Learning Throttle Valve Control Using Policy Search

Bastian Bischoff[1], Duy Nguyen-Tuong[1], Torsten Koller[1], Heiner Markert[1], and Alois Knoll[2]

[1] Robert Bosch GmbH, Corporate Research, Robert-Bosch-Str. 2, 71701 Schwieberdingen, Germany
[2] TU Munich, Robotics and Embedded Systems, Boltzmannstr. 3, 85748 Garching at Munich, Germany

Abstract. The throttle valve is a technical device used for regulating a fluid or a gas flow. Throttle valve control is a challenging task, due to its complex dynamics and demanding constraints for the controller. Using state-of-the-art throttle valve control, such as model-free PID controllers, time-consuming and manual adjusting of the controller is necessary. In this paper, we investigate how reinforcement learning (RL) can help to alleviate the effort of manual controller design by automatically learning a control policy from experiences. In order to obtain a valid control policy for the throttle valve, several constraints need to be addressed, such as no-overshoot. Furthermore, the learned controller must be able to follow given desired trajectories, while moving the valve from any start to any goal position and, thus, multi-targets policy learning needs to be considered for RL. In this study, we employ a policy search RL approach, Pilco [2], to learn a throttle valve control policy. We adapt the Pilco algorithm, while taking into account the practical requirements and constraints for the controller. For evaluation, we employ the resulting algorithm to solve several control tasks in simulation, as well as on a physical throttle valve system. The results show that policy search RL is able to learn a consistent control policy for complex, real-world systems.

1 Introduction

The throttle valve, as shown in Figure 1, is an important and widely-used technical device for many industrial and automotive applications, such as for pressure control in gasoline combustion engines and flow regulation in air conditioning and heat pumps. Usually, the throttle valve system consists of a valve and an actuator, e.g. a DC-motor. The throttle valve control task is to move the valve from arbitrary positions to given desired positions by regulating the actuator inputs.

Controlling the throttle valve is a challenging task. Due to the spring-damper design of the valve system, we have a highly dynamic behavior. As many unknown nonlinearities are involved, such as complex friction, accurate physical models of the valve dynamics are hard to obtain. In practice, the valve needs to be

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part I, LNAI 8188, pp. 49–64, 2013.

controlled at a very high rate, e.g. 200Hz, and desired valve positions need to be reached as fast as possible. While requiring a fast control performance, no overshoot is allowed here, i.e. the valve position must not exceed the desired position. This requirement is essential for pressure control in gasoline combustion engines for automotive application, as addressed in this paper. Here, an open valve corresponds to a car acceleration and, thus, an overshoot during the valve control would result in undesirable jerks of engine torque. These constraints, e.g. unknown nonlinearities and fast control without overshoot, make the throttle valve controller design difficult in practice.

Fig. 1. Example of a throttle valve used in combustion engines for automotive applications

In the literature, several approaches are discussed to tackle challenges of throttle valve control based on methods of classical control theory [12–14]. These approaches usually involve tedious, manual tunning of controller parameters. Furthermore, profound knowledge of the physical system is required in order to obtain a good parametrization of the controller in this case. These limitations motivate the approach used in this study. We investigate how machine learning techniques, especially, Reinforcement Learning (RL), can be employed to successfully *learn* a control policy from experience, while incorporating required practical constraints. Beside the mentioned challenges, several RL problems need to be tackled, such as learning multi-targets and handling large data during the learning process. In this paper, we employ a probabilistic, model-based RL approach, e.g. the probabilistic inference for control algorithm (Pilco) [2], for learning the control policy. We modify Pilco taking in account the discussed requirements. The method is implemented and evaluated in simulation, as well as on a real throttle valve system. The evaluation shows the feasibility of the presented RL approach and, thus, indicates the suitability of RL for real-world, industrial applications.

The remainder of the paper is organized as follows: in the next section, we introduce the throttle valve system and motivate the use of RL. In Section 3, we briefly review the basic idea behind RL and introduce Pilco. Section 4 shows how probabilistic RL, especially, Pilco, can be modified to match the required constraints. Evaluation of our method in simulation, as well as on a real throttle valve, is provided in Section 5. Finally, a conclusion is given in Section 6.

2 The Throttle Valve System

Throttle valve systems are widely used in many industrial applications, ranging from semi-conductor manufacturing to cooling systems for nuclear power plants. However, one of the most important applications can be found in automotive control, where throttle valves are employed to regulate the flow of air entering a

combustion engine. As shown in Figure 1, the valve system basically consists of a DC-motor, a spring and a flap with position sensors. Depending on the position of the flap, the gasoline-to-air ratio in the combustion chamber is adjusted and, subsequently, influences the torque generated by the engine. The dynamics of the throttle valve system can be simplified by the model [10] to be

$$\begin{bmatrix}\dot{\alpha}(t)\\ \dot{\omega}(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -K_s & -K_d\end{bmatrix}\begin{bmatrix}\alpha(t)\\ \omega(t)\end{bmatrix} + \begin{bmatrix}0\\ C_s - K_f\,\mathrm{sgn}(\omega(t))\end{bmatrix} + \begin{bmatrix}0\\ T(t)\end{bmatrix}, \tag{1}$$

where α and ω are the flap angle and corresponding angular velocity, T is the actuator input. The parameters K_s, K_d, K_f and C_s are dynamics parameters and need to be identified for a given system [18]. Model-based valve controllers rely on this model [5] and, thus, identification of dynamics parameters is necessary. However, parameter identification using sampled data can be time-consuming and difficult. It it hard to create sufficiently rich data in order to obtain plausible dynamics parameters. Furthermore, the parameters that optimally fit a data set, are often not physically consistent and, hence, physical consistency constraints have to be imposed on the identification problem [17]. Using data-based nonparametric models for RL — as employed in this paper — for learning optimal control policies can help to overcome these limitations.

2.1 Requirements for Throttle Valve Control

As the throttle valve is a real-time system, precise and fast control is crucial to provide optimal performance. In order to obtain fast control, we employ a fixed radial-basis function structure as parametrization of the controller, which can be evaluated in real-time. As shown in Section 5, the learned controller can be evaluated at a frequency of about 200Hz. Furthermore, for learning the dynamics used for model-based RL we employ a NARX-structure [9] to represent the system dynamics, as described in Section 4.1. A well-approximated dynamics model is prerequisite for learning a good control policy with model-based RL.

Typical RL problems are goal oriented [1], i.e. RL is formulated for reaching single, desired goal positions. However, when employing throttle valve control, trajectory tracking is inevitable. The learned policy needs to be able to follow desired trajectory and, thus, multi-target RL as described in Section 4.2 is required here. It is shown in the evaluation that the learned policy can generalize well for unknown goals and trajectories.

In addition to the fast control requirement, no overshoot of the trajectory is essential. As an overshoot corresponds to undesirable jerks in engine torque, the valve trajectory must not go beyond the desired valve position. On the other hand, it is required that the valve moves to the desired position as close and fast as possible. Taking in account these requirements, we design an appropriate cost function in Section 4.3. The cost is defined such that the requirements are accomplished and the resulting RL formulation remains solvable in closed form.

3 A Brief Review on RL

In this section, we provide a short review on the basic concepts of RL [1, 3]. Subsequently, we proceed to discuss the probabilistic policy search approach Pilco [2].

3.1 General Setting

In RL, we consider an agent and its interactions with the environment. The state of the learning agent is defined by $s \in S$. The agent can apply actions $a \in A$ and, subsequently, moves to a new state s' with probability given as $p(s'|s, a)$. The controller $\pi : S \to A$ determines in every state the action which should be used. If the controller is applied for T timesteps, we get a state-action sequence $\{s_0, a_0\}, \{s_1, a_1\}, \ldots, \{s_{T-1}, a_{T-1}\}, s_T$, which we call *rollout* of the controller. In case of uncertainty and noise, multiple rollouts will not be identical and the rollout must be described by probability distributions. The environment rates states s_i with a cost function $c : S \to \mathbb{R}$ (and, thus, gives a rating for the controller). The goal of the learning algorithm is to find a controller, which minimizes the expected sum $J(\pi)$ of collected cost, i.e.

$$\min_{\pi} J(\pi),\ J(\pi) = \sum_{t=0}^{T} E_{s_t}(c(s_t)), \tag{2}$$

where $p(s_0), \ldots, p(s_T)$ are the resulting state distributions on application of the controller π. The cost function c must be set according to the learning goal. The tuples s_i, a_i, s_{i+1} are saved as experience and are used to optimize the controller. RL algorithms differ in the way they use this experience to learn a new, improved controller π. Two important properties that characterize RL techniques, are model-free and model-based, as well as Policy Search and Value-function. Next we will shortly describe the approaches and examine their suitability for throttle valve control.

3.2 Approaches in Reinforcement Learning

Model-based RL describes algorithms, where the experience samples s_i, a_i, s_{i+1} are used to learn a dynamics model $f : S \times A \to S$ and the controller π is optimized using the dynamics model as internal simulation. *Model-free* RL algorithms, on the other hand, directly optimizes the controller π without usage of a dynamics model. In the last decades, model-free RL got much attention mainly because for discrete state and action sets, convergence guarantees can be given. However, often many trials are necessary to obtain a good controller. Model-based RL methods potentially use the data more efficient, but it is well known that *model bias* can strongly degrade the learning performance [4]. Here, a controller might succeed in simulation but fails when applied to the real system, if the model does not describe the complete system dynamics. To address

this problem, it is important to incorporate uncertainty — which can result from a lack of experience or due to stochasticity in the system — and to employ probabilistic dynamics models.

Besides model-free and model-based, RL methods can be divided into *policy search* algorithms and *value-function* approaches. Policy search methods directly operate on the parameters θ of a controller π_θ to optimize the sum of expected cost given in Equation (2). Therefore, a parametrized control structure has to be defined by the user. This allows to include prior knowledge about good control strategies, but the structure also limits the set of strategies that can be learned. In contrast to policy search, value-function approaches try to estimate a long-term cost for each state.

3.3 Pilco: A Model-Based Probabilistic Policy Search

For the throttle valve control task — as for many physical systems — it is not possible to perform several thousands of experiments on the real system until the learning process converges. Thus, it is important to reduce interactions with the system. This favours model-based RL approaches. Furthermore, the throttle valve has high requirements on the control frequency due to the fast system dynamics. For example, evaluation of the controller given the current system state must take at most 5ms. Therefore, policy search RL, with a control structure that can be evaluated fast, seems to be an appropriate approach.

Pilco[1] is a model-based RL algorithm, which uses Gaussian processes (GP) for modeling the system dynamics [2]. Based on this probabilistic dynamics model, a control policy can be inferred. The controller can, for example, be parametrized as a radial basis function network (RBF) with Gaussian shaped basis function. Thus, the controller is given by

$$\pi(s) = \sum_{i=0}^{N} w_i \phi_i(s),$$

with $\phi_i(s) = \sigma_f^2 \exp(-\frac{1}{2}(s - s_i)^T \Lambda (s - s_i))$, $S = [s_0, s_1, \ldots, s_N]^T$ the set of support points and w representing the weight vector. The hyperparameters of the controller are given by $\Lambda = \text{diag}(l_1^2, \ldots l_D^2)^{-1}$ and σ_f. The number N of support points is a free parameter that needs to be adjusted. The support points, the weight vector and the hyperparameters build the set θ of control parameters that need to be optimized during learning.

For learning the controller, we start with a Gaussian state distribution $p(s_0)$. The RBF network controller returns an action for every state, therefore we get a distribution $p(a_t) = \int p(a_t|s_t)p(s_t)ds_t$. This distribution is analytically approximated to a Gaussian. Given the state distribution $p(s_t)$ and the approximated Gaussian action distribution $p(a_t)$, the joint distribution $p(s_t, a_t)$ is approximated. The dynamics model takes this distribution as input and returns the distribution $p(s_{t+1})$ for the next state. The expected cost $J(\theta) = \sum_{t=0}^{T} E_{s_t}(c(s_t))$

[1] We thank Marc Deisenroth for providing us the Pilco code.

Algorithm 1. Pilco: model-based policy search

1: $D \coloneqq D_{init}, \theta :=$ random $\triangleright$ initialize dynamics data set D and control parameters θ
2: **for** $e := 1$ **to** *Episodes* **do**
3: Learn dynamics model $GP_{dynamics} : S \times A \to S$
4: Improve policy:
5: Estimate rollout $p(s_0), p(s_1), \ldots, p(s_T)$ of π_θ using $GP_{dynamics}$
6: Rate policy-parameters $J(\theta) = \sum_{t=0}^{T} E_{s_t}(c(s_t))$
7: Adapt policy-parameters $\theta = \theta + \nabla J(\theta)$
8: Apply controller π_θ on system, $D := D \cup \{s_0, \pi_\theta(s_0) = a_0, s_1, \ldots\}$
9: **end for**

can subsequently be computed from these rollout results. Based on generated cost values, the controller can now be optimized. The optimization step can be performed using gradient descend procedure, where analytical gradients are computed on the hyperparameters. The resulting algorithm [2] is summarized in Algorithm 1.

4 Learning Throttle Valve Control with RL

In Section 2, we described the throttle valve system and the task specific requirements for throttle valve control. In this section, we adapt the Pilco algorithm described in the previous section taking into account the desired requirements. First, we show how the system dynamics can be modeled using Gaussian processes while employing a NARX-structure with state and action feedback. Additionally, to handle the large amount of dynamics data occurring during learning, an information gain criterion is used for selecting informative data points. As the learned controller must be able to follow arbitrary trajectories, we describe the setting for multiple start and goal states. Finally, we define a cost function addressing the no-overshoot restriction, while the integrals involved in the Pilco learning process can still be solved analytically. The analytical gradients, which are essential for policy optimizing, will be provided.

4.1 Modeling Throttle Valve Dynamics

A dynamics model describes the behavior of a system, i.e. the probability $p(s'|s, a)$ that the system state changes to s' when action a is applied in state s. Here, the actions a correspond to the input voltage u of the DC-motor. The opening angle α of the valve changes dynamically depending on the input voltage. As shown in Equation (1), the dynamics of the valve can be approximated by a second-order system. Due to this insight, the RL state s_t is defined as $s_t = [\alpha_t, \alpha_{t-1}, \alpha_{t-2}, u_{t-1}, u_{t-2}]$. Thus, the resulting state s_t has the well-known *Nonlinear Autoregressive Exogenous* (NARX) structure [9], as shown in Figure 2. For modeling the dynamics, nonparametric Gaussian process regression is employed to predict α_{t+1} given a state s_t and the current action u_t.

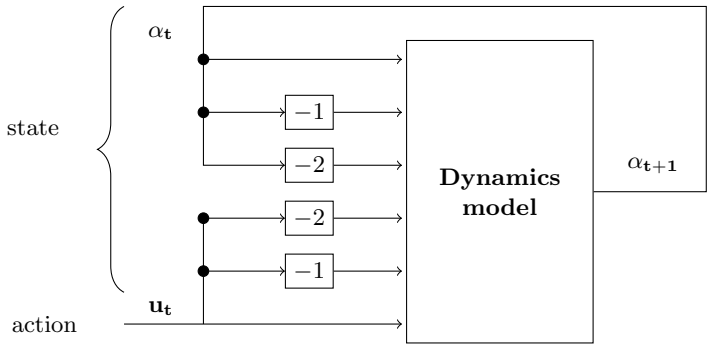

Fig. 2. NARX-structure with state and action feedback to model the throttle valve dynamics. The current valve opening α_t, as well as past valve openings $\alpha_{t-1}, \alpha_{t-2}$ and input voltage u_{t-1}, u_{t-2} are jointly used to predict α_{t+1}. For modeling the dynamics, GP regression is employed.

A further challenge in modeling the throttle valve dynamics is the amount of data generated by the system. Dynamics data is sampled at a high frequency, e.g. 200 samples per second, leading to massive data sets. Here, we employ an information gain criterion [8] to reduce the sampled data while retaining relevant dynamics information. This significantly reduces the overall learning time as well as the memory requirements. A GP is employed for modeling the dynamics, the information gain criterion can be computed analytically and efficiently. See [8] by Seeger et. al. for more details.

4.2 Multiple Start and Goal States

For throttle valve control, it is important that the learned controller can follow arbitrary trajectories. Thus, the controller must be able to move the valve from any given start position to any goal position. One approach would be to learn seperate controllers for a set of goal states and combine these controllers for arbitrary goal positions. However, this is complex for non-linear systems and may not be optimal globally. Instead, we include the goal position g as input to the controller, i.e. $u = \pi_\theta(s, g)$, as described in [11]. Now, one joint controller is learned for all goal positions. This formulation allows to set the goal state dynamically on controller application.

In the standard Pilco framework, the control parameters θ are optimized with respect to the sum of expected cost, $J(\theta) = \sum_{t=0}^{T} E(c(s_t))$, where $s_t \sim \mathcal{N}(\mu, \Sigma)$ is a Gaussian distribution. For multiple start states s_t^i and goal states g_i, the objective function can be modified (see [11]) to

$$J(\theta) = \sum_{i=0}^{|Z|} \sum_{t=0}^{T} E(c(s_t^i, g^i)) \,, \tag{3}$$

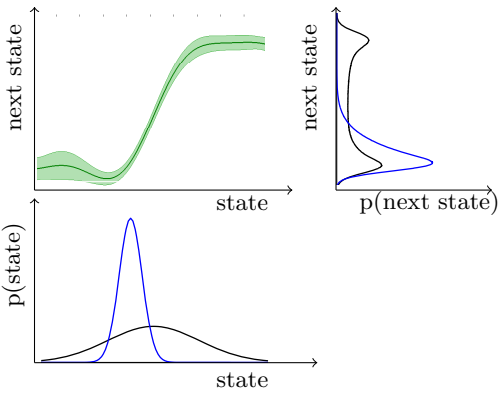

Fig. 3. The upper left figure shows the Gaussian process dynamics model for a fixed action. The lower picture shows two state distributions. The right upper plot shows the corresponding next state distributions, after mapping the state distributions through the dynamics GP. When the start distribution is too broad, the state distribution after mapping is too complicated to approximate by a Gaussian as performed in Pilco.

where Z represents pairs of start and goal states, e.g. $Z = \{(s_0^0, g^0), (s_0^1, g^1), \ldots\}$. The start distributions given by $s_0^i \in Z$ and σ_0 as well as the goal states g^i in Z are determined such that they cover the relevant state space parts. The variance of the start distributions $s_0^i \in Z$ needs to be chosen appropriately. Given a very broad start distribution s_0, the next state distribution — after mapping it through the dynamics GP model — can be difficult and, thus, more complicated when approximated by a Gaussian as done by Pilco. However, when the start variance is small, performance might me suboptimal for some start states not covered by the start distributions. Figure 3 illustrates the effects when mapping different state distributions through the GP dynamics model.

4.3 Cost Function for Learning Throttle Valve Control

In this section, we define an appropriate cost function for policy optimization. The saturated immediate cost [2] given by $c_d(s, g) = 1 - \exp(-\|s - g\|^2 / (2d^2))$ with goal state g is a general, task unspecific cost function. Here, the hyperparameter d describes the width of the cost. However, this cost function is not appropriate for learning throttle valve control, as it does not avoid valve trajectory overshoot. Taking in account the no-overshoot restriction, we introduce an *asymmetric saturating cost function*

$$c(s, g) = \begin{cases} c_{d_1}(s, g), & \text{if } s \leq g \\ c_{d_2}(s, g), & \text{otherwise} \end{cases}, \tag{4}$$

where c_{d_i} is the saturating cost with width d_i and g is the goal state. This continuous, smooth function can be described as a saturating cost function with variable steepness on both sides of the goal depending on the parameters d_1, d_2.

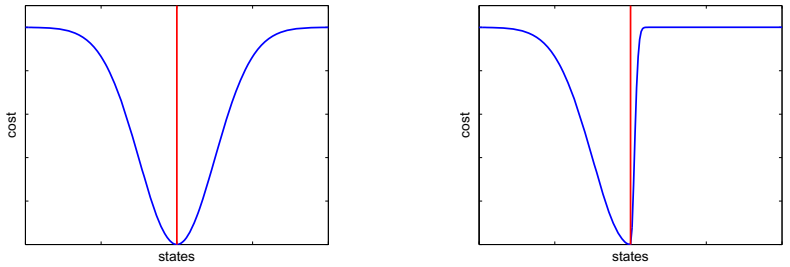

Fig. 4. The left figure shows the symmetric saturating cost over states given on the x-axis. The goal state is indicated by the red vertical line. The right figure shows the asymmetric saturating cost with variable steepness on both sides. Overshooting the goal state (here, states right of the goal state) implies high costs, while approaching the goal from a state left of the goal leads to decreasing cost.

In contrast to the usual symmetric saturating cost, this allows us to assign a decreasing cost when the state converges to the goal state while overshoot is punished, see Figure 4.

To estimate the cost for a given set of parameters in the probabilistics model-based RL framework, the expected cost $E(c(s,g))$ is required for a Gaussian state distribution $s \sim \mathcal{N}(\mu_s, \Sigma_s)$. For the asymmetric saturating cost in Equation (4), $E(c(s,g))$ is given as

$$\begin{aligned} E(c(s,g)) &= \int_{-\infty}^{g} c_{d_1}(s,g)p(s)ds + \int_{g}^{\infty} c_{d_2}(s,g)p(s)ds \\ &= \frac{1}{\sqrt{2\pi\sigma^2}}\left[\int_{-\infty}^{g} e^{\ell_1}ds + \int_{g}^{\infty} e^{\ell_2}ds\right] = \frac{1}{\sqrt{2\pi\sigma^2}}\left[\frac{w_1}{v_1}r_1 + \frac{w_2}{v_2}r_2\right] \end{aligned}$$

with

$$\ell_i = -\left(1/d_i^2 + 1/\sigma^2\right)\left[s - \left(\frac{g\sigma^2 + \mu d_i^2}{\sigma^2 d_i^2}\right)\right]^2 - \frac{(g-\mu)^2}{d_i^2 + \sigma^2},\ \ w_i = e^{\left(\frac{-(g-\mu)^2}{d_i^2+\sigma^2}\right)}$$

$$v_i = \sqrt{\frac{1}{d_i^2} + \frac{1}{\sigma^2}},\ \ u_i = \frac{g\sigma^2 + \mu d_i^2}{\sigma^2 + d_i^2},\ r_1 = \int_{-\infty}^{v_1(g-u_1)} e^{-q^2}dq,\ r_2 = \int_{v_2(g-u_2)}^{\infty} e^{-q^2}dq$$

Using the error function $\int_{-\infty}^{b} e^{-q^2}dq = \frac{\sqrt{\pi}}{2}\left(\mathrm{erf}(b) + 1\right)$, we have

$$r_1 = \frac{\sqrt{\pi}}{2}\left(\mathrm{erf}(v_1\,(g-u_1)) + 1\right),\ r_2 = \frac{\sqrt{\pi}}{2}\left(1 - \mathrm{erf}(v_2\,(g-u_2))\right)$$

Given $E(c(s,g))$, the gradients for the policy optimization can be given as

$$\frac{\delta E(c(s,g))}{\delta \mu} = \frac{1}{2\sqrt{2\pi\sigma^2}} \sum_{i=1}^{2} \frac{w_i}{v_i} \frac{1}{d_i^2 + \sigma^2} \left[2r_i(g-\mu) + (-1)^i v_i e^{-(v_i(g-u_i))^2} \right] ,$$

$$\frac{\delta E(c(s,g))}{\delta \sigma^2} = \frac{-1}{2\sigma^2} E(c(s)) + \sum_{i=1}^{2} \frac{w_i}{v_i} \left[\frac{r_i}{2\sigma^4 v_i^2} + \frac{r_i(g-\mu)^2}{(\sigma^2 + d_i^2)^2} + (-1)^i e^{-(v_i(g-u_i)^2)} \frac{\sqrt{2}}{\pi} (\delta_{v_i}(g-u_i) - \delta_{u_i} v_i) \right]$$

where

$$\delta_{u_i} = \frac{d_i^2(g-\mu)}{(\sigma^2 + d_i^2)^2} , \ \delta_{v_i} = \frac{1}{2} v_i^{-1} \sigma^{-4} , \ \delta_{w_i} = e^{\left(\frac{-(g-\mu)^2}{\sigma^2 + d_i^2} \right)} \frac{(g-\mu)^2}{(\sigma + d_i^2)^2} .$$

5 Evaluations

In this section, we evaluate the presented RL approach on a throttle valve simulation, as well as on a physical throttle valve system. The experiment setting and learning process are described in detail.

5.1 Simulation Results

First, learning is performed on a throttle valve simulator. We employed the Pilco algorithm as given in Algorithm 1 with an RBF control structure using 40 base functions. The controller can be evaluated in approximately 1ms, which allows for a control frequency of at most 1000Hz. In all experiments, a control frequency of 200Hz was used.

As a first step, we learn a controller for a single start and target state, i.e. a trajectory starting from $\alpha_0 = 70°$ with desired valve opening $g = 10°$. To obtain initial dynamics data D_{init}, random actions are applied twice for 0.16 seconds leading to two trajectories starting in α_0. A NARX-structure is established for modeling the dynamics (see Section 4.1). Optimization is performed with respect to the asymmetric saturating cost with width $d_1 = 0.5, d_2 = 3.5$. Figure 5 shows the learning process over several episodes. It can be seen that the learning converges to a near optimal solution after 4 Episodes. Next, we compare the learning result for the symmetric saturating cost with the asymmetric cost function introduced in Section 4.3. It can be seen in Figure 6 that the asymmetric cost function significantly reduces the overshoot (on the right) compared to the standard symmetric saturating cost (on the left).

So far, the learned controller was optimized for a single trajectory from a single start angle to a single goal angle. We now employ multiple start and goal states as described in Section 4.2. Here, we choose 10 different combinations of start positions and goal positions covering the state space equally, e.g.

$$\begin{aligned} \tilde{Z} &= \left\{ (\alpha_0^0, g^0), (\alpha_0^1, g^1), \ldots \right\} \\ &= \{ (30,40), (30,45), (30,60), (40,55), (45,55), \\ &\quad (45,60), (55,40), (60,30), (60,45), (60,50) \} . \end{aligned}$$

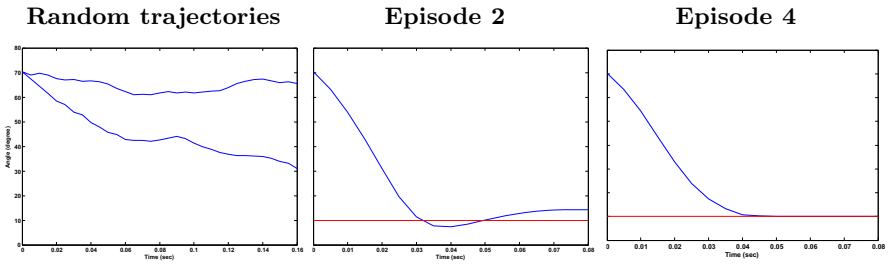

Fig. 5. The figure shows the learning process for the start angle $\alpha_0 = 70$ with goal angle $g = 10$. The left-most plot shows the two random trajectories used as initial dynamics data set. After 2 episodes, the learned controller gets close to the learning goal, but still overshoots and does not reach the goal angle accurately. After 4 episodes, the learning has converged and the resulting control policy shows a near optimal behavior.

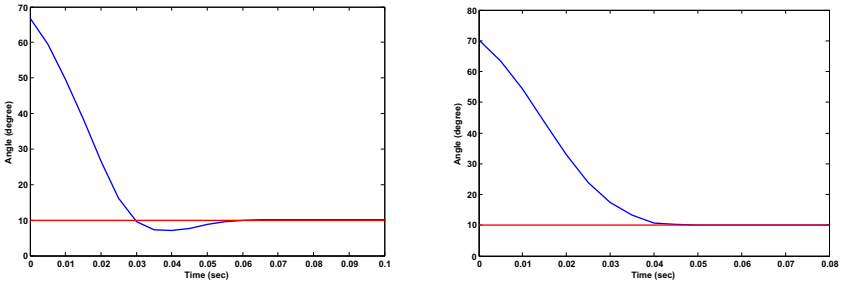

Fig. 6. On the left, the learning result for the symmetric saturating cost function is shown, the result for the asymmetric saturating cost is shown on the right. While the symmetric cost leads to significant overshoot, the behavior for the asymmetric cost is near optimal.

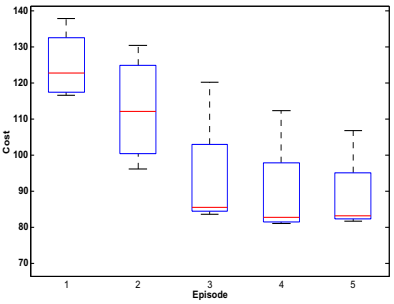

Fig. 7. The figure shows the cost over episodes of 4 independent learning attempts in simulation. In 3 of 4 cases, the learning converges to a near optimal solution after 3 episodes.

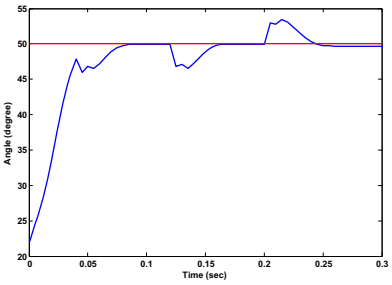

Fig. 8. The controller is applied to move the valve from 22 degree valve opening to 50 degree on the throttle valve simulator. At times 0.03, 0.13 and 0.23 torque disturbances are introduced. The controller handles the disturbance well and returns to the goal angle quickly.

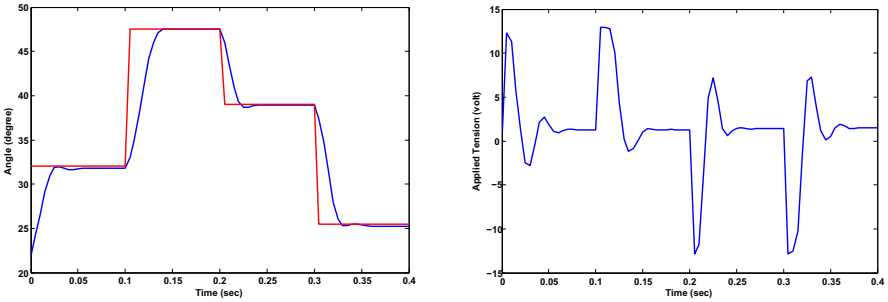

Fig. 9. The learned controller (blue) is applied on the simulated throttle valve with a step trajectory (red) as desired goal trajectory. The left plot shows the valve angle over time, while the right plot shows the applied voltage. The results show, that the learned controller successfully matches the desired trajectory.

The standard deviation of the Gaussian start state distributions was set to $\sigma_0 = 3°$. We repeated 4 independent runs. The cost over episodes for the runs are shown in Figure 7. In 3 out of 4 cases, the optimal solution was already found after 3 episodes. Next, the learned controller of a successful run is evaluated with respect to its robustness towards torque disturbances. Figure 8 shows a trajectory, where three disturbances are applied, the learned controller returns the valve to the desired angle in a robust manner as expected. Finally, we apply the learned controller to follow an arbitrary step trajectory. The resulting trajectory as well as the voltage applied by the controller is shown in Figure 9.

5.2 Real Throttle Valve Results

In this section, the RL controller is learned on a real throttle valve system. The performance of the RL controller is tested on different control tasks.

For learning on the real system, we use the same setting as described for simulation in the previous section. Again, an RBF control structure using 40 basis functions is employed, data is sampled at a rate of 200Hz. Here, we directly handle the case of multiple start and goal states on the real system. As in simulation, the controller is optimized with respect to 10 combinations of start states and goal states. In each episode, additional dynamics data is sampled by application of the controller for 1.2 seconds. In this 1.2s timeslot, a step trajectory consisting of the 10 start/goal combinations is employed as desired trajectory. The information gain criterion significantly reduces the overall size of the dynamics data set, e.g. the 1200 dynamics samples gathered after 4 episodes are reduced to a training set of only 300 elements.

Figure 10 shows the cost of controller application after each episode. The learning was stopped after 4 episodes, since the controller already reached near optimal performance. The controller learned after 4 episodes is evaluated on various desired trajectories. In Figure 11, the performance of two of the 10 learned trajectories is illustrated. Figure 12 shows the application of the learned controller on a more complex trajectory with arbitrary steps. Further, we used a

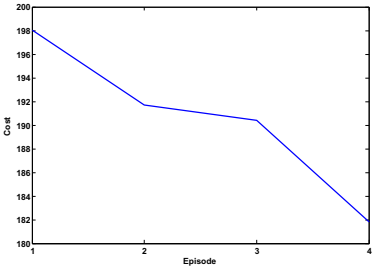

Fig. 10. In the learning process a rollout is performed after each learning episode. The figure shows the accumulated cost (see Equation (3)) for each episode rollout for the asymmetric saturating cost function and $d_1 = 2, d_2 = 0.5$.

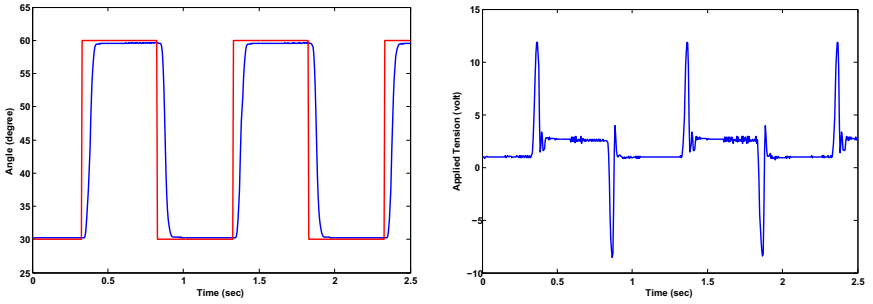

Fig. 11. The learned controller (blue) is applied on the physical throttle valve system with a step trajectory (red) as desired trajectory. While the left figure shows the valve angle over time, the voltage over time is shown on the right. As can be seen, the learned controller performs well and is able to follow the desired trajecotry in a robust manner.

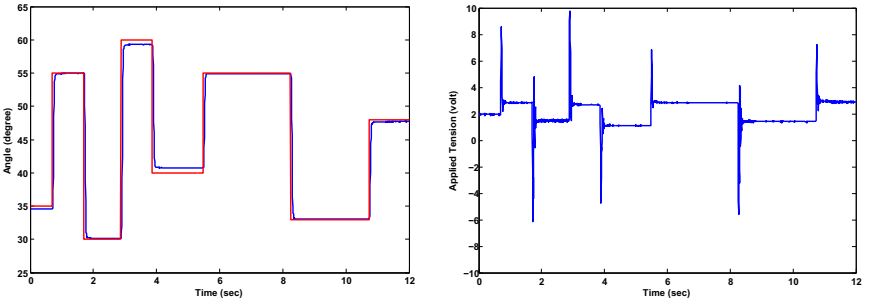

Fig. 12. As a next test case, we used a more complex step trajectory (red). The learned controller (blue) is able to follow the step trajectory, while the accurarcy varies with the goal angle in a range of approximately 1 degree.

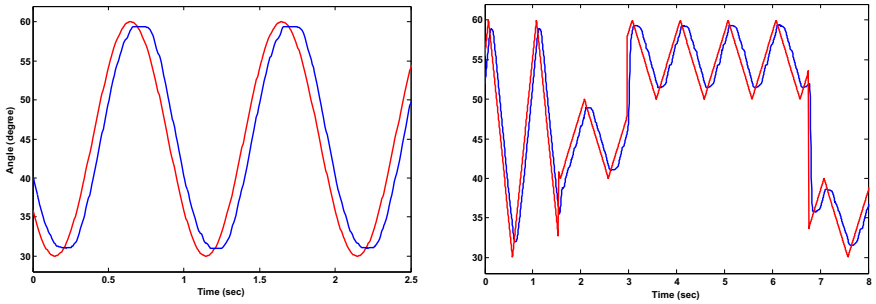

Fig. 13. The figure on the left shows the learned controller (blue) following a desired sine trajectory (red) on the physical valve. The resulting trajectory is shifted by a small amount of time, because of the time required to reach a new desired goal angle. On the right, the controller follows a ramp trajectory of varying steepness and amplitude.

sine as well as variable ramps as desired trajectories, see Figure 13. In all cases, the learned controller was able to follow the desired trajectories in an accurate and robust manner without significant overshoot.

Finally, we compare the learned controller to a classical PID controller with manually tuned parameters. Furthermore, we test the robustness of both controllers towards torque disturbances. The discrete time PID structure is give as

$$u_t = K_P e_t + K_I \sum_{i=0}^{t} e_i dt + K_D \frac{e_t - e_{t-1}}{dt}, \tag{5}$$

where T is the current time index, error $e_t = \alpha_t - g$, $1/dt$ equals the control frequency. The gains K_P, K_I, K_D are free parameters and need to be adjusted. This tuning involves several trade-offs, such as accuracy versus no overshoot. It must be kept in mind that inappropriate parameters lead to unstable control behavior that potentially damages the system. For the subsequent experiments, we use the parameters obtained after extensive tuning with help of a system expert.

Figure 14 shows the learned controller compared to PID control. While both controllers are able to follow the desired trajectory, the learned controller outperforms the PID control in terms of accuracy. Next, we examine both controllers with respect to disturbances. Figure 15 illustrates the behavior when a constant disturbance is introduced for a small amount of time. Compared to the learned controller, the impact of the disturbance is significantly higher for the PID control. This results from a slightly longer time period until the PID controller counteracts the disturbance. Furthermore, the accuracy of the PID control is significantly reduced after the disturbance due to the integration element I (see Equation (5)). More advanced methods of control theory help to improve the results compared to the standard PID control. However, this often increases the number of free parameters that need to be tuned.

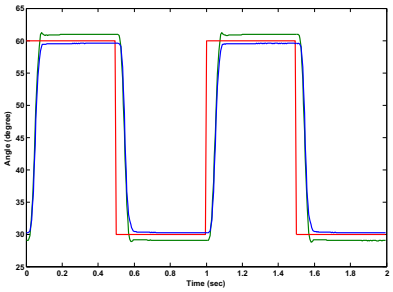

Fig. 14. The figure shows application of the learned controller (blue) and the PID controller (green) on the physical throttle valve. Both controller are able to follow the desired step trajectory (red), the accuracy of the learned controller exceeds the PID performance.

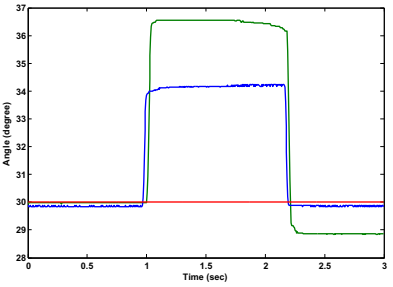

Fig. 15. At timestep 1s, a torque disturbance is introduced to the system until timestep 2.2s. The learned controller (blue) handles the disturbance well, while the PID control (green) shows a stronger impact of the disturbance and deteriorated performance afterwards.

A video was created to illustrate the learning process described in chapter 3, algorithm 1. A controller is learned over the course of 4 episodes and the learning progress is shown through rollouts on the throttle valve system on each episode: `www.youtube.com/watch?v=-HpzKsxios4`.

6 Conclusion

In this study, we investigate how throttle valve control can be learned from experience, while showing a practical application of probabilistics RL on a real-world problem. A throttle valve is an important industrial device to regulate flows of gas or fluids and has various application, e.g. pressure regulation in combustion engines. As analytical models are hard to obtain due to complex dynamics and unknown nonlinearities, model-based position control of the throttle valve is a challenging problem. In this paper, we modify the probabilistic inference for control algorithm (Pilco) to match the requirements of throttle valve control, such as no-overshoot restriction. We evaluated the approach in simulation, as well as on a real throttle valve system. The results show that policy search RL is able to learn a consistent control policy for complex, real-world systems.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press (1998)
2. Deisenroth, P.M., Rasmussen, C.E.: PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In: ICML, pp. 465–472 (2011)

3. Wiering, M., van Otterlo, M.: Reinforcement Learning: State-of-the-Art. Adaptation, Learning, and Optimization. Springer (2012)
4. Deisenroth, M.P.: Efficient Reinforcement Learning using Gaussian Processes. PhD Thesis, Karlsruhe (2010)
5. Yuan, X., Wang, Y., Wu, L.: SVM-Based Approximate Model Control for Electronic Throttle Valve. Transactions on Vehicular Technology 57(5) (2008)
6. Nentwig, M., Mercorelli, P.: Throttle valve control using an inverse local linear model tree based on a fuzzy neural network. In: 7th International Conference on Cybernetic Intelligent Systems (2008)
7. Yuan, X., Wang, Y., Lianghong, W., Xizheng, X., Sun, W.: Neural Network Based Self-Learning Control Strategy for Electronic Throttle Valve. Transactions on Vehicular Technology 59(8) (2010)
8. Seeger, M., Williams, C.K.I., Lawrence, N.D.: Fast Forward Selection to Speed Up Sparse Gaussian Process Regression. In: 9th International Workshop on Artificial Intelligence and Statistics (2003)
9. Leontaritis, I.J., Billings, S.A.: Input-output Parametric Models for Nonlinear Systems. International Journal of Control 41, 303–344 (1985)
10. Griffiths, P.G.: Embedded Software Control Design for an Electronic Throttle Body. Master's Thesis, Berkeley, California (2000)
11. Deisenroth, M.P., Fox, D.: Multiple-Target Reinforcement Learning with a Single Policy. In: ICML Workshop on Planning and Acting with Uncertain Models (2011)
12. Nakamura, H., Masashi, M.: Thottle valve positioning control apparatus. United States Patent 5, 852, 996 (1998)
13. Al-samarraie, S.A., Abbas, Y.K.: Design of Electronic Throttle Valve Position Control System using Nonlinear PID Controller. International Journal of Computer Applications 59, 27–34 (2012)
14. Wang, H., Yuan, X., Wang, Y., Yang, Y.: Harmony search algorithm-based fuzzy-PID controller for electronic throttle valve. Neural Computing and Applications 22, 329–336 (2013)
15. Deisenroth, M.P., Rasmussen, C.E., Fox, D.: Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. RSS (2011)
16. Fisher Controls International LLC: Control Valve Handbook, 4th edn. (2005)
17. Ting, J., D'Souza, A., Schaal, S.: A Bayesian Approach to Nonlinear Parameter Identification for Rigid-Body Dynamics. Neural Networks (2009)
18. Garcia, C.: Comparison of Friction Models Applied to a Control Valve. Control Eng. Pract. 16(10), 1231–1243 (2008)