Optimal control of Furuta pendulum

Minh Quang Nguyen
M.Sc. Embedded Systems Engineering
Albert-Ludwig University Freiburg
MatrNr. 5366992

Rean Clive Fernandes
M.Sc. Embedded Systems Engineering
Albert-Ludwig University Freiburg
MatrNr. 5250057

***Abstract*—This paper investigates the implementation of optimal control on a simulated Furuta pendulum, using Model Predictive Control based on different methodologies namely, Direct Multiple Shooting, Direct Collocation and Real-Time Iterative with CasADi [1] on MATLAB for optimization.**

***Index Terms*—Furuta Pendulum, Model Predictive Control, Direct Multiple Shooting, Direct Collocation, Real-Time Iteration, CasADi, IPOPT, qpOASES**

I. Introduction

The inverted pendulum is considered as the standard base for implementation and demonstration of control laws. The Furuta pendulum is an instance of inverted pendulum. Unlike a cart-pole model of an inverted pendulum, the Furuta pendulum uses the rotary arm to swing up the pendulum. In this project, we implement the swing-up control and test its stability with simulation using methodologies that we learned in the Numerical Optimal Control course.

The structure of the report is as follows: In section II, we describe the model and the dynamics of the Furuta pendulum in the Ordinary Differential Equation and Differential Algebraic Equation forms. Section III and section IV show the Optimal Control Problem, its Nonlinear Program and subsequently the open-loop control schemes. Section V details the closed-loop methodology. Section VI is devoted to the results of simulation of the swing-up and moving to reach the desired set point, along with stabilizing the pendulum in the presence of external disturbances. Section VII involves the discussion related to our results, implementation and future work. Section VIII gives our acknowledgements.

II. System Modelling

In the model of the Furuta pendulum described in Fig. 1, the mass of the pendulum is uniformly distributed along the rod. θ_1 is the angular position of the rotary arm and θ_2 is the angular position of the pendulum. The arm is free to move only in the x-y plane and the pendulum can only rotate orthogonally with the axis of the arm.

A. Physical model of the pendulum

The model of the pendulum has been described in the Euler-Lagrange form where, M models the masses of the system, C models the Coriolis force and G models the gravitational force, and

$$M\left(\theta_1,\theta_2\right)\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix}+C\left(\theta_1,\theta_2,\dot{\theta}_1,\dot{\theta}_2\right)\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix}+G\left(\theta_1,\theta_2\right)=\begin{bmatrix}u\\ 0\end{bmatrix} \tag{1}$$

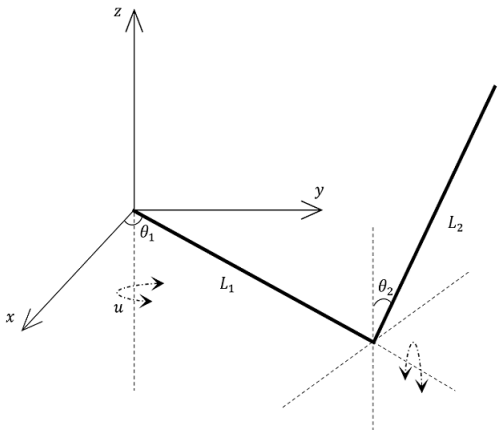

Fig. 1. Model of the pendulum in the Cartesian coordinate.

where

$$M\left(\theta_1,\theta_2\right)=\begin{bmatrix}M_{11} & M_{12}\\ M_{21} & M_{22}\end{bmatrix} \tag{2}$$

$$M_{11}=I_1+m_2\,L_1^2+0.25\,m_2\,L_2^2\,sin^2(\theta_2) \tag{3}$$

$$M_{12}=M_{21}=-0.5\,m_2\,L_2\,L_1\,cos(\theta_2) \tag{4}$$

$$M_{22}=I_2+0.25\,m_2\,L_2^2 \tag{5}$$

$$C\left(\theta_1,\theta_2,\dot{\theta}_1,\dot{\theta}_2\right)=\begin{bmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{bmatrix} \tag{6}$$

$$C_{11}=0.5\,m_2\,L_2^2\,sin(\theta_2)\,cos(\theta_2)\,\dot{\theta}_2 \tag{7}$$

$$C_{12}=0.5\,m_2\,L_2\,L_1\,sin(\theta_2)\,\dot{\theta}_2 \tag{8}$$

$$C_{21}=-0.25\,m_2\,L_2^2\,sin(\theta_2)\,cos(\theta_2)\,\dot{\theta}_1 \tag{9}$$

$$C_{22}=0 \tag{10}$$

$$G\left(\theta_1,\theta_2\right)=\begin{bmatrix}0 & -0.5\,m_2\,g\,L_2\,sin(\theta_2)\end{bmatrix}^T \tag{11}$$

A description of variables and parameters of the model is given in Table I. This model is based on [2], however, we choose the reference point for potential energy as a point in x-y plane. In [2], the pendulum has 0 potential energy when it is in the upright stable position.

TABLE I
PARAMETERS AND VARIABLES OF THE MODEL

Symbol	Value	Unit	Description
m_2	0.3	kg	Mass of the pendulum
L_1	0.2	m	Length of the rotary arm
L_2	0.3	m	Length of the pendulum
I_1	0050	$\mathrm{kg.m^2}$	Moment of inertia of the rotary arm
I_2	0.0025	$\mathrm{kg.m^2}$	Moment of inertia of the pendulum
g	9.81	$\mathrm{m/s^2}$	Gravitational acceleration
θ_1		rad	Angular position of the rotary arm
θ_2		rad	Angular position of the pendulum
u		N.m	Torque as control input

B. Model Definition

The state variable is taken based on the angular positions and velocities as,

$$x = \begin{bmatrix} \theta_1 & \theta_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T \tag{12}$$

1) Ordinary Differential Equation Formulation: The Ordinary Differential Equation (ODE) for the dynamics is expressed as,

$$x = \begin{bmatrix} \theta_1 & \theta_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T \tag{13}$$

$$\dot{x} = f(x,u) = \begin{bmatrix} x_3 \\ x_4 \\ M(x)^{-1}\left(-C(x)\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} - G(x) + \begin{bmatrix} u \\ 0 \end{bmatrix}\right) \end{bmatrix} \tag{14}$$

From (2), (3), (4), (5), and $M_{11}M_{22} > 0.25\, m_2^2\, L_2^2\, L_1^2$, we can show,

$$\begin{aligned} \det(M(x)) &= M_{11}M_{22} - M_{12}M_{21} \\ &= M_{11}M_{22} - 0.25\, m_2^2\, L_2^2\, L_1^2\, cos^2(\theta_2) \\ &> 0.25\, m_2^2\, L_2^2\, L_1^2\, sin^2(\theta_2) \end{aligned}$$

Therefore, $\det(M(x)) > 0$ and $M(x)$ is always invertible.

2) Differential-Algebraic Equation Formulation: The system as expressed in a Differential-Algebraic Equation (DAE) form, is

$$z = \begin{bmatrix} \ddot{\theta}_1 & \ddot{\theta}_2 \end{bmatrix}^T \tag{15}$$

$$\dot{x} = f(x,z,u) = \begin{bmatrix} x_3 & x_4 & z \end{bmatrix}^T \tag{16}$$

$$0 = g(x,z,u) = M(x)z + C(x)\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + G(x) - \begin{bmatrix} u \\ 0 \end{bmatrix} \tag{17}$$

Because $M(x)$ is invertible, $\frac{\partial g}{\partial z}$ has full rank and this DAE system is of index 1.

III. Optimal Control Problem Formulation

In this section, we define the Optimal Control Problem (OCP) of open-loop control and its Nonlinear Program (NLP). We use the ODE of the system for this formulation, however, the DAE form can also be used with changes to decision variables and equality constraints.

A. Optimal Control Problem

The initial pose for the system is set as $x(0) = [0, \pi, 0, 0]^T$, which is the pose when the pendulum is not actuated. Our goal is to swing up the pendulum (i.e. drive the system to the set point $\bar{x}$) using the least energy. Therefore, for the open-loop control, the Lagrange term of the Bolza cost function is

$$L(x(t), u(t)) = u(t)^2 \tag{18}$$

and we impose a terminal constraint at time T. We also describe the limit of actuator by restricting control u between $3\,\mathrm{N.m}$ and $-3\,\mathrm{N.m}$. The OCP is then formulated as

$$\min_{x(\cdot),u(\cdot)} \int_0^T L(x(t), u(t))\, dt \tag{19}$$

$$\text{subject to} \quad x(0) - x_0 = 0 \tag{20}$$

$$\dot{x}(t) - f(x(t), u(t)) = 0, \quad t \in [0,T] \tag{21}$$

$$x(T) - \bar{x} = 0 \tag{22}$$

$$-3 \le u(t) \le 3 \tag{23}$$

In the above mentioned OCP, one needs to give the system enough time so that the pendulum swings up at the end of the time horizon, given the constraints in control input. In other words, we should choose an appropriate T, based on the capability of the actuator.

B. Discrete time NLP

The discretized Lagrange term is defined as

$$l_i(s_i, u_i) = \int_{t_i}^{t_{i+1}} L(x_i(t, s_i, u_i), u_i)\, dt \tag{24}$$

The discretized NLP is then given as

$$\min_{s,u} \sum_{i=0}^{N-1} l_i(s_i, u_i) \tag{25}$$

$$\text{subject to} \quad x(0) - s_0 = 0 \tag{26}$$

$$s_{i+1} - F(s_i, u_i) = 0, \quad i = 0, ..., N-1 \tag{27}$$

$$s_N - \bar{x} = 0 \tag{28}$$

$$-3 \le u_i \le 3 \tag{29}$$

Because the dynamic model of the system is nonlinear, our formulated OCP and NLP are non-convex.

IV. Open-loop Control

The open-loop control of the pendulum swings it up to the upright position and makes it stay there. It was implemented using Direct Multiple Shooting and also Direct Collocation. We set the number of control intervals as $N = 40$, distributed over a time horizon of $T = 1.5\,\mathrm{s}$ and we solved the NLP with MATLAB using CasADi [1] framework with IPOPT [5] as NLP solver.

A. Swing-up using Direct Multiple Shooting

We implemented Direct Multiple Shooting with an explicit Fourth Order Runge-Kutta (RK-4) integrator to derive the next state from the previous state and the control input. The number of RK-4 steps per time interval was chosen as 1.

B. Swing-up using Direct Collocation

In the Direct Collocation method, Orthogonal Collocation was used as an implicit integrator, with Lagrangian basis polynomials of order 3 and Legendre collocation points. We used the DAE model of the system for Direct Collocation with extension in constraints of the NLP as mentioned in section 14.4.3 of [4].

V. Closed-loop Control

After solving the open-loop OCP, we used Model Predictive Control (MPC) to provide feedback control for the system. We apply only the first input in the optimal control sequence u^* obtained from the open-loop NLP and shift the prediction horizon one step forward. We implemented MPC with Direct Multiple Shooting, Direct Collocation and Real-Time Iteration. We used the ODE and RK-4 integrator to implement Real-Time Iteration, with the qpOASES [3] to solve the subsequent QP. During the simulation, MPC was expected to swing up the pendulum, move the system to another desired set point and then stabilize the pendulum when an external impulse acted on it. The impulse was modelled as a sudden change in the angular velocity of the pendulum. We set the total time of simulation $T_f = 15\,\mathrm{s}$, and a time horizon of length $T = 1.5\,\mathrm{s}$. We varied the number of control intervals to compare performance, as is dealt with in section VI. The source code is available here.

VI. Results

A. Open-loop Control

For swing up, we had the set point as $\bar{x} = [0, 0, 0, 0]^T$. As can be seen from Fig. 2, the Direct Multiple Shooting and the Direct Collocation approaches were able to find an optimal sequence of control inputs to swing up the pendulum. The resultant state trajectories and control input sequences for both controllers are similar because the system dynamics model is non-stiff i.e. we have not accounted for the friction, which leads to our system not being very stable, due to lack of natural damping. With different initial guesses for the open-loop NLP, the two controllers give different state trajectories and controls, highlighting the non-convexity of the NLP. As we solve the optimal control inputs for every time step in the beginning itself, any external disturbance to the system (for example, an unexpected impulse acting on the pendulum) will throw the system off the planned trajectory, since there is no feedback involved. This gives rise to our motivation for using closed loop control.

B. MPC for closed-loop Control

1) State Trajectory and Control Sequence Comparison: The set point for swing-up was set as $[\pi, 0, 0, 0]^T$ and the second set point was $[\pi/2, 0, 0, 0]^T$. We do not use any initialization strategy here for the Direct Multiple Shooting and the Direct Collocation methods (i.e. we set all initial guesses to zero). In Fig. 3, all three controllers were able to bring the system to the set point and deal with the external impulse within a good settling time. We see however, that

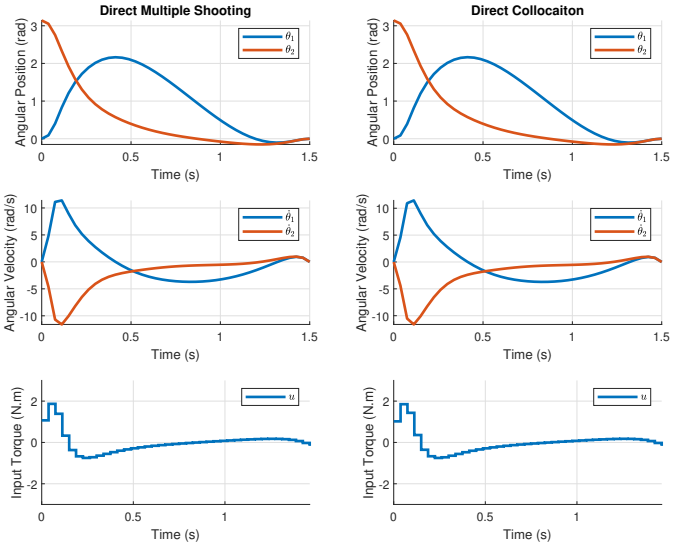

Fig. 2. Open-loop control for the system.

there are differences in the swing up stage among the three approaches. The Direct Multiple Shooting approach gives us the best result, due to lower jump between control signals, whereas the other two approaches show a high jump in the control signal and subsequent state trajectories. In a real-time application, a gentle control transition is preferred to put lesser load on the actuator. When it comes to changing set point or dealing with the impact of an external impulse, all three controllers have similar control signals and thus state trajectories.

2) Runtime Comparison: We kept the same setup for the simulation as in last section and evaluated the runtimes for two different number of control intervals , $N \in \{30, 40\}$. For initialization of the subsequent NLPs in MPC with Direct Multiple Shooting (DMS) and Direct Collocation (DC), we used the following methods:

- No initialization
- Simple warm-start (i.e. repeating optimal value from last MPC iteration)
- Shift initialization

They are respectively shown with index $i = 0, 1, 2$ in Table II. The Real-Time Iteration (RTI) does not require an initialization strategy.

Table II shows that the RTI method takes the least time

TABLE II
Runtime with different schemes and configurations

Control Intervals	Scheme	Initialization Strategy		
		$i = 0$	$i = 1$	$i = 2$
$N = 40$	DMS	30.328 s	31.245 s	31.765 s
	DC	30.202 s	25.291 s	25.838 s
	RTI	19.519 s		
$N = 30$	DMS	17.261 s	17.404 s	16.972 s
	DC	16.198 s	14.877 s	14.637 s
	RTI	7.221 s		

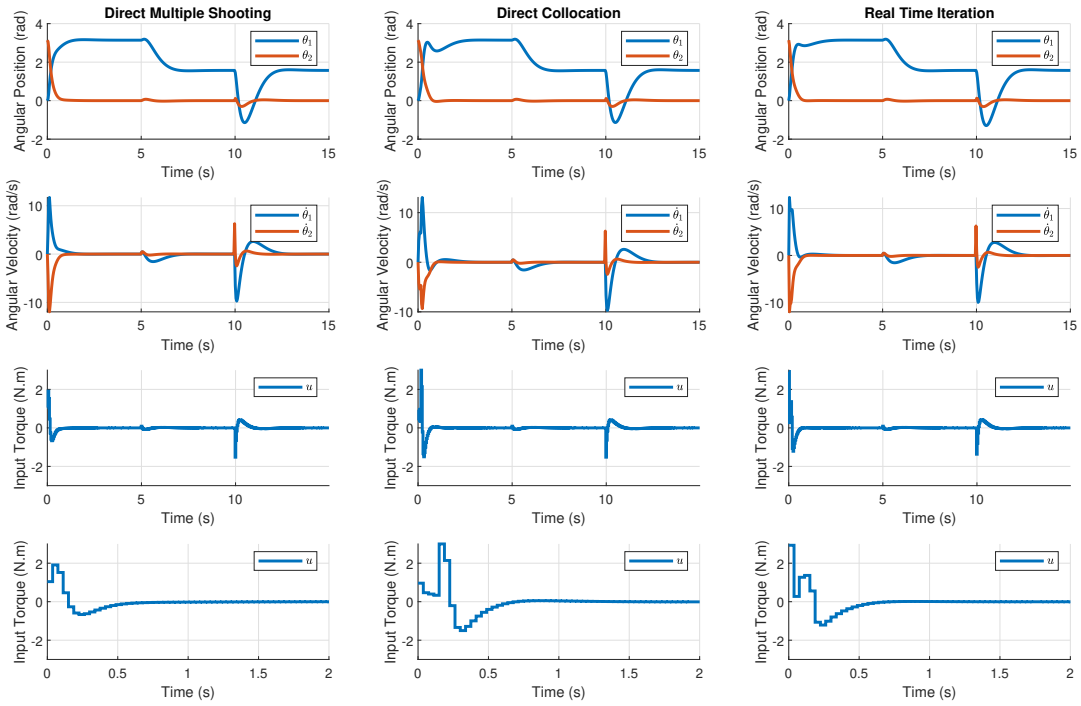

Fig. 3. Closed-loop control for the system. Comparison between methods used for MPC. The fourth row is the enlarged version of the third row in the first 2 seconds of the simulation. At 5 s, set point of the system was changed. At 10 s, an external impulse hits the pendulum.

amongst the three controllers, and the runtime is even significantly reduced when $N = 30$. However, when the number of control intervals is $N = 30$, RTI gives a control sequence with higher jump compared to its control sequence with $N = 40$, causing larger fluctuation in the state trajectories during swing up, which is shown in Fig. 4. This might be explained by the fact that we need to linearize the constraints of the NLP for every step. In our case, since these constraints enforce continuity of the state trajectory, we need a sufficient number of control intervals to achieve good quality control input. The Direct Collocation method gives a slightly better runtime compared to that of Direct Multiple Shooting. From Table II, we can say that initialization strategies help reduce the runtime for the Direct Collocation method, although the same could not be realized for Direct Multiple Shooting.

Fig. 4 and Table II also shows that decreasing the number of control intervals to $N = 30$ and using a simple warm-start reduces the runtime and still yields the same state trajectories and control sequence. By using initialization methods, we do not see the jump in control sequences of Direct Collocation during the swing up phase.

C. Angular Periodicity

The Furuta pendulum is built based on rotational movement, which means that we encounter angular periodicity. An interesting implication of this fact is that for the movement of rotary arm, it could be cheaper to rotate $-\pi/2$ (i.e. $\pi/2$ clockwise) to reach the set point $\bar{\theta}_1 = 3\pi/2$. The constraint stated in (22) does not take into account the periodicity of the system. Similarly, the integrators that we have used so far could not distinguish between periodic variables and non-periodic variables. We partially overcome this problem with a modification in the NLP. In this part, we choose the ODE representation, RK-4 integrator and Direct Multiple Shooting method for a MPC simulation. First, we introduce the vector $w\left(x\left(t\right)\right)$ as

$$w\left(x\left(t\right)\right) = \begin{bmatrix} cos\left(x_1\right) - cos\left(\bar{x}_1\right) \\ sin\left(x_1\right) - sin\left(\bar{x}_1\right) \\ cos\left(x_2\right) - cos\left(\bar{x}_2\right) \\ sin\left(x_2\right) - sin\left(\bar{x}_2\right) \\ x_3 - \bar{x}_3 \\ x_4 - \bar{x}_4 \end{bmatrix} \tag{30}$$

As long as we bring $w\left(x\left(t\right)\right)$ to zero, we ensure that our state variables converge to the set point. With Q as the diagonal weighting matrix, the augmented cost function is written as

$$L_1\left(x\left(t\right), u\left(t\right)\right) = w\left(x\left(t\right)\right)^T Q w\left(x\left(t\right)\right) + u\left(t\right)^2 \tag{31}$$

If we choose the matrix Q with low weights, we can drive the system to the set point expending the same energy compared to the original cost function. After eliminating the terminal constraint (22), we get the following OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_1\left(x\left(t\right), u\left(t\right)\right) dt \tag{32}$$

$$\text{subject to} \quad x\left(0\right) - x_0 = 0 \tag{33}$$

$$\dot{x}\left(t\right) - f\left(x\left(t\right), u\left(t\right)\right) = 0, \quad t \in [0, T] \tag{34}$$

$$-3 \leq u\left(t\right) \leq 3 \tag{35}$$

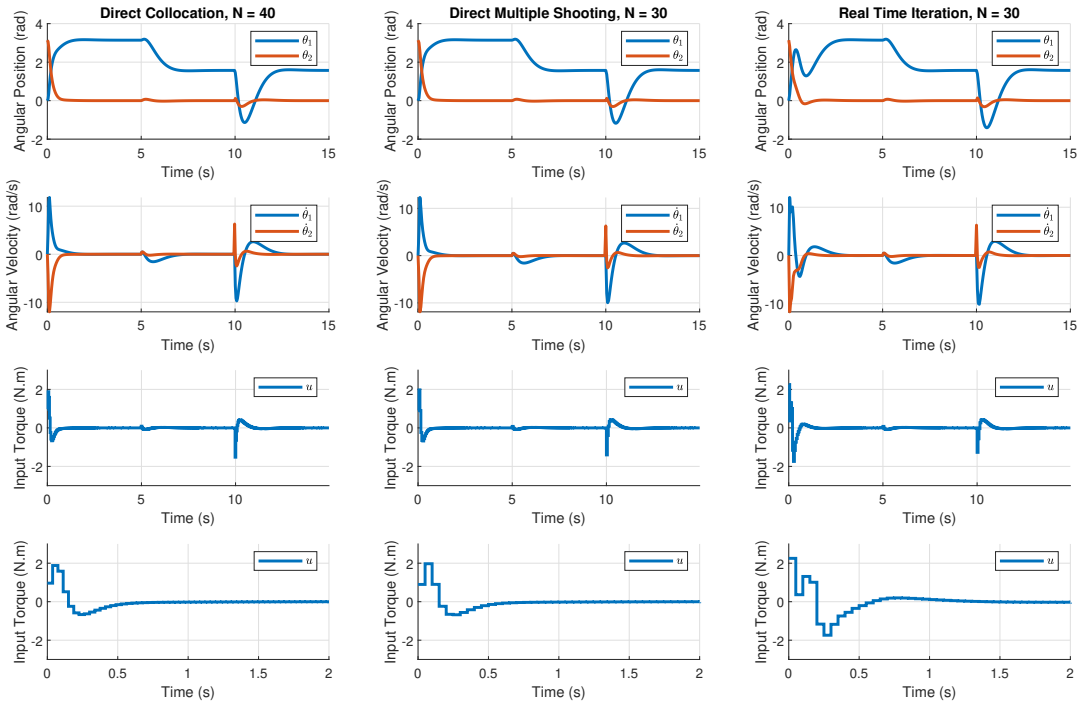

Fig. 4. Comparison of difference MPC schemes and configurations. Simple warm-start initialization was used for Direct Multiple Shooting and Direct Collocation. The fourth row is the enlarged version of the third row in the first 2 seconds of the simulation.

With the new NLP which is formulated from the above OCP, we resolve the angular periodicity without violating the *linear independence constraint qualification* (LICQ). In our simulation, we set the swing up point as $[5\pi/2, 0, 0, 0]^T$ and the second set point was $[7\pi/2, 0, 0, 0]^T$. We chose $Q = \text{diag}\,(0.05, 0.05, 0.05, 0.05, 0.01, 0.01)$, with number of control intervals per horizon as $N = 30$. In Fig. 5, during swing up and set point movement, both approaches spend the same energy, however, giving different state trajectories. For the controller considering angular periodicity, we see that there is an overshoot around $\pi/2$ in θ_1 i.e. the angular position of the rotary arm, which leads to the pendulum stabilizing without the arm completing another full revolution to reach $5\pi/2$ which is essentially $\pi/2$, with a remarkably less settling time. In comparison, the other controller, without consideration for angular periodicity, drives the rotary arm to $5\pi/2$ before stabilizing, with a longer settling time. The similarity in the expended energy for both controllers can be explained by observing that the controller with periodicity consideration uses more energy for the overshoot, while the other controller uses the energy for an extra revolution of the arm.

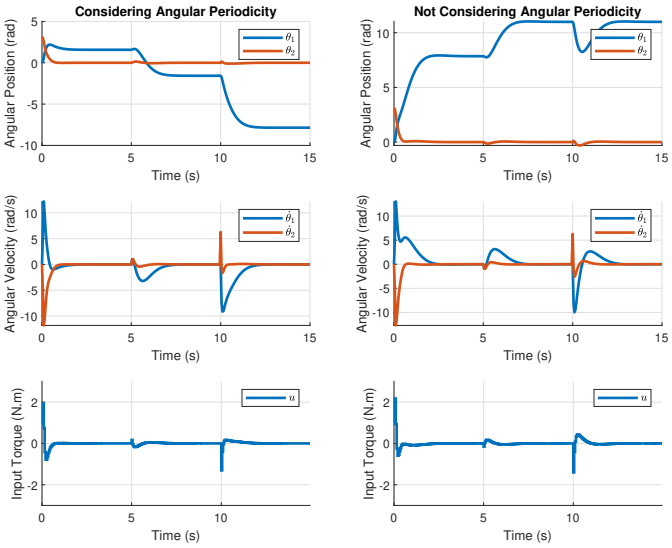

Fig. 5. Angular periodicity consideration.

When the external impulse hits the pendulum, the two controllers exhibit different behaviors. While the controller without the angular periodicity consideration tries not to drift away from the set point, the other controller just tries to keep the pendulum from falling down while simultaneously utilizing the initial velocity that the impulse gave the pendulum to drive the rotary arm to the right position by rotating a 2π angle. This leads to a lower required control input at the time when the impulse hits. Interestingly, after the external disturbance, there is no difference in settling time of the system in these two approaches, only that the first controller tries to fight against impact of the impulse, while the second one uses the energy from the impulse itself to drive the pendulum to the set point. We observe however, that it takes slightly more runtime for the controller which considers angular periodicity.

VII. Discussion

We implemented Model Predictive Control for the ideal Furuta Pendulum using Direct Multiple Shooting, Direct Collocation and Real-Time Iteration methods. Direct Multiple Shooting offers a simple implementation with the ODE model, but it requires inversion of the matrix $M(x)$. The implementation Direct Collocation is trickier but it comes with great flexibility. It works well with stiff systems, which extends to a large class of mechanical and mechatronics systems. In addition, it is extended to work with DAE model with minor modifications to the dynamics, which is used alternatively to model mechanical systems. Our evaluation shows that Real-Time Iteration has the best runtime among the three approaches with a reasonable output, and we observe that different choices of the first linearization point for Real-Time Iteration will usually lead to different solutions of subsequent QPs or might even result in an infeasible QP. The performance of the different controllers in terms of swing up, set point change and stability under external disturbances have also been evaluated. By augmenting the cost function with a slight change to the OCP and NLP formulations, the angular periodicity has been mentioned and addressed here, without much difference in the runtime compared to that of our original problem formulation.

Our model is capable of only partially describing the real system dynamics. An important factor that we did not model in our consideration is the impact of friction forces, thus leading to a significant plant-model mismatch. Additionally, in our model, the torque is a direct control input. In reality however, electric motors usually take voltage as the input and convert it to a torque, therefore, the relationship between an input voltage and its output torque was also not modelled. Moreover, neither state disturbances nor the measurement noise were considered. As a result, many modifications and improvements; for example, adding an online state estimator and using model with higher complexity can be done to alleviate model-plant mismatch.

Finally, the runtime measurements were done on a laptop and given that we use a very simple model, there is scope to further refine the implementation to better enable online and real-time control in embedded devices. With the basic knowledge of nonlinear optimization, Model Predictive Control and know-how in the implementation of control schemes, in the future, we will be able to formulate optimal control problems and solve them with fast embedded solvers for nonlinear optimal control and MPC such as `acados`.

VIII. Acknowledgment

We would like to thank Florian Messerer, Andrea Ghezzi and Prof. Dr. Moritz Diehl for the supervision in this project.

References

[1] Joel A E Andersson et al. “CasADi – A software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[2] J L Duart et al. “Dynamic Modeling and Simulation of a Rotational Inverted Pendulum”. In: *Journal of Physics: Conference Series* 792 (Jan. 2017), p. 012081. DOI: 10.1088/1742-6596/792/1/012081.

[3] H.J. Ferreau et al. “qpOASES: A parametric active-set algorithm for quadratic programming”. In: *Mathematical Programming Computation* 6.4 (2014), pp. 327–363.

[4] Sébastien Gros and Moritz Diehl. *Numerical Optimal Control*. Apr. 2022.

[5] A. Wächter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming* 106.1 (2006), pp. 25–57.