Gaussian Process Based Model-free Control with Q-Learning ⋆

Jan Hauser * **Daniel Pachner** ** **Vladimír Havlena** *

* *Department of Control Engineering, Faculty of Electrical Engineering of Czech Technical University in Prague, Technicka 2, 166 27 Praha 6, Czech Republic (e-mail: {hauseja3,havlena}@fel.cvut.cz)*
** *Honeywell HBT Architecture & Innovation Team, V Parku 2326/18, 148 00 Prague, Czech Republic (e-mail: daniel.pachner@honeywell.com)*

Abstract: The aim of this paper is to demonstrate a new algorithm for Machine Learning (ML) based on Gaussian Process Regression (GPR) and how it can be used as a practical control design technique. An optimized control law for a nonlinear process is found directly by training the algorithm on noisy data collected from the process when controlled by a sub-optimal controller. A simplified nonlinear Fan Coil Unit (FCU) model is used as an example for which the fan speed control is designed using the off-policy Q-learning algorithm. Additionally, the algorithm properties are discussed, i.e. learning process robustness, GP kernel functions choice. The simulation results are compared to a simple PI, designed based on a linearized model.

1. INTRODUCTION

Model-free control techniques assume that no mathematical model of the controlled process is available and the controller is designed from the measurement data. One such approach would collect the data in advance during some time window to use it offline for a controller design. A different approach would attempt to use the data in the real time to improve the control continuously. In this article the former offline approach is considered, i.e. the situation when some sub-optimal controller was already in use and the data were collected and can be used to optimize or improve that controller. Many existing control design techniques first create a model from data to use it for a control design method afterwards, which makes sense if some reliable modeling information, e.g. model structure, is available. A different approach, used in this paper, is the controller designed directly from the data, without creating any process model. This approach can have some advantages especially if little or nothing is known about the process or if the process is nonlinear and no analytical control design method is available.

The Q-learning is an off-policy machine learning (ML) iterative algorithm (Sutton and Barto, 2018), which approximates certain function satisfying the Bellman equation. This Q function then defines a controller. Q-learning was developed for Markov Decision Process (MDP) with finite number of states and later generalized to continuous state spaces (van Hasselt and Wiering, 2007; Gaskett et al., 1999). If the analytical form of the Q function is unknown in continuous state spaces, it may be represented by an universal function approximating method such as Neural Network or Gaussian Process Regression (GPR).

GPR is a non-parametric regression technique (Rasmussen and Williams, 2006) which is able to approximate any continuous target function uniformly. GP uses various covariance (kernel) functions (Williams, 1999) to define the data covariance matrices. The choice of the kernel can have significant impact on the accuracy of the regression model.

The contribution of this paper is the practical and efficient combination of Q-learning and GPR resulting in unbiased estimate of the Q function. It is shown in (Bradtke and Barto, 1996) that there is a bias introduced in parametric least squares estimate, due to the errors-in-variables (Young, 1984). The paper mentioned above describes how to solve such a case for parametric estimate. Here in this paper, it is shown how to prevent this bias in non-parametric estimate (GP). Often the training data are simulated by a model and the statistical properties of the algorithm are thus less important. Here a smaller dataset from a process affected by unmeasured disturbances is targeted (of a size $\sim 10^3$). This requires the information in the data to be used efficiently.

A simple Fan Coil Unit (FCU) model approximation is used to demonstrate the approach. FCU is a nonlinear system widely used for both air heating and cooling in buildings. A linear control design cannot achieve optimal FCU control in terms of energy consumption and user comfort (Arguello-Serrano and Velez-Reyes, 1999). FCU model used here is highly simplified to make the result easier to interpret. It is supposed that ML will be able to optimize a real FCU control based on several days data.

For reader's understanding, some essential theory is briefly introduced. In Section 2, the other papers from this

⋆ This work has been supported by the projects 18-26278S and SGS19/174/OHK3/3T/13 sponsored by Grant Agency of the Czech Republic.

stream are commented, then the GPR algorithm and the Q-learning mechanism are briefly described. Following Section 3 describes the main contribution and novelty of this paper, the unbiased Q-learning algorithm using GPR. In Section 4, a reduced FCU model is explained. Next Section 5 presents the results of these techniques and Section 6 concludes and proposes a future work.

2. BACKGROUND

In order to understand presented algorithm properly, the GP and Q-learning principles are presented here as a background material. Also the related work is commented in this section.

2.1 Gaussian Process Regression

GPR is a supervised learning regression model, which can be also described as a distribution over functions (Rasmussen and Williams, 2006). It is the function value estimator for an unknown function $f(\mathbf{x})$ considering any dataset $(\mathbf{X}, \mathbf{y})$, which can be written as

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')\right),$$

where mean $m(\mathbf{x})$ and covariance function $\kappa(\mathbf{x}, \mathbf{x}')$ are given a priori up to some hyperparameters and are defined as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}\left[f(\mathbf{x})\right], \\ \kappa(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f(\mathbf{x}') - m(\mathbf{x}')\right)^\top\right], \end{aligned}$$

where $\mathbb{E}$ is the expectation, $\mathbf{x}$ and $\mathbf{x}'$ are a pair of vectors in the data space. A dataset is a number of such vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. In this article $f(\mathbf{x}) \in \mathbb{R}$.

Assume the finite training set $\mathbf{X}$ and the finite testing set $\mathbf{X}_p$, then GPR can predict $z_j = f(\mathbf{x}_{p_j})$, where $\mathbf{x}_{p_j} \in \mathbf{X}_p$, by using data $\mathbf{x}_i \in \mathbf{X}$ and their function values $f(\mathbf{x}_i)$. The function values $f(\mathbf{X})$ themselves do not need to be accessible but rather their noisy measurements $y_i = f(\mathbf{x}_i) + \varepsilon_i$, where ε_i is independent identically distributed (i.i.d) Gaussian noise with variance σ_n^2. The prior covariance of the noisy values is defined as

$$\operatorname{cov}(\mathbf{y}) = \kappa(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}. \tag{1}$$

Then the prior joint probability distribution function (p.d.f) can be defined for training and testing sets values as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} \sim \begin{bmatrix} f(\mathbf{X}) + \boldsymbol{\varepsilon} \\ f(\mathbf{X}_p) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_p) \end{bmatrix}, \begin{bmatrix} \kappa(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & \kappa(\mathbf{X}, \mathbf{X}_p) \\ \kappa(\mathbf{X}_p, \mathbf{X}) & \kappa(\mathbf{X}_p, \mathbf{X}_p) \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{m} \\ \mathbf{m}_z \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_n^2 \mathbf{I} & \mathbf{K}_{fz} \\ \mathbf{K}_{zf} & \mathbf{K}_{zz} \end{bmatrix}\right),$$

where $\mathcal{N}$ is normal distribution defined by mean and covariance, $\mathbf{K}_{fz}$ is a $n \times n_p$ matrix of the covariances of all pairs of training and testing datasets, and $\mathbf{K}_{ff}$, $\mathbf{K}_{zz}$, $\mathbf{K}_{zf}$ analogously. Let's also use a notation $\mathbf{K}_{yy} = \mathbf{K}_{ff} + \sigma_n^2 \mathbf{I}$ for covariance of noisy measurements $\mathbf{y}$ (1). There are many useful covariance functions $\kappa(\mathbf{x}, \mathbf{x}')$ called kernels, e.g. squared exponential (SE)

$$\begin{aligned} \kappa(\mathbf{x}, \mathbf{x}') &= \sigma_f^2 \exp\left(-\frac{1}{2}\left(\mathbf{x} - \mathbf{x}'\right)^\top \boldsymbol{\Lambda}^{-1}\left(\mathbf{x} - \mathbf{x}'\right)\right), \\ \boldsymbol{\Lambda} &= \operatorname{diag}(\lambda_1^2, \ldots, \lambda_n^2), \end{aligned}$$

or polynomial kernel of d-degree

$$\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^\top \mathbf{x}' + c\right)^d,$$

where $c \geq 0$. These kernel functions are also scalable by their hyperparameters, i.e. these are signal variance σ_f^2 and length-scale $\boldsymbol{\Lambda}$ for SE kernel or degree d and soft-margin c for polynomial kernel.

If $\mathbf{y}$ is known, then the posterior conditional normal distribution of $\mathbf{z}$ can be defined. The predictive GPR relationships are following

$$\begin{aligned} p(\mathbf{z}|\mathbf{y}) &= \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z), \\ \boldsymbol{\mu}_z &= \mathbb{E}\left[\mathbf{z}|\mathbf{y}\right] = \mathbf{m}_z + \mathbf{K}_{zf}\mathbf{K}_{yy}^{-1}(\mathbf{y} - \mathbf{m}), \qquad (2) \\ \boldsymbol{\Sigma}_z &= \mathbf{K}_{zz} - \mathbf{K}_{zf}\mathbf{K}_{yy}^{-1}\mathbf{K}_{fz}, \qquad (3) \end{aligned}$$

where $\boldsymbol{\mu}_z$ is a mean vector and $\boldsymbol{\Sigma}_z$ is a covariance matrix.

There is an important relationship between the Kalman filter equations and (2, 3) which will be used later. Specifically, we remind that the term $\mathbf{G} = \mathbf{K}_{zf}\mathbf{K}_{yy}^{-1}$ is the Kalman gain matrix. Assuming the unknown continuous function f is a GP, then training points from dataset $\mathbf{X}$ and the observed function values $\mathbf{y}$ define the posterior expectations (predictions) $\mathbf{z}$ for any test points over a dataset $\mathbf{X}_p$.

2.2 Q-learning

This section describes the basic principles of Q-learning algorithm (Sutton and Barto, 2018), which is a model-free off-policy reinforcement learning approach. Then the generalized policy iteration algorithm based on Bellman equation is pointed out.

For purpose of this section, let's highlight the analogies and slight differences between two closely related fields: Optimal Control Theory (OCT) and MDP. At a discrete time k, the vectors of states $\mathbf{x}_k \in \mathcal{X}$ and inputs $\mathbf{u}_k \in \mathcal{U}$ are usually considered in OCT for a process model, whereas MDP uses the Markov process states $\mathbf{s}_k \in \mathcal{S}$ and the agent's actions $\mathbf{a}_k \in \mathcal{A}$ analogously. Note that the sets $\mathcal{X}$ and $\mathcal{U}$ are usually real vector spaces in control problems whereas $\mathcal{S}$ and $\mathcal{A}$ may often be finite sets in MDP. The process model itself is an analogy of probability transition matrix $p(\mathbf{s}_{k+1}|\mathbf{s}_k, \mathbf{a}_k)$ of MDP. Control law, or a state feedback $\mathbf{u}_k = C(\mathbf{x}_k)$ in OCT is an analogy of a deterministic policy $\mathbf{a}_k = \pi(\mathbf{s}_k)$. A stochastic policy, usually not used in OCT, defines the joint p.d.f. $\pi(\mathbf{s}_k, \mathbf{a}_k)$ instead of an explicit function. An important difference exists between the reward $r(\mathbf{s}_k, \mathbf{a}_k, \mathbf{s}_{k+1})$ used in MDP (bounded, to be

maximized) and loss function $\ell(\mathbf{x}_k, \mathbf{u}_k)$ used in OCT (not bounded in general, to be minimized, almost never depending on $\mathbf{x}_{k+1}$). The ML theory will be discussed below mostly with OCT notations and assumptions.

Q Function Generally, Q function is a scalar function of a state-input (state-action) pair, which maps to real values

$$Q : \mathbf{x} \times \mathbf{u} \to \mathbb{R}.$$

It is possible to talk either about Q function $Q^\pi(\mathbf{x}, \mathbf{u})$ pertaining to a given policy π or the function $Q^*(\mathbf{x}, \mathbf{u})$, which pertains to the optimal policy π^*. Q (and Q^*) describes the expected total discounted loss received by the controller starting from $\mathbf{x}$ with a control action $\mathbf{u}$ and following with the policy π (optimal π^*) thereafter. Q^*, as function of $\mathbf{u}$, is thus a measure of quality of selecting the control action $\mathbf{u}$ in a given state $\mathbf{x}$. Q^* is minimized by the optimal control action(s) because it can only be made worse. There is also an important parallel between Q function and the value (cost-to-go) function V used in Dynamic Programming. It is also related to Lyapunov function and stability theory. V is not used for purpose of this paper. For a policy π, not necessarily optimal, and an instantaneous loss $\ell(\mathbf{x}_k, \mathbf{u}_k)$ at time $k \in \mathbb{N}$, the Q is defined as

$$Q^\pi(\mathbf{x}_k, \mathbf{u}_k) = l(\mathbf{x}_k, \mathbf{u}_k) + \mathbb{E}\left[\sum_{i=1}^{\infty} \gamma^i \ell(\mathbf{x}_{k+i}, \mathbf{u}^\pi_{k+i}) \middle| \mathbf{x}_k, \mathbf{u}_k \right], \quad (4)$$

where $\gamma \in (0, 1]$ is a discount factor. Q^* is defined as follows

$$Q^*(\mathbf{x}_k, \mathbf{u}_k) = l(\mathbf{x}_k, \mathbf{u}_k) + \mathbb{E}\left[\sum_{i=1}^{\infty} \gamma^i \min_{\mathbf{u}_{k+i}} \ell(\mathbf{x}_{k+i}, \mathbf{u}_{k+i}) \middle| \mathbf{x}_k, \mathbf{u}_k \right]. \quad (5)$$

The relationship between the function Q^* and the optimal policy π^* is

$$\pi^*(\mathbf{x}) = \arg\min_{\mathbf{u}} Q^*(\mathbf{x}, \mathbf{u}). \quad (6)$$

The Bellman equation (in optimal control apps more usually expressed in terms of the value function $V(\mathbf{x}) = \min_{\mathbf{u}} Q(\mathbf{x}, \mathbf{u})$) provides the recursive approach for finding the functions Q^π and Q^*. It follows directly from (4, 5). For an optimal policy π^*, function $Q^*(\mathbf{x}_k, \mathbf{u}_k)$ must satisfy

$$Q^*(\mathbf{x}_k, \mathbf{u}_k) = \ell(\mathbf{x}_k, \mathbf{u}_k) + \gamma \mathbb{E}\left[\min_{\mathbf{u}_{k+1}} Q^*(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) \middle| \mathbf{x}_k, \mathbf{u}_k \right], \quad (7)$$

and for general policy π analogously by using (4).

Generalized Policy Iteration Generalized Policy iteration (GPI) algorithm (Sutton and Barto, 2018) calculates Q function from Bellman equation (4) for current policy and then improves the current policy by seeking for minimum values of Q^π with respect to $\mathbf{u}$ for each $\mathbf{x}$. Such minimizing $\mathbf{u}$ defines the new policy. This process is repeated until convergence to Q^*. The starting policy is selected as stabilizing in order to ensure the initial Q^π is finite.

2.3 Related Work

There are many related papers presented in this article stream. Various of them uses the GP for on-policy learning, an example of such approach is the commonly used *state action reward state action* or SARSA (Sutton and Barto, 2018). For these algorithms the value function is learned for the same policy the samples are collected from. On the other hand, the off-policy learning of value function directly approximates the optimal value function regardless the policy samples are collected from, still there could be e.g. a requirement for initial policy to be stable. An example of this behaviour is the already presented Q-learning algorithm (Chowdhary et al., 2014). In this paper, Q-learning is selected in order to find optimal control strategy for any process data. It is also used in combination with GP, which in our algorithm finds unbiased estimate of the Q function.

The common usage of this combination with temporal differences (TD) learning (Engel et al., 2003, 2005) results in biased Q function estimate. This error is described and solved for parametric estimate in (Bradtke and Barto, 1996) and the unbiased non-parametric estimate method is the main contribution of this paper. Such a bias is introduced due to the errors-in-variables (Young, 1984). More generally, this is also called regression dilution bias (A Fuller, 2019) and it causes biasing of the regression slope towards zero. Having a noise in the dependent variable causes the uncertainty. On the other hand, having a noise in the independent variable causes the above mentioned bias.

There could be also found articles presenting model-based reinforcement learning algorithms (Jung and Stone, 2010; Rasmussen and Kuss, 2004). In those papers, GP is commonly used not only for estimation of the value function (Q function respectively), but also for estimation of the process dynamics. It was found non-necessary and expensive for the purpose of control, only input-output data are used in a batch form. Which leads us to the next common problem burdening this area. The difference between batch and online usage of the algorithms. For the purpose of this paper, online approach is being investigated.

Last but not least, many of the papers are using some kind of sparsification methods to reduce the complexity of calculations. An example of sparsification approach is (Csato and Opper, 2002) or the induction points approach (Bijl et al., 2015). This paper does not use sparsification, but considers it as possible way to lower the dimensionality. Presented algorithm eliminates repeated calculation of operations which are computationally heavy.

3. Q-LEARNING WITH GPR

In this section, the unbiased Q-learning algorithm with GPR is introduced. For a finite (in number of states and

actions) MDP, the Q-learning algorithms may approximate the Q function at every element of the state-action space using the observed samples $\mathbf{x}_k, \mathbf{u}_k$, which were encountered during interaction with a system.

This paper considers GPR as a continuous Q^π function approximation method and then optimizes the control action using (2.2.2) via numerical minimization. Let us define the set of training and prediction points for the GPR as concatenations of points in the state-action space

$$\mathbf{X}_p^\pi = \left\{ \begin{array}{c} \mathbf{x}_2, \mathbf{u}_2^\pi \\ \vdots \\ \mathbf{x}_k, \mathbf{u}_k^\pi \end{array} \right\}, \; \mathbf{X} = \left\{ \begin{array}{c} \mathbf{x}_1, \mathbf{u}_1 \\ \vdots \\ \mathbf{x}_{k-1}, \mathbf{u}_{k-1} \end{array} \right\}.$$

Here $\mathbf{X}$ is a collection of state-action pairs visited by the process whereas $\mathbf{X}_p^\pi$ is a collection of states observed as results of the actions accompanied with the actions the evaluated strategy π would presumably apply there. Note $\mathbf{X}_p^\pi$ is known without actually applying the strategy π. That is why the approach may use a historical dataset to optimize π. Also, the concatenation of the losses will be used

$$\boldsymbol{\ell} = \begin{bmatrix} \ell(\mathbf{x}_1, \mathbf{u}_1) \\ \vdots \\ \ell(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \end{bmatrix},$$

and let the Q function be a GP with known kernel.

Firstly, the commonly used approach resulting in biased estimate is described. The notation $\mathbf{f}$ (shortened $f(\mathbf{X})$) and $\mathbf{z}$ for the vectors of unknown function values used in GPR context will be preserved. Also denote

$$\mathbf{z}^\pi = Q^\pi(\mathbf{X}_p^\pi), \quad \tilde{\mathbf{z}}^\pi = \mathbb{E}[\mathbf{z}^\pi | \mathbf{X}], \quad \mathbf{f}^\pi = Q^\pi(\mathbf{X}),$$

where $\mathbf{f}^\pi = \boldsymbol{\ell} + \gamma \tilde{\mathbf{z}}^\pi$ and $\tilde{\mathbf{z}}^\pi$ is the expectation of $\mathbf{z}^\pi$ conditioned by $\mathbf{X}$. The joint p.d.f. of $\mathbf{f}^\pi$ and $\tilde{\mathbf{z}}^\pi$ is

$$\begin{bmatrix} \mathbf{f}^\pi \\ \tilde{\mathbf{z}}^\pi \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m} \\ \mathbf{m}_{\tilde{z}} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{f\tilde{z}} \\ \mathbf{K}_{\tilde{z}f} & \mathbf{K}_{\tilde{z}\tilde{z}} \end{bmatrix} \right).$$

Using (2) to find the estimates $\begin{bmatrix} \hat{\mathbf{f}}^\pi & \hat{\mathbf{z}}^\pi \end{bmatrix}^\top$ of $\begin{bmatrix} \mathbf{f}^\pi & \mathbf{z}^\pi \end{bmatrix}^\top$ leads to two cases. For deterministic processes, where $\tilde{\mathbf{z}}^\pi = \mathbf{z}^\pi$, $\mathbf{K}_{f\tilde{z}} = \mathbf{K}_{fz}$, $\mathbf{K}_{\tilde{z}f} = \mathbf{K}_{zf}$ and $\mathbf{K}_{\tilde{z}\tilde{z}} = \mathbf{K}_{zz}$, the estimation result is straightforward. Unfortunately, for stochastic processes, where $\tilde{\mathbf{z}}^\pi \neq \mathbf{z}^\pi$, the covariances $\mathbf{K}_{f\tilde{z}}$, $\mathbf{K}_{\tilde{z}f}$ and $\mathbf{K}_{\tilde{z}\tilde{z}}$ are unknown. Not $\tilde{\mathbf{z}}^\pi$ but only $\mathbf{z}^\pi$ is known, which is a situation similar to the errors-in-variables regression models (Young, 1984). Then using $\mathbf{K}_{fz}$ as an approximation of $\mathbf{K}_{f\tilde{z}}$, and similarly for the other covariances $\mathbf{K}_{zf}, \mathbf{K}_{zz}$, causes such estimate is biased in general. Such a biased estimation algorithm is e.g. used in (Engel et al., 2003). This bias is eliminated in (Bradtke and Barto, 1996) for parametric estimation algorithm, i.e. Least-Squares Temporal Difference. Comparison of this algorithm and the biased estimation is shown on a simple first order system with quadratic Q function

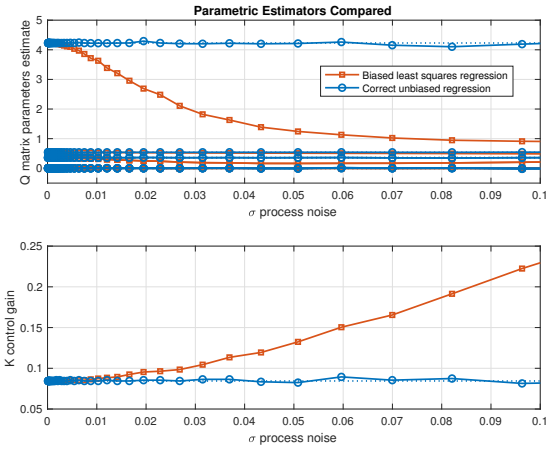

Fig. 1. Comparison of two parametric estimators of Q function for simple first order process showing the above discussed bias impact. The unbiased parametric regression is consistent with (Bradtke and Barto, 1996). At the top figure, the estimates of five elements of (8) (omitting the constant element q_1) are compared to the true values found by solving the Riccati equation. Note that couple of elements are overlapping around zero. The bottom figure compares control gains K calculated by minimizing of estimated Q functions over $\mathbf{u}$ at some specific x_k. Results are presented for increasing noise variance of independent variables on sufficiently big set of data ($\sim 10^4$).

$$Q(x_k, u_k) = \frac{1}{2} \begin{bmatrix} 1 \\ x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} q_1 & q_2 & q_3 \\ q_2 & q_4 & q_5 \\ q_3 & q_5 & q_6 \end{bmatrix} \begin{bmatrix} 1 \\ x_k \\ u_k \end{bmatrix}, \tag{8}$$

where parameters q are estimated and compared to the true values found by solving the Riccati equation. Also control gains K are compared. These gains were calculated by minimizing of estimated Q functions (8) over $\mathbf{u}$ at some specific x_k. See Fig. (1).

The following part of this section describes how to eliminate this bias of $\mathbf{z}^\pi$ for non-parametric estimation using GP. The notation $\mathbf{y}$ for the noisy observed values from GPR context is preserved as well. The noisy realization of $\mathbf{f}^\pi$ is $\mathbf{y}^\pi = \boldsymbol{\ell} + \gamma \mathbf{z}^\pi$.

The conditional means based on (2) and (7) are for this non-parametric estimator

$$\mathbb{E}\left[\begin{array}{c} \mathbf{f}^\pi \\ \mathbf{z}^\pi \end{array} \middle| \mathbf{y}^\pi \right] = \begin{bmatrix} \mathbf{m} \\ \mathbf{m}_z \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ff} \\ \mathbf{K}_{zf}^\pi \end{bmatrix} \mathbf{K}_{yy}^{-1} \left(\boldsymbol{\ell} + \gamma \mathbf{z}^\pi - \mathbf{m} \right). \tag{9}$$

The equation (9) may seem useless because the Q estimates depend on $\mathbf{z}^\pi$ value, which is not known. Recall that Q function is neither measured nor observed. However, consider the left hand side equals to the true values $\mathbf{f}^\pi$ and $\mathbf{z}^\pi$ for $k \to \infty$, sufficient excitation in the state-action space, and when $\mathbf{m} = \mathbf{f}^\pi$, $\mathbf{m}_z = \mathbf{z}^\pi$ respectively. The $\mathbf{f}^\pi$ and $\mathbf{z}^\pi$ thus represent a fixed point of the following iterations (index π dropped)

$$\begin{bmatrix} \mathbf{f}^{(i+1)} \\ \mathbf{z}^{(i+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{(i)} \\ \mathbf{z}^{(i)} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ff} \\ \mathbf{K}_{zf} \end{bmatrix} \mathbf{K}_{yy}^{-1} \left(\boldsymbol{\ell} + \gamma \mathbf{z}^{(i)} - \mathbf{f}^{(i)} \right). \quad (10)$$

An estimate may be calculated starting the iterations from $\mathbf{m}, \mathbf{m}_z$. Instead of actually iterating (10), a system of linear equations, which satisfy the fixed point values, can be solved. However, this system of linear equations will typically be ill-conditioned. In general, the update (9) does not uniformly reduce the uncertainty and the respective mapping is not a contraction. One may now use the Kalman filter and GPR analogy to understand that the latter happens because some linear combinations of the Q function values are not observable (Kwakernaak, 1972) and some information from the starting values $\mathbf{m}, \mathbf{m}_z$ does not vanish. We propose to regularize this situation by shifting the unobservable poles from 1 to some stable real pole $1-\xi$, i.e. inside the unit circle by $\xi > 0$. Practically, this means that the unobservable Q function values will be estimated as zeros ($\mathbf{m}$ could also be used). This regularization adds a fictitious time update step to the Kalman filter, shrinking the right hand side of (10) by the factor of $1-\xi$. It results in the following estimates

$$\begin{bmatrix} \hat{\mathbf{f}}^{\pi} \\ \hat{\mathbf{z}}^{\pi} \end{bmatrix} = \left((1-\xi) \left[\mathbf{G}^{\pi}, \; -\gamma \mathbf{G}^{\pi} \right] - \xi \mathbf{I} \right)^{-1} \mathbf{G}^{\pi} \boldsymbol{\ell}, \quad (11)$$

with the Kalman gain matrix $\mathbf{G}^{\pi}$ defined as

$$\mathbf{G}^{\pi} = \begin{bmatrix} \mathbf{K}_{ff} \\ \mathbf{K}_{zf}^{\pi} \end{bmatrix} \mathbf{K}_{yy}^{-1}.$$

Now imagine that the Q function value at a general query point $f_q^{\pi} = Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k^q)$ is also updated in (9). Such value may be queried by a numerical method trying to improve the current policy. This estimate may now be obtained reusing the above estimates $\hat{\mathbf{f}}^{\pi}, \hat{\mathbf{z}}^{\pi}$ as well as $\mathbf{K}_{yy}^{-1}$ and recalculating only a row kernel matrix $\mathbf{K}_{qf}^{\pi}$ which is the only datum changed by the query point. The result is

$$\hat{f}_q^{\pi} = \mathbf{K}_{qf}^{\pi} \mathbf{K}_{yy}^{-1} \left(\frac{1-\xi}{\xi} \left(\hat{\mathbf{f}}^{\pi} - \gamma \hat{\mathbf{z}}^{\pi} \right) - \frac{1}{\xi} \boldsymbol{\ell} \right). \quad (12)$$

Without proofs, we state several statistical properties of the estimates (11). They are unbiased (under GPR assumptions) except of the bias towards zero caused by $\xi > 0$. However, they are not optimal because $\mathbf{y} - \mathbf{f}$ are in general not normal, independent and homeoskedastic. Also, it should be noted that the uncertainty of this estimate cannot be calculated by (3), but must be estimated in a different way.

The flow of calculations is in Algorithm 1. In the first step, it calculates the kernel matrix $\mathbf{K}_{ff}$ and matrix inversion $\mathbf{K}_{yy}^{-1}$ for later usage as it depends only on the data, not the optimized policy. Then data matrix $\mathbf{X}_p^{\pi^{(i)}}$ and kernel matrix $\mathbf{K}_{zf}^{\pi^{(i)}}$ have to be updated according to actual policy $\pi^{(i)}$, starting with some known stabilizing policy $\pi^{(1)}$. $Q^{\pi^{(i)}}$ estimate is also calculated for each policy $\pi^{(i)}$ by solving a linear system of equations (11). All available data can be used in this step. This unbiased estimate of $Q^{\pi^{(i)}}$ is then minimized at each state $\mathbf{x}_j$ using actions $\mathbf{u}_j$ at step (2c) in order to define a new improved policy $\pi^{(i+1)}$, where $\mathbf{x}_j, \mathbf{u}_j$ are predefined according to process/system. The minimization is iterative, evaluating the Q function using (12), which is an inner product. The stop condition is based on either a number of iterations limit or vanishing difference between the Q function values evaluated at last two policies $\pi^{(i)}$ and $\pi^{(i-1)}$. Finally, the GP defines the optimal control action at any process state implicitly by (6).

Require: $\mathbf{X}, \mathbf{X}_p^{\pi^{(1)}}, \boldsymbol{\ell}, \epsilon, \gamma, \xi, \pi^{(1)}, \mathbf{x}_j, \mathbf{u}_j$
(1) calculate kernel $\mathbf{K}_{ff}$ and kernel inversion $\mathbf{K}_{yy}^{-1}$
(2) $i = 1$, **repeat**
  (a) update $\mathbf{X}_p^{\pi^{(i)}}$ and kernel $\mathbf{K}_{zf}^{\pi^{(i)}}$ according to new $\pi^{(i)}$
  (b) calculate $\hat{\mathbf{f}}^{(i)}, \hat{\mathbf{z}}^{(i)}$ using (11)
  (c) improve policy $\pi^{(i+1)}(\mathbf{x}_j) \leftarrow \arg\min_{\mathbf{u}_j} Q^{\pi^{(i)}}(\mathbf{x}_j, \mathbf{u}_j)$ using (12)
  (d) $i = i + 1$
(3) **until** $\max \left| \hat{\mathbf{f}}^{(i-1)} - \hat{\mathbf{f}}^{(i)} \right| < \epsilon$

Algorithm 1. Q-learning with GPR

4. FAN COIL UNIT

This section introduces the simplified FCU model used for testing the algorithm from previous section. FCU is a common air conditioning system, which is inherently nonlinear (Arguello-Serrano and Velez-Reyes, 1999). Usually installed in building interiors, it consists of a speed controllable electrical fan, a copper coil flown with heating and/or cooling liquid (a heat exchanger), and an air supply. It mixes the recirculated interior air with primary (outdoor) air. This air mixture is then heated/cooled according to the air temperature setpoint error by flowing through the coil. Then such air is supplied into the interior and mixed. The goal is to achieve the temperature set-point maintaining the interior CO_2 fraction and relative humidity at acceptable limits. Except of the obvious air heating and cooling effect, the heat supplied to or removed from the air can also be related to water evaporation or condensation in the unit. It thus makes a difference whether a FCU changes temperature of more air by less or vice versa. A model based optimal controller cannot be supplied by the unit manufacturer because the process model involves model of the interior, including its volume, thermal capacities, thermal insulation, solar and thermal load predictions, typical CO_2 and humidity loads. That is why model-free control or ML techniques may come into account. If such controllers could periodically re-optimize their behavior using ML techniques, significant amounts of energy could be saved world wide.

4.1 Model

Only the room air temperature T_z [°C] state is taken into consideration for purpose of this paper. Considering the

perfect air mixing in the interior, it is described by the differential equation

$$\dot{T}_z(t) = \frac{f(t)}{V}\left(T_s(t) - T_z(t)\right) + \frac{q_L(t)}{c_p V \rho},$$

where $T_s(t)$ [°C] is supply air temperature, $q_L(t)$ [W] is net heat load/loss, $f(t)$ [m^3/s] is air flow, V [m^3] is volume of the interior, ρ [kg/m^3] is air density and c_p[J/kg K] is air spec. thermal capacity. Let us define the volume independent control action as $u(t) = \tau f(t)/V$, i.e. the relative fraction of the air replaced per time unit τ (e.g. one hour). The supply air temperature $T_s(t)$ is a nonlinear function of air flow rate $u(t)$. The nonlinearity of $T_s(t)$ for purpose of this paper was approximated by the rational function

$$T_s(t) = \frac{u(t)T_z(t) + eT_0}{u(t) + e}, \tag{13}$$

where e is a heat exchanger size factor and T_0 [°C] is the maximum supply air temperature. The maximum supply air temperature decreases from T_0 (considered 40°C) to T_z asymptotically when the air flow increases. For simplicity, we neglected the primary air. The heat losses were considered as $\tau q_{L_0}/c_p V \rho = -7$°C, i.e. the room temperature would drop by this amount per unit time if the air-conditioning would be stopped.

5. RESULTS

This section presents the results. Model from Section 4 was considered in discrete time form obtained by the Euler method with the sampling rate $\tau/200$. The GPI algorithm was applied in order to find the optimal control policy. Loss function ℓ was defined as

$$\ell(x_k, u_k) = (T_k - T_{sp})^2 + u_k^4, \tag{14}$$

where setpoint temperature was $T_{sp} = 22$°C . GPR used the product of a polynomial kernel (degree two) and SE kernel as the kernel function. Recall that product of kernels is again a kernel. This choice is based on the fact known from linear control theory (Kwakernaak, 1972): a linear model with quadratic loss has quadratic Q. Hence, our choice defines a locally linear control law.

Training dataset consists of $2,000$ points ~ 10 hours. T_{z_k} is the only state x_k of the process. See Fig. (2). The data used for learning were generated by simulation. The control u_k was selected as random bounded input.

Q function was calculated by the proposed method and optimal control policy π^* was found after several policy iterations (around five suffice). Values $\gamma = 0.999$ and $\xi = 10^{-5}$ were used in the algorithm. It is important that u_k^π must excite the process to cover the state-action space. The control actions must be partly randomized to get a valid training dataset. This is related to the well known problem of exploration-exploitation trade-off. Fig. (3) shows the trajectory of optimal policy as a curve connecting the minima of Q function with respect to u.

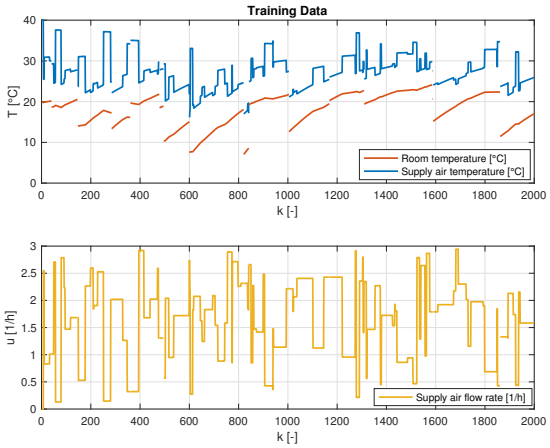

Fig. 2. Q-learning training data consists of $2,000$ data points (~ 10 hours) divided into 15 continuous time intervals during which the room was heated from a cold condition. The room temperature $T_{z_k} \sim x_k$, supply air flow rate u_k, and also supply air temperature T_s calculated from (13) are shown.

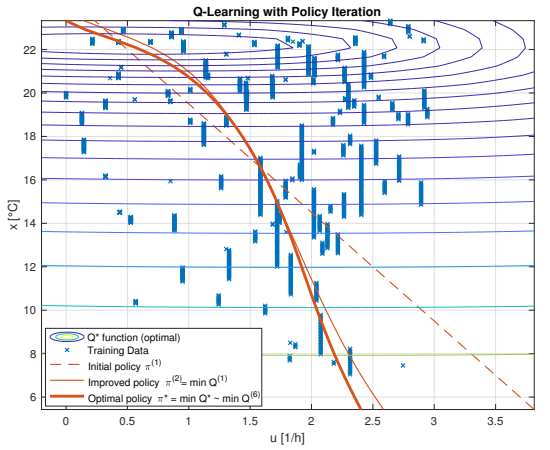

Fig. 3. Q-learning with GPR and GPI. Q^* contours and policies $\pi^{(1)}, \pi^{(2)}$ and π^* (highlighted) calculated from (6).

This Q learning result makes good sense intuitively. The air exchange rate u_k equals the heat losses when the room temperature is at the set point. Then it increases if the temperature is lower in order to heat the interior. Therefore, there is a negative feedback as expected. However, this feedback gain becomes smaller when the control error is greater because the large air flows are less effective for heating due to limited heat exchanger effectiveness and the decreasing supply air temperature. Instead, the electrical fan noise and the air flow would just annoy the occupants. Also, the fan would use more electricity. All this is modeled by the second term in ℓ. As such, the policy resembles a proportional feedback controller with variable gain. It does not have any integral action whereas a proportional integral derivative (PID) controller would be normally used for similar purpose. However, it can be shown that the integral action can be added by augmenting the state space with temperature time difference and considering the time

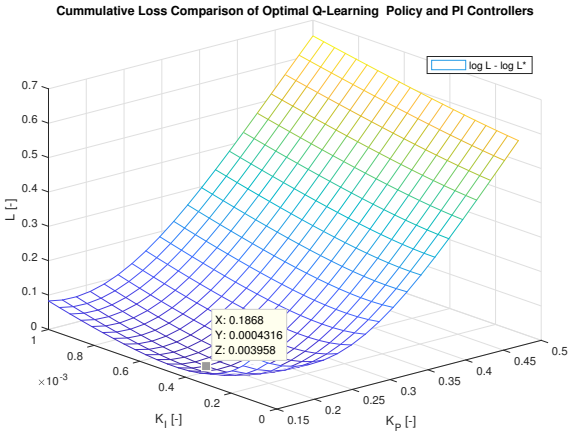

Fig. 4. Comparison of PI controllers cumulative losses L with optimal policy cumulative loss $L*$.

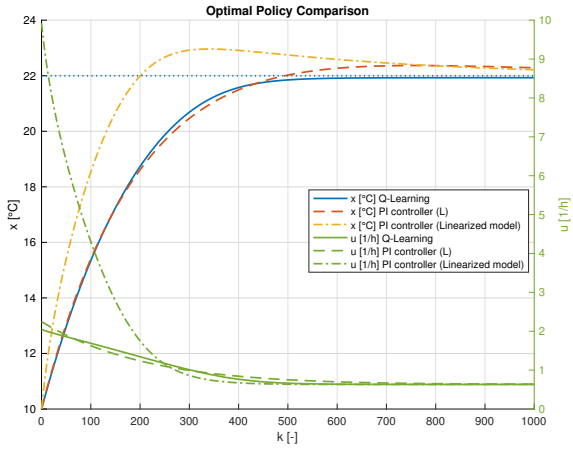

Fig. 5. Comparison of Q-learning optimal control policy, PI controller designed by cumulative loss comparison and PI controller designed from linearized model of FCU. The results are presented on nonlinear FCU model.

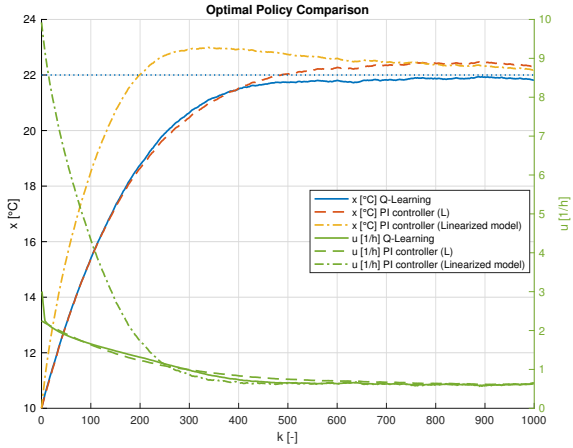

Fig. 6. Comparison of Q-learning optimal control policy, PI controller designed by cumulative loss comparison and PI controller designed from linearized model of FCU. The results are presented on nonlinear FCU model with noisy net heat load/heat loss q_L.

difference $u_k - u_{k-1}$ as the control action. Recall that the integral action is important in order to reject unmeasured slow disturbances.

5.1 Q-learning Evaluation

The results from Q-learning were compared to multiple PI controllers in terms of the cumulative loss function L values. This function represents the sum of all losses during an episode, i.e. $L = \sum_k \ell_k$. An assuredly optimal value of cumulative loss L^* is given by the optimal policy π^* from Q learning. A grid of proportional and integral PI constants was considered so that it obviously contained the optimal PI values (local minimum). All L values were calculated using (14), starting at the same initial condition 10°C over next 1,000 sampling periods considering $T_{sp} = 22$°C, so the results are comparable. These values were compared in Fig. (4). The best PI controller parameters from the grid were selected for Fig. (5). A PI controller designed for a linearized model of FCU at $x_0 = 22\,[°\mathrm{C}], u_0 = 1\,[1/\mathrm{h}]$ is also visualized. It can be observed that the best PI almost matches the result of the Q learning whereas the model linearization based result is different.

Next, the controllers and the Q learning were tested considering the net heat load/heat loss q_L not constant but uniformly distributed over $[-9, -5]$. The result is shown in Fig. (6). Note that Q-learning designed controller is robust towards such a noise. It should be noted that the same noise was used to generate the learning data for this test, not only when simulating the controller.

6. CONCLUSIONS

This paper described a practical approach of using unbiased GPR based Q-learning algorithm to find a control law for a completely unknown nonlinear process based on a historical dataset of a medium size $\sim 10^3$. Engineers face such problem often and a solution is of practical interest. An efficient GPR approach was used to calculate an unbiased Q function value estimate in any point in the state-action space. The generalized policy iterations were used to optimize the controller. This method typically converges rapidly. The optimal control law was then fully defined by the minima of a GP. This represents a numerical optimization in a low dimensional space of controller outputs and is thus numerically trackable. Although not explained in this paper, GP can be globally minimized even if it is not convex, see (Franey et al., 2011). An unbiased Q estimate makes the method insensitive to noise affecting the process. Presented non-parametric GPR approach is consistent with the Least-Squares Temporal Difference Learning (LSTD) (Bradtke and Barto, 1996), which is an unbiased parametric estimation algorithm. The approach can be integrated with the GPR sparse form in order to lower the dimensionality. However, details of this reduction are currently a subject of research. The approach of unbiased estimation was verified on a simple linear model against Q function calculated as a solution of Riccati equation, this verification is not part of the paper in detail. It was then tested on a simplified nonlinear one-input one-state FCU simulation model and an optimal

control policy π^* was calculated. The result makes sense intuitively, the feedback gain is gradually decreasing with the control error. A grid of PI controllers were compared in terms of cumulative loss L towards L^*, found by the Q-learning. The best PI controller from the grid was slightly worse than L^*. However, such direct controller search is impractical without a simulation model because it requires evaluating many controllers from defined initial conditions and affected by defined disturbances. Also, a PI controller was designed based on a linearized FCU model and PI tuning rules. This traditional approach could actually be used in practice together with, for example, Ziegler Nichols PID calibration method. The controllers were compared in Section 5 and the performance of PI designed using linearized FCU was shown to be worse than both Q-learning and the PI found by direct search. Here the problem may be both linearization point and PID tuning rule which does not minimize the loss (14). Although the example was a single input single output control problem, the method applies to multidimensional problems without modifications. The whole process was described for reader's understanding on the high level. Many technical details were mentioned just briefly. However, the method is simple and straightforward.

The main pitfalls of the process may be also pointed out. It is necessary to choose a kernel function and several hyperparameters. However, these choices affect the accuracy, the method should still converges to the same Q function asymptotically with growing dataset. The optimization of hyperparameters in connection with the proposed approach is our current research topic. Although result with only one kernel (SE times quadratic) was presented, different kernels were also tried with similar results, the hyperparameters were chosen reasonably and the kernels were smooth. The method assumes the process state is measured without error. This is not realistic in most control applications. The state may be often approximated by measurements and lagged measurements and control actions. Although the Q-learning seems to work well if the approximation is reasonable, an optimal state approximation is currently investigated in terms of dimension versus accuracy trade-off. The training dataset must provide exploration and cannot be obtained by running a fixed controller. A control action randomization is necessary. A stabilizing initial controller is required. This does not seem to be a serious constraint in many practical applications such as building control.

Overall, the method gives reasonably consistent results. Also note that theoretical proofs for declared statements regarding the used method are work in progress.

REFERENCES

A Fuller, W. (2019). Measurement error models / wyne a. fuller. *SERBIULA (sistema Librum 2.0)*.

Arguello-Serrano, B. and Velez-Reyes, M. (1999). Nonlinear control of a heating, ventilating, and air conditioning system with thermal load estimation. *IEEE Transactions on Control Systems Technology*, 7, 56 – 63.

Bijl, H., van Wingerden, J.W., and T. B. Schˆn, M.V. (2015). Online sparse gaussian process regression using fitc and pitc approximations. *IFAC-PapersOnline - 17th IFAC Symposium on System Identification. ed. / Y Zhao; E-W Bai; J-F Zhang. Laxenburg, Austria : IFAC*, 48, 703–708.

Bradtke, S.J. and Barto, A.G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1), 33 – 57.

Chowdhary, G., Liu, M., Grande, R., Walsh, T., How, J., and Carin, L. (2014). Off-policy reinforcement learning with gaussian processes. *IEEE/CAA Journal of Automatica Sinica*, 1(3), 227–238. doi: 10.1109/JAS.2014.7004680.

Csato, L. and Opper, M. (2002). Sparse on-line gaussian processes. *Neural computation*, 14, 641–68. doi: 10.1162/089976602317250933.

Engel, Y., Mannor, S., and Meir, R. (2003). Bayes meets bellman: The gaussian process approach to temporal difference learning. volume 1, 154–161.

Engel, Y., Mannor, S., and Meir, R. (2005). Reinforcement learning with gaussian processes. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, 201–208. ACM, New York, NY, USA. doi:10.1145/1102351.1102377. URL `http://doi.acm.org/10.1145/1102351.1102377`.

Franey, M., Ranjan, P., and Chipman, H. (2011). Branch and bound algorithms for maximizing expected improvement functions. *Journal of Statistical Planning and Inference*, 141, 42 – 55.

Gaskett, C., Wettergreen, D., and Zelinsky, A. (1999). Q-learning in continuous state and action spaces. In *Australian Joint Conference on Artificial Intelligence*, 417–428.

Jung, T. and Stone, P. (2010). Gaussian processes for sample efficient reinforcement learning with rmax-like exploration. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, ECML PKDD'10, 601–616. Springer-Verlag, Berlin, Heidelberg.

Kwakernaak, H. (1972). *Linear Optimal Control Systems*. John Wiley & Sons, Inc., New York, NY, USA.

Rasmussen, C.E. and Williams, K.I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, London, England.

Rasmussen, C. and Kuss, M. (2004). Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*, 751–759. Max-Planck-Gesellschaft, MIT Press, Cambridge, MA, USA.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, London, England.

van Hasselt, H. and Wiering, M.A. (2007). Reinforcement learning in continuous action spaces. *IEEE International Symposium on Adaptive Dynamic Programming and Rein-forcement Learning (ADPRL-07)*, 272–279.

Williams, C.K.I. (1999). *Prediction with Gaussian processes: from linear regression to linear prediction and beyond*, 599–621. MIT.

Young, P. (1984). *Recursive estimation and time-series analysis. An introduction*. doi:10.1007/978-3-642-82336-7.