_0) p(x_0) \, \mathrm{d}x_0$$

Update: Learn from z_1 with Bayes' rule

$$p_{\mathrm{filter}}(x_1 \mid z_1) = \frac{p(z_1 \mid x_1) \, p_{\mathrm{predict}}(x_1)}{p(z_1)}$$

Together:

$$p_{\mathrm{filter}}(x_1 \mid z_1) = \int \frac{p(z_1 \mid x_1) \, p(x_1 \mid x_0)}{p(z_1)} p(x_0) \, \mathrm{d}x_0 \, .$$

Simplifying the problem to just a single step

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Simplified problem:** A generic state-space model with only a single time step:

Unknown initial state:	$p(x_0)$
Transition model:	$p(x_1 \mid x_0)$
Observation model:	$p(z_1 \mid x_1)$

- **Goal:** Learn from z_1 and marginalize out x_0

$$p_{\mathrm{UP}}(x_1 \mid z_1) = \int p(x_1 \mid z_1, x_0) p(x_0) \, \mathrm{d}x_0$$

$$= \int \frac{p(z_1 \mid x_1) \, p(x_1 \mid x_0)}{p(z_1 \mid x_0)} p(x_0) \, \mathrm{d}x_0$$

- What **Bayesian filtering** computes: **Predict:** Marginalize out x_0

$$p_{\mathrm{predict}}(x_1) = \int p(x_1 \mid x_0) p(x_0) \, \mathrm{d}x_0$$

Update: Learn from z_1 with Bayes' rule

$$p_{\mathrm{filter}}(x_1 \mid z_1) = \frac{p(z_1 \mid x_1) \, p_{\mathrm{predict}}(x_1)}{p(z_1)}$$

Together:

$$p_{\mathrm{filter}}(x_1 \mid z_1) = \int \frac{p(z_1 \mid x_1) \, p(x_1 \mid x_0)}{p(z_1)} p(x_0) \, \mathrm{d}x_0 \, .$$

Simplifying the problem to just a single step

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Simplified problem:** A generic state-space model with only a single time step:

Unknown initial state:	$p(x_0)$
Transition model:	$p(x_1 \mid x_0)$
Observation model:	$p(z_1 \mid x_1)$

- **Goal:** Learn from z_1 and marginalize out x_0

$$p_{\text{UP}}(x_1 \mid z_1) = \int p(x_1 \mid z_1, x_0) p(x_0) \, \mathrm{d}x_0 = \int \frac{p(z_1 \mid x_1)\, p(x_1 \mid x_0)}{p(z_1 \mid x_0)} p(x_0) \, \mathrm{d}x_0$$

- What **Bayesian filtering** computes: **Predict:** Marginalize out x_0

$$p_{\text{predict}}(x_1) = \int p(x_1 \mid x_0) p(x_0) \, \mathrm{d}x_0$$

Update: Learn from z_1 with Bayes' rule

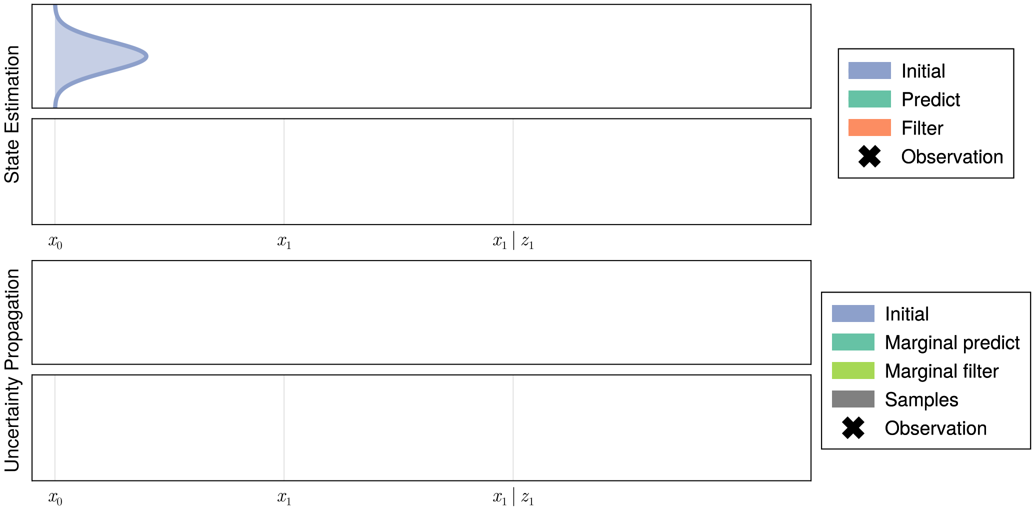
$$p_{\text{filter}}(x_1 \mid z_1) = \frac{p(z_1 \mid x_1)\, p_{\text{predict}}(x_1)}{p(z_1)}$$

Together:

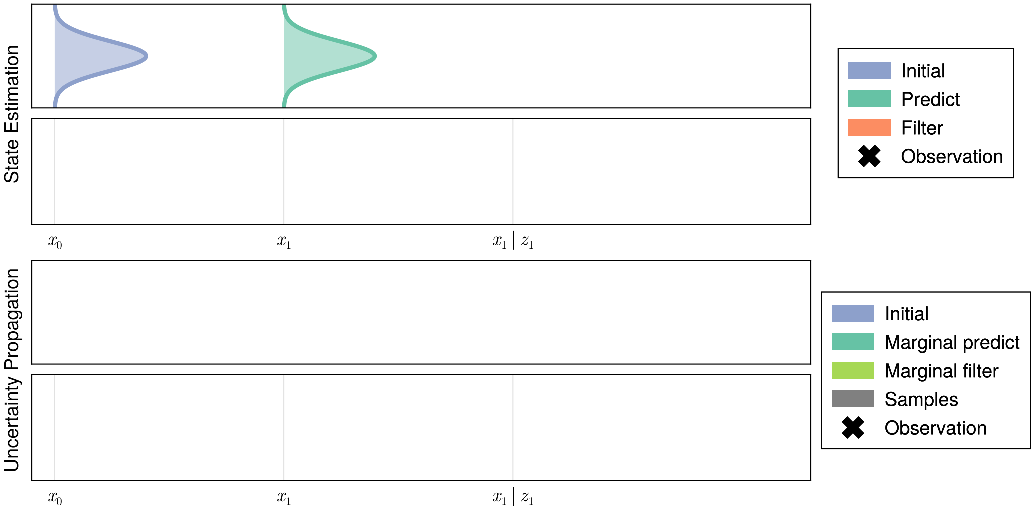
$$p_{\text{filter}}(x_1 \mid z_1) = \int \frac{p(z_1 \mid x_1)\, p(x_1 \mid x_0)}{p(z_1)} p(x_0) \, \mathrm{d}x_0 \, .$$

$$p_{\text{UP}} \neq p_{\text{filter}}$$

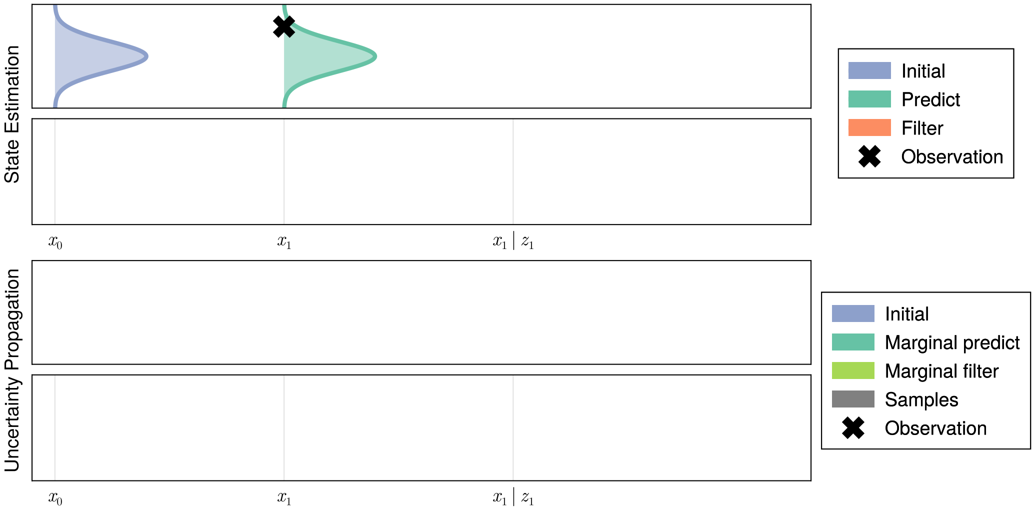
A visual demonstration



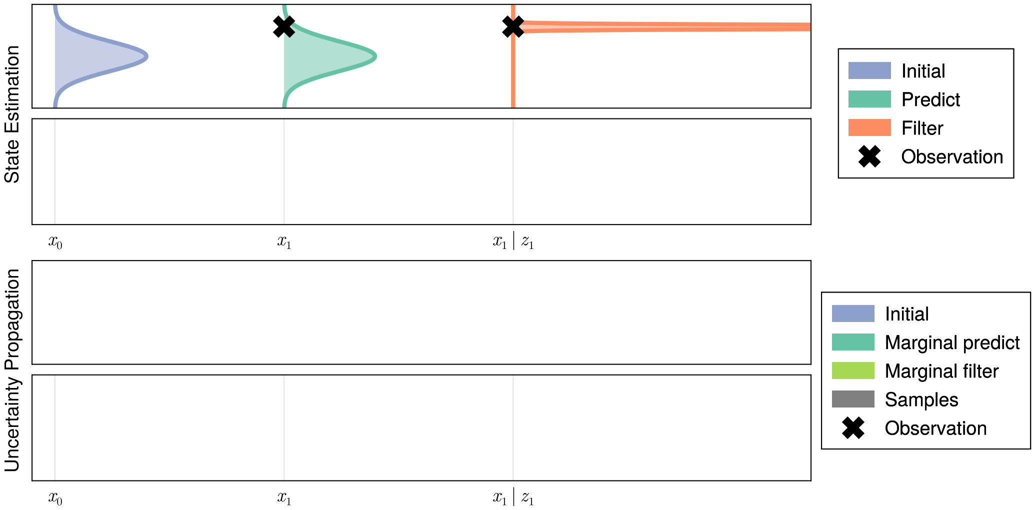
A visual demonstration



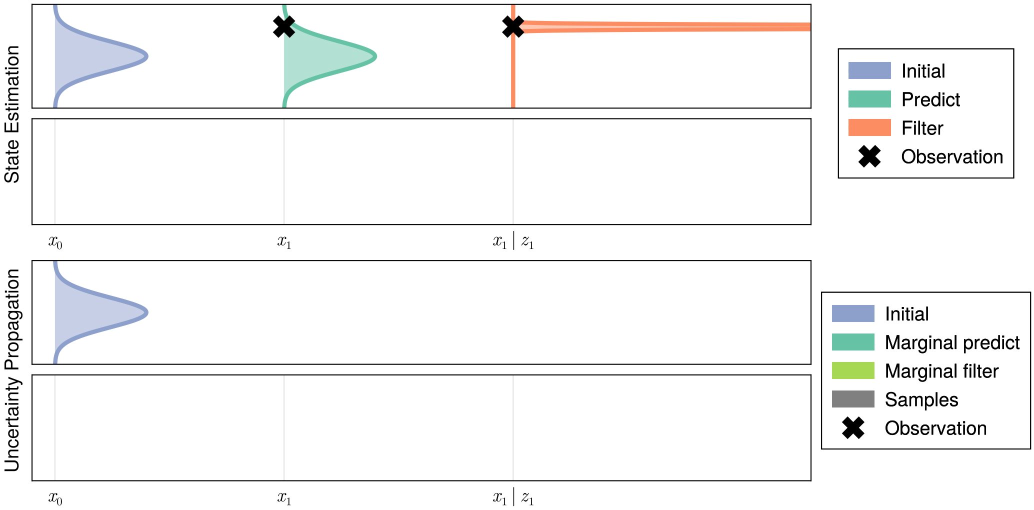
A visual demonstration



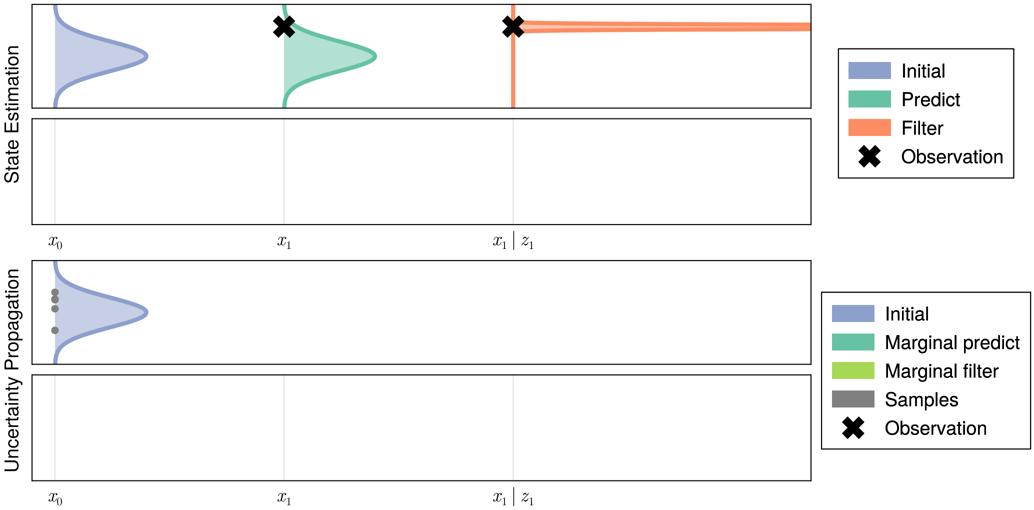
A visual demonstration



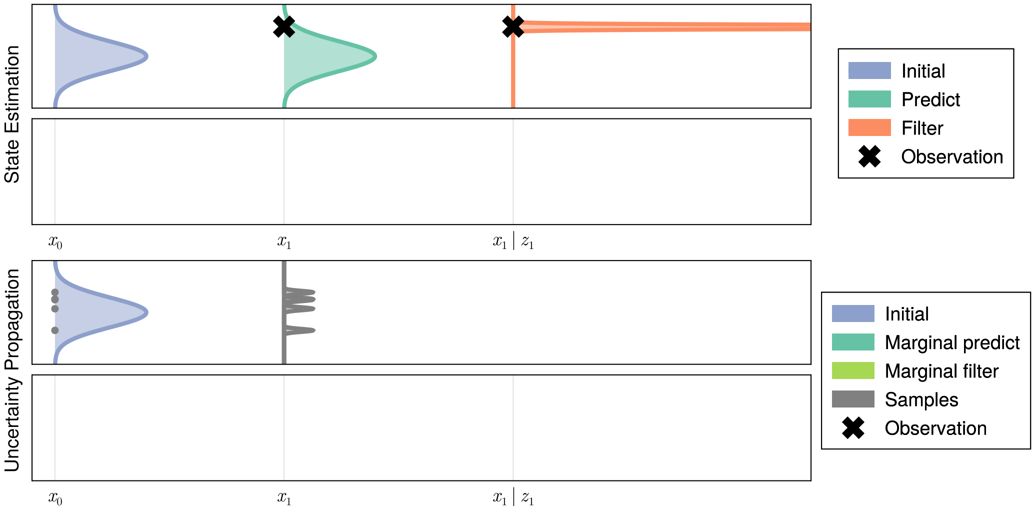
A visual demonstration



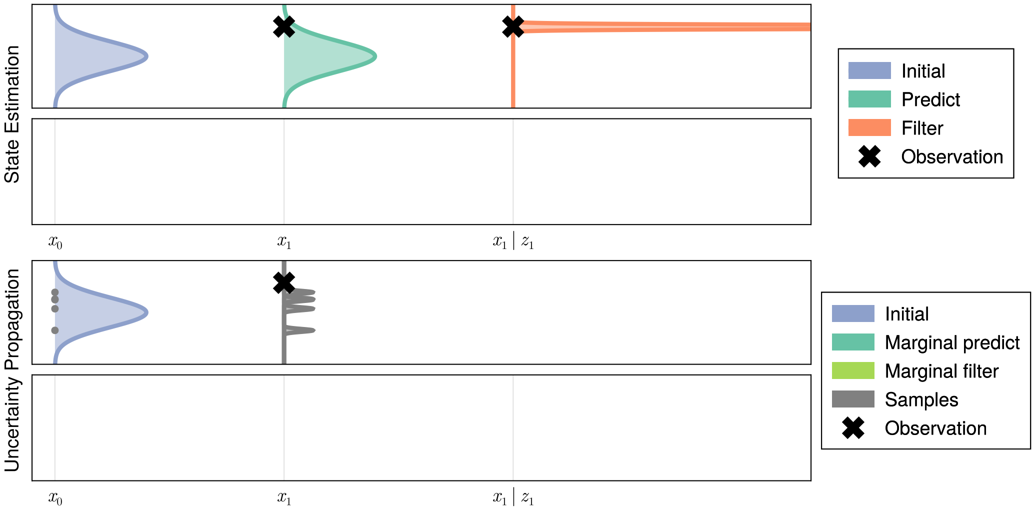
A visual demonstration



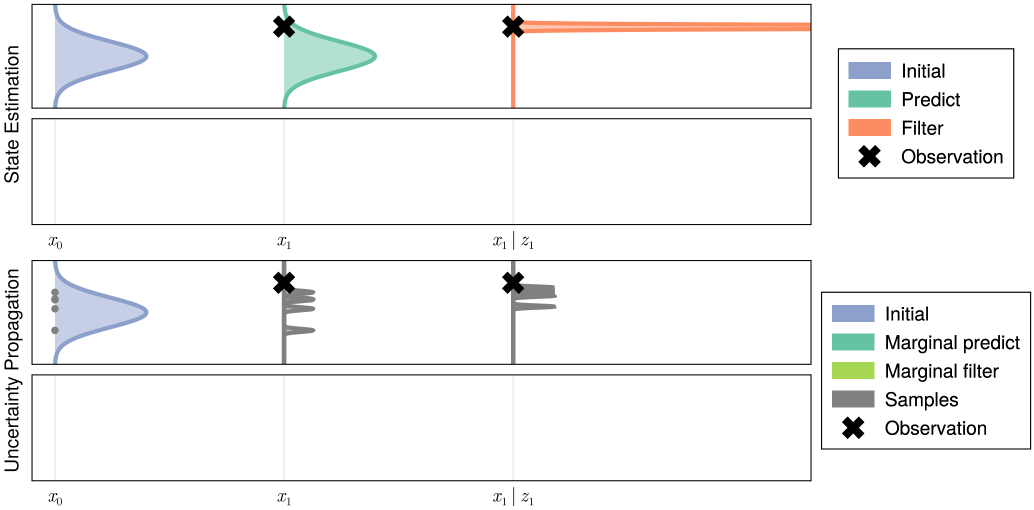
A visual demonstration



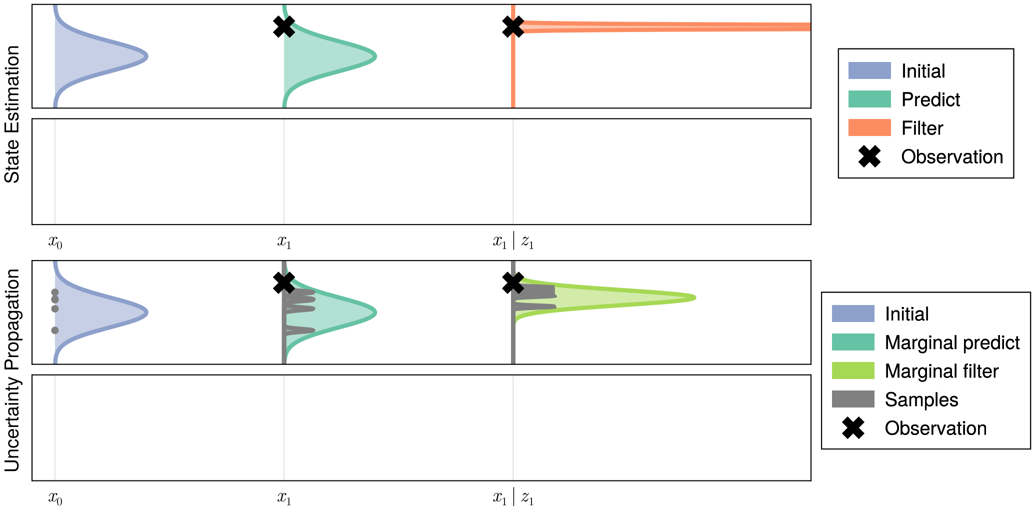
A visual demonstration



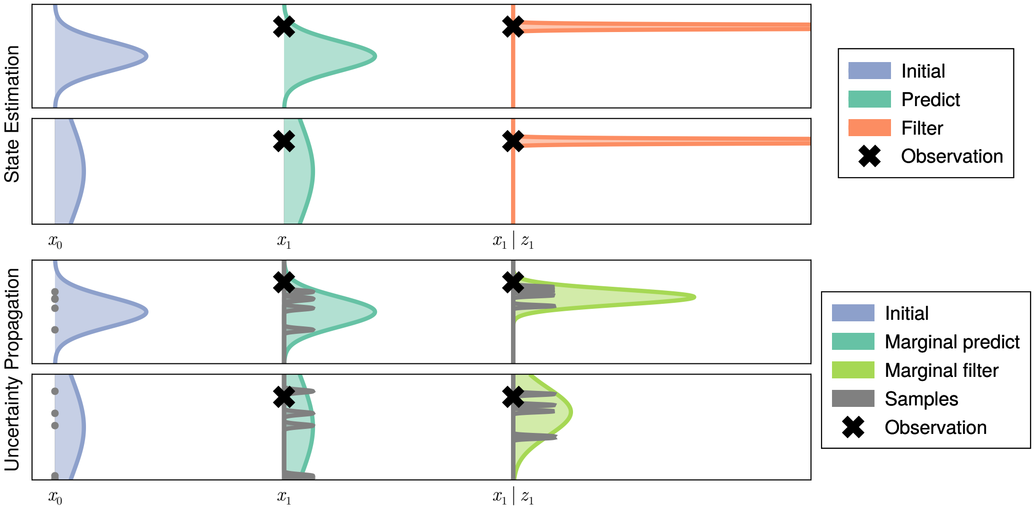
A visual demonstration



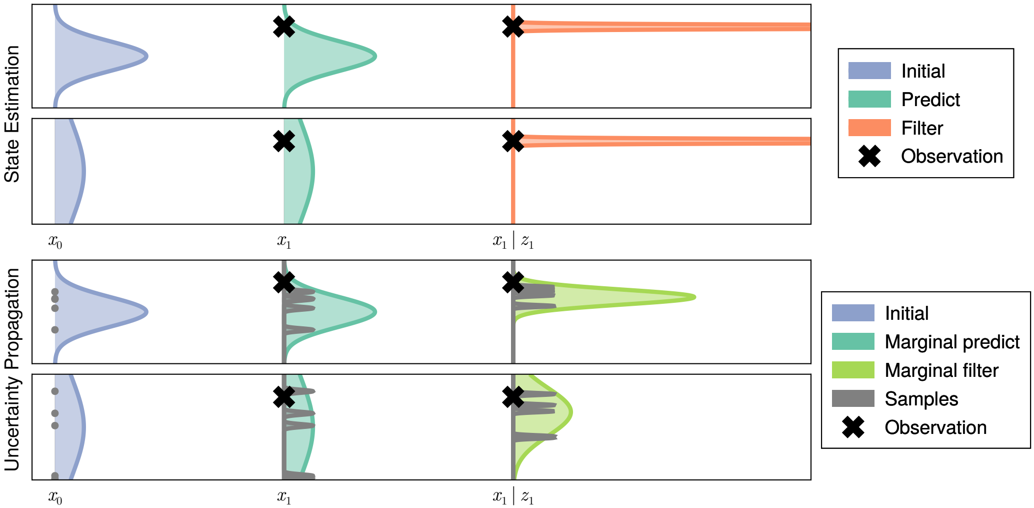
A visual demonstration



A visual demonstration



A visual demonstration



$\Rightarrow$ **Bayesian filters perform *state estimation* and not *uncertainty propagation*!**

Back to ODEs

- **Problem:** Ordinary differential equations with model uncertainty:

$$\begin{aligned} \dot{y}(t) &= f_\theta(y(t), t), \qquad t \in [0, T], \\ y(0) &= c_\theta, \\ \theta &\sim p(\theta). \end{aligned}$$

Uncertain ODEs and how to actually solve them

- **Problem:** Ordinary differential equations with model uncertainty:

$$\begin{aligned} \dot{y}(t) &= f_\theta(y(t), t), \qquad t \in [0, T], \\ y(0) &= c_\theta, \\ \theta &\sim p(\theta). \end{aligned}$$

- **Goal:** Compute the mean and covariance of $y(t)$:

$$\mathbb{E}[g(y_\theta(t))]_{p(\theta)} = \int g(y_\theta(t)) p(\theta) \, \mathrm{d}\theta$$

with $g(y) = y$ and $g(y) = (y - \mathbb{E}[y])^2$

Uncertain ODEs and how to actually solve them

- **Problem:** Ordinary differential equations with model uncertainty:

$$\begin{aligned} \dot{y}(t) &= f_\theta(y(t), t), \qquad t \in [0, T], \\ y(0) &= c_\theta, \\ \theta &\sim p(\theta). \end{aligned}$$

- **Goal:** Compute the mean and covariance of $y(t)$:

$$\mathbb{E}[g(y_\theta(t))]_{p(\theta)} = \int g(y_\theta(t))p(\theta)\,\mathrm{d}\theta$$

  with $g(y) = y$ and $g(y) = (y - \mathbb{E}[y])^2$

- **Approach:** Approximate unknown $y_\theta(t)$ with probabilistic numerical solution $p_{\text{PN}}(y(t) \mid \theta)$:

$$\mathbb{E}[g(y_\theta(t))]_{p(\theta)} \approx \int\int g(y(t))p_{\text{PN}}(y(t) \mid \theta)p(\theta)\,\mathrm{d}\theta\,\mathrm{d}y(t)$$

Uncertain ODEs and how to actually solve them

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Problem:** Ordinary differential equations with model uncertainty:

$$\begin{aligned} \dot{y}(t) &= f_\theta(y(t), t), \qquad t \in [0, T], \\ y(0) &= c_\theta, \\ \theta &\sim p(\theta). \end{aligned}$$

- **Goal:** Compute the mean and covariance of $y(t)$:

$$\mathbb{E}[g(y_\theta(t))]_{p(\theta)} = \int g(y_\theta(t)) p(\theta) \, \mathrm{d}\theta$$

with $g(y) = y$ and $g(y) = (y - \mathbb{E}[y])^2$

- **Approach:** Approximate unknown $y_\theta(t)$ with probabilistic numerical solution $p_{\text{PN}}(y(t) \mid \theta)$:

$$\mathbb{E}[g(y_\theta(t))]_{p(\theta)} \approx \int g(y(t)) \left(\int p_{\text{PN}}(y(t) \mid \theta) p(\theta) \, \mathrm{d}\theta \right) \mathrm{d}y(t)$$

Approximate uncertainty propagation via numerical quadrature

- **Step 1:** Approximate $\int p_{\text{PN}}(y(t) \mid \theta) p(\theta) \,\mathrm{d}\theta$ with some quadrature scheme:

$$\int p(y(t) \mid \theta) p(\theta) \,\mathrm{d}\theta \approx \sum_{i=1}^{N} w_i \cdot p_{\text{PN}}(y(t) \mid \theta_i),$$

with *nodes* $\theta_i \in \mathbb{R}^e$ and weights $w_i \in \mathbb{R}$.

Approximate uncertainty propagation via numerical quadrature

- **Step 1:** Approximate $\int p_{\text{PN}}(y(t) \mid \theta)p(\theta)\,\mathrm{d}\theta$ with some quadrature scheme:

$$\int p(y(t) \mid \theta)p(\theta)\,\mathrm{d}\theta \approx \sum_{i=1}^{N} w_i \cdot p_{\text{PN}}(y(t) \mid \theta_i),$$

with *nodes* $\theta_i \in \mathbb{R}^e$ and weights $w_i \in \mathbb{R}$. We obtain a Gaussian mixture distribution:

$$\int p(y(t) \mid \theta)p(\theta)\,\mathrm{d}\theta \approx \sum_{i=1}^{N} w_i \cdot \mathcal{N}(\mu_i(t), \Sigma_i(t))$$

Approximate uncertainty propagation via numerical quadrature

- **Step 1:** Approximate $\int p_{\mathrm{PN}}(y(t) \mid \theta)p(\theta)\,\mathrm{d}\theta$ with some quadrature scheme:

$$\int p(y(t) \mid \theta)p(\theta)\,\mathrm{d}\theta \approx \sum_{i=1}^{N} w_i \cdot p_{\mathrm{PN}}(y(t) \mid \theta_i),$$

  with *nodes* $\theta_i \in \mathbb{R}^e$ and weights $w_i \in \mathbb{R}$. We obtain a Gaussian mixture distribution:

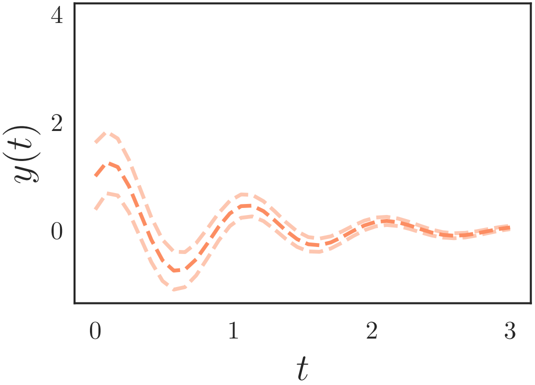
$$\int p(y(t) \mid \theta)p(\theta)\,\mathrm{d}\theta \approx \sum_{i=1}^{N} w_i \cdot \mathcal{N}(\mu_i(t), \Sigma_i(t))$$

- **Step 2:** Compute the expectation and covariance of the Gaussian mixture:

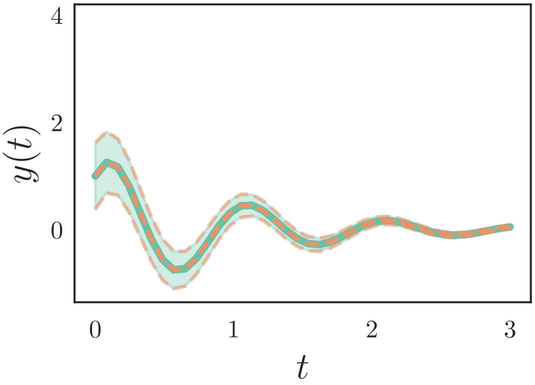
$$\mathbb{E}[y(t)]_{p(y(t))} = \sum_{i=1}^{N} w_i \mu_i(t),$$

$$\mathbb{V}[y(t)]_{p(y(t))} = \sum_{i=1}^{N} w_i \left[\Sigma_i(t) + (\mu_i(t) - \bar{\mu}(t))\,(\mu_i(t) - \bar{\mu}(t))^{\intercal}\right],$$

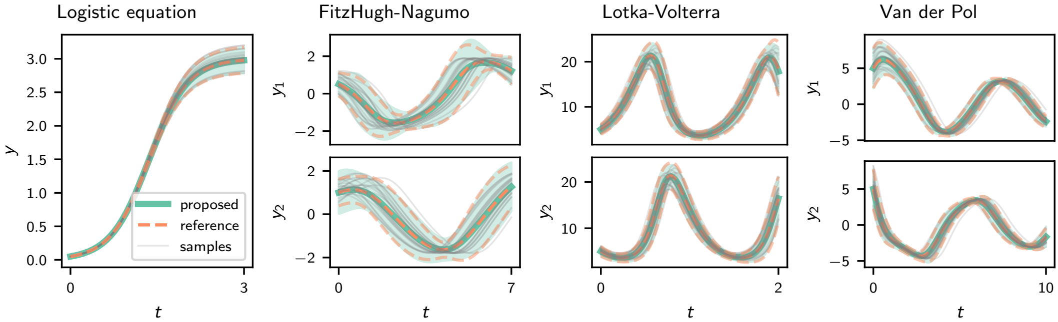
Applying the this algorithm to the github example



Applying the this algorithm to the github example



Examples on more linear and nonlinear ODEs



Applying this to ODE filters vs. non-PN methods

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

$$\mathbb{E}[y(t)]_{p(y(t))} = \sum_{i=1}^{N} w_i \mu_i(t)$$

$$\mathbb{V}[y(t)]_{p(y(t))} = \sum_{i=1}^{N} w_i \Sigma_i(t) + \sum_{i=1}^{N} w_i \left(\mu_i(t) - \bar{\mu}(t)\right) \left(\mu_i(t) - \bar{\mu}(t)\right)^\intercal$$

Applying this to ODE filters vs. non-PN methods

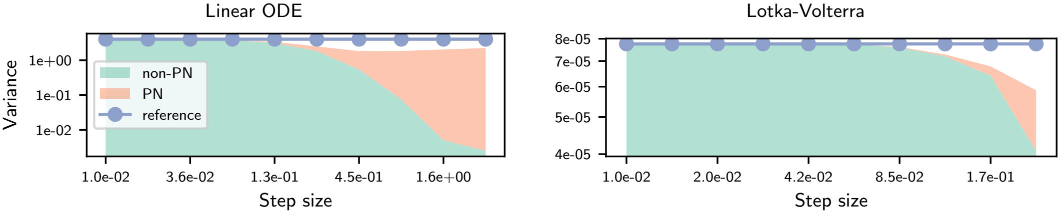
$$\mathbb{E}[y(t)]_{p(y(t))} = \sum_{i=1}^{N} w_i \mu_i(t)$$

$$\mathbb{V}[y(t)]_{p(y(t))} = \underbrace{\sum_{i=1}^{N} w_i \Sigma_i(t)}_{\text{PN}} + \underbrace{\sum_{i=1}^{N} w_i \left(\mu_i(t) - \bar{\mu}(t)\right)\left(\mu_i(t) - \bar{\mu}(t)\right)^\intercal}_{\text{non-PN}}$$

Applying this to ODE filters vs. non-PN methods

$$\mathbb{E}[y(t)]_{p(y(t))} = \sum_{i=1}^{N} w_i \mu_i(t)$$

$$\mathbb{V}[y(t)]_{p(y(t))} = \underbrace{\sum_{i=1}^{N} w_i \Sigma_i(t)}_{\text{PN}} + \underbrace{\sum_{i=1}^{N} w_i \left(\mu_i(t) - \bar{\mu}(t)\right)\left(\mu_i(t) - \bar{\mu}(t)\right)^\intercal}_{\text{non-PN}}$$



Conclusion

Concluding remarks:

Concluding remarks:

- ODE filters seem like they can do uncertainty propagation out-of-the-box, *but they can't*!

Concluding remarks:

- ODE filters seem like they can do uncertainty propagation out-of-the-box, *but they can't*!
- More generally, *just because an algorithm operates on probability distributions does not imply it computes the right quantity*! (e.g. marginalization vs. inference)

Concluding remarks:

- ODE filters seem like they can do uncertainty propagation out-of-the-box, *but they can't*!
- More generally, *just because an algorithm operates on probability distributions does not imply it computes the right quantity*! (e.g. marginalization vs. inference)
- A simple ad-hoc solution: Marginalize approximately via sampling / numerical quadrature.

Concluding remarks:

- ODE filters seem like they can do uncertainty propagation out-of-the-box, *but they can't*!
- More generally, *just because an algorithm operates on probability distributions does not imply it computes the right quantity*! (e.g. marginalization vs. inference)
- A simple ad-hoc solution: Marginalize approximately via sampling / numerical quadrature.
- **Open:** How to propagate uncertainty in ODEs with PN for both time and space discretization?

Concluding remarks:

- ODE filters seem like they can do uncertainty propagation out-of-the-box, *but they can't*!
- More generally, *just because an algorithm operates on probability distributions does not imply it computes the right quantity*! (e.g. marginalization vs. inference)
- A simple ad-hoc solution: Marginalize approximately via sampling / numerical quadrature.
- **Open:** How to propagate uncertainty in ODEs with PN for both time and space discretization?

Hennig, Osborne, Girolami. *Probabilistic numerics and uncertainty in computations*. 2015.

> Many open questions remain for this exciting field. In the long run, probabilistic formulations may allow the propagation of uncertainty through pipelines of computation, and thus the active control of computational effort through hierarchical, modular computations.

Concluding remarks:

- ODE filters seem like they can do uncertainty propagation out-of-the-box, *but they can't*!
- More generally, *just because an algorithm operates on probability distributions does not imply it computes the right quantity*! (e.g. marginalization vs. inference)
- A simple ad-hoc solution: Marginalize approximately via sampling / numerical quadrature.
- **Open:** How to propagate uncertainty in ODEs with PN for both time and space discretization?

Hennig, Osborne, Girolami. *Probabilistic numerics and uncertainty in computations*. 2015.

> Many open questions remain for this exciting field. In the long run, probabilistic formulations may allow the propagation of uncertainty through pipelines of computation, and thus the active control of computational effort through hierarchical, modular computations.

Thanks!