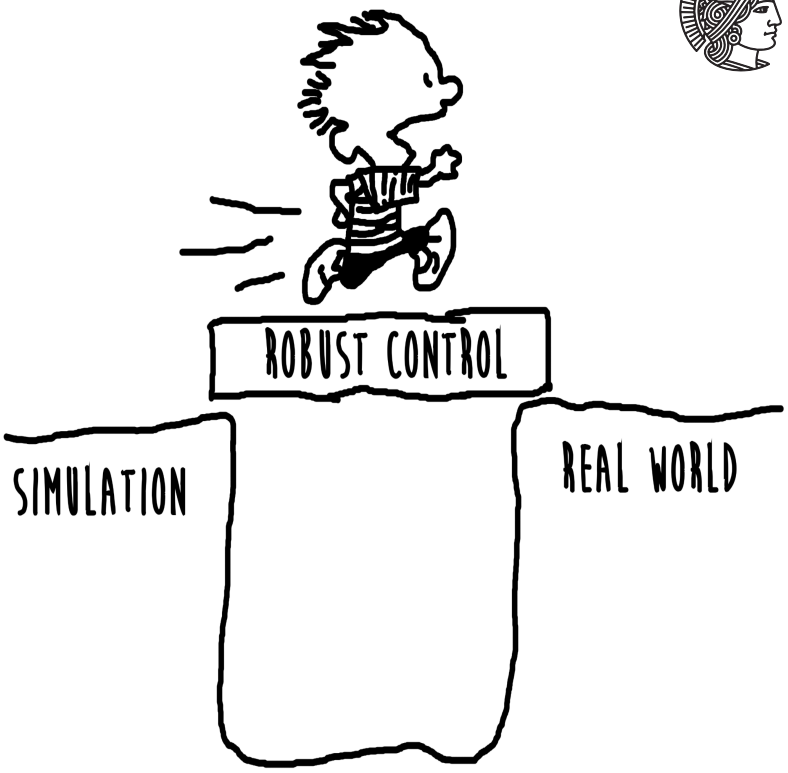
Benchmarking Robust Control against Reinforcement Learning Methods on a Robotic Balancing Problem

Markus Lamprecht

Master Thesis, *15. August 2018– 14. February 2019*

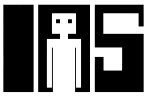
Supervisors: Prof. Heinz Koeppl, Prof. Jan Peters, Dr. Michael Gienger
and M.Sc Fabio Muratore

Departement of Electrical Engineering and Information Technology



Technical University Darmstadt
Bioinspired Communication Systems Lab
Departement of Electrical Engineering and Information Technology
www.bcs.tu-darmstadt.de
Prof. Heinz Koeppl

Technical University Darmstadt
Intelligent Autonomous Systems Lab
Computer Science Department
www.ias.informatik.tu-darmstadt.de
Prof. Jan Peters



Honda Research Institute EU
www.honda-ri.de
Dr. Michael Gienger and M.Sc Fabio Muratore

Erklärung zur Abschlussarbeit gemäß § 22 Abs. und § 23 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Markus Lamprecht, die vorliegende Master-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Darmstadt, 14. February 2019 _____

Markus Lamprecht

English translation for information purposes only:

Thesis Statement pursuant to § 22 paragraph 7 and § 23 paragraph 7 of APB TU Darmstadt I herewith formally declare that I, Markus Lamprecht, have written the submitted thesis independently pursuant to § 22 paragraph 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form. I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once. In the submitted thesis the written copies and the electronic version for archiving are pursuant to § 23 paragraph 7 of APB identical in content.

Abstract

As simulators represent an inexpensive, fast and save test environment, it is a common practice to evaluate and optimize controllers of robotic systems in physics simulators, before applying them to the real robot. Identifying precise physics parameters that are required by the simulators is difficult. Thus one can observe a drop in performance when testing the designed controller on the real system. One approach to bridge this reality gap is to design robust controllers that are able to stabilize the system for parameter uncertainties. Within this thesis a Multi-Model Pole Placement (MMPP) and a $\mathscr{H}_2$ fixed-structure robust controller is designed. The robustness of both is examined on a seven DoF Schunk arm, that balances a ball on a plate. The static controllers that were designed with respect to a precise model are able to stabilize the Ball-on-Plate system for different radii and rolling friction coefficients of the ball. The balancing behaviour is simulated in the robot control system environment which is developed by the Honda Research Institute Europa. These controllers are compared against a robust controller that is designed using Proximal Policy Optimization (PPO). In the simulation the neuronal network trained with PPO revealed a faster balancing behaviour compared to the MMPP and $\mathscr{H}_2$ fixed-structure controller. On the real robot a Linear Quadratic Regulator (LQR) controller with an additional integrative part was able to balance balls with different radii and for different rolling friction coefficients. Using this controller the reality gap could be crossed.

Keywords: Robust control, Multi-Model Pole Placement, fixed-structure $\mathscr{H}_2$ controller, Physics simulation, LQR, PPO.

Contents

Symbols and Abbreviations

Symbols

Symbol	Description	Unit
x	Ball position in Plate coordinates along x_p - Axis	m
y	Ball position in Plate coordinates along y_p - Axis	m
z	Ball position in Plate coordinates along z_p - Axis	m
$\mathbf{r_B}$	Ball position vector in solid coordinate system	m
$\mathbf{v_B} = \dot{\mathbf{r}}_B$	Ball velocity vector in solid coordinate system	m/sec
$\omega_\mathbf{B}$	Ball angular velocity vector in solid coordinate system	rad/sec
α_B	Ball rotation angle around its x_B axis	rad
β_B	Ball rotation angle around its y_B axis	rad
$\omega_\mathbf{P}$	Plate angular velocity vector in solid coordinate system	rad/sec
α	Plate angle. $\alpha > 0$ when the plate rotates around x_P[1]	rad
β	Plate angle. $\beta > 0$ when the plate rotates around y_P[2]	rad
γ	Plate angle around z_P (with right hand rule).	rad
x_p	Axis of plate coordinate frame	
y_p	Axis of plate coordinate frame	
z_p	Axis of plate coordinate frame	
w	Width of the plate	m
l	Length of plate	m
h	Height of plate	m
F_x	Force along X_{world}	N
F_y	Force along Y_{world}	N
F_z	Force along Z_{world}	N
τ_x	Moment around of x_p	Nm
τ_y	Moment around of y_p	Nm
X_{World}	Axis of World coordinate frame	
Y_{World}	Axis of World coordinate frame	
Z_{World}	Axis of World coordinate frame	
X	Plate position in World coordinates along X_{World} - Axis	m
Y	Plate position in World coordinates along Y_{World} - Axis	m
Z	Plate position in World coordinates along Z_{World} - Axis	m

[1] with right hand rule
[2] with right hand rule

Symbols

Symbol	Description	Unit
g	Gravitational constant	$\frac{m}{s^2}$
d	Thickness of a hollow sphere	m
r	Radius of the ball	m
m, m_B	Mass of the ball	kg
m_P	Mass of the plate	kg
j_B	Constant for inertia of the ball	kg m^2
$\mathbf{I}_B$	Inertia matrix of the ball	kg m^2
$\mathbf{I}_P$	Inertia matrix of the plate	kg m^2
L	Lagrange parameter	Nm
T	Kinetic energy parameter of Lagrange approach	Nm
V	Potential energy parameter of Lagrange approach	Nm
$\mathbf{q}$	Vector with generalized coordinates	
$\mathbf{Q}$	Vector with generalized forces and torques used in Lagrange approach	
$\mathbf{R}$	$\mathbf{R}_x$ and $\mathbf{R}_y$ are rotational matrices around the x and y axis	
$\mathbf{x}$	Vector with the states of the state space representation of the BoP system	
$\mathbf{u}$	Vector with the control inputs of the state space representation of the BoP system	
$\mathbf{A}$	System matrix of the state space representation of the BoP system	
$\mathbf{B}$	Control input matrix of the state space representation of the BoP system	
$\mathbf{C}$	Output matrix of the state space representation of the BoP system	
R	Friction coefficient	
P	Dissipation function	
μ_R	Rolling friction constant of the Ball	
A	Contact surface that causes the air friction of the Ball	m^2
ρ	Air density of the surrounding medium of the ball	kg/m^3
c_W	Drag coefficient	

Acronyms

AS	Actuator Saturation
ASL	Actuator Saturation Limit
BoP	Ball-on-Plate
BT	Ball Tracking
cBoP	coupled Ball-on-Plate
dBoP	decoupled Ball-on-Plate
DoF	Degree of Freedom
EP	Equilibrium Point
EPOpt	Ensemble Policy Optimization
FIFO	First In - First Out
FK	Forward Kinematics
FTS	Force Torque Sensor
HSV	Huge Saturation Value
IK	Inverse Kinematics
KF	Kalman Filter
LFT	Linear Fractional Transformation
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
LWA	Light Weight Arm
MIMO	Multiple-Input-Multiple-Output
MMPP	Multi-Model Pole Placement
NP	Nominal Performance
NS	Nominal Stability
OpenCV	Open Computer Vision (a library for image processing)
PPO	Proximal Policy Optimization
TRPO	Trust Region Policy Optimization
RARL	Robust Adversarial Reinforcement Learning
Rcs	Robot Control System
RL	Reinforcement Learning
ROS	Robot Operating System
RP	Robust Performance
RS	Robust Stability
s. t.	subject to
SRR	Step Response Requirements
SRRP	Step Response Robust Performance
STD	Standard Deviation
SSV	Structured Singular Value

1 Introduction

As simulators represent an inexpensive, fast, safe and easy to change test environment, it is a common practice today to analyse controllers of robotic systems with Multibody physics simulators[1] such as Bullet, Vortex or MuJoCo before applying them to the real world. However, every physics simulation is by definition an approximation of the real world. One reason is the fact that simulators require accurate, but often unknown physical parameters. Moreover, physics simulations struggle with precisely modelling non-linear behaviour as contact reaction. Thus, applying controllers which have been examined in idealized simulators to the real system one can observe a drop in performance. This problem called reality gap can be significantly enough to lead to instability and thereby to complete failure. Overcoming this issue is a recent research topic.

One approach to bridge the reality gap is to design robust controllers that are able to stabilize a system and guarantee a minimal performance even though the exact physics parameters are not known. A robust controller can either be designed as a static controller that does not change its structure or parameter values or as a non-static controller which is changing during the control of the system. Non-static controllers like adaptive controllers [1, p.319] determine the feedback control with respect to previously identified plant physics parameters. Åström et. al. compared robust and adaptive controllers and concluded that robust controllers with static gains will respond faster to variations in process parameters if these are within the design specifications. Adaptive systems respond more slowly, but have less steady state error. In general it can be said, that adaptive controllers achieve a higher performance if the identification process of the process parameters converges [2]. Thus, the disadvantage of adaptive control is that a precise identification is required and stability cannot easily be proven. For this reason static robust controllers are examined in this thesis. Control theory methods for designing robust controllers with a static gain can be divided in two groups. The first group considers the exact uncertainties limits, whereas the second group assumes a generic uncertainty. $\mathscr{H}_\infty$ loop shaping is e.g. a method that does not require the precise uncertainty range, but optimizes the open loop of a system assuming a generic uncertainty [3, p. 134]. Algorithms that are based on precise knowledge of the uncertainty range are e.g. Multi-Model Pole Placement (MMPP). This method aims to place the poles of a defined set of models within a particular region. Another method of the first group is to design a fixed-structure $\mathscr{H}_2$ controller.

Besides using methods from control theory, Treede [4] demonstrated the design of robust controllers using novel methods from Reinforcement Learning (RL) for example Ensemble Policy

[1] For detailed information about these simulators see: *Bullet*, *Vortex* and *MuJoCo*.

Optimization (EPOpt) or Trust Region Policy Optimization (TRPO).
The aim of this thesis was to design robust controllers with methods from control theory and compare these controllers against methods from RL. In order to benchmark the designed controllers a Ball-on-Plate (BoP) was utilized. Balancing a ball on a plate is a challenging problem, because it represents an underactuated system with non-negligible dynamics and changing contact situation.

In a first step the model of a BoP-system for seven Degree of Freedom (DoF) Schunk arm is derived (see Chapter 2). Secondly an image-based Ball Tracking (BT) algorithm was developed to obtain the position and velocity of the ball (see Chatper 3). In a third step, a LQR, MMPP and a fixed-structure $\mathscr{H}_2$ controller are designed (see Chapter 4). Chapter 5 describes the robust controller designed with methods from RL. The controllers are evaluated using the Robot Control System (Rcs) simulation environment which is developed by the Honda Research Institute Europa (see Chapter 6). In Chapter 7 certain problems (such as measurement noise, image processing time delay) that occurred, when applying the designed controllers to the robot, are discussed. Additionally, a LQR controller with an integrative part is analysed on the real robot. With the experience gained from these experiments, a guideline of how to cross the reality gap is given. The results are summarized in Chapter 8.

2 Modelling of the Ball-on-Plate system

In order to determine a control law for balancing a ball on a plate, a model of the system is required. This chapter describes the setup of the Ball-on-Plate (BoP) system. Moreover, two models are discussed for this setup. The first model considers the plate angles as actuating variables. In the following, this model is denoted as decoupled Ball-on-Plate (dBoP) system, because the tilting angles can be controlled directly by two motors. The second model additionally controls the end-effector position of the robot. This model is denoted as coupled Ball-on-Plate (cBoP) system. Both models are validated by using Matlab. The code is given in Appendix E.

2.1 Setup of the Ball-on-Plate system

The setup of the real BoP system constructed at the Honda Research Laboratory is illustrated in Figure 2.1.

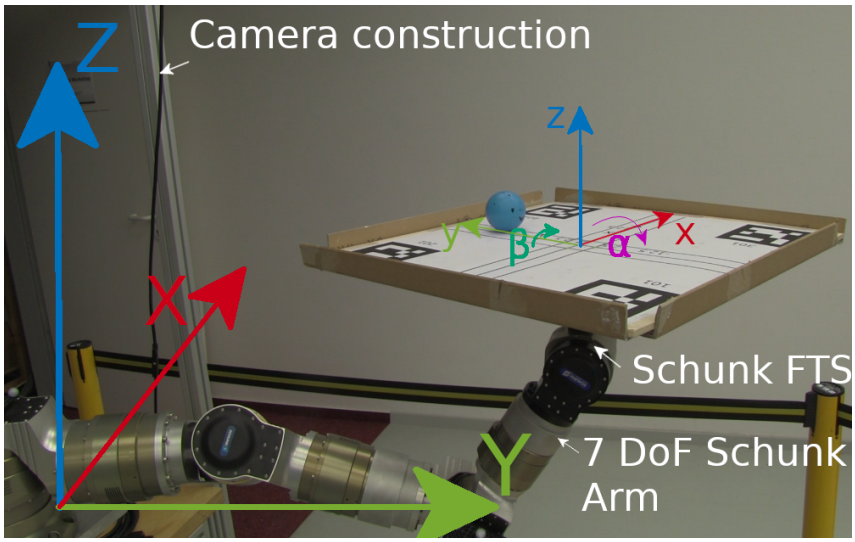


Figure 2.1.: Ball-on-Plate setup: Seven DoF Schunk Arm, Realsense R200 Camera and FTS to track the ball lying on the plate

A Realsense R200[1] camera or the Force Torque Sensor (FTS) mounted on the robot's end-effector observes the ball position relatively to the Plate. A seven DoF Light Weight Arm (LWA) by Schunk is used to control the pose of the plate's end-effector. The pose of the plate's end-effector is the tip position and orientation of the LWA. It consists of the distance to the world coordinate frame $(X,\ Y,\ Z)$ and the inclination angles of the plate $(\alpha,\ \beta)$, which are defined

[1] For additional information see *datasheet Realsense R200*.

defined around the axis of the local plate coordinate frame $(x,\ y,\ z)$. The ball itself is measured in the plate coordinate system.

Given the task of controlling the ball to a desired relative position $(x_{des},\ y_{des})$, it is required to measure the states of the system that are defined as the plate position and orientation in world coordinates, the relative position of the ball and the derivatives of these states. The measurement vector is given by

$$x_{meas} = \begin{bmatrix} X & Y & Z & \alpha & \beta & x & y & \dot{X} & \dot{Y} & \dot{Z} & \dot{\alpha} & \dot{\beta} & \dot{x} & \dot{y} \end{bmatrix}^T .$$

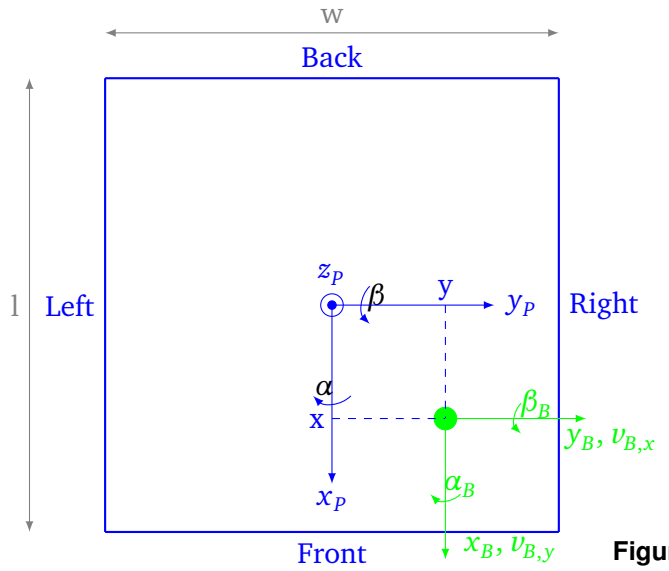


Figure 2.2.: Top view of the BoP system

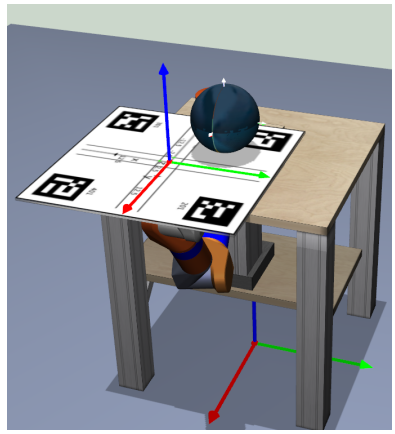


Figure 2.3.: All coordinate frames of the BoP system

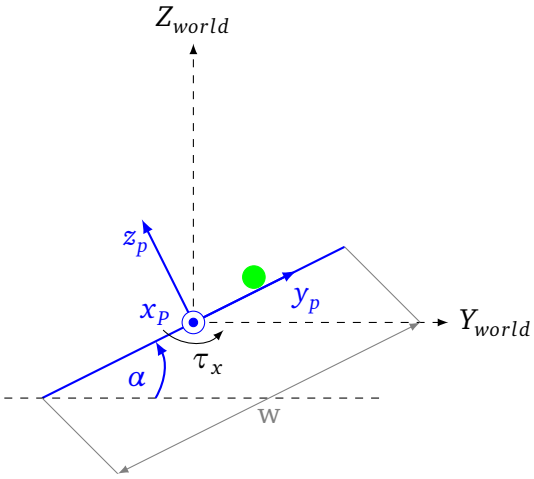


Figure 2.4.: Right up view ($\alpha > 0$) of BoP system

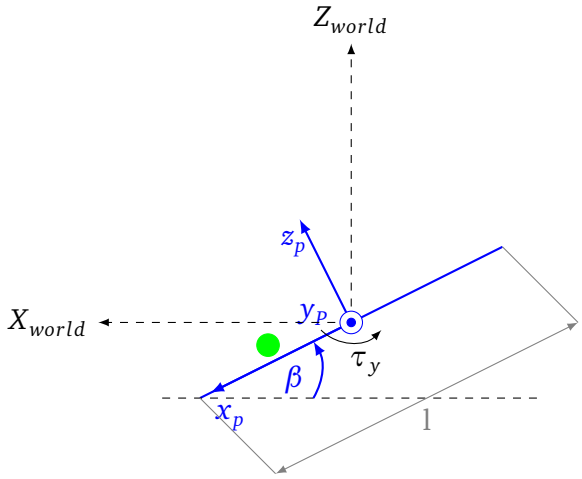


Figure 2.5.: Back up view ($\beta > 0$) of BoP system

In order to model the BoP system, a decoupled Ball-on-Plate (dBoP) and coupled Ball-on-Plate (cBoP) system is given. The dBoP system considers the angles of the plate as actuating variables, whereas the cBoP system additionally considers the position of the plate as actuating variables. In total three coordinate systems are defined for both systems.

Figure 2.2 shows the frame of the ball (axis: x_B, y_B, angles: α_B, β_B) and the frame of the plate (axis: x_P, y_P, z_P, angles: α, β). Figure 2.4 and 2.5 show the definition of the angles of the plate relative to the world coordinate frame (axis: X_{World}, Y_{World}, Z_{World}). Figure 2.3 shows all coordinate frames.

In order to understand the BoP system more deeply, the following facts are summarized:

- The world frame coordinates are written with capital letters and the plate frame coordinates with lower case letters.
- α, β are the angles measured between the coordinate system of plate and the world.
- The position of the ball measured in coordinates of the plate is: ${}_P\mathbf{r}_B = [x, y, z]^T$.
- The position of the plate measured in coordinates of the world is: ${}_W\mathbf{r}_P = [X, Y, Z]^T$
- $\alpha > 0 \rightarrow y \downarrow$: If $\alpha > 0$, right side of plate goes up and the y position of the ball is decreasing.
- $\beta > 0 \rightarrow x \uparrow$: If $\beta > 0$, back side of plate goes up and the x position of the ball is increasing.
- Additionally, the coordinate frames of the real cBoP system are depicted in Figure 2.1.

2.2 Decoupled Ball-on-Plate system

In the decoupled four DoF BoP system the inclination angles of the plate can be manipulated. The position of the end-effector of the robot holding the plate is constant. The state vector is: $\mathbf{x} = \begin{bmatrix} x & y & \alpha & \beta & \dot{x} & \dot{y} & \dot{\alpha} & \dot{\beta} \end{bmatrix}^T$
To derive a model for this system the Lagrange formalism is used. Thereby it is important to note that the total energy of a system remains constant. For this reason it does not matter in which coordinate frame the modelling is done. However, the easiest is to do it in coordinates of the plate.

2.2.1 Assumptions

Assumptions to model the dBoP system:

- Ball is not rotating around its vertical axis ($\gamma = 0$).
- Ball is lying on the plate ($z = r$).
- The inertia of the ball is considered as point mass.
- The arm of the robot holding the plate is assumed to rotate the plate around α and β. The plate is not assumed to move. Thus there is no force on the ball but the torques τ_x, τ_y around α and β. This means $X = Y = Z = F_x = F_y = F_z = 0$
- The friction of the ball on the plate is ignored.

In order to calculate the equations of motions of the dBoP system, the Lagrange formalism is used. The energies are calculated in the plate frame.
The kinetic energy can be calculated by using

$$\begin{aligned} T &= T_{B,trans} + T_{B,rot} + T_{P,rot} \\ &= \frac{1}{2} m_B \cdot_P \mathbf{v}_B^T \cdot_P \mathbf{v_B} + \frac{1}{2} \cdot_P \omega_B^T \cdot \mathbf{I}_B \cdot_P \omega_B + + \frac{1}{2} \cdot_P \omega_P^T \cdot \mathbf{I}_P \cdot_P \omega_P \,, \end{aligned} \tag{2.1}$$

with

$${}_P\mathbf{r}_B = \begin{bmatrix} x \\ y \\ r \end{bmatrix}, \quad {}_P\mathbf{v}_B = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix}, \quad {}_P\omega_B = {}_P\omega_P + {}_P\omega_{B,rel} = \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{-\dot{y}}{r} \\ \frac{\dot{x}}{r} \\ 0 \end{bmatrix} .$$

The inertia of a ball with Radius r and mass m_B can be calculated as

$$\mathbf{I_B} = j_b \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \begin{array}{ll} \text{for a hollow sphere with } d << r & j_b = \frac{2}{3} m_B r^2 \\ \text{for a full sphere} & j_b = \frac{2}{5} m_B r^2 \end{array} .$$

The inertia of a plate with the dimensions $l \times w \times h$ (length x width x height), mass m_P and the inertia of the ball considered as point mass at (x, y, r)[2] can be determined with

$$\mathbf{I_P} = \begin{bmatrix} J_{P,xx} + m_B(x^2 + r^2) & 0 & 0 \\ 0 & J_{P,yy} + m_B(y^2 + r^2) & 0 \\ 0 & 0 & I_{P,zz} + m_B(x^2 + y^2) \end{bmatrix}$$
$$J_{P,xx} = \frac{m_P}{12}(l^2 + h^2), \quad J_{P,yy} = \frac{m_P}{12}(w^2 + h^2), \quad I_{P,zz} = \frac{m_P}{12}(w^2 + l^2) \,.$$

Calculating

$$T_{B,trans} = \frac{1}{2} m_B \cdot (\dot{x}^2 + \dot{y}^2) \,, \tag{2.2}$$
$$T_{B,rot} = \frac{1}{2} j_b \cdot [(\dot{\alpha} - \frac{\dot{y}}{r})^2 + (\dot{\beta} + \frac{\dot{x}}{r})^2] \,, \tag{2.3}$$
$$T_{P,rot} = \frac{1}{2} [\dot{\alpha}^2 (J_{P,xx} + m_B(x^2 + r^2)) + \dot{\beta}^2 (J_{P,yy} + m_B(y^2 + r^2))] \,, \tag{2.4}$$

the kinetic energy T can be obtained

$$\begin{aligned} T = \frac{1}{2} [&\dot{\alpha}^2 (J_{P,xx} + m_B(x^2 + r^2) + j_b) + \dot{\beta}^2 (J_{P,yy} + m_B(y^2 + r^2) + j_b) \\ &+ \dot{x}^2 (\frac{j_b}{r^2} + m_B) + \dot{y}^2 (\frac{j_b}{r^2} + m_B) + 2 \cdot j_b (\frac{\dot{x}\dot{\beta}}{r} - \frac{\dot{y}\dot{\alpha}}{r})] \,. \end{aligned} \tag{2.5}$$

[2] The mas of the ball is included in the calculation of the inertia of the ball with the Parallel Axis theorem.

The potential energy V can be calculated as

$$V = m_B \cdot g \cdot_W r_{B,z} \ , \tag{2.6}$$

with

$$\begin{aligned} {}_W r_{B,z} &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{R}_y \cdot \mathbf{R}_x \cdot_P \mathbf{r}_B \\ &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ r \end{bmatrix} \\ &= r\cos(\alpha)\cos(\beta) - \sin(\beta)x + \cos(\beta)\sin(\alpha)y \ . \end{aligned} \tag{2.7}$$

With the kinetic and potential energy the Lagrange formalism

$$L = T(\mathbf{q},\dot{\mathbf{q}}) - V(\mathbf{q}), \tag{2.8}$$

$$Q_i^* = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i}, \quad i = 1,2,3,4, \quad \mathbf{q} = [x, y, \alpha, \beta]^T, \quad \mathbf{Q} = [0, 0, \tau_x, \tau_y]^T \tag{2.9}$$

$$Q_i^* = \frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} + \frac{\partial V}{\partial q_i} - \frac{\partial T}{\partial q_i}, \tag{2.10}$$

can be applied. The result

$$\begin{aligned} L = \frac{1}{2}[&\dot{\alpha}^2(J_{P,xx} + m_B(x^2 + r^2) + j_b) + \dot{\beta}^2(J_{P,yy} + m_B(y^2 + r^2) + j_b) \\ &+ \dot{x}^2(\frac{j_b}{r^2} + m_B) + \dot{y}^2(\frac{j_b}{r^2} + m_B) + 2 \cdot j_b(\frac{\dot{x}\dot{\beta}}{r} - \frac{\dot{y}\dot{\alpha}}{r})] \\ &- [r\cos(\alpha)\cos(\beta) - \sin(\beta)x + \cos(\beta)\sin(\alpha)y]. \end{aligned}$$

is used to calculate the equations of motion. Therefore the partial derivates are

$$\begin{aligned} &\frac{\partial T}{\partial \dot{x}} = (\frac{j_b}{r^2} + m_B)\dot{x} + \frac{j_b\dot{\beta}}{r}, \quad \rightarrow \frac{d}{dt}\frac{\partial T}{\partial \dot{x}} = (\frac{j_b}{r^2} + m_B)\ddot{x} + \frac{j_b\ddot{\beta}}{r} \\ &\frac{\partial T}{\partial \dot{y}} = (\frac{j_b}{r^2} + m_B)\dot{y} - \frac{j_b\dot{\alpha}}{r}, \quad \rightarrow \frac{d}{dt}\frac{\partial T}{\partial \dot{y}} = (\frac{j_b}{r^2} + m_B)\ddot{y} - \frac{j_b\ddot{\alpha}}{r} \\ &\frac{\partial T}{\partial \dot{\alpha}} = \dot{\alpha}(J_{P,xx} + m_B(x^2 + r^2) + j_b) - j_b\frac{\dot{y}}{r} \\ &\quad \rightarrow \frac{d}{dt}\frac{\partial T}{\partial \dot{\alpha}} = \ddot{\alpha}(J_{P,xx} + m_B(x^2 + r^2) + j_b) + 2\dot{\alpha}m_B\dot{x}x - j_b\frac{\ddot{y}}{r} \\ &\frac{\partial T}{\partial \dot{\beta}} = \dot{\beta}(J_{P,yy} + m_B(y^2 + r^2) + j_b) + j_b\frac{\dot{x}}{r} \\ &\quad \rightarrow \frac{d}{dt}\frac{\partial T}{\partial \dot{\alpha}} = \ddot{\beta}(J_{P,yy} + m_B(y^2 + r^2) + j_b) + 2\dot{\beta}m_B\dot{y}y + j_b\frac{\ddot{x}}{r} \\ &\frac{\partial T}{\partial x} = m_B\dot{\alpha}^2x, \quad \frac{\partial T}{\partial y} = m_B\dot{\beta}^2y, \quad \frac{\partial T}{\partial \alpha} = 0, \quad \frac{\partial T}{\partial \beta} = 0 \\ &\frac{\partial V}{\partial x} = m_Bg(-\sin(\beta)), \quad \frac{\partial V}{\partial y} = m_Bg\cos(\beta)\sin(\alpha) \\ &\frac{\partial V}{\partial \alpha} = m_Bg(-\sin(\alpha)\cos(\beta)r + \cos(\beta)\cos(\alpha)y) \\ &\frac{\partial V}{\partial \beta} = m_Bg(-\sin(\beta)\cos(\alpha)r - \sin(\beta)\sin(\alpha)y - \cos(\beta)x) \end{aligned}$$

are calculated first. Summarizing the above terms leads to the non-linear differential equations of motion

$$
\begin{aligned}
0 &= (\frac{j_b}{r^2} + m_B)\ddot{x} + \frac{j_b\ddot{\beta}}{r} + m_B g(-\sin(\beta)) - m_B\dot{\alpha}^2 x \\
0 &= (\frac{j_b}{r^2} + m_B)\ddot{y} - \frac{j_b\ddot{\alpha}}{r} + m_B g\cos(\beta)\sin(\alpha) - m_B\dot{\beta}^2 y \\
\tau_x &= \ddot{\alpha}(J_{P,xx} + m_B(x^2+r^2) + j_b) + 2\dot{\alpha}m_B\dot{x}x - j_b\frac{\ddot{y}}{r} \\
&\quad + m_B\cdot g\cdot\cos(\beta)\cdot(-\sin(\alpha)r + \cos(\alpha)y) \\
\tau_y &= \ddot{\beta}(J_{P,yy} + m_B(y^2+r^2) + j_b) + 2\dot{\beta}m_B\dot{y}y + j_b\frac{\ddot{x}}{r} \\
&\quad - m_B g(\sin(\beta)\cos(\alpha)r + \sin(\beta)\sin(\alpha)y + \cos(\beta)x)\ .
\end{aligned}
$$

Note that one can see the Coriolis force in the above equation $F_{Coriolis} = m_B\dot{\beta}^2 y$.

2.2.3 Constructing a simulator

In order to simulate the non-linear equations of motion, the system of equations of motions has to be solved for the second derivatives

$$
\begin{bmatrix}
(\frac{j_b}{r^2}+m_B) & 0 & 0 & \frac{j_b}{r} \\
0 & (\frac{j_b}{r^2}+m_B) & -\frac{j_b}{r} & 0 \\
0 & -\frac{j_b}{r} & (J_{P,xx}+m_B(x^2+r^2)+j_b) & 0 \\
\frac{j_b}{r} & 0 & 0 & (J_{P,yy}+m_B(y^2+r^2)+j_b)
\end{bmatrix}
\cdot
\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix}
= \begin{bmatrix}
m_B g\sin(\beta) + m_B\dot{\alpha}^2 x \\
-m_B g\cos(\beta)\sin(\alpha) + m_B\dot{\beta}^2 y \\
\tau_x - 2m_B\dot{x}\dot{\alpha}x - m_B g\cos(\beta)\cdot(-\sin(\alpha)r + \cos(\alpha)y) \\
\tau_y - 2m_B\dot{y}\dot{\beta}y + m_B g(\sin(\beta)\cos(\alpha)r + \sin(\beta)\sin(\alpha)y + \cos(\beta)x)
\end{bmatrix}.
$$

The equations required for simulation can then be calculated by inverting the above system[3]:

$$
\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix} =
\begin{bmatrix}
\frac{r^2\left(m_B x\dot{\alpha}^2 + g m_B\sin(\beta)\right)\left(m_B r^2 + m_B y^2 + J_{Pyy} + j_b\right)}{J_{Pyy}j_b + m_B^2 r^4 + 2j_b m_B r^2 + j_b m_B y^2 + m_B^2 r^2 y^2 + J_{Pyy} m_B r^2} - \frac{j_b r\left(\tau_y + g m_B(x\cos(\beta) + r\cos(\alpha)\sin(\beta) + y\sin(\alpha)\sin(\beta)) - 2\dot{\beta}\dot{y}m_B y\right)}{J_{Pyy}j_b + m_B^2 r^4 + 2j_b m_B r^2 + j_b m_B y^2 + m_B^2 r^2 y^2 + J_{Pyy} m_B r^2} \\
\frac{\left(\dot{\beta}^2 m_B y - g m_B\cos(\beta)\sin(\alpha)\right)\left(J_{Pxx} r^2 + j_b r^2 + m_B r^4 + m_B r^2 x^2\right)}{J_{Pxx}j_b + m_B^2 r^4 + 2j_b m_B r^2 + j_b m_B x^2 + m_B^2 r^2 x^2 + J_{Pxx} m_B r^2} - \frac{j_b r\left(2\dot{\alpha}\dot{x}m_B x - \tau_x + g m_B\cos(\alpha)(y\cos(\alpha) - r\sin(\alpha))\right)}{J_{Pxx}j_b + m_B^2 r^4 + 2j_b m_B r^2 + j_b m_B x^2 + m_B^2 r^2 x^2 + J_{Pxx} m_B r^2} \\
\frac{j_b r\left(\dot{\beta}^2 m_B y - g m_B\cos(\beta)\sin(\alpha)\right)}{J_{Pxx}j_b + m_B^2 r^4 + 2j_b m_B r^2 + j_b m_B x^2 + m_B^2 r^2 x^2 + J_{Pxx} m_B r^2} - \frac{\left(m_B r^2 + j_b\right)\left(2\dot{\alpha}\dot{x}m_B x - \tau_x + g m_B\cos(\alpha)(y\cos(\alpha) - r\sin(\alpha))\right)}{J_{Pxx}j_b + m_B^2 r^4 + 2j_b m_B r^2 + j_b m_B x^2 + m_B^2 r^2 x^2 + J_{Pxx} m_B r^2} \\
\frac{\left(m_B r^2 + j_b\right)\left(\tau_y + g m_B(x\cos(\beta) + r\cos(\alpha)\sin(\beta) + y\sin(\alpha)\sin(\beta)) - 2\dot{\beta}\dot{y}m_B y\right)}{J_{Pyy}j_b + m_B^2 r^4 + 2j_b m_B r^2 + j_b m_B y^2 + m_B^2 r^2 y^2 + J_{Pyy} m_B r^2} - \frac{j_b r\left(m_B x\dot{\alpha}^2 + g m_B\sin(\beta)\right)}{J_{Pyy}j_b + m_B^2 r^4 + 2j_b m_B r^2 + j_b m_B y^2 + m_B^2 r^2 y^2 + J_{Pyy} m_B r^2}
\end{bmatrix}.
$$

2.2.4 Linearisation and torque control

All equation of motion derived in Section 2.2.3 are used to construct the non-linear system

$$
\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{y}(t) = \mathbf{g}(\mathbf{x}) = \begin{bmatrix} x & y \end{bmatrix}^T \tag{2.11}
$$

$$
\mathbf{x} = \begin{bmatrix} x & y & \alpha & \beta & \dot{x} & \dot{y} & \dot{\alpha} & \dot{\beta} \end{bmatrix}^T, \quad \mathbf{u} = \begin{bmatrix} \tau_x & \tau_y \end{bmatrix}^T\ . \tag{2.12}
$$

[3] This is done by Matlab.

This system is linearised around the Equilibrium Point (EP) of (2.13) by utilizing

$$\mathbf{x}(t=0)=\mathbf{x}_0=\mathbf{0}^T,\quad \mathbf{u}(t=0)=\mathbf{u}_0=\mathbf{0}^T \tag{2.13}$$

$$\dot{\mathbf{x}}-\mathbf{f}(\mathbf{x},\,\mathbf{u})|_{\mathbf{EP}}\approx\frac{\partial\mathbf{f}(\mathbf{x},\,\mathbf{u})}{\partial\mathbf{x}}|_{EP}\cdot(\mathbf{x}(t)-\mathbf{x}_0)+\frac{\partial\mathbf{f}(\mathbf{x},\,\mathbf{u})}{\partial\mathbf{u}}|_{EP}\cdot(\mathbf{u}(t)-\mathbf{u}_0) \tag{2.14}$$

$$\Delta\dot{\mathbf{x}}=\mathbf{A}\Delta\mathbf{x}+\mathbf{B}\Delta\mathbf{u}\ . \tag{2.15}$$

The result of the linearisation around the EP of $\mathbf{0}$ is given as

$$A=\begin{bmatrix}
0 & 0 & 0 & 0 & 1&0&0&0\\
0 & 0 & 0 & 0 & 0&1&0&0\\
0 & 0 & 0 & 0 & 0&0&1&0\\
0 & 0 & 0 & 0 & 0&0&0&1\\
-\frac{g\,j_b\,m_B\,r}{kk_2} & 0 & 0 & \frac{g\,m_B\,r^2\left(m_B\,r^2+J_{Pyy}+j_b\right)}{kk_2}-\frac{g\,j_b\,m_B\,r^2}{kk_2} & 0&0&0&0\\
0 & -\frac{g\,j_b\,m_B\,r}{kk_1} & \frac{g\,j_b\,m_B\,r^2}{kk_1}-\frac{g\,m_B\left(J_{Pxx}\,r^2+j_b\,r^2+m_B\,r^4\right)}{kk_1} & 0 & 0&0&0&0\\
0 & -\frac{g\,m_B\left(m_B\,r^2+j_b\right)}{kk_1} & \frac{g\,m_B\,r\left(m_B\,r^2+j_b\right)}{kk_1}-\frac{g\,j_b\,m_B\,r}{kk_1} & 0 & 0&0&0&0\\
\frac{g\,m_B\left(m_B\,r^2+j_b\right)}{kk_2} & 0 & 0 & \frac{g\,m_B\,r\left(m_B\,r^2+j_b\right)}{kk_2}-\frac{g\,j_b\,m_B\,r}{kk_2} & 0&0&0&0
\end{bmatrix},$$

$$kk_1=J_{\mathrm{Pxx}}\,j_b+{m_B}^2\,r^4+2\,j_b\,m_B\,r^2+J_{\mathrm{Pxx}}\,m_B\,r^2,\quad kk_2=J_{\mathrm{Pyy}}\,j_b+{m_B}^2\,r^4+2\,j_b\,m_B\,r^2+J_{\mathrm{Pyy}}\,m_B\,r^2,$$

$$B=\begin{bmatrix}
0 & 0\\
0 & 0\\
0 & 0\\
0 & 0\\
0 & -\frac{j_b\,r}{kk_2}\\
\frac{j_b\,r}{kk_1} & 0\\
\frac{m_B\,r^2+j_b}{kk_1} & 0\\
0 & \frac{m_B\,r^2+j_b}{kk_2}
\end{bmatrix},\quad C=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\end{bmatrix}.$$

With these A, B, C matrices a Linear Time Invariant (LTI) system is obtained.

2.2.5 Linearisation and acceleration control

It is possible to use the last two equations of motion and receptively solve them for $\ddot{x},\ddot{y}$

$$\ddot{x}=-\frac{(\frac{j_b\ddot{\beta}}{r}+m_Bg(-\sin(\beta))-m_B\dot{\alpha}^2x)}{\frac{j_b}{r^2}+m_B}=\frac{\frac{-j_b\ddot{\beta}}{r}+m_Bg\sin(\beta)+m_B\dot{\alpha}^2x}{\frac{j_b}{r^2}+m_B} \tag{2.16}$$

$$\ddot{y}=-\frac{-\frac{j_b\ddot{\alpha}}{r}+m_Bg\cos(\beta)\sin(\alpha)-m_B\dot{\beta}^2y}{\frac{j_b}{r^2}+m_B}=\frac{\frac{j_b\ddot{\alpha}}{r}-m_Bg\cos(\beta)\sin(\alpha)+m_B\dot{\beta}^2y}{\frac{j_b}{r^2}+m_B}\ . \tag{2.17}$$

If the motors controlling the inclination angle of the plate are controlled by $\ddot{\alpha}$ and $\ddot{\beta}$, the linearisation can be done as follows

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad \mathbf{y}(t) = \mathbf{g}(\mathbf{x}) = \begin{bmatrix} x & y \end{bmatrix}^T$$
$$\mathbf{x} = \begin{bmatrix} x & y & \alpha & \beta & \dot{x} & \dot{y} & \dot{\alpha} & \dot{\beta} \end{bmatrix}^T, \quad \mathbf{u} = \begin{bmatrix} \ddot{\alpha} & \ddot{\beta} \end{bmatrix}^T .$$

Linearisation around the EP of $\mathbf{x} = \mathbf{x}_0$, $\mathbf{u} = \mathbf{u}_0$ with

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{\mathrm{x}} & \dot{\mathrm{y}} & \dot{\alpha} & \dot{\beta} & \frac{\dot{\alpha}^2 m_B x + g m_B \sin(\beta) - \frac{\ddot{\beta} j_b}{r}}{m_B + \frac{j_b}{r^2}} & \frac{\dot{\beta}^2 m_B y + \frac{\ddot{\alpha} j_b}{r} - g m_B \cos(\beta)\sin(\alpha)}{m_B + \frac{j_b}{r^2}} & \ddot{\alpha} & \ddot{\beta} \end{bmatrix},$$

results in

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{g m_B}{m_B + \frac{j_b}{r^2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{g m_B}{m_B + \frac{j_b}{r^2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -\frac{j_b}{r\left(m_B + \frac{j_b}{r^2}\right)} \\ \frac{j_b}{r\left(m_B + \frac{j_b}{r^2}\right)} & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

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

With these A, B, C matrices a LTI system is obtained.

2.3 Coupled Ball-on-Plate system

One problem of the dBoP system are its invariant zeros[4], which cannot be moved by the controller. As real non minimum phase zeros (invariant zeros) imply undershoot in the step response of linear systems, the ball of the dBoP will always respond by moving in the opposite direction when applying a step on the system. In order to investigate, if the invariant zeros can be removed by adding more DoF to the system, the cBoP system is modelled as well.
In the coupled seven DoF BoP system the position of the end-effector's is additionally manipulated. Moreover, the rolling friction of the ball is considered. The state vector is
$\mathbf{x} = \begin{bmatrix} X & Y & Z & \alpha & \beta & x & y & \dot{X} & \dot{Y} & \dot{Z} & \dot{\alpha} & \dot{\beta} & \dot{x} & \dot{y} \end{bmatrix}^T$.
To derive a model for this system the Lagrange formalism is used. For the cBoP system the rotational kinetic energy does not has to be transformed to the world frame, but the linear kinetic energy has to be transformed[5].

[4] The invariant zeros of the dBoP are obtained by using the *tzero* function of Matlab.

[5] The rotational kinetic energy is independent of a coordinate frame. For this reason it does not have to be transformed to the world frame.

2.3.1 Assumptions

The assumptions used for the cBoP system consist of the first three assumptions of the dBoP system and the following:

- The ball is assumed to roll all the time. Thus no sticking friction has to be considered.
- The ball is assumed to not slide.

2.3.2 Modelling the friction of the ball

In order to incorporate the friction of the ball into the model, the Lagrange formalism has to be extended. The generalized force vector $\mathbf{Q}^*$ becomes the sum of the conservative forces and the non-conservative forces (friction). The additional non-conservative forces are given by the derivation of the dissipation function P in the following equations

$$L = T(\mathbf{q},\dot{\mathbf{q}}) - V(\mathbf{q}) \tag{2.18}$$

$$Q_i^* = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} + \frac{\partial P}{\partial \dot{q}_i}, \quad i = 1...7, \tag{2.19}$$

$$\mathbf{q} = [X,Y,Z,\alpha,\beta,x,y]^T, \quad \mathbf{Q} = [F_x, F_y, F_z, \tau_x, \tau_y, 0, 0]^T \ . \tag{2.20}$$

To model the friction R the dissipation function P is utilized

$$R_j = -\frac{\partial P}{\partial \dot{q}_j} \ .$$

In general, the friction can be calculated as the integrated sum of a velocity dependent function $h(v)$

$$R_j = -\frac{\partial}{\partial \dot{q}_j} \sum_{i=1}^{N} \int_0^{v_i} h_i(\hat{v}_i) d\,\hat{v}_i \ .$$

Solving the above equation for P gives

$$P = \sum_{i=1}^{N} \int_0^{v_i} h_i(\hat{v}_i) d\,\hat{v}_i \ . \tag{2.21}$$

The rolling friction of an object that is rolling without sliding is given as [6]

$$\mathbf{R}_{roll} = -\mu_R F_N \frac{\mathbf{v}}{|\mathbf{v}|} \ . \tag{2.22}$$

[6] μ_R is the rolling friction coefficient, F_N is the normal force and v is the velocity of the object.

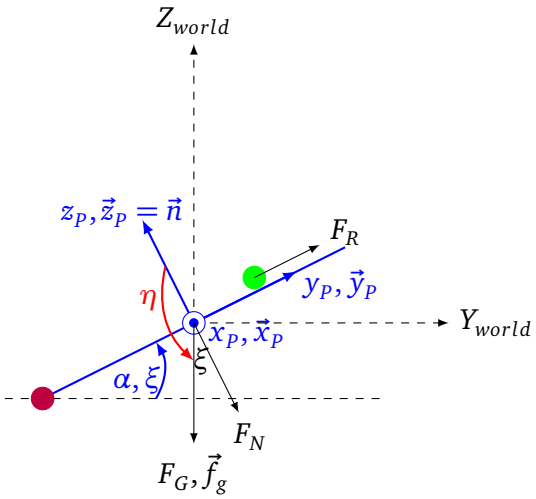


Figure 2.6.: Explanation of the angle ξ

Using 2.22 and 2.21 the dissipation function of a ball on a plate can be derived as [p. 13 -17][5]

$$h_{roll}(v) = \mu_R F_N = \mu_R m_B g \xi$$
$$P_{roll} = \int_0^{v_B} \mu_R m_B g \xi \cdot d\hat{v}_B = \mu_R m_B g \xi \sqrt{\dot{x}_B^2 + \dot{y}_B^2}\,.$$

To calculate the rolling friction, the angle between the normal force F_N and F_G denoted as ξ has to be obtained. For a deeper understanding of this angle, assume that $\alpha > 0$, $\beta > 0$. Then the plate "stands" on one contact point (purple point in Figure 2.6) in the front left corner. The angle between the front left contact point and the standing plate is named ξ. Note that $\alpha = \xi$ if $\beta = 0$ and $\beta = \xi$ if $\alpha = 0$.

With respect to the world frame the vectors

$$\vec{y}_P = \begin{bmatrix} \cos(\alpha) \\ 0 \\ \sin(\alpha) \end{bmatrix}, \quad \vec{x}_P = \begin{bmatrix} 0 \\ \cos(\beta) \\ \sin(\beta) \end{bmatrix}, \quad \vec{n} = \vec{x}_P \times \vec{y}_P = \begin{bmatrix} -\sin(\alpha)\cos(\beta) \\ -\cos(\alpha)\sin(\beta) \\ \cos(\alpha)\cos(\beta) \end{bmatrix}$$
$$\vec{f}_g = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \quad \xi = 180 - \eta, \quad \xi = 180 - \eta = \arccos(-\eta) \tag{2.23}$$
$$\cos(\eta) = \frac{\vec{n} \cdot \vec{f}_g}{|\vec{n}| \cdot |\vec{f}_g|} = \frac{-\cos(\alpha)\cos(\beta)}{\sqrt{\sin^2(\alpha)\cos^2(\beta) + \cos^2(\alpha)}} = \frac{-\cos(\alpha)\cos(\beta)}{\sqrt{1 - \sin^2(\alpha)\sin^2(\beta)}}$$

can be calculated, in order to obtain ξ.

With ξ the total dissipation function P

$$P = P_{roll} \tag{2.24}$$
$$= \sqrt{\dot{x}^2 + \dot{y}^2} \cdot \underbrace{\mu_R m_B g}_{k_R} \cdot \arccos\left(\frac{\cos(\alpha)\cos(\beta)}{\sqrt{1 - \sin^2(\alpha)\sin^2(\beta)}}\right)$$

is given.

2.3.3 Equations of motion

To calculate the equations of motion the Lagrange formalism is used. The kinetic energy of the cBoP system can be derived as

$$\begin{aligned} T &= T_{B,trans} + T_{P,trans} + T_{B,rot} + T_{P,rot} \\ &= \frac{1}{2} m_B \cdot_W \mathbf{v}_B^T \cdot_W \mathbf{v_B} + \frac{1}{2} m_P \cdot_W \mathbf{v}_P^T \cdot_W \mathbf{v_P} + \frac{1}{2} \cdot_P \omega_B^T \cdot \mathbf{I}_B \cdot_P \omega_B + + \frac{1}{2} \cdot_P \omega_P^T \cdot \mathbf{I}_P \cdot_P \omega_P \ , \end{aligned} \quad (2.25)$$

with[7]

$$ {}_W\mathbf{v}_B = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} + \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ 0 \end{bmatrix} \times (\mathbf{R} \cdot \begin{bmatrix} x \\ y \\ r \end{bmatrix}) + \mathbf{R} \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} $$

$$ {}_W\mathbf{v}_B = \begin{bmatrix} \dot{\beta}(r\cos(\alpha)\cos(\beta) - \sin(\beta)x + \cos(\beta)\sin(\alpha)y) + \cos(\beta)\dot{x} + \dot{X} + \sin(\alpha)\sin(\beta)\dot{y} \\ \cos(\alpha)\dot{y} - \dot{\alpha}(r\cos(\alpha)\cos(\beta) - \sin(\beta)x + \cos(\beta)\sin(\alpha)y) + \dot{Y} \\ \dot{Z} - \dot{\beta}(\cos(\beta)x + r\cos(\alpha)\sin(\beta) + \sin(\alpha)\sin(\beta)y) - \sin(\beta)\dot{x} - \dot{\alpha}(r\sin(\alpha) - \cos(\alpha)y) + \cos(\beta)\sin(\alpha)\dot{y} \end{bmatrix} $$

Inserting ${}_W\mathbf{v}_B$ in equation 2.25 gives

$$\begin{aligned} T_{B,trans} = (m_B((\dot{\alpha}(r\sin(\alpha) - \cos(\alpha)y) + \dot{\beta}(\cos(\beta)x + r\cos(\alpha)\sin(\beta) + \sin(\alpha)\sin(\beta)y) + \\ \sin(\beta)\dot{x} - \dot{Z} - \cos(\beta)\sin(\alpha)\dot{y})^2 + (\dot{\beta}(r\cos(\alpha)\cos(\beta) - \sin(\beta)x + \cos(\beta)\sin(\alpha)y) + \\ \cos(\beta)\dot{x} + \dot{X} + \sin(\alpha)\sin(\beta)\dot{y})^2 + (\cos(\alpha)\dot{y} - \dot{\alpha}(r\cos(\alpha)\cos(\beta) - \sin(\beta)x + \\ \cos(\beta)\sin(\alpha)y) + \dot{Y})^2))/2 \end{aligned}$$

The other kinetic energies

$$\begin{aligned} T_{P,trans} &= \frac{1}{2} m_P \cdot (\dot{X}^2 + \dot{Y}^2 + \dot{Z}^2) \\ T_{B,rot} &= \frac{1}{2} j_b \cdot [(\dot{\alpha} - \frac{\dot{y}}{r})^2 + (\dot{\beta} + \frac{\dot{x}}{r})^2] \\ T_{P,rot} &= \frac{1}{2} [\dot{\alpha}^2 (J_{P,xx} + m_B(x^2 + r^2)) + \dot{\beta}^2 (J_{P,yy} + m_B(y^2 + r^2))] \end{aligned}$$

can be calculated more simple as they do not have to be transformed into the world frame. The total kinetic energy is then given as

$$\begin{aligned} T_{ges} = (i_b(\dot{\beta} + \dot{x}/r)^2 + i_b(\dot{\alpha} - \dot{y}/r)^2 + \dot{\alpha}^2(I_{Pxx} + m_B(r^2 + x^2)) + \dot{\beta}^2(I_{Pyy} + . \\ m_B(r^2 + y^2)) + m_B\big(((r\sin(\alpha) - \cos(\alpha)y)\dot{\alpha} + (\cos(\beta)x + r\cos(\alpha)\sin(\beta) + \\ \sin(\alpha)\sin(\beta)y)\dot{\beta} + \sin(\beta)\dot{x} - \dot{Z} - \cos(\beta)\sin(\alpha)\dot{y})^2 + (\dot{\beta}(r\cos(\alpha)\cos(\beta) - \\ \sin(\beta)x + \cos(\beta)\sin(\alpha)y) + \cos(\beta)\dot{x} + \dot{X} + \sin(\alpha)\sin(\beta)\dot{y})^2 + (\cos(\alpha)\dot{y} - \\ \dot{\alpha}(r\cos(\alpha)\cos(\beta) - \sin(\beta)x + \cos(\beta)\sin(\alpha)y) + \dot{Y})^2\big) + m_P\dot{X}^2 + m_P\dot{Y}^2 + \\ m_P\dot{Z}^2)/2 \end{aligned}$$

[7] $\mathbf{R} = \mathbf{R}_y \cdot \mathbf{R}_x$. The $\mathbf{R}_x$ and $\mathbf{R}_y$ matrix is given in equation 2.7.

The potential energy V can be calculated as

$$
\begin{aligned}
V &= m_B \cdot g \cdot_W r_{B,z} + m_P \cdot g \cdot [0,0,Z]^T \\
{}_W r_{B,z} &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} (\cdot \mathbf{R}_y \cdot \mathbf{R}_x \cdot_P \mathbf{r}_B) \\
&= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \left(\begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ r \end{bmatrix} + \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) \\
&= Z - \sin(\beta)x + r\cos(\alpha)\cos(\beta) + \cos(\beta)\sin(\alpha)y \\
\rightarrow V &= (Z - \sin(\beta)x + r\cos(\alpha)\cos(\beta) + \cos(\beta)\sin(\alpha)y)gm_B + m_P Z g \ .
\end{aligned}
$$

Next the Lagrange formalism (see equation 2.18) is used to calculate the equations of motion. The equations of motion of the cBoP system with rolling friction are given in Appendix A. As solving the non-linear equations of motion for the seven second derivatives (given in Appendix A) results in really long terms, it is computationally very expensive to construct a non-linear simulator for the cBoP system.

2.3.4 Linearisation and acceleration control

However, if the task is to only control the x and y position of the ball on the plate, (A.6) and (A.7) can be solved. Solving these equations for $\ddot{x}$ and for $\ddot{y}$ results in (A.8) and (A.9). These two equations of motion are linearised by (2.13) around

$$
\begin{aligned}
\mathbf{x}_0 &= [0\ 0\ 0\ 0.001\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0.0001]^T \\
\mathbf{u}_0 &= [0\ 0\ 0\ 0\ 0]^T \ .
\end{aligned}
$$

In order to prevent a division through zero the cBoP system should be linearised around an EP, which has a non zero ball velocity and a minimal inclination angle. The matrices of the system can subsequently be derived as

$$
A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{g\,m_B}{m_B + \frac{j_b}{r2}} & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{m_B}{10^7\left(m_B + \frac{j_b}{r2}\right)} & -\frac{2\,k_A + 10\,k_R}{m_B + \frac{j_b}{r2}} & 0 \\
0 & 0 & 0 & -\frac{k_R + g\,m_B}{m_B + \frac{j_b}{r2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{2\,k_A}{m_B + \frac{j_b}{r2}}
\end{bmatrix}
$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -\frac{m_B}{m_B+\frac{j_b}{r^2}} & 0 & 0 & 0 & -\frac{m_B\, r+\frac{j_b}{r}}{m_B+\frac{j_b}{r^2}} \\ 0 & -\frac{m_B}{m_B+\frac{j_b}{r^2}} & -\frac{m_B}{10^3\left(m_B+\frac{j_b}{r^2}\right)} & \frac{m_B\, r+\frac{j_b}{r}}{m_B+\frac{j_b}{r^2}} & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

With these matrices the state space representation of the cBoP system is derived.

2.4 Comparison of both models

The main difference between the dBoP and the cBoP system is that the dBoP system has three additional DoF. With these additional DoF the system is much more complex, because it is required to calculate the inverse kinematics[8] to control the system. Additionally, it can be summarized that the observability (Obsv) and the controllability (Ctrb) of both systems depend on the output ($\mathbf{y}$) vector and thereby on the $\mathbf{C}$ matrix. Moreover, the minimal phase (Min), the eigenvalues (Eig) and invariant zeros (Inv zeros) are evaluated in Table 2.1[9].
It is conspicuous that the cBoP system has no invariant zeros for $\mathbf{y} = \begin{bmatrix} x & y \end{bmatrix}^T$. Thus, this system will not undershoot. Moreover, it is expected that the three additional DoF can be used to design a faster and more stable controller. If the control input vector can be weighted, it is also possible to disable the cBoP system to become a dBoP system. For these reasons, the cBoP system is chosen for further analysis.

Sys	$\mathbf{y}$	Ctrb	Obsv	Min	Eig	Inv zeros
cBoP	$\begin{bmatrix} X \\ Y \\ Z \\ x \\ y \end{bmatrix}$	yes	yes	no	$\begin{bmatrix} -0.19696 \\ -0.54732 \\ 0(x12) \end{bmatrix}$	$\begin{bmatrix} -10.4636 \\ -10.4897 \\ 10.4636 \\ 10.4897 \end{bmatrix}$
cBoP	$\begin{bmatrix} x \\ y \end{bmatrix}$	yes	no	yes	$\begin{bmatrix} -0.19696 \\ -0.54732 \\ 0(x12) \end{bmatrix}$	––
dBoP	$\begin{bmatrix} x \\ y \end{bmatrix}$	yes	yes	no	$\begin{bmatrix} 0(x8) \end{bmatrix}$	$\begin{bmatrix} -19.5756(x2) \\ 19.5756(x2) \end{bmatrix}$

Table 2.1.: Comparison of the coupled and decoupled BoP system

[8] The inclination angles of the dBoP system can directly be controlled by two motors.

[9] The concrete eigenvalues and invariant zeros given in Table 2.1 are calculated with the linearised A, B matrices for the pool ball and the wooden plate (as parameter values). The code for these calculations is given in the *cBoP.m* script (see Appendix E).

3 Ball tracking

In order to control the ball on the plate it is necessary to quickly, precisely and smoothly measure the position and velocity of the ball. Ball Tracking can be done by analysing different sensor data. One approach is doing RGB Image Processing. Another one is tracking the point cloud of the ball. A third one is calculating the position of the ball by using force and torque sensors at the end-effector of the LWA. The advantages and disadvantages of these methods are discussed in the following sections.

3.1 Ball tracking with RGB images

In order to measure the position and velocity of balls by analysing RGB images, an algorithm is invented. This algorithm uses OpenCV to detect circles within a defined HSV color range. The center and radius for each of these circles is determined in image coordinates. Using the radius, the center position and the projection matrix of the Realsense Camera R200 the position of the ball in 3D coordinates relative to the camera can be calculated. This position can be mapped to the frame of the plate by using Aruco markers [1] on the plate. The velocity of each ball is estimated by using a Kalman Filter (KF). This algorithm is explained in more detail in Figure 3.1. The algorithm is implemented within a ROS Indigo environment. Figure 3.3 shows the benefit of using a KF compared to raw measurements. For this experiment the ball was first placed on the origin of the plate and then accelerated in y-direction on a rail. This experimental setup is illustrated in Figure 3.2.

The maximum camera input stream of the Realsense R200 camera for RGB images is 60 Hz (at a resolution of 640x480). For observing one ball an output frequency of 58Hz can be reached. Moreover a static positioning Standard Deviation (STD)[2] of ≤ 1 mm is reached (see Figure 3.3). The downside of this algorithm is its need of the true radius to detect the position of the ball. Moreover, the performance depends on lightning conditions. In order to compensate high noise, filtering is required. However, this results in a time delay which should be prevented. For this reason, no additional filters are used.

3.2 Ball tracking with point clouds

The Realsense R200 outputs a point cloud stream at 30 Hz. Analysing these point clouds the radius and position of the ball in the camera frame can be determined. Using aruco markers

[1] To calculate the transformation matrix of the Aruco markers to the camera, an extended and improved version of the Robot Operating System (ROS) package *aruco_eye* is used.

[2] The STD was calculated with respect to the sample STD.

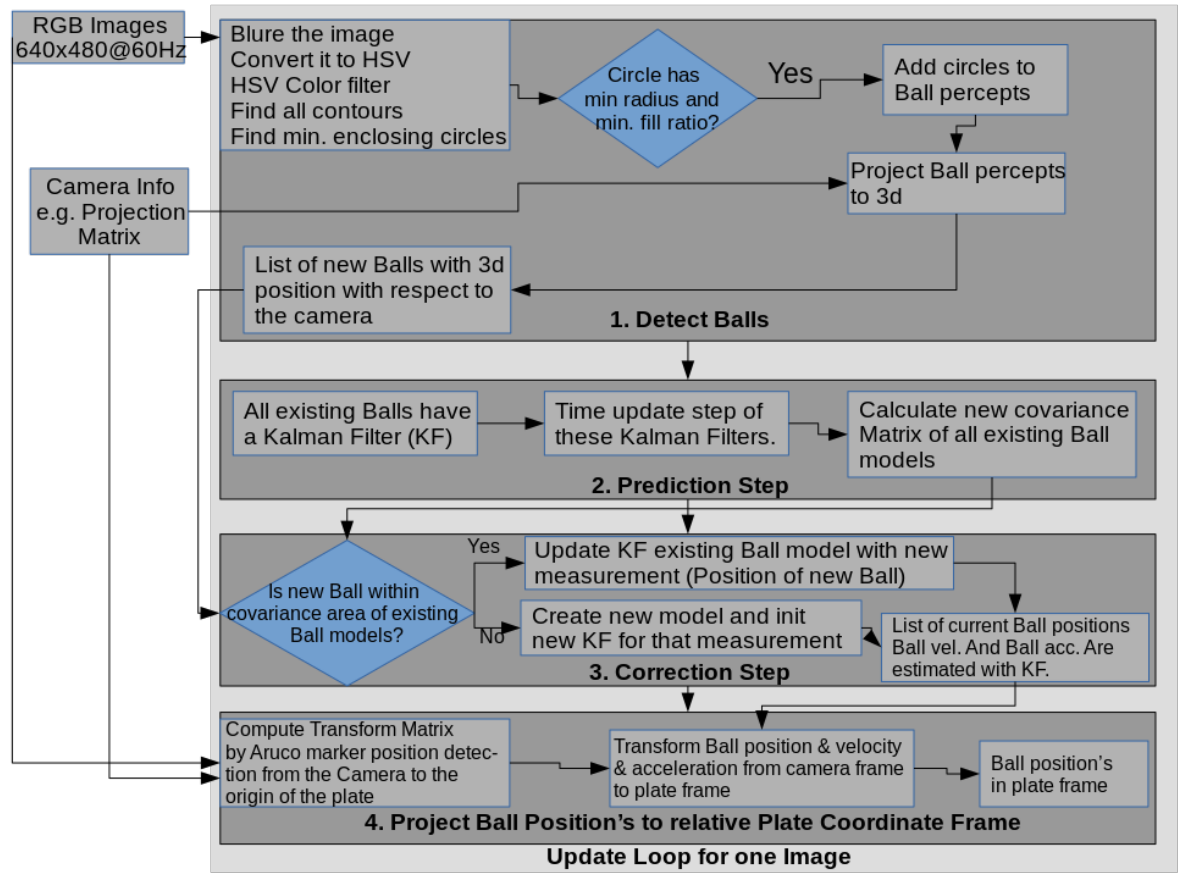


Figure 3.1.: Multiple RGB image-based Ball Tracking algorithm

Figure 3.2.: Visualization of BT algorithm and aruco marker detection

on the plate, this position can be transformed to plate coordinates. With an additional KF the velocity of the ball can be estimated as well. The invented algorithm, that uses a particle filter

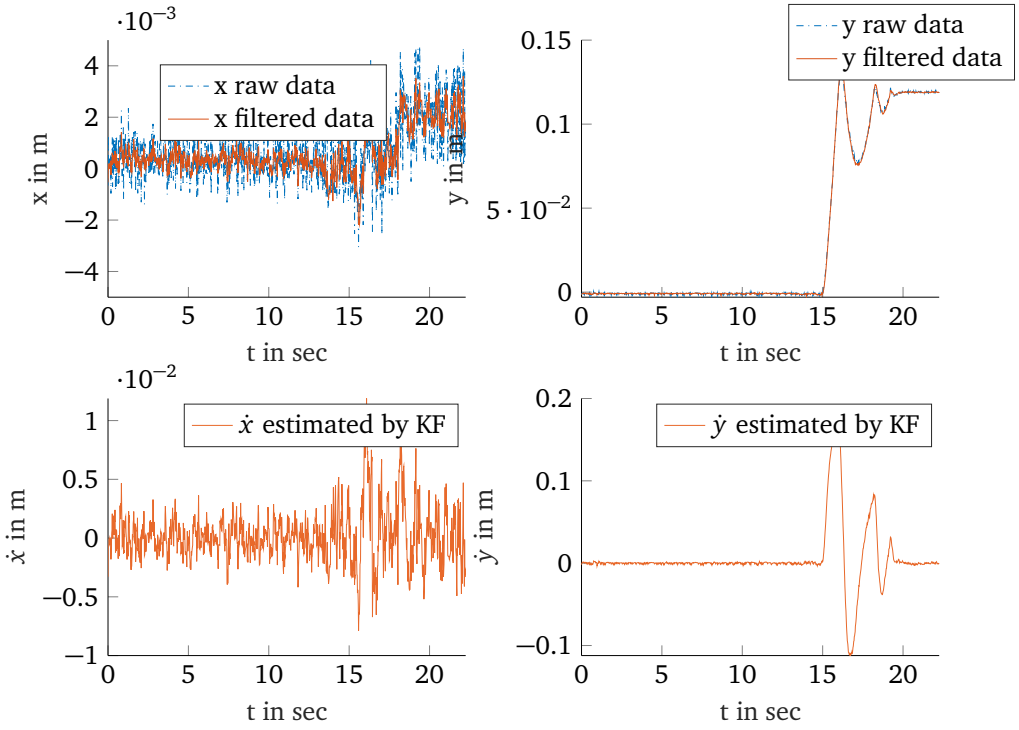


Figure 3.3.: Comparison of the raw and the Kalman filtered position of the ball. The velocity is estimated by the Kalman Filter.

and the Point Cloud Library (PCL)[3], reaches an output position frequency of 25 Hz. Moreover, the algorithm sometimes lost the position of the ball. In such cases it is required to reinitialize it. However, one advantage of this algorithm is that it does not depend on lightning conditions.

3.3 Ball tracking by torque measurements

The used setup (see Figure 2.1) includes a Force Torque Sensor (FTS) by Schunk that is mounted to the end-effector of the LWA. Using these torques (M_x, M_y) and the mass of the ball (m), the position of a single ball lying on the plate can be determined [6]

$$\begin{aligned} x &= \frac{M_y}{m(g\cos(\alpha) - \ddot{X}\sin(\alpha))} \\ y &= \frac{M_x}{m(g\cos(\beta) - \ddot{Y}\sin(\beta))} \end{aligned} . \tag{3.1}$$

The advantage of this method is that no camera is required. Moving the ball by hand over the plate, the performance of this method was convincing (output stream of 500 Hz, static position STD of five millimeter). In (3.1) no friction force and torque is assumed. Thus when using this method for different rolling friction coefficients of the ball, a friction model has to be included in (3.1). Additionally, it was observed that (3.1) does not consider dynamic changes of the

[3] This algorithm is based on *PCL Tracking object in real time*.

tilting angles. In an experiment α was tilted between $\pm 5°$ in a sinusoidal way and the position of the ball was tracked by the using the BT algorithm and (3.1). With respect to Figure 3.4 one can see that the BT algorithm provides a more accurate solution as the ball was rolling from one side of the plate $y = 22.5\,cm$ to the other side[4]. Instead the solution obtained using the torque measurements was much slower and inaccurate. For this reason this method was not further

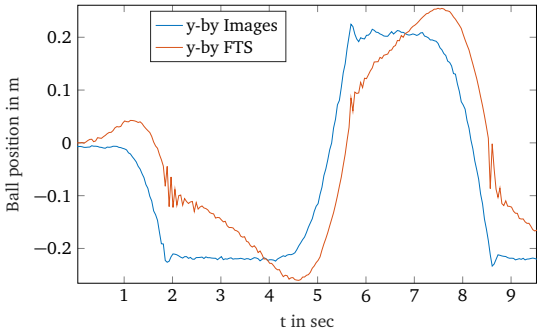


Figure 3.4.: Comparison of the BT algorithm with the method of (3.1) that utilizes torque measurements

investigated. The velocity of the ball can be calculated by derivating the position. As this results in noisy data a KF should be used [6].

3.4 Summary

Table 3.1 compares the performance of the approaches above. The image-based BT algorithm was extensively tested and the most precise one. Thus it was used to obtain the measurements of the position and velocity of the ball for the experiments on the real robot.

Measurements	Hardware	Frequency	static STD	Other
RGB images	Realsense R200	58 Hz	0.9 mm	Radius required.
Point clouds	Realsense R200	25 Hz	5 mm	Information of the radius is not necessary.
Torques	FTS by Schunk	500	5 mm	No camera construction required. Mass required.

Table 3.1.: Different methods of Ball Tracking

[4] In this experiment the plate was surrounded by a fence which prohibits the ball to fall of the plate.

4 Robust controller design

This chapter describes general aspects concerning robust control (in Section 4.1). Moreover, the following (robust) controllers for the uncertain (see Section 4.2) cBoP system with respect to Actuator Saturation Limits (ASLs) are described:

1. As a baseline a LQR controller is designed in Section 4.6.
2. A robust Multi-Model Pole Placement (MMPP) controller is designed in Section 4.7.
3. The benefit of a $\mathscr{H}_2$ controller is discussed in Section 4.8.
4. A fixed-structure $\mathscr{H}_2$ robust controller is designed in Section 4.9.

The objective of the design of the above controllers is to stabilize the uncertain cBoP and fullfill a minimum performance (as defined in Section 4.4). To apply the designed controller the control circuit described in Section 4.5 is used. The controller design is based on the linearised cBoP system of Section 2.3.4.

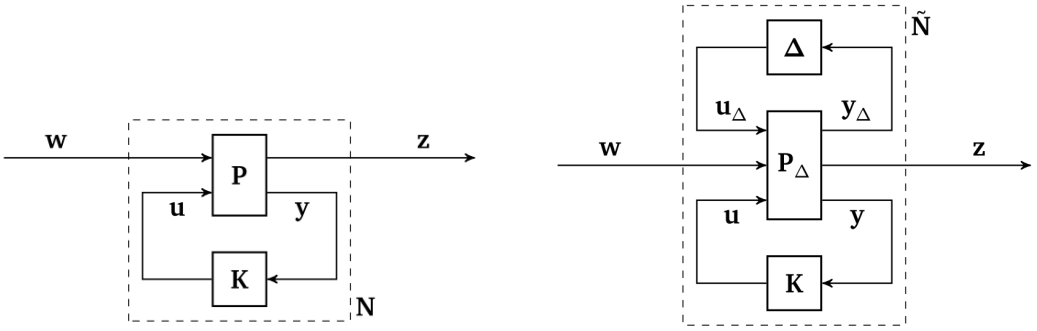
4.1 What is Robust Control

The objective of robust control is to guarantee stability and a specified minimum performance, if a system is working within predefined uncertainty margins. The task of robust control is to find the controller **K** for the control circuit in Figure 4.1 (b) to achieve Robust Performance (RP). In order to explain RP other terms are explained in the following.

4.1.1 Generalized plant

Robust control is usually done for a generalized plant **P**. It contains the plant and additional weights which are used for the controller design. The controller **K** is connected between the measurement signals (control outputs) **y** and the control signals (control inputs) **u** of the generalized plant.

To design a robust controller the system performance is optimized from the performance inputs **w** to the performance outputs **z** of the generalized nominal plant **P**. Thereby **w** is a vector that includes reference signals, noises and disturbances, **z** is a vector that includes all controlled signals and tracking errors and the nominal plant represents the system with no uncertainties. The control input vector and the control output vector of the generalized plant is denoted as **u** and **y**. Figure 4.1 (a) shows the regulated generalized plant **N**. If the plant has uncertainties



(a) Regulated generalized plant **(b)** Approach for design of robust controllers

Figure 4.1.: Generalized plant of the nominal system (a) and of a system with uncertainties (b). This framework is used for the design of robust controllers.

another framework for robust control depicted in Figure 4.1 (b) is utilized. The set of all possible uncertainties[1] is denoted as Δ and $\tilde{\mathbf{N}}$ is the transfer function from $\mathbf{w}$ to $\mathbf{z}$ [8, p. 2].

A state space representation can be transformed to a generalized plant by partitioning the input($\mathbf{B}$), output($\mathbf{C}$) and feedthrough matrix ($\mathbf{D}$)

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ \mathbf{u}\end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix}\mathbf{z}\\ \mathbf{y}\end{bmatrix} = \begin{bmatrix}\mathbf{C}_1\\ \mathbf{C}_2\end{bmatrix}\mathbf{x} + \begin{bmatrix}\mathbf{D}_{11} & \mathbf{D}_{12}\\ \mathbf{D}_{21} & \mathbf{D}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ \mathbf{u}\end{bmatrix}. \tag{4.2}$$

The generalized plant can also be represented as transfer function

$$\mathbf{P}(s) = \begin{bmatrix}\mathbf{P}_{11}(s) & \mathbf{P}_{12}(s)\\ \mathbf{P}_{21}(s) & \mathbf{P}_{22}(s)\end{bmatrix} = \left[\begin{array}{c|cc}\mathbf{A} & \mathbf{B}_1 & \mathbf{B}_2\\ \hline \mathbf{C}_1 & \mathbf{D}_{11} & \mathbf{D}_{21}\\ \mathbf{C}_2 & \mathbf{D}_{21} & \mathbf{D}_{22}\end{array}\right].$$

The transfer function of a generalized plant is given by

$$\mathbf{P}(s) = \mathbf{P}_{21}(\mathbf{s}\mathbf{I} - \mathbf{P}_{11})^{-1}\mathbf{P}_{12} + \mathbf{P}_{22}\,. \tag{4.3}$$

Figure 4.2 shows the block diagram of the generalized plant.

[1] For just parametric uncertainties Δ is a diagonal matrix. Each diagonal entry contains a parametric uncertainty. However Δ can also be used to describe more complex uncertainties as dynamic and parametric, structured and unstructured. For more details about describing the uncertainty of a system see [7, p. 260 - 280].

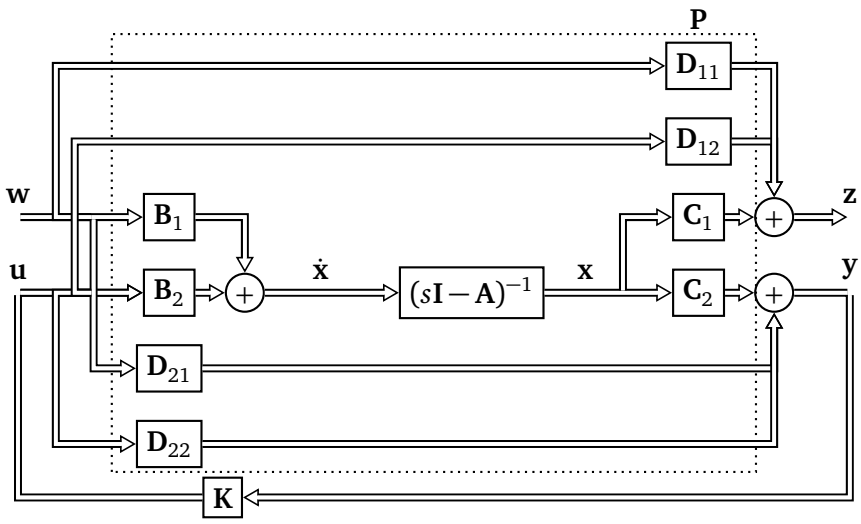


Figure 4.2.: Block diagram of generalized plant.

4.1.2 Linear Fractional Transformation

In order to compute **N** in Figure 4.1(a) the concept of Linear Fractional Transformation (LFT) is applied [2]. Using

$$\mathbf{z} = \mathbf{P}_{11}\mathbf{w} + \mathbf{P}_{12}\mathbf{u}$$
$$\mathbf{y} = \mathbf{P}_{21}\mathbf{w} + \mathbf{P}_{22}\mathbf{u}$$
$$\mathbf{u} = \mathbf{K}\mathbf{y}\ ,$$

N can be calculated

$$\mathbf{N} = \mathbf{P}_{11} + \mathbf{P}_{12}\mathbf{K}\cdot(\mathbf{I} - \mathbf{P}_{22}\mathbf{K})^{-1}\cdot\mathbf{P}_{21}\ . \tag{4.4}$$

N can also be written as lower LFT of **P** and **K**:

$$\mathbf{N} = F_l(\mathbf{P},\ \mathbf{K})\ .$$

Note that the upper linear LFT ($\mathbf{F}_u(\cdot,\cdot)$) is required if the uncertainty Δ is incooperated in the generalized plant [3, p. 247 - 253].

4.1.3 Well-posedness

A closed loop control circuit is well-posed if all transfer matrices of the closed loop control circuit are well-defined and proper [8, p. 66 - 68]. A transfer function is *proper* if the degree of the numerator is less or equal the degree of the denominator.

[2] Note that in Figure 4.1 **P** and **K** are state space systems. Thus, **K** is not represented as constant controller matrix (as e.g. in an LQR controller design).

4.1.4 Internal Stability

Figure 4.3 shows a block diagram that is used to explain internal stability.

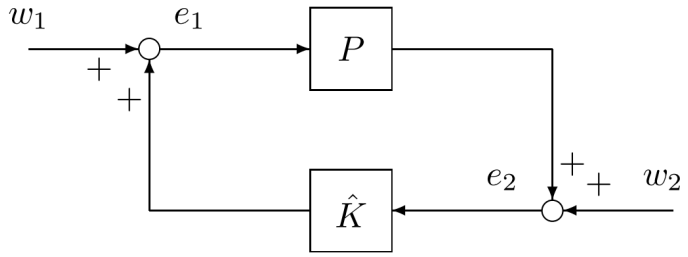


Figure 4.3.: Internal stability analysis diagram

Assume that $\mathbf{P}$ and $\hat{\mathbf{K}}$ are transfer functions that can be represented as state space matrices

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right] \qquad \hat{\mathbf{K}} = \left[\begin{array}{c|c} \mathbf{A}_K & \mathbf{B}_K \\ \hline \mathbf{C}_K & \mathbf{D}_K \end{array}\right].$$

Thereby the matrices with subscript K denote the state space matrices that are summarized in the transfer function $\hat{\mathbf{K}}$.

The system is proper if $\begin{bmatrix} \mathbf{I} & -\mathbf{D}_K \\ -\mathbf{D} & \mathbf{I} \end{bmatrix}^{-1}$ exists [8, p. 68].

The system of Figure 4.3 is said to be *internally stable* if the feedback is *well-posed* and the transfer matrix

$$\begin{bmatrix} \mathbf{I} & -\hat{\mathbf{K}} \\ -\mathbf{P} & \mathbf{I} \end{bmatrix}^{-1} \tag{4.5}$$

from $(w_1,\ w_2)$ to $(e_1,\ e_2)$ belongs to $\mathscr{RH}_\infty$[3]. The second argument can simply be checked by proving that these four transfer functions

$$(\mathbf{I}-\hat{\mathbf{K}}\mathbf{P})^{-1}, \quad \hat{\mathbf{K}}\cdot(\mathbf{I}-\mathbf{P}\hat{\mathbf{K}})^{-1}, \quad \mathbf{P}\cdot(\mathbf{I}-\hat{\mathbf{K}}\mathbf{P})^{-1} \quad \text{and} \quad (\mathbf{I}-\mathbf{P}\hat{\mathbf{K}})^{-1}$$

are stable. This can be summarized to: A control circuit is *internal stable* if all transfer functions of the closed loop control circuit are stable [8, p. 68 - 71] and [3, p. 121].

4.1.5 Robust Stability (RS) for MIMO systems

Consider the uncertain $\mathbf{N}\Delta$-system in Figure 4.4 [4] for which the transfer function from $\mathbf{w}$ to $\mathbf{z}$ is given by the upper LFT

$$\tilde{\mathbf{N}} = F_u(\mathbf{N},\ \Delta) = \mathbf{N}_{22} + \mathbf{N}_{21}\Delta(\mathbf{I}-\mathbf{N}_{11}\Delta)^{-1}\mathbf{N}_{21}\ . \tag{4.6}$$

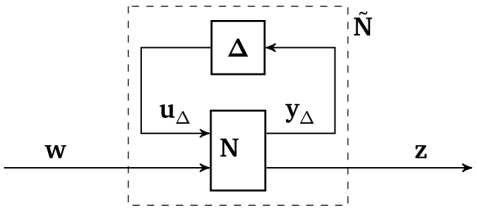


Figure 4.4.: $\mathbf{N}\Delta$-structure for robust stability and robust performance analysis

First assume $\Delta = 0$ and all transfer functions of $\mathbf{N}$ are stable. Then the system achieves Nominal Stability (NS). This means that a system has NS if $\mathbf{N}$ is *internally stable*. For Robust Stability (RS) suppose that Δ and all transfer functions of $\mathbf{N}$ are stable. With these assumptions the only possible source of instability in (4.6) is the feedback term $(\mathbf{I} - \mathbf{N}_{11}\Delta)^{-1}$. Thus, the stability of the system in Figure 4.4 is equivalent to the stability in Figure 4.5.

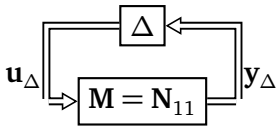


Figure 4.5.: $\mathbf{M}\Delta$-structure for robust stability analysis

To check RS the so called $\mathbf{M}\Delta$-structure (with $\mathbf{M} = \mathbf{N}_{11}$) depicted in Figure 4.5 is used. RS can now be formulated depending on the kind(structured, unstructured, complex, real) of the uncertainty in different ways [7, p. 301 - 316]. One simple representation of RS is given by the *Determinant stability condition* for real or complex perturbations [5] [7, p. 301]:

Assume the nominal system M(s) and the perturbations $\Delta(s)$ are stable. Consider the convex set of perturbations Δ, such that if Δ' is an allowed perturbation then so is $c\Delta$ where c is any real scalar such that $|c| \leq 1$. Then the $M\Delta$-system in Figure 4.5 is stable for all allowed perturbations (we have RS) if and only if

Nyquist plot of $det(\mathbf{I} - \mathbf{M}\Delta(s))$ does not encircle the origin, $\forall\Delta$

$$\Longleftrightarrow det(\mathbf{I} - \mathbf{M}\Delta(j\omega)) \neq 0, \forall\omega, \forall\Delta$$

$$\Longleftrightarrow \lambda_i(M\Delta) \neq 1, \forall i, \forall\omega, \forall\Delta.$$

[3] $\mathscr{RH}_\infty$ describes the set of stable, proper and rational transfer functions.

[4] This control diagram can be obtained by calculating the lower LFT of $\mathbf{P}_\Delta$ and $\mathbf{K}$ in Figure 4.1 (b): $\mathbf{N} = F_l(\mathbf{P}_\Delta, \mathbf{K})$

[5] In the following, the equations $\lambda_i(M\Delta)$ represent the eigenvalues of $(M\Delta)$.

4.1.6 Robust Performance for MIMO systems

Rearrange the uncertain system into the $\mathbf{N}\Delta$-structure of Figure 4.6. Assume Nominal Stability (NS) so that $\mathbf{N}$ is internally stable. Then

$$\begin{aligned} \text{RP} &\Longleftrightarrow \|\tilde{\mathbf{N}}\|_\infty < 1,\ \forall \|\Delta\|_\infty \leq 1 \\ &\Longleftrightarrow \mu_{\hat{\Delta}}(\mathbf{N}(j\omega)) < 1,\ \forall \omega \end{aligned} \tag{4.7}$$

holds. In (4.7) μ is the Structured Singular Value (SSV)[6] computed with respect to the general uncertainty structure

$$\hat{\Delta} = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta_P \end{bmatrix}.$$

Δ_P is a full complex perturbation with the same dimensions as $\tilde{\mathbf{N}}^T$ [7, p.316].

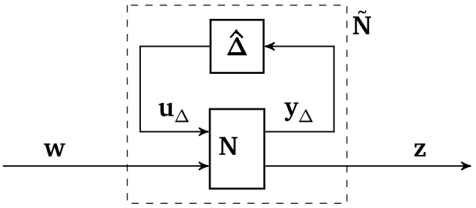


Figure 4.6.: $\mathbf{N}\hat{\Delta}$-structure for robust performance analysis

4.1.7 Summary

The above terms can be summarized with [7, p.300]

$$\begin{aligned} &\text{NS} \Longleftrightarrow \mathbf{N} \text{ is internally stable} \\ &\text{NP} \Longleftrightarrow \|\mathbf{N}_{22}\|_\infty < 1;\ \text{ and NS} \\ &\text{RS} \Longleftrightarrow \tilde{\mathbf{N}} = F_u(\mathbf{N},\ \Delta) \text{ is stable } \forall \Delta, \|\Delta\|_\infty \leq 1;\ \text{and NS} \\ &\text{RP} \Longleftrightarrow \|\tilde{\mathbf{N}}\|_\infty < 1, \quad \forall \Delta,\ \|\Delta\|_\infty <= 1; \quad \text{and NS}\ . \end{aligned}$$

Thereby NP is the nominal performance and $\|\cdot\|_\infty$ is the $\mathscr{H}_\infty$norm. These terms can also be formulated as μ conditions [7, p. 319].

[6] For simplicity a definition of the SSV is not given in this thesis. More about the SSV can be read in [8, p. 187-200].

4.2 Uncertain Ball-on-Plate System

In order to benchmark robust control on the cBoP system, the objective is to find a static and stable controller with a minimum performance for a ball with uncertain parameters. If ideal rolling friction is assumed, it is important to note that the mass of the ball does not impact the system and thereby the performance of the controller. This is due to the fact, that different masses are accelerated by the same gravitation. In case of non-ideal rolling friction it was observed, that varying the radius of the ball impacts the performance of the system. Moreover, changing the rolling friction coefficient (μ_r) influences the system with and without ideal assumptions.

The uncertain cBoP system that is used for robust control is described by the following two parametric uncertainties

$$\begin{aligned} r &\in \left[20.0 \cdot 10^{-3}\text{m} \quad 80 \cdot 10^{-3}\text{m}\right] \\ \mu_r &\in \left[0.0295 \quad 0.1\right] . \end{aligned} \tag{4.8}$$

The uncertainty of the radius varies from a table tennis ball to a hand ball. The rolling friction coefficient varies from the rolling friction of wood on metal to a rolling friction coefficient of ($\mu_r = 0.1$) which corresponds to the friction of car wheels on solid sand. Ideal rolling friction slows the ball down with a constant force $F_R = F_N \mu_r$. The ball starts to accelerate if $F > F_R$. In the real world it is assumed that there is a drop between the friction of the ball with zero velocity($\|v = 0\|$) denoted as static friction and the friction of the ball with ($\|v > 0\|$). These non-linear effects can be prevented by a higher upper rolling friction coefficient. On purpose the upper bound of the rolling friction is defined quite high, as this allows to overcome non-linear effects between static and rolling friction. In particular a static friction coefficient of $\mu_H = 0.1$ corresponds to the stiction of wood on metal[7]. Figure 4.7 illustrates the relation between the force and the friction of a ball.

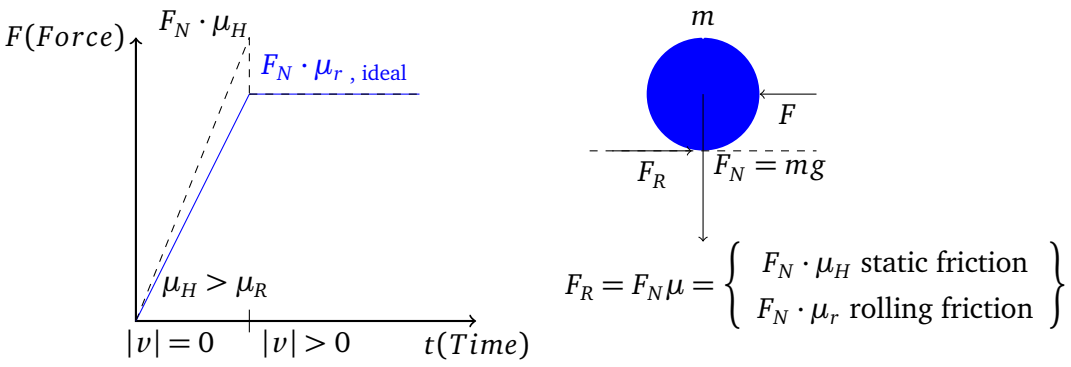


Figure 4.7.: Static friction and rolling friction of the ball

The balancing behaviour of the ball with an initial position of $x_0 = -0.1$ $y_0 = 0.1$ is evaluated by the designed controllers (see Section 4.6 and 4.9). The response of the closed-loop controllers

[7] For more details see *Engineer's handbook* .

are plotted for the pool ball and the wooden plate (see Appendix B). Each plot shows the four corner models denoted as (a) - (d) (see Figure 4.8). Moreover, the nominal system (denoted as (nom)) which represents the true values of the pool ball is plotted. A wooden plate is assumed for the nominal system. Thus, a low rolling friction coefficient of $\mu_r = 0.00295$ is assumed for the nominal system.

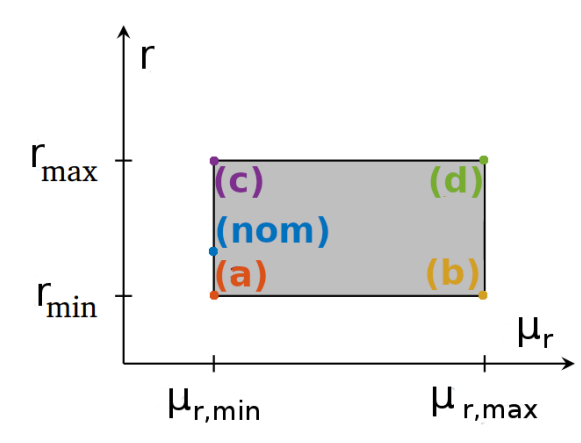


Figure 4.8.: Region of uncertainties

4.3 Actuator Saturation Limits of the BoP system

In order to design a robust controller for a real system it is important to consider the limits of the actuator saturation of the motors of this system. Our system is a seven DoF Schunk arm. It has seven motors (joints denoted as $\mathbf{q}$) which have saturation limits. The limits of the Schunk arm are given in Table 4.1. Note that the LWA consists of three different Universal Rotary Actuators PRL sizes from 120 to 80. For more details read the *datasheet*. The output of the designed

Joint	PRL	Peak torque[Nm]	max vel. $[°/s]$	max acc. $[°/s^2]$	angle range[°]
1	120	372	25	100	$-180, 180$
2	120	372	25	100	$-270, 90$
3	100	176	24	95	$-115, 115$
4	100	176	24	95	$-270, -90$
5	80	41.4	25	100	$-110, 115$
6	80	41.4	25	100	$-360, 0$
7	80	41.4	25	100	$-120, 120$

Table 4.1.: Joint limits of the Schunk LWA

controller however is given in Task Space and not in Joint Space. This means the controller computes a desired control signal $(\ddot{X},\ \ddot{Y},\ \ddot{Z},\ \ddot{\alpha},\ \ddot{\beta})$ for the pose of the end-effector of the LWA. The joint limits can be calculated in task space by using

$$\dot{\mathbf{x}}_{P,max} = \mathbf{J} \cdot \dot{\mathbf{q}}_{max}$$

and

$$\ddot{\mathbf{x}}_{P,max} = \dot{\mathbf{J}} \cdot \dot{\mathbf{q}}_{max} + \mathbf{J} \cdot \ddot{\mathbf{q}}_{max}$$

with

$$\mathbf{x}_P = \begin{bmatrix} X & Y & Z & \alpha & \beta \end{bmatrix}^T, \mathbf{J} = \frac{\partial \mathbf{x}_P}{\partial \mathbf{q}} .$$

The so determined task space ASLs depend on the current joint configuration of the LWA and the time. Thus, no constant ASLs can be specified. Designing a controller having a constant ASL would be desirable. For this reason the following limits are defined in task space:

- $\ddot{X} \leq 0.25\,\text{m/sec}^2$
- $\ddot{Y} \leq 0.25\ \text{m/sec}^2$
- $\ddot{Z} \leq 0.25\,\text{m/sec}^2$
- $\ddot{\alpha} \leq 100\,^\circ/\text{sec}^2$
- $\ddot{\beta} \leq 100\,^\circ/\text{sec}^2$.

In Chapter 6 it is verified, that if the above task space ASLs are not exceeded, the joint space limits are not exceeded, too. During the design of the controllers (see Section 4.6 to 4.9) the above task space ASLs are considered.

4.4 Required performance of the controllers

Testing Robust Performance (RP) of a control circuit is difficult. For this reason a custom term of Step Response Robust Performance (SRRP) is defined. SRRP has a closed-loop system if:

- it achieves Robust Stability as defined in Section 4.1,
- it fullfills the Step Response Requirements (SRR) (as defined below) for all models of the uncertain cBoP system and
- it does not exceed the ASL of the task-space ($\ddot{X}$, $\ddot{Y}$, $\ddot{Z}$, $\ddot{\alpha}$, $\ddot{\beta}$) as defined in Section 4.3 for the analysis of the model and the ASL of the joint-space for the analysis of the Rcs simulation.

The Step Response Requirements (SRR) are given as:

- Max. overshoot of o$\leq$ 60 %.
- Max. undershoot of u$\leq$ 10 %.
- The Ball reaches a $\pm$5 cm band around the desired value within 20 s
- Max. steady state error of sse $\leq$ 0.05m.

4.5 Control circuit

The control circuit is depicted in Figure 4.9. This picture shows, that in order to control the BoP system, a desired plate pose $\mathbf{x_{P,des}}$ is obtained. To reach this pose desired joint angles $\mathbf{q_{des}}$ of the Schunk arm have to be calculated. These angles are determined by Inverse Kinematics (IK) and then applied to the motors of the Schunk arm. If the BoP System is simulated, the position $\mathbf{x}_B = \begin{bmatrix} x & y \end{bmatrix}^T$ and velocity $\dot{\mathbf{x}}_B$ of the ball can be measured. If the real robot is used, the position and velocity of the ball is determined with a ball tracking algorithm or with the FTS (see Chapter 3). The position of the plate $\mathbf{x}_P = \begin{bmatrix} X & Y & Z & \alpha & \beta \end{bmatrix}^T$ and velocity $\dot{\mathbf{x}}_P$ can be calculated with Forward Kinematics (FK). With the ball and the state of the plate a measurement vector $\mathbf{x_{meas}} = \begin{bmatrix} \mathbf{x_P} & \mathbf{x_B} & \dot{\mathbf{x}}_P & \dot{\mathbf{x}}_B \end{bmatrix}^T$ is constructed. This measurement vector is then multiplied by a feed back constant control matrix $\mathbf{K}$. The control input vector $\mathbf{u}$ that is applied to the plant consists of the difference of the reference input vector $\mathbf{w}$ multiplied with the feed forward matrix $\mathbf{F}$ and the feed back term. $\mathbf{K}$ is constructed by using different control design methods explained in the next Sections. $\mathbf{F}$ is designed so that there is no steady state error.

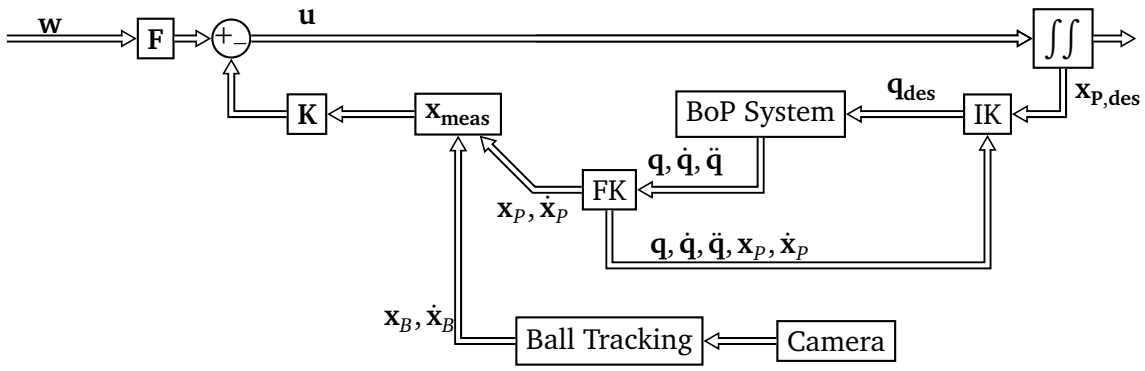


Figure 4.9.: Control circuit to apply the controllers to the real robot (In the Rcs simulation the position and velocity of the ball is directly obtained by the simulation.).

4.6 Design of LQR controller

In a first step a Linear Quadratic Regulator (LQR) controller is designed as a baseline to control the cBoP system. Given the continuous-time linear cBoP system (see Section 2.3.4)

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \ ,$$

with a quadratic cost function

$$J(\mathbf{x}, \mathbf{u}) = \int_0^\infty \mathbf{x}(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)\mathbf{R}\mathbf{u}(t)dt \ , \tag{4.9}$$

the feedback control law that minimizes the value of the cost can be obtained by

$$\mathbf{u} = -\mathbf{K} \cdot \mathbf{x} + \mathbf{F} \cdot \mathbf{w} \ . \tag{4.10}$$

The feed back matrix **K** can be obtained by solving the continuous time Riccati differential equation [9, p. 358]. Therefore the semi-positive Matrix **Q**, that weights the states and the positive Matrix **R** that weights the actuating variables have to be constructed first.

Calculating the IK for big variations of the position of the end-effector compared to the inclination angles of the end-effector is more computational expensive[8]. For this reason it is desired, that the end-effector does not move more than ±20 cm (in X, Y, Z direction) away from its initial position. This aspect can be considered in the design of the LQR controller by punishing the states of the position of the end-effector more than inclination angles in the **R** matrix. In order to have a fast step response the position of the ball is weighted much more compared to the other states in the **Q** matrix. When designing the system it should be considered, that high dynamic parts (such as much overshoot) should be prevented, because these are amplified when the controller is applied to the non-linear real BoP system [p. 16][10]. Thus, the states of the position of the ball should not be weighted to high. These considerations result in the following **Q** and **R** matrix

$$\begin{aligned} \mathbf{Q} &= \mathrm{diag}(1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 50\ \ 50\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1) \\ \mathbf{R} &= \mathrm{diag}(50\ \ 50\ \ 50\ \ 0.2\ \ 0.2)\ . \end{aligned} \tag{4.11}$$

Using the parameters for the pool ball and the wooden plate, as defined in Appendix B, the feed back matrix **K** is calculated[9]. With respect to [11, p. 183] the feed forward matrix can now be derived as

$$\mathbf{F} = (\mathbf{C}(\mathbf{BK}-\mathbf{A})^{-1}\mathbf{B})^{-1}\ .$$

Figure 4.10 shows the balancing behaviour of the LQR controller for the cBoP system for an initial position of $x_0 = -0.1, y_0 = 0.1$. The LQR controller is designed for the linearised cBoP nominal system. Besides the nominal system the balancing behaviour is plotted for the four corner systems (a)-(d) of the uncertainty. For each of these, the rise time t_r, the overshoot o and the settling time T_s is given[10]. Additionally, 20 random systems within the bounds of uncertainty mentioned in 4.8 are displayed. The Figure illustrates a nice balancing behaviour for the nominal system. However, if the rolling friction coefficient is smaller, the response tends to higher settling times. Figure 4.11 shows that the ASL as defined in Section 4.3 are not exceeded. In order to check if the LQR controller achieves RS, the *robstab* Matlab command which is explained in Table D.1 is utilized. Using *robstab* RS is established for the designed LQR controller (see *lqr_design.m* lines 45 - 61). It should be noted, that if the LQR controller is designed for a rolling friction coefficient of $\mu_r = \frac{\mu_{r,max}+\mu_{r,min}}{2}$, the system response has an increased settling time and overshoot for lower rolling friction coefficients. The balancing behaviour for

[8] This is due to the attributes of the IK algorithm. Additionally, finding a solution for the IK algorithm is easier if the actual positions of the end-effector are similar to the desired positions.

[9] $\mu_r = 0.0295$ is used for the nominal system.

[10] The definition of these terms is given at *https://de.mathworks.com/help/control/ref/stepinfo.html*.

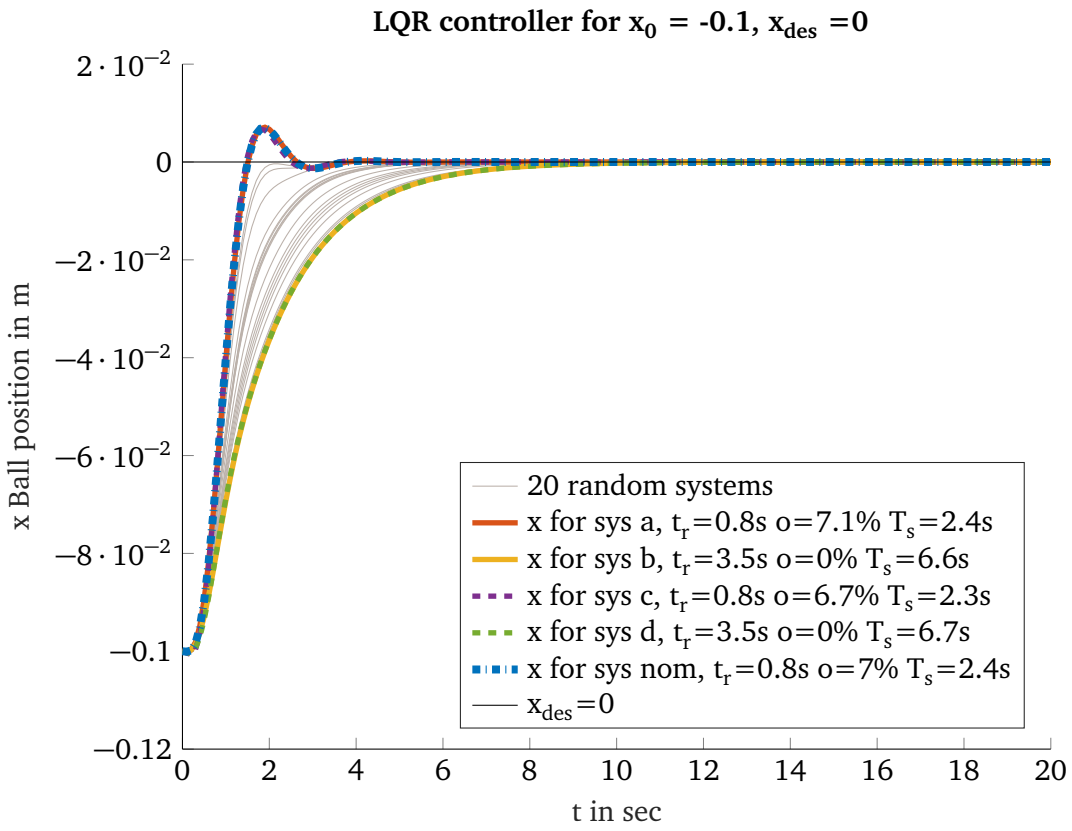


Figure 4.10.: Balancing behaviour of LQR controller for an initial position of the ball of $x_0 = -0.1, y_0 = 0.1$ (simulated)

these LQR controller is given in Figure C.1. Thus, it can be concluded, that designing a LQR controller with a low rolling friction coefficient leads to a stable controller for the defined uncertainties, because a higher rolling friction coefficient slows down the system response. The designed LQR controller achieves RS and fulfils the SRR.

4.7 Design of Multi-Model Pole Placement controller

One robust control approach to cope parameter uncertainties is Multi-Model Pole Placement (MMPP) [12]. MMPP aims to put all eigenvalues of models with different parameters (depicted in Figure 4.8) within a particular pole region (depicted in Figure 4.12). The better a feed back controller matrix $\mathbf{K}$ is able to put all eigenvalues within this pole region Γ the more robust is the controller with respect to these predefined uncertainties. Applying MMPP to the uncertainties (see Section 4.2) of the cBoP system, a stable controller with a minimum performance can be optimized. To achieve this performance (as defined in Section 4.4) a desired pole region of a damping ratio $\varphi \approx 79°$, a minimum speed [11] of -0.1δ and a maximum speed of -10δ is designed. Figure 4.15 shows this desired region Γ. By using MMPP a feed back matrix $\mathbf{K}$

[11] The minimum speed is a stability margin. The closer this margin to the origin of the complex plane, the more likely it is that the system tends to instability.

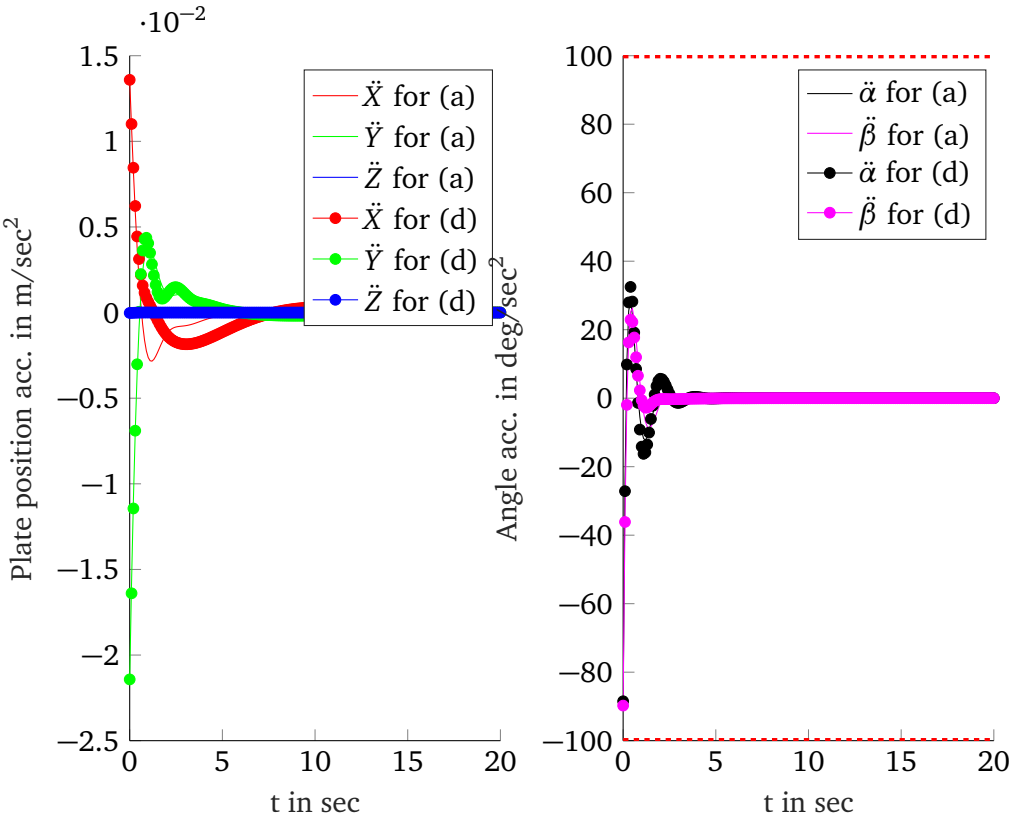


Figure 4.11.: Control input $\mathbf{u}$ of LQR controller (simulated)

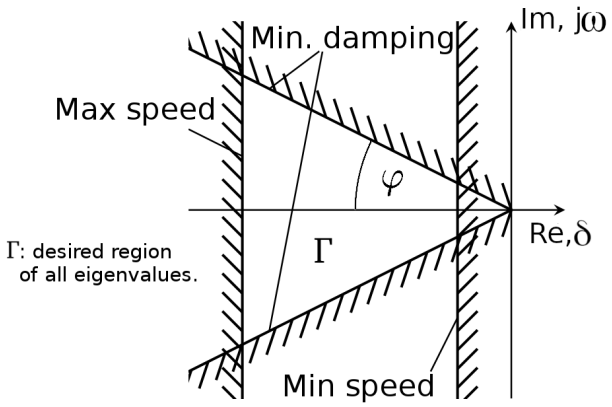


Figure 4.12.: Performance explanation of desired pole region [13].

is optimized in a way that the eigenvalues of all corner models are placed in the desired pole region. Figure 4.15 shows the poles of the closed loop systems of the four corner models (a)-(d). All eigenvalues are lying in the desired region Γ.

The result of the step response is given in Figure 4.13. The corresponding control input is given in Figure 4.14.

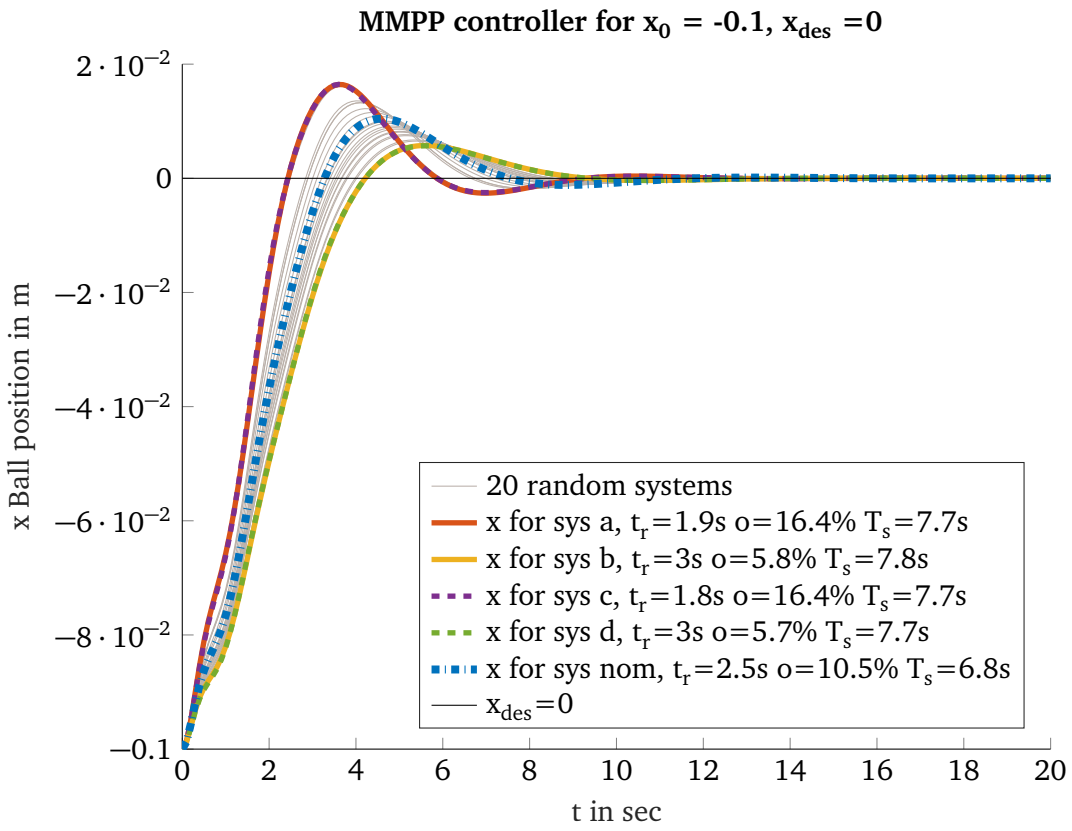


Figure 4.13.: Balancing behaviour of MMPP controller for an initial position of the ball of $x_0 = -0.1, y_0 = 0.1$ (simulated by Matlab)

With respect to the script *mmpp_design.m* (Line 67 - 85) RS could be proven with the Matlab command *robstab* [12].

Comparing Figure 4.10 (LQR controller) and Figure 4.13 (MMPP controller) the settling time for high rolling friction coefficients μ_r is significantly less for the MMPP controller. Thus, the MMPP controller achieves a more robust step response performance. The SRR as defined in Section 4.4 could be achieved with the MMPP controller. However, the ASL of the task-space (depicted as dashed red lines in Figure 4.14) are exceeded. The MMPP method cannot be used to weight certain control input paths. This means it is not possible to weight the angles of the end-effector more than the position[13]. Additionally, it is not possible to weight the states. Thus, the controller aims to stabilize the system around its EP and does not focus on controlling the position of the ball. As it is desirable to weight the inputs and the states it is necessary to design other robust controllers which are discussed in the next sections.

[12] For details about the Matlab commands see Table D.1.

[13] This aspect is illustrated in Figure 4.14. In this Figure the acceleration of the position are weighted very less compared to the acceleration of the angles.

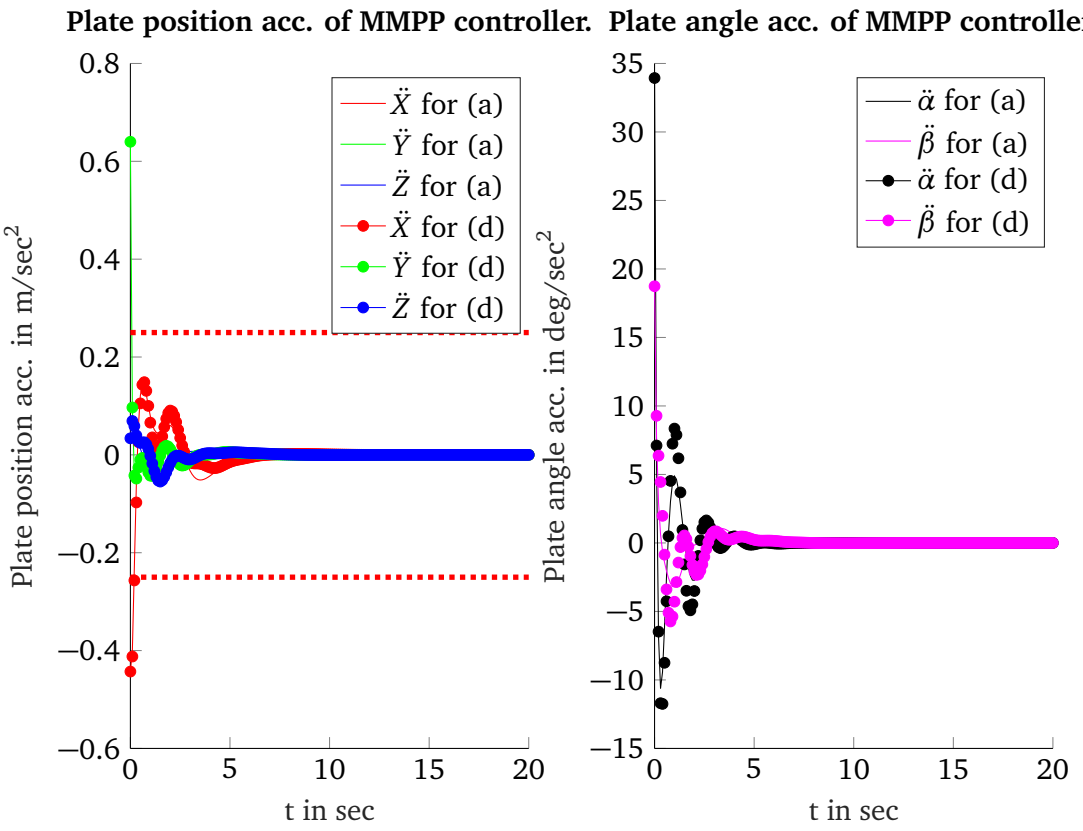


Figure 4.14.: Control input $\mathbf{u}$ of MMPP controller (simulated by Matlab)

4.8 Design of $\mathscr{H}_2$ controller

As it is not possible to weight the control inputs with the MMPP method, another control design which allows to weight the states and control inputs similar to the LQR approach , would be desirable. In a first step the LQR design (see (4.9)) can be formulated as a special $\mathscr{H}_2$ problem [8, p. 257 - 261]. The general $\mathscr{H}_2$ problem is given as

$$\min_{\mathbf{K}} ||\mathbf{F}_l(\mathbf{P}(s),\ \mathbf{K}(s))||_2 \quad \text{under the condition that } \mathbf{K}(s) \text{ stabilizes } \mathbf{P}(s) \text{ internal.} \tag{4.12}$$

The generalized plant $\mathbf{P}$ is given as

$$\mathbf{P}(s) = \left[\begin{array}{c|cc} \mathbf{A} & \mathbf{B}_1 & \mathbf{B}_2 \\ \hline \mathbf{C}_1 & \mathbf{0} & \mathbf{D}_{12} \\ \mathbf{C}_2 & \mathbf{D}_{21} & \mathbf{0} \end{array}\right]. \tag{4.13}$$

The objective of the $\mathscr{H}_2$ approach is to find the minimum $\mathbf{K}$ of the lower LFT of the generalized plant $\mathbf{P}$ and $\mathbf{K}$. [8, p. 261 - 265]. In order to solve (4.12) by common algorithms as e.g. Matlab *h2syn* it is required that

1. $(\mathbf{A},\ \mathbf{B_2})$ is controllable (1a), $(\mathbf{C}_2,\ \mathbf{A})$ is observable (1b),

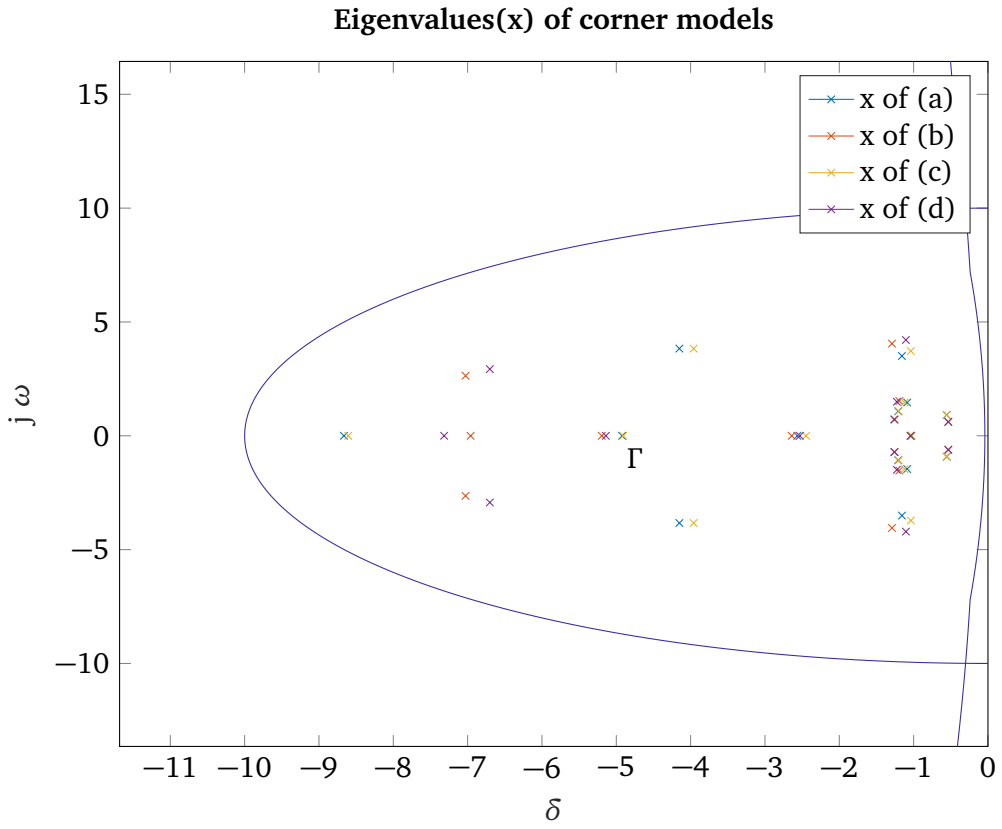


Figure 4.15.: Desired region of closed loop poles Γ and closed loop poles(x) of the four corner models

2. $\mathbf{D}_{12}$ has full column (2a) and $\mathbf{D}_{21}$ full row rank (2b),
3. $\begin{bmatrix} \mathbf{A}-j\omega\mathbf{I} & \mathbf{B}_2 \\ \mathbf{C}_1 & \mathbf{D}_{12} \end{bmatrix}$ has full column rank $\forall\omega$
4. and $\begin{bmatrix} \mathbf{A}-j\omega\mathbf{I} & \mathbf{B}_1 \\ \mathbf{C}_2 & \mathbf{D}_{21} \end{bmatrix}$ has full row rank $\forall\omega$.

If these requirements [3, p. 384] are achieved, it is guaranteed, that there is a solution of the $\mathscr{H}_2$ optimization problem common algorithms can find. A closed solution can be found in [3, p. 369, 370]. A LQR controller is a special $\mathscr{H}_2$ problem. Thus, a LQR controller can be formulated as a $\mathscr{H}_2$ problem with the following generalized plant $\mathbf{P}$ [3, p. 372]

$$\mathbf{P}(s) = \left[\begin{array}{c|cc} \mathbf{A} & \mathbf{S} & \mathbf{B} \\ \hline \begin{bmatrix} \mathbf{Q}^{0.5} \\ \mathbf{0} \end{bmatrix} & \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} & \begin{bmatrix} \mathbf{0} \\ \mathbf{R}^{0.5} \end{bmatrix} \\ \mathbf{C}_2 & \mathbf{0} & \mathbf{0} \end{array}\right]. \tag{4.14}$$

$\mathbf{S}$ in (4.14) is an arbitrary ($\neq \mathbf{0}$) matrix. In general $\mathbf{S}$ is a diagonal matrix. Each diagonal entry reflects the initial distortion of this state.

A special case of the $\mathscr{H}_2$ Optimization problem known as "full state feedback" problem, is when $\mathbf{C}_2 = \mathbf{I}$ in (4.14) holds. In this case each state is measurable and a solution can be found if condition (1a), (2b), (3) and (4) is satisfied [3, p. 390 - 393].

Considering the BoP system, each state is measurable. Thus, using a generalized plant of (4.14) allows to design a controller and be able to weight the control inputs and the states. However, this full state feedback problem can only be solved (e.g. with the Matlab *h2syn* command) for a nominal system. For this reason another method has to be utilized to design a robust controller.

4.9 Design of fixed-structure $\mathscr{H}_2$ controller using *systune*

$\mathscr{H}_2$ or $\mathscr{H}_\infty$ controller are optimal controllers with respect to the defined weighted inputs and outputs. This means there is no other controller that reaches a smaller $\mathscr{H}_2$ or $\mathscr{H}_\infty$ norm of the generalized plant $\mathbf{P}(s)$ with its given weights[14]. These controllers, however have disadvantages:

- Depending on the weights the computed controller is of high order.
- Commonly optimizing algorithms require measurement noise to compute a controller.
- Additional requirements such as limitation of the control input can only be considered by additional terms in the cost function.
- Integrative components are not supported.

Considering these disadvantages, it might be useful to formulate the controller design problem as

$$\min_{\mathbf{K}\in\mathscr{K}} \sum_i \|\mathbf{F}_l(\mathbf{P}_i(s),\ \mathbf{K}(s))\| \quad \text{under the condition that } \mathbf{K}(s) \text{ stabilizes } \mathbf{P}(s) \text{ internal,} \tag{4.15}$$

$$\|\mathbf{F}_l(\mathbf{P}_{s.t.,j}(s),\ \mathbf{K}(s))\| \leq \gamma_{max,j}, \forall j\ . \tag{4.16}$$

$\mathscr{K}$ is the set of valid controllers, which can contain a specification of the structure or the maximum order. Being able to use multiple generalized plants $\mathbf{P}_i$, makes it possible to weight certain control signal and reference signal paths. $\mathbf{P}_{s.t.j}(s)$ allows to formulate additional constraints for the optimization process.

Optimizing (4.15) leads to non-convex and non-smooth problems. To solve these problems numerical optimization methods (e.g. Matlab *systune*) are required. These methods do not guarantee to find a global or even local minimum[13].

In the following, a robust controller that uses the generalized plant (4.14) is designed by using the Matlab *systune* command. Table D.2 describes the inputs and outputs of this function.

The control circuit that is used to design the robust controller is depicted in Figure 4.16.

The so derived $\mathbf{K}$ is then used to construct the control law

$$\mathbf{u} = -(-\mathbf{K})\mathbf{y} + \mathbf{F}\mathbf{w}\ ,$$

[14] A weight is e.g. the $\mathbf{Q}$ or $\mathbf{R}$ matrix in the $\mathscr{H}_2$ design of the previous section.

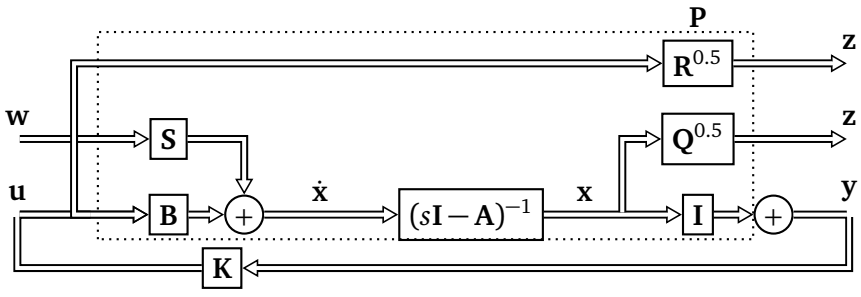


Figure 4.16.: Block diagram of the control circuit that is used to design the fixed-structure $\mathscr{H}_2$ controller

that is applied to the actual control circuit depicted in Figure 4.9. The prefilter matrix **F** is designed in a way that the closed control circuit has no steady state error. **F** is given by

$$\mathbf{F} = -(\mathbf{C}(\mathbf{A}+\mathbf{B}\mathbf{K})\cdot\mathbf{B})^{-1} \ .$$

With respect to Figure 4.16 the optimization problem can be formulated as

$$\min_{\mathbf{K}\in\mathscr{K}} \left\| \begin{bmatrix} x \leftarrow w_6 \\ y \leftarrow w_7 \end{bmatrix} \right\|_2 \text{ under the condition that } \mathbf{K}(s) \text{ stabilizes } \mathbf{P}(s) \text{ internal,}$$

$$\left\| \begin{bmatrix} \ddot{X}, \ddot{Y}, \ddot{Z} \leftarrow w_6 \\ \ddot{X}, \ddot{Y}, \ddot{Z} \leftarrow w_7 \end{bmatrix} \right\|_2 \leq 0.25, \left\| \begin{bmatrix} \ddot{\alpha},\ \ddot{\beta} \leftarrow w_6 \\ \ddot{\alpha},\ \ddot{\beta} \leftarrow w_7 \end{bmatrix} \right\|_2 \leq 100,$$

with

$$\mathscr{K} = \left\{ \begin{bmatrix} k_1 & \dots & k_{14} \end{bmatrix} \mid k_1, \dots, k_{14} \in \mathbb{R} \right\} \ .$$

Using *systune* this optimization problem can be numerically approximated by considering the ASL as HardGoals and the SRR as SoftGoals[15]. Additionally, *TuingGoal.Poles* was used to define a desired pole region, that prevents eigenvalues being close to the origin. The Matlab script of the fixed-structure $\mathscr{H}_2$ controller design is given in script *h2_systune.m*. Additionally, the RS of this controller could be proven with *robstab* [16]. Figure 4.17 illustrates the balancing behaviour of the linear cBoP for an initial position of the ball of $x_0 = -0.1$, $y_0 = 0.1$. The SRR could be achieved for this system. Additionally, Figure 4.18 shows that the task-space ASL are not exceeded.

[15] In particular a PT2 function can be definded as desired step response.

[16] The script *h2_systune_lin.m* shows the specific *systune* adjustments to full fill the ASL and SRR as well as the RS-analysis for this system.

4.10 Summary

The design of the MMPP controller revealed that it is desirable to be able to weight the states and the control input. The LQR and the fixed-structure $\mathscr{H}_2$ controller offer these possibilities. Additionally, tuning a generalized plant with *systune* it is possible to consider saturation limits and step response requirements.

Thus, using *systune* to design fixed-structure $\mathscr{H}_2$ controller represents a powerful method to consider ASL and SRR. In particular Table 6.1 compares the designed controllers. The SRR such as the worst overshoot, undershoot etc. are listed of all analysed system responses for each controller. The $\mathscr{H}_2$ controller model of this table is a redesigned $\mathscr{H}_2$ robust controller as discussed in Section 6.3.

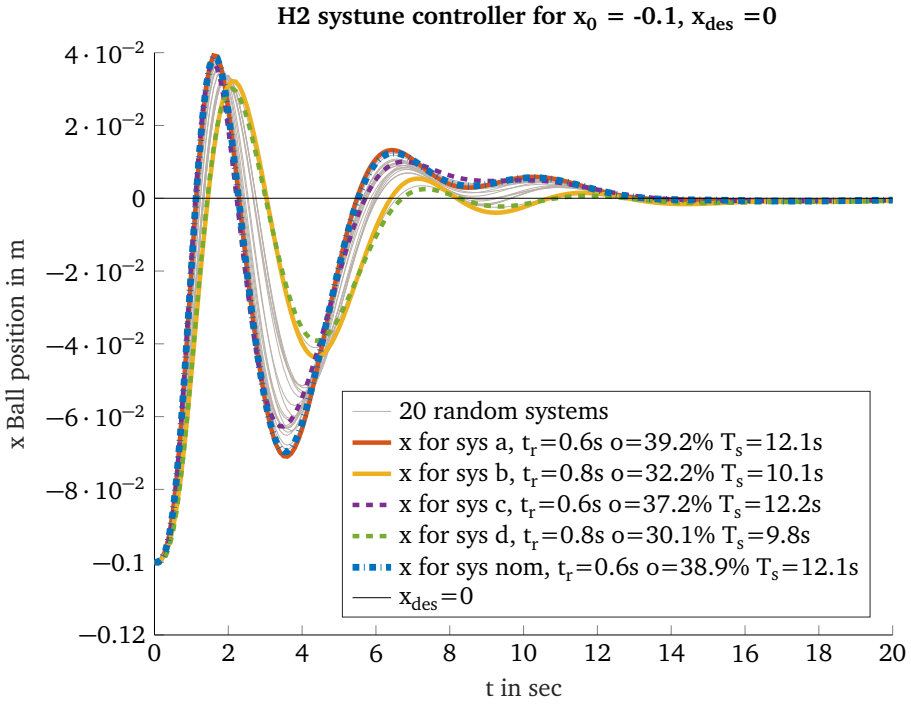


Figure 4.17.: Balancing behaviour of fixed-structure $\mathscr{H}_2$ controller for an initial position of the ball of $x_0 = -0.1$, $y_0 = 0.1$ (simulated)

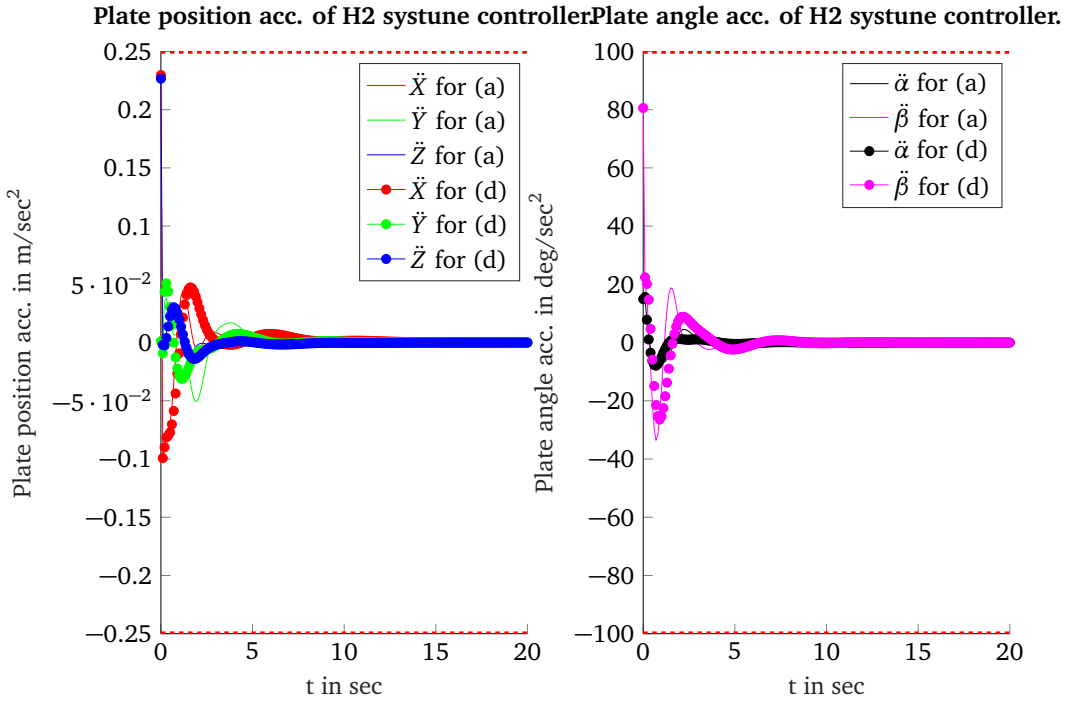


Figure 4.18.: Control input $\mathbf{u}$ of fixed-structure $\mathscr{H}_2$ controller (simulated)

5 Robust controller design with methods from Reinforcement Learning

Training a robust policy on real robots is time-consuming as the world state has to be reset after each rollout and of the risk that the robot is damaging itself or its environment during exploration. One approach to solve these problems is to create a model of the robot and train the policy in simulation. Policies which use exact physics constants to train the parameters of a neuronal network tend to overfit. Hence, a policy that has been trained to achieve a high performance in the simulation might fail in the real world. One approach to bridge this phenomena known as the reality-gap is modularizing the task[1] [14], training a policy with random initial states and learning a policy that is robust to changes in physics parameters. This approach was investigated for the decoupled Ball-on-Plate (dBoP) system in [4] and [15]. In simulation a neuronal network was trained using Proximal Policy Optimization (PPO). The reward function is based on the quadratic cost function[15, p. 12]

$$r(\mathbf{s}_t, \mathbf{a}_t) = \exp(c(\mathbf{s}_t^T \mathbf{Q} \mathbf{s}_t + \mathbf{a}_t^T \mathbf{R} \mathbf{a}_t)) \tag{5.1}$$

Here $r(\mathbf{s}_t, \mathbf{a}_t)$ is the reward, $\mathbf{s}_t$ are the continuous states at time step t, $\mathbf{a}_t$ are the continuous actions at time step t and $\mathbf{Q}$ and $\mathbf{R}$ are matrices to weight the states and the control inputs (similar to the LQR controller design). For more details refer to [4] and [15].

In this thesis a trained neuronal network (see Figure 6.8) for the cBoP system was given to compare the classical robust controllers with the approach above. The neuronal network was designed in *Pytorch*.

Training directly the physics simulation is one major advantage of designing a robust controller with RL techniques: non-linear behaviour as well as discretisation can be learned as well. Thus, there is no simulation gap between the linearised models and the physics simulation compared to the classical robust controllers (as discussed above). Moreover, no complex friction model of the BoP-system has to be derived. Additionally, the trained neural network can be easily extended with additional hidden layers or nodes if the learned policy does not maximize the reward. Disadvantages are that tuning hyperparameters of RL methods is tedious and the time expensive training.

The result of the simulated neuronal network is discussed in Section 6.4.

[1] Modularizing the task means that a policy is not learned end-to-end (sensor inputs to direct motor commands), but learned from the sensor input to desired position's of the robot in task space.

6 Simulation results

Before applying the designed controllers to the real robot, the controllers are first tested with the Robot Control System (Rcs). Rcs is an open source[1] set of C and C++ libraries for robot control and simulation. One major advantage is that the real hardware can directly be integrated instead of a physics-engine that is used for simulation. In this thesis Rcs was used to verify if the joint space Actuator Saturation Limit (ASL) (as defined in Section 4.3) are exceeded. For the simulation the Vortex[2] physics engine was utilized. The simulation parameters of Vortex can be found in the Appendix B. Whereas Chapter 4 evaluates the behaviour of the continuous linearised model, the Rcs simulation considers the non-linear behaviour of the BoP system and discretely obtains the feedback with 100 Hz. Additionally, the measurement noise (for the ball or the joints), the time delay of the measurements of the ball as well as errors of the motor (time-delay, discretization) were neglected in this chapter. The control circuit that was used in the Rcs simulation is given in Figure 4.9.

6.1 Linear Quadratic Regulator (LQR) controller

The balancing behaviour of the LQR controller depicted in Figure 6.1 and simulated by Rcs is slightly different to the model of the LQR controller. As the desired position was not reached for higher rolling friction coefficients, it can be assumed that the influence of the rolling friction coefficient in the Rcs simulation was higher than in the model. The joint space ASL were not exceeded for the system responses.

In order to decrease the steady-state error an additional integral part can be added to the controller as depicted in Figure 6.2.

Thus, the control law becomes[3]

$$\mathbf{u} = -\mathbf{K} \cdot \mathbf{x} + \int \mathbf{K}_I \cdot (\mathbf{F} \cdot \mathbf{w} - \mathbf{x_B}), \text{ with } \mathbf{x_B} = \begin{bmatrix} x & y \end{bmatrix}^T. \tag{6.1}$$

[1] Rcs can be downloaded from *https://github.com/HRI-EU/Rcs*.
[2] In this thesis Vortex 6.8.1 was used.
[3] Note that for this control law the **F** matrix has to be redesigned and becomes a 2x2 Matrix.

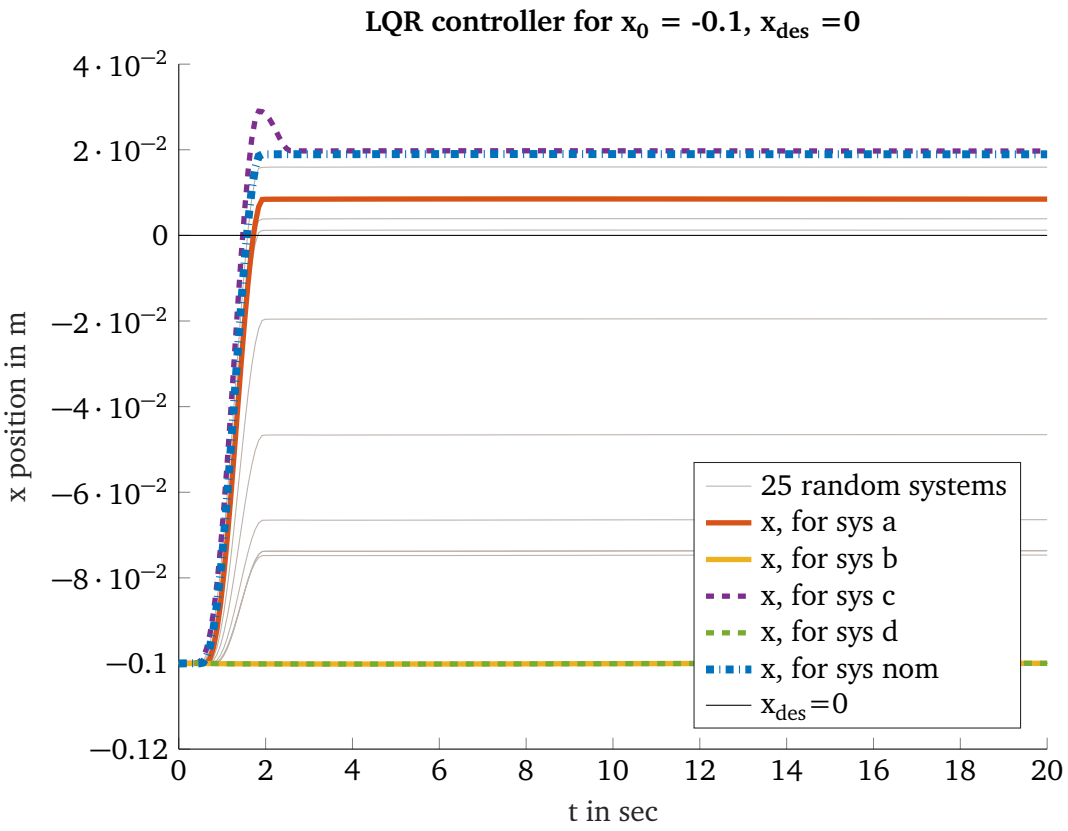


Figure 6.1.: Rcs simulation of LQR controller as designed in Section 4.6

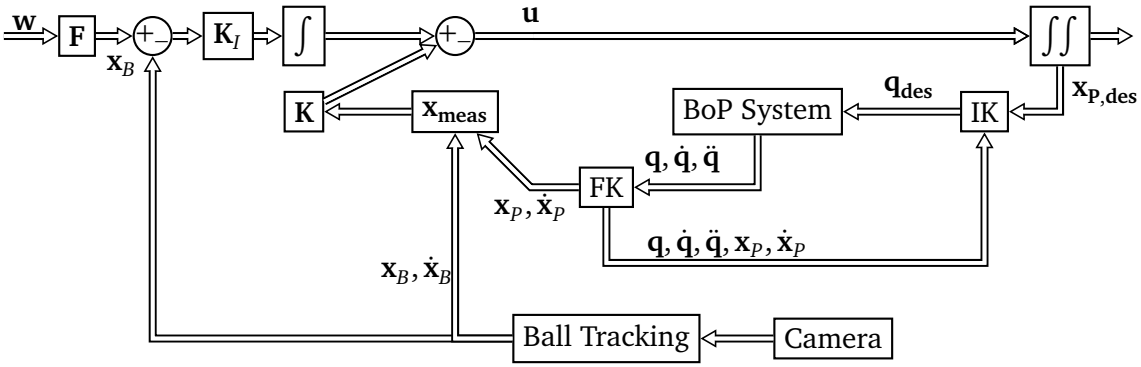


Figure 6.2.: Control circuit with additional integral part and $\mathbf{K}_I$ matrix to cope steady-state errors.

If the set-point is not reached the position error is integrated and should be added to the angle accelerations of the control input vector (adding it to the position accelerations often causes the robot to move to unreachable end-effector positions.). Thereby an integral matrix of

$$\mathbf{K}_I = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} \tag{6.2}$$

was selected. The balancing behaviour of this LQR-I controller is illustrated in Figure 6.3. The

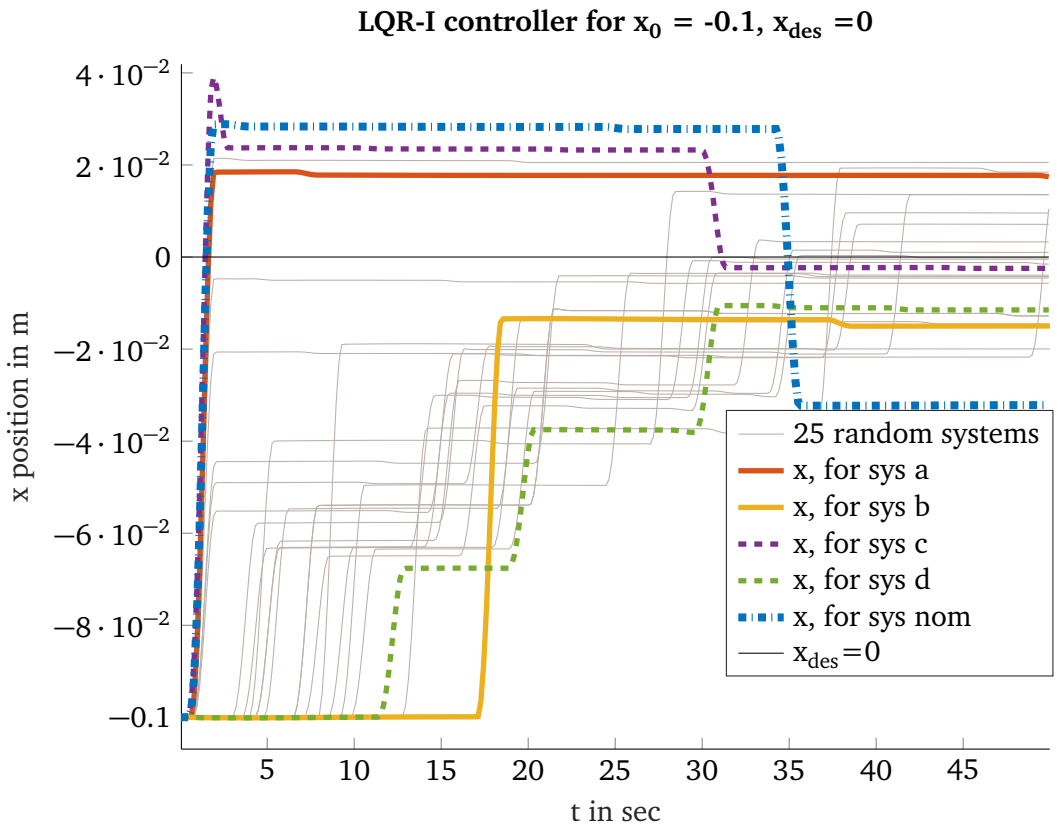


Figure 6.3.: Rcs simulation of LQR-I controller as designed in Section 4.6 with $\mathbf{K}_I$ of (6.2)

integrative part forces also the system (b) and (d) to move towards the origin. Thus, the steady-state error using an integral part can be reduced for these systems to sse$\leq \pm 2$ cm. With respect to the response of the nominal system in Figure 6.3 it is suspicious that integrating the position error for a long time leads to small jumps around the desired position (see (nom) for t>30 s). This disadvantage however does not cause instability.

It can be summarized that the simulated LQR controller responds less intensive to higher friction coefficients than the simulation of the linearised model. Using an additional integrative decreases the worst steady-state error of the analyses systems and forces the ball to move (even for higher rolling friction coefficients). The LQR and LQR-I controller are both stable for the uncertainties and the joint space ASL are not exceeded. It should be noted that the integrative part can *wind-up* and finally destabilize the system. In this case an *Anti-Windup* filter should be included. However, in the Rcs experiments this problem was not observed.

6.2 Multi-Model Pole Placement (MMPP) controller

The Rcs simulation of the MMPP controller manages the ball to move for all rolling friction coefficients as illustrated in Figure 6.4. The corresponding control input vector $\mathbf{u}$ as well as the actual joint positions $\dot{\mathbf{q}}$ velocities $\ddot{\mathbf{q}}$ and $\dddot{\mathbf{q}}$ accelerations for the nominal system is given in Figure C.2. As the task-space limits of the linearised model are already exceeded, it is not surprising

that the joint and task space limits of the simulated controller are also exceeded. In particular comparing the acceleration task-space limits of the linearised and the simulated controller, one observes that they are similar. However, at the example of the MMPP controller effects that only occur in the simulation can be shown:

- in Figure 6.4 one can observe, that the MMPP controller responses aggressive to steady-state errors. This dynamic response results in a magnification of non-linear effects that cause the ball to swing. These non-linear effects can be additionally amplified if the distance of the ball to the origin (which is the EP) is higher.
- another reason for the different response of the simulated controller is the discretisation. Figure C.3 shows the Rcs simulation sampled with 1 kHz. The corresponding control input vector **u** is given in Figure C.4. The figures show that the discretisation impacts the simulated controller. With a smaller time step in the simulation, the controller responds similar to the model. However, in this case one can still observe that for the maximum rolling friction coefficient and the minimum radius the ball does not move at all (see Figure C.3). Additionally, it should be noted that with a lower time step, the joint ASL decrease (see Figure C.4).

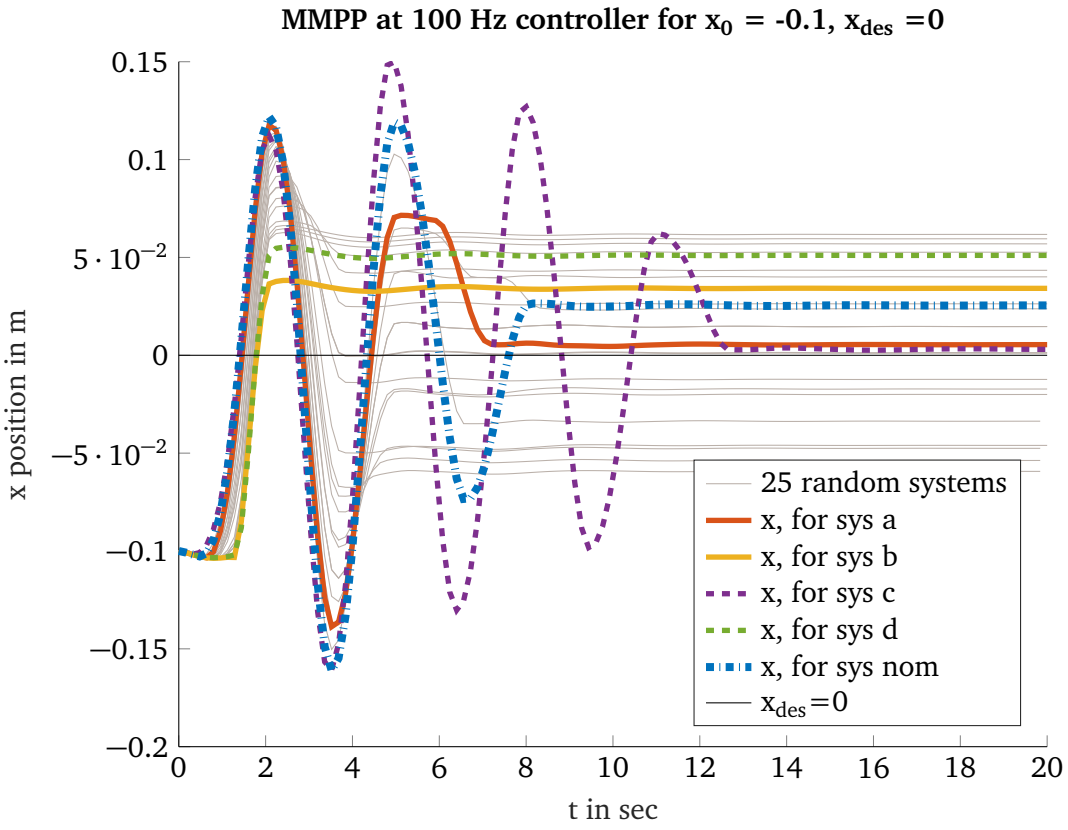


Figure 6.4.: Rcs simulation of MMPP controller with 100 Hz

The simulation of the MMPP controller revealed a strong impact of the discretisation and non-linear effects. Additionally, it showed that the chosen constant ASL in task space can be used to estimate the joint space ASL: as well as the modelled MMPP controller exceeds the task-space

limits, the simulated MMPP controller exceeds the joint-space limits. For this reason the MMPP controller should not be applied to the real robot.

6.3 Fixed-structure $\mathscr{H}_2$ controller

Applying the fixed-structure $\mathscr{H}_2$ controller as designed in Section 4.9 to the Rcs simulation results in an unstable response. The reason for this is a high overshooting which is additionally increased by non-linearities. As the step response goal causes these high overshoots, the fixed-structure controller was redesigned with different tuning goals. Additionally, the desired region of the eigenvalues was shifted to the left on the real axis. Thereby the controller becomes more stable and can be applied to the discrete Rcs simulation. Moreover, the position of the end-effector (in task-space) was limited, in order to prevent that the robot moves to positions which are physically not possible[4]. The modelled balancing behaviour of this redesigned controller is given in Figure C.6.

Figure 6.5 shows the balancing behaviour of the Rcs simulation of this redesigned controller.

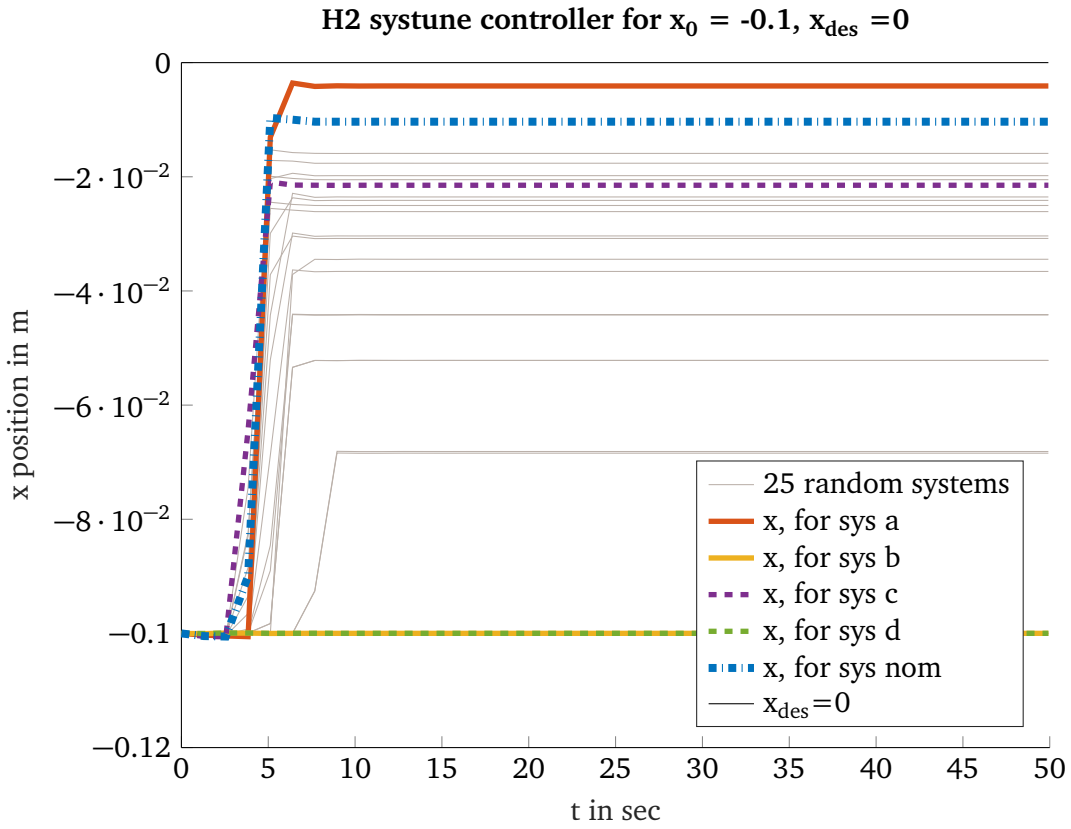


Figure 6.5.: Rcs simulation of fixed-structure $\mathscr{H}_2$ controller

[4] The script *h2_systune_re.m* shows the tuning goals of the redesigned controller.

Similar to the LQR controller the influence of the rolling friction coefficient is higher compared to the model. This causes high steady-state errors or prevents the ball to move (e.g. for system (b) or (d)). For this reason an integral part (as described in Section 6.1) of

$$\mathbf{K}_I = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -1.5 \\ 0 & 0 \end{bmatrix} \tag{6.3}$$

was added. It was observed that adding an integral part only for $\ddot{\beta}$ is sufficient enough to decrease the steady state errors in x and y direction. Figure 6.6 illustrates that thereby the steady-state error could significantly be decreased.

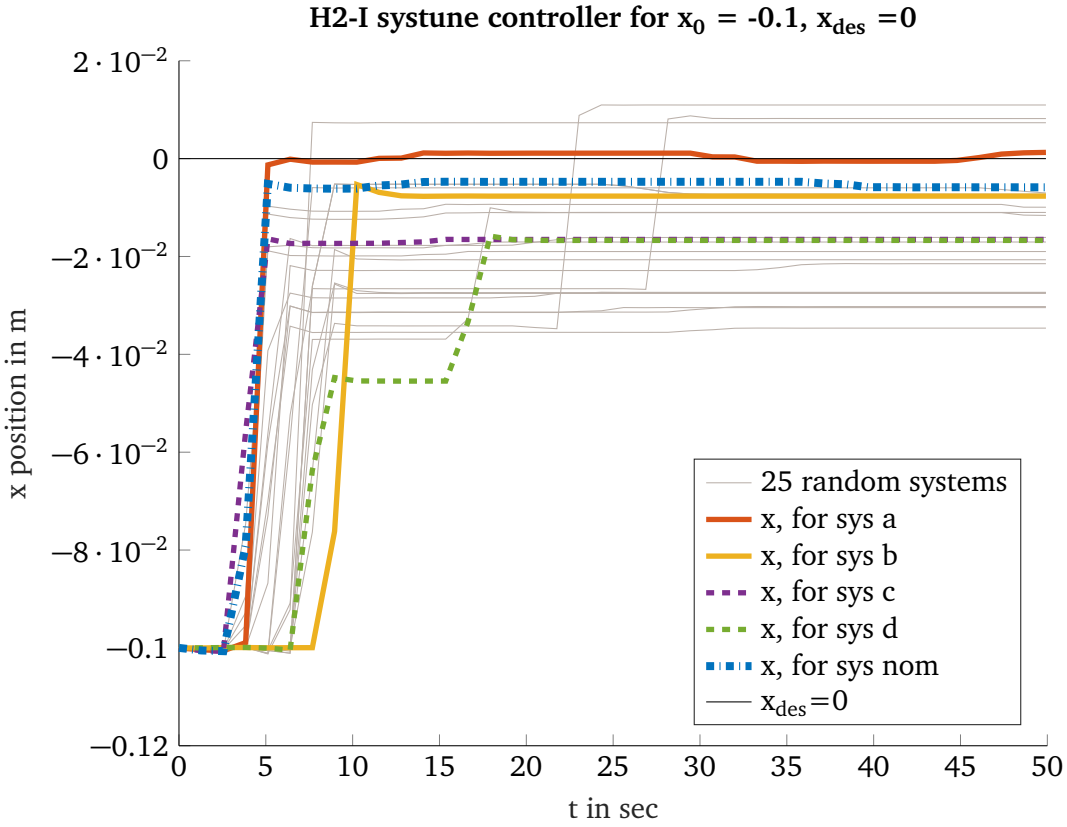


Figure 6.6.: Rcs simulation of fixed-structure $\mathscr{H}_2$-I controller

For both controllers the ASL in joint space were not exceeded. As an example the accelerations of the joint and task-space for the nominal system of the $\mathscr{H}$-2-I controller are illustrated in Figure 6.7. One can observe that even though the ASL of velocity task-space are exceeded, the corresponding joint limits are not crossed. Thus, the limits of the task-space might be chosen to conservative or the controller balances the position of the ball in a way that prevents the need of high joint accelerations and velocities.

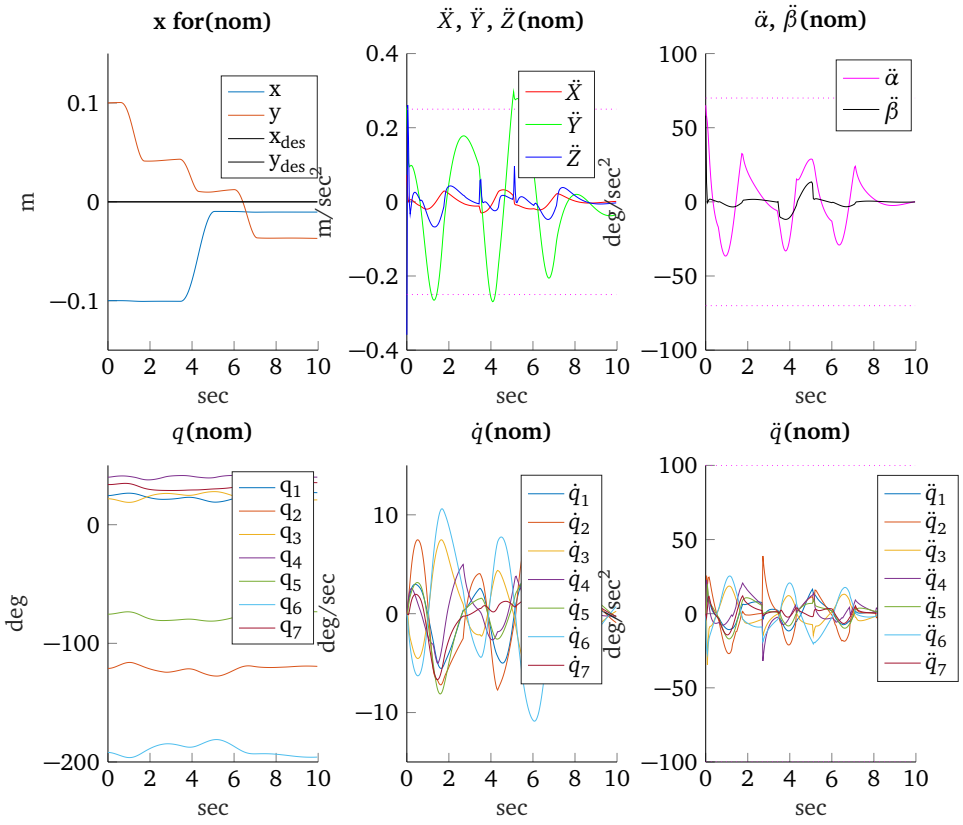


Figure 6.7.: ASL of the task- and joint-space of the Rcs Simulation of the fixed-structure $\mathscr{H}_2$-I controller

Further investigations (see Appendix C.4) revealed that linearising the model around just one EP causes an exceeding of the joint space ASL if the initial position of the ball has a grater distance ($y = 0.2\,\mathrm{m}$) to the origin.

Comparing the balancing behaviour of the $\mathscr{H}$-2-I controller with the LQR-I controller, it is conspicuous that the responses of the $\mathscr{H}$-2-I controller are all similar and lie between a tight band. On the other hand the LQR-I controller responds different for higher and lower rolling friction coefficients. Thus, comparing these two controller designs one can observe that using robust control does not only guarantee stability for the modelled uncertainty but also allows to define a minimum performance (e.g. in terms of SRR).

6.4 Robust controller designed with Reinforcement Learning

The control circuit that was used to evaluate the robustness of the fully-connected neuronal network (input layer 14 nodes, 2 hidden layers with each 64 nodes, output layer 5 nodes) is depicted in Figure 6.8.

The balancing behaviour of this neuronal network is tested by evaluating the system response of the four corner uncertainty systems (a)-(d), the nominal system (nom) and additionally 20

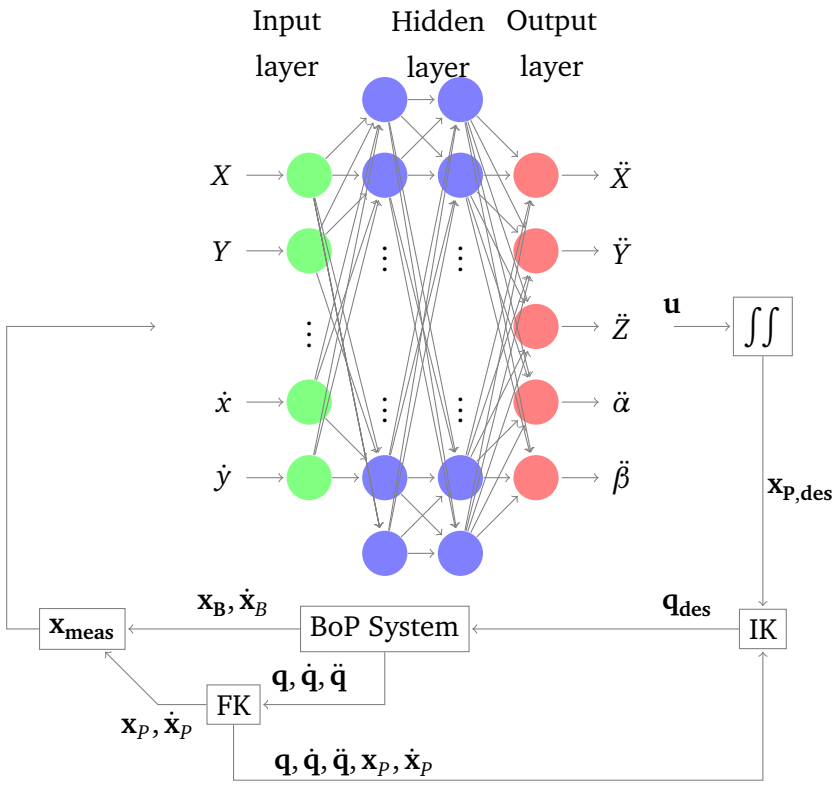


Figure 6.8.: Control circuit with the trained neuronal network

random systems within the uncertainty bounds (see Section 4.2). The result is depicted in Figure 6.9.

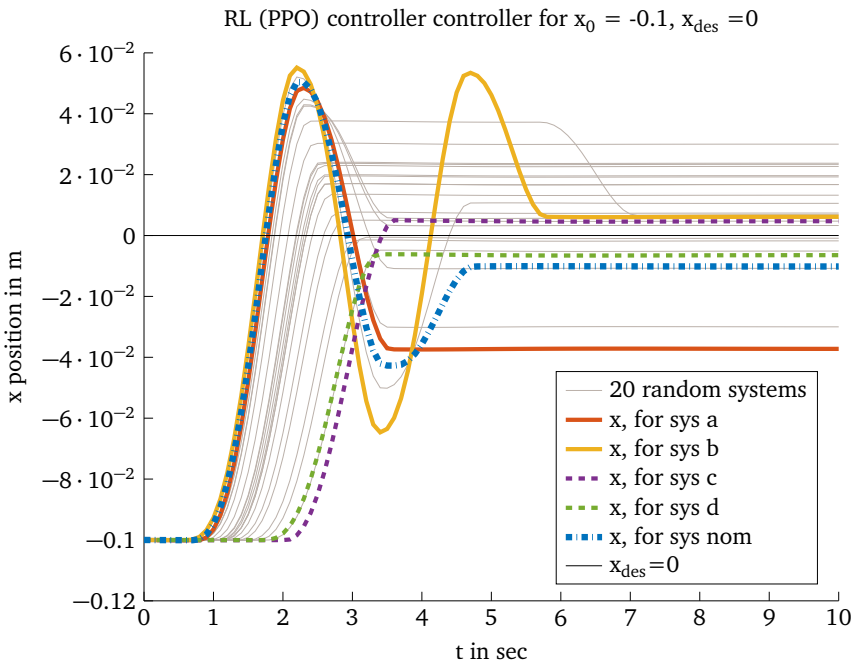


Figure 6.9.: Rcs simulation of robust controller designed with PPO

One can observe that the neuronal network trained with PPO overshoots up to 55 % for low rolling friction coefficients. Additionally, the system response for low radii and low rolling friction coefficients have a higher steady state error[5]. The task and joint space limits for the controller were not exceeded. Thus, the controller designed with methods from RL is a robust controller that is competitive to the $\mathscr{H}_2$-I controller: Input states and output states can be weighted using (5.1), the network can be scaled easily (by adding additional nodes, or hidden layers), non-linear behaviour as well as discretization can be learned directly and no model of the system has to be derived. However, it was not yet investigated how to formulate additional goals (such as a certain step response behaviour). Moreover, the controller designed with methods from RL learned much more parameters compared to the 70 parameters of the $\mathbf{K}$ feedback matrix. For this reason it was already suspected to outperform the $\mathscr{H}_2$-I controller.

6.5 Summary

When simulating the linearised controllers, it can be observed, that the discretisation of 100 Hz causes an unstable response of the MMPP controller. Furthermore, the impact of the rolling friction coefficient was significantly higher for the LQR and fixed-structure $\mathscr{H}_2$ controller. Adding an integrative part revealed a system response with less steady state error. The only linearised controller that fulfils the joint-space ASL and the SRR in the Rcs simulation is the fixed-structure $\mathscr{H}_2$-I controller. Benchmarking the classical robust controller ($\mathscr{H}_2$-I) against the controller designed with methods from RL, one can observe that the $\mathscr{H}_2$-I is slower and undershoots less as depicted in Table 6.1. As the controller trained with PPO is able to learn non-linear behaviour as well as discretization and is much more easy to extend it is expected to outperform the $\mathscr{H}_2$-I controller on the real robot.

[5] This error might be removable by adding an integrative part. Within this thesis this possibility was not investigated.

	LQR Model	LQR Sim	LQR-I Sim	MMPP Model	MMPP Sim	$\mathscr{H}_2$ Model	$\mathscr{H}_2$ Sim	$\mathscr{H}_2$-I Sim	PPO Sim
Figure	4.10	6.1	6.3	4.13	6.4	C.6	6.6	6.6	6.9
Full fills ASL	√	x	√	x	x	√	√	√	√
RS	√	–	–	√	–	√	√	√	–
Full fills SRR	√	x	x	√	x	√	x	√	√
Weight inputs	√	√	√	x	x	√	√	√	√
Weight states	√	√	√	x	x	√	√	√	√
Worst overshoot [%]	7	29.0	39.0	16.4	151.1	0.1	0.0	11.0	55.16
Worst undershoot [%]	0.0	0.1	0.1	0.0	61.6	0.0	0.7	1.4	0.1
Longest time to reach ±5 cm band[s]	1.5	Inf	20.4	3.2	9.8	4.1	Inf	9.2	4.9
Worst steady-state error[cm]	0.0	10	3.2	0.0	6.1	0.0	10	3.4	3.7

Table 6.1.: Comparison of different robust controllers. The worst case balancing behaviour (overshoot, undershoot, longest time to reach, worst steady-state error) of all system responses for one controller is listed. Sim is the result of the Rcs Simulation. PPO corresponds to the controller designed with methods from RL.

7 Robot results

This chapter discusses how noise and time delay of the image processing impacts the Rcs simulation. In a next step robot control modes as well as motor restrictions are explained. Finally, the LQR controller is evaluated on the real robot.

7.1 Measurement noise

The following noise effects are observed:

- noise of the position measurements of the end-effector with a static STD of $< 0.1\,\text{mm}$
- noise of the position measurements of the ball with a static STD of $0.9\,\text{mm}$ and of the velocity measurements of the ball with a static STD of $2.2\,\frac{mm}{s}$ (see Figure 3.3).

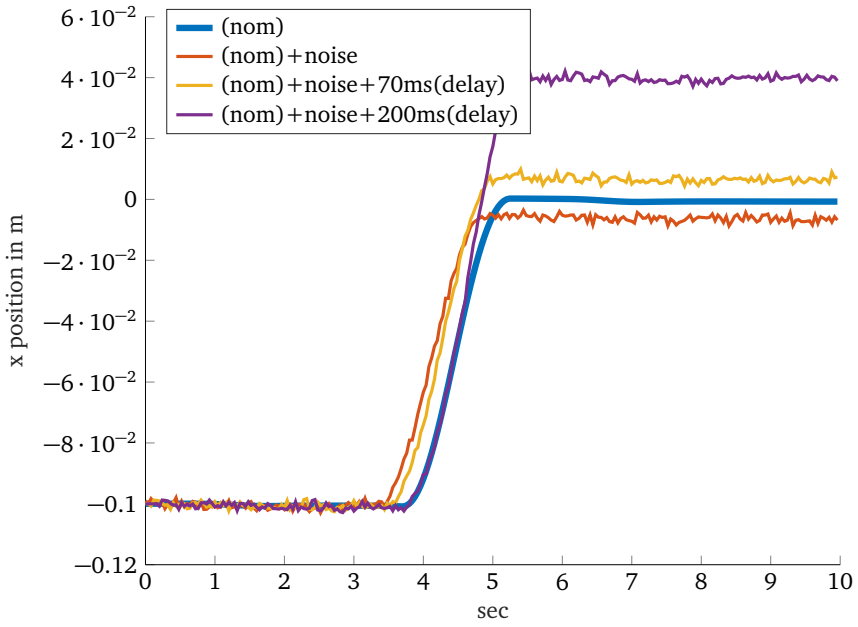


Figure 7.1.: Rcs Simulation of fixed-structure $\mathcal{H}_2$-I controller of the nominal system with additional noise and delayed position and velocity of the ball compared to the end-effector position measurements.

Figure 7.1 illustrates how the balancing behaviour of the simulated fixed-structure $\mathcal{H}_2$-I controller is influenced by this noise. The above noise effects impact the system to react earlier and causes a higher steady-state error. It can be summarized that the noise of this amount has no critical influence to the system.

7.2 Time Delay of image processing

In order to obtain the delay between the position of the end-effector and the position of the ball, the ball was fixed at the origin. By moving the position of the plate in a sinusoidal way along the Y position and measuring the position of the end-effector $Y - Y_{\text{offset}}$ as well as the position of the ball y_f relatively to a fixed initial coordinate system, the delay of the image processing can be calculated by cross-correlation. Figure 7.2 illustrates both, the measurements and the cross-correlation. The maximum time difference (delay) of these measurements sampled with 100 Hz is 70 ms. To align both, the measurements of the ball should be either predicted into the future using the Kalman Filter or the measurements of the end-effector should be delayed by using e.g. a First In - First Out (FIFO) filter. If the measurements are not aligned the system response countersteers with a delay in order to control the position of the ball. If this delay is to high the system response tends to swing up. If the position and velocity of the ball is delayed by 70 ms or 200 ms the simulated fixed-structure $\mathscr{H}_2$-I controller is still able to balance the ball. Figure 7.1 illustrates that this time delay causes a higher steady-state error. In particular, it should be noted, that if the system response overshoots, this time delay is critical as the response might swing up and cause instability.

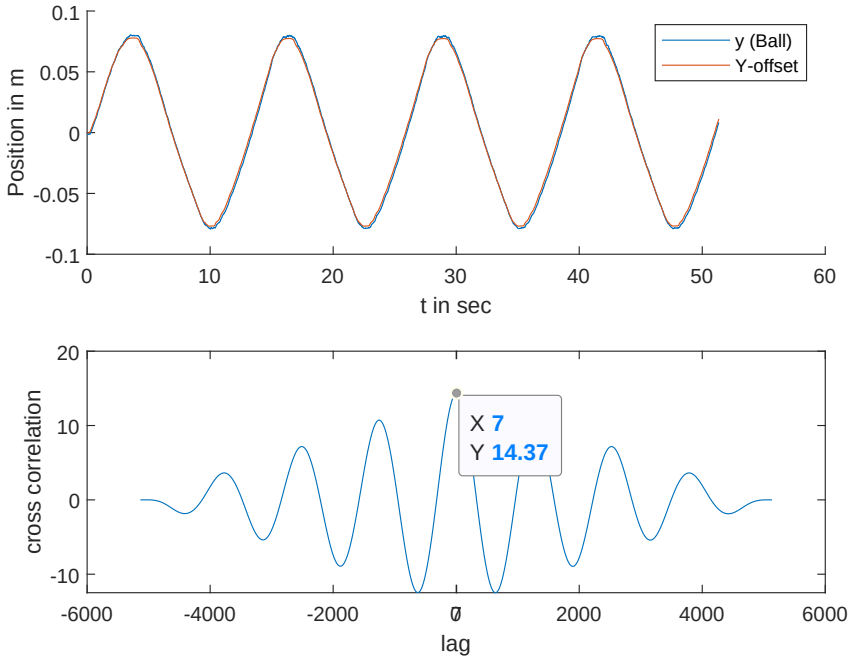


Figure 7.2.: Top: Y — offset-position of the end-effector of the robot and y_f position of the ball around a static coordinate system. Bottom: Cross-correlation between both position measurement curves.

It can be summarised that a time delay of 70 ms does not disturb the simulated fixed-structure $\mathscr{H}_2$-I controller. This low impact, however, is due to the fact that this controller does not overshoot and magnify the time delay by swinging. Tests on the real robot (see Section C.5) revealed

that this time delay of the image processing does not have a huge impact and was therefore neglected.

7.3 Real motor restrictions

Rcs offers two different control modes to apply the desired joint positions to the Schunk motors:

- the mode *Control Velocity* offers three filters for the velocity: *Slow*, *Medium* and *Fast*. Depending on the filter, the motor commands are internally smoothed and decelerated before being applied to the motors.
- the mode *Move Step* is a non filtered mode and directly passes the commands (unfiltered) to the motors.

In an experiment the plate was moved around the inclination angle α in a sinusoidal way once using the *Control Velocity - Fast* mode and once using the *Move Step* mode. Figure 7.3 illustrates the result of this experiment.

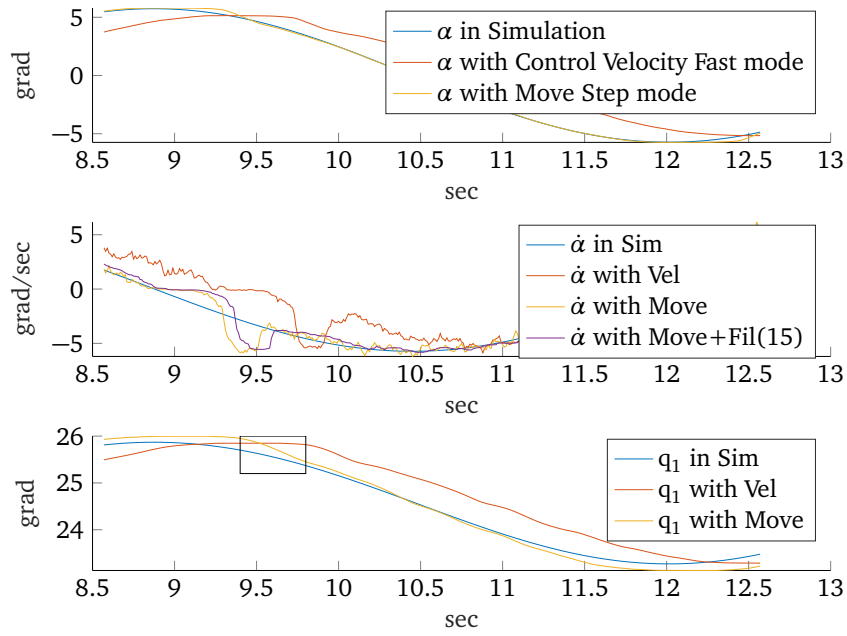


Figure 7.3.: Moving the plate around angle α in a sinusoidal way. Top: α in Simulation (Sim) and on real robot with control mode *Velocity Fast* (Vel) and *Move Step* (Move), Middle: $\dot{\alpha}$ of the same experiment. Additionally, $\dot{\alpha}$ of the *Move Step* control mode with a median filter of window size 15 is illustrated, Bottom: joint angle q_1 of same experiment.

From this experiment one can observe:

1. using the *Control Velocity Fast* mode there is a delay of up to 0.5 sec between the desired and the actual task-space and joint space value. In Figure 7.3 only q_1 is shown as an example. However, this delay is present for all joints. It is too high to balance the ball on the plate. Thus, the *Control Velocity Fast* mode should not be used.

2. the velocity of the angle $\dot{\alpha}$ is very noisy[1] and there are non-linear jumps (at around 9.5 sec). These jumps cannot be smoothed significantly using a median filter of window size 15. Increasing the window size has the downside of an undesired delay. Thus, a KF or a reduced Luenberger observer should be used to measure the states more exactly.

3. the black rectangle in the bottom plot illustrates a deadband or dead zone behaviour of the brushless DC servo Schunk motors: In order to turn the motors, a certain minimum amount of a current deviation is needed. Thus, if the desired angle is smaller than 0.1 ° the motor does not turn. This effect causes a non-linear behaviour which is not desirable and not avoidable. In particular, it results in more jerkily movements that are delayed by up to 0.2 sec.

It can be summarized that the real motors have a non linear behaviour that causes noisy velocity measurements with occasionally jumps of $\dot{\alpha}$ and $\dot{\beta}$. Thus, these states have to be measured more precisely e.g. using a Kalman Filter. Moreover, the unfiltered *Move Step* control mode should be used [2]. Experiments on the real robot (see Section C.5) revealed that the ball could only be balanced by using the *Move Step* control mode. Within this thesis using a KF to measure $\dot{\alpha}$ and $\dot{\beta}$ was not further investigated.

7.4 Including the model

In order to successfully apply the controllers to the robot, the control circuit of Figure 4.9 could not be used. The problem of this method is, that the noisy measurement vector $\mathbf{x}$ is multiplied with $-\mathbf{K}$ and two times integrated before being applied to the robot. This noisy measurement and double integration causes a $\mathbf{x}_{P,des}$ that was to slow to balance the ball. Analysing the the Rcs simulation of the LQR controller with the control circuit with an included model (see Figure 7.4) makes it clearer: including the model in the control circuit forces the controller to follow this model and results in a faster and more smooth step response as illustrated in Figure 7.5.

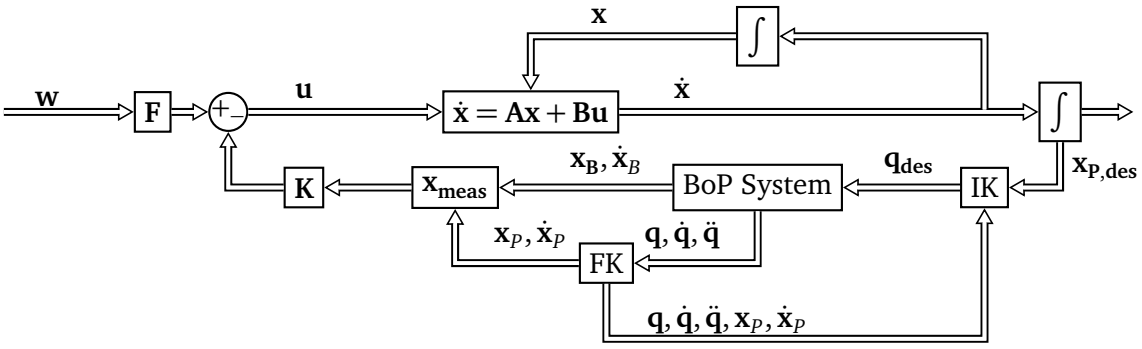


Figure 7.4.: Control circuit that includes the model.

[1] Note that the pose of the end-effector is measured by using FK. The velocity of this pose is then simply be derived by using finite differences. This method results in noisy measurements.

[2] In the current Rcs simulation the deadband of the motors was not included. For further work with the BoP system it is advised to incooperate these non-linearities in the simulation.

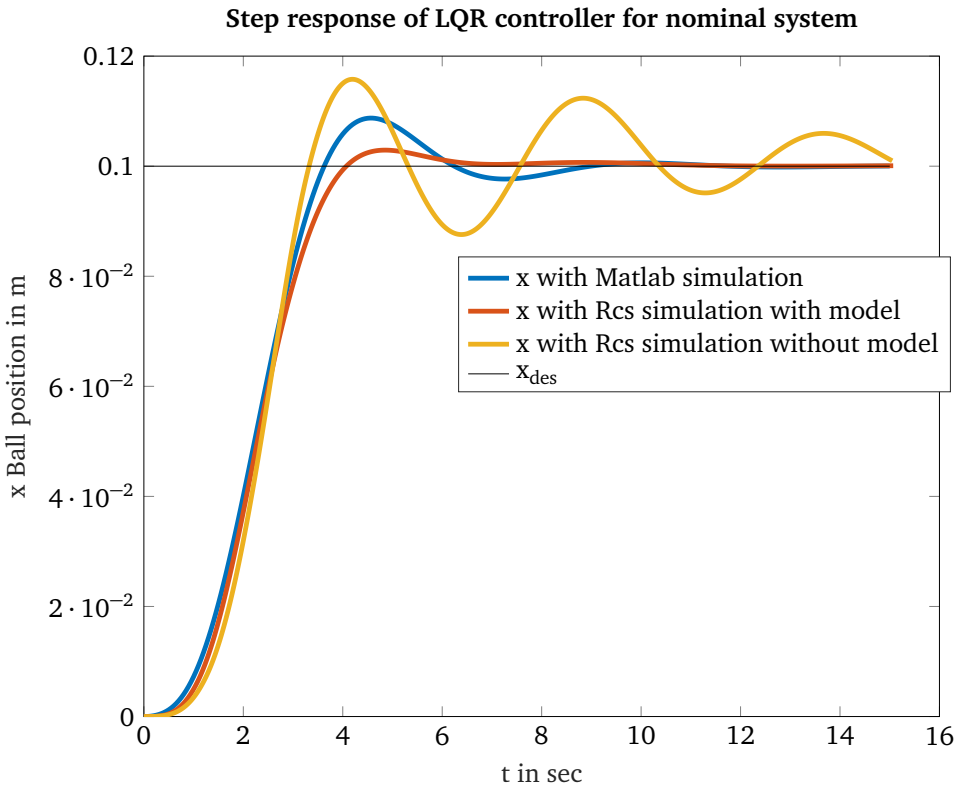


Figure 7.5.: Step response of LQR controller for cBoP system. The result of the Matlab simulation is compared with the control circuit that includes the model and with one that does not include the model.

On the real robot the positive effect of not integrating noisy data twice but interpreting $\mathbf{u}$ as the input of a model that the plant should follow made a balancing of the ball possible.

7.5 LQR-I controller

With the above considerations the LQR-I controller that is designed for the pool ball assuming no rolling friction and the following

$$\mathbf{Q} = \text{diag}(0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 2.0\ \ 2.0\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 1\ \ 1)$$
$$\mathbf{R} = \text{diag}(80\ \ 80\ \ 80\ \ 1\ \ 1)$$

matrices to weight the states and the inputs is utilized to evaluate robust control on the real robot. It was necessary to redesign the LQR controller in order to apply it successfully to the real robot. If the parameters of the simulation (4.11) were utilized, it was observed that the robot reacts to aggressive and thereby exceeds the maximum joint velocity limits. One reason for this behaviour are the non-linear effects explained in Section 7.3. For this experiment the control circuit of Figure 7.4 was used.

The feedback matrix $\mathbf{K}$ that was designed in that way is used for different friction coefficients (from μ_1 (low) over μ_2 (middle, tissue under the ball), to μ_3 (high, piece of cloth material under the ball)) and different balls (described in B) as depicted in Figure 7.6.

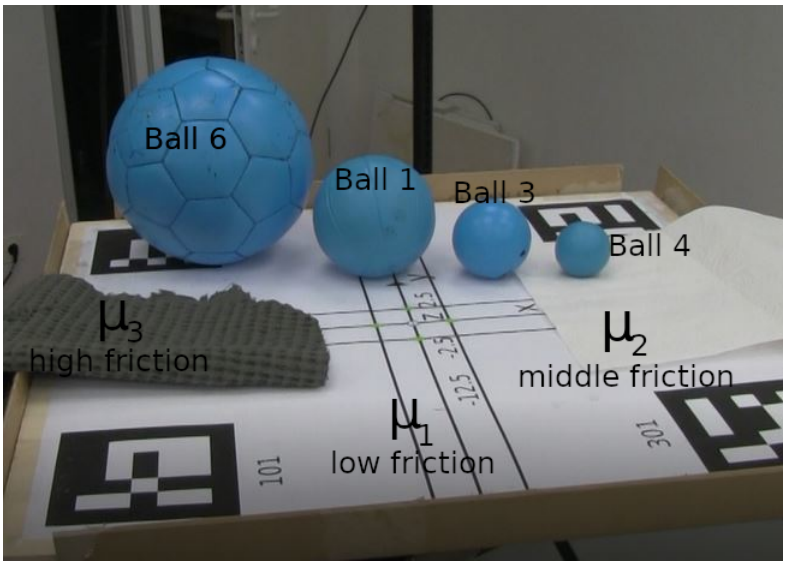


Figure 7.6.: Balls and friction coefficients used for the real robot experiment

The LQR-I controller was able to balance ball 3, 4 (except for μ_3) and 6 (except for μ_3) as depicted in Figure 7.7.

Thus, it can be concluded that the LQR-I controller could not handle the high rolling friction coefficient for different ball sizes.

It can be summarized that the LQR-I controller revealed a robust balancing behaviour for different ball and physics parameters. Thus, this controller could be used to cross the reality gap.

Due to the camera construction the robust fixed-structure $\mathscr{H}_2$ controller as well as the controller designed with RL methods could not be tested so far. Both controllers required the position of end-effector to move in a bigger region that was not covered by the camera. Thus, the ball could not be tracked for these controllers. Additionally, it should be noted that a camera construction that has a greater distance to the plate causes more noisy measurements (especially for small balls) as the resolution of the image decreases.

Additionally, a LQR controller that allows more movement of the position of the end-effector is analysed in Appendix C.5. Moreover, plate imperfections are investigated in Appendix C.6.

7.6 How to cross the reality gap

With the results of the above chapter and the experience gained from the experiments a guideline to cross the reality gap is presented:

1. identify the real system
    - determine the discretisation, resolution, limits(max. acceleration, max. speed etc.), time delays etc. of your sensors and motors.
    - try to measure (or estimate) physics parameters (dimensions, contact frictions, masses etc.).
    - with respect to the identified values determine parametric and dynamic uncertainties of the system.
2. try to include the non-linear effects of the identified behaviour of the system as well as motor limits in the simulation as good as possible.
3. design a robust controller for the parametric and dynamic uncertainties. If the simulation does not simulate non-linear behaviour correctly (such as contact friction or motor resolution, deadbands etc.) extend the uncertainty range for these parameters.

In order to design a robust controller using control theory the following guideline is advised:

- derive the equations of motion (e.g. by Lagrange) of your system.
- design a continuous LQR-I controller as a base line of the linearised nominal model.
- using the identified parametric and dynamic uncertainties design a robust controller (e.g. a fixed-structure $\mathscr{H}_2$ controller).
- test the performance of the controllers in simulation.
- in case the discretization error is to high: discretise the system.

- in case the impact of the non-linearities is to high: Linearise the system around many EP and interpolate between them (Gain-scheduling controller)[3].
- evaluate the controllers on the real system.
- measure the states required for the controller. If the measurements are too noisy consider to design an observer (reduced or KF).

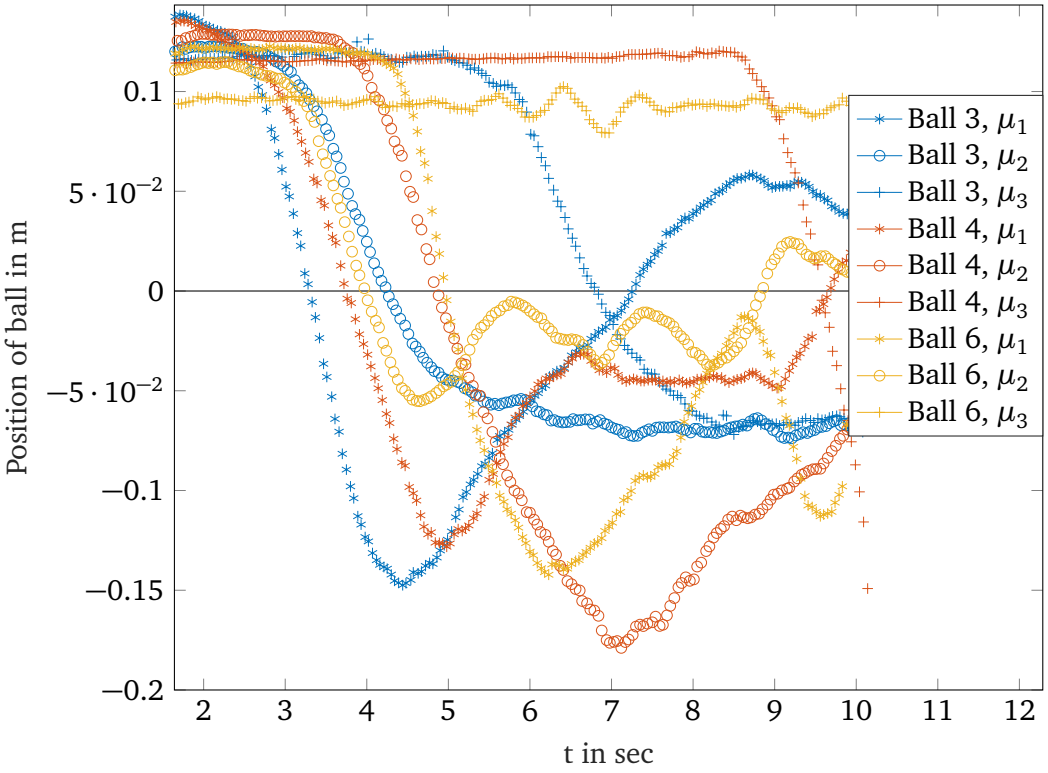


Figure 7.7.: Balancing behaviour of LQR-I controller on real robot.

[3] In this case a robust controller has to be designed for each system. Thus, RS cannot be proven anymore.

8 Discussion and Outlook

In this thesis the model of the 4-DoF and the 7-DoF Ball-on-Plate system which additionally includes the rolling friction of the ball is derived. Using this model two robust controllers are designed for the parametric uncertainties of the radius and the rolling friction coefficient of the ball. The first one is designed with the multi-model pole placement method and the second one is a $\mathscr{H}_2$ fixed-structure controller. Simulating these controllers in Matlab revealed that the first controller was not able to meet the actuator saturation limits in task-space. Moreover, the first controller has the disadvantage of not being able to weight certain states or control inputs. In contrast the introduced design method of a $\mathscr{H}_2$ fixed-structure controller offers to weight the states, the control inputs and to formulate additional goals to meet e.g. step response requirements. With respect to the uncertainties robust stability is proven for this controller. Both controllers were compared with a linear quadratic regulator that was designed as a baseline.

In a next step the controllers were evaluated in the Robot Control System simulation which utilizes the Vortex physics engine. As expected the controller designed with the multi-model pole placement method was not able to meet the joint space actuator saturation limits. Additionally, it turned out that the impact of the rolling friction coefficient in the Robot Control System simulation was higher compared to the linearised model. Thus, the linear quadratic regulator and the $\mathscr{H}_2$ fixed-structure controller were not able to move the ball for high friction coefficients. This issue could be solved by adding an additional integrative part. In total the robust $\mathscr{H}_2$ fixed-structure controller had the fastest and smoothest system response (of the classical controllers) for the different radii and rolling friction coefficients of the balls. In terms of robustness it was determined, that especially the controller designed with multi-model pole placement has higher discretization errors, that caused the robot to exceed its joint space actuator saturation limits. In order to decrease non-linear effects, the $\mathscr{H}_2$ fixed-structure controller was designed to prevent dynamic behaviour (such as overshooting). Moreover, it was observed that linearising the equations of motion around just one equilibrium point, results in instability, if the ball has a greater distance of $\pm 20\,cm$ to its equilibrium point. As these design handicaps, restrict the minimum performance of the linearised controllers, it can be concluded that their robust performance is limited, too.

In contrast the controller designed with methods from reinforcement learning does not suffer from linearisation and discretization errors, because it is able to directly learn non-linear and discrete behaviour. For this reason the neuronal network trained with policy proximal optimization was able to control the balls for different radii and rolling friction coefficients faster. Additionally, the task-space actuator saturation limits were not exceeded. However, as the cur-

rent reward function does not consider overshoot, the controller designed with methods from reinforcement learning overshot more compared to the fixed-structure $\mathscr{H}_2$ controller. Comparing the scalability of both designs (classical and RL) it was concluded, that a neuronal network can more easily be extended (by adding additional hidden layers or nodes).
It can be summarized, that designing a robust controller with methods from reinforcement learning is a competitive approach compared to methods from classical control theory. As neuronal networks are more easily to scale and able to learn non-linear and discrete behaviour, they have the potential to significantly outperform static robust controllers designed with methods from classical control theory.

However, both designs revealed that the rolling friction coefficient impacts the balancing behaviour most. In particular, the maximum rolling friction coefficient and the minimum radius was the most problematic case. Applying the controllers to the real robot imposed a series of challenges. It was investigated that the measurement of the pose of the end-effector as well as the position and velocity of the ball had a minor impact. The same applies to the time delay between the measurements of the robot and the ball. However, the impact of the deadband of the motors, that causes non-linear artefacts of the velocity of the inclination angles of the plate $(\dot{\alpha}, \dot{\beta})$, is more critical.
Nevertheless, the reality gap could be crossed by a control circuit that includes the model. Thereby, the noisy measurement data is not just double integrated but forced to follow the model. On the real robot the LQR-I controller was able to robustly balance balls of different radii and rolling friction coefficients.

In the future, the fixed-structure $\mathscr{H}_2$ controller as well as the trained neuronal network should be tested on the real robot. In order to improve the performance of the classical controllers, it should be evaluated to discretize and linearise the equations of motion of the Ball-on-Plate system around a grid of equilibrium points. For each equilibrium point a fixed-structure $\mathscr{H}_2$ robust controller can be designed. Combining these controllers by using the gain-scheduling method non-linear effects can be decreased. The robust controller designed with methods from reinforcement learning can be further improved by extending the reward function to consider step response requirements (such as overshoot). The robustness of both controllers can be further improved by including not only parametric uncertainties but also dynamic uncertainties (such as time-delays, non-linear motor behaviour) in the design. Thereby the reality gap can be narrowed. Regarding the real robot, a different initial position that provides a higher operational area should be chosen and it should be considered to use motors with a higher maximum velocity speed to achieve a higher performance.

A Equations of motion of the coupled Ball-on-Plate system

The equations of motion of the cBoP system derived by Matlab are given as:

$$F_X = (m_B+m_P)\ddot{X}+((m_B(2rcos(\alpha)cos(\beta)-2sin(\beta)x+2cos(\beta)sin(\alpha)y))/2)\ddot{\beta}+m_Bcos(\beta)\ddot{x}+m_Bsin(\alpha)sin(\beta)\ddot{y}-(m_B(2(sin(\beta)\dot{x}+ cos(\beta)x\dot{\beta}-cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})\dot{\beta}+2sin(\beta)\dot{x}\dot{\beta}- 2cos(\alpha)sin(\beta)\dot{\alpha}\dot{y}-2cos(\beta)sin(\alpha)\dot{y}\dot{\beta}))/2 \tag{A.1}$$

$$F_Y = (m_B+m_P)\ddot{Y}+(-(m_B(2rcos(\alpha)cos(\beta)-2sin(\beta)x+2cos(\beta)sin(\alpha)y))/2)\ddot{\alpha}+m_Bcos(\alpha)\ddot{y}+(m_B(2(sin(\beta)\dot{x}+cos(\beta)x\dot{\beta}- cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})\dot{\alpha}-2sin(\alpha)\dot{y}\dot{\alpha}))/2 \tag{A.2}$$

$$F_Z = (m_B+m_P)\ddot{Z}+(-(m_B(2rsin(\alpha)-2cos(\alpha)y))/2)\ddot{\alpha}+(-(m_B(2cos(\beta)x+2rcos(\alpha)sin(\beta)+2sin(\alpha)sin(\beta)y))/2)\ddot{\beta}+(-m_Bsin(\beta))\ddot{x}+m_Bcos(\beta)sin(\alpha)\ddot{y}+gm_B+gm_P-(m_B(2\dot{\beta}(cos(\beta)\dot{x}-sin(\beta)x\dot{\beta}+sin(\alpha)sin(\beta)\dot{y}-rsin(\alpha)sin(\beta)\dot{\alpha}+cos(\alpha)sin(\beta)y\dot{\alpha}+cos(\beta)sin(\alpha)y\dot{\beta}+rcos(\alpha)cos(\beta)\dot{\beta})+2\dot{\alpha}(sin(\alpha)y\dot{\alpha}-cos(\alpha)\dot{y}+rcos(\alpha)\dot{\alpha})+2cos(\beta)\dot{\beta}\dot{x}-2cos(\alpha)cos(\beta)\dot{\alpha}\dot{y}+2sin(\alpha)sin(\beta)\dot{\beta}\dot{y}))/2 \tag{A.3}$$

$$\tau_x = (-(m_B(2rcos(\alpha)cos(\beta)-2sin(\beta)x+2cos(\beta)sin(\alpha)y))/2)\ddot{Y}+(-(m_B(2rsin(\alpha)-2cos(\alpha)y))/2)\ddot{Z}+(I_{Pxx}+j_b+m_B(r^2+x^2)+(m_B(2(rsin(\alpha)-cos(\alpha)y)^2+2(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)^2))/2)\ddot{\alpha}+m_B(rsin(\alpha)-cos(\alpha)y)(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)\ddot{\beta}+m_Bsin(\beta)(rsin(\alpha)-cos(\alpha)y)\ddot{x}+(-j_b/r-(m_B(2cos(\alpha)(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+2cos(\beta)sin(\alpha)(rsin(\alpha)-cos(\alpha)y)))/2)\ddot{y}+(m_B(2(cos(\alpha)\dot{y}-\dot{\alpha}(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+\dot{Y})(sin(\beta)\dot{x}+cos(\beta)x\dot{\beta}-cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})-2((sin(\beta)\dot{x}+cos(\beta)x\dot{\beta}-cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})\dot{\alpha}-sin(\alpha)\dot{y}\dot{\alpha})(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+2(sin(\alpha)y\dot{\alpha}-cos(\alpha)\dot{y}+rcos(\alpha)\dot{\alpha})((rsin(\alpha)-cos(\alpha)y)\dot{\alpha}+(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)\dot{\beta}+sin(\beta)\dot{x}-\dot{Z}-cos(\beta)sin(\alpha)\dot{y})+2(rsin(\alpha)-cos(\alpha)y)(\dot{\beta}(cos(\beta)\dot{x}-sin(\beta)x\dot{\beta}+sin(\alpha)sin(\beta)\dot{y}-rsin(\alpha)sin(\beta)\dot{\alpha}+cos(\alpha)sin(\beta)y\dot{\alpha}+cos(\beta)sin(\alpha)y\dot{\beta}+rcos(\alpha)cos(\beta)\dot{\beta})+\dot{\alpha}(sin(\alpha)y\dot{\alpha}-cos(\alpha)\dot{y}+rcos(\alpha)\dot{\alpha})+cos(\beta)\dot{\beta}\dot{x}-cos(\alpha)cos(\beta)\dot{\alpha}\dot{y}+sin(\alpha)sin(\beta)\dot{\beta}\dot{y})))/2-gm_B(rcos(\beta)sin(\alpha)-cos(\alpha)cos(\beta)y)+2m_Bx\dot{\alpha}\dot{x} \tag{A.4}$$

$$\tau_y = ((m_B(2rcos(\alpha)cos(\beta)-2sin(\beta)x+2cos(\beta)sin(\alpha)y))/2)\ddot{X}+(-(m_B(2cos(\beta)x+2rcos(\alpha)sin(\beta)+2sin(\alpha)sin(\beta)y))/2)\ddot{Z}+m_B(rsin(\alpha)-cos(\alpha)y)(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)\ddot{\alpha}+(I_{Pyy}+j_b+m_B(r^2+y^2)+(m_B(2(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)^2+2(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)^2))/2)\ddot{\beta}+((m_B(2cos(\beta)(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+2sin(\beta)(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)))/2+j_b/r)\ddot{x}+(-(m_B(2cos(\beta)sin(\alpha)(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)-2sin(\alpha)sin(\beta)(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)))/2)\ddot{y}+(m_B(2(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)(\dot{\beta}(cos(\beta)\dot{x}-sin(\beta)x\dot{\beta}+sin(\alpha)sin(\beta)\dot{y}-rsin(\alpha)sin(\beta)\dot{\alpha}+cos(\alpha)sin(\beta)y\dot{\alpha}+cos(\beta)sin(\alpha)y\dot{\beta}+rcos(\alpha)cos(\beta)\dot{\beta})+\dot{\alpha}(sin(\alpha)y\dot{\alpha}-cos(\alpha)\dot{y}+rcos(\alpha)\dot{\alpha})+cos(\beta)\dot{\beta}\dot{x}-cos(\alpha)cos(\beta)\dot{\alpha}\dot{y}+sin(\alpha)sin(\beta)\dot{\beta}\dot{y})-2(\dot{\beta}(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+cos(\beta)\dot{x}+\dot{X}+sin(\alpha)sin(\beta)\dot{y})(sin(\beta)\dot{x}+cos(\beta)x\dot{\beta}-cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})-2(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)((sin(\beta)\dot{x}+cos(\beta)x\dot{\beta}-cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})\dot{\beta}+sin(\beta)\dot{x}\dot{\beta}-cos(\alpha)sin(\beta)\dot{\alpha}\dot{y}-cos(\beta)sin(\alpha)\dot{y}\dot{\beta})+2((rsin(\alpha)-cos(\alpha)y)\dot{\alpha}+(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)\dot{\beta}+sin(\beta)\dot{x}-\dot{Z}-cos(\beta)sin(\alpha)\dot{y})(cos(\beta)\dot{x}-sin(\beta)x\dot{\beta}+sin(\alpha)sin(\beta)\dot{y}-rsin(\alpha)sin(\beta)\dot{\alpha}+cos(\alpha)sin(\beta)y\dot{\alpha}+cos(\beta)sin(\alpha)y\dot{\beta}+rcos(\alpha)cos(\beta)\dot{\beta})))/2-gm_B(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)+2m_By\dot{\beta}\dot{y} \tag{A.5}$$

$$0 = m_Bcos(\beta)\ddot{X}+(-m_Bsin(\beta))\ddot{Z}+m_Bsin(\beta)(rsin(\alpha)-cos(\alpha)y)\ddot{\alpha}+((m_B(2cos(\beta)(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+2sin(\beta)(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)))/2+j_b/r)\ddot{\beta}+((m_B(2cos(\beta)^2+2sin(\beta)^2))/2+j_b/r^2)\ddot{x}+(m_B(2sin(\beta)(\dot{\beta}(cos(\beta)\dot{x}-sin(\beta)x\dot{\beta}+sin(\alpha)sin(\beta)\dot{y}-rsin(\alpha)sin(\beta)\dot{\alpha}+cos(\alpha)sin(\beta)y\dot{\alpha}+cos(\beta)sin(\alpha)y\dot{\beta}+rcos(\alpha)cos(\beta)\dot{\beta})+\dot{\alpha}(sin(\alpha)y\dot{\alpha}-cos(\alpha)\dot{y}+rcos(\alpha)\dot{\alpha})+cos(\beta)\dot{\beta}\dot{x}-cos(\alpha)cos(\beta)\dot{\alpha}\dot{y}+sin(\alpha)sin(\beta)\dot{\beta}\dot{y})-2cos(\beta)((sin(\beta)\dot{x}+cos(\beta)x\dot{\beta}-cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})\dot{\beta}+sin(\beta)\dot{x}\dot{\beta}-cos(\alpha)sin(\beta)\dot{\alpha}\dot{y}-cos(\beta)sin(\alpha)\dot{y}\dot{\beta})-2sin(\beta)\dot{\beta}(\dot{\beta}(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+cos(\beta)\dot{x}+\dot{X}+sin(\alpha)sin(\beta)\dot{y})+2cos(\beta)\dot{\beta}((rsin(\alpha)-cos(\alpha)y)\dot{\alpha}+(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)\dot{\beta}+sin(\beta)\dot{x}-\dot{Z}-cos(\beta)sin(\alpha)\dot{y})))/2+k_A\dot{x}-gm_Bsin(\beta)+(k_Racos((cos(\alpha)cos(\beta)))/(1-sin(\alpha)^2sin(\beta)^2)^(1/2))\dot{x})/(2(\dot{x}^2+\dot{y}^2)^(1/2)) \tag{A.6}$$

$$0 = m_B sin(\alpha)sin(\beta)\ddot{X} + m_B cos(\alpha)\ddot{Y} + m_B cos(\beta)sin(\alpha)\ddot{Z} + (-j_b/r - (m_B(2cos(\alpha)(rcos(\alpha)cos(\beta) - sin(\beta)x + cos(\beta)sin(\alpha)y) + \quad (A.7)$$
$$2cos(\beta)sin(\alpha)(rsin(\alpha) - cos(\alpha)y)))/2)\ddot{\alpha} + (-(m_B(2cos(\beta)sin(\alpha)(cos(\beta)x + rcos(\alpha)sin(\beta) + sin(\alpha)sin(\beta)y) - 2sin(\alpha)sin(\beta)(rcos(\alpha)cos(\beta) -$$
$$sin(\beta)x + cos(\beta)sin(\alpha)y)))/2)\ddot{\beta} + ((m_B(2cos(\alpha)^2 + 2cos(\beta)^2 sin(\alpha)^2 + 2sin(\alpha)^2 sin(\beta)^2))/2 + j_b/r^2)\ddot{y} + k_A\dot{y} + (m_B(2cos(\alpha)((sin(\beta)\dot{x} +$$
$$cos(\beta)x\dot{\beta} - cos(\beta)sin(\alpha)\dot{y} + rcos(\beta)sin(\alpha)\dot{\alpha} + rcos(\alpha)sin(\beta)\dot{\beta} - cos(\alpha)cos(\beta)y\dot{\alpha} + sin(\alpha)sin(\beta)y\dot{\beta})\dot{\alpha} - sin(\alpha)\dot{y}\dot{\alpha}) -$$
$$2cos(\beta)sin(\alpha)(\dot{\beta}(cos(\beta)\dot{x} - sin(\beta)x\dot{\beta} + sin(\alpha)sin(\beta)\dot{y} - rsin(\alpha)sin(\beta)\dot{\alpha} + cos(\alpha)sin(\beta)y\dot{\alpha} + cos(\beta)sin(\alpha)y\dot{\beta} + rcos(\alpha)cos(\beta)\dot{\beta}) +$$
$$\dot{\alpha}(sin(\alpha)y\dot{\alpha} - cos(\alpha)\dot{y} + rcos(\alpha)\dot{\alpha}) + cos(\beta)\dot{\beta}x - cos(\alpha)cos(\beta)\dot{\alpha}\dot{y} + sin(\alpha)sin(\beta)\dot{\beta}\dot{y}) - 2sin(\alpha)sin(\beta)((sin(\beta)\dot{x} + cos(\beta)x\dot{\beta} -$$
$$cos(\beta)sin(\alpha)\dot{y} + rcos(\beta)sin(\alpha)\dot{\alpha} + rcos(\alpha)sin(\beta)\dot{\beta} - cos(\alpha)cos(\beta)y\dot{\alpha} + sin(\alpha)sin(\beta)y\dot{\beta})\dot{\beta} + sin(\beta)\dot{x}\dot{\beta} - cos(\alpha)sin(\beta)\dot{\alpha}\dot{y} -$$
$$cos(\beta)sin(\alpha)\dot{y}\dot{\beta}) - 2sin(\alpha)\dot{\alpha}(cos(\alpha)\dot{y} - \dot{\alpha}(rcos(\alpha)cos(\beta) - sin(\beta)x + cos(\beta)sin(\alpha)y) + \dot{Y}) - 2cos(\alpha)cos(\beta)\dot{\alpha}((rsin(\alpha) -$$
$$cos(\alpha)y)\dot{\alpha} + (cos(\beta)x + rcos(\alpha)sin(\beta) + sin(\alpha)sin(\beta)y)\dot{\beta} + sin(\beta)\dot{x} - \dot{Z} - cos(\beta)sin(\alpha)\dot{y}) + 2sin(\alpha)sin(\beta)\dot{\beta}((rsin(\alpha) -$$
$$cos(\alpha)y)\dot{\alpha} + (cos(\beta)x + rcos(\alpha)sin(\beta) + sin(\alpha)sin(\beta)y)\dot{\beta} + sin(\beta)\dot{x} - \dot{Z} - cos(\beta)sin(\alpha)\dot{y}) + 2cos(\alpha)sin(\beta)\dot{\alpha}(\dot{\beta}(rcos(\alpha)cos(\beta) -$$
$$sin(\beta)x + cos(\beta)sin(\alpha)y) + cos(\beta)\dot{x} + \dot{X} + sin(\alpha)sin(\beta)\dot{y}) + 2cos(\beta)sin(\alpha)\dot{\beta}(\dot{\beta}(rcos(\alpha)cos(\beta) - sin(\beta)x + cos(\beta)sin(\alpha)y) +$$
$$cos(\beta)\dot{x} + \dot{X} + sin(\alpha)sin(\beta)\dot{y})))/2 + (k_R acos((cos(\alpha)cos(\beta))/(1 - sin(\alpha)^2 sin(\beta)^2)^{(}1/2))\dot{y})/(2(\dot{x}^2 + \dot{y}^2)^{(}1/2)) + g m_B cos(\beta)sin(\alpha)$$

The solution of (A.6) and (A.7) is given as

$$\ddot{x} = -(((m_B(2cos(\beta)(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+2sin(\beta)(cos(\beta)x+ \quad (A.8)$$
$$rcos(\alpha)sin(\beta) + sin(\alpha)sin(\beta)y)))/2 + j_b/r)\ddot{\beta} + (m_B(2sin(\beta)(\dot{\beta}(cos(\beta)\dot{x} -$$
$$sin(\beta)x\dot{\beta}+sin(\alpha)sin(\beta)\dot{y}-rsin(\alpha)sin(\beta)\dot{\alpha}+cos(\alpha)sin(\beta)y\dot{\alpha}+cos(\beta)sin(\alpha)y\dot{\beta}+$$
$$rcos(\alpha)cos(\beta)\dot{\beta})+\dot{\alpha}(sin(\alpha)y\dot{\alpha}-cos(\alpha)\dot{y}+rcos(\alpha)\dot{\alpha})+cos(\beta)\dot{\beta}\dot{x}-cos(\alpha)cos(\beta)\dot{\alpha}\dot{y}+$$
$$sin(\alpha)sin(\beta)\dot{\beta}\dot{y})-2cos(\beta)(\dot{\beta}(sin(\beta)\dot{x}+cos(\beta)x\dot{\beta}-cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+$$
$$rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})+sin(\beta)\dot{\beta}\dot{x}-cos(\alpha)sin(\beta)\dot{\alpha}\dot{y}-$$
$$cos(\beta)sin(\alpha)\dot{\beta}\dot{y}) - 2sin(\beta)\dot{\beta}(\dot{\beta}(rcos(\alpha)cos(\beta) - sin(\beta)x + cos(\beta)sin(\alpha)y) +$$
$$cos(\beta)\dot{x} + \dot{X} + sin(\alpha)sin(\beta)\dot{y}) + 2cos(\beta)\dot{\beta}(\dot{\alpha}(rsin(\alpha)-cos(\alpha)y)+\dot{\beta}(cos(\beta)x+$$
$$rcos(\alpha)sin(\beta) + sin(\alpha)sin(\beta)y) + sin(\beta)\dot{x} - \dot{Z} - cos(\beta)sin(\alpha)\dot{y})))/2 + k_A\dot{x} +$$
$$m_Bcos(\beta)\ddot{X} - m_Bsin(\beta)\ddot{Z} - gm_Bsin(\beta) + m_Bsin(\beta)(rsin(\alpha) - cos(\alpha)y)\ddot{\alpha} +$$
$$(k_Racos((cos(\alpha)cos(\beta))/(1-sin(\alpha)^2sin(\beta)^2)^(1/2))\dot{x})/(2(\dot{x}^2+\dot{y}^2)^(1/2)))/((m_B(2cos(\beta)^2+$$
$$2sin(\beta)^2))/2 + j_b/r^2) ,$$

$$\ddot{y} = -(k_A\dot{y} - (j_b/r + (m_B(2cos(\alpha)(rcos(\alpha)cos(\beta) - sin(\beta)x + cos(\beta)sin(\alpha)y) + \quad (A.9)$$
$$2cos(\beta)sin(\alpha)(rsin(\alpha)-cos(\alpha)y)))/2)\ddot{\alpha}+(m_B(2cos(\alpha)(\dot{\alpha}(sin(\beta)\dot{x}+cos(\beta)x\dot{\beta}-$$
$$cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+sin(\alpha)sin(\beta)y\dot{\beta})-$$
$$sin(\alpha)\dot{\alpha}\dot{y})-2cos(\beta)sin(\alpha)(\dot{\beta}(cos(\beta)\dot{x}-sin(\beta)x\dot{\beta}+sin(\alpha)sin(\beta)\dot{y}-rsin(\alpha)sin(\beta)\dot{\alpha}+$$
$$cos(\alpha)sin(\beta)y\dot{\alpha}+cos(\beta)sin(\alpha)y\dot{\beta}+rcos(\alpha)cos(\beta)\dot{\beta})+\dot{\alpha}(sin(\alpha)y\dot{\alpha}-cos(\alpha)\dot{y}+$$
$$rcos(\alpha)\dot{\alpha})+cos(\beta)\dot{\beta}\dot{x}-cos(\alpha)cos(\beta)\dot{\alpha}\dot{y}+sin(\alpha)sin(\beta)\dot{\beta}\dot{y})-2sin(\alpha)sin(\beta)(\dot{\beta}(sin(\beta)\dot{x}+$$
$$cos(\beta)x\dot{\beta}-cos(\beta)sin(\alpha)\dot{y}+rcos(\beta)sin(\alpha)\dot{\alpha}+rcos(\alpha)sin(\beta)\dot{\beta}-cos(\alpha)cos(\beta)y\dot{\alpha}+$$
$$sin(\alpha)sin(\beta)y\dot{\beta})+sin(\beta)\dot{\beta}\dot{x}-cos(\alpha)sin(\beta)\dot{\alpha}\dot{y}-cos(\beta)sin(\alpha)\dot{\beta}\dot{y})-2sin(\alpha)\dot{\alpha}(cos(\alpha)\dot{y}-$$
$$\dot{\alpha}(rcos(\alpha)cos(\beta)-sin(\beta)x+cos(\beta)sin(\alpha)y)+\dot{Y})-2cos(\alpha)cos(\beta)\dot{\alpha}(\dot{\alpha}(rsin(\alpha)-$$
$$cos(\alpha)y) + \dot{\beta}(cos(\beta)x + rcos(\alpha)sin(\beta) + sin(\alpha)sin(\beta)y) + sin(\beta)\dot{x} - \dot{Z} -$$
$$cos(\beta)sin(\alpha)\dot{y})+2sin(\alpha)sin(\beta)\dot{\beta}(\dot{\alpha}(rsin(\alpha)-cos(\alpha)y)+\dot{\beta}(cos(\beta)x+rcos(\alpha)sin(\beta)+$$
$$sin(\alpha)sin(\beta)y)+sin(\beta)\dot{x}-\dot{Z}-cos(\beta)sin(\alpha)\dot{y})+2cos(\alpha)sin(\beta)\dot{\alpha}(\dot{\beta}(rcos(\alpha)cos(\beta)-$$
$$sin(\beta)x+cos(\beta)sin(\alpha)y)+cos(\beta)\dot{x}+\dot{X}+sin(\alpha)sin(\beta)\dot{y})+2cos(\beta)sin(\alpha)\dot{\beta}(\dot{\beta}(rcos(\alpha)cos(\beta)-$$
$$sin(\beta)x + cos(\beta)sin(\alpha)y) + cos(\beta)\dot{x} + \dot{X} + sin(\alpha)sin(\beta)\dot{y})))/2 + m_Bcos(\alpha)\ddot{Y} -$$
$$(m_B(2cos(\beta)sin(\alpha)(cos(\beta)x+rcos(\alpha)sin(\beta)+sin(\alpha)sin(\beta)y)-2sin(\alpha)sin(\beta)(rcos(\alpha)cos(\beta)-$$
$$sin(\beta)x+cos(\beta)sin(\alpha)y))\ddot{\beta})/2+(k_Racos((cos(\alpha)cos(\beta))/(1-sin(\alpha)^2sin(\beta)^2)^(1/2))\dot{y})/(2(\dot{x}^2+$$
$$\dot{y}^2)^(1/2))+m_Bcos(\beta)sin(\alpha)\ddot{Z}+m_Bsin(\alpha)sin(\beta)\ddot{X}+gm_Bcos(\beta)sin(\alpha))/((m_B(2cos(\alpha)^2+$$
$$2cos(\beta)^2sin(\alpha)^2 + 2sin(\alpha)^2sin(\beta)^2))/2 + j_b/r^2) .$$

B Parameters and constants

The inertia is calculated as follows:

- if the Ball is solid: $j_b = \frac{2}{5} m_b r^2$
- if the ball is hollow $j_b = \frac{2}{3} m_b r^2$.

Ball constants

Number	Ball	Radius of ball	Mass of ball	Inertia	Solid/Hollow
1	Small basket ball	45.5 mm	55 g	75.9092 kg/mm^2	hollow
2	Tennis ball	33.4 mm	57 g	42.391 kg/mm^2	hollow
3	Pool ball	29.8 mm	181 g	64.294 kg/mm^2	solid
4	Table tennis ball	21.2 mm	2 g	0.59925 kg/mm^2	hollow
5	Golf ball	20.5 mm	6 g	1.681 kg/mm^2	hollow
6	Hand ball	77.99 mm	165 g	669.068 kg/mm^2	hollow

Plate parameters

Symbol	Description	gazebo plexiglass plate	real wood plate
m_P	Mass of the Plate	2.975 kg	0.91 kg
$I_{P,xx}$	Inertia of the Plate	0.062 kg/m^2	0.018961 kg/m^2
$I_{P,yy}$	Inertia of the Plate	0.062 kg/m^2	0.018961 kg/m^2
l	Length of the plate	0.5 m	0.5 m
w	Width of the plate	0.5 m	0.5 m
h	Height of the plate	0.01 m	0.006 m

Vortex parameters

Table B.1 shows the most relevant Vortex parameters that are used in the Rcs Simulation in Chapter 6. For a detailed description of the parameters see *Vortex Studio SDK Classes Documentation*, *Vortex contact parameters description* and [4]. The value refers to the parameter value that is used in the Rcs simulation for the material of the plate and the ball. As the plate and the ball are assumed to be in contact all the time, both should have the same contact material properties.

Parameter	Description	Value
friction model	friction model along this axis	ScaledBoxFast
friction coefficient(μ_{fc})	The static friction (stiction) coefficient μ_H is calculated by: $\mu_H = \mu_{fc} \cdot k$. This coefficient is equal to the kinetic friction coefficient as defined in the Coulomb friction law. Note that this coefficient is used for the Vortex ScaledBox and ScaledBoxFast friction model.	0.2
rolling friction coefficient	The rolling friction coefficient used in Vortex is the actual dimensionless rolling friction coefficient μ_r multiplied with the curvature radius of the contacting object which was in our case the radius of the ball.	0 - $r \cdot \mu_r$
static friction scale (k)	The static friction scale parameter k is used to obtain the stiction by multiplying k with the friction coefficient μ_{fc}. For the BoP system it is assumed that the ball is not sliding. Thus there is no kinetic friction. This means the kinetic and static friction is the same. For this reason k was set to 1.0.	1.0
slip	When greater than 0.0, this value adds viscosity (in s/kg). If set to 0.0, the friction is dry.	0.0
integrated slip displacement	Enable or disable slip displacement detection	true
compliance	The reciprocal of Stiffness (material softness) (in m/N).	0.0001
damping	Value at the contact point to get the system to reach equilibrium (in kg/s). As a rule of thumb, Damping should be ten times Stiffness.	100000

Table B.1.: Vortex parameters that are used in Simulation

C Additional results

C.1 LQR controller

Figure C.1 shows the balancing behaviour of another LQR controller that was designed with $\mu_r = \frac{\mu_{r,max}+\mu_{r,min}}{2}$.

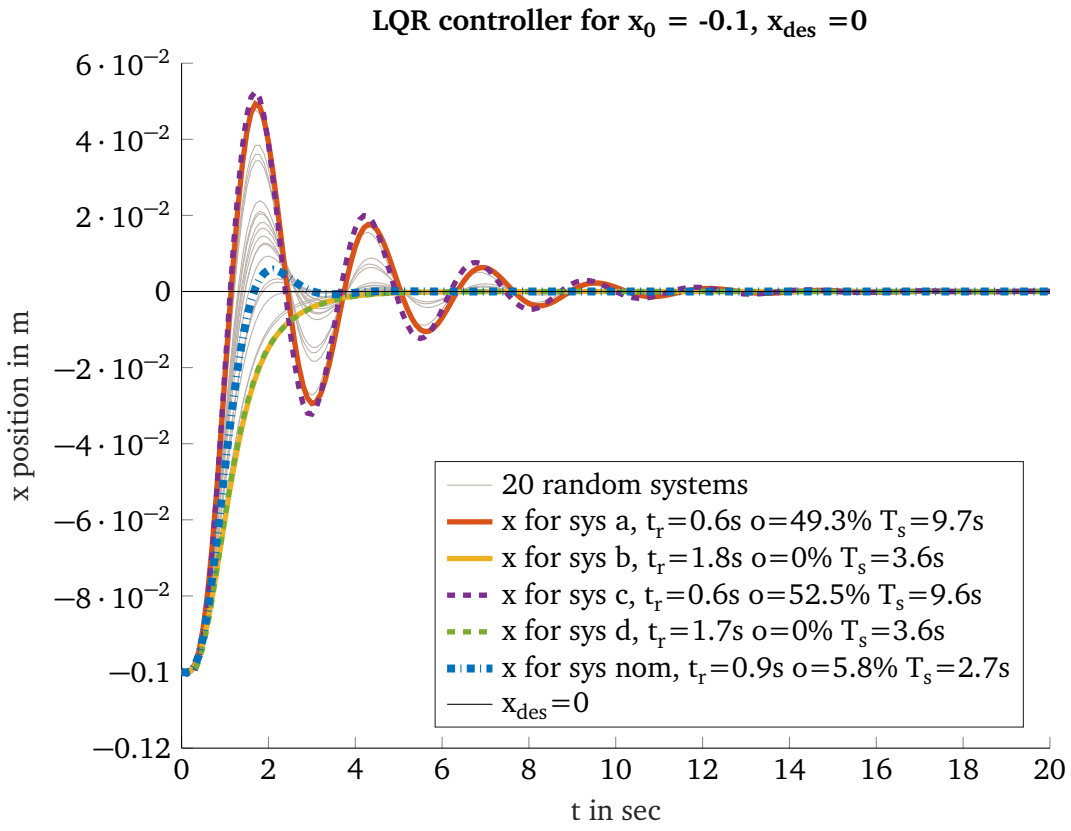


Figure C.1.: Balancing behaviour of LQR controller that was designed with $\mu_r = \frac{\mu_{r,max}+\mu_{r,min}}{2}$ for an initial position of the ball of $x_0 = -0.1$

C.2 MMPP controller

In order to compare discretization errors, the MMPP controller was once sampled with 100 Hz (see Figure C.2) and once with 1 kHz (see Figure C.4). When comparing the joint limits of both it is conspicuous that the joint limits of the MMPP controller sampled with 1 kHz are exceeded much less.

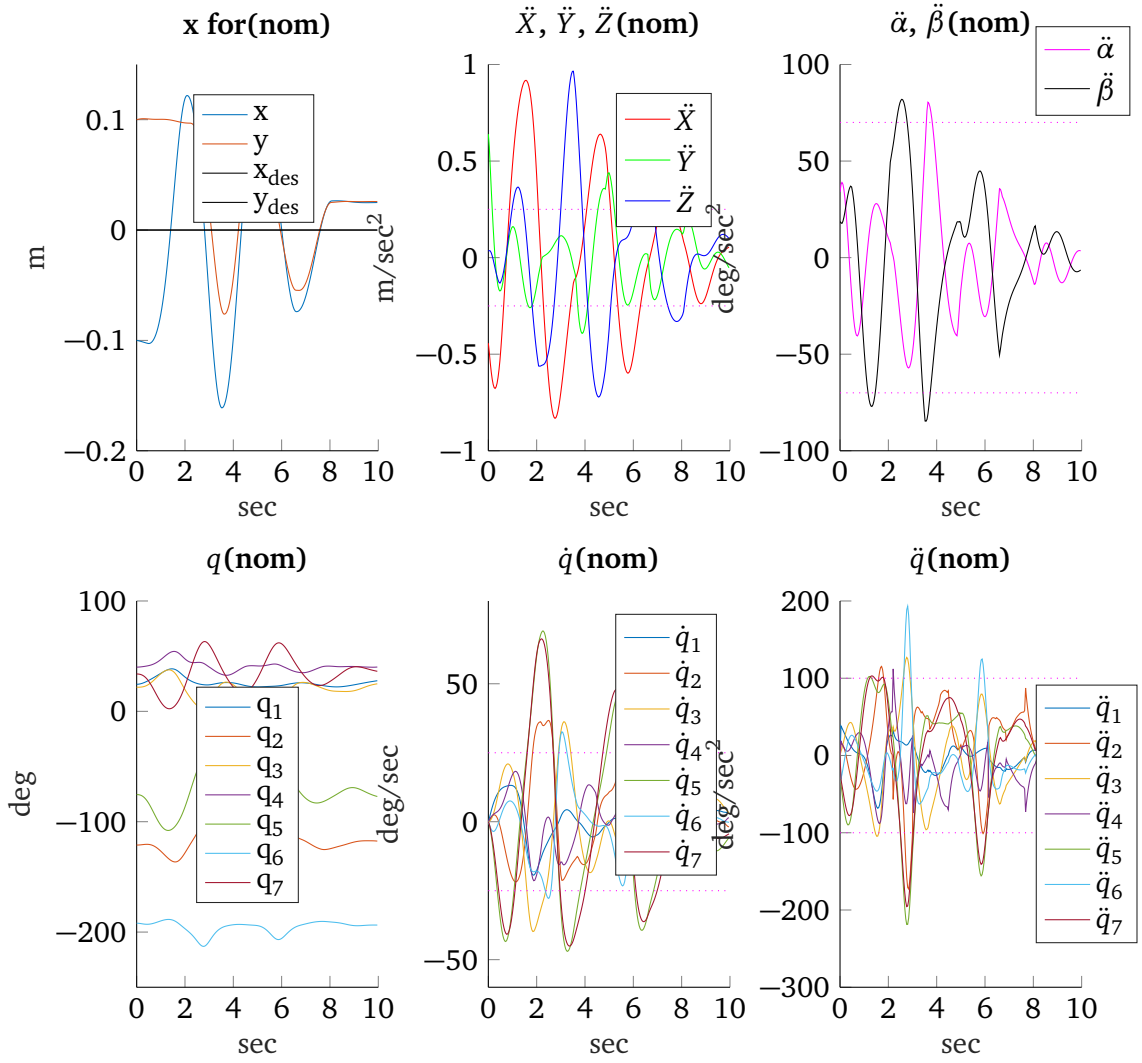


Figure C.2.: Rcs Simulation of MMPP controller sampled with 100 Hz of the nominal system

C.3 Fixed-structure $\mathscr{H}_2$ controller

Figure C.6 shows the redesigned $\mathscr{H}_2$ controller. The tuning goals for this controller are given in the script *h2_systune_re.m* (see Appendix E).

C.4 Linearisation problems

In an additional experiment, the Ball was placed at the initial position of $x_0 = -0.2, y_0 = 0.2$. The y-trajectory for system (c) of this experiment is illustrated in Figure C.5. It was observed, that if the ball has a greater distance to the origin, the ball is accelerated more in order to be balanced. These higher dynamics cause a system response with high overshooting. For

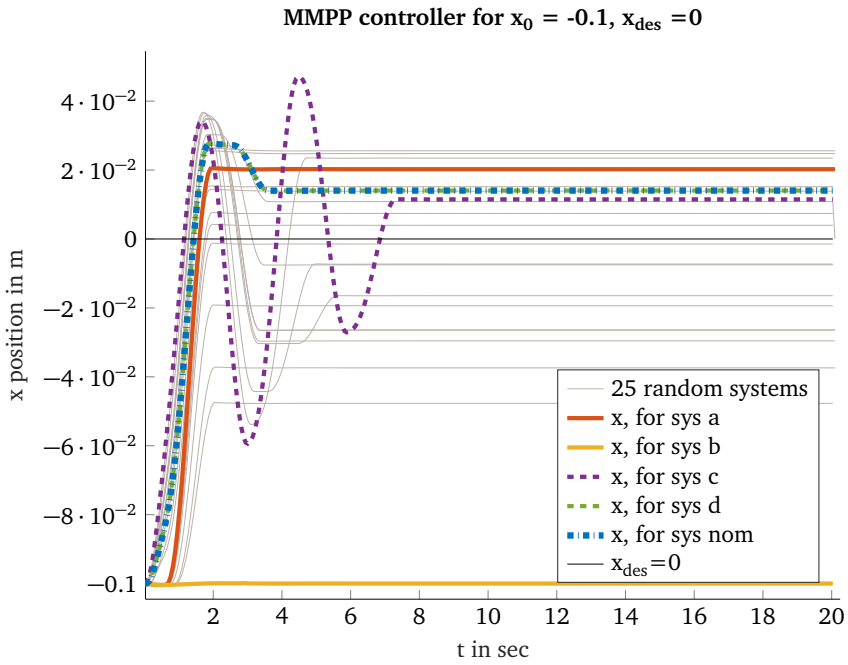


Figure C.3.: Rcs Simulation of MMPP controller sampled with 1 kHz

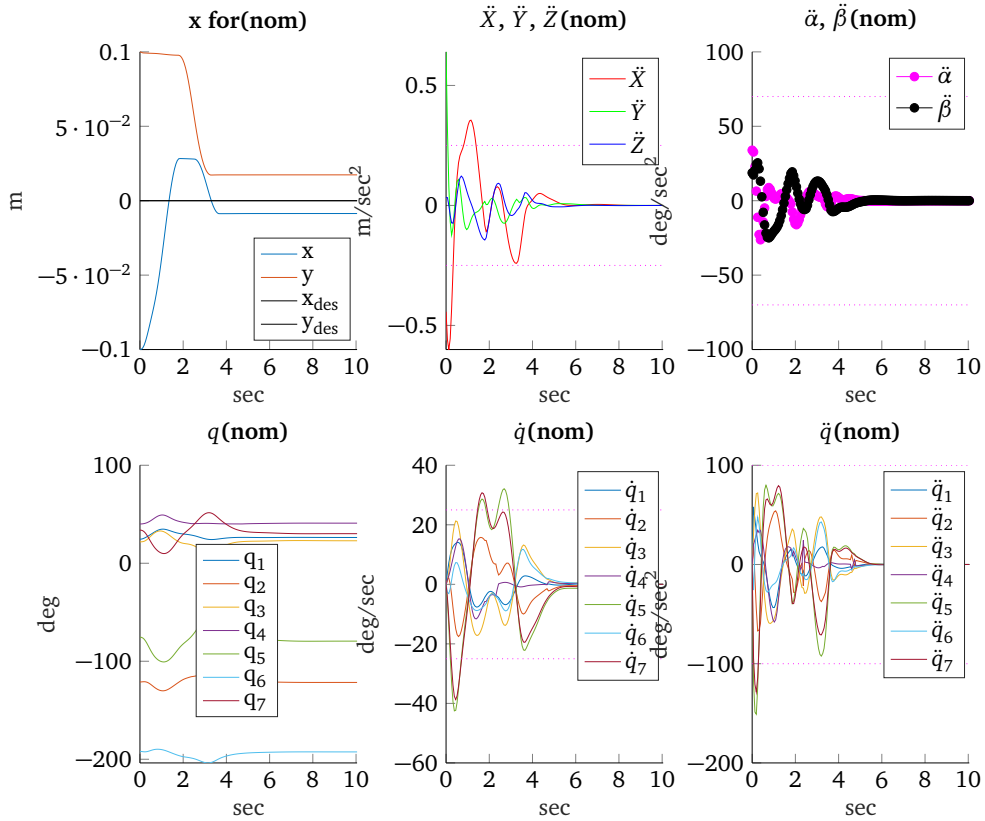


Figure C.4.: Rcs Simulation of MMPP controller sampled with 1 kHz for nominal system joint sates

this reason, high dynamics should be prevented in order to decrease non-linear effects. The experiment was executed with the fixed-structure $\mathscr{H}_2$-I controller.

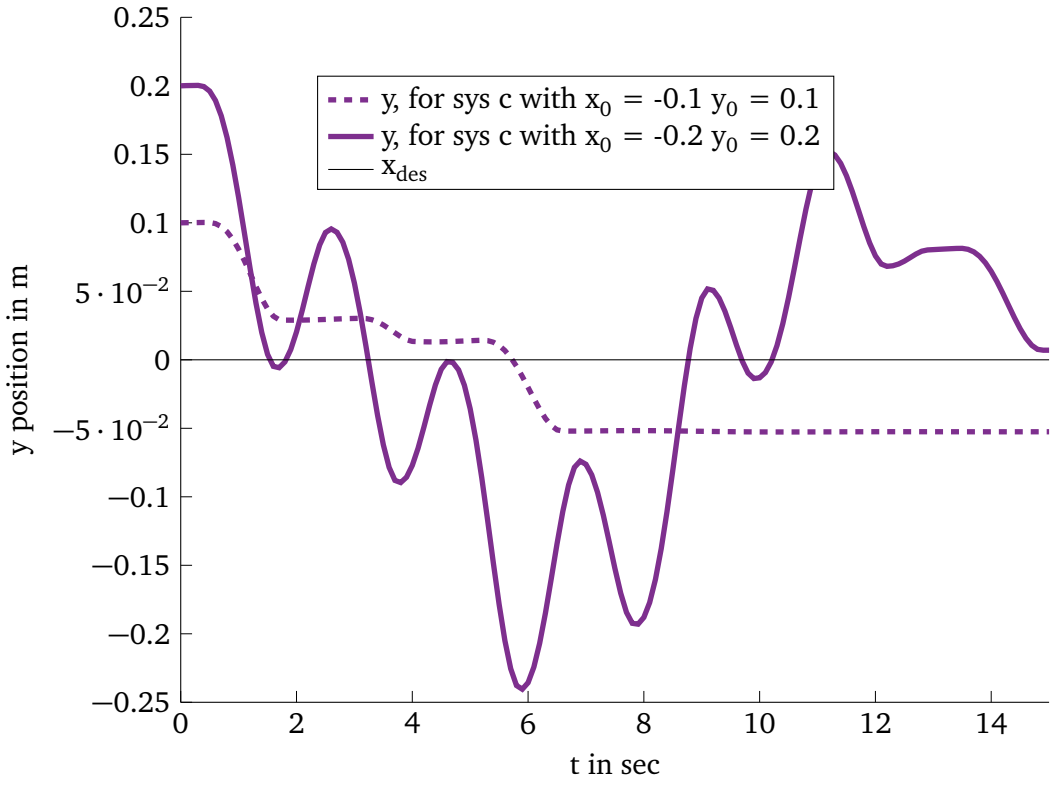


Figure C.5.: Balancing behaviour of the fixed-structure $\mathscr{H}_2$-I controller for different initial positions

C.5 LQR controller (real robot)

A different LQR controller with

$$\mathbf{Q} = \text{diag}(0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.5\ \ 0.5\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 1\ \ 1)$$
$$\mathbf{R} = \text{diag}(1\ \ 1\ \ 1\ \ 1\ \ 1)\,.$$

is analysed. Allowing the position of the end-effector to move in a wider range has the advantage of not using the noisy acceleration of the measurements of the angles $\ddot{\alpha}, \ddot{\beta}$. This positive effect results in a much smoother balancing behaviour (the ball is balancing in a certain band of $\pm 3cm$ around the origin[1]) as depicted in Figure C.7. As the position of the end-effector is very limited, the robot is more likely to reach its joint limits. Another disadvantage is that in case of a higher rolling-friction coefficient (red line) the ball is not rolling at all.

C.6 Plate imperfections

The angles of inclination of the real robot could not be aligned perfectly to be 0.0°. The initial position of the plate was always a little bit inclined by $\alpha \approx 1.8°, \beta \approx 0.2°$. This fact was evaluated by controlling the position of the ball in a square. Figure C.8 shows the x-position of

[1] The reason that this band is not perfectly around the origin is due to plate imperfections (screws etc.).

the ball plotted over the y-position of the ball. One can observe that the x-position in simulation is often on the left side of the x-position of the real robot. Figure C.8 shows the x, and y position of the ball plotted over the time. The differences of the simulation and the real robot are caused by a wrong initialization position of the plate, imperfections of the shape of the plate and the ball as well as the noisy measurement data.

For this experiment a LQR controller with

$$\mathbf{Q} = \text{diag}(0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 5.0\ \ 5.0\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 0.1\ \ 2\ \ 2)$$
$$\mathbf{R} = \text{diag}(5\ \ 5\ \ 5\ \ 1\ \ 1)$$

was utilized.

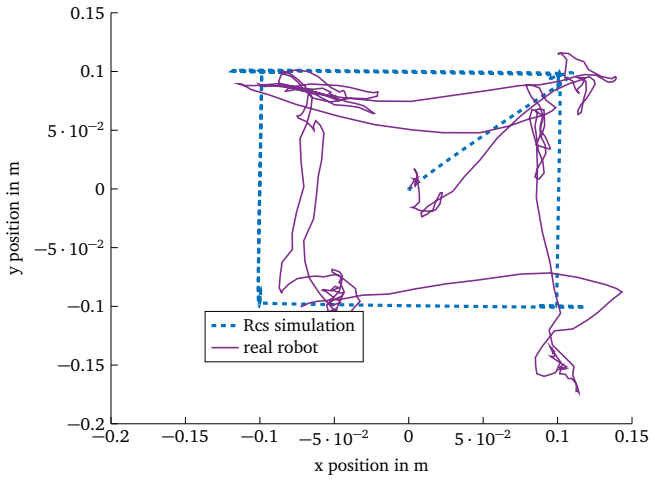


Figure C.8.: Controlling the position of the plate to follow a square: The x-position is plotted over the y-position of the ball.

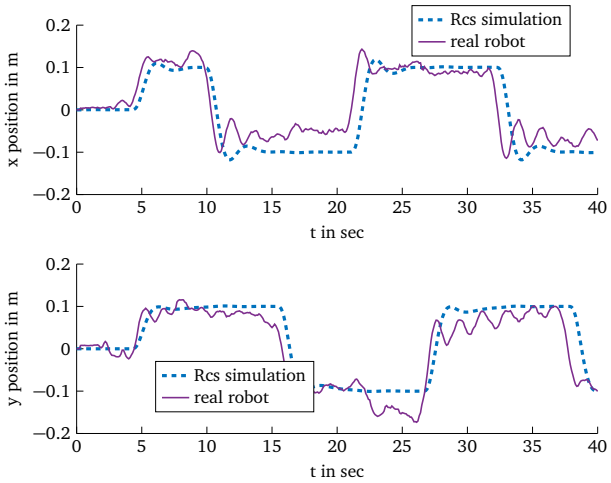


Figure C.9.: Controlling the position of the plate to follow a square: The x-position and the y-position is plotted over the time.

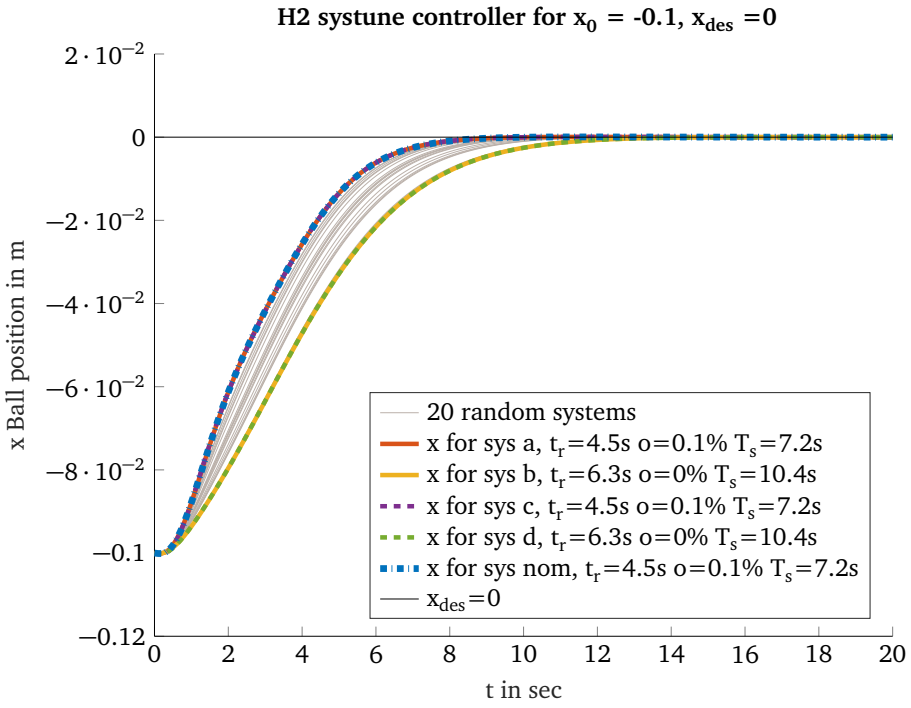


Figure C.6.: Balancing behaviour of the fixed structure $\mathscr{H}_2$ controller (simulation)

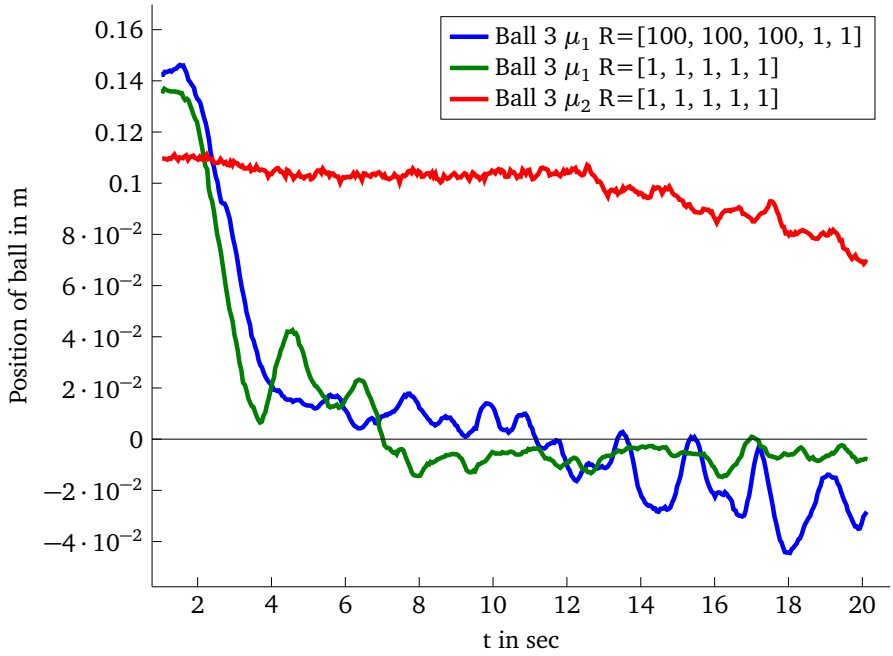


Figure C.7.: Balancing behaviour of LQR controller with different weighted control input vector $\mathbf{u}$ (on real robot)

D Matlab function explanation

Table D.1 explains the Matlab *robstab* and *robgain* commands. Within this thesis Matlab R2018b was utilized. Note that in older Matlab versions *robstab* is replaced by *robuststab*. For more detailed information of these commands see *robstab doc* and *robgain doc*.

	[stabmarg, wcu, info] = robstab(usys, w)
stabmarg	*stabmarg* specifies the scalar factor (γ) by which the uncertainties can be amplified until the limit ($\gamma \cdot \Delta$) of RS is reached. A *LowerBound* and *UpperBound* is given which specifies the region of this factor. The exact stability margin is guaranteed to be no smaller than the LowerBound. In other words, for all modeled uncertainty with normalized magnitude up to LowerBound, the system is guaranteed stable. If the *LowerBound* and *LowerBound* are greater than one, the system has RS. Assume a *LowerBound* of 1.59 and a *UpperBound* of 1.6. This means that the system can tolerate up to 60% more of parameter uncertainty until it reaches instability.
wcu	Specifies the uncertain parameters which are closest to the nominal values, for which the system no longer has RS.

	[perfmarg, wcu] = robgain(usys, gamma)
perfmarg	*perfmarg* specifies the scalar factor (γ) by which the requirements of uncertainty and performance can be amplified ($\gamma \cdot \hat{\Delta}$) until the limit of RP is reached (with respect to the input gamma). A *LowerBound* and *UpperBound* is given which specifies the region of this factor. The exact margin is guaranteed to be no smaller than the *LowerBound*. In other words, for all modeled uncertainty with normalized magnitude up to *LowerBound*, the system is guaranteed to have peak gain below gamma. The exact margin is guaranteed to be no larger than *UpperBound*. In other words, some uncertain-element values associated with this magnitude exist that drive the peak gain above gamma. If the *LowerBound* and *UpperBound* are below the specified gamma, the system achieves RP.
wcu	Specifies the uncertain parameters which are closest to the nominal values, for which the system has no longer RP.

Table D.1.: Explanation of Matlab *robstab* and *robgain* commands.

Table D.2 explains the Matlab *systune* funtion. For more details see *Matlab systune function*.

	[CL, fSoft, gHard, info] = systune(CL0, SoftGoals, HardGoals, Options)
CL0	System model (of type *gnss*) that can contain uncertainties as well as a fixed controller structure with adjustable parameters. *systune* internally optimizes certain $\mathscr{H}_2$ or $\mathscr{H}_\infty$ norms to find these adjustable parameters.
SoftGoals	Many different sub constrains in frequency or time domain can be specified to find a controller with the desired behaviour. E.g. using *TuningGoal.StepResp* a desired step response can be formulated.
HardGoals	Hard Goals as e.g. ASL can be formulated.
Options	Options allow to specify internal parameters of the optimization process.
CL	The closed loop system (with the designed controller inserted).
fSoft	Contains the value of each soft goal for the best overall run. Each tuning goal evaluates to a scalar value, and systune minimizes the maximum value of the soft goals, subject to satisfying all the hard goals.
gHard	gHard contains the value of each hard goal for the best overall run (the run that achieved the smallest value for max(fSoft), subject to max(gHard)<1. All entries of gHard are less than 1 when all hard goals are satisfied. Entries greater than 1 indicate that systune could not satisfy one or more design constraints.
info	Structure with additional information of the optimization process as e.g. *wcPert*, which contains the worst combinations of uncertain parameters.

Table D.2.: Explanation of the Matlab *systune* function.

E Code snippets

The Matlab code that is used in this thesis is attached with a CD. If you do not have the CD please send me (MarkusLamprecht@live.de) a request to receive the code. Table E.1 lists the different Matlab functions and scripts.

Name	Description
dBoP.m	Code of the modelling of the dBoP system.
cBoP.m	Code of the modelling of the cBoP system.
getBall.m	Code of the modelling of the getBall function.
getPlate.m	Code of the modelling of the getPlate function.
lqr_design.m	LQR design main script.
getBoPFriction.m	getBoPFriction function used for LQR design.
getUncertainty.m	getUncertainty function used for LQR design.
mmpp_design.m	MMPP main optimization script.
mmpp_opt.m	optimization function used for MMPP design.
h2_systune.m	$\mathscr{H}_2$ systune main optimization script.
h2_systune_lin.m	$\mathscr{H}_2$ systune Hard and Soft Goals as well as RS-analysis for the controller of Section 4.9.
h2_systune_re.m	$\mathscr{H}_2$ systune Hard and Soft Goals the redesigned controller of Section 6.3.
genP.m	generate generalized plant function.

Table E.1.: Matlab functions and scripts that are used in this thesis

List of Figures

List of Tables

Bibliography

[1] Z. C. Böcker Joachim, Hartmann Irmfried, *Nichtlineare und adaptive Regelungssysteme*, SpringerVerlag, Ed. SpringerVerlag, 1986.

[2] K. Åström, L. Neumann, and P. Gutman, "A comparison between robust and adaptive control of uncertain systems," *IFAC Proceedings Volumes*, vol. 20, no. 2, pp. 43 – 48, 1987, 2nd IFAC Workshop on Adaptive Systems in Control and Signal Processing 1986, Lund, Sweden, 30 June-2 July 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667017559352

[3] K. Zhou, J. Doyle, and K. Glover, *Robust and Optimal Control*, ser. Feher/Prentice Hall Digital and. Prentice Hall, 1996. [Online]. Available: https://books.google.de/books?id=RPSOQgAACAAJ

[4] F. Treede, "Learning robust control policies from simulations with perturbed parameters," Master's thesis, TU Darmstadt, 2017.

[5] P. P. Uwer, *Theoretische Physik - Mechanik*, T. scriptum is available online at: http://jaguar.biologie.hu-berlin.de/downloads/TP_SS2010/skript_mechanik.pdf, Ed. Humbold-Universität zu Berlin, 2010.

[6] D. W. Kwang-Kyu Lee, Georg Bätz, "Basketball robot: Ball-on-plate with pure haptic information," in *IEEE International Conference on Robotics and Automation*. Pasadena, CA, USA: IEEE, 2008, pp. 2410–2415.

[7] I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. New York, NY, USA: John Wiley & Sons, Inc., 1996.

[8] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. Prentice-Hall, 1998.

[9] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990.

[10] N. Wettstein, "Balancing a ball on a plate using stereo vision," Master's thesis, ETH-Zürich, 2013.

[11] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton university press, 2004. [Online]. Available: http://www.cds.caltech.edu/~murray/books/AM08/pdf/am07-complete_17Jul07.pdf

[12] O. Föllinger, U. Konigorski, B. Lohmann, G. Roppenecker, and A. Trächtler, *Regelungstechnik*, 12th ed. VDE-Verlag, 2016.

[13] E. Lenz, *Robuste Regelung*, T. script is avaliable at the rtm institute of the TU Darmstadt, Ed. Technische Universität Darmstadt, Institut für Automatisierungstechnik, Fachgebiet Regelungstechnik und Mechatronik, 2018.

[14] I. Clavera, D. Held, and P. Abbeel, "Policy transfer via modularity and reward guiding," in *International Conference on Intelligent Robots and Systems (IROS)*, September 2017.

[15] F. Muratore, F. Treede, M. Gienger, and J. Peters, "Domain randomization for simulation-based policy optimization with transferability assessment," in *Conference on Robot Learning (CoRL)*, 2018. [Online]. Available: https://www.ias.informatik.tu-darmstadt.de/uploads/Team/FabioMuratore/Muratore_Treede_Gienger_Peters--SPOTA_CoRL2018.pdf