Predictive Control for Swing-up and Balancing of an Underactuated Inverted Double Pendulum

Jannik J. Grothusen, Justus Karnath, Vincent Lenz, Richard Wittmuess

Video: https://youtu.be/6fgMRzRNHnU
Code: https://github.com/J4nn1K/me231a-final-project

***Abstract*—This paper describes the design of a MPC controller to carry out the swing-up process of a double pendulum into the upper unstable equilibrium position. Furthermore, a non-linear energy based MPC, a linear reference tracking MPC and a LQR state feedback controller are used comparatively to keep the pendulum in that position.**

***Index Terms*—Model Predictive Control, Inverted Double Pendulum, LQR Controller**

I. Introduction

The double inverted pendulum is an under-actuated robot that poses a major challenge in many control approaches. It consists of two pendulums connected with an actuated pivot point on the base which allows it turn in any direction. In this paper, it is assumed that the beams are mass-loaded elements whose weight force acts at the center of gravity. The scope of this paper is to investigate the performance of different control approaches to keep the pendulum in a vertical position. Several methods have been explored to accomplish this. Balancing is evaluated using a LQR state feedback controller, a linear reference tracking MPC and a nonlinear MPC with a cost-function that minimizes kinetic energy while maximizing potential energy. This paper is based on the publication *Energy Based Control of the Pendubot* by Isabelle Fantoni, Rogelio Lozano, and Mark W. Spong [1].

II. Double Pendulum Model

A. Nominal Model

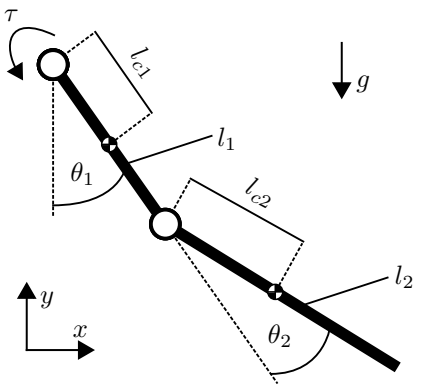

Fig. 1. Sketch of the double pendulum

The double pendulum seen in figure 1 consists of two rods of length l_1 and l_2 with their respective centers of mass at l_{c1} and l_{c2}. The rods are connected at their end and the first rod has an actuated connection to the base of the pendulum.

B. State Space Formulation

To describe the inverted double pendulum we use the angle of the actuated joint θ_1 and the relative angle θ_2 of second joint. To describe a compact state space model, we choose

$$\boldsymbol{q} = [\theta_1, \theta_2]^T \tag{1}$$

which concludes

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} \text{ and } \dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix}. \tag{2}$$

Therefore, the desired unstable equilibrium point can be written as $\hat{\boldsymbol{x}} = \begin{bmatrix} \pi & 0 & 0 & 0 \end{bmatrix}^T$. We can now write the non-linear equations of motion

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \boldsymbol{\tau_g}(\boldsymbol{q}) + \boldsymbol{B}\boldsymbol{u} \tag{3}$$

in state space representation as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{0} & -\boldsymbol{M}^{-1}\boldsymbol{C} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ \boldsymbol{M}^{-1}\boldsymbol{\tau_g} + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{u} \end{bmatrix}. \tag{4}$$

Furthermore, we define the input as the moment

$$\boldsymbol{u} = \tau \tag{5}$$

and with $\sin(\theta_1)$ abbreviated as s_1, and c_{1+2} for $\cos(\theta_1 + \theta_2)$ we can write the matrices as

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + 2m_2 l_1 l_{c2} c_2 & I_2 + m_2 l_1 l_{c2} c_2 \\ I_2 + m_2 l_1 l_{c2} c_2 & I_2 \end{bmatrix} \tag{6}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -2m_2 l_1 l_{c2} s_2 \dot{q}_2 & -m_2 l_1 l_{c2} s_2 \dot{q}_2 \\ m_2 l_1 l_{c2} s_2 \dot{q}_1 & 0 \end{bmatrix} \tag{7}$$

$$\boldsymbol{\tau_g}(\mathbf{q}) = \begin{bmatrix} -m_1 g l_{c1} s_1 - m_2 g \left(l_1 s_1 + l_{c2} s_{1+2} \right) \\ -m_2 g l_{c2} s_{1+2} \end{bmatrix} \tag{8}$$

$$\mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{9}$$

The rods' centers of mass are formulated relative to the base. The global location for further computations is are calculated as

$$\boldsymbol{p_{c1}} = \begin{bmatrix} l_{c1} s_1 \\ -l_{c1} c_1 \end{bmatrix}, \quad \boldsymbol{p_{c2}} = \begin{bmatrix} l_1 s_1 + l_{c2} s_{1+2} \\ -l_1 c_1 - l_{c2} c_{1+2} \end{bmatrix}. \tag{10}$$

C. Linearized Model

Linearized around a constant position $\boldsymbol{y}_S$ we can formulate the problem as

$$\boldsymbol{M}\ddot{\boldsymbol{\eta}} + \boldsymbol{Q}\boldsymbol{\eta} = \boldsymbol{h} \tag{11}$$

with

$$\begin{aligned} \boldsymbol{M} &= |\boldsymbol{M}|_{\boldsymbol{y}_S} = \text{ const.} \\ \boldsymbol{Q} &= \left.\frac{\partial \boldsymbol{k}}{\partial \boldsymbol{q}}\right|_{\boldsymbol{y}_S} - \left.\frac{\partial \boldsymbol{\tau_g}}{\partial \boldsymbol{q}}\right|_{\boldsymbol{y}_S} = \text{ const.} \\ \boldsymbol{h} &= \boldsymbol{\tau_g}(\boldsymbol{y}_S, \underbrace{0}_{=\dot{\boldsymbol{y}}_S}) - \boldsymbol{k}(\boldsymbol{y}_S, \underbrace{0}_{=\dot{\boldsymbol{y}}_S}) \end{aligned} \tag{12}$$

where

$$\boldsymbol{k} = \boldsymbol{C}\boldsymbol{q}. \tag{13}$$

For $\boldsymbol{Q}$ this results in

$$\boldsymbol{Q} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \tag{14}$$

with

$$\begin{aligned} Q_{11} &= -gl_1m_1\cos(x_0) - gm_2(L_1\cos(x_0) \\ &\quad + l_2\cos(x_0 + x_1)) \\ Q_{12} &= 2L_1l_2m_2x_2x_3\cos(x_1) + L_1l_2m_2x_3^2\cos(x_1) \\ &\quad - gl_2m_2\cos(x_0 + x_1) \\ Q_{21} &= -gl_2m_2\cos(x_0 + x_1) \\ Q_{22} &= -L_1l_2m_2x_2^2\cos(x_1) - gl_2m_2\cos(x_0 + x_1). \end{aligned} \tag{15}$$

III. Controller Design

A. Controller for swing-up

In this section, the design of three different controllers is shown. The first controller uses Model Predictive Control (MPC) and is primarily designed for the swing-up of the pendulum. Therefore the non-linear system is used. Furthermore, instead of a classical reference-tracking approach for the cost function design, the kinetic and potential energy of the system are used. In particular, the potential energy is maximized to favor the upmost unstable equilibrium point and the kinetic energy is minimized for stabilizing it. By doing so one can avoid the problem of multiple representations of the equilibrium point with multiples of 2π. The only constraints used are the input constraints and model equations, which are discretized using the zero order hold discretization.

B. Controllers for stabilization

1) Reference MPC: To swing the double pendulum to its upright position, we employ the MPC described in III-A that utilizes a nonlinear model of the system. As the pendulum approaches the upright position, we transition to using a linear model to approximate the system's behavior for small deviations from this equilibrium point.

By implementing the linearized model and performing reference tracking we modified the MPC to optimize the control actions. Hereby, the reference signal that is tracked is the upright unstable equilibrium point:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \dot{\hat{q}}_1 \\ \dot{\hat{q}}_2 \end{bmatrix} = \begin{bmatrix} \pi \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{16}$$

It is worth noting that the optimization problem resulting from this modification is convex, in contrast to the nonlinear case, which makes it more computationally efficient. Keep in mind that if the deviations from the upright position become too large, the linear model is no longer sufficient and we switch back to the swing-up MPC exploiting the nonlinear model.

2) LQR: In order to achieve balance in the upright position, we use a Linear-Quadratic Regulator (LQR) controller in another approach. As we near this area, we can utilize the linear model derived in the section II-C to find the optimal controller that minimizes the cost function J [2]:

$$J = \frac{1}{2}\int_0^\infty \left(\tilde{\boldsymbol{x}}^T\boldsymbol{Q}\tilde{\boldsymbol{x}} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}\right) dt \tag{17}$$

Here, the value of x in this equation represents the error vector between the actual and desired signals, with the desired signal being the upright position.

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} - \begin{bmatrix} \pi \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{18}$$

Q and R are positive definite matrices that act as weightings in the controller design. In order to design the LQR controller, we must solve the continuous Riccati equation [3]:

$$-\boldsymbol{P}\boldsymbol{A} - \boldsymbol{A}^T\boldsymbol{P} - \boldsymbol{Q} + \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} = \boldsymbol{0} \tag{19}$$

which allows us to define the optimal control law as follows:

$$\boldsymbol{u}^* = -\boldsymbol{K}\boldsymbol{x} \tag{20}$$

Where:

$$\boldsymbol{K} = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} \tag{21}$$

In our case, we have chosen the weightings Q and R as follows:

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \boldsymbol{R} = 1 \tag{22}$$

IV. Experimental Results

A. Simulator

Due to the lack of access to a physical system for testing, a simulator is being constructed in order to replicate the actual system with as much accuracy as possible. The main sources of error that would have been encountered had a real system been tested on are discretization errors and model mismatch.

It was aimed to address model mismatch by mapping the error of the internal model of the controller over the state

space and using interpolation to correct the model by adding the current error to the constraint equation in order to learn from the error during the simulation process and improve performance over time. However, due to the inability to access the optimization variables during runtime using Pyomo [5], the desired error mapping and interpolation could not be implemented, thus the exact model parameters were used for both the simulation and the controller.

Simulation of the double pendulum using nonlinear differential equations has been employed to validate the controller. To reduce the discretization error, a more exact integration scheme than the zero-order hold discretization was used to solve the model, allowing for a more accurate representation of the system.

Both fast and slow dynamics are exhibited by the double pendulum, resulting in it being a stiff system requiring numerical integration techniques capable of accurately representing both dynamics. Hence, we have chosen to use Backward Differentiation Formulas (BDFs) as our stiff integrator to solve the ODEs. Specifically, we have implemented the order 1-5 BDF from the scipy library. This integrator is particularly effective because it can achieve a high level of accuracy with smaller time steps compared to traditional Runge-Kutta methods. The small time steps used in the simulator are necessary to accurately capture the behavior of the double pendulum.

However, the time step used in the simulator is smaller than the time step used in the controller due to the computational cost of using a smaller time step for the controller.

B. Results

In order to replicate a realistic system, the input was limited to u = 4. This value was selected randomly from a range after analyzing the impact of the inputs on the system. It was important to ensure that the system was physically capable of bringing the pendulum to the desired position, while also preventing that an over-dimensioning of the manipulated variables leads to trivial results.

Due to limitations in computing power, the controller frequency had to be fixed. After conducting several test series on different systems, it was determined that setting the controller frequency to 20 Hz would provide the best balance of accuracy and computational feasibility. Lower frequencies resulted in better accuracy, but were no longer practical from a computational standpoint.

Figure 2 shows the cost function at each step for various prediction horizons. It can be seen that for short horizons of 0.1 and 0.2 seconds, the cost reaches a local minimum at the equilibrium position of the double pendulum, where the actuated rod is pointing upwards and the second rod is facing downwards. With a short horizon like this, it is not possible for the optimizer to plan ahead and achieve the desired swing-up.

For the horizon of 1 second, the controller can be seen building up energy, which corresponds to the oscillations of the cost function. The system is able to do this by simulating the system over a longer horizon, allowing it to oversee the

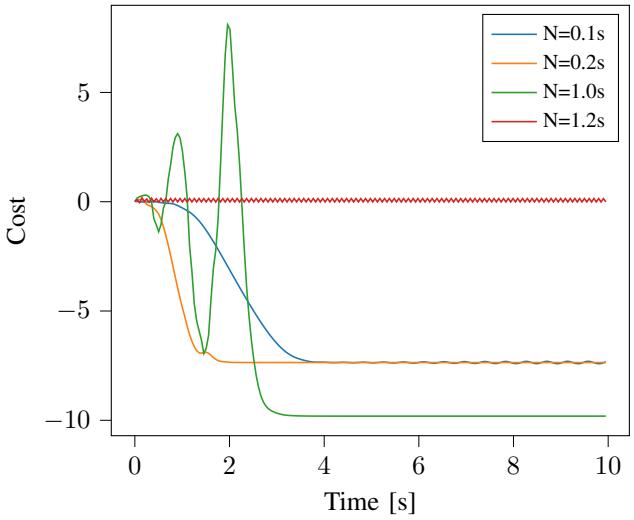

Fig. 2. Cost per timestep for different prediction horizons.

spikes in the cost function. Eventually, the system reaches the desired optimum and is able to stabilize the pendulum.

However, for some other configurations such as the 1.2 second horizon, the controller encountered numerical issues. The likely cause of this occurring even with longer horizons is the accumulation of discretization error during the internal simulation, which eventually reaches a point where the optimization result no longer accurately reflects the true problem.

To address this issue, it would be necessary to use a higher control frequency or a different discretization strategy. However, increasing the frequency is not a viable option in this case, as the current setup already requires a speed-up by a factor of 3 to approach real-time applications.

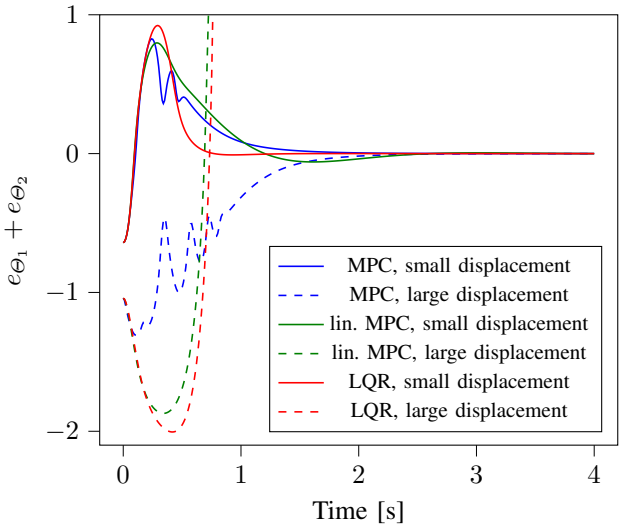

Fig. 3. Reference tracking error of tested control algorithms.

Nevertheless, it is generally easier to control the pendulum in the upper equilibrium position. Figure 3 compares the three implemented controllers: the nonlinear MPC used for

the swing-up, the linear MPC using reference tracking, and the continuous LQR controller for the control in around that position. The performance of controlling the system back to the desired equilibrium point is similar for all three controllers, with slight advantages for the LQR. This is likely due to model errors in the MPC controllers, which initially react faster but then have to correct again and lose some of their desired performance. For larger initial deflections, the limitations of the linear system used for the second MPC and the design of the LQR controller become apparent, and the controllers are unable to stabilize the system. While it is possible to significantly reduce the run time of the MPC controllers by using shorter horizons and a linear system, there is no clear advantage over the LQR controller in terms of performance.

It was obvious that, the controller model mismatch and accumulated error while solving the system increase when longer time horizons are used. Using too short horizons leads to simulations that are not able to reach the upper equilibrium point and therefore settle in local optima. There are also configurations in between where the horizon is long enough to overcome local optima but because of the discussed model mismatch is still not able to settle and swings around for the entire simulation time. Using the described set up it is possible to tune the controller to achieve a successful swing-up. For extracting a general tuning strategy better model knowledge and a more exact discretization would be necessary.

Once the swing-up is done successfully, a linear MPC using a reference tracking cost function as well as a LQR controller can take over.

V. Conclusion

In our work, we demonstrated the use of a model predictive control approach to control the inverted double pendulum in the upright unstable equilibrium position. Our initial plan was to swing up the pendulum with an MPC that uses the nonlinear model of the pendulum's system and then switch to either a LQR controller or an MPC with reference tracking, both of which use a linearized model. However, we were unable to implement this plan. Instead, we demonstrated that it is possible to efficiently control the system with a linearized model for small displacements around the unstable equilibrium point. Both the MPC and the LQR performed similarly, with the LQR showing better runtime results. These controllers are not reliable for larger displacements, however, as the system becomes unstable. In these cases, the MPC using the nonlinear model performs better in comparison and therefore must be used for these cases.

We also found that, in a simulated environment, there are limitations on the prediction horizon for the MPC. If the prediction horizon is set too high, it can lead to an excessive amount of computational errors, resulting in unreliable results. It is therefore important to find an appropriate balance in the prediction horizon to achieve reliable results.

In future research, it would be beneficial to examine ways to improve the discretization strategy of the model in order to reduce discretization errors, while also trying to enhance performance in terms of computation time. This could result in more reliable results and a more efficient optimization process.

Additionally, it would be valuable to compare and validate our simulated results with experiments of the developed MPC on a real physical system.

References

[1] Fantoni, Isabelle, Rogelio Lozano, and Mark W. Spong. "Energy based control of the pendubot." IEEE Transactions on Automatic Control 45.4 (2000): 725-729.

[2] Seif-El-Islam Hasseni. "Hybrid Control of a Pendubot System Using Nonlinear H_∞ and LQR" in WSEAS TRANSACTIONS on SYSTEMS and CONTROL (2021)

[3] Lavretsky E, and Wise KA. "Optimal Control and the Linear Quadratic Regulator" in Lavretsky E, and Wise KA (eds.), Robust and Adaptive Control With Aerospace Applications, Springer, 2013, pp. 27-50.

[4] Russ Tedrake. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832). Downloaded on December 14th 2022 from http://underactuated.mit.edu/

[5] Hart, William E., Jean-Paul Watson, and David L. Woodruff. "Pyomo: modeling and solving mathematical programs in Python." Mathematical Programming Computation 3(3) (2011): 219-260.