Aalto University
School of Electrical
Engineering

Probabilistic Solvers for ODEs and PDEs

Simo Särkkä

Aalto University, Finland

November, 2022

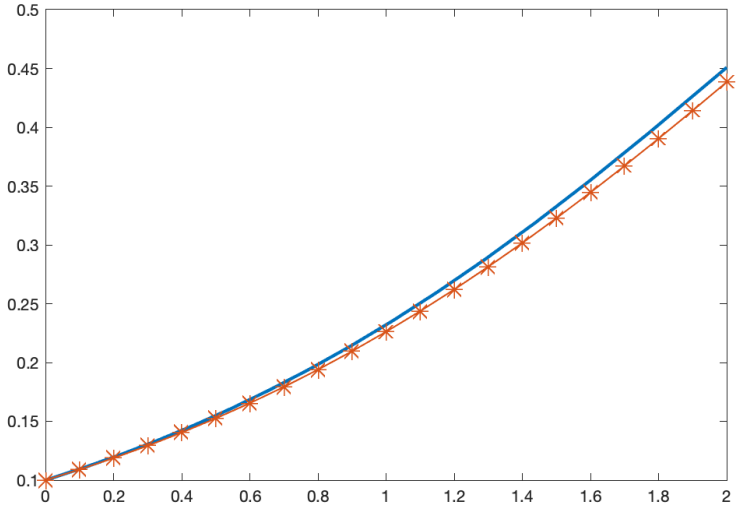
Contents

Problem formulation

- Consider a ordinary differential equation (ODE) for $\mathbf{x}(t) \in \mathbb{R}^d$:

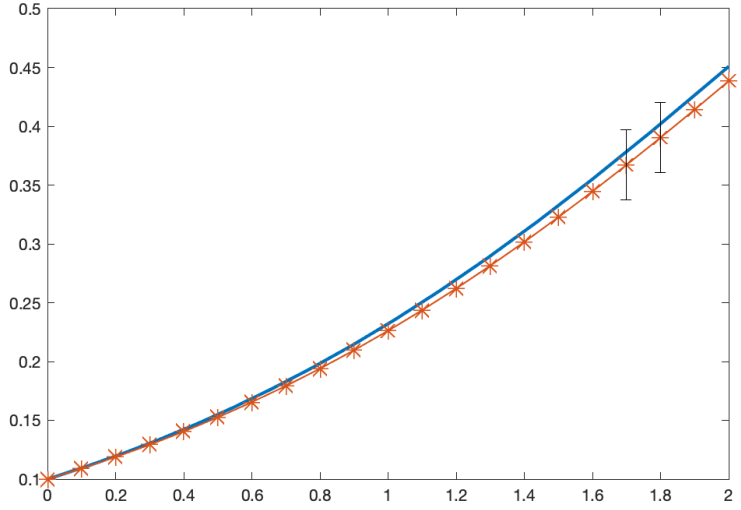
$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(t), t), \qquad \mathbf{x}(0) = \mathbf{x}_0.$$

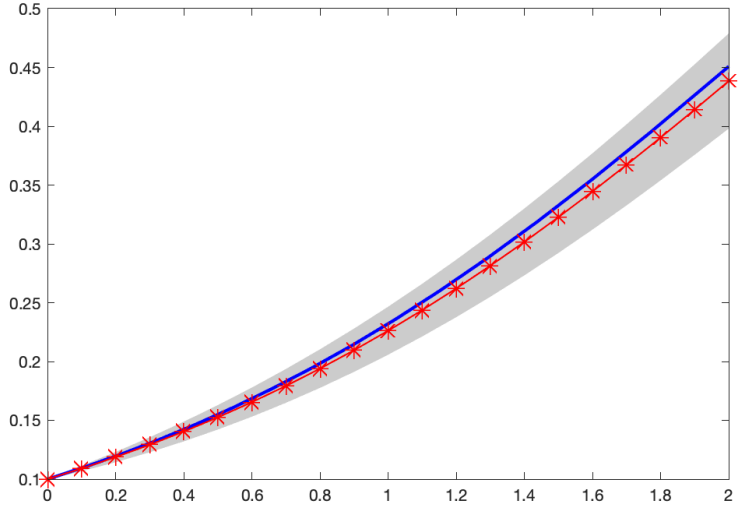
- The aim is to find an approximate solution $\hat{\mathbf{x}}(t)$ such that $\hat{\mathbf{x}}(t_n) \approx \mathbf{x}(t_n)$ on some points $0 = t_0 < t_1 < \cdots < t_N = T$.
- Function $\mathbf{f}(\cdot)$ is only evaluated at points $\hat{\mathbf{x}}(t_n)$, and some nearby points.
- The approximate solution $\hat{\mathbf{x}}(t)$ is called a numerical solver.



Problem formulation (cont.)

- Classically the error $\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$ is quantified in terms of worst-case error.
- The error is typically quantified using Taylor's theorem.
- In probabilistic ODE solvers the error is quantified probabilistically.
- The probabilistic solvers also have worst-case bounds in terms of Sobolev norms.





Classical ODE solving is polynomial fitting

- Classical ODE solvers can be seen as piece-wise polynomial approximations to the solution.
- Euler method is a piece-wise linear approximation:

$$x(t) \approx x(t_0) + \frac{dx(t_0)}{dt}(t - t_0) = x(t_0) + f(x(t_0))(t - t_0).$$

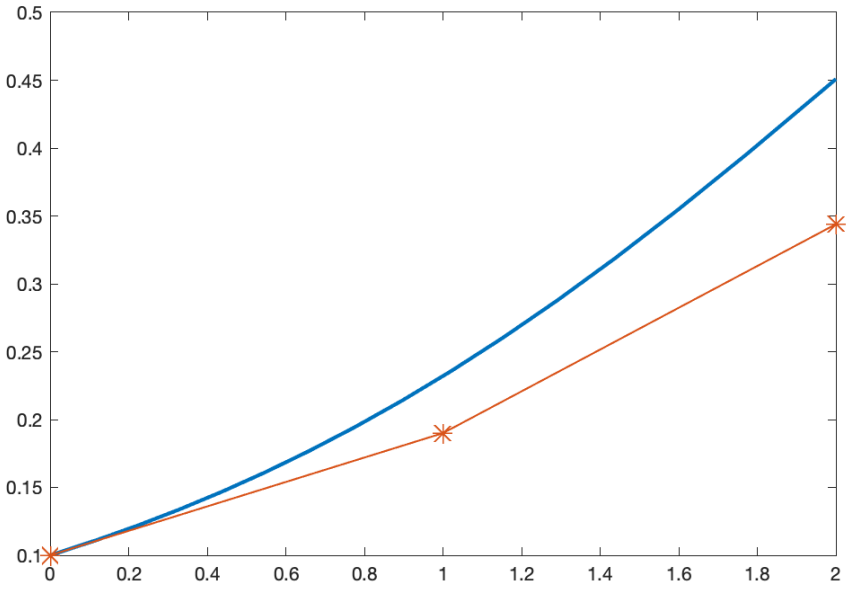
- Runge–Kutta methods are based on higher order polynomial fitting:

$$\begin{aligned} x(t) &\approx x(t_0) + \frac{dx(t_0)}{dt}(t - t_0) + \frac{1}{2}\frac{d^2x(t_0)}{dt^2}(t - t_0) + \cdots \\ &= c_0 + c_1(t - t_0) + c_2(t - t_0)^2 + \cdots \end{aligned}$$

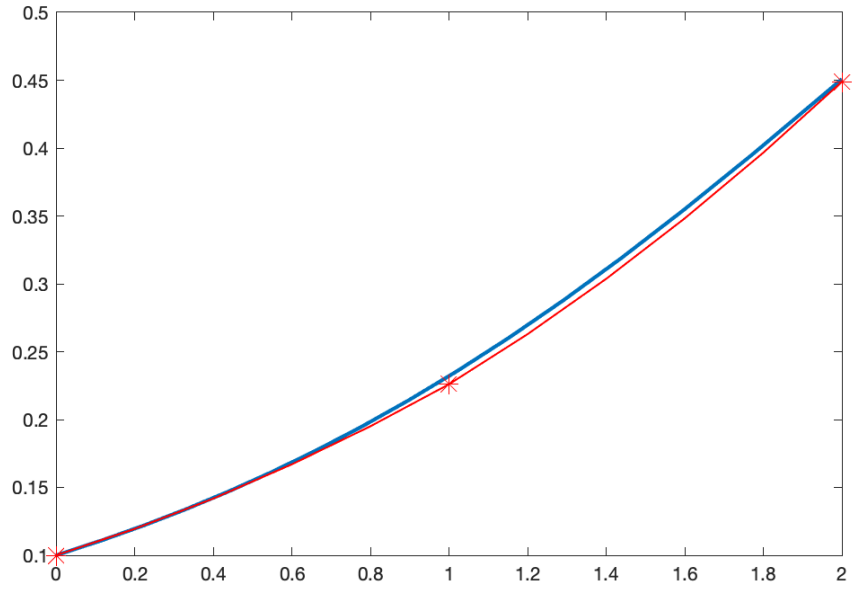
- The worst-case error analysis possible using Taylor's theorem.

Classical ODE solving is polynomial fitting (cont.)

Linear approximation:

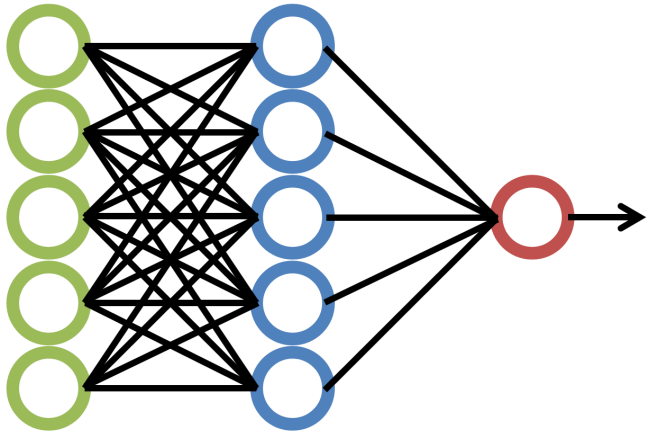


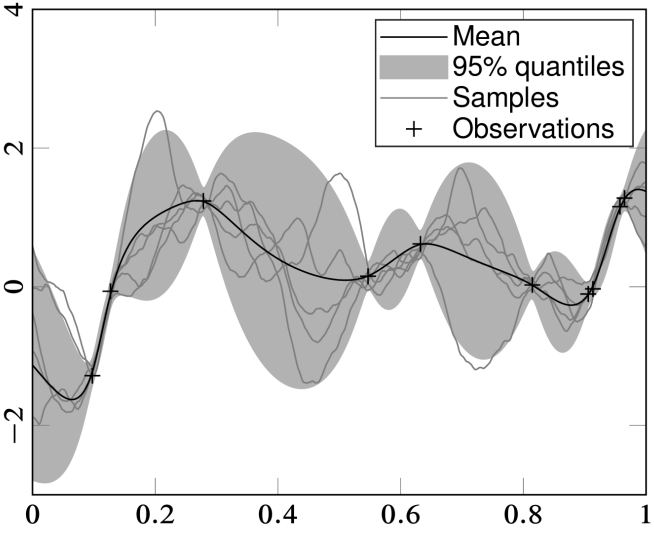
Polynomial approximation:



Going beyond polynomial fitting

- Machine learning and statistics provide other than polynomial regression models.
- For example, neural networks are flexible, but slow to train (= fit).
- Gaussian processes (GPs) in turn are fast to fit to data, and they also provide error bounds.
- Probabilistic ODE solvers replace the polynomial approximation with a GP.





Gaussian process regression [1/5]

- Gaussian process regression considers predicting the value of an unknown function

$$y = g(x)$$

  at a certain test point x^* based on a finite number of training samples (x_j, y_j) observed from it.
- As we are dealing with functions of time, let's replace x with t:

$$y = g(t).$$

- In classic regression, we postulate parametric form of $g(t; \theta)$ and estimate the parameters θ.
- In GP regression, we instead assume that $g(t)$ is a sample from a Gaussian process with a covariance function, e.g.,

$$K(t, t') = s^2 \exp\left(-\frac{1}{2\ell^2}||t - t'||^2\right).$$

Gaussian process regression [2/5]

- Let's denote the vector of observed points as $\mathbf{y} = (y_1, \ldots, y_N)$, and test point value as $y^* = g(t^*)$.
- Gaussian process assumption implies that

$$\begin{pmatrix} \mathbf{y} \\ y^* \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{K}(t_{1:N}, t_{1:N}) & \mathbf{K}^\mathsf{T}(t^*, t_{1:N}) \\ \mathbf{K}(t^*, t_{1:N}) & K(t^*, t^*) \end{pmatrix} \right)$$

  where
  - $\mathbf{K}(t_{1:N}, t_{1:N}) = [K(t_i, t_j)]$ is the covariance of observed points,
  - $K(t^*, t^*)$ is the (co)variance of the test point,
  - $\mathbf{K}(t^*, t_{1:N}) = [K(t^*, t_j)]$ is the cross covariance.
- By using the computation rules of Gaussian distributions

$$\mathrm{E}[y^* \mid \mathbf{y}] = \mathbf{K}(t^*, t_{1:N})\, \mathbf{K}^{-1}(t_{1:N}, t_{1:N})\, \mathbf{y}$$

$$\mathrm{Var}[y^* \mid \mathbf{y}] = K(t^*, t^*) - \mathbf{K}(t^*, t_{1:N})\, \mathbf{K}^{-1}(t_{1:N}, t_{1:N})\, \mathbf{K}^\mathsf{T}(t^*, t_{1:N}).$$

- These equations can be used for interpolating or extrapolating the value of $y^* = g(t^*)$ at any test point t^*.

Gaussian process regression [3/5]

- In practice, the measurements usually have noise:

$$y_n = g(t_n) + e_n, \qquad e_n \sim \mathcal{N}(0, \sigma^2).$$

- We want to estimate the value of the "clean" function $g(t^*)$ at a test point t^*.
- Due to the Gaussian process assumption we now get

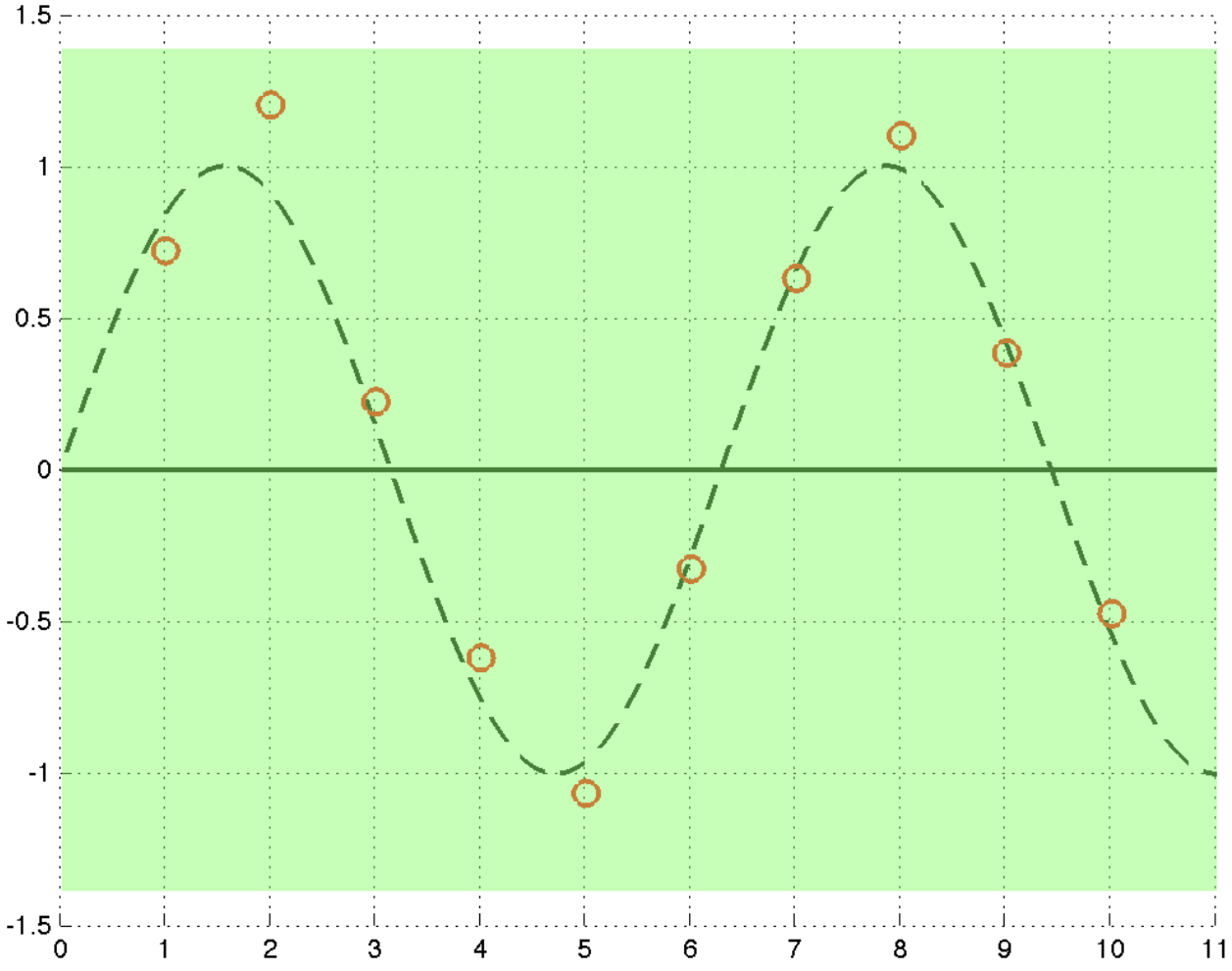
$$\begin{pmatrix} \mathbf{y} \\ g(t^*) \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{K}(t_{1:N}, t_{1:N}) + \sigma^2 \mathbf{I} & \mathbf{K}^{\mathsf{T}}(t^*, t_{1:N}) \\ \mathbf{K}(t^*, t_{1:N}) & K(t^*, t^*) \end{pmatrix} \right)$$

- The conditional mean and variance are given as

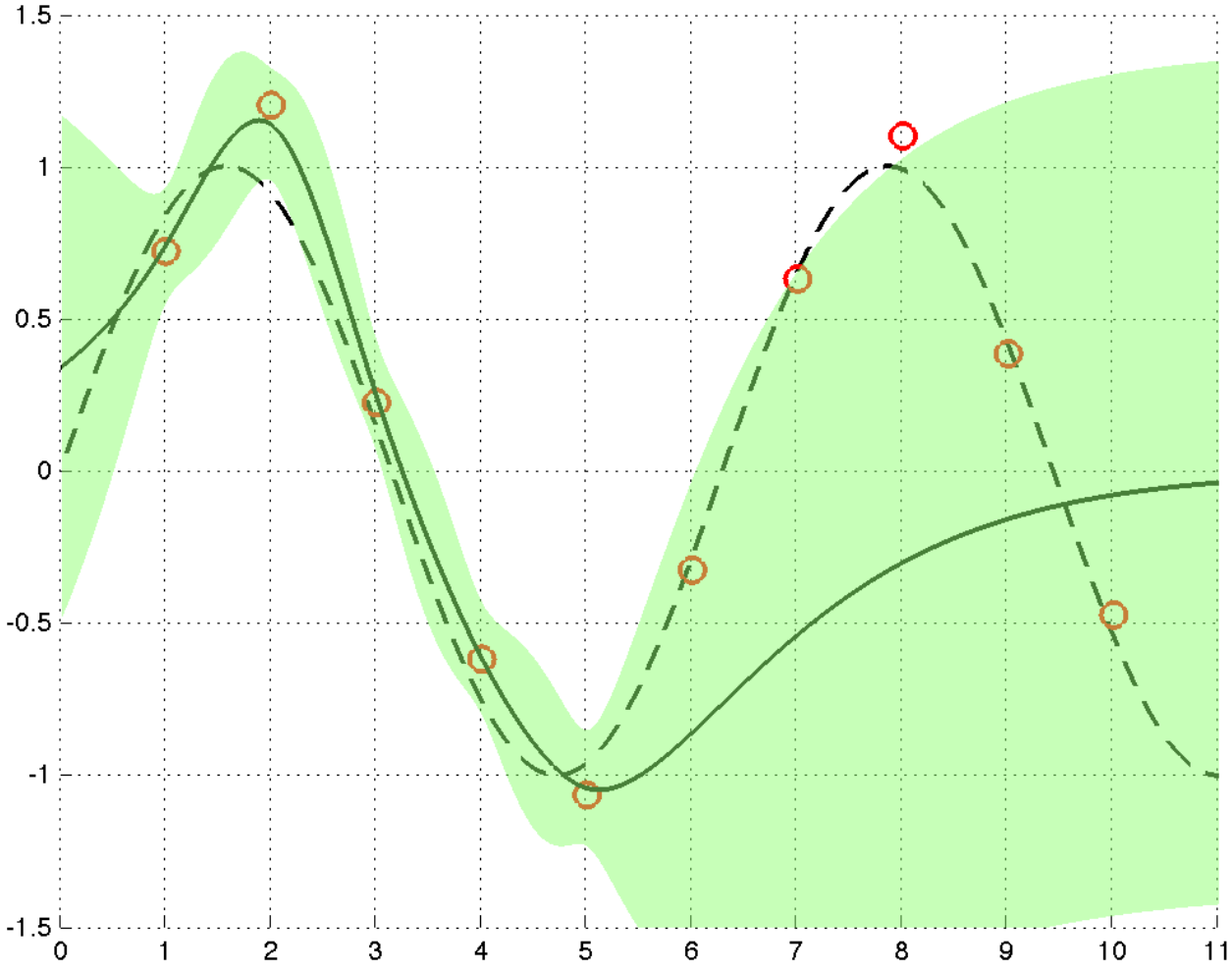
$$\begin{aligned} \mathrm{E}[g(t^*) \,|\, \mathbf{y}] &= \mathbf{K}(t^*, t_{1:N}) \left(\mathbf{K}(t_{1:N}, t_{1:N}) + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y} \\ \mathrm{Var}[g(t^*) \,|\, \mathbf{y}] &= K(t^*, t^*) \\ &\quad - \mathbf{K}(t^*, t_{1:N}) \left(\mathbf{K}(t_{1:N}, t_{1:N}) + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{K}^{\mathsf{T}}(t^*, t_{1:N}). \end{aligned}$$

- These are the Gaussian process regression equations in their typical form - scalar special cases though.

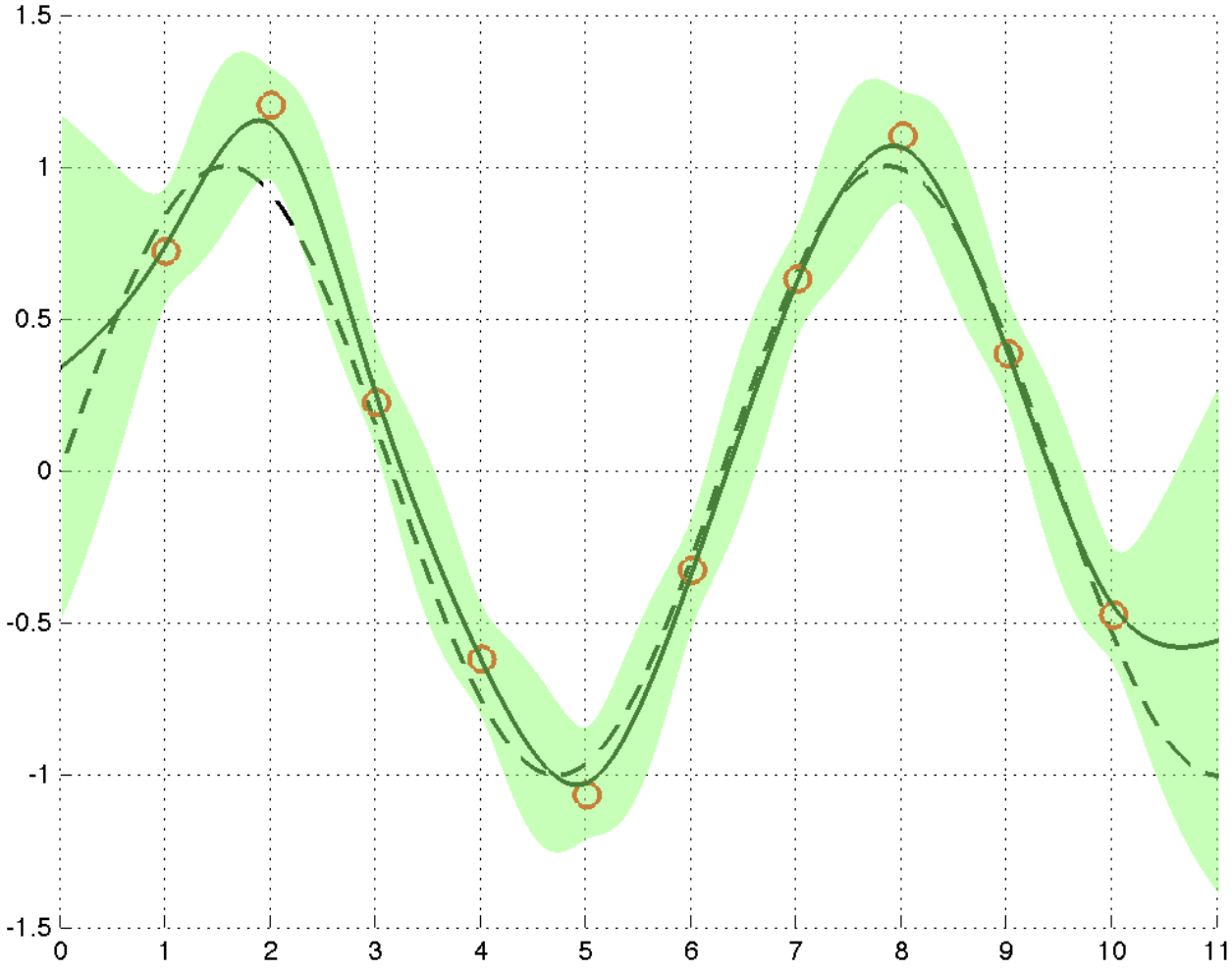
Gaussian process regression [4/5]



Gaussian process regression [4/5]



Gaussian process regression [4/5]



Gaussian process regression [5/5]

- We can also do GP regression with derivative measurements

$$\dot{y}_n = \frac{dg}{dt}(t_n) + e_n, \qquad e_n \sim \mathcal{N}(0, \sigma^2).$$

- The conditional mean and variance only change a bit

$$\mathrm{E}[g(t^*) \,|\, \mathbf{z}] = \frac{\partial \mathbf{K}}{\partial t}(t^*, t_{1:N}) \left(\frac{\partial^2 \mathbf{K}(t_{1:N}, t_{1:N})}{\partial t \, \partial t'} + \sigma^2 \mathbf{I} \right)^{-1} \dot{\mathbf{y}}$$

$$\mathrm{Var}[g(t^*) \,|\, \mathbf{z}] = \frac{\partial \mathbf{K}}{\partial t}(t^*, t_{1:N}) - \frac{\partial \mathbf{K}}{\partial t}(t^*, t_{1:N}) \left(\frac{\partial^2 \mathbf{K}(t_{1:N}, t_{1:N})}{\partial t \, \partial t'} + \sigma^2 \mathbf{I} \right)^{-1} \frac{\partial \mathbf{K}^\mathsf{T}}{\partial t}(t^*, t_{1:N}).$$

GP solution to an ODE

- Let us consider an ODE

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), t), \qquad x(0) = x_0.$$

- We now aim to use a GP regressor

$$g(t) \sim \mathcal{GP}(0, k(t, t'))$$

  to approximate the solution $x(t) \approx g(t)$.
- The approach is to condition the Gaussian process on the ODE at the selected grid:

$$\frac{dg}{dt}(t_n) - f(g(t_n), t_n) = 0.$$

- This defines a non-linear likelihood (actually a constraint) that can be handled with non-linear GP methods.

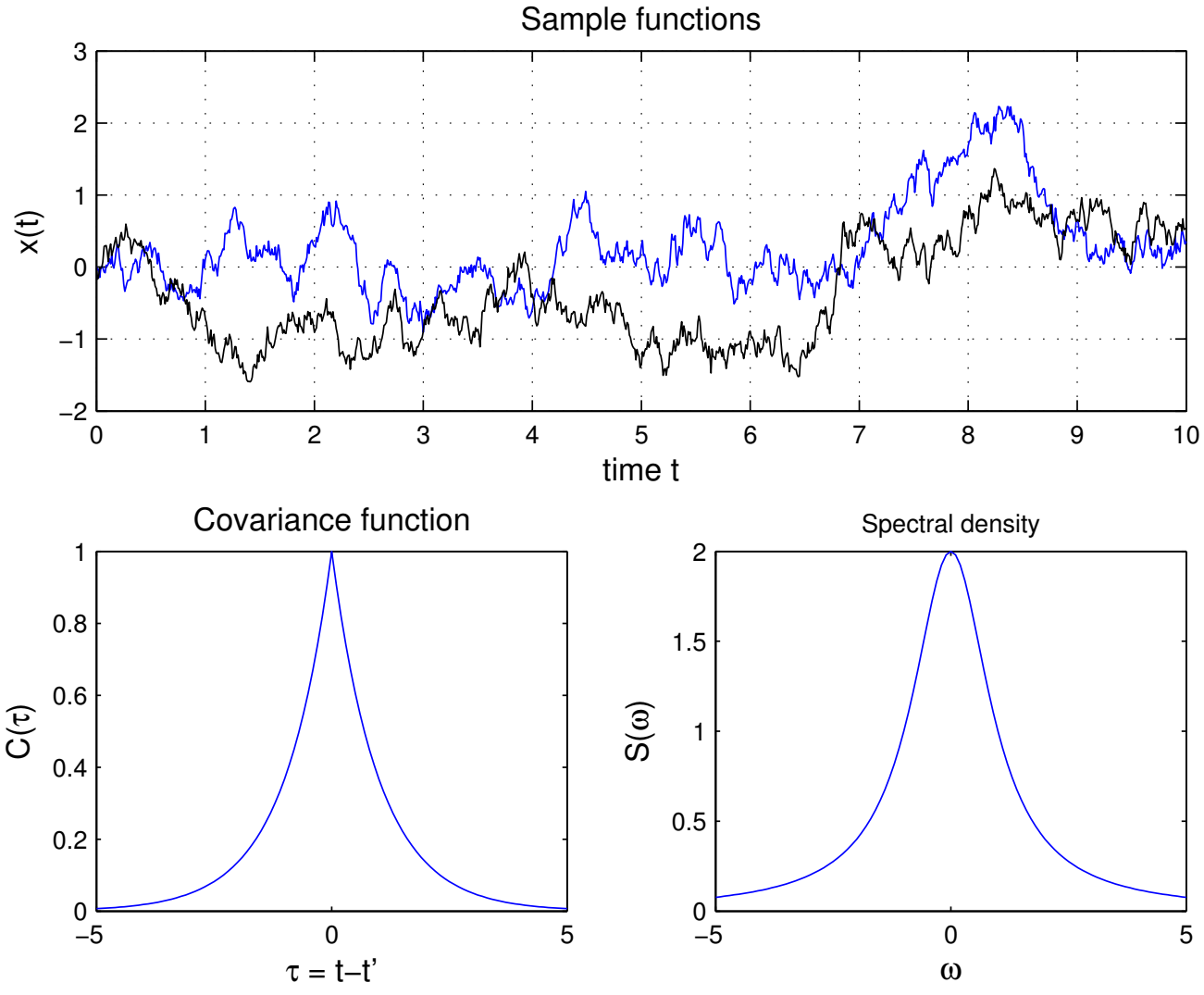
Computational complexity of GP regression

- The GP-regression has cubic computational complexity $O(N^3)$ in the number of measurements N.
- This results from the inversion of the $N \times N$ matrix:

$$\begin{aligned}
\mathrm{E}[g(t^*) \,|\, \mathbf{y}] &= \mathbf{K}(t^*, t_{1:N}) \left(\mathbf{K}(t_{1:N}, t_{1:N}) + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y} \\
\mathrm{Var}[g(t^*) \,|\, \mathbf{y}] &= K(t^*, t^*) \\
&\quad - \mathbf{K}(t^*, t_{1:N}) \left(\mathbf{K}(t_{1:N}, t_{1:N}) + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{K}^{\mathsf{T}}(t^*, t_{1:N}).
\end{aligned}$$

- We could also use GP-based ODE solver step-by-step – loses uncertainty information.
- Various sparse, reduced-rank, and related approximations have been developed for this purpose.
- Here we can use another method – we reduce GP regression into Kalman filtering/smoothing problem which has linear $O(N)$ complexity – for functions of time.

Representations of temporal Gaussian processes



Representations of temporal Gaussian processes

- Example: Ornstein-Uhlenbeck process – path representation as a stochastic differential equation (SDE):

$$\frac{dg(t)}{dt} = -\lambda\, g(t) + w(t),$$

  where $w(t)$ is a white noise process.
- The mean and covariance functions:

$$m(t) = 0$$
$$k(t, t') = \exp(-\lambda|t - t'|)$$

- Spectral density:

$$S(\omega) = \frac{2\lambda}{\omega^2 + \lambda^2}$$

- Ornstein-Uhlenbeck process $g(t)$ is Markovian in the sense that given $g(t)$ the past $\{g(s), s < t\}$ does not affect the distribution of the future $\{g(s'), s' > t\}$.

State-space Gaussian process regression [1/3]

- Consider a Gaussian process regression problem

$$g(t) \sim \mathcal{GP}(0, k(t, t'))$$
$$y_n = g(t_n) + e_n, \qquad e_n \sim \mathcal{N}(0, \sigma^2_{\text{noise}}).$$

- We can can now convert this to state estimation problem:

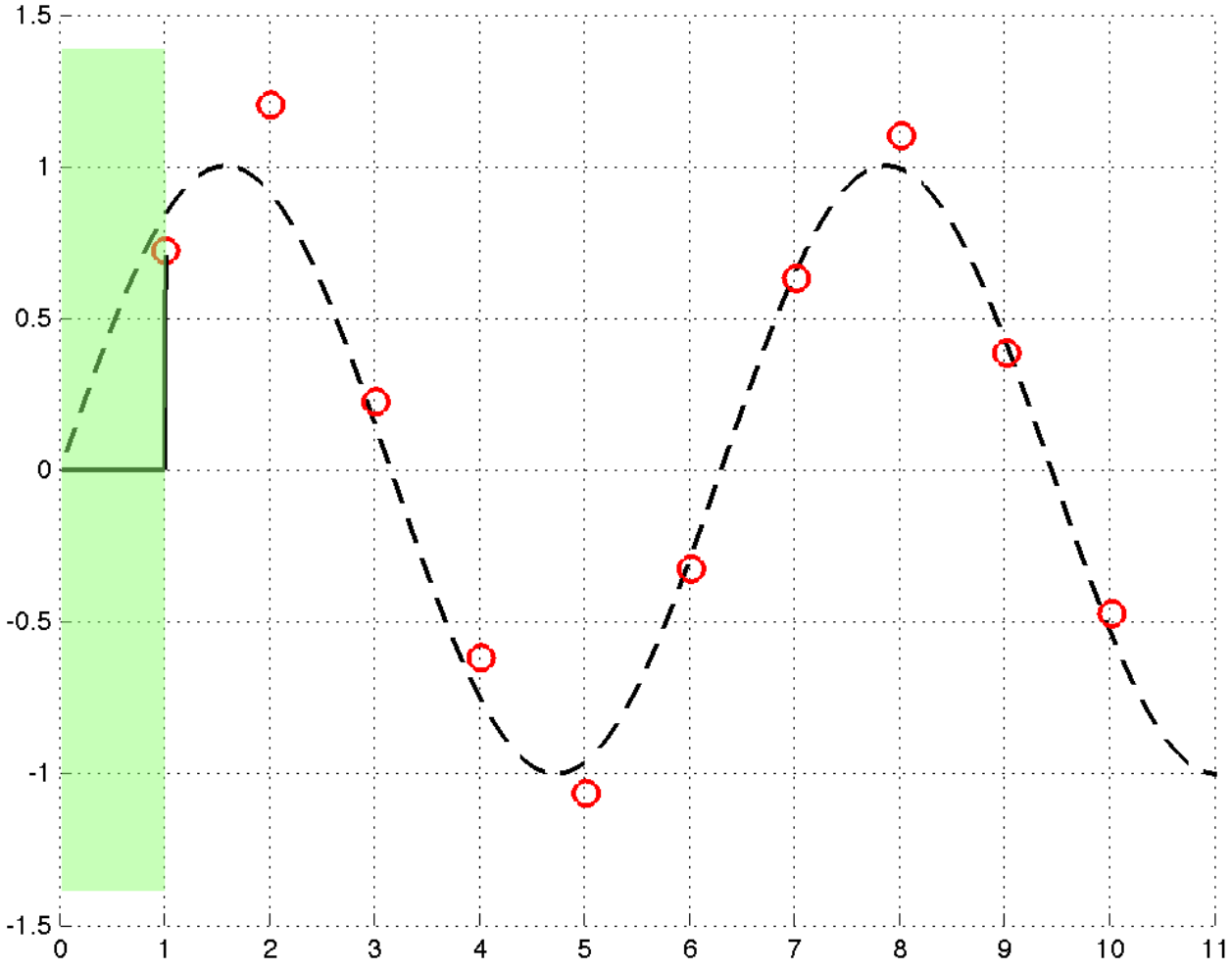
$$\frac{d\mathbf{g}(t)}{dt} = \mathbf{F}\,\mathbf{g}(t) + \mathbf{L}\,w(t)$$
$$y_n = \mathbf{H}\,\mathbf{g}(t_n) + e_n.$$

- This can further be converted into a discrete-time state-space model (here $\mathbf{g}_n = \mathbf{g}(t_n)$)

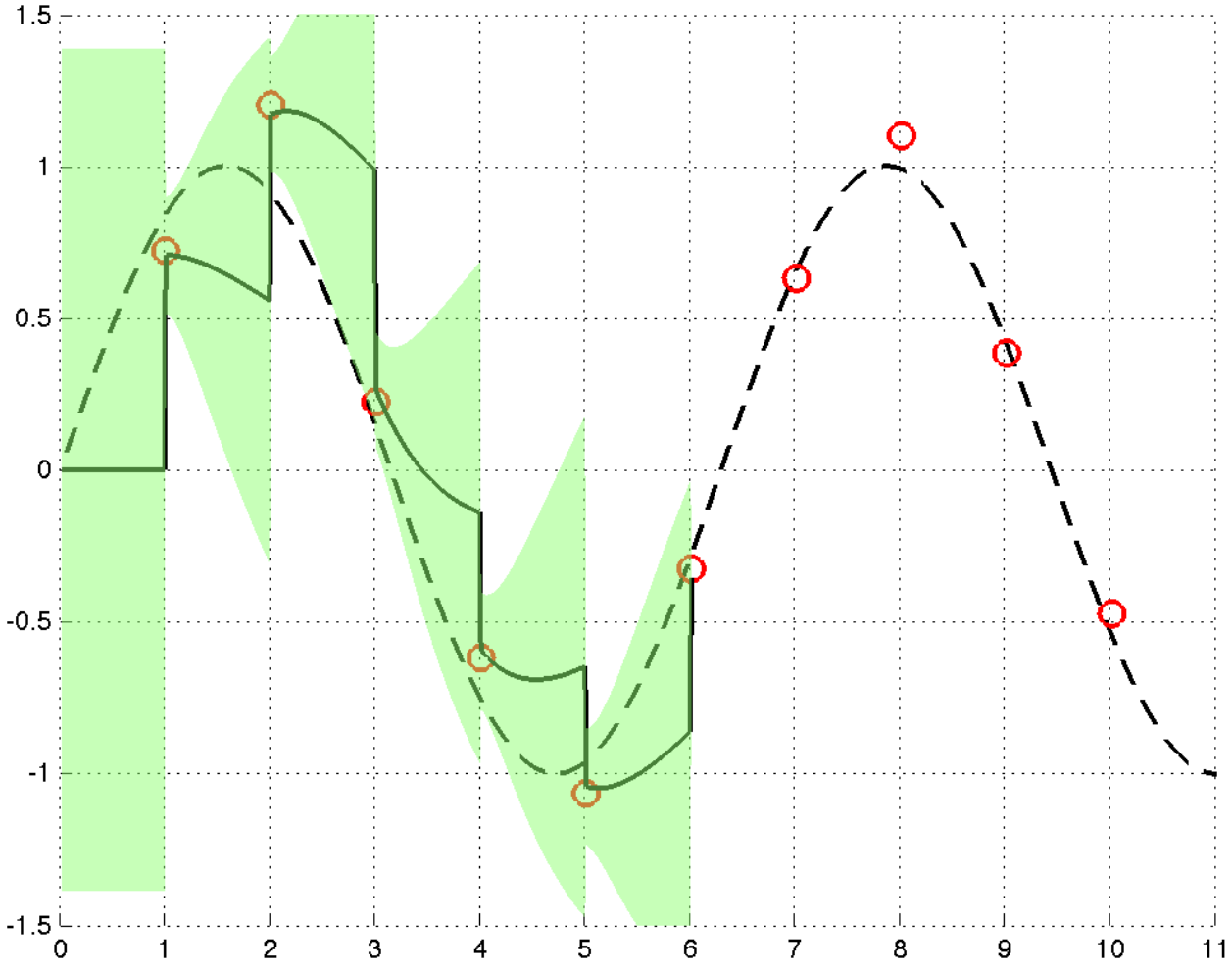
$$\mathbf{g}_n = \mathbf{A}_n\,\mathbf{g}_{n-1} + \mathbf{q}_{n-1},$$
$$y_n = \mathbf{H}\,\mathbf{g}_n + e_n.$$

- The GP-regression solution $p(g(t^*) \mid y_1, \ldots, y_N)$ can now be computed in $O(N)$ time with Kalman filter and smoother.

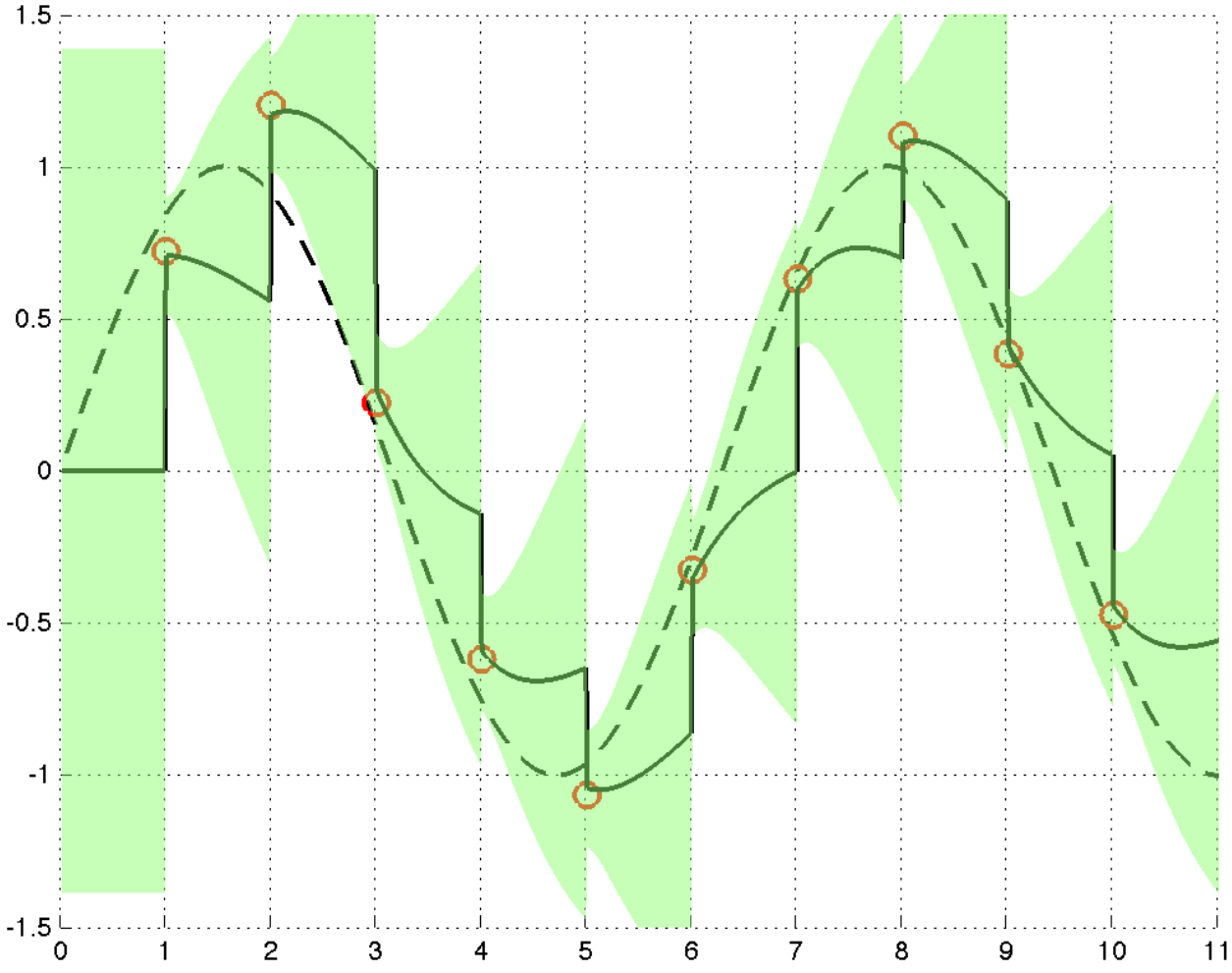
State-space Gaussian process regression [2/3]



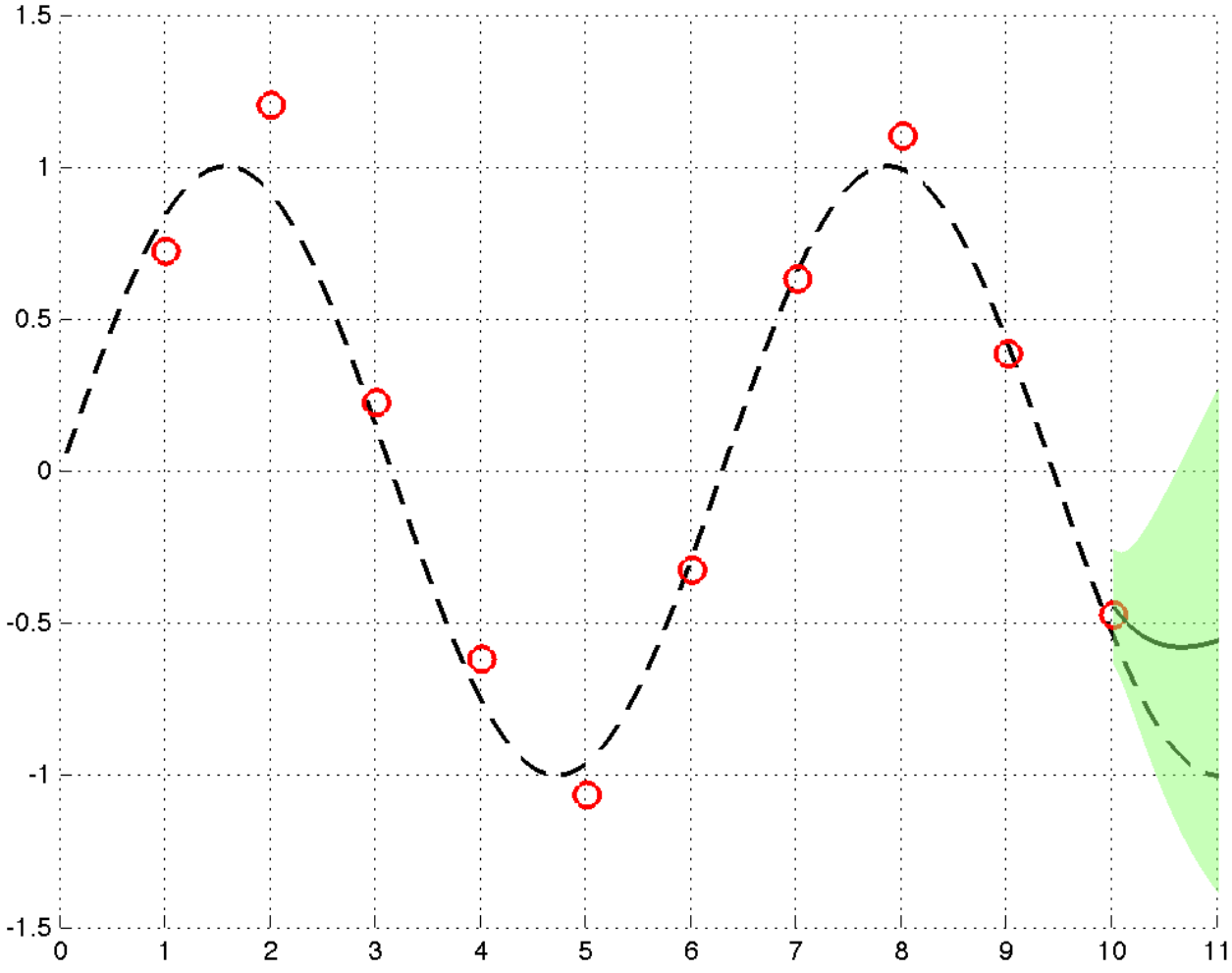
State-space Gaussian process regression [2/3]



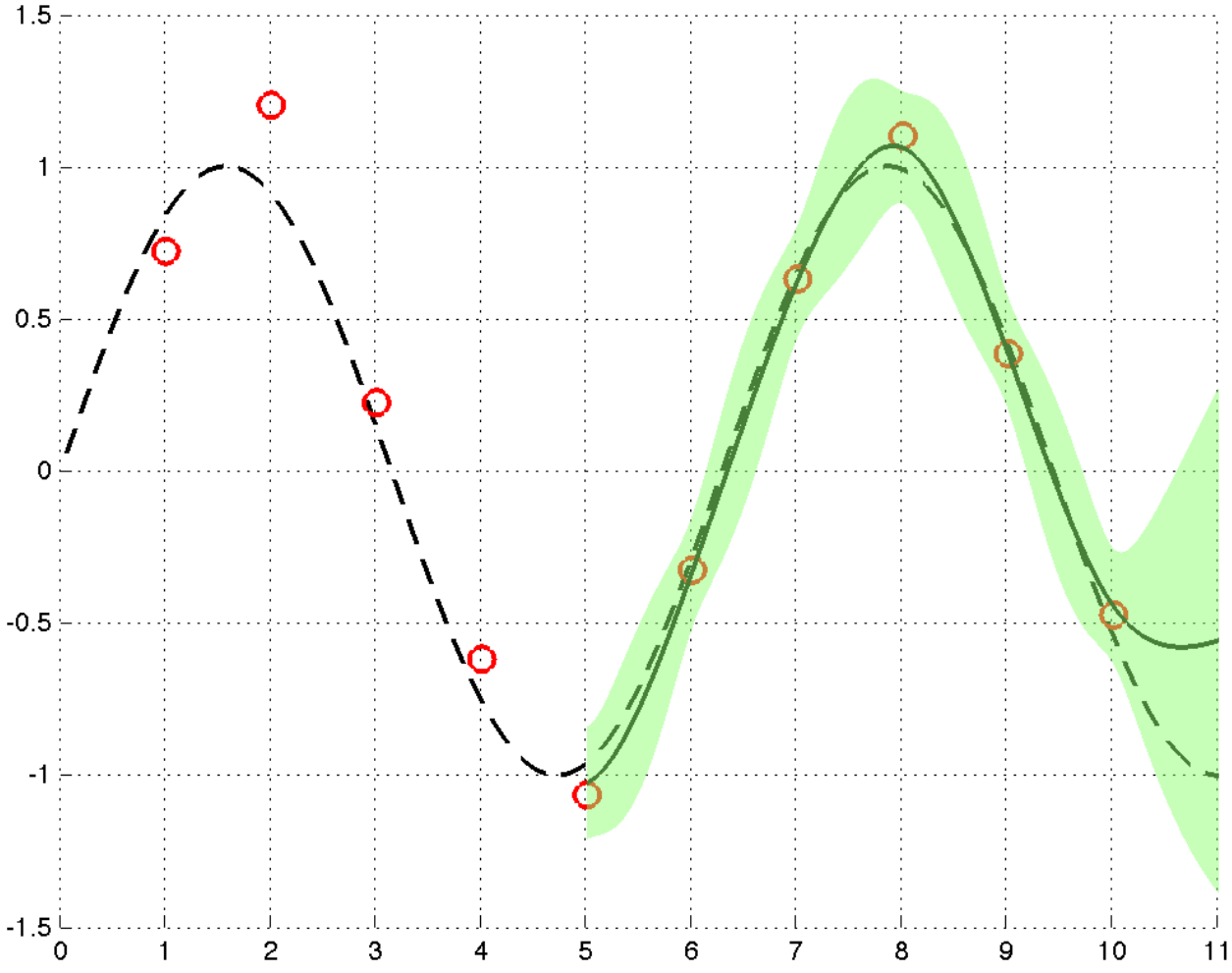
State-space Gaussian process regression [2/3]



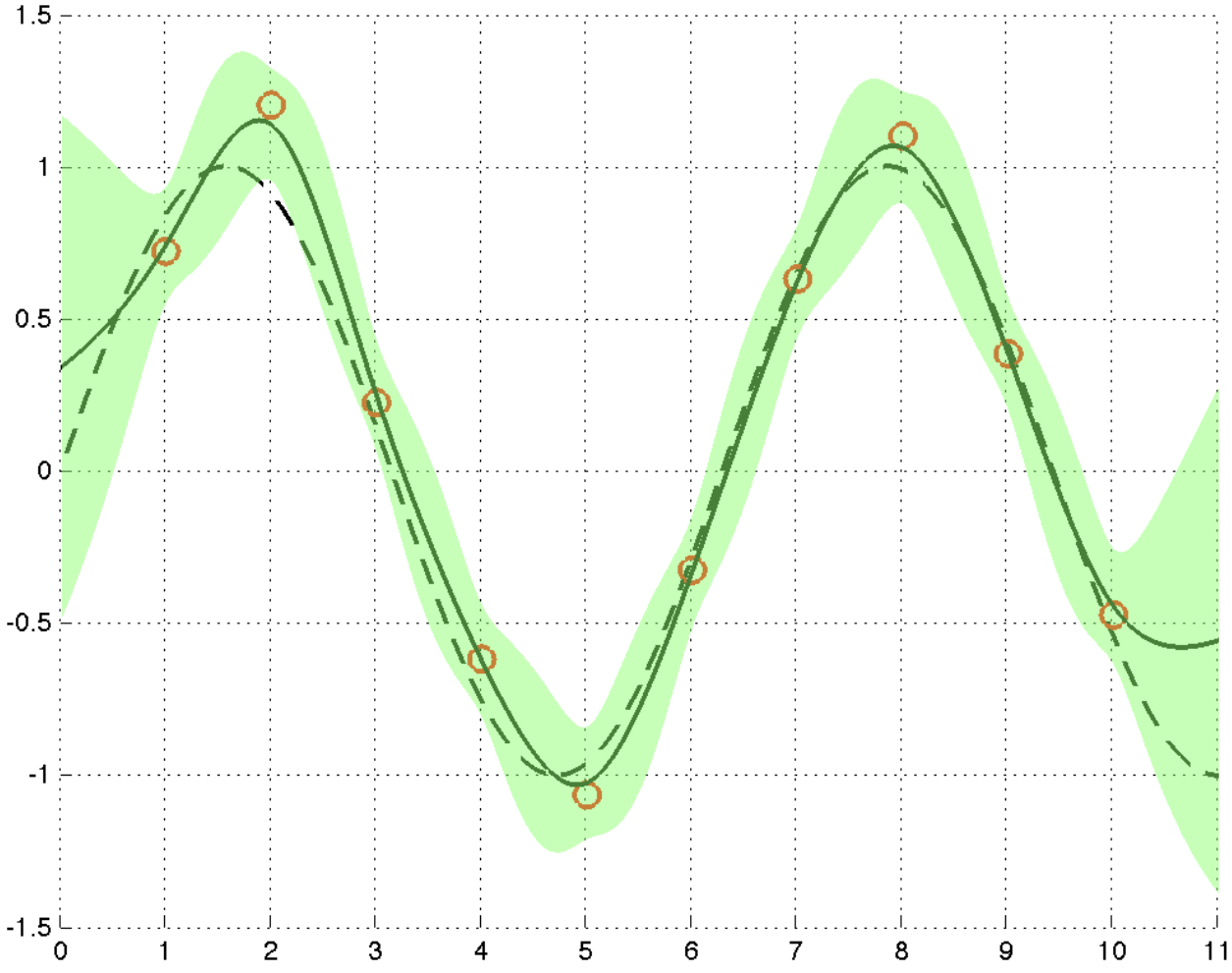
State-space Gaussian process regression [2/3]



State-space Gaussian process regression [2/3]



State-space Gaussian process regression [2/3]



State-space Gaussian process regression [3/3]

- The state $\mathbf{g}(t)$ of the state-space GP regression typically contains the time derivative dg/dt as a component.
- Henceforth, derivative observations can be handled with a simple change of the observation model:

$$\mathbf{g}_n = \mathbf{A}_n \mathbf{g}_{n-1} + \mathbf{q}_{n-1},$$
$$\dot{y}_n = \mathbf{C}\, \mathbf{g}_n + e_n.$$

- For example, if the state is $\mathbf{g} = (g, dg/dt)$, then observing g corresponds to

$$y_n = \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{\mathbf{H}} \mathbf{g}_n + e_n.$$

- Observing dg/dt then corresponds to

$$\dot{y}_n = \underbrace{\begin{pmatrix} 0 & 1 \end{pmatrix}}_{\mathbf{C}} \mathbf{g}_n + e_n.$$

State-Space GP ODE solvers

- Conditioning on the solution to $dx/dt = f(x, t)$ now corresponds to the constraint

$$\mathbf{C}\,\mathbf{g}_n - f(\mathbf{H}\,\mathbf{g}_n, t_n) = 0.$$

- If we write $h_n(\mathbf{g}_n) = \mathbf{C}\,\mathbf{g}_n - f(\mathbf{H}\,\mathbf{g}_n, t_n)$, this corresponds to a pseudo measurement model

$$z_n = h_n(\mathbf{g}_n) + \epsilon_n,$$

  where we observe $z_n = 0$ and ϵ_n has a zero variance.
- Combining with the state-space GP then gives

$$\begin{aligned}\mathbf{g}_n &= \mathbf{A}_n\,\mathbf{g}_{n-1} + \mathbf{q}_{n-1},\\ z_n &= h_n(\mathbf{g}_n) + \epsilon_n.\end{aligned}$$

- But this is just a non-linear filtering/smoothing problem!

Non-linear filters and smoothers as probabilistic ODE solvers

- We can now use any non-linear Bayesian filter as an explicit probabilistic ODE solver.
- For example, extended Kalman filter (EKF), unscented Kalman filter (UKF), particle filter (the last with a catch).
- The iterated extended Kalman smoother (IEKS) can be used to compute the MAP estimate of the trajectory.
- The IEKS corresponds to a form of global implicit probabilistic ODE solver.

Example: Logistic equation (from Tronarp, Särkkä, Hennig, 2021)

Equation: $dy/dt = r\,y(1-y)$.

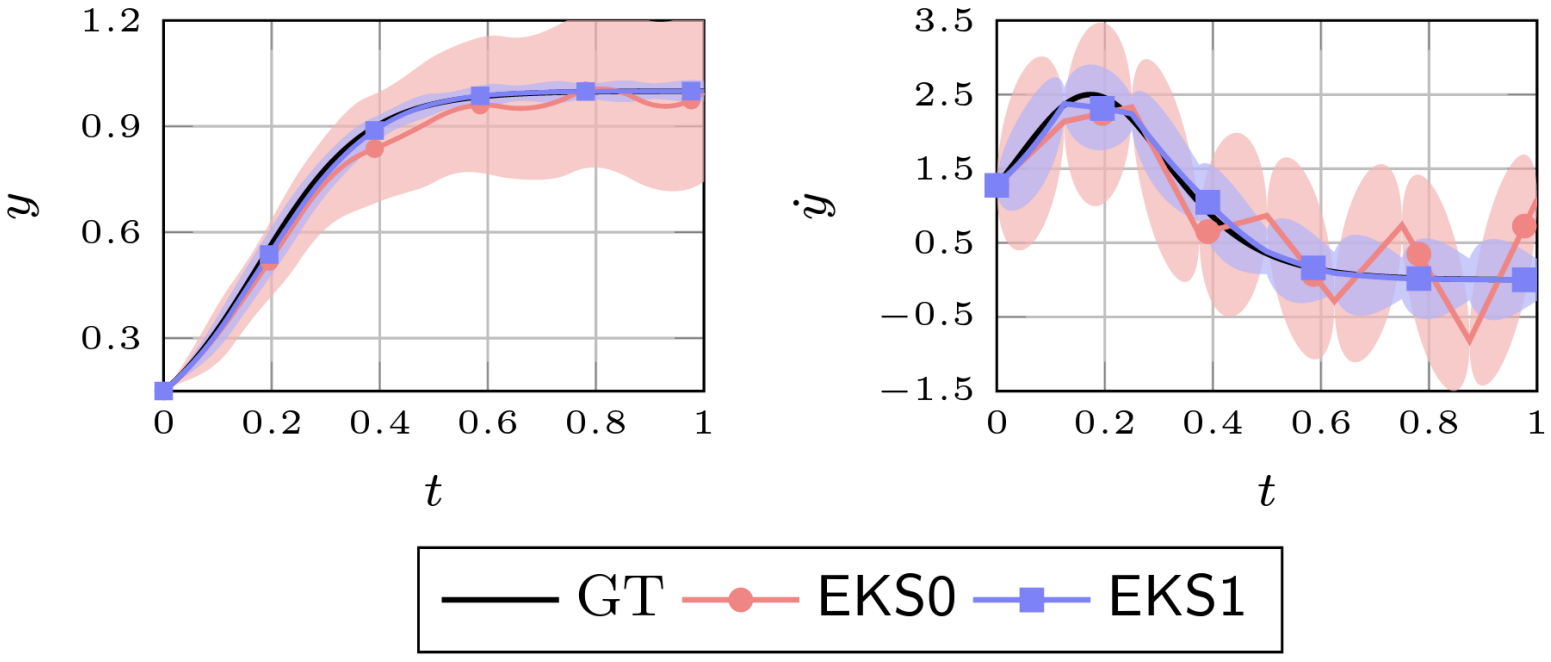


Fig. 3 Reconstruction of the logistic map (left) and its derivative (right) with two standard deviation credible bands for EKS0 (red) and EKS1 (blue).

Extension to Cauchy-type PDEs

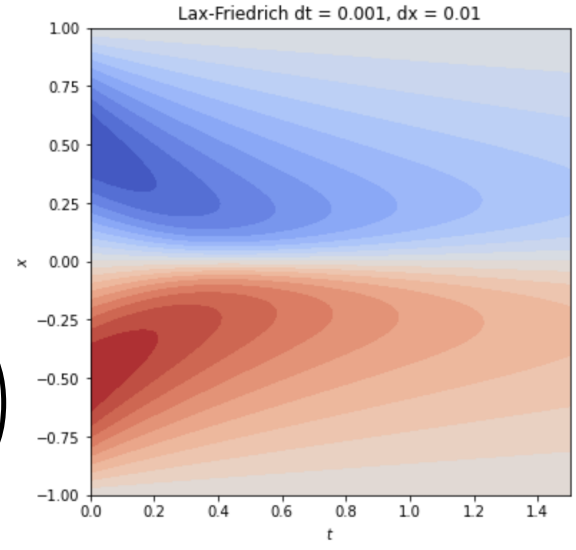
- We can also extend the approach to Cauchy-type of PDEs such as

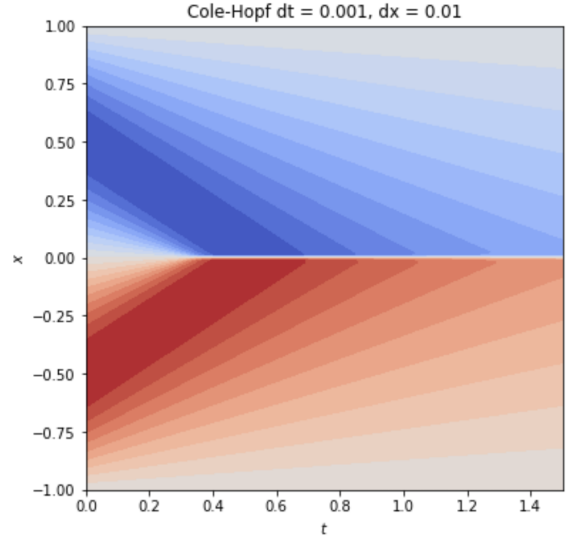
$$\frac{\partial \mathbf{x}(t,\mathbf{r})}{\partial t} = f\left(\mathbf{x}, \frac{\partial \mathbf{x}(t,\mathbf{r})}{\partial \mathbf{r}}, \frac{\partial^2 \mathbf{x}(t,\mathbf{r})}{\partial \mathbf{r}^2}, \dots\right)$$

- This includes, for example, Burger's equation (sorry for notation change):

$$u_t + (F(u))_x = 0.$$

- Hard non-linear PDE with shocks.





Recipe for probabilistic solving of PDEs

- A simple approach is to space-discretize the PDE which results in a high-dimensional ODE (method of lines).
- The GP prior should be spatio-temporal, which can be represented as infinite-dimensional SDE.
- The infinite-dimensional SDE can be space-discretized in analogous way.
- Possible methods:
  - Finite-difference methods.
  - Ritz–Galerkin methods.
  - Finite element method (FEM).
- The resulting finite-dimensional state-estimation problem can be tackled with EKF, UKF, IEKS, PF, etc.

Example: Approximating Burger's as an ODE via discretization

- The Burger's equation can be space-discretized as

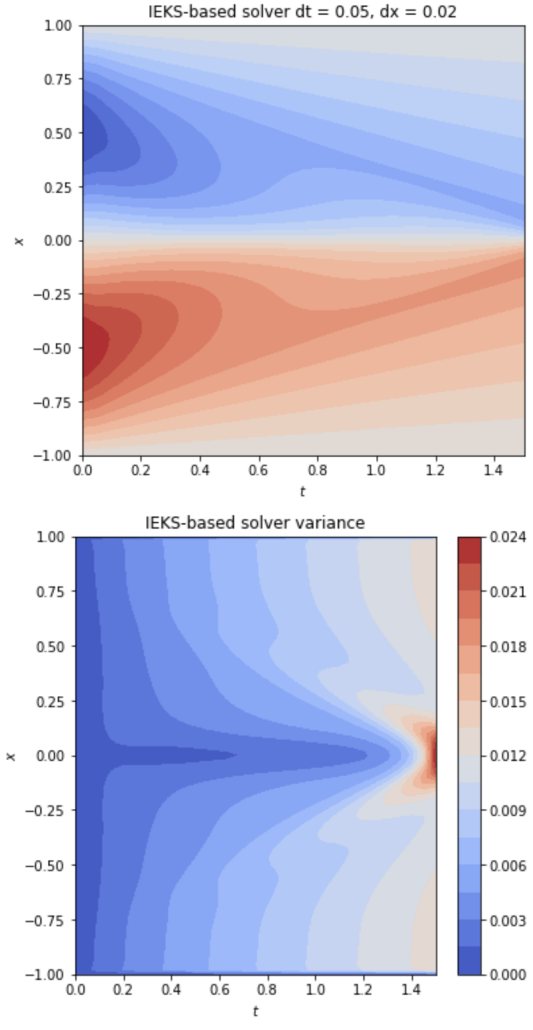
$$\frac{du_j}{dt} + \frac{1}{2\Delta x}(F(u_{j+1}(t)) - F(u_{j-1}(t))) = 0,$$

where $u_j(t) = u(t, x_j)$.

- We can then formulate the GP prior as infinite-dimensional SDE

$$\frac{\partial \mathbf{g}(x,t)}{\partial t} = \mathcal{A}\,\mathbf{g}(x,t) + \mathbf{L}\,\mathbf{w}(x,t),$$

where $\mathcal{A}$ is a (pseudo) differential operator.



Conclusion

- Probabilistic ODE solvers aim to provide probabilistic uncertainty bounds for ODE solutions.
- Based on replacing the classical polynomial approximation with a Gaussian process (GP) regressor.
- GP-based ODE solvers can be reformulated as Bayesian filtering and smoothing problems.
- Explicit obtained solvers via EKF, UKF, and PF, implicit global solution with IEKS.
- The concept can be extended to partial differential equations (PDEs) of Cauchy form.

References

References (incomplete list)

Kersting H, Hennig P (2016). **Active uncertainty calibration in Bayesian ODE solvers.** *Uncertainty in Artificial Intelligence (UAI).*

Nicholas Krämer, Jonathan Schmidt, Philipp Hennig (2022). **Probabilistic Numerical Method of Lines for Time-Dependent Partial Differential Equations.** *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics.*

Simo Särkkä (2013). **Bayesian Filtering and Smoothing.** *Cambridge University Press.*

S. Särkkä, A. Solin, and J. Hartikainen (2013). **Spatio-Temporal Learning via Infinite-Dimensional Bayesian Filtering and Smoothing.** *IEEE Signal Processing Magazine, Volume 30, Issue 4, Pages 51-61.*

Schober M, Duvenaud D, Hennig P (2014). **Probabilistic ODE solvers with Runge–Kutta means.** *In: Advances in Neural Information Processing Systems (NIPS).*

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

J. Skilling. **Bayesian solution of ordinary differential equations.** *Maximum Entropy and Bayesian Methods, Seattle, 1991.*

Filip Tronarp, Hans Kersting, Simo Särkkä, Philipp Hennig (2019). **Probabilistic Solutions To Ordinary Differential Equations As Non-Linear Bayesian Filtering: A New Perspective. Statistics and Computing,** *Volume 29, pages 1297-1315.*

Filip Tronarp, Simo Särkkä, and Philipp Hennig (2021). **Bayesian ODE Solvers: The Maximum A Posteriori Estimate.** *Statistics and Computing 31, 23.*