

PROBABILISTIC NUMERICS FOR ORDINARY DIFFERENTIAL EQUATIONS

Nathanael Bosch

22. November 2022

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN



impris-is



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



Background

- ▶ Ordinary differential equations and how to solve them
- ▶ State estimation with extended Kalman filtering & smoothing



Background

- ▶ Ordinary differential equations and how to solve them
- ▶ State estimation with extended Kalman filtering & smoothing

Central statement: ODE solving is state estimation

- ▶ “ODE filters”: **How to solve ODEs with extended Kalman filtering and smoothing**
- ▶ *Bells and whistles* to make ODE filters work even better
 - ▶ Uncertainty calibration
 - ▶ Square-root filtering



Background

- ▶ Ordinary differential equations and how to solve them
- ▶ State estimation with extended Kalman filtering & smoothing

Central statement: ODE solving is state estimation

- ▶ "ODE filters": **How to solve ODEs with extended Kalman filtering and smoothing**
- ▶ *Bells and whistles* to make ODE filters work even better
 - ▶ Uncertainty calibration
 - ▶ Square-root filtering

Fun with ODE filters

- ▶ Generalizing ODE filters to other related problems (higher-order ODEs, DAEs, ...)
- ▶ Latent force inference: Joint GP regression on both ODEs and data



Background: **Ordinary Differential Equations
and how to solve them**

Background: Ordinary Differential Equations and how to solve them



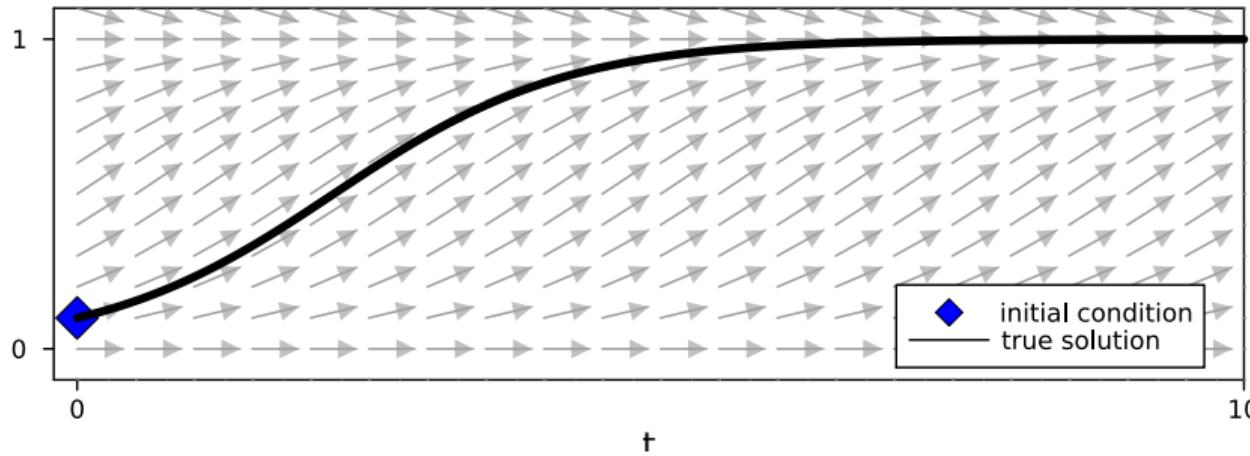
Numerical ODE solvers try to estimate an unknown function by evaluating the vector field

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$, and initial value $x(0) = x_0$. Goal: "Find x ".

► Simple example: Logistic ODE

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





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

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$, and initial value $x(0) = x_0$. Goal: "Find x ".

Numerical ODE solvers:

- ▶ Forward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t), t)$$



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

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$, and initial value $x(0) = x_0$. Goal: "Find x ".

Numerical ODE solvers:

- ▶ Forward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t), t)$$

- ▶ Backward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t + h), t + h)$$



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

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$, and initial value $x(0) = x_0$. Goal: "Find x ".

Numerical ODE solvers:

- ▶ Forward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t), t)$$

- ▶ Backward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t + h), t + h)$$

- ▶ Runge–Kutta:

$$\hat{x}(t + h) = \hat{x}(t) + h \sum_{i=1}^s b_i f(\tilde{x}_i, t + c_i h)$$



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

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$, and initial value $x(0) = x_0$. Goal: "Find x ".

Numerical ODE solvers:

- ▶ Forward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t), t)$$

- ▶ Backward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t + h), t + h)$$

- ▶ Runge–Kutta:

$$\hat{x}(t + h) = \hat{x}(t) + h \sum_{i=1}^s b_i f(\tilde{x}_i, t + c_i h)$$

- ▶ Multistep:

$$\hat{x}(t + h) = \hat{x}(t) + h \sum_{i=0}^{s-1} b_i f(\hat{x}(t - ih), t - ih)$$



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

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$, and initial value $x(0) = x_0$. Goal: "Find x ".

Numerical ODE solvers:

- ▶ Forward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t), t)$$

- ▶ Backward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t + h), t + h)$$

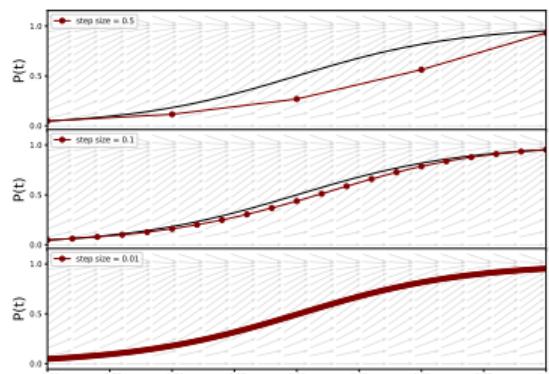
- ▶ Runge–Kutta:

$$\hat{x}(t + h) = \hat{x}(t) + h \sum_{i=1}^s b_i f(\tilde{x}_i, t + c_i h)$$

- ▶ Multistep:

$$\hat{x}(t + h) = \hat{x}(t) + h \sum_{i=0}^{s-1} b_i f(\hat{x}(t - ih), t - ih)$$

Forward Euler for different step sizes:



⇒ It is "correct" only in the limit $h \rightarrow 0!$



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

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

with $t \in [0, T]$, vector field $f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$, and initial value $x(0) = x_0$. Goal: "Find x ".

Numerical ODE solvers:

- ▶ Forward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t), t)$$

- ▶ Backward Euler:

$$\hat{x}(t + h) = \hat{x}(t) + hf(\hat{x}(t + h), t + h)$$

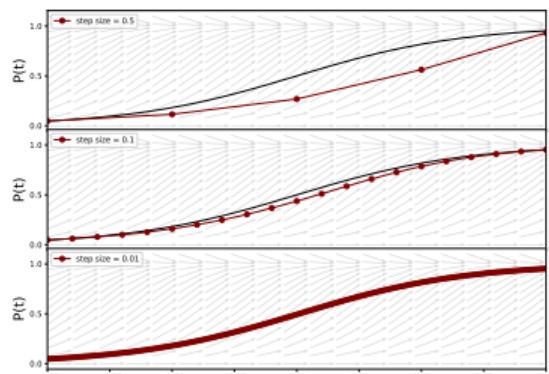
- ▶ Runge–Kutta:

$$\hat{x}(t + h) = \hat{x}(t) + h \sum_{i=1}^s b_i f(\tilde{x}_i, t + c_i h)$$

- ▶ Multistep:

$$\hat{x}(t + h) = \hat{x}(t) + h \sum_{i=0}^{s-1} b_i f(\hat{x}(t - ih), t - ih)$$

Forward Euler for different step sizes:



⇒ It is "correct" only in the limit $h \rightarrow 0!$

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



Background: **Bayesian State Estimation with Extended Kalman filtering and smoothing**



Background: Extended Kalman filtering and smoothing

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

Non-linear Gaussian state-estimation problem:

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

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

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

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



Background: Extended Kalman filtering and smoothing

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

Non-linear Gaussian state-estimation problem:

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

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

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

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

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

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

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

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

Likelihood: $p(y_i | y_{1:i-1}) \approx \mathcal{N}(y_i; \hat{y}_i, S_i).$

Background: Extended Kalman filtering and smoothing

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

Non-linear Gaussian state-estimation problem:

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

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

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

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

PREDICT

$$\mu_{i+1}^P = f(\mu_i),$$

$$\Sigma_{i+1}^P = J_f(\mu_i)\Sigma_i J_f(\mu_i)^\top + Q_i.$$

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

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

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

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

Likelihood: $p(y_i | y_{1:i-1}) \approx \mathcal{N}(y_i; \hat{y}_i, S_i).$



Background: Extended Kalman filtering and smoothing

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

Non-linear Gaussian state-estimation problem:

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

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

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

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

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

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

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

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

Likelihood: $p(y_i | y_{1:i-1}) \approx \mathcal{N}(y_i; \hat{y}_i, S_i).$

PREDICT

$$\mu_{i+1}^P = f(\mu_i),$$

$$\Sigma_{i+1}^P = J_f(\mu_i)\Sigma_i J_f(\mu_i)^\top + Q_i.$$

UPDATE

$$\hat{z}_i = m(\mu_i^P),$$

$$S_i = J_m(\mu_i^P)\Sigma_i^P J_m(\mu_i^P)^\top + R_i,$$

$$K_i = \Sigma_i^P J_m(\mu_i^P)^\top S_i^{-1},$$

$$\mu_i = \mu_i^P + K_i(z_i - \hat{z}_i),$$

$$\Sigma_i = \Sigma_i^P - K_i S_i K_i^\top.$$

Background: Extended Kalman filtering and smoothing

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

Non-linear Gaussian state-estimation problem:

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

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

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

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

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

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

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

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

Likelihood: $p(y_i | y_{1:i-1}) \approx \mathcal{N}(y_i; \hat{y}_i, S_i).$

PREDICT

$$\mu_{i+1}^P = f(\mu_i),$$

$$\Sigma_{i+1}^P = J_f(\mu_i)\Sigma_i J_f(\mu_i)^\top + Q_i.$$

UPDATE

$$\hat{z}_i = m(\mu_i^P),$$

$$S_i = J_m(\mu_i^P)\Sigma_i^P J_m(\mu_i^P)^\top + R_i,$$

$$K_i = \Sigma_i^P J_m(\mu_i^P)^\top S_i^{-1},$$

$$\mu_i = \mu_i^P + K_i(z_i - \hat{z}_i),$$

$$\Sigma_i = \Sigma_i^P - K_i S_i K_i^\top.$$

Similarly SMOOTH.



Today: ***Probabilistic numerical ODE solutions***

or "How to treat ODEs as the state estimation problem that they really are"



Probabilistic numerical ODE solutions

or "How to treat ODEs as the state estimation problem that they really are"



Probabilistic numerical ODE solutions

or "How to treat ODEs as the state estimation problem that they really are"

$$p \left(x(t) \mid x(0) = x_0, \{ \dot{x}(t_n) = f(x(t_n), t_n) \}_{n=1}^N \right)$$



Probabilistic numerical ODE solutions

or "How to treat ODEs as the state estimation problem that they really are"

$$p \left(x(t) \mid x(0) = x_0, \{ \dot{x}(t_n) = f(x(t_n), t_n) \}_{n=1}^N \right)$$

To solve an ODE with Gaussian filtering and smoothing, we need:

1. Prior:
2. Likelihood:
3. Data:



Probabilistic numerical ODE solutions

or "How to treat ODEs as the state estimation problem that they really are"

$$p \left(x(t) \mid x(0) = x_0, \{ \dot{x}(t_n) = f(x(t_n), t_n) \}_{n=1}^N \right)$$

To solve an ODE with Gaussian filtering and smoothing, we need:

1. **Prior:**
2. Likelihood:
3. Data:

Prior: General Gauss–Markov processes

You saw this in lecture 5 (I believe)

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

- **Continuous Gauss–Markov prior:** Let $X(t) = [X^{(0)}(t), X^{(1)}(t), \dots, X^{(q)}(t)]^\top$ be the solution of a linear time-invariant (LTI) stochastic differential equation (SDE):

$$\begin{aligned} dX(t) &= FX(t) dt + \Gamma dW(t), \\ X(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \end{aligned}$$

with F such that $dX^{(i)}(t) = X^{(i+1)}(t)dt$. Then, we use $X^{(i)}(t)$ to model the i -th derivative of $x(t)$.

Examples: Integrated Wiener process, Integrated Ornstein–Uhlenbeck process, Matérn process.

Prior: General Gauss–Markov processes

You saw this in lecture 5 (I believe)

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

- **Continuous Gauss–Markov prior:** Let $X(t) = [X^{(0)}(t), X^{(1)}(t), \dots, X^{(q)}(t)]^\top$ be the solution of a linear time-invariant (LTI) stochastic differential equation (SDE):

$$\begin{aligned} dX(t) &= FX(t) dt + \Gamma dW(t), \\ X(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \end{aligned}$$

with F such that $dX^{(i)}(t) = X^{(i+1)}(t)dt$. Then, we use $X^{(i)}(t)$ to model the i -th derivative of $x(t)$.

Examples: Integrated Wiener process, Integrated Ornstein–Uhlenbeck process, Matérn process.

- **Discrete transition densities:** $X(t)$ can be described in discrete time with

$$X(t+h) \mid X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h)),$$

where $(A(h), Q(h))$ are given by

$$A(h) = \exp(Fh), \quad Q(h) = \int_0^h A(h-\tau)\Gamma\Gamma^\top A(h-\tau)^\top d\tau.$$

The transition matrices $(A(h), Q(h))$ can be computed with the "matrix fraction decomposition"; see for instance Särkkä & Solin, "Applied Stochastic Differential Equations", 2013.

Prior: General Gauss–Markov processes

You saw this in lecture 5 (I believe)

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

- **Continuous Gauss–Markov prior:** Let $X(t) = [X^{(0)}(t), X^{(1)}(t), \dots, X^{(q)}(t)]^\top$ be the solution of a linear time-invariant (LTI) stochastic differential equation (SDE):

$$\begin{aligned} dX(t) &= FX(t) dt + \Gamma dW(t), \\ X(0) &\sim \mathcal{N}(\mu_0, \Sigma_0), \end{aligned}$$

with F such that $dX^{(i)}(t) = X^{(i+1)}(t)dt$. Then, we use $X^{(i)}(t)$ to model the i -th derivative of $x(t)$.

Examples: **Integrated Wiener process**, Integrated Ornstein–Uhlenbeck process, Matérn process.

- **Discrete transition densities:** $X(t)$ can be described in discrete time with

$$X(t+h) \mid X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h)),$$

where $(A(h), Q(h))$ are given by

$$A(h) = \exp(Fh), \quad Q(h) = \int_0^h A(h-\tau)\Gamma\Gamma^\top A(h-\tau)^\top d\tau.$$

The transition matrices $(A(h), Q(h))$ can be computed with the "matrix fraction decomposition"; see for instance Särkkä & Solin, "Applied Stochastic Differential Equations", 2013.



Prior: The q -times integrated Wiener process

A very convenient prior with closed-form transition densities

- **q -times integrated Wiener process prior:** $X(t) \sim \text{IWP}(q)$

$$dX^{(i)}(t) = X^{(i+1)}(t) dt, \quad i = 0, \dots, q-1,$$

$$dX^{(q)}(t) = \sigma dW(t),$$

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



Prior: The q -times integrated Wiener process

A very convenient prior with closed-form transition densities

- **q -times integrated Wiener process prior:** $X(t) \sim \text{IWP}(q)$

$$dX^{(i)}(t) = X^{(i+1)}(t) dt, \quad i = 0, \dots, q-1,$$

$$dX^{(q)}(t) = \sigma dW(t),$$

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

- **Discrete-time transitions:**

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

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

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

for any $i, j = 0, \dots, q$. (one-dimensional case).
 (proof: [Kersting et al., 2020])

Prior: The q -times integrated Wiener process

A very convenient prior with closed-form transition densities

- **q -times integrated Wiener process prior:** $X(t) \sim \text{IWP}(q)$

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

- **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).
 (proof: [Kersting et al., 2020])

- **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:** $X(t) \sim \text{IWP}(q)$

$$dX^{(i)}(t) = X^{(i+1)}(t) dt, \quad i = 0, \dots, q-1,$$

$$dX^{(q)}(t) = \sigma dW(t),$$

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

- **Discrete-time transitions:**

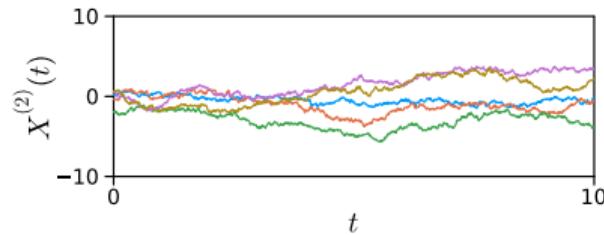
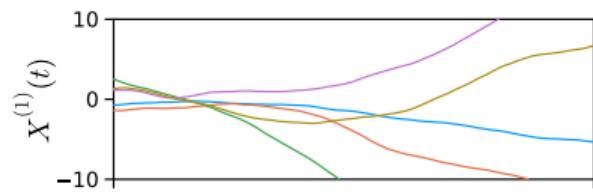
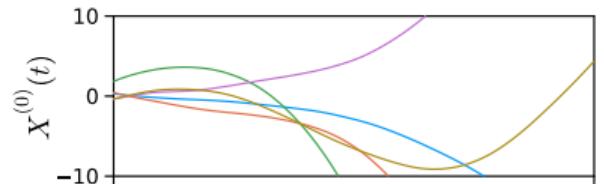
$$X(t+h) | X(t) \sim \mathcal{N}\left(X(t+h); A(h)X(t), \sigma^2 Q(h)\right),$$

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

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

for any $i, j = 0, \dots, q$. (one-dimensional case).
(proof: [Kersting et al., 2020])

- **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:** $X(t) \sim \text{IWP}(q)$

$$dX^{(i)}(t) = X^{(i+1)}(t) dt, \quad i = 0, \dots, q-1,$$

$$dX^{(q)}(t) = \sigma dW(t),$$

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

- **Discrete-time transitions:**

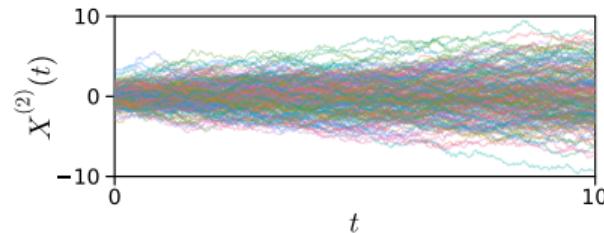
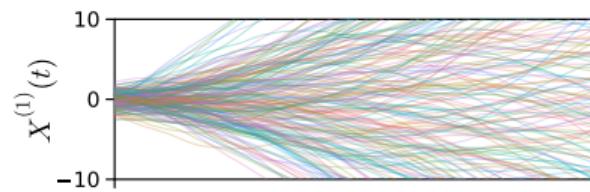
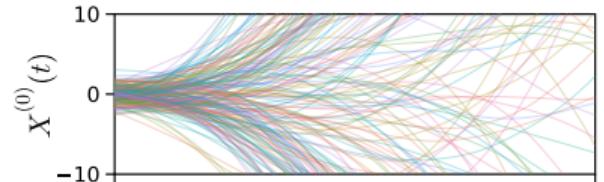
$$X(t+h) | X(t) \sim \mathcal{N}\left(X(t+h); A(h)X(t), \sigma^2 Q(h)\right),$$

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

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

for any $i, j = 0, \dots, q$. (one-dimensional case).
 (proof: [Kersting et al., 2020])

- **Example:** IWP(2)





Probabilistic numerical ODE solutions

How to treat ODEs as the state estimation problem that they really are

$$p \left(x(t) \mid x(0) = x_0, \{ \dot{x}(t_n) = f(x(t_n), t_n) \}_{n=1}^N \right)$$

To solve an ODE with Gaussian filtering and smoothing, we need:

1. **Prior:** q -times integrated Wiener process prior

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

2. Likelihood:
3. Data:

Probabilistic numerical ODE solutions



How to treat ODEs as the state estimation problem that they really are

$$p \left(x(t) \mid x(0) = x_0, \{ \dot{x}(t_n) = f(x(t_n), t_n) \}_{n=1}^N \right)$$

To solve an ODE with Gaussian filtering and smoothing, we need:

1. Prior: q -times integrated Wiener process prior

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

2. **Likelihood:**
3. **Data:**



The likelihood model and the data

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

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

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



The likelihood model and the data

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

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

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

using $X(t)$

$$\Leftrightarrow \quad X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$



The likelihood model and the data

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

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

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

$$\stackrel{\text{using } X(t)}{\Leftrightarrow} \quad X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right)$$



The likelihood model and the data

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

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

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

$$\stackrel{\text{using } X(t)}{\Leftrightarrow} \quad X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

The likelihood model and the data

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

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

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

$$\stackrel{\text{using } X(t)}{\Leftrightarrow} \quad X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

The likelihood model and the data

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

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

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

$$\stackrel{\text{using } X(t)}{\Leftrightarrow} \quad X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

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

The likelihood model and the data

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

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

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

$$\stackrel{\text{using } X(t)}{\Leftrightarrow} \quad X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

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

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

$$Z(t_i) \mid X(t_i) \sim \mathcal{N}(m(X(t_i), t_i), R)$$

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

where z_i is a realization of $Z(t_i)$.

The likelihood model and the data

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

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

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

$$\stackrel{\text{using } X(t)}{\Leftrightarrow} \quad X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

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

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

$$Z(t_i) \mid X(t_i) \sim \mathcal{N}(m(X(t_i), t_i), 0)$$

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

where z_i is a realization of $Z(t_i)$.

The likelihood model and the data

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

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

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

$$\stackrel{\text{using } X(t)}{\Leftrightarrow} \quad X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

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

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

$$Z(t_i) \mid X(t_i) \sim \delta(m(X(t_i), t_i))$$

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

where z_i is a realization of $Z(t_i)$.
 (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 but intractable goal:** Want $x(t)$ to satisfy the ODE

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

using $X(t)$
 \Leftrightarrow

$$X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

$$\Leftrightarrow m(X(t_i), t_i) = 0$$

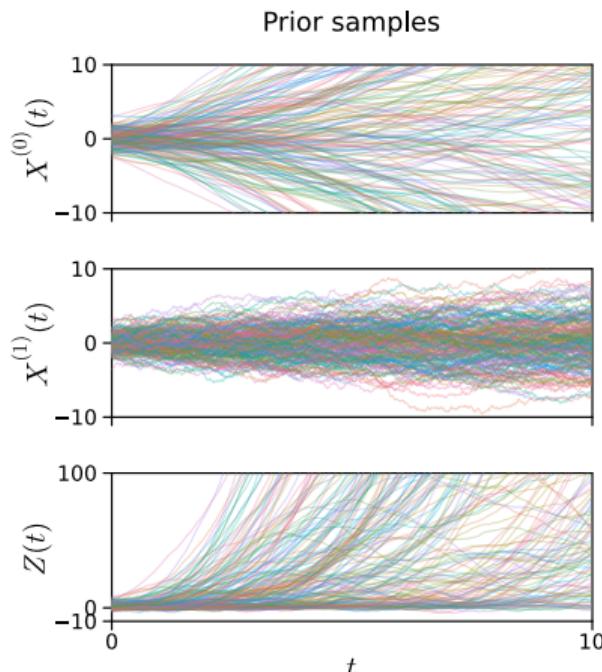
- This motivates a *noiseless measurement model* and **data**:

$$Z(t_i) \mid X(t_i) \sim \delta(m(X(t_i), t_i))$$

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

where z_i is a realization of $Z(t_i)$.
 $(\delta$ is the Dirac distribution)

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



(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 but intractable goal:** Want $x(t)$ to satisfy the ODE

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

using $X(t)$
 \Leftrightarrow

$$X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

$$\Leftrightarrow m(X(t_i), t_i) = 0$$

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

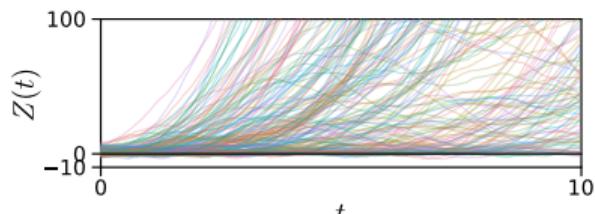
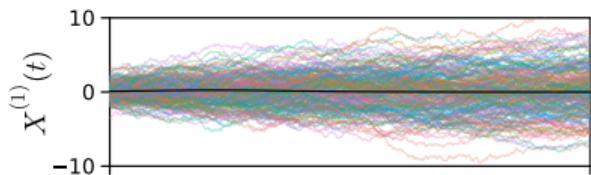
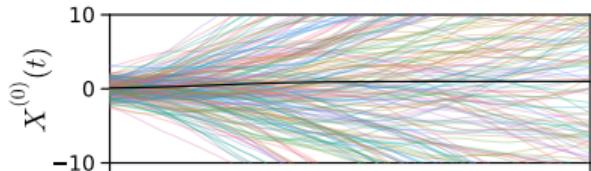
$$Z(t_i) \mid X(t_i) \sim \delta(m(X(t_i), t_i))$$

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

where z_i is a realization of $Z(t_i)$.
 $(\delta$ is the Dirac distribution)

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

Prior samples & ODE solution



(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 but intractable goal:** Want $x(t)$ to satisfy the ODE

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

using $X(t)$
 \Leftrightarrow

$$X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

$$\Leftrightarrow m(X(t_i), t_i) = 0$$

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

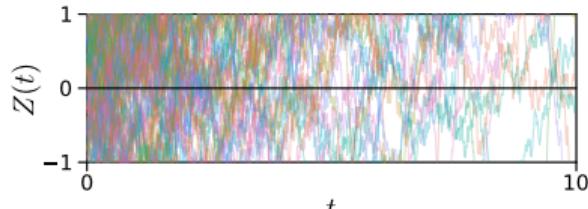
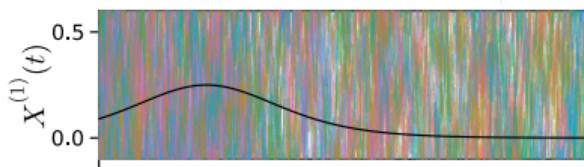
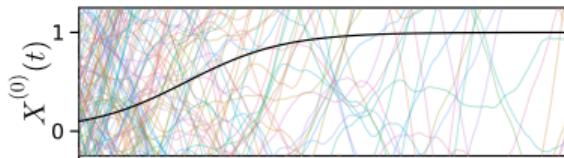
$$Z(t_i) \mid X(t_i) \sim \delta(m(X(t_i), t_i))$$

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

where z_i is a realization of $Z(t_i)$.
 $(\delta$ is the Dirac distribution)

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

Prior samples & ODE solution (zoomed)



(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 but intractable goal:** Want $x(t)$ to satisfy the ODE

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

using $X(t)$
 \Leftrightarrow

$$X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

$$\Leftrightarrow m(X(t_i), t_i) = 0$$

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

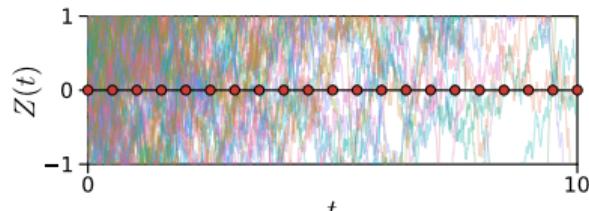
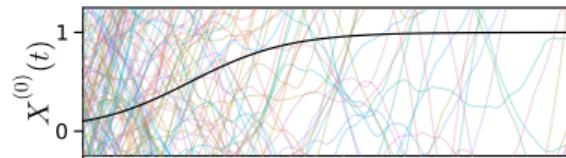
$$Z(t_i) \mid X(t_i) \sim \delta(m(X(t_i), t_i))$$

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

where z_i is a realization of $Z(t_i)$.
 $(\delta$ is the Dirac distribution)

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

Prior samples & ODE solution & "Data"



(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 but intractable goal:** Want $x(t)$ to satisfy the ODE

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

using $X(t)$
 \Leftrightarrow

$$X^{(1)}(t) = f\left(X^{(0)}(t), t\right)$$

$$0 = X^{(1)}(t) - f\left(X^{(0)}(t), t\right) =: m(X(t), t).$$

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

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

$$\Leftrightarrow m(X(t_i), t_i) = 0$$

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

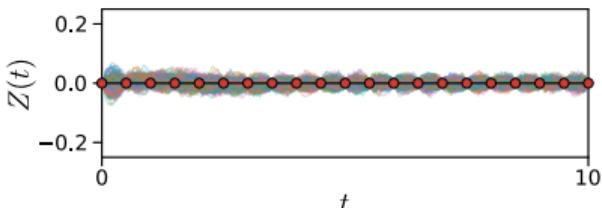
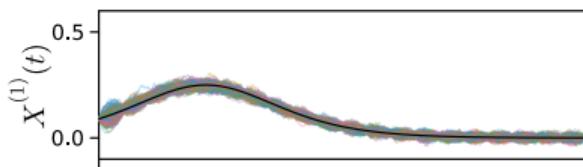
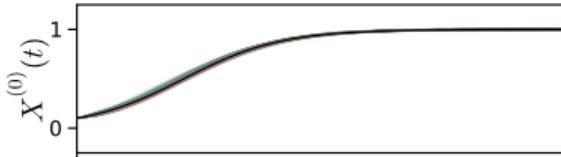
$$Z(t_i) \mid X(t_i) \sim \delta(m(X(t_i), t_i))$$

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

where z_i is a realization of $Z(t_i)$.
 $(\delta$ is the Dirac distribution)

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

Posterior samples & ODE solution



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

Spoiler: This is the thing we want!



Probabilistic numerical ODE solutions

How to treat ODEs as the state estimation problem that they really are

$$p \left(x(t) \mid x(0) = x_0, \{ \dot{x}(t_n) = f(x(t_n), t_n) \}_{n=1}^N \right)$$

To solve an ODE with Gaussian filtering and smoothing, we need:

1. Prior: q -times integrated Wiener process prior:

$$X(t+h) \mid X(t) \sim \mathcal{N}(A(h)X(t), Q(h))$$

2. **Likelihood:** $Z(t) \mid X(t) \sim \delta(X^{(1)}(t) - f(X^{(0)}(t), t))$
3. **Data:** $\mathcal{D}_{\text{PN}} = \{Z_i\}$, with $(Z(t_i) =)z_i = 0$ on a discrete time grid $t_i \in \mathbb{T}$.



Probabilistic numerical ODE solutions

How to treat ODEs as the state estimation problem that they really are

$$p \left(x(t) \mid x(0) = x_0, \{ \dot{x}(t_n) = f(x(t_n), t_n) \}_{n=1}^N \right)$$

To solve an ODE with Gaussian filtering and smoothing, we need:

1. Prior: q -times integrated Wiener process prior:

$$X(t+h) \mid X(t) \sim \mathcal{N}(A(h)X(t), Q(h))$$

2. Likelihood: $Z(t) \mid X(t) \sim \delta(X^{(1)}(t) - f(X^{(0)}(t), t))$
3. Data: $\mathcal{D}_{\text{PN}} = \{z_i\}$, with $(Z(t_i) =) z_i = 0$ on a discrete time grid $t_i \in \mathbb{T}$.

This describes a state-space model



Probabilistic numerical ODE solutions

How to treat ODEs as the state estimation problem that they really are

$$p \left(x(t) \mid x(0) = x_0, \{ \dot{x}(t_n) = f(x(t_n), t_n) \}_{n=1}^N \right)$$

To solve an ODE with Gaussian filtering and smoothing, we need:

1. Prior: q -times integrated Wiener process prior:

$$X(t+h) \mid X(t) \sim \mathcal{N}(A(h)X(t), Q(h))$$

2. Likelihood: $Z(t) \mid X(t) \sim \delta(X^{(1)}(t) - f(X^{(0)}(t), t))$
3. Data: $\mathcal{D}_{\text{PN}} = \{z_i\}$, with $(Z(t_i) =) z_i = 0$ on a discrete time grid $t_i \in \mathbb{T}$.

This describes a state-space model \Rightarrow solve with EKF/EKS!



The extended Kalman ODE filter – the state-space model

Bringing the last slides all together

For a given initial value problem $\dot{x}(t) = f(x(t), t)$ on $[0, T]$ with $x(0) = x_0$, we have:



The extended Kalman ODE filter – the state-space model

Bringing the last slides all together

For a given initial value problem $\dot{x}(t) = f(x(t), t)$ on $[0, T]$ with $x(0) = x_0$, we have:

Initial distribution:

$$X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$$

Prior / dynamics model:

$$X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$$

Likelihood / measurement model:

$$Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right)$$

Data:

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$



The extended Kalman ODE filter – the state-space model

Bringing the last slides all together

For a given initial value problem $\dot{x}(t) = f(x(t), t)$ on $[0, T]$ with $x(0) = x_0$, we have:

Initial distribution:

$$X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$$

Prior / dynamics model:

$$X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$$

Likelihood / measurement model:

$$Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right)$$

Data:

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

One thing is still missing:



The extended Kalman ODE filter – the state-space model

Bringing the last slides all together

For a given initial value problem $\dot{x}(t) = f(x(t), t)$ on $[0, T]$ with $x(0) = x_0$, we have:

Initial distribution:

$$X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$$

Prior / dynamics model:

$$X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$$

Likelihood / measurement model:

$$Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right)$$

Data:

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

One thing is still missing: **What about the initial value??**



The extended Kalman ODE filter – the state-space model

Bringing the last slides all together

For a given initial value problem $\dot{x}(t) = f(x(t), t)$ on $[0, T]$ with $x(0) = x_0$, we have:

Initial distribution:

$$X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$$

Prior / dynamics model:

$$X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$$

Likelihood / measurement model:

$$Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right)$$

Data:

$$z_i \triangleq 0, \quad i = 1, \dots, N.$$

One thing is still missing: **What about the initial value??** Just add another measurement at $t = 0$:

$$Z^{\text{init}} | X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0.$$



The extended Kalman ODE filter

We can solve ODEs with basically just an extended Kalman filter

Algorithm The extended Kalman ODE filter

```
1 procedure EXTENDED KALMAN ODE FILTER( $(\mu_0^-, \Sigma_0^-), (A, Q), (f, x_0), \{t_i\}_{i=1}^N$ )
2    $\mu_0, \Sigma_0 \leftarrow \text{KF\_UPDATE}(\mu_0^-, \Sigma_0^-, E_0, 0_{d \times d}, x_0)$                                 // Initial update to fit the initial value
3   for  $k \in \{1, \dots, N\}$  do
4      $h_k \leftarrow t_k - t_{k-1}$                                                  // Step size
5      $\mu_k^-, \Sigma_k^- \leftarrow \text{KF\_PREDICT}(\mu_{k-1}, \Sigma_{k-1}, A(h_k), Q(h_k))$           // Kalman filter prediction
6      $m_k(X) := E_1 X - f(E_0 X, t_k)$                                          // Define the non-linear observation model
7      $\mu_k, \Sigma_k \leftarrow \text{EKF\_UPDATE}(\mu_k^-, \Sigma_k^-, m_k, 0_{d \times d}, \mathbf{0}_d)$           // Extended Kalman filter update
8   end for
9   return  $(\mu_k, \Sigma_k)_{k=1}^N$ 
10 end procedure
```

Recall: The state $X(t)$ is a stack of q derivatives $X = [X^{(0)}, X^{(1)}, \dots, X^{(q)}]^T$.

For convenience, define projection matrices E_i to map to the i -th derivative: $E_i X = X^{(i)}$.



The extended Kalman ODE filter

We can solve ODEs with basically just an extended Kalman filter

Algorithm The extended Kalman ODE filter

```
1 procedure EXTENDED KALMAN ODE FILTER( $(\mu_0^-, \Sigma_0^-), (A, Q), (f, x_0), \{t_i\}_{i=1}^N$ )
2    $\mu_0, \Sigma_0 \leftarrow \text{KF\_UPDATE}(\mu_0^-, \Sigma_0^-, E_0, 0_{d \times d}, x_0)$                                 // Initial update to fit the initial value
3   for  $k \in \{1, \dots, N\}$  do
4      $h_k \leftarrow t_k - t_{k-1}$                                                  // Step size
5      $\mu_k^-, \Sigma_k^- \leftarrow \text{KF\_PREDICT}(\mu_{k-1}, \Sigma_{k-1}, A(h_k), Q(h_k))$           // Kalman filter prediction
6      $m_k(X) := E_1 X - f(E_0 X, t_k)$                                          // Define the non-linear observation model
7      $\mu_k, \Sigma_k \leftarrow \text{EKF\_UPDATE}(\mu_k^-, \Sigma_k^-, m_k, 0_{d \times d}, \mathbf{0}_d)$           // Extended Kalman filter update
8   end for
9   return  $(\mu_k, \Sigma_k)_{k=1}^N$ 
10 end procedure
```

Recall: The state $X(t)$ is a stack of q derivatives $X = [X^{(0)}, X^{(1)}, \dots, X^{(q)}]^T$.

For convenience, define projection matrices E_i to map to the i -th derivative: $E_i X = X^{(i)}$.

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



The extended Kalman ODE filter – building blocks

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

Algorithm Kalman filter prediction

```
1 procedure KF_PREDICT( $\mu$ ,  $\Sigma$ ,  $A$ ,  $Q$ )
2    $\mu^P \leftarrow A\mu$                                 // Predict mean
3    $\Sigma^P \leftarrow A\Sigma A^\top + Q$            // Predict covariance
4   return  $\mu^P, \Sigma^P$ 
5 end procedure
```

Algorithm Extended Kalman filter update

```
1 procedure EKF_UPDATE( $\mu$ ,  $\Sigma$ ,  $h$ ,  $R$ ,  $y$ )
2    $\hat{y} \leftarrow h(\mu)$                          // evaluate the observation model
3    $H \leftarrow J_h(\mu)$                           // Jacobian of the observation model
4    $S \leftarrow H\Sigma H^\top + R$             // Measurement covariance
5    $K \leftarrow \Sigma H^\top S^{-1}$           // Kalman gain
6    $\mu^F \leftarrow \mu + K(y - \hat{y})$         // update mean
7    $\Sigma^F \leftarrow \Sigma - KSK^\top$         // update covariance
8   return  $\mu^F, \Sigma^F$ 
9 end procedure
```

(KF_UPDATE analog but with affine h)



DEMO TIME: The extended Kalman ODE filter in code

`demo.jl`

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?

Uncertainty calibration or “how to choose prior hyperparameters”

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



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

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

Uncertainty calibration or “how to choose prior hyperparameters”

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



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

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

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

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

Uncertainty calibration or “how to choose prior hyperparameters”

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



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

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

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

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

- ▶ **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:

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



DEMO TIME: Calibrated vs uncalibrated posteriors

demo.jl



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`



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 $M = BB^\top$.**



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 $M = BB^T$.
- ▶ **Kalman filtering and smoothing in square-root form – a minimal derivation:**



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): $\Lambda = (I - GA)\Sigma(I - GA)^\top + G\Lambda G^\top + GQG^\top$



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 $M = M_L(M_L)^\top, B = B_L(B_L)^\top, C = C_L(C_L)^\top$:

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



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 $M = M_L(M_L)^\top, B = B_L(B_L)^\top, C = C_L(C_L)^\top$:

$$\begin{aligned}M &= ABA^\top + C, \\ \Leftrightarrow M_L(M_L)^\top &= AB_L(B_L)^\top A^\top + C_L(C_L)^\top\end{aligned}$$



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 $M = M_L(M_L)^\top, B = B_L(B_L)^\top, C = C_L(C_L)^\top$:

$$\begin{aligned} M &= ABA^\top + C, \\ \Leftrightarrow M_L(M_L)^\top &= AB_L(B_L)^\top A^\top + C_L(C_L)^\top = [AB_L \quad C_L] \cdot [AB_L \quad C_L]^\top \end{aligned}$$



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 $M = M_L(M_L)^\top, B = B_L(B_L)^\top, C = C_L(C_L)^\top$:

$$\begin{aligned} M &= ABA^\top + C, \\ \Leftrightarrow M_L(M_L)^\top &= AB_L(B_L)^\top A^\top + C_L(C_L)^\top = [AB_L \quad C_L] \cdot [AB_L \quad C_L]^\top \\ \text{doing QR} \left(\begin{matrix} AB_L & C_L \end{matrix}^\top \right) &\Leftrightarrow = R^\top Q^\top QR \end{aligned}$$



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 $M = M_L(M_L)^\top, B = B_L(B_L)^\top, C = C_L(C_L)^\top$:

$$\begin{aligned} M &= ABA^\top + C, \\ \Leftrightarrow M_L(M_L)^\top &= AB_L(B_L)^\top A^\top + C_L(C_L)^\top = [AB_L \quad C_L] \cdot [AB_L \quad C_L]^\top \\ \text{doing QR} \left(\begin{bmatrix} AB_L & C_L \end{bmatrix}^\top \right) &= R^\top Q^\top QR = R^\top R. \qquad \Rightarrow M_L := R^\top \end{aligned}$$



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 $M = M_L(M_L)^\top, B = B_L(B_L)^\top, C = C_L(C_L)^\top$:

$$\begin{aligned} M &= ABA^\top + C, \\ \Leftrightarrow M_L(M_L)^\top &= AB_L(B_L)^\top A^\top + C_L(C_L)^\top = [AB_L \quad C_L] \cdot [AB_L \quad C_L]^\top \\ \text{doing QR} \left(\begin{bmatrix} AB_L & C_L \end{bmatrix}^\top \right) &\Leftrightarrow = R^\top Q^\top QR = R^\top R. \Rightarrow M_L := R^\top \end{aligned}$$

⇒ PREDICT/UPDATE/SMOOTH can be formulated directly on square-roots to preserve PSD-ness!



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 $M = M_L(M_L)^\top, B = B_L(B_L)^\top, C = C_L(C_L)^\top$:

$$\begin{aligned} M &= ABA^\top + C, \\ \Leftrightarrow M_L(M_L)^\top &= AB_L(B_L)^\top A^\top + C_L(C_L)^\top = [AB_L \quad C_L] \cdot [AB_L \quad C_L]^\top \\ \text{doing QR} \left(\begin{bmatrix} AB_L & C_L \end{bmatrix}^\top \right) &\Leftrightarrow = R^\top Q^\top QR = R^\top R. \Rightarrow M_L := R^\top \end{aligned}$$

⇒ PREDICT/UPDATE/SMOOTH can be formulated directly on square-roots to preserve PSD-ness!

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



DEMO TIME: Solving on extremely small step sizes with square-root filtering

demo.jl



Intermediate summary

- ▶ *ODE solving is state estimation*
- ▶ *We can estimate ODE solutions with extended Kalman filtering/smoothing, in a stable and calibrated way*

Next: **Extending ODE filters**

1. *Flexible information operators*: The ODE filter formulation extends to other numerical problems
2. *Latent force inference*: Joint GP regression on both ODEs and data



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

[Bosch et al., 2022]

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

$$\dot{x}(t) = f(x(t), t), \quad x(0) = x_0.$$

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

Initial distribution: $X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$

Prior / dynamics model: $X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$

ODE likelihood: $Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right), \quad z_i \triangleq 0$

Initial value likelihood: $Z^{\text{init}} | X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$

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



[Bosch et al., 2022]

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

$$\ddot{x}(t) = f(\dot{x}(t), x(t), t), \quad x(0) = x_0, \quad \dot{x}(0) = \dot{x}_0.$$

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

Initial distribution: $X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$

Prior / dynamics model: $X(t+h) \mid X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$

ODE likelihood: $Z(t_i) \mid X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right), \quad z_i \triangleq 0$

Initial value likelihood: $Z^{\text{init}} \mid X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$



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

[Bosch et al., 2022]

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

$$\ddot{x}(t) = f(\dot{x}(t), x(t), t), \quad x(0) = x_0, \quad \dot{x}(0) = \dot{x}_0.$$

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

Initial distribution: $X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$

Prior / dynamics model: $X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$

ODE likelihood: $Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(2)}(t_i) - f(X^{(1)}(t_i), X^{(0)}(t_i), t_i)\right), \quad z_i \triangleq 0$

Initial value likelihood: $Z^{\text{init}} | X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$

Initial derivative likelihood: $Z_1^{\text{init}} | X(0) \sim \delta\left(Z_1^{\text{init}}; X^{(1)}(0)\right), \quad z_1^{\text{init}} \triangleq \dot{x}_0$



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

[Bosch et al., 2022]

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

$$M\dot{x}(t) = f(x(t), t), \quad x(0) = x_0.$$

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

Initial distribution: $X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$

Prior / dynamics model: $X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$

ODE likelihood: $Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right), \quad z_i \triangleq 0$

Initial value likelihood: $Z^{\text{init}} | X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$

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



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

$$M\dot{x}(t) = f(x(t), t), \quad x(0) = x_0.$$

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

Initial distribution: $X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$

Prior / dynamics model: $X(t+h) \mid X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$

DAE likelihood: $Z(t_i) \mid X(t_i) \sim \delta\left(Z(t_i); M\dot{X}^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right), \quad z_i \triangleq 0$

Initial value likelihood: $Z^{\text{init}} \mid X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$

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



[Bosch et al., 2022]

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

$$\dot{x}(t) = f(x(t), t), \quad x(0) = x_0, \quad g(x(t), \dot{x}(t)) = 0.$$

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

Initial distribution: $X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$

Prior / dynamics model: $X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$

ODE likelihood: $Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right), \quad z_i \triangleq 0$

Initial value likelihood: $Z^{\text{init}} | X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$



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

[Bosch et al., 2022]

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

$$\dot{x}(t) = f(x(t), t), \quad x(0) = x_0, \quad g(x(t), \dot{x}(t)) = 0.$$

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

Initial distribution: $X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$

Prior / dynamics model: $X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$

ODE likelihood: $Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right), \quad z_i \triangleq 0$

Conservation law likelihood: $Z_i^c(t_i) | X(t_i) \sim \delta\left(Z_i^c(t_i); g(X^{(0)}(t_i), X^{(1)}(t_i))\right), \quad z_i^c \triangleq 0$

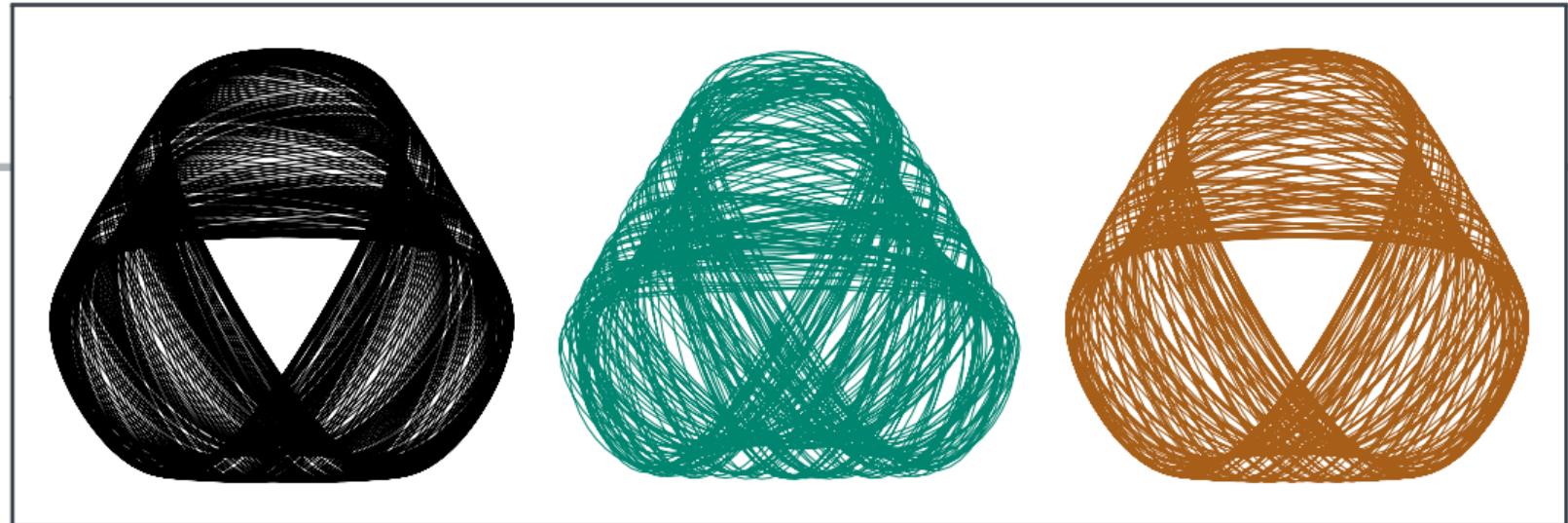
Initial value likelihood: $Z^{\text{init}} | X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$



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

[Bosch et al., 2022]

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



Initial value likelihood:

$$Z^{\text{init}} | X(0) \sim \delta(Z^{\text{init}}; X^{(0)}(0)),$$

$$z^{\text{init}} \triangleq x_0$$



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

[Bosch et al., 2022]

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

$$\dot{x}(t) = f(x(t), t), \quad x(0) = x_0, \quad g(x(t), \dot{x}(t)) = 0.$$

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

Initial distribution: $X(0) \sim \mathcal{N}(X(0); \mu_0, \Sigma_0)$

Prior / dynamics model: $X(t+h) | X(t) \sim \mathcal{N}(X(t+h); A(h)X(t), Q(h))$

ODE likelihood: $Z(t_i) | X(t_i) \sim \delta\left(Z(t_i); X^{(1)}(t_i) - f(X^{(0)}(t_i), t_i)\right), \quad z_i \triangleq 0$

Conservation law likelihood: $Z_i^c(t_i) | X(t_i) \sim \delta\left(Z_i^c(t_i); g(X^{(0)}(t_i), X^{(1)}(t_i))\right), \quad z_i^c \triangleq 0$

Initial value likelihood: $Z^{\text{init}} | X(0) \sim \delta\left(Z^{\text{init}}; X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$

The measurement model provides a very flexible way to easily encode desired properties!



DEMO TIME: Solving a second-order ODE

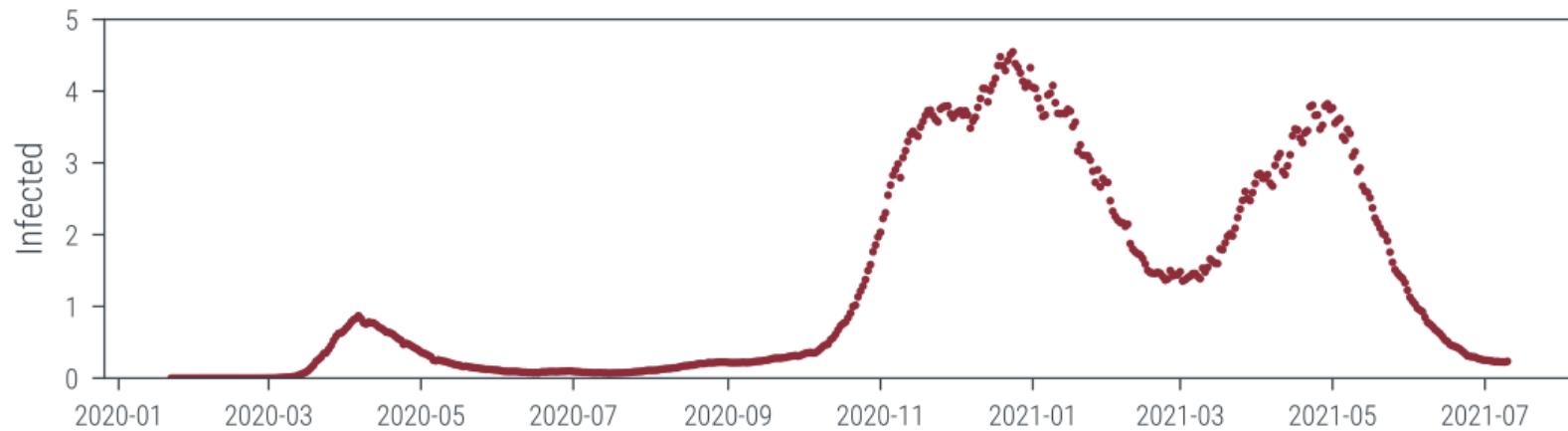
demo.jl



Next: **Combine ODEs and GP regression via *latent force inference***

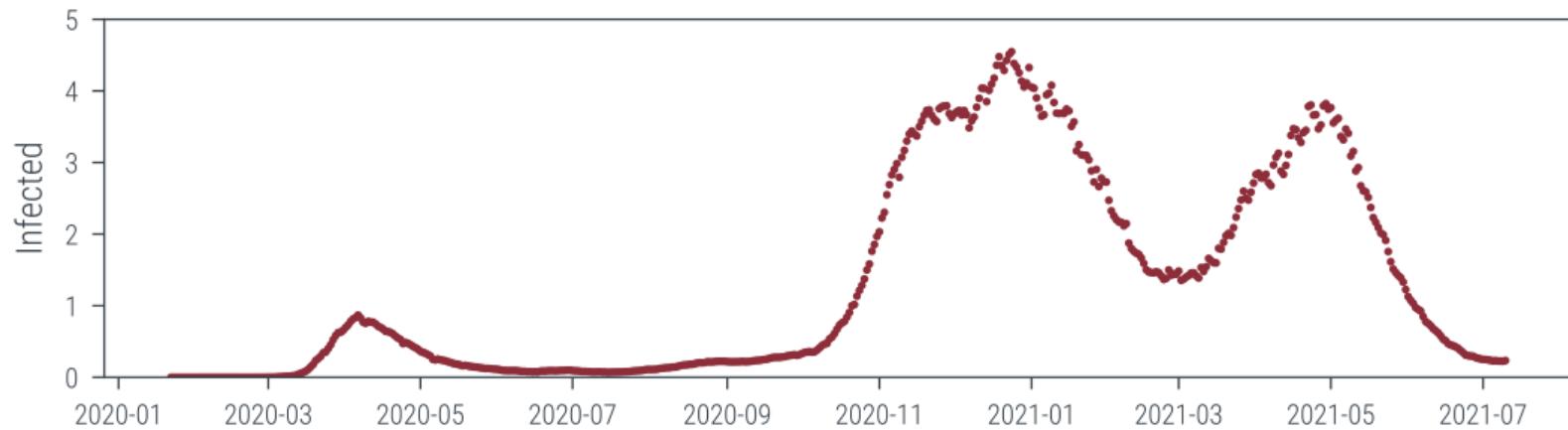
Latent force inference: GP regression on both ODEs and data

An example we know all too well: COVID-19



Latent force inference: GP regression on both ODEs and data

An example we know all too well: COVID-19

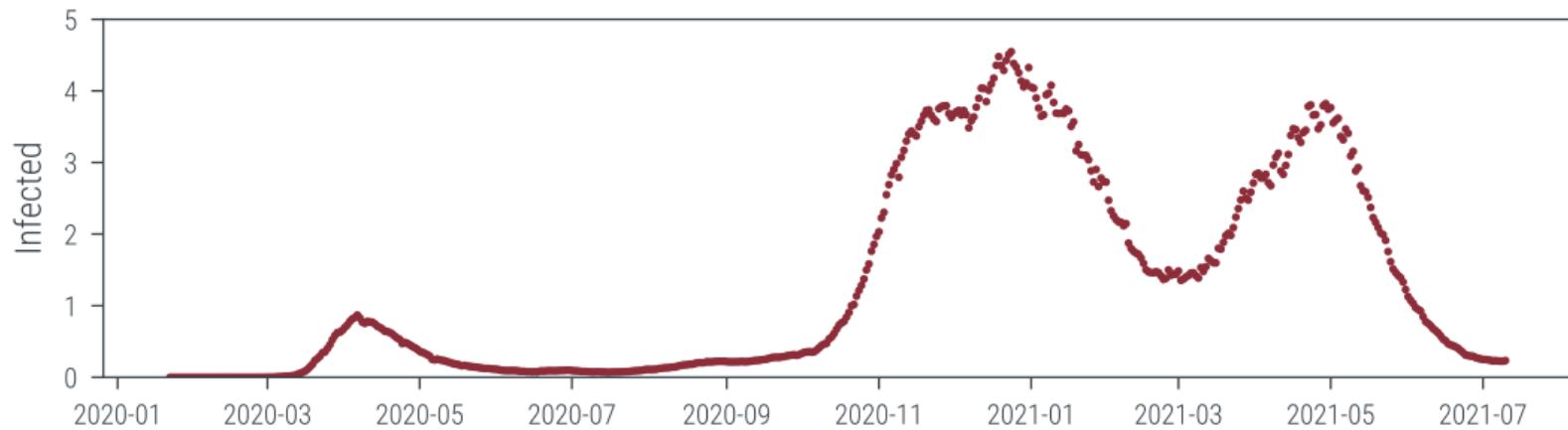


ODE dynamics:

$$\frac{d}{dt} \begin{bmatrix} S(t) \\ I(t) \\ R(t) \\ D(t) \end{bmatrix} = \begin{bmatrix} -\beta \cdot S(t)I(t)/P \\ \beta \cdot S(t)I(t)/P - \gamma I(t) - \eta I(t) \\ \gamma I(t) \\ \eta I(t) \end{bmatrix}$$

Latent force inference: GP regression on both ODEs and data

An example we know all too well: COVID-19

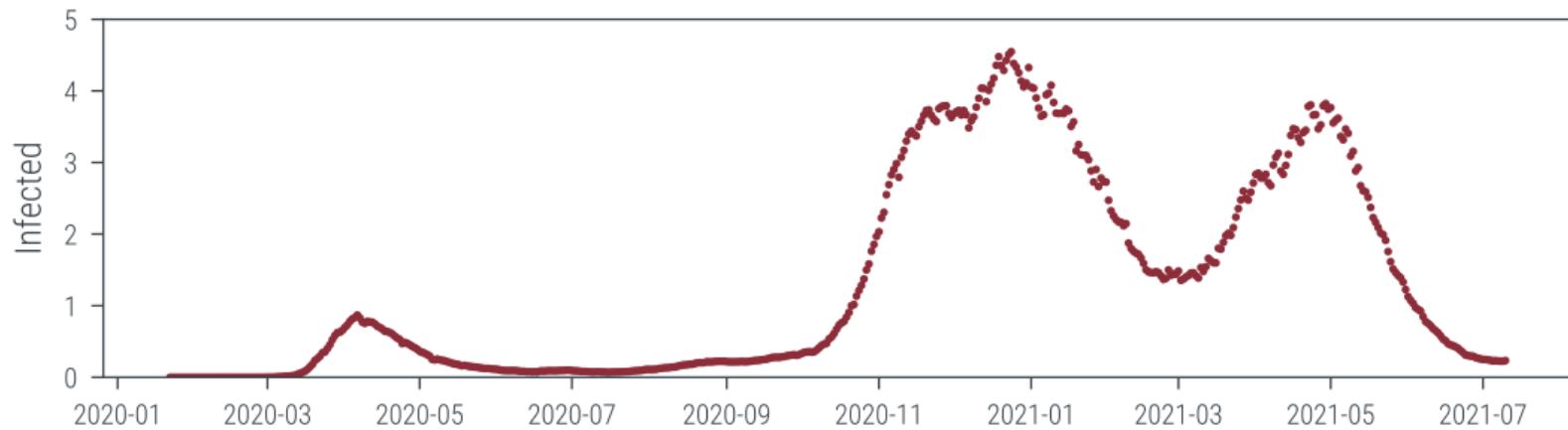


ODE dynamics with time-varying contact rate:

$$\frac{d}{dt} \begin{bmatrix} S(t) \\ I(t) \\ R(t) \\ D(t) \end{bmatrix} = \begin{bmatrix} -\beta(t) \cdot S(t)I(t)/P \\ \beta(t) \cdot S(t)I(t)/P - \gamma I(t) - \eta I(t) \\ \gamma I(t) \\ \eta I(t) \end{bmatrix}$$

Latent force inference: GP regression on both ODEs and data

An example we know all too well: COVID-19



ODE dynamics with time-varying contact rate:

$$\frac{d}{dt} \begin{bmatrix} S(t) \\ I(t) \\ R(t) \\ D(t) \end{bmatrix} = \begin{bmatrix} -\beta(t) \cdot S(t)I(t)/P \\ \beta(t) \cdot S(t)I(t)/P - \gamma I(t) - \eta I(t) \\ \gamma I(t) \\ \eta I(t) \end{bmatrix}$$

Latent force model: Gauss–Markov prior

$$\beta(t+h) \mid \beta(t) \sim \mathcal{N}(A_\beta(h)\beta(t), Q_\beta(h))$$

Data:

$$y_i \mid x(t_i) \sim \mathcal{N}(Hx(t_i), \sigma^2 I)$$



Latent force inference: GP regression on both ODEs and data

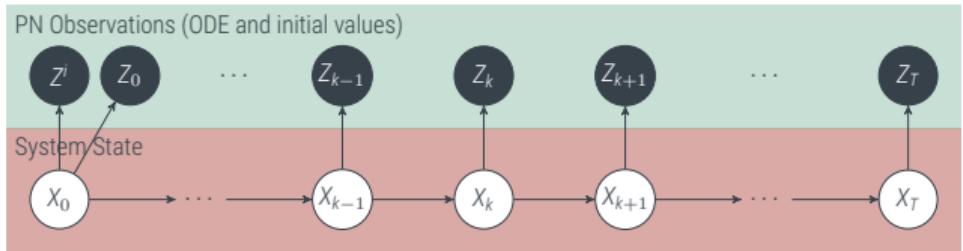
Once again we can just build a custom state-space model for the problem setup of interest

Paper: [Schmidt et al., 2021]

Initial value problem:

$$\dot{x}(t) = f(x(t), t), \quad x(0) = x_0.$$

ODE filter setup:





Latent force inference: GP regression on both ODEs and data

Once again we can just build a custom state-space model for the problem setup of interest

Paper: [Schmidt et al., 2021]

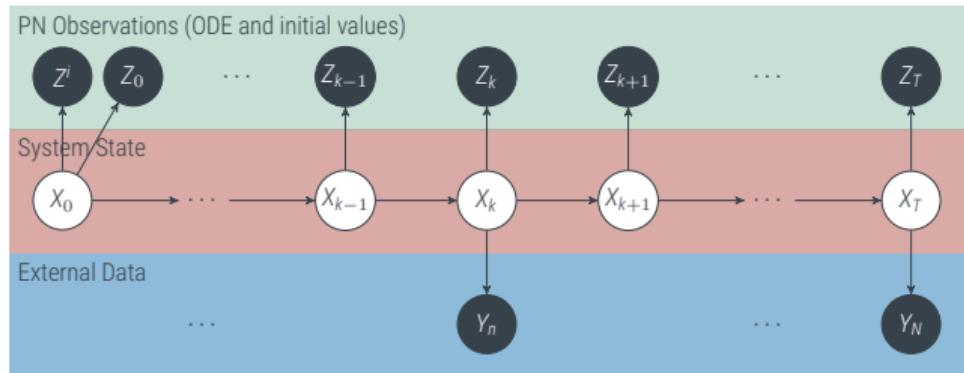
Initial value problem:

$$\dot{x}(t) = f(x(t), t), \quad x(0) = x_0.$$

External observations / data:

$$y_i = Hx(t_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

ODE filter setup:





Latent force inference: GP regression on both ODEs and data

Once again we can just build a custom state-space model for the problem setup of interest

Paper: [Schmidt et al., 2021]

Initial value problem:

$$\dot{x}(t) = f(x(t), \beta(t), t), \quad x(0) = x_0.$$

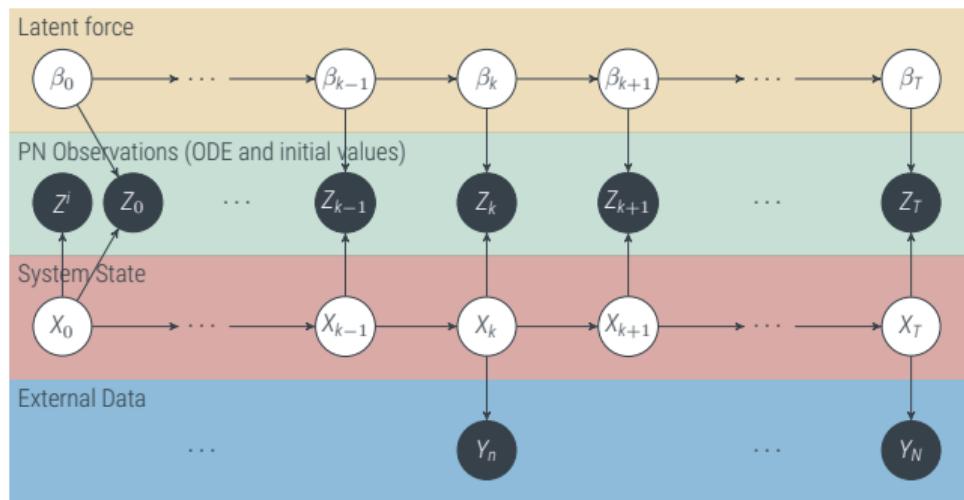
External observations / data:

$$y_i = Hx(t_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Latent Gauss–Markov process:

$$\beta(t+h) \mid \beta(t) \sim \mathcal{N} (A_\beta(h)\beta(t), \sigma_\beta^2 Q_\beta(h)).$$

ODE filter setup:





Latent force inference: GP regression on both ODEs and data

Once again we can just build a custom state-space model for the problem setup of interest

Paper: [Schmidt et al., 2021]

Initial value problem:

$$\dot{x}(t) = f(x(t), \beta(t), t), \quad x(0) = x_0.$$

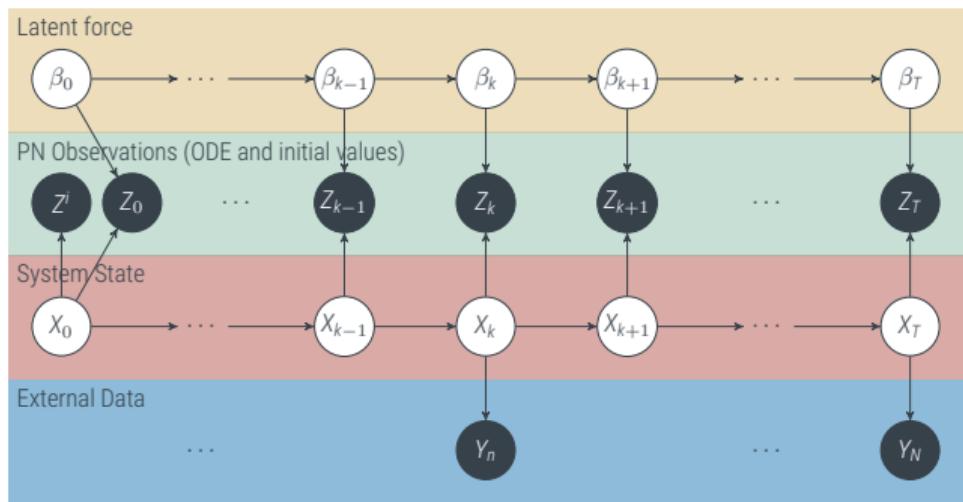
External observations / data:

$$y_i = Hx(t_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Latent Gauss–Markov process:

$$\beta(t+h) \mid \beta(t) \sim \mathcal{N} (A_\beta(h)\beta(t), \sigma_\beta^2 Q_\beta(h)).$$

ODE filter setup:



Again: **This is just state-space model**



Latent force inference: GP regression on both ODEs and data

Once again we can just build a custom state-space model for the problem setup of interest

Paper: [Schmidt et al., 2021]

Initial value problem:

$$\dot{x}(t) = f(x(t), \beta(t), t), \quad x(0) = x_0.$$

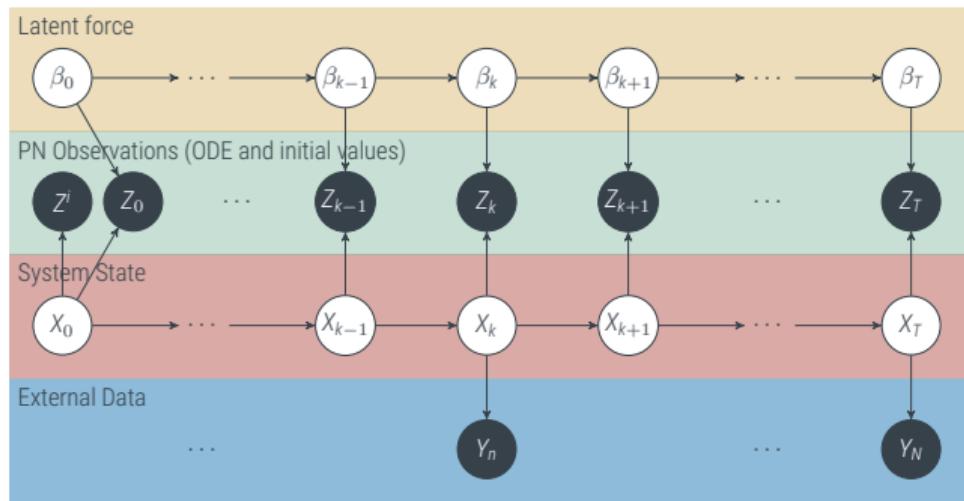
External observations / data:

$$y_i = Hx(t_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Latent Gauss–Markov process:

$$\beta(t+h) \mid \beta(t) \sim \mathcal{N} (A_\beta(h)\beta(t), \sigma_\beta^2 Q_\beta(h)).$$

ODE filter setup:



Again: **This is just state-space model \Rightarrow inference with EKF/EKS!**



Formally we obtain the **probabilistic state estimation problem**:

State initial distribution: $X(0) \sim \mathcal{N}(\mu_0, \Sigma_0)$

State dynamics: $X(t + h) | X(t) \sim \mathcal{N}(A(h)X(t), Q(h))$

Latent force initial distribution: $\beta(0) \sim \mathcal{N}(\mu_0^\beta, \Sigma_0^\beta)$

Latent force dynamics: $\beta(t + h) | \beta(t) \sim \mathcal{N}(A_\beta(h)\beta(t), Q_\beta(h))$

ODE likelihood: $Z(t_i) | X(t_i), \beta(t_i) \sim \delta\left(X^{(1)}(t_i) - f(X^{(0)}(t_i), \beta(t_i), t_i)\right), \quad z_i \triangleq 0$

Initial value likelihood: $Z^{\text{init}} | X(0) \sim \delta\left(X^{(0)}(0)\right), \quad z^{\text{init}} \triangleq x_0$

Data likelihood: $Y_i | X(t_i) \sim \mathcal{N}\left(HX^{(0)}(t_i), \sigma^2 I\right), \quad y_i \in \mathcal{D}_y$

Latent force inference: Writing down the state estimation problem



Formally we obtain the probabilistic state estimation problem, *simplified by stacking* $\tilde{X} = [X, \beta]$:

Initial distribution: $\tilde{X}(0) \sim \mathcal{N}(\tilde{\mu}_0, \tilde{\Sigma}_0)$

Prior / dynamics model: $\tilde{X}(t+h) | \tilde{X}(t) \sim \mathcal{N}(\tilde{A}(h)\tilde{X}(t), \tilde{Q}(h))$

ODE likelihood: $Z(t_i) | \tilde{X}(t_i) \sim \delta(E_1\tilde{X}(t_i) - f(E_0\tilde{X}(t_i), E_\beta\tilde{X}(t_i), t_i)), \quad z_i \triangleq 0$

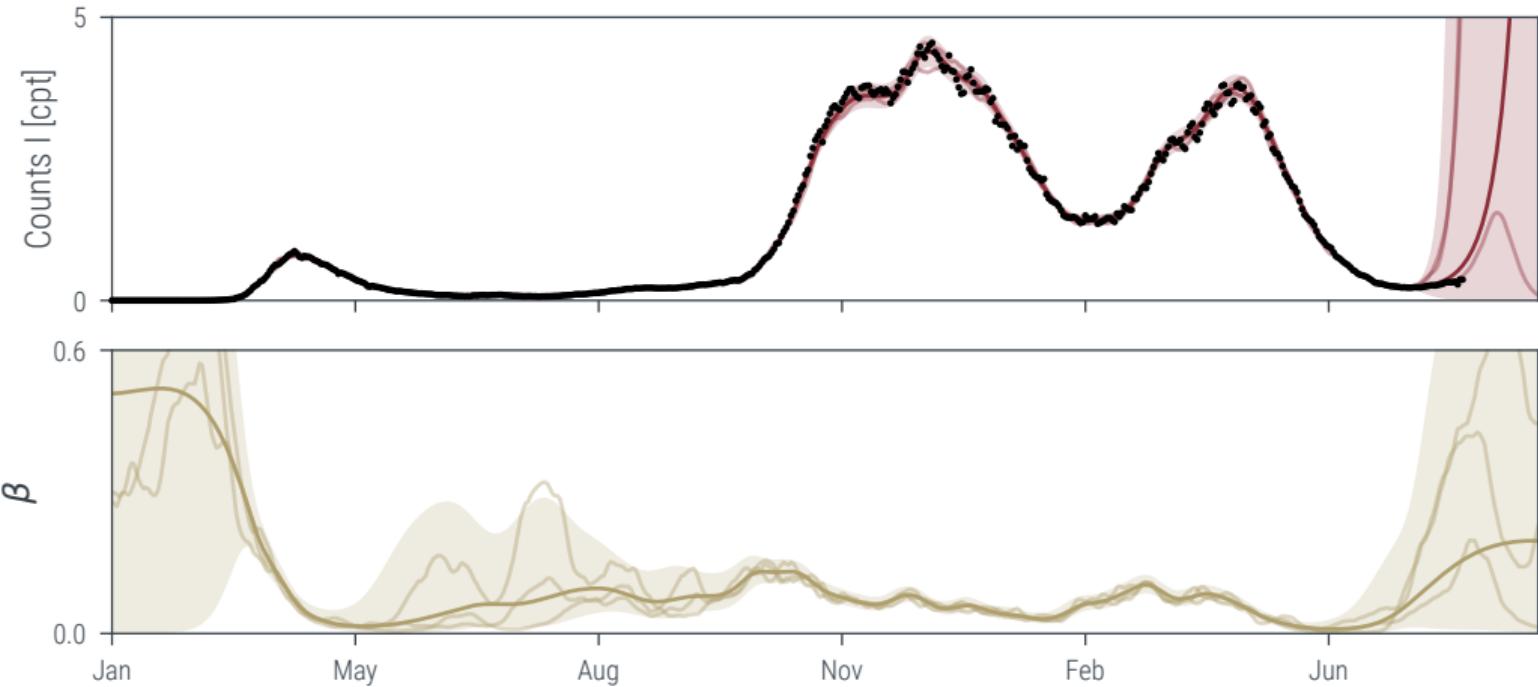
Initial value likelihood: $Z^{\text{init}} | \tilde{X}(0) \sim \delta(E_0\tilde{X}(0)), \quad z^{\text{init}} \triangleq x_0$

Data likelihood: $Y_i | \tilde{X}(t_i) \sim \mathcal{N}(HE_0\tilde{X}(t_i), \sigma^2 I), \quad y_i \in \mathcal{D}_y$

with $E_0\tilde{X} := X^{(0)}$, $E_1\tilde{X} := X^{(1)}$, $E_\beta\tilde{X} := \beta$.

Latent force inference: Results

Posteriors over infections and contact rates in a single forward-backward pass





Outlook



Probabilistic Numerics: Computation as Machine Learning

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



Probabilistic Numerics: Computation as Machine Learning

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

References for topics not covered today:

- ▶ ODE filter theory and details:
 - ▶ Convergence rates: [Kersting et al., 2020, Tronarp et al., 2021]
 - ▶ Other filtering algorithms (e.g. IEKS and particle filter): [Tronarp et al., 2019, Tronarp et al., 2021]
 - ▶ Step-size adaptation and more calibration: [Bosch et al., 2021]
 - ▶ Scaling ODE filters to high dimensions: [Krämer et al., 2022]
- ▶ More related differential equation problems:
 - ▶ Boundary value problems (BVPs): [Krämer and Hennig, 2021]
 - ▶ Partial differential equations (PDEs): [Krämer et al., 2022]
- ▶ Inverse problems
 - ▶ Parameter inference in ODEs with ODE filters: [Tronarp et al., 2022]
 - ▶ Efficient latent force inference: [Schmidt et al., 2021]



Come to the Probabilistic Numerics Spring School!

Register at <https://probnumschool.org/>

Probabilistic Numerics Spring School *Tübingen 2023*

March 27 - 29, 2023



Summary

- ▶ *ODE solving is state estimation*
⇒ treat initial value problems as state estimation problems
- ▶ “*ODE filters*”: **How to solve ODEs with Bayesian filtering and smoothing**
- ▶ *Bells and whistles*: Uncertainty calibration & Square-root filtering
- ▶ *Flexible information operators* to solve more than just standard ODEs
- ▶ *Latent force inference*: Joint GP regression on both ODEs and data

Please give Feedback:



Software packages



<https://github.com/nathanaelbosch/ProbNumDiffEq.jl>
] add ProbNumDiffEq



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



<https://github.com/pnkraemer/tornadox>
pip install tornadox



Bibliography I

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



Bibliography II

- ▶ 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.
- ▶ 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.



Bibliography IV

- ▶ Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. (2019).
Probabilistic solutions to ordinary differential equations as nonlinear bayesian filtering: a new perspective.
Stat. Comput., 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.