NMPC for double inverted pendulum

Enni Mattern, Marvin Trapp, Jan Nouruzi-Pur, Welf Rehberg

***Abstract*—This paper describes a control algorithm for the double inverted pendulum, based on the concept of model predictive control (MPC). Due to it's non-linearity and instability in the upright position, a full-on nonlinear model predictive controller without linearization has been implemented. The controller design consists of two aspects. First the pendulum has to be raised from the initial hanging position. Second the Stabilization process has to be completed at a defined target position. Therefore a new target positions gets defined and has to be reached whilst keeping track of the given path constraints. The nonlinear optimization problem is solved via the CasADi library.**
The implemented controller acts according to the required behavior as the swing-up can be achieved from the initial hanging position and the repositioning does not violate the constraints.

***Index Terms*—double inverted pendulum, nonlinear model predictive control, CasADi, orthogonal collocation on finite elements, unstable equilibrium point, path constraints**

I. Introduction and Motivation

The stabilization of the double inverted pendulum (DIP) is a widely used reference task for the applicability of different controller designs on nonlinear systems. Therefore several other research papers present different approaches to the stabilization of the DIP, e.g. [Qian et al., 2011], [Zhong and Röck, 2001] and [Moysis, 2016]. In this work a model based predictive controller will be utilized to stabilize and relocate the DIP.
The nonlinear differential equations of motion result from the overall structure of the DIP, which is further described in section II-B. Additionally the input and path constraints underline the nonlinear system behavior and inhibit the usage of controller designs based on linear time invariant system theory like the linear quadratic regulator or PID and state space approaches. Thus a nonlinear model predictive control is implemented, as it does not require any form of linearization of the system dynamics and deals well with state and input constraints.
The non-linearity further leads to a nonlinear optimization problem, which is solved with the help of CasADi. CasADi is an open source software library, implementable in MATLAB, octave or python. It provides tools to solve nonlinear or quadratic programs and interfaces to several optimization problems solvers like IPOPT or CPLEX [Anderson et al., 2019]. The necessary discretization of the optimization problem is derived with a discretization scheme called orthogonal collocation on finite elements.
This paper is structured as followed: Section II provides a general overview of the implemented software architecture, further analysis of the system model equations and a description of the cost function, discretization scheme and the implementation of the path constraints. In section III the results of two test cases are presented and evaluated. A final summary and the conclusion are given in section IV.

II. Methodology

A. Software architecture

The code was developed in a generic manner to maintain an overview during the development process and for reusability purposes. Therefore we structured functionalities in three different modules and a main module that acts as a interface for the user to work with these modules. Figure 1 visualizes the modules and their relation by arrows, which indicate where the modules are imported.

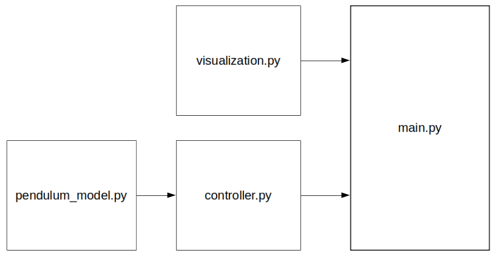

Fig. 1. Software architecture, consisting of modules and their relation to each other

From an algorithmic point of view the core implementation can be found in $controller.py$. Here the MPC-loop and the underlying optimization problem are implemented. This module imports $pendulum\_model.py$, which defines system-related equations. The $controller.py$ defines the controller as a python-class. This allows for easy instantiation of controller-objects in $main.py$, where hyperparameters, controller objectives and state constraints can be specified.

Beside MPC-related algorithms, utilities to visualize and animate system behaviour and predictions are defined in $visualization.py$.

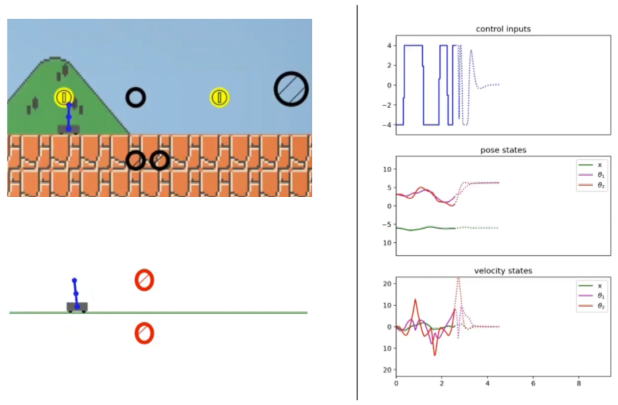

Fig. 2. Animation of the control problem. The visualization module can animate state sequences in two different styles (left) and corresponding plots with past, present and predicted states and control inputs (right)

Figure 2 shows how the implemented animation looks like. It is capable of animating a state sequence with the respective state constraints. A graph view shows past with current states and control inputs additional to the predictions that are computed at every time step.

B. Model

The setup of the double inverted pendulum system is shown in figure 3. It consists of two joints and two pendulums that are mounted on

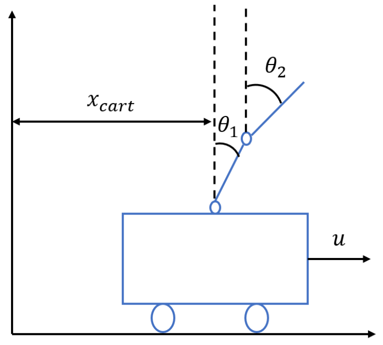

Fig. 3. Setup of the double inverted pendulum system

a cart. The cart can be moved left to right in one dimension. To do so, a force u that is applied to the cart acts as an input to the system. This poses a challenge as the double inverted pendulum is an underactuated system with three degrees of freedom and only one control input.

The system has four equilibrium points where the two pendulums are in up-up, down-down, up-down or down-up position. The goal of the swing-up controller is to bring the pendulum from the stable down-down position into the upright position. Following parameters are used for modelling the system:

- mass of the cart: $m_0 = 0.6kg$
- mass of the two pendulums: $m_1 = m_2 = 0.2kg$
- length of the two pendulums: $L_1 = L_2 = 0.5m$
- distance from a joint to the pendulum link center of mass: $l_1 = l_2 = 0.25m$

As a first step, the state vector θ is introduced with the generalized coordinates shown in figure 3, where x_{cart} is the car's displacement and θ_1 and θ_2 are the respective angles of each pendulum.

$$\theta = [x_{cart}, \theta_1, \theta_2]^T \tag{1}$$

With this states vector, [Bogdanov, 2004] uses the Lagrange mechanics principle to establish a mathematical model of the system, while friction is neglected:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = Q \tag{2}$$

Where L is the Lagrangian and Q the generalized forces acting on the system. The Lagrangian is the difference between the kinetic and potential energies:

$$L = E_{kin} - E_{pot} \tag{3}$$

The energies can be written as the sum of the energies of the cart and the two pendulums:

$$E_{kin} = E_{kin0} + E_{kin1} + E_{kin2} \tag{4}$$
$$E_{pot} = E_{pot0} + E_{pot1} + E_{pot2} \tag{5}$$

Where the moments of inertia of the pendulum links, with respect to its center of mass, I_i are given by:

$$I_i = \frac{m_i L_i^2}{3} \tag{6}$$

for $i = 1, 2$ and the gravitation constant g, the energies of the individual components are:

$$\begin{aligned}
E_{kin0} &= \frac{1}{2} m_0 \dot{x}_{cart}^2 \\
E_{kin1} &= \frac{1}{2} m_1 \dot{x}_{cart}^2 + \frac{1}{2}(m_1 l_1^2 + I_1)\dot{\theta}_1^2 \\
&\quad + m_1 l_1 \dot{x}_{cart}\dot{\theta}_1 \cos\theta_1 \\
E_{kin2} &= \frac{1}{2} m_2 \dot{x}_{cart}^2 + \frac{1}{2} m_2 L_1^2 \dot{\theta}_1^2 + \frac{1}{2}(m_2 l_2^2 + I_2)\dot{\theta}_2^2 \\
&\quad + m_2 L_1 \dot{x}_{cart}\dot{\theta}_1 \cos\theta_1 + m_2 l_2 \dot{x}_{cart}\dot{\theta}_2 \cos\theta_2 \\
&\quad + m_2 L_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \\
E_{pot0} &= 0 \\
E_{pot1} &= m_1 g l_1 \cos\theta_1 \\
E_{pot2} &= m_2 g (L_1 \cos\theta_1 + l_2 \cos\theta_2)
\end{aligned} \tag{7}$$

Written in matrix form the system equation is:

$$D\ddot{\theta} + C\dot{\theta} + G = Hu \tag{8}$$

where

$$D = \begin{pmatrix} d_1 & d_2\cos\theta_1 & d_3\cos\theta_2 \\ d_2\cos\theta_1 & d_4 & d_5\cos(\theta_1-\theta_2) \\ d_3\cos\theta_2 & d_5\cos(\theta_1-\theta_2) & d_6 \end{pmatrix} \tag{9}$$

$$C = \begin{pmatrix} 0 & -d_2\sin\theta_1\dot{\theta}_1 & -d_3\sin\theta_2\dot{\theta}_2 \\ 0 & 0 & d_5\sin(\theta_1-\theta_2)\dot{\theta}_2 \\ 0 & -d_5\sin(\theta_1-\theta_2)\dot{\theta}_1 & 0 \end{pmatrix} \tag{10}$$

$$G = \begin{pmatrix} 0 \\ -f_1\sin\theta_1 \\ f_2\sin\theta_2 \end{pmatrix} \tag{11}$$

$$H = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T \tag{12}$$

The elements of D, C, G are given by:

$$\begin{aligned}
d_1 &= m_0 + m_1 + m_2 \\
d_2 &= (\frac{1}{2}m_1 + m_2)L_1 \\
d_3 &= \frac{1}{2}m_2 L_2 \\
d_4 &= (\frac{1}{3}m_1 + m_2)L_1^2 \\
d_5 &= \frac{1}{2}m_2 L_1 L_2 \\
d_6 &= \frac{1}{3}m_2 L_2^2 \\
f_1 &= (\frac{1}{2}m_1 + m_2)L_1 g \\
f_2 &= \frac{1}{2}m_2 L_2 g
\end{aligned} \tag{13}$$

To find a state representation in the form of

$$\dot{x} = f(x, u) \tag{14}$$

we introduce a new state vector x:

$$x = [x_{cart}, \theta_1, \theta_2, \dot{x}_{cart}, \dot{\theta}_1, \dot{\theta}_2]^T \tag{15}$$

With that state vector, the Lagrange equations can be reformulated into a 6th-order system of ordinary differential equations:

$$\dot{x} = \begin{pmatrix} 0 & I \\ 0 & -D^{-1}C \end{pmatrix} x + \begin{pmatrix} 0 \\ -D^{-1}G \end{pmatrix} + \begin{pmatrix} 0 \\ D^{-1}H \end{pmatrix} u \tag{16}$$

Where I is the identity matrix and D, C, G and H the system matrices.

C. Algorithm

1) MPC loop and optimization problem: On the highest level of abstraction from an algorithmic point of view, the mpc loop is executed. The implemented loop can be described as following:

- solve the optimization problem using the nonlinear solver ipopt[1] through the interface provided by CasADI
- simulate the system with the first obtained control input
- update the initial state for the optimization problem

These three steps loop either for an fixed number of iteration steps or until the current velocity states fall under a threshold for a certain time. As the controlled variable of the system is directly connected to the velocity of the system, the system can be considered as stable, if the velocity remains close to zero for a respected time frame. Therefore the Frobenius norm of the final ten velocity values will be evaluated at each iteration. If the norm falls below a certain threshold the system is considered as stable. The Forbenius norm is defined as:

$$\|A\|_F := \sqrt{\sum_{i=1}^{m} |a_m|^2}, \text{ for each velocity-vector element } m. \quad (17)$$

The threshold is defined as 1.5 because it results in a reliable stability detection in this application. An adjustment of the threshold should be considered if the specifications of the optimization problem changes. To continuously obtain reasonable solutions in more complex cases and to speed up the computation it was necessary to warm start the optimization problem with the optimization solution we obtained in the previous iteration.

To determine an optimal input sequence $u(t)$ we consider the following continuous-time optimal control problem for the time interval $t \in [t_0, t_F]$:

$$\begin{aligned}
\underset{x(t),u(t)}{\text{minimize}} \quad & \int_{t=t_0}^{t_F} J_{stage} + J_{terminal} && (18)\\
\text{subject to} \quad & \dot{x}(t) = f(x(t), u(t)),\ x(0) = x_0 && (19)\\
& u(t) \in [-4N, 4N],\ x(t) \in \mathbb{X} \setminus \mathbb{X}_o && (20)\\
& x(t_F) \in \mathbb{X}_F \setminus \mathbb{X}_o && (21)\\
& t \in [t_0, t_F] && (22)
\end{aligned}$$

Where (18) is a generalized formulation for the objective to minimize the costs, consisting of the stage costs and the terminal cost. This is subject to the dynamic constraint (19), given by the system representation in (8). The input and state constraints are given in (20), considering that the states cannot be part of a subset $\mathbb{X}_o$ that describes the physical limitations created by the obstacles.

2) Cost function: In the following chapter the underlying cost function and its derivation is described. The main idea is to minimize the difference between the declared target pose and the actual pose of the system.

The complete cost function consists of two parts, which are the sum of the stage costs and the terminal cost. Where the stage costs penalize the state of the system at each time step and the terminal cost penalizes the system state at the end of the prediction horizon. Considering that the chosen state vector of the system consists of positions and velocities the cost function is able to minimize the difference between the target state vector and the actual one. It is desirable to distinguish what to penalize in the stage cost and the terminal cost.

[1]https://github.com/coin-or/Ipopt

For example it is not necessary to minimize the velocities of the pendulum during the swing up, but to ensure that the pendulum is positioned stable, at the end of the given time horizon. Therefore they have to be considered in the terminal cost. In addition each individual component is provided with a parameter to flexible weight the different terms. These parameters are represented by *Q1-Q3*, *R* for the stage costs and by *P1-P4* for the terminal cost.

There are two basic approaches to achieve equality between the target state and the actual state in the described system. On the one hand to directly penalize the position difference, on the other hand the energy based approach, described in [Zhong and Röck, 2001]. Thereby the potential and kinetic energy terms are used to determine whether the system is in the desired position or not. To further ensure that the designed controller avoids high frequency input fluctuations, a penalty on the input is included in the stage costs as well.

Quadratic approach: For the direct approach, in comparison to a general linear quadratic regulator, it is important to assure that the alternating nature of angular positions is considered. This can be achieved by minimizing the sinus of the position difference with a factor. Since the system has to be in rest position to be considered as stable, in the terminal cost all velocities are simply minimized as well. The resulting stage cost and terminal cost functions have the following form.

Stage cost:

$$\begin{aligned}&Q1 \cdot (x_{cart} - x_{targ})^2 + Q2 \cdot \sin(0.5 \cdot (\theta_1 - \theta_{1,targ}))^2 \\ &+ Q3 \cdot sin(0.5 \cdot (\theta_2 - \theta_{2,targ}))^2 + R \cdot u^2\end{aligned} \quad (23)$$

Terminal cost:

$$\begin{aligned}&P1 \cdot (x_{cart} - x_{targ})^2 + P2 \cdot \sin(0.5 \cdot (\theta_1 - \theta_{1,targ}))^2 \\ &+ P3 \cdot sin(0.5 \cdot (\theta_2 - \theta_{2,targ}))^2 \\ &+ P4 \cdot (\dot{x_{cart}}^2 + \dot{\theta_1}^2 + \dot{\theta_2}^2)\end{aligned} \quad (24)$$

Energy based approach: The second approach makes use of two of the in chapter II B mentioned equilibrium points of the double pendulum. Which can be expressed by the minimization and maximization of the former described potential (E_{pot}) and kinetic energy (E_{kin}) terms. To reach the upright position, the stage cost function maximizes the overall potential energy of the system. Whereas the terminal cost minimizes its kinetic energy.

Stage cost:

$$Q1 \cdot (x_{cart} - x_{targ})^2 + Q2 \cdot (-E_{pot}) + R \cdot u^2 \quad (25)$$

Terminal cost:

$$P1 \cdot (x_{cart} - x_{targ})^2 + P2 \cdot (-E_{pot}) + P3 \cdot E_{kin} \quad (26)$$

A simulation of the mpc-Loop leads to the conclusion that the energy based approach clearly outperforms the quadratic approach for the defined control targets due to faster convergence.

3) OC: To design a control law an appropriate discretization scheme is necessary. Possible methods are multiple shooting and orthogonal collocation on finite elements, as both can deal well with path constraints and unstable systems [Biegler, 2014]. However orthogonal collocation was used due to its easier implementation.

The states are now approximated using collocation at colloctaion points t_j:

$$x(t) \approx x_k(t) := \sum_{j=0}^{n_{deg}} x_k^j l^j(t),\ l^i(t) = \prod_{\substack{j=0 \\ j \neq i}}^{n_{deg}} \frac{t - t_j}{t_i - t_j} \tag{27}$$

This leads us to a discrete time formulation for a prediction horizon N:

$$\underset{x(t),u(t)}{\text{minimize}} \quad \sum_{i=0}^{N-1} J_{stage} + J_{terminal} \tag{28}$$

$$\text{subject to} \quad \sum_{i=0}^{n_{deg}} x_k^i \frac{\delta l^i(t_k^j)}{\delta t} = f(x_k^j, u(k)), x(0) = x_0 \tag{29}$$

$$\forall k = 0, ..., N-1, j = 0, ..., n_{deg} \tag{30}$$

$$u(k) \in [-4N, 4N],\ x(k) \in \mathbb{X} \setminus \mathbb{X}_o \tag{31}$$

$$x(k = N) \in \mathbb{X}_F \setminus \mathbb{X}_o \tag{32}$$

Where the differential equation $\dot{x} = f(x, u)$ was substituted by the collocation constraints seen in (29). For the collocation points we chose a number of three Legendre-Gauss points which results in a polynomial degree of $n_{deg} = 2$.

4) Obstacles: It was convenient to define the obstacles in a way that incorporating them into the optimization problem is straightforward. These obstacles are denoted in (31) and (32) as $\mathbb{X}_o$. Implementing them in form of circle inequality equations is an appropriate form to do so. The following equation shows the exact constraint that was used:

$$(x - x_{center})^2 + (y - y_{center})^2 < r^2 \tag{33}$$

The continuous differentiability property of this equation eases the incorporation of the equation into the solver. If an point (x, y) satisfies this constraint, it lies outside an circle with coordinates (x_{center}, y_{center}) and radius r. As an approximation, only the endpoints of the two pendulum-rods were constrained in this way. Figure 4 visualizes the inequality equation. This showed to be sufficient in the implemented scenarios. However, improving this approximation by restricting more points on the rods to follow the constraint can be done easily.

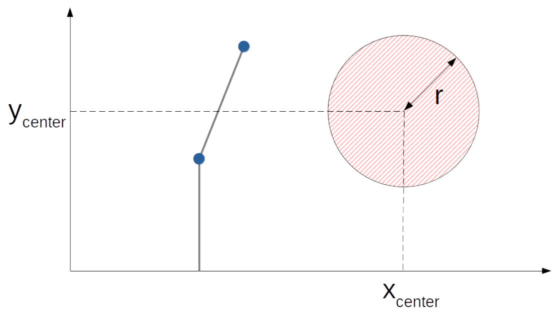

Fig. 4. The constraint prevents defined points (here marked in blue) to enter restricted circle areas.

III. Results and Evaluation

1) Scenario One: The first test scenario intends to show the general applicability of the designed model predictive control algorithm. Therefore two target positions and path constraints between them have been defined. Figure 5 gives a general impression of the implemented target positions and the path constraints between them. The red areas should be strictly avoided by the controller as they represent the path constraints. The utilised parameter configuration of the controller is shown in table I. The definition of the hyperparameters was conducted via try and error. The objectives of the first scenario are defined as following:

1) Swing up and stabilise the pendulum in such a way that it comes to rest at target position 1
2) Transfer of the pendulum from target position 1 to target position 2 under consideration of the path restrictions and re-stabilization

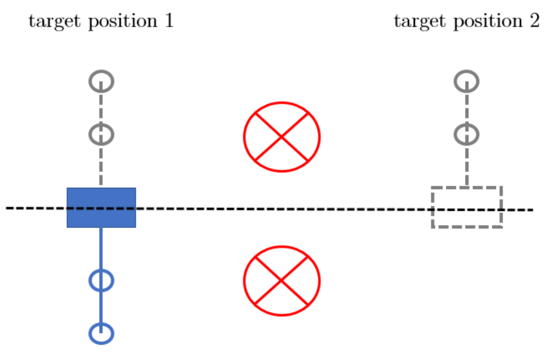

Fig. 5. Obstacle setup of the first scenario

TABLE I
Parameter configuration of the cost function utilised in Scenario One

Parameter	Value
Q1	0
Q2	1
Q3	1
R	0.001
P1	5
P2	1
P3	1
Prediction horizon	50
Initial States Vector	$[-6.0, \pi, \pi, 0, 0, 0]$

Figure 6 shows the simulated system behavior at three different phases. Phase 1 shows the DIP after the completed swing-up and stabilization phase. Phase 2 shows the pendulum while passing through the area with path constraints. Phase 3 shows the System after the second stabilization phase. The state, input and cost function trajectories are presented in Figure 7. The transition between both controller instances is taken after 98 simulation steps.

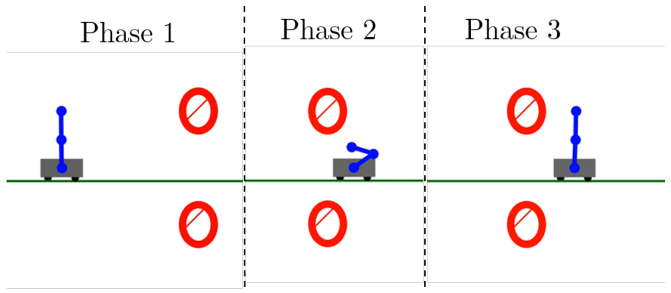

Fig. 6. Cart at relevant positions during scenario one

The computation time highly differs probably due on the amount of restricting path constraints and further possible side effects. Most iteration of the optimization problem take less than 500 ms of computation time. However there are instances with a required computation time of over 1 s up to 46 s as the worst case even with the utilization of the warmstart.

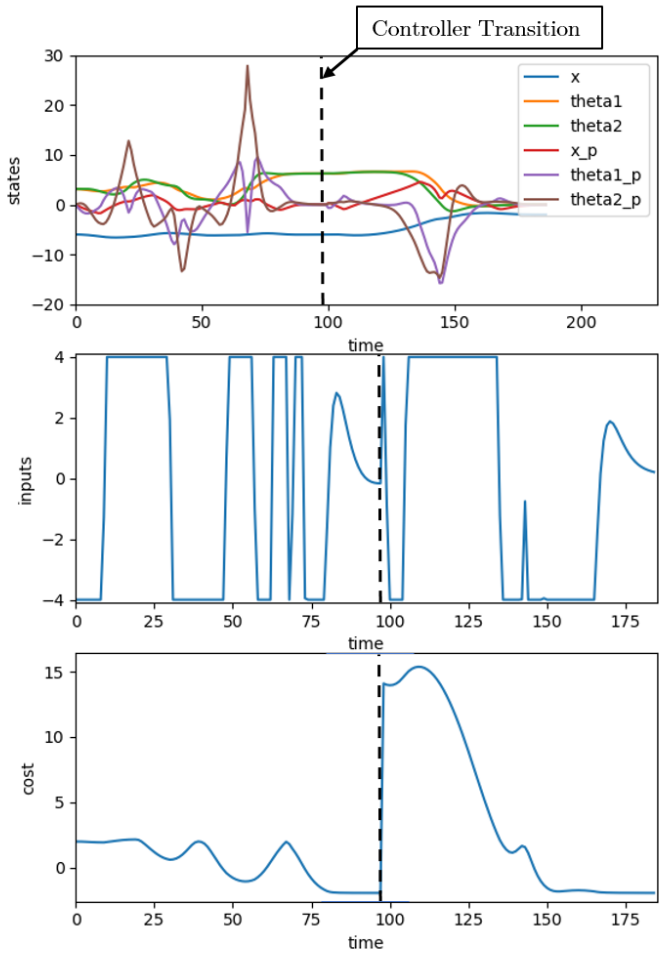

Fig. 7. State, input and cost trajectories for scenario 1

2) Scenario Two: For the second scenario a complicated multistage problem was chosen. The obstacle course consists of two sub problems, each of which is similar to the setup of scenario one. It contains upper and an lower restricted areas in different constellations. The in chapter II C 1 described transition conditions control the point in time at which the different controllers start to operate. In Figure 8 the chosen obstacle setup for the second scenario is shown. The red circles again define the areas which the pendulum has to avoid.

As for the first scenario different objectives are defined for the sections of the main problem.

1) Swing up and stabilization of the pendulum in the initial position
2) Minimize the deviation between actual and target position, including again stabilization at the chosen target.
3) Repetition of objective two for the new target position.

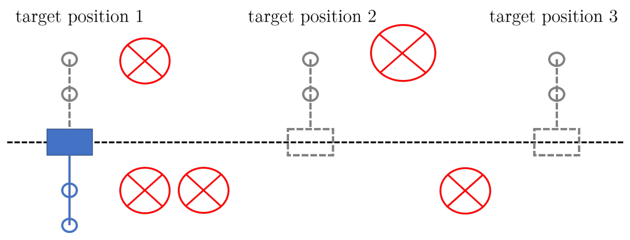

Fig. 8. Obstacle setup of the second scenario

In order to minimize the time which is needed to pass all chosen obstacles and to improve general performance each controller is initialized with a different set of parameters. The parameters used are summarized in table II.

TABLE II
PARAMETER CONFIGURATION OF THE COST FUNCTION UTILISED IN SCENARIO TWO

Parameter	Value Cont. 1	Value Cont. 2	Value Cont. 3
Q1	0	0	0
Q2	1	0.5	1
Q3	0	0.8	0.8
R	0.001	0.001	0.001
P1	1	1	1
P2	1	1	1
P3	1	1	1
Prediction horizon	30		
Initial States Vector	$[-6.5, \pi, \pi, 0, 0, 0]$		

The obtained state values, optimal inputs and costs are shown in Figure 9. The vertical dashed lines mark the transition between the different objectives. The plotted curves show that, with the chosen transition condition and controller parameter, it was quite possible to minimize the time taken by the controller to overcome the obstacle course.

IV. CONCLUSION

The presented nonlinear model predictive controller was designed based on the kinetic and potential energy terms of the system. As presented in section III the simulated behavior shows that the controller can clearly achieve the desired behavior. The controller is able to stabilize the pendulum from any given initial starting position and performs transitions between different target positions without violating path constraints. Therefore the parameter configuration of each controller entity and it's cost function has to be adapted depending on the control problem. In addition the usage of the predicted system behavior to warmstart the optimization solver was crucial to get reliable results. As the computational time for certain optimization problems takes comparatively long and the measurement noise is neglected, the implementation in a real systems has to be further examined. The same applies for model uncertainties, which are also not considered.

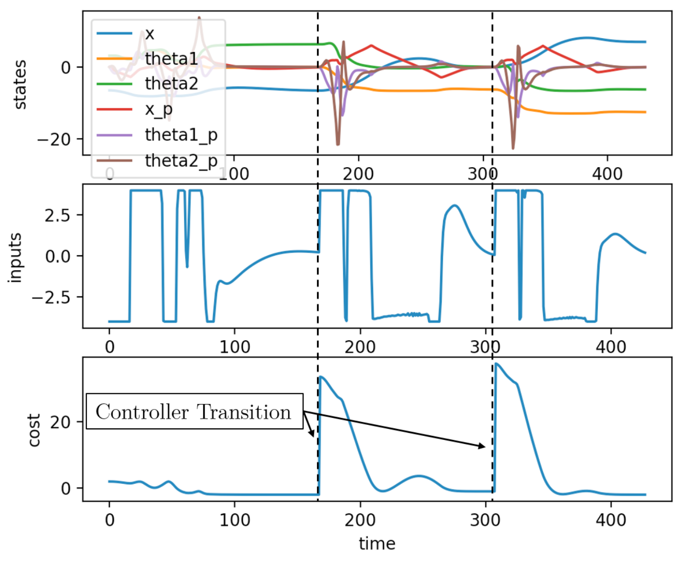

Fig. 9. State values, inputs and costs of the second scenario

References

[Anderson et al., 2019] ANDERSSON, J.A.E., GILLIS, J., HON, G., *et al.*, 2019. CasADi: a software framework for nonlinear optimization and optimal control. In: *Math. Prog. Comp.* 11, 1–36. https://doi.org/10.1007/s12532-018-0139-4

[Biegler, 2014] BIEGLER, L, 2014. Nonlinear Programming Strategies for Dynamic Chemical Process Optimization. In: *Theo. Found. Chem. Eng.* 48, 541–554. doi: https://doi.org/10.1134/S0040579514050157

[Bogdanov, 2004] BOGDANOV, A., 2004. Optimal control of a double inverted pendulum on a cart. Oregon Health and Science University, Tech. Rep. CSE-04-006, OGI School of Science and Engineering, Beaverton, OR, 2004

[Moysis, 2016] MOYSIS, Lazaros, 2016. Balancing a double inverted pendulum using optimal control and Laguerre functions, DOI: 10.13140/RG.2.1.2948.6486

[Qian et al., 2011] QIAN, Q., DONGMEI, D., FENG, L. and YONGCHUAN, T., 2011. Stabilization of the double inverted pendulum based on discrete-time model predictive control. In: *2011 IEEE International Conference on Automation and Logistics (ICAL)*, Chongqing, pp. 243-247, doi: 10.1109/ICAL.2011.6024721

[Zhong and Röck, 2001] ZHONG, Wei and ROCK, H., 2001. 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)*, Mexico City, Mexico, pp. 896-901, doi: 10.1109/CCA.2001.973983