What We Talk About

When We Talk About

Probabilistic Numerics

LAI Reading Group
18 March 2026

Rui-Yang Zhang

Why?

Why?

- I love probabilities

Why?

- I love probabilities
- and I like numerical algorithms

Why?

- I love probabilities
- and I like numerical algorithms
- so I should also like probability + numerics … ?

Why?

- I love probabilities
- and I like numerical algorithms
- so I should also like probability + numerics … ?

Why?

- I love probabilities
- and I like numerical algorithms
- so I should also like probability + numerics … ?

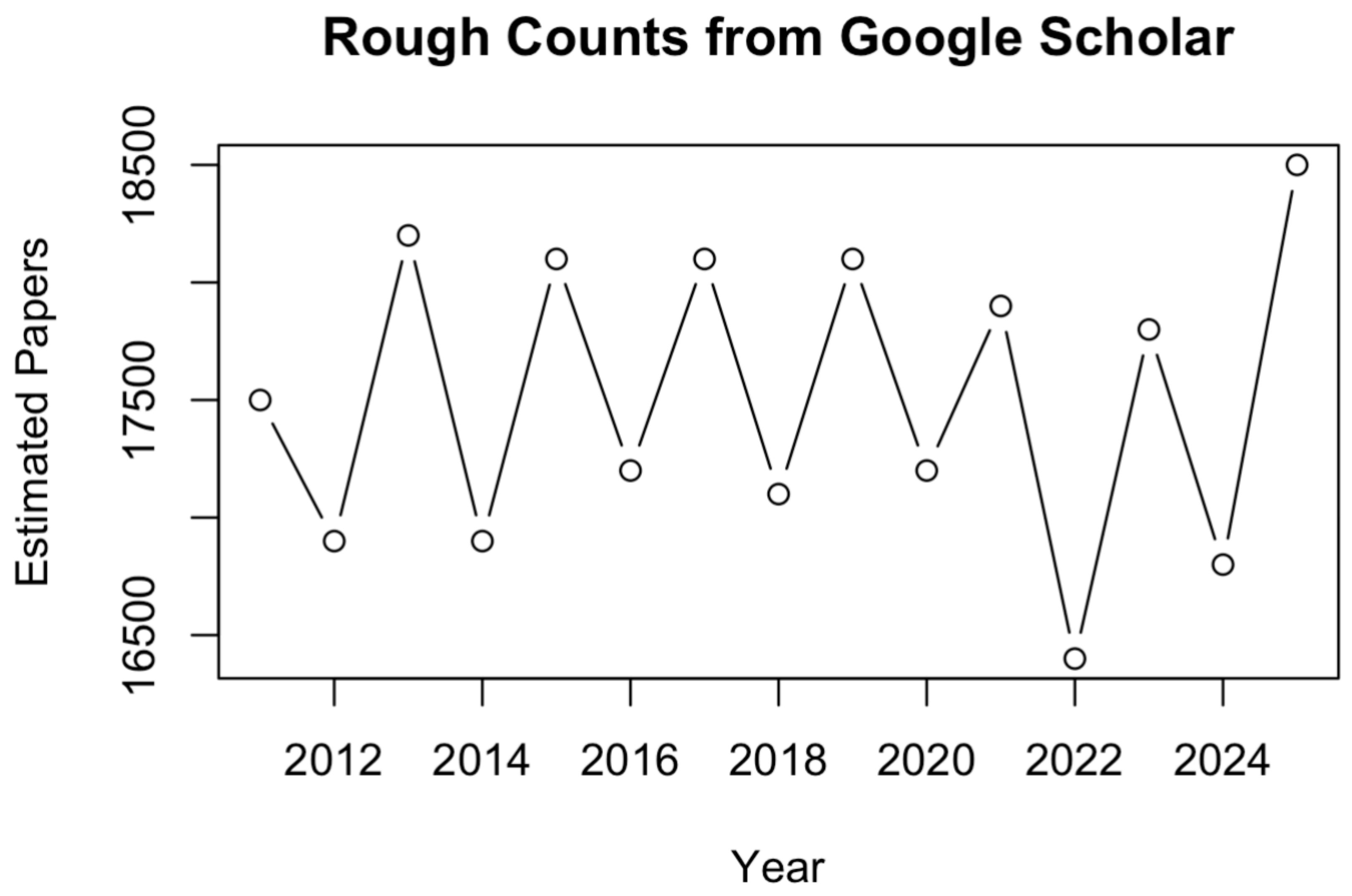
- also was looking for new research ideas / areas and this area appears to use the tools I know

Why?

- I love probabilities
- and I like numerical algorithms
- so I should also like probability + numerics … ?

- also was looking for new research ideas / areas and this area appears to use the tools I know
- read some papers and had some idea, etc. etc.

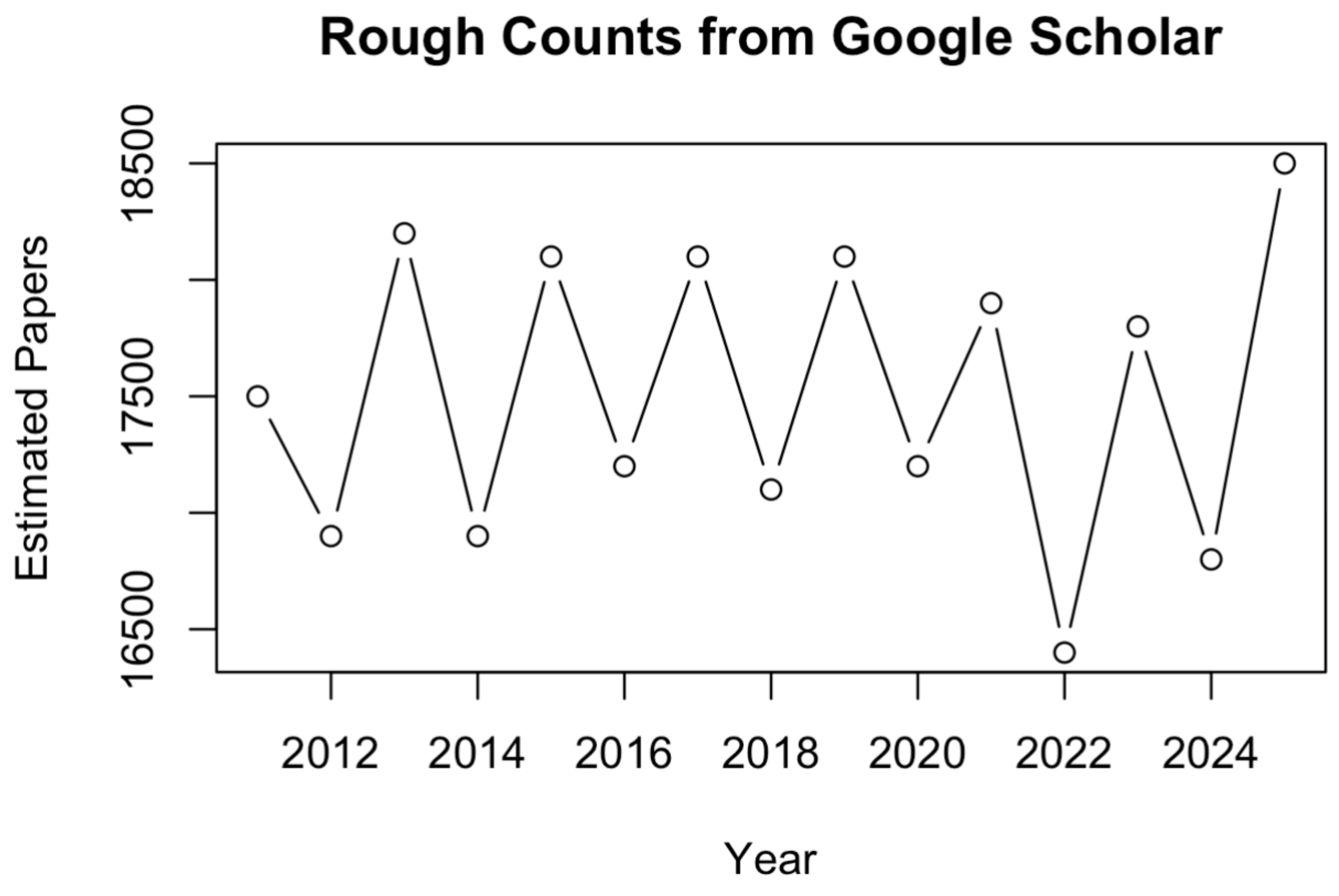
Why?



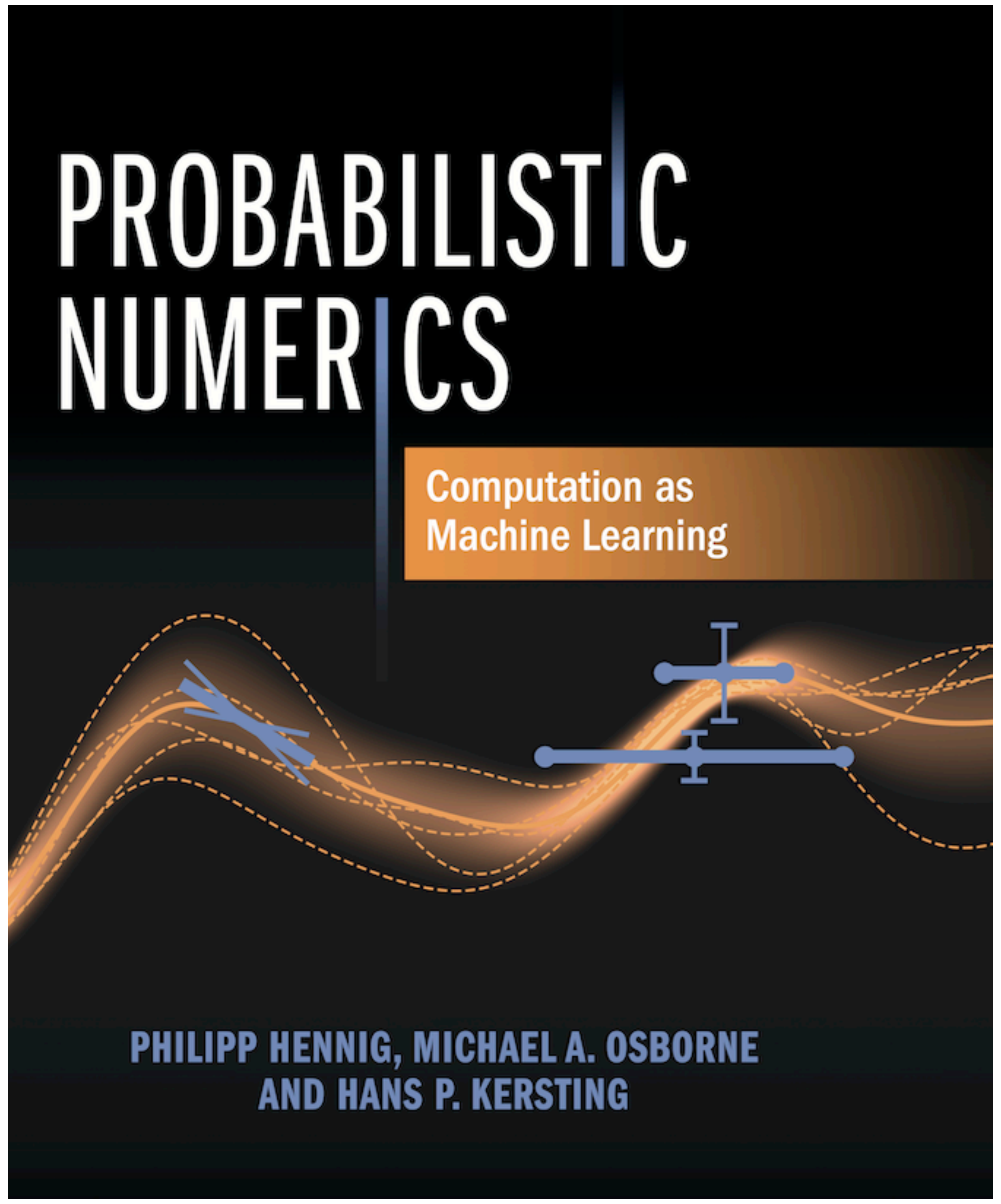
- I love probabilities
- and I like numerical algorithms
- so I should also like probability + numerics … ?

- also was looking for new research ideas / areas and this area appears to use the tools I know
- read some papers and had some idea, etc. etc.

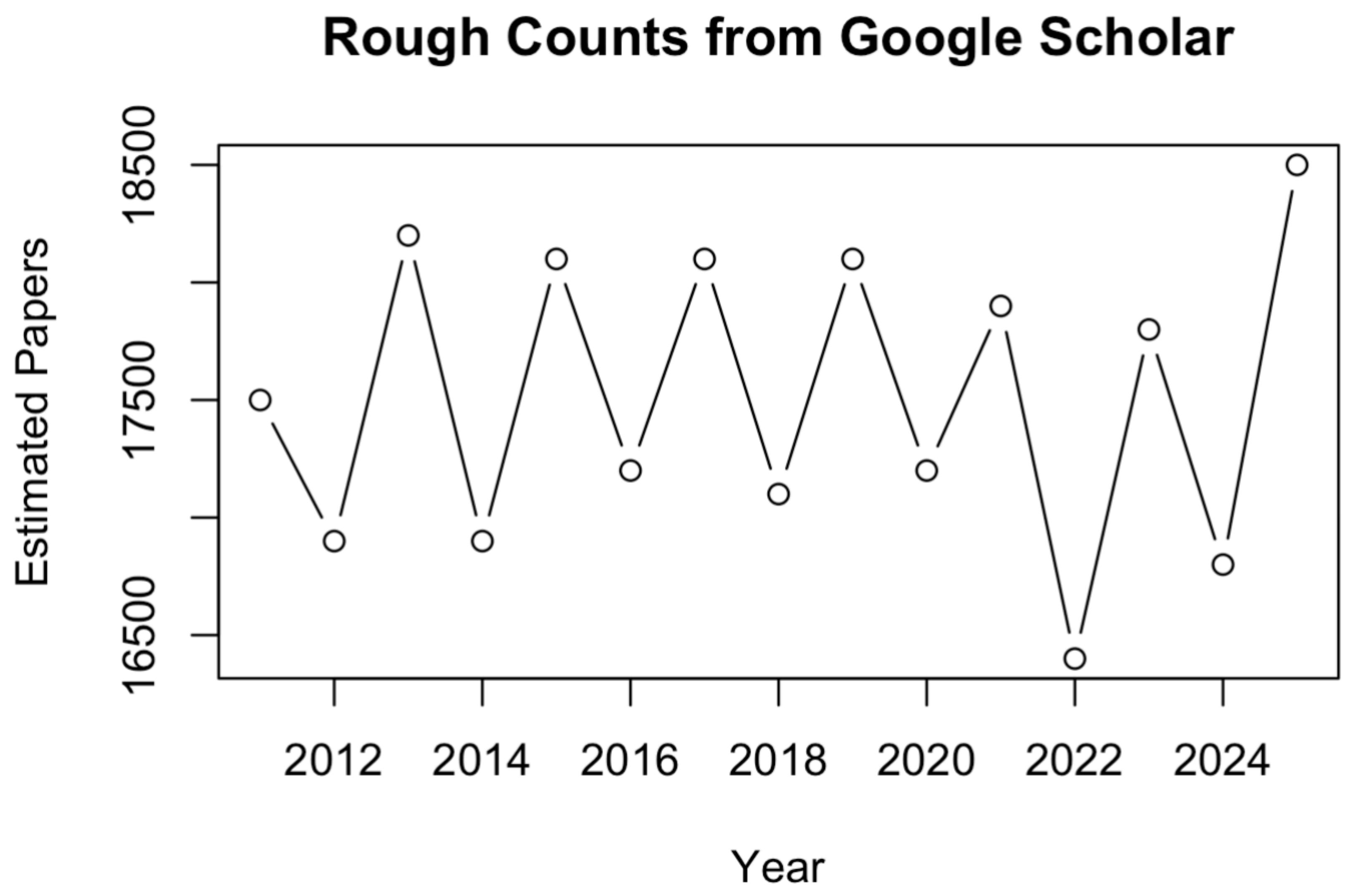
Why?



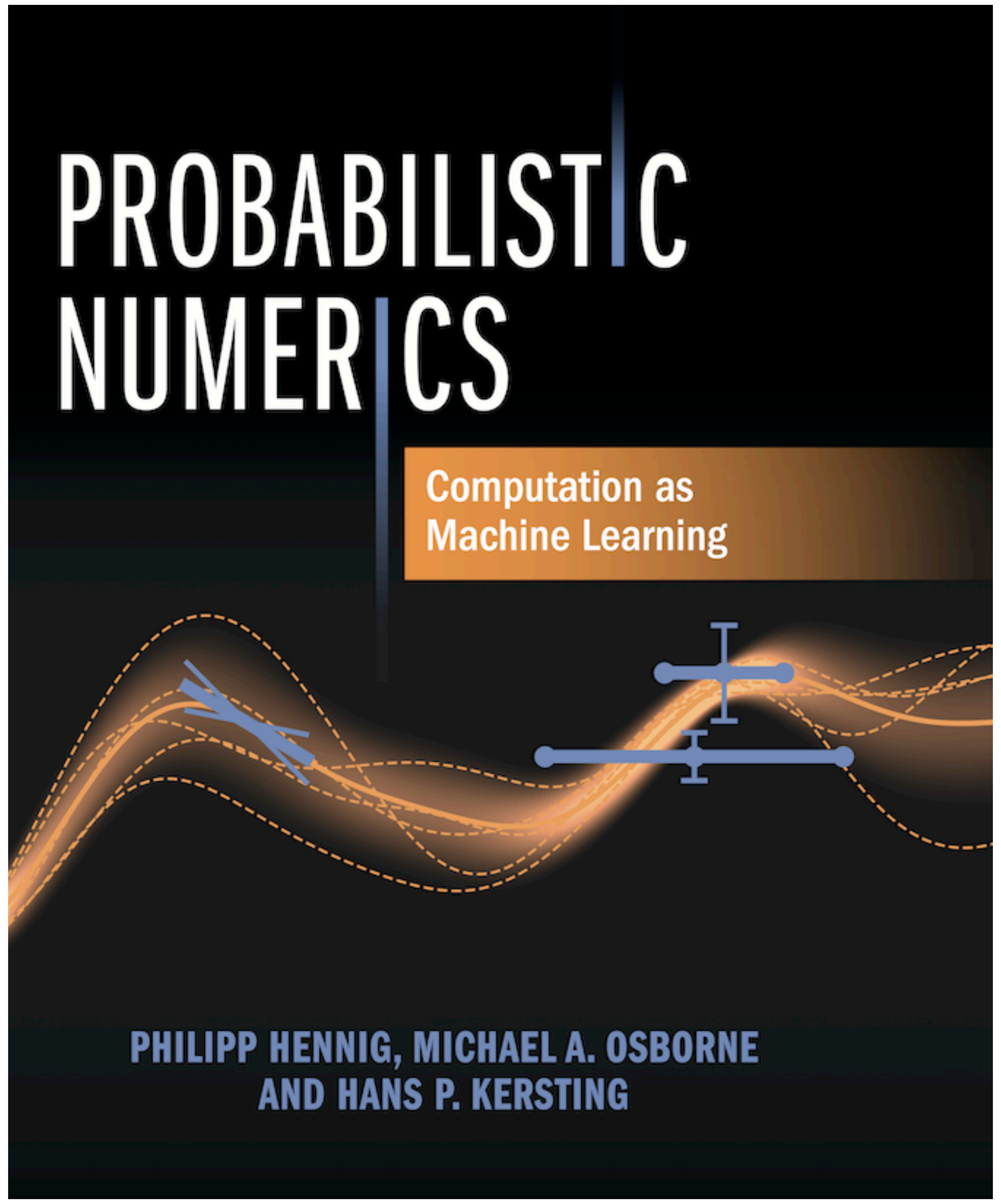
- I love probabilities
- and I like numerical algorithms
- so I should also like probability + numerics … ?
- also was looking for new research ideas / areas and this area appears to use the tools I know
- read some papers and had some idea, etc. etc.



Why?



- I love probabilities
- and I like numerical algorithms
- so I should also like probability + numerics … ?



- also was looking for new research ideas / areas and this area appears to use the tools I know
- read some papers and had some idea, etc. etc.

So, what is *Probabilistic Numerics* ?

So, what is *Probabilistic Numerics* ?

PROCEEDINGS A

rspa.royalsocietypublishing.org

Research

CrossMark
click for updates

Cite this article: Hennig P, Osborne MA, Girolami M. 2015 Probabilistic numerics and uncertainty in computations. *Proc. R. Soc. A* **471**: 20150142.
http://dx.doi.org/10.1098/rspa.2015.0142

Probabilistic numerics and uncertainty in computations

Philipp Hennig[1], Michael A. Osborne[2] and Mark Girolami[3]

[1]Department of Empirical Inference, Max Planck Institute for Intelligent Systems, Tübingen, Germany
[2]Department of Engineering Science, University of Oxford, Oxford, UK
[3]Department of Statistics, University of Warwick, Warwick, UK

Algorithms for numerical tasks, including linear algebra, integration, optimization and solving differential equations, that return uncertainties in their calculations.

So, what is *Probabilistic Numerics* ?

PROCEEDINGS A

rspa.royalsocietypublishing.org

Research

CrossMark
click for updates

Cite this article: Hennig P, Osborne MA, Girolami M. 2015 Probabilistic numerics and uncertainty in computations. *Proc. R. Soc. A* **471**: 20150142.
http://dx.doi.org/10.1098/rspa.2015.0142

Probabilistic numerics and uncertainty in computations

Philipp Hennig[1], Michael A. Osborne[2] and Mark Girolami[3]

[1]Department of Empirical Inference, Max Planck Institute for Intelligent Systems, Tübingen, Germany
[2]Department of Engineering Science, University of Oxford, Oxford, UK
[3]Department of Statistics, University of Warwick, Warwick, UK

Algorithms for numerical tasks, including linear algebra, integration, optimization and solving differential equations, that return uncertainties in their calculations.

Home > Statistics and Computing > Article

A modern retrospective on probabilistic numerics

Open access | Published: 04 October 2019
Volume 29, pages 1335–1351, (2019) Cite this article

A *statistical* treatment of the errors and/or approximations that are made en route to the output of a deterministic numerical method.

ProbNum = Numerical Algorithms with Result UQ

ProbNum = Numerical Algorithms with Result UQ

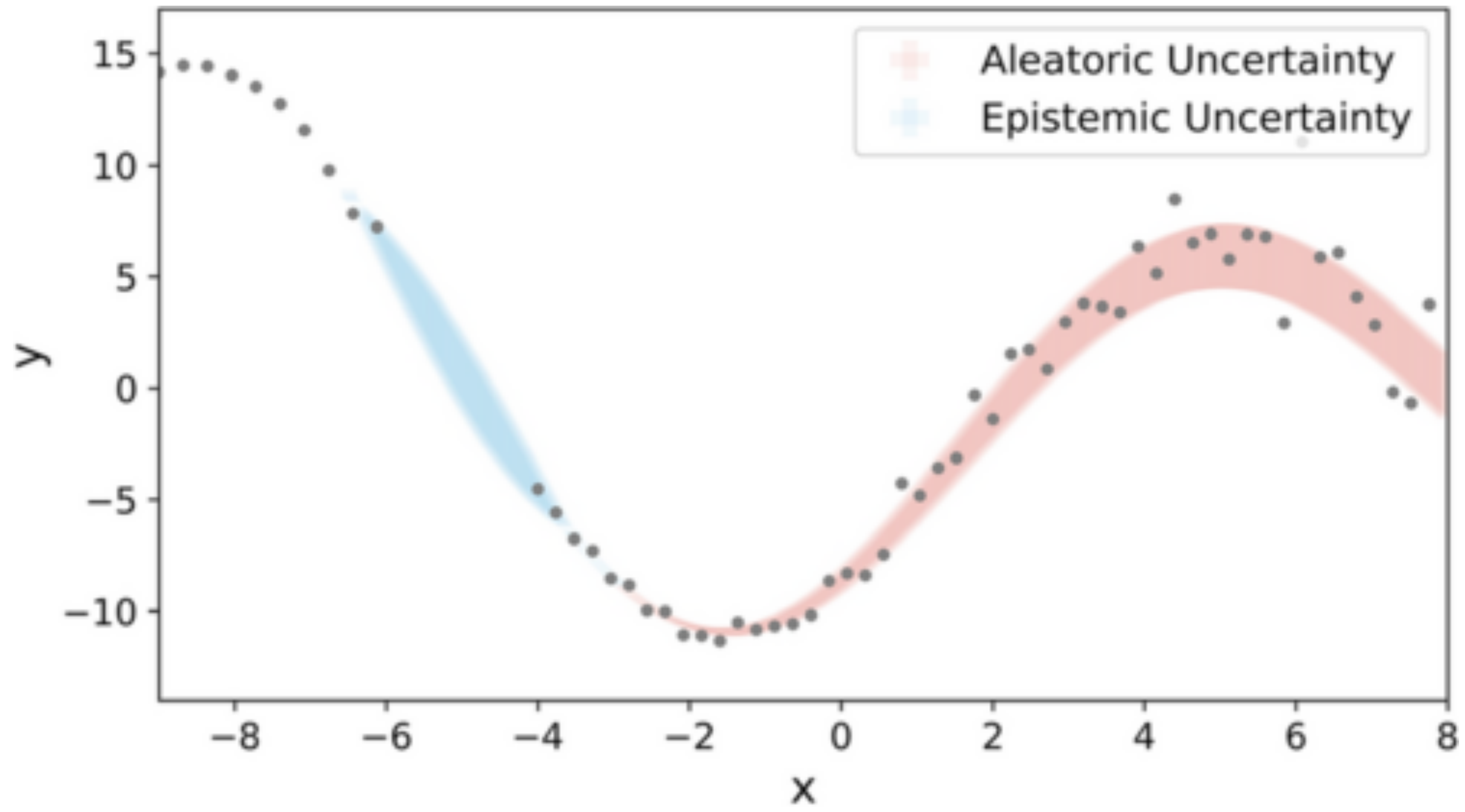
UQ sounds nice, but …

- What do these uncertainties mean?
    - Epistemic (Reducible) & Aleatoric (Irreducible)

ProbNum = Numerical Algorithms with Result UQ

UQ sounds nice, but …

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)



ProbNum = Numerical Algorithms with Result UQ

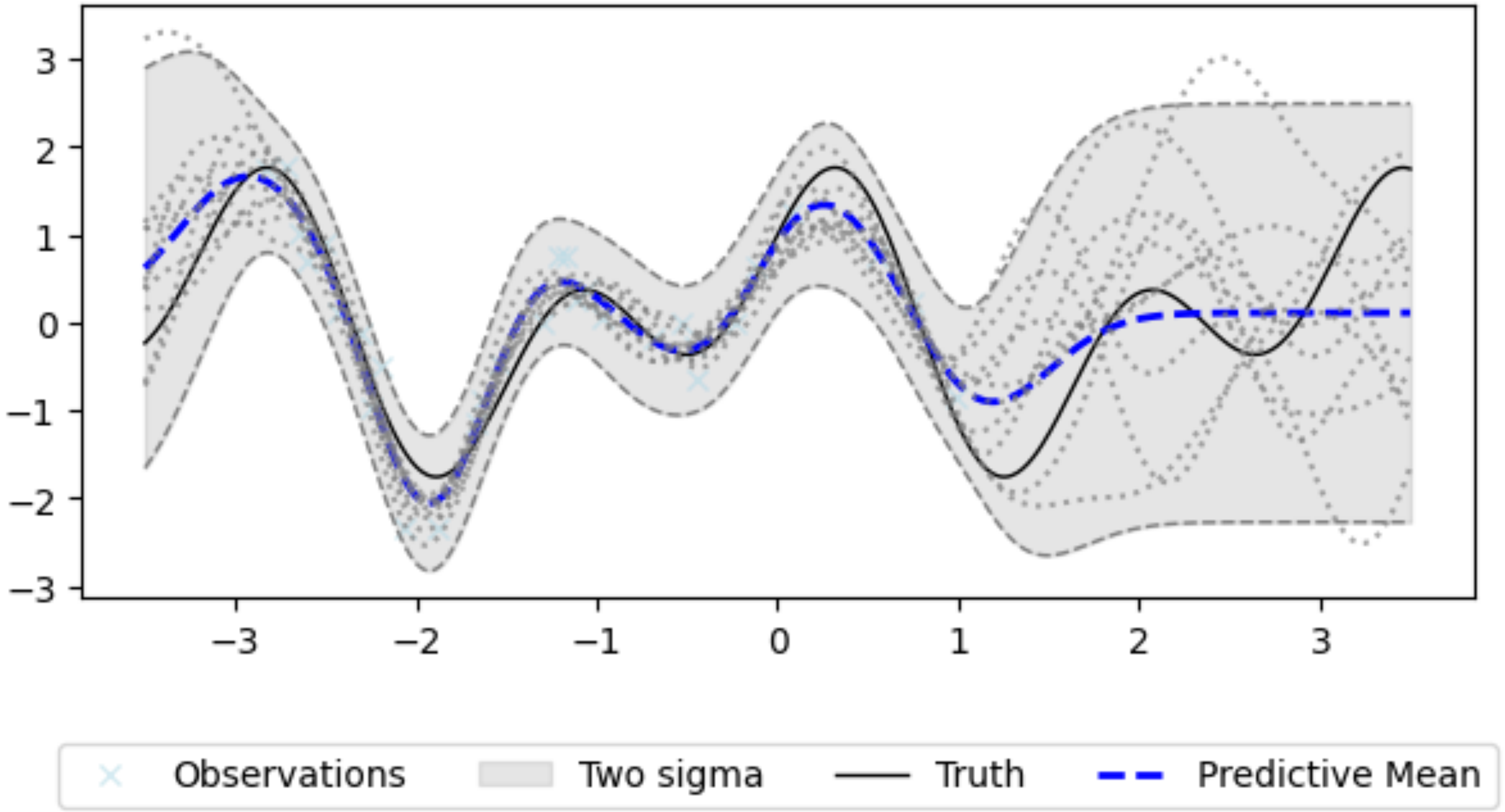
UQ sounds nice, but …

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples

ProbNum = Numerical Algorithms with Result UQ

UQ sounds nice, but …

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples



ProbNum = Numerical Algorithms with Result UQ

UQ sounds nice, but …

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
  - Sequential Data Acquisition

- Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
- Sequential Data Acquisition

ProbNum = Numerical Algorithms with Result UQ

UQ sounds nice, but …

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
  - Sequential Data Acquisition
- How much do we have to pay for these uncertainties?

Relevance of ProbNum for ML

Hennig (2023): "… (Probabilistic Numerics) never mattered more"

https://www.youtube.com/watch?v=0Q1ZTLHULcw

Relevance of ProbNum for ML

Hennig (2023): "… (Probabilistic Numerics) never mattered more"

Probabilistic Numerics Spring School 2023

https://www.youtube.com/watch?v=0Q1ZTLHULcw

Relevance of ProbNum for ML

Hennig (2023): "… (Probabilistic Numerics) never mattered more"

Probabilistic Numerics Spring School 2023

Three Ways *Probabilistic Numerics* Can Help **ML / AI**:

https://www.youtube.com/watch?v=0Q1ZTLHULcw

Relevance of ProbNum for ML

Hennig (2023): "… (Probabilistic Numerics) never mattered more"

Probabilistic Numerics Spring School 2023

Three Ways *Probabilistic Numerics* Can Help **ML / AI**:

- Cohesive synergy of mechanistic (physics) and empirical (data) information

https://www.youtube.com/watch?v=0Q1ZTLHULcw

Relevance of ProbNum for ML

Hennig (2023): "… (Probabilistic Numerics) never mattered more"

Probabilistic Numerics Spring School 2023

Three Ways *Probabilistic Numerics* Can Help **ML / AI**:

- Cohesive synergy of mechanistic (physics) and empirical (data) information
- NN with uncertainties

https://www.youtube.com/watch?v=0Q1ZTLHULcw

Relevance of ProbNum for ML

Hennig (2023): "… (Probabilistic Numerics) never mattered more"

Probabilistic Numerics Spring School 2023

Three Ways *Probabilistic Numerics* Can Help **ML / AI**:

- Cohesive synergy of mechanistic (physics) and empirical (data) information
- NN with uncertainties
- Smarter NN training

https://www.youtube.com/watch?v=0Q1ZTLHULcw

Relevance of ProbNum for ML

Hennig (2023): “… (Probabilistic Numerics) never mattered more”

Probabilistic Numerics Spring School 2023

Three Ways *Probabilistic Numerics* Can Help **ML / AI**:

- Cohesive synergy of mechanistic (physics) and empirical (data) information
- NN with uncertainties
- Smarter NN training

https://www.youtube.com/watch?v=0Q1ZTLHULcw

Relevance of ProbNum for ML

Hennig (2023): "… (Probabilistic Numerics) never mattered more"

Probabilistic Numerics Spring School 2023

Three Ways *Probabilistic Numerics* Can Help **ML / AI**:

- Cohesive synergy of mechanistic (physics) and empirical (data) information
- NN with uncertainties
- Smarter NN training

https://www.youtube.com/watch?v=0Q1ZTLHULcw

NN with Uncertainties

Being Bayesian, Even Just a Bit, Helps

**Being Bayesian, Even Just a Bit,
Fixes Overconfidence in ReLU Networks**

Agustinus Kristiadi [1] Matthias Hein [1] Philipp Hennig [1 2]

NN with Uncertainties

Being Bayesian, Even Just a Bit, Helps

Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks

Agustinus Kristiadi[1] **Matthias Hein**[1] **Philipp Hennig**[1,2]

- Bayesianize NN: don't have to have proper BNN, but NN with Bayes flavour

NN with Uncertainties

Being Bayesian, Even Just a Bit, Helps

**Being Bayesian, Even Just a Bit,
Fixes Overconfidence in ReLU Networks**

Agustinus Kristiadi [1] **Matthias Hein** [1] **Philipp Hennig** [1 2]

- Bayesianize NN: don't have to have proper BNN, but NN with Bayes flavour
- Laplace Approximation (Lanya's talk)

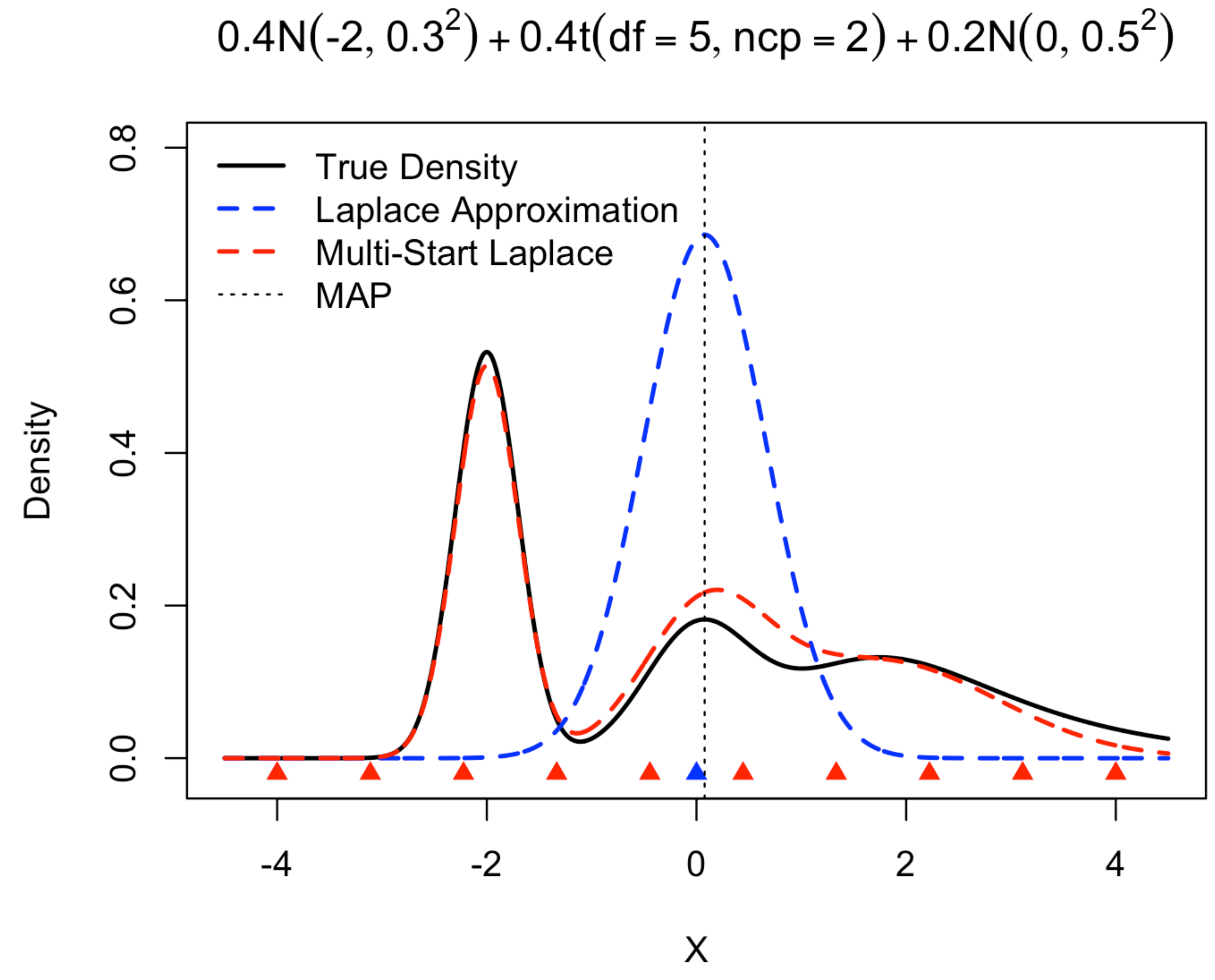
NN with Uncertainties

Being Bayesian, Even Just a Bit, Helps

Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks

Agustinus Kristiadi [1] **Matthias Hein** [1] **Philipp Hennig** [1 2]

- Bayesianize NN: don't have to have proper BNN, but NN with Bayes flavour
- Laplace Approximation (Lanya's talk)



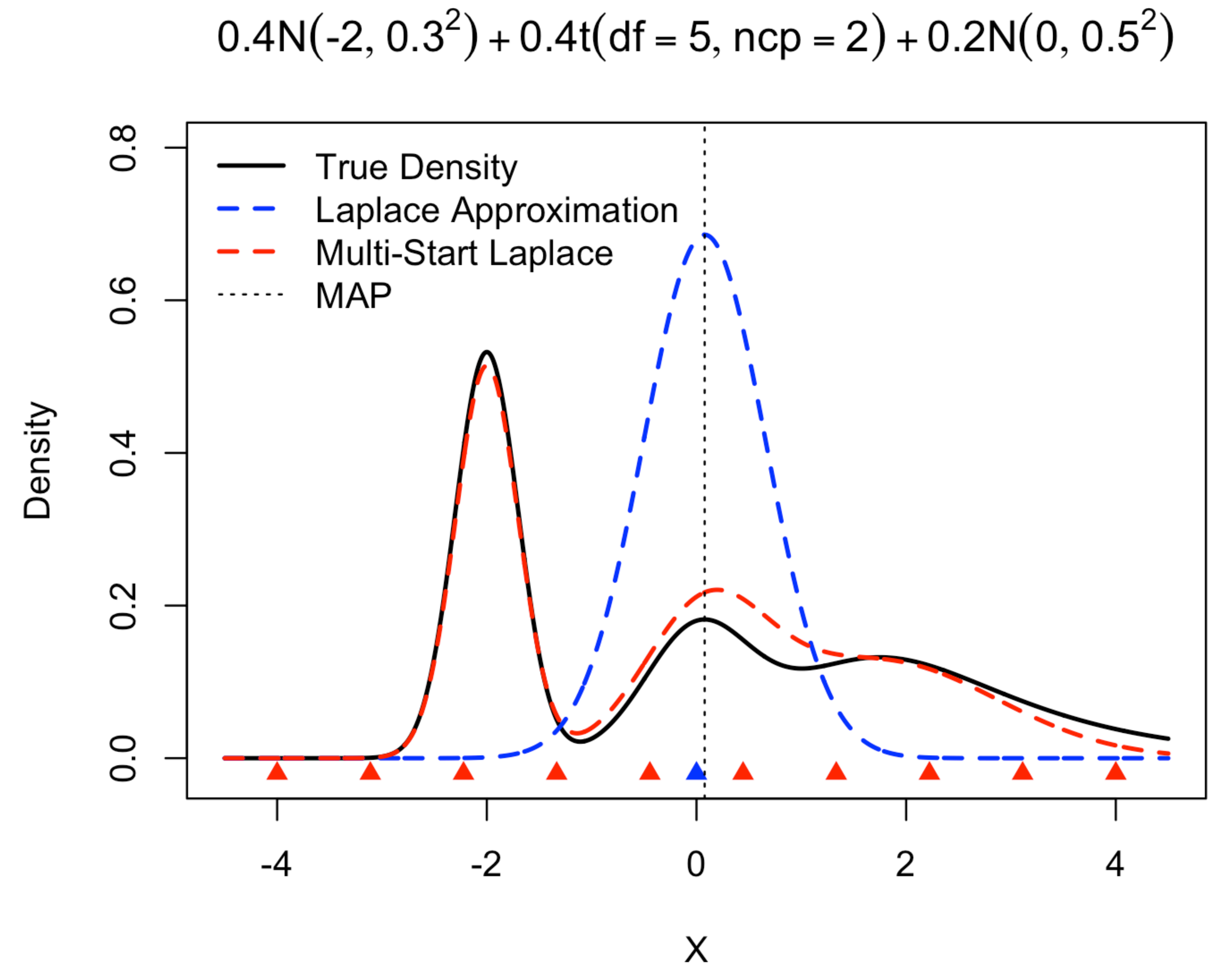
NN with Uncertainties

Being Bayesian, Even Just a Bit, Helps

Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks

Agustinus Kristiadi [1] **Matthias Hein** [1] **Philipp Hennig** [1 2]

- Bayesianize NN: don't have to have proper BNN, but NN with Bayes flavour
- Laplace Approximation (Lanya's talk)
- Bayesian Neural Network (Jixiang's talk)



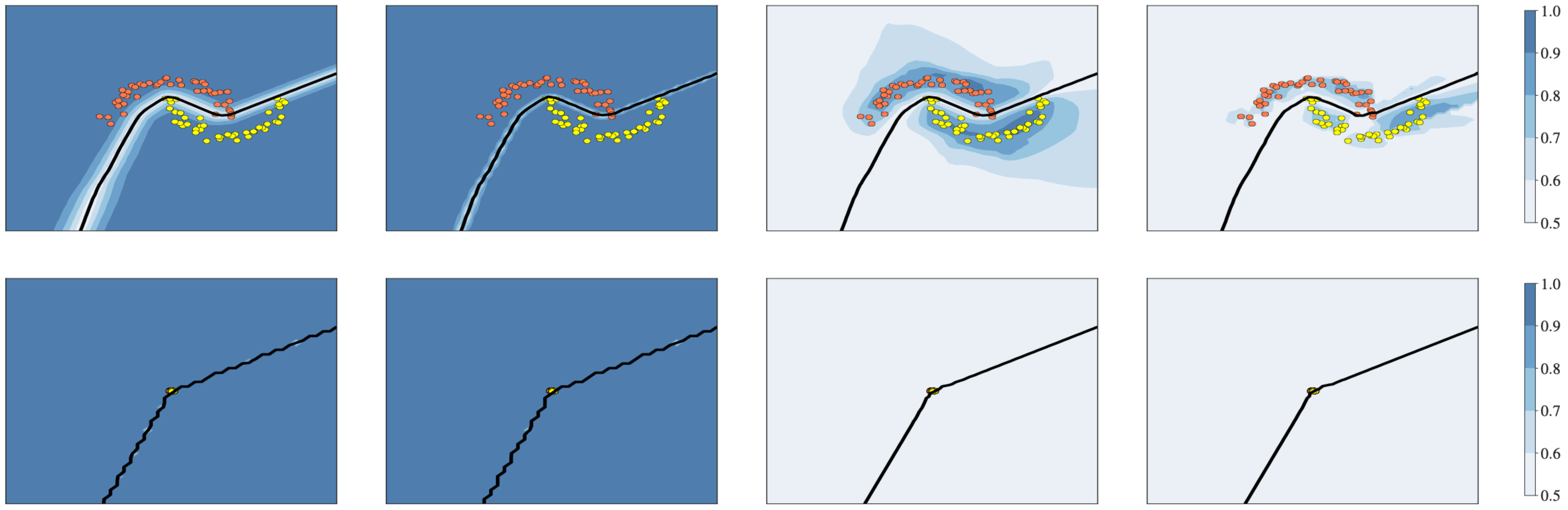
NN with Uncertainties

Being Bayesian, Even Just a Bit, Helps

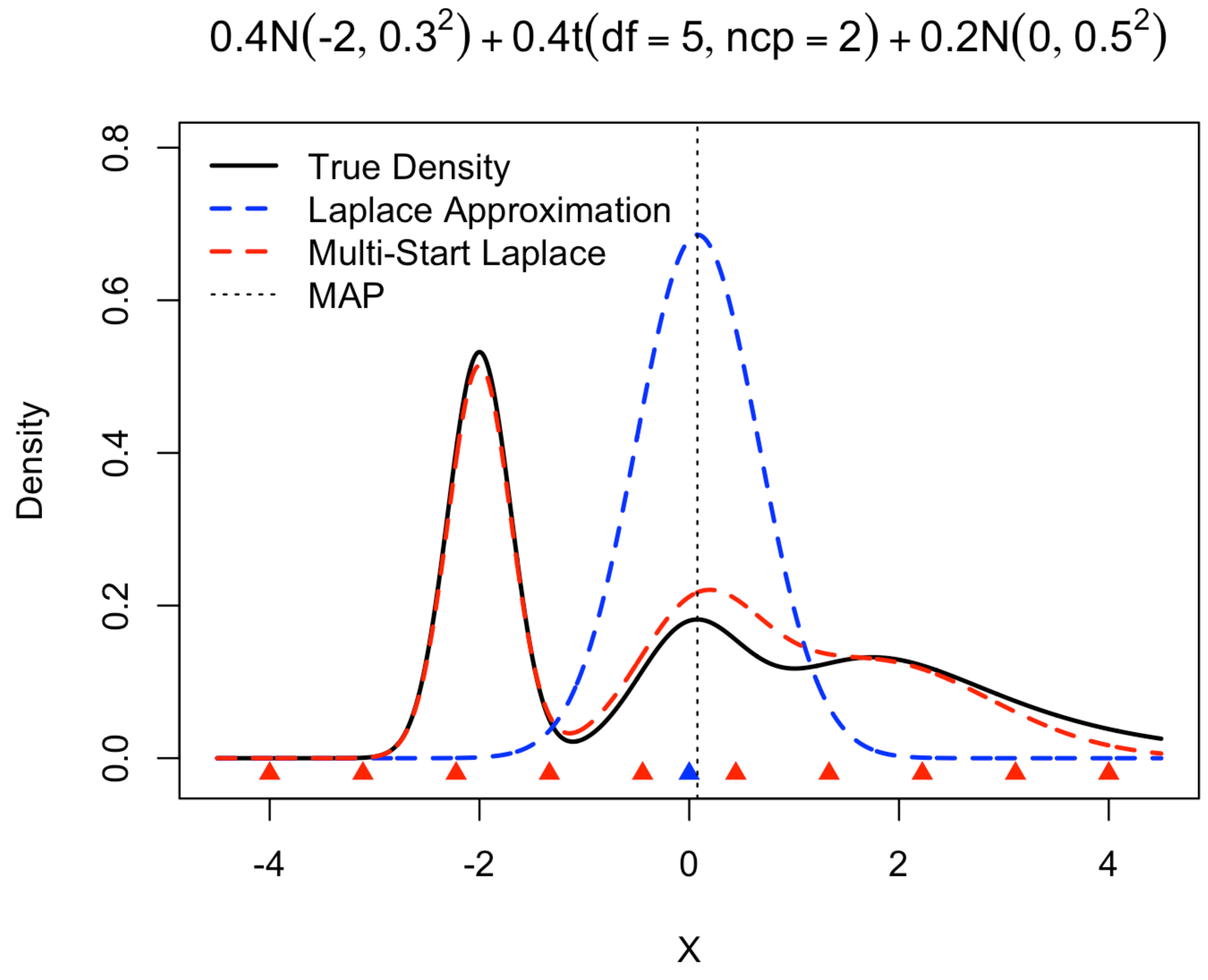
Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks

Agustinus Kristiadi [1] **Matthias Hein** [1] **Philipp Hennig** [1 2]

- Bayesianize NN: don't have to have proper BNN, but NN with Bayes flavour
- Laplace Approximation (Lanya's talk)
- Bayesian Neural Network (Jixiang's talk)



(a) MAP (b) Temp. scaling (c) Bayesian (last-layer) (d) Bayesian (all-layer)



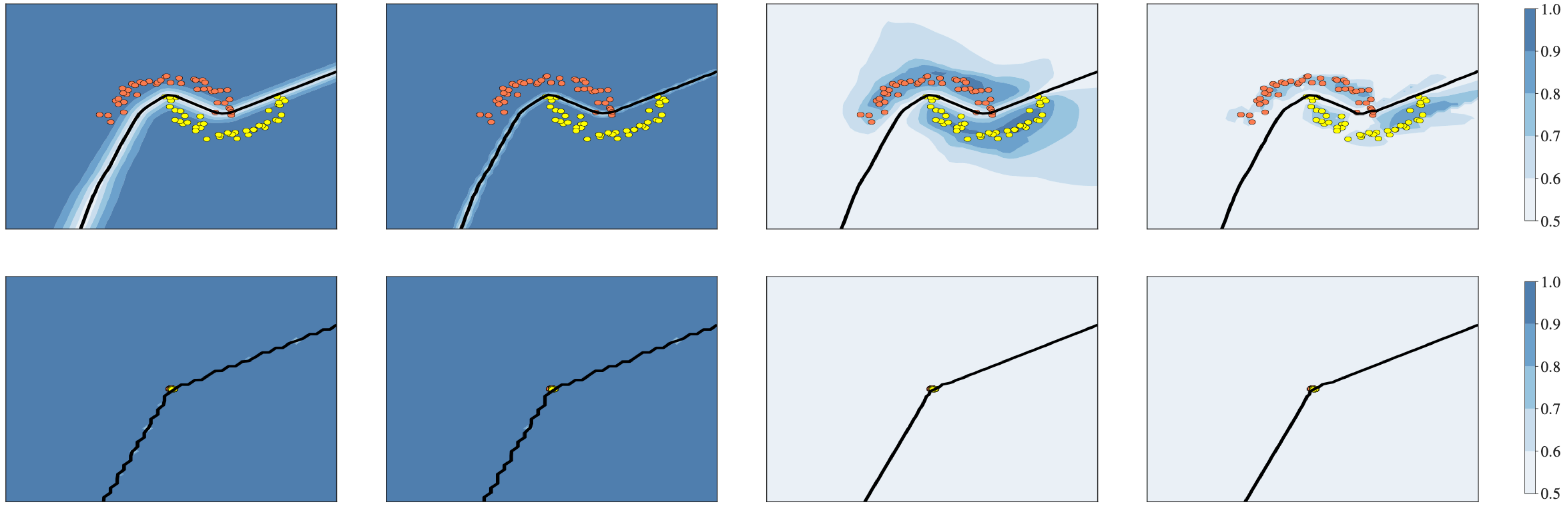
NN with Uncertainties

Being Bayesian, Even Just a Bit, Helps

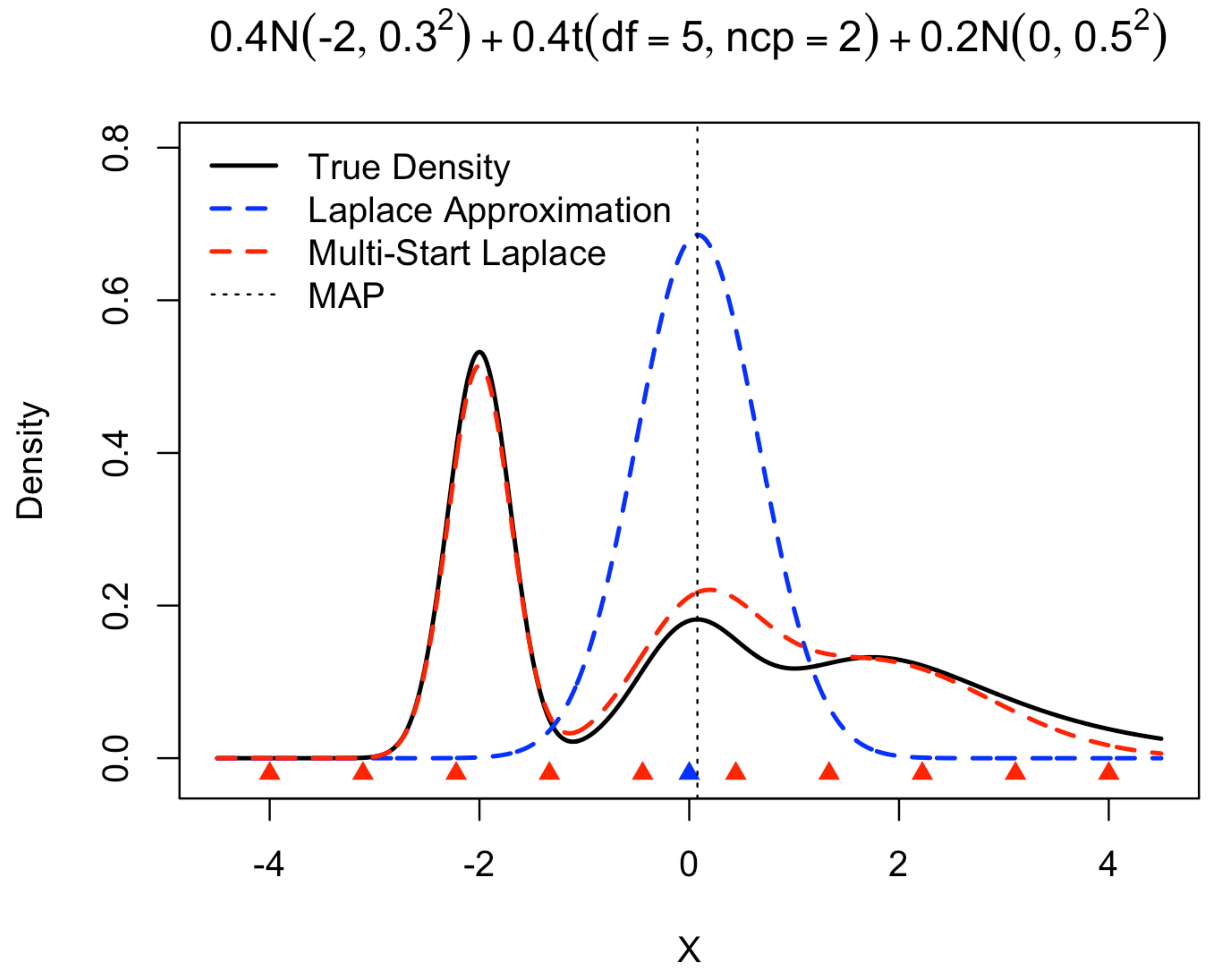
Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks

Agustinus Kristiadi[1] **Matthias Hein**[1] **Philipp Hennig**[1,2]

- Bayesianize NN: don't have to have proper BNN, but NN with Bayes flavour
- Laplace Approximation (Lanya's talk)
- Bayesian Neural Network (Jixiang's talk)



(a) MAP (b) Temp. scaling (c) Bayesian (last-layer) (d) Bayesian (all-layer)



Conformal Prediction?

Smarter NN Training

Smarter NN Training

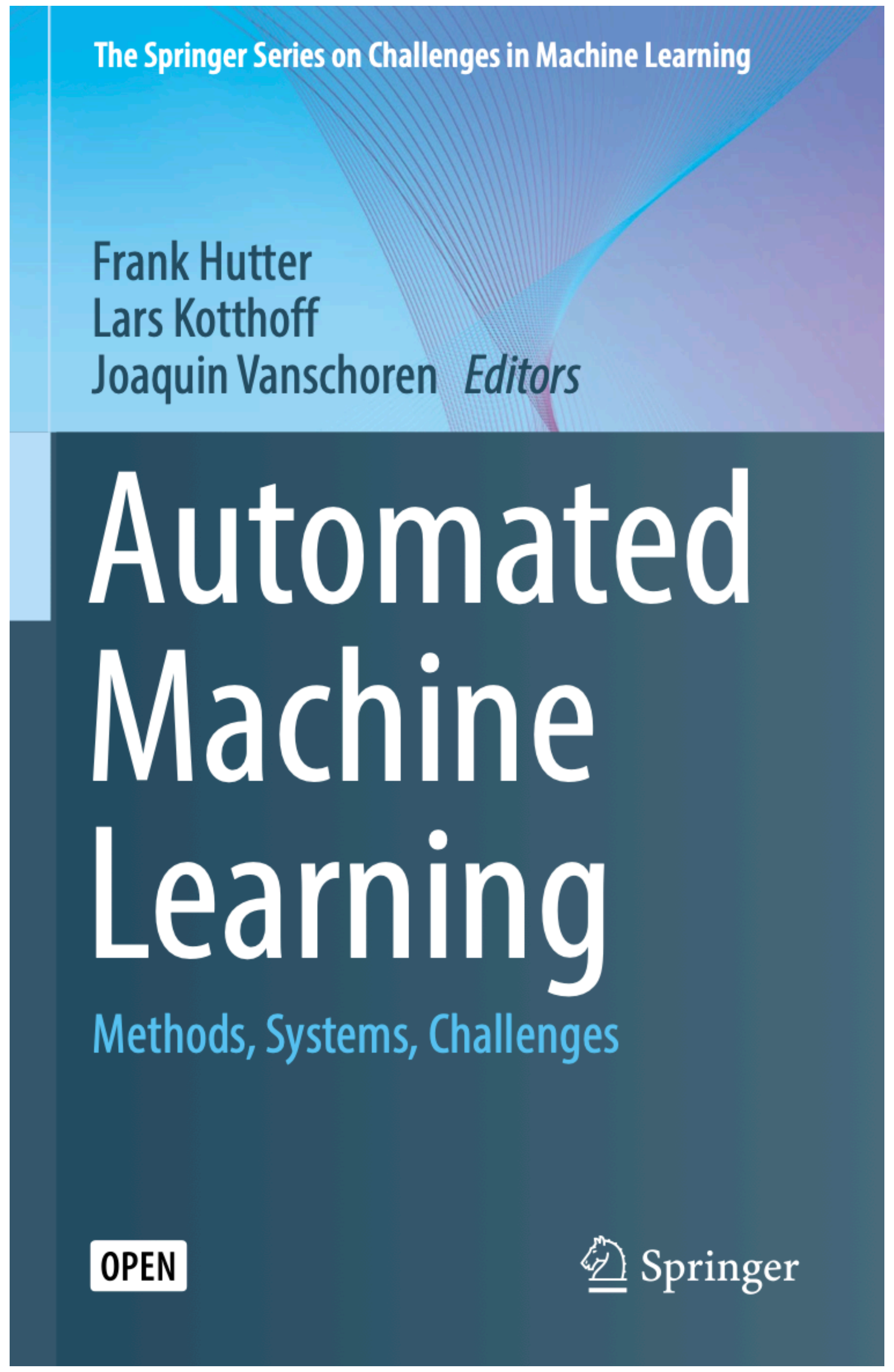
- Beyond manual knob turning for hyperparameter (learning rates, tolerance, etc) tunings.

Smarter NN Training

- Beyond manual knob turning for hyperparameter (learning rates, tolerance, etc) tunings.
- Bayesian optimisation & Intelligent data acquisition (Freddie's talk)

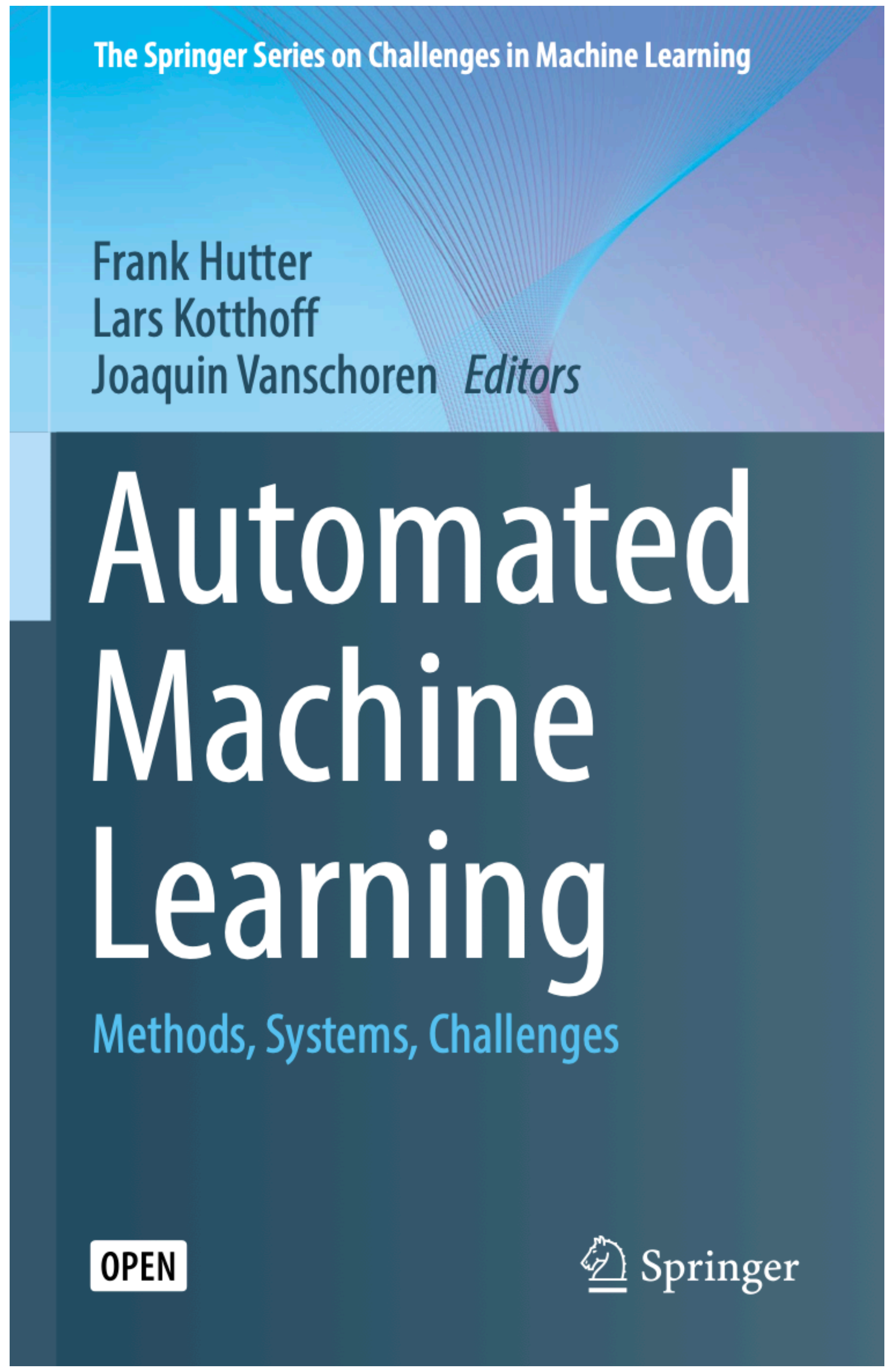
Smarter NN Training

- Beyond manual knob turning for hyperparameter (learning rates, tolerance, etc) tunings.
- Bayesian optimisation & Intelligent data acquisition (Freddie's talk)



Smarter NN Training

- Beyond manual knob turning for hyperparameter (learning rates, tolerance, etc) tunings.
- Bayesian optimisation & Intelligent data acquisition (Freddie's talk)
- (Are we just calling existing things ProbNum … ?)



Physics-Data Spectrum

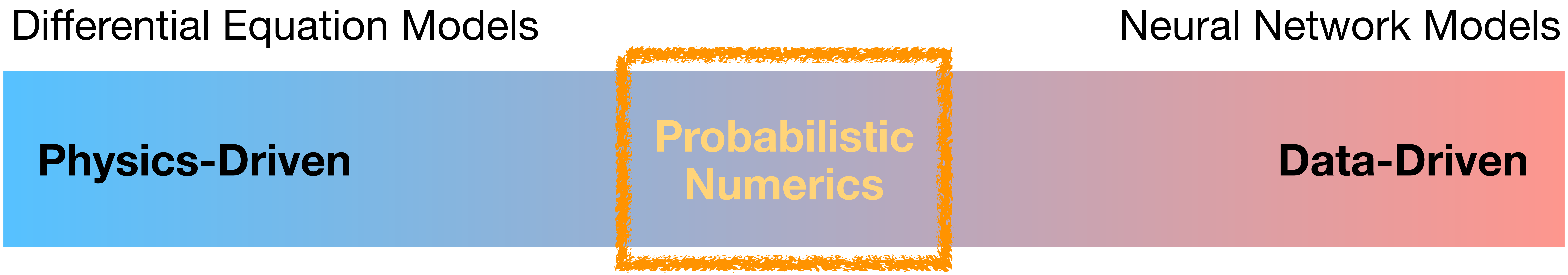
Differential Equation Models

Neural Network Models

Physics-Driven

Data-Driven

Physics-Data Spectrum



Physics-Data Spectrum

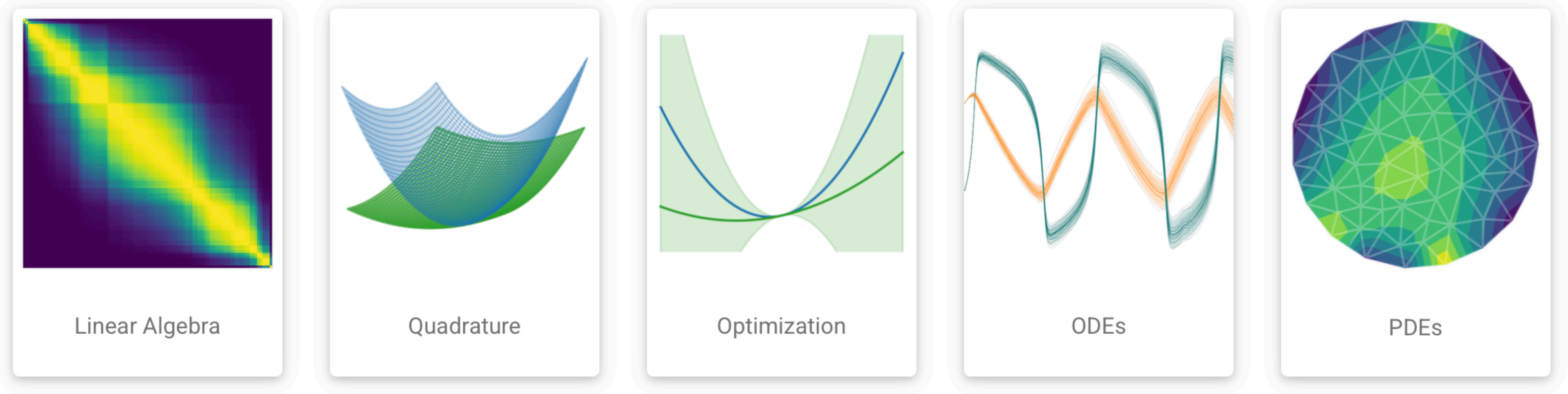
Differential Equation Models

Neural Network Models

Physics-Driven

Probabilistic Numerics

Data-Driven



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

Physics-Data Spectrum

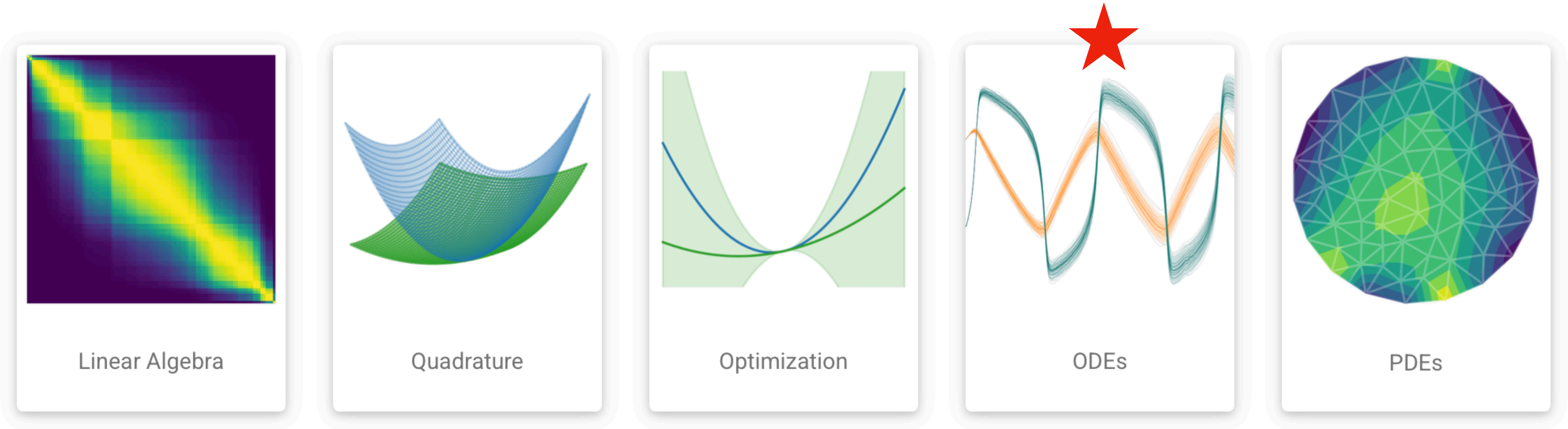
Differential Equation Models

Neural Network Models

Physics-Driven

Probabilistic Numerics

Data-Driven



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

Classical ODE Solver

Classical ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Classical ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Euler's Method $\quad x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$

Classical ODE Solver

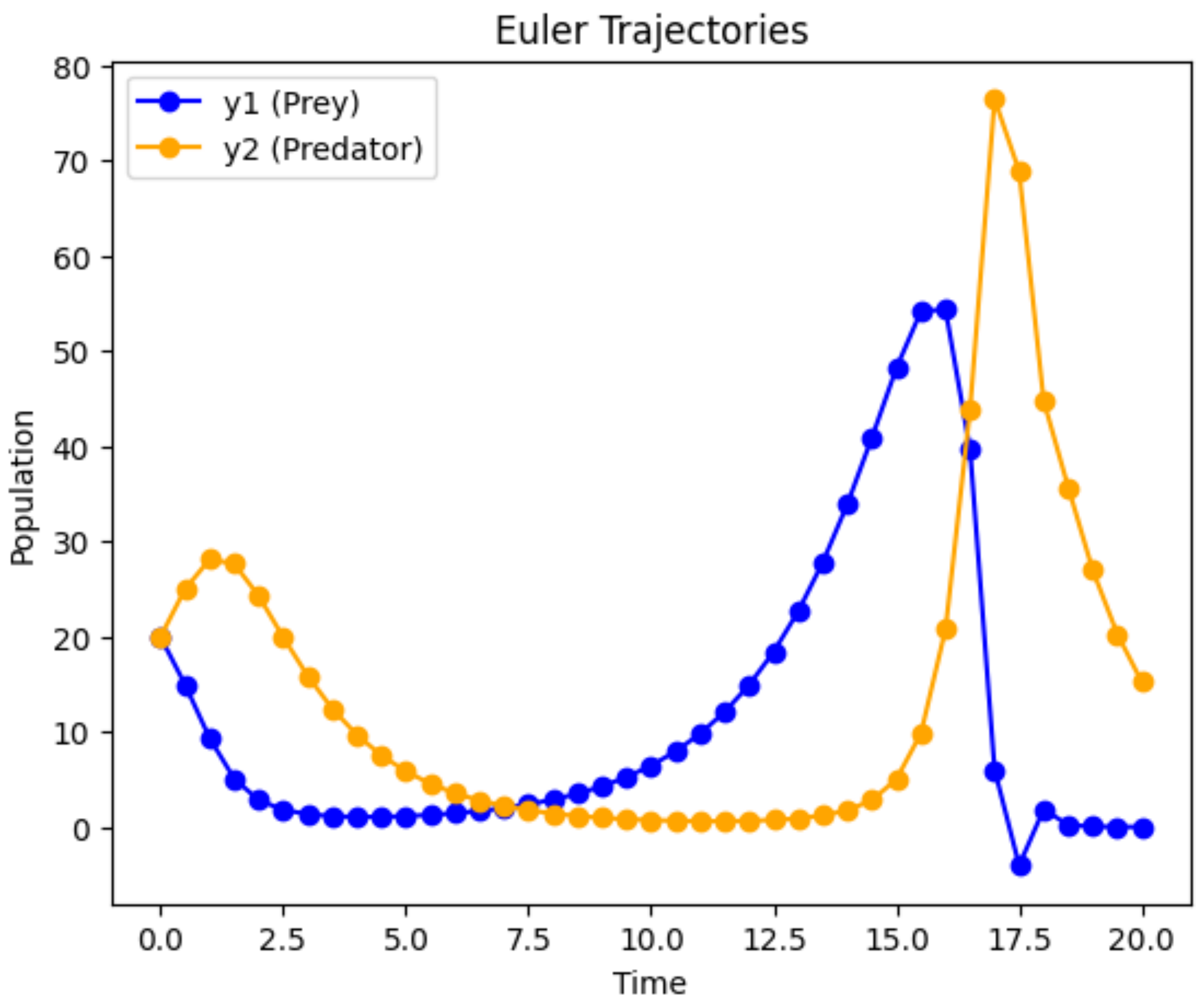
$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

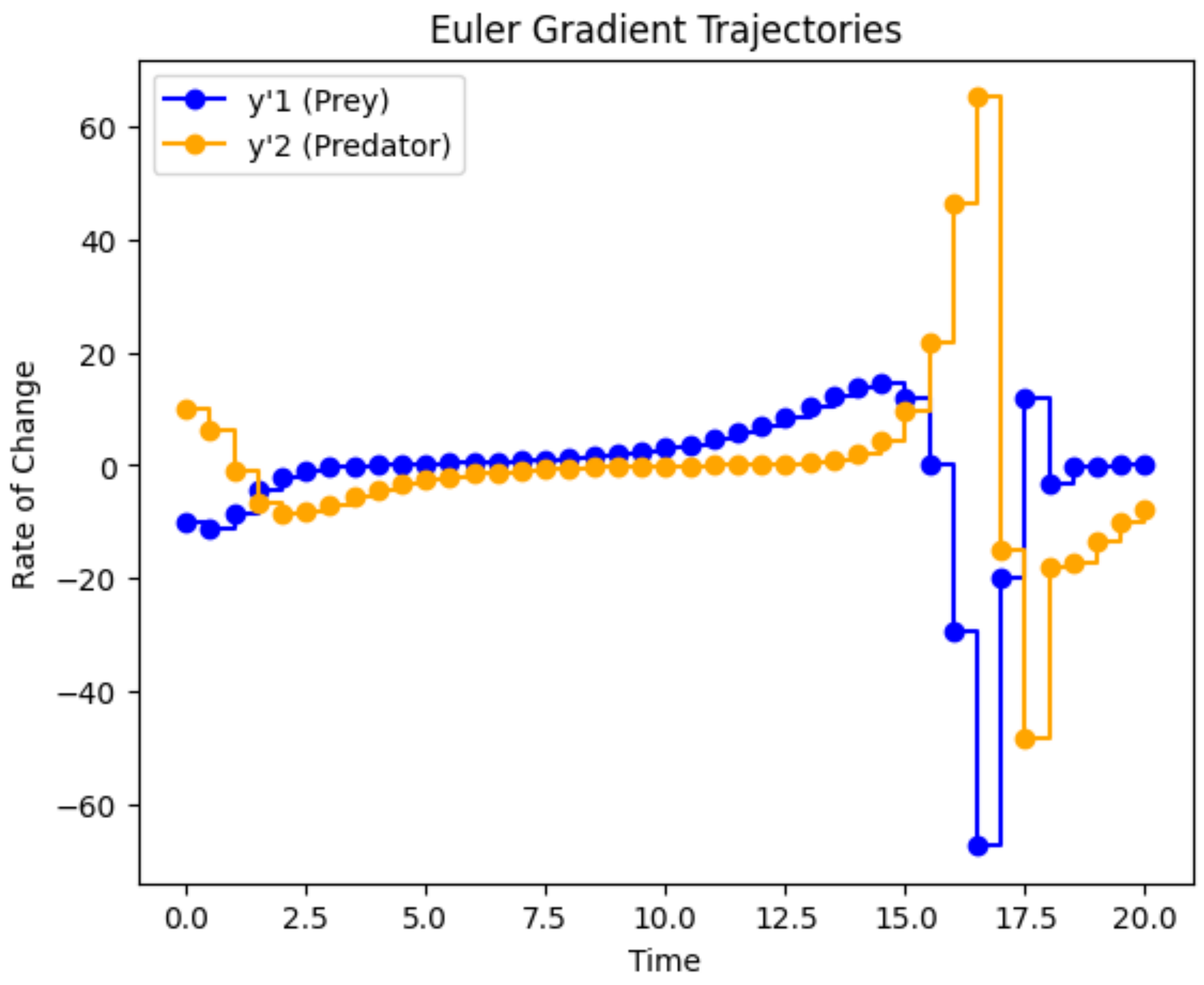
Euler's Method $\quad x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$

<u>Lotka-Volterra</u>

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$





Probabilistic ODE Solver

Probabilistic ODE Solver

IVP $$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Residual $$z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

Probabilistic ODE Solver

IVP $$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Residual $$z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

Euler's Method: Freeze gradient between consecutive discretisation points.

Probabilistic ODE Solver

IVP $$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Residual $$z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

Euler's Method: Freeze gradient between consecutive discretisation points.

Collocation Method: Ensuring residual conditions are met at discretisation points.

Probabilistic ODE Solver

IVP $$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Residual $$z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

Euler's Method: Freeze gradient between consecutive discretisation points.

Collocation Method: Ensuring residual conditions are met at discretisation points.

Probabilistic ODE Solver


Home > Statistics and Computing > Article

Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective

Open access | Published: 18 September 2019

Volume 29, pages 1297–1315, (2019) Cite this article

You have full access to this open access article

Download PDF

Save article

Filip Tronarp, Hans Kersting, Simo Särkkä & Philipp Hennig

IVP $$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Residual $$z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

Euler's Method: Freeze gradient between consecutive discretisation points.

★ Collocation Method: Ensuring residual conditions are met at discretisation points.

Probabilistic ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0, \qquad z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

★ Collocation Method: Ensuring residual conditions are met at discretisation points.

Probabilistic ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0, \qquad z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

★ Collocation Method: Ensuring residual conditions are met at discretisation points.

ODE Solve as Bayesian Inference

Probabilistic ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0, \qquad z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

★ Collocation Method: Ensuring residual conditions are met at discretisation points.

<u>ODE Solve as Bayesian Inference</u>

- **Goal**: Estimate discretised (probabilistic) trajectory $\{X_{t_k}\}_{k=0}^{N}$ satisfying the ODE conditions as best as possible.

Probabilistic ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0, \qquad z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

★ Collocation Method: Ensuring residual conditions are met at discretisation points.

ODE Solve as Bayesian Inference

- **Goal**: Estimate discretised (probabilistic) trajectory $\{X_{t_k}\}_{k=0}^{N}$ satisfying the ODE conditions as best as possible.
- **Prior**: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

Probabilistic ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0, \qquad z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

★ Collocation Method: Ensuring residual conditions are met at discretisation points.

ODE Solve as Bayesian Inference

- **Goal**: Estimate discretised (probabilistic) trajectory $\{X_{t_k}\}_{k=0}^{N}$ satisfying the ODE conditions as best as possible.
- **Prior**: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.
- **Data**: Collocation points $\{z_{t_k} = 0\}_{k=1}^{N}$.

Probabilistic ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0, \qquad z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

★ Collocation Method: Ensuring residual conditions are met at discretisation points.

ODE Solve as Bayesian Inference

- **Goal**: Estimate discretised (probabilistic) trajectory $\{X_{t_k}\}_{k=0}^{N}$ satisfying the ODE conditions as best as possible.
- **Prior**: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.
- **Data**: Collocation points $\{z_{t_k} = 0\}_{k=1}^{N}$.
- **Posterior**: Standard Bayes posterior.

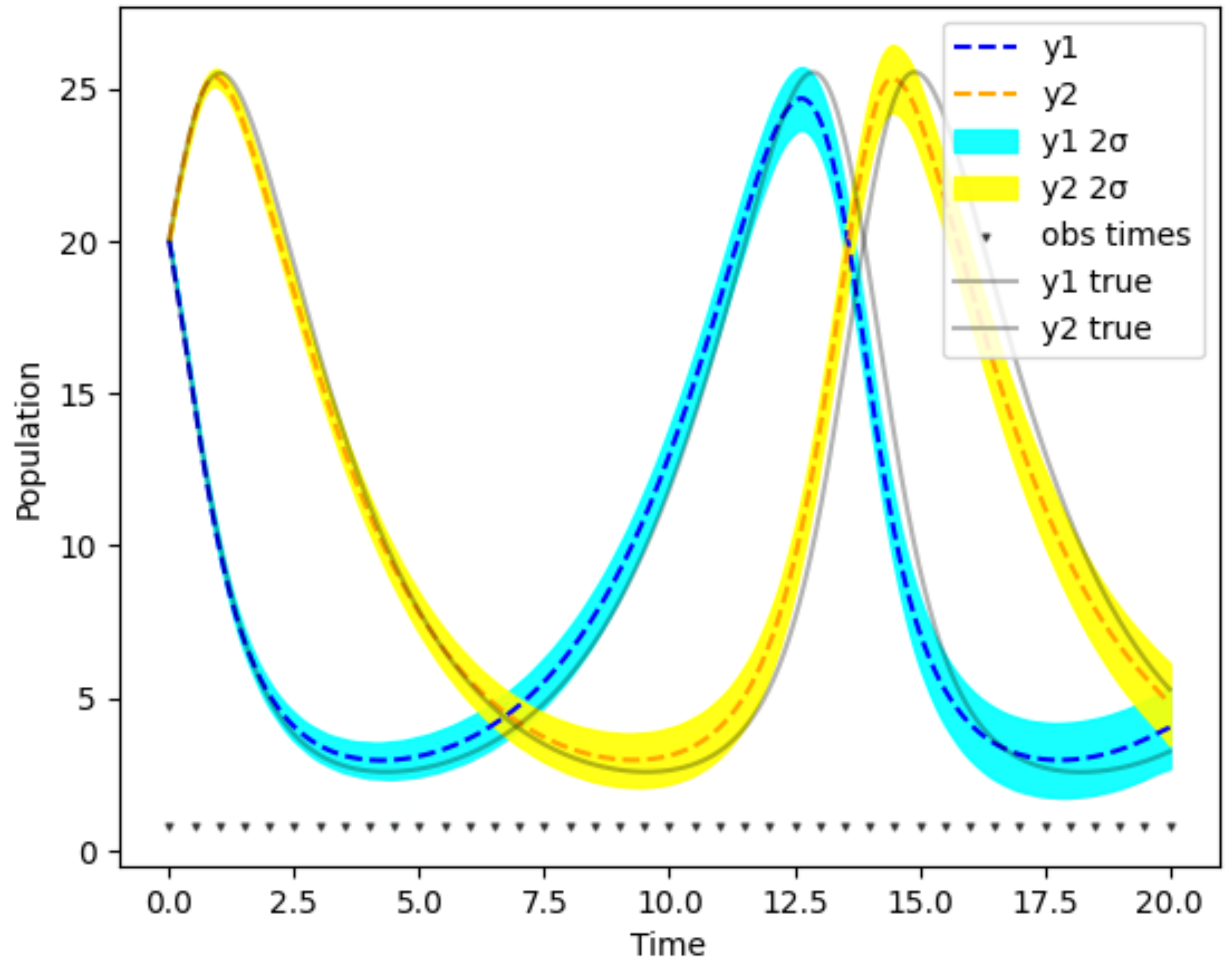
Probabilistic ODE Solver

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0, \qquad z(t) := \frac{d}{dt}y(t) - f(y(t), t) = 0$$

★ Collocation Method: Ensuring residual conditions are met at discretisation points.

ODE Solve as Bayesian Inference

- **Goal**: Estimate discretised (probabilistic) trajectory $\{X_{t_k}\}_{k=0}^{N}$ satisfying the ODE conditions as best as possible.
- **Prior**: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.
- **Data**: Collocation points $\{z_{t_k} = 0\}_{k=1}^{N}$.
- **Posterior**: Standard Bayes posterior.



Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

- Need to be stochastic and sufficiently expressive.

Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

- Need to be stochastic and sufficiently expressive.
- Need to contain both the trajectories and its derivative(s).

Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

- Need to be stochastic and sufficiently expressive.
- Need to contain both the trajectories and its derivative(s).
- Trajectories and derivative(s) dynamics need to be coupled.

Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

- Need to be stochastic and sufficiently expressive.
- Need to contain both the trajectories and its derivative(s).
- Trajectories and derivative(s) dynamics need to be coupled.
- Should make subsequent calculations easy (*will return to it later*).

Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

- Need to be stochastic and sufficiently expressive.
- Need to contain both the trajectories and its derivative(s).
- Trajectories and derivative(s) dynamics need to be coupled.
- Should make subsequent calculations easy (*will return to it later*).

$$d\mathbf{X}_t = d\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = F\mathbf{X}_t dt + L dW_t, \qquad \frac{d}{dt}X_t^1 = X_t^2$$

Probabilistic ODE Solver

<u>WANT</u>: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

$$d\mathbf{X}_t = d\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = F\mathbf{X}_t dt + L dW_t, \qquad \frac{d}{dt}X_t^1 = X_t^2$$

Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

$$d\mathbf{X}_t = d\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = F\mathbf{X}_t dt + L dW_t, \qquad \frac{d}{dt}X_t^1 = X_t^2$$

Example: Integrated Wiener Process (order-1, dim-1)

$$d\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} dt + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} dW_t^1 \\ dW_t^2 \end{bmatrix} = \begin{bmatrix} X_t^2 dt \\ dW_t^2 \end{bmatrix}$$

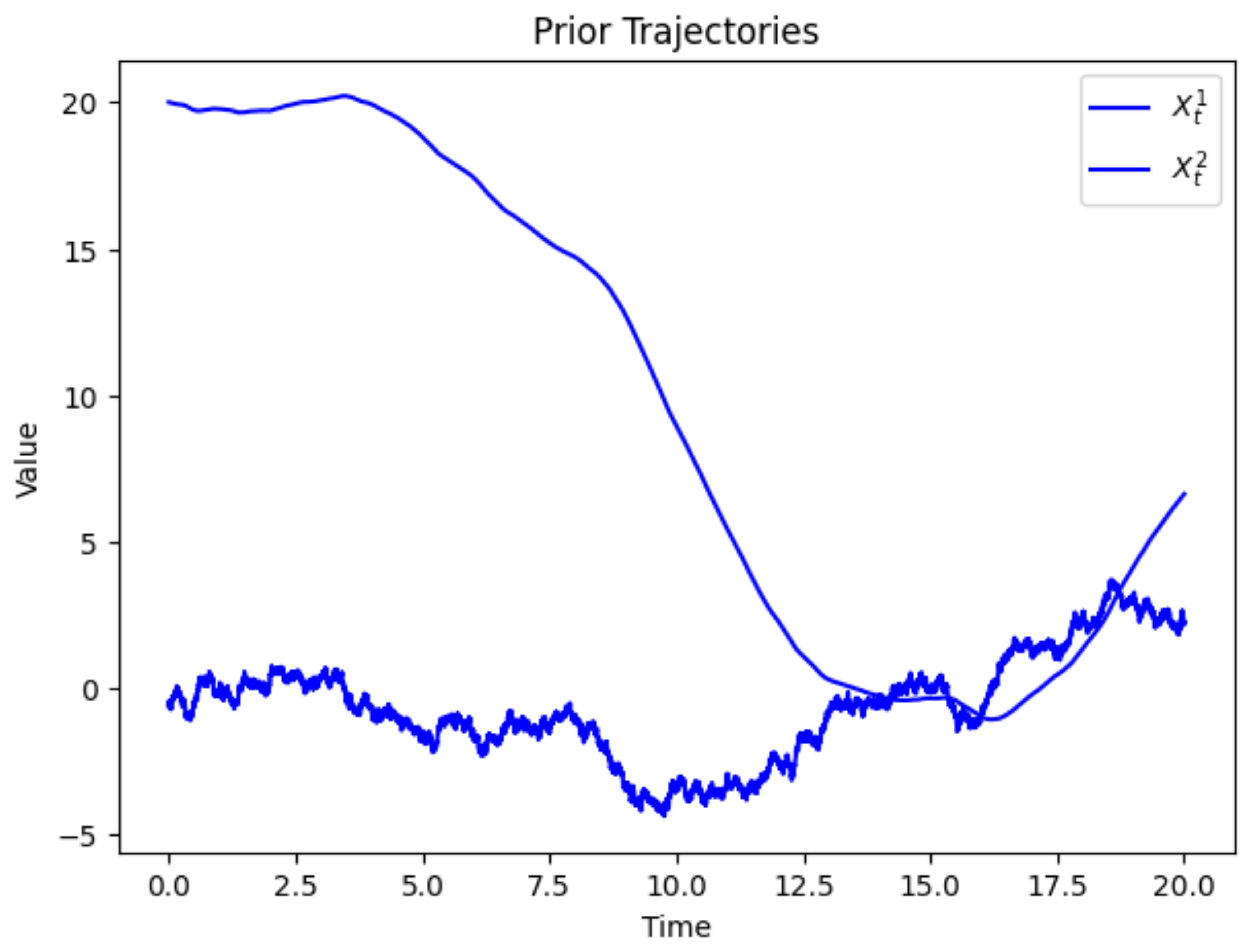
Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

$$d\mathbf{X}_t = d\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = F\mathbf{X}_t dt + L dW_t, \qquad \frac{d}{dt}X_t^1 = X_t^2$$

Example: Integrated Wiener Process (order-1, dim-1)

$$d\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} dt + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} dW_t^1 \\ dW_t^2 \end{bmatrix} = \begin{bmatrix} X_t^2 dt \\ dW_t^2 \end{bmatrix}$$



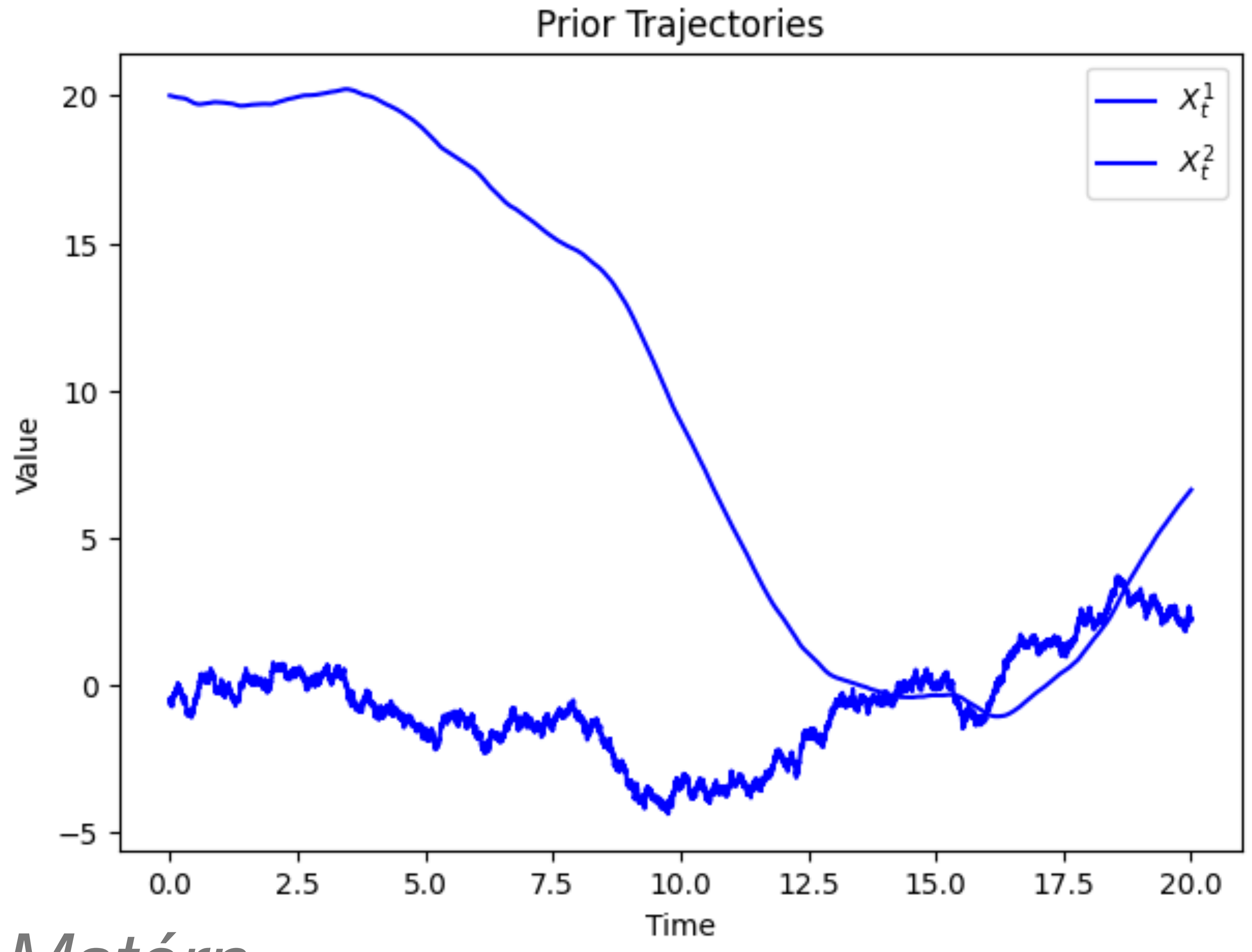
Probabilistic ODE Solver

WANT: Sufficiently flexible prior dynamics for $\{X_{t_k}\}_{k=0}^{N}$.

$$d\mathbf{X}_t = d\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = F\mathbf{X}_t dt + L dW_t, \qquad \frac{d}{dt}X_t^1 = X_t^2$$

Example: Integrated Wiener Process (order-1, dim-1)

$$d\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} dt + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} dW_t^1 \\ dW_t^2 \end{bmatrix} = \begin{bmatrix} X_t^2 dt \\ dW_t^2 \end{bmatrix}$$



other SDE priors are also possible, e.g. Matérn

Probabilistic ODE Solver

WANT: Collocation points $\{z_{t_k} = 0\}_{k=1}^{N}$.

Probabilistic ODE Solver

WANT: Collocation points $\{z_{t_k} = 0\}_{k=1}^{N}$.

For ODE on $[0,T]$, pick timestamps $0 < t_1 < \ldots < t_N = T$.

Probabilistic ODE Solver

WANT: Collocation points $\{z_{t_k} = 0\}_{k=1}^{N}$.

For ODE on $[0,T]$, pick timestamps $0 < t_1 < \ldots < t_N = T$.

does not have to be uniform

Probabilistic ODE Solver

WANT: Collocation points $\{z_{t_k} = 0\}_{k=1}^{N}$.

For ODE on $[0,T]$, pick timestamps $0 < t_1 < \ldots < t_N = T$.

does not have to be uniform

Enforce residuals $z_k := z_{t_k} = X_{t_k}^2 - f(X_{t_k}^1, t_k) = 0$ for all N points.

Probabilistic ODE Solver

<u>WANT</u>: Collocation points $\{z_{t_k} = 0\}_{k=1}^{N}$.

For ODE on $[0,T]$, pick timestamps $0 < t_1 < \ldots < t_N = T$.

does not have to be uniform

Enforce residuals $z_k := z_{t_k} = X^2_{t_k} - f(X^1_{t_k}, t_k) = 0$ for all N points.

<u>Notation</u>: Define $C, \dot{C}$ such that $C\mathbf{X}_t = X^1_t, \dot{C}\mathbf{X}_t = X^2_t$; so $z_k = \dot{C}\mathbf{X}_{t_k} - f(C\mathbf{X}_{t_k}, t_k) = 0$

Probabilistic ODE Solver

WANT: Standard Bayes posterior.

Probabilistic ODE Solver

WANT: Standard Bayes posterior.

Prior

$$d\mathbf{X}_t = F\mathbf{X}_t dt + L dW_t$$

+

Data

$$\{z_k = \dot{C}\mathbf{X}_{t_k} - f(C\mathbf{X}_{t_k}, t_k) = 0\}_{k=1}^{N}$$

Probabilistic ODE Solver

WANT: Standard Bayes posterior.

Prior

$$d\mathbf{X}_t = F\mathbf{X}_t dt + L dW_t$$

+

Data

$$\{z_k = \dot{C}\mathbf{X}_{t_k} - f(C\mathbf{X}_{t_k}, t_k) = 0\}_{k=1}^{N}$$

Can be phrased as a State Space Model !!

State Space Model

State Space Model

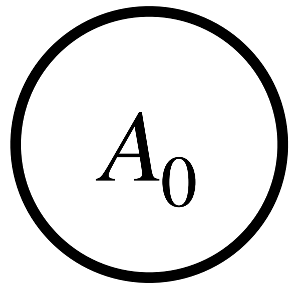
(Latent Dynamics) $A_{n+1} \mid \left(A_{0:n} = a_{0:n}, B_{0:n} = b_{0:n}\right) \sim P(\,\cdot \mid a_n)$

(Observations) $B_{n+1} \mid \left(A_{0:n+1} = a_{0:n+1}, B_{0:n} = b_{0:n}\right) \sim g(\,\cdot \mid a_{n+1})$

State Space Model

(Latent Dynamics) $A_{n+1} \,|\, \left(A_{0:n} = a_{0:n}, B_{0:n} = b_{0:n}\right) \sim P(\,\cdot\,|\, a_n)$

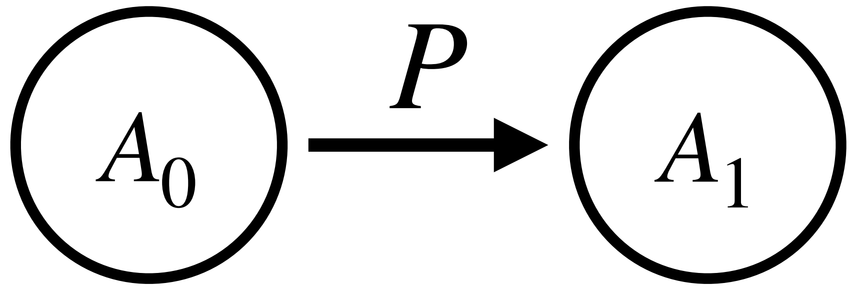
(Observations) $B_{n+1} \,|\, \left(A_{0:n+1} = a_{0:n+1}, B_{0:n} = b_{0:n}\right) \sim g(\,\cdot\,|\, a_{n+1})$



State Space Model

(Latent Dynamics) $A_{n+1} \,|\, \left(A_{0:n} = a_{0:n}, B_{0:n} = b_{0:n}\right) \sim P(\,\cdot\,|\,a_n)$

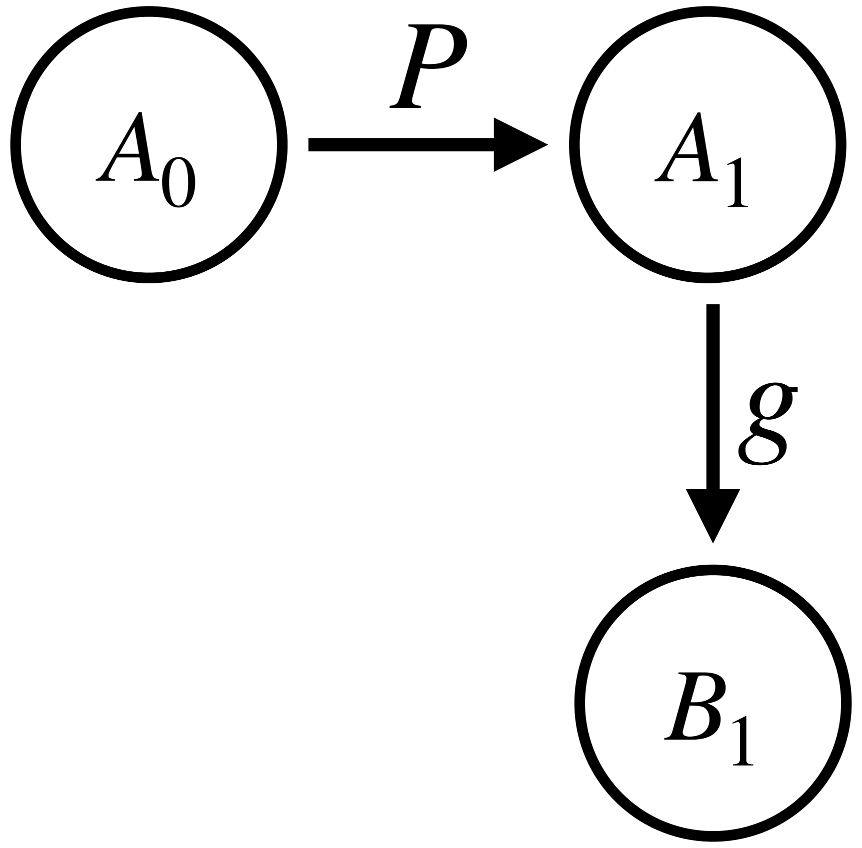
(Observations) $B_{n+1} \,|\, \left(A_{0:n+1} = a_{0:n+1}, B_{0:n} = b_{0:n}\right) \sim g(\,\cdot\,|\,a_{n+1})$



State Space Model

(Latent Dynamics) $A_{n+1} \mid \left(A_{0:n} = a_{0:n}, B_{0:n} = b_{0:n}\right) \sim P(\,\cdot\mid a_n)$

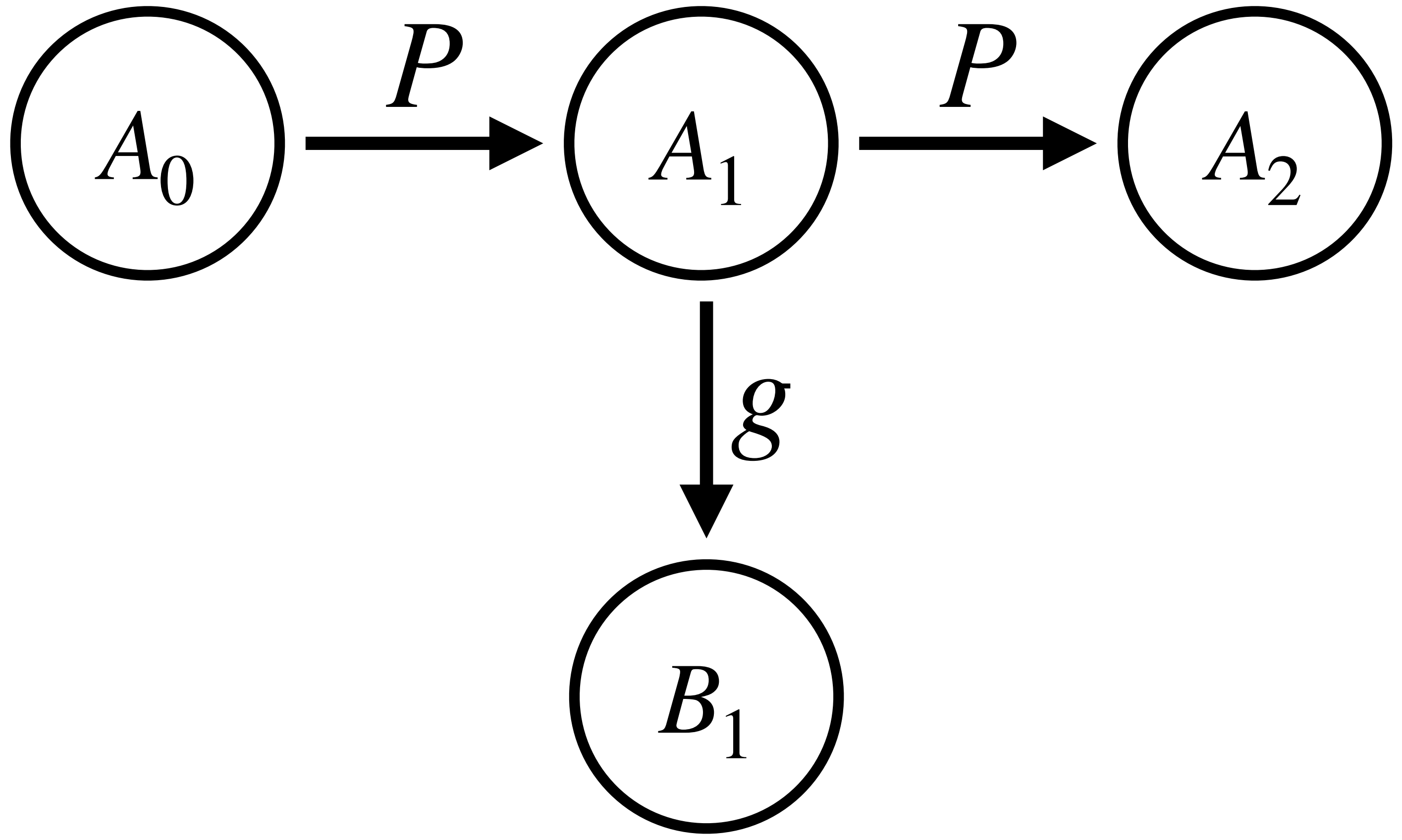
(Observations) $B_{n+1} \mid \left(A_{0:n+1} = a_{0:n+1}, B_{0:n} = b_{0:n}\right) \sim g(\,\cdot\mid a_{n+1})$



State Space Model

(Latent Dynamics) $A_{n+1} \mid \left(A_{0:n} = a_{0:n}, B_{0:n} = b_{0:n}\right) \sim P(\,\cdot \mid a_n)$

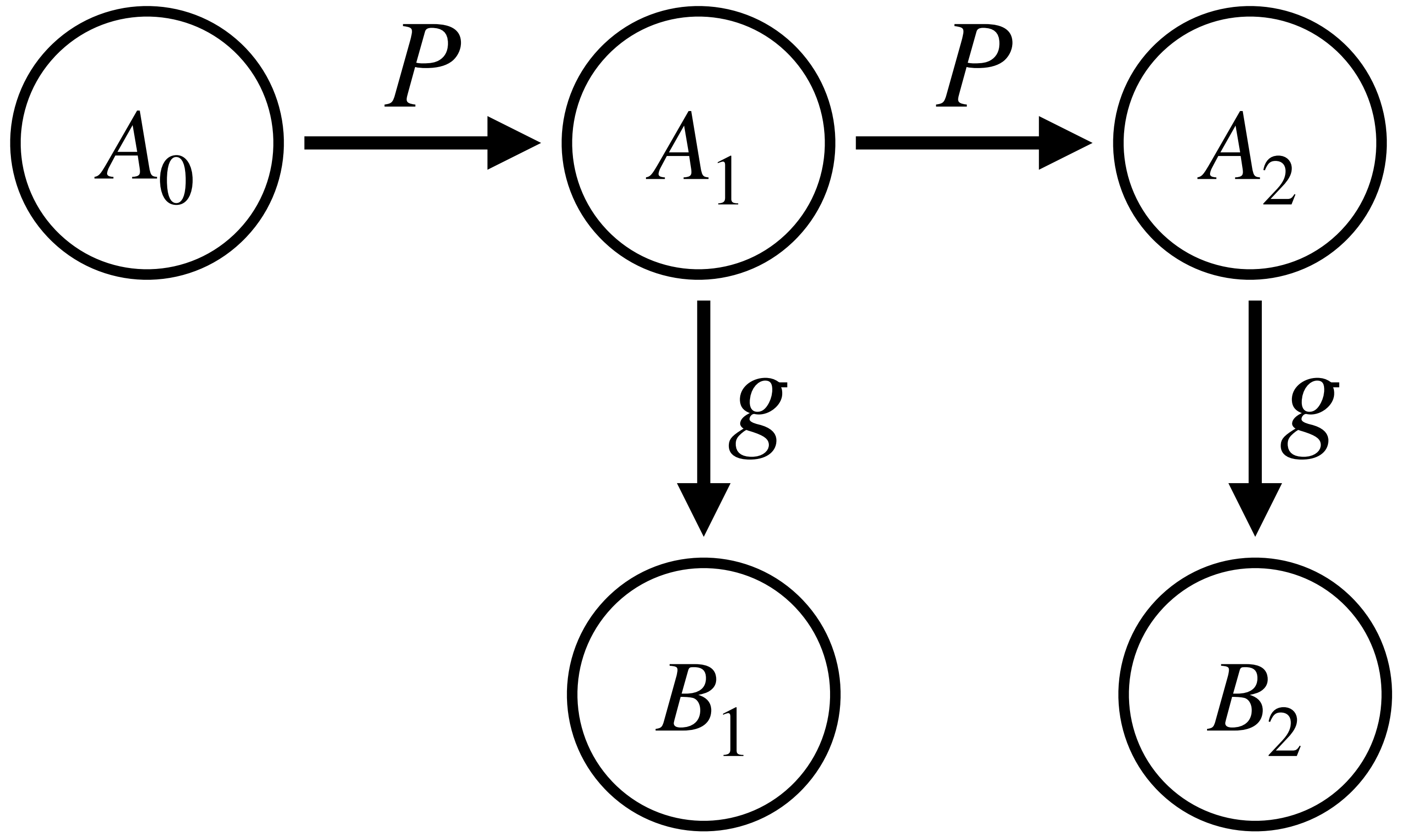
(Observations) $B_{n+1} \mid \left(A_{0:n+1} = a_{0:n+1}, B_{0:n} = b_{0:n}\right) \sim g(\,\cdot \mid a_{n+1})$



State Space Model

(Latent Dynamics) $A_{n+1} \,|\, \left(A_{0:n} = a_{0:n}, B_{0:n} = b_{0:n}\right) \sim P(\,\cdot\,|\, a_n)$

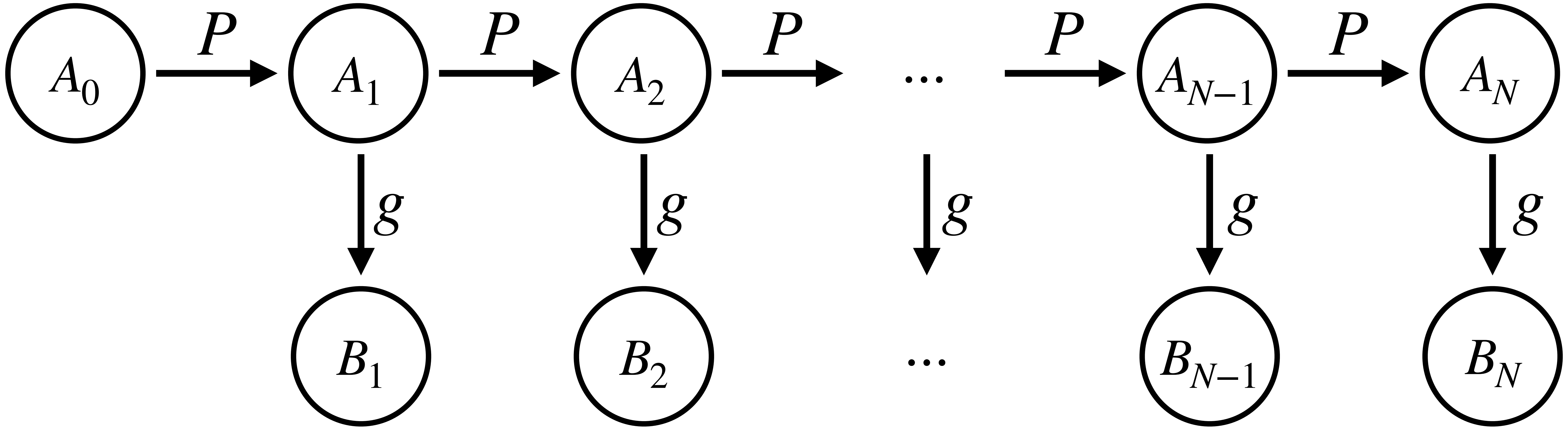
(Observations) $B_{n+1} \,|\, \left(A_{0:n+1} = a_{0:n+1}, B_{0:n} = b_{0:n}\right) \sim g(\,\cdot\,|\, a_{n+1})$



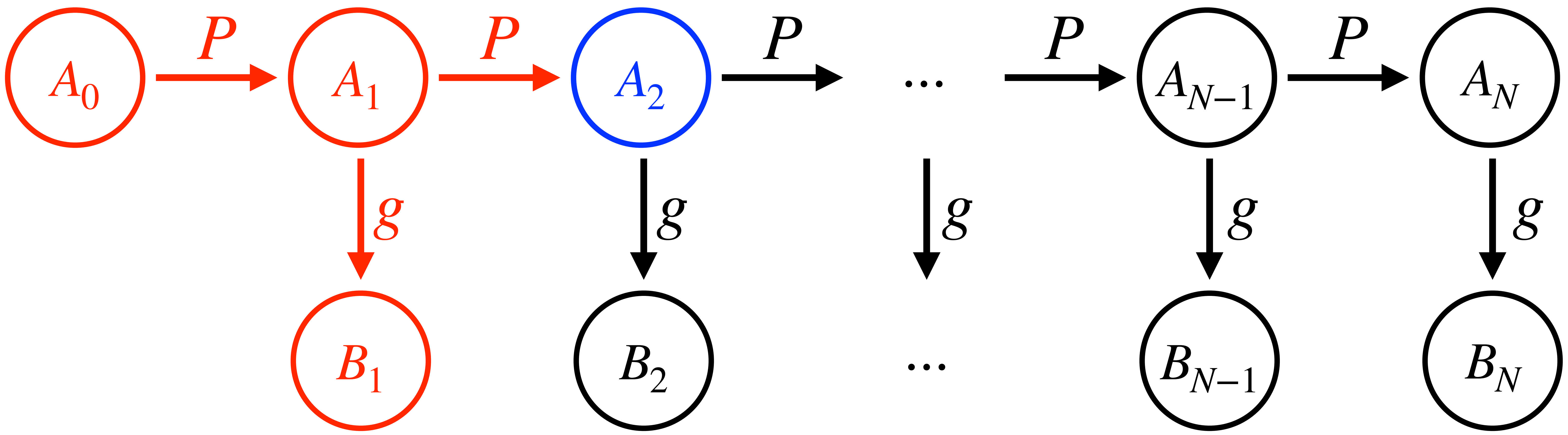
State Space Model

(Latent Dynamics) $A_{n+1} \,|\, \left(A_{0:n} = a_{0:n}, B_{0:n} = b_{0:n}\right) \sim P(\,\cdot\,|\,a_n)$

(Observations) $B_{n+1} \,|\, \left(A_{0:n+1} = a_{0:n+1}, B_{0:n} = b_{0:n}\right) \sim g(\,\cdot\,|\,a_{n+1})$

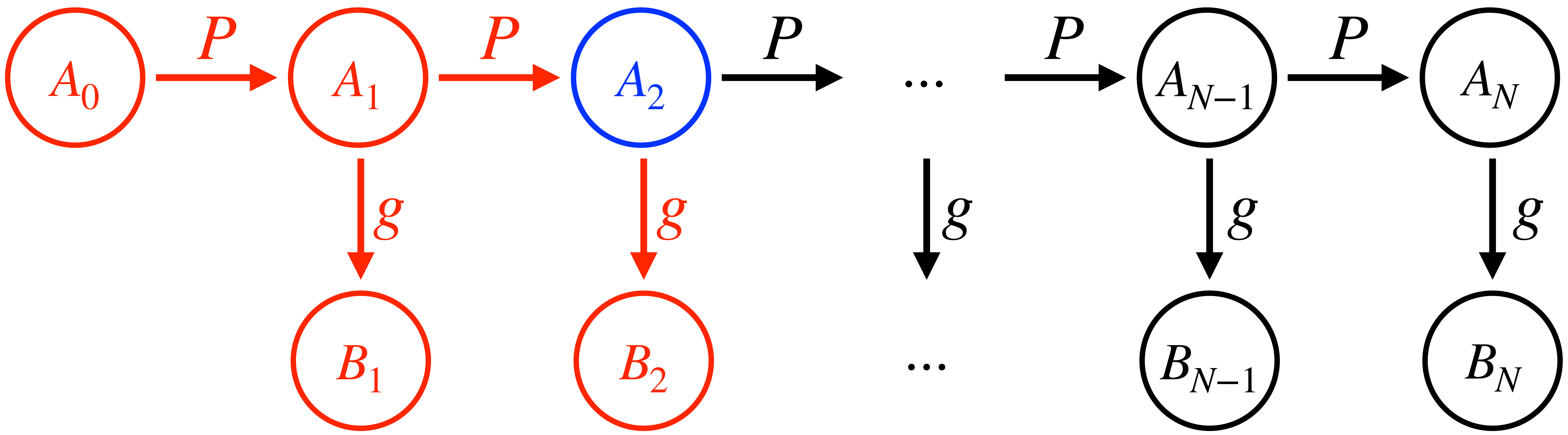


State Space Model



(Predicting Distribution) $p\left(A_n \,|\, A_{0:n-1} = a_{0:n-1}\right)$

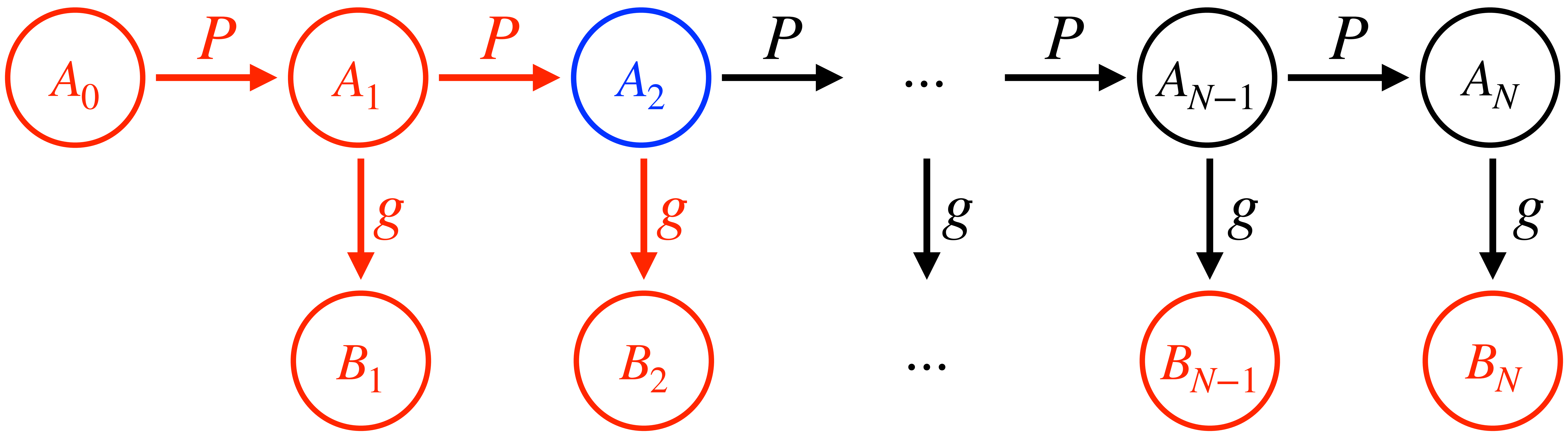
State Space Model



(Predicting Distribution) $p\left(A_n \,|\, A_{0:n-1} = a_{0:n-1}\right)$

(Filtering Distribution) $p\left(A_n \,|\, B_{0:n} = b_{0:n}\right)$

State Space Model



(Predicting Distribution) $p\left(A_n \mid A_{0:n-1} = a_{0:n-1}\right)$

(Filtering Distribution) $p\left(A_n \mid B_{0:n} = b_{0:n}\right)$

(Smoothing Distribution) $p\left(A_n \mid B_{0:N} = b_{0:N}\right)$

State Space Model

State Space Model

Example: Linear Gaussian Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

State Space Model

<u>Example: Linear Gaussian Model</u>

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

What are the distributions of … ?

State Space Model

<u>Example: Linear Gaussian Model</u>

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

What are the distributions of … ?

- predicting distribution $A_{n+1|n} \sim N(\mu_n^P, \Sigma_n^P)$

State Space Model

<u>Example: Linear Gaussian Model</u>

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

What are the distributions of … ?

- predicting distribution $A_{n+1|n} \sim N(\mu_n^P, \Sigma_n^P)$
- filtering distribution $A_{n+1|n+1} \sim N(\mu_n^F, \Sigma_n^F)$

State Space Model

Example: Linear Gaussian Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

What are the distributions of … ?

- predicting distribution $A_{n+1|n} \sim N(\mu_n^P, \Sigma_n^P)$
- filtering distribution $A_{n+1|n+1} \sim N(\mu_n^F, \Sigma_n^F)$
- smoothing distribution $A_{n+1|N} \sim N(\mu_n^S, \Sigma_n^S)$

Example: Linear Gaussian Model

State Space Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Starting with a known initial distribution $A_{0|0} \sim N(\mu_0^F, \Sigma_0^F)$.

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Starting with a known initial distribution $A_{0|0} \sim N(\mu_0^F, \Sigma_0^F)$.
- Propagate $A_{1|0} \sim N(G\mu_0^F, G\Sigma_0^F G^T + Q) = N(\mu_0^P, \Sigma_0^P)$.

Example: Linear Gaussian Model

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Starting with a known initial distribution $A_{0|0} \sim N(\mu_0^F, \Sigma_0^F)$.
- Propagate $A_{1|0} \sim N(G\mu_0^F, G\Sigma_0^F G^T + Q) = N(\mu_0^P, \Sigma_0^P)$.
- Given observation B_1:

State Space Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Starting with a known initial distribution $A_{0|0} \sim N(\mu_0^F, \Sigma_0^F)$.
- Propagate $A_{1|0} \sim N(G\mu_0^F, G\Sigma_0^F G^T + Q) = N(\mu_0^P, \Sigma_0^P)$.
- Given observation B_1:
  - Calculate *innovation* $v_1 = B_1 - H\mu_0^P$.

State Space Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Starting with a known initial distribution $A_{0|0} \sim N(\mu_0^F, \Sigma_0^F)$.
- Propagate $A_{1|0} \sim N(G\mu_0^F, G\Sigma_0^F G^T + Q) = N(\mu_0^P, \Sigma_0^P)$.
- Given observation B_1:
  - Calculate *innovation* $v_1 = B_1 - H\mu_0^P$.
  - Calculate *innovation covariance* $S_1 = H\Sigma_0 H^T + R$.

Example: Linear Gaussian Model

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Starting with a known initial distribution $A_{0|0} \sim N(\mu_0^F, \Sigma_0^F)$.
- Propagate $A_{1|0} \sim N(G\mu_0^F, G\Sigma_0^F G^T + Q) = N(\mu_0^P, \Sigma_0^P)$.
- Given observation B_1:
  - Calculate *innovation* $v_1 = B_1 - H\mu_0^P$.
  - Calculate *innovation covariance* $S_1 = H\Sigma_0 H^T + R$.
  - Calculate *Kalman gain* $K_1 = \Sigma_0^P H^T S_1^{-1}$.

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Starting with a known initial distribution $A_{0|0} \sim N(\mu_0^F, \Sigma_0^F)$.
- Propagate $A_{1|0} \sim N(G\mu_0^F, G\Sigma_0^F G^T + Q) = N(\mu_0^P, \Sigma_0^P)$.
- Given observation B_1:
  - Calculate *innovation* $v_1 = B_1 - H\mu_0^P$.
  - Calculate *innovation covariance* $S_1 = H\Sigma_0 H^T + R$.
  - Calculate *Kalman gain* $K_1 = \Sigma_0^P H^T S_1^{-1}$.
  - Obtain filtered state $A_{1|1} \sim N(\mu_0^P + K_1 v_1, \Sigma_0^P - K_1 S_1 K_1^T) = N(\mu_1^F, \mu_1^F)$.

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Starting with a known initial distribution $A_{0|0} \sim N(\mu_0^F, \Sigma_0^F)$.
- Propagate $A_{1|0} \sim N(G\mu_0^F, G\Sigma_0^F G^T + Q) = N(\mu_0^P, \Sigma_0^P)$.
- Given observation B_1:
  - Calculate *innovation* $v_1 = B_1 - H\mu_0^P$.
  - Calculate *innovation covariance* $S_1 = H\Sigma_0 H^T + R$.
  - Calculate *Kalman gain* $K_1 = \Sigma_0^P H^T S_1^{-1}$.
  - Obtain filtered state $A_{1|1} \sim N(\mu_0^P + K_1 v_1, \Sigma_0^P - K_1 S_1 K_1^T) = N(\mu_1^F, \mu_1^F)$.

Kalman Filter!

Obtained by Gaussian algebra.

State Space Model

Example: Linear Gaussian Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

Example: Linear Gaussian Model

State Space Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Filter out all observations $B_1, B_2, \ldots, B_N$.

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Filter out all observations $B_1, B_2, \ldots, B_N$.
- Set smoothed state $A_{N|N} \sim N(\mu_N^F, \Sigma_N^F) = N(\mu_N^S, \Sigma_N^S)$.

State Space Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Filter out all observations $B_1, B_2, \ldots, B_N$.
- Set smoothed state $A_{N|N} \sim N(\mu_N^F, \Sigma_N^F) = N(\mu_N^S, \Sigma_N^S)$.
- Going backwards

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Filter out all observations $B_1, B_2, \ldots, B_N$.
- Set smoothed state $A_{N|N} \sim N(\mu_N^F, \Sigma_N^F) = N(\mu_N^S, \Sigma_N^S)$.
- Going backwards
  - Calculate *smoother gain* $J_{N-1} = \Sigma_{N-1|N-1} G^T (\Sigma_{N|N-1})^{-1}$.

State Space Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Filter out all observations $B_1, B_2, \ldots, B_N$.
- Set smoothed state $A_{N|N} \sim N(\mu_N^F, \Sigma_N^F) = N(\mu_N^S, \Sigma_N^S)$.
- Going backwards
  - Calculate *smoother gain* $J_{N-1} = \Sigma_{N-1|N-1} G^T (\Sigma_{N|N-1})^{-1}$.
  - Calculate *smoother mean* $\mu_{N-1}^S = \mu_{N-1}^F + J_{N-1}(\mu_N^S - \mu_N^P)$.

Example: Linear Gaussian Model

State Space Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Filter out all observations $B_1, B_2, \ldots, B_N$.
- Set smoothed state $A_{N|N} \sim N(\mu_N^F, \Sigma_N^F) = N(\mu_N^S, \Sigma_N^S)$.
- Going backwards
  - Calculate *smoother gain* $J_{N-1} = \Sigma_{N-1|N-1} G^T (\Sigma_{N|N-1})^{-1}$.
  - Calculate *smoother mean* $\mu_{N-1}^S = \mu_{N-1}^F + J_{N-1}(\mu_N^S - \mu_N^P)$.
  - Calculate *smoother covariance* $\Sigma_{N-1}^S = \Sigma_N^S + J_{N-1}(\Sigma_N^P - \Sigma_N^P) J_{N-1}^T$.

State Space Model

$$\begin{cases} A_{n+1} & = GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} & = HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Filter out all observations $B_1, B_2, \ldots, B_N$.
- Set smoothed state $A_{N|N} \sim N(\mu_N^F, \Sigma_N^F) = N(\mu_N^S, \Sigma_N^S)$.
- Going backwards
  - Calculate *smoother gain* $J_{N-1} = \Sigma_{N-1|N-1} G^T (\Sigma_{N|N-1})^{-1}$.
  - Calculate *smoother mean* $\mu_{N-1}^S = \mu_{N-1}^F + J_{N-1}(\mu_N^S - \mu_N^P)$.
  - Calculate *smoother covariance* $\Sigma_{N-1}^S = \Sigma_N^S + J_{N-1}(\Sigma_N^P - \Sigma_N^P) J_{N-1}^T$.
  - Obtain smoothed state $A_{N-1|N-1} \sim N(\mu_{N-1}^S, \Sigma_{N-1}^S)$.

Example: Linear Gaussian Model

State Space Model

$$\begin{cases} A_{n+1} &= GA_n + \xi_n, \qquad \xi_n \sim N(0,Q) \\ B_{n+1} &= HA_{n+1} + \varepsilon_n, \qquad \varepsilon_n \sim N(0,R) \end{cases}$$

- Filter out all observations $B_1, B_2, \ldots, B_N$.
- Set smoothed state $A_{N|N} \sim N(\mu_N^F, \Sigma_N^F) = N(\mu_N^S, \Sigma_N^S)$.
- Going backwards
  - Calculate *smoother gain* $J_{N-1} = \Sigma_{N-1|N-1} G^T (\Sigma_{N|N-1})^{-1}$.
  - Calculate *smoother mean* $\mu_{N-1}^S = \mu_{N-1}^F + J_{N-1}(\mu_N^S - \mu_N^P)$.
  - Calculate *smoother covariance* $\Sigma_{N-1}^S = \Sigma_N^S + J_{N-1}(\Sigma_N^P - \Sigma_N^P) J_{N-1}^T$.
  - Obtain smoothed state $A_{N-1|N-1} \sim N(\mu_{N-1}^S, \Sigma_{N-1}^S)$.

RTS Smoother!

Obtained by Gaussian algebra.

Probabilistic ODE Solver

Prior

$$d\mathbf{X}_t = F\mathbf{X}_t dt + L dW_t$$

+

Data

$$\{z_k = \dot{C}\mathbf{X}_{t_k} - f(C\mathbf{X}_{t_k}, t_k) = 0\}_{k=1}^N$$

Probabilistic ODE Solver

Prior

$$dX_t = FX_t dt + LdW_t$$

+

Data

$$\{z_k = \dot{C}\mathbf{X}_{t_k} - f(C\mathbf{X}_{t_k}, t_k) = 0\}_{k=1}^{N}$$

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} & = G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} & = \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

Probabilistic ODE Solver

<u>Prior</u>

$$d\mathbf{X}_t = F\mathbf{X}_t dt + L dW_t$$

+

<u>Data</u>

$$\{z_k = \dot{C}\mathbf{X}_{t_k} - f(C\mathbf{X}_{t_k}, t_k) = 0\}_{k=1}^{N}$$

Exact transition of Linear SDE

<u>(Sometimes) Linear Gaussian Model</u>

$$\begin{cases} \mathbf{X}_{n+1} & = G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} & = \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

Probabilistic ODE Solver

Prior

$$dX_t = FX_t dt + L dW_t$$

+

Data

$$\{z_k = \dot{C}X_{t_k} - f(CX_{t_k}, t_k) = 0\}_{k=1}^{N}$$

Exact transition of Linear SDE

(Sometimes) Linear Gaussian Model

$$\begin{cases} X_{n+1} = G_h X_n + \xi_n, & \xi_n \sim N(0, Q_h) \\ Z_{n+1} = \dot{C}X_{n+1} - f(CX_{n+1}, t_{n+1}) + \varepsilon_n, & \varepsilon_n \sim N(0, R) \end{cases}$$

Could be zero

Probabilistic ODE Solver

Prior

$$dX_t = FX_t dt + L dW_t$$

+

Data

$$\{z_k = \dot{C}X_{t_k} - f(CX_{t_k}, t_k) = 0\}_{k=1}^{N}$$

Exact transition of Linear SDE

(Sometimes) Linear Gaussian Model

$$\begin{cases} X_{n+1} &= G_h X_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} &= \dot{C}X_{n+1} - f(CX_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

(Potential) Nonlinearity Culprit!

Could be zero

Probabilistic ODE Solver

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} & = G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} & = \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

Probabilistic ODE Solver

<u>(Sometimes) Linear Gaussian Model</u>

$$\begin{cases} \mathbf{X}_{n+1} & = G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} & = \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

- When f is non-linear, we can either (a) linearise it [EKF], or (b) use particle approximations [PF].

Probabilistic ODE Solver

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} &= G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} &= \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

- When f is non-linear, we can either (a) linearise it [EKF], or (b) use particle approximations [PF].
- While both are approximations, particle filter is *asymptotically true* (unbiased and consistent) in the number of particles while EKF is *not*.

Probabilistic ODE Solver

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} &= G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} &= \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

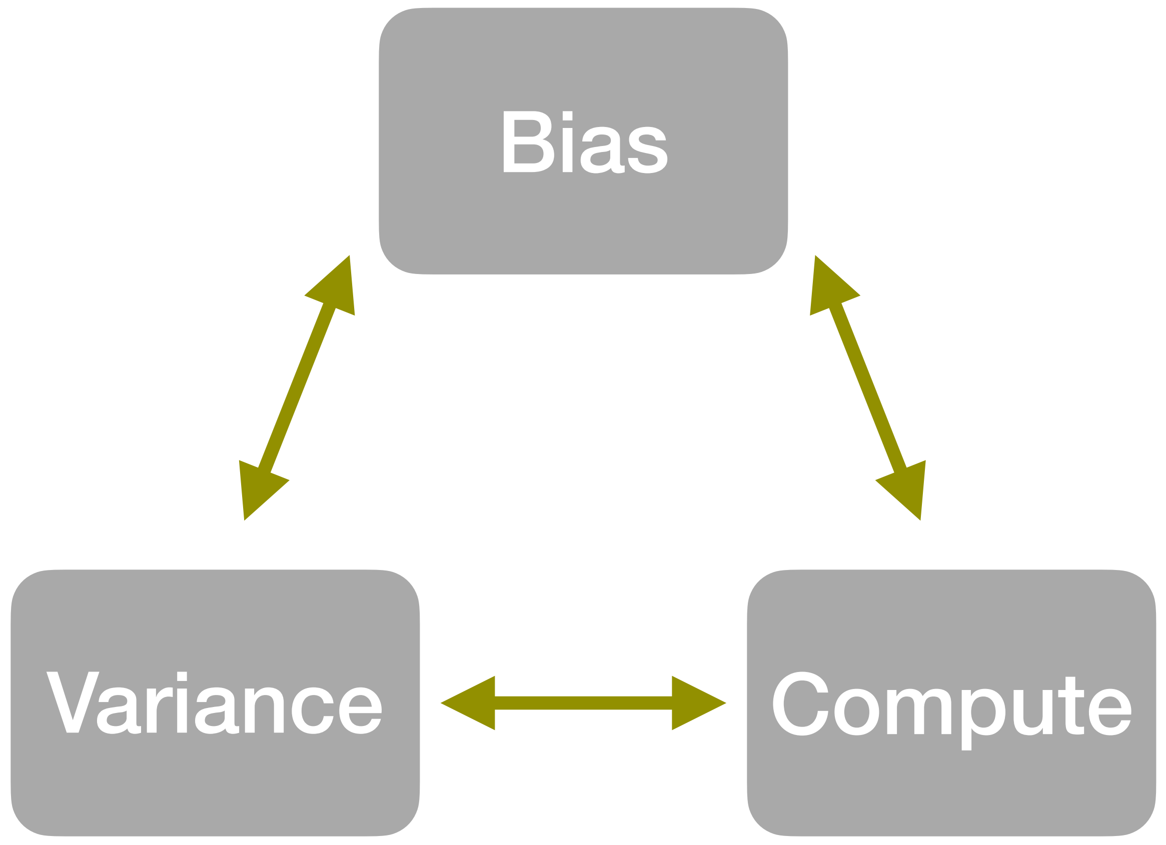
- When f is non-linear, we can either (a) linearise it [EKF], or (b) use particle approximations [PF].
- While both are approximations, particle filter is *asymptotically true* (unbiased and consistent) in the number of particles while EKF is *not*.
- However, EKF is much faster to implement.

Probabilistic ODE Solver

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} &= G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} &= \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

- When f is non-linear, we can either (a) linearise it [EKF], or (b) use particle approximations [PF].
- While both are approximations, particle filter is *asymptotically true* (unbiased and consistent) in the number of particles while EKF is *not*.
- However, EKF is much faster to implement.



Probabilistic ODE Solver

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} & = G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} & = \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

Probabilistic ODE Solver

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} & = G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} & = \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

EKF linearises by replacing nonlinearities with truncated Taylor expansion.

Probabilistic ODE Solver

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} &= G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} &= \dot{C} \mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

EKF linearises by replacing nonlinearities with truncated Taylor expansion.

[EK0] $\qquad f(C\mathbf{X}_k, t_k) \approx f(C\mu_k^P, t_k)$

Probabilistic ODE Solver

(Sometimes) Linear Gaussian Model

$$\begin{cases} \mathbf{X}_{n+1} & = G_h \mathbf{X}_n + \xi_n, \qquad \xi_n \sim N(0, Q_h) \\ Z_{n+1} & = \dot{C}\mathbf{X}_{n+1} - f(C\mathbf{X}_{n+1}, t_{n+1}) + \varepsilon_n, \qquad \varepsilon_n \sim N(0, R) \end{cases}$$

EKF linearises by replacing nonlinearities with truncated Taylor expansion.

[EK0] $$f(C\mathbf{X}_k, t_k) \approx f(C\mu_k^P, t_k)$$

[EK1] $$f(C\mathbf{X}_k, t_k) \approx f(C\mu_k^P, t_k) + J_f(C\mu_k^P, t_k)C(\mathbf{X}_k - \mu_k^P)$$

J_f is the Jacobian of $y \mapsto f(y, t)$

Implementation Details

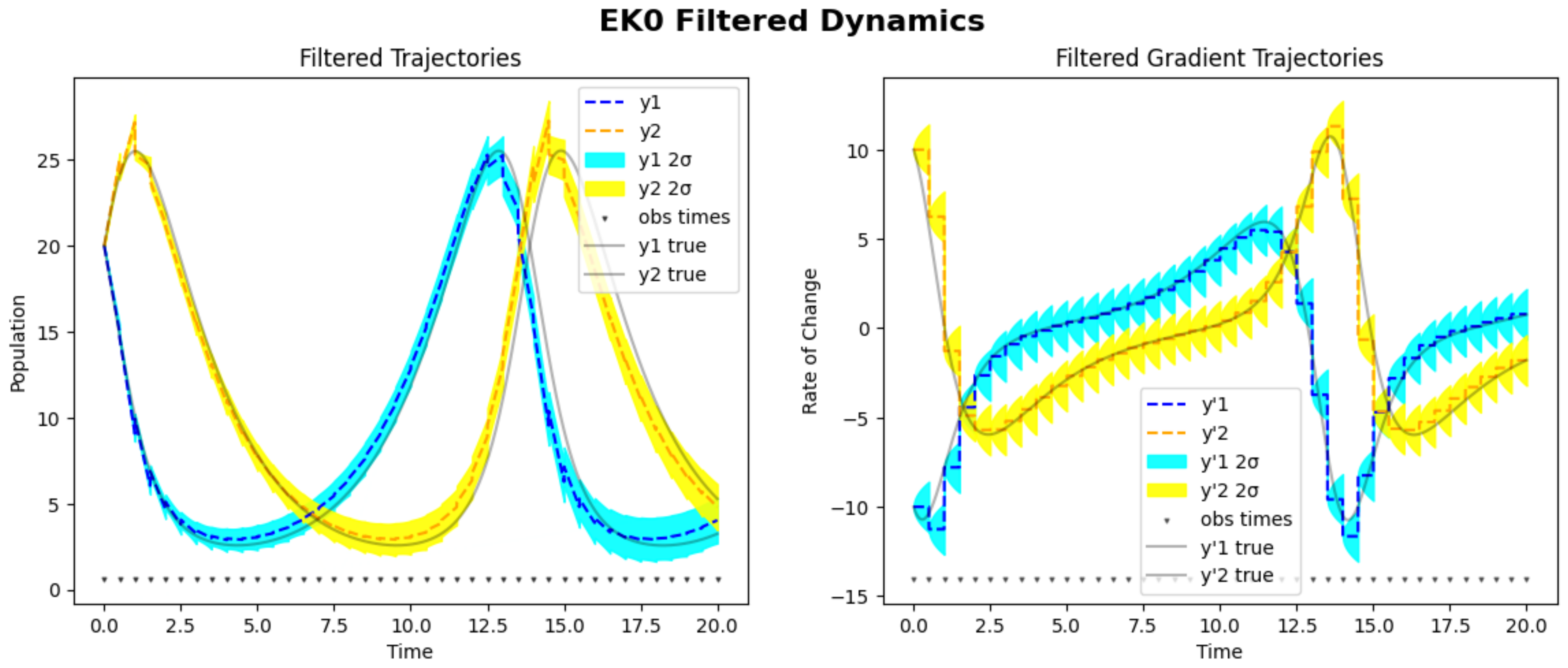
- d-Dimensional trajectories can be incorporated by expanding the matrices using Kronecker product with identity matrix I_d.
- One can include timestamps with no observations too, just propagate and not assimilate.
- When observation times are not uniform, make sure the transition matrices are re-computed.

Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

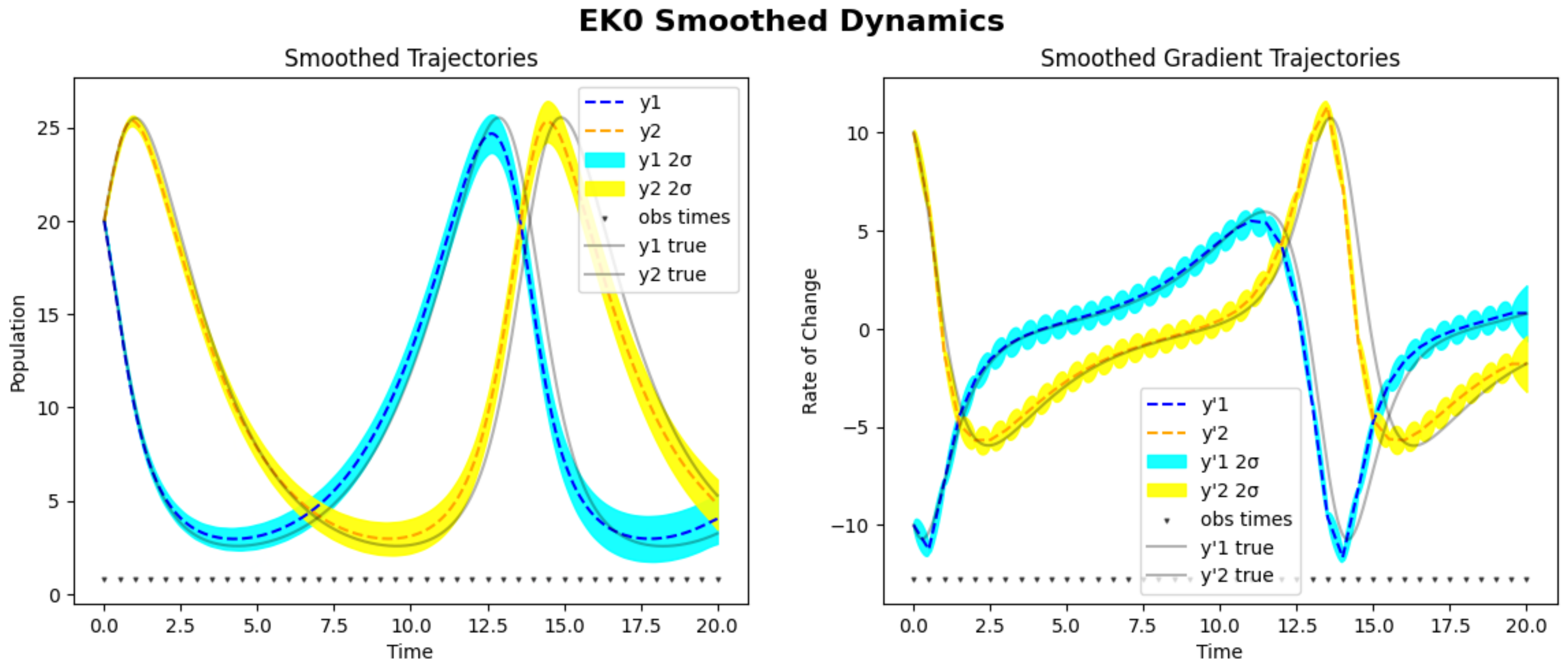


Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$



Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
  - Sequential Data Acquisition
- How much do we have to pay for these uncertainties?

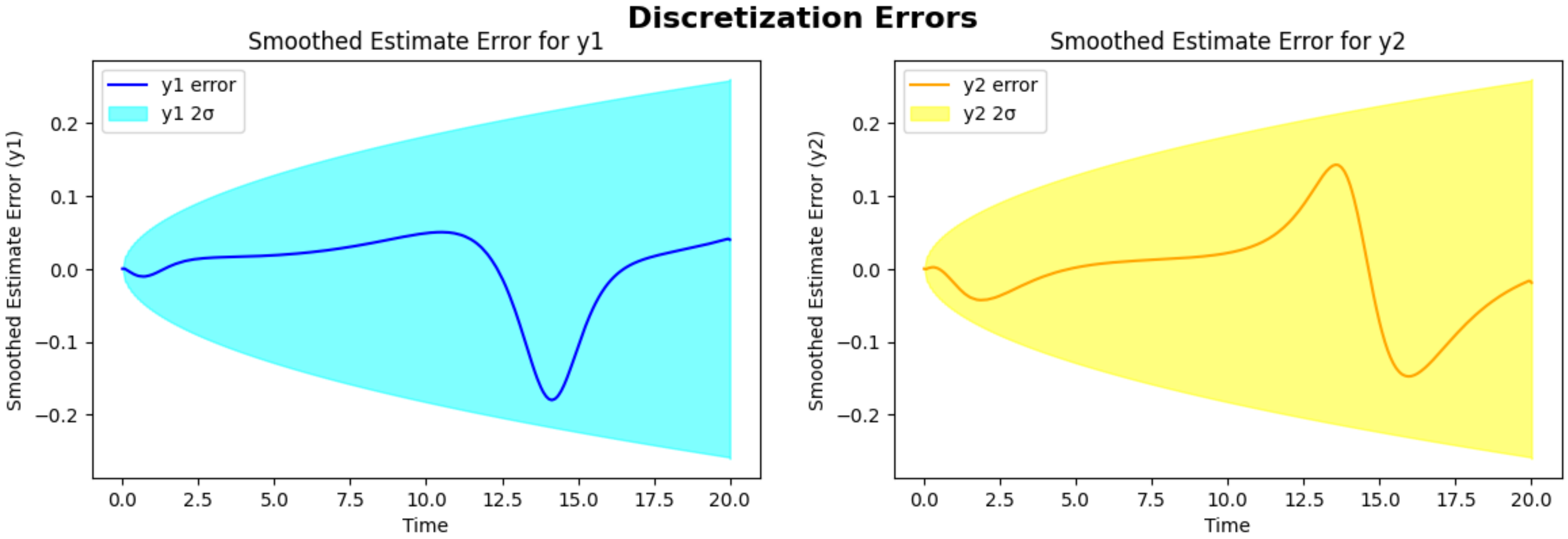
Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)



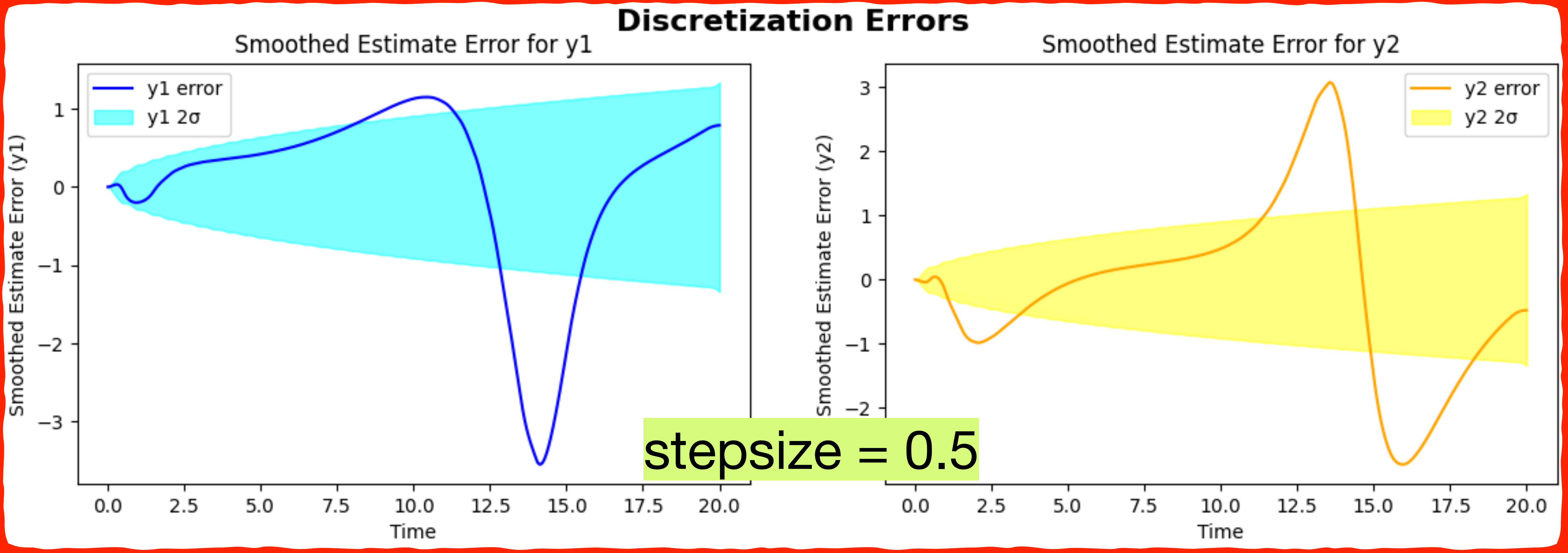
Prob

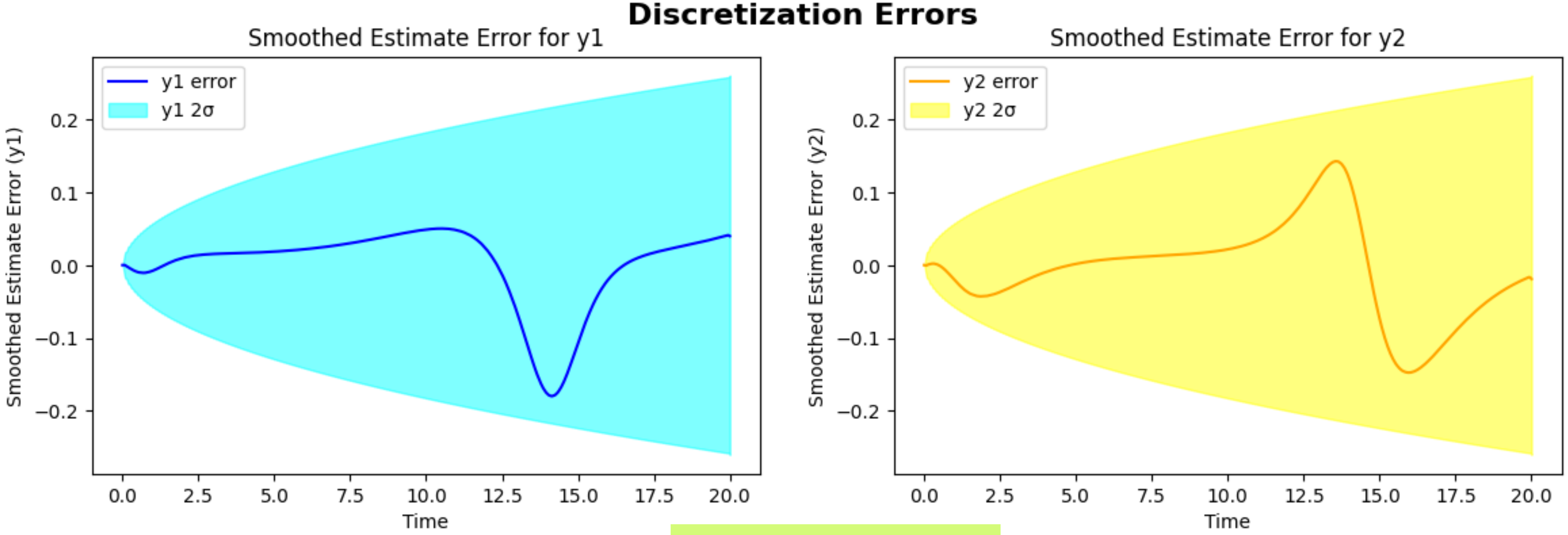
$y_1(t)y_2(t)$

$5y_1(t)y_2(t)$

)

- What c
  - Epis





stepsize = 0.1

Probabilistic ODE Solver

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples

Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples

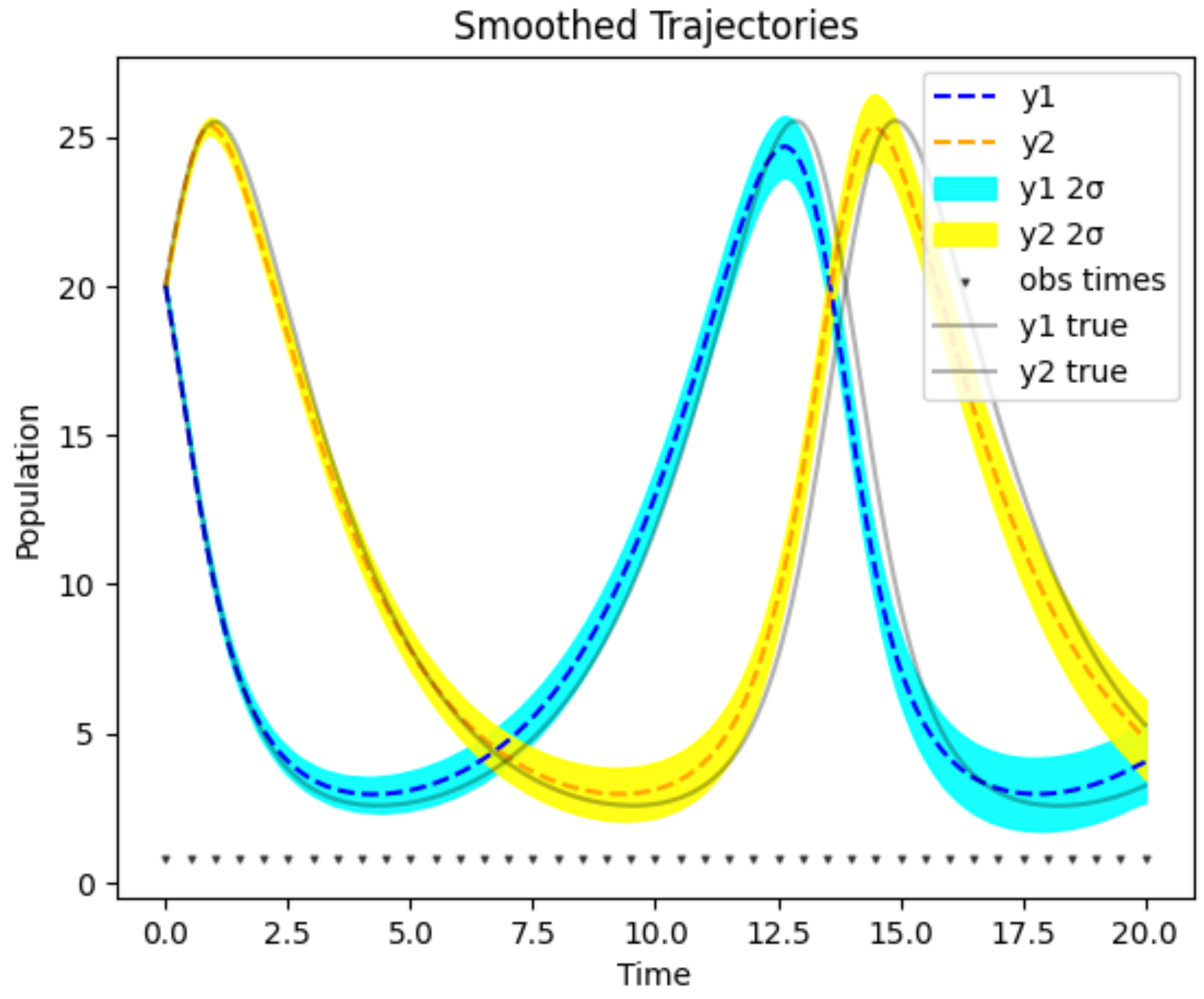
Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples



E.g. we are 95% confident that $y_1(20) \in [-0.624,\ 2.202]$

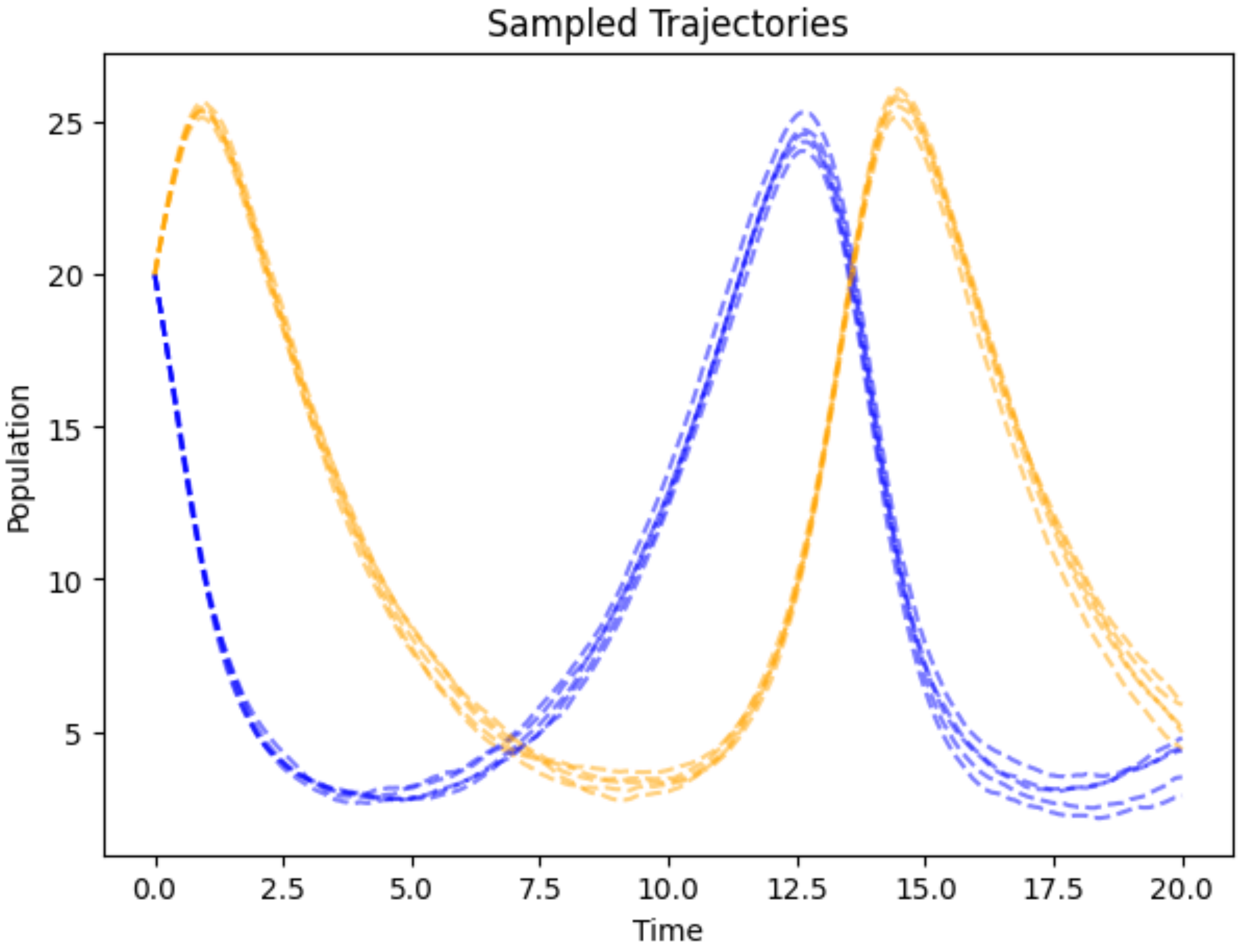
Probabilistic ODE Solver

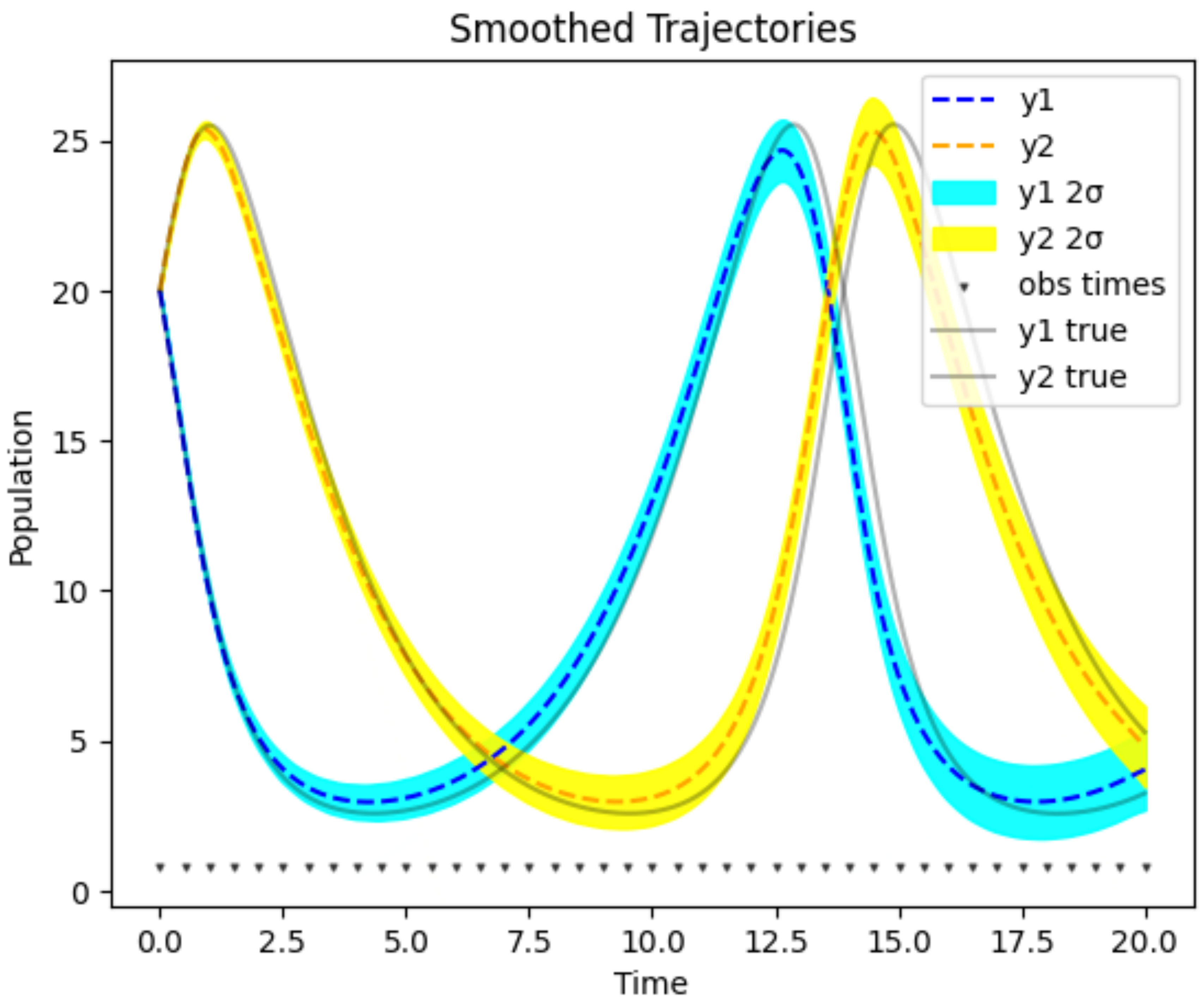
Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

- W
  - ible)
- H
  -

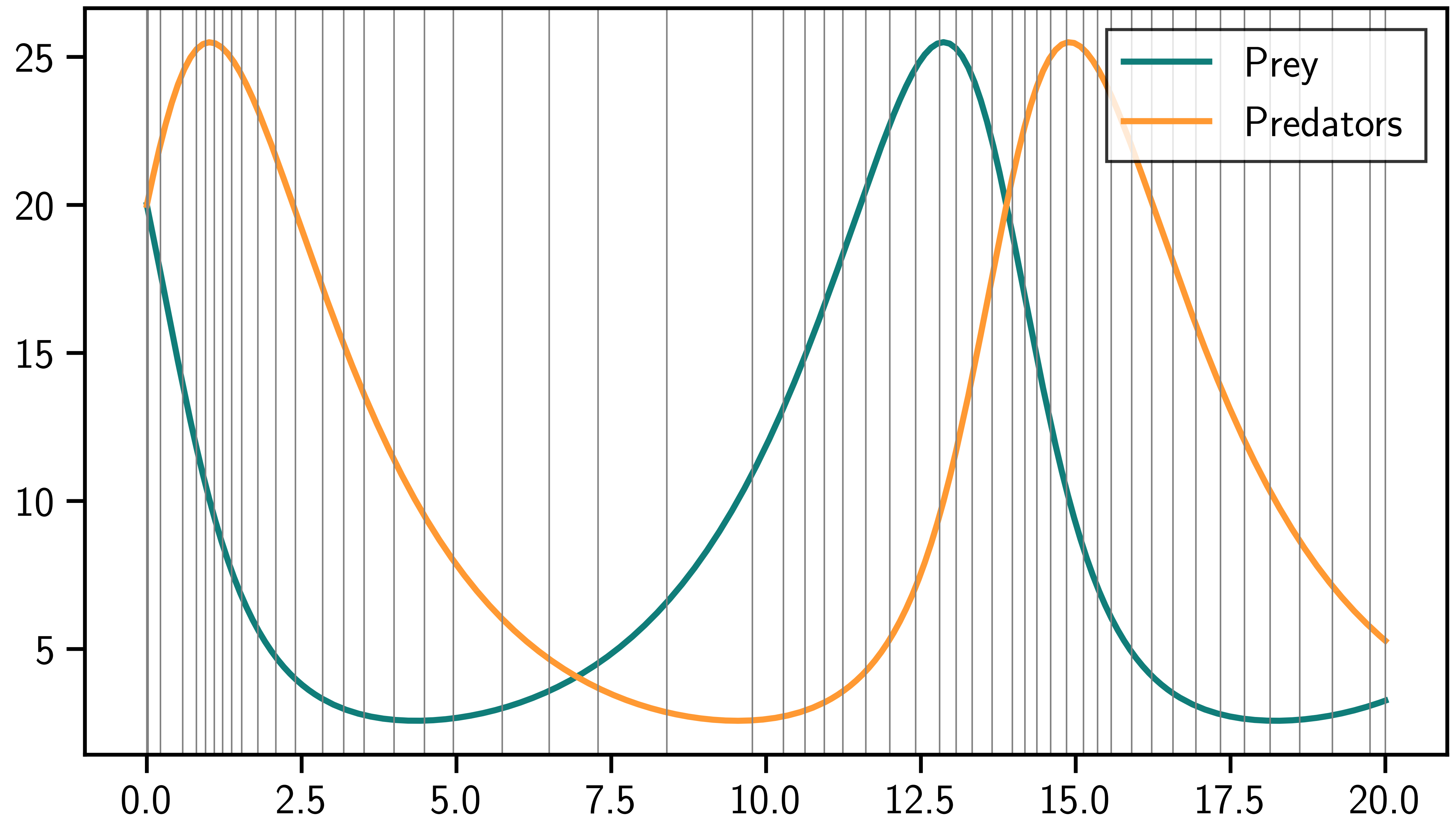




E.g. we are 95% confident that $y_1(20) \in [-0.624,\ 2.202]$

Probabilistic ODE Solver

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
  - Sequential Data Acquisition
  - Adaptive stepsizes



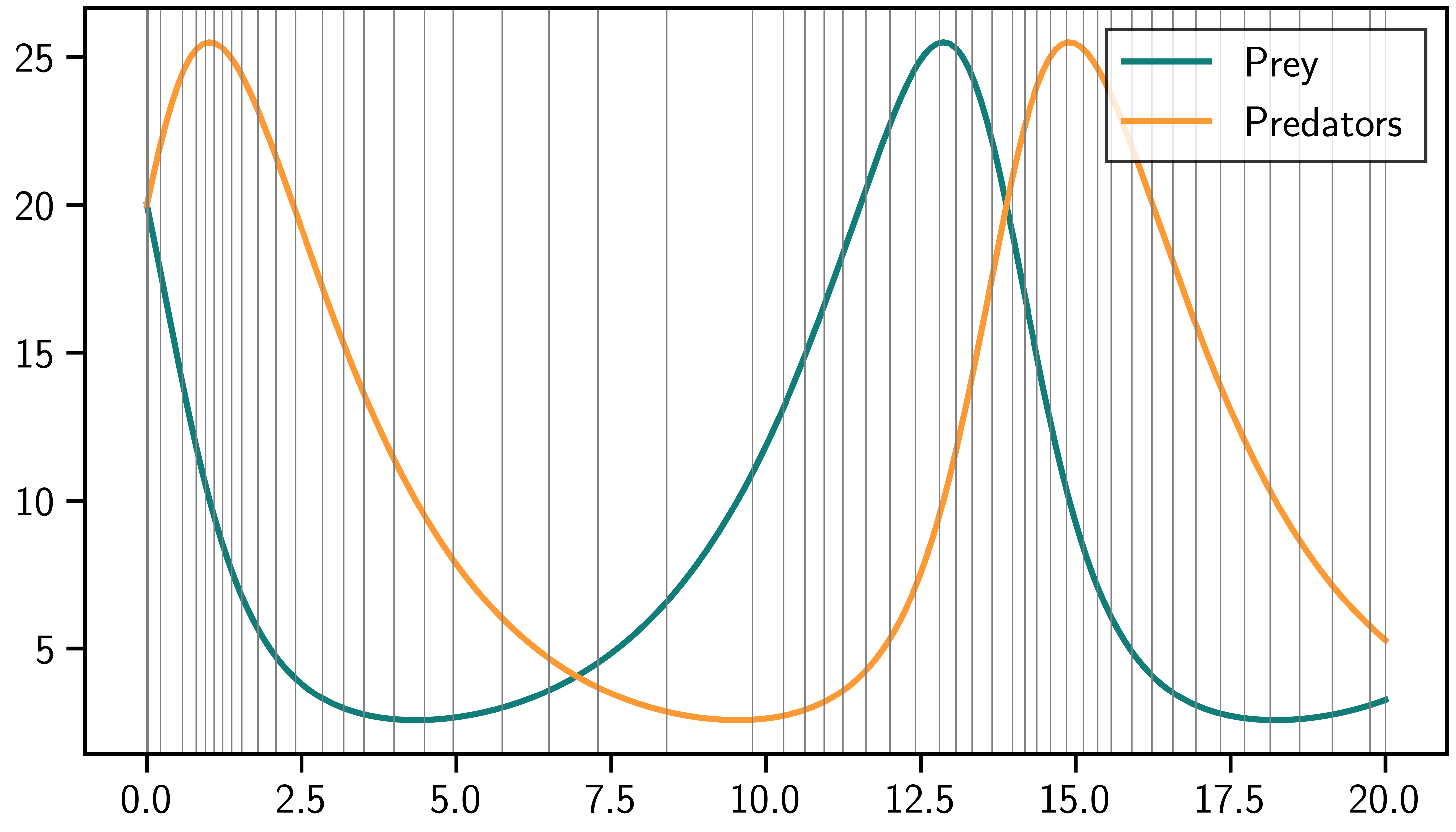
Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
  - Sequential Data Acquisition
  - Adaptive stepsizes



Probabilistic ODE Solver

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
  - Sequential Data Acquisition
- How much do we have to pay for these uncertainties?

Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
  - Sequential Data Acquisition
- How much do we have to pay for these uncertainties?

Probabilistic ODE Solver

Lotka-Volterra

$$\begin{bmatrix} dy_1(t) \\ dy_2(t) \end{bmatrix} = \begin{bmatrix} 0.5y_1(t) - 0.05y_1(t)y_2(t) \\ -0.5y_2(t) + 0.05y_1(t)y_2(t) \end{bmatrix}$$

$$y_1(0) = y_2(0) = 20$$

- What do these uncertainties mean?
  - Epistemic (Reducible) & Aleatoric (Irreducible)
- How are uncertainties represented?
  - Credible Intervals, Posterior Samples
- What can we do with these uncertainties?
  - Sequential Data Acquisition
- How much do we have to pay for these uncertainties?

Roughly, for the same stepsize, probabilistic solvers take 10s-X compute.

and much more time to code up …

Probabilistic ODE Solver

Remark (Relation to Bayesian quadrature). *Before introducing more ODE filters, let us briefly clarify the relation to Bayesian quadrature (BQ) – namely that the EKF0/EKS0 is a* generalisation of BQ *in the following sense: if the ODE is really just an integral (i.e. $x'(t) = g(t)$), then its solution is given by*

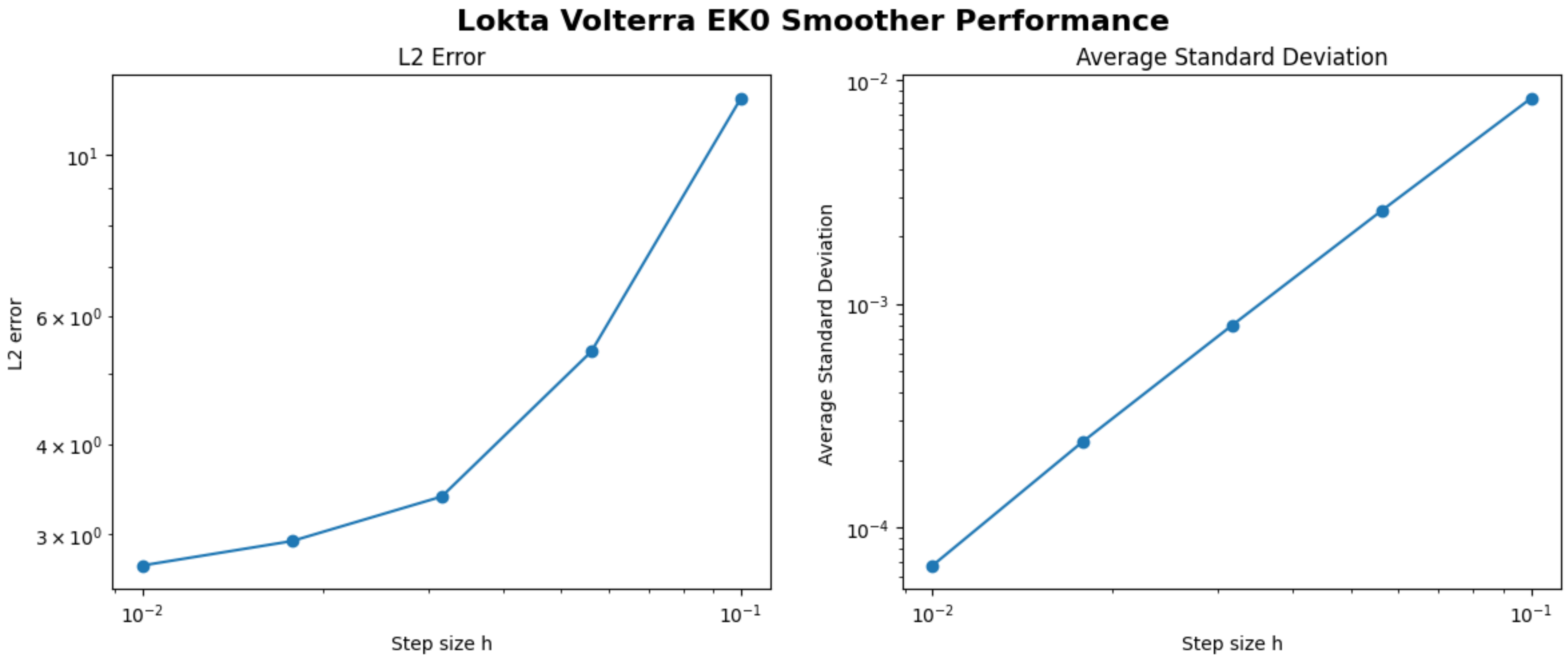
$$x(t) = x_0 + \int_0^t g(s)\,\mathrm{d}s. \tag{38.27}$$

Thus, computing $x(t)$ by approximating the integral in Eq. (38.27) with the Kalman-filter version of BQ (Algorithm 11.2 from §11.2) is equivalent to solving the ODE

$$x'(s) = g(s),\ s \in [0,t], \qquad \textit{with initial value } x(0) = x_0,$$

by the EKF0 or EKS0.[31]

Probabilistic ODE Solver



Convergence results exist too …

Alternative Probabilistic ODE Solvers

Collocation is not the only way …

Alternative Probabilistic ODE Solvers

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Collocation is not the only way …

Euler's Method $$x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$$

Alternative Probabilistic ODE Solvers

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Collocation is not the only way …

Euler's Method $\quad x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$

Option 1: Inject noise to drift.

Alternative Probabilistic ODE Solvers

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Collocation is not the only way …

Euler's Method $\quad x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$

Option 1: Inject noise to drift.

Home > Statistics and Computing > Article

Statistical analysis of differential equations: introducing probability measures on numerical solutions

Open access | Published: 02 June 2016

Volume 27, pages 1065–1082, (2017) Cite this article

Patrick R. Conrad, Mark Girolami, Simo Särkkä, Andrew Stuart & Konstantinos Zygalakis

Alternative Probabilistic ODE Solvers

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Collocation is not the only way …

Euler's Method $$x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$$

Home › Statistics and Computing › Article

Statistical analysis of differential equations: introducing probability measures on numerical solutions

Open access | Published: 02 June 2016

Volume 27, pages 1065–1082, (2017) Cite this article

You have full access to this open access article

Download PDF Save article

Patrick R. Conrad, Mark Girolami, Simo Särkkä, Andrew Stuart & Konstantinos Zygalakis

Option 1: Inject noise to drift.

basically turn ODEs into SDEs

Alternative Probabilistic ODE Solvers

Collocation is not the only way …

Alternative Probabilistic ODE Solvers

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Collocation is not the only way …

Euler's Method $$x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$$

Alternative Probabilistic ODE Solvers

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Collocation is not the only way …

Euler's Method $$x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$$

Option 2: Random stepsizes

Alternative Probabilistic ODE Solvers

$$\frac{d}{dt}y(t) = f(y(t), t), \qquad y(0) = y_0$$

Collocation is not the only way …

Euler's Method $\quad x_{n+h} = x_n + f(x_n, t_n) \cdot h, \qquad x_0 = y_0$

Option 2: Random stepsizes

Home > Statistics and Computing > Article

Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration

Published: 14 February 2020

Volume 30, pages 907–932, (2020) Cite this article

Assyr Abdulle & Giacomo Garegnani

Wait, how about combining Physics and data?

Wait, how about combining Physics and data?

Stay tuned for my CSML talk (23 Apr) !