Probabilistic Numerical Solvers for Ordinary Differential Equations

SCML 2024

Nathanael Bosch

22. March 2024

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

imprs-is

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

Background

- Ordinary differential equations and how to solve them

Background

- Ordinary differential equations and how to solve them

Central statement: **ODE solving is state estimation**

- "ODE filters": **How to solve ODEs with extended Kalman filtering and smoothing**

Background

- Ordinary differential equations and how to solve them

Central statement: **ODE solving is state estimation**

- "ODE filters": **How to solve ODEs with extended Kalman filtering and smoothing**

Showcasing ODE filters

- Generalizing ODE filters to higher-order ODEs, systems with conserved quantities, BVPs, DAEs, …
- Parameter inference with ODE filters

Background: Ordinary Differential Equations and how to solve them

Background: Ordinary Differential Equations and how to solve them

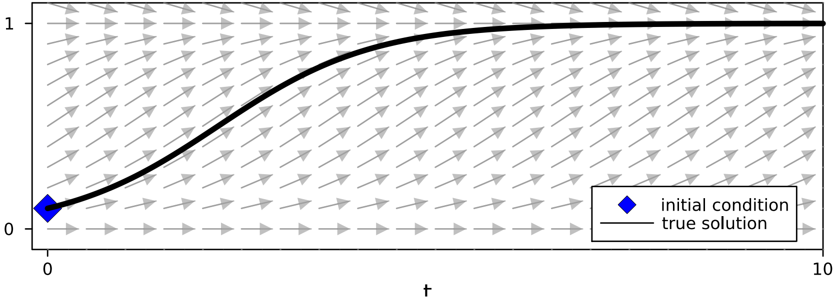
Numerical ODE solvers try to estimate an unknown function by evaluating the vector field

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, and initial value $y(0) = y_0$. Goal: "Find y".

- **Simple example**: Logistic ODE

$$\dot{y}(t) = y(t)\,(1 - y(t))\,, \qquad t \in [0, 10], \qquad y(0) = 0.1.$$



Background: Ordinary Differential Equations and how to solve them

Numerical ODE solvers try to estimate an unknown function by evaluating the vector field

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, and initial value $y(0) = y_0$. Goal: "Find y".

Numerical ODE solvers:

- Forward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t), t)$$

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, and initial value $y(0) = y_0$. Goal: "Find y".

Numerical ODE solvers:

- Forward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t), t)$$
- Backward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t+h), t+h)$$

Background: Ordinary Differential Equations and how to solve them

Numerical ODE solvers try to estimate an unknown function by evaluating the vector field

$$\dot{y}(t) = f\left(y(t), t\right)$$

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, and initial value $y(0) = y_0$. Goal: "Find y".

Numerical ODE solvers:

- Forward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t), t)$$
- Backward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t+h), t+h)$$
- Runge–Kutta:
  $$\hat{y}(t+h) = \hat{y}(t) + h \sum_{i=1}^{s} b_i f(\tilde{y}_i, t + c_i h)$$

Numerical ODE solvers try to estimate an unknown function by evaluating the vector field

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, and initial value $y(0) = y_0$. Goal: "Find y".

Numerical ODE solvers:

- Forward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t), t)$$
- Backward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t+h), t+h)$$
- Runge–Kutta:
  $$\hat{y}(t+h) = \hat{y}(t) + h\sum_{i=1}^{s} b_i f(\tilde{y}_i, t + c_i h)$$
- Multistep:
  $$\hat{y}(t+h) = \hat{y}(t) + h\sum_{i=0}^{s-1} b_i f(\hat{y}(t-ih), t-ih)$$

Background: Ordinary Differential Equations and how to solve them

Numerical ODE solvers try to estimate an unknown function by evaluating the vector field

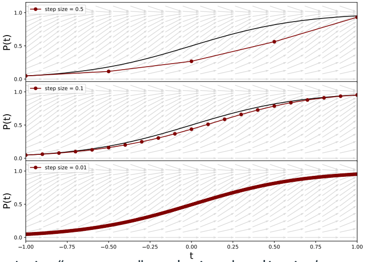
$$\dot{y}(t) = f(y(t), t)$$

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, and initial value $y(0) = y_0$. Goal: "Find y".

Numerical ODE solvers:

- Forward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t), t)$$
- Backward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t+h), t+h)$$
- Runge–Kutta:
  $$\hat{y}(t+h) = \hat{y}(t) + h\sum_{i=1}^{s} b_i f(\tilde{y}_i, t + c_i h)$$
- Multistep:
  $$\hat{y}(t+h) = \hat{y}(t) + h\sum_{i=0}^{s-1} b_i f(\hat{y}(t - ih), t - ih)$$

Forward Euler for different step sizes:



$\Rightarrow$ It is "correct" only in the limit $h \to 0$!

Background: Ordinary Differential Equations and how to solve them

Numerical ODE solvers try to estimate an unknown function by evaluating the vector field

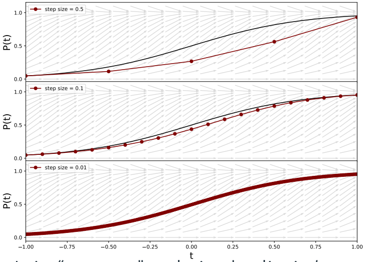
$$\dot{y}(t) = f(y(t), t)$$

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, and initial value $y(0) = y_0$. Goal: "Find y".

Numerical ODE solvers:

- Forward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t), t)$$
- Backward Euler:
  $$\hat{y}(t+h) = \hat{y}(t) + hf(\hat{y}(t+h), t+h)$$
- Runge–Kutta:
  $$\hat{y}(t+h) = \hat{y}(t) + h\sum_{i=1}^{s} b_i f(\tilde{y}_i, t + c_i h)$$
- Multistep:
  $$\hat{y}(t+h) = \hat{y}(t) + h\sum_{i=0}^{s-1} b_i f(\hat{y}(t-ih), t-ih)$$

Forward Euler for different step sizes:



$\Rightarrow$ It is "correct" only in the limit $h \to 0$!

Numerical ODE solvers **estimate** $y(t)$ *by evaluating f on a discrete set of points.*

Probabilistic numerical ODE solvers

or “How to treat ODE solving as a Bayesian state estimation problem”

Probabilistic numerical ODE solvers

Bayes' theorem to the rescue

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

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

Probabilistic formulation of an ODE solver:

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

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

Probabilistic formulation of an ODE solver:

- Prior: $y \sim \mathcal{GP}$

Probabilistic numerical ODE solvers

Bayes' theorem to the rescue

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

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

Probabilistic formulation of an ODE solver:

- Prior: $y \sim \mathcal{GP}$
- Likelihood / data:
  - Initial data: $y(0) = y_0$
  - ODE data: $\dot{y}(t_i) = f(y(t_i), t_i)$, for some $\{t_j\}_{j=1}^N \subset [0, T]$

Probabilistic numerical ODE solvers

Bayes' theorem to the rescue

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

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

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

Probabilistic formulation of an ODE solver:

- **Prior:** $y \sim \mathcal{GP}$
- **Likelihood / data:**
  - Initial data: $y(0) = y_0$
  - ODE data: $\dot{y}(t_i) = f(y(t_i), t_i)$, for some $\{t_j\}_{j=1}^N \subset [0, T]$
- **Inference:** Bayes' rule

Prior: Gauss–Markov process priors

Gauss–Markov processes make GPs go fast

See also: *Särkkä & Solin, "Applied Stochastic Differential Equations", 2013*

- **Continuous Gauss–Markov process prior:**
  $y(t)$ defined as the output of a *linear time-invariant (LTI) stochastic differential equation (SDE)*:

$$
\begin{aligned}
x(0) &\sim \mathcal{N}(\mu_0^-, \Sigma_0^-), \\
\mathrm{d}x(t) &= Fx(t)\mathrm{d}t + \sigma \Gamma \mathrm{d}w(t), \\
y^{(m)}(t) &= E_m x(t), \quad m = 1, \dots, \nu.
\end{aligned}
$$

  $x(t)$ is the *state-space representation* of $y(t)$.
  Examples: Integrated Wiener process, Integrated Ornstein–Uhlenbeck process, Matérn process.

Prior: Gauss–Markov process priors

Gauss–Markov processes make GPs go fast

See also: *Särkkä & Solin, "Applied Stochastic Differential Equations", 2013*

- **Continuous Gauss–Markov process prior:**
  $y(t)$ defined as the output of a *linear time-invariant (LTI) stochastic differential equation (SDE)*:

$$
\begin{aligned}
x(0) &\sim \mathcal{N}(\mu_0^-, \Sigma_0^-), \\
\mathrm{d}x(t) &= Fx(t)\mathrm{d}t + \sigma\Gamma \mathrm{d}w(t), \\
y^{(m)}(t) &= E_m x(t), \quad m = 1, \dots, \nu.
\end{aligned}
$$

  $x(t)$ is the *state-space representation* of $y(t)$.
  Examples: Integrated Wiener process, Integrated Ornstein–Uhlenbeck process, Matérn process.
- **Discrete transition densities:** $x(t)$ can be described in discrete time as

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

  with

$$
A(h) = \exp\left(Fh\right), \qquad Q(h) = \int_0^h A(h-\tau)\Gamma\Gamma^\top A(h-\tau)^\top \tau.
$$

Prior: Gauss–Markov process priors

Gauss–Markov processes make GPs go fast

See also: *Särkkä & Solin, "Applied Stochastic Differential Equations", 2013*

- **Continuous Gauss–Markov process prior:**
  $y(t)$ defined as the output of a *linear time-invariant (LTI) stochastic differential equation (SDE)*:

$$
\begin{aligned}
x(0) &\sim \mathcal{N}(\mu_0^-, \Sigma_0^-), \\
\mathrm{d}x(t) &= Fx(t)\mathrm{d}t + \sigma\Gamma \mathrm{d}w(t), \\
y^{(m)}(t) &= E_m x(t), \qquad m = 1, \dots, \nu.
\end{aligned}
$$

  $x(t)$ is the *state-space representation* of $y(t)$.
  Examples: **Integrated Wiener process**, Integrated Ornstein–Uhlenbeck process, Matérn process.
- **Discrete transition densities:** $x(t)$ can be described in discrete time as

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

  with

$$
A(h) = \exp\left(Fh\right), \qquad Q(h) = \int_0^h A(h-\tau)\Gamma\Gamma^\top A(h-\tau)^\top \tau.
$$

Prior: The q-times integrated Wiener process

A very convenient prior with closed-form transition densities

- **q-times integrated Wiener process prior**: $y(t) \sim \mathrm{IWP}(q)$, defined with $x(t) := [x^{(0)}(t), x^{(1)}(t), \dots, x^{(q)}(t)]$ as

$$
\begin{aligned}
x(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \\
\mathrm{d}x^{(i)}(t) &= x^{(i+1)}(t)\mathrm{d}t, \qquad i = 0, \dots, q-1, \\
\mathrm{d}x^{(q)}(t) &= \sigma \mathrm{d}W(t).
\end{aligned}
$$

  Then $x^{(i)} =: E_i x$ models the i-th derivative of y.

Prior: The q-times integrated Wiener process

A very convenient prior with closed-form transition densities

- q-times integrated Wiener process prior: $y(t) \sim \mathrm{IWP}(q)$, defined with $x(t) := [x^{(0)}(t), x^{(1)}(t), \dots, x^{(q)}(t)]$ as

$$
\begin{aligned}
x(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \\
\mathrm{d}x^{(i)}(t) &= x^{(i+1)}(t)\mathrm{d}t, \qquad i = 0, \dots, q-1, \\
\mathrm{d}x^{(q)}(t) &= \sigma \mathrm{d}W(t).
\end{aligned}
$$

  Then $x^{(i)} =: E_i x$ models the i-th derivative of y.

- **Discrete-time transitions**:

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

$$
[A(h)]_{ij} = \mathbb{I}_{i \leq j} \frac{h^{j-i}}{(j-i)!},
$$

$$
[Q(h)]_{ij} = \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!},
$$

for any $i, j = 0, \dots, q$. (one-dimensional case).

Prior: The q-times integrated Wiener process

A very convenient prior with closed-form transition densities

- q-times integrated Wiener process prior: $y(t) \sim \mathrm{IWP}(q)$, defined with $x(t) := [x^{(0)}(t), x^{(1)}(t), \dots, x^{(q)}(t)]$ as

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0),$$
$$\mathrm{d}x^{(i)}(t) = x^{(i+1)}(t)\mathrm{d}t, \qquad i = 0, \dots, q-1,$$
$$\mathrm{d}x^{(q)}(t) = \sigma \mathrm{d}W(t).$$

  Then $x^{(i)} =: E_i x$ models the i-th derivative of y.

- **Discrete-time transitions**:

$$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), \sigma^2 Q(h)\right),$$
$$[A(h)]_{ij} = \mathbb{I}_{i \leq j} \frac{h^{j-i}}{(j-i)!},$$
$$[Q(h)]_{ij} = \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!},$$

  for any $i, j = 0, \dots, q$. (one-dimensional case).

- **Example**: IWP(2)

$$A(h) = \begin{pmatrix} 1 & h & \frac{h^2}{2} \\ 0 & 1 & h \\ 0 & 0 & 1 \end{pmatrix},$$
$$Q(h) = \begin{pmatrix} \frac{h^5}{20} & \frac{h^4}{8} & \frac{h^3}{6} \\ \frac{h^4}{8} & \frac{h^3}{3} & \frac{h^2}{2} \\ \frac{h^3}{6} & \frac{h^2}{2} & h \end{pmatrix}.$$

Prior: The q-times integrated Wiener process

A very convenient prior with closed-form transition densities

- q-times integrated Wiener process prior: $y(t) \sim \mathrm{IWP}(q)$, defined with $x(t) := [x^{(0)}(t), x^{(1)}(t), \dots, x^{(q)}(t)]$ as

$$
\begin{aligned}
x(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \\
dx^{(i)}(t) &= x^{(i+1)}(t)dt, \qquad i = 0, \dots, q-1, \\
dx^{(q)}(t) &= \sigma dW(t).
\end{aligned}
$$

  Then $x^{(i)} =: E_i x$ models the i-th derivative of y.

- **Discrete-time transitions:**

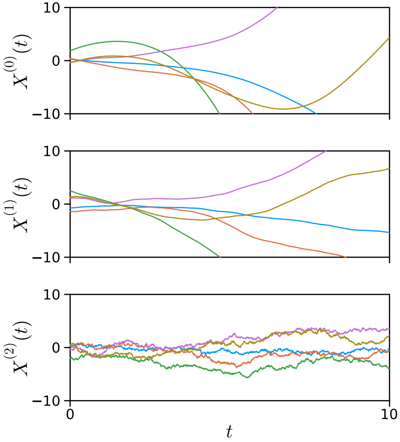
$$
x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), \sigma^2 Q(h)\right),
$$

$$
[A(h)]_{ij} = \mathbb{I}_{i \leq j} \frac{h^{j-i}}{(j-i)!},
$$

$$
[Q(h)]_{ij} = \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!},
$$

  for any $i, j = 0, \dots, q$. (one-dimensional case).

- **Example**: IWP(2)



Prior: The q-times integrated Wiener process

A very convenient prior with closed-form transition densities

- q-times integrated Wiener process prior: $y(t) \sim \text{IWP}(q)$, defined with $x(t) := [x^{(0)}(t), x^{(1)}(t), \dots, x^{(q)}(t)]$ as

$$x(0) \sim \mathcal{N}(\mu_0, \Sigma_0),$$
$$dx^{(i)}(t) = x^{(i+1)}(t)dt, \qquad i = 0, \dots, q-1,$$
$$dx^{(q)}(t) = \sigma dW(t).$$

  Then $x^{(i)} =: E_i x$ models the i-th derivative of y.

- **Discrete-time transitions**:

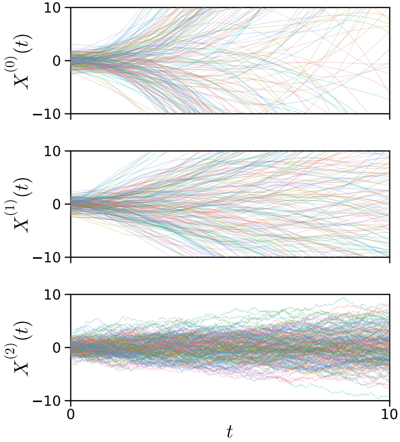
$$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), \sigma^2 Q(h)\right),$$

$$[A(h)]_{ij} = \mathbb{I}_{i \leq j} \frac{h^{j-i}}{(j-i)!},$$

$$[Q(h)]_{ij} = \frac{h^{2q+1-i-j}}{(2q+1-i-j)(q-i)!(q-j)!},$$

  for any $i, j = 0, \dots, q$. (one-dimensional case).

- **Example**: IWP(2)



The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

$$\dot{y}(t) = f\left(y(t), t\right)$$

$$\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) = f\left(E_0 x(t), t\right)$$

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

$$
\begin{aligned}
\dot{y}(t) &= f\left(y(t), t\right) \\
\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) &= f\left(E_0 x(t), t\right) \\
0 &= E_1 x(t) - f\left(E_0 x(t), t\right)
\end{aligned}
$$

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

$$\dot{y}(t) = f\left(y(t), t\right)$$

$$\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) = f\left(E_0 x(t), t\right)$$

$$0 = E_1 x(t) - f\left(E_0 x(t), t\right) =: m(x(t), t).$$

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

$$\overset{\text{using } x(t)}{\Leftrightarrow} \quad E_1 x(t) = f(E_0 x(t), t)$$

$$0 = E_1 x(t) - f(E_0 x(t), t) \; =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \quad t_i \in \mathbb{T} = \{t_i\}_{i=1}^{N} \subset [0, T],$$

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

$$\overset{\text{using } x(t)}{\Leftrightarrow} \quad E_1 x(t) = f(E_0 x(t), t)$$

$$0 = E_1 x(t) - f(E_0 x(t), t) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \quad t_i \in \mathbb{T} = \{t_i\}_{i=1}^{N} \subset [0, T],$$

$$\Leftrightarrow \quad m(x(t_i), t_i) = 0$$

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

$$\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) = f(E_0 x(t), t)$$

$$0 = E_1 x(t) - f(E_0 x(t), t) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \qquad t_i \in \mathbb{T} = \{t_i\}_{i=1}^N \subset [0, T],$$

$$\Leftrightarrow \qquad m(x(t_i), t_i) = 0$$

- This motivates a **measurement model** and **data**:

$$z(t_i) \mid x(t_i) \sim \mathcal{N}\left(m(x(t_i), t_i), R\right)$$

$$z(t_i) \triangleq 0, \qquad i = 1, \dots, N.$$

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

$$\dot{y}(t) = f\left(y(t), t\right)$$

$$\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) = f\left(E_0 x(t), t\right)$$

$$0 = E_1 x(t) - f\left(E_0 x(t), t\right) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \qquad t_i \in \mathbb{T} = \{t_i\}_{i=1}^N \subset [0, T],$$

$$\Leftrightarrow \qquad m(x(t_i), t_i) = 0$$

- This motivates a *noiseless* **measurement model** and **data**:

$$z(t_i) \mid x(t_i) \sim \mathcal{N}\left(m(x(t_i), t_i), 0\right)$$

$$z(t_i) \triangleq 0, \qquad i = 1, \dots, N.$$

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

$$\dot{y}(t) = f\left(y(t), t\right)$$

$$\overset{\text{using } x(t)}{\Leftrightarrow} \quad E_1 x(t) = f\left(E_0 x(t), t\right)$$

$$0 = E_1 x(t) - f\left(E_0 x(t), t\right) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \qquad t_i \in \mathbb{T} = \{t_i\}_{i=1}^N \subset [0, T],$$

$$\Leftrightarrow \qquad m(x(t_i), t_i) = 0$$

- This motivates a *noiseless* **measurement model** and **data**:

$$z(t_i) \mid x(t_i) \sim \delta\left(m(x(t_i), t_i)\right)$$

$$z(t_i) \triangleq 0, \qquad i = 1, \dots, N.$$

(δ is the Dirac distribution)

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

$$\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) = f(E_0 x(t), t)$$

$$0 = E_1 x(t) - f(E_0 x(t), t) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \quad t_i \in \mathbb{T} = \{t_i\}_{i=1}^N \subset [0, T],$$

$$\Leftrightarrow \qquad m(x(t_i), t_i) = 0$$

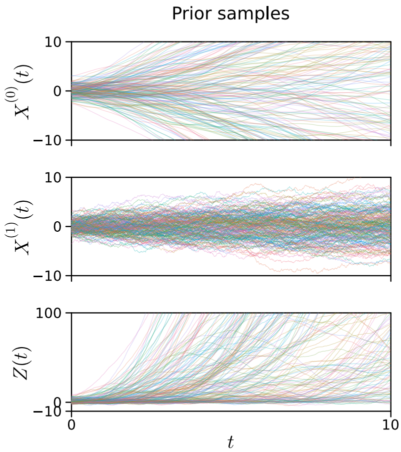
- This motivates a *noiseless* **measurement model** and **data**:

$$z(t_i) \mid x(t_i) \sim \delta(m(x(t_i), t_i))$$

$$z(t_i) \triangleq 0, \qquad i = 1, \dots, N.$$

(δ is the Dirac distribution)

Example: Logistic ODE $\dot{y} = y(1 - y)$



(here: $Z = X^{(1)} - X^{(0)}(1 - X^{(0)})$)

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

$$\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) = f(E_0 x(t), t)$$

$$0 = E_1 x(t) - f(E_0 x(t), t) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \quad t_i \in \mathbb{T} = \{t_i\}_{i=1}^N \subset [0, T],$$

$$\Leftrightarrow \qquad m(x(t_i), t_i) = 0$$

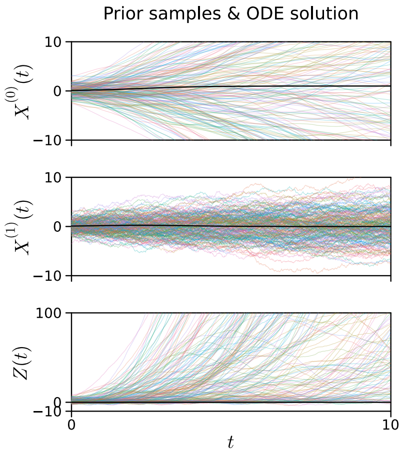
- This motivates a *noiseless* **measurement model** and **data**:

$$z(t_i) \mid x(t_i) \sim \delta(m(x(t_i), t_i))$$

$$z(t_i) \triangleq 0, \qquad i = 1, \dots, N.$$

(δ is the Dirac distribution)

Example: Logistic ODE $\dot{y} = y(1 - y)$



(here: $Z = X^{(1)} - X^{(0)}(1 - X^{(0)})$)

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

$$\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) = f(E_0 x(t), t)$$

$$0 = E_1 x(t) - f(E_0 x(t), t) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \quad t_i \in \mathbb{T} = \{t_i\}_{i=1}^N \subset [0, T],$$

$$\Leftrightarrow \qquad m(x(t_i), t_i) = 0$$

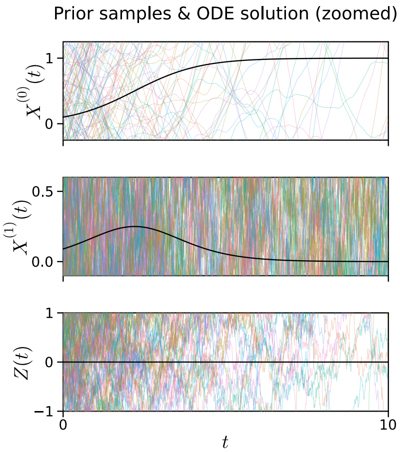
- This motivates a *noiseless* **measurement model** and **data**:

$$z(t_i) \mid x(t_i) \sim \delta(m(x(t_i), t_i))$$

$$z(t_i) \triangleq 0, \qquad i = 1, \dots, N.$$

(δ is the Dirac distribution)

Example: Logistic ODE $\dot{y} = y(1 - y)$



(here: $Z = X^{(1)} - X^{(0)}(1 - X^{(0)})$)

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

$$\overset{\text{using } x(t)}{\Leftrightarrow} \quad E_1 x(t) = f(E_0 x(t), t)$$

$$0 = E_1 x(t) - f(E_0 x(t), t) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \quad t_i \in \mathbb{T} = \{t_i\}_{i=1}^N \subset [0, T],$$

$$\Leftrightarrow \quad m(x(t_i), t_i) = 0$$

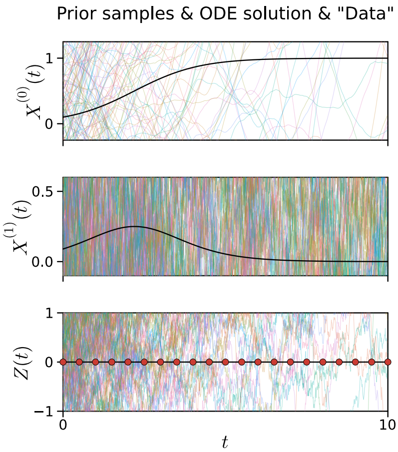
- This motivates a *noiseless* **measurement model** and **data**:

$$z(t_i) \mid x(t_i) \sim \delta(m(x(t_i), t_i))$$

$$z(t_i) \triangleq 0, \qquad i = 1, \ldots, N.$$

(δ is the Dirac distribution)

Example: Logistic ODE $\dot{y} = y(1 - y)$



(here: $Z = X^{(1)} - X^{(0)}(1 - X^{(0)})$)

The likelihood model and the data

The likelihood and data relate the prior to the desired posterior: the numerical ODE solution

- **Ideal goal (intractable):** Want $y(t)$ to satisfy the ODE

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

$$\overset{\text{using } x(t)}{\Leftrightarrow} \qquad E_1 x(t) = f(E_0 x(t), t)$$

$$0 = E_1 x(t) - f(E_0 x(t), t) =: m(x(t), t).$$

- **Easier goal:** Satisfy the ODE *on a discrete time grid*

$$\dot{y}(t_i) = f(y(t_i), t_i), \quad t_i \in \mathbb{T} = \{t_i\}_{i=1}^N \subset [0, T],$$

$$\Leftrightarrow \qquad m(x(t_i), t_i) = 0$$

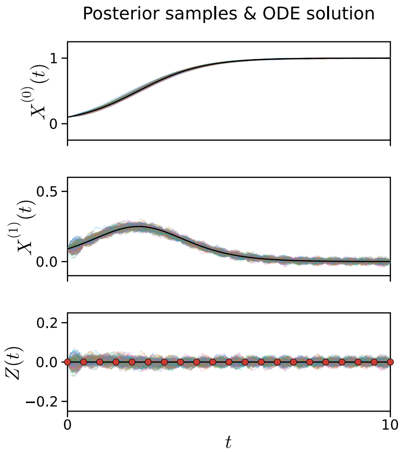
- This motivates a *noiseless* **measurement model** and **data**:

$$z(t_i) \mid x(t_i) \sim \delta(m(x(t_i), t_i))$$

$$z(t_i) \triangleq 0, \qquad i = 1, \dots, N.$$

(δ is the Dirac distribution)

Example: Logistic ODE $\dot{y} = y(1 - y)$



(here: $Z = X^{(1)} - X^{(0)}(1 - X^{(0)})$)

Spoiler: **This is the thing we want!**

Inference: Extended Kalman filtering and smoothing

Bayesian filters and smoothers estimate an unknown state (often continuous) from observations

Given a non-linear Gaussian state-estimation problem:

$$
\begin{aligned}
\text{Initial distribution:} \quad & x_0 \sim \mathcal{N}\left(x_0; \mu_0, \Sigma_0\right), \\
\text{Prior / dynamics:} \quad & x_{i+1} \mid x_i \sim \mathcal{N}\left(x_{i+1}; g(x_i), Q_i\right), \\
\text{Likelihood / measurement:} \quad & z_i \mid x_i \sim \mathcal{N}\left(z_i; m(x_i), R_i\right), \\
\text{Data:} \quad & \mathcal{D} = \{z_i\}_{i=1}^{N}.
\end{aligned}
$$

Inference: Extended Kalman filtering and smoothing

Bayesian filters and smoothers estimate an unknown state (often continuous) from observations

Given a non-linear Gaussian state-estimation problem:

$$\begin{aligned}
\text{Initial distribution:} \quad & x_0 \sim \mathcal{N}\left(x_0; \mu_0, \Sigma_0\right), \\
\text{Prior / dynamics:} \quad & x_{i+1} \mid x_i \sim \mathcal{N}\left(x_{i+1}; g(x_i), Q_i\right), \\
\text{Likelihood / measurement:} \quad & z_i \mid x_i \sim \mathcal{N}\left(z_i; m(x_i), R_i\right), \\
\text{Data:} \quad & \mathcal{D} = \{z_i\}_{i=1}^{N}.
\end{aligned}$$

The extended Kalman filter/smoother (EKF/EKS) recursively computes Gaussian approximations:

$$\begin{aligned}
\text{Predict:} \quad & p(x_i \mid z_{1:i-1}) \approx \mathcal{N}(x_i; \mu_i^P, \Sigma_i^P), \\
\text{Filter:} \quad & p(x_i \mid z_{1:i}) \approx \mathcal{N}(x_i; \mu_i, \Sigma_i), \\
\text{Smooth:} \quad & p(x_i \mid z_{1:N}) \approx \mathcal{N}(x_i; \mu_i^S, \Sigma_i^S), \\
\text{Likelihood:} \quad & p(z_i \mid z_{1:i-1}) \approx \mathcal{N}(z_i; \hat{z}_i, S_i).
\end{aligned}$$

Inference: Extended Kalman filtering and smoothing

Bayesian filters and smoothers estimate an unknown state (often continuous) from observations

Given a non-linear Gaussian state-estimation problem:

Initial distribution: $x_0 \sim \mathcal{N}\left(x_0; \mu_0, \Sigma_0\right),$

Prior / dynamics: $x_{i+1} \mid x_i \sim \mathcal{N}\left(x_{i+1}; g(x_i), Q_i\right),$

Likelihood / measurement: $z_i \mid x_i \sim \mathcal{N}\left(z_i; m(x_i), R_i\right),$

Data: $\mathcal{D} = \{z_i\}_{i=1}^N.$

The extended Kalman filter/smoother (EKF/EKS) recursively computes Gaussian approximations:

Predict: $p(x_i \mid z_{1:i-1}) \approx \mathcal{N}(x_i; \mu_i^P, \Sigma_i^P),$

Filter: $p(x_i \mid z_{1:i}) \approx \mathcal{N}(x_i; \mu_i, \Sigma_i),$

Smooth: $p(x_i \mid z_{1:N}) \approx \mathcal{N}(x_i; \mu_i^S, \Sigma_i^S),$

Likelihood: $p(z_i \mid z_{1:i-1}) \approx \mathcal{N}(z_i; \hat{z}_i, S_i).$

EKF PREDICT

$$\mu_{i+1}^P = g(\mu_i),$$
$$\Sigma_{i+1}^P = J_g(\mu_i)\Sigma_i J_g(\mu_i)^\top + Q_i.$$

EKF UPDATE

$$\hat{z}_i = m(\mu_i^P),$$
$$S_i = J_m(\mu_i^P)\Sigma_i^P J_m(\mu_i^P)^\top + R_i,$$
$$K_i = \Sigma_i^P J_m(\mu_i^P)^\top S_i^{-1},$$
$$\mu_i = \mu_i^P + K_i\left(y_i - \hat{y}_i\right),$$
$$\Sigma_i = \Sigma_i^P - K_i S_i K_i^\top.$$

Similarly SMOOTH.

Probabilistic numerical ODE solvers in code

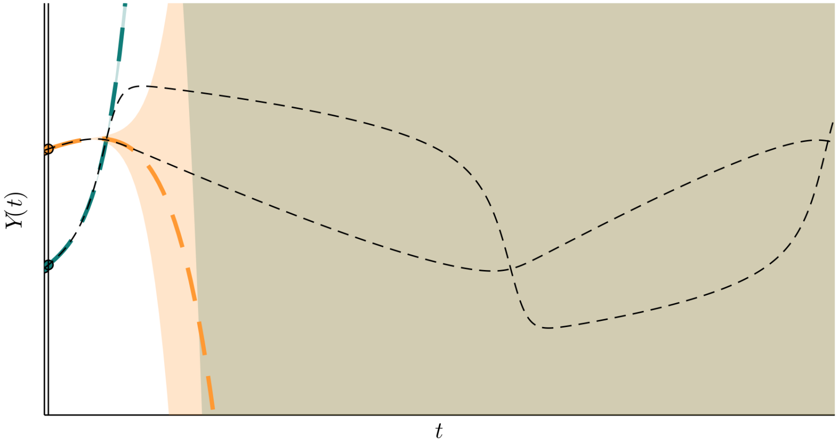
We can solve ODEs with basically just an extended Kalman filter

Algorithm The extended Kalman ODE filter

procedure EXTENDED KALMAN ODE FILTER$((\mu_0^-, \Sigma_0^-), (A, Q), (f, y_0), \{t_i\}_{i=1}^N)$
    $\mu_0, \Sigma_0 \leftarrow \text{KF\_UPDATE}(\mu_0^-, \Sigma_0^-, E_0, 0_{d \times d}, y_0)$ // Initial update to fit the initial value
    for $k \in \{1, \dots, N\}$ **do**
        $h_k \leftarrow t_k - t_{k-1}$ // Step size
        $\mu_k^-, \Sigma_k^- \leftarrow \text{KF\_PREDICT}(\mu_{k-1}, \Sigma_{k-1}, A(h_k), Q(h_k))$ // Kalman filter prediction
        $m_k(x) := E_1 x - f(E_0 x, t_k)$ // Define the non-linear observation model
        $\mu_k, \Sigma_k \leftarrow \text{EKF\_UPDATE}(\mu_k^-, \Sigma_k^-, m_k, 0_{d \times d}, \vec{0}_d)$ // Extended Kalman filter update
    end for
    return $(\mu_k, \Sigma_k)_{k=1}^N$
end procedure

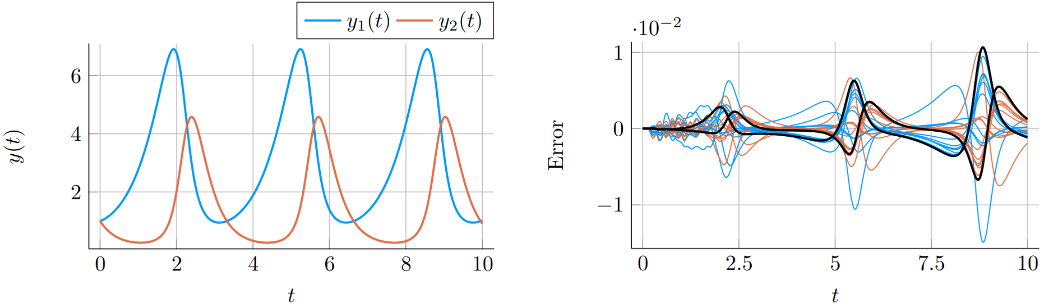
EXTENDED KALMAN ODE SMOOTHER: Just run a RTS smoother after the filter!

Probabilistic numerical ODE solvers in action



Probabilistic numerical ODE solutions

EBERHARD KARLS UNIVERSITÄT TÜBINGEN



The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]

The state of filtering-based probabilistic numerical ODE solvers

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]
  - Connection to multi-step methods in Nordsieck form [Schober et al., 2019]

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]
  - Connection to multi-step methods in Nordsieck form [Schober et al., 2019]
  - Complexity: $\mathcal{O}(d^3)$ for the A-stable semi-implicit method,
    $\mathcal{O}(d)$ for an explicit version with coarser covariances [Krämer et al., 2022]

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]
  - Connection to multi-step methods in Nordsieck form [Schober et al., 2019]
  - Complexity: $\mathcal{O}(d^3)$ for the A-stable semi-implicit method, $\mathcal{O}(d)$ for an explicit version with coarser covariances [Krämer et al., 2022]
  - Step-size adaptation and calibration [Bosch et al., 2021]

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]
  - Connection to multi-step methods in Nordsieck form [Schober et al., 2019]
  - Complexity: $\mathcal{O}(d^3)$ for the A-stable semi-implicit method, $\mathcal{O}(d)$ for an explicit version with coarser covariances [Krämer et al., 2022]
  - Step-size adaptation and calibration [Bosch et al., 2021]
  - Parallel-in-time formulation with $\mathcal{O}(\log(N))$ complexity [Bosch et al., 2023a]

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]
  - Connection to multi-step methods in Nordsieck form [Schober et al., 2019]
  - Complexity: $\mathcal{O}(d^3)$ for the A-stable semi-implicit method,
    $\mathcal{O}(d)$ for an explicit version with coarser covariances [Krämer et al., 2022]
  - Step-size adaptation and calibration [Bosch et al., 2021]
  - Parallel-in-time formulation with $\mathcal{O}(\log(N))$ complexity [Bosch et al., 2023a]
- More related differential equation problems:
  - Higher-order ODEs, DAEs, Hamiltonian systems [Bosch et al., 2022]
  - Boundary value problems (BVPs) [Krämer and Hennig, 2021]
  - Partial differential equations (PDEs) via method of lines [Krämer et al., 2022]

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]
  - Connection to multi-step methods in Nordsieck form [Schober et al., 2019]
  - Complexity: $\mathcal{O}(d^3)$ for the A-stable semi-implicit method,
    $\mathcal{O}(d)$ for an explicit version with coarser covariances [Krämer et al., 2022]
  - Step-size adaptation and calibration [Bosch et al., 2021]
  - Parallel-in-time formulation with $\mathcal{O}(\log(N))$ complexity [Bosch et al., 2023a]
- More related differential equation problems:
  - Higher-order ODEs, DAEs, Hamiltonian systems [Bosch et al., 2022]
  - Boundary value problems (BVPs) [Krämer and Hennig, 2021]
  - Partial differential equations (PDEs) via method of lines [Krämer et al., 2022]
- Inverse problems
  - Probabilistic numerics-based parameter inference in ODEs [Tronarp et al., 2022]
  - Efficient inference of time-varying latent forces [Schmidt et al., 2021]

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]
  - Connection to multi-step methods in Nordsieck form [Schober et al., 2019]
  - Complexity: $\mathcal{O}(d^3)$ for the A-stable semi-implicit method,
    $\mathcal{O}(d)$ for an explicit version with coarser covariances [Krämer et al., 2022]
  - Step-size adaptation and calibration [Bosch et al., 2021]
  - Parallel-in-time formulation with $\mathcal{O}(\log(N))$ complexity [Bosch et al., 2023a]
- More related differential equation problems:
  - Higher-order ODEs, DAEs, Hamiltonian systems [Bosch et al., 2022]
  - Boundary value problems (BVPs) [Krämer and Hennig, 2021]
  - Partial differential equations (PDEs) via method of lines [Krämer et al., 2022]
- Inverse problems
  - Probabilistic numerics-based parameter inference in ODEs [Tronarp et al., 2022]
  - Efficient inference of time-varying latent forces [Schmidt et al., 2021]

Probabilistic Numerics: Computation as Machine Learning

Philipp Hennig, Michael A. Osborne, Hans P. Kersting, 2022

The state of filtering-based probabilistic numerical ODE solvers

- Properties and features:
  - Polynomial convergence rates [Kersting et al., 2020, Tronarp et al., 2021]
  - A-stability [Tronarp et al., 2019]
  - L-stable probabilistic exponential integrators [Bosch et al., 2023b]
  - Connection to multi-step methods in Nordsieck form [Schober et al., 2019]
  - Complexity: $\mathcal{O}(d^3)$ for the A-stable semi-implicit method, $\mathcal{O}(d)$ for an explicit version with coarser covariances [Krämer et al., 2022]
  - Step-size adaptation and calibration [Bosch et al., 2021]
  - Parallel-in-time formulation with $\mathcal{O}(\log(N))$ complexity [Bosch et al., 2023a]
- More related differential equation problems:
  - Higher-order ODEs, DAEs, Hamiltonian systems [Bosch et al., 2022]
  - Boundary value problems (BVPs) [Krämer and Hennig, 2021]
  - Partial differential equations (PDEs) via method of lines [Krämer et al., 2022]
- Inverse problems
  - Probabilistic numerics-based parameter inference in ODEs [Tronarp et al., 2022]
  - Efficient inference of time-varying latent forces [Schmidt et al., 2021]

Probabilistic Numerics: Computation as Machine Learning

Philipp Hennig, Michael A. Osborne, Hans P. Kersting, 2022

Flexible Information Operators

or: *"How to solve other problems than ODEs with essentially the same algorithm as before"*

Flexible Information Operators

or: *"How to solve other problems than ODEs with essentially the same algorithm as before"*
(it's all just likelihood models)

Extending ODE filters to other related differential equation problems

ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Initial value problem with first-order ODE

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

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_1 x(t_i) - f(E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$

Extending ODE filters to other related differential equation problems

ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Initial value problem with second-order ODE

$$\ddot{y}(t) = f(\dot{y}(t), y(t), t), \quad y(0) = y_0, \quad \dot{y}(0) = \dot{y}_0.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_1 x(t_i) - f(E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$

Extending ODE filters to other related differential equation problems

ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Initial value problem with second-order ODE

$$\ddot{y}(t) = f(\dot{y}(t), y(t), t), \quad y(0) = y_0, \quad \dot{y}(0) = \dot{y}_0.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_2 x(t_i) - f(E_1 x(t_i), E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$
Initial derivative likelihood:	$z_1^{\text{init}} \mid x(0) \sim \delta\left(z_1^{\text{init}}; E_1 x(0)\right),$	$z_1^{\text{init}} \triangleq \dot{y}_0$

Extending ODE filters to other related differential equation problems

ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Initial value problem with first-order ODE and conserved quantities

$$\dot{y}(t) = f(y(t), t), \quad y(0) = y_0. \quad g(y(t), \dot{y}(t)) = 0.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_1 x(t_i) - f(E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$

Extending ODE filters to other related differential equation problems

ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Initial value problem with first-order ODE and conserved quantities

$$\dot{y}(t) = f(y(t), t), \quad y(0) = y_0. \quad g(y(t), \dot{y}(t)) = 0.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_1 x(t_i) - f(E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Conservation law likelihood:	$z_i^c(t_i) \mid z(t_i) \sim \delta\left(z_i^c(t_i); g(E_0 x(t), E_1 x(t))\right),$	$z_i^c \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$

Extending ODE filters to other related differential equation problems

ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Initial value problem with second-order ODE and conserved quantities

$$\ddot{y}(t) = f(\dot{y}(t), y(t), t), \quad y(0) = y_0, \quad \dot{y}(0) = \dot{y}_0. \quad g(y(t), \dot{y}(t)) = 0.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_2 x(t_i) - f(E_1 x(t_i), E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Conservation law likelihood:	$z_i^c(t_i) \mid z(t_i) \sim \delta\left(z_i^c(t_i); g(E_0 x(t), E_1 x(t))\right),$	$z_i^c \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$
Initial derivative likelihood:	$z_1^{\text{init}} \mid x(0) \sim \delta\left(z_1^{\text{init}}; E_1 x(0)\right),$	$z_1^{\text{init}} \triangleq \dot{y}_0$

Extending ODE filters to other related differential equation problems

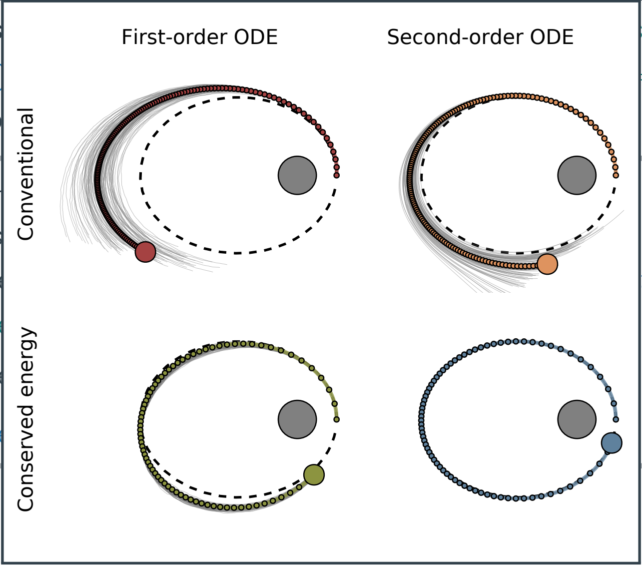
ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems ... erved quantities

$\ddot{y}(t$ ... $= 0.$

This leads to the **pro**

Initial distr		
Prior / dynamics		
ODE lik	$t_i), t_i))$,	$z_i \triangleq 0$
Conservation law lik		$z_i^c \triangleq 0$
Initial value lik		$z^{\text{init}} \triangleq y_0$
Initial derivative lik		$z_1^{\text{init}} \triangleq \dot{y}_0$

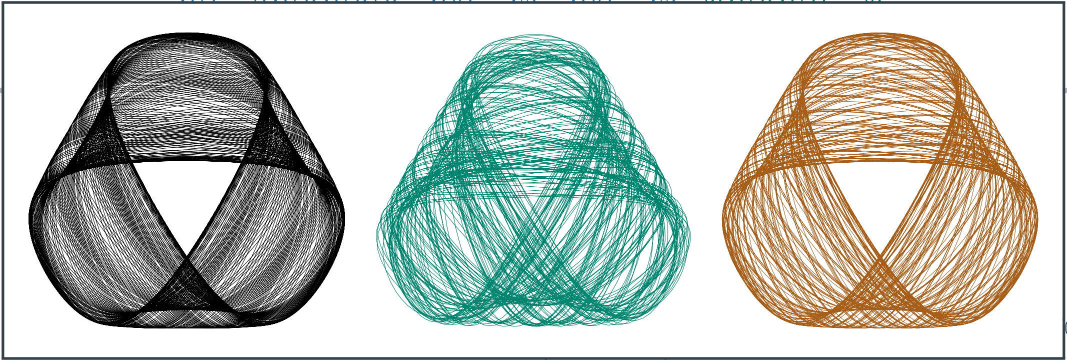


ODE filters can solve much more than the ODEs that we saw so far!

[Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Initial value problem with second-order ODE and conserved quantities

$$\ddot{y}(t) = f(\dot{y}(t), y(t), t), \quad y(0) = y_0, \quad \dot{y}(0) = \dot{y}_0, \quad g(y(t), \dot{y}(t)) = 0.$$



0

Initial derivative likelihood: $z_1^{\text{init}} \mid x(0) \sim \delta\left(z_1^{\text{init}}; E_1 x(0)\right)$, $z_1^{\text{init}} \triangleq \dot{y}_0$

Numerical problems setting: Initial value problem with *differential-algebraic equation* (DAE)

$$0 = F\left(\dot{y}(t), y(t), t\right), \quad y(0) = y_0.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_1 x(t_i) - f(E_0 x(t_i), t_i)\right),$ $\quad z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$ $\quad z^{\text{init}} \triangleq y_0$

Numerical problems setting: Initial value problem with *differential-algebraic equation* (DAE)

$$0 = F\left(\dot{y}(t), y(t), t\right), \quad y(0) = y_0.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
DAE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); F\left(E_1 x(t_i), E_0 x(t_i), t_i\right)\right),$	$z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$

Extending ODE filters to other related differential equation problems

ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Boundary value problem (BVP) with first-order ODE

$$\dot{y}(t) = f(y(t), t), \quad Ly(0) = y_0, \quad Ry(T) = y_T.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_1 x(t_i) - f(E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; E_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$

Numerical problems setting: Boundary value problem (BVP) with first-order ODE

$$\dot{y}(t) = f(y(t), t), \quad Ly(0) = y_0, \quad Ry(T) = y_T.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_1 x(t_i) - f(E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; LE_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$
Boundary value likelihood:	$z_1^R \mid x(T) \sim \delta\left(z_1^R; RE_0 x(T)\right),$	$z_1^{\text{init}} \triangleq y_T$

Extending ODE filters to other related differential equation problems

ODE filters can solve much more than the ODEs that we saw so far! [Bosch et al., 2022, Krämer and Hennig, 2021]

Numerical problems setting: Boundary value problem (BVP) with first-order ODE

$$\dot{y}(t) = f(y(t), t), \quad Ly(0) = y_0, \quad Ry(T) = y_T.$$

This leads to the **probabilistic state estimation problem:**

Initial distribution:	$x(0) \sim \mathcal{N}\left(x(0); \mu_0^-, \Sigma_0^-\right)$	
Prior / dynamics model:	$x(t+h) \mid x(t) \sim \mathcal{N}\left(x(t+h); A(h)x(t), Q(h)\right)$	
ODE likelihood:	$z(t_i) \mid x(t_i) \sim \delta\left(z(t_i); E_1 x(t_i) - f(E_0 x(t_i), t_i)\right),$	$z_i \triangleq 0$
Initial value likelihood:	$z^{\text{init}} \mid x(0) \sim \delta\left(z^{\text{init}}; LE_0 x(0)\right),$	$z^{\text{init}} \triangleq y_0$
Boundary value likelihood:	$z_1^R \mid x(T) \sim \delta\left(z_1^R; RE_0 x(T)\right),$	$z_1^{\text{init}} \triangleq y_T$

The measurement model provides a very flexible way to easily encode desired properties.
But it's all just Bayesian state estimation! $\Rightarrow$ Inference with Bayesian filtering and smoothing.

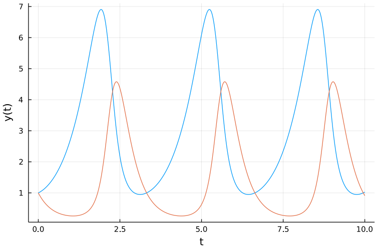
Probabilistic Numerics for ODE Parameter Inference

Using the ODE solution as a "physics-enhanced" prior for regression

Forward Problem

$$\dot{y}_\theta = f_\theta(y_\theta, t) \qquad y_\theta(t_0) = y_0(\theta).$$

solve $\Longrightarrow$



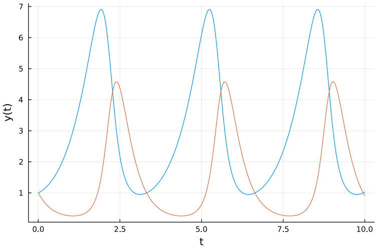
“Forward” and “Inverse” Problems

Going from formula to plot, or from plot to formula

Forward Problem

$$\dot{y}_\theta = f_\theta(y_\theta, t) \qquad y_\theta(t_0) = y_0(\theta).$$

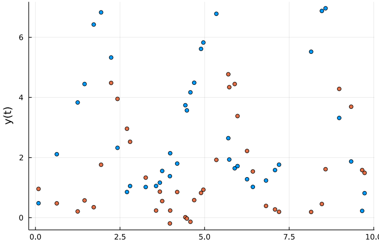
solve $\Longrightarrow$



Inverse Problem

$$p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta) p(\theta)$$

find $\Longleftarrow$



"Forward" and "Inverse" Problems

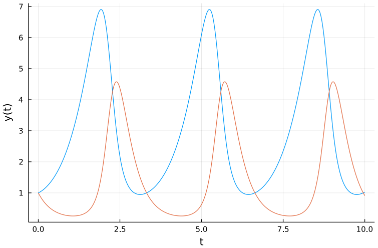
Going from formula to plot, or from plot to formula

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Forward Problem

$$\dot{y}_\theta = f_\theta(y_\theta, t) \qquad y_\theta(t_0) = y_0(\theta).$$

solve ⟹

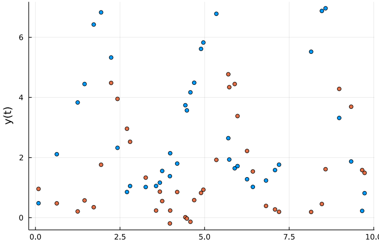


Inverse Problem

$$p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta) p(\theta)$$

find ⟸

Problem: The *marginal likelihood* $p(\mathcal{D} \mid \theta) = \prod_{i=1}^{N} \mathcal{N}\left(u(t_i); y_\theta(t_i), R_\theta\right)$ is intractable (because y_θ is intractable)



Between classic integration and gradient matching

We're doing both: Integrating first, then GP regression

- Classical Numerical Integration
  - (i) Solve the IVP to compute $\hat{y}_\theta(t)$
  - (ii) Approximate the marginal likelihood as $\widehat{\mathcal{M}}(\theta) = \prod_n \mathcal{N}(u(t_n); \hat{y}_\theta(t_n), R_\theta)$
  - (iii) Optimize to get $\hat{\theta} = \arg\max \widehat{\mathcal{M}}(\theta)$

Between classic integration and gradient matching

We're doing both: Integrating first, then GP regression

- Classical Numerical Integration
  - (i) Solve the IVP to compute $\hat{y}_\theta(t)$
  - (ii) Approximate the marginal likelihood as $\widehat{\mathcal{M}}(\theta) = \prod_n \mathcal{N}(u(t_n); \hat{y}_\theta(t_n), R_\theta)$
  - (iii) Optimize to get $\hat{\theta} = \arg\max \widehat{\mathcal{M}}(\theta)$
- Gradient Matching
  - (i) Fit a curve $\hat{y}(t)$ *to the data* $\{u(t_i)\}_{i=1}N$
  - (ii) Estimate θ by minimizing $\dot{\hat{y}}(t) - f_\theta(\hat{y}(t))$

Exists in both classic (splines) or probabilistic versions (GPs)

Between classic integration and gradient matching

We're doing both: Integrating first, then GP regression

- Classical Numerical Integration
  - (i) Solve the IVP to compute $\hat{y}_\theta(t)$
  - (ii) Approximate the marginal likelihood as $\widehat{\mathcal{M}}(\theta) = \prod_n \mathcal{N}(u(t_n); \hat{y}_\theta(t_n), R_\theta)$
  - (iii) Optimize to get $\hat{\theta} = \arg\max \widehat{\mathcal{M}}(\theta)$
- Gradient Matching
  - (i) Fit a curve $\hat{y}(t)$ *to the data* $\{u(t_i)\}_{i=1}N$
  - (ii) Estimate θ by minimizing $\dot{\hat{y}}(t) - f_\theta(\hat{y}(t))$

  Exists in both classic (splines) or probabilistic versions (GPs)
- Probabilistic Numerical Integration

Between classic integration and gradient matching

We're doing both: Integrating first, then GP regression

- Classical Numerical Integration
  - (i) Solve the IVP to compute $\hat{y}_\theta(t)$
  - (ii) Approximate the marginal likelihood as $\widehat{\mathcal{M}}(\theta) = \prod_n \mathcal{N}(u(t_n); \hat{y}_\theta(t_n), R_\theta)$
  - (iii) Optimize to get $\hat{\theta} = \arg\max \widehat{\mathcal{M}}(\theta)$
- Gradient Matching
  - (i) Fit a curve $\hat{y}(t)$ *to the data* $\{u(t_i)\}_{i=1}N$
  - (ii) Estimate θ by minimizing $\dot{\hat{y}}(t) - f_\theta(\hat{y}(t))$

  Exists in both classic (splines) or probabilistic versions (GPs)
- **Probabilistic Numerical Integration**

$$\widehat{\mathcal{M}}_{PN}(\theta, \kappa) = \int \underbrace{\prod_n \mathcal{N}\left(u(t_n); y(t_n), R_\theta\right)}_{\text{Likelihood}} \cdot \underbrace{p_{PN}\left(y(t_{1:N}) \mid \theta, \kappa\right)}_{\text{PN ODE Solution}} \mathrm{d}y(t_{1:N}) \tag{1}$$

Between classic integration and gradient matching

We're doing both: Integrating first, then GP regression

- Classical Numerical Integration
  - (i) Solve the IVP to compute $\hat{y}_\theta(t)$
  - (ii) Approximate the marginal likelihood as $\widehat{\mathcal{M}}(\theta) = \prod_n \mathcal{N}(u(t_n); \hat{y}_\theta(t_n), R_\theta)$
  - (iii) Optimize to get $\hat{\theta} = \arg\max \widehat{\mathcal{M}}(\theta)$
- Gradient Matching
  - (i) Fit a curve $\hat{y}(t)$ *to the data* $\{u(t_i)\}_{i=1}N$
  - (ii) Estimate θ by minimizing $\dot{\hat{y}}(t) - f_\theta(\hat{y}(t))$

  Exists in both classic (splines) or probabilistic versions (GPs)
- **Probabilistic Numerical Integration**

$$\widehat{\mathcal{M}}_{PN}(\theta, \kappa) = \int \underbrace{\prod_n \mathcal{N}\left(u(t_n); y(t_n), R_\theta\right)}_{\text{Likelihood}} \cdot \underbrace{p_{PN}\left(y(t_{1:N}) \mid \theta, \kappa\right)}_{\text{PN ODE Solution}} \mathrm{d}y(t_{1:N}) \tag{1}$$

  - (i) *Probabilistically* solve IVP to compute $p_{\mathrm{PN}}(y(t) \mid \theta, \kappa)$

Between classic integration and gradient matching

We're doing both: Integrating first, then GP regression

- Classical Numerical Integration
  - (i) Solve the IVP to compute $\hat{y}_\theta(t)$
  - (ii) Approximate the marginal likelihood as $\widehat{\mathcal{M}}(\theta) = \prod_n \mathcal{N}(u(t_n); \hat{y}_\theta(t_n), R_\theta)$
  - (iii) Optimize to get $\hat{\theta} = \arg\max \widehat{\mathcal{M}}(\theta)$
- Gradient Matching
  - (i) Fit a curve $\hat{y}(t)$ *to the data* $\{u(t_i)\}_{i=1}N$
  - (ii) Estimate θ by minimizing $\dot{\hat{y}}(t) - f_\theta(\hat{y}(t))$

  Exists in both classic (splines) or probabilistic versions (GPs)
- Probabilistic Numerical Integration

$$\widehat{\mathcal{M}}_{PN}(\theta, \kappa) = \int \underbrace{\prod_n \mathcal{N}\left(u(t_n); y(t_n), R_\theta\right)}_{\text{Likelihood}} \cdot \underbrace{p_{PN}\left(y(t_{1:N}) \mid \theta, \kappa\right)}_{\text{PN ODE Solution}} \mathrm{d}y(t_{1:N}) \tag{1}$$

  - (i) *Probabilistically* solve IVP to compute $p_{\text{PN}}(y(t) \mid \theta, \kappa)$
  - (ii) Perform Kalman filtering on the data, with p_{PN} as a *"physics-enhanced"* prior

Between classic integration and gradient matching

We're doing both: Integrating first, then GP regression

- Classical Numerical Integration
  - (i) Solve the IVP to compute $\hat{y}_\theta(t)$
  - (ii) Approximate the marginal likelihood as $\widehat{\mathcal{M}}(\theta) = \prod_n \mathcal{N}(u(t_n); \hat{y}_\theta(t_n), R_\theta)$
  - (iii) Optimize to get $\hat{\theta} = \arg\max \widehat{\mathcal{M}}(\theta)$
- Gradient Matching
  - (i) Fit a curve $\hat{y}(t)$ *to the data* $\{u(t_i)\}_{i=1}N$
  - (ii) Estimate θ by minimizing $\dot{\hat{y}}(t) - f_\theta(\hat{y}(t))$

  Exists in both classic (splines) or probabilistic versions (GPs)
- Probabilistic Numerical Integration

$$\widehat{\mathcal{M}}_{PN}(\theta, \kappa) = \int \underbrace{\prod_n \mathcal{N}\left(u(t_n); y(t_n), R_\theta\right)}_{\text{Likelihood}} \cdot \underbrace{p_{PN}\left(y(t_{1:N}) \mid \theta, \kappa\right)}_{\text{PN ODE Solution}} \mathrm{d}y(t_{1:N}) \tag{1}$$

  - (i) *Probabilistically* solve IVP to compute $p_{\mathrm{PN}}(y(t) \mid \theta, \kappa)$
  - (ii) Perform Kalman filtering on the data, with p_{PN} as a *"physics-enhanced"* prior
  - (iii) Optimize the approximate marginal likelihood

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

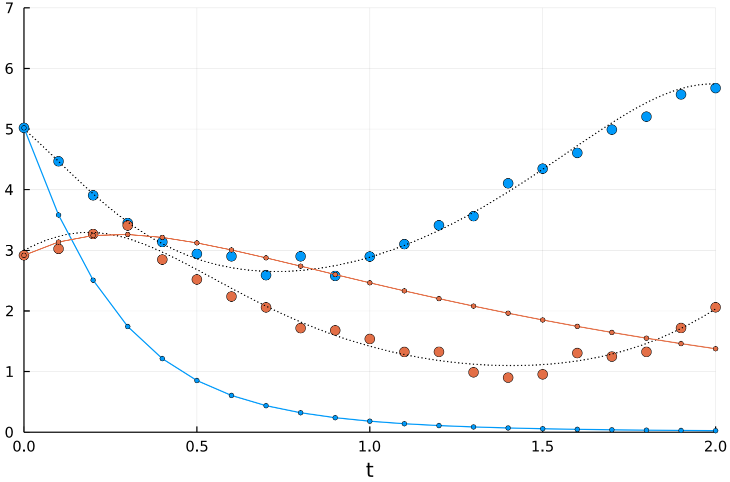


Figure: i=1

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

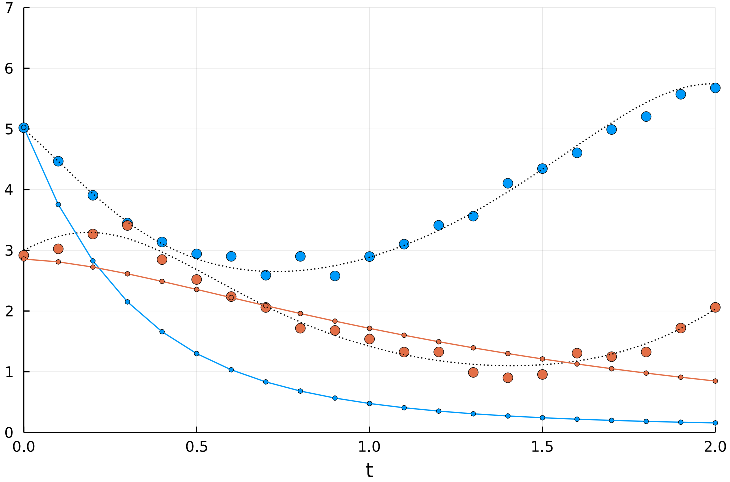


Figure: i=2

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

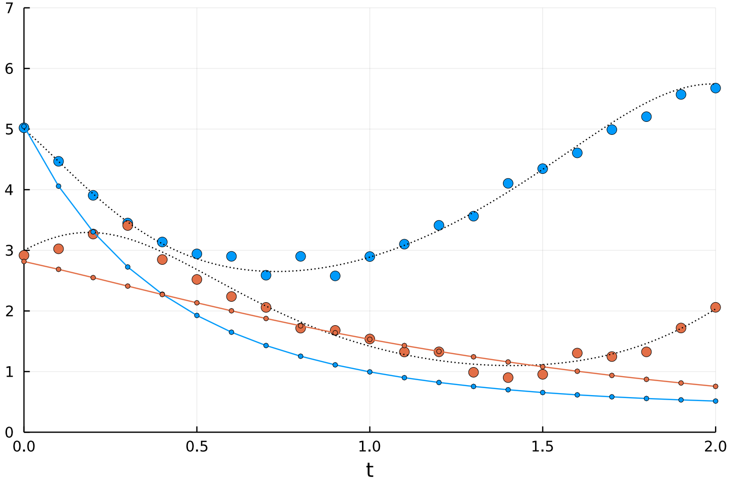


Figure: i=3

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

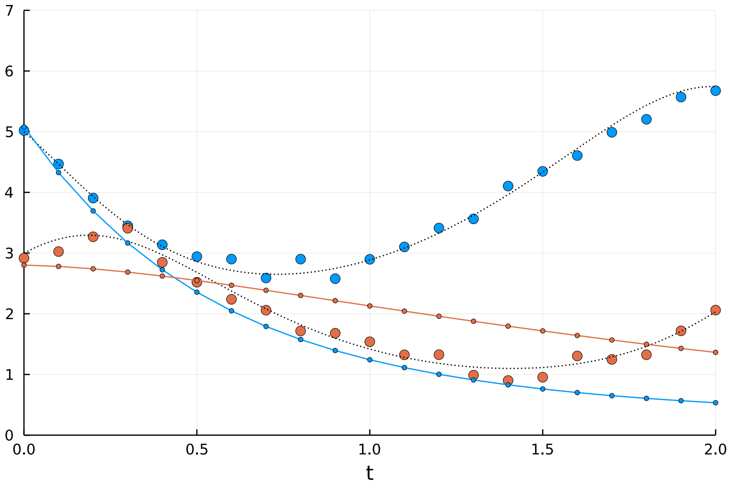


Figure: i=4

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

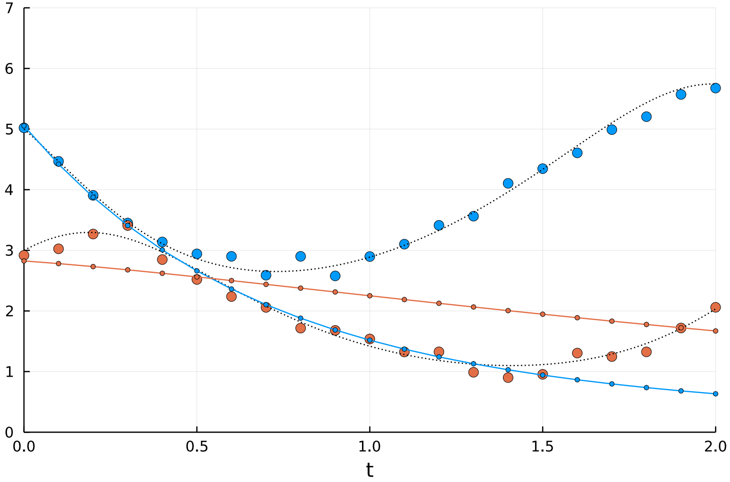


Figure: i=5

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

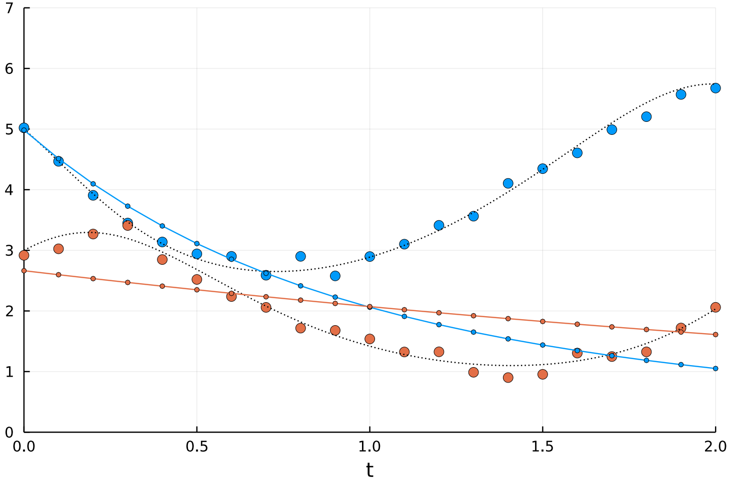


Figure: i=10

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

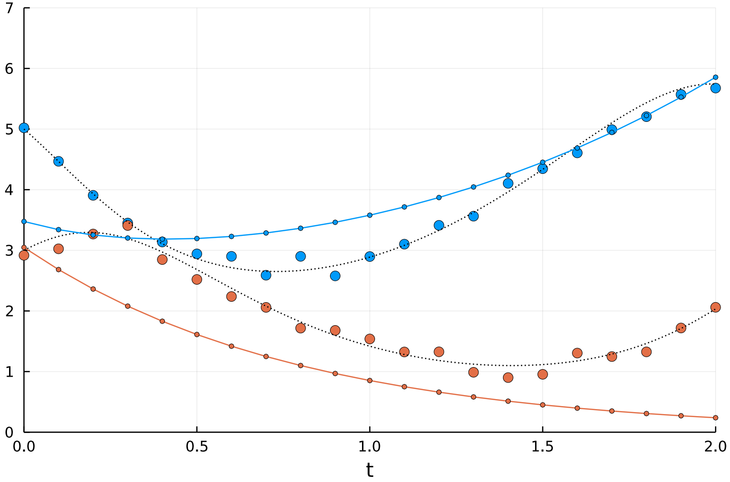


Figure: i=15

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

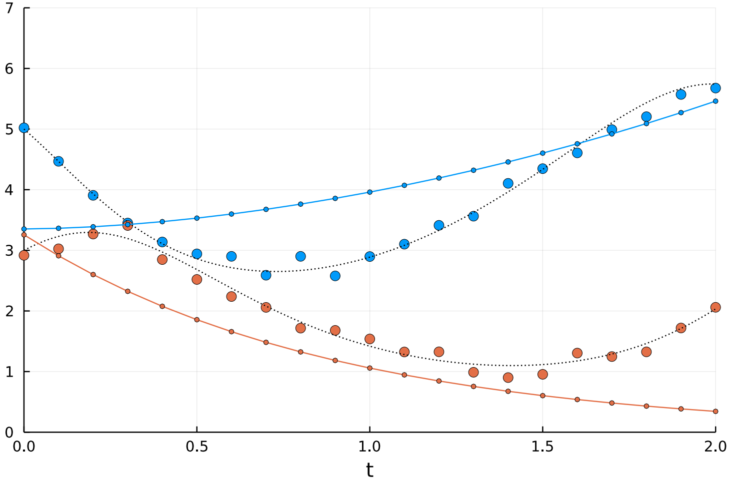


Figure: i=20

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

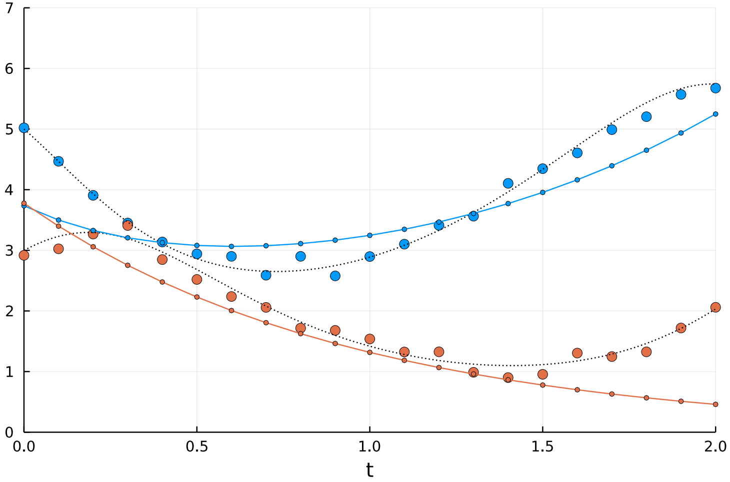


Figure: i=25

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

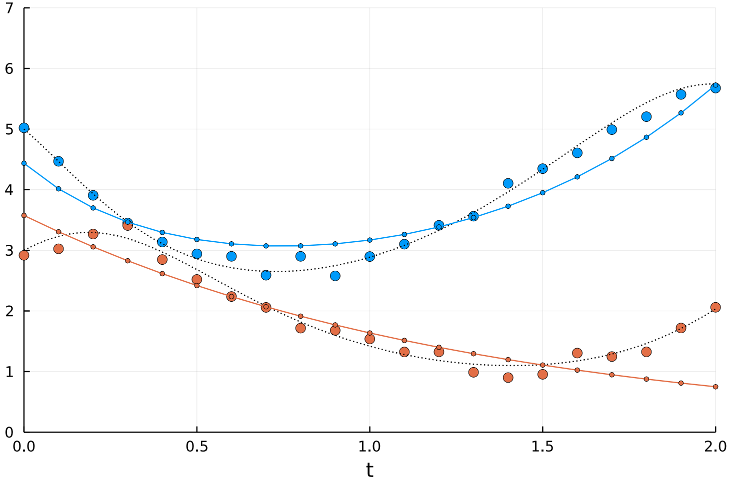


Figure: i=30

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

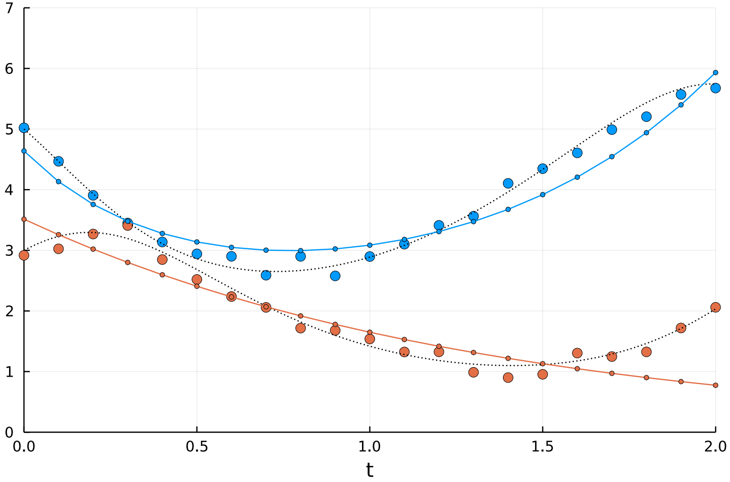


Figure: i=35

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

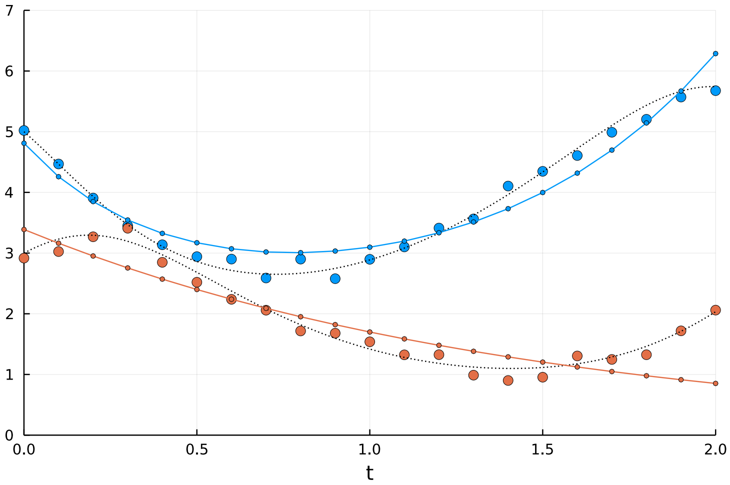


Figure: i=40

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

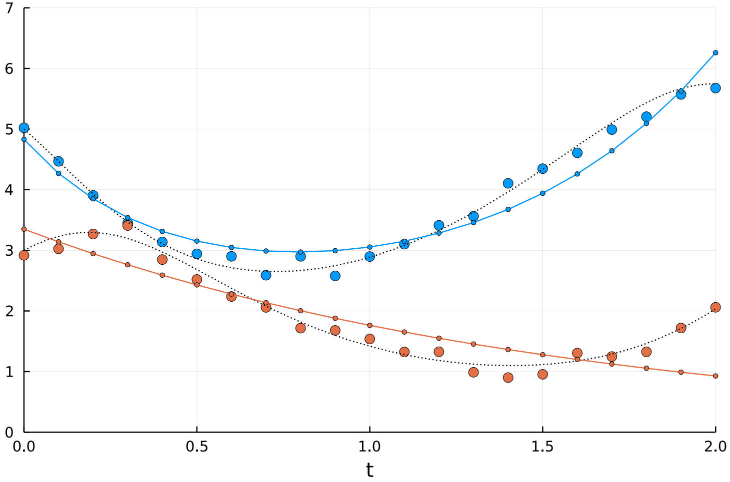


Figure: i=45

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

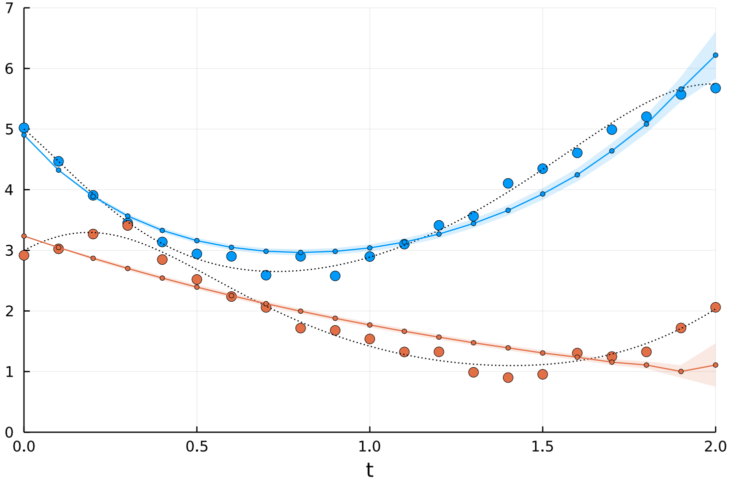


Figure: i=50

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

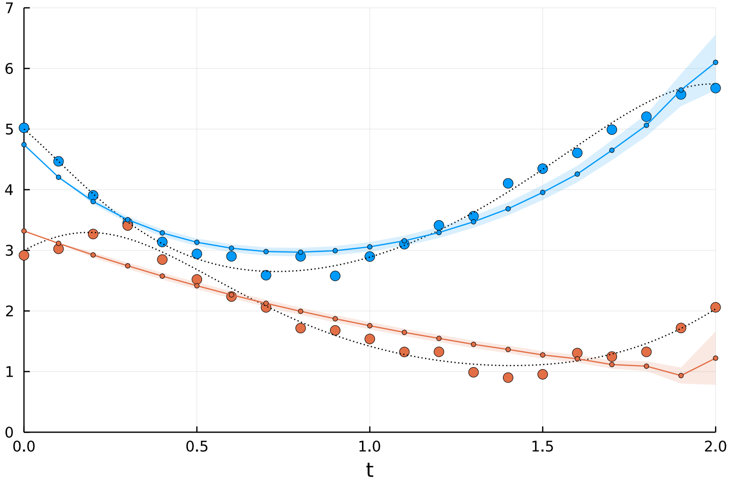


Figure: i=55

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

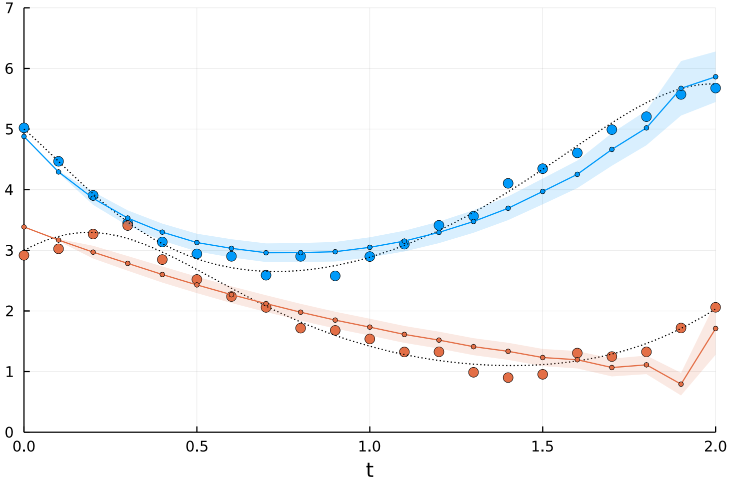


Figure: i=60

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

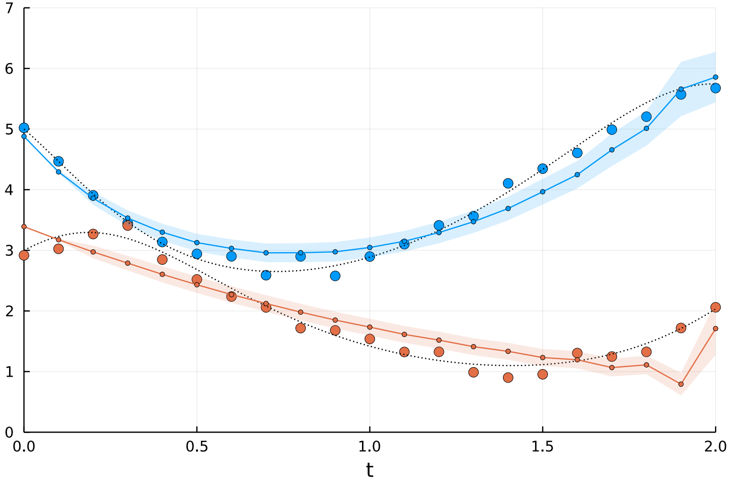


Figure: i=61

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

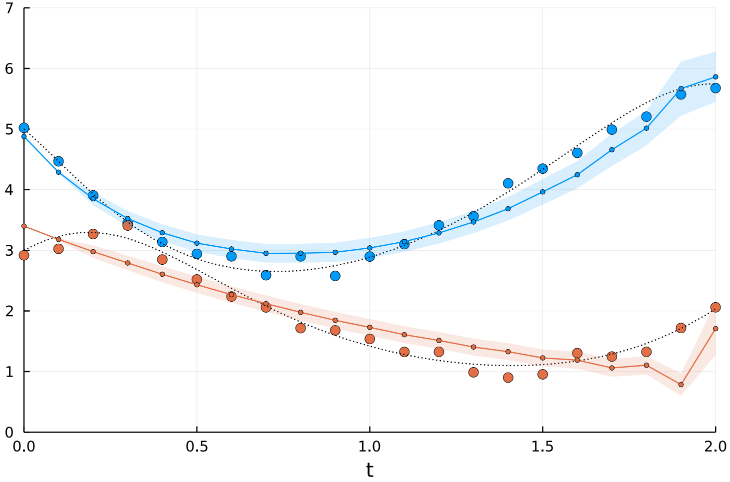


Figure: i=62

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

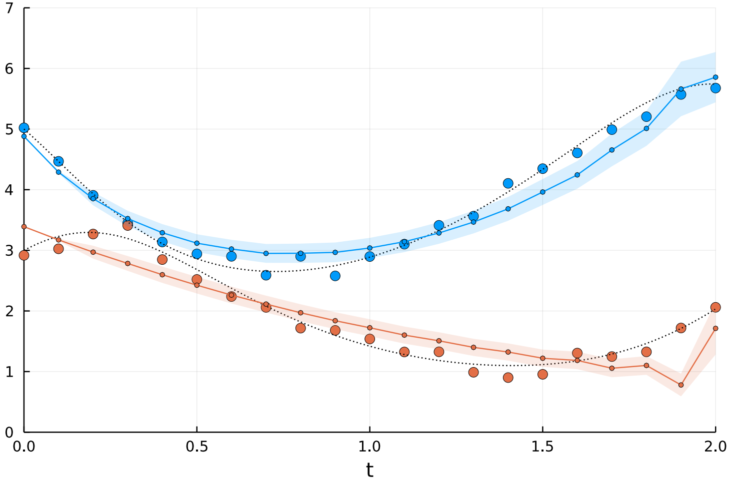


Figure: i=63

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

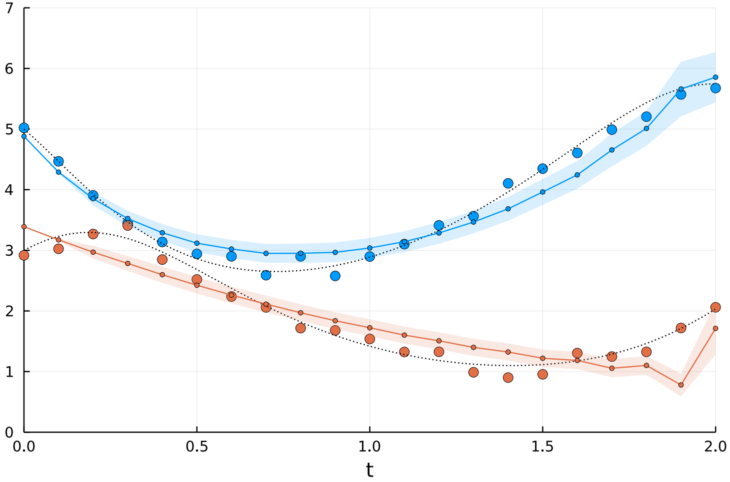


Figure: i=63

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

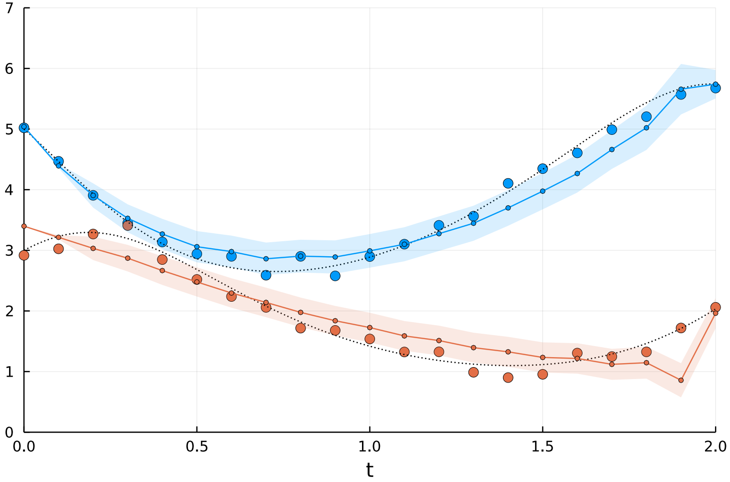


Figure: i=64

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

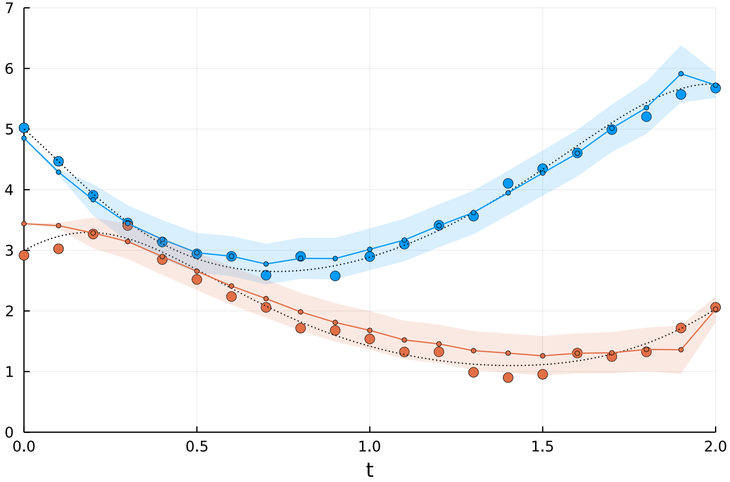


Figure: i=65

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

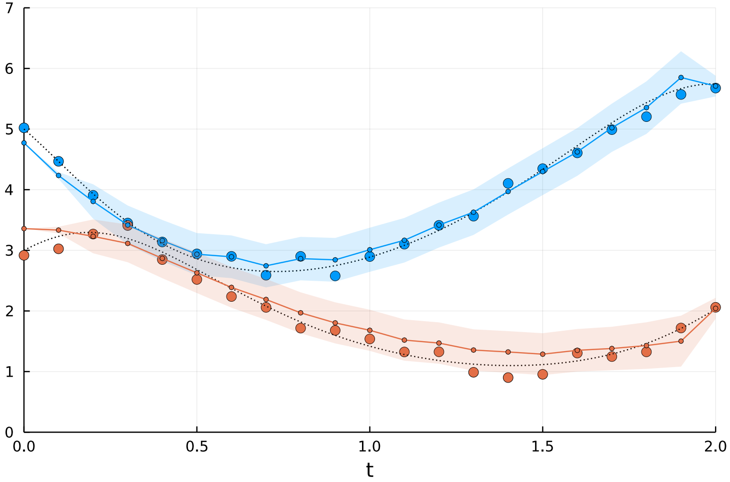


Figure: i=66

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

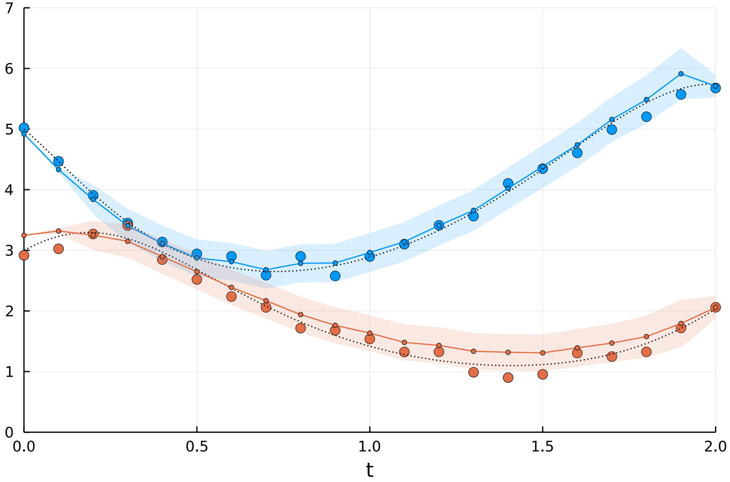


Figure: i=67

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

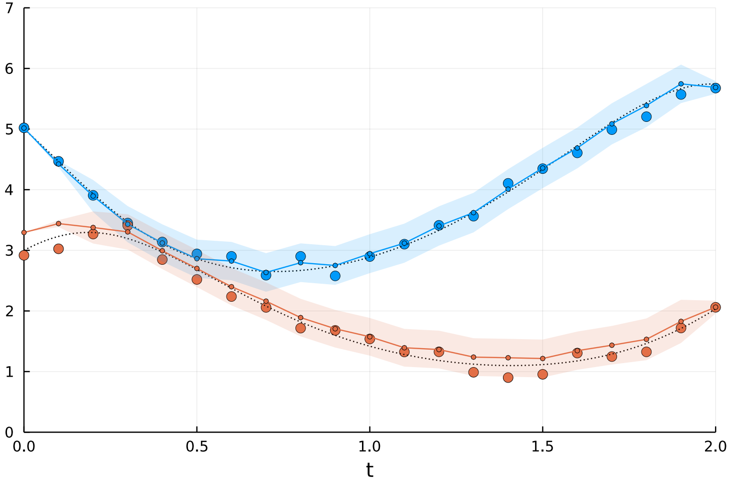


Figure: i=68

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

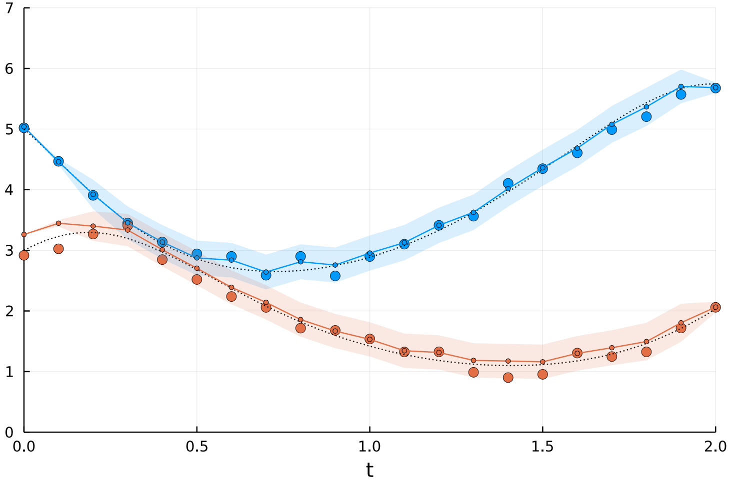


Figure: i=69

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

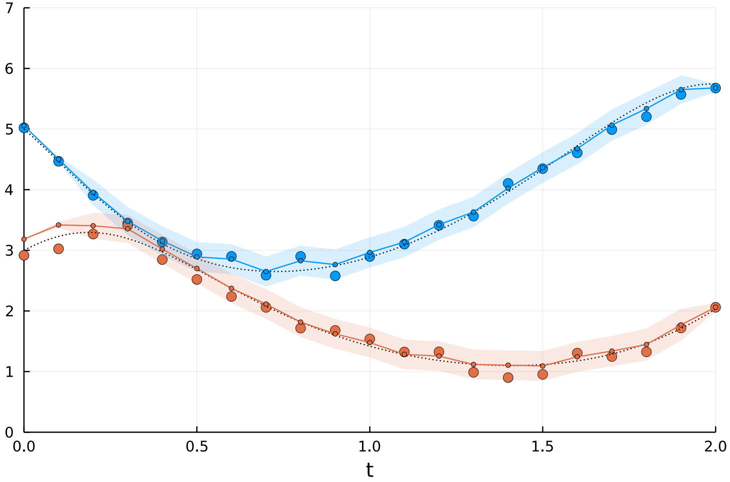


Figure: i=70

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

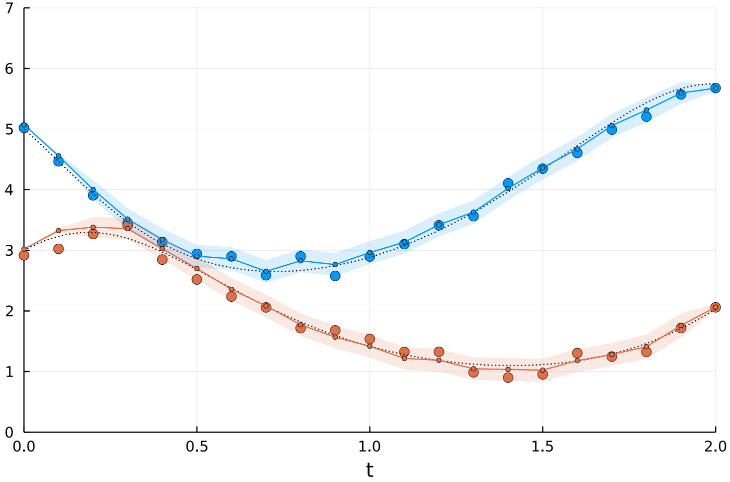


Figure: i=71

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

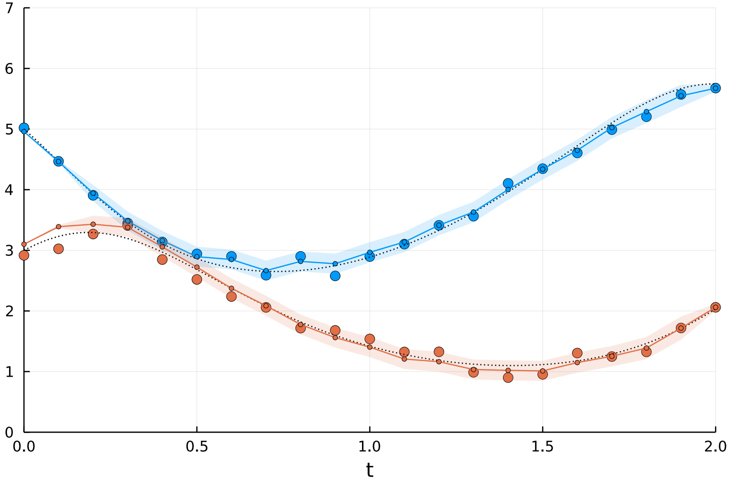


Figure: i=72

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

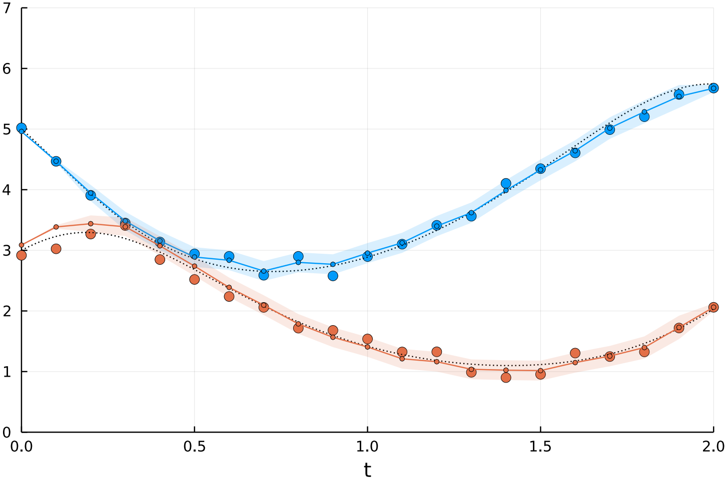


Figure: i=73

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

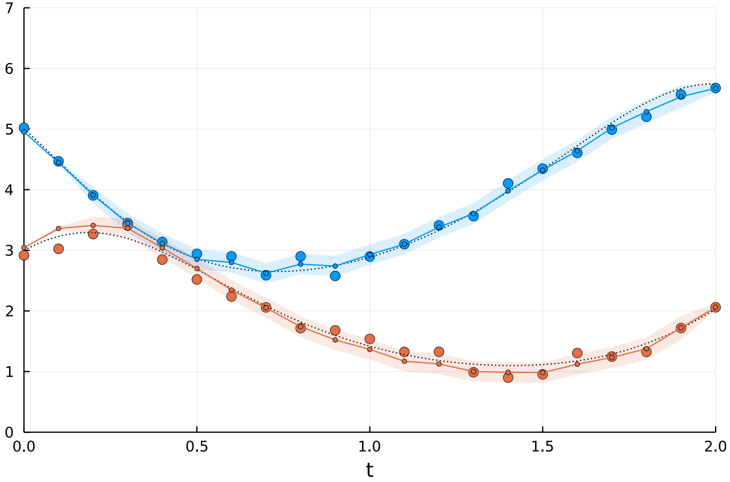


Figure: i=74

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

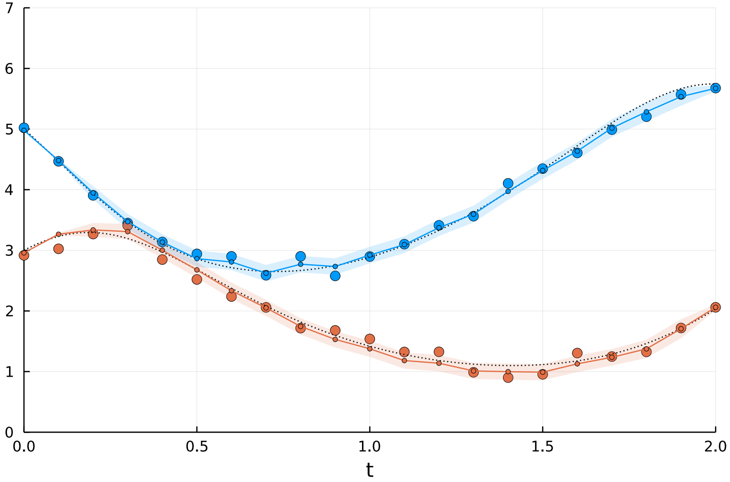


Figure: i=75

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

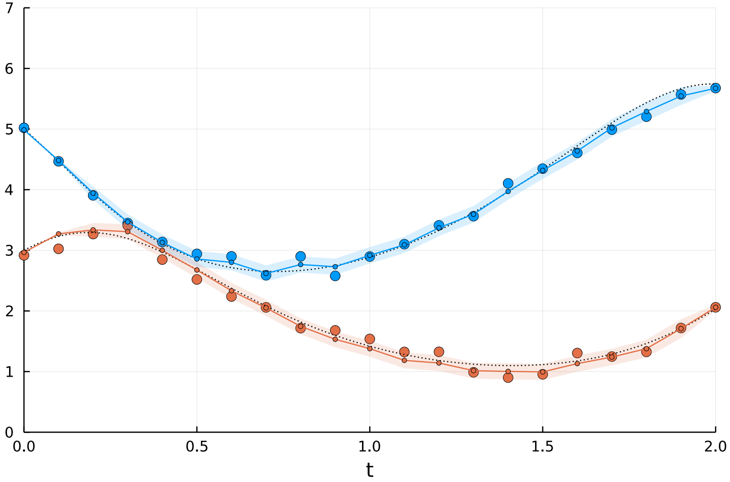


Figure: i=76

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

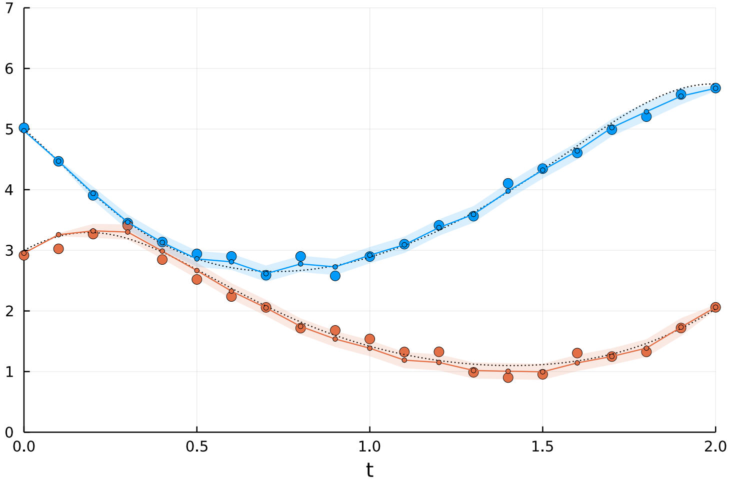


Figure: i=77

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

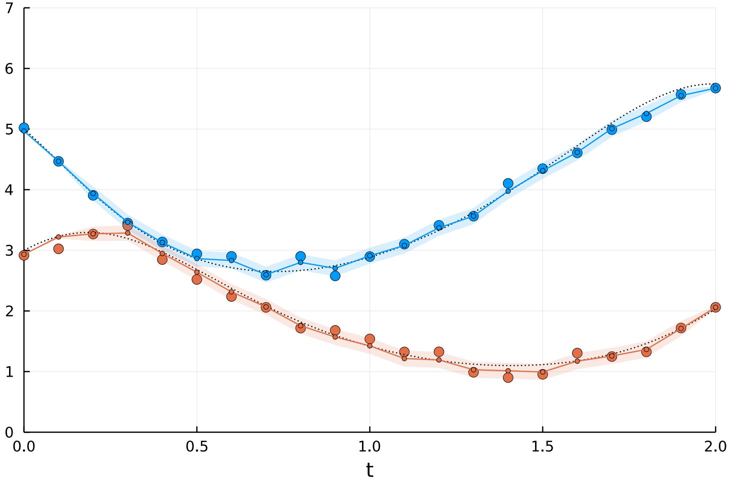


Figure: i=78

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

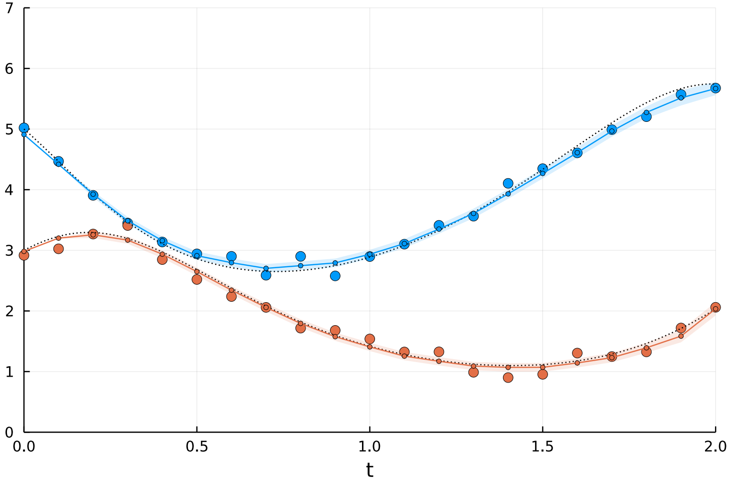


Figure: i=79

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

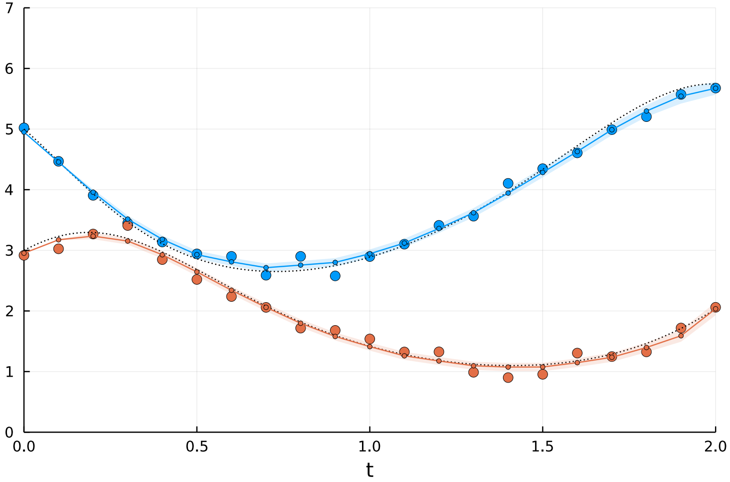


Figure: i=80

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

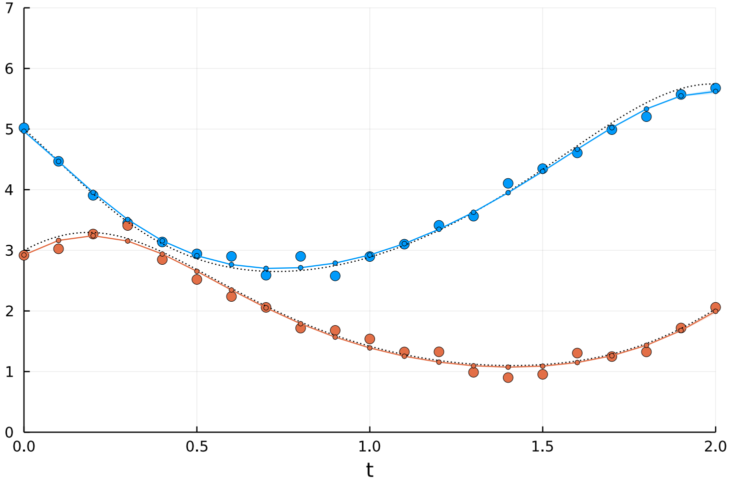


Figure: i=90

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

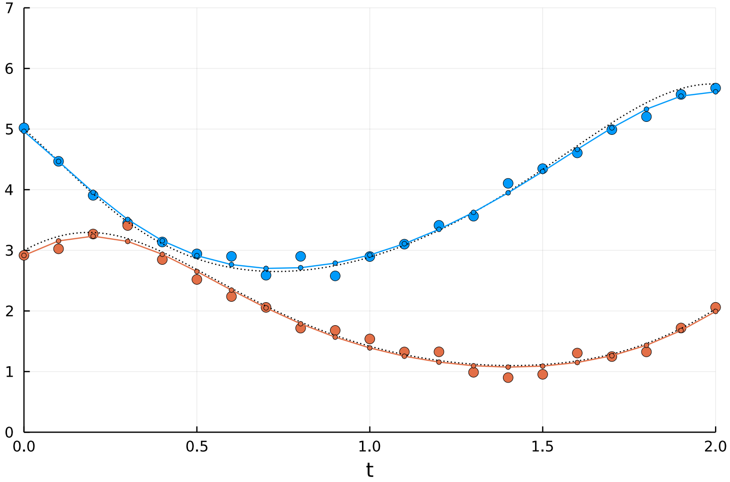


Figure: i=100

Example: Probabilistic Numerical Integration

Optimizing ODE parameters and prior hyperparameters jointly

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

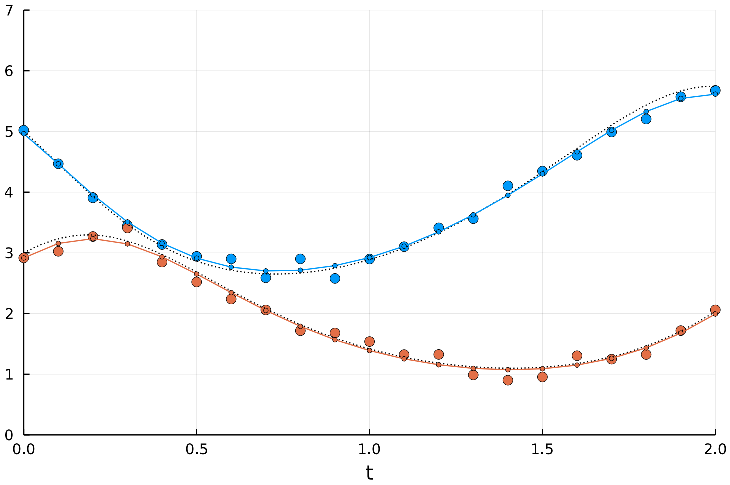


Figure: i=100 DONE

Probabilistic numerics can help escape local optima

By becoming uncertain enough about the ODE solution the method can interpolate the data and continue from there

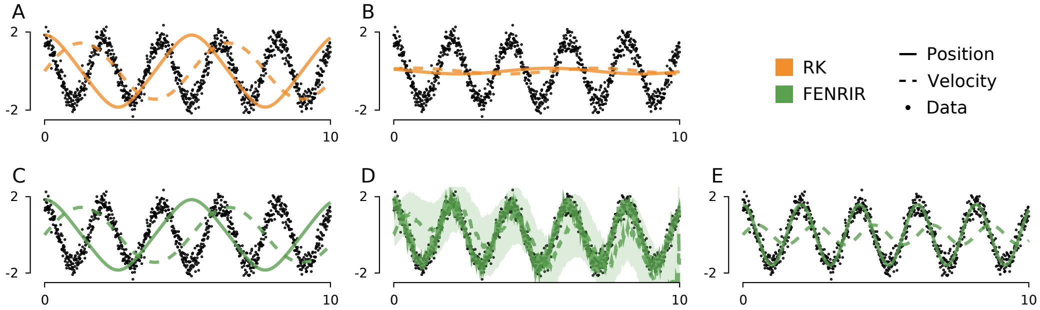
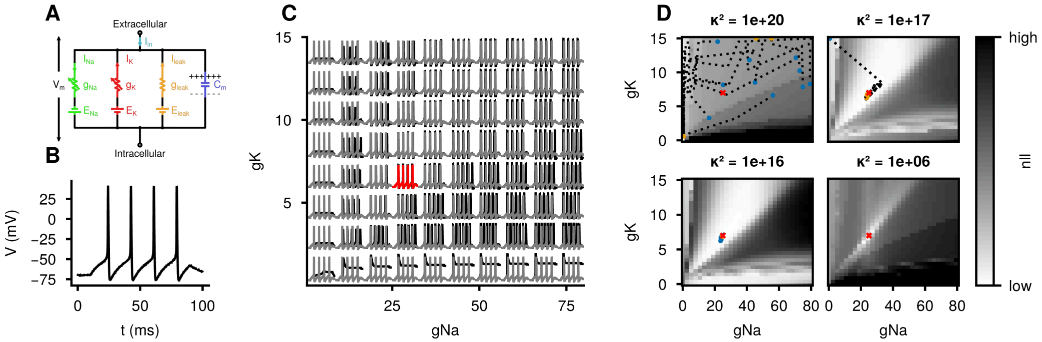


Figure: Learning the length of a simple pendulum with Runge–Kutta (RK) and probabilistic numerics (FENRIR). Out-of-phase initial condition shown on the left, optimization progress shown left to right.

Gradient-based parameter inference in a Hodgkin–Huxley neuron



Summary

- *ODE solving is state estimation*
  $\Rightarrow$ treat initial value problems as state estimation problems
- *"ODE filters"*: **How to solve ODEs with Bayesian filtering and smoothing**
- *Flexible information operators* to solve more than just standard ODEs
- *Parameter inference:* Being uncertain about the ODE solution allows you to update on data

Software packages

```
https://github.com/nathanaelbosch/ProbNumDiffEq.jl
]add ProbNumDiffEq
```

```
https://github.com/probabilistic-numerics/probnum
pip install probnum
```

```
https://github.com/pnkraemer/probdiffeq
pip install probdiffeq
```

Bibliography I

- Bosch, N., Corenflos, A., Yaghoobi, F., Tronarp, F., Hennig, P., and Särkkä, S. (2023a).
  Parallel-in-time probabilistic numerical ODE solvers.
- Bosch, N., Hennig, P., and Tronarp, F. (2021).
  Calibrated adaptive probabilistic ODE solvers.
  In Banerjee, A. and Fukumizu, K., editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3466–3474. PMLR.
- Bosch, N., Hennig, P., and Tronarp, F. (2023b).
  Probabilistic exponential integrators.
  In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Bosch, N., Tronarp, F., and Hennig, P. (2022).
  Pick-and-mix information operators for probabilistic ODE solvers.
  In Camps-Valls, G., Ruiz, F. J. R., and Valera, I., editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 10015–10027. PMLR.

Bibliography II

- Kersting, H., Sullivan, T. J., and Hennig, P. (2020).
  Convergence rates of gaussian ode filters.
  Statistics and Computing, 30(6):1791–1816.
- Krämer, N., Bosch, N., Schmidt, J., and Hennig, P. (2022).
  Probabilistic ODE solutions in millions of dimensions.
  In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11634–11649. PMLR.
- Krämer, N. and Hennig, P. (2020).
  Stable implementation of probabilistic ode solvers.
  CoRR.
- Krämer, N. and Hennig, P. (2021).
  Linear-time probabilistic solution of boundary value problems.
  In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11160–11171. Curran Associates, Inc.

Bibliography III

- Krämer, N., Schmidt, J., and Hennig, P. (2022).
  Probabilistic numerical method of lines for time-dependent partial differential equations.
  In Camps-Valls, G., Ruiz, F. J. R., and Valera, I., editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 625–639. PMLR.
- Schmidt, J., Krämer, N., and Hennig, P. (2021).
  A probabilistic state space model for joint inference from differential equations and data.
  In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12374–12385. Curran Associates, Inc.
- Schober, M., Särkkä, S., and Hennig, P. (2019).
  A probabilistic model for the numerical solution of initial value problems.
  Statistics and Computing, 29(1):99–122.

Bibliography IV

- Tronarp, F., Bosch, N., and Hennig, P. (2022).
  Fenrir: Physics-enhanced regression for initial value problems.
  In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 21776–21794. PMLR.
- Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. (2019).
  Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective.
  Statistics and Computing, 29(6):1297–1315.
- Tronarp, F., Särkkä, S., and Hennig, P. (2021).
  Bayesian ode solvers: the maximum a posteriori estimate.
  Statistics and Computing, 31(3):23.

BACKUP

Background: Extended Kalman filtering and smoothing

Bayesian filters and smoothers estimate an unknown state (often continuous) from observations

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Non-linear Gaussian state-estimation problem:

Initial distribution: $x_0 \sim \mathcal{N}(x_0; \mu_0, \Sigma_0)$,

Prior / dynamics: $x_{i+1} \mid x_i \sim \mathcal{N}(x_{i+1}; f(x_i), Q_i)$,

Likelihood / measurement: $z_i \mid x_i \sim \mathcal{N}(z_i; m(x_i), R_i)$,

Data: $\mathcal{D} = \{z_i\}_{i=1}^N$.

The extended Kalman filter/smoother (EKF/EKS) recursively computes Gaussian approximations:

Predict: $p(x_i \mid z_{1:i-1}) \approx \mathcal{N}(x_i; \mu_i^P, \Sigma_i^P)$,

Filter: $p(x_i \mid z_{1:i}) \approx \mathcal{N}(x_i; \mu_i, \Sigma_i)$,

Smooth: $p(x_i \mid z_{1:N}) \approx \mathcal{N}(x_i; \mu_i^S, \Sigma_i^S)$,

Likelihood: $p(z_i \mid z_{1:i-1}) \approx \mathcal{N}(z_i; \hat{z}_i, S_i)$.

EKF PREDICT

$$
\begin{aligned}
\mu_{i+1}^P &= f(\mu_i), \\
\Sigma_{i+1}^P &= J_f(\mu_i)\Sigma_i J_f(\mu_i)^\top + Q_i.
\end{aligned}
$$

EKF UPDATE

$$
\begin{aligned}
\hat{z}_i &= m(\mu_i^P), \\
S_i &= J_m(\mu_i^P)\Sigma_i^P J_m(\mu_i^P)^\top + R_i, \\
K_i &= \Sigma_i^P J_m(\mu_i^P)^\top S_i^{-1}, \\
\mu_i &= \mu_i^P + K_i (y_i - \hat{y}_i), \\
\Sigma_i &= \Sigma_i^P - K_i S_i K_i^\top.
\end{aligned}
$$

Similarly SMOOTH.

The extended Kalman ODE filter – building blocks

The well-known predict and update steps for (extended) Kalman filtering

Algorithm Kalman filter prediction

procedure KF_PREDICT(μ, Σ, A, Q)
  $\mu^P \leftarrow A\mu$ // Predict mean
  $\Sigma^P \leftarrow A \Sigma A^\top + Q$ // Predict covariance
  return μ^P, Σ^P
end procedure

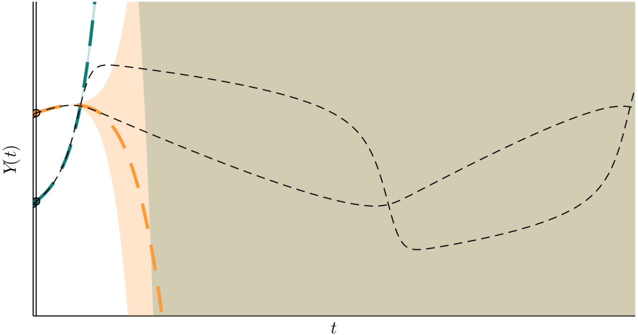
Algorithm Extended Kalman filter update

procedure EKF_UPDATE(μ, Σ, h, R, y)
  $\hat{y} \leftarrow h(\mu)$ // evaluate the observation model
  $H \leftarrow J_h(\mu)$ // Jacobian of the observation model
  $S \leftarrow H \Sigma H^\top + R$ // Measurement covariance
  $K \leftarrow \Sigma H^\top S^{-1}$ // Kalman gain
  $\mu^F \leftarrow \mu + K(y - \hat{y})$ // update mean
  $\Sigma^F \leftarrow \Sigma - K S K^\top$ // update covariance
  return μ^F, Σ^F
end procedure

(KF_UPDATE analog but with affine h)

Local calibration and step-size adaptation

Fixed steps — the vanilla way as introduced so far

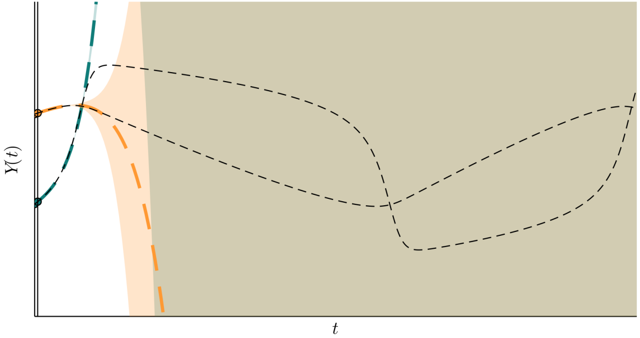


Local calibration and step-size adaptation

Fixed steps – the vanilla way as introduced so far

Calibration

- *Problem*: The Gauss–Markov prior has hyperparameters. How to choose them?
- Most notably: The *diffusion* σ (basically acts as an output scale)

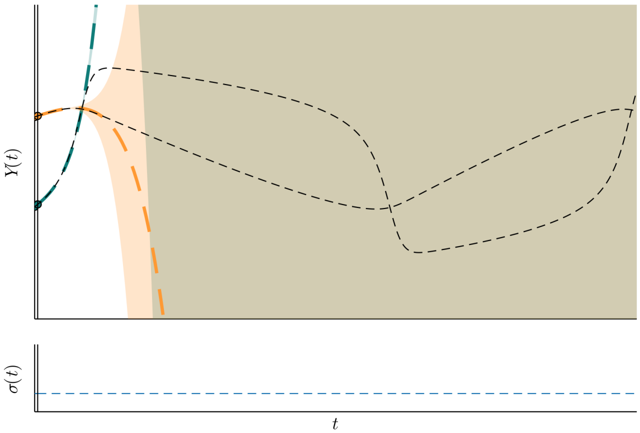


Local calibration and step-size adaptation

Local calibration by estimating a time-varying diffusion model $\sigma(t)$

Calibration

- *Problem*: The Gauss–Markov prior has hyperparameters. How to choose them?
- Most notably: The *diffusion* σ (basically acts as an output scale)
- *Solution*: (Quasi-)MLE (can be done in closed form here)



Local calibration and step-size adaptation

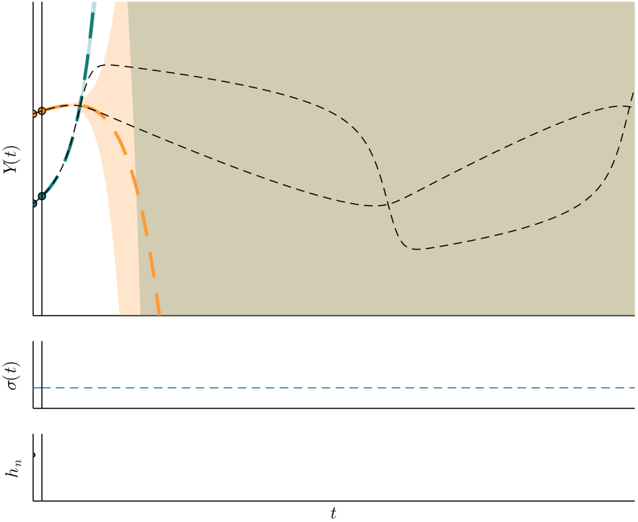
Adaptive step-size selection via local error estimation from the measurement residuals

Calibration

- *Problem*: The Gauss–Markov prior has hyperparameters. How to choose them?
- Most notably: The *diffusion* σ (basically acts as an output scale)
- *Solution*: (Quasi-)MLE (can be done in closed form here)

Step-size adaptation

- Local error estimates from measurement residuals
- Step-size selection with PI-control (similar as in classic solvers)



Prior: The ν-times integrated Wiener process

A very convenient prior with closed-form transition densities

- **ν-times integrated Wiener process prior**: $x(t) \sim \mathrm{IWP}(q)$

$$\begin{aligned} \mathrm{d}x^{(i)}(t) &= x^{(i+1)}(t)\mathrm{d}t, \qquad i = 0, \dots, q-1, \\ \mathrm{d}x^{(q)}(t) &= \sigma \mathrm{d}W(t), \\ x(0) &\sim \mathcal{N}(\mu_0, \Sigma_0). \end{aligned}$$

- Corresponds to Taylor-polynomial + perturbation:

$$x^{(0)}(t) = \sum_{m=0}^{q} x^{(m)}(0) \frac{t^m}{m!} + \sigma \int_0^t \frac{t-\tau}{q!} \mathrm{d}W(\tau)$$

On linearization strategies and their influence on A-Stability

We can actually approximate the Jacobian in the EKF and still get sensible results / algorithms!

[Tronarp et al., 2019]

- Measurement model: $m(x(t), t) = x^{(1)}(t) - f(x^{(0)}(t), t)$
- A standard extended Kalman filter computes the Jacobian of the measurement mode: $J_m(\xi) = E_1 - J_f(E_0\xi, t)E_0 \setminus \Rightarrow$ This algorithm is often called **EK1**.
- Turns out the following also works: $J_f \approx 0$ and then $J_m(\xi) \approx E_1 \setminus \Rightarrow$ The resulting algorithm is often called **EK0**.

A comparison of **EK1** and **EK0**:

	Jacobian	type	A-stable	uncertainties	speed
EK1	$H = E_1 - J_f(E_0\mu^p)E_0$	semi-implicit	yes	more expressive	slower ($O(Nd^3q^3)$)
EK0	$H = E_1$	explicit	no	simpler	faster ($O(Ndq^3)$)

Uncertainty calibration or "how to choose prior hyperparameters"

Hyperparameters of the prior have a strong influence on posteriors – so we need to estimate them

[Tronarp et al., 2019]

- **Problem**: The prior hyperparameter σ strongly influences covariances. How to choose it?
- **Standard approach**: Maximize the marginal likelihood:

$$\hat{\sigma} = \arg\max p(\mathcal{D}_{\text{PN}} \mid \sigma) = p(z_{1:N} \mid \sigma) = p(z_1 \mid \sigma) \prod_{k=2}^{N} p(z_k | z_{1:k-1}, \sigma).$$

- The EKF provides Gaussian estimates $p(z_k \mid z_{1:k-1}) \approx \mathcal{N}(z_k; \hat{z}_k, S_k)$.
  $\Rightarrow$ Quasi-maximum likelihood estimate:

$$\hat{\sigma} = \arg\max p(\mathcal{D}_{\text{PN}} \mid \sigma) = \arg\max \sum_{k=1}^{N} \log p(z_k \mid z_{1:k-1}, \sigma)$$

- **In our specific context** there is a closed-form solution (proof: [Tronarp et al., 2019]):

$$\hat{\sigma}^2 = \frac{1}{Nd} \sum_{i=1}^{N} (z_i - \hat{z}_i)^\top S_i^{-1} (z_i - \hat{z}_i),$$

  and we don't even need to run the filter again! Just adjust the estimated covariances:

$$\Sigma_i \leftarrow \hat{\sigma}^2 \cdot \Sigma_i, \quad \forall i \in \{1, \dots, N\}.$$

Numerically stable implementation: Square-root filtering

When steps get small numerical stability suffers – so better work with matrix square-roots directly [Krämer and Hennig, 2020]

- **Problem:** The computed covariances can have negative eigenvalues due to finite precision arithmetic and numerical round-off, in particular with small step sizes. Failure example: `demo.jl`
- It holds: **A matrix $M \in \mathbb{R}^{d \times d}$ is positive semi-definite if and only if there exists a matrix $B \in \mathbb{R}^{d \times d}$ such that $M = BB^\top$.**
- **Kalman filtering and smoothing in square-root form – a minimal derivation**:
  - Central operation in `PREDICT`/`UPDATE`/`SMOOTH`: $M = ABA^\top + C$.
    - Predict: $\Sigma^P = A \Sigma A^\top + Q$
    - Update (in Joseph form): $\Sigma = (I - KH)\Sigma^P (I - KH)^\top + KRK^\top$
    - Smooth (in Joseph form): $\Lambda = (I - GA)\Sigma(I - GA)^\top + G\Lambda^+ G^\top + GQG^\top$
  - **This can be formulated on the square-root level**: Let $M = M_L (M_L)^\top, B = B_L (B_L)^\top, C = C_L (C_L)^\top$:

$$M = ABA^\top + C,$$

$$\Leftrightarrow \quad M_L (M_L)^\top = AB_L (B_L)^\top A^\top + C_L (C_L)^\top = \begin{bmatrix} AB_L & C_L \end{bmatrix} \cdot \begin{bmatrix} AB_L & C_L \end{bmatrix}^\top$$

$$\overset{\text{doing QR}\left(\begin{bmatrix} AB_L & C_L \end{bmatrix}^\top\right)}{\Leftrightarrow} \quad = R^\top Q^\top Q R = R^\top R. \qquad \Rightarrow M_L := R^\top$$

$\Rightarrow$ `PREDICT`/`UPDATE`/`SMOOTH` can be formulated directly on square-roots to preserve PSD-ness!

$\Rightarrow$ **To solve ODEs in a stable way, use the square-root Kalman filters / smoothers!**

Visual Example: EKF

IVP:

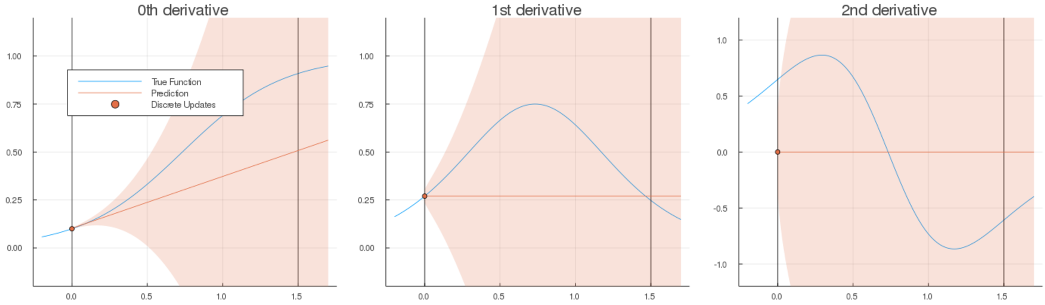
$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 0:



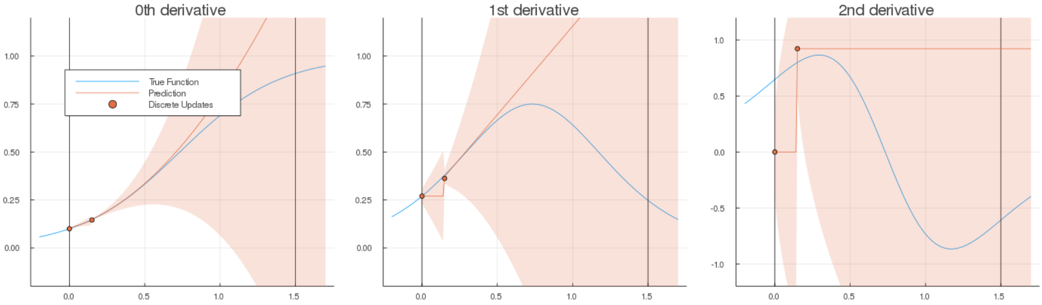
Visual Example: EKF

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 1:

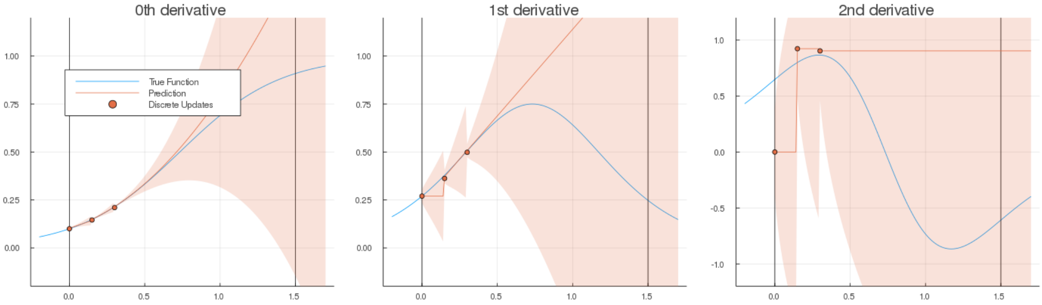


Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 2:

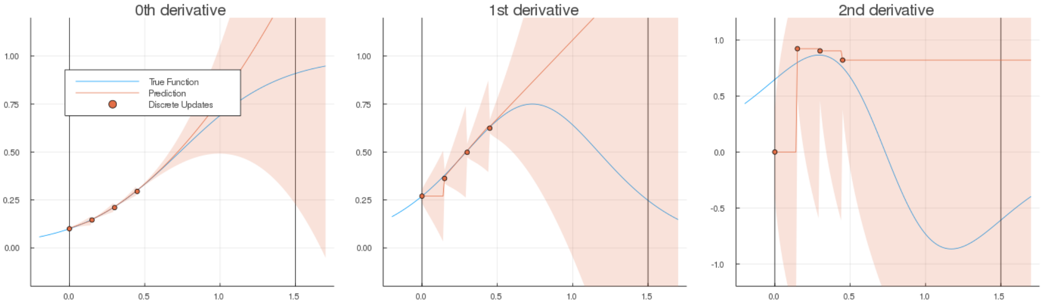


Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 3:

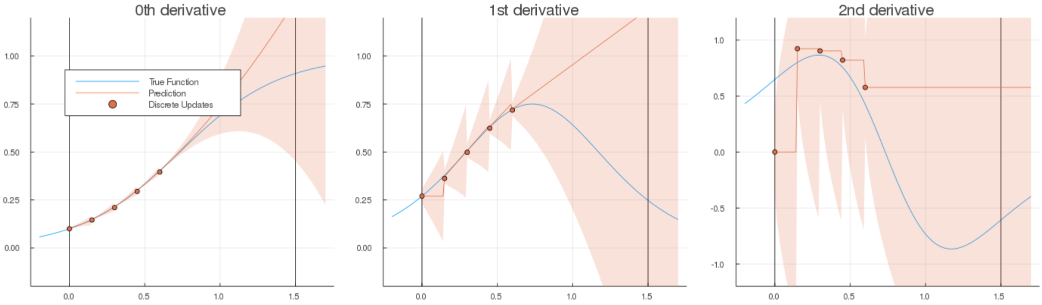


Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 4:



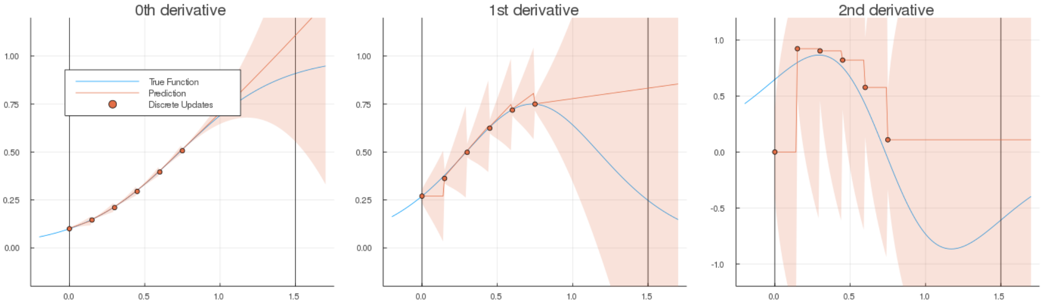
Visual Example: EKF

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 5:

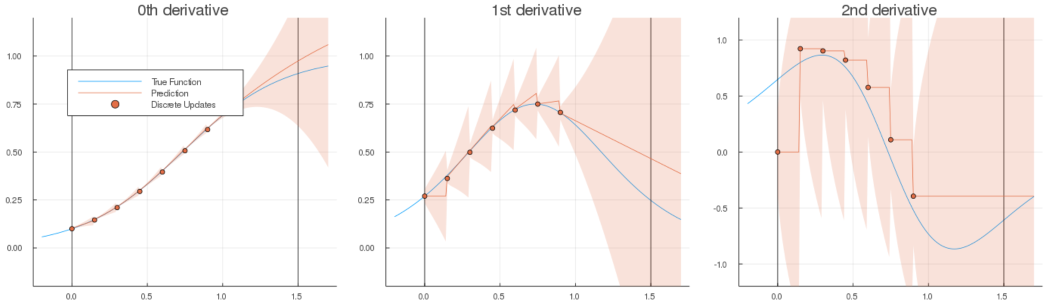


Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 6:

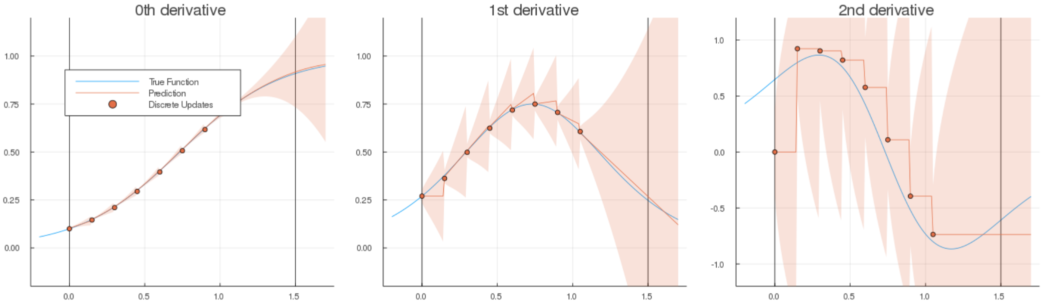


Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 7:

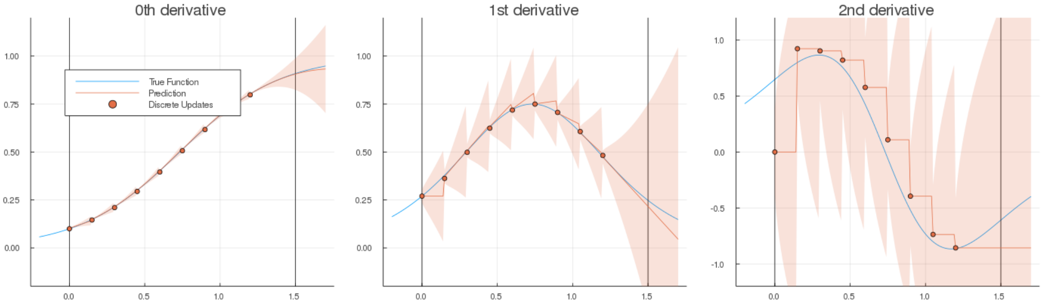


Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 8:

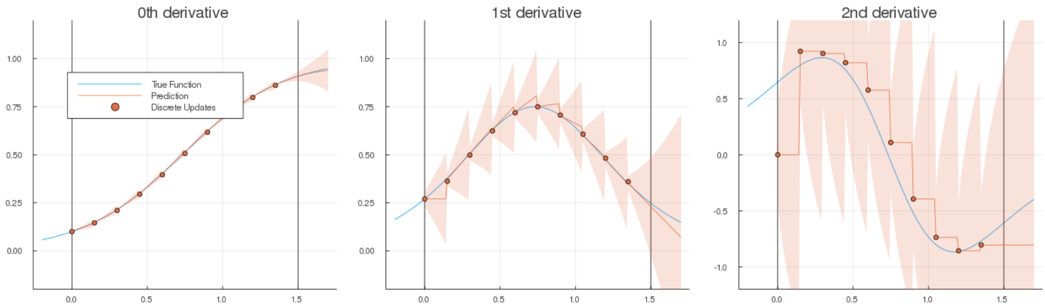


Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 9:



Visual Example: EKF

IVP:

$$y'(t) = 3y(1-y), \qquad y(0) = 0.1, \qquad t \in [0, 1.5].$$

Step 10: