Gaussian Process Based Model-less Control with Q-Learning*

Jan Hauser,[1] Daniel Pachner[2] and Vladimír Havlena[1]

***Abstract*— The aim of this paper is to demonstrate how Machine Learning (ML) based on Gaussian Process Regression (GPR) 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.**

I. INTRODUCTION

Model-less 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 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 [10], 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 [11], [6]. If the analytical form of the Q function is unknown in continuous state spaces, it may be represented by a universal function approximating method such as Neural Network or Gaussian Process Regression (GPR).

*This work was supported by ...?

[1]Jan Hauser and Vladimír Havlena are with Department of Control Engineering, Faculty of Electrical Engineering of Czech Technical University in Prague, Technicka 2, 166 27 Praha 6, Czech Republic. Email: {hauseja3,havlena}@fel.cvut.cz

[2]Daniel Pachner is with Honeywell ACS Global Laboratory Prague, V Parku 2326/18, 148 00 Prague, Czech Republic Email: daniel.pachner@honeywell.com

GPR is a non-parametric regression technique [9] which is able to approximate any continuous target function uniformly. Unfortunately, the runtime computational requirement is $\mathcal{O}(n^3)$ and the memory requirement is $\mathcal{O}(n^2)$ for n data points. Various sparse GPR techniques were developed to overcome this computational complexity [2], [4], [7]. One of the conventional GPR sparse method is to use a set of size m induction input points which reduces the computational complexity to $\mathcal{O}(nm^2)$ (runtime) and $\mathcal{O}(nm)$ (memory). These induction points reduce the whole dataset into a smaller number of representing data points which are distributed across the data space to preserve as much information as possible. GP uses various covariance (kernel) functions [12] 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. ML algorithms are normally supposed to process a large dataset (of size $\sim 10^5$ and more) to yield sufficiently accurate results. 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 [1]. 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 II, the GPR algorithm is explained. Section III describes the Q-Learning mechanism and how it connects to GPR. In Section IV, a reduced FCU model is explained. Next Section V presents the results of these techniques and Section VI concludes and proposes a future work.

II. GAUSSIAN PROCESS

In this section, GPR method is briefly described. The GPR technique is then used with Q-Learning algorithm in next section.

A. Gaussian Process Regression

GPR is a supervised learning regression model, which can be also described as a distribution over functions [9]. 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 $\mathbf{X}_*$, then GPR can predict $f(\mathbf{x}_{j*})$, where $\mathbf{x}_{j*} \in \mathbf{X}_*$, 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 $\mathbf{y}_i = f(\mathbf{x}_i) + \boldsymbol{\varepsilon}_i$, where $\boldsymbol{\varepsilon}$ is a vector of independent identically distributed (i.i.d) Gaussian noise with variance σ_n^2. The prior covariance of the noisy values is defined as

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

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

$$\begin{bmatrix} \mathbf{y} \\ f(\mathbf{X}_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \kappa(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & \kappa(\mathbf{X}, \mathbf{X}_*) \\ \kappa(\mathbf{X}_*, \mathbf{X}) & \kappa(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{m} \\ \mathbf{m}_* \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_n^2 \mathbf{I} & \mathbf{K}_{f*} \\ \mathbf{K}_{*f} & \mathbf{K}_{**} \end{bmatrix}\right),$$

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

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\left|\mathbf{x} - \mathbf{x}'\right|^2\right) + \sigma_n^2 \mathbf{I},$$

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, length-scale l^2 and noise variance σ_n^2 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 $\mathbf{f}_*$ (shortened expression of $f(\mathbf{X}_*)$) can be defined. The predictive GPR relationships are following

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{y}) &= \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*), \\ \boldsymbol{\mu}_* &= \mathbb{E}\left[\mathbf{f}_*|\mathbf{y}\right] = \mathbf{m}_* + \mathbf{K}_{*f}\overline{\mathbf{K}}_{ff}^{-1}(\mathbf{y} - \mathbf{m}), \quad (1) \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_{*f}\overline{\mathbf{K}}_{ff}^{-1}\mathbf{K}_{f*}, \quad (2) \end{aligned}$$

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

There is an important relationship between the Kalman filter equations and (1, 2) which will be used later. Specifically, we remind that the term $\mathbf{G} = \mathbf{K}_{*f}\overline{\mathbf{K}}_{ff}^{-1}$ 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) $\mathbf{f}_*$ for any test points over a dataset $\mathbf{X}_*$. Unfortunately, these simple calculations can get very expensive due to the inverse of matrix $\overline{\mathbf{K}}_{ff}$, which is of size $n \times n$ where n is the number of training data points. This is the operation which costs $\mathcal{O}(n^3)$. This is the moment when sparse GPR is taken into account [2].

III. Q-LEARNING

This section describes the basic principles of Q-Learning algorithm [10] which is a model-less reinforcement learning approach. Then the policy iteration algorithm based on Bellman equation is introduced.

For purpose of this section, let's highlight the analogies and slight differences between two closely related fields: Control Theory (CT), and MDP respectively. 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 CT 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 $p(\mathbf{s}_{k+1}|\mathbf{a}_k, \mathbf{s}_k)$ of MDP. Control law, or a state feedback $\mathbf{u}_k = C(\mathbf{x}_k)$ in CT is an analogy of a deterministic policy $\mathbf{a}_k = \pi(\mathbf{s}_k)$. A stochastic policy, usually not used in CT, defines the joint p.d.f. $\pi(\mathbf{a}_k, \mathbf{s}_k)$ instead of an explicit function. An important difference exists between the reward $r(\mathbf{a}_k, \mathbf{s}_k, \mathbf{s}_{k+1})$ used in MDP (bounded, to be maximized) and loss function $\ell(\mathbf{x}_k, \mathbf{u}_k)$ used in CT (not bounded in general, to be minimized, almost never depending on $\mathbf{x}_{k+1}$). The ML theory will be discussed below mostly with CT notations and assumptions.

A. Q Function

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

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

It is possible to talk either about Q function $Q^\pi(\mathbf{u}, \mathbf{x})$ 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 $\ell(\mathbf{u}_k, \mathbf{x}_k)$ at time $k \in \mathbb{N}$, the Q is defined

$$Q^\pi(\mathbf{u}_k, \mathbf{x}_k) = l(\mathbf{u}_k, \mathbf{x}_k) + \mathbb{E}\left[\sum_{i=1}^{\infty} \gamma^i \ell(\mathbf{u}^\pi_{k+i}, \mathbf{x}_{k+i}) \middle| \mathbf{u}_k, \mathbf{x}_k \right], \quad (3)$$

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

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

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

$$\pi^*(\mathbf{x}) = \arg\min_{\mathbf{u}} Q^*(\mathbf{u}, \mathbf{x}). \quad (5)$$

The Bellman equation (more usually expressed in terms of V) provides the recursive approach for finding the functions Q^π and Q^*. It follows directly from (3,4). For an optimal policy π^*, function $Q^*(x_k, u_k)$ must satisfy

$$Q^*(\mathbf{u}_k, \mathbf{x}_k) = \ell(\mathbf{u}_k, \mathbf{x}_k) + \gamma \mathbb{E}\left[\min_{\mathbf{u}_{k+1}} Q^*(\mathbf{u}_{k+1}, \mathbf{x}_{k+1}) \middle| \mathbf{u}_k, \mathbf{x}_k \right], \quad (6)$$

and for general policy π analogously by using (3).

B. Policy Iteration

Policy iteration algorithm calculates Q function from Bellman equation (6) for a 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.

C. Q-Learning with Gaussian Process Regression

Last step in this section is to introduce Q-Learning algorithm with GPR. For a finite (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. An example of such approach is the commonly used *state action reward state action* or SARSA [10].

This paper considers GPR as a continuous Q^π function approximation method and then optimizes the control action using (III-B) 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

$$\mathbf{X}^\pi_* = \begin{bmatrix} \mathbf{u}^\pi_2, \mathbf{x}_2 \\ \vdots \\ \mathbf{u}^\pi_k, \mathbf{x}^\pi_k \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{u}_1, \mathbf{x}_1 \\ \vdots \\ \mathbf{u}_{k-1}, \mathbf{x}_{k-1} \end{bmatrix}.$$

Here $\mathbf{X}$ is a collection of state-action pairs visited by the process whereas $\mathbf{X}^\pi_*$ is a collection of states observed as results of the actions accompanied with the actions the evaluated strategy π would presumably apply there. Note $\mathbf{X}^\pi_*$ 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

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

and let the Q function be a GP with known kernel. The notation $\mathbf{f}$ for the vector of unknown function values used in GPR context will be preserved

$$\mathbf{f}^\pi_* = Q^\pi(\mathbf{X}^\pi_*), \quad \mathbf{f}^\pi = Q^\pi(\mathbf{X}).$$

The noisy realization of $\mathbf{f}^\pi = \boldsymbol{\ell} + \gamma \mathbb{E}[Q^\pi(\mathbf{X}^\pi_*)]$ is $\mathbf{y}^\pi = \boldsymbol{\ell} + \gamma \mathbf{f}^\pi_*$. Then based on (1) and (6), the conditional means are

$$\mathbb{E}\left[\begin{matrix} \mathbf{f}^\pi \\ \mathbf{f}^\pi_* \end{matrix} \middle| \mathbf{y}^\pi \right] = \begin{bmatrix} \mathbf{m} \\ \mathbf{m}_* \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ff} \\ \mathbf{K}^\pi_{*f} \end{bmatrix} \overline{\mathbf{K}}^{-1}_{ff} \left(\boldsymbol{\ell} + \gamma \mathbf{f}^\pi_* - \mathbf{m} \right). \quad (7)$$

The expression (7) may seem useless because the Q estimates depend on $\mathbf{f}^\pi_*$ value which we do not know. Recall that Q function is neither measured nor observed. However, consider the left hand side equals the true values $\mathbf{f}^\pi$ and $\mathbf{f}^\pi_*$ for $k \to \infty$ and sufficient excitation in the state-action space, and when $\mathbf{m} = \mathbf{f}^\pi$ and $\mathbf{m}_* = \mathbf{f}^\pi_*$. The $\mathbf{f}^\pi$ and $\mathbf{f}^\pi_*$ thus represent a fixed point of the following iterations (index π dropped)

$$\begin{bmatrix} \mathbf{f}^{(i+1)} \\ \mathbf{f}^{(i+1)}_* \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{(i)} \\ \mathbf{f}^{(i)}_* \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ff} \\ \mathbf{K}_{*f} \end{bmatrix} \overline{\mathbf{K}}^{-1}_{ff} \left(\boldsymbol{\ell} + \gamma \mathbf{f}^{(i)}_* - \mathbf{f}^{(i)} \right). \quad (8)$$

An estimate may be calculated starting the iterations from $\mathbf{m}, \mathbf{m}_*$. Instead of actually iterating (8), 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 (7) 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 observalbe [8] and some information from the starting values $\mathbf{m}, \mathbf{m}_*$ 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. slightly to the left by $\xi \geq 0$. 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 (8) by the factor of $1-\xi$. It results in the following estimates

$$\begin{bmatrix} \hat{\mathbf{f}}^\pi \\ \hat{\mathbf{f}}^\pi_* \end{bmatrix} = \left((1-\xi)\begin{bmatrix} \mathbf{G}^\pi, & -\gamma\mathbf{G}^\pi \end{bmatrix} - \xi\mathbf{I}\right)^{-1}\mathbf{G}^\pi\ell, \quad (9)$$

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

$$\mathbf{G}^\pi = \begin{bmatrix} \mathbf{K}_{ff} \\ \mathbf{K}^\pi_{*f} \end{bmatrix}\overline{\mathbf{K}}^{-1}_{ff}.$$

Now imagine that the Q function value at a general query point $f^\pi_q = Q^\pi(\mathbf{u}^q_k, \mathbf{x}_k)$ is also updated in (7). 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 $\hat{\mathbf{f}}^\pi, \hat{\mathbf{f}}^\pi_*$ as well as $\overline{\mathbf{K}}^{-1}_{ff}$ and recalculating only a row kernel matrix $\mathbf{K}^\pi_{qf}$ which is the only datum changed by the query point. The result is

$$\hat{f}^\pi_q = \mathbf{K}^\pi_{qf}\overline{\mathbf{K}}^{-1}_{ff}\left(\frac{1-\xi}{\xi}\left(\hat{\mathbf{f}^\pi} - \gamma\hat{\mathbf{f}^\pi_*}\right) - \frac{1}{\xi}\boldsymbol{\ell}\right). \quad (10)$$

Without proofs we state several statistical properties of the estimates (9). 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 (2) but must be estimated in a different way.

The flow of calculations is in Algorithm 1. In the first step, it calculates the matrix inversion $\overline{\mathbf{K}}^{-1}_{ff}$ for later uses as it depends only on the data, not the optimized policy. $Q^{\pi^{(i)}}$ estimate is calculated for each policy $\pi^{(i)}$, starting with some known stabilizing policy $\pi^{(1)}$, solving a linear system of equations (9). All available data can be used in this step. This $Q^{\pi^{(i)}}$ is then minimized at each state from $\mathbf{X}^\pi_*$ at 2c to define a new improved policy $\pi^{(i+1)}$. The minimization is iterative evaluating the Q function using (10) 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 (5).

Algorithm 1 Q-Learning with GPR

Require: $\kappa([\mathbf{u},\mathbf{x}],[\mathbf{u}',\mathbf{x}']), \epsilon, \gamma, \xi, \mathbf{x}_j, \mathbf{u}_j, \boldsymbol{\ell}_j, \pi^{(1)}$
$1 \leq j \leq k$

1) calculate kernel and its inversion $\overline{\mathbf{K}}^{-1}_{ff}$
2) $i = 1$, **repeat**
   a) update $\mathbf{X}^{\pi^{(i)}}_*$ and kernel $\mathbf{K}^{(i)}_{*f}$
   b) calculate $\hat{\mathbf{f}}^{(i)}, \hat{\mathbf{f}}^{(i)}_*$ using (9)
   c) $\pi^{(i+1)}(\mathbf{x}_j) \leftarrow \arg\min_{\mathbf{u}_j} Q^{\pi^{(i)}}(\mathbf{u}_j, \mathbf{x}_j)$
   d) $i = i + 1$
3) **until** $\max\left|\hat{\mathbf{f}}^{(i-1)} - \hat{\mathbf{f}}^{(i)}\right| < \epsilon$

IV. 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 non-linear [1]. 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-less control or ML techniques may come into consideration. If such controllers could periodically re-optimize their behavior using ML techniques, significant amounts of energy could be saved world wide.

A. 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[$\mathrm{J}/\mathrm{kg\,K}$] 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

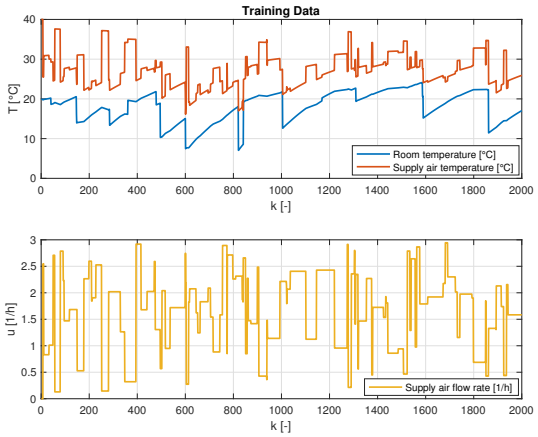

Fig. 1. Q-Learning training data consists of $2,000$ data points ~ 10 hours. The room temperature $T_{z_k} \sim x_k$, supply air flow rate u_k, and also supply air temperature T_s calculated from (11) shown.

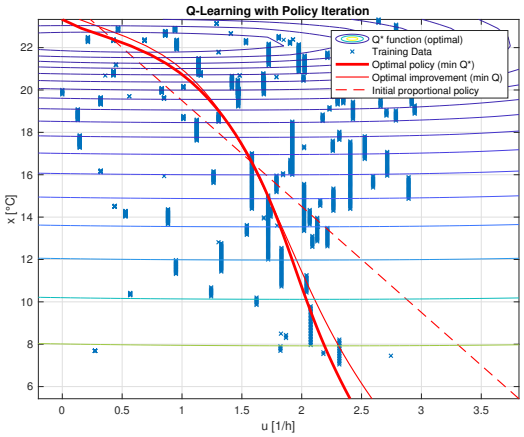

Fig. 2. Q-Learning with GPR and Policy Iteration. Q^* contours and policies $\pi^{(2)}$ and π^* (highlighted) calculated from (5).

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

$$T_s(t) = \frac{u(t)T_z(t) + eT_0}{u(k) + e}, \tag{11}$$

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.

V. RESULTS

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

$$\ell(u_k, x_k) = (T_k - T_{sp})^2 + u_k^4, \tag{12}$$

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 [8]: 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. (1). 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 valid a training dataset. This is related to the well known problem of exploration-exploitation trade-off. Fig. (2) shows the trajectory of optimal policy as a curve connecting the minima of Q function with respect to u.

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 augmenting the state space with temperature time difference and considering the time difference $u_k - u_{k-1}$ as the control action. Recall that the integral action is important in order to reject unmeasured slow disturbances.

A. 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

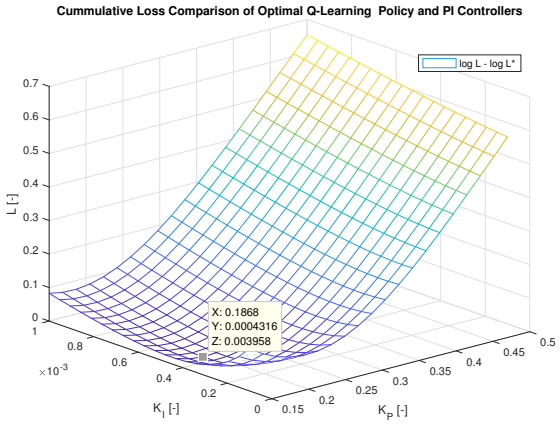

Fig. 3. Comparison of PI controllers cumulative losses L with optimal policy cumulative loss $L*$.

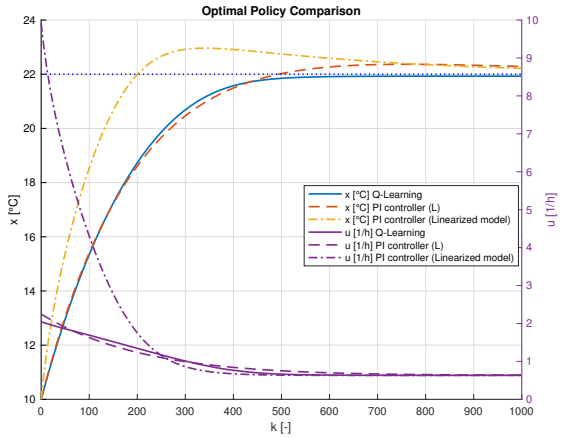

Fig. 4. Comparison of Q-Learning optimal control policy, PI controller designed by cumulative loss comparison and PI controller designed from linearized model of FCU. The results are presented on nonlinear FCU model.

using (12) 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. (3). The best PI controller parameters from the grid were selected for Fig. (4). A PI controller designed for a linearized model of FCU at $u_0 = 1\,[1/\mathrm{h}]\,, x_0 = 22\,[°\mathrm{C}]$ 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 $-7 \pm \Delta$. The result is shown in Fig. (5). 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.

VI. CONCLUSIONS

This paper described a practical approach of using GPR based Q-Learning algorithm to find a control law for a

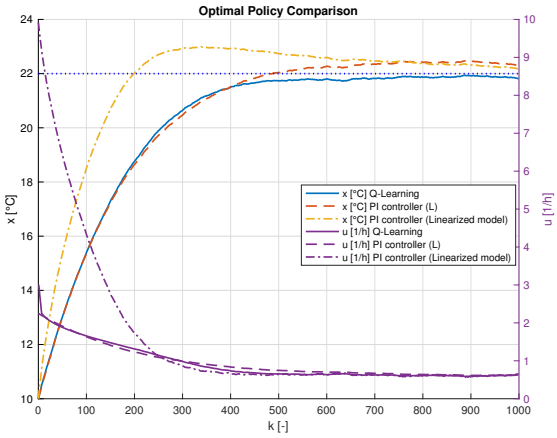

Fig. 5. Comparison of Q-Learning optimal control policy, PI controller designed by cumulative loss comparison and PI controller designed from linearized model of FCU. The results are presented on nonlinear FCU model with noisy net heat load/heat loss q_L.

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 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 tractable. Although not explained in this paper, GP can be globally minimized even if it is not convex, see [5]. An unbiased Q estimate makes the method insensitive to noise affecting the process. It is more accurate than just solving the so-called temporal difference equation in the least squares sense which results in a biased estimate in the presence of process noise [3]. The approach can be integrated with the GPR sparse form in order to lower the dimensionality. However, details of this reduction is currently a subject of research. The approach was verified on a simple linear model against Q function calculated as a solution of Riccati equation, tested on a simplified 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 V 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 the choice of the linearization point. 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 provided 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.

REFERENCES

[1] B. Argüello-Serrano and M. Vélez-Reyes. Nonlinear control of a heating, ventilating, and air conditioning system with thermal load estimation. *IEEE Transactions on Control Systems Technology*, 7:56 – 63, 1999.

[2] H. Bijl, J. W. van Wingerden, and M. Verhaegen T. B. Schön. 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, 2015.

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

[4] J. Q. Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

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

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

[7] M. F. Huber. Recursive gaussian process: On-line regression and learning. *Pattern Recognition Letters*, 45:85–91, 2014.

[8] H. Kwakernaak. *Linear Optimal Control Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1972.

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

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

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

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