Optimal control of double inverted pendulum

Bašić, Mate

University of Zagreb
Faculty of Mechanical Engineering and Naval Architecture

MASTER'S THESIS

Mate Bašić

Zagreb, 2024.

University of Zagreb
Faculty of Mechanical Engineering and Naval Architecture

MASTER'S THESIS

Optimal control of double inverted pendulum

Mentor:
prof. dr. sc. Andrej Jokić

Student:
Mate Bašić

Zagreb, 2024.

Statement

I hereby declare that I have made this thesis independently using the knowledge acquired during my studies and the cited references.

Zagreb, May 2024. Mate Bašić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Procesno-energetski, konstrukcijski, inženjersko modeliranje i računalne simulacije i brodostrojarski

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 24 - 06 / 1**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Mate Bašić** JMBAG: 0035214749

Naslov rada na hrvatskom jeziku: **Optimalno upravljanje dvostrukim inverznim njihalom**

Naslov rada na engleskom jeziku: **Optimal control of double inverted pendulum**

Opis zadatka:

The double inverted pendulum is a nonlinear mechanical system frequently used as a benchmark example to test various control methods, both in simulations and in laboratory setups. A desirable feature of this system is that we can build its' rather accurate model, while the system is suitable for testing both linear (control in proximity of equilibrium point) and nonlinear control algorithms (global control).
The main goal of this thesis is to design a controller which stabilizes the pendulum in the upward position. In addition to stabilization, a controller should be optimal, with the objective function appropriately penalizing deviations from the desired equilibrium, as well as the control effort.

The thesis should include the following:

- Develop and present a mathematical model of double inverted pendulum dynamics suitable for controller synthesis.
- Suitably formulate the optimal control problem, including actuator and state constraints.
- Design the optimal LQR controller based on the developed model.
- Design the model predictive controller (MPC) which incorporates actuator/state constraints.
- Present simulation results of the closed-loop behavior, wherein the inverted pendulum is simulated based on the full nonlinear model.

The thesis should include reference list of used literature as well as acknowledgments to any assistance obtained.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
7. ožujka 2024.	9. svibnja 2024.	13. – 17. svibnja 2024.

Zadatak zadao:

Prof.dr.sc. Andrej Jokić

Predsjednik Povjerenstva:

Prof. dr. sc. Tanja Jurčević Lulić

Contents

List of Figures

List of Symbols

Summary

In this thesis a model based control of an inverted pendulum on a cart is considered. The system's dynamics is nonlinear, and for control synthesis purposes the nonlinear model is linearized around the upward position. The linearized model is then used for synthesis of linear quadratic regulator (LQR) and model predictive controller (MPC). In the MPC setting, terminal cost and terminal constraints have been incorporated and their impact on the closed-loop behavior has been analyzed. It is also shown that MPC can be used for double pendulum position reference tracking. Finally, two state estimation techiques are compared and analysed, the extended Kalman filter (EKF) and the unscented Kalman filter (UKF).

Keywords: Double inverted pendulum on a cart, linear quadratic controller (LQR), model predictive control (MPC), extended Kalman filter (EKF), unscented Kalman filter (UKF)

1 Introduction

Optimal control is an approach to the synthesis of the closed-loop behaviour in which contol actions are chosen to minimize some objective function. In that sense, model predictive control (MPC) is one of its most widely used techniques, in the same time being theoretically very general, but also practically applicable to the specific control problems. Although the theory development started in the final years of 1960's, the first succesful industrial applications of the model predictive control occured in the petrochemical industry in the 1970's [1]. Since then, its usage has been continously increased not only in various other fields of production, but also economy in a broader sense.

The classical perspective to a generic control problem is shown in Figure 1.1, where the controller block takes in the error and computes the actuator signal values, which in turn influence the controlled process plant.

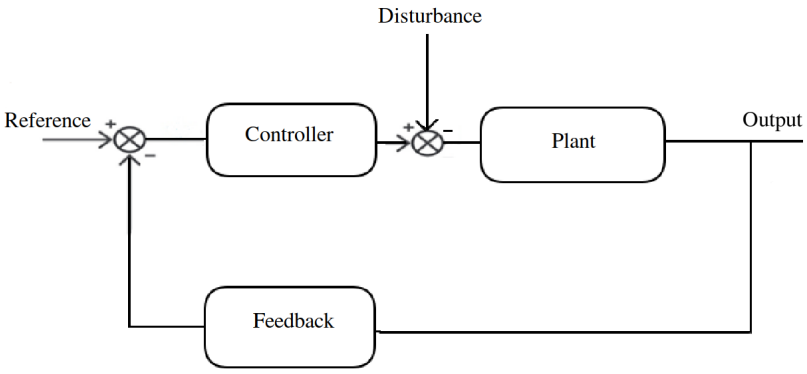

Figure 1.1: Classical approach to a generic system control problem

The main idea of the MPC is to compute the control actions by solving the constrained optimisation problem in every time step, where the cost function of this problem is formulated based on the desired system behavior. This scheme is shown in Figure 1.2, where block **P** stands for process and r for reference, with y being a measured output.

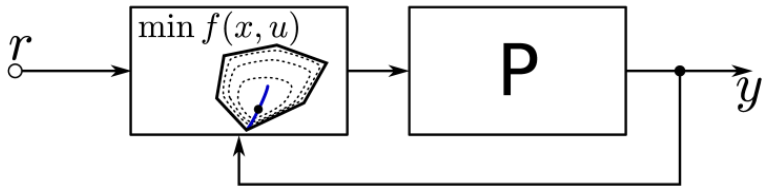

Figure 1.2: MPC approach to a generic system control problem [1]

The topic of this thesis is to investigate the possibilities of such a control approach on the double inverted pendulum model. The thesis itself is divided in six chapters. After the introduction, the second chapter briefly presents theoretical backgrund neccessary for MPC control problem definition. The topic of the third chapter is the derivation of the double inverted pendulum on a cart model, which is one of the few usual benchmark problems for a particular controller performance assessment. In the fourth chapter, LQR and then MPC control laws are presented and their performance is analysed. The topic of the fifth chapter is the state estimation, often an unavoidable prerequisite for any system control. Finally, the sixth chapter closes the thesis with the concluding remarks.

2 Theoretical background

In order to develop the idea behind the generic MPC scheme shown in Figure 1.2, a few concepts from control theory have to be introduced.

2.1. Receding horizon control

Model predictive control is a control strategy based on receding horizon control (RHC), which is control principle best illustrated using extension to Figure 1.2 shown in Figure 2.1.

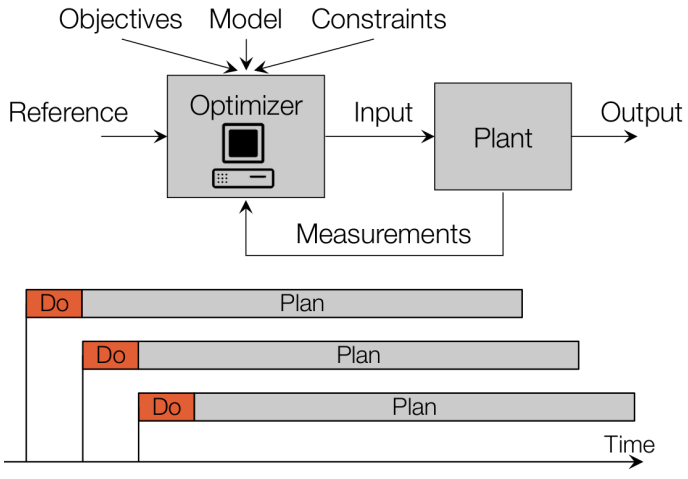

Figure 2.1: Receding horizon control principle [1]

So, the main point is that controller takes into consideration the behaviour of the system over some finite time window (horizon) in the nearest future, with the calculated

control sequence acting along the entire horizon. Then, only the first control action is applied, and the controlled system evolves to the next state, which is again measured or estimated. This way a feedback action is introduced, which accounts for the model innacurateness, and makes RHC actually a closed-loop control principle. Once the state of the system in this new time step is obtained, the whole procedure of computing the control actions for the entire horizon and then applying the first one is repeated.
Regarding the MPC, the above mentioned planning of the control actions sequence on the moving horizon is carried out by solving an optimisation problem. An important part in formulating an optimisation problem is the cost function, which can include both the distance of the system state from the desired reference and the amplitudes of actuators signal, thus introducing the possibility of compromising between strict reference tracking and power consuption. Another MPC benefit of key importance is the possibility to include the state and actuator signal constraints directly in the optimisation problem. This way, the controller actions on the entire horizon will be calculated in a way which minimizes the cost function, but also taking care not to bring the state of the system outside the desired area throughout the process. Also, instead of *ad-hoc* saturation of the actuator signals, MPC takes account of the actuators' limits when formulating the optimisation problem, therefore having a better model of realistic system possibilities over the entire horizon. These aspects distinguish the MPC from the Linear Quadratic Regulator (LQR), which is actually its base.

2.2. Convex optimisation

Since solving the optimisation problem is the key part of the MPC control strategy, its type has of course great implications to the eventual controller feasibility. Although initial classification of the optimisation problems mainly divided them to linear and non-linear ones, modern control theory distiungishes convex and non-convex problem types as a more important classification factor [2]. Theferore, this section briefly introduces the features of convex optimisation problems.
One of the crucial properties of a convex optimisation problem is that every locally optimal solution is also globally optimal. Before we continue with a more formal definition of a convex optimization problem, we first present a standard form of an optimisation

problem as follows:

$$\begin{aligned} &\min_{x \in \mathcal{X}} f_0(x) \\ \text{subject to: } &g_j(x) \leq 0 \quad j = 1, \ldots, m \\ &h_i(x) = 0 \quad i = 1, \ldots, p. \end{aligned} \tag{2.1}$$

Here, $\mathcal{X}$ stands for optimisation problem project variable domain, f for cost function, g_j for inequality constraint function, and h_i for equality constraint function. So, optimisation problem defined by 2.1 is convex if the following holds [1]:

- the problem domain $\mathcal{X}$ is a convex set
- the objective function f_0 is a convex function
- the inequality constraint functions g_j are all convex
- The equality constraint functions $h_i(x) = a_i^\top x - b_i$ are all affine

In this definition, a few more theoretical terms are arising, so their explanations also have to be briefly discussed.

2.2.1. Affine function

Basically, affine function is a composition of a linear function and a translation. In context of linear algebra, linear function is the m-dimensional function of the form $f_m(x) = Ax_n$, where x is a n-dimensional vector, so A is $m \times n$-dimensional matrix. The kernel space of such function spans a linear subspace, and is defined as a set of solutions of vector equation $Ax_n = 0$, of more compactly written as $\{x \mid Ax = 0\}$, as depicted for 2-D space example in Figure 2.2. The important characteristic of such a subspace is that it passes through the origin. Depending on the dimensions of the independent variable vector x and the matrix A, this definition can encompass points, lines, planes or hyperplanes for higher-order dimensionality.

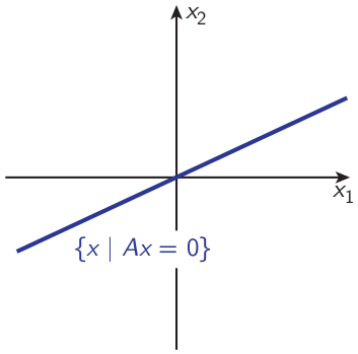

Figure 2.2: A linear subspace example, $\{x \mid Ax = 0\}$ [1]

An affine set is defined by adding a vector of constant offset to the linear subspace, or mathematically written affine set in n-dimensional domain is $\{x \in \mathbb{R}^n \mid Ax = b\}$. For two-dimensional space and one-dimensional equation $Ax = b$, an affine set is a one-dimensional line, as shown in Figure 2.3.

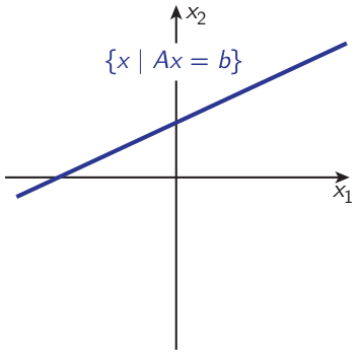

Figure 2.3: An affine set example, $\{x \in \mathbb{R}^n \mid Ax = b\}$ [1]

In an analogous manner as the solution of the linear function formed the linear subspace, an affine subspace can be defined by solving an equation containing an affine function $f(x) = 0$, such that it has the form $f(x) = Ax - b$.

2.2.2. Convex set

A set $\mathcal{X}$ is convex if and only if for any pair of points x and y in $\mathcal{X}$, any convex combination of x and y lies in $\mathcal{X}$, or to express is using mathematical notation:

$$\mathcal{X} \text{ is convex} \quad \Leftrightarrow \lambda x + (1-\lambda)y \in \mathcal{X}, \forall \lambda \in [0,1], \forall x, y \in \mathcal{X}$$

An intuitive interpretation of the statement above is that all line segments with endpoints in the convex set are completely within this set, as visualised in Figure 2.4.

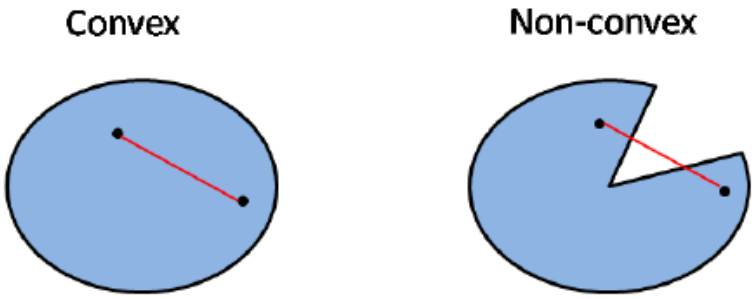

Figure 2.4: Convex vs non-convex set [1]

2.2.3. Convex function

An explanation of a convex set is also a prerequisite for a convex function definition. So, a function is a convex function if and only if its domain is a convex set, and if it satisfies the following property:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y), \quad \forall \lambda \in (0,1), \quad \forall x, y \in \text{dom}(f) \tag{2.2}$$

Geometrically, what the statement above essentially says is that a value of a function between some points which belong to this function has to be below the line connecting these two points. This concept is best visualised with the help of a sketch like the one shown in Figure 2.5.

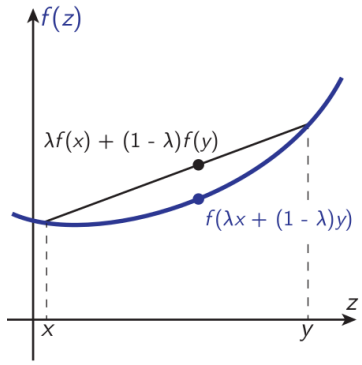

Figure 2.5: Convex function definition

If the function is differentiable, the same principle can be expressed using the gradient of a function. So, if the function is convex, a first order approximation of a function is its global underestimator, as shown in Figure 2.6.

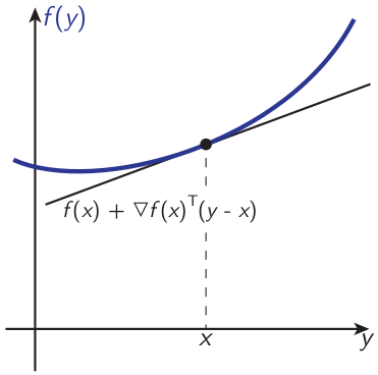

Figure 2.6: Convex function definition for differentiable function

Now when all definitions neccessary to classify a convex optimisation problem are clarified, a convex model predictive control problem can be addressed, as the central topic of this thesis.

2.3. MPC stability and recursive feasibility

A crucial property of any closed-loop controlled dynamical system is its stability, and a process controlled by model predictive controller is not an exception in that

sense. Concerning stability of an autonoumous non-linear system in the vicinity of an equilibrium, a system is Lyapunov stable if it stays forever near this equlibrium point, when initiated from its proximity. If this equlibrium point was a coordinate system origin and δ and ϵ some (small) distances, that would mathematically be written as follows [3]:

$$\forall \varepsilon > 0, \quad \exists \delta > 0, \quad \|x\left(t_0\right)\| < \delta \Rightarrow \|x(t)\| < \varepsilon, \quad t \geq 0 \tag{2.3}$$

Moreover, it is said that $x = 0$ is asymptotically stable equilibrium point if the system converges to it, not only stays in its proximity, or more compactly:

$$\forall \delta > 0, \quad \|x\left(t_0\right)\| < \delta \Rightarrow \|x(t)\| \to 0, \quad t \to \infty \tag{2.4}$$

In his pioneering work on non-linear dynamical system stability, Lyapunov established so called Lyapunov stability criterion, which includes finding a Lyapunov scalar function $V(x)$, such that:

- $V(x) = 0$ if and only if $x = 0$
- $V(x)$ is a positive definite function, i.e. for every $x \neq 0$, $V(x) > 0$
- $\dot{V}(x)$ is negative definite function, i.e. for every $x \neq 0$, $\dot{V}(x) < 0$

If there is such a function with contionous first-order partial derivations, then the autonomous system $\dot{x} = f(x)$ is asymptotically stable [3].
On the other hand, recursive feasibility is not so widespread term in control systems area as the stability, but more specific to the receding horizon optimal control problems with finite horizon. What can happen is that after some time, closed-loop trajectory may lead the system to the states where optimisation problem becomes infeasible. It is important to note that this infeasibility can occur even without model innacurateness or external disturbances. Also, the set of initial conditions from which the MPC-controlled system leads to instability depends mostly on the horizon length. Actually, the problem of infeasibility occurs due to deviation between closed-loop trajectory, and open-loop prediction for the remaining part of the horizon [1]. If control horizon would be infinitely long, then the optimisation result in the initial step would yield a control actions sequence which would be optimal regarding the cost function. Again, with no model mismatch or disturbances, after the control action from the first step was carried out, the system

would end in the second step exactly where the solution from the initial step predicted it. Then in the second step, the control sequence and the predicted system trajectory would exactly match the solution from the first step, because the new optimisation problem is the same as the initial one, but just without the first step. This is actually called the Bellman's principle of optimality, which basically states that the solution of the optimisation problem would not differ if a point on the original solution was chosen as a new starting point. To visualise this, a sketch from Figure 2.7 can help, where the goal of some optimisation problem is to find an optimal trajectory from from the starting point **A** to the goal **J**.

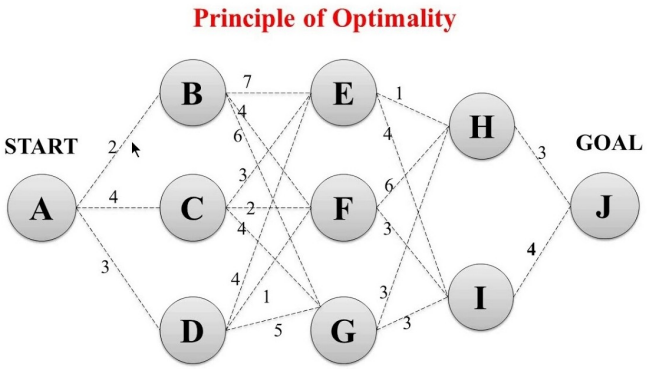

Figure 2.7: Bellman's prinicple of optimality [4]

For example, if the optimal solution would go through points **C**, **F** and **H**, then optimal path to the goal from the point **C** would again pass through the points **F** and **H**. But, in the receding horizon control problem, this is not the case, because the prediction horizon is shifted towards future in every time step (see again Figure 2.1). Therefore, if the horizon is short enough, closed-loop system trajectory can substantially differ from the predictions made in previous time step for the remainig horizon, as shown schematically for two generic systems in Figure 2.8. Consequently, in some cases this may cause the optimisation problem infeasibility even without any model mismatch, and especially when the predition horizons are shorter.

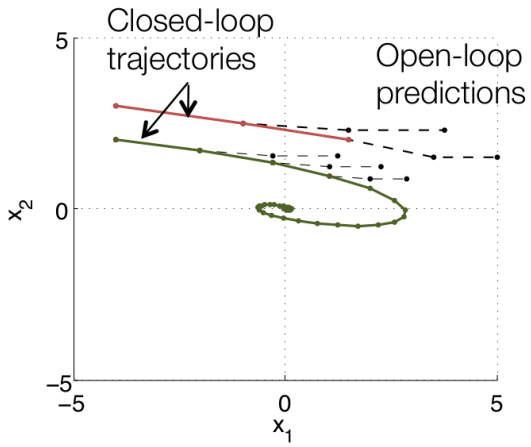

Figure 2.8: Deviation between system trajectory and its prediction from earlier steps for RHC problem [1]

In practice, those problems of stability and infeasibility are often solved by extaneding the prediction horizon and checking the controller perfomance by sampling [1]. The idea behind ensuring these requirements theoretically is to introduce terminal constraints and terminal cost, in way which would substitute the remaining horizon until infinity, thus imitating the infinite horizon control.

2.3.1. Invariant set

In order to be able to define the outcomes of stability and recursive feasibility proof, the notion of the invariant set has to be introduced.
For an autonomous discrete system $x(t+1) = f(x(t))$, a set $\mathcal{O}$ is positively invariant if the system stays forever within this set, when initiated from the point which belongs to it. Mathematically written, this sentence amounts to the following statement:

$$x(0) \in \mathcal{O} \Rightarrow x(t) \in \mathcal{O}, \quad \forall t \in \mathbb{N}_+ \tag{2.5}$$

Then, the positively invariant set which contains all positively invariant sets for some autonomous system is called maximal positively invariant set $\mathcal{O}_\infty$ [1]. If the allowed system states are contained in the set denoted by $\mathcal{X}$, and Ω_i indicates the sets calculated in the intermediate steps, then the maximal invariant set can be obtained using the idea

explained by the following pseudocode [5]:

$$
\begin{aligned}
&Input: \quad \mathcal{X} \\
&Output: \quad \mathcal{O}_\infty \\
&\quad \Omega_0 \leftarrow \mathcal{X} \\
&\quad loop \\
&\quad\quad \Omega_{i+1} \leftarrow \text{pre}\,(\Omega_i) \cap \Omega_i \\
&\quad\quad if \quad \Omega_{i+1} = \Omega_i \quad then \\
&\quad\quad\quad return \quad \mathcal{O}_\infty = \Omega_i \\
&\quad\quad end \quad if \\
&\quad end \quad loop
\end{aligned}
$$

where operator pre (Ω_i) designates the operation of generating preset. In this sense, a preset of a set Ω_i is defined as the set from which all the states evolve to Ω_i in a single time step. So the idea behind the pseudocode above is to generate the set of admissible states as the inital set, and calculate its preset as the next candidate for the invariant set. The procedure is repeated until the candidate is completely within the bounds of its preset, which means by definition that the autonomous system cannot exit that current preset, so for this system this is the positively invariant set. Moreover, since the code is stopped the first time this happens, the current-step preset is then actually the maximal postively invariant set. An example of this process for a 2-D system example is shown in Figure 2.9, where it can be seen that all the trajectories from the maximal positively invariant set converge to the origin without leaving the set, as opposed to the red trajectories which are initiated outside it.

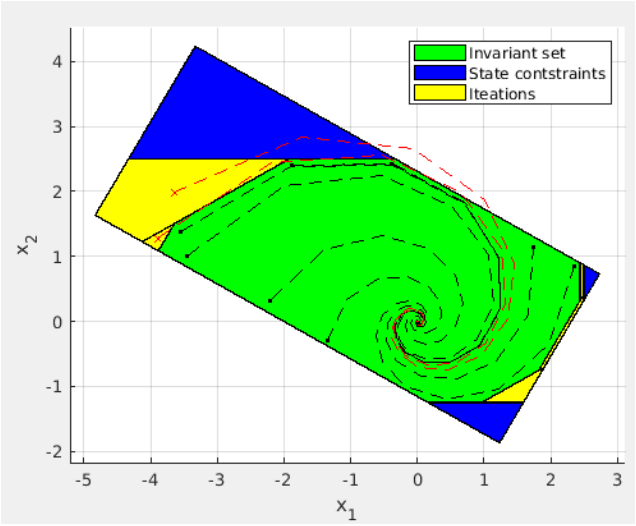

Figure 2.9: Maximal invariant set calculation procedure

2.4. MPC stability theorem

Although the derivation of proof for MPC stability conditions are out of the scope of this thesis, all the prerequisites neccesary to introduce this theorem have been presented. The discrete-time MPC problem from the initial point x_0 is denoted as follows:

$$\begin{aligned} J_0^*(x_0) = \min_{U_0} \quad & p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \\ \text{subj. to} \quad & x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N-1 \\ & x_k \in \mathcal{X}, u_k \in \mathcal{U}, \quad k = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(t) \end{aligned} \tag{2.6}$$

Here, besides the state constraints set $\mathcal{X}$, there is also a special terminal state x_N and control variables u_k constraints sets denoted respectively by $\mathcal{X}_f$ and $\mathcal{U}$. The project variable U_o is here the set of all the inputs u_k from the initial point x_0 over the entire horizon, $U_o = \{u_0, \ldots, u_{N-1}\}$. A and B are state and input matrices for a linear system, respectively. Stage cost is marked by $q(x_k, u_k)$ and the final state cost by $p(x_N)$.
Three assumptions are required for the stability theorem, which then also implies recursive feasibility, because a system cannot be stable or unstable if it's not feasible:

- Stage cost $q(x_k, u_k)$ is a positive definite function, i.e. it is strictly positive and only zero at the origin
- Terminal set $\mathcal{X}_f$ is invariant under some local control law $v(x_k)$, with all the state and inputs constraints satisfied in $\mathcal{X}_f$
- Terminal cost $p(x_N)$ is a continuous Lyapunov function in the terminal set $\mathcal{X}_f$ and satisfies the following relation with the stage cost $q(x_k, u_k)$:

$$p(x_{k+1}) - p(x_k) \leq -q(x_k, v(x_k))$$

Under these assumptions, the closed-loop system under the MPC control law $u_0^*(x)$ is asymptotically stable and the set $\mathcal{X}_f$ is positive invariant for the system $x(k+1) = Ax + Bu_0^*(x)$ [1]. It is important to note that the local control law $v(x_k)$ does not have to be the one which will be realised, the theorem requires only that there is one which can satisfy all of the prescribed constraints. Using this theorem, a MPC control law can be insured to be stable even without long prediction horizons, which can be beneficiary in terms of computational power required for its realisation.

3 Pendulum model

In this chapter the model of the standard control problem of double inverted pendulum on a cart (DIPC) is derived and analysed, with the DIPC system shown in Figure 3.1.

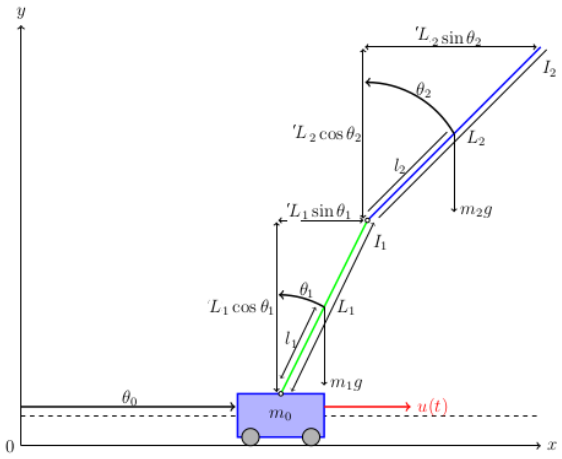

Figure 3.1: Double inverted pendulum on a cart [6]

Marked red in the Figure 3.1, the system input variable $u(t)$ is the force which acts on the cart, and three elements of the state vector $\boldsymbol{\theta}$ are the cart position θ_0, and two angles of the double pendulum links, θ_1 and θ_2 respectively.

3.1. Governing equations

To be able to synthesize any kind of model-based controller, a mathematical model of the controlled system has to be derived. If the frictional forces in DIPC system are considered, it is a non-conservative system with single exogenous input, the control force $u(t)$. The system governing equations can be derived using Langrange equations [7]:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{\theta}}}\right) - \frac{\partial L}{\partial \boldsymbol{\theta}} + \frac{\partial \Phi}{\partial \dot{\boldsymbol{\theta}}} = \mathbf{q} \tag{3.1}$$

Here, the partial derivation of the scalar variable with respect to vector variable as e.g. $\frac{\partial L}{\partial \boldsymbol{\theta}}$ denotes the gradient of the Langranian function $L(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta})$. Also, $\mathbf{q}$ denotes the vector of generalised forces, i.e. the external forces or moments acting along the chosen system generalised coordinates, in our case $\theta_{0,1,2}$, which actually comprise the vector $\boldsymbol{\theta}$. As shown in [7], if frictional forces depend linearly on the time derivatives of generalised coordinates (velocities), they can be inserted into Langrange equations using Rayleigh dissipation function Φ. Although it has a form of energy, physically Φ does not represent an energy [7]. Rayleigh function is defined so that it's derivative with repect to generalised velocity yields generalised forces which arise from viscous friction, denoted by $\boldsymbol{\mu}$:

$$\begin{aligned} \Phi =& \frac{1}{2}(d_0\dot{\theta_0}^2 + d_1\dot{\theta_1}^2 + d_2\dot{\theta_2}^2) \\ \boldsymbol{\mu} =& -\frac{\partial \Phi}{\partial \dot{\boldsymbol{\theta}}} \end{aligned} \tag{3.2}$$

This results in friction vector being comprised of a one frictional force for a cart, and two frictional moments for pendulum links:

$$\boldsymbol{\mu} = \begin{bmatrix} -d_0\dot{\theta}_0 \\ -d_1\dot{\theta}_1 \\ -d_2\dot{\theta}_2 \end{bmatrix} \tag{3.3}$$

Note that these definitions of generalised coordinates and Rayleigh disspiation function imply that the entire system friction is modelled with repect to the apsolute, not relative movement between components. Interpretated physically, this means that for pendulum links the joint friction in disregarded, and air resistance is taken into account. L in 3.1 stands for Langrangian function, which is defined as a difference between system kinetic

and potential energy.

$$L = E_{kin} - E_{pot} \tag{3.4}$$

Firstly, kinetic and potential energy expressions dependent on system coordinates are calculated from DIPC system physical properties, such as cart and link masses $m_{1,2,3}$, dimensions $L_{1,2}$ and rotatinal inertias $I_{1,2}$ (see Figure 3.1). To denote three rigid bodies which comprise the DIPC system, indexes (0), (1) and (2) are used on the variables for kinetic and potential energy E_{kin} and E_{pot}, as in [6]).

$$\begin{aligned} E_{kin} &= E_{kin}^{(0)} + E_{kin}^{(1)} + E_{kin}^{(2)} \\ E_{\text{pot}} &= E_{\text{pot}}^{(0)} + E_{\text{pot}}^{(1)} + E_{\text{pot}}^{(2)} \end{aligned} \tag{3.5}$$

Here, it is important to note that the cart has one degree of freedom and therefore only translational component of kinetic energy, while the pendulum links move planarly. So, the kinetic and potential energy expressions of the cart are simply:

$$\begin{aligned} E_{kin}^{(0)} &= \frac{1}{2} m_0 \dot{\theta}_0^2 \\ E_{\text{pot}}^{(0)} &= 0 \end{aligned} \tag{3.6}$$

Regarding the pendulum links, in order to rightfully use the simple superposition of translational and rotational kinetic energy, the links translational velocity and also rotational inertia around the axis perpendicular to the plane of movement have to be calculated with respect to the center of mass of the each link. To show their contributions to the system kinetic and potential energy, their expressions in Carthesian coordinate system will be converted to the chosen set of generalised coordinates. For the first link, they are equal to:

$$\begin{aligned} E_{kin}^{(1)} &= E_{kin}^{(1)}(\text{trans}) + E_{kin}^{(1)}(\text{rot}) \\ &= \frac{1}{2} m_1 \left(\dot{x}_1^2 + \dot{y}_1^2 \right) + \frac{1}{2} I_1 \dot{\theta}_1^2 \\ &= \frac{1}{2} m_1 \left\{ \left(\frac{d}{dt} \left[\theta_0 + l_1 \sin \theta_1 \right] \right)^2 + \left(\frac{d}{dt} \left[l_1 \cos \theta_1 \right] \right)^2 \right\} + \frac{1}{2} I_1 \dot{\theta}_1^2 \\ &= \frac{1}{2} m_1 \dot{\theta}_0^2 + \frac{1}{2} \left(m_1 l_1^2 + I_1 \right) \dot{\theta}_1^2 + m_1 l_1 \dot{\theta}_0 \dot{\theta}_1 \cos \theta_1 \\ E_{\text{pot}}^{(1)} &= m_1 g y_1 \\ &= m_1 g l_1 \cos \theta_1 \end{aligned} \tag{3.7}$$

Regarding the second link, the kinetic energy expression is:

$$\begin{aligned} E_{\text{kin}}^{(2)} =& \frac{1}{2} m_2 \left(\dot{x}_2^2 + \dot{y}_2^2\right) + \frac{1}{2} I_2 \dot{\theta}_2^2 \\ =& \frac{1}{2} m_2 \left\{ \left(\frac{d}{dt} \left[\theta_0 + L_1 \sin\theta_1 + l_2 \sin\theta_2\right] \right)^2 + \left(\frac{d}{dt} \left[L_1 \cos\theta_1 + l_2 \cos\theta_2\right] \right)^2 \right\} \\ &+ \frac{1}{2} I_2 \dot{\theta}_2^2, \end{aligned} \tag{3.8}$$

which after some mathematical manipulation simplifies into:

$$\begin{aligned} E_{\text{kin}}^{(2)} =& \frac{1}{2} m_2 \dot{\theta}_0^2 + \frac{1}{2} m_2 L_1^2 \dot{\theta}_1^2 + \frac{1}{2} \left(m_2 l_2^2 + I_2\right) \dot{\theta}_2^2 + m_2 L_1 \dot{\theta}_0 \dot{\theta}_1 \cos\theta_1 \\ &+ m_2 l_2 \dot{\theta}_0 \dot{\theta}_2 \cos\theta_2 + m_2 L_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos\left(\theta_1 - \theta_2\right). \end{aligned} \tag{3.9}$$

The expression for its potential energy is again relatively simple:

$$\begin{aligned} E_{\text{pot}}^{(2)} &= m_1 g y_2 \\ &= m_1 g \left(L_1 \cos\theta_1 + l_2 \cos\theta_2\right) \end{aligned} \tag{3.10}$$

After potential and kinetic energy expressions for all system components are derived, an expression for Lagrangian can be formed using 3.4, and subsequently its partial and time derivatives required to resolve the left-hand side of the Lagrange equation 3.1. For the right-hand side, a vector of generelised forces is very simple, because there is only one external force, which acts on a cart exactly in the direction of the first generalised coordinate θ_0:

$$\mathbf{q} = \begin{pmatrix} u(t) \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u(t)$$

After everything is inserted into Lagrange equations 3.1, and if $l_{1,2}$ are taken as auxilliary parameters equal to half of the pendulum links lengths $l_{1,2} = 0.5 L_{1,2}$, the final set of system governing equations is the following:

$$\begin{aligned} \left(m_0 + m_1 + m_2\right) \ddot{\theta}_0 + \left(m_1 l_1 + m_2 L_1\right) \cos\theta_1 \ddot{\theta}_1 + m_2 l_2 \cos\theta_2 \ddot{\theta}_2 \\ - \left(m_1 l_1 + m_2 L_1\right) \sin\theta_1 \dot{\theta}_1^2 - m_2 l_2 \sin\theta_2 \dot{\theta}_2^2 = u(t) - d_0 \dot{\theta}_0 \\ \left(m_1 l_1^2 + m_2 L_1^2 + I_1\right) \ddot{\theta}_1 + \left(m_1 l_1 + m_2 L_1\right) \cos\theta_1 \ddot{\theta}_0 \\ + m_2 L_1 l_2 \cos\left(\theta_1 - \theta_2\right) \ddot{\theta}_2 + m_2 L_1 l_2 \sin\left(\theta_1 - \theta_2\right) \dot{\theta}_2^2 \\ - g \left(m_1 l_1 + m_2 L_1\right) \sin\theta_1 = -d_1 \dot{\theta}_1 \\ m_2 l_2 \cos\theta_2 \ddot{\theta}_0 + m_2 L_1 l_2 \cos\left(\theta_1 - \theta_2\right) \ddot{\theta}_1 + \left(m_2 l_2^2 + I_2\right) \ddot{\theta}_2 \\ - m_2 L_1 l_2 \sin\left(\theta_1 - \theta_2\right) \dot{\theta}_1^2 - m_2 g l_2 \sin\theta_2 = -d_2 \dot{\theta}_2 \end{aligned} \tag{3.11}$$

The derivation of the left-hand side of equation can also be done using *Matlab Symbolic Toolbox*, with the code shown in Appendix A. As expected, it can be seen that this is highly non-linear second-order system. The idea now is to write it down in more compact form and then convert it to first-order system. If the $\boldsymbol{\theta}$ is chosen as a vector representing state variables $\theta_{0,1,2}$, the equation 3.11 can be rewritten in the vectorial form:

$$\mathbf{D}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C_0}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{G}(\boldsymbol{\theta}) = \mathbf{H}u + \boldsymbol{\mu}\dot{\boldsymbol{\theta}} \tag{3.12}$$

To simplify the equations, the auxilliary pendulum half-lengths are expressed in terms of full lengths, and link inertias are expressed using their definition for rods, regarding the axis perpendicular to the plane of motion and going through the center of mass:

$$I_{1,2} = \frac{1}{12} m_{1,2} L_{1,2}^2$$

If the frictional forces $\boldsymbol{\mu}(\dot{\boldsymbol{\theta}})$ are put together with the terms under the matrix $\mathbf{C_0}$, then the new matrix C has the following form:

$$\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \mathbf{C_0}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \boldsymbol{d}I,$$

where I denotes the identity matrix. This way, the vector-form of governing equations is sligtlhy reformulated:

$$\mathbf{D}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{G}(\boldsymbol{\theta}) = \mathbf{H}u, \tag{3.13}$$

with the matrices $\mathbf{D}(\boldsymbol{\theta})$, $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, $\mathbf{G}(\boldsymbol{\theta})$ and $\mathbf{H}$ having the following form:

$$\mathbf{D}(\boldsymbol{\theta}) = \begin{bmatrix} m_0 + m_1 + m_2 & \left(\frac{1}{2}m_1 + m_2\right) L_1 \cos\theta_1 & \frac{1}{2}m_2 L_2 \cos(\theta_1 + \theta_2) \\ \left(\frac{1}{2}m_1 + m_2\right) L_1 \cos\theta_1 & \left(\frac{1}{3}m_1 + m_2\right) L_1^2 & \frac{1}{2}m_2 L_1 L_2 \cos(\theta_1 - \theta_2) \\ \frac{1}{2}m_2 L_2 \cos\theta_2 & \frac{1}{2}m_2 L_1 L_2 \cos(\theta_1 - \theta_2) & \frac{1}{3}m_2 L_2^2 \end{bmatrix}$$

$$\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{bmatrix} d_0 & -\left(\frac{1}{2}m_1 + m_2\right) L_1 \sin\theta_1 \dot{\theta}_1 & -\frac{1}{2}m_2 L_2 \sin\theta_2 \dot{\theta}_2 \\ 0 & d_1 & \frac{1}{2}m_2 L_1 L_2 \sin(\theta_1 - \theta_2)\dot{\theta}_2 \\ 0 & -\frac{1}{2}m_2 L_1 L_2 \sin(\theta_1 - \theta_2)\dot{\theta}_1 & d_2 \end{bmatrix}$$

$$\mathbf{G}(\boldsymbol{\theta}) = \begin{bmatrix} 0 \\ -\left(\frac{1}{2}m_1 + m_2\right) L_1 g \sin\theta_1 \\ -\frac{1}{2}m_2 g L_2 \sin\theta_2 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$$

(3.14)

Nextly, if the arguments are now disregarded for the sake of compactness, the highest-order derivative of $\boldsymbol{\theta}$ is as follows:

$$\ddot{\boldsymbol{\theta}} = -\mathbf{D}^{-1}\mathbf{C}\dot{\boldsymbol{\theta}} - \mathbf{D}^{-1}\mathbf{G} + \mathbf{D}^{-1}\mathbf{H}u \tag{3.15}$$

To convert the second-order system 3.15 to the first-order system of equations, a standard procedure of extending the state vector with the first derivatives is employed. Therefore, the new expanded state vector $\mathbf{x}$ is defined in the following manner:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \end{bmatrix}.$$

If this state vector redefinition is used for restating the system equations 3.4, the result is:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & I \\ 0 & -\mathbf{D}^{-1}\mathbf{C} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ -\mathbf{D}^{-1}\mathbf{G} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{D}^{-1}\mathbf{H} \end{bmatrix} u \tag{3.16}$$

Compactly written, the goverening equation is:

$$\dot{\mathbf{x}} = \tilde{\mathbf{A}}(\mathbf{x})\mathbf{x} + \tilde{\mathbf{B}}(\mathbf{x})u + \mathbf{L}(\mathbf{x}) \tag{3.17}$$

with matrix terms:

$$\tilde{\mathbf{A}}(\mathbf{x}) = \begin{bmatrix} 0 & I \\ 0 & -\mathbf{D}^{-1}\mathbf{C} \end{bmatrix}, \quad \tilde{\mathbf{B}}(\mathbf{x}) = \begin{bmatrix} 0 \\ \mathbf{D}^{-1}\mathbf{H} \end{bmatrix}, \quad \mathbf{L}(\mathbf{x}) = \begin{bmatrix} 0 \\ -\mathbf{D}^{-1}\mathbf{G} \end{bmatrix} \tag{3.18}$$

3.1.1. Alternative formulation

Alternatively, the angle of the second pendulum link can be defined relatively to the first one, as shown in Figure 3.2. The advantage of this approach is that now the same Rayleigh dissipation function 3.2 describes the frictional generalised forces as moments which damp the pendulum oscilation proportional to their relative rotational velocity. As explained earlier, with the first approach, the entire friction was modelled with regard to pendulum absolute anguar velocity, which is physically less plausible. The code used for deriving these equations using *Matlab Symbolic Toolbox* is shown in Appendix B.

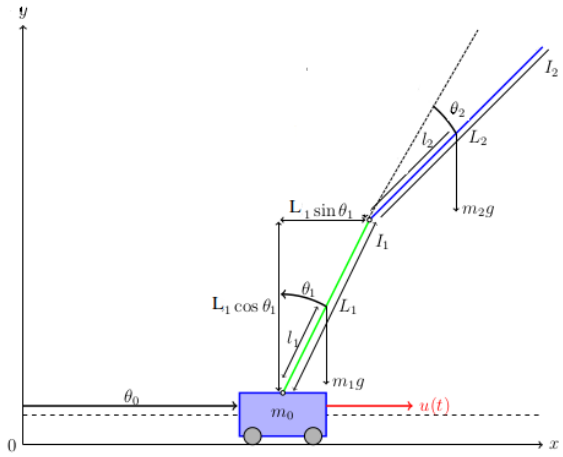

Figure 3.2: Double inverted pendulum on a cart alternative model derivation

The resulting equation looks the same as the original one 3.13, but the the matrices $\mathbf{D}(\boldsymbol{\theta})$, $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, $\mathbf{G}(\boldsymbol{\theta})$ and $\mathbf{H}$ differ slightly, assuming the following form (cf 3.14):

$$
\mathbf{D}(\boldsymbol{\theta}) = \begin{bmatrix} m_0 + m_1 + m_2 & \left(\frac{1}{2}m_1 + m_2\right) L_1 \cos\theta_1 + \frac{1}{2}m_2 L_2 cos(\theta_1 + \theta_2) & \frac{1}{2}m_2 L_2 \cos(\theta_1 + \theta_2) \\ \left(\frac{1}{2}m_1 + m_2\right) L_1 \cos\theta_1 & \left(\frac{1}{3}m_1 + m_2\right) L_1^2 & \frac{1}{2}m_2 L_1 L_2 \cos(\theta_1 - \theta_2) \\ \frac{1}{2}m_2 L_2 \cos\theta_2 & \frac{1}{2}m_2 L_1 L_2 \cos(\theta_1 - \theta_2) & \frac{1}{3}m_2 L_2^2 \end{bmatrix}
$$

$$
\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{bmatrix} d_0 & -\left(\frac{1}{2}m_1 + m_2\right) L_1 \sin\theta_1 \dot{\theta}_1 - m_2 L_2 sin(\theta_1 + \theta_2)(\frac{1}{2}\dot{\theta}_1 + \dot{\theta}_2) & -\frac{1}{2}m_2 L_2 \sin(\theta_1 + \theta_2)\dot{\theta}_2 \\ 0 & d_1 & -m_2 L_1 L_2 \sin\theta_2 (\frac{1}{2}\dot{\theta}_2 + \dot{\theta}_1) \\ 0 & \frac{1}{2}m_2 L_1 L_2 \sin\theta_2 \dot{\theta}_1 & d_2 \end{bmatrix}
$$

$$
\mathbf{G}(\boldsymbol{\theta}) = \begin{bmatrix} 0 \\ -(\frac{1}{2}m_1 + m_2) L_1 g sin(\theta_1) - \frac{1}{2}m_2 g L_2 sin(\theta_1 + \theta_2) \\ -\frac{1}{2}m_2 g L_2 sin(\theta_1 + \theta_2) \end{bmatrix}
$$

$$
\mathbf{H} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}
$$

(3.19)

In order to test both model derivations, the simulation of the DIPC system oscilating freely around pendulums ponited downwards has been put through, with the same parameters and the same initial conditions. Since in both derivations coordinates θ_0

and θ_1 are defined in the same way, comparison of their trajectories is shown in Figure 3.3, where it can be seen that the two models definitely do not behave identically.

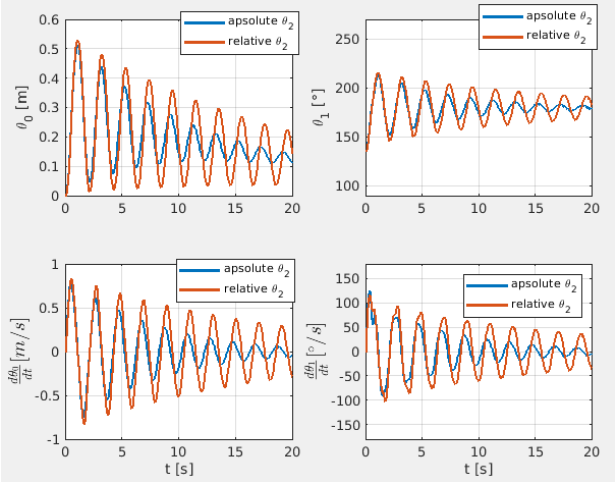

Figure 3.3: θ_0 and θ_1 trajectories comparison

The explanation for this discrepancy lies in the aforementioned different definition of the damping. While in the original model [6], it is proportional to the absolute angular velocity of the pendulum, in the second case it depends on the relative angular velocity between the links. If the damping of the second link is disregarded, the two models behave the same way, as shown in Figure 3.4.

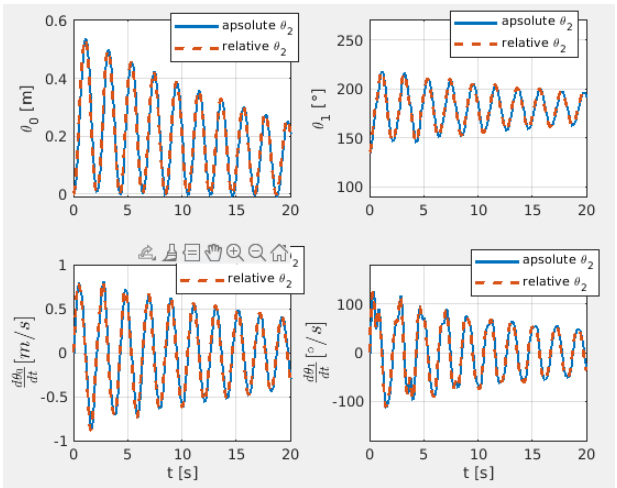

Figure 3.4: θ_0 and θ_1 trajectories comparison, $d_2 = 0$

3.2. Model linearisation

The theory of system dynamics and control is much more developed and versatile for linear systems compared to the general non-linear type. Therefore, it is often desired to obtain a linear model which describes the system dynamics accurately enough. To do so, model-governing equations 3.11 or 3.17 can be reformulated as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.20}$$

Then, model linearised around some equilibrium state $\mathbf{x_0}$ and corresponding stationary input signal $\mathbf{u_0}$ can be derived by computing Jacobian matrices $\mathbf{A}_J$ and $\mathbf{B}_J$ of the vector equation 3.20.

$$\begin{aligned} \mathbf{A}_J &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}_0 \\ \mathbf{u}=u_0}} \\ \mathbf{B}_J &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}_0 \\ \mathbf{u}=u_0}} \end{aligned} \tag{3.21}$$

This way, a new linear model is of the form:

$$\dot{\mathbf{x}} = \mathbf{A}_J\mathbf{x} + \mathbf{B}_J\mathbf{u} \tag{3.22}$$

The DIPC system from Figure 3.2 has two distinct equlibrium points, one stable with both links pointed downwards, and one unstable with both links in upward position. Although the first goal of the control system is to keep the pendulum in upward position, linear and non-linear passive models cannot be compared in the vicinity of that position, because it's unstable. This feature will make the passive system leave the state around which it was linearised, i.e. the pendulums will collapse. Therefore, the comparison of the full model and it's linearised counterpart will be carried out by observing free oscilations around the downward position. More specifically, to induce the non-linearities in second pendulum link movement equations, the simulation initial conditions will be with the first link being displaced from the stationary downward position and the second link aligned with it, like shown in Figure 3.5, with θ_2 equal to zero.

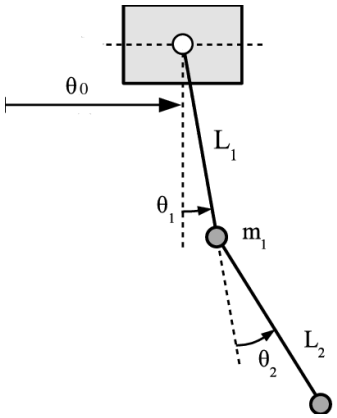

Figure 3.5: Free oscilations for comparison of non-linear and linearised model

The complete procedure of deriving the linear model using *Matlab Symbolic Toolbox* is shown in Appendix C. Since the second, relative θ_2 definition (Figure 3.2) was used and the linearisation was done around pendulum-down position, θ_0 and θ_2 oscilate around zero in both non-linear and linearised model, and therefore these two coordinates will be used for comparison. Firstly, the results are shown for the large initial θ_1 angle of 45°, and then in decreasing order until initial θ_1 equal to 5°.

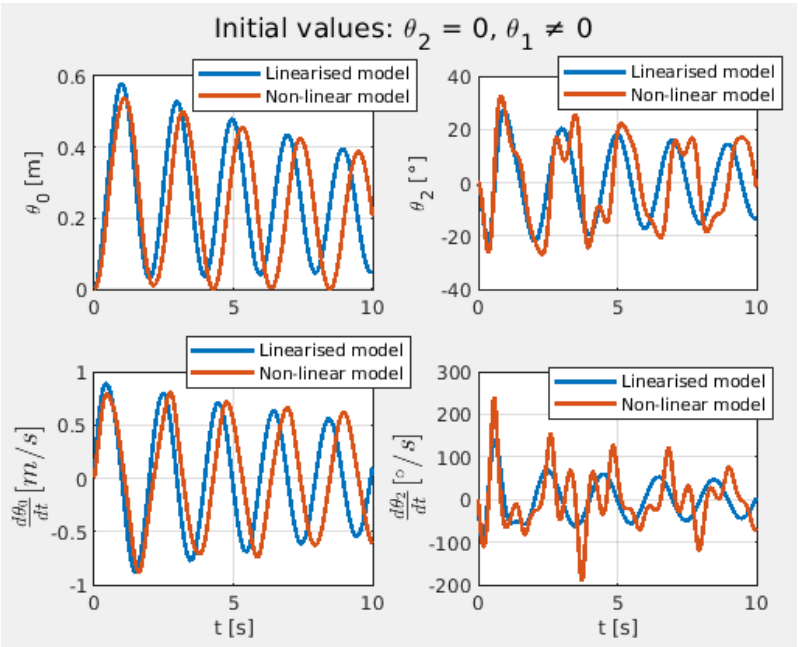

Figure 3.6: θ_0 and θ_2 in linear and non-linear model comparison, $\theta_{1,init} = 45°$

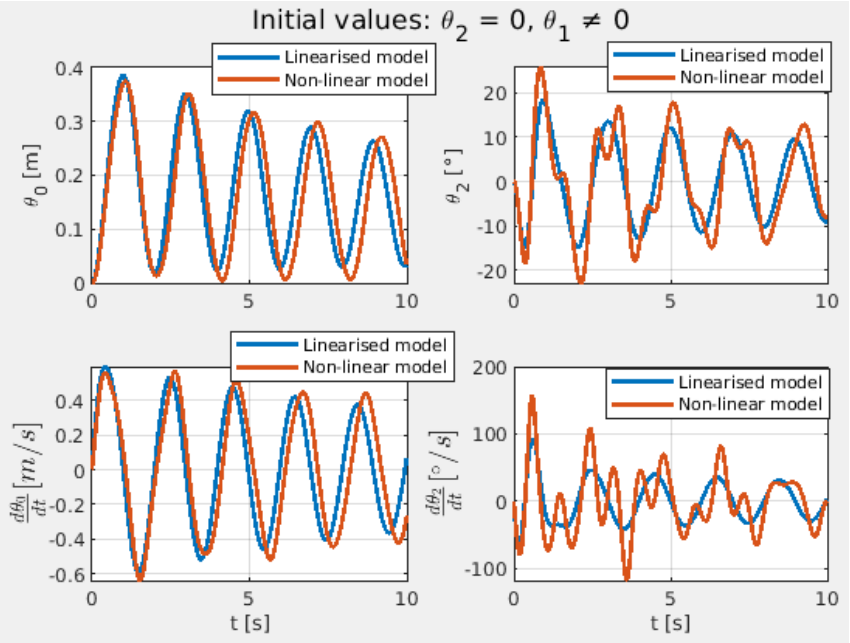

Figure 3.7: θ_0 and θ_2 in linear and non-linear model comparison, $\theta_{1,init} = 30°$

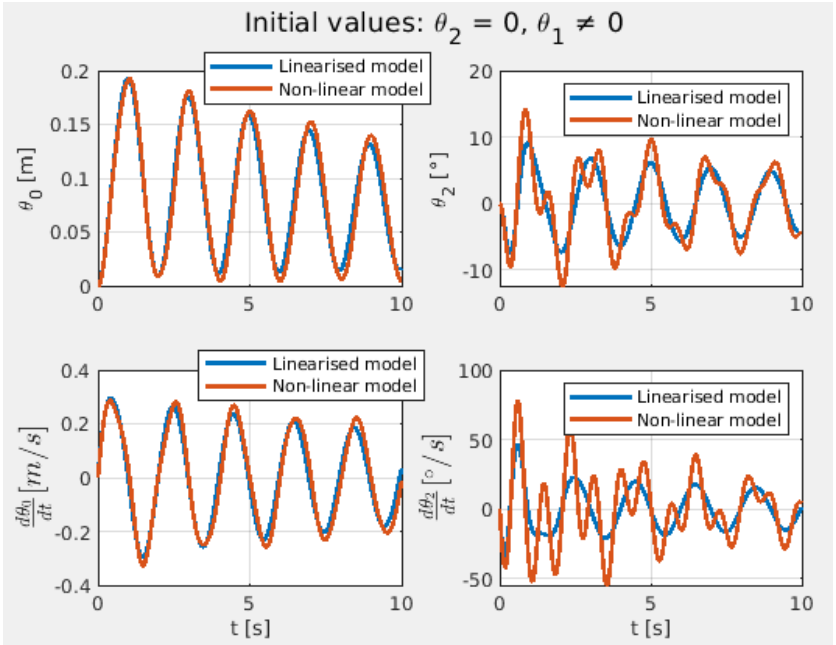

Figure 3.8: θ_0 and θ_2 in linear and non-linear model comparison, $\theta_{1,init} = 15°$

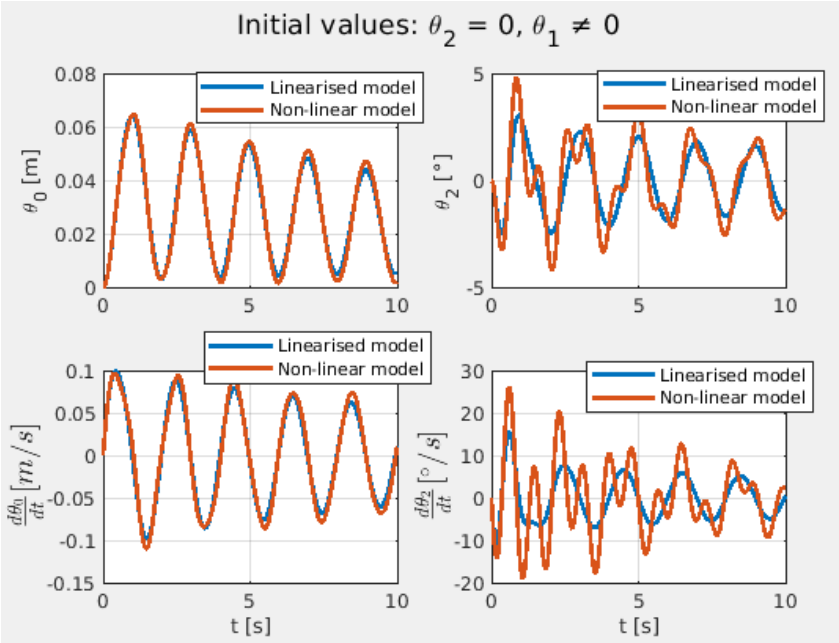

Figure 3.9: θ_0 and θ_2 in linear and non-linear model comparison, $\theta_{1,init} = 5°$

Looking at these figures, it can be seen that considerable amount of linearised model mismatch compared to the non-linear one present at large oscilation angles such as 45°, gradually lowers as the initial inclination is decreased, which can be described as expected. Also, it is interesting to note that the non-linear dynamics of the second link are also significant at the lowest oscilation angles below 5°, which can best be seen on the bottom-right plot, which compares the time derivation of θ_2 from both models. This implies that the model is very non-linear even near the equilibrium point, which definitely poses an additional challange for the control laws based on the linearised model.

4 DIPC system control

Double inverted pendulum is inherently unstable system at the upward oriented equilibrium point, therefore the task of the control algorithm is to stabilize it. Simililarly to the full non-linear system, its version linearised around the upeard position also has unstable equilibrium, which can be confirmed by inspecting the system closed-loop poles, shown in Figure 4.1.

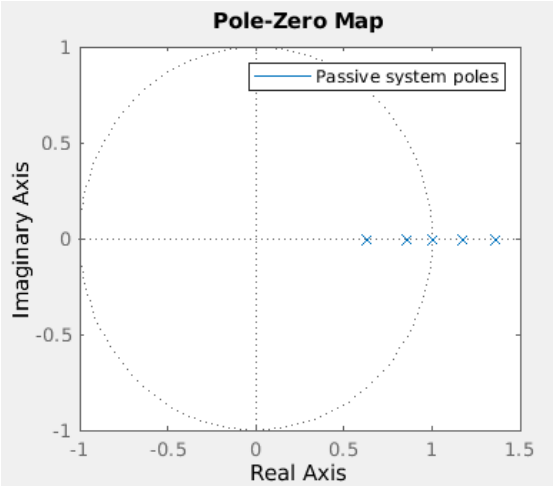

Figure 4.1: Poles of the passive DIPC system, linearised around the upward position

Here the system is discretised using *Matlab c2d* function, because the contoller synthesis is done in discrete domain. Therefore, unstable poles from the right-hand side complex half-plane from the contionous domain transform into poles outside the unit cir-

cle for discrete systems. Hence their presence in pole plot of the passive system linearised around unstable equilibrium confirms the intuition regarding the system instability.

4.1. LQR controller

The first controller applied to the DIPC system in this work is the LQR controller. The term LQR is the abbreviation for linear quadratic regulator, because it deals with linear systems where the optimality is found using quadratic cost function. If the continous-time linear system described with 3.22 is converted into discrete-time system, its generic form is:

$$\mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} + \mathbf{B}\mathbf{u_k} \tag{4.1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are state and input matrices for linear discrete-time state space model, which can be calculated from their continous-time counterparts. Furthermore, the cost function J is here defined in a quadratic form:

$$J = \mathbf{x_N}^T\mathbf{Q_N}\mathbf{x_N} + \sum_{k=0}^{N-1}\left(\mathbf{x_k}^T\mathbf{Q}\mathbf{x_k} + \mathbf{u_k}^T\mathbf{R}\mathbf{u_k}\right) \tag{4.2}$$

where $\mathbf{Q}$ is a symetric positive semi-definite matrix representing the weighting factors for the state vector elements distance from the origin, and the $\mathbf{R}$ symetric positive semi-definite matrix containing weights for input signal vector elements in the optimal control problem. Solution to this problem leads to feedback control law of the form:

$$\mathbf{u_k} = -\mathbf{K}_k\mathbf{x_k} \tag{4.3}$$

where $\mathbf{K}$ stands for LQR gain matrix, derived by solving the recursive Riccati equation. Generally, matrix $\mathbf{K}_k$ changes its values as the time propagates, as marked with index k in the equation 4.3. But in the specific case, where the prediction horizon N is infinite, the LQR gain matrix $\mathbf{K}$ obtains a constant value. This is the case where considering infinite prediction horizon, which is desirable in any optimal control problem, leads to a simpler solution then in the finite-horizon problem. Therefore, this solution will be used in the following analysis of the LQR-controlled DIPC system. As already mentioned, LQR control law is optimal control law for linear systems, therefore $\mathbf{K}$ matrix is computed using linearised discrete-time model. In case of DIPC system, there

is only one input signal, therefore LQR gain matrix for the infinite horizon is now a constant row vector:

$$u_k = -\mathbf{K}\mathbf{x_k} \tag{4.4}$$

So, when this control law is applied to the linearised DIPC system of the form 4.1, the resulting closed-loop system has the dynamics of the autonomous system, determined by the following equation:

$$\mathbf{x_{k+1}} = (\mathbf{A} - \mathbf{BK})\mathbf{x_k} \tag{4.5}$$

As already stated, the goal of a controller is to stabilise the system. By inspecting the closed-loop poles of the system determined by the new state matrix $\mathbf{A} - \mathbf{BK}$, it can be seen that the linearised closed-loop system indeed is stable. As shown in Figure 4.2, all the poles of the discrete-time system are inside the unit circle.

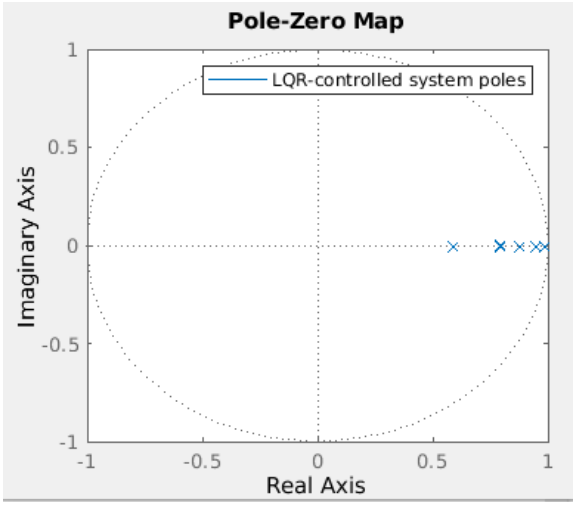

Figure 4.2: Poles of the LQR-controlled linearised DIPC system

Important note here is that this closed-loop stability analysis is only valid for linearised system. As shown in previous chepter DIPC system is highly non-linear, even in the vicinity of the equilibrium point. That is the reason why the LQR performance has to be tested with the full non-linear system, as shown in the following section.

4.1.1. LQR controller performance on the DIPC system

Since the tuning of the LQR controller is out of the scope of this work, only a few examplary cases of values of **Q** and **R** weighting matrices are presented here. As there is only one input signal in the examined DIPC system, the **R** matrix is reduced to a scalar. Regarding the **Q** matrix, its values determine the weights of the each element of the state vector, which is in the case of DIPC system consisted of three coordinates $\theta_{0,1,2}$ and their derivatives $\dot{\theta}_{0,1,2}$:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \end{bmatrix} \tag{4.6}$$

Since the goal of this controller is not to keep the system velocities low while bringing the system into the origin, the relative weights on the $\boldsymbol{\theta}$ are set to a three order of magnitudes greater values than the ones for $\dot{\boldsymbol{\theta}}$:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & 0.001 \times \mathbf{I}_{3x3} \end{bmatrix} \tag{4.7}$$

Similarly to the comparison of non-linear and linearised system, the chosen initial state is both pendulum links aligned and tilted from the upward position, $\theta_2 = 0, \theta_1 \neq 0$. If the initial θ_1 is chosen to be 15°, the system non-linearity is not significant enough to surpass the stabilising property of LQR controller, as shown in Figure 4.3.

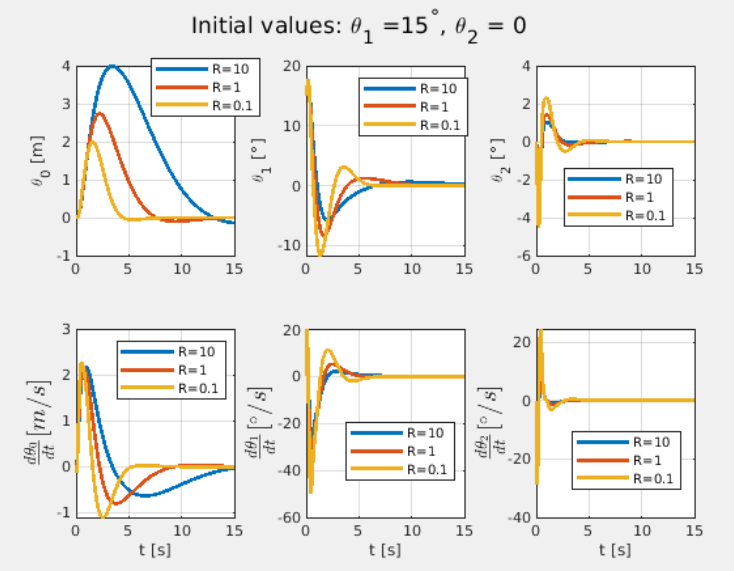

Figure 4.3: LQR controller performance - $\theta_{1,init} = 15°$

This figure also shows the effect of changing the relative weights of R and $\mathbf{Q}$ matrix. In the first case when $R = 10$, the cost of the input signal is order of magnitude larger than the cost of state being out of the desired position, hence the controller returns the system relatively slowly, as best seen on the upper-left plot. The situation is opposite for $10\times$ lower R value, where the controller acts much more aggressively. The difference is also visible when inspecting the input signal magnitudes, as in Figure 4.4.

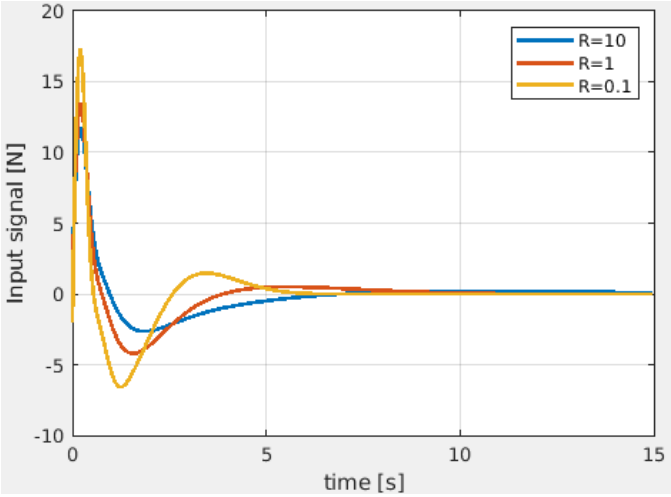

Figure 4.4: LQR controlled input signal comparison for various values of R

Finally, Figure 4.5 shows how limited the application of the linear controller actually is, because the non-linearity of the DIPC system becomes too significant at larger initial angles.

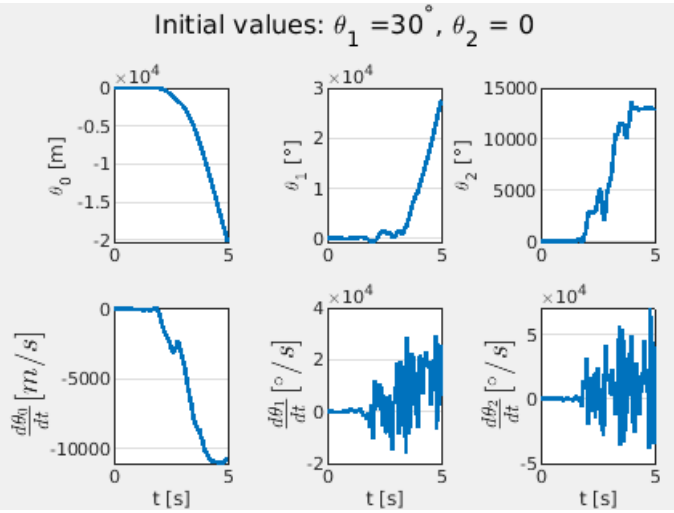

Figure 4.5: LQR controller collapses at $\theta_{1,init} = 30°$

4.1.2. LQR controller with constraints

The limitation coming from the relatively narrow area of applicability of the linearised system implies that the another goal of the controller should be keeping the system state inside some bounds. In the case of DIPC system, these state constraints may be dependent on each other, because at larger θ_1 angles, smaller values of θ_2 can be allowed, in order for linear controller to be able to handle it. Therefore, a set of chosen state constraints are the following:

$$\begin{aligned} |\theta_{1,2}| &< 15° \\ |\theta_1 + \theta_2| &< 20° \end{aligned} \tag{4.8}$$

Graphically, these constraints can be represented with the following polytope:

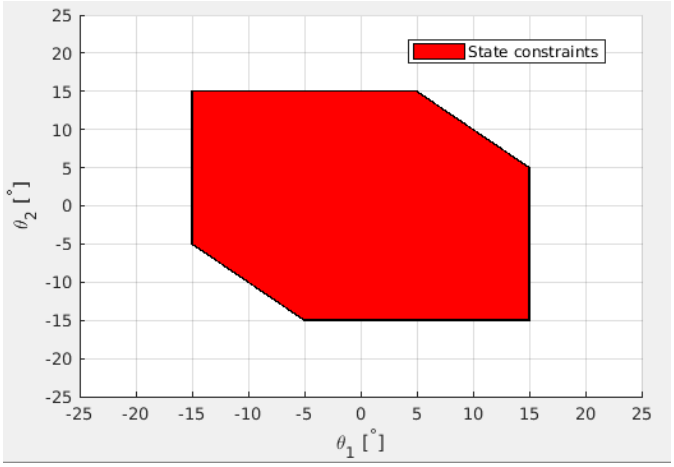

Figure 4.6: Polyhedron representing chosen state constraints

Moreover, hardware limitations often require the input signal to be confined within some boundary value u_{max}. However, since LQR controller is synthesed using the state vector 4.6, LQR gain matrix is actually 1×6 vector. In other words, it depends on all the three coordinates $\theta_{0,1,2}$ and their derivatives $\dot{\theta}_{0,1,2}$. Therefore, the full state vector equivalent of the polyhedron shown in the Figure 4.6 is actually six-dimensional. For some practical reasons, it also makes sense to constrain all the other elements of the state vector. For example, the cart slider is not infinitely long, so θ_0 has to be bounded. Furthermore, it is very likely that the linear model for frictional generelised forces loses its fidelity for very high relative velocities, so it makes sense to put another set of limits on $\dot{\theta}_{0,1,2}$. Adding all these constraints will actually close the 6-dimensional polyhedron from all sides, making it a polytope, or mathematically $|\boldsymbol{x}| < \boldsymbol{x}_{max}$. Since six-dimensional object cannot be visualised, to get the sense of consequences of implementing input signal constraints in LQR controller DIPC system, this 6D polytope can be sliced along the plane spanned by θ_1 and θ_2 coordinates, as shown in Figure 4.7 and Figure 4.8. Here, the red regions represent the above described polytope slice along θ_1 and θ_2 coordinates, while the cyan regions show the same slice, but also with the input signal contraints included. Mathematically, the cyan region is therefore defined as the slice along the polytope defined by:

$$\{\boldsymbol{x} \mid \boldsymbol{x}| < \boldsymbol{x}_{max}\} \cap \{\boldsymbol{x} \mid u = -\mathbf{K}\mathbf{x} < u_{max}\} \tag{4.9}$$

In the first example shown in Figure 4.7, input force maximal value of 40N was used.

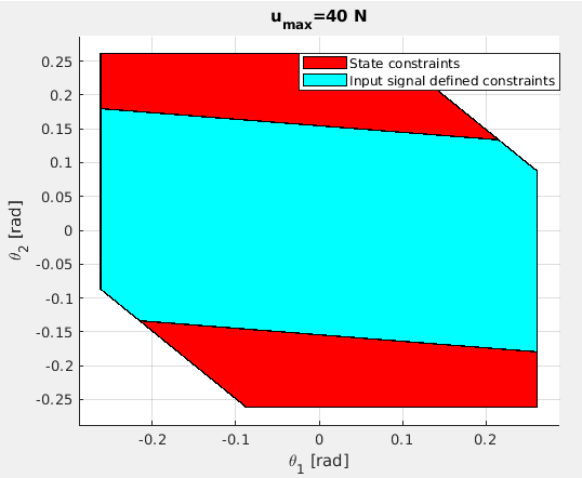

Figure 4.7: State constraints polytope slice along θ_1,θ_2 plane, $u_{max} = 40$N

For comparison, if much stricter input signal limitation of 10N is imposed, the same polytope slice is much thinner:

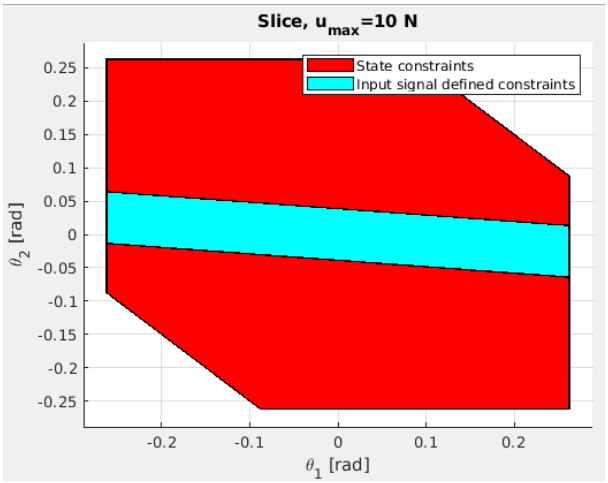

Figure 4.8: State constraints polytope slice along θ_1,θ_2 plane, $u_{max} = 10$N

At first, Figure 4.8 could be explained that with this input signal constraint, θ_1 and θ_2 coordinates neccesarily have to be kept within the narrow band marked cyan. But, since the Figure 4.8 shows the polytope slice, the correct interpretation should be that the cyan set boundaries are valid only if all the other elements of the state vector (θ_0 and $\dot{\theta}_{0,1,2}$) are equal to zero. In a general case, it could be that in some other combinations

of other state variables, LQR controller does not yield the input signal which breaks the imposed limit. To determine if there are any regions of allowed θ_1 and θ_2 states where the input signal constraint is always broken, a 6D polytope has to be projected onto θ_1,θ_2 plane, not sliced along it. In Figure 4.9, this projection of 6D polytope is marked cyan, and it can be seen that it actually covers entire set of θ_1,θ_2 coordinates allowed by initially imposed state constraints (eq. 4.8). In other words, even with the relatively strict input signal constraint of 10N, all of the θ_1,θ_2 pairs allowed by the initial state constraints 4.8 can still be reached without braking the input signal constraint, providing that all the other state variables values are located accordingly, actually matching the second set definition criterion from equation 4.9.

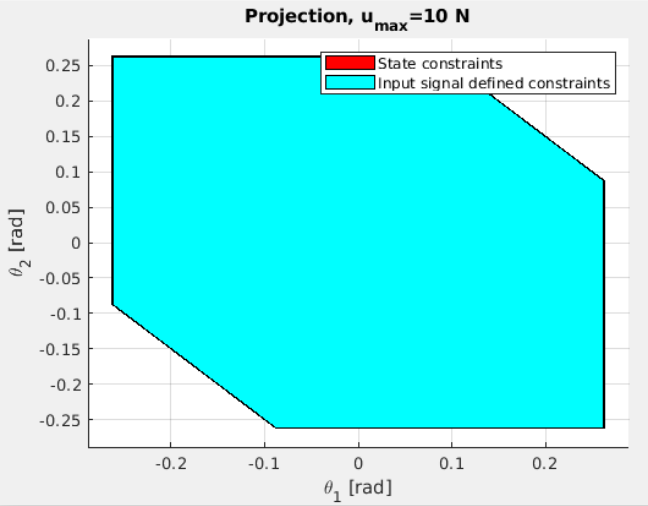

Figure 4.9: Constraints polytope projection on θ_1,θ_2 plane, $u_{max} = 10$N

So, the goal now is to find the set of initial θ_1,θ_2 coordinates, for which the LQR controller will not break any of the imposed state or input constraints during the entire procedure of stabilising the DIPC system in the upward position. To tackle this problem, maximal invariant sets introduced in the second chapter can be used. Since all the other state vector initial values are assumed to be zero, the goal is to find only the 2D cross-section of 6D invariant set, sliced along θ_1,θ_2 plane. By using the logic behind the pseudocode from the second chapter (with its implementation shown in Appendix D), the maximal invariant set for linearised closed-loop DIPC system (eq. 4.5) can be obtained. Its slice along θ_1,θ_2 plane with the chosen input signal limit of $u_{max} = 30$N is

shown in Figure 4.10:

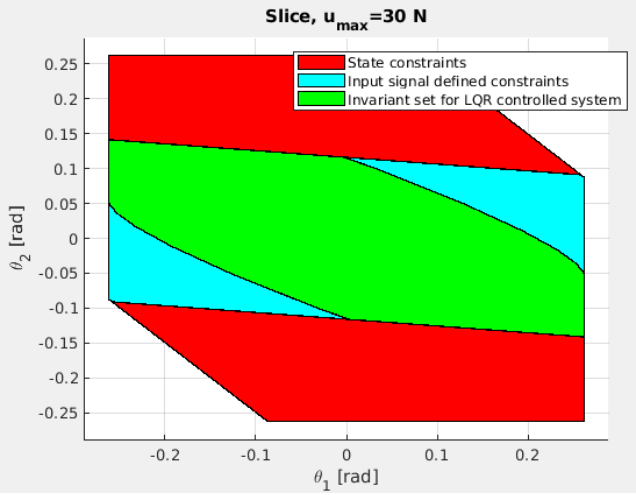

Figure 4.10: Maximal θ_1,θ_2 plane, $u_{max} = 30$N

To implement a discrete-time closed-loop simulation which will illustrate that the derived invariant set shown in Figure 4.10 really is invariant, the full non-linear system has to be discretised. Since the non-linear state space DIPC model is in the continous-time domain, its integration can be done by manual implementation of e.g. fourth order Runge-Kutta method. To check this approach and inspect the required sampling time, the same LQR-controlled initial value problem was simulated using the proposed manual integration algorithm and *Matlab* built-in integration function for general non-stiff problems *ode45*, with the results shown in Figure 4.11 and Figure 4.12.

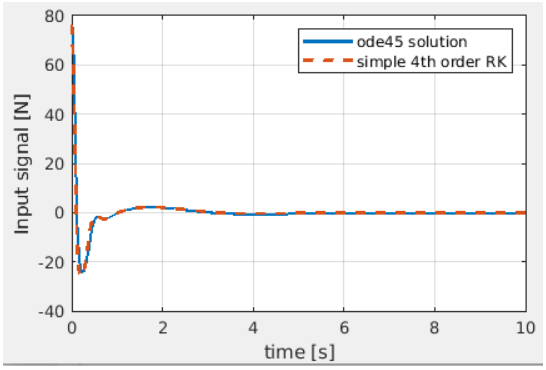

Figure 4.11: Manual implementation of Runge-Kutta integration algorithm vs *Matlab ode45* function, input signal comparison

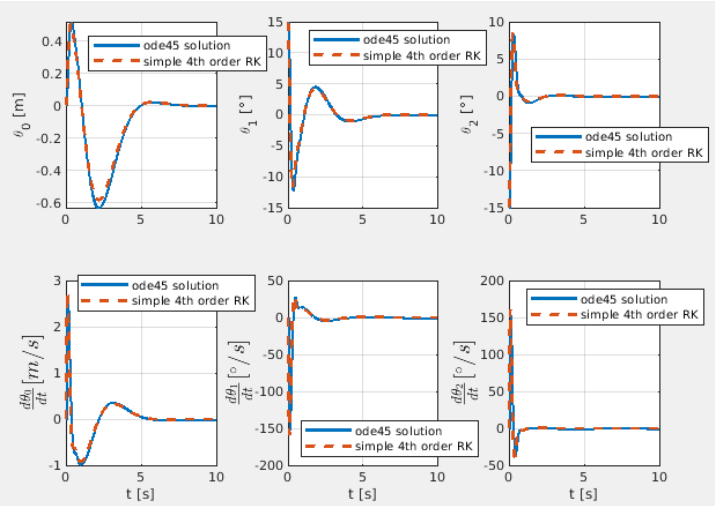

Figure 4.12: Manual implementation of Runge-Kutta integration algorithm vs *Matlab ode45* function, state variables comparison

These two figures suggest that the non-linear DIPC system initial value problem can be integrated using the proposed integration algorithm. However, in some situations with more aggresive input signal value change amplitudes and consequently greater state change from one step to another, the presented simple integration scheme can diverge. For example, if the state measurement noise is not filtered well enough, the state estimation is also somewhat noisy, which directly leads to the input signal richer with higher frequencies, especially if the state feedback controller with high gain is used. In another words, if the input signal is defined as $u = -Kx$, the noiser x directly results in noisier u. The alternative therefore is to use the same *Matlab* built-in integration algorithm repeatedly in every step, with the integration time equal to the timestep duration. The proposed algorithm can now be used for discrete-time simulation, with the θ_1,θ_2 inital values chosen from various locations within the invariant set. As shown in Figure 4.13, the correspoding trajectories support the claim that the system really doesn't exit the invariant set.

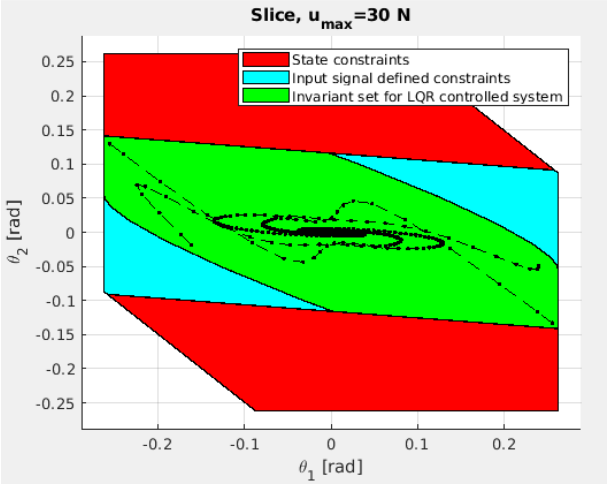

Figure 4.13: θ_1,θ_2 trajectories from various initial values within the invariant set

Also, Figure 4.14 proves that all four trajectories from various corners of invariant set do not break the chosen input and imposed sate constraints (equations 4.8).

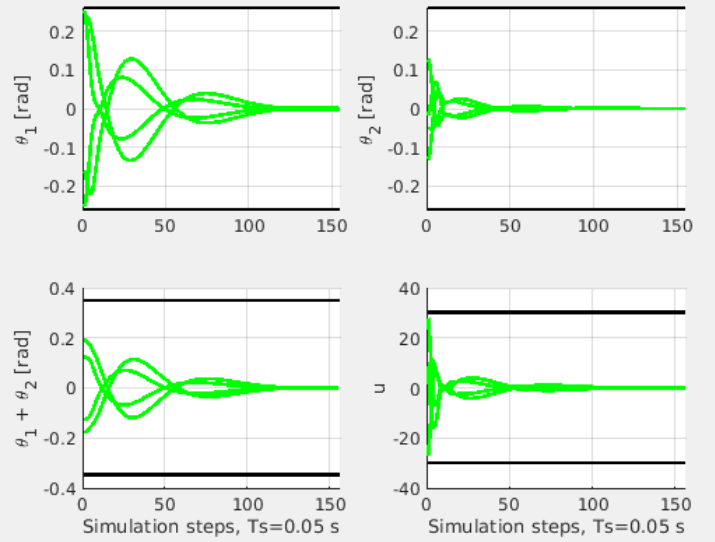

Figure 4.14: θ_1,θ_2 and input signal trajectories and the corresponding constraints

On the other hand, Figure 4.15 and Figure 4.16 show the same plots, but comparing the initial values of θ_1 and θ_2 coordinates, one chosen inside the derived invariant set,

and one outside it.

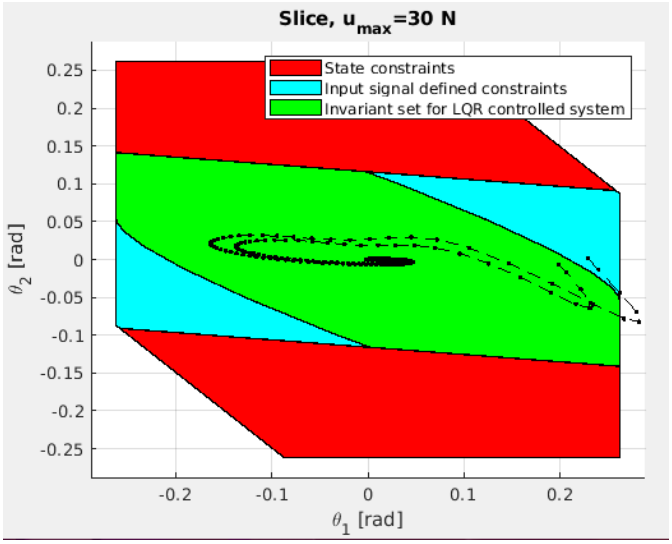

Figure 4.15: θ_1,θ_2 trajectories comparison, relative to the invariant set

Figure 4.16 emphasises the subtle difference between the two trajectories, and it can be seen how the system trajectory, when it's initiated from the area outside the invariant set (marked red in plots), breaks the imposed constraint on the θ_1 coordinate.

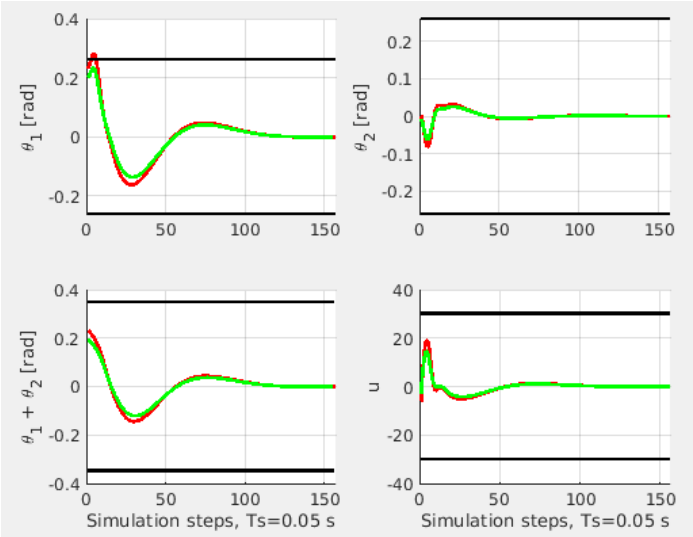

Figure 4.16: θ_1,θ_2 and input signal trajectories comparison of different initial values, and the corresponding constraints

4.2. MPC controller performance on DIPC system

The next step in the optimal controller assessment is the implementation of the Model Predictive Control (MPC) algorithm, already introduced in the initial sections of this thesis. Essentially it is similar to the LQR controller, but with more complicated formulation of the optimisation problem, since it implicitly takes into account also the state and input signal constraints. In the previous section it was shown that the LQR controller can stabilise the DIPC system in the upward position without braking any of the imposed constraints, but only in case of initating the problem from the invariant set. In another words, the region of the attraction of the LQR-controlled system is limited to that set, so the goal of this section is to investigate if the more sophsticated MPC can increase this set of permissible initial values of θ_1 and θ_2 coordinates. Another practical approach to constuct the input signal would be *ad-hoc* saturation of the signal value coming from the LQR controller. But in that case, the original LQR optimality guarantee is lost, because it is no longer a linear problem, which is not taken into account when deriving LQR controller optimal gain. Even more importantly, the system can diverge even from the initial points which satisfy all the constraints, which means that the stability as the crucial property is also possibly compromised. Regarding the MPC controller, using multiparametric programming solution (*Matlab* function *mpsolve*), the MPC region of attraction can be computed. The term region of attraction is here used synonymously with the feasible set, i.e. set of all initial points for which the MPC problem is stable and recursively feasible. In this case, it means that MPC can bring the closed-loop system to the origin while obeying all of the constraints, therefore the term region of attraction is coined. However, since the state vector has 6 elements and state constraint vector has 14 elements, the number of critical regions is very large, so it takes very long for the *Matlab* to find the final attraction region. Therefore, only the short prediction horizons are considered now. Actually, prediction horizon length is here of a key importance - the longer it is, the greater is the attraction region from which the MPC can bring the system to the chosen terminal set, if the one is imposed. For DIPC system MPC controller design, both terminal cost and terminal set was used in the problem formulation, in order to guarantee the MPC stability and the recursive feasibility even for shorter prediction horizons, as discussed in the second chapter. If the LQR-controlled closed-loop system invariant set is set as the terminal set, and the

solution to the discrete algebraic Riccati equation as the terminal cost, it can be proved that with the chosen stage cost function all the three requirements needed for MPC stability theorem are met, as shown in [1].

Initially a discrete time prediction horizon length of two steps is chosen, which corresponds to the horizon of 0.1 second. Since the LQR invariant set is chosen as the terminal one, the MPC controller expands this attraction region, as shown in Figure 4.17. Figure 4.18 confirms that the system initiated from cyan region obeys the imposed constraints.

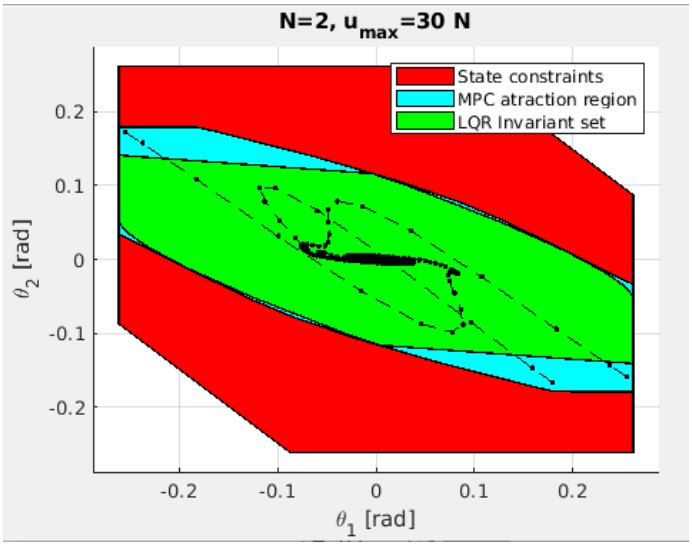

Figure 4.17: MPC attraction region for $N = 2$, compared to LQR

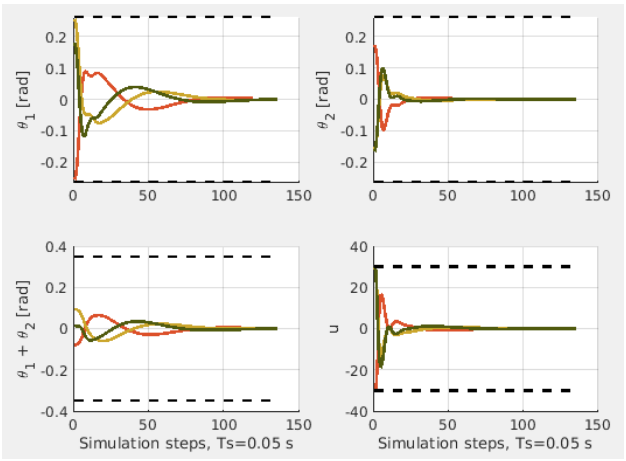

Figure 4.18: MPC-controlled θ_1,θ_2 and input signal trajectories

To ilustrate the importance of the terminal cost and set constraint in such a control problem, the same procedure of stabilizing the DIPC system was simulated without them. Looking at the Figure 4.17, it can be seen that from the initial angles of θ_1,$\theta_2 \approx [0.2, -0.15]$rad, which corresponds to roughly 11° and −8°, MPC with the terminal state constraint can stabilize the system with the prediction horizon length of 2 steps. On the other hand, disregarding the terminal state make the MPC-controlled system with the same weight matrices unstable even for prediction horizon equal to 30, as shown in Figure 4.19.

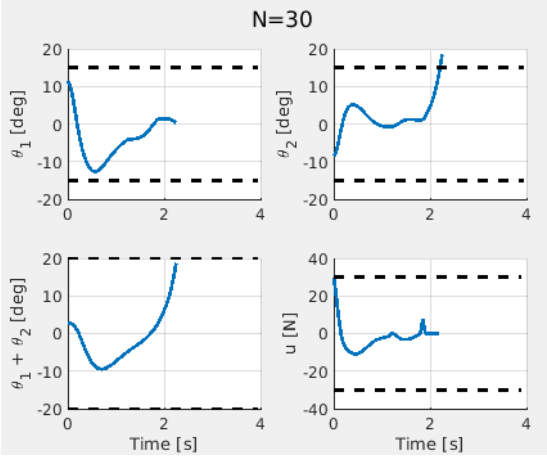

Figure 4.19: MPC-controlled system is unstable with terminal state disregarded, even for longer prediction horizon

This leads to a conclusion that if the linear control problem is of a type where LQR controller is almost suitable, it maybe makes sense to upgrade it using the linear MPC with the LQR solution for terminal cost and state contstraint, thus enabling the use of very short prediction horizons, which possibly does not demand as high computational power increase. If the prediction horizon is increased beyond $N = 2$, the *solvemp* function returns error message that the feasible set is wrong. Further investigation of this problem would involve accessing the implementation details of this built-in *Multi-Parametric Toolbox* function, which is out of the scope of this thesis.

4.2.1. MPC reference tracking

Another type of control problem which often occurs in practice is the desired state reference tracking. Regarding the MPC controller, there are generally two possible cases:

- the reference on the entire horizon is constant
- the variable reference trajectory is known for the entire horizon.

The first case appears in a situation where the reference is constant exogeneus input, like from e.g. human operator. For example, if the vehicle yaw rate reference in the torque vectoring system is calculated from the vehicle speed and steering wheel angle as the instantenous data without using any of the prediction algorithms, they are constant for the entire horizon. This is is so because the controller can not know in advance if the driver intends to keep the vehicle in the same state permanently or not. In the same example of torque vectoring, the changing reference can occur in the case of autonomous driving, where some superpositioned motion planning algorithm sets the vehicle yaw rate reference, and then it can be time-varying on the prediction horizon. In case of DIPC system, the cart position coordinate θ_0 reference tracking is an example of control problem which can be used to analyse the MPC controller capabilities.
If the first case from the list above with the constant reference on the entire horizon is studied, the MPC performance is shown on upper-most plot in Figure 4.20. The second and third row of the same figure confirm that neither of the imposed state or input signal contraint is violated. Also, the upper-most plot shows that the controller is rather conservative, in the sense that it keeps the DIPC system very far from any of the constraints, and therefore guiding the system from one to another reference position relatively slowly. In an attempt to speed up the closed-loop system response, first logical step is to tune the objective function by decreasing the stage cost of the signal value, denoted R. If R is decreased from the initial 0.1 to the value of 0.005, the resulting controller is much more aggresive, as shown in Figure 4.21. This can be seen by comparing the second and third row from the Figure 4.21 with the ones from 4.20, where an increase in both of the control signal and system coordinates trajectory amplitudes can be detected.

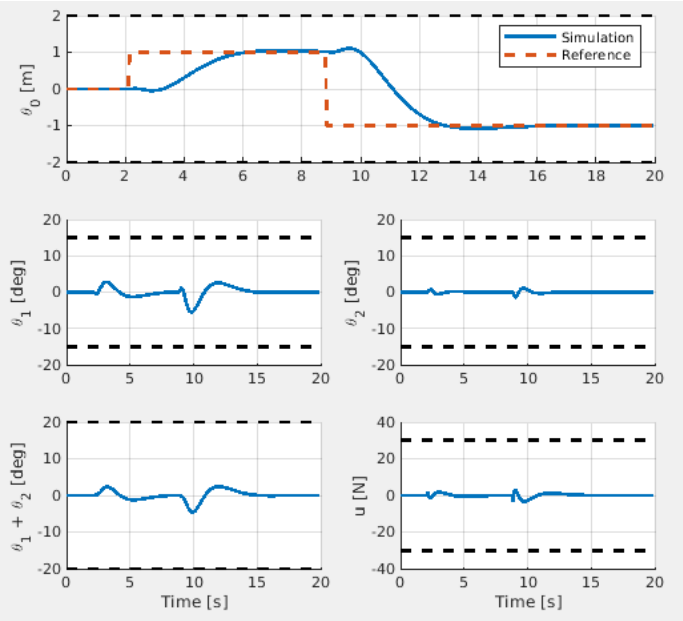

Figure 4.20: MPC θ_0 reference tracking with the constant reference on the prediction horizon, $R = 0.1$

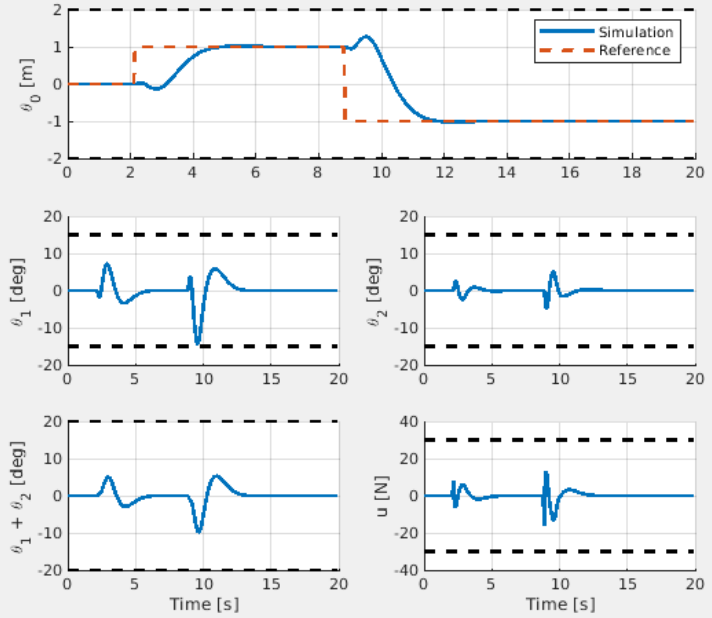

Figure 4.21: MPC θ_0 reference tracking with the constant reference on the prediction horizon, $R = 0.005$

However, if the weight factor R is decreased further to the value of $R = 0.0001$, it causes the problem infeasibility at around 9.5s, as shown in Figure 4.22. As mentioned earlier, the too short prediction horizon causes the optimal solution to guide the system into the region where eventually no feasible solution can be found. As explained earlier, it can happen even if the model is theoretically perfect, but in this case the controller synthesis is based on the linearised model, whereas the closed-loop simulation is done on the full non-linear model.

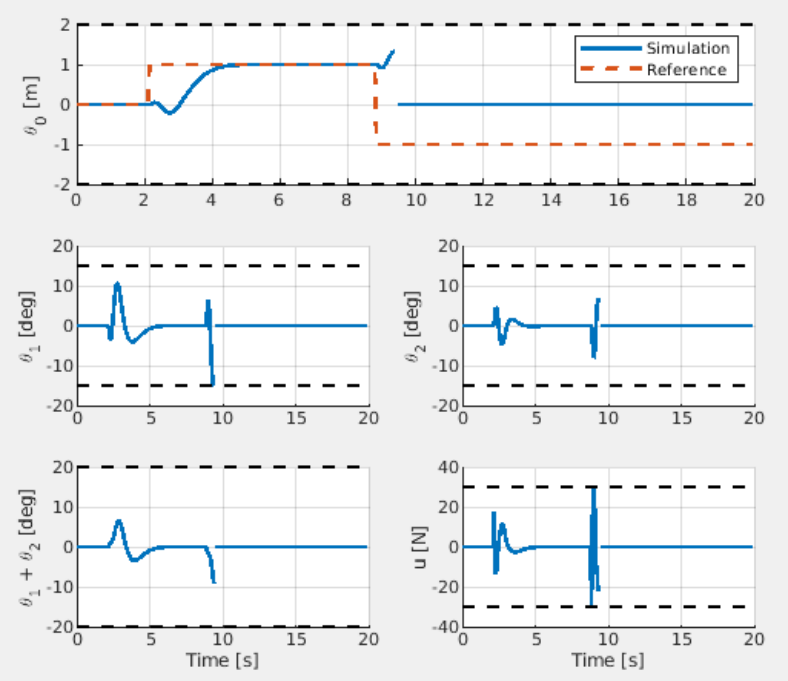

Figure 4.22: MPC infeasibility problem with the constant reference on the prediction horizon, $R = 0.0001$

So, if the plant model is non-linear and no terminal state cost and constraints are used, the choice of prediction horizon length should be done simultaneously with the controller tuning in the simulation phase, in order to avoid the controlled system failures. The second type of the MPC reference tracking problem is the one where the change in reference is known in advance. This is where the full potential of this control technique can come to light, as shown in Figure 4.23.

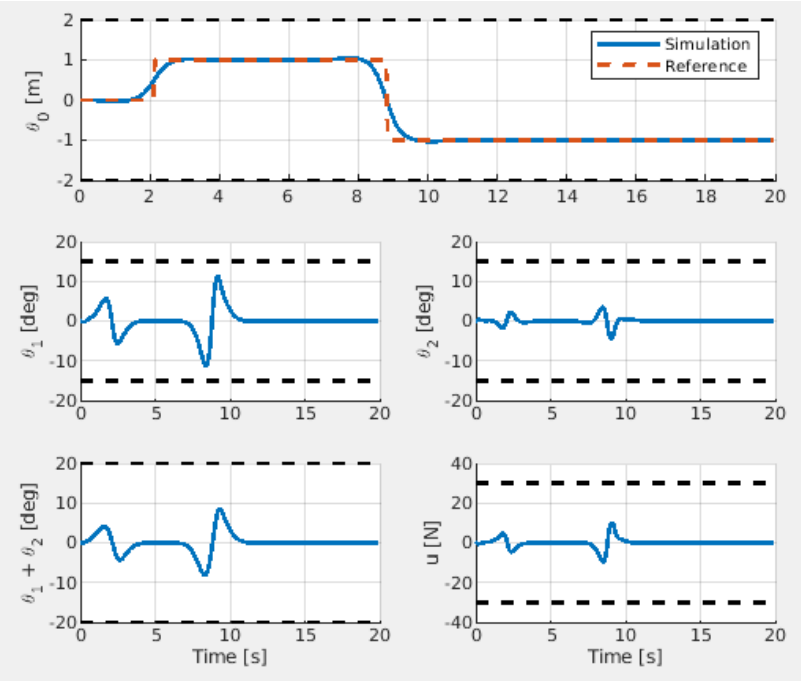

Figure 4.23: MPC perfomance with the reference change known in advance

Here the upper-most plot is again the most informative, because it shows how the controller starts the manouver of changing the cart position even before the reference actually changes, in order to minimise the overall deviation from the reference during the transient. To be more precise, it minimises the deviation on the prediction horizon, therefore it makes sense to choose its length according to the system general inertia, i.e. how long does it take to move the system from one reference position to another.
The two control problems of stabilising the inverted pendulum and reference tracking can be tested together in one procedure, by simply setting the initial θ_1 and θ_2 coordinates of the prodeure above to some value other than zero. Figure 4.24 shows this example, where e.g. initial angles are $\theta_1 = 10°$ and $\theta_2 = -5°$.

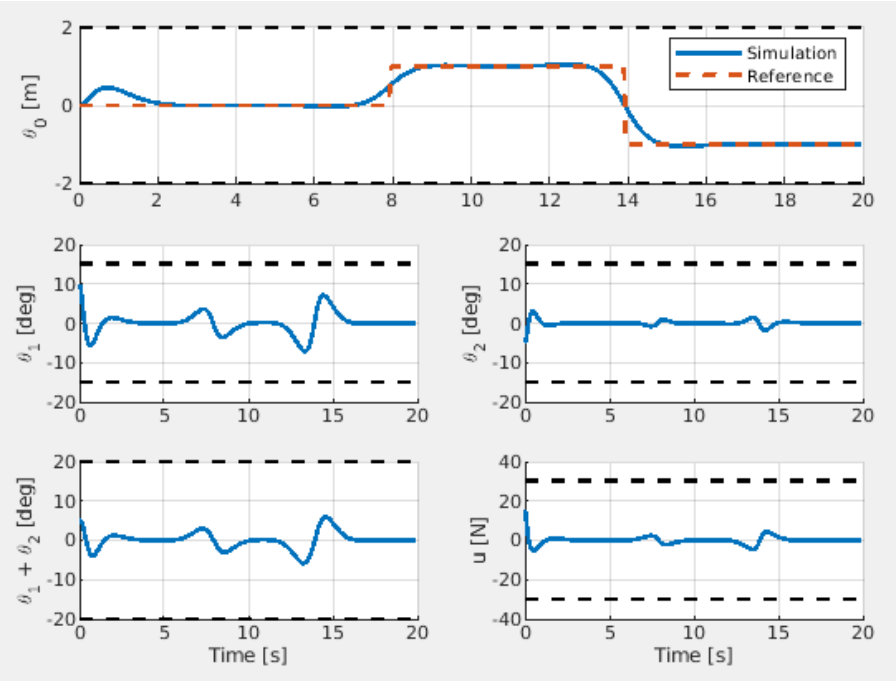

Figure 4.24: MPC stabilisation and reference tracking combined

However, even for this relatively small initial deviation from the equlibrium, a relatively long prediction horizon is still necessary to ensure optimisation problem feasibility, because in this problem formulation no terminal constraints were imposed. The next idea is therefore to combine the theoretical assurance for linear MPC problem feasibility derived for pendulum stabilisation around the upward equlibrium point, with the practical problem of reference tracking. In the presented θ_0 reference tracking problem all of the reference state vector elements are zero except the θ_0 coordinate. Although the theoretical backgorund for using LQR solution as the terminal contraints to guarantee the problem feasibility were derived for stabilisation around origin, the same LQR is here used for reference tracking since all of the reference vectors are analogous, in the sense there are no stationary input signal values in the reference state. So, it could be stated that when θ_0 reference is changed, actually the origin of the six-dimensional coordinate system was translated along the first (θ_0) coordinate, so the control problem is again stabilisation around the origin, with all the theoretical MPC stability proof assumptions being applicable. In this context, it is also important to stress out that there exist a unneglectable mismatch between linearised model used for the MPC syn-

tesis and the non-linear one used for simulation, which means that infeasibility is the possible problem to be dealt with by simulation experments in any case.
But the limiting factor in the use of the presented terminal cost and constraints is the distance between the two reference states, beacuse the controller is iven a hard constraint that it has to bring the system in the region around the new reference state from which the LQR controller can stabilise it without braking any of the state and input signal constraints, as shown previously. Therefore, the difference between required prediction horizon lengths with and the without the use of the terminal constraints is not as drastic as in the case of pure stabilisation problem. If the same procedure from Figure 4.24 with the combination of initial stabilisation and reference tracking is used, using the terminal cost and constraints the required horizon length is 25 steps, and without them 35 steps (with the simulation test increment of 5 steps). The corresponding plots of these two simulations are shown in Figure 4.25 and Figure 4.26.

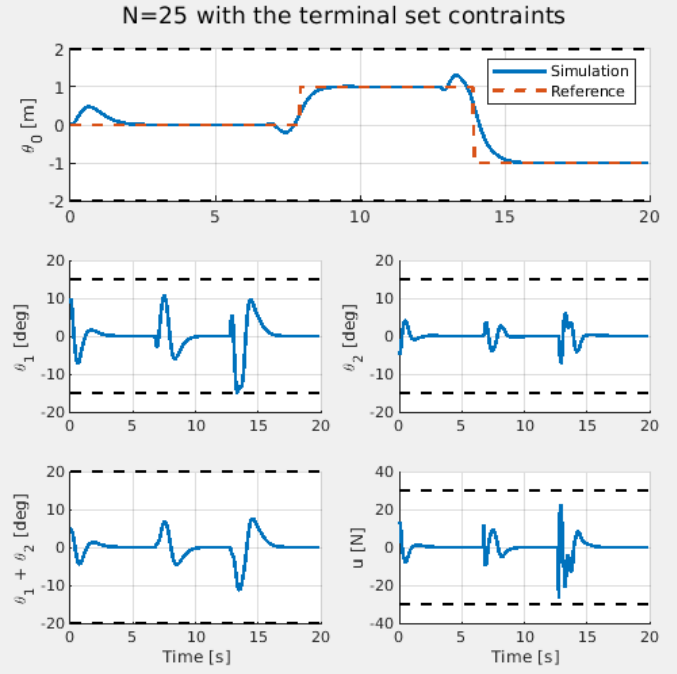

Figure 4.25: MPC reference tracking with the terminal cost and contraints, $N_{min} = 25$

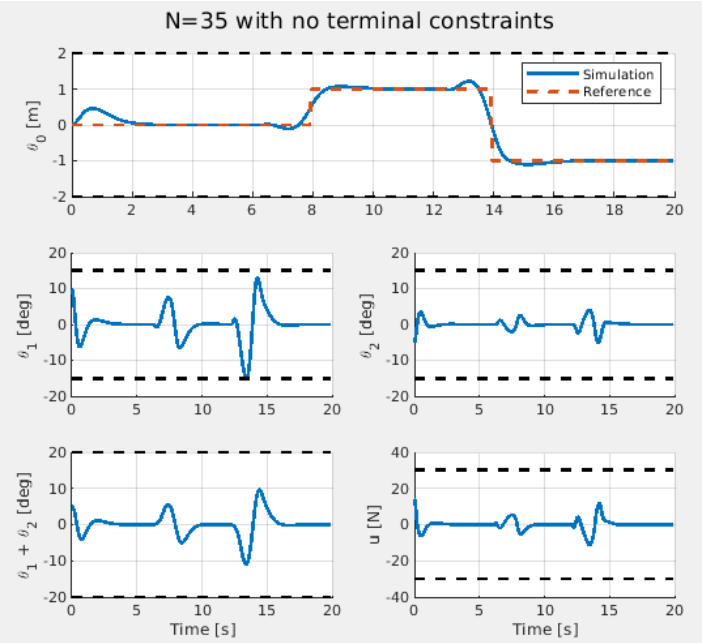

Figure 4.26: MPC reference tracking without the terminal cost and constraints, $N_{min} = 35$

Moreover, by comparing the two plots, it can also be seen that using the terminal cost and constraint and the same values of weight matrices results in a bit more agresive MPC controller regarding the input signal magnitudes and frequency. Another remark to this simulated experiment is that the difference of required prediction horizons between the terminally constrained and non-constrained MPC definitely depends on the combination of sampling time and distance between reference states, so further investigation would maybe show that this difference would maybe be even more significant in some different experiment scenario. However, the effect of changing experiment circumstances could also have the opposite effect. For example, in the modified test of stabilisation and θ_0 reference tracking, the initial angles are set to $\theta_1 = 5°$ and $\theta_2 = -5°$, and θ_0 reference is altered to be closer to the origin (± 0.4m), but with the θ_0 constraints placed relatively close to the reference states (± 0.5m). This way, all the other parameters being the same, the controller with terminal cost and constraints fails if the horizon is set to 30, while the one without them passes the test, as shown in Figure 4.27 and Figure 4.28. The problematic part for the terminally constrained MPC is when it has to bring the system within the LQR invariant set without moving the cart much in the

opposite direction to induce the desired system movement because it's close to the θ_0 constraint. So, this situation would represent the case where adding terminal cost and constraints does not result in the decrease of the required prediction horizon.

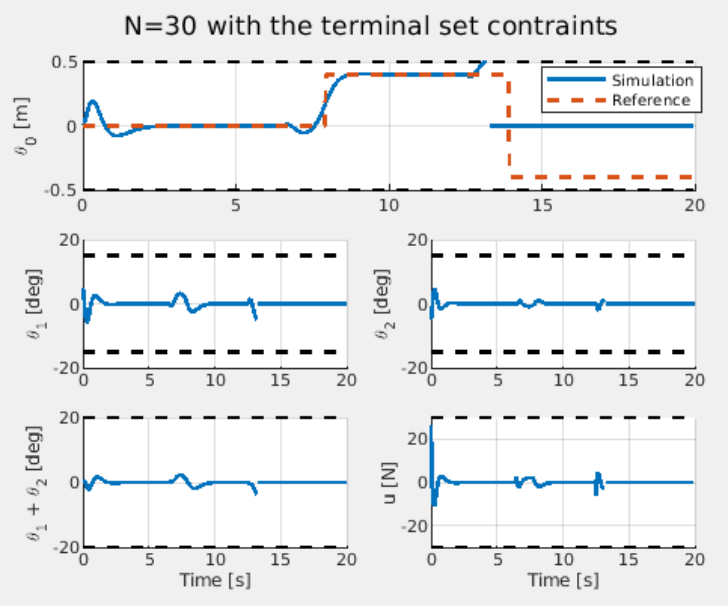

Figure 4.27: Test which MPC with the terminal constraints fails for $N = 30$

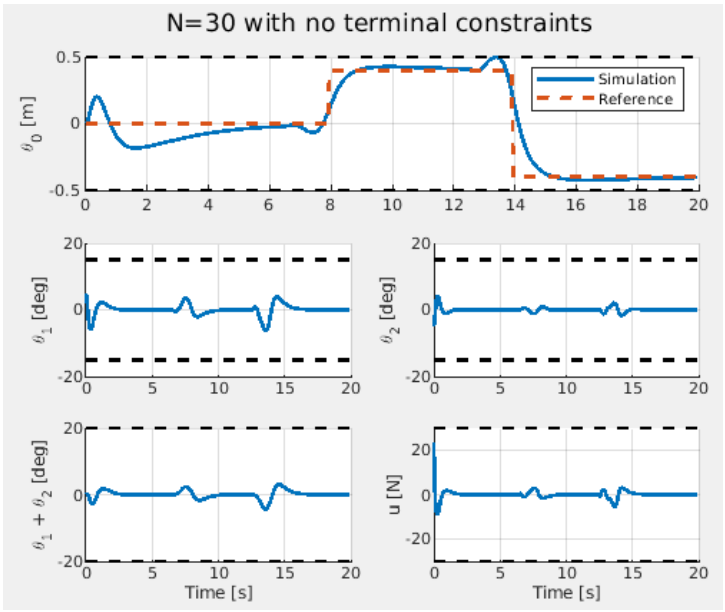

Figure 4.28: Test which MPC with no terminal constraints passes for $N = 30$

5 State estimation

Since system state measurements are almost always corrupted with measurement noise, and sometimes state variables cannot even be measured, state estimation is often a very important branch of a system's control. Regarding previously presented control techniques for DIPC system, elements of the state vector were so far assumed to be known quantitites. In reality, all of three coordinates and their respective angular velocities cannot be measured without some noise. Therefore, the topic of this chapter is to investigate the possibility of combining the measuerements with the knowledge of the physical system model, in order to obtain the state vector estimate accurate enough for control.

5.1. Extended Kalman filter (EKF)

Kalman filter is an recursive algorithm for estimation of the state variables. Developed by the mathematician Rudolph E. Kalman in ithe 1960s, in its base form it is applicable to linear systems. Its generalisation for non-linear systems is called extended Kalman filter (EKF), which is based on system linearisation around estimated state from the previous step [8]. Since the filter inherently involves the model and measurement innacuracies, the starting point form of non-linear system for EKF implementation is the general state space model:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}(t)) \end{aligned} \tag{5.1}$$

Also, due to the step-by step linearisation in the Kalman filtering algorithm, the input and output non-linear functions $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ and $\mathbf{h}(\mathbf{x}(t))$ have to be continously differentiable. To obtain an approximate linear model which will be used to predict the next system state, equations 5.1 have to linearised around the current state. This yields Jacobian matrices, similar to the controller synthesis procedure, but extended with the simple output equation (cf 3.22):

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}_J\mathbf{x} + \mathbf{B}_J\mathbf{u} \\ \mathbf{y} &= \mathbf{C}_J\mathbf{x}, \end{aligned} \tag{5.2}$$

where $\mathbf{A}_J$,$\mathbf{B}_J$,$\mathbf{C}_J$ are defined by the previous step operating point and input signal:

$$\begin{aligned} \mathbf{A}_J &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}_0 \\ \mathbf{u}=u_0}} \\ \mathbf{B}_J &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}_0 \\ \mathbf{u}=u_0}} \\ \mathbf{C}_J &= \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} \end{aligned} \tag{5.3}$$

For the discrete-time applications, the continous linearised state space equations should be discretised. After addition of the process and measurement noise vectors $\mathbf{w}$ and $\mathbf{v}$, final linearised EKF model equations are as follows:

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{F}(k-1)\mathbf{x}(k-1) + \mathbf{G}(k-1)\mathbf{u}(k-1) + \boldsymbol{w}(k-1) \\ \mathbf{y}(k) &= \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k) \end{aligned} \tag{5.4}$$

Here, k marks the sampling step, and $\mathbf{w}$ and $\mathbf{v}$ are process and measurement noise vectors. Esentially, the matrices $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$ are discrete-time Jacobians calculated numerically from their continous-time counterparts $\mathbf{A}_J$, $\mathbf{B}_J$, $\mathbf{C}_J$. In case that the non-linear system is derived in the discrete-time form, the matrices $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$ would be obtained by differentiation around the operating point with the zero noise vectors (using only the deterministic model part):

$$\begin{aligned} \mathbf{F}(k-1) &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}(k-1) \\ \mathbf{u}(k-1) \\ \mathbf{w}=\underline{0}}} \\ \boldsymbol{G}(k-1) &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}(k-1) \\ \mathbf{u}(k-1) \\ \mathbf{w}=\underline{0}}} \\ \mathbf{H}(k) &= \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}(k-1) \\ \mathbf{v}=\underline{0}}} \end{aligned} \tag{5.5}$$

Generally, Kalman filter is comprised of two substeps, where the first one corresponds to the model-based state prediction, and the second one to the predicted state correction based on freshly obtained measurements. So, the result of first substep of the k-th estimation step is *a-priori* state estimation $\hat{\mathbf{x}}(k \mid k-1)$ and estimation error covariance matrix $\mathbf{P}(k \mid k-1)$, based on the model and the information from the previous step. Then in the second part, the previously *a-priori* estimated state is corrected by fresh measurements $\mathbf{y}(k)$ using the Kalman gain matrix $\mathbf{K}$, thus obtaining final *a-posteriori* state estimate $\hat{\mathbf{x}}(k \mid k)$ and measurement covariance matrix $\mathbf{P}(k \mid k)$. The Kalman filter gain is calculated using model and measurement covariance matrices $\mathbf{Q}_{\mathbf{est}}$ and $\mathbf{R}_{\mathbf{est}}$ and the previous measurement error covariance matrix $\mathbf{P}$, in a way which is minimises the covariance of *a-posteriori* state estimation error [9]. It is important to note that this optimality, theoretically proved for linear Kalman filter (LKF) is lost for the non-linear model and the EKF. This is so because in the derivation of the Kalman filter it is assumed that the state and measurement noises are Gaussian, which is the property preserved after linear transformation in the *a-priori* state prediction step [10]. However, the non-linear model corrupts this assumption [11], this way losing the optimality guarantee. Moreover, it is also hard to prove that EKF estimate is assymptotically stable, which means that with no noise the estimation error converges to zero [10]. But, to exploit the accuracy of the genuine non-linear model, the EKF estimator uses the original equations 5.1 in the *a-priori* prediction step, as well as for the calculation of the measurement residual $\tilde{\mathbf{y}}$, which is the difference between predicted and actually measured values of the output vector. So, the resulting extended Kalman equations read [11] [9]:

$$\begin{aligned}
&\hat{\mathbf{x}}(k \mid k-1) = \mathbf{f}(\hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k-1)) \\
&\tilde{\mathbf{y}}(k \mid k-1) = \mathbf{y}(k) - \mathbf{h}(\hat{\mathbf{x}}(k \mid k-1)) \\
&\mathbf{P}(k \mid k-1) = \mathbf{F}(k-1)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^T(k-1) + \mathbf{Q}_{\mathbf{est}}(k-1) \\
&\mathbf{K}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}^T(k) \left[\mathbf{H}^T(k)\mathbf{P}(k \mid k-1)\mathbf{H}^T(k) + \mathbf{R}_{\mathbf{est}}(k)\right]^{-1} \\
&\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{K}(k)\tilde{\mathbf{y}}(k \mid k-1) \\
&\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{K}(k)\mathbf{H}(k)\mathbf{P}(k \mid k-1)
\end{aligned} \tag{5.6}$$

The EKF approaches to the problem of non-linear dynamic system state estimation essentially by approximating the non-linear transformation function by linearising it around the previous-step estimate. As mentioned earlier, the key part of the Kalman filter algorithm is the propagation of uncertain states through the system dynamics,

where these variables are assumed to be of Gaussian distribution [12]. Basically, the filter pretends that the transformation function is linear, and approximate the transformed variable covariance accordingly. In some cases, this can cause considerable errors in the posterior mean (new state estimate) and estimated variables covariance, even sometimes leading to the filter divergence [9] [8] [12] [13]. Second set limitations is connected to the succesive Jacobian matrices calculation, which for some systems can be hard [14] or even impossible [13]. Generally, it could be stated that if the non-linear dynamical system is differentiable and if the linearised version accurately enough describes the original non-linear one, with the relatively small measurement noise the EKF algorithm often works well [10].

5.2. Unscented Kalman filter

The problem of calculating covariance of the Gaussian variables put through the non-linear transformation is adressed in the unscented Kalman filter (UKF) from different perspective. While the EKF uses the linearised function approximation, the UKF use the set of carefully chosen sample points [12] (so called sigma points), and propagates them through the original dynamical system. This transformation is called the unscented transformation (UT) and is based on the intuition that the probability distribution can be approximated easier than the non-linear function whose output uncertainty (to the Gaussian-shaped input) has to be described [13]. So, instead of linearising the transformation function, the UT uses weighted samples around the previously calculated state estimate and passes them through the non-linear transformation, and then calculates the new mean (*a-priori* estimate of the new state) and its covariance. Although at first this may resemble particle filters [13] [14], the crucial difference is that sample points are not drawn randomly, but selected assigned weights in the way which preserves higher-order infromation about the distribution [13]. This way, only a small number of sample points (actually $2n_x+1$, where n_x is the state dimensionality) are neccessary to reconstruct the covariance matrix of the transformed sample points [14]. Also, the final choice of the sigma points distribution and their corresponding weighting is not unique, but can be tuned according to the properties of the observed dynamical system. This is somewhat different than the EKF, where the only tuning parameter is the process noise covariance matrix, in the case where measurement noise covariance is readily available from the

measurements itself, as mentioned in [9].

The difference in general approaches between EKF and UKF is shown schematically in Figure 5.1, where on the left side the true mean and covariance of some 2-dimensional system is shown before and after the transformation. The middle part shows the distribution approximation obtained by transforming middle point and linerasing the transformation function around it, as essentially done in EKF. The right side of the figure represents the UT approach with the weighted sigma points as described above, and it can be seen that for this example UT can approximate the transformed variable distribution better than the algorithm where the transformation function itself is linearised.

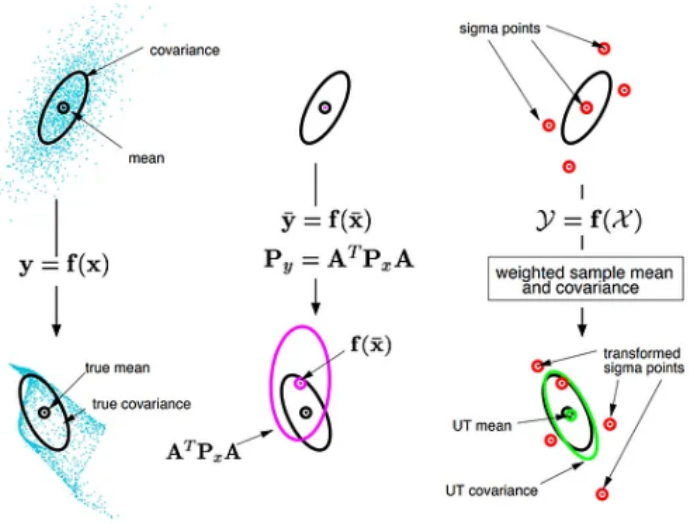

Figure 5.1: UKF vs EKF approach to covariance estimation comparison [12]

5.2.1. The scaled unscented transformation

Although at first the UT and its deployment in UKF theoretically offers an improvement over the EKF without significant increase in the computational effort [12], it also has its drawbacks. The first is related to the property that when estimating systems of higher dimensionality, the spread of sigma points increases as well [14] [15]. Although the transformation properties are still preserved, sigma points may then capture non-local transformation properties, which often has undesirable effect on state estimation [14] [15]. In the DIPC example, if the sampling points in θ_1 and θ_2 corrdinate direction are 60° away from the current estimate, their outputs from the dynamical system will

not provide much of the useful information about the covariance of the new *a-priori* estimated state, simply because the system behaves completely different on these coordinates. Also, some of recommendations for the scaling parameters [15] can lead to the negative parameters for the problems with number of states higher than 3, which can consequentally cause non-positive semidefinite covariance matrix estimate. This property is required for the Cholesky decomposition, which is numerically stable algorithm for taking matrix square root used in the UKF [13] [15], as it will shown later. Therefore, the attempt to overcome this unsuitability of the UT for the higher-dimensional state estimation problems represents the scaled unscented transformation.
In short, this approaches tackles the spread of the sigma points by deriving their alternative set moved towards the mean, and then derives an alternative weighting to each point, in order to preserve second order accuracy both for the mean and covariance [14]. To fully describe the UT, its scaled version and its implementation within the UKF, the entire algorithm will be explained step by step.
So, in the general non-linear state space example, both the state and output equations are non-linear functions. In the context of UT, they represent the non-linear transformation function, so the beginning is actually the same as for the EKF:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}(t)) \end{aligned} \tag{5.7}$$

Next up, the sigma points $\boldsymbol{\sigma_i}$ and their weights for the UT need to be defined, using the scaling factors α, β and κ. The first sigma point is the mean μ, i.e. previously estimated state, and the remaining $2n_x$ are distributed symetrically around it:

$$\begin{aligned} &\boldsymbol{\sigma_0} = \mu, \qquad \boldsymbol{\sigma}_{\pm i} = \boldsymbol{\mu} \pm \left[\sqrt{(n_x + \lambda)}\mathbf{S_x}\right]_i, \\ &\omega_0 = \tfrac{\lambda}{n_x+\lambda}, \quad \omega_{\pm i} = \tfrac{1}{2(n_x+\lambda)}, \end{aligned} \tag{5.8}$$

Here i denotes the i^{th} column vector in the expressions for the symetrically distributed sigma points. λ stands for composed scaling parameter, defined by

$$\lambda = \alpha^2 \left(n_x + \kappa\right) - n_x$$

and $\boldsymbol{S}_x$ represents decomposition of the covariance matrix, such that

$$\boldsymbol{S}_x \boldsymbol{S}_x^\top = \boldsymbol{P_x}.$$

ω_i denotes weighting factors for the each sigma point, derived so that they represent the prior distribution (mean and covariance), according to the expressions:

$$\begin{aligned} \boldsymbol{\mu} &= \sum_{i=0}^{2n_x} \omega_i \boldsymbol{\sigma_i} \\ \mathbf{P} &= \sum_{i=0}^{2n_x} \omega_i \left(\boldsymbol{\sigma_i} - \boldsymbol{\mu}\right) \left(\boldsymbol{\sigma_i} - \boldsymbol{\mu}\right)^T. \end{aligned} \tag{5.9}$$

It has to be noted that the weights ω_i are not explicity bounded to be between zero and one, but negative weights are also permitted, with the neccesary condition that their sum is equal to one.

$$\sum_{i=0}^{2n_x} \omega_i = 1$$

However, some sources state that negative values could lead to the numerical issues in the algorithm execution [15], and therefore one of the conditions used for setting the tuning parameters in this thesis is to avoid this situation. So far, "regular" UT sigma points and their corresponding weights were presented, and in the scaled UT, the sigma points are modified as follows:

$$\boldsymbol{\sigma_i}' = \boldsymbol{\sigma_0} + \alpha \left(\boldsymbol{\sigma_i} - \boldsymbol{\sigma_0}\right) \tag{5.10}$$

Since α is limited to be in the set $(0, 1]$, the scaled sigma points are either equal to the non-scaled (for $\alpha = 1$), or approaching the zeroth sigma point, i.e. previous state estimate. To preserve the properties of the UT, the weighting coefficents also have to be corrected, which is done as follows:

$$\omega_i' = \begin{cases} \omega_0/\alpha^2 + (1 - 1/\alpha^2) & i = 0 \\ \omega_i/\alpha^2 & i \neq 0 \end{cases} \tag{5.11}$$

Important is that in [14] it is shown that the covariance matrix of the transformed sigma points is guaranteed to be positive semidefinite if all of the transformed weighting coefficients are non-negative, so the line of reasoning to avoid the negative weights is also confirmed. Nextly, if the transformed sigma points are denoted z_i, in case of non-linear transformation from the system state equation, they are defined by:

$$z_i = f(\sigma_i) \tag{5.12}$$

After the transformation of the of the sigma points through the system state equation, the *a-priori* estimate of the state variable vector (mean), and its covariance matrix are obtained via:

$$\begin{aligned} \boldsymbol{\mu_z} &\approx \sum_{i=0}^{2n_x} \omega_i \boldsymbol{z_i} \\ \boldsymbol{P}_z &\approx \sum_{i=0}^{2n_x} \omega_i \left(\boldsymbol{z_i} - \boldsymbol{\mu_z}\right)\left(\boldsymbol{z_i} - \boldsymbol{\mu_z}\right)^T \\ &\quad + \left(1 - \alpha^2 + \beta\right)\left(\mathbf{z}_0 - \boldsymbol{\mu_z}\right)\left(\mathbf{z}_0 - \boldsymbol{\mu_z}\right)^T, \end{aligned} \tag{5.13}$$

where β is additional scaling paramter, used to add extra weight to the zeroth transformed sigma point [14] [15].

5.2.2. Scaled UT applied in UKF

As already mentioned, the above explained procedure of the scaled unscented transformation can be deployed in the unscented Kalman filter used for state estimation for non-linear state space, as the one described by equations 5.1. Actually, the scaled UT is applied twice, so the UKF algorithm can be summarised as follows:

- Firsly, the UT is applied on the state equation with the previous state estimate being zeroth sigma point, thus obtaining the *a-priori* new state estimate. To calculate the *a-priori* estimate covariance, process noise covariance matrix is simply added to the covariance matrix obtained by UT.

- Then, the same procedure is applied to the non-linear output equation, hence getting the model-based best guess of the measured values. Instead of adding the process noise, the measurement noise covariance matrix is added to the calculate the corresponding covariance matrix.

- The third step is to calculate the Kalman gain matrix from the covariance matrices, using the expressions below [16]:

$$\begin{aligned} \mathbf{P}_{xz} &= \sum_{i=0}^{2n_x} \omega_i \left(\boldsymbol{\sigma_i} - \boldsymbol{\mu}_x\right)\left(\boldsymbol{z_i} - \boldsymbol{\mu_z}\right)^T \\ \mathbf{K}_{ukf} &= \mathbf{P}_{xz}\mathbf{P}_z^{-1} \end{aligned}$$

- After obtaining fresh measurements $y(k)$, final two steps are to correct *a-priori* estimete via Kalman gain matrix, thus obtaining final *a-posteriori* new state estimate $\hat{\mathbf{x}}$, and also to update the state estimate covariance matrix $\mathbf{P}$ using Kalman gain matrix [16]:

$$\hat{\mathbf{x}} = \boldsymbol{\mu}_x + \mathbf{K}_{ukf}(\mathbf{y}(k) - \boldsymbol{\mu}_{\boldsymbol{z}})$$
$$\mathbf{P} = \mathbf{P} - \mathbf{K}_{ukf}\mathbf{P}\mathbf{K}_{ukf}^T$$

Although the expressions for the Kalman gain and the estimate covariance matrix update look differently then in the case of EKF (cf. 5.6), in [16] it is shown that mathematically they represent the same thing.

5.3. EKF and UKF performance on DIPC system

This section present the closed-loop performance of the MPC-controlled DIPC system, if the state vector is estimated only from noisy position and angle measurements, so only three out of six elements of the the state vector are measured.
The chosen simulation and controller environment is the MPC without the terminal cost and constraints, with the θ_0 reference varying on the prediction horizon, but close to the θ_0 constraint, like the one from Figure 4.28. The measurements are corrupted with the white noise characterised by its standard deviation, where the corresponding noise numerical data was taken from the datasheets of commercially available linear potentiometier [17] for cart position measurement and Hall-effect angle sensor [18]. The procedure of comparing the performance of the two filters will be as follows:

- The measurement noise covariance matrix $\mathbf{R}_{\mathbf{est}}$ is calculated as the diagonal matrix, with the members on the main daigonal being sqaured standard deviation values from the sensor datasheets
- The process noise covariance matrix $\mathbf{Q}_{\mathbf{est}}$ is tuned the goal to achieve the desired EKF state estimation performance, which is used as a benchmark for the UKF
- The UKF scaling factors α and κ are tuned to compare the UKF performance with the EKF

Initially, both estimation algorithms can be compared in the open loop, i.e. with the use of simulated state variables as the system inputs, so as if all the state vector variables

were perfectly measured. Such a state estimation of both the estimators is shown in Figure 5.2, with the zoomed plots in Figure 5.3. From them it can be seen that both estimators similarly assess the θ_0 coordinate. For $\dot{\theta}_0$ and also $\dot{\theta}_1$ UKF shows less noise, but also small stationary offset. Regarding θ_1 and θ_2, it can be deduced that EKF relies heavily on the measurements, while the UKF has more significant filtering action. Since the system is very unstable, increasing the model reliability in the filter, i.e. setting the lower values into the $\mathbf{Q}_{\mathbf{est}}$ matrix, causes it the system in the closed loop to diverge, threfore these open loop estimation results are considered as a starting point for the next subsections.

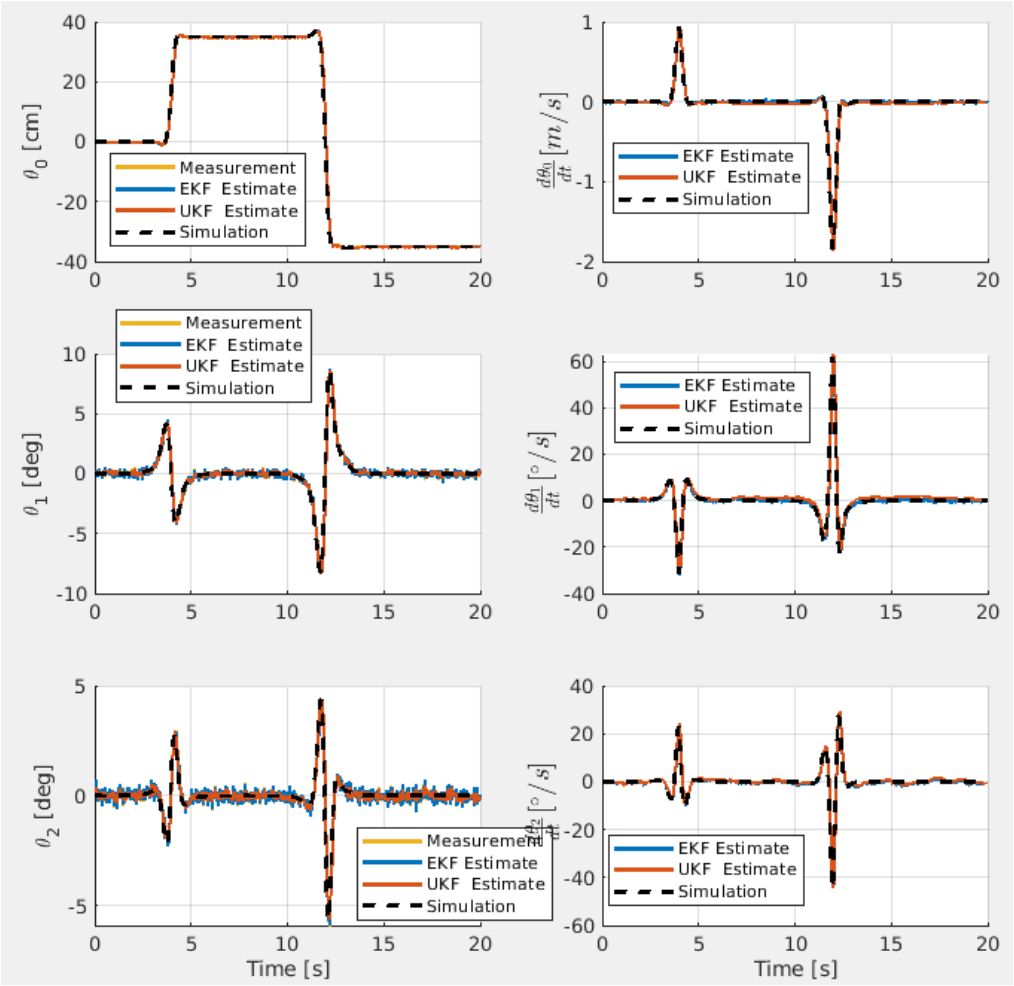

Figure 5.2: EKF and UKF state estimation in open loop

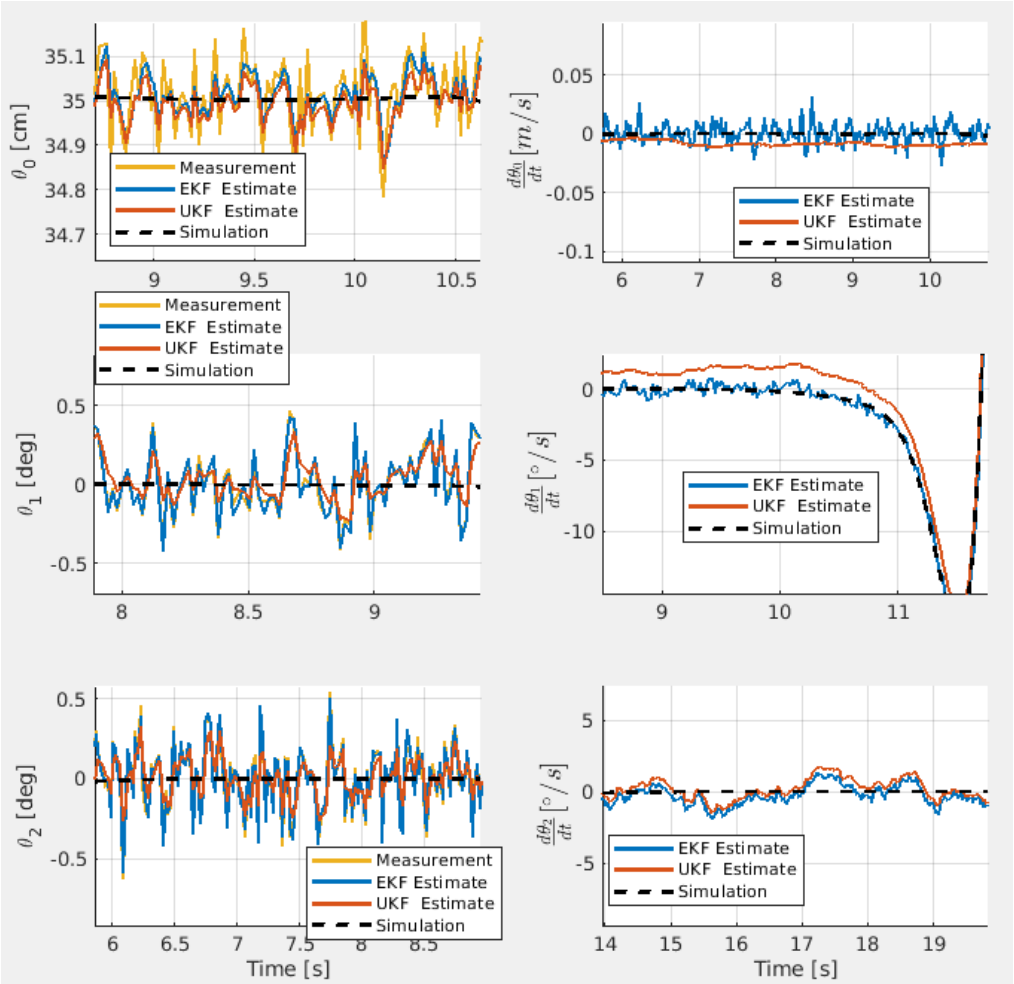

Figure 5.3: EKF and UKF state estimation in open loop, zoomed view

5.3.1. EKF estimate as the feedback

If the EKF is used as the feedback, the noisy state estimate influences the reduced precision in the pendulum positioning, i.e. θ_0 coordinate, as seen in Figure 5.4. The estimation algorithms comparison shown in Figure 5.5 shows that when EKF is used in the closed loop, the two estimators behave almost the same.

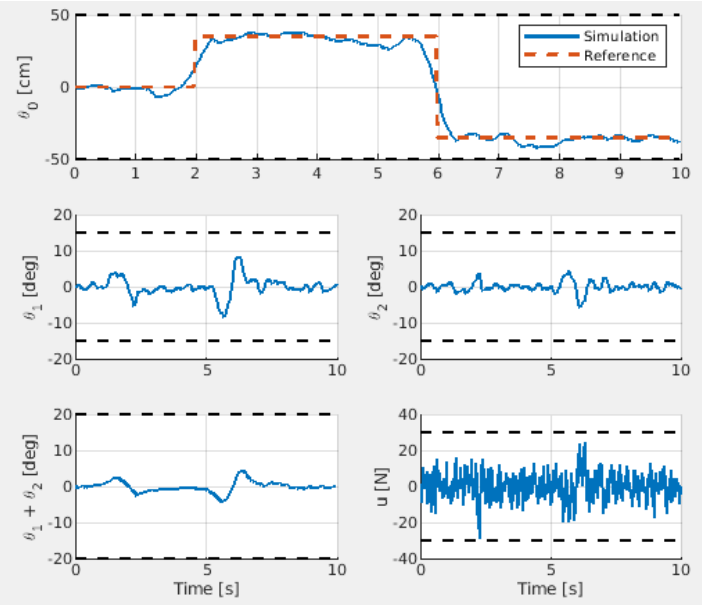

Figure 5.4: Test procedure with the loop closed with EKF estimate

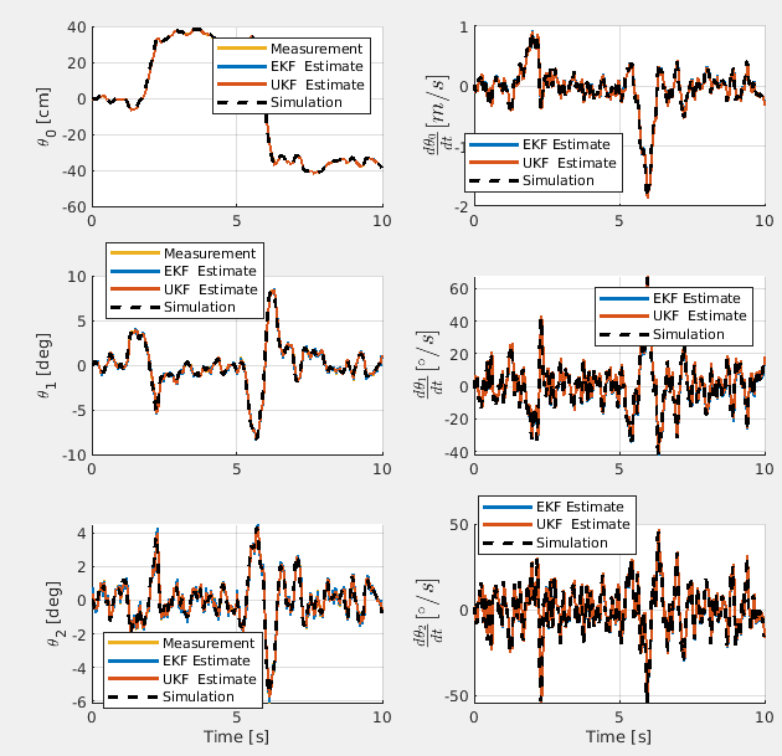

Figure 5.5: Estimated state variables, EKF in feedback loop

Inspecting the enlarged view of the estimated variables comparison shown in Figure 5.6 confirms the previous conclusion that in case where the actual system is oscilating due to the noisy state estimate, both the estimators yield practically the same results. The middle left plot in Figure 5.6 shows that UKF still exhibits a bit better filtering action, while EKF again relies almost completely on measurements.

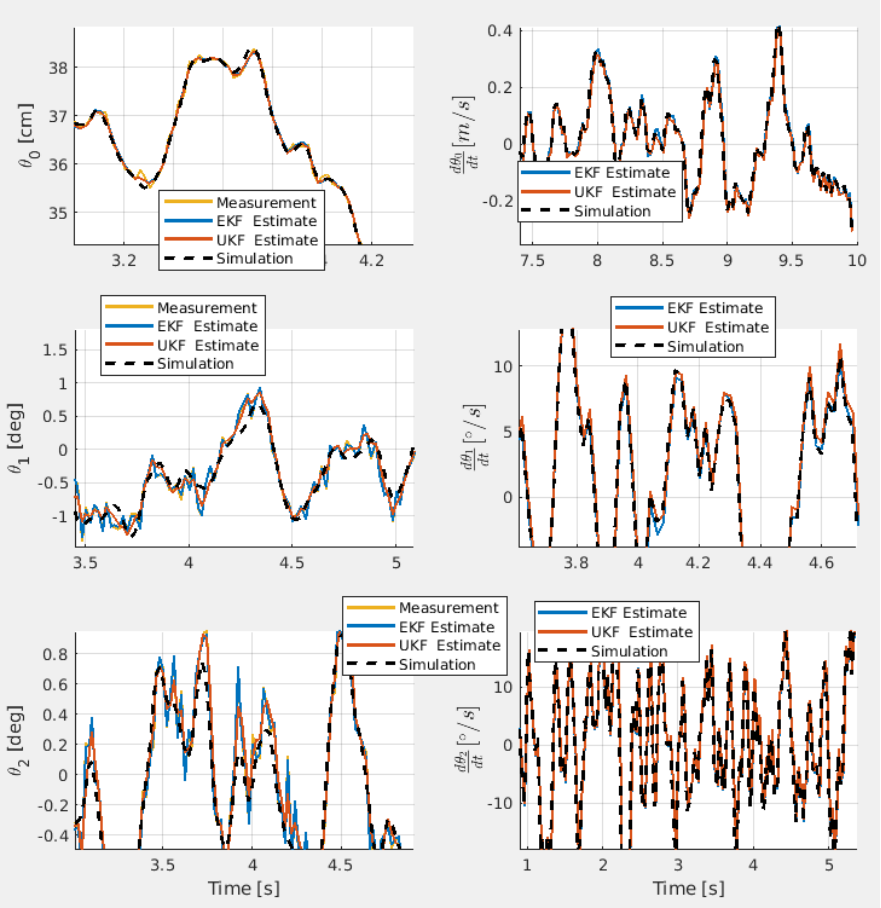

Figure 5.6: Estimated state variables, EKF in feedback loop - zoom

In spite of oscilating motion of the pendulum cart (θ_0 coordinate), the presented EKF manages to give the estimate of the non-measured state vector elements ($\dot{\theta}_{0,1,2}$) well enough that the pendulum remains stabilised in the upward position.

5.3.2. UKF estimated state in the feedback

Second option discussed in this thesis is the use of the UKF estimator output in the feedback loop. The state variables trajectories in this case are shown in Figure 5.7, where it can be seen that compared to the EKF, slighly less noisier estimator gives a bit less noise in the control signal and consequentually in some state variable trajectories, as for example θ_1. However, a more significant difference between the use of the two algorithms is the small stationary offset present in the controlled variable θ_0.

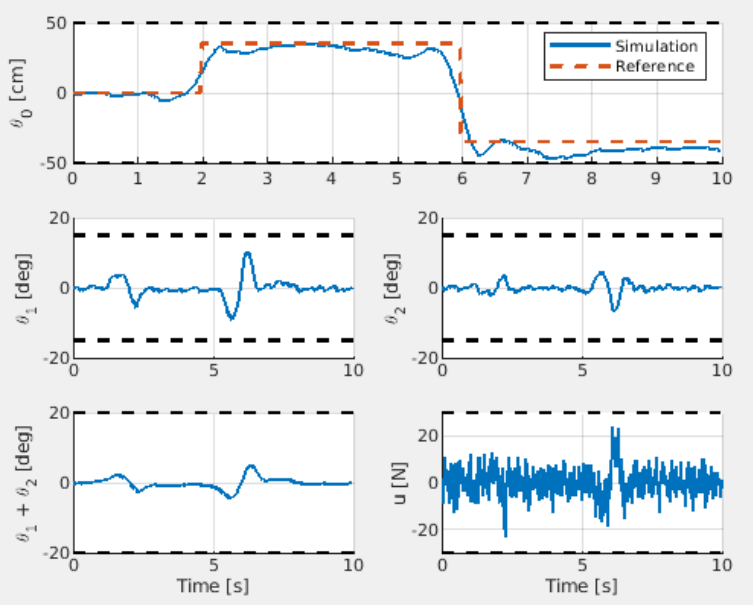

Figure 5.7: Test procedure with the loop closed with UKF estimate

The state variable estimation plots at first look similar to the case where EKF estimator is deployed, so again both EKF and UKF yield practically the same estimates (see Figure 5.5). However, the enlarged view od the first link angle derivative $\dot{\theta}_1$ estimate reveals that UKF tends to drift off from the simulation trajectory, while the EKF follows it better, as shown in Figure 5.8. This could explain why the same controller with the UKF estimate in the feedback loop does not follow the desired θ_0 reference. If the θ_1 is offset, the predictive controller expects both of the links will start falling to the side, and so it has to wait until the pendulum links are tilted, so it can catch them in the same time bringing the cart to the desired position. Since it then does not initiate any cart movement, the pendulum link never assume any significant tilting angle, and therefore

the cart remains stuck in the offset position.

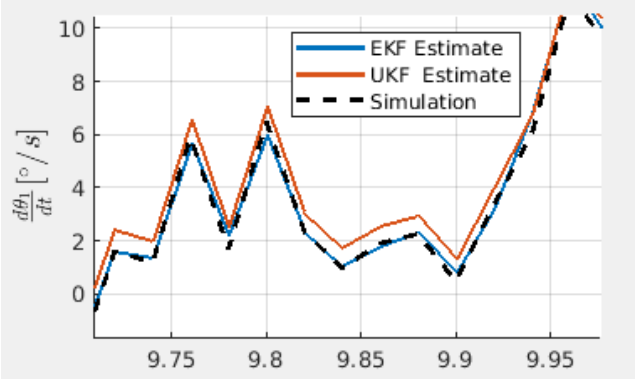

Figure 5.8: $\dot{\theta}_1$ estimate with UKF in feedback loop

This effect is even exagurated when inifinite horizon LQR controller is deployed with the UKF estimate in the feedback loop, as shown in Figure 5.9.

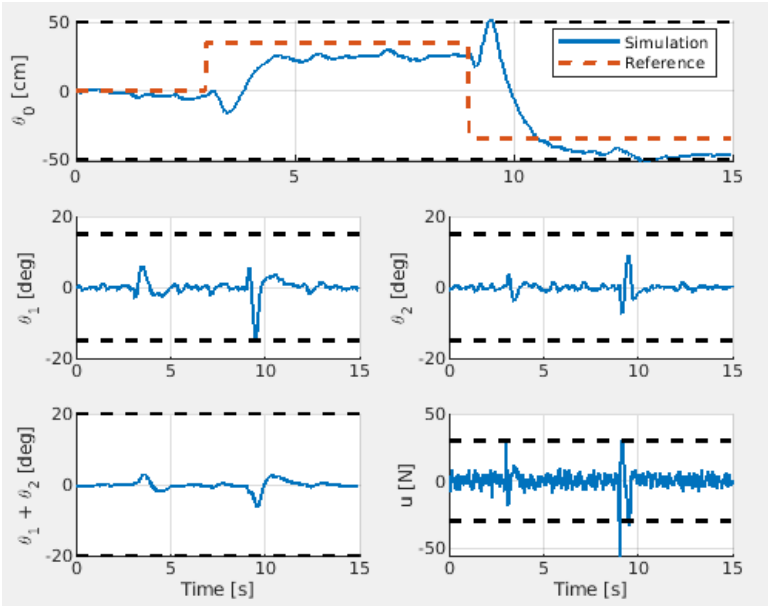

Figure 5.9: LQR controlled system with the UKF estimate

As expected, looking at the enlarged view of the $\dot{\theta}_1$ estimate plot shown in 5.10, it can be seen that again not negligible amount of this state variable estimate offset is present.

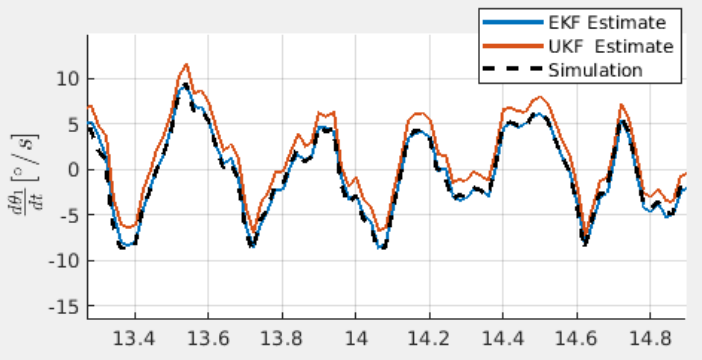

Figure 5.10: $\dot{\theta}_1$ estimate with UKF in feedback loop and LQR controller

Further investigation of this effect could include deployment of the offset-free model predictive controller (as in [19] [20]), if the UKF was to be employed. The interesting part of the problem is the fact that adjusting the UKF tuning parameters does not affect the estimate offset.

If the EKF was to be used, and the presented system accuracy was not satisfactory, the solution would be to increase the sensor quality in terms of the output noise. For example, if the noise amplitudes were halved and the EKF estimator would be deployed, the resulting DIPC positioning accuracy would be better, as shown in Figure 5.11.

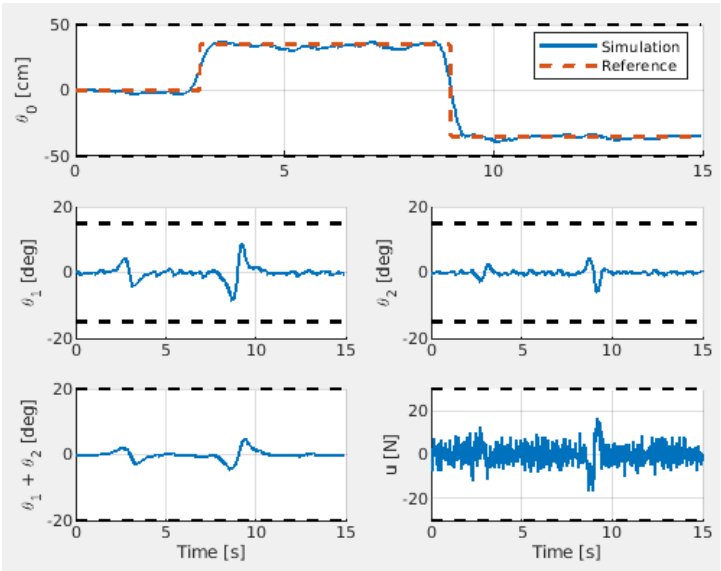

Figure 5.11: Test procedure with EKF estimator and improved sensors

6 Conclusion

In this thesis the double inverted pendulum on a cart dynamical model was derived using the first principles equations. Model linearisation was carried out, with the comparison of the original model and its linearised counterpart, where it was shown that the presented system is highly non-linear. Inifite horizon linear quadratic controller (LQR) was used to stabilise the system, and the invariant set for LQR-controlled system was analysed. Then, linear model predictive controller (MPC) was deployed, and subsequently it was shown how the usage of the terminal cost and constraints can decrease the required prediction horizon, which is important factor for the system hardware requirements. Subsequently, MPC was used for reference tracking problems, with the constraints designed to constraint the system's trajectories to the applicability region of the linear model used in the controller synthesis. Simulation results of the closed-loop behaviour was presented for two cases, when the reference signal change is known in advance, i.e. over the whole prediction horizon, and also when this knowledge is not available. Finally, it was assumed that only position sensors were available, so the analysis of the state estimation algorithms was performed and presented. For this puprose, the extended and unscented Kalman filters (EKF and UKF) were derived and analysed. It was shown that by using the UKF the measurement noise is filtered slightly better, but also that the closed-loop system exhibits stationary offset when UKF estimate is used as a feedback. Regarding the EKF, it was shown that it can be used for the double pendulum positioning problem, but that for increased accuracy higher sensor requirements are needed.

7 Appendices

All the codes are written in *Matlab*, version *R2022b*. As the sidenote, some linebreaks were added to the text version of the original code, in order to fit the page. Since there is no NDA, original code is available upon request.

7.1. Appendix A

Appendix A shows the code which calculates the terms on the left-hand side of the Langrage equations 3.1, using the θ_2 definition from figure 3.1

```
syms theta0(t) theta1(t) theta2(t) m0 m1 m2 g L1 L2
J1=1/12*m1*L1^2;
J2=1/12*m2*L2^2;

% kinetic energy of the cart
Ek0=1/2*m0*(diff(theta0,t))^2;

% potential and kinetic energy of the first link
Ep1=m1*g*L1*0.5*cos(theta1);
x1=theta0+1/2*L1*sin(theta1);
y1=1/2*L1*cos(theta1);
Ek1t=1/2*m1*((diff(x1))^2+(diff(y1))^2);
Ek1r=1/2*J1*(diff(theta1))^2;
Ek1=Ek1r+Ek1t
```

```
Ek1s=simplify(Ek1)

% potential and kinetic energy of the second link
Ep2=m2*g*(L1*cos(theta1)+1/2*L2*cos(theta2));
x2=theta0+L1*sin(theta1)+1/2*L2*sin(theta2);
y2=L1*cos(theta1)+1/2*L2*cos(theta2);
Ek2t=1/2*m2*((diff(x2))^2+(diff(y2))^2);
Ek2r=1/2*J2*(diff(theta2))^2;
Ek2=Ek2r+Ek2t;
Ek2s=simplify(Ek2)

Ek=Ek0+Ek1+Ek2
Eks=simplify(Ek, Steps=30)
Ep=simplify(Ep1+Ep2);

Lag=Ek-Ep;

Lag0=diff(diff(Lag,diff(theta0)),t)-diff(Lag,theta0);
Lag0s=simplify(Lag0)

Lag1=diff(diff(Lag,diff(theta1)),t)-diff(Lag,theta1);
Lag1s=simplify(Lag1)

Lag2=diff(diff(Lag,diff(theta2)),t)-diff(Lag,theta2);
Lag2s=simplify(Lag2)
```

7.2. Appendix B

Appendix B shows the code which calculates the terms on the left-hand side of the Langrage equations 3.1, using the θ_2 definition from figure 3.2.

```
syms theta0(t) theta1(t) theta2(t) m0 m1 m2 g L1 L2
J1=1/12*m1*L1^2;
```

```
J2=1/12*m2*L2^2;

% kinetic energy of the cart
Ek0=1/2*m0*(diff(theta0,t))^2;

% potential and kinetic energy of the first link
Ep1=m1*g*L1*0.5*cos(theta1);
x1=theta0+1/2*L1*sin(theta1);
y1=1/2*L1*cos(theta1);
Ek1t=1/2*m1*((diff(x1))^2+(diff(y1))^2);
Ek1r=1/2*J1*(diff(theta1))^2;
Ek1=Ek1r+Ek1t
Ek1s=simplify(Ek1)

% potential and kinetic energy of the second link
Ep2=m2*g*(L1*cos(theta1)+1/2*L2*cos(theta1+theta2));
x2=theta0+L1*sin(theta1)+1/2*L2*sin(theta1+theta2);
y2=L1*cos(theta1)+1/2*L2*cos(theta1+theta2);
Ek2t=1/2*m2*((diff(x2))^2+(diff(y2))^2);
Ek2r=1/2*J2*(diff(theta2)+diff(theta1))^2;
Ek2=Ek2r+Ek2t;
Ek2s=simplify(Ek2)

Ek=Ek0+Ek1+Ek2
Eks=simplify(Ek, Steps=30)
Ep=simplify(Ep1+Ep2);

Lag=Ek-Ep;

Lag0=diff(diff(Lag,diff(theta0)),t)-diff(Lag,theta0);
Lag0s=simplify(Lag0)

Lag1=diff(diff(Lag,diff(theta1)),t)-diff(Lag,theta1);
```

```
Lag1s=simplify(Lag1, Steps=30)

Lag2=diff(diff(Lag,diff(theta2)),t)-diff(Lag,theta2);
Lag2s=simplify(Lag2, Steps=30)
```

7.3. Appendix C

Appendix C shows the code which derives the linearised model of the DIPC system 3.22 using *Matlab Symbolic Toolbox*:

```
syms theta0(t) theta1(t) theta2(t) m0 m1 m2 g L1 L2 d0 d1 d2 u
dtheta0=diff(theta0)
dtheta1=diff(theta1)
dtheta2=diff(theta2)

% non-linear model matrices
% do not fit the pdf page in the code form
% see the paper for expressions
H=[1;0;0]
G=[0;-(0.5*m1+m2)*L1*g*sin(theta1)-0.5*m2*L2*g*sin(theta1+theta2);
...-0.5*m2*g*L2*sin(theta1+theta2)]
D=[m0+m1+m2, (0.5*m1+m2)*L1*cos(theta1)+1/2*m2*L2*cos(theta1+theta2),
... 0.5*m2*L2*cos(theta1+theta2);
... (0.5*m1+m2)*L1*cos(theta1)+1/2*m2*L2*cos(theta1+theta2),
... (1/3*m1+m2)*L1^2 + 1/3*m2*L2^2+m2*L1*L2*cos(theta2),
... 1/3*m2*L2^2+0.5*m2*L1*L2*cos(theta2);
...0.5*m2*L2*cos(theta1+theta2),
... 0.5*m2*L1*L2*cos(theta2)+1/3*m2*L2^2, 1/3*m2*L2^2]
D_inv=D\eye(3)
C=[d0, -(0.5*m1+m2)*L1*sin(theta1)*dtheta1-
...m2*L2*sin(theta1+theta2)*(dtheta2+0.5*dtheta1),
... -0.5*m2*L2*sin(theta1+theta2)*dtheta2;
...0, d1, -0.5*m2*L1*L2*sin(theta2)*dtheta2-L1*L2*m2*sin(theta2)*dtheta1;
```

```
... 0, 0.5*m2*L1*L2*sin(theta2)*dtheta1, d2]

A_=[zeros(3,3),eye(3); zeros(3,3),-D_inv*C]
B_=[0;0;0;D_inv*H]
L=[0;0;0;-D_inv*G]

% intermediate step for jacobian derivation
Aj=formula(A_)

Bj=formula(B_)
Lj=formula(L)
x_=[theta0,theta1,theta2,dtheta0,dtheta1,dtheta2]

% defining the non-linear model in the form x_dot=f(x,u)
f1=Aj(1,:)*x_'+Lj(1)+Bj(1)*u
f2=Aj(2,:)*x_'+Lj(2)+Bj(2)*u
f3=Aj(3,:)*x_'+Lj(3)+Bj(3)*u
f4=Aj(4,:)*x_'+Lj(4)+Bj(4)*u
f5=Aj(5,:)*x_'+Lj(5)+Bj(5)*u
f6=Aj(6,:)*x_'+Lj(6)+Bj(6)*u

% vector of equations
f=[f1,f2,f3,f4,f5,f6];

% general jacobian calculations
A1=jacobian(f,x_)
B1=jacobian(f,u)

% calculation of the jacobian matrices
% of the linearisation around pendulum-down position
theta0=0
theta1=pi
theta2=0 % theta_2 is defined relatively
```

```
dtheta0=0
dtheta1=0
dtheta2=0

A1s=subs(A1)
B1s=subs(B1)
```

7.4. Appendix D

Appendix D shows the code which derives the maximal invariant set for linearised DIPC system with the chosen state and input constraints, and then performs a simulation of the closed-loop system, with the initial state chosen by the click on the invariant set slice displayed in plotted graph.

7.4.1. Simulation from initial states chosen by mouse click

```
%function which plots the polyhedron which
% represnts the set of constraints
%on theta1 and theta2

%Parameters
m0=1.5;
m1=0.5;
m2=0.5;
L1=1;
L2=1;
g=9.81;
d0=0.1;
d=0.1;

d1=d;
d2=d;
```

```
% Linearised system dynamics - from live script (for theta1,2=0):
matrices Ac and Bc too long to fit the pdf, therfore omitted
see: Lagrange_dpic_livescript_jacob Appendix B

sys = ss(Ac,Bc,eye(6),zeros(6,1)); %D matrix is zero

% Conversion from continous to discrete system
Ts=0.05; % sample time
sysd = c2d(sys, Ts);
[A,B,C,D] = ssdata(sysd);

% Criterion weight matrices
Q = [eye(3),zeros(3);zeros(3),eye(3)*0.001];
R =0.1;

%LQR controller gain
[K,~,~] = dlqr(A,B,Q,R);
Acl = A-B*K; % closed loop system matrix

%% State constraints - theta_0,1,2 theta_dot_0,1,2 (meters and radians)
f0=[2, 15*pi/180,15*pi/180,10,720*pi/180,720*pi/180]';
% additional constraints on sum of theta_1 and theta_2
%abs(theta1+theta2)<20 deg
F0=[0 1 1 0 0 0];
F = [eye(6);-eye(6);F0;-F0]; f=[f0;f0;20*pi/180;20*pi/180];

P1 = Polyhedron (F,f);
Pcut1 = slice(P1, [1 4 5 6]);
plot(Pcut1, 'Color', 'r');
hold on

Umax=30; %[N]
%Input signal constraints
```

```
% U = {u | Mu <= m}
M = [1;-1]; m = [Umax;Umax];
% u=-Kx, therefore another polyhedron of states represents the input signal
% constraint

% intersection of input constraints with the state
% constraints X = {x | Fx <= f}
FF = [F;-M*K]; ff = [f;m];

P2 = Polyhedron (FF,ff);
Pcut2 = slice(P2, [1 4 5 6]);
plot(Pcut2, 'Color', 'c');
hold on




% %% projection of entire polytope onto a theta1,2 plane
% figure(2)
% plot(Pcut1, 'Color', 'r');
% hold on
% Pproj=projection(P2,[2 3]);
% plot (Pproj,'Color', 'c')
%
% legend('State constraints','Input signal defined constraints')
% xlabel('\theta_1 [rad]')
% ylabel('\theta_2 [rad]')
% title ('Projection, u_{max}=10 N')

%% Maximal invariant set

% Xf = maxInvar(Acl, Polyhedron(FF,ff));
% Finvar = Xf.A;
```

```
% finvar = Xf.b;

Pcut_invar = slice(Xf, [1 4 5 6]);
plot(Pcut_invar, 'Color', 'g')


%% closed-loop simulation
h=Ts; %time step for manual runge kuuta implementation
traj = cell(0); % cell which will contain all the trajectories

while 1
     fprintf('\nClick on figure for trajectory (right-click for exit)\n')
    [x,y,button] = ginput(1);
if button > 1, break; end
    sol.x = [];
    sol.x(:,1) = [0;x;y;0;0;0];

    if all(Finvar*sol.x(:,1) <= finvar)
    % x is inside invariant set, for plot color
        sol.feasible = 1;
    else
        sol.feasible = 0;
    end

    i = 1;
    while norm(sol.x(:,end)) > 1e-2
    % Simulation lasts until the system converges to origin
      % discrete LQR controlled optimal actions
      sol.u(:,i) = -K*sol.x(:,i);

      % Application of the control action to the system
      % --> manual 4th order runge kutta implementation -
      %for integration of non-linear system
```

```
      % rename dpic into odefun_dpic_newtheta_RK
      k_1 = dpic(sol.x(:,i), m0, m1, m2, L1, L2, g, d0, d,sol.u(:,i));
      k_2 = dpic(sol.x(:,i)+0.5.*h.*k_1, m0, m1, m2, L1, L2, g, d0, d,sol.u(:,i));
      k_3 = dpic(sol.x(:,i)+0.5.*h.*k_2, m0, m1, m2, L1, L2, g, d0, d,sol.u(:,i));
      k_4 = dpic(sol.x(:,i)+k_3.*h, m0, m1, m2, L1, L2, g, d0, d,sol.u(:,i));
      sol.x(:,i+1) = sol.x(:,i) + (1/6).*(k_1+2.*k_2+2.*k_3+k_4).*h;
       % main equation

      i = i + 1;
    end

    % each trajectory is a separte cell object
    traj{end+1} = sol;

    % Plotting the trajectores
    plot(sol.x(2,:),sol.x(3,:),'--k');
    plot(sol.x(2,:),sol.x(3,:),'.k','markersize',5);
end
legend('State constraints','Input signal defined constraints',..
...'Invariant set for LQR controlled system')
xlabel('\theta_1 [rad]')
ylabel('\theta_2 [rad]')
title ('Slice, u_{max}=30 N')
figure;
hold on; grid on;

num_traj = numel(traj);

for i=1:num_traj
    sol = traj{i};
    o = ones(1,size(sol.x,2));
```

```
if sol.feasible
    color = 'g';
else
    color = 'r';
end

subplot(2,2,1)
hold on; grid on;
plot(sol.x(2,:),'Color',color,'markersize',20,'linewidth',2);
plot(1:size(sol.x,2),f(2)*o,'k','linewidth',2)
plot(1:size(sol.x,2),-f(8)*o,'k','linewidth',2)
ylabel('\theta_1 [rad]')

subplot(2,2,2)
hold on; grid on;
plot(sol.x(3,:),'Color',color,'markersize',20,'linewidth',2);
plot(1:size(sol.x,2),f(3)*o,'k','linewidth',2)
plot(1:size(sol.x,2),-f(9)*o,'k','linewidth',2)
ylabel('\theta_2 [rad]')

subplot(2,2,3)
o = ones(1,size(sol.u,2));
hold on; grid on;
plot(sol.x(3,:)+sol.x(2,:),'Color',color,'markersize',20,'linewidth',2);
plot(1:size(sol.u,2),f(13)*o,'k','linewidth',2)
plot(1:size(sol.u,2),-f(14)*o,'k','linewidth',2)
ylabel('\theta_1 + \theta_2 [rad]'); xlabel('Simulation steps, Ts=0.05 s')

subplot(2,2,4)
o = ones(1,size(sol.u,2));
hold on; grid on;
plot(sol.u,'Color',color,'markersize',20,'linewidth',2);
plot(1:size(sol.u,2),m(1)*o,'k','linewidth',2)
```

```
    plot(1:size(sol.u,2),-m(2)*o,'k','linewidth',2)
    ylabel('u'); xlabel('Simulation steps, Ts=0.05 s')
end
```

7.4.2. Maximal invariant set function

```
% Function which calculates maximal invariant set for automous system
% x_dot=Ax, with polyhedron set of constraints

function Xf = maxInvar(A, X)

while 1
  prevX = X;  %invariant set candidate
  T = X.A; t = X.b;
  preX  = Polyhedron(T*A,t); % preset of X calculation

%function polyhedron creates polyhedral set
%from matrix A and vector b such that {x | Ax <= b}
  X     = intersect(preX,X);
     %if X is completely within its preset, then it means that X is maximal
     %invariant set
     % translated, this means that the intersection of X and its preset is
     % equal to X
  if prevX == X
break
  end
end
Xf = X;

end
```

Bibliography

[1] M. Zeilinger; C. Jones; F. Borrelli; M. Morari. Lecture slides from model predictive control, part one – introduction, 2016.

[2] Andrej Jokić. Lecture slides from the optimisation course, 2020.

[3] Josip Kasać. Lecture slides from the fuzzy and digital control course, 2022.

[4] Mr. Keshav M. Jadhav. Introduction to dynamic programming and bellman's principle of optimality, accessed in May 2024.

[5] Branimir Novoselnik. Practical lectures from predictive control course, 2023.

[6] Ian J. P. Crowe-Wright. Control theory: The double pendulum inverted on a cart. Master's thesis, University of New Mexico, 2018.

[7] Dragan Pustaić; Hinko Wolf; Zdenko Tonković. *Mehanika III.* Golden marketing - Tehnička knjiga, 2005.

[8] Eric L. Haseltine and James B. Rawlings. Critical evaluation of extended kalman filtering and moving-horizon estimation. *Industrial and Engineering Chemistry Research,44, 2451-2460*, 2005.

[9] Mario Hrgetić. *Vehicle Dynamics State Estimation Based On Sensor Fusion By Adaptive Kalman Filter.* PhD thesis, Faculty of Electrical Engineering and Computing, 2015.

[10] California Institute of Technology Henrik Sandberg. Moving horizon estimation lecture slides, 2006.

[11] Kris Kitani. Extended kalman filter lecture slides, Carnegie Mellon University.

[12] Rudolph van der Merwe Eric A. Wan. The unscented kalman filter for nonlinear estimations. *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000.

[13] JEFFREY K. UHLMANN SIMON J. JULIER. Unscented filtering and nonlinear estimation. *PROCEEDINGS OF THE IEEE, VOL. 92, NO. 3*, 2004.

[14] SIMON J. JULIER. The scaled unscented transformation. *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, 2002.

[15] Caroline Svahn; Kristin Nielsen; Hector Rodriguez-Deniz; Gustaf Hendeby. Ukf parameter tuning for local variation smoothing. *Proceedings of the 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2021.

[16] Cyrill Stachniss. Slam course - 06 - unscented kalman filter, lecture slides, 2013/2014.

[17] Waycom series lmi12-sl and lmi12-se linear potentiometer datasheet, accessed in January 2024.

[18] Allegro microsystemsy a1334 angular hall sensor datasheet, accessed in January 2024.

[19] Spahija; Švec; Matuško; Ileš. Successive linearization based predictive vehicle torque vectoring. *2019 International Conference on Electrical Drives and Power Electronics*, 2019.

[20] Moritz M. Diehl James B. Rawlings, David Q. Mayne. *Model Predictive Control: Theory, Computation, and Design, 2nd Edition*. Nob Hill Publishing, 2017.