Aalto University
School of Electrical Engineering

Probabilistic differential equation solving as Bayesian filtering and smoothing

Simo Särkkä

Aalto University, Finland

March 28, 2023

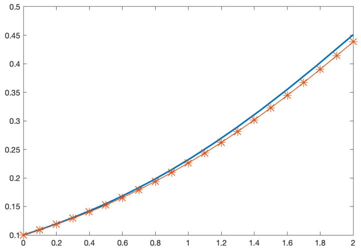
Contents

Problem formulation

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

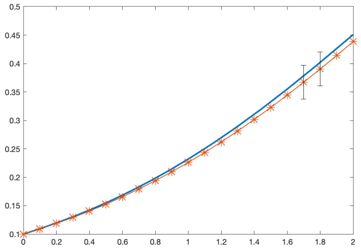
$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(t), t), \qquad \mathbf{x}(0) = \mathbf{x}_0.$$

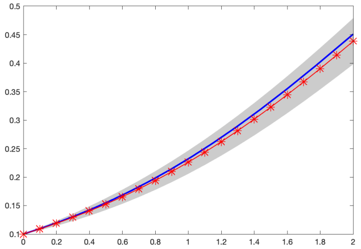
- The aim is to find an approximate solution $\hat{\mathbf{x}}(t)$ such that $\hat{\mathbf{x}}(t_n) \approx \mathbf{x}(t_n)$ on some points $0 = t_0 < t_1 < \cdots < t_N = T$.
- Function $\mathbf{f}(\cdot)$ is only evaluated at points $\hat{\mathbf{x}}(t_n)$, and some nearby points.
- The approximate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

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

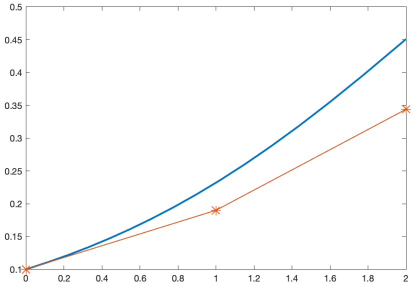
- Runge–Kutta methods are based on higher order polynomial fitting:

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

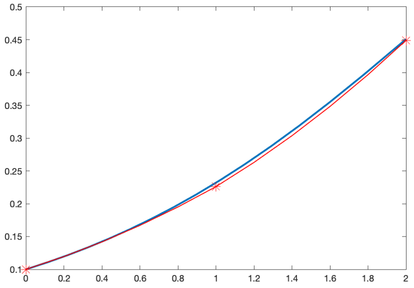
- The worst-case error analysis possible using Taylor's theorem.

Classical ODE solving is polynomial fitting (cont.)

Linear approximation:

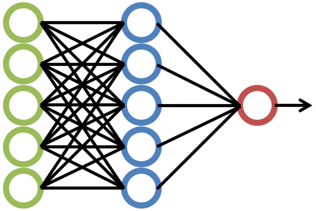


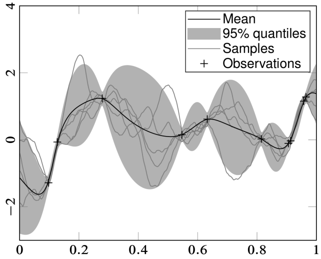
Polynomial approximation:



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 $g(t; \boldsymbol{\theta})$ and estimate the parameters $\boldsymbol{\theta}$.
- 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

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

- 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

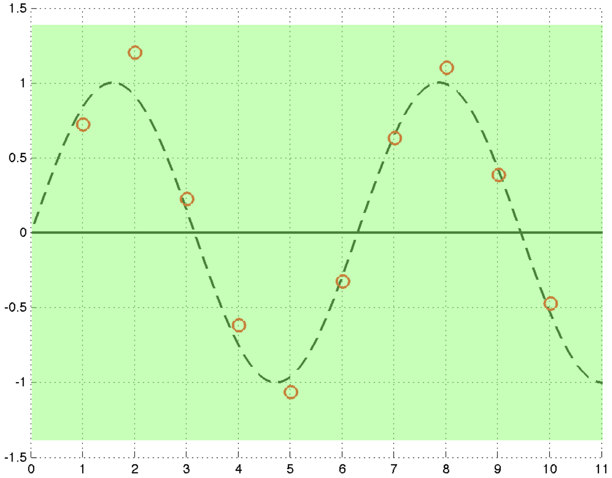
$$\begin{pmatrix} \mathbf{y} \\ g(t^*) \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{K}(t_{1:N}, t_{1:N}) + \sigma^2 \mathbf{I} & \mathbf{K}^\mathsf{T}(t^*, t_{1:N}) \\ \mathbf{K}(t^*, t_{1:N}) & K(t^*, t^*) \end{pmatrix} \right)$$

- The conditional mean and variance are given as

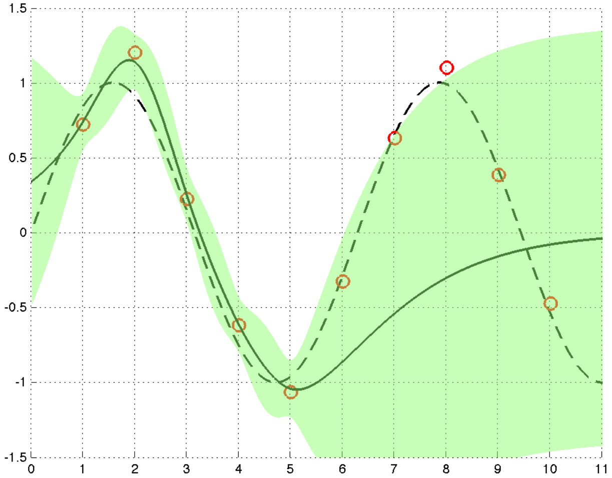
$$\begin{aligned} \mathrm{E}[g(t^*) \mid \mathbf{y}] &= \mathbf{K}(t^*, t_{1:N}) \, (\mathbf{K}(t_{1:N}, t_{1:N}) + \sigma^2 \mathbf{I})^{-1} \, \mathbf{y} \\ \mathrm{Var}[g(t^*) \mid \mathbf{y}] &= K(t^*, t^*) \\ &\qquad - \mathbf{K}(t^*, t_{1:N}) \, (\mathbf{K}(t_{1:N}, t_{1:N}) + \sigma^2 \mathbf{I})^{-1}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

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

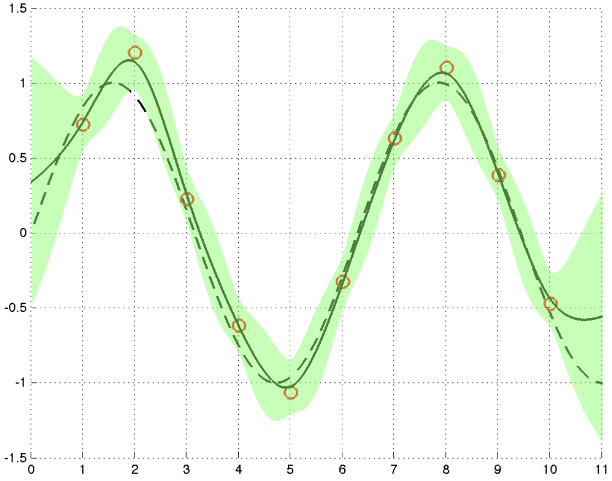
Gaussian process regression [4/5]



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

$$\begin{aligned}
\mathrm{E}[g(t^*) \mid \dot{\mathbf{y}}] &= \frac{\partial \mathbf{K}}{\partial t}(t^*, t_{1:N}) \left(\frac{\partial^2 \mathbf{K}(t_{1:N}, t_{1:N})}{\partial t \, \partial t'} + \sigma^2 \mathbf{I} \right)^{-1} \dot{\mathbf{y}} \\
\mathrm{Var}[g(t^*) \mid \dot{\mathbf{y}}] &= \frac{\partial \mathbf{K}}{\partial t}(t^*, t_{1:N}) \\
&\quad - \frac{\partial \mathbf{K}}{\partial t}(t^*, t_{1:N}) \left(\frac{\partial^2 \mathbf{K}(t_{1:N}, t_{1:N})}{\partial t \, \partial t'} + \sigma^2 \mathbf{I} \right)^{-1} \frac{\partial \mathbf{K}^{\mathsf{T}}}{\partial t}(t^*, t_{1:N}).
\end{aligned}$$

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.

Probabilistics State Space Models

- Generally, Markov model for the state $\mathbf{x}_k \in \mathbb{R}^n$:

$$\mathbf{x}_k \sim p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}).$$

- Likelihood distribution of the measurement $\mathbf{y}_k \in \mathbb{R}^m$:

$$\mathbf{y}_k \sim p(\mathbf{y}_k \,|\, \mathbf{x}_k).$$

- Has the form of hidden Markov model (HMM):

$$\begin{tikzcd}
\text{observed:} & \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \mathbf{y}_4 & \\
\text{hidden:} & \mathbf{x}_1 \arrow[u] \arrow[r] & \mathbf{x}_2 \arrow[u] \arrow[r] & \mathbf{x}_3 \arrow[u] \arrow[r] & \mathbf{x}_4 \arrow[u] \arrow[r] & \ldots
\end{tikzcd}$$

Probabilistics State Space Models: Example

Example (Gaussian random walk)

Gaussian random walk model can be written as

$$\begin{aligned} x_k &= x_{k-1} + w_{k-1}, \quad & w_{k-1} &\sim \mathcal{N}(0, q) \\ y_k &= x_k + e_k, & e_k &\sim \mathcal{N}(0, r), \end{aligned}$$

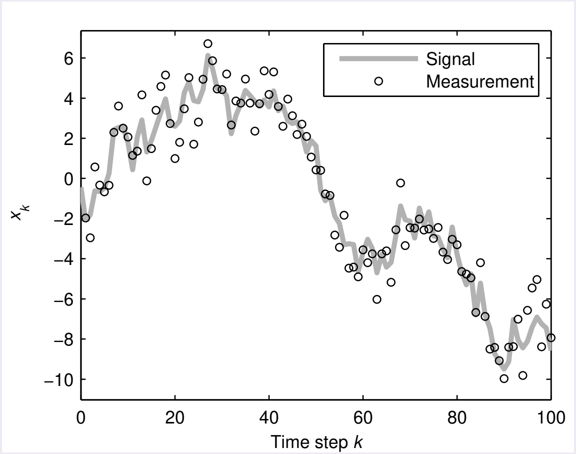
where x_k is the hidden state and y_k is the measurement. In terms of probability densities the model can be written as

$$\begin{aligned} p(x_k \mid x_{k-1}) &= \frac{1}{\sqrt{2\pi q}} \exp\left(-\frac{1}{2q} (x_k - x_{k-1})^2 \right) \\ p(y_k \mid x_k) &= \frac{1}{\sqrt{2\pi r}} \exp\left(-\frac{1}{2r} (y_k - x_k)^2 \right) \end{aligned}$$

which is a discrete-time state space model.

Probabilistics State Space Models: Example (cont.)

Example (Gaussian random walk (cont.))



Probabilistics State Space Models: Further Examples

- Linear Gauss-Markov model – which defines a Gaussian process:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{A}_{k-1}\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1}\\ \mathbf{y}_k &= \mathbf{H}_k\,\mathbf{x}_k + \mathbf{r}_k,\end{aligned}$$

- Gaussian driven non-linear model – a non-Gaussian process:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})\\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{r}_k).\end{aligned}$$

- Continuous-discrete-time models

$$\begin{aligned}\frac{d\mathbf{x}}{dt} &= \mathbf{a}(\mathbf{x}) + \mathbf{L}(\mathbf{x})\,\mathbf{w}(t)\\ \mathbf{y}_k &\sim p(\mathbf{y}_k \,|\, \mathbf{x}(t_k)).\end{aligned}$$

Bayesian Filtering, Prediction and Smoothing

- In principle, we could just use the (batch) Bayes' rule

$$
\begin{aligned}
& p(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \mathbf{y}_1, \ldots, \mathbf{y}_T) \\
& \qquad = \frac{p(\mathbf{y}_1, \ldots, \mathbf{y}_T \mid \mathbf{x}_1, \ldots, \mathbf{x}_T)\, p(\mathbf{x}_1, \ldots, \mathbf{x}_T)}{p(\mathbf{y}_1, \ldots, \mathbf{y}_T)},
\end{aligned}
$$

- Curse of computational complexity: complexity grows more than linearly with the measurements (typically $O(T^3)$).
- However, we can compute the following in $O(T)$ time:
  - Filtering distributions:

$$p(\mathbf{x}_k \mid \mathbf{y}_1, \ldots, \mathbf{y}_k), \qquad k = 1, \ldots, T.$$

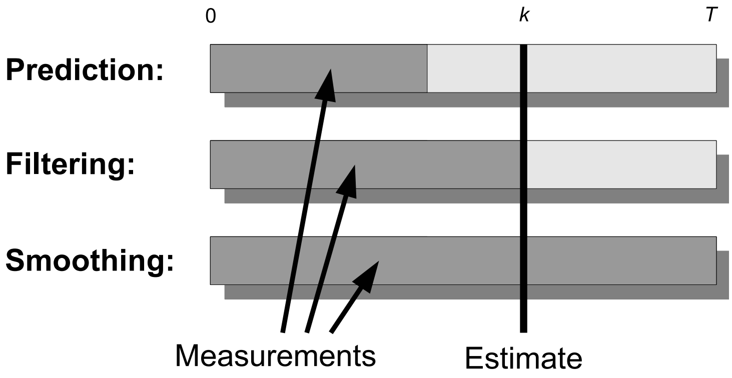
  - Prediction distributions:

$$p(\mathbf{x}_{k+n} \mid \mathbf{y}_1, \ldots, \mathbf{y}_k), \qquad k = 1, \ldots, T, \quad n = 1, 2, \ldots,$$

  - Smoothing distributions:

$$p(\mathbf{x}_k \mid \mathbf{y}_1, \ldots, \mathbf{y}_T), \qquad k = 1, \ldots, T.$$

Bayesian Filtering, Prediction and Smoothing (cont.)



Bayesian Filter: Principle

- Bayesian optimal filter computes the distribution

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$$

- Given the following:
  1. Prior distribution $p(\mathbf{x}_0)$.
  2. State space model:

$$\begin{aligned}\mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k),\end{aligned}$$

  3. Measurement sequence $\mathbf{y}_{1:k} = \mathbf{y}_1, \ldots, \mathbf{y}_k$.
- Computation is based on recursion rule for incorporation of the new measurement $\mathbf{y}_k$ into the posterior:

$$p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) \longrightarrow p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$$

Bayesian Optimal Filter: Formal Equations

Optimal filter

- Initialization: The recursion starts from the prior distribution $p(\mathbf{x}_0)$.
- Prediction: by the Chapman-Kolmogorov equation

$$p(\mathbf{x}_k \,|\, \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}) \, p(\mathbf{x}_{k-1} \,|\, \mathbf{y}_{1:k-1}) \, \mathrm{d}\mathbf{x}_{k-1}.$$

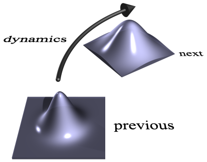
- Update: by the Bayes' rule

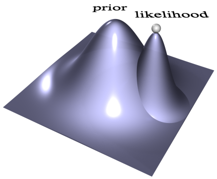
$$p(\mathbf{x}_k \,|\, \mathbf{y}_{1:k}) = \frac{1}{Z_k} p(\mathbf{y}_k \,|\, \mathbf{x}_k) \, p(\mathbf{x}_k \,|\, \mathbf{y}_{1:k-1}).$$

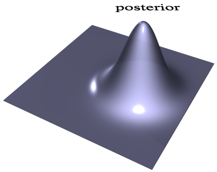
- The normalization constant $Z_k = p(\mathbf{y}_k \,|\, \mathbf{y}_{1:k-1})$ is given as

$$Z_k = \int p(\mathbf{y}_k \,|\, \mathbf{x}_k) \, p(\mathbf{x}_k \,|\, \mathbf{y}_{1:k-1}) \, \mathrm{d}\mathbf{x}_k.$$

Bayesian Optimal Filter: Graphical Explanation







On prediction step the distribution of previous step is propagated through the dynamics.

Prior distribution from prediction and the likelihood of measurement.

The posterior distribution after combining the prior and likelihood by Bayes' rule.

Filtering Algorithms

- Kalman filter is the classical optimal filter for linear-Gaussian models.
- Extended Kalman filter (EKF) is linearization based extension of Kalman filter to non-linear models.
- Unscented Kalman filter (UKF) is sigma-point transformation based extension of Kalman filter.
- Gauss-Hermite and Cubature Kalman filters (GHKF/CKF) are numerical integration based extensions of Kalman filter.
- Particle filter forms a Monte Carlo representation (particle set) to the distribution of the state estimate.
- Grid based filters approximate the probability distributions on a finite grid.
- Mixture Gaussian approximations are used, for example, in multiple model Kalman filters and Rao-Blackwellized Particle filters.

Bayesian Smoothing Problem

- We have a probabilistic state space model:

$$\begin{aligned} \mathbf{y}_k &\sim p(\mathbf{y}_k \,|\, \mathbf{x}_k) \\ \mathbf{x}_k &\sim p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}) \end{aligned}$$

- Assume that the filtering distributions $p(\mathbf{x}_k \,|\, \mathbf{y}_{1:k})$ have already been computed for all $k = 0, \ldots, T$.
- We want recursive equations of computing the smoothing distribution for all $k < T$:

$$p(\mathbf{x}_k \,|\, \mathbf{y}_{1:T}).$$

- The recursion will go backwards in time, because on the last step, the filtering and smoothing distributions coincide:

$$p(\mathbf{x}_T \,|\, \mathbf{y}_{1:T}).$$

Bayesian Smoothing Equations

Bayesian Smoothing Equations

The Bayesian smoothing equations consist of prediction step and backward update step:

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) = \int p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k})\, \mathrm{d}\mathbf{x}_k$$

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) = p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \int \left[\frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \right] \mathrm{d}\mathbf{x}_{k+1}$$

The recursion is started from the filtering (and smoothing) distribution of the last time step $p(\mathbf{x}_T \mid \mathbf{y}_{1:T})$.

Smoothing Algorithms

- Rauch-Tung-Striebel (RTS) smoother is the closed form smoother for linear Gaussian models.
- Extended, unscented, and sigma-point RTS smoothers are the approximate nonlinear smoothers corresponding to EKF, UKF, and sigma-point filters.
- Gaussian RTS smoothers: cubature RTS smoother, Gauss–Hermite RTS smoothers and various others.
- Iterated smoothers: iterated extended Kalman smoother (IEKS), iterated posterior linearization smoother (IPLS).
- Particle smoothing is based on approximating the smoothing solutions via Monte Carlo.
- Rao-Blackwellized particle smoother is a combination of particle smoothing and RTS smoothing.

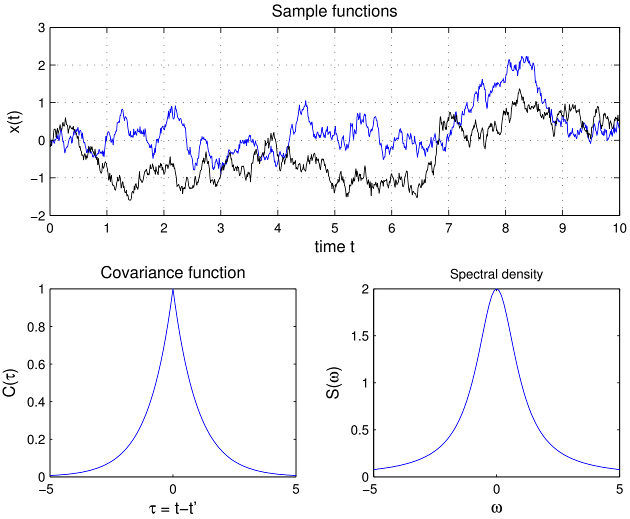
Computational complexity of GP regression

- 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:

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

- In practice, we use Cholesky factorization and do not invert explicitly – but still the $O(N^3)$ problem remains.
- 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 ($T = N$).

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 [1/4]

- Consider a Gaussian process regression problem

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

- By defining $\mathbf{x}(t) = (g(t), dg(t)/dt, d^2 g(t)/dt^2, \ldots)$, we can convert this to state estimation problem:

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

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

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

State-space Gaussian process regression [2/4]

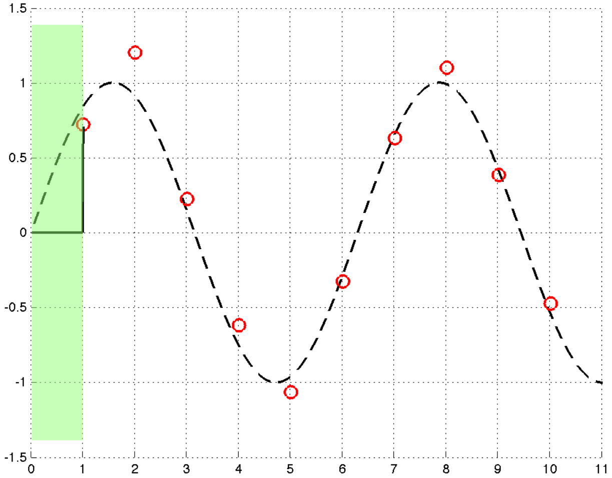
- The GP-regression solution

$$p(g(t^*) \mid y_1, \ldots, y_N) = p(x_1(t^*) \mid y_1, \ldots, y_N)$$

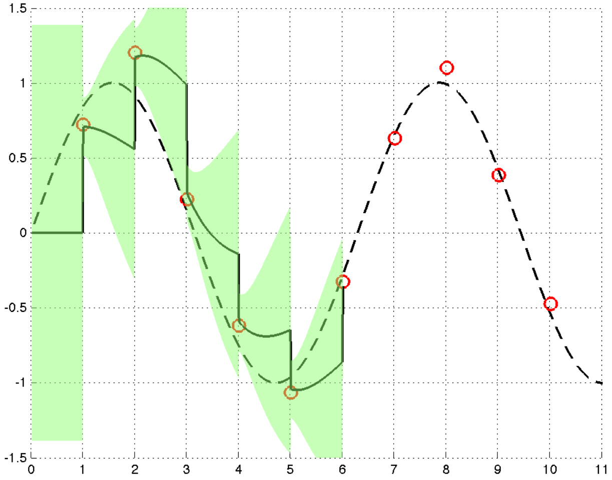
  can now be computed in $O(N)$ time with Kalman filter and smoother.

- In practice we do the following:
  1. Form the matrices **F**, **L**, **H** from the GP covariance.
  2. Augment the test points among the observed data.
  3. Discretize the system by the matrix exponentials and Lyapunov solvers.
  4. Run Kalman filter forwards on the data.
  5. Run Kalman smoother backward on the data.
  6. Collect the GP regression solution by collecting the related state mean and covariance blocks.

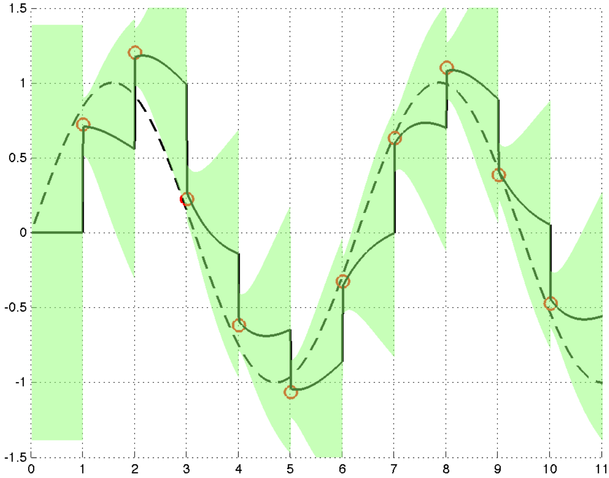
State-space Gaussian process regression [3/4]



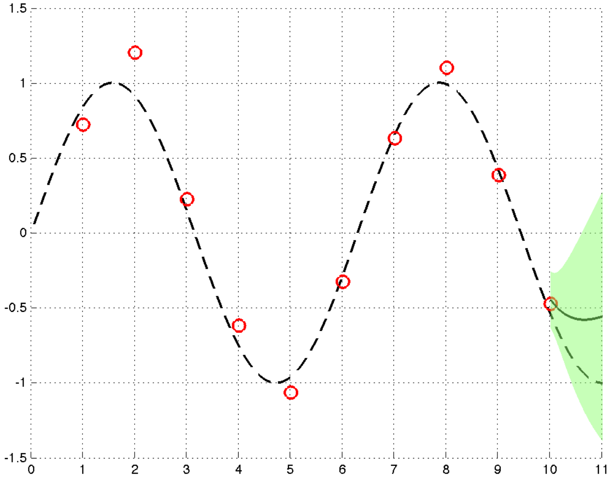
State-space Gaussian process regression [3/4]



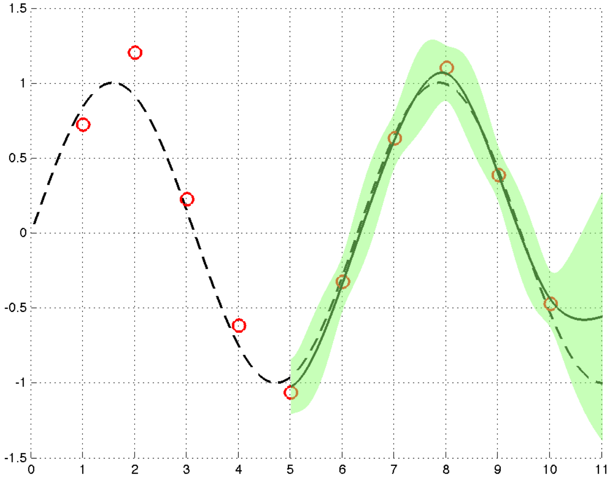
State-space Gaussian process regression [3/4]



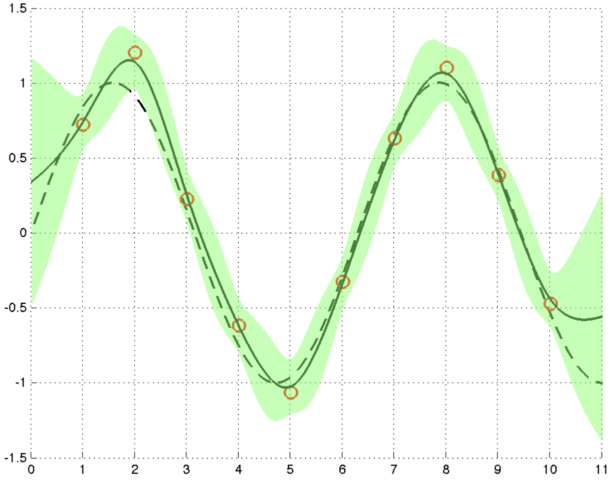
State-space Gaussian process regression [3/4]



State-space Gaussian process regression [3/4]



State-space Gaussian process regression [3/4]



State-space Gaussian process regression [4/4]

- The state $\mathbf{x}(t)$ 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:

$$\begin{aligned} \mathbf{x}_n &= \mathbf{A}_n \, \mathbf{x}_{n-1} + \mathbf{q}_{n-1}, \\ \dot{y}_n &= \mathbf{C} \, \mathbf{x}_n + e_n. \end{aligned}$$

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

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

- Observing dg/dt then corresponds to

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

State-Space GP ODE solvers

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

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

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

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

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

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

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

Non-linear filters and smoothers as probabilistic ODE solvers

- We have a large collection of Bayesian filters and smoothers which are applicable to the state space model

$$\begin{aligned} \mathbf{x}_n &= \mathbf{A}_n \, \mathbf{x}_{n-1} + \mathbf{q}_{n-1}, \\ z_n &= \mathbf{C} \, \mathbf{x}_n - f(\mathbf{H} \, \mathbf{x}_n, t_n) + \epsilon_n. \end{aligned}$$

- We can 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: $dx/dt = r\,x(1-x)$.

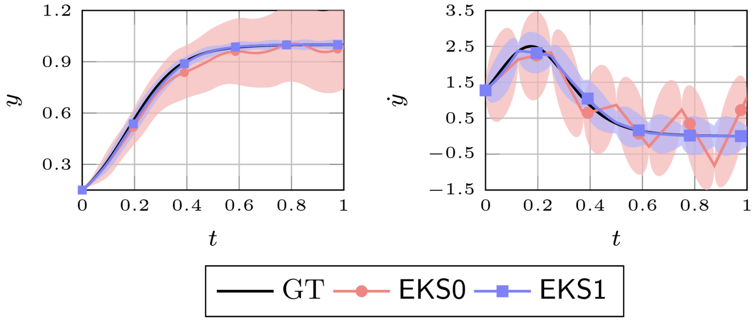


Fig. 3 Reconstruction of the logistic map (left) and its derivative (right) with two standard deviation credible bands for EKS0 (red) and EKS1 (blue).

Example: Pendulum

- The pendulum obeys an ODE

$$\frac{d^2\alpha}{dt^2} = -g \sin(\alpha).$$

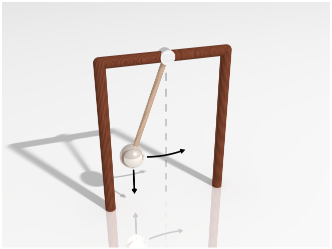
- By defining $x_1 = \alpha$ and $x_2 = d\alpha / dt$, we get

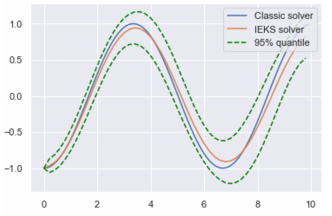
$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -g \sin(x_1) \end{pmatrix},$$

which is an ODE of the form

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}).$$

- Example code in JAX $\Rightarrow$





Further directions

- Parameter estimation (of GP hyperparameters) can be done using state-space methods.
- Parallel Kalman filtering and smoothing methods and their non-linear extensions.
- Numerically stable square-root filters and smoothers.
- Partial differential equations via the method of lines.
- Latent force models are also closely related to state estimation, filters, and smoothers.

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:
  1. Convert the GP into its state-space form.
  2. Form state-space model with non-linear measurement model encoding $dx/dt - f(x,t) = 0$.
  3. Solve the state inference problem using a Bayesian filter or smoother.
- Explicit obtained solvers via EKF, UKF, and PF, implicit global solution with IEKS.

References (incomplete list)

Philipp Hennig, Michael A. Osborne, Hans P. Kersting (2022). **Probabilistic Numerics: Computation as Machine Learning.** *Cambridge University Press (CUP).*

Nicholas Krämer, Jonathan Schmidt, Philipp Hennig (2022). **Probabilistic Numerical Method of Lines for Time-Dependent Partial Differential Equations.** *Proc. AISTATS.*

Simo Särkkä, Mauricio A. Álvarez, Neil D. Lawrence (2019). **Gaussian Process Latent Force Models for Learning and Stochastic Control of Physical Systems.** *IEEE Trans. Autom. Control, 64(7):2953-2960.*

Simo Särkkä and Ángel F. Garcia-Fernández (2021). **Temporal Parallelization of Bayesian Smoothers.** *IEEE Transactions on Automatic Control, 66(1):299-306.*

Simo Särkkä and Arno Solin (2019). **Applied Stochastic Differential Equations.** *CUP.*

Simo Särkkä and Lennart Svensson (2023). **Bayesian Filtering and Smoothing,** 2nd ed. *CUP.* (in press)

Simo Särkkä, Arno Solin, and Jouni Hartikainen (2013). **Spatio-Temporal Learning via Infinite-Dimensional Bayesian Filtering and Smoothing.** *IEEE Sig. Proc. Magazine, 30(4):51-61.*

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.*

Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, and Simo Särkkä (2021) **Parallel Iterated Extended and Sigma-Point Kalman Smoothers.** *Proc. ICASSP.*