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

Jan Hauser[1] and 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. Furthermore, sparse form of GPR is suggested and applied to reduce computational effort. 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. effect of the sparse GPR, 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 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 simple analytical control design method is available.

The Q-Learning is an off-policy machine learning (ML) iterative algorithm [**?**], which approximates certain function satisfying the Bellman equation. This Q function then defines a controller. To solve problems in continuous state spaces, Q function can be represented by a universal function approximating method such as Neural Network or Gaussian Process Regression (GPR). This approach is very closely related to Markov Decision Process (MDP) and Approximate Dynamic Programming (ADP) approaches.

GPR is a non-parametric regression technique [**?**] which is able to approximate any function using noisy data. It can be applied even if the analytical form of the function is unknown. 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 [**?**],[**?**],[*cite: Guber?*] techniques were developed to overcome this computational complexity. 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 [**?**] to define the data covariance matrices. The choice of the kernel can have significant impact on the accuracy of the regression result.

The contribution of this paper is the practical combination of Q-Learning and GPR resulting in unbiased estimate of Q function and control policy improvement. ML algorithms normally needs a large historical dataset (of size $\sim 10^5$ and more) to find sufficient result, here a way smaller dataset is necessary (of a size $\sim 10^3$). 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 control in terms of energy consumption and user comfort [**?**]. FCU model used here is highly simplified to make the result easier to interpret.

For reader's understanding, some essential theory is briefly introduced. In Section II, the GPR algorithm is explained. Moreover, its sparse approximation is suggested later in text. 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 [**?**]. It is

*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

the function value estimator for an unknown function $f(\mathbf{x})$ considering any dataset $(\mathbf{x}, \mathbf{y})$, which can be written as

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

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

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

where $\mathbf{x}$ is a vector representing the dataset $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$.

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 $\mathbf{y}_i = f(\mathbf{X}) + \varepsilon_i$, where ε 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

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

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

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

where $\mathbf{K}_{f*}$ is a $n \times n_*$ matrix of the covariances of all pairs of training and testing datasets, i.e. $\mathbf{K}_{ff}$, $\mathbf{K}_{**}$, $\mathbf{K}_{*f}$ 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 $k(\mathbf{x}, \mathbf{x}')$ called kernels, e.g. squared exponential (SE)

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

or polynomial kernel of d-degree

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

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

$$\begin{aligned} p(\mathbf{f}_* | \mathbf{y}) &= \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*), && (1) \\ \boldsymbol{\mu}_* &= \mathbb{E}\left[\mathbf{f}_* | \mathbf{y}\right] = \mathbf{m}_* + \mathbf{K}_{*f} \overline{\mathbf{K}}_{ff}^{-1}(\mathbf{y} - \mathbf{m}), && (2) \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_{*f} \overline{\mathbf{K}}_{ff}^{-1} \mathbf{K}_{f*}. && (3) \end{aligned}$$

There is an important relationship between the Kalman filter equations and (2, 3), this relationship 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 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 [**?**].

III. Q-LEARNING

This section describes the basic principles of Q-Learning algorithm [**?**] 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. The states x_k and inputs u_k are usually considered in CT for a process model, whereas MDP uses the Markov process states $s_k \in \mathbf{S}$ and the agent's actions $a_k \in \mathbf{A}$. Note that the sets $\mathbf{S}$ and $\mathbf{A}$ are usually some real vector spaces in control problems whereas they may often be finite sets in MDP. The process model itself is an analogy of probability transition matrix $p(s_{k+1}|a_k, s_k)$ of MDP. Control law, or the state feedback $u_k = C(x_k)$ in CT is an analogy of a deterministic policy $u_k = \pi(s_k)$. A stochastic policy defines the joint p.d.f. $\pi(a_k, s_k)$ instead of an explicit function. An important difference exists between the reward $r_k(a_k, s_k, s_{k+1})$ used in MDP (bounded, to be maximized) and loss function $\ell_k(x_k, u_k)$ used in CT (often not bounded, to be minimized, almost never depending on x_{k+1}). The ML theory will be discussed below with CT notation.

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 $Q^\pi(u_k, x_k)$ pertaining to a given policy π or the Q function $Q^*(x_k, u_k)$ achieved after convergence. Q (and Q^*) describes the expected total discounted loss $\ell(u_k, x_k)$ received by the controller starting from x_k with a control action u_k and following with the policy π (and optimal π^*) thereafter. Q^*, as function of u_k, is thus a measure of quality of selecting the control action u_k in a given state x_k. 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 DP. 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, the Q is defined

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

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

$$Q^*(u_k, x_k) = \\ l(u_k, x_k) + \mathbb{E}\left[\min_{u_{k+i}} \sum_{i=1}^{\infty} \gamma^i \ell(u_{k+i}, x_{k+i}) \middle| u_k, x_k \right]. \quad (5)$$

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

$$\pi^*(x_k) = \arg\min_{u_k} Q^*(u_k, x_k). \quad (6)$$

The Bellman optimality principle equation (more usually expressed in terms of V) provides the recursive approach for finding the Q function Q^*. It follows directly from (4, 5). For some policy π, function $Q^{\pi}(x_k, u_k)$ must satisfy

$$Q^{\pi}(u_k, x_k) = \\ \ell(u_k, x_k) + \gamma \mathbb{E}\left[Q^{\pi}(u^{\pi}_{k+1}, x_{k+1}) \middle| u_k, x_k \right], \quad (7)$$

and for optimal policy π^* analogously by using (5).

B. Policy Iteration

Policy iteration algorithm calculates Q function from Bellman equation for a current policy and then improves the current policy by seeking for minimum values of Q^{π}_k with respect to u_k for each x_k. Such minimizing u_k defines the new policy. This process is repeated until convergence. It converges to the Q^* and provides expected cumulative loss, which is finite for the initial π. It means the starting policy can be selected randomly but must be 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 finite (in number of states and actions) MDP, the Q-Learning algorithms approximate the Q function using the observed samples x_k, u_k, which were encountered during interaction with a system. Nice example of commonly used Q-Learning is state action reward state action or SARSA [?].

This paper considers GPR as Q function approximation and then optimize the control action using (III-B). Method based on GPR is proposed to calculate Q, because the action-state space is a vector space where f.e. SARSA cannot be used directly. Let us define the set of training and prediction points for the GPR as concatenations of points in the action-state space

$$\mathbf{X}_* = \begin{bmatrix} u^{\pi}_1, x_1 \\ \vdots \\ u^{\pi}_k, x^{\pi}_k \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} u_0, x_0 \\ \vdots \\ u_{k-1}, x_{k-1} \end{bmatrix}. \quad (8)$$

Also, the concatenation of the losses will be used

$$\ell = \begin{bmatrix} \ell_0 \\ \vdots \\ \ell_{k-1} \end{bmatrix}, \quad (9)$$

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

$$\mathbf{f}_* = Q^{\pi}(\mathbf{X}_*), \quad \mathbf{f} = Q^{\pi}(\mathbf{X}). \quad (10)$$

The noisy realization of $\mathbf{f} = \ell + \gamma \mathbb{E}[Q^{\pi}(\mathbf{X}_*)]$ is $\mathbf{y} = \ell + \gamma \mathbf{f}_*$. Then based on (2) and (7), the conditional means are

$$\mathbb{E}\left[\begin{matrix} \mathbf{f} \\ \mathbf{f}_* \end{matrix} \middle| \mathbf{y} \right] = \\ \begin{bmatrix} \mathbf{m} \\ \mathbf{m}_* \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ff} \\ \mathbf{K}_{*f} \end{bmatrix} \overline{\mathbf{K}}^{-1}_{ff} \left(\ell + \gamma \mathbf{f}_* - \mathbf{m} \right). \quad (11)$$

The expression (11) may seem useless because the Q estimates depend on $\mathbf{f}_*$ true 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 $\mathbf{f}$ and $\mathbf{f}_*$ for $k \to \infty$ and sufficient excitation in the action-state space, and when $\mathbf{m} = \mathbf{f}$ and $\mathbf{m}_* = \mathbf{f}_*$. From this one gets a system of linear equations the true values satisfy. This system of linear equations may have one or infinitely many solutions. One may now use the Kalman filter and GPR analogy to understand that the latter happens when some of the function values are not observalbe [?]. Although they may be observable in theory, the Kalman gain may be so small that the system would get ill-conditioned. We propose to regularize this situation shifting the unobservable poles of the filter from 1 to some stable real pole $(1 - \xi) > \gamma$, i.e. slightly to the left. Practically, this means that the unobservable Q function values will be estimated as zeros. With this regularization, we have the following estimates

$$\begin{bmatrix} \hat{\mathbf{f}} \\ \hat{\mathbf{f}}_* \end{bmatrix} = \left((1 - \xi) \begin{bmatrix} \mathbf{G} & -\gamma \mathbf{G} \end{bmatrix} - \xi \mathbf{I} \right)^{-1} \mathbf{G} \ell, \quad (12)$$

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

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

Without proofs we state several statistical properties of the estimates (12). They are unbiased (under GPR assumptions) except of the small bias towards zero caused by $\xi > 0$. However, they are not efficient unless $\mathbf{y} - \mathbf{f}$ would be independent and homeoskedastic. Also, it should be noted

Algorithm 1 Q-Learning with GPR

```
Require: $\mathbf{X}, \mathbf{X}_*, \pi^{(1)}$
  1) while stop condition
       a) $\pi^{(i)} \to Q^{\pi^{(i)}}(\mathbf{X}, \mathbf{X}_*)$
       b) $\pi^{(i+1)} \leftarrow \arg\min_{\mathbf{u}} Q^{\pi^{(i)}}$
       c) $i = i + 1$
  2) end
```

that the uncertainty of this estimate cannot be calculated by (3) but must be estimated in a different way.

Not only $\hat{\mathbf{f}}, \hat{\mathbf{f}}_*$ estimates should be considered for optimization purpose. Also some general points $\hat{\mathbf{f}}_{**}$ are used during seeking for minimum of Q function in order to find π^*. These points are not part of the dataset.

The final algorithm is described in (1). The first step (1a) is to find GPR estimate of $Q^{\pi^{(i)}}$ by using (12). Next step (1b) is to apply (III-B) to minimize found $Q^{\pi^{(i)}}$ and find new policy $\pi^{(i+1)}$. The stop condition is either a number of iterations or a difference between control policies $\pi^{(i)}$ and $\pi^{(i+1)}$.

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 [**?**]. 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. This control problem is significantly complex and non-linear. 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, 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, a significant amounts of energy could be saved world wide.

A. Model

Only the room air temperature T_z $[°\mathrm{C}]$ 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 $T_s(t)$ $[°\mathrm{C}]$ is supply air temperature, $q_L(t)$ $[W]$ is net heat load/loss, $f(t)$ $\left[\mathrm{m}^3/\mathrm{s}\right]$ is air flow, V $\left[\mathrm{m}^3\right]$ is volume of the interior, ρ $\left[\mathrm{kg/m}^3\right]$ is air density and $c_p[\mathrm{J/kg\,K}]$ 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 $u(t)$. The nonlinearity of $T_s(t)$ for purpose of this paper was approximated by the rational function

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

where e is a heat exchanger size factor and T_0 $[°\mathrm{C}]$ is the maximum supply air temperature. The nonlinearity models how the maximum supply air temperature asymptotically decrease from T_0 (considered 40°C) to T_z when the air flow increases. For simplicity, we neglected the primary air. The heat losses were considered as $\tau q_{L_0}/c_p V\rho = -7°\mathrm{C}$, 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 difference equation form using 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

$$l(x_k, u_k) = \left(T_k - T_{sp}\right)^2 + u_k^4, \qquad (14)$$

where setpoint temperature was $T_{sp} = 22°\mathrm{C}$. 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 [**?**]: 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). Value $\gamma = 1 - 1e^{-3}$ was used in Q-Leatning algorithm. Also note here, that it is very important to let u_k^π excite in order to let the algorithm explore more input-state space. This is 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,

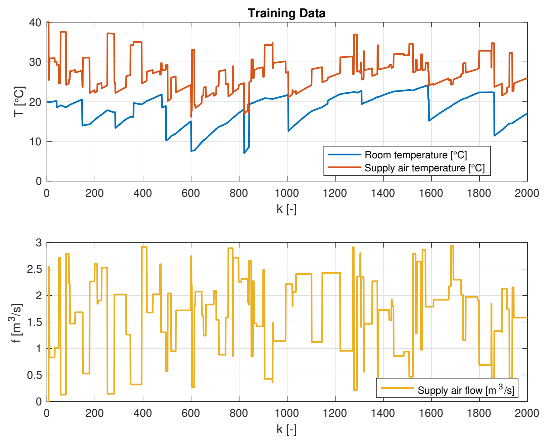

Fig. 1. Q-Learning training data consists of $2,000$ data points $\sim$ 10 hours. The room temperature $T_{z_k} \sim x_k$, supply air flow rate u_k, and also supply air temperature T_s calculated from (13) shown.

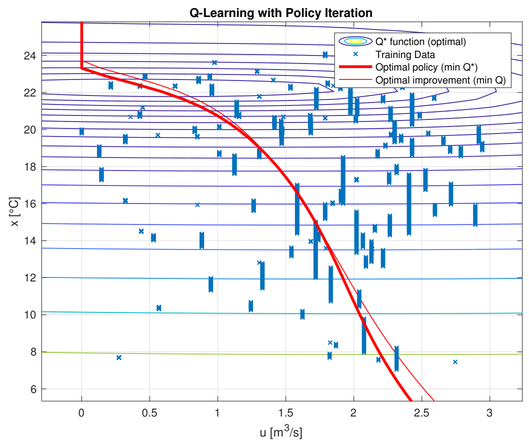

Fig. 2. Q-Learning with GPR and Policy Iteration. Q function Q^* is presented and optimal control policy π^* was calculated from (6) and its curve highlighted here.

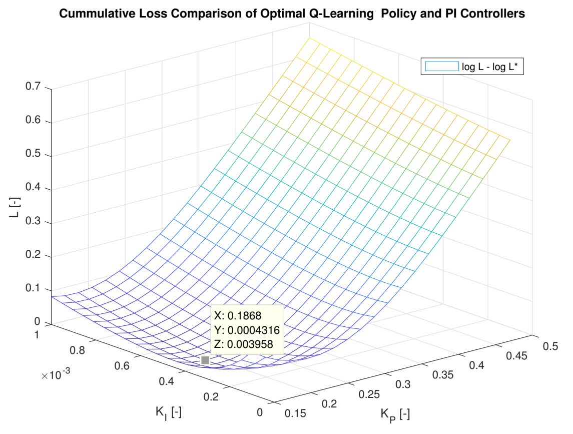

Fig. 3. Comparison of PI controllers cumulative losses L with optimal policy cumulative loss $L*$.

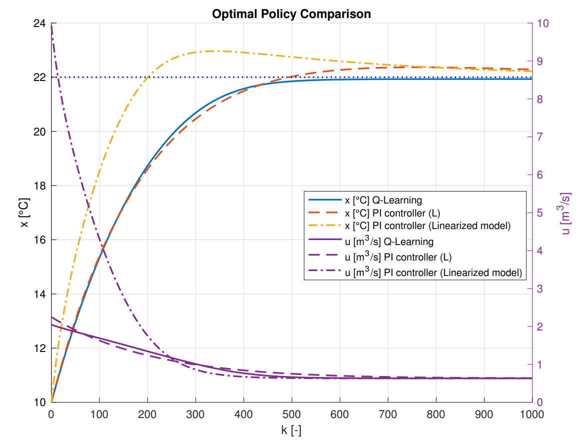

Fig. 4. Comparison of Q-Learning optimal control policy, PI controller designed by cumulative loss comparison and PI controller designed from linearized model of FCU. The results are presented on nonlinear FCU model.

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 would just annoy the occupants. Also, the fan would use more electricity. As such, the policy resembles a proportional feedback controller with variable gain. It does not have any integral action whereas a proportional derivative integral (PID) controller would be normally used for similar purpose. However, it can be shown that the intergal action can be added to Q learned policy 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 their cumulative loss function L values. This function represents the sum of all losses during all episodes, i.e. $L = \sum_k \ell_k$ and the 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). The cumulative loss was calculated for each in the same way as L^* was calculated, i.e. using (14), so the cumulative losses are made comparable. The initial room temperature was $10°\mathrm{C}$ and the cumulative loss was calculated for 1,000 sampling periods. These cumulative loss values were then compared to L^* in Fig. (3) and the best PI controller parameters from the grid were selected for Fig. (4). A PI controller designed for a linearized model of FCU at $u_0 = 1, x_0 = 22\,[°\mathrm{C}]$ is also visualized. It can be observed that the best PI almost matches the result of the Q learning.

Next, the controllers and the Q learning were tested

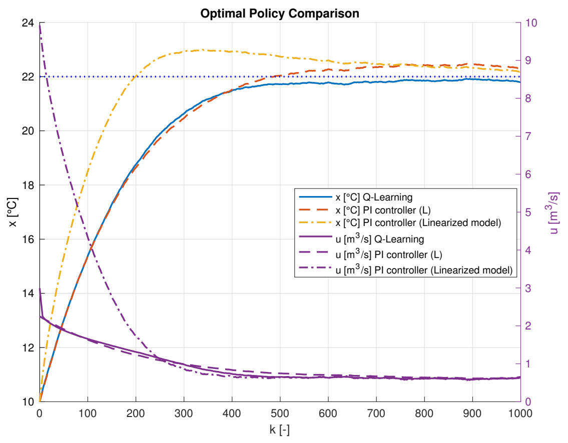

Fig. 5. Comparison of Q-Learning optimal control policy, PI controller designed by cumulative loss comparison and PI controller designed from linearized model of FCU. The results are presented on nonlinear FCU model with noisy net heat load/heat loss q_L.

considering the net heat load/heat loss q_L not constant but uniformly distributed over $[-7, 0]$. 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.

Note, that the dataset size is very important and if only a small dataset is available, the Q-Learning result degrades significantly.

VI. CONCLUSIONS

This paper described a practical approach of using GPR based Q-Learning algorithm to find a control law for a completely unknown nonlinear process based on a historical dataset of size $\sim 10^3$. Engineers face such problem often and a solution is of practical interest. GPR approach was used to estimate Q function value in any point in the action value space. The policy iterations were used to optimize the controller as this method converges rapidly. It requires the initial control to be stabilizing. This does not seem to be a serious constraint in many practical applications such as building control. The optimal control law was then fully defined by the minima of Q function Q^*. Although not explained in this paper, GPR can be globally minimized even if the function f is not convex provided the kernel is either convex after a transformation [?]. It was shown how GPR can be used to get an unbiased Q estimate which is not sensitive to noise affecting the process. Note that the proposed method is more accurate than just solving the temporal difference equation in the least squares sense which results in a bias [?]. 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 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 with the assuredly optimal cumulative loss L^*, which was found by the Q-learning. The best controller from the grid is slightly worse than L^*. Such direct controller optimization may be impractical in reality because it is very time consuming. Also, another PI controller was designed based on a linearized FCU model. This traditional approach could be actually used in practice together with, for example, Ziegler Nichols PID calibration method. The controllers were compared in Section V and the performance of both PI found by direct search and PI designed using linearized FCU were shown to be worse than Q-Learning policy π^*. The whole process was described for reader's understanding on the high level. Many technical details were mentioned just briefly. However, the method is quite simple and straightforward.

The main pitfalls of the process may be also pointed out. Firstly, it is necessary to choose several parameters, i. e. the hyperparameters for GP and the training dataset. Their optimization and diagnostics is possible and it is current research topic. Although result with only one kernel (SE times quadratic) was presented, we also tried different kernels with similar results provided the hyperparameters were chosen reasonably and the kernels were smooth. Tuning of hyperparameters is not discussed in this paper. Next interesting problem is the tuning of sparse GPR parameters. Overall, the method gives reasonably consistent results not overly sensitive to the data or parameters.