SLOVAK UNIVERSITY OF TECHNOLOGY
IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reg. No.: FCHPT-NaN

Nonlinear Model Predictive Control of Rotary Pendulum

SEMESTRAL THESIS

2020 **Alexey Morozov**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reg. No.: FCHPT-NaN

Nonlinear Model Predictive Control of Rotary Pendulum

SEMESTRAL THESIS

Study programme:	Process Control
Study field:	5.2.14. Automation
Training workspace:	Institute of Information Engineering, Automation, and Mathematics
Thesis supervisor:	Ing. Martin Klaučo, PhD.

2020 **Alexey Morozov**

Abstract

This work is dedicated to designing control strategies for the rotational inverted pendulum or Furuta's pendulum. Those strategies are Linear-quadratic regulator (LQR) with the Swing-up controller, Model Predictive Control (MPC) method with the Swing-up controller and the Nonlinear Model Predictive Control (NMPC) strategy. Those controllers are designed in a computing environment MATLAB, for NMPC has used MATMPC toolbox as well as the CasADi toolbox. After verifying in simulations those controllers are used to control the real process.

Contents

Chapter 1

Introduction

Nowadays the PID controller is the most commonly used controller in process control engineering. They are cheap, easy to tune and could be easily implemented in PLC what makes it a good choice for single-input and single-output (SISO) systems without significant time delay. However it almost unsuitable for multiple-input and multiple-output (MIMO) systems don't handle constraints and disturbances, and we can't guarantee that the PID controller ensures the best process behavior. That's why advanced control strategies are required.

This work is dedicated to designing the optimal controllers for the fast nonlinear process with complex dynamics. And the rotational inverted pendulum or Furuta's pendulum is a great example of such a process. It was invented in 1992 at Tokyo Institute of Technology by Katsuhisa as an example of a complex nonlinear oscillator. This pendulum is underactuated and extremely non-linear due to gravitational forces and Coriolis interactions.

As the model of the process behavior is available the final thesis is consists of two main parts: first is designing of the controllers and their testing in simulations and the second part is the final tuning and controllers testing on the real pendulum. The process is controlled in three different ways: first is the LQR combined with the Swing-up controller, the second is MPC controller with the Swing-up controller and the third is the NMPC controller. This work demonstrates how advanced control strategies could be used to control a complex process.

Chapter 2

Theory

2.1 Furuta's Pendulum

Rotational inverted pendulum or Furuta's pendulum composes of two main parts: motor-driven arm, which rotates in the horizontal plane and a pendulum, attached to that arm, which freely rotates in the vertical plane. The system is underactuated and extremely nonlinear due to the gravitational forces and the coupling arising from the Coriolis and centripetal forces. The schematic representation of the pendulum is shown in 2.1

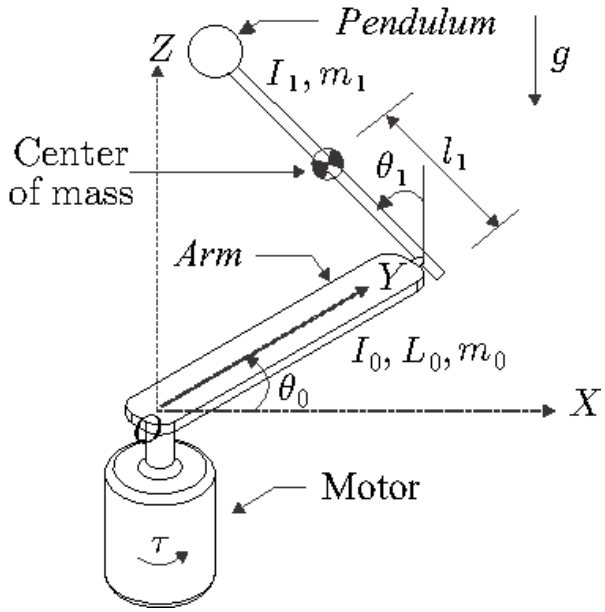

Figure 2.1: Furuta's Pendulum

The symbols in the figure indicate the following:

- g - gravitational acceleration [$\mathrm{m\,s^{-2}}$]
- m_0 - mass of arm [kg]
- m_1 - mass of pendulum [kg]
- L_0 - length of arm [m]
- L_1 - length of pendulum [m]
- l_1 - location of the pendulums center of mass [m]
- I_0 - moment of inertia of arm [$\mathrm{kg\,m^{-2}}$]
- I_1 - moment of inertia of pendulum [$\mathrm{kg\,m^{-2}}$]
- θ_0 - arm angle [rad]
- θ_1 - pendulum angle [rad]
- τ - motor torque [V]

2.1.1 State-Space Representation

To obtain the state representation of the process we must define our state variables first:

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^\intercal = \begin{bmatrix} \theta_0 & \dot{\theta_0} & \theta_1 & \dot{\theta_1} \end{bmatrix}^\intercal \tag{2.1}$$

Control variable:

$$u = \tau \tag{2.2}$$

Then we can write our state equations:

$$\dot{x_1} = \dot{\theta_0} \tag{2.3a}$$

$$\dot{x_2} = \frac{\gamma(\epsilon\dot{\theta_0}^2 + \rho) - \delta(\tau + \beta\dot{\theta_1}^2 - \sigma\dot{\theta_0}\dot{\theta_1})}{\gamma^2 - \alpha\delta} \tag{2.3b}$$

$$\dot{x_3} = \dot{\theta_1} \tag{2.3c}$$

$$\dot{x_4} = \frac{\gamma(\tau + \beta\dot{\theta_1}^2 - \sigma\dot{\theta_0}\dot{\theta_1}) - \alpha(\epsilon\dot{\theta_0}^2 + \rho)}{\gamma^2 - \alpha\delta} \tag{2.3d}$$

where

$$\alpha = I_0 + L_0^2 m_1 + l_1^2 m_1 \sin^2\theta_1 \tag{2.4}$$

$$\beta = L_0 m_1 l_1 \sin\theta_1 \tag{2.5}$$

$$\gamma = L_0 m_1 l_1 \cos\theta_1 \tag{2.6}$$

$$\delta = I_1 + l_1^2 m_1 \tag{2.7}$$

$$\epsilon = l_1^2 m_1 \sin\theta_1 \cos\theta_1 \tag{2.8}$$

$$\rho = m_1 g l_1 \sin\theta_1 \tag{2.9}$$

$$\tau = 2 l_1^2 m_1 \sin\theta_1 \cos\theta_1 \tag{2.10}$$

Now these non-linear differential equations we can write in the form of matrices:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} \dot{\theta_0} \\ \frac{\gamma(\epsilon \dot{\theta_0}^2 + \rho) - \delta(\tau + \beta \dot{\theta_1}^2 - \sigma \dot{\theta_0} \dot{\theta_1})}{\gamma^2 - \alpha\delta} \\ \dot{\theta_1} \\ \frac{\gamma(\tau + \beta \dot{\theta_1}^2 - \sigma \dot{\theta_0} \dot{\theta_1}) - \alpha(\epsilon \dot{\theta_0}^2 + \rho)}{\gamma^2 - \alpha\delta} \end{bmatrix} \tag{2.11}$$

or:

$$\dot{x} = f(x,u) = \begin{bmatrix} f_1(x,u) \\ f_2(x,u) \\ f_3(x,u) \\ f_4(x,u) \end{bmatrix} \tag{2.12}$$

So that's our non-linear dynamic model of the process. But only the NMPC controller is able to operate with such a model. So, to make that model suitable for LQR and MPC controller we can approximate that non-linear model by a linear model as follows:

$$\dot{x} = Ax + Bu \tag{2.13}$$

And the constant matrices are derived as:

$$A = \begin{bmatrix} \frac{\partial f_1(x,u)}{\partial x_1} & \frac{\partial f_1(x,u)}{\partial x_2} & \frac{\partial f_1(x,u)}{\partial x_3} & \frac{\partial f_1(x,u)}{\partial x_4} \\ \frac{\partial f_2(x,u)}{\partial x_1} & \frac{\partial f_2(x,u)}{\partial x_2} & \frac{\partial f_2(x,u)}{\partial x_3} & \frac{\partial f_2(x,u)}{\partial x_4} \\ \frac{\partial f_3(x,u)}{\partial x_1} & \frac{\partial f_3(x,u)}{\partial x_2} & \frac{\partial f_3(x,u)}{\partial x_3} & \frac{\partial f_3(x,u)}{\partial x_4} \\ \frac{\partial f_4(x,u)}{\partial x_1} & \frac{\partial f_4(x,u)}{\partial x_2} & \frac{\partial f_4(x,u)}{\partial x_3} & \frac{\partial f_4(x,u)}{\partial x_4} \end{bmatrix} \tag{2.14}$$

respektive:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-g L_0 l_1^2 m_1^2}{(m_1 L_0^2 + I_0)(m_1 l_1^2 + I_1) - L_0^2 l_1^2 m_1^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g l_1 m_1 (m_1 L_0^2 + I_0)}{(m_1 L_0^2 + I_0)(m_1 l_1^2 + I_1) - L_0^2 l_1^2 m_1^2} & 0 \end{bmatrix} \tag{2.15}$$

$$B = \begin{bmatrix} \frac{\partial f_1(x,u)}{\partial u} \\ \frac{\partial f_2(x,u)}{\partial u} \\ \frac{\partial f_3(x,u)}{\partial u} \\ \frac{\partial f_4(x,u)}{\partial u} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{m_1 L_1^2 + I_1}{(m_1 L_0^2 + I_0)(m_1 l_1^2 + I_1) - L_0^2 l_1^2 m_1^2} \\ 0 \\ \frac{-L_0 l_1 m_1}{(m_1 L_0^2 + I_0)(m_1 l_1^2 + I_1) - L_0^2 l_1^2 m_1^2} \end{bmatrix} \tag{2.16}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.17}$$

$$D = 0 \tag{2.18}$$

The linearized equations of motion for the simplified system are now could be derived for two equilibrium positions: upright and downward. The reason is that at the downward position the system's output, which is the position of the pendulum, has a stable point at "$+\pi$" and "$-\pi$", while at the upright position system has no stable point.

The model, obtained by linearization around the upright operation point, is used for fulfilling the main control objective, which is stabilizing the pendulum at the upright position. The second model is used to simulate process behavior during initial excitation by a Swing-up controller.

2.2 Controller Synthesis

2.2.1 LQR design

The Linear Quadratic Regulator (LQR) is a well-known method that provides optimally controlled feedback gains to enable the closed-loop stable and high performance design of systems.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx, C = I^{n \times n} \end{aligned} \tag{2.19}$$

which requires full-state feedback (all n states are measurable). The feedback gain is a matrix K and the feedback control law takes the form:

$$u = -Kx \tag{2.20}$$

Than closed-loop system dynamics can be written:

$$\dot{x} = (A - BK)x \tag{2.21}$$

So there appears condition that poles of $(A-BK)$ must be in in stable, suitably-damped locations in the complex plane.

2.2.1.1 Derivation of LQR

Towards a generic procedure for solving optimal control problems, we derive a methodology based on the calculus of variations. The problem statement for a fixed final time t_f is:

$$\begin{aligned} J = \min_u \quad & \Phi(t_f) + \frac{1}{2}\int_{t_0}^{t_f} L(x(t), u(t), t)\, dt \\ s.t. \quad & \dot{x} = f(x(t), u(t), t) \\ & x(t_0) = x_0 \end{aligned} \tag{2.22}$$

In the case of the Linear Quadratic Regulator (with zero terminal cost), we set $\Phi(x(t_f)) = 0$, and $L = x^\intercal Q x + u^\intercal R u$ solve the new optimization problem

$$J = \min_u \frac{1}{2}\int_{t_0}^{t_f} x^\intercal Q x + u^\intercal R u\, dt \tag{2.23}$$

Where Q and R are positive semidefinite weighting matrices for states and control input respectively. Now we define new variable H as:

$$H = \frac{1}{2}(x^\intercal Q x + u^\intercal R u) + \lambda^\intercal (Ax + Bu) \tag{2.24}$$

As we are looking for minimum, we apply the condition, that $\frac{\partial H(x,u,\lambda)}{\partial u}$ is equal to zero.

$$\frac{\partial H(x, u, \lambda)}{\partial u} = Ru + B^\intercal \lambda = 0 \tag{2.25}$$

then:

$$u = -R^{-1} B^\intercal \lambda \tag{2.26}$$

where

$$\lambda = Px \tag{2.27}$$

and finally

$$u = -R^{-1} B^\intercal P x \tag{2.28}$$

And that is our optimal control law for full-state feedback LQR. Matrix P is calculated by solving Riccati equation in continuous time:

$$Q + A^\intercal P + PA - PBR^{-1}B^\intercal P = 0 \tag{2.29}$$

2.2.2 MPC controller design

MPC uses a model of the system to make predictions about the system's future behavior. MPC solves an online optimization algorithm to find the optimal control action that drives the predicted output to the reference. MPC can handle MIMO systems that may have interactions between their inputs and outputs. It can also handle input and output constraints. MPC has preview capability; it can incorporate future reference information into the control problem to improve controller performance.

2.2.2.1 MPC formulation

MPC controller requires the model of the system in the discrete time formulation that is used to predict systems behavior in the future:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{2.30}$$

State predictions:

$$\begin{aligned} \hat{x}_{k+1} &= Ax_k + Bu_k \\ \hat{x}_{k+2} &= A\hat{x}_{k+1} + Bu_{k+1} \\ &= A^2x_k + ABu_k + Bu_{k+1} \\ \hat{x}_{k+3} &= A\hat{x}_{k+2} + Bu_{k+2} \\ &= A^3x_k + A^2Bu_k + ABu_{k+1} + Bu_{k+2} \\ &\vdots \\ \hat{x}_{k+N} &= A^Nx_k + \sum_{j=k}^{k+N-1} A^jBu_{k+N-j-1} \end{aligned} \tag{2.31}$$

Output Predictions:

$$\begin{aligned} \hat{y}_{k+1} &= C\hat{x}_{k+1} \\ &= CAx_k + CBu_k \\ \hat{y}_{k+2} &= C\hat{x}_{k+2} \\ &= CA^2x_k + CABu_k + CBu_{k+1} \\ \hat{y}_{k+2} &= C\hat{x}_{k+2} \\ &= CA^3x_k + CA^2Bu_k + CABu_{k+1} + CBu_{k+2} \end{aligned} \tag{2.32}$$

Now we can obtain predictive system model:

$$\begin{bmatrix} \hat{y}_{k+1} \\ \hat{y}_{k+2} \\ \vdots \\ \hat{y}_{k+N} \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} x_k + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1} & CA^{N-2} & \cdots & CB \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix} \tag{2.33}$$

respektive:

$$\hat{Y} = Y_0 + GU \tag{2.34}$$

And now the whole optimization problem could be written as a QP problem with constrains:

$$\begin{aligned} J = \min_u \sum_k^{k+N} & x^\intercal Q_x x + u^\intercal Q_u u \\ s.t. \quad & x_{k+1} = Ax_k + Bu_k \\ & y_k = Cx_k \\ & u_{min} \leq u_k \leq u_{max} \end{aligned} \tag{2.35}$$

Where Q_x and Q_u are positive semidefinite weighting matrices for states and control input respectively. By solving that optimization problem we get the optimal value for control input for every simulation step.

2.2.3 Swing-Up controller design

For initial excitation of the system we use the energy-based swing-up controller. The strategy with this controller is that we increase the amplitude of swings by increasing the energy of the system with every swing. The energy is added by controlling arms movements and depends on the actual energy of the pendulum. The actual energy of the pendulum can be calculated from the actual position of the pendulum and its velocity:

$$E = \frac{m_1 g l_1}{2}((\frac{\dot{\theta_1}}{\omega_0})^2 + \cos\theta_1 - 1) \tag{2.36}$$

Than the control law has following form:

$$u = k_v E sign(\dot{\theta_1}\cos\theta_1) \tag{2.37}$$

Where element $sign(\dot{\theta_1}\cos\theta_1)$ determines direction i which the force will be applied and $k_v E$ is the gain of the controller.

Chapter 3

Simulation Results

3.1 LQR

As first we design a Linear-Quadratic regulator to control the process in the upright position. To simulate the system's behavior we use matrices A and B, which were linearised relative to the upper operating point.
To design an LQR we have to define weight matrices Q and R:

$$Q = \begin{bmatrix} 0.6 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \quad R = 1 \tag{3.1}$$

Then we calculate state-feedback gain:

$$K = \begin{bmatrix} -0.5309 & -0.5347 & -6.9738 & -0.9858 \end{bmatrix} \tag{3.2}$$

Now we choose the initial condition:

$$x_0 = \begin{bmatrix} -1 & -2 & 0.5 & 2 \end{bmatrix}^\intercal \tag{3.3}$$

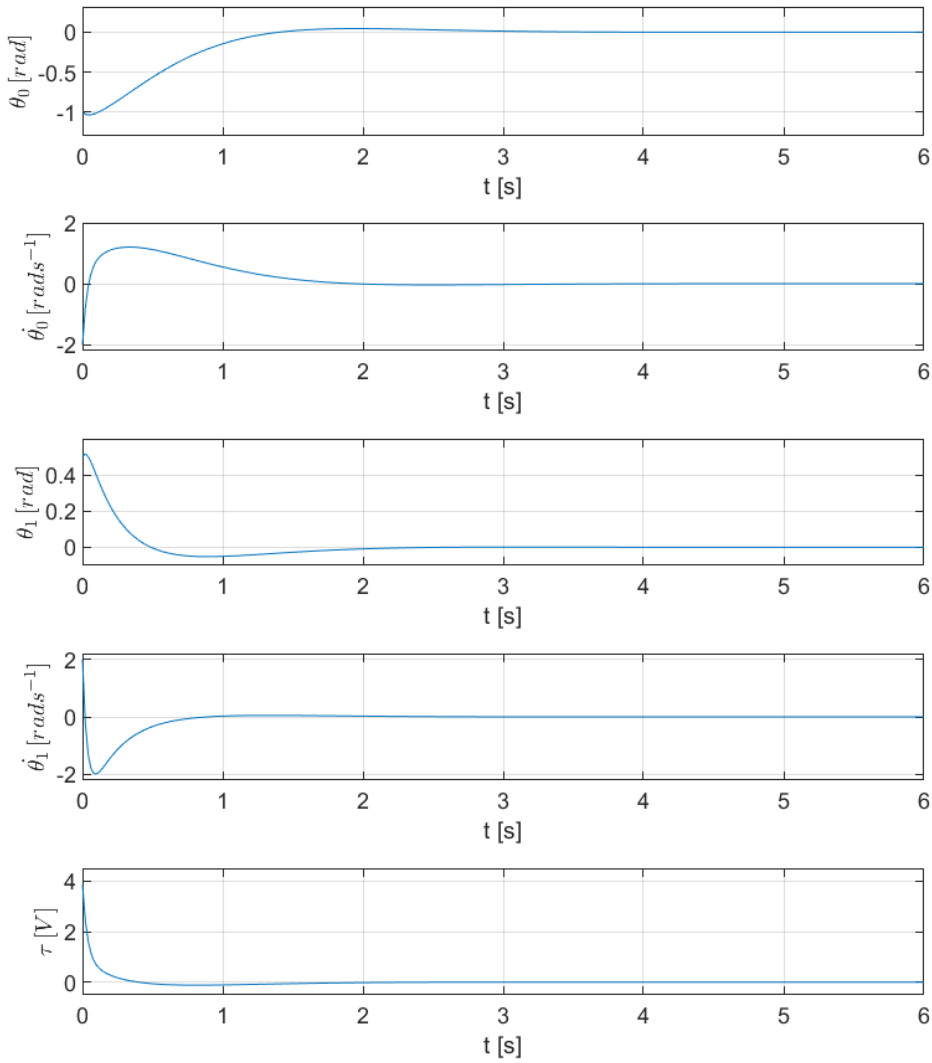

Figure 3.1: Control with LQR: simulation

3.2 MPC

An MPC comtoller will be used with the same purpose as LQR: to control the pendulum at the upright position. The optimization problem **??** is solved via using *quadprog* solver in MATLLAB. To design an MPC controller we have to define weight matrices Q_x and Q_u:

$$Q_x = \begin{bmatrix} 1.5 & 0 & 0 & 0 \\ 0 & 0.08 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix} \quad Q_u = 1 \tag{3.4}$$

And set constrains for the control input:

$$\begin{aligned} u_{min} &= -5 \\ u_{max} &= 5 \end{aligned} \tag{3.5}$$

We use the same initial condition as with the LQR

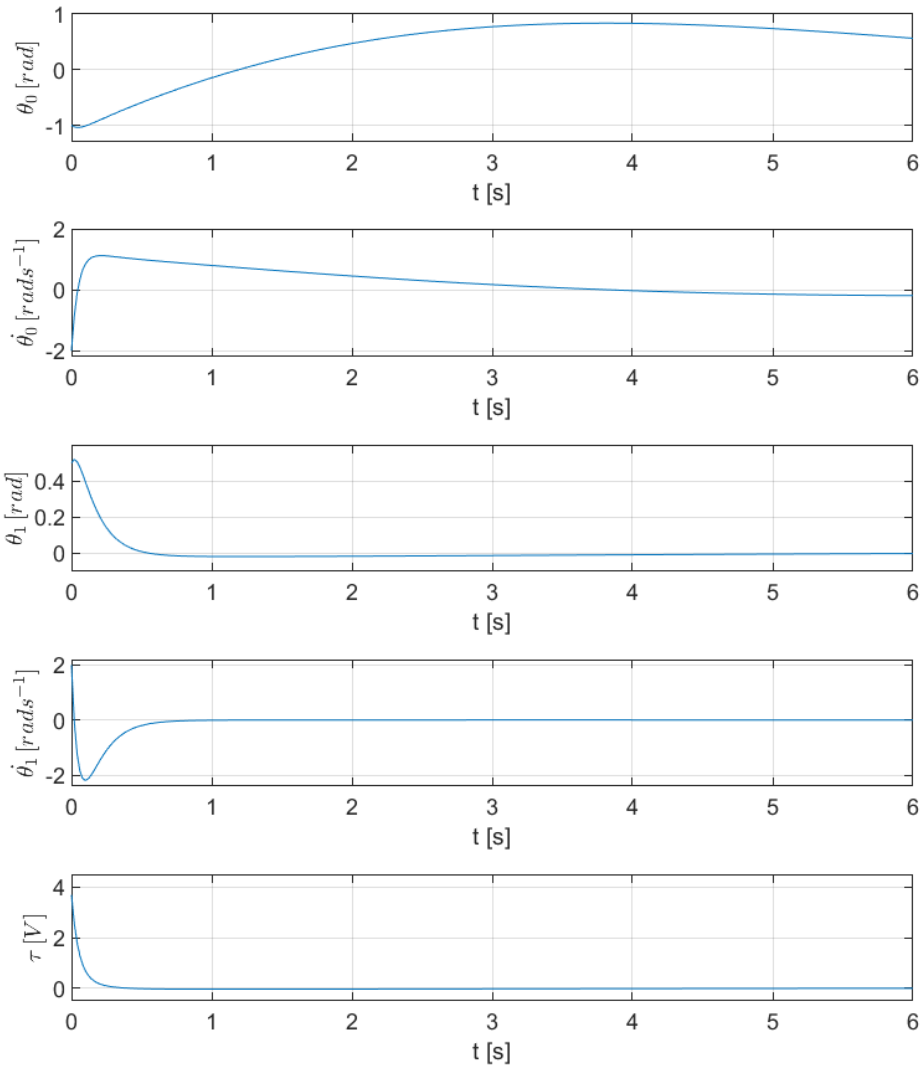

Figure 3.2: Control with MPC: simulation

3.3 Swing-Up

The main purpose of the Swing-Up controller is to swing the pendulum from the downside position into the upright position where the control of the process will be taken by another controller.

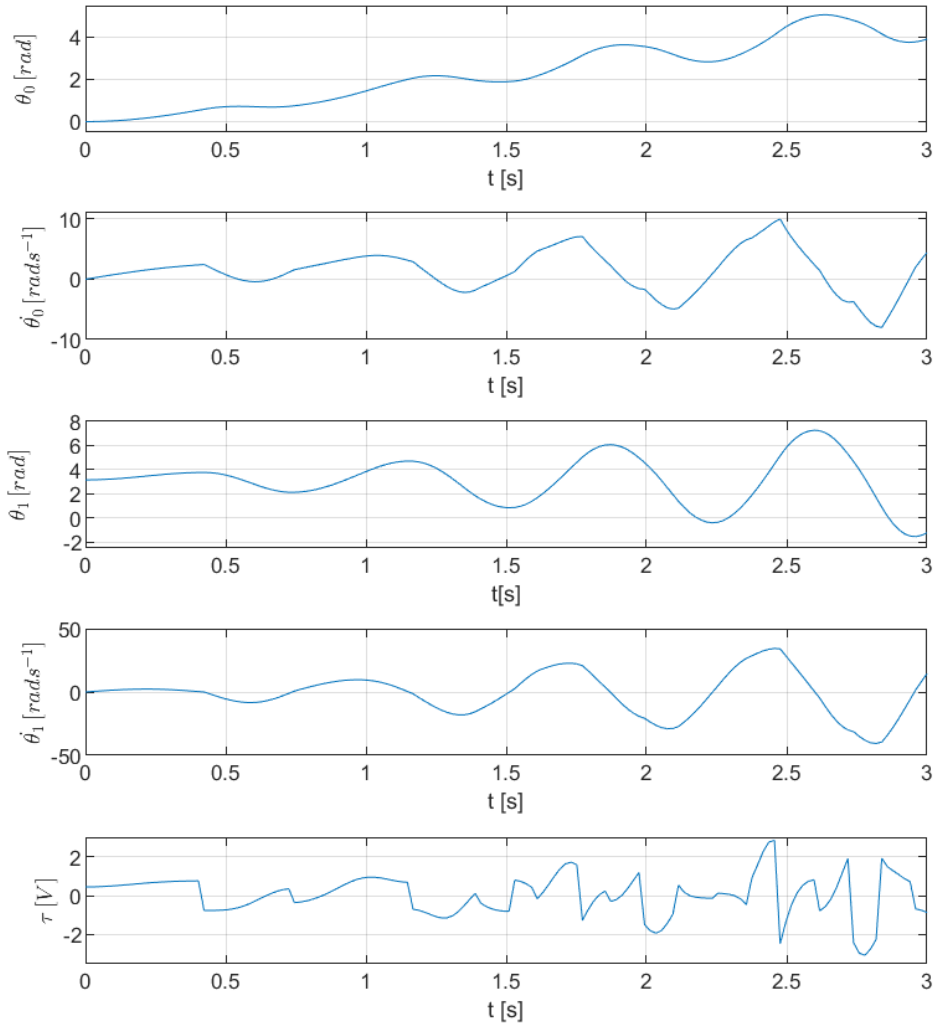

Figure 3.3: Control with Swing-Up: simulation

3.4 Combined Control Strategy

Now we combine the Swing-Up controller with an LQR or MPC controller to fully control the process. The strategy is the following: at the beginning, we oscillate the pendulum from the steady-state at the downside position with the Swing-Up controller. With every swing the amplitude of swings is increasing. And as the position of the pendulum reaches the value of 0.5 radians, we are switching to another controller. And also we have to recalculate the predicted states because while we control the system with the Swing-Up controller, states are calculated via matrices A and B, which were linearised relative to downside operating point. While LQR and MPC controller require matrices A and B, which were linearised relative to the upright operating point.

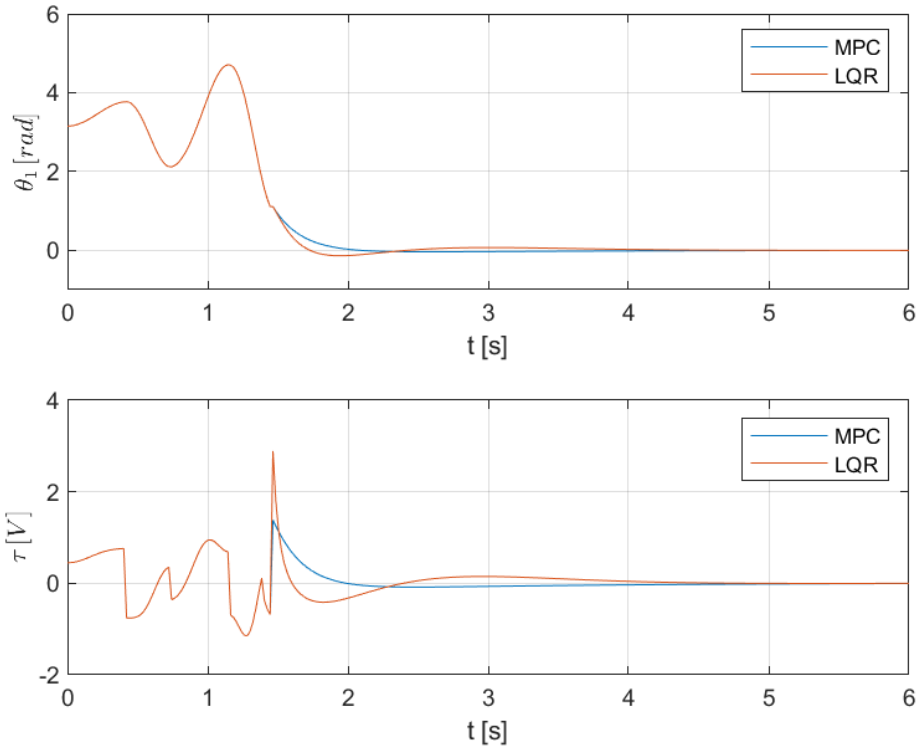

Figure 3.4: Combined LQR/Swing-Up and MPC/Swing-up control: simulation

3.5 NMPC

NMPC controller is able to fully control the process by itself. To design such controller we use freely availabe *MATMPC* toolbox, which could be downloaded from *https://github.com/chenyutao36/MATMPC*. Also additional software is required:

- **CasAdi** - the state-of-the-art automatic/algorithmic differentiation toolbox.
- **MinGW-w64 C/C++ Compiler** - algorithmic routines are compiled into MEX functions using this compiler.

As we have all the toolboxes, we use nonlinear model of the process 2.11 and weight matrices Q_x and Q_u, which are designed as:

$$Q_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \quad Q_u = 0.1 \tag{3.6}$$

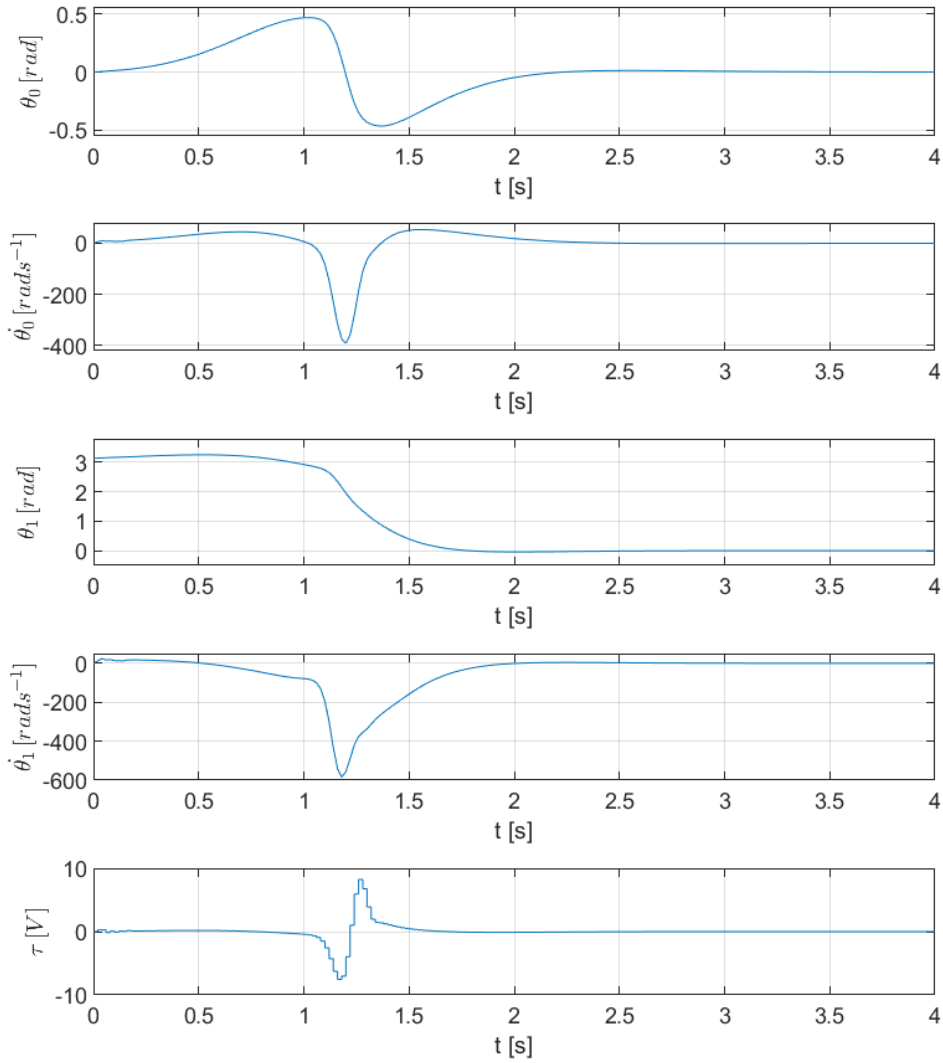

Figure 3.5: NMPC control: simulation

Chapter 4

Conclusions

The goal of this thesis was to design a set of controllers for the underactuated fast and extremely non-linear process, which was Furuta's pendulum. For that purpose Swing-Up, LQR, MPC and NMPC controllers were designed.
In the beginning, we got the nonlinear dynamic model of the process. But only NMPC controller can control the process, which is described by such model. To make it suitable for Swing-Up, LQR and MPC controllers we had to linearise it relative to two operating points: at the downside position, where the process is stable, and at the upright position, where the process is unstable. The reason for that is that the process is controlled by LQR and MPC controllers when the pendulum is in the upper position, and it is controlled by a Swing-Up controller when the pendulum is in the lower position.
All controllers were designed using MATLAB and also tested in simulations. By evaluating simulation results we can conclude that all controllers were designed properly, and could be used to control the real process.

Bibliography

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, USA, 7th edition, 2009.