Inference in Time Series

Cheng Soon Ong
Marc Peter Deisenroth

December 2020

Setting

- Time-series model

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{\epsilon}, \qquad \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \quad \boldsymbol{\epsilon} \sim \mathcal{N}\big(\mathbf{0}, \boldsymbol{Q}\big)$$

Setting

- Time-series model

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{\epsilon}, \qquad \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$$

- Compute an expected utility of a state trajectory $\boldsymbol{\tau} := (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T)$

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})]$$

Setting

- Time-series model

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{\epsilon}, \qquad \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$$

- Compute an expected utility of a state trajectory $\boldsymbol{\tau} := (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T)$

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})]$$

  - Reinforcement learning and optimal control
  - Demand forecasting (logistics)
  - Weather/climate forecasts

Setting

- Time-series model

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{\epsilon}, \qquad \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$$

- Compute an expected utility of a state trajectory $\boldsymbol{\tau} := (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T)$

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})]$$

  - Reinforcement learning and optimal control
  - Demand forecasting (logistics)
  - Weather/climate forecasts

- Challenge: Long-term predictions and uncertainty propagation

Approaches

- **Deterministic inference via iterative computation**
  - Iteratively determine marginal distributions $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$
  - Compute expectations $\mathbb{E}_{\boldsymbol{x}_t}[u(\boldsymbol{x}_t)]$ and compute utilities of the form

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})] = \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{x}_t}[u(\boldsymbol{x}_t)] = \sum_{t=0}^{T} \int u(\boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t$$

Approaches

- **Deterministic inference via iterative computation**
  - Iteratively determine marginal distributions $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$
  - Compute expectations $\mathbb{E}_{\boldsymbol{x}_t}[u(\boldsymbol{x}_t)]$ and compute utilities of the form

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})] = \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{x}_t}[u(\boldsymbol{x}_t)] = \sum_{t=0}^{T} \int u(\boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t$$

- **Stochastic inference via trajectory sampling**
  - Generate sample trajectories $\boldsymbol{\tau}^{(s)} = (\boldsymbol{x}_0^{(s)}, \ldots, \boldsymbol{x}_T^{(s)})$
  - Monte-Carlo integration

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})] \approx \frac{1}{S} \sum_{s=1}^{S} U(\boldsymbol{\tau}^{(s)})$$

Deterministic Approximate Inference

Deterministic approximate inference

- Iteratively compute marginals

$$p(\boldsymbol{x}_{t+1}) = \int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)d\boldsymbol{x}_t$$

$$= \int \mathcal{N}\big(f(\boldsymbol{x}_t),\, \boldsymbol{Q}\big)p(\boldsymbol{x}_t)d\boldsymbol{x}_t$$

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}\big(\boldsymbol{0},\, \boldsymbol{Q}\big)$$

Deterministic approximate inference

- Iteratively compute marginals

$$p(\boldsymbol{x}_{t+1}) = \int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)d\boldsymbol{x}_t$$
$$= \int \mathcal{N}\big(f(\boldsymbol{x}_t),\, \boldsymbol{Q}\big)p(\boldsymbol{x}_t)d\boldsymbol{x}_t$$

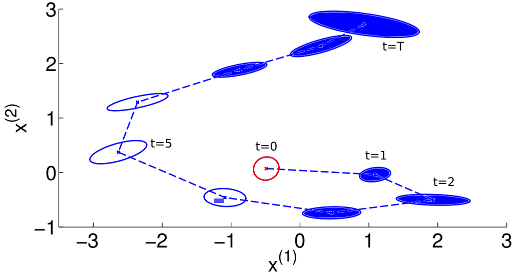
$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}\big(\boldsymbol{0},\, \boldsymbol{Q}\big)$$

No closed-form solution for nonlinear f

Iterative Gaussian approximation

- Common approach: Iterative Gaussian approximation of marginals:

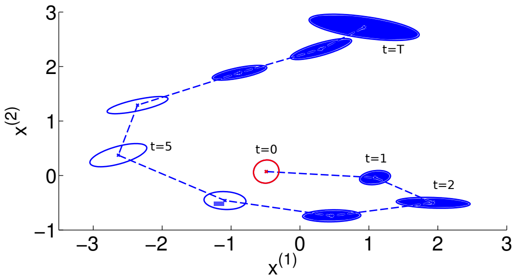
$$p(\boldsymbol{x}_t) \approx \mathcal{N}\big(\boldsymbol{\mu}_t,\, \Sigma_t\big)$$



Iterative Gaussian approximation

- Common approach: Iterative Gaussian approximation of marginals:

$$p(\boldsymbol{x}_t) \approx \mathcal{N}\big(\boldsymbol{\mu}_t,\, \Sigma_t\big)$$



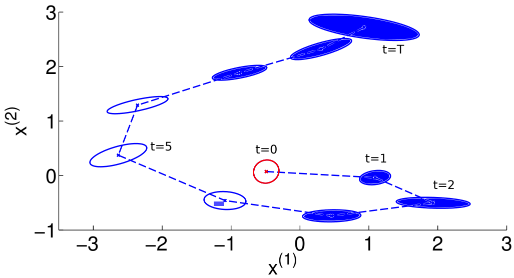
- **Linearization**

⏩ Extended Kalman filter

Iterative Gaussian approximation

- Common approach: Iterative Gaussian approximation of marginals:

$$p(\boldsymbol{x}_t) \approx \mathcal{N}\big(\boldsymbol{\mu}_t,\, \Sigma_t\big)$$

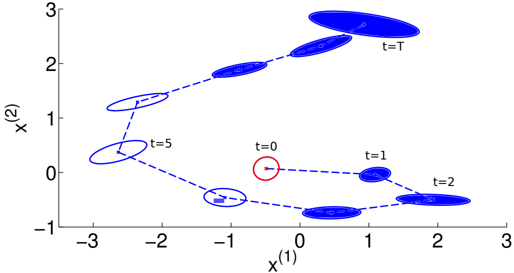


- **Linearization** ⏩ Extended Kalman filter
- **Unscented transformation** ⏩ Unscented Kalman filter

Iterative Gaussian approximation

- Common approach: Iterative Gaussian approximation of marginals:

$$p(\boldsymbol{x}_t) \approx \mathcal{N}\big(\boldsymbol{\mu}_t,\, \Sigma_t\big)$$



- **Linearization** ⏩ Extended Kalman filter
- **Unscented transformation** ⏩ Unscented Kalman filter
- **Moment matching** ⏩ Assumed density filter

Two approaches

$$p(\boldsymbol{x}_{t+1}) = \int \mathcal{N}\left(\boldsymbol{x}_{t+1} \middle| f(\boldsymbol{x}_t), \boldsymbol{Q}\right) p(\boldsymbol{x}_t) d\boldsymbol{x}_t \approx \mathcal{N}\left(\boldsymbol{x}_{t+1} \middle| \boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}\right)$$

Two approaches

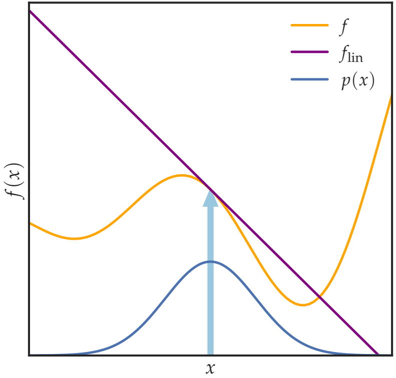
$$p(\boldsymbol{x}_{t+1}) = \int \mathcal{N}\left(\boldsymbol{x}_{t+1} \middle| f(\boldsymbol{x}_t), \boldsymbol{Q}\right) p(\boldsymbol{x}_t) d\boldsymbol{x}_t \approx \mathcal{N}\left(\boldsymbol{x}_{t+1} \middle| \boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}\right)$$

- Approximate f ⏩ Linearization (e.g., Smith et al., 1962)
- Approximate $p(\boldsymbol{x}_t)$ ⏩ Unscented transformation (Julier & Uhlmann, 1995)

Approach 1: Linearization

Key idea (e.g., Smith et al., 1962; Ohab & Stubberud, 1965)

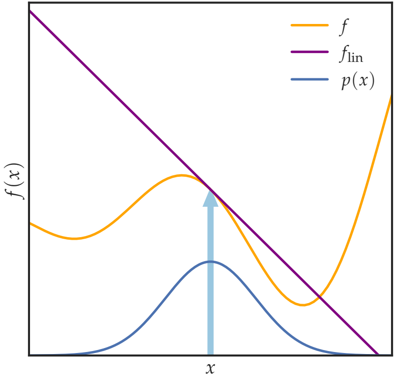
1. Locally linearize f around mean $\boldsymbol{\mu}_t$
2. Compute predictive distribution (Gaussian) for linearized function in closed form



Approach 1: Linearization

Key idea (e.g., Smith et al., 1962; Ohab & Stubberud, 1965)

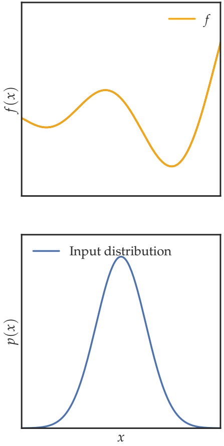
1. Locally linearize f around mean $\boldsymbol{\mu}_t$
2. Compute predictive distribution (Gaussian) for linearized function in closed form



- Linearization: First-order Taylor-series expansion around $\boldsymbol{\mu}_t$
  - Gradient (Jacobian) $df/d\boldsymbol{x}_t$ of f evaluated at $\boldsymbol{\mu}_t$
- Key insight: Gaussians can be pushed through linear functions in closed form

How it works

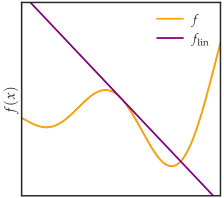
- Compute gradient $\boldsymbol{J}_t := df/d\boldsymbol{x}_t|_{\boldsymbol{x}_t=\boldsymbol{\mu}_t}$

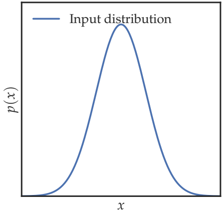


How it works

- Compute gradient $\boldsymbol{J}_t := df/d\boldsymbol{x}_t|_{\boldsymbol{x}_t=\boldsymbol{\mu}_t}$
- Linearized model:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{\mu}_t) + \boldsymbol{J}_t(\boldsymbol{x} - \boldsymbol{\mu}_t)$$





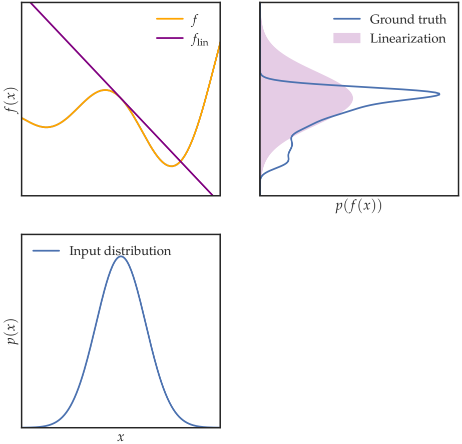
How it works

- Compute gradient $\boldsymbol{J}_t := df/d\boldsymbol{x}_t|_{\boldsymbol{x}_t=\boldsymbol{\mu}_t}$
- Linearized model:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{\mu}_t) + \boldsymbol{J}_t(\boldsymbol{x} - \boldsymbol{\mu}_t)$$

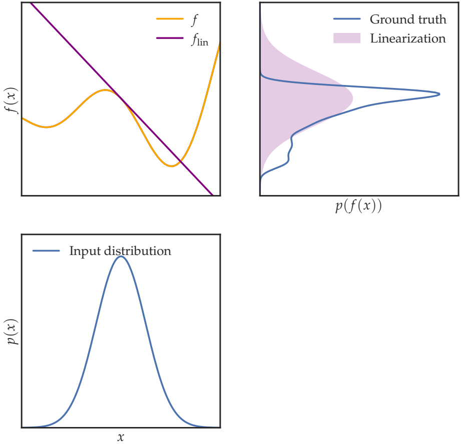
- Approximate predictive distribution is Gaussian:

$$p(f(\boldsymbol{x}_t)) \approx \mathcal{N}\left(f(\boldsymbol{\mu}_t),\ \boldsymbol{J}_t\boldsymbol{\Sigma}_t\boldsymbol{J}_t^\top\right)$$
$$p(\boldsymbol{x}_{t+1}) \approx \mathcal{N}\left(f(\boldsymbol{\mu}_t),\ \boldsymbol{J}_t\boldsymbol{\Sigma}_t\boldsymbol{J}_t^\top + \boldsymbol{Q}\right)$$



Linearization: Properties

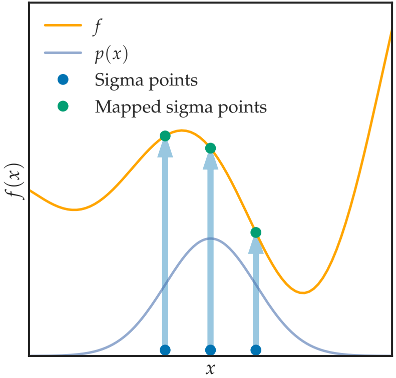
- Conceptually straightforward
- Requires differentiable f
- Tends to underestimate true covariance matrix ⏩ Overconfidence
- Scales cubically in the dimension of $\boldsymbol{x}$
- Widely used in engineering
  (e.g., navigation systems, GPS, Apollo missions)



Approach 2: Unscented transformation

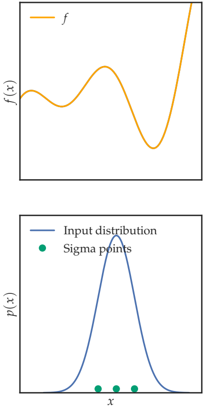
Key idea (Julier & Uhlmann, 1995)

1. Approximate $p(\boldsymbol{x}_t)$ using a small set of deterministically chosen sigma points
2. Map sigma points through f
3. Compute a weighted average of the mean and covariance of the predictive distribution.



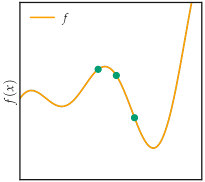
How it works

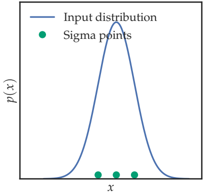
- Determine $2D+1$ sigma points
  $\mathcal{X}_t = \{\boldsymbol{\mu}_t \pm \alpha\left(\sqrt{\boldsymbol{\Sigma}_t}\right)_i, \; i = 1, \ldots, D\}$



How it works

- Determine $2D+1$ sigma points
  $\mathcal{X}_t = \{\boldsymbol{\mu}_t \pm \alpha\left(\sqrt{\boldsymbol{\Sigma}_t}\right)_i,\ i = 1, \ldots, D\}$
- Map sigma points through f to get $f(\mathcal{X}_t)$



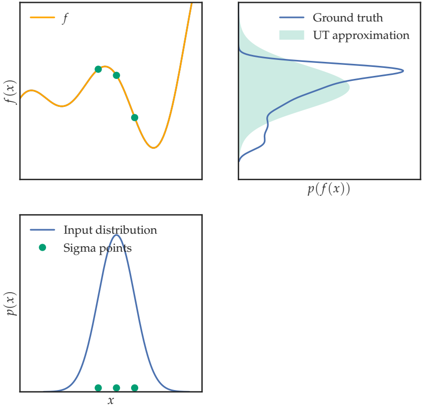


How it works

- Determine $2D+1$ sigma points
  $\mathcal{X}_t = \{\boldsymbol{\mu}_t \pm \alpha\big(\sqrt{\boldsymbol{\Sigma}_t}\big)_i, \;\; i = 1, \dots, D\}$
- Map sigma points through f to get $f(\mathcal{X}_t)$
- Compute mean/covariance of predictive distribution $p(f(\boldsymbol{x}_t))$ as a weighted average

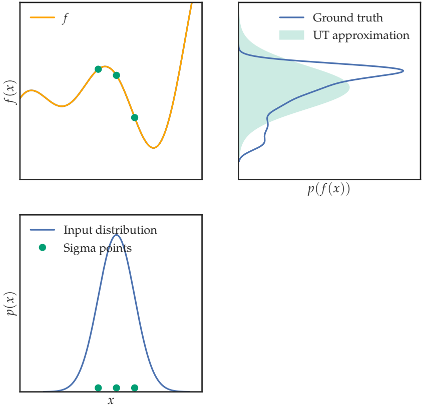
$$\boldsymbol{\mu}_{t+1} \approx \sum_{d=1}^{2D+1} w_d^{\mu} f(\mathcal{X}_t^{(d)})$$

$$\boldsymbol{\Sigma}_{t+1} \approx \sum_{d=1}^{2D+1} w_d^{\Sigma} \big(f(\mathcal{X}_t^{(d)}) - \boldsymbol{\mu}_{t+1}\big)\big(f(\mathcal{X}_t^{(d)}) - \boldsymbol{\mu}_{t+1}\big)^\top$$



Unscented transformation: Properties

- Not a Monte-Carlo method: Sigma points are deterministic, not random
- No explicit calculation of Jacobians
  ⏩ f can be non-differentiable
- Input distribution does not need to be Gaussian
- Higher accuracy (covariance) than linearization (Julier & Uhlmann, 2004)



Stochastic Approximate Inference

Stochastic approximate inference

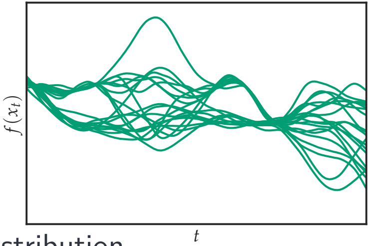
- Sample trajectories $\boldsymbol{\tau}^{(i)} = \left(\boldsymbol{x}_0^{(i)}, \dots, \boldsymbol{x}_T^{(i)}\right)$:
  1. Sample initial state: $\boldsymbol{x}_0^{(i)} \sim p(\boldsymbol{x}_0)$
  2. For $t = 1, \dots, T$:

$$\boldsymbol{x}_t^{(i)} \sim p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^{(i)}) = \mathcal{N}\left(\boldsymbol{x}_t \middle| f(\boldsymbol{x}_{t-1}^{(i)}), \boldsymbol{Q}\right)$$

Stochastic approximate inference

- Sample trajectories $\boldsymbol{\tau}^{(i)} = \left(\boldsymbol{x}_0^{(i)}, \ldots, \boldsymbol{x}_T^{(i)}\right)$:
  1. Sample initial state: $\boldsymbol{x}_0^{(i)} \sim p(\boldsymbol{x}_0)$
  2. For $t = 1, \ldots, T$:

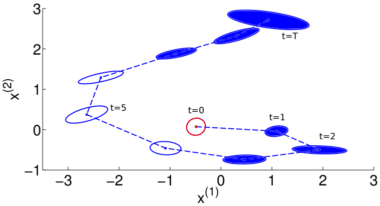
$$\boldsymbol{x}_t^{(i)} \sim p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^{(i)}) = \mathcal{N}\left(\boldsymbol{x}_t \middle| f(\boldsymbol{x}_{t-1}^{(i)}), \boldsymbol{Q}\right)$$

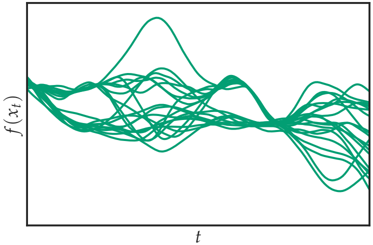


- No parametric restriction to a specific kind of distribution
- Have to store all samples (particles) ⏩ Potential memory issue
- Sequential Monte Carlo (particle filter)
  (e.g., Doucet et al., 2000; Thrun et al., 2005;)

Discussion: Long-term predictions

	Deterministic	Stochastic
Density representation	Parametric	Particles
Bias	Yes	No
Time correlation	No	Yes
Speed	Fast	(Slow)
Parallelization		Easy
Memory consumption	Low	(High)
Gradients	Deterministic	Stochastic





Inference in Gaussian Process Time Series Models

Setting

- Time-series model

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + \boldsymbol{\epsilon}, \qquad \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \quad \boldsymbol{\epsilon} \sim \mathcal{N}\big(\mathbf{0}, \boldsymbol{Q}\big), \quad f \sim GP(\mu, k)$$

Setting

- Time-series model

$$x_{t+1} = f(x_t) + \epsilon, \qquad x_0 \sim p(x_0), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}), \quad f \sim GP(\mu, k)$$

- Two approaches for long-term predictions:
  - Deterministic approximate inference of the marginals $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$
  - Stochastic approximate inference by sampling trajectories $\boldsymbol{\tau}^{(i)} = (\boldsymbol{x}_0^{(i)}, \ldots, \boldsymbol{x}_T^{(i)})$

Deterministic approximate inference

$$p(\boldsymbol{x}_{t+1}) = \int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)d\boldsymbol{x}_t$$

Deterministic approximate inference

$$\begin{aligned} p(\boldsymbol{x}_{t+1}) &= \int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)d\boldsymbol{x}_t \\ &= \int \int p(f(\boldsymbol{x}_t)|f, \boldsymbol{x}_t)p(f)df\, p(\boldsymbol{x}_t)d\boldsymbol{x}_t \end{aligned}$$

Deterministic approximate inference

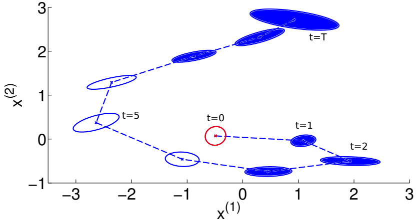
$$p(\boldsymbol{x}_{t+1}) = \int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)d\boldsymbol{x}_t$$
$$= \int \int p(f(\boldsymbol{x}_t)|f, \boldsymbol{x}_t)p(f)df\, p(\boldsymbol{x}_t)d\boldsymbol{x}_t$$

Approaches:

- Linearization (e.g., Ko & Fox, 2009)
- Unscented transformation (e.g., Ko & Fox, 2009)
- Moment matching (e.g., Deisenroth et al., 2009)

Long-term predictions

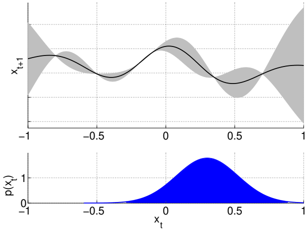
- Iteratively compute $p(\boldsymbol{x}_1), \dots, p(\boldsymbol{x}_T)$



Long-term predictions

- Iteratively compute $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$

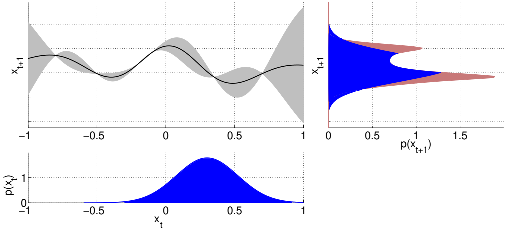
$$\underbrace{p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)}_{\text{GP prediction}} \underbrace{p(\boldsymbol{x}_t)}_{\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)}$$



Long-term predictions

- Iteratively compute $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$

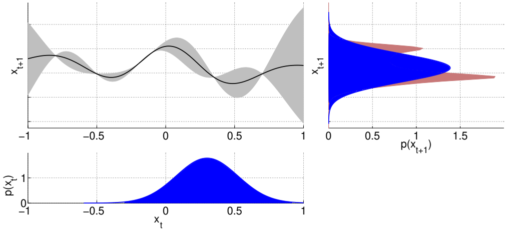
$$\boxed{p(\boldsymbol{x}_{t+1})} = \iint \underbrace{\boxed{p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)}}_{\text{GP prediction}} \underbrace{\boxed{p(\boldsymbol{x}_t)}}_{\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)} \, df \, d\boldsymbol{x}_t$$



Long-term predictions

- Iteratively compute $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$

$$p(\boldsymbol{x}_{t+1}) = \iint \underbrace{p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)}_{\text{GP prediction}} \underbrace{p(\boldsymbol{x}_t)}_{\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)} \, df \, d\boldsymbol{x}_t$$

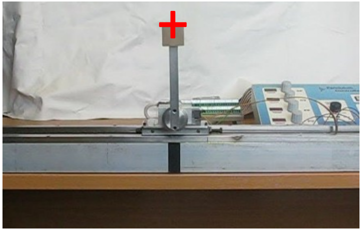


  - GP moment matching (Girard et al., 2003; Quiñonero-Candela et al., 2003)
  - Key ingredient: Computing kernel expectations

Example: Model-based reinforcement learning

- Learn dynamics of a physical system from data ⏩ Gaussian process

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\epsilon}, \quad f \sim GP$$
$$\boldsymbol{u}_t = \pi(\boldsymbol{x}_t; \boldsymbol{\theta})$$



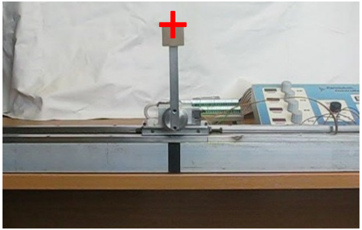
From Deisenroth & Rasmussen (2011)
https://www.youtube.com/PilcoLearner

Example: Model-based reinforcement learning

- Learn dynamics of a physical system from data ⏩ Gaussian process
- Given the learned system, find policy parameters $\boldsymbol{\theta}^*$ that minimize an expected long-term cost

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})] = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{x}_t}[c(\boldsymbol{x}_t)]$$

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\epsilon}, \quad f \sim GP$$
$$\boldsymbol{u}_t = \pi(\boldsymbol{x}_t; \boldsymbol{\theta})$$



From Deisenroth & Rasmussen (2011)
https://www.youtube.com/PilcoLearner

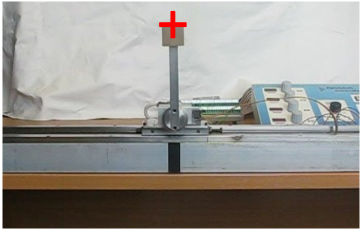
Example: Model-based reinforcement learning

- Learn dynamics of a physical system from data ⏩ Gaussian process
- Given the learned system, find policy parameters $\boldsymbol{\theta}^*$ that minimize an expected long-term cost

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})] = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{x}_t}[c(\boldsymbol{x}_t)]$$

- GP moment matching for long-term predictions

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\epsilon}, \quad f \sim GP$$
$$\boldsymbol{u}_t = \pi(\boldsymbol{x}_t; \boldsymbol{\theta})$$



From Deisenroth & Rasmussen (2011)
https://www.youtube.com/PilcoLearner

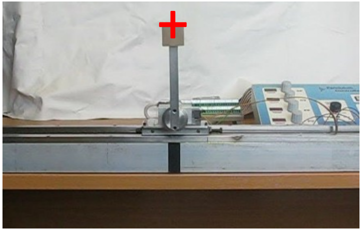
Example: Model-based reinforcement learning

- Learn dynamics of a physical system from data ⏩ Gaussian process
- Given the learned system, find policy parameters $\boldsymbol{\theta}^*$ that minimize an expected long-term cost

$$\mathbb{E}_{\boldsymbol{\tau}}[U(\boldsymbol{\tau})] = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{x}_t}[c(\boldsymbol{x}_t)]$$

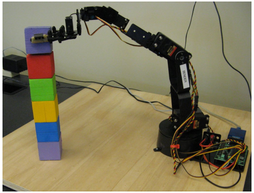
- GP moment matching for long-term predictions
- Gradient descent to find $\boldsymbol{\theta}^*$

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\epsilon}, \quad f \sim GP$$
$$\boldsymbol{u}_t = \pi(\boldsymbol{x}_t; \boldsymbol{\theta})$$

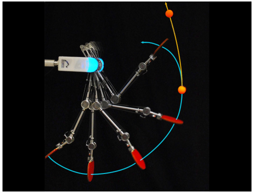


From Deisenroth & Rasmussen (2011)
https://www.youtube.com/PilcoLearner

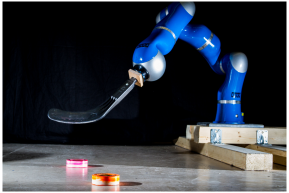
Wide Applicability



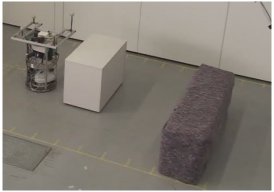
Deisenroth et al. (2011)



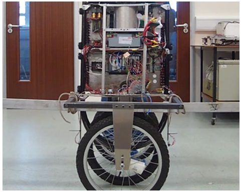
Englert et al. (2013)



Kupcsik et al. (2017)



Bischoff et al. (2013b)



McHutchon (2014)

Bischoff et al. (2013a)

▶▶▶ Application to a wide range of robotic systems

Stochastic approximate inference

- Generating a function draw: Sampling from a T-dimensional multivariate Gaussian
  T: Number of query points

Figure: Generated with GPflow (Matthews et al., 2017)

Stochastic approximate inference

- Generating a function draw: Sampling from a T-dimensional multivariate Gaussian
  T: Number of query points
- Drawing a sample from a GP **scales cubically in T**

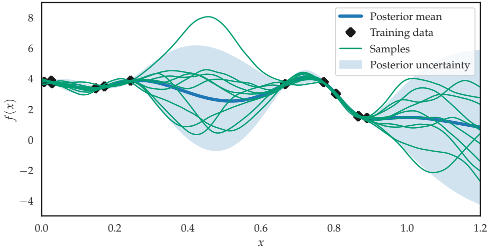


Figure: Generated with GPflow (Matthews et al., 2017)

Stochastic approximate inference

- Generating a function draw: Sampling from a T-dimensional multivariate Gaussian
  T: Number of query points
- Drawing a sample from a GP **scales cubically in T**
- There are some ways around this in low dimensions (e.g., Särkkä et al., 2013; Solin et al., 2018) or by making structural assumptions (e.g., Pleiss et al., 2018)

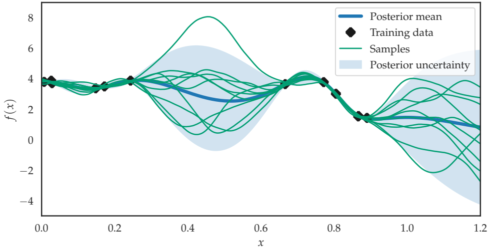


Figure: Generated with GPflow (Matthews et al., 2017)

Stochastic approximate inference

- Generating a function draw: Sampling from a T-dimensional multivariate Gaussian
  T: Number of query points
- Drawing a sample from a GP **scales cubically in T**
- There are some ways around this in low dimensions (e.g., Särkkä et al., 2013; Solin et al., 2018) or by making structural assumptions (e.g., Pleiss et al., 2018)
  - Let's try something else

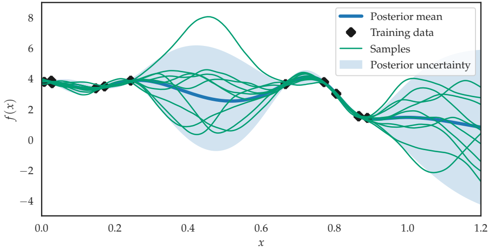


Figure: Generated with GPflow (Matthews et al., 2017)

Decoupled sampling (Wilson et *a*l., 2020*a*)

Key idea

Sample functions from a Gaussian process by exploiting **Matheron's rule** (for Gaussian random variables):

$$\text{posterior} = \text{prior} + \text{data-dependent update}$$

Decoupled sampling (Wilson et al., 2020*a*)

Key idea

Sample functions from a Gaussian process by exploiting **Matheron's rule** (for Gaussian random variables):

$$\text{posterior} = \text{prior} + \text{data-dependent update}$$

- Think about the posterior in terms of samples, not in terms of (conditional) distributions

Decoupled sampling (Wilson et al., 2020*a*)

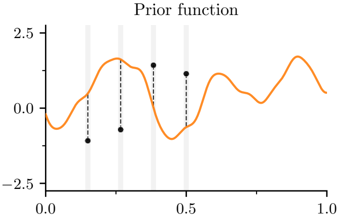
Key idea

Sample functions from a Gaussian process by exploiting **Matheron's rule** (for Gaussian random variables):

$$\text{posterior} = \text{prior} + \text{data-dependent update}$$

- Think about the posterior in terms of samples, not in terms of (conditional) distributions
- Samples from the posterior can be obtained through a two-step procedure:
  1. Sample from prior ⏩ Source of randomness
  2. "Correct" sample using a data-dependent update term
     ⏩ Deterministic transformation

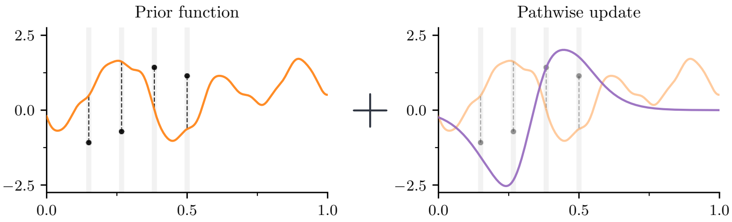
Illustration: Decoupled sampling (Wilson et al., 2020*a*)



$$\underbrace{f^{(s)}(\cdot)}_{\text{sample from prior}}$$

1. Sample from the prior

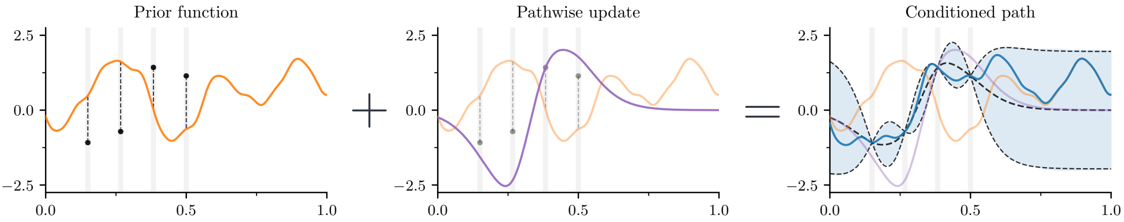
Illustration: Decoupled sampling (Wilson et al., 2020*a*)



$$\underbrace{f^{(s)}(\cdot)}_{\text{sample from prior}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{y} - f^{(s)}(\boldsymbol{X}))}_{\text{data-dependent update}}$$

1. Sample from the prior
2. Add data-dependent update term

Illustration: Decoupled sampling (Wilson et al., 2020a)



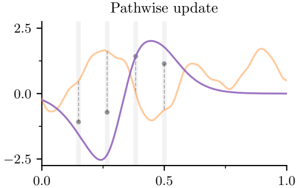
$$\underbrace{f^{(s)}(\cdot)}_{\text{sample from prior}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{y} - f^{(s)}(\boldsymbol{X}))}_{\text{data-dependent update}} = \underbrace{f^{(s)}(\cdot) | \boldsymbol{X}, \boldsymbol{y}}_{\text{sample from posterior}}$$

1. Sample from the prior
2. Add data-dependent update term

Sample from the posterior

Properties

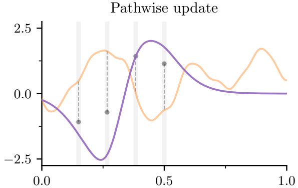
$$\underbrace{f^{(s)}(\cdot)}_{\text{sample from prior}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{y} - f^{(s)}(\boldsymbol{X}))}_{\text{data-dependent update}} = \underbrace{f^{(s)}(\cdot)|\boldsymbol{X}, \boldsymbol{y}}_{\text{sample from posterior}}$$



- Update term depends on error/residual between the prior sample and data $\boldsymbol{y}$

Properties

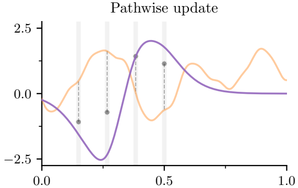
$$\underbrace{f^{(s)}(\cdot)}_{\substack{\text{sample}\\\text{from prior}}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{y} - f^{(s)}(\boldsymbol{X}))}_{\text{data-dependent update}} = \underbrace{f^{(s)}(\cdot)|\boldsymbol{X}, \boldsymbol{y}}_{\substack{\text{sample}\\\text{from posterior}}}$$



- Update term depends on error/residual between the prior sample and data $\boldsymbol{y}$
- Update term is a mapping from prior to posterior

Properties

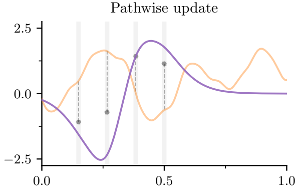
$$\underbrace{f^{(s)}(\cdot)}_{\text{sample from prior}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{y} - f^{(s)}(\boldsymbol{X}))}_{\text{data-dependent update}} = \underbrace{f^{(s)}(\cdot) | \boldsymbol{X}, \boldsymbol{y}}_{\text{sample from posterior}}$$



- Update term depends on error/residual between the prior sample and data $\boldsymbol{y}$
- Update term is a mapping from prior to posterior
- Different representations for prior and update terms
  (e.g., RFF for prior and finite basis-function representation for update)

Properties

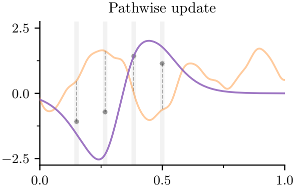
$$\underbrace{f^{(s)}(\cdot)}_{\text{sample from prior}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{y} - f^{(s)}(\boldsymbol{X}))}_{\text{data-dependent update}} = \underbrace{f^{(s)}(\cdot) | \boldsymbol{X}, \boldsymbol{y}}_{\text{sample from posterior}}$$



- Update term depends on error/residual between the prior sample and data $\boldsymbol{y}$
- Update term is a mapping from prior to posterior
- Different representations for prior and update terms (e.g., RFF for prior and finite basis-function representation for update)
  - Sampling from RFF prior scales linearly in the number T of test inputs

Properties

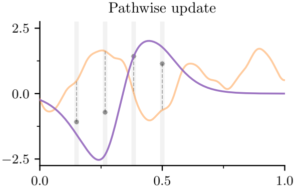
$$\underbrace{f^{(s)}(\cdot)}_{\text{sample from prior}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{y} - f^{(s)}(\boldsymbol{X}))}_{\text{data-dependent update}} = \underbrace{f^{(s)}(\cdot) | \boldsymbol{X}, \boldsymbol{y}}_{\text{sample from posterior}}$$



- Update term depends on error/residual between the prior sample and data $\boldsymbol{y}$
- Update term is a mapping from prior to posterior
- Different representations for prior and update terms (e.g., RFF for prior and finite basis-function representation for update)
  - Sampling from RFF prior scales linearly in the number T of test inputs
  - Update term can be computed linearly in the number T of test inputs

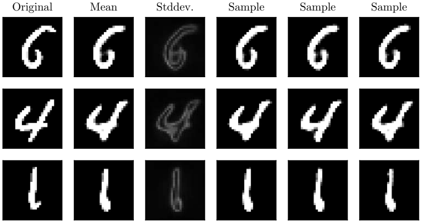
Properties

$$\underbrace{f^{(s)}(\cdot)}_{\substack{\text{sample}\\ \text{from prior}}} + \underbrace{k(\cdot, \boldsymbol{X})\boldsymbol{K}^{-1}(\boldsymbol{y} - f^{(s)}(\boldsymbol{X}))}_{\text{data-dependent update}} = \underbrace{f^{(s)}(\cdot)|\boldsymbol{X}, \boldsymbol{y}}_{\substack{\text{sample}\\ \text{from posterior}}}$$



- Update term depends on error/residual between the prior sample and data $\boldsymbol{y}$
- Update term is a mapping from prior to posterior
- Different representations for prior and update terms (e.g., RFF for prior and finite basis-function representation for update)
  - Sampling from RFF prior scales linearly in the number T of test inputs
  - Update term can be computed linearly in the number T of test inputs
- **Functions can be sampled efficiently** (linearly in the number of test inputs)

Applications

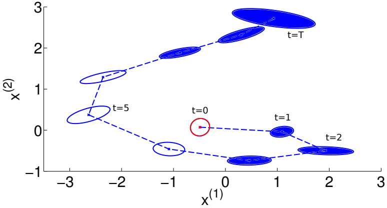


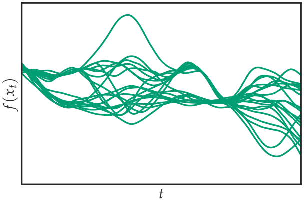
From Wilson et al. (2020b)

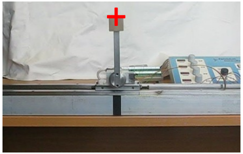
From Borovitskiy et al. (2020)

- Deep convolutional GP auto-encoders (Wilson et al., 2020b)
- Bayesian optimization with Thompson sampling (Wilson et al., 2020a)
- Sampling from GPs on manifolds (Borovitskiy et al., 2020)
- Model-based reinforcement learning

Summary







- Propagate uncertainty through a nonlinear dynamical system
- Deterministic approximate inference (linearization, unscented transformation)
- Stochastic approximate inference (sampling)
- Examples in the context of GP dynamical systems

References

Bischoff, B., Nguyen-Tuong, D., Koller, T., Markert, H., and Knoll, A. (2013a). Learning Throttle Valve Control Using Policy Search. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*.

Bischoff, B., Nguyen-Tuong, D., Markert, H., and Knoll, A. (2013b). Learning Control Under Uncertainty: A Probabilistic Value-Iteration Approach. In *Proceedings of the European Symposium on Artificial Neural Networks*.

Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2020). Matern Gaussian Processes on Riemannian Manifolds. In *Advances in Neural Information Processing Systems*.

Deisenroth, M. P., Huber, M. F., and Hanebeck, U. D. (2009). Analytic Moment-based Gaussian Process Filtering. In *Proceedings of the International Conference on Machine Learning*.

Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the International Conference on Machine Learning*.

References (cont.)

Deisenroth, M. P., Rasmussen, C. E., and Fox, D. (2011). Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*.

Doucet, A., Godsill, S. J., and Andrieu, C. (2000). On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 10:197–208.

Englert, P., Paraschos, A., Peters, J., and Deisenroth, M. P. (2013). Model-based Imitation Learning by Probabilistic Trajectory Matching. In *Proceedings of the International Conference on Robotics and Automation*.

Girard, A., Rasmussen, C. E., Quiñonero Candela, J., and Murray-Smith, R. (2003). Gaussian Process Priors with Uncertain Inputs—Application to Multiple-Step Ahead Time Series Forecasting. In *Advances in Neural Information Processing Systems*.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422.

References (cont.)

Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. (1995). A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators. In *Proceedings of the American Control Conference*.

Ko, J. and Fox, D. (2009). GP-BayesFilters: Bayesian Filtering using Gaussian Process Prediction and Observation Models. *Autonomous Robots*, 27(1):75–90.

Kupcsik, A., Deisenroth, M. P., Peters, J., Poha, L. A., Vadakkepata, P., and Neumann, G. (2017). Model-based Contextual Policy Search for Data-Efficient Generalization of Robot Skills. *Artificial Intelligence*.

Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. (2017). GPflow: A Gaussian Process Library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6.

McHutchon, A. (2014). *Nonlinear Modelling and Control usingGaussian Processes*. PhD thesis, University of Cambridge.

References (cont.)

Ohab, R. F. and Stubberud, A. R. (1965). A Technique for Estimating the State of a Nonlinear System. *IEEE Transactions on Automatic Control*, 10:150–155.

Pleiss, G., Gardner, J. R., Weinberger, K. Q., and Wilson, A. G. (2018). Constant-Time Predictive Distributions for Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*.

Quiñonero-Candela, J., Girard, A., Larsen, J., and Rasmussen, C. E. (2003). Propagation of Uncertainty in Bayesian Kernel Models—Application to Multiple-Step Ahead Forecasting. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Särkkä, S., Solin, A., and Hartikainen, J. (2013). Spatiotemporal Learning via Infinite-Dimensional Bayesian Filtering and Smoothing: A Look at Gaussian Process Regression Through Kalman Filtering,. *IEEE Signal Processing Magazine*, 30(4):51–61.

Smith, G. L., Schmidt, S. F., and McGee, L. A. (1962). Application of Statistical Filter Theory to the Optimal Estimation of Position and Velocity on Board a Circumlunar Vehicle. Technical report, NASA.

References (cont.)

Solin, A., Hensman, J., and Turner, R. E. (2018). Infinite-Horizon Gaussian Processes. In *Advances in Neural Information Processing Systems*.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.

Wilson, J. T., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2020a). Efficiently Sampling Functions from Gaussian Process Posteriors. In *Proceedings of the International Conference on Machine Learning*.

Wilson, J. T., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2020b). Pathwise Conditioning of Gaussian Processes. *arXiv:2011.04026*.