Comparison of Model-Based, Model-Free Reinforcement Learning, and Optimal Control

Julia Ströbel, Sebastian Hügler, and Jan Brüdigam

***Abstract*— The control of dynamical systems can be achieved by a variety of approaches with different advantages and drawbacks. In this paper, implementations of model-based reinforcement learning (RL), model-free RL, and optimal control for linear and non-linear dynamical systems are compared and discussed.**

I. INTRODUCTION

The increase of computational power in recent years has allowed for the successful implementation of learning algorithms to control a wide range of dynamical systems. Nonetheless, classical approaches to control problems also provide useful solutions for this class of systems. In the following, model-based RL, model-free RL, and optimal control algorithms are presented, evaluated and discussed.

II. SYSTEMS

This section shows the three models that will be controlled by the different algorithms.

A. Spring-Mass

The spring-mass system can be described by a system of differential equations

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u, \tag{1}$$

with the parameters $k = 1\,\text{N/m}$ (spring) and $m = 1\,\text{kg}$ (mass).

B. Pendulum

The dynamics of the pendulum are described in (2).

$$\dot{x} = \begin{pmatrix} x_2 \\ \frac{u - b\,x_2 - m\,g\,s\sin(x_1)}{m\,s^2} \end{pmatrix}, \tag{2}$$

with the input u, the parameters $b = 0.2\,\text{sNm/rad}$ (friction), $m = 1\,\text{kg}$ (mass), $g = 9.82\,\text{m/s}^2$ (gravity), and $s = 1\,\text{m}$ (length of the pendulum).

C. Cart-Pole

The cart-pole dynamics are as follows:

$$\dot{x} = \begin{pmatrix} x_2 \\ \frac{2\,m\,l\,x_4^2\,\mathrm{s}_3 + 3\,m\,g\,\mathrm{s}_3\,\mathrm{c}_3 + 4(u - c\,x_2)}{4(M+m) - 3\,m\,\mathrm{c}_3^2} \\ x_4 \\ \frac{-3\,m\,l\,x_4^2\,\mathrm{s}_3\,\mathrm{c}_3 - 6(M+m)\,g\,\mathrm{s}_3 - 6(u - c\,x_2)\,\mathrm{c}_3}{l(4(M+m) - 3\,m\,\mathrm{c}_3^2)} \end{pmatrix}, \tag{3}$$

with the parameters $m = 0.2\,\text{kg}$ (mass of pendulum), $M = 1\,\text{kg}$ (mass of cart), $l = 0.5\,\text{m}$ (length of the pole), $g = 9.82\,\text{m/s}^2$ (gravity), $c = 0.2\,\text{sN/m}$ (friction), and where s_3 and c_3 stand for $\sin(x_3)$ and $\cos(x_3)$, respectively.

III. MODEL-BASED RL

In model based RL a model of the system to be controlled is learned using measurement data and a control policy is optimized using the learned model.

A. Algorithm

In this work the PILCO approach from [1] is investigated. In PILCO a probabilistic dynamic model for long term policy planning is trained. The unknown system dynamics are assumed with

$$x_t = f\left(x_{t-1}, u_{t-1}\right), \tag{4}$$

where $x \in \mathbb{R}^D$ are the system states, $u \in \mathbb{R}^U$ are the control inputs. The aim of PILCO is to find a policy π for the unkown system, i.e. the optimal parameters θ of a controller.

a) Learning Model Dynamics: The learning of the model dynamics described in (4) is done using a Gaussian Process (GP), where $(x_{t-1}, u_{t-1}) \in \mathbb{R}^{D+F}$ serve as inputs and $\Delta_t = x_t - x_{t-1} + \epsilon \in \mathbb{R}^D$ with $\epsilon \sim \mathcal{N}(0, \Sigma_\epsilon)$ serve as outputs. For the GP a squared exponential kernel (SQE) and a prior mean $m \equiv 0$ is chosen. The parameters of the SQE are trained using evidence maximization. Finally, the one step predictions based on the GP are:

$$p(x_t \mid x_{t-1}, u_{t-1}) = \mathcal{N}\left(x_t \mid \mu_t, \mathbf{\Sigma}_t\right), \tag{5}$$

$$\mu_t = x_{t-1} + \mathbb{E}_f\left[\Delta_t\right], \; \mathbf{\Sigma}_t = \mathrm{var}_f\left[\Delta_t\right] \tag{6}$$

Here the mean $\mathbb{E}_f\left[\Delta'\right]$ and the variance $\mathrm{var}_f\left[\Delta'\right]$ of a test input Δ' can be calculated using the GP's hyperparameters [1].

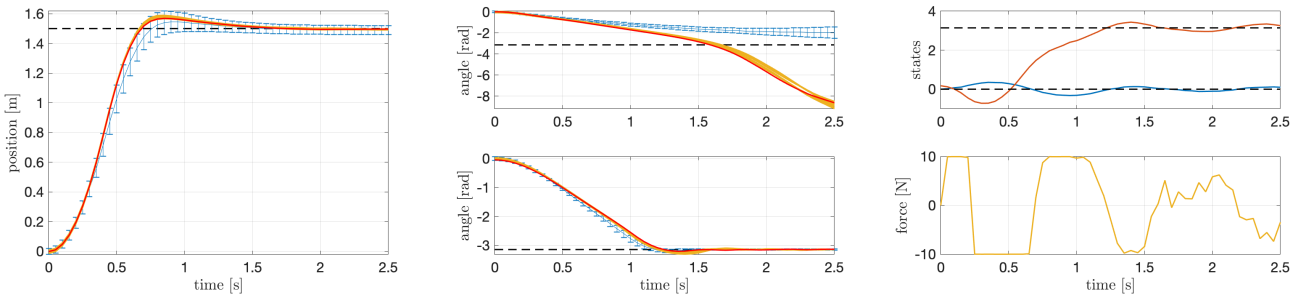
b) Policy Evaluation: In order to predict a successor state x_t based on the policy input $u_{t-1} = \pi\left(x_{t-1}, \theta\right)$ a joint distribution $p\left(\tilde{x}_{t-1}\right) = p\left(x_{t-1}, u_{t-1}\right)$ has to be determined. This is done by first integrating out the state x_{t-1} where μ_u and $\mathbf{\Sigma}_u$ are obtained. Secondly, by additionally calculating the cross-covariance $\mathrm{cov}_f = [x_{t-1}, u_{t-1}]$ the joint state-policy distribution can be approximated by a Gaussian. This result is then used for predicting $p\left(\Delta_t\right)$ with

$$p\left(\Delta_t\right) = \int p\left(f(\tilde{x}_{t-1}) \mid \tilde{x}_{t-1}\right) p\left(\tilde{x}_{t-1}\right) \mathrm{d}\tilde{x}_{t-1}. \tag{7}$$

Often there is no analytically feasible solution for (7). Therefore the Gaussian of $p\left(\Delta_t\right)$ is approximated using moment matching or linearization [2]. With the Gaussian of $p\left(\Delta_t\right)$ an approximate Gaussian is obtained for $p\left(x_t\right)$ by

$$\mu_t = \mu_{t-1} + \mu_\Delta, \tag{8}$$

$$\mathbf{\Sigma}_t = \mathbf{\Sigma}_{t-1} + \mathrm{cov}\left[x_{t-1}, \Delta_t\right] + \mathrm{cov}\left[\Delta_t, x_{t-1}\right]. \tag{9}$$



(a) Spring-mass system (trial #1)

(b) Pendulum (trial #1 - top, trial #2 - bottom)

(c) Cart-pole, initial angle: 10°, angle (orange), position (blue), policy (yellow)

Fig. 1: Figures (a) and (b) show GP training trajectories (red), additional rollouts (yellow), predicted system trajectories (95% confidence, blue) and target states (black).

Finally, the expected cost is calculated according to

$$\mathbb{E}\left[c\left(x_t\right)\right] = \int c\left(x_t\right) \mathcal{N}\left(x_t \mid \mu_t, \mathbf{\Sigma}_t\right) \mathrm{d}x_t. \tag{10}$$

The cost $c(x)$ considering the target state x_{target} is an inverted squared exponential function for each state, where the width is determined by σ_c. Using this cost function the resulting partial derivatives over μ_t and Σ_t for the gradients in the policy optimization are calculable in a closed form [2].

B. Results

The PILCO framework for MATLAB from [3] is applied to the three introduced systems, using a radial basis function (RBF) controller for the policy. Figure 1a shows that the spring-mass system is controlled on trail no. 1 after optimizing the GP with the data of random exploration in the beginning. This is due to the fact that the linear system has a low complexity. The pendulum in figure 1b is controlled after trial no. 2. At trail no. 1 the state space of the system is only explored in the angular region from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. The result is that the GP is only trained with data out of this region and the uncertainty of the GP model grows when the planned trajectory leaves this region (see blue errorbars in fig. 1b). Afer trail no. 2 however, the system is stabilized at $-\pi$ for all following trials. This shows that PILCO can handle unkown nonlinear systems effectively. This becomes even more obvious when considering results of the cart-pole trajectories. It takes 7 trials for the swing up from 10° inital pole angle, 3 trials from 90° and 2 trials from 180°. After the swing up, the steady state is controlled stably with slight inaccuarcies. Figure 1c shows the swing up from 10°. Remarkable is the capability to handle the complexity of the trajectory for this swing up with only a few trials. A linear controller would not be able to plan such a trajectory.

IV. MODEL-FREE POLICY SEARCH

This section shows an approach of a model-free policy search using Policy learning by Weighting Exploration with Returns (PoWER).

A. Algorithm

The PoWER-algorithm is based on applying varying policies to the unknown system and optimizing these policies according to their rewards. The policy π_θ applied to the system in this work is based on Dynamic Movement Primitives (DMP)

$$\pi_\theta = \ddot{y} = \tau^2 \alpha_y \beta_y (g - y(t)) - \tau^2 \alpha_y \dot{y}(t) + \tau^2 f(z(t)), \tag{11}$$

with parameters $\tau, \alpha_y, \beta_y > 0$ [4]. The function f is represented by a radial basis function (RBF) network

$$f(z(t)) = \frac{\sum_{i=1}^{N} \phi_i \omega_i}{\sum_{i=1}^{N} \phi_i} z, \tag{12}$$

where ω_i corresponds to the weights of the RBFs ϕ_i. Usually, five to twenty radial basis functions ϕ_i per system's degree of freedom (DoF) are used. The variable $z(t)$ is a system state resulting from the dynamics

$$\dot{z} = -\tau\, \alpha_z\, z, \tag{13}$$

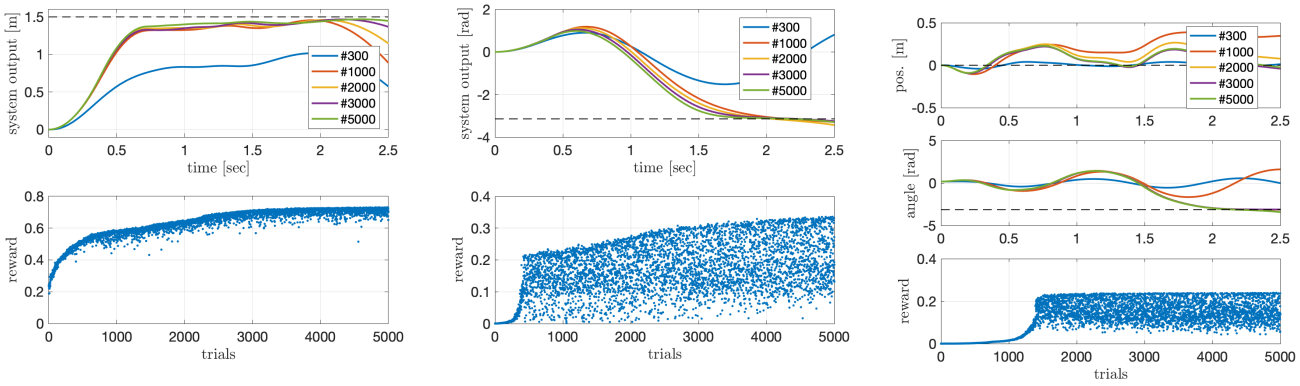
with $\alpha_z > 0$ [4]. To optimize the policy π_θ, the policy parameter vector θ (containing weights ω_i of RBF ϕ_i) is explored in parameter space by adding a gaussian noise $\varepsilon \sim N(0, \sigma_\varepsilon)$ to the parameters in iteration n [5]:

$$\theta_{temp} = \theta_n + \varepsilon \tag{14}$$

The policy obtained by temporary parameters θ_{temp} are rolled out on the system and the system trajectory x_t is measured. A cumulative reward R assesses the quality of the rollout using the trajectory x_t:

$$R(\theta_{temp}) = \sum_{t=0}^{T-1} r_t(x_t) \tag{15}$$

The reward r_t at time t is calculated using a squared exponential function. The procedure of applying different policies using different parameters θ_{temp}, which are obtained by the gaussian noise added to the base parameter θ_n, is repeated k times at each iteration n. The parameter update for the next iteration $n+1$ is then done by ordering the



(a) Spring-mass system

(b) Pendulum

(c) Cart-pole, initial angle: 10°

Fig. 2: Figures (a) to (c) show the system output over time as well as the reward at the increasing iteration steps.

temporary parameters θ_{temp} by their rewards $R(\theta_{temp})$ from best to worst. The update law is then

$$\theta_{n+1} = \theta_n + \frac{\Delta\theta_1 R(\theta_1) + ... + \Delta\theta_{10} R(\theta_{10})}{R(\theta_1) + ... + R(\theta_{10})}, \tag{16}$$

with $\Delta\theta_k = \theta_{temp,k} - \theta_n$, $k = 1, 2, ..., 10$. The mentioned steps are repeated until the reward $R(\theta_n)$ is greater than a threshold close to 1, or a maximum number of iterations is reached.

B. Results

In figure 2 the results of the three systems spring-mass, pendulum, and cart-pole are shown. For each system it takes around 3,000 trials to reach the target positions with a sufficient quality (see figure 2). At first, the three reward curves of the systems show a fast improvement. After about 500 trials the rewards of the spring-mass system and the pendulum show a moderated linear improvement; the reward of the cart-pole has its significant point at about 1,500 trials. The appearances of the three graphs differ: While the reward of the spring-mass system shows comparatively little scattering and a maximum of about 0.75 (out of 1), the rewards of the two other systems have larger variations and lower maxima. This is due to the fact that leaving a point with higher rewards entails a higher penalty in these two systems. Because the policy decreases with z, it is difficult for the algorithm to hold the target position in the end of the rollout horizon.

V. OPTIMAL CONTROL

The overarching concept in optimal control is to optimize control inputs and system states according to a specific cost function.

Since there are many different forms of optimal control, in this section, LQR was applied to the linear spring-mass system, iLQR to the pendulum, and MPC to the cart-pole. This highlights the different capabilities of these approaches.

A. Algorithms

The continuous LQR algorithm aims at minimizing the cost function depicted in (17).

$$J_1 = \int_0^\infty \left(x^T Q\, x + u^T R\, u\right) dt, \tag{17}$$

with weight matrices Q and R. Since there was no specifications for input and states, the weights were simply set to $Q = \mathrm{diag}(1, 1)$ and $R = 0.1$. This results in the well known control law $u = -K\, x$ which transforms (1) into

$$\dot{x} = (A - B\, K)x, \tag{18}$$

with the optimal state feedback gain K.

With a goal position different from the origin, a prefilter matrix L as shown in (19) is necessary.

$$L = \left(C(B\, K - A)^{-1} B\right)^{-1} \tag{19}$$

Since LQR is lacking an integral component, a simple PI controller ($K_P = K_I = 1$) was added to drive the system to the desired goal position under disturbances (see fig. 3a).

The iLQR approach for the pendulum was based on [6] and their Matlab toolbox.

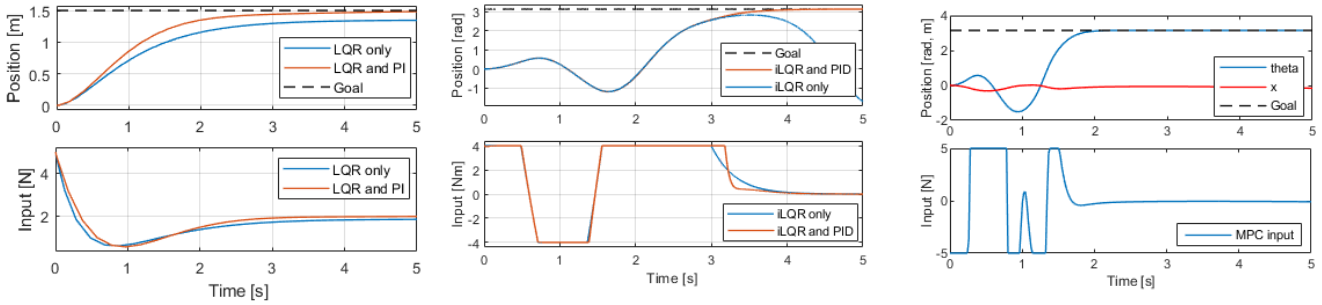
The input was limited to $|u| < 4\,\mathrm{Nm}$, so that the goal can only be reached by swinging back and forth.

The cost function for the pendulum problem is defined as

$$J_2 = \frac{1}{2}\overline{x}_N^T P\, \overline{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(\overline{x}_k^T Q\, \overline{x}_k + u_k^T R\, u_k\right), \tag{20}$$

with the time horizon N, the deviation of the current state from the desired state $\overline{x}_k = (x_k - x_{des})$, the final weight matrix P, and the familiar Q and R matrices.

Without any other specifications, the input weight was set to a rather low value of $R = 10^{-3}$ and $Q = \mathrm{diag}(10^{-3}, 0)$ to allow for necessary swinging and full speed during the run. The final cost was set to $P = \mathrm{diag}(100, 0.1)$ to ensure the upright position. With a time horizon of $N = 5000$ at a $t = 1\,\mathrm{ms}$ sample time, the runtime was 5 seconds.



(a) Comparison of LQR and LQR+PI. Position x_1 of the spring-mass system (top) and control input u (bottom).

(b) Comparison of iLQR result and closed-loop simulation. Angular position x_1 of the pendulum (top) and control input u (bottom).

(c) MPC of the cart-pole. Angle $\theta = x_3$ of the pendulum and position $x = x_1$ of the car (top) and control input u (bottom).

Fig. 3: Simulation results with optimal control for the spring-mass system (a), pendulum (b), and cart-pole (c).

For the implementation, the optimized (open-loop) input trajectory $\underline{u}$ was used as feed forward control, while a PID controller minimized the difference between the optimized (estimated) trajectory $\underline{x}$ and the actual state. Without this setup, even the slightest perturbation of the system (e.g. numerical errors) leads to a failed upswing since the goal position is an unstable equilibrium point (see fig. 3b).

The MPC approach was based on the paper and toolbox from [7] which relies on the Matlab function "fmincon".

The input was constrained to $|u| < 5\,\mathrm{N}$, and the cart position was limited to $|x_1| < 4\,\mathrm{m}$. The cost function is the same as in (20), with $P = \mathrm{diag}(0, 10, 10, 10)$ (end position of the cart is irrelevant as long as it stays within the bounds), $Q = \mathrm{diag}(20, 10, 10, 0)$ (cart should remain in the center during upswing, while angular speed is irrelevant), and $R = 0.01$. The prediction horizon was set to $N = 20$ at a sample time of $t = 20\,\mathrm{ms}$.

Since MPC is a closed-loop control strategy, no additional controller was necessary to stabilize the system (see fig. 3c).

B. Results

For the spring-mass system without disturbances, the goal was reach by the LQR controller. However, with an added constant disturbance of $d = 0.5\,\mathrm{N}$, the PI controller was necessary as shown in figure 3a. Note that using only the PI controller without LQR would render the system unstable.

For the pendulum, the iLQR algorithm converged after 52 iterations. The resulting trajectory requires three swings to reach to goal position due to the input constraints.

Figure 3b displays the stand-alone iLQR trajectory and one with an added PID controller. Both trajectories are almost identical, and there is only a small deviation in the control input at the end of the upswing (compare figure 3b bottom at 3 seconds). However, this slight change is crucial to keep the pendulum in the upright position. Without the added PID controller, the system becomes unstable since iLQR is not a closed-loop controller.

The MPC controller managed to stabilize the pendulum of the cart-pole in the upright position for all initial angles. The successful swing-up for an initial angle of 10° is presented in figure 3c.

VI. Comparison and Discussion

As the results show, optimal control is an effective method as it does not need any interactions with the real system and still delivers high accuracy with reasonable computational effort. However, a system model has to be known or derivable by linearization, and the parameterization of the weights requires some work. For unknown or highly non-linear models this method might fail, which is when the presented reinforcment learning approaches appear to be valid options. The advantage of PILCO is that model learning and simulative parameter optimizations require only few system interactions for stable closed loop control. The optimization algorithm is, however, computationally expensive, since matrix inversions during GP training are needed. By contrast, the calculation of DMP-policies is faster, but requires several system interactions to yield an (open loop) policy. Both RL algorithms show slight deviations around the goal position due to their inherent structure (see figures 1, 2b, 2c).

VII. Conclusion

Summarizing the results, optimal control presents a good starting point for controlling a system when the dynamics are (partially) known. More complex sub-dynamics (e.g. friction) can then be tackled with reinforcement learning. Here, DMP policies appear to be a good choice for repetitive tasks, while PILCO is a strong algorithm for a general control scheme.

References

[1] Deisenroth, "Efficient reinforcement learning using Gaussian processes", KIT Scientific Publishing, vol. 9, 2010.

[2] Deisenroth, Fox, and Rasmussen, "Gaussian processes for data-efficient learning in robotics and control", IEEE transactions on pattern analysis and machine intelligence 37.2 (2015): 408-423.

[3] Deisenroth, Fox, and Rasmussen, "PILCO website", 2013. Available at: http://mlg.eng.cam.ac.uk/pilco/ [Accessed: 27-Jan-2019].

[4] Studywolf, "Dynamic Movement Primitives Part 1: The Basics", 2013. Available at: https://studywolf.wordpress.com/2013/11/16/dynamic-movement-primitives-part-1-the-basics/ [Accessed: 28-Jan-2019].

[5] Kober and Peters, "Policy search for motor primitives in robotics", Advances in Neural Information Processing Systems (2009): 849-856.

[6] Tassa, Mansard, and Todorov, "Control-limited differential dynamic programming", IEEE International Conference on Robotics and Automation (ICRA), 2014.

[7] Gruene and Pannek, "Nonlinear Model Predictive Control", Springer Verlag, 2017.