Comparison of Model-Based and 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. The results show +one sentence about what results show+**

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. Therefore, the aim of this paper is to present and discuss algorithms and results, as well as advantages and drawbacks of different control approaches, namely model-based reinforcement learning (RL), model-free RL, and optimal control. These methods are applied to a spring-mass system, a pendulum, and a cart-pole system.

The algorithms and results for the three approaches are presented in Sec. II, Sec. III, and Sec. IV, while a discussion of the findings is provided in Sec. V and a summarizing conclusion is drawn in Sec. VI.

II. SYSTEMS

The following three models are controlled by the three different control algorithms:

A. Spring Damper Mass

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

$$\ddot{x} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{pmatrix}}_{A} x + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix}}_{B} u, \tag{1}$$

with the parameters $k = 1\,\mathrm{N/m}$ (spring) and $m = 1\,\mathrm{kg}$ (mass), while the output of the system is given as

$$y = \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{C} x. \tag{2}$$

B. Pendulum

The dynamics of the pendulum are described in (3).

$$\dot{x} = \begin{pmatrix} x_2 \\ \frac{u - b\,x_2 - m\,g\,s\sin(x_1)}{m\,s^2} \end{pmatrix}, \tag{3}$$

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

C. Cart Pole

The cart-pole dynamics are as follows:

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

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

D. Simulations

For the comparison of the following algorithms the simulation horizon for all models is 2.5 seconds and the maximum executable force for the spring damper mass and cart pole is 10 N and the maximum torque for the pendulum is 10 Nm. The initial state of spring damper mass (pendulum) is 0 m (0°) and 0 $\frac{m}{s}$ (0$\frac{rad}{s}$). The cart pole starts at the initial angles 10°, 90°and 180°and zero position of the cart.

III. MODEL-BASED RL

In model based reinforcement learning a model of the system to be controlled is learned using measurement data. Secondly a policy is optimized using the learned model in order to control the real world system afterwards.

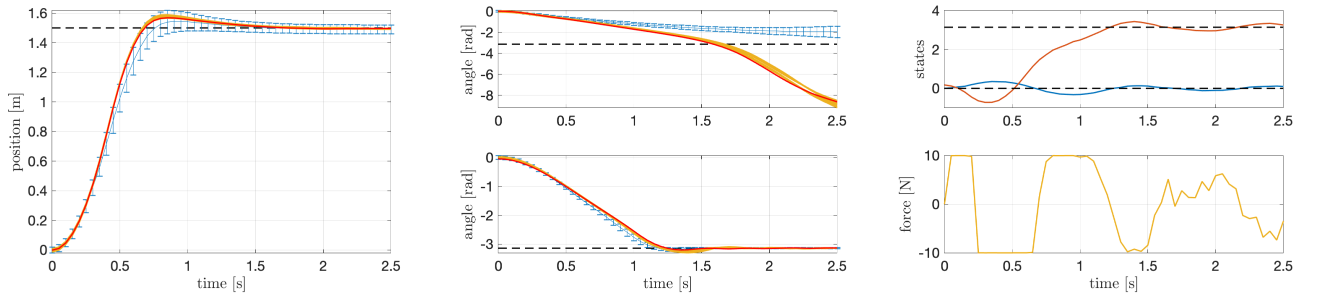
A. Algorithm

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

$$x_t = f\left(x_{t-1}, u_{t-1}\right) \tag{5}$$

where $x \in \mathbb{R}^D$ are the real valued system states, $u \in \mathbb{R}^U$ are the control inputs and f is the unkown transition dynamics. The aim of PILCO is to find a policy π for the unkown system, i.e. the optimal parameters θ of a controller, to minimize a cumulative cost $J^\pi\left(\theta\right)$.

a) Learning Model Dynamics: The learning of the model dynamics described in equation (5) is done using a gaussian process (GP), where $\left(x_{t-1}, u_{t-1}\right) \in \mathbb{R}^{D+F}$ serve as inputs and $\Delta_t = x_t - x_{t-1} + \epsilon \in \mathbb{R}^D$ with $\epsilon \sim \mathcal{N}\left(0, \Sigma_\epsilon\right)$ serve as the GP's outputs. For the GP the squared exponential kernel and a prior mean $m \equiv 0$ is chosen. The parameters (length scales, signal/ noise variance) are



(a) Spring Damper Mass (trail #1)

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

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

Fig. 1: Figures (a)-(b) show trajectories used for training of GP (red), additional rollouts not used for GP training (yellow), predicted system trajectory with confidence interval of 95% (blue) and target states (black).

trained using evidence maximization. Finally the one step predictions based on the GP are:

$$p(x_t \mid x_{t-1}, u_{t-1}) = \mathcal{N}\left(x_t \mid \mu_t, \boldsymbol{\Sigma}_t\right), \tag{6}$$

$$\mu_t = x_{t-1} + \mathbb{E}_f\left[\Delta_t\right], \; \boldsymbol{\Sigma}_t = \text{var}_f\left[\Delta_t\right] \tag{7}$$

Here the mean $\mathbb{E}_f\left[\Delta'\right]$ and the variance $\text{var}_f\left[\Delta'\right]$ 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 $\boldsymbol{\Sigma}_u$ are obtained. Secondly by additionally calculating the cross-covariance $\text{cov}_f = \left[x_{t-1}, u_{t-1}\right]$ the joint state-policy distribution can be approximated by a Gaussian. This result is then used for predicting $p\left(\Delta_t\right)$ with

$$p\left(\Delta_t\right) = \int p\left(f\left(\tilde{x}_{t-1}\right) \mid \tilde{x}_{t-1}\right) p\left(\tilde{x}_{t-1}\right) \mathrm{d}\tilde{x}_{t-1} \tag{8}$$

Often there is no analytically feasible solution for equation (8). Therefore the Gaussian of $p\left(\Delta_t\right)$ is approximated using moment matching or linearization [2]. Wtih the Gaussian of $p\left(\Delta_t\right)$ an approximate Gaussian is obtained for $p\left(x_t\right)$ by

$$\mu_t = \mu_{t-1} + \mu_\Delta, \tag{9}$$

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_{t-1} + \text{cov}\left[x_{t-1}, \Delta_t\right] + \text{cov}\left[\Delta_t, x_{t-1}\right], \tag{10}$$

$$\text{cov}\left[x_{t-1}, \Delta_t\right] = \text{cov}\left[x_{t-1}, u_{t-1}\right] \boldsymbol{\Sigma}_u^{-1} \text{cov}\left[u_{t-1}, \Delta_t\right]. \tag{11}$$

Finally the the expected cost is calculated according to

$$\mathbb{E}\left[c\left(x_t\right)\right] = \int c\left(x_t\right) \mathcal{N}\left(x_t \mid \mu_t, \boldsymbol{\Sigma}_t\right) \mathrm{d}x_t. \tag{12}$$

The cost $c(x)$ considering the target state x_{target} is a 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

In this part the PILCO algorithm is applied to the three previous introduced systems. The simulations have been done using the online available PILCO framework for MATLAB [3]. For the policy a radial basis function (RBF) controller is used.

Figure 1a shows that the spring damper mass system is controlled on trail #1 after optimizing the GP with the data of random exploration in the beginning. This is due to the fact the system is a linear one and the complexity is quite low. The pendulum in figure 1b is controlled after the trial #2. At trail #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 (s. blue errorbars in fig. 1b). Afer trail #2 however, the system is stabilized at $-\pi$ for all following trials. This shows that PILCO can handle unkown nonlinear systems quite effectively. This becomes even more obvious when considering results of the cart-pole trajectories. It takes 7 trials for the swing up from 10°, 3 trails from 90°and 2 trails from 180°. After the swing up the steady state is controlled stable with slight inaccuarcies. Figure 1c shows the swing up from 10 °. Remarkable for this swing up is that the controller uses gravity to let the pole first swing in the other direction and swing it to its target position afterwards. This is due to the restriction of the controller to 10 N maximum force. An linear controller would not be able to plan such a trajectory.

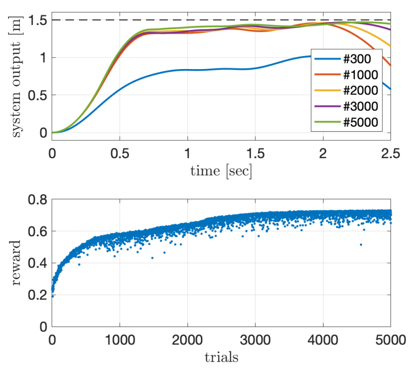
IV. MODEL-FREE POLICY SEARCH

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

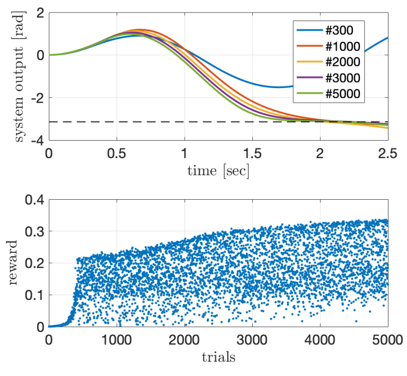
A. Algorithm

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 on the system in this work is based on Dynamic Movement Primitives (DMP)

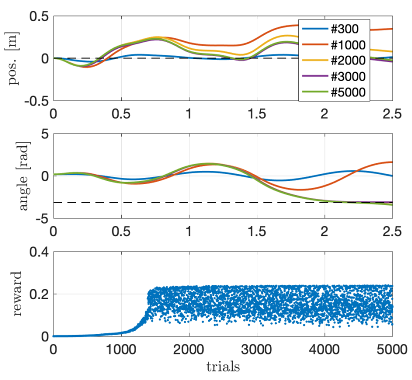
$$\pi_\theta = \ddot{y} = \tau^2 \alpha_y \beta_y (g - y(t) - \tau^2 \alpha_y \dot{y}(t) + \tau^2 f(z(t)) \tag{13}$$



(a) Mass-Spring-Damper



(b) Pendulum



(c) Cartpole, initial angle: 10°

Fig. 2: Figures (a)-(c) show the system output as well as the policy over the time at different iteration steps (300 - 5000 trails).

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

$$f(z(t)) = \frac{\sum_{i=1}^{N} \phi_i \omega_i}{\sum_{i=1}^{N} \phi_i} z \tag{14}$$

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

$$\dot{z} = -\tau\, \alpha_z\, z \tag{15}$$

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

$$\theta_{temp} = \theta_n + \varepsilon \tag{16}$$

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:

$$R(\theta_{temp}) = \sum_{t=0}^{T-1} r_t(x_t) \tag{17}$$

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

$$\theta_{n+1} = \theta_n + \frac{\Delta\theta_1 R(\theta_1) + ... + \Delta\theta_{10} R(\theta_{10})}{R(\theta_1) + ... + R(\theta_{10})} \tag{18}$$

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 the following the results of the three systems mass-spring-damper, pendulum and cartpole 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 mass-spring-damper and the pendulum show a moderated linear improvement; the reward of the cartpole has its significant point at about 1.500 trials. The appearance of the three graphs differ: While the reward of the mass-spring-damper 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 higer rewards, entails a higher penalty in these two systems. Because the policy decreases with z, it is difficult for the alorithm 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.

For linear systems, the method of choice is a linear-quadratic regulator (LQR) that optimizes a linear-quadratic (LQ) cost function. When dealing with non-linear systems, LQR can be iteratively applied to locally LQ problems. This method is called iterative LQR (iLQR). Another method to handle non-linear systems is model-predictive control (MPC). In MPC, an optimization problem to obtain optimal inputs and state trajectories is only solved for a finite time horizon and continually updated during runtime.

For this section, LQR was applied to the linear spring-mass system, iLQR to the pendulum, and MPC to the cart-pole system to highlight the different capabilities of these approaches.

A. Algorithms

The continuous LQR aims at minimizing the cost function depicted in (19).

$$J_1 = \int_0^\infty \left(x^T Q\, x + u^T R\, u\right) dt, \tag{19}$$

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 an optimal input $u = -K\, x$ which transforms (1) into

$$\dot{x} = (A - B\, K)x. \tag{20}$$

For the spring-mass system, K was computed using the Matlab function "lqr".

Since the goal position is not the origin but rather $x = (1.5\ 0)^T$, a prefilter matrix L as shown in (21) is necessary.

$$L = \left(C(B\, K - A)^{-1}B\right)^{-1} \tag{21}$$

Since LQR is lacking an integral component, a simple PI controller ($K_P = K_I = 1$) was added to drive the system into the desired goal position even under disturbances. Note that the system would also be stable for any other controller gain but experience higher overshooting.

The iLQR approach for the pendulum was based on [+add reference for Tassa, Mansard and Todorov, 'Control-Limited Differential Dynamic Programming', ICRA 2014+] and their "iLQG/DDP trajectory optimization" toolbox for Matlab.

To make the problem more interesting, the input is limited to $|u| < 4\,\mathrm{Nm}$, so that the goal cannot be achieved in a single swing but only by moving back and forth.

The iLQR (or the similar iLQG) algorithm is a shooting method. This means that an input trajectory is applied to the system (forward pass), and then the input trajectory is optimized (backwards pass). This process is repeated iteratively, until the input trajectory converges.

The cost function for the pendulum problem is defined as

$$J_2 = \frac{1}{2}\overline{x}_N^T P\, \overline{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(\overline{x}_k^T Q\, \overline{x}_k + u_k^T R\, u_k\right), \tag{22}$$

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 $R = 10^{-3}$.

Even though the desired state is the upright position of the pendulum at zero speed, during the run, only a modest punishment is placed on the correctness of the position to leave enough room for swinging back and forth, and the speed is not punished at all since it is required to drive the pendulum in the desired position. Therefore, this matrix was set to $Q = \mathrm{diag}(10^{-3}, 0)$.

For the final state, the position is the most important factor, while the speed should at least be close to zero, which leads to $P = \mathrm{diag}(100, 0.1)$.

The iLQR algorithm was set to a time horizon of $N = 5000$ at a $t = 1\,\mathrm{ms}$ sample time, which equals a runtime of 5 seconds.

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

The MPC implementation was based on the "nmpc" function from [+cite Nonlinear Model Predictive Control Theory and Algorithms, Grne, Lars, Pannek, Jrgen +]. This function aggregates system dynamics, constraints, cost function, and optimization parameters to solve the optimization problem with 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 (22), 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 $t = 20\,\mathrm{ms}$, and the simulation ran for 4 seconds.

Since MPC is a closed-loop control strategy, no additional controller was necessary.

B. Results

For the spring-mass system without disturbances, the goal was reach by the LQR controller with $K =$. However, with an added constant disturbance of $d = 0.5\,\mathrm{N}$, the LQR controller alone fails to reach the desired position, while the addition of the PI controller solves this issue as shown in figure (3).

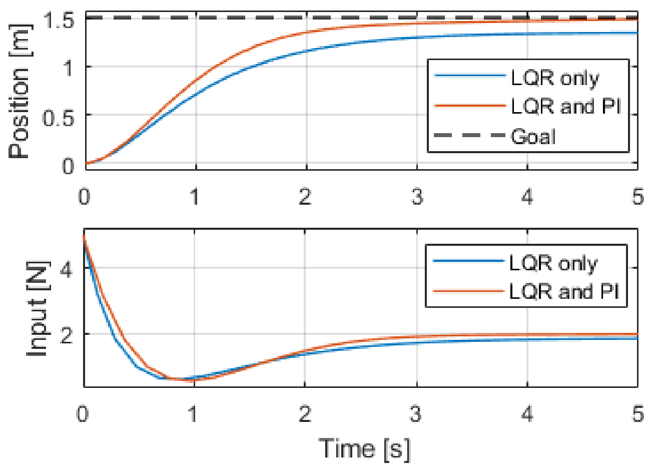


Fig. 3: Comparison of LQR and LQR+PI controller. Position x_1 of the spring-mass system (top) and control input u (bottom).

Note that only using the PI controller is not sufficient and would cause the system to become unstable.

For the pendulum, the iLQR algorithm converged to a cost of 156.70 after 52 iterations. The resulting trajectory is one initial swing to the left, followed by a complete upswing to the right. This trajectory seems reasonable, since reaching the goal with just a single swing is impossible due to the restrictions on the control input. Figure (4) displays the trajectory that the algorithm found and the simulated trajectory with an added PID controller.

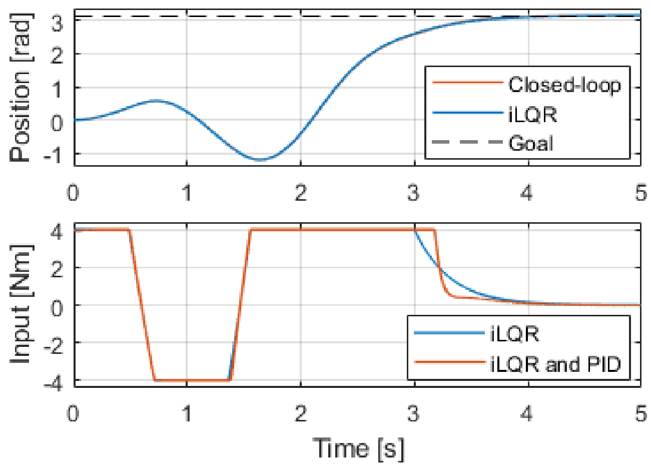


Fig. 4: Comparison of iLQR result and closed-loop simulation. Angular position x_1 of the pendulum (top) and control input u (bottom).

Both trajectories are almost identical, and there is only a small deviation in the control input at the end of the upswing (compare figure (4) 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 remains instable as shown in figure (5), since iLQR is not a closed-loop controller.

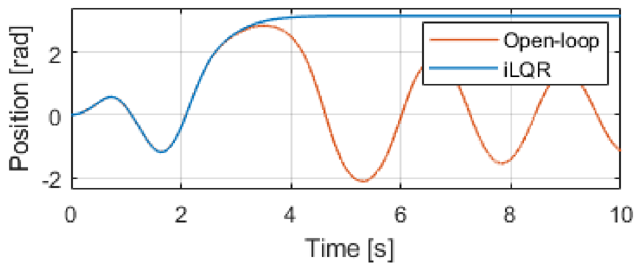


Fig. 5: Comparison of iLQR result and open-loop simulation. Angular position x_1 of the pendulum.

The MPC controller managed to stabilize the pendulum of the cart-pole in the upright position after 135 time steps or 2.7 seconds. Due to the input constraints, the car had to move left and right a few times to allow the pendulum to swing back and forth until it reached the upright position. The successful swing-up is presented in figure (6).

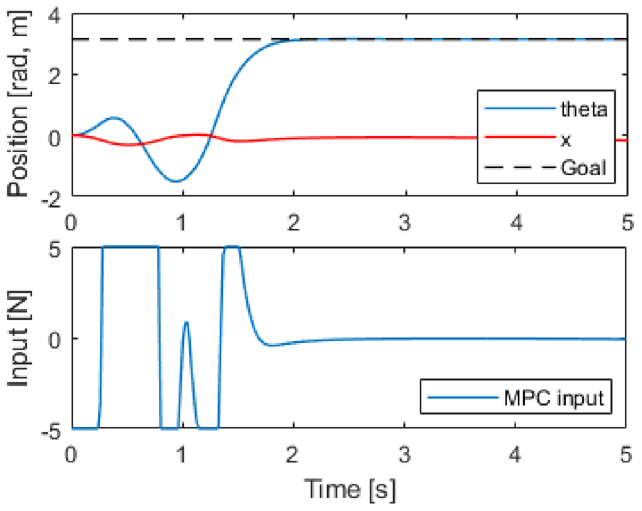


Fig. 6: The swing-up of the cart-pole system. Angular position $\theta = x_3$ of the pendulum and horizontal displacement $x = x_1$ of the car (top) and control input u (bottom).

Since MPC is a closed-loop control strategy, the system remained in the upright position after reaching it initially.

VI. Comparison and Discussion

+TBD+

VII. Conclusion

+TBD+

References

[1] Deisenroth, Marc, and Carl E. Rasmussen. ”PILCO: A model-based and data-efficient approach to policy search.” Proceedings of the 28th International Conference on machine learning (ICML-11). 2011.

[2] Deisenroth, Marc Peter, Dieter Fox, and Carl Edward 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, Marc Peter, Dieter Fox, and Carl Edward Rasmussen. ”PILCO website”, 2013. [Online]. Available: http://mlg.eng.cam.ac.uk/pilco/. [Accessed: 27- Jan- 2019]