Double Inverted Pendulum Swing-Up and Stabilization using MPC and RL

Ali Alouane (397293), Adrian Pfisterer (382140), Jonas Rauch (381293)

I. Introduction

THIS project aims to control a double inverted pendulum using model predictive control (MPC). The starting position is set to the origin of coordinates with both pendulums hanging down. The MPC approach shall be able to, within physical limitations, bring cart and pendulums in any arbitrary position while avoiding given obstacles.

The model of a double pendulum can be used in real life applications for example in the field of robotics. Furthermore it is use-full in understanding fundamental principals of pendulum behaviour, which itself is a widely used model for a variety of systems.

For the problem at hand we will use a direct MPC approach utilizing orthogonal collocation. An essential part of the problem finding suitable cost function.

In addition a reinforcement learning approach is used, namely soft actor critic (SAC). The results of both methods will be compared.

II. Methodology

A. Model

The double inverted pendulum is composed of two poles and one cart which are all connected to each other by three friction less joints as shown in Figure 1.

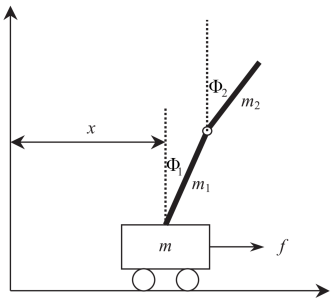


Fig. 1. Double inverted pendulum system [7]

The cart is able to move horizontally in the positive and negative x direction in response to an discreetly applied input force $f = u(t)$. The goal of swinging up the pendulum is assumed to be achieved if both poles are in the up-up position where Φ_1 and Φ_2 are the angles of the first and second pole from the vertical direction respectively. For $\Phi_1 = \Phi_2 = 0$ both poles are in the desired up-up position.

The systems state can be expressed as in 6. Since only the states x and $\dot{x}$ can be controlled the system is considered under actuated. In addition no friction and therefore no internal damping is considered. This represents the worst case scenario for implementing various control strategies [7]. The mathematical system model is derived analogously to Zhong and Roeck [7] under the assumption that the masses of the cart and each pole are concentrated in their centers of gravity respectively.

The derivation of the differential equation describing the model is based on the Euler-Lagrange approach:

$$Q_q = \frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}}\right] - \frac{\partial L}{\partial q} \tag{1}$$

where:

Q_q : external forces acting on the system

$L : T - V$

T : kinetic energy

V : potential energy

$q = [x, \Phi_1, \Phi_2]^T$

according to [7]. These equation allow the system to be described as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = Q_q \tag{2}$$

where:

$$M(q) = \begin{bmatrix} h_1 & h_2 cos\Phi_1 & h_3 cos\Phi_2 \\ h_2 cos\Phi_1 & h_4 & h_5 cos(\Phi_1 - \Phi_2) \\ h_3 cos\Phi_2 & h_5 cos(\Phi_1 - \Phi_2) & h_6 \end{bmatrix}$$

$$C(q,\dot{q}) = \begin{bmatrix} 0 & -h_2\dot{\Phi}_1 sin\Phi_1 & -h_3\dot{\Phi}_2 sin\Phi_2 \\ 0 & 0 & h_5\dot{\Phi}_2 sin(\Phi_1 - \Phi_2) \\ 0 & -h_5\dot{\Phi}_1 sin(\Phi_1 - \Phi_2) & 0 \end{bmatrix}$$

$$q = \begin{bmatrix} x \\ \Phi_1 \\ Phi_2 \end{bmatrix}, \quad g(q) = \begin{bmatrix} 0 \\ -h_7 sin\Phi_1 \\ -h_8 sin\Phi_2 \end{bmatrix}, \quad Q_q = \begin{bmatrix} f \\ 0 \\ 0 \end{bmatrix}. \tag{3}$$

The following terms where used to simplify equations 3:

$$\begin{aligned} h_1 &= m + m_1 + m_2 & h_2 &= m_1 l_1 + m_2 l_2 \\ h_3 &= m_2 l_2 & h_4 &= m_1 l_1^2 + m_2 L_1^2 + J_1 \\ h_5 &= m_2 l_2 L_1 & h_6 &= m_2 l_2^2 + J_2 \\ h_7 &= m_1 l_1 g + m_2 L_1 g & h_8 &= m_2 l_2 g. \end{aligned} \tag{4}$$

Here m_1 and m_2 donate the mass of pole 1 and 2 respectively while m represents the mass of the cart. Using the same indexing pattern L_1 and L_2 are the lengths of each pole and

$l_i = 0.5L_i$ for both poles such that $i \in \{1, 2\}$. The poles inertia is defined as $J_i = m_i L_i^2/12, \quad i \in \{1, 2\}$ and g is the gravitational force.
To obtain the ODEs describing the systems dynamics equation 2 is solved for $\ddot{q}$:

$$\ddot{q} = M(q)^{-1} \left(-C(q, \dot{q})\dot{q} + g(q) + Q_q\right). \tag{5}$$

Since $det(M(q)) > 0$ for all q $M(q)$ is invertible [7]. After computing equation 5 the state space model $\dot{x}_s$ results in:

$$\dot{x}_s = \frac{\partial}{\partial t}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = [x, \Phi_1, \Phi_2, \dot{x}, \dot{\Phi}_1, \dot{\Phi}_2]^T. \tag{6}$$

The model parameters chosen can be found in table I.

TABLE I
MODEL PARAMETERS

Parameter	Symbol	Value
Mass cart	m	0.6 kg
Mass first pole	m_1	0.2 kg
Mass second pole	m_2	0.2 kg
Length first pole	L_1	0.5 m
Length second pole	L_2	0.5 m
Gravitational force	g	9.81 m/s^2

B. MPC Approach

Model predictive control can be described as a form of repeated optimal control (OC). MPC will run a optimal control problem solver for each sampling instant. With k as the kth iteration of the prediction horizon it will calculate the OC inputs for iteration $k + 1$ based on the state calculated in iteration k. This is repeated until a terminal condition is met or the maximum step count is reached.

1) Discretization: The system states must be discretized in order to get a finite state space vector as well as a finite input vector. This will be done using the orthogonal collocation approach, which will approximate the states with interpolation polynomials. This is a full discretization approach, which is well suited for unstable systems and in addition can also handle path constraints [6]. First the Lagrange polynomials and their time derivatives are calculated. Then the collocation equations for each collocation point are evaluated. The differential equation $\dot{x} = f(x, u)$ is substituted by :

$$\sum_{i=0}^{n_{deg}} x_k^i \cdot \frac{\partial l^i(t_j)}{\partial t} = f(x_k^j, u_k^j) \tag{7}$$

$$\text{for } k = 0, \ldots, N-1, j = 0, \ldots, n_{deg} \tag{8}$$

The orthogonal collocation can give better accuracy with less variables compared to the explicit Euler approach [6]. We will not look into direct approaches like single or multiple shooting, since these methods are not well suited for very unstable systems like the present DIP.

2) Cost function: One major part of this project was to find a suitable cost function. An intuitive approach is to use a quadratic set-point tracking objective in the cost function. This was formulated as follows:

$$\begin{aligned} J_{stage} =& w_{stage,phi} \cdot ((x_1 - \phi_{1,target})^2 \\ &+ (x_2 - \phi_{2,target})^2) \\ &+ w_{stage,cart} \cdot (x_c - x_{c,target})^2 \\ J_{terminal} =& w_{terminal,phi} \cdot ((x_1 - \phi_{1,terminal})^2 \\ &+ (x_2 - \phi_{2,terminal})^2) \\ &+ w_{terminal,cart} \cdot (x_c - x_{c,terminal})^2 \end{aligned}$$

The different states have different weights for the stage and terminal cost. For the stage cost we choose higher weights for the angles ($w_{stage,phi}$) and for the terminal cost higher weights for the cart position ($w_{terminal,cart}$). This was done to first erect the pendulum and move it to the target position in the end.

Secondly we implemented a state and energy based cost function. We redefined the state vector by subtracting the target vector.

$$\vec{x}_{new} = \vec{x} - \vec{x}_{target}$$

All further calculations will use this new state vector. Thereby the target state optimization is already integrated into the cost function as stated above. The pendulum is fully erect for pole angles of $2 \cdot \pi \cdot k$ with $k \in \mathbb{Z}$. To reflect this in our cost function we use following equation:

$$J_{phi1,2} = \sin(0.5 \cdot x_1)^2 + \sin(0.5 \cdot x_2)^2$$

We initialize the stage and terminal cost function as follows:

$$J_{s,state} = \vec{x}_{new}^T \cdot Q_s \cdot \vec{x}_{new} + u^T \cdot R \cdot u \tag{9}$$

$$J_{t,state} = \vec{x}_{new}^T \cdot Q_t \cdot \vec{x}_{new} \tag{10}$$

Note that this will not be sensitive towards multiples of $2 \cdot \pi$ for the angles. Therefor we set the angle weights to zero and then define the correct cost with separate weights:

$$\begin{aligned} J_{s,angle} &= w_{s,phi} \cdot J_{phi,1,2} \\ J_{t,angle} &= w_{t,phi} \cdot J_{phi,1,2} \end{aligned}$$

To achieve the energy based cost function we need to calculate the potential as well as the kinetic energy. The potential energy for each pole joint can be calculated as follows:

$$\begin{aligned} E_{kin,1} =& \frac{1}{2} \cdot m \cdot x_3^2 \\ E_{kin,2} =& \frac{1}{2} \cdot m1 \cdot ((x_3 + l_1 \cdot x_4 \cdot \cos(x_1))^2 \\ &+ (l_1 \cdot x_4 \cdot \sin(x_1))^2) + 1/2 \cdot J_1 \cdot x_4^2 \\ E_{kin,3} =& \frac{1}{2} \cdot m_2 \cdot ((x_3 + L_1 \cdot x_4 \cdot \cos(x_1) \\ &+ l_2 \cdot x_5 \cdot \cos(x_2))^2 + (L_1 \cdot x_4 \cdot \sin(x_1) \\ &+ l_2 \cdot x_5 \cdot sin(x_2))^2) + 1/2 \cdot J_2 \cdot x_4^2 \\ E_{kin} =& E_{kin,1} + E_{kin,2} + E_{kin,3} \end{aligned}$$

For the potential energy we get:

$$E_{pot} = m_1 \cdot g \cdot l_1 \cdot \cos(x_1) + m_2 \cdot g \cdot (L_1 \cdot \cos(x_1) + l_2 \cdot \cos(x_2))$$

We get the total energy by adding potential and kinetic energy:

$$E_{total} = E_{kin} + E_{pot} \tag{11}$$

The potential energy in the target state is then:

$$E_{pot,target} = E_{pot}\big|_{\phi_1=\phi_2=0} = m_1 \cdot g \cdot l_1 + m_2 \cdot g \cdot (L_1 + l_2)$$

The kinetic energy shall be zero since we do not want cart movement in the target position. Using this we define the energy based cost, weigh them and finally add them to the cost functions:

$$J_{s,energy} = w_{s,energy} \cdot (E_{target} - E_{ges})^2$$
$$J_{t,energy} = w_{t,energy} \cdot ((E_{target} - E_{pot})^2 + E_{kin}^2)$$

Our total cost function is then:

$$J_s = J_{s,state} + J_{s,energy} + J_{s,angle} \tag{12}$$
$$J_t = J_{t,state} + J_{t,energy} + J_{t,angle} \tag{13}$$

3) Constraints: First we add the collocation and continuity constraints which are needed for the orthogonal collocation. The input is limited to $-4N \leq F \leq 4N$ and is added as constraint as well. The states don't need specific constraints, although choosing to restrictive limits may lead to a infeasible problem. Therefore the state constraints may be chosen arbitrarily, but adequately high. Since we will use circular obstacles, we need to add obstacle constraints in the form of circular equations. Note that here x and y refer to the actual coordinates and do not represent states. Points which lay within the obstacle will meet following equation:

$$(x - x_{obs})^2 + (y - y_{obs})^2 - r_{obs}^2 >= 0$$

For simplicity we will add constraints for the pole joints and for each pole midpoint. Those points are calculated as follows:

$$p_{cart} = x_0$$
$$p_{mid,1} = (x_0 + l1 \cdot \sin(x_1), l1 \cdot \cos(x_1))$$
$$p_{end,1} = (x_0 + L1 \cdot \sin(x_1), L1 \cdot \cos(x_1))$$
$$p_{mid,2} = (p_{end,1,x} + l2 \cdot \sin(x_2), p_{end,1,y} + l2 \cdot \cos(x_2))$$
$$p_{end,2} = (p_{end,1,x} + L2 \cdot \sin(x_2), p_{end,1,y} + L2 \cdot \cos(x_2))$$

Inserting into the circular equation leads to the constraints which need to be added. For each point above we calculate the constraint as follows:

$$g_p = (p_x - x_{obs})^2 + (p_y - y_{obs})^2 - r_{obs}^2$$

4) Target switching: Figure 4 shows the process of state switching. On the left the pendulum is erected in set-point $x_{s,1}$. The trace for the movement to the second set-point $x_{s,2}$ is shown. This visualizes the path the pendulum takes. When $x_{s,1}$ is achieved we need to set the target state to $x_{s,2}$. Therefore we have to be able to detect when the pendulum reaches $x_{s,1}$.

We define target values for phi and the cart position as well as a threshold. Based on this values we will figure which solver is to be used. If following equations are full-filled the system is expected to be in the set-point. With T as threshold:

$$|\sin(0.5 \cdot \phi_1) - \sin(0.5 \cdot \phi_{1,target})| \leq T \tag{14}$$
$$|\sin(0.5 \cdot \phi_2) - \sin(0.5 \cdot \phi_{2,target})| \leq T \tag{15}$$
$$|x_c - x_{c,target}| \leq T \tag{16}$$
$$\tag{17}$$

The accuracy can easily be varied by adjusting the threshold value. If these equations are met the next set-point is pursued or the final state is reached. In our case we want to first change the set-point and if the second set point is reached we want the MPC loop to end.

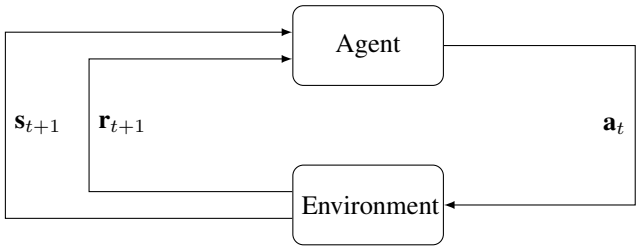
C. Reinforcement Learning Approach

In this section, we used reinforcement learning to build a control system for the stabilisation and swing up maneuver of the double inverted pendulum. In contrast to MPC methods used in section II-B.4, Reinforcement learning (RL) algorithms aim to learn the optimal control strategy using system interaction.

Previous work that uses RL approaches to solve the DIP, show good results for basic balancing as well as for the swing-up problem using probabilistic inference for the learning Control algorithm (PILCO) [4]. However, this method is computationally expensive, when it comes to high-dimensional problems and its a model based approach, that aims to estimate the system dynamic of the environment. In our work, we try to adopt a model free method, that aims to find the best policy with the maximum accumulated reward.

In the robotic field, soft actor critic (SAC) [2] outperforms state-of-the-art model free deep RL methods, such as deep deterministic policy gradient (DDPG) algorithm [5], when it comes to sample efficiency and reward convergence.

1) Environment: In RL, we can formulate the problem as policy search in a fully observed *Markov Decision Process* (MDP), defined by a tuple (S, A, p, r). The figure below II-C.1 shows, how the agent and the environment interact continually on each transition. The agent selects an action $a_t \in A$ and the environment responds to the action with the corresponding reward $r_t : S \times A \rightarrow [r_{min}, r_{max}]$ and moves to the next state s_{t+1} according to the state transition probability $p : S \times S \times A \rightarrow [0, \infty)$. The agent seeks an optimal policy π^* that maximizes the cumulative discounted rewards $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. In our work, the discount factor was set to $\gamma = 0.99$.



In MDP, we used the same *state* space $\mathbf{s} \in \mathbb{R}^6$defined in equation 6. Furthermore, the *action* of the agent corresponds to selecting a continuous force **u** from an certain interval applied to the cart. In our case we used $\mathbf{u} \in [-4N, 4N]$.
Primarily, the *state transition probabilities* are unknown to the agent, but intrinsically given in equation 5.

Lastly, we need to define a suitable reward function for the two different maneuvers. Generally, the reward is negative sum of the different cost function. For balancing, the agent starts a new episode once $|\Phi_1| > \frac{\pi}{2}$, that is, when the first pendulum falls. For swing-up, we penalizes any position of the pendulum system such that :

$$s = \begin{bmatrix} x & \Phi_1 & \Phi_2 \end{bmatrix}^T \neq \begin{bmatrix} 0 & 2\pi n & 2\pi m \end{bmatrix}^T \tag{18}$$

where $n, m \in \mathbb{Z}$. Additionally, we used a modified version of the CartPole-v0 environment in OpenAI Gym [1] to simulate the physical system of the DIP which behaves similarly to the model proposed above. The reason for this are the predefined model wrappers that can be used with popular RL framworks such as PyTorch and Tensorflow.

2) Soft Actor Critic (SAC): The important feature of SAC is entropy regularization. The policy is trained to maximize a trade-off between expected return and entropy. The term of *Entropy* is a measure of randomness in the policy function. Therefore, a high entropy in the policy encourages exploration. The Equation 19 shows the objective function consisting of both a reward term and an entropy term H weighted by a so-called temperature parameter that determines the relative importance of both terms.

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha H(\pi(. \mid s_t)) \right], \tag{19}$$

Generally, actor critic methods, consists of two part:

- The **Critic** uses the state-value function $V_\psi(s_t)$ to estimate the soft value and the Q-function $Q_\Phi(s_t, a_t)$ to estimate the soft action a_t.
- The **Actor** updates the policy distribution $\pi_\phi(a_t \mid s_t)$in the direction suggested by the Critic (with policy gradients)

In our work, we applied the SAC algorithm introduced in [3], that uses two soft Q-functions to speed up training. Primarily, the soft Q-function $Q_\Phi(\mathbf{s}_t, \mathbf{a}_t)$ can be modeled as expressive neural network, and the policy $\pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t)$ as Gaussian with mean and covariance given by the neural network. Next, $Q_\Phi(\mathbf{s}_t, \mathbf{a}_t)$ parameters can be trained to minimize the soft Bellman residual expressed in $J_Q(\Phi)$ by applying the stochastic gradients $\hat{\nabla}_{\Phi_i} J_Q(\Phi_i)$. Finally, the policy parameters can be learned by minimizing the expected KL-divergence expressed in $J_\pi(\phi)$ and update the policy weights by $\lambda_Q \hat{\nabla}_{\Phi_i} J_Q(\Phi_i)$. Additionally, the SAC algorithm updates the temperature of the entropy α by presenting the average entropy of the policy as constrain while optimizing. The used algorithm is listed in 1, where it alternates between collecting experience from the environment with the current policy and updating the approximation functions using stochastic gradient descent from batches sampled from the replay Buffer D.

Algorithm 1: Soft Actor-Critic

Result: Write here the result
Initialize parameter vectors $\psi, \bar{\psi}, \theta, \phi$.
for *each iteration* **do**
  for *each environment step* **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t)$
    $\mathrm{s}_{t+1} \sim p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$
    $\mathrm{D} \leftarrow D \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end
  for *each gradient step* **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)\ for\ i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
  end
end

III. Results

A. MPC

Before focusing on the control tasks of avoiding obstacles and switching between set-points the controller needs to able to swing up both pendulums. This is considered accomplished if the set-point $x_{s,1} = [0, 0, 0, 0, 0, 0]^T$ is reached whilst using an input force of $|u| \leq 4\ N$.
First the cost functions 9 and 10 are utilized to tackle this task. The matrices are chosen as:

$$Q_s = \text{diag}(10, 1, 1, 0.1, 0.1, 0.1)$$
$$Q_t = \text{diag}(100, 1, 1, 0.1, 0.1, 0.1)$$

This is done to penalize the cart for moving to far away from the defined set-point as well as emphasizing the controller to reach the designated angles θ_1 and θ_2. Velocity's are not of a big concern here. Unfortunately the input constraint could not be met when using this quadratic set-point tracking cost function. Only for inputs $|u| \geq 60\ N$ a feasible solution to the optimal control problem is found by the solver. As a consequence energy based terms are included in the cost function as stated in equation 12 and 13. Here the state tracking Terms $J_{s,state}$ and $J_{t,state}$ are used to penalize all velocity's as well as deviations of the cart position from the given set-point. Energy based terms $J_{s,energy}$ and $J_{t,energy}$ are responsible for reaching the designated up-up position. Therefore weights and matrices are chosen as follows:

$$Q_s = \text{diag}(10, 0, 0, 0, 1, 1, 1)$$
$$Q_t = \text{diag}(100, 0, 0, 0, 1, 1, 1)$$
$$w_{s,phi} = 1, \quad w_{t,phi} = 10$$
$$w_{s,energy} = 1, \quad w_{t,energy} = 10$$

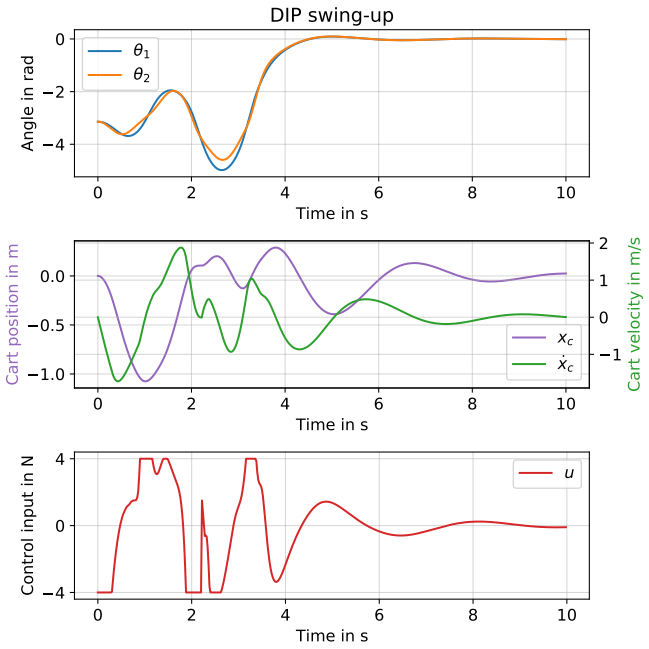


Fig. 2. Simulation of the implemented DIP swing-up MPC controller with prediction horizon $\boldsymbol{N = 50}$ and time step $\boldsymbol{\Delta t = 0.02}$s

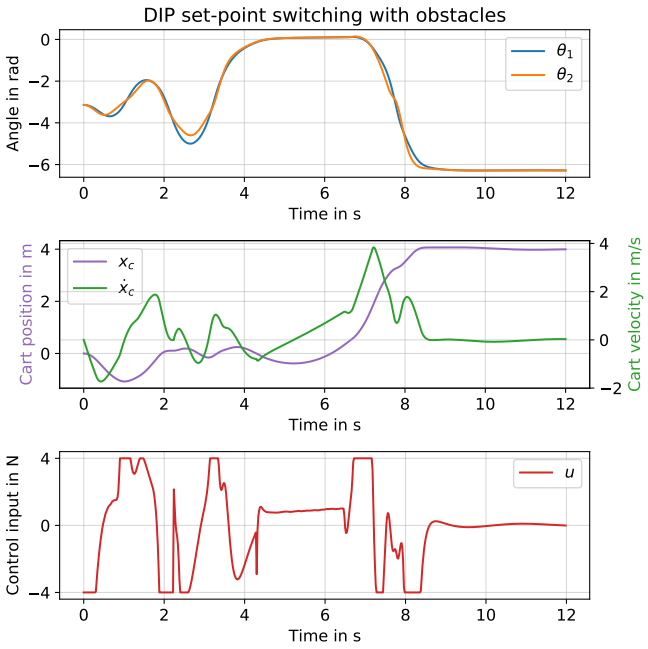


Fig. 3. Controller reaching set-point $\boldsymbol{x_{s,1}}$ and transitioning to set-point $\boldsymbol{x_{s,2}}$ while avoiding obstacles with prediction horizon $\boldsymbol{N = 50}$ and time step $\boldsymbol{\Delta t = 0.02}$s

This change to the cost function enables the MPC controller to swing up the DIP while obeying the given input constraints. In Figure 2 a simulation of the optimal trajectory found by the energy based controller is visualized.

For facilitating the MPC controller to transition between set-points only the state tracking part in the cost function needs to be altered since solely the cart position changes. This is done after the first set-point $x_{s,1}$ is reached as described in II-B.4. The second set-point is chosen as $x_{s,2} = [4, 0, 0, 0, 0, 0]^T$. In addition obstacles (circles) should be avoided while transitioning from $x_{s,1}$ to $x_{s,2}$. Here two obstacles are defined between the two set-points. One circle is placed at $x = 2, y = 1.3$ the other one at $x = 2, y = -1.3$ both circles poses a radius of $r = 0.6$. They are included in the optimal control problem as constraints. The trajectory found by the MPC controller is shown in Figure 3. The DIP's motion after reaching $x_{s,1}$ is additionally visualized in Figure 4.

It can be observed in figure 3 that $x_{s,1}$ is reached after 4.2 seconds and the DIP is now in the up-up position. Afterwards the controller pursues set-point $x_{s,2}$ which is accomplished after 9 seconds. Figure 4 shows that during the transition between set points both obstacles are avoided.

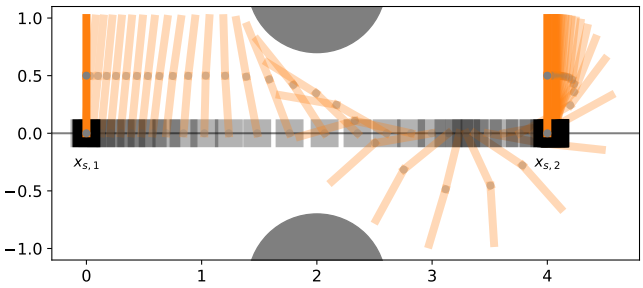


Fig. 4. DIP motion while transitioning from set-point $\boldsymbol{x_{s,1}}$ to set-point $\boldsymbol{x_{s,2}}$

TABLE II
PRE-DEFINED HYPERPARAMETERS FOR THE SAC ALGORITHM

Hyperparameters	Values
γ: Discount factor for the reward	0.99
τ : Target smoothing coefficient	0.005
lr: learning rate	0.003
updates-per-step : model updates per simulator step	0.003
nodes in hidden layer	256

B. RL

In this part, we describe the achieved experiments for both maneuvers. Due to the lack of time, we only varied the values of the entropy α. Otherwise, we set the remaining parameters as follows II:

1) Balancing: In each new episode, the state was initialized as follows:

$$s_0 = \begin{bmatrix} 0 & \theta_{1_0} & \theta_{2_0} & 0 & 0 & 0 \end{bmatrix}^T \tag{20}$$

where $\theta_{1_0}, \theta_{2_0} \sim U\,[-0.1, 0.1]$. In addition, we set the maximum number of steps in a single episode to 200. If the agent reached the limit, the episode was considered as successful.

The figure 5 describes the influence of the entropy α on the convergence of the reward. Thus, applying $\alpha = 0.2$ shows faster convergence in compare to other values. However, the agent takes around 800 episode to reach the maximum accumulated reward. To overcome this problem , tuning other hyperparameters can boost further the reward to reach its maximum.

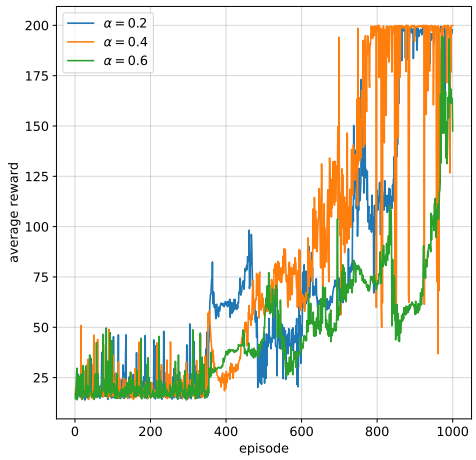


Fig. 5. Typical learning curve for balancing for different entropy values

2) Swing up: In each new episode, the state was initialized as follows:

$$s_0 = \begin{bmatrix} 0 & \theta_{1_0} & \theta_{2_0} & 0 & 0 & 0 \end{bmatrix}^T \tag{21}$$

where $\theta_{1_0}, \theta_{2_0} \sim U[-\pi, \pi]$. We set 200 as maximum number of steps in a single episode.

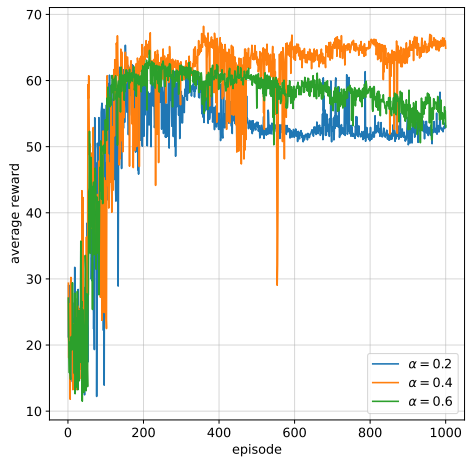


Fig. 6. Typical learning curve for swing-up for different entropy values

Nevertheless, we were not able to swing-up the pendulum neither reach meaningful convergence in reasonable number of episodes. Instead, we observed that the agent was able only to stabilize the first pendulum and didn't try to swing the second one up. However the agent learned successfully to balance both pendulums. A typical learning curve is given in figure 6 above, where *average reward* is total episode reward divided by episode length. It shows the slightly enhancement of the reward function when applying $\alpha = 0.4$. However, even with longer number episodes, the agent shows no improvement. We assume, that the reason behind the unsuccessful results for the swing-up problem, is most likely that we used the same reward function for both problems even when the later is a complex than the balancing maneuver. Additionally, we need to do more fine tuning of SAC hyperparamters and try multiple combinations.

IV. Conclusion

The implemented MPC controller is able to swing-up the DIP, stabilize it and transition between set-points. This is mainly accomplished by utilizing an energy based cost function. Here further optimization of the weights in the cost function is a possible starting point to reach the desired set-point even faster. Especially a parameter sensitivity study can be carried out to optimize the controller. Although a study like this is very time consuming since the *ipopt* solver used is not parallelizable. Therefore the simulation in this project involving set-point transitioning took up to 40 minutes to compute on a AMD Ryzen 5 3600 CPU clocked at 4.3 GHz. In addition the solver is not thread safe which makes running multiple simulations in parallel even harder. A possible approach for reducing the time needed to solve the MPC problem may be to reduce the total number of constraints. For this purpose other discretization approaches should be tested. In addition the constraints responsible for obstacle avoidance can be adapted to use a straight-line equation instead of multiple points.
We found that appropriate values for the prediction horizon and the time step are necessary for the existence of a region of attraction. Even slight changes to both parameters may result in an unfeasible solution of the optimal control problem. This is mainly caused by the DIP's chaotic behaviour. Likewise different discretization schemes can be evaluated to cope with this sensibility.
Regarding the RL approach a lot of potential for improvements exists. Although we were able to stabilize the pendulum adjustments to the cost function are necessary to achieve comparable results to the MPC controller proposed.

References

[1] Openai gym. `https://gym.openai.com/`.
[2] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.
[3] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.
[4] Michael Hesse, Julia Timmermann, Eyke Hüllermeier, and Ansgar Trächtler. A reinforcement learning strategy for the swing-up of the double pendulum on a cart. *Procedia Manufacturing*, 24:15–20, 01 2018.
[5] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.
[6] Prof. Sergio Lucia. Model predictive control: Lecture 5: Discretization techniques for nonlinear model predictive control. 2020.
[7] Wei Zhong and H. Rock. Energy and passivity based control of the double inverted pendulum on a cart. In *Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01) (Cat. No.01CH37204)*, pages 896–901, 2001.