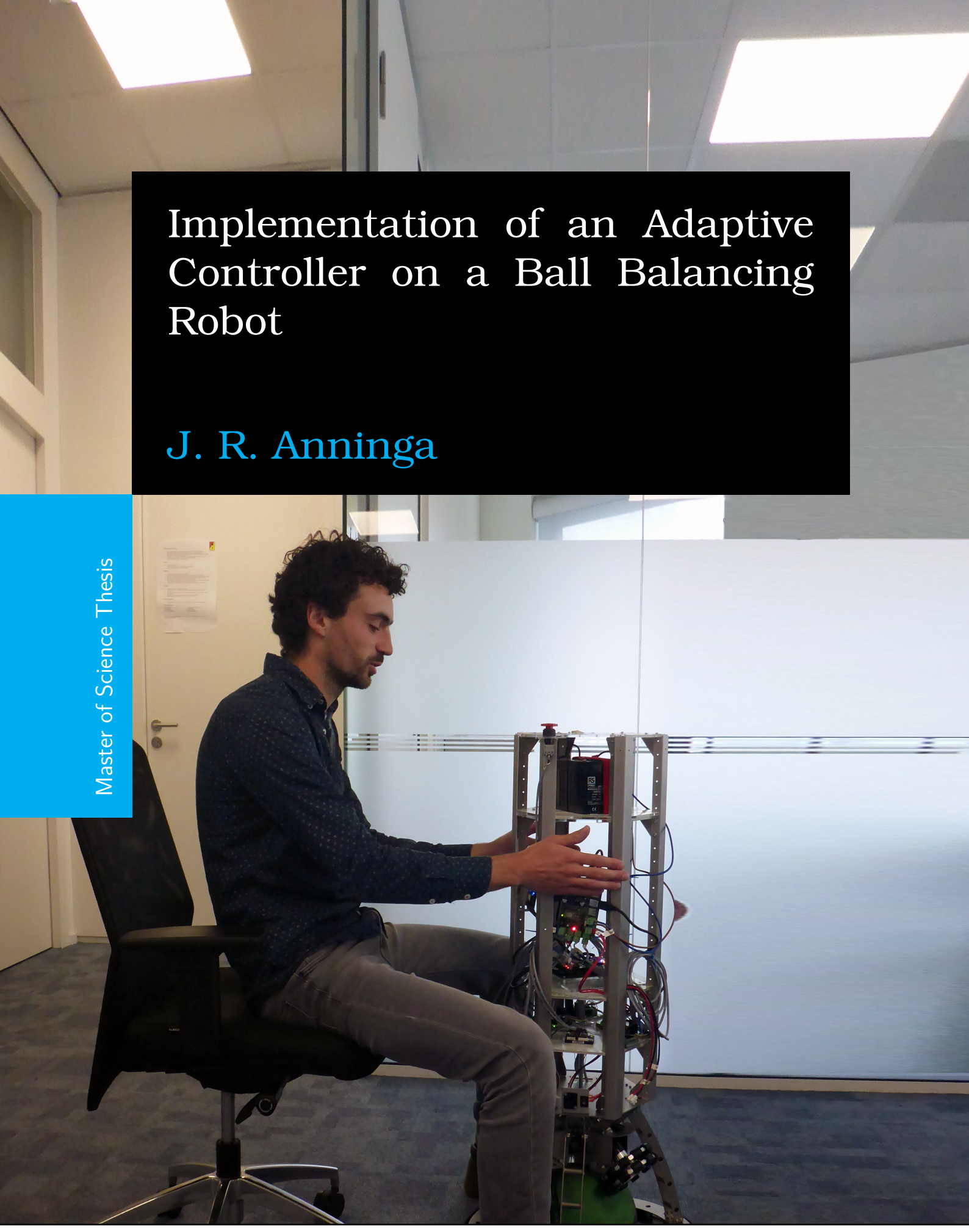


Implementation of an Adaptive Controller on a Ball Balancing Robot

J. R. Anninga

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Implementation of an Adaptive Controller on a Ball Balancing Robot

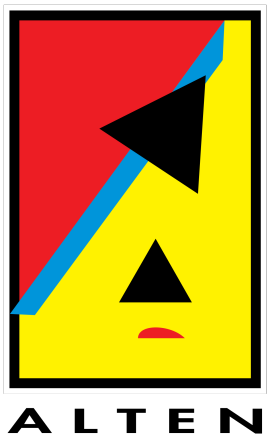
Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

J. R. Anninga

October 16, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology



The work in this thesis was supported by ALTEN. Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Implementation of an Adaptive Controller on a Ball Balancing Robot

by

J. R. Anninga

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: October 16, 2019

Supervisor(s):

dr.ir. S. Baldi

ir. D. Hanssen

Reader(s):

dr.ir. A.J.J. van den Boom

dr.ir. M. Wiertlewski

Abstract

A Ball Balancing Robot (BBR) is a highly agile machine which is an unstable system. To stabilize, it uses three omni wheels connected to motors as an input of the system. Since it has less inputs than outputs, the BBR is an underactuated system. Due to the underactuated character and the non-minimum phase plant, the BBR is an interesting mechanism to study the effectiveness of control design.

An adaptive controller will be designed for the BBR such that it can cope with a changing environment, for example added mass to the robot or change of the surface it rides on. Due to the non-minimum phase characteristic of the plant, an Adaptive Pole Placement Controller (APPC) is the most suitable adaptive controller for the BBR.

In order to study which adaptive law in combination with the APPC gives the best performance simulations are done. For these simulations a 2D planar model is derived, which is compared and verified with the actual system.

Table of Contents

"Everything should be made as simple as possible, but no simpler"
— *Albert Einstein*

Chapter 1

Introduction

A Ball Balancing Robot (BBR), as the name suggests, is a robot which can balance and ride on a ball. It uses three omni-wheels connected to motors as an input of the system such that it can balance. Moreover, it is a highly agile machine but also statically unstable. The problem with statically stable robots is that they can tip over. Since they are not actuated, the statically stable robots cannot recover from a push. To overcome this problem, they normally have a low centre of gravity (cog) which makes them less manoeuvrable. The BBR, however, has a high cog, hence its high agility. Another advantage is that the BBR does not have to turn around its axis before it can ride to a certain direction in comparison with a Segway where that is needed. Despite of those benefits it has not a real purpose yet, but it is an interesting machine to study since, as stated before, it is a statically unstable robot and it is highly agile. Moreover it is an under actuated system since it has more outputs (angle and position) than inputs and it is a non-minimum phase plant.

ALTEN is a consultancy and engineering organisation that works in the hightech sector and IT. Due to the fact that they do not have an own products to show, they developed the BBR. By using the BBR they draw attention at fairs and can showcase their skills to the people such that they can learn about ALTEN. The BBR at ALTEN is designed to balance on different sizes of balls, it will have a different surface to ride on each time it is demonstrated on another fair, and the cog can change due to variation in placement of the battery. In other words, it has a changing plant.

An adaptive controller is able to adapt itself to a changing plant. Research on autopilots for high performance aircraft was one of the main motivations for research on adaptive control [1]. The changing plant was due to the wide operating range of speeds and altitudes. The BBR does not have to operate in a wide range of speeds and certainly not in different altitudes, but its plant will change. Therefore it will be interesting to study how well an adaptive controller will work on a BBR and how well it can adapt to a changing environment. In collaboration with ALTEN an adaptive controller is designed and implemented for their BBR.

First the layout of the BBR at ALTEN and the different controllers previously implemented on the BBR are explained in Section 1-1. Section 1-2 shows earlier research on the BBR by

other universities. Finally, the research questions and the structure of this thesis is explained in Section 1-3.

1-1 BBR Alten

The BBR at Alten is showed in Figure 1-1. The ball on which it balances, is a medicine ball of 3 kg. It has 3 omnidirectional wheels actuated by 3 brushless motors. The motors are controlled by motor controllers. These have two different control settings, it can be either velocity or torque controlled. In the body off the robot are different sensors in order to measure its tilt angle and angular rate: an accelerometer, a gyroscope and a magnetometer. The motors include encoders, which provide data to calculate the position of the BBR. An on-board computer uses the sensor data to calculate the required torques to balance the robot and move it to the desired place. Earlier research has been done by van der Blonk [2] and Verdier [3].

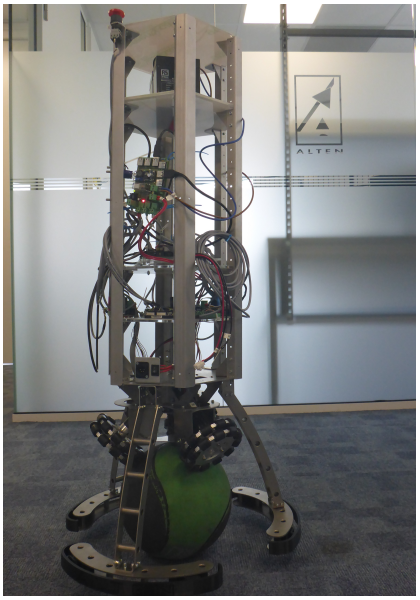


Figure 1-1: BBR from Alten

The BBR is controlled now using a cascaded PID controller using the velocity controlled setting of the motor controllers. Cascaded control consist of two control loops: the inner-loop is a normal feedback controller, while the output of the outer-loop controller will be the reference signal for the inner-loop. This structure is used for the BBR by others as well [4], [5], [6]. The cascaded control is used at ALTEN for two different settings: station keeping and balancing. Balancing means that the robot has no position reference signal and only the angle of the robot is controlled. In this setting the robot can be taken along by hand such that you can "walk" with the robot. Station keeping means that the robot has a position reference signal and is also balancing. The BBR can be pushed and it will go back to the initial position.

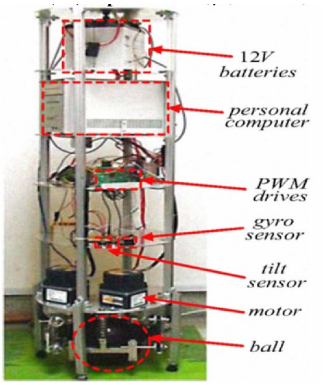
1-2 Other BBRs

The first BBR (The Ballbot) was build at Carnegie Mellon University (CMU) [5]. Since then also other universities designed, built and studied this highly agile robot.

The Ballbot can be seen in Figure 1-2. To actuate the robot, it is using an inverted mouse ball principle. It uses two rollers to actuate and two rollers for guiding. Later on they used all four rollers to actuate the robot for better performance. They also installed an extra motor to control its yaw (rotation around the z-axis). It is 1.5 m tall such that it has more or the less the same height as a person and its weight is 45 kg. It has three extendable legs to be statically stable as well.

Figure 1-2: Ballbot from Carnegie Mellon University

At the National Chung Hsing Universit (NCHU) they made two BBRs. One with an inverted mouse ball principle [7] (Figure 1-3a), the same as CMU, and one with three omnidirectional wheels [8] as can be seen in Figure 1-3b.



(a) BBR with inverted mouse drive

(b) BBR with omnidirectional wheels

Figure 1-3: Two BBRs from National Chung Hsing University

At the Eidgenössische Technische Hochshule (ETH) in Zürich they made Rezero [9]. They used three omnidirectional wheels to actuate the BBR. They used a system to pull the ball to the motors such that the wheels would not slip over the ball. Rezero is shown in Figure 1-4.

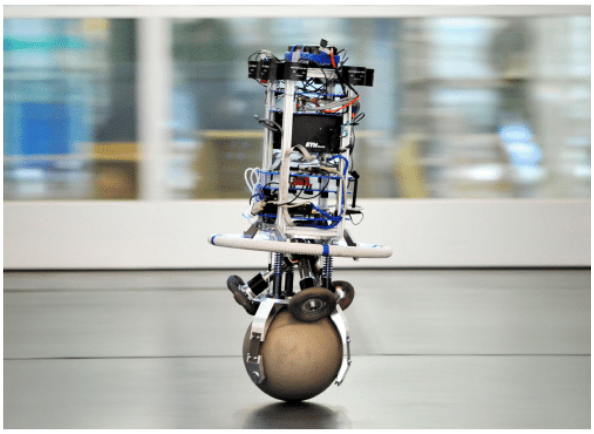


Figure 1-4: Rezero from the ETH

In Japan at the Tohoku Gakuin University (TGU) they made a similar BBR to ETH [10]. It has 3 omnidirectional wheels but the cog is lower than the one from ETH as can be seen in Figure 1-5.

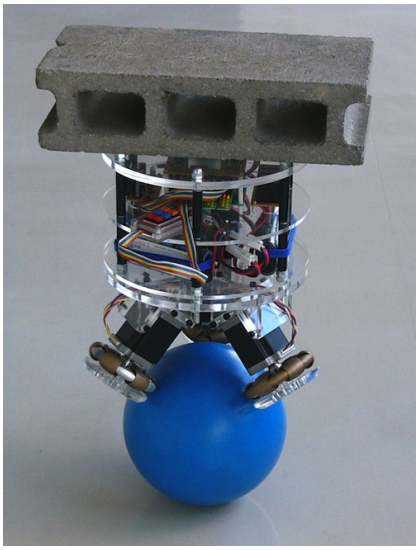


Figure 1-5: BBR from Tohoku Gakuin University

Beside those four BBRs there are a lot more of these robots. Some people make them just for fun and even Honda made a BBR to play around with.

1-3 Structure of the thesis

This Chapter gave an introduction into the thesis. It showed what a BBR is, what the challenges are when controlling one and earlier research on the BBR. From this, four different research questions resulted. First, in Chapter 2 the model is derived using the Lagrange method. It is a good practice to derive a model and it gives good insights into the dynamics of the BBR. As stated before, the BBR can have a changing plant, therefore it is interesting to see if an adaptive controller can cope with changes. Hence, the first question is:

What is a suitable adaptive controller to implement on the BBR?

This question is addressed in Chapter 3. Moreover, the control design and the different adaptive control algorithms which are implemented on the BBR are explained. Since the adaptive controller will be implemented on a real set up, the controller not only will be tuned, but also the influence of the different parts needs to be studied. Therefore the next question is:

What is the influence of the different parts of the adaptive controller on the performance of the controller using simulations?

The results are discussed in Chapter 4 using `MATLAB` and `SIMULINK`. Also, the different adaptive controllers and compared.

When the adaptive controller is implemented on the real BBR, it first needs a non-adaptive working controller to start with, using the same structure as the adaptive controller. So the third question is:

Which type of control is able to control the real BBR regarding both balancing and station keeping?

This question is answered in Chapter 5. When there is a non-adaptive controller working on the BBR the adaptive controller is implemented on the robot. It is interesting to see which controller performs the best and hence the last question:

Which controller shows the best performance on the set up of the BBR?

In Chapter 5 the results are shown for the adaptive controllers implemented on the real BBR. To conclude, Chapter 6 gives a conclusion on the project and recommendations for further research and improvements on the BBR.

Chapter 2

Model

The dynamics for the Ball Balancing Robot (BBR) have been derived in earlier research [2] [3] [9]. Even though, it is a good practise to derive a model and it gives good insights into the dynamics of the BBR. In the sections below a model is derived for the BBR. The BBR is modelled in three 2D models for each plane: $x-y$, $x-z$ and $y-z$ plane. First the assumptions used for the model are described in Section 2-1. Section 2-2 describes the coordinates used for the model. The equations of motion for the different planes are derived in Section 2-3 using the Lagrangian method. The identification method to identify the parameters of the model are described in Section 2-4. The Chapter is finished with the linearization of the model in Section 2-5.

2-1 Assumptions

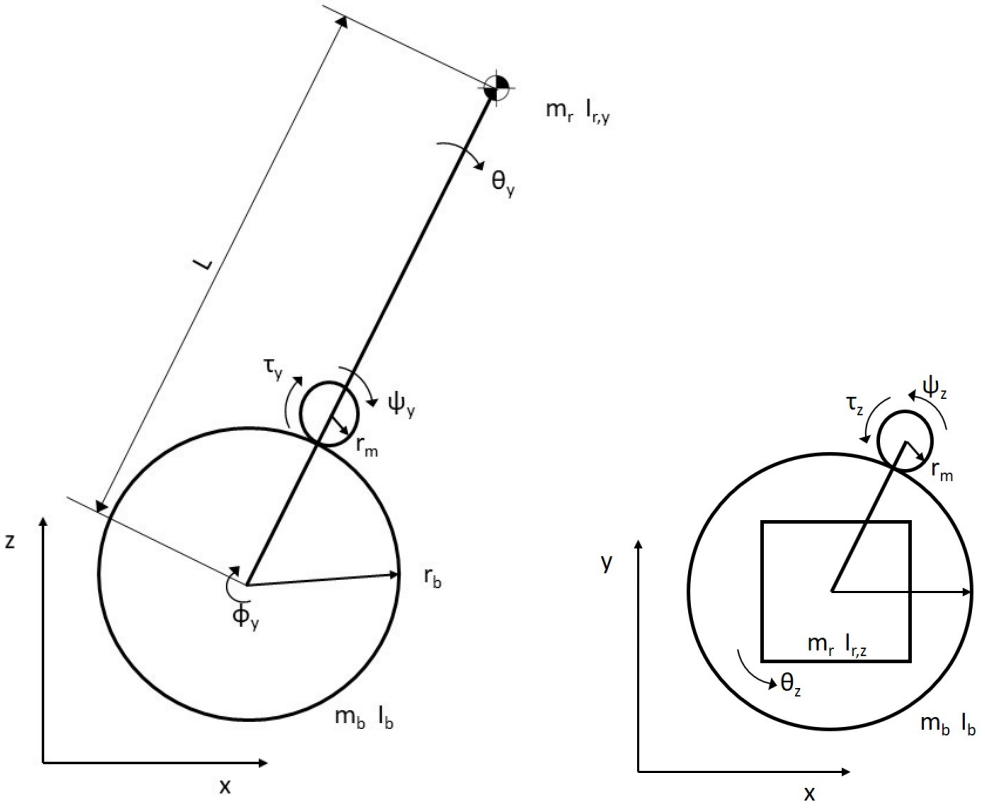
For this model the following assumptions are used:

- The ball will be modelled as a rigid sphere
- The body of the robot will be modelled as a rigid box
- There is no slip between the ball and the floor nor between the wheels and the ball
- The robot rides on a flat surface
- There is no time delay between the measurement of the sensors and the actuation of the motors
- Friction is modelled only between the floor and the ball using static friction, there is no friction modelled between the ball and the wheels.

Since a 2D model is derived instead of a 3D model, the wheels are replaced by virtual wheels in the model. Those wheels have the same dimensions as the real wheels and are placed in plane with the $x-y$, $x-z$ and $y-z$ plane.

2-2 Coordinates

Figure 2-1a shows a schematic drawing of the BBR in the $x-z$ plane. The model for the $x-z$ and $y-z$ are the same, therefore only the $x-z$ will be described. Figure 2-1b shows a schematic drawing in the $x-y$ plane.



(a) $x-z$ plane **(b)** $x-y$ plane

Figure 2-1: Schematic drawing of the ball balancing robot

The minimal coordinates q for those models are:

$$q_{xz} = \begin{bmatrix} \theta_y \\ \phi_y \end{bmatrix}, \quad q_{yz} = \begin{bmatrix} \theta_x \\ \phi_x \end{bmatrix}, \quad q_{xy} = \begin{bmatrix} \theta_z \end{bmatrix} \tag{2-1}$$

where θ_i is the rotation of the body of the robot around the i-axis (x, y and z) and ϕ_i is the angle of the ball around the i-axis (only x and y). As can be seen from Equation (2-1), θ_z is the only coordinate used for the $x-y$ plane . This comes from the assumption that there is no slip between the ball and the floor and therefore the ball does not rotate around the z-axis

The minimal coordinates can be rewritten in Cartesian coordinates in the following manner for the $x-z$ plane:

$$\begin{bmatrix} x_r \\ z_r \end{bmatrix} = \begin{bmatrix} r_b\phi_y + l\sin(\theta_y) \\ -l\cos(\theta_y) \end{bmatrix}, \quad \begin{bmatrix} x_b \\ z_b \end{bmatrix} = \begin{bmatrix} r_b\phi_y \\ 0 \end{bmatrix}, \quad \begin{bmatrix} x_m \\ z_m \end{bmatrix} = \begin{bmatrix} r_b\phi_y + (r_b + r_m)\sin(\theta_y) \\ -(r_b + r_m)\cos(\theta_y) \end{bmatrix} \tag{2-2}$$

where r_b is the radius of the ball, r_m the radius of the wheel and l is the distance between the centre of gravity (cog) of the ball and the robot.

For the $x-y$ plane the minimal coordinates can be rewritten in Cartesian coordinates in the following manner:

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} (r_b + r_m)\cos(\theta_z) \\ (r_b + r_m)\sin(\theta_z) \end{bmatrix} \tag{2-3}$$

2-3 Equations of motion

To get the equations of motion the Lagrangian method [11] is used. The following steps lead to the equations of motion:

- Determine the kinetic energy T.
- Determine the potential energy V.
- Determine the non-conservative generalized torques τ.
- Write down the Lagrangian $L(q, \dot{q}) = T - V$.
- Use the Euler-Lagrange equation to get the equations of motion $\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = \tau_i$.
- Add the friction term.

First the equations of motion are derived for the $x-z$ plane. As mentioned before, the $y-z$ plane has the same equations of motion. Afterwards the equations of motion for the $x-y$ plane are derived.

2-3-1 $x-z$ plane

Energy of the ball

The kinetic energy of the ball T_b is:

$$T_b = \frac{1}{2}m_b v_b^2 + \frac{1}{2}I_b\omega^2 \tag{2-4}$$

$$= \frac{1}{2}m_b r_b^2 \dot{\phi}_y^2 + \frac{1}{2}I_b\dot{\phi}_y^2 \tag{2-5}$$

where m_b is the mass of the ball and I_b is the inertia of the ball.

Since the ball cannot jump the potential energy V_b is zero:

$$V_b = 0 \tag{2-6}$$

Energy robot

The kinetic energy for the body of the robot T_r is given by the following equation:

$$T_r = \frac{1}{2} m_r v_r^2 + \frac{1}{2} I_{r,y} \omega^2 \tag{2-7}$$

where m_r is the mass of the robot and $I_{r,y}$ is the inertia of the robot around the y axis. v_r^2 can be calculated as follows using the Cartesian coordinates from Equation (2-2):

$$\begin{aligned} v_r^2 =& \dot{x}_r^2 + \dot{z}_r^2 \\ =& (r_b \dot{\phi}_y + l\cos(\theta_y)\dot{\theta}_y)^2 + (\dot{\theta}_y l \sin(\theta_y))^2 \\ =& r_b^2 \dot{\phi}_y^2 + 2 r_b \dot{\phi}_y \dot{\theta}_y l \cos(\theta_y) + l^2 \dot{\theta}_y^2 \end{aligned}$$

This is substituted into Equation (2-7):

$$T_r = \frac{1}{2} m_r (r_b^2 \dot{\phi}_y^2 + 2 r_b \dot{\phi}_y \dot{\theta}_y l \cos(\theta_y) + l^2 \dot{\theta}_y^2) + \frac{1}{2} I_{r,y} \dot{\theta}_y^2 \tag{2-8}$$

The potential energy V_r is given by the following equation:

$$V_r = m_r g l \cos(\theta_y) \tag{2-9}$$

Energy motor

The kinetic energy T_m is given by the following equation:

$$T_m = \frac{1}{2} m_m v_m^2 + \frac{1}{2} I_{m,y} \omega^2 \tag{2-10}$$

where m_m is the mass of the motor and wheel, $I_{m,y}$ is the inertia of the wheel and motor around the y axis. v_m^2 is calculated as follows:

$$\begin{aligned} v_m^2 =& \dot{x}_m^2 + \dot{z}_m^2 \\ =& (r_b \dot{\phi}_y + (r_b + r_m)\cos(\theta_y)\dot{\theta}_y)^2 + ((r_b + r_m)\sin(\theta_y)\dot{\theta}_y)^2 \\ =& r_b^2 \dot{\phi}_y^2 + 2 r_b (r_b + r_m)\cos(\theta_y)\dot{\phi}_y \dot{\theta}_y + (r_m + r_b)^2 \dot{\theta}_y^2 \end{aligned}$$

This equation is substituted into Equation (2-10) to give the following equation:

$$T_m = \frac{1}{2} m_m (r_b^2 \dot{\phi}_y^2 + 2 r_b (r_b + r_m)\cos(\theta_y)\dot{\phi}_y \dot{\theta}_y + (r_m + r_b)^2 \dot{\theta}_y^2) + \frac{1}{2} I_{m,y} \dot{\psi}_y^2 \tag{2-11}$$

ψ_y is not one of the minimal coordinates so $\dot{\psi}_y$ needs to be rewritten into minimal coordinates. To do that Figure 2-2 is used.

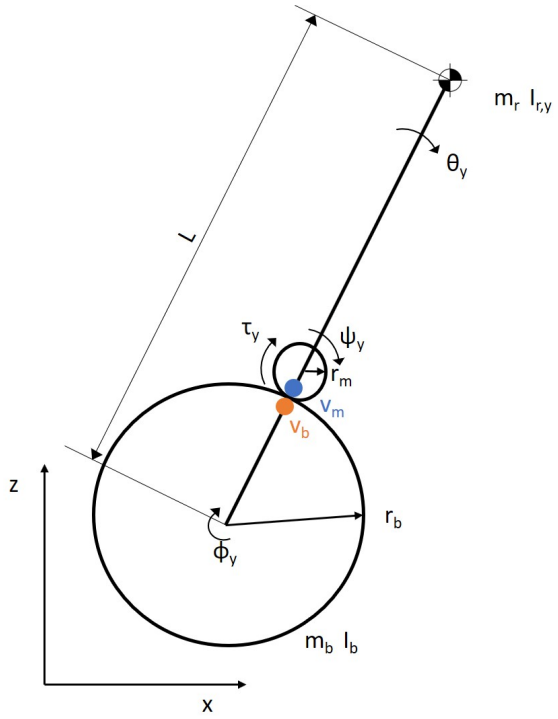


Figure 2-2: Schematic drawing of the ball balancing robot, $x-z$ plane. $v_b = v_m$ is used to rewrite ψ_y into minimal coordinates.

Obviously, the speed of the ball v_b and the speed of the motor v_m at the point where they touch should be equal if there is no slip between the ball and the wheel.

$$\begin{aligned} v_b &= \begin{bmatrix} \dot{x}_b \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\phi}_y \\ 0 \end{bmatrix} \times \begin{bmatrix} r_b \sin(\theta_y) \\ 0 \\ -r_b \cos(\theta_y) \end{bmatrix} \\ &= \begin{bmatrix} \dot{x}_b \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -\dot{\phi}_y r_b \cos(\theta_y) \\ 0 \\ -\dot{\phi}_y r_b \sin(\theta_y) \end{bmatrix} \end{aligned}$$

$$\begin{aligned} v_m &= \begin{bmatrix} \dot{x}_b \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\theta}_y \\ 0 \end{bmatrix} \times \begin{bmatrix} (r_b + r_m)\sin(\theta_y) \\ 0 \\ -(r_b + r_m)\cos(\theta_y) \end{bmatrix} + \begin{bmatrix} 0 \\ -\dot{\psi}_y \\ 0 \end{bmatrix} \times \begin{bmatrix} -r_m \sin(\theta_y) \\ 0 \\ r_m \cos(\theta_y) \end{bmatrix} \\ &= \begin{bmatrix} \dot{x}_b \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -\dot{\theta}_y (r_b + r_m)\cos(\theta_y) \\ 0 \\ -\dot{\theta}_y (r_b + r_m)\sin(\theta_y) \end{bmatrix} + \begin{bmatrix} -\dot{\psi}_y r_m \cos(\theta_y) \\ 0 \\ -\dot{\psi}_y r_m \sin(\theta_y) \end{bmatrix} \end{aligned}$$

From those equations it follows that:

$$\dot{\psi}_y = -\dot{\theta}_y \frac{r_b + r_m}{r_m} + \dot{\phi}_y \frac{r_b}{r_m} \tag{2-12}$$

When filling this in, the kinetic energy T_m is:

$$\begin{aligned} T_m = \frac{1}{2} m_m (r_b^2 \dot{\phi}_y^2 + 2 r_b (r_b + r_m) \cos(\theta_y) \dot{\phi}_y \dot{\theta}_y + (r_m + r_b)^2 \dot{\theta}_y^2) \\ + \frac{1}{2} I_{m,y} \Big(\frac{r_b}{r_m} \Big)^2 \dot{\phi}_y^2 - 2 \frac{r_b (r_b + r_m)}{r_m^2} \dot{\phi}_y \dot{\theta}_y + \Big(\frac{r_b + r_m}{r_m} \Big)^2 \dot{\theta}_y^2 \end{aligned} \tag{2-13}$$

The potential energy of the motor V_m is:

$$V_m = m_m g (r_b + r_m) \cos(\theta_y) \tag{2-14}$$

Non-conservative torques

To get the non-conservative torques τ_{ext} Equation (2-12) is used.

$$\tau_{ext} = \begin{bmatrix} -\frac{r_b + r_m}{r_m} & \frac{r_b}{r_m} \end{bmatrix}^\top \tau_y \tag{2-15}$$

Lagrangian

The Lagrangian is as follows:

$$L(q, \dot{q}) = T - V \tag{2-16}$$

$$L(\theta_y, \phi_y, \dot{\theta}_y, \dot{\phi}_y) = T_b + T_r + T_m - V_b - V_r - V_m \tag{2-17}$$

To get the equations of motion the Euler-Lagrange equation is used:

$$\frac{d}{dt} \Big(\frac{\partial L}{\partial \dot{q}_i} \Big) - \frac{\partial L}{\partial q_i} = \tau_{ext} \tag{2-18}$$

The above equation can be rewritten in the following form:

$$M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) = \tau_{ext} \tag{2-19}$$

When all the energy equations are filled in the Lagragian and the Euler-Lagrangian equation is used to get the equations of motion, then the resulting system matrices M, C, G and τ_{ext} are:

$$M(\theta_y, \phi_y) = \begin{bmatrix} m_r l^2 + m_m r_t^2 + I_{r,y} + I_{m,y} \Big(\frac{r_t}{r_m} \Big)^2 & (m_r l + m_m r_t) r_b \cos(\theta_y) - I_{m,y} \frac{r_b r_t}{r_m^2} \\ (m_r l + m_m r_t) r_b \cos(\theta_y) - I_{m,y} \frac{r_b r_t}{r_m^2} & (m_m + m_r + m_b) r_b^2 + I_{b,y} + I_{m,y} \Big(\frac{r_b}{r_m} \Big)^2 \end{bmatrix} \tag{2-20}$$

$$C(\theta_y, \phi_y, \dot{\theta}_y, \dot{\phi}_y) = \begin{bmatrix} 0 & 0 \\ -(m_r r_b l + m_m r_b r_t)\sin(\theta_y)\dot{\theta} & 0 \end{bmatrix} \tag{2-21}$$

$$G(\theta_y, \phi_y) = \begin{bmatrix} -(m_r g l + m_m g r_t)\sin(\theta_y) \\ 0 \end{bmatrix} \tag{2-22}$$

$$\tau_{ext} = \begin{bmatrix} -\frac{r_t}{r_m} & \frac{r_b}{r_m} \end{bmatrix}^T \tau_y \tag{2-23}$$

where r_t is the sum of r_b and r_m.

Friction

When friction is added to the model, Equation (2-19) is rewritten as follows:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + D(\dot{q}) = \tau_{ext} \tag{2-24}$$

where $D(\dot{q})$ is the friction term. Since there is only friction acting on the ball, $D(\dot{q})$ is written as:

$$D(\dot{q}) = \begin{bmatrix} 0 \\ D_\phi(\dot{\phi}_y) \end{bmatrix} \tag{2-25}$$

where $D_\phi(\dot{\phi}_y)$ is described by:

$$D_\phi(\dot{\phi}_y) = \begin{cases} \min(D_{sta}, |\sum T_{tot}|) \cdot \text{sign}(\sum T_{tot}) & \text{if} \quad \dot{\phi}_y = 0 \\ D_{sta} \cdot \text{sign}(\dot{\phi}_y) & \text{if} \quad \dot{\phi}_y \neq 0 \end{cases} \tag{2-26}$$

where D_{sta} is the static friction constant in Nm and $\sum T_{tot}$ is the sum of the torques working on the ball: $\tau_{ext} - C(q, \dot{q})\dot{q} - G(q) - D(\dot{q})$.

2-3-2 $x - y$ plane

The same strategy is used to get the equation of motion for the $x - y$ plane. Since we look into the $x - y$ plane, all the potential energies are zero.

Energy ball

One assumption is that there is no slip between the ball and the floor, therefore the rotation around the z-axis of the ball is equal to zero and the kinetic energy of the ball is equal to zero.

Energy robot

The kinetic energy of the robot T_r is as follows:

$$T_r = \frac{1}{2} I_{r,z} \dot{\theta}_z^2 \tag{2-27}$$

Energy motor

The kinetic energy of the motor T_m is as follows:

$$\begin{aligned} T_m =& \frac{1}{2} m_m v_m^2 + \frac{1}{2} I_{m,z} \dot{\psi}_z^2 && (2\text{-}28)\\ =& \frac{1}{2} m_m (r_b + r_m)^2 \dot{\theta}_y^2 + \frac{1}{2} I_{m,z} \Big(\frac{r_b}{r_m}\Big)^2 \dot{\theta}_z^2 && (2\text{-}29) \end{aligned}$$

Non-conservative torques

The non-conservative torques τ_{ext} are:

$$\tau_{ext} = -\frac{r_b}{r_m} \tau_z \tag{2-30}$$

Lagrangian

Putting all the energies in the Lagrangian and using the Euler-Lagrange equation, the equation of motion is:

$$(I_{r,z} + I_{m,z} \Big(\frac{r_b}{r_m}\Big)^2 + m_m (r_b + r_m)^2) \ddot{\theta}_z = -\frac{r_b}{r_m} \tau_z \tag{2-31}$$

Such that the system matrices are:

$$\begin{aligned} M(\theta_z) =& I_{r,z} + I_{m,z} \Big(\frac{r_b}{r_m}\Big)^2 + m_m (r_b + r_m)^2 && (2\text{-}32)\\ C(\theta_z, \dot{\theta}_z) =& 0 && (2\text{-}33)\\ G(\theta_z) =& 0 && (2\text{-}34)\\ \tau_{ext} =& -\frac{r_b}{r_m} && (2\text{-}35) \end{aligned}$$

Friction

There is no friction added to this model since the ball is not rotating around the z-axis.

2-4 Identification

The parameters in the above parametric models need to be identified such that the model can be used for simulations and the control design. First the parameters are estimated, afterwards the parameters are identified using closed loop identification methods.

2-4-1 Estimation

Table 2-1 shows the values for the different parameters. How these parameters are estimated is found in Appendix A-1

Table 2-1: Values of the model parameters

Parameter	Symbol	Value	Unit
mass ball	m_b	3.0	kg
mass robot	m_r	14.66	kg
mass wheel	m_w	0.55	kg
Inertia robot x-axis	$I_{r,x}$	1.946	kgm^2
Inertia robot y-axis	$I_{r,y}$	1.946	kgm^2
Inertia robot x-axis	$I_{r,z}$	0.181	kgm^2
Inertia omni wheel rotation-axis	$I_{ow,r}$	$6.938 \cdot 10^{-4}$	kgm^2
Inertia motor	I_m	$2.42 \cdot 10^{-6}$	kgm^2
Inertia gears	I_g	$8.0 \cdot 10^{-8}$	kgm^2
Inertia virtual wheel x-axis	$I_{w,x}$	$2.2 \cdot 10^{-3}$	kgm^2
Inertia virtual wheel y-axis	$I_{w,y}$	$2.2 \cdot 10^{-3}$	kgm^2
Inertia virtual wheel z-axis	$I_{w,z}$	$4.3 \cdot 10^{-3}$	kgm^2
radius of the ball	r_b	0.115	m
radius of the wheel	r_w	0.06	m
distance between cog ball and robot	l	0.454	m
angle between wheels and z-axis	α	$1/4\pi$	rad
gear ratio	k	28:1	

2-4-2 Closed-loop identification

The parameters are identified using closed loop identification described in [12]. There are two methods to identify a closed loop system. The direct and indirect way. The direct way works the same as open loop system identification. The information about the input and output of the system is used to identify the system. The basic prediction error method can be used to identify the system. With indirect closed loop identification the output of the system and for instance the reference signal are used for identification. Now the controller is also taken into account. The controller too can be identified if it is unknown using this method, or when the controller is known, this information can be used.

The direct method is the most convenient method to use for identification and is therefore used. A PID controller is designed and implemented on the BBR for balancing only and not for station keeping. The torque and both the angle of the robot and the ball are logged and

used for the system identification, such that a Single Input Multiple Output (SIMO) system needs to be identified.

Since the results of the closed-loop identification are less good then expected, the parameters are used described in Table 2-1. The results of the closed-loop identification and the methods used are described in Appendix A-2.

2-5 Linearization

For control purposes the non-linear model is linearised. First Equation (2-19) is rewritten in the following manner:

$$\ddot{q} = M(q)^{-1}(\tau_{ext} - C(q,\dot{q})\dot{q} - G(q)) \tag{2-36}$$

The linear model is described in state space form:

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u}\\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u}\end{aligned}$$

where $\mathbf{u}$ is the input of the system τ_{ext}.

2-5-1 $x-z$ plane

For the $x-z$ plane the states $\mathbf{x}$ are $(\dot{\theta}_y, \dot{\phi}_y, \theta_y, \phi_y)$ and the input $\mathbf{u} = \tau_y$. The state space matrices are calculated as follows:

$$A_y = \begin{bmatrix} \frac{\partial \ddot{\theta}_y}{\partial \dot{\theta}_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} & \frac{\partial \ddot{\theta}_y}{\partial \dot{\phi}_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} & \frac{\partial \ddot{\theta}_y}{\partial \theta_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} & \frac{\partial \ddot{\theta}_y}{\partial \phi_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} \\ \frac{\partial \ddot{\phi}_y}{\partial \dot{\theta}_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} & \frac{\partial \ddot{\phi}_y}{\partial \dot{\phi}_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} & \frac{\partial \ddot{\phi}_y}{\partial \theta_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} & \frac{\partial \ddot{\phi}_y}{\partial \phi_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad B_y = \begin{bmatrix} \frac{\partial \ddot{\theta}_y}{\partial \tau_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} \\ \frac{\partial \ddot{\phi}_y}{\partial \tau_y}\Big|_{\substack{\mathbf{x}=0\\ \mathbf{u}=0}} \\ 0 \\ 0 \end{bmatrix}$$

$$C_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad D_y = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The values of the parameters that are used for the model are shown in Table 2-1. When filling in those values the state space matrices A_y and B_y are:

$$A_y = \begin{bmatrix} 0 & 0 & 22.90 & 0 \\ 0 & 0 & -63.24 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad B_y = \begin{bmatrix} -2.84 \\ 14.76 \\ 0 \\ 0 \end{bmatrix} \tag{2-37}$$

The state space model is rewritten in four separate transfer functions. The transfer functions are used to calculate the poles $p_{\mathbf{x}}$ and zeros $z_{\mathbf{x}}$ of the system. The transfer functions with corresponding poles and zeros are shown below:

$$\begin{aligned} G_{\dot{\theta_y}}(s) &= \frac{-2.84s}{s^2 - 22.9}, \quad p_{\dot{\theta_y}} = [4.79 \ -4.79], \quad z_{\dot{\theta_y}} = [0] \\ G_{\dot{\psi_y}}(s) &= \frac{14.76s^2 - 158.6}{s^3 - 22.9s}, \quad p_{\dot{\psi_y}} = [0 \ 4.79 \ -4.79], \quad z_{\dot{\psi_y}} = [3.28 \ -3.28] \\ G_{\theta_y}(s) &= \frac{-2.84}{s^2 - 22.9}, \quad p_{\theta_y} = [4.79 \ -4.79], \quad z_{\theta_y} = [\ ] \\ G_{\psi_y}(s) &= \frac{14.76s^2 - 158.6}{s^4 - 22.9s^2}, \quad p_{\psi_y} = [0 \ 0 \ 4.79 \ -4.79], \quad z_{\psi_y} = [3.28 \ -3.28] \end{aligned}$$

From the positive poles and zeros it can be concluded that the system is unstable (because of the positive pole) and is non-minimum phase (because of the positive zero).

2-5-2 $x - y$ plane

The states $\mathbf{x}$ for the $x - y$ plane are $(\dot{\theta}_z, \theta_z)$ and the input $\mathbf{u} = \tau_z$. Using the values in Table 2-1, the state space matrices become:

$$A_z = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad B_z = \begin{bmatrix} -8.96 \\ 0 \end{bmatrix} \quad C_z = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad D_z = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2-38}$$

The state space model is rewritten into two transfer functions. These are used to calculate the poles $p_{\mathbf{x}}$ and the zeros $z_{\mathbf{x}}$ of the system. The transfer functions and the poles and zeros are shown below:

$$\begin{aligned} G_{\dot{\theta_z}}(s) &= \frac{-8.97}{s}, \quad p_{\dot{\theta_z}} = [0], \quad z_{\dot{\theta_z}} = [] \\ G_{\theta_z}(s) &= \frac{-8.97}{s^2}, \quad p_{\theta_z} = [0 \ 0], \quad z_{\theta_z} = [] \end{aligned}$$

Chapter 3

Control Design

As stated before, adaptive control is used to control the Ball Balancing Robot (BBR). The structure of an adaptive controller is showed in Figure 3-1. The adaptive law uses the in- and outputs of the plant to adapt the adaptive controller. How the controller is adjusted can be in a direct or indirect way. With a direct adaptive controller, the control effort is adapted directly by the adaptation law. When an indirect adaptive controller is used, the plant parameters are adapted first, these new plant parameters are used to change the control effort.

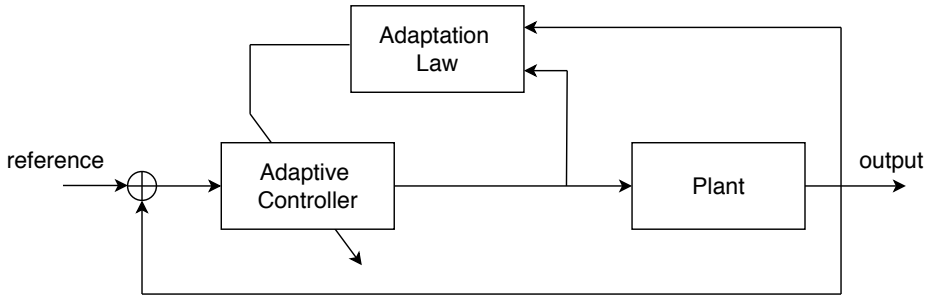


Figure 3-1: Adaptive control scheme

A study of existing literature [13] shows which adaptive controller would be suitable to implement on the BBR. The following adaptive controllers were studied:

- Self-Tuning Control
- Model Reference Adaptive Control (MRAC)
- Adaptive Pole Placement Controller (APPC)
- Robust Adaptive Control

Literature showed that the Self-Tuning Control [14] and MRAC [1] is not suitable as an adaptive controller for the BBR. Both adaptive controllers need a minimum phase plant such that the adaptive controller stays bounded. The APPC does not need a minimum phase plant [15] and can therefore be used. The robust adaptive controllers are an extension on the standard adaptive controllers to make them robust. The robust adaptive controller can be used in combination with the APPC.

Since the BBR is an unstable system, a stable pole placement controller is designed as a base for the APPC. The design process is described in Section 3-1. Afterwards the APPC is described in Section 3-2. The different adaptation laws are shown in Section 3-3. The Chapter is finished with an explanation about robust adaptive control, this is found in Section 3-4.

For this whole Chapter, the model for the $x - z$ plane is used. So a 4x4 A matrix and a 4x1 B matrix is used.

3-1 Pole placement

First a non adaptive pole placement controller is designed. With a pole placement controller, the poles of the closed loop system are placed at a desired location. It uses the following structure (see Figure 3-2):

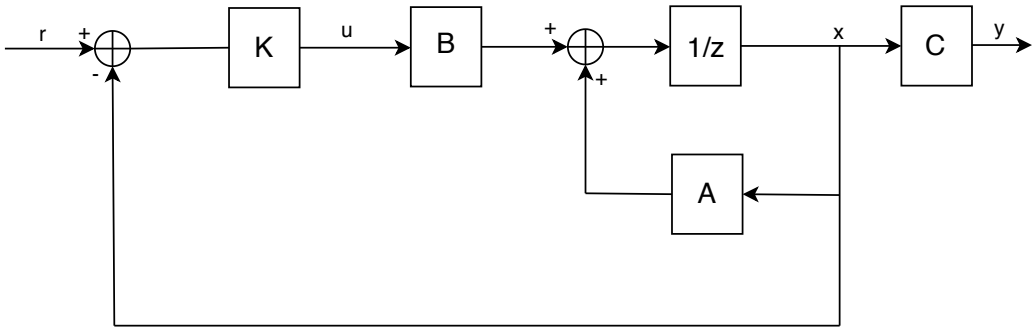


Figure 3-2: Block diagram for a pole placement controller

The controller gain matrix K is calculated using Ackermann's formula [16]:

$$K = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} W_c^{-1} P_d(A_d) \tag{3-1}$$

where W_c is the controllability matrix:

$$W_c = \begin{bmatrix} B_d & A_d B_d & A_d^2 B_d & A_d^3 B_d \end{bmatrix} \tag{3-2}$$

where A_d and B_d are the discrete time state space matrices, they are calculated as follows [16]:

$$A_d = I + A\Psi \tag{3-3}$$
$$B_d = \Psi B \tag{3-4}$$
$$\Psi = IT_s + \frac{AT_s^2}{2!} + \frac{A^2T_s^3}{3!} + \cdots + \frac{A^iT_s^{i+1}}{(i+1)!} + \cdots \tag{3-5}$$

where T_s is the time step and I is the identity matrix. Ψ is approximated by the first two terms: $\Psi = IT_s + \frac{AT_s^2}{2!}$. $P_d(A_d)$ is a function of A_d and P_d which is a polynomial where the roots are equal to the desired discrete poles $z^4 + p_{d,1}z^3 + p_{d,2}z^2 + p_{d,3}z + p_{d,4}$. $P_d(A_d)$ is calculated as follows:

$$P_d(A_d) = {A_d}^4 + {A_d}^3 p_{d,1} + {A_d}^2 p_{d,2} + {A_d}^1 p_{d,3} + {A_d}^0 p_{d,4} \tag{3-6}$$

Two of the poles will be chosen to lay around the dominant poles of the open loop system [17], the other two are chosen to be a few factors faster than the dominant poles. The two continuous time dominant poles are calculated using the second order polynomial $s^2 + 2\zeta_{dr}\omega_n s + \omega_n^2$ where ζ_{dr} is the desired damping ratio and ω_n the desired natural frequency. The poles of the second order polynomial are then:

$$p_{1,2} = -\zeta_{dr}\omega_n \pm j\omega_n\sqrt{1 - {\zeta_{dr}}^2} \tag{3-7}$$

The continuous time poles are transformed into the discrete time poles with the following equation:

$$p_{d,i} = e^{p_i T_s} \tag{3-8}$$

The algorithm described above is used for static pole placement, the controller does not change. The desired poles and the linearized state space matrices A_d and B_d are used to calculate the controller gain matrix K. The same algorithm is used for adaptive pole placement as described in Section 3-2.

3-2 Adaptive Pole Placement

The APPC makes sure the poles of the closed-loop system are always on the same location even though the plant is changing. When the plant is changing, an online parameter estimation algorithm (the adaptation law) is updating the model. The APPC is using this changing model to change the controller gains K.

The state space model described by Equation (2-37) is parametrized as follows:

$$A = \begin{bmatrix} 0 & 0 & a_1 & 0 \\ 0 & 0 & a_2 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \\ 0 \\ 0 \end{bmatrix} \tag{3-9}$$

This state space model is transformed in a parametric model in the following manner. First the state space form is transformed into two differential equations:

$$\dot{x}_1 = a_1 x_3 + b_1 u \tag{3-10}$$
$$\dot{x}_2 = a_2 x_3 + b_2 u \tag{3-11}$$

Equation (3-10) and Equation (3-11) are discretized to get the difference equations:

$$x_1(k+1) - x_1(k) = a_1 T_s x_3(k) + b_1 T_s u(k) \tag{3-12}$$
$$x_2(k+1) - x_2(k) = a_2 T_s x_3(k) + b_2 T_s u(k) \tag{3-13}$$

Where T_s is the time step The difference equation is rewritten into a parametric model:

$$\zeta = \theta^{*\top} \phi \tag{3-14}$$
$$\underbrace{\begin{bmatrix} x_1(k+1) - x_1(k) \\ x_2(k+1) - x_2(k) \end{bmatrix}}_{\zeta} = \underbrace{\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}}_{\theta^{*\top}} \underbrace{\begin{bmatrix} T_s x_3(k) \\ T_s u(k) \end{bmatrix}}_{\phi} \tag{3-15}$$

ζ is called the observation, θ^* the regressor and ϕ the regressand. ζ and ϕ can be measured and θ^* is the vector with the ideal model parameters. Since the adaptation laws do not work with θ^* being a matrix, Equation (3-15) is rewritten as follows:

$$\underbrace{\begin{bmatrix} x_1(k+1) - x_1(k) \end{bmatrix}}_{\zeta_1} = \underbrace{\begin{bmatrix} a_1 & b_1 \end{bmatrix}}_{{\theta_1}^{*\top}} \underbrace{\begin{bmatrix} T_s x_3(k) \\ T_s u(k) \end{bmatrix}}_{\phi} \tag{3-16}$$
$$\underbrace{\begin{bmatrix} x_2(k+1) - x_2(k) \end{bmatrix}}_{\zeta_2} = \underbrace{\begin{bmatrix} a_2 & b_2 \end{bmatrix}}_{{\theta_2}^{*\top}} \underbrace{\begin{bmatrix} T_s x_3(k) \\ T_s u(k) \end{bmatrix}}_{\phi} \tag{3-17}$$

Now the model is in the form of a parametric model, an online parameter estimation algorithm is used, known as adaptation laws. Different adaptation laws are described in Section 3-3.

When the parameters are estimated they are plugged in Equation (3-9) to get the state space matrices A and B. These are filled in Equation (3-3) and Equation (3-4) to get the discrete time state space matrices A_d and B_d. The discrete time state space matrices are then filled in Equation (3-1) and using Equation (3-2) and Equation (3-6) to get the gain matrix K. This process is repeated each time step to adapt the pole placement controller such that it performs in an optimum way.

3-3 Adaptation laws

In this Section the following adaptation laws are described:

- Gradient Based (GB) method
- Recursive Least Squares (RLS) method

3-3-1 Gradient Based

First an estimate of $\zeta(k)$ (denoted as $\hat{\zeta}(k)$) is calculated using the last estimate of $\theta(k-1)$ and the measurements $\phi(k)$.

$$\hat{\zeta}(k) = \theta^\top(k-1)\phi(k) \tag{3-18}$$

A certain cost will be minimized to update $\theta(k)$ using the estimation error:

$$\epsilon(k) = \zeta(k) - \hat{\zeta}(k) \tag{3-19}$$

An example of a cost function is:

$$J(\theta) = \epsilon^2 = (\zeta(k) - \theta^\top(k-1)\phi(k))^2 \tag{3-20}$$

The cost function is minimized with respect to θ using the gradient method [15]:

$$\theta(k) = \theta(k-1) + \Gamma\epsilon(k)\phi(k) \tag{3-21}$$

Where Γ is the adaptation gain which indicates how fast the adaptation will go.

3-3-2 Recursive Least Squares

With the GB method described above, a fixed adaptation gain Γ is used. While with the RLS method the adaptation gain, in this case called covariance matrix P, is adapted. The algorithm [18] is described by the following Equations:

$$e(k) = \zeta(k) - \theta^\top(k-1)\phi(k) \tag{3-22}$$

$$\theta(k) = \theta(k-1) + \frac{P(k-1)\epsilon(k)\phi(k)}{1 + \phi^\top(k)P(k-1)\phi(k)} \tag{3-23}$$

$$P(k) = P(k-1) - \frac{P(k-1)\phi(k)\phi^\top(k)P(k-1)}{1 + \phi^\top(k)P(k-1)\phi(k)} \tag{3-24}$$

3-4 Robust Adaptive Control

The adaptive laws described above are designed for an ideal model. Since the adaptive controller will be implemented on a real system, the BBR, it will be influenced by: disturbances, noise, unmodelled dynamics and time delays. The parametric model is rewritten as follows:

$$\zeta = \theta^{*\top}\phi + \eta \tag{3-25}$$

η is an unknown function that represents the modelling errors [15] which can have a big influence on the stability. There can happen a few different things which are described below.

Parameter drift means that the parameter estimation drifts away to infinity because of the unknown dynamics η.

High gain instability is described as follows [1]: "*The adaptive control law can generate a high gain feedback which excites the unmodelled dynamics and leads to instability and unbounded solutions.*"

Fast adaptation instability is similar to high gain instability. Instead of the controller gain amplifying the unmodelled dynamics, the adaptation gain Γ amplifies the unmodelled dynamics η which can result in false adaptation and unbounded solutions.

High frequency instability comes from high frequencies in the reference signal. When those same high frequencies are not covered in the model, it can happen that those high unmodelled frequencies are excited which can lead to instability.

To overcome the problems described above, only the adaptive laws have to be changed. This will result in robust adaptive laws. To describe the different robust adaptive laws, the GB method is used as a base. The RLS method can be changed in a similar manner. The following robust adaptive laws are described:

- Deadzone
- Leakage
- Static Normalization
- Dynamic Normalization

3-4-1 Dead Zone

Using Equation (3-25) the estimation error becomes:

$$\epsilon = \zeta - \theta^{\top}\phi = -\tilde{\theta}^{\top}\phi + \eta \tag{3-26}$$

If the parameter error times the regressand $|\tilde{\theta}^{\top}\phi|$ is big, $|\epsilon|$ will be big and the parameter estimation will be driven by $\tilde{\theta}\phi$. But if $|\tilde{\theta}^{\top}\phi|$ is small, the parameter estimation is driven by η and therefore we want to switch of the parameter estimation if $|\epsilon|$ gets small enough as follows [1]:

$$\theta(k) = \begin{cases} \theta(k-1) + \Gamma\epsilon(k)\phi(k) & \text{if } |\epsilon| > g_0 \\ \theta(k-1) & \text{otherwise} \end{cases} \tag{3-27}$$

where g_0 is an upper bound of the modelling error η.

3-4-2 Leakage

Some of the problems described above come from the pure integrator in the adaptive law. It can be changed in the following manner :

$$\theta(k) = \theta(k-1) + \Gamma\epsilon(k)\phi(k) - \sigma_l(k)\Gamma\theta(k) \tag{3-28}$$

where $\sigma_l(k) > 0$ should be designed. The easiest is to take $\sigma_l(k) = \sigma$. Since σ_l only is needed when the parameters are out of bounds, a better choice of $\sigma_l(k)$ would be:

$$\sigma_l(k) = \sigma, \quad \sigma = \begin{cases} 0 & \text{if } |\theta| < M_0 \\ \sigma_0 & \text{if } |\theta| \geq M_0 \end{cases} \tag{3-29}$$

where $M_0 > 0$ and $\sigma_0 > 0$ are design constants. M_0 should be chosen large enough such that $M_0 > |\theta^*|$. This is a discontinuous switching, it can also be made continuous of course.

3-4-3 Static Normalization

It can happen with an adaptation law without normalization that the signals are not bounded. To overcome this, adaptation laws with normalization can be used. A normalization signal is used as described in [1]. Instead of using the estimation error described by Equation (3-19), the following normalized error equation is used:

$$\epsilon = \frac{\zeta - \hat{\zeta}}{m^2} \tag{3-30}$$

where $m^2 = 1 + n_s^2$. n_s is the normalizing signal and is described by:

$$n_s^2 = \phi^\top \phi, \text{ or} \tag{3-31}$$
$$n_s^2 = \phi^\top P \phi \tag{3-32}$$

where $P = P^\top > 0$

3-4-4 Dynamic Normalizing

To make sure the controlled plant will be stable, all signals should be bounded. In Section 3-4-3 the static normalization was explained. But if η is not bounded from above by $|u|$, another normalization signal needs to be designed. If η is related to u through some transfer it can be shown that all signals will be bounded using the following dynamic normalizing signal [1]:

$$\epsilon(k) = \frac{\zeta(k) - \hat{\zeta}(k)}{m(k)^2} \tag{3-33}$$

$$m(k)^2 = 1 + u(k-1)^2 + n_s(k)^2, \quad n_s(k)^2 = m_s(k) \tag{3-34}$$

$$m_s(k) = m_s(k-1) - \left(\delta_0 m_s(k-1) + u(k-1)^2\right)T_s, \quad m_s(0) = 0 \tag{3-35}$$

where $\delta_0 > 0$. Another normalization function can be used as well:

$$\epsilon(k) = \frac{\zeta(k) - \hat{\zeta}(k)}{m(k)^2} \tag{3-36}$$

$$m(k)^2 = 1 + \phi(k)^\top \phi(k) + n_s(k)^2, \quad n_s(k)^2 = m_s(k) \tag{3-37}$$

$$m_s(k) = m_s(k-1) - \left(\delta_0 m_s(k-1) + \phi(k)^\top \phi(k)\right)T_s, \quad m_s(0) = 0 \tag{3-38}$$

3-5 Conclusion

In this Chapter we have seen how the standard pole placement controller is designed using Ackerman's formula. The same algorithm is used for the APPC which can be combined with different adaptation laws. Namely the GB method, RLS method and these two methods combined with robust adaptive laws. These adaptation laws estimate each time step new state space matrices which are used to adapt the controller gain matrix K.

Chapter 4

Simulations

In this chapter the influence of the different parameters for the adaptive controllers is studied. The controllers are tuned by looking for an optimum for the following measures:

- Convergence time of adaptive controller
- Control effort
- Deviation of the angles

Besides finding the optimum, it is also investigated if a certain parameter can induce an unstable system. This knowledge will be used for tuning the adaptive controllers on the real Ball Balancing Robot (BBR). Finally, the different adaptive controllers are compared with each other by changing the plant during the simulation.

Since the BBR is working in discrete time, all the simulations described in this Chapter are done also in discrete time. The BBR works with a fixed time step of 0.005 s, therefore the time step T_s for the simulations is 0.005 s as well.

4-1 Simple pole placement

Before an adaptive pole placement controller can be implemented, a stable simple pole placement controller needs to be designed. In this section the design and the results of a simple pole placement controller are described.

The method described in Section 3-1 is used for placing the poles. The dominant poles are calculated using Equation (3-7) with a damping ratio of $\zeta = 0.7$ and a natural frequency of $\omega_n = 1.8$. The location of the poles are:

$$p_{1,2} = -1.26 \pm 1.29i \tag{4-1}$$

The other poles are chosen a few times faster than the dominant poles. These are:

$$p_3 = -3.78, \;\; p_4 = -3.91 \tag{4-2}$$

For the simulation an initial value for the states is chosen as follows: $x_0 = [0\ 0\ 0.17\ 0]$. This is equivalent to the robot having an initial angle of 10 degrees. This is also the maximum allowable angle for the BBR before the electronics shut down as a safety measure.

The results of the states and the torque can be seen in Figure 4-1 and Figure 4-2 respectively. As can be seen the closed loop system is stabilized by the controller. If the poles would be chosen faster (more negative) the torque would be higher and therefore the system would react faster. But, since the maximum applied torque on the wheels is around 4 Nm and the maximum allowable angle is 10 degrees, the poles can not be chosen faster. On the other hand, if the poles are chosen slower the deviation of the angle of the ball would be bigger. Therefore the placement of poles as described above is a compromise between the deviation of the angles and the control effort.

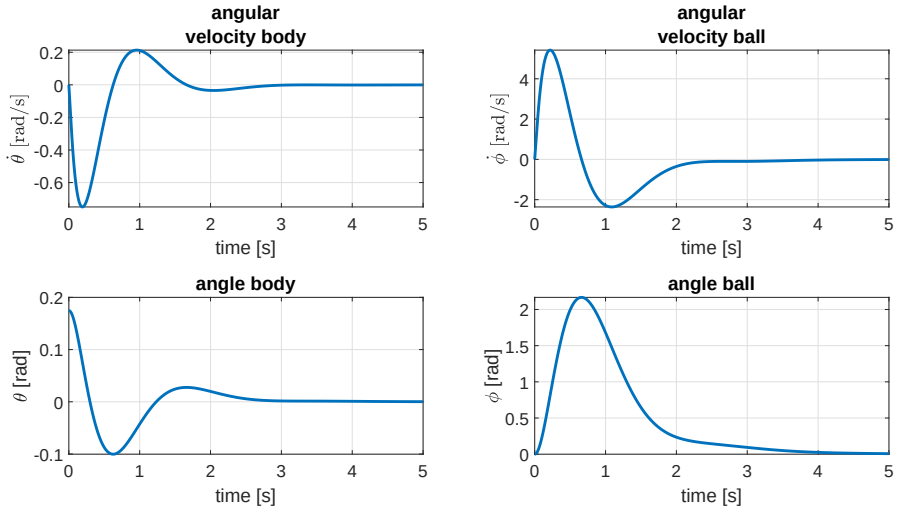


Figure 4-1: Response for a initial angle of the robot of 10 degrees.

4-2 Tuning adaptive controllers

To tune the adaptive controllers the simplified model without friction (Equation (3-9)) is used. It is expected that the influence of changing the different parameters of an adaptive controller will be the same on the simplified model and as on the more complex model. Therefore, for simplicity and understanding it is more convenient to use the simplified model.

For the simulations θ^* (the real parameter values) are calculated using the parameters given in Table 2-1. The initial values for $\theta(0)$ are chosen to be 10% off the real values θ^*.

In Figure 4-3 the disturbance signal is shown which will be used to disturb the system. It consist of a triangular impulse for the disturbance on the input of the system, as if the robot

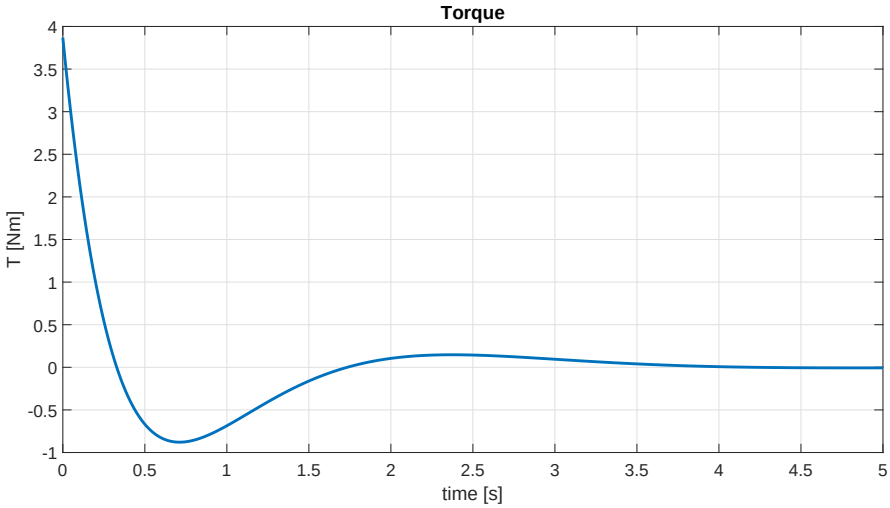


Figure 4-2: Torque for a initial angle of the robot of 10 degrees.

is pushed. So the disturbance torque is added to the calculated input torque. The signal is repeated each 5 seconds such that the system is excited enough to let the adaptive controller converge.

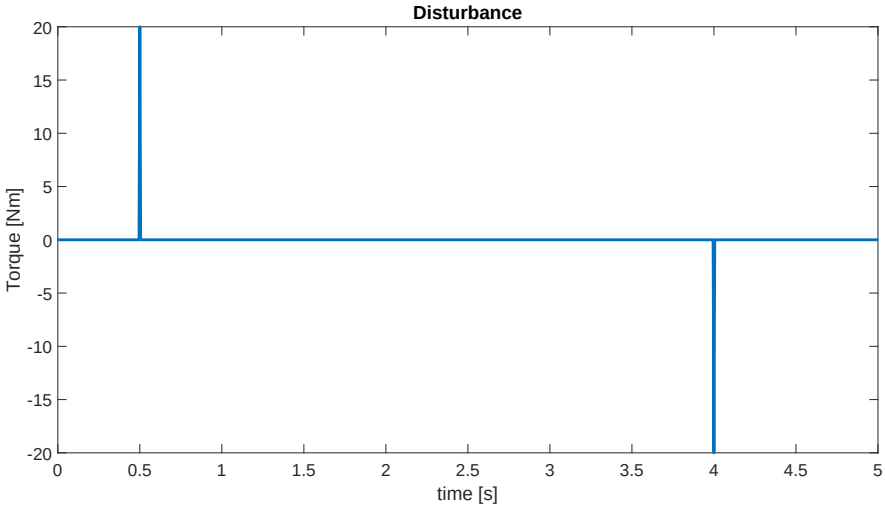


Figure 4-3: Disturbance signal for simulations. Signal will be repeated each five seconds.

4-2-1 Gradient based method

For the Gradient Based (GB) method (described in Section 3-3-1) the influence on the controller performance is studied for the following subjects:

- Adaptation gain Γ
- Placement of the poles

- Initial values of model parameters $\theta(0)$

Adaptation gain Figure 4-4 shows the results of the parameter estimation using different adaptive gains. The lower Γ consist of the following values:

$$\Gamma_{low} = \begin{bmatrix} 2 \cdot 10^5 & 0 \\ 0 & 2 \cdot 10^3 \end{bmatrix}$$

The higher Γ is: $\Gamma_{high} = 5\,\Gamma_{low}$.

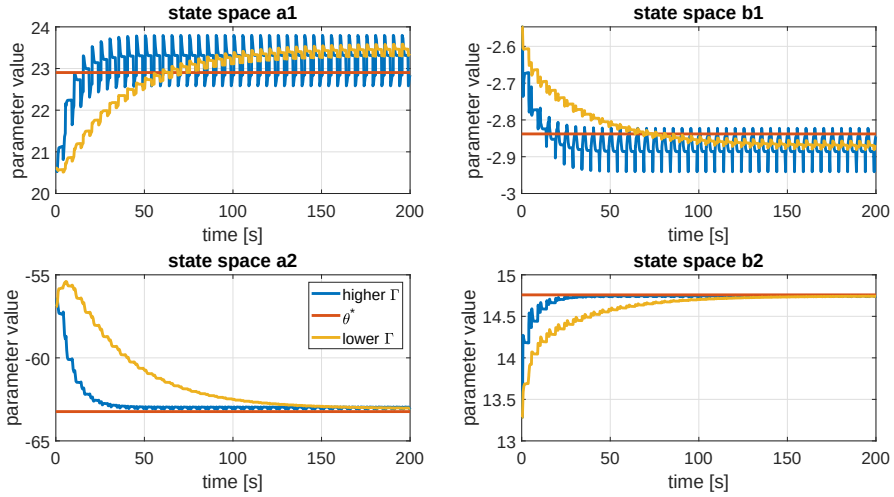


Figure 4-4: Difference in the parameter estimation for a high and low adaptation gain Γ.

The value of the adaptation gain indicates how fast the adaptive controller adapts. The higher the value, the faster the adaptive controller converges. But when the adaptive controller has converged and the system is again perturbed, the parameter estimations will have a bigger overshoot. This can be seen in Figure 4-5.

This more aggressive reaction on a perturbation also is seen in Figure 4-6 which shows the controller gains. The values where the parameters converges to, are the same for a higher and lower Γ. There is almost no difference in the control effort and the deviation of the angles when changing the adaptation gain. Therefore these are not shown.

If the gains are too high the system can become unstable, so there will be a certain optimum for the adaptive gain Γ. For now the lower Γ is used such that the influence of the other parameters are better visible.

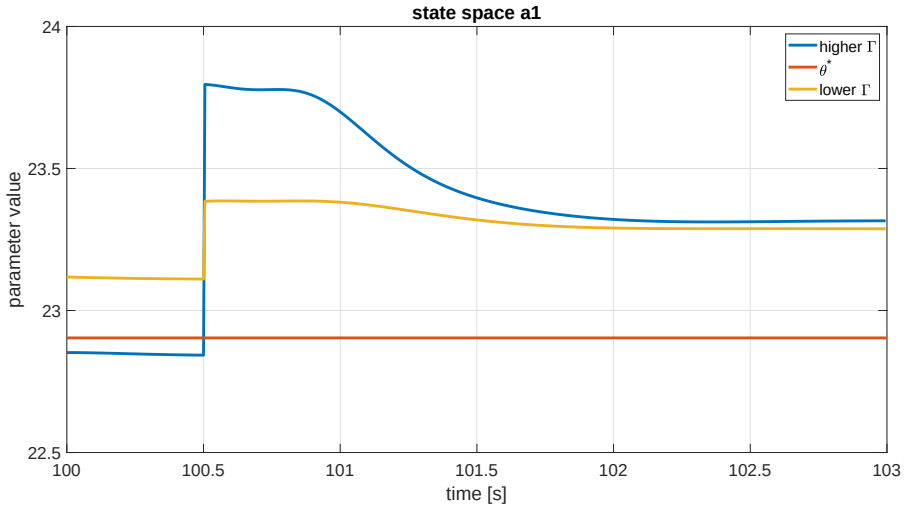


Figure 4-5: Difference in the parameter estimation for a high and low adaptation gain Γ when the adaptive controller has converged.

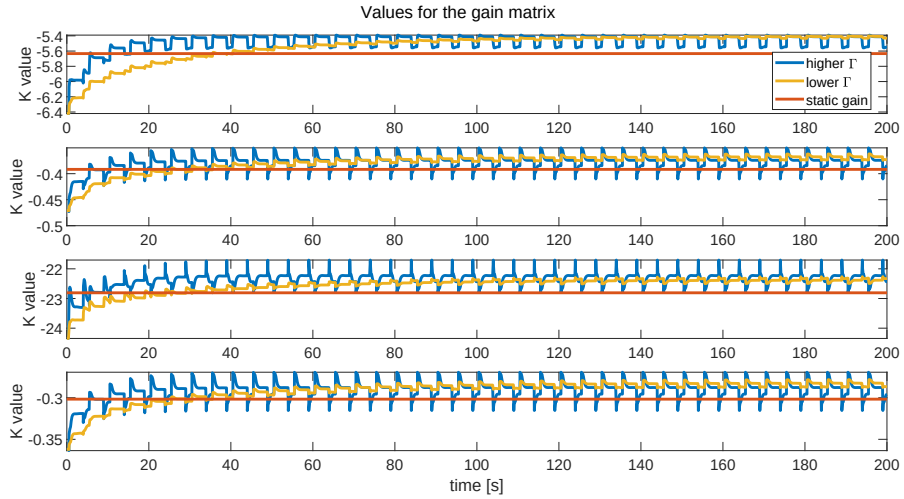


Figure 4-6: Difference in the controller gains K for a high and low adaptation gain Γ. The non adaptive static gain is shown as reference.

Placement of the poles When changing the location of the poles, the system will react differently of course. This can be seen in Figure 4-7. Here the slow poles correspond to the same poles describes in Equation (4-1) and (4-2):

$$p = -1.26 \pm 1.29i \quad -3.78 \quad -3.91 \tag{4-3}$$

The fast poles correspond to dominant poles with a natural frequency ω_n which is four times bigger such that the poles are located as follows:

$$p = -5 \pm 5.14i \quad -15.12 \quad -15.62 \tag{4-4}$$

The faster the poles, the faster the system reacts, the faster system is stabilized and the bigger the overshoot for especially the angle of the robot.

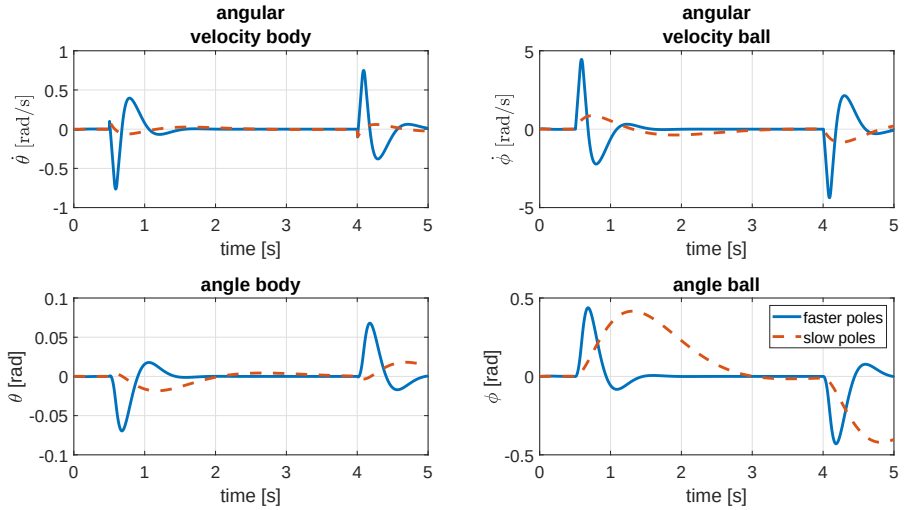


Figure 4-7: Influence of changing the location of the poles on the states of the system.

There is a maximum torque implemented in the simulations of 4 Nm since this is also the case for the real BBR. This maximum is reached with the faster poles as can be seen in Figure 4-8.

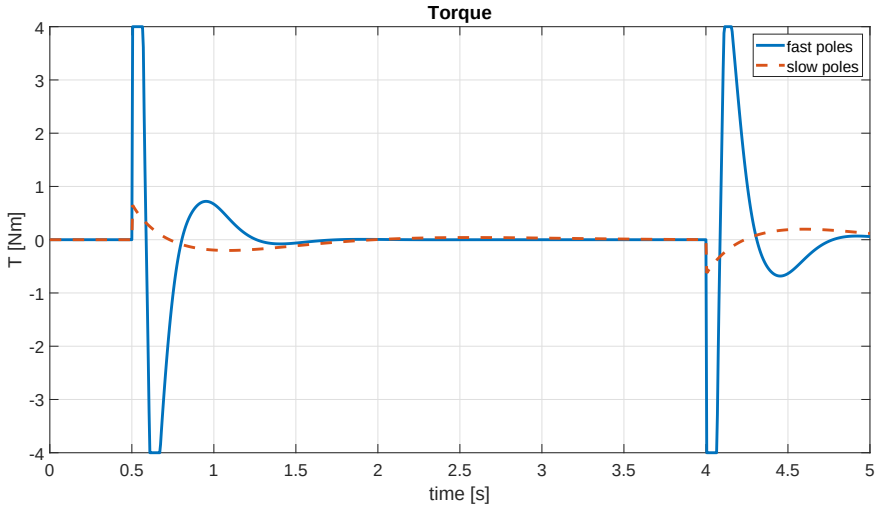


Figure 4-8: Influence of changing the location of the poles on the input of the system.

It can also be seen that the faster poles have a bigger control effort. The bigger control effort and the bigger overshoot of the angle of the robot can be seen also in Figure 4-9. In this Figure the estimation of the parameters θ are shown. Since the the control effort and overshoot is bigger, the estimation of the parameters will go faster. Again there is an optimum for the location of the poles between the convergence time of the adaptive controller, the control effort and the deviation of the angles. For now the slower poles are used.

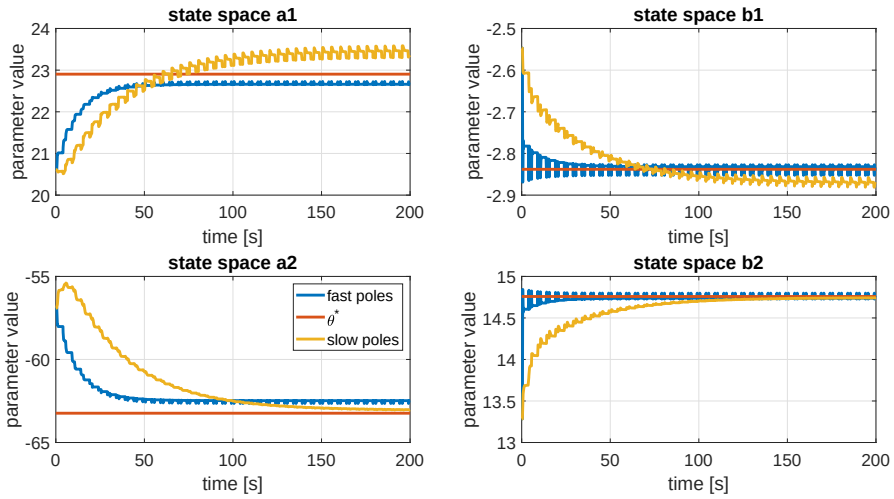


Figure 4-9: Influence on the model parameter estimations when changing the location of the poles.

Initial values of model parameters $\theta(0)$ In this paragraph the influence of the initial values of the model parameters is studied. At the end of the paragraph we want to know what a good value is to chose for $\theta(0)$ and if there is a region for $\theta(0)$ such the adaptive controller does not converge and becomes unstable.

Figure 4-10 shows the influence on the parameter estimation θ when using either $\theta(0) = 1.1\,\theta^*$ or $\theta(0) = 0.9\,\theta^*$. Besides the mirror effect from changing the initial values, a big difference can not be seen. For both values, the adaptive controller converges with the same speed. Moreover for the deviation of the angles and the control effort again a mirror effect is seen. But the amplitude for the signals is almost the same for the different initial values. This can be seen in Table 4-1 where the RMS values are shown of the body angle, the ball angle and the torque applied on the wheels.

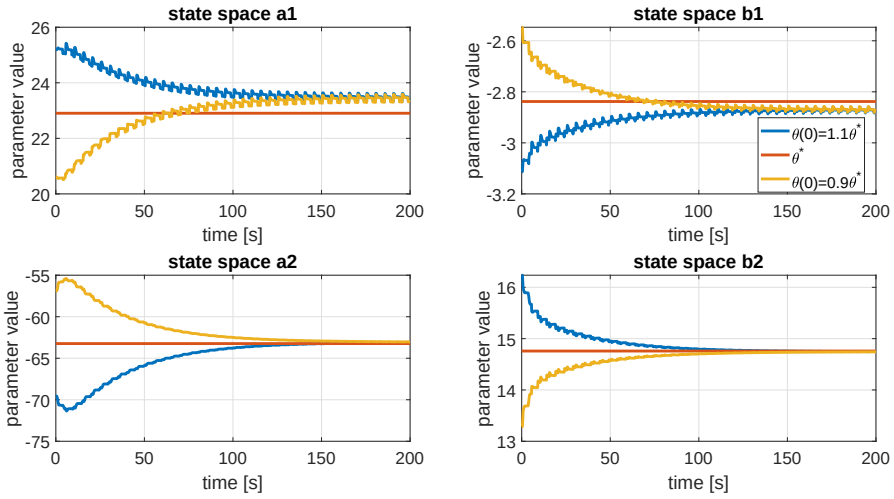


Figure 4-10: Influence on the parameter estimation when changing the initial value $\theta(0)$.

Table 4-1: RMS values for body angle, ball angle and torque when changing initial values $\theta(0)$.

	RMS body angle	RMS ball angle	RMS Torque
$\theta(0) = 1.1\,\theta^*$	0.0102	0.2752	0.1349
$\theta(0) = 0.9\,\theta^*$	0.0102	0.2699	0.1362

Since there is not a big difference between using $\theta(0) = 1.1\,\theta^*$ or $\theta(0) = 0.9\,\theta^*$ and the actual value θ^* is unknown, a bigger difference between the initial is chosen: $\theta(0) = 1.5\,\theta^*$ and $\theta(0) = 0.5\,\theta^*$. Again a big difference can not be seen between the two different initial values and therefore the results are not shown.

Only when $\theta(0)$ is chosen close to zero ($\theta(0) < 0.1\,\theta^*$) a big difference is noticed. The smaller the estimated parameters, the bigger the controller gain K resulting in bigger inputs. If $\theta(0)$ is chosen too small, the system becomes unstable since the input will be too big.

From this paragraph it can be concluded that the best value for $\theta(0)$ is the best possible estimate of θ^*. There is no region found close to θ^* which makes the adaptive controller unstable. A too big starting value $\theta(0)$ is not really possible, even a choice of $\theta(0) = 5 \cdot \theta^*$ did not make the system unstable. A to low $\theta(0)$ can result in an unstable system. If the initial values $\theta(0)$ happens to be to low, it can always be increased.

4-2-2 Recursive Least Squares method

With the Recursive Least Squares (RLS) method (described in Section 3-3-2) the following subjects are studied, specifically their influence on the performance of the controller when they are changed:

- Initial values of $P(0)$
- Placement of the poles
- Initial values of the model parameters $\theta(0)$

Initial values for P(0) The covariance matrix P is changing overtime (see Equation (3-24)) and therefore needs an initial value to start with. In this paragraph is studied if there is an value for $P(0)$ such that the system becomes unstable and if there is an optimum for $P(0)$.

In Figure 4-11 the influence is shown when using a lower and higher $P(0)$. The lower $P(0)$ comprises of:

$$P(0)_{low} = \begin{bmatrix} 1 \cdot 10^7 & 0 \\ 0 & 1 \cdot 10^4 \end{bmatrix} \tag{4-5}$$

The higher $P(0)$ is calculated as: $P(0)_{high} = 1000 \cdot P(0)_{low}$. As can be seen the higher $P(0)$ the faster the parameter estimation converges. There is no limit on how high this value can be made. In other words, the system does not become unstable.

There is not a big difference between a higher and lower $P(0)$ when looking at the control effort and the deviation of the angles. This is shown in Table 4-2.

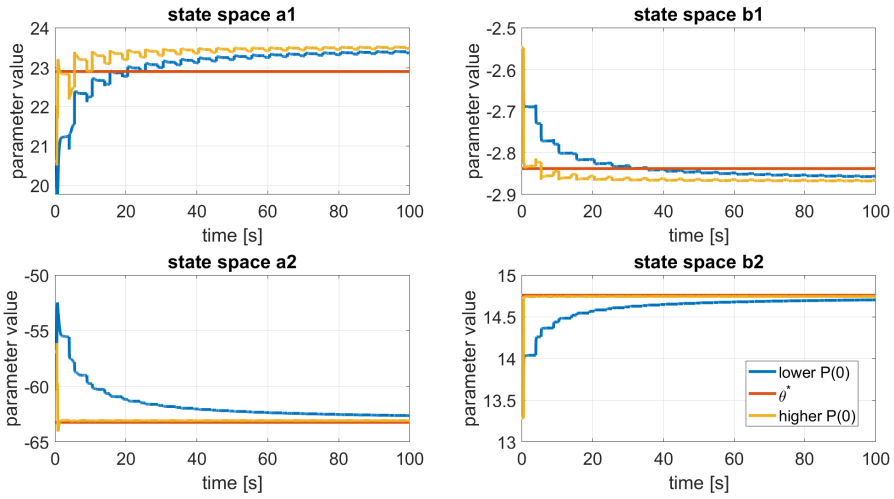


Figure 4-11: Influence on the parameter estimation when changing the initial value $P(0)$.

Table 4-2: RMS values for body angle and ball angle when changing initial values $P(0)$.

	RMS body angle	RMS ball angle	RMS Torque
$P(0)$ lower	0.0101	0.2686	0.1355
$P(0)$ higher	0.0101	0.2697	0.1353

To conclude it is better to use a higher $P(0)$ since the convergences time is smaller and there is not a notable difference in the control effort and the deviation of the angles. Even though, for the following simulations a lower $P(0)$ is used as described in Equation (4-5) since the influence of changing the other parameters can be studied better when using a lower $P(0)$.

Placement of the poles The same different poles are used as described in Equation (4-3) and Equation (4-4). Changing the poles using the RLS method has the same influence on the deviation of the angle (Figure 4-7) and the control effort (Figure 4-8) as when using the GB method.

The influence on the parameter estimation is shown in Figure 4-12. The same conclusion can be drawn for the GB and the RLS method when changing the location of the poles: Using the faster the poles described in Equation (4-4), the smaller the convergences time of the adaptive controller and the bigger the deviation of the angles and the control effort.

Initial values of the model parameters $\theta(0)$ The difference in using either $\theta(0) = 0.9\,\theta^*$ or $\theta(0) = 1.1\,\theta^*$ for an initial estimate of the parameters θ are shown in Figure 4-13. The same conclusions can be drawn when changing $\theta(0)$ using either the GB and the RLS method (see Section 4-2-1).

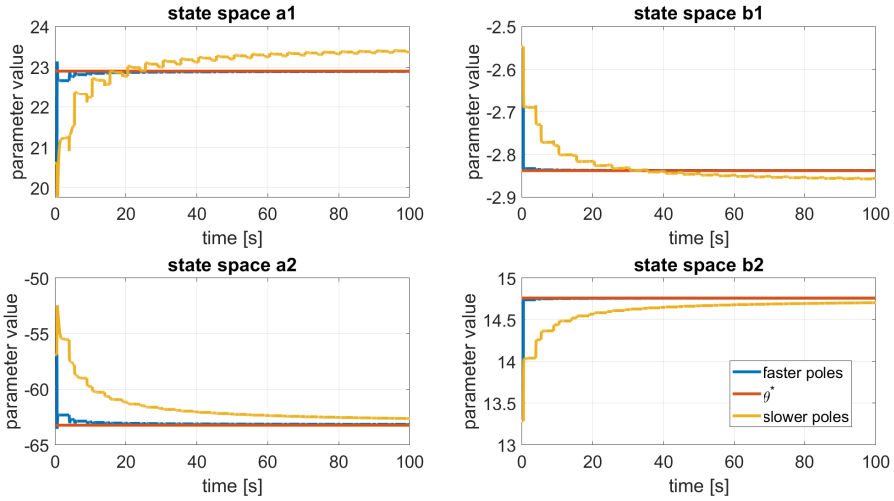


Figure 4-12: Influence on the parameter estimation when changing the location of the poles.

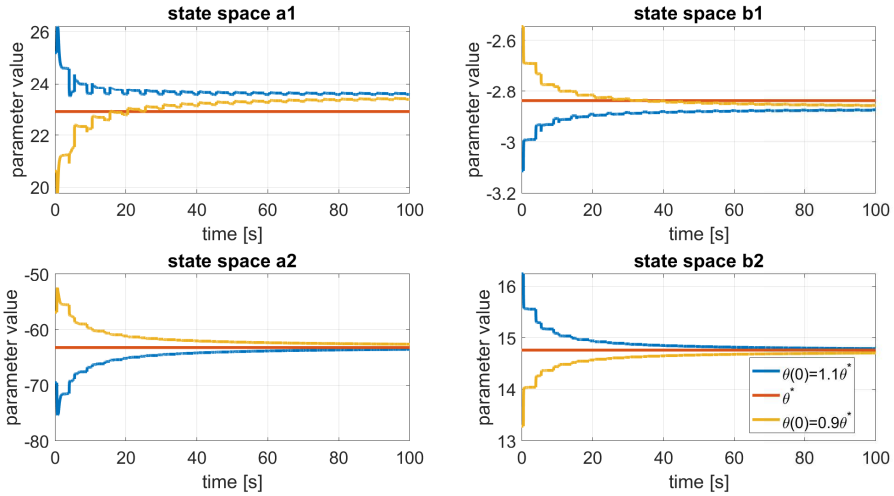


Figure 4-13: Influence on the parameter estimation when changing the initial value $\theta(0)$.

4-3 Robust adaptive laws

In Section 4-2 a linear known state space model without noise and the same known disturbances were used to study the influence of the different parameters of the adaptive controllers. In the real world, and thus on the BBR as well, we need to deal with a non-linear plant with noise and unknown disturbances. This section shows what the influence is of these real world problems on the adaptive control. First, noise is added to the system, secondly different disturbances are applied to the system. Before we study if the problems can be solved using robust adaptive laws, we need to know what those problems exactly are using the GB method. Those problems are described in Section 4-3-1 and Section 4-3-2. Afterwards the influence of the robust adaptive laws are studied in Section 4-3-3. These robust adaptive laws are an extension on the non robust adaptive laws GB method and the RLS method.

4-3-1 Noise

Measurement noise is added to the states of the system in the form of band limited white noise. In Figure 4-14, a comparison between the GB method is shown for the cases where there is noise and where there is no noise. Besides a slightly noisy estimation of the parameters, the parameters converge to a different value.

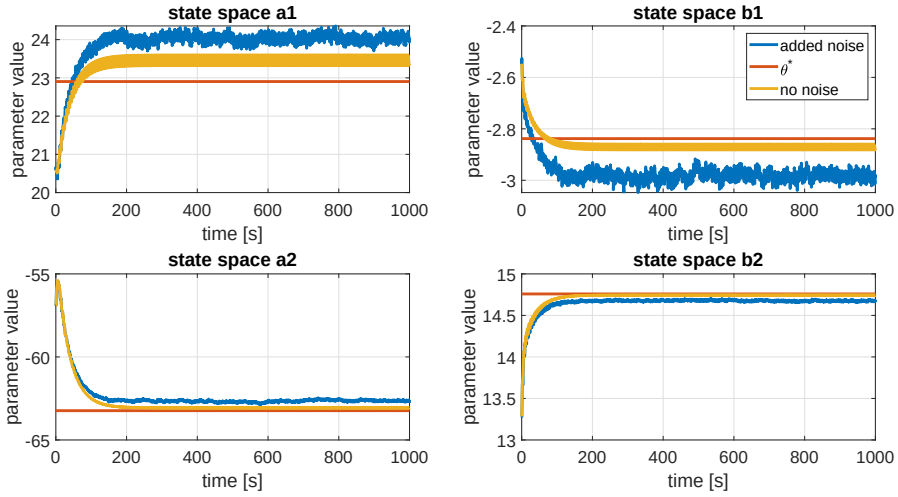


Figure 4-14: Influence on the parameter estimation when measurement noise is added to the system. Each five seconds a disturbance.

That the parameters converge to a different value due to added noise is shown more clearly in Figure 4-15. In this Figure the influence when the system is not disturbed and only noise is added to the system is shown. The phenomenon shown is called parameter drift and was first studied by Egardt [19].

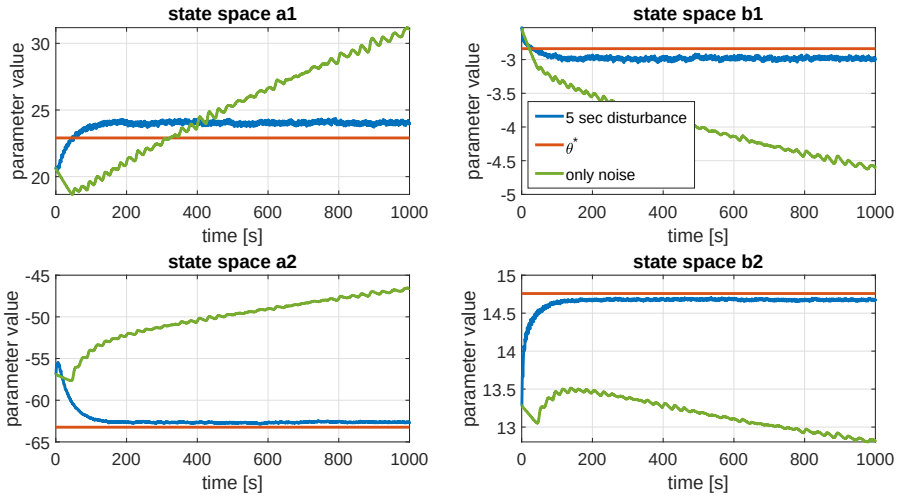


Figure 4-15: Influence on the parameter estimation when measurement noise is added to the system.

So there are two problems encountered when noise is added to the system, namely: parameter drift (when only noise is used), and the parameters do not converge to the same values.

4-3-2 Disturbance

In this paragraph it is studied more thoroughly what the influence is when a different disturbance is applied on the system. In Figure 4-16 the results are shown.

The "1 amplitude" line corresponds when the disturbance is used showed in Figure 4-3. The other lines are obvious and do not need an explanation.

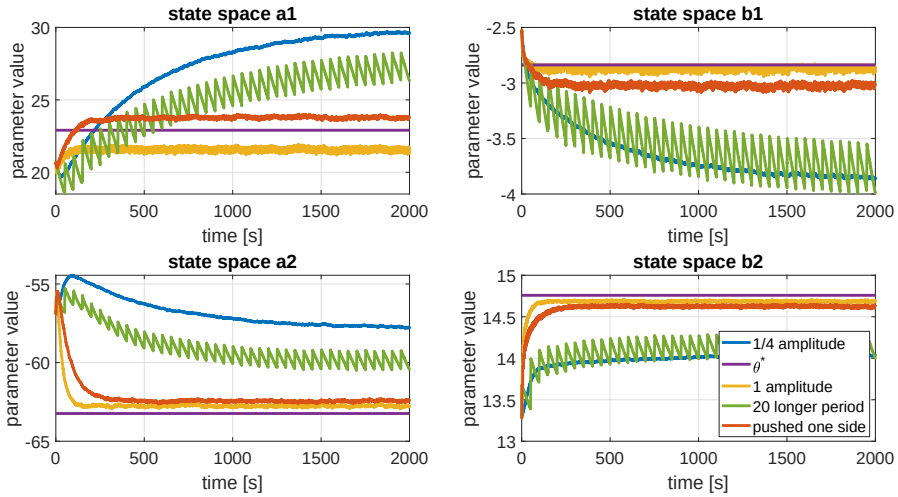


Figure 4-16: Influence on the parameter estimation when different disturbances are used.. GB method.

Figure 4-16 shows that a different disturbance results in a different parameter estimation. A different parameter estimation will result of course in a different gain matrix as well. This is shown in Figure 4-17.

These different gain matrices have influence how the system will react, in Figure 4-18 and Figure 4-19 the results are shown for the two most outlying gain matrices showed in Figure 4-17 and the gain matrix calculated using normal pole placement:

- gain matrix 1 = [-3.3 -0.27 -17.3 -0.22] is the gain matrix for 1/4 amplitude
- gain matrix 2 = [-5.19 -0.37 -21.9 -0.29] is the gain matrix for 1 amplitude
- normal pp = [-5.63 -0.39 -22.80 -0.30] is the gain matrix for normal pole placement

As is shown, the system reacts differently when different disturbances are used, even though the parameters do not converge to the same value, the system is still stabilized.

In the next paragraphs it is studied if the robust adaptive laws can solve the problem that the parameters converge to different values when noise is added to the system and different

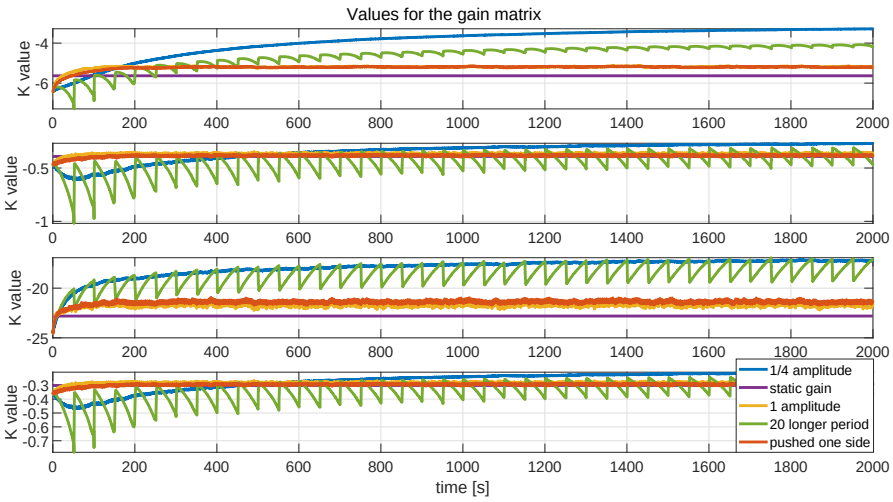


Figure 4-17: Influence on gain matrix when using different disturbances. The non adaptive static gain is shown as reference. GB method.

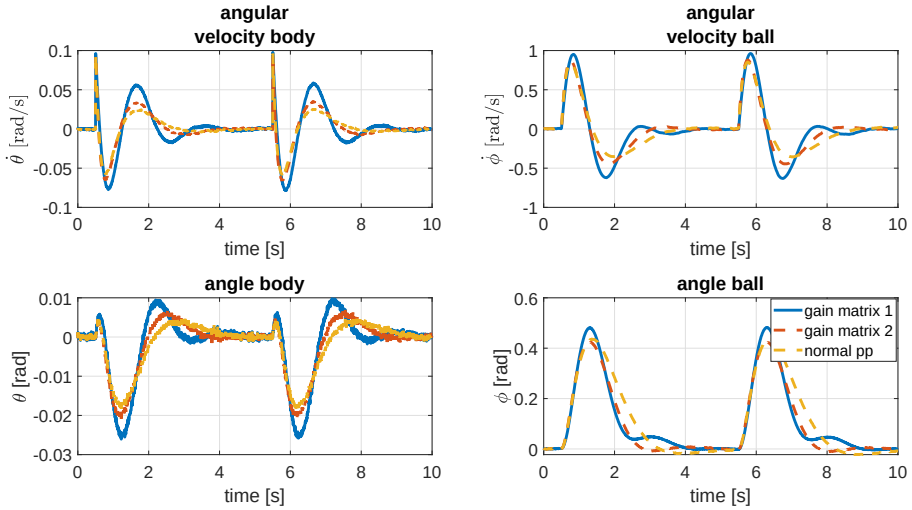


Figure 4-18: Influence on the outputs when using different gain matrices due to different disturbances used to perturb the system and adapt the system. To compare the different controllers the same perturbations is used for 10 seconds.

disturbances are used.

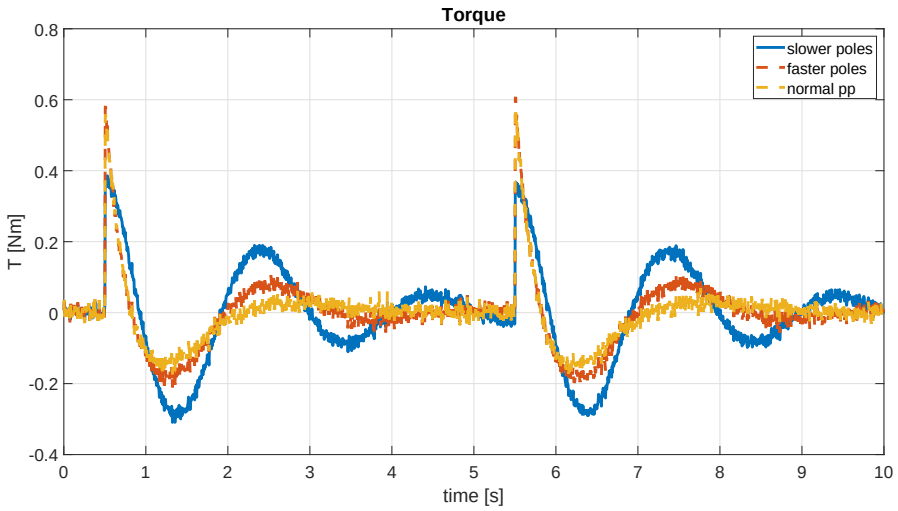


Figure 4-19: Influence on the inputs when using different gain matrices due to different disturbances used to perturb the system and adapt the system. To compare the different controllers the same perturbations is used for 10 seconds.

4-3-3 Implementing robust adaptive laws

In Section 4-3-1 and Section 4-3-2 the problems when measurement noise is added or a different disturbances are used is described. The following paragraphs show if robust adaptive laws can solve those problems. The GB method is extended with the following robust adaptive laws:

- Dead zone
- Leakage
- Static normalization
- Dynamic normalization

These laws are described in Section 3-4.

Dead zone The algorithm used is described in Section 3-4-1. Since there are two outputs, the angle of the robot and ball, there are two estimation errors ϵ_1, ϵ_2 and therefore two different values for $g_{0,1}$, $g_{0,2}$. Those are $g_{0,1} = 5 \cdot 10^{-4}$, $g_{0,2} = 1 \cdot 10^{-3}$.

In the three figures (Figure 4-20, Figure 4-21, Figure 4-22) below the results are shown for GB method with additional dead zone.

Dead zone does not help when noise is added to the sensor data and each 5 seconds there is a disturbance on the system. When the system is not disturbed, but only noise is added to the sensor data, the parameter estimation drifts away. Dead zone slows down this process, but can not stop it totally with using the values described above for g_0. When those values are increased, it can be made sure that there is no parameter drift when there is only added

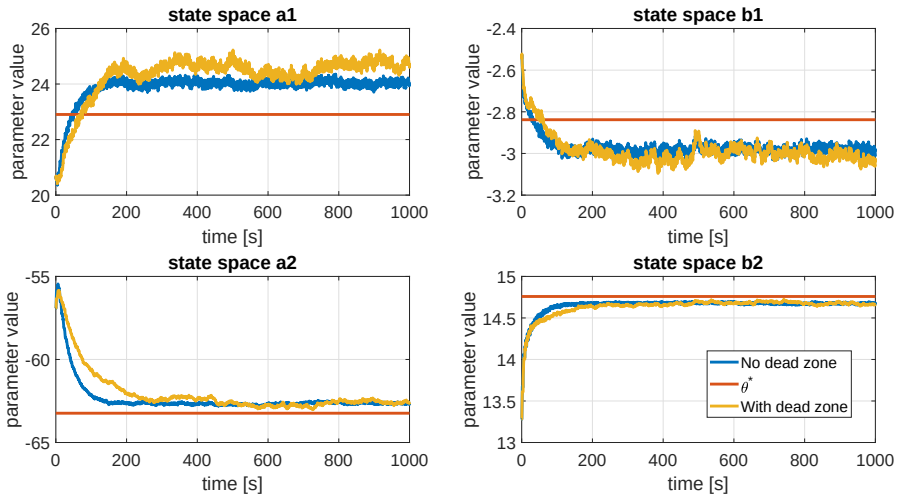


Figure 4-20: Influence on the parameter estimation when using dead zone. Each five seconds a disturbance.

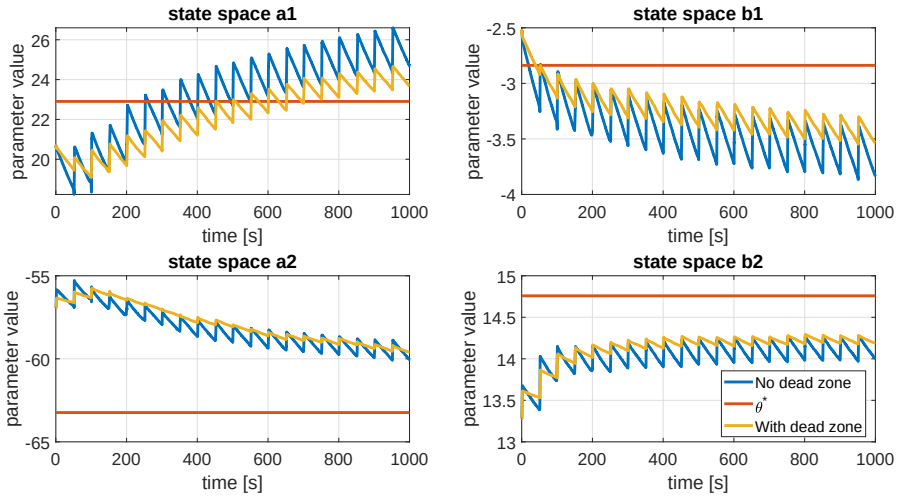


Figure 4-21: Influence on the parameter estimation when using dead zone. Each 50 seconds a disturbance.

noise and no disturbances. If the system is also perturbed by disturbances, a higher value for g_0 will give worse results.

Dead zone implemented on RLS, did not improve the results. It gives more or the less the same results with GB method and RLS.

From the results showed above, it can be concluded that the adaptive controller does not give better results when dead zone is implemented.

Leakage Leakage is used to make sure the system parameters keeps bounded using the adaptation law described in Section 3-4-2.

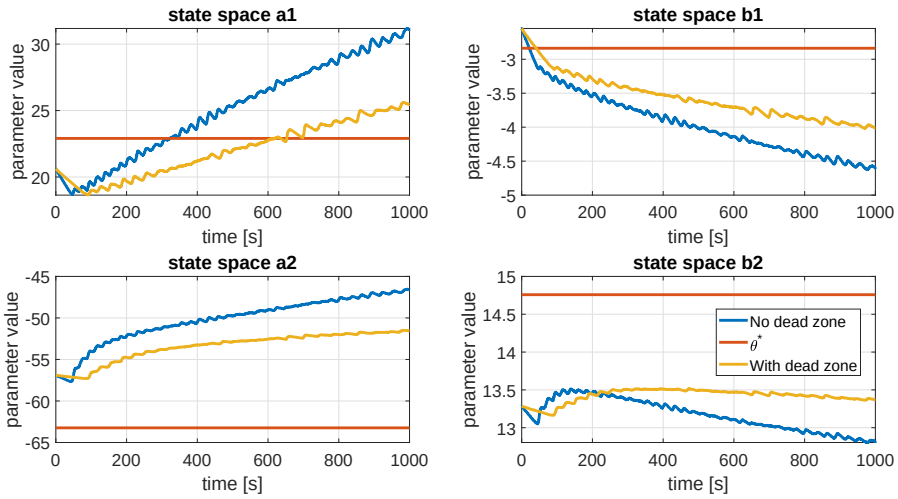


Figure 4-22: Influence on the parameter estimation when using dead zone. No disturbance, only noise on states.

The three figures below show the results of the GB method with added leakage described by the adaptive law in Equation (3-28) using a value for $\sigma_l(t) = 1 \cdot 10^{-10}$. Different disturbance signals are used.

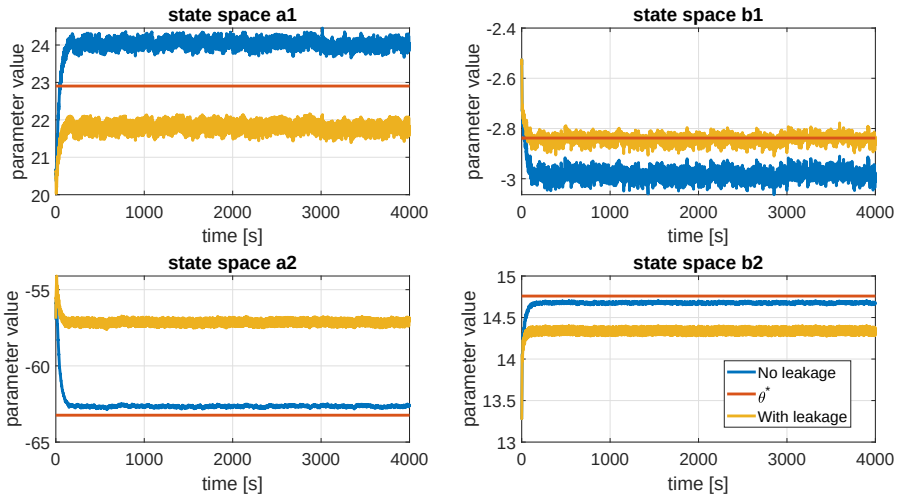


Figure 4-23: Influence on the parameter estimation when using leakage Each five seconds a disturbance.

As can be seen in the Figures, the parameter estimation converge to a different value when using leakage. When no disturbance is applied to the system but only noise is added, this difference is the biggest. Even though the parameter estimation does converge, this is not favourable.

Equation (3-29) was used as well as adaptive law. This adaptive law did not give better results and are therefore not shown. Leakage was applied also using RLS, the results were

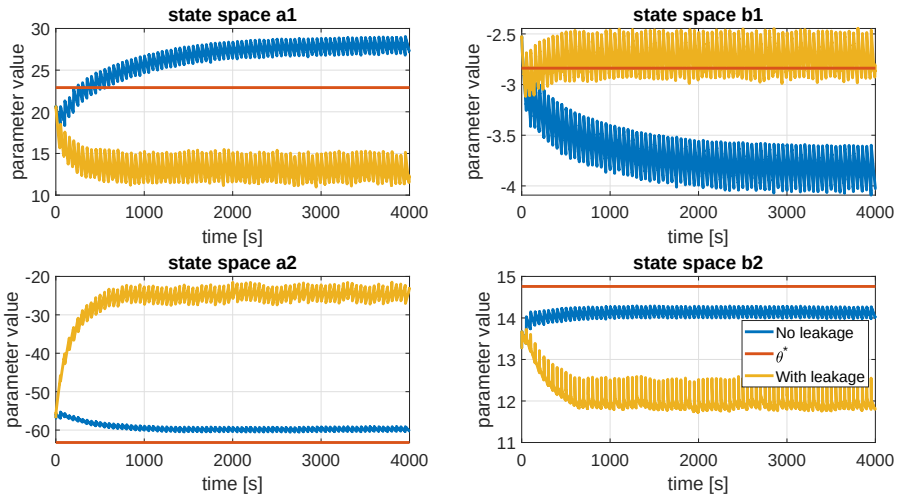


Figure 4-24: Influence on the parameter estimation when using leakage Each 50 seconds a disturbance.

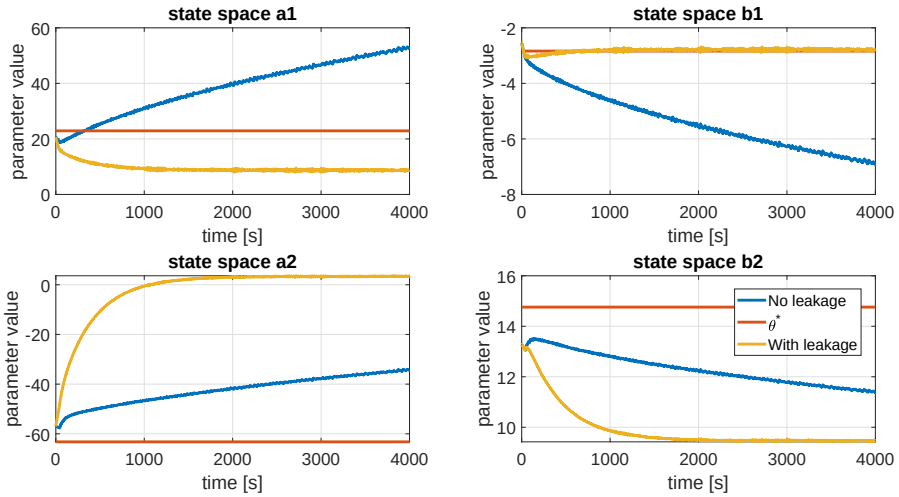


Figure 4-25: Influence on the parameter estimation when using leakage. No disturbance, only noise.

similar and therefore not shown.

From the results, it can be concluded that leakage does not make the adaptive controller perform better.

Static Normalization A normalization signal is used as described in Section 3-4-3.

P is tuned as:

$$P = \begin{bmatrix} 25 \cdot 10^6 & 0 \\ 0 & 1 \cdot 10^6 \end{bmatrix} \tag{4-6}$$

the order of magnitude of the body angle is 25 times smaller than the torque, therefore P is dimensioned as such.

The results of using the two different static normalization signals (Equation (3-31) and Equation (3-32)) are shown in Figure 4-26.

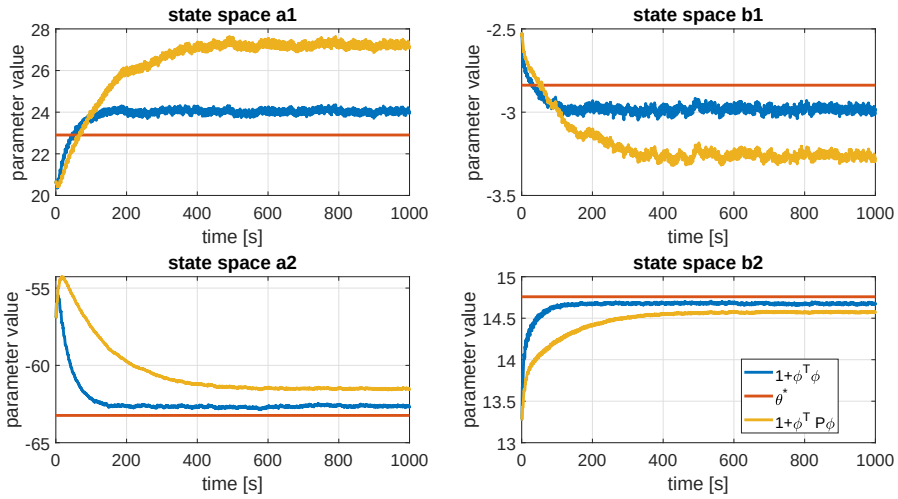


Figure 4-26: Influence on the parameter estimation when using either static normalization or gained static normalization. Each five seconds a disturbance.

Normalization makes the adaptation slower, the model parameters do not converge closer to the real values. This is the same for when using a different disturbance signal or no disturbance at all. With these results it can be concluded that static normalization does not make the adaptive controller better.

Dynamic Normalization Instead of using the static normalization function, a dynamic normalization function is used (Section 3-4-4). Both Equation (3-35) and Equation (3-38) are used with $\delta_0 = 0.0001$. Equation (3-35) gave better results, those results are shown in the figures below for the GB method with added dynamic normalization.

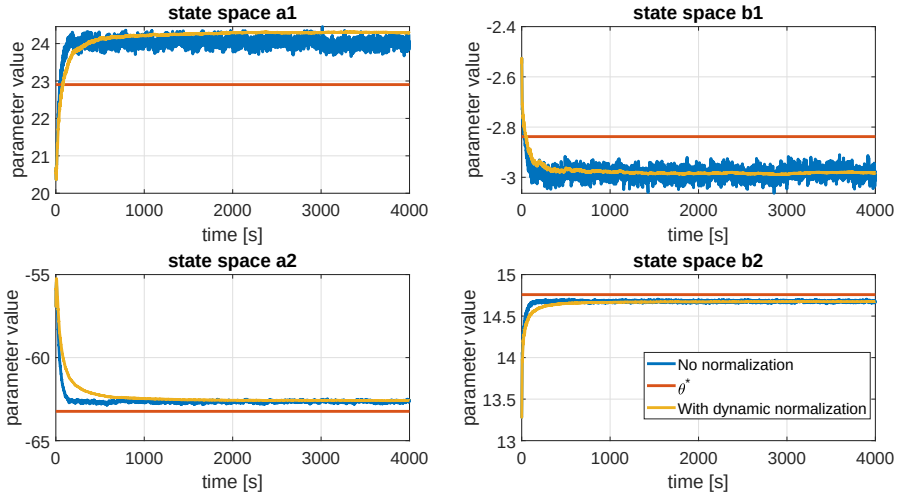


Figure 4-27: Influence on the parameter estimation when using dynamic normalization. Each five seconds a disturbance. GB

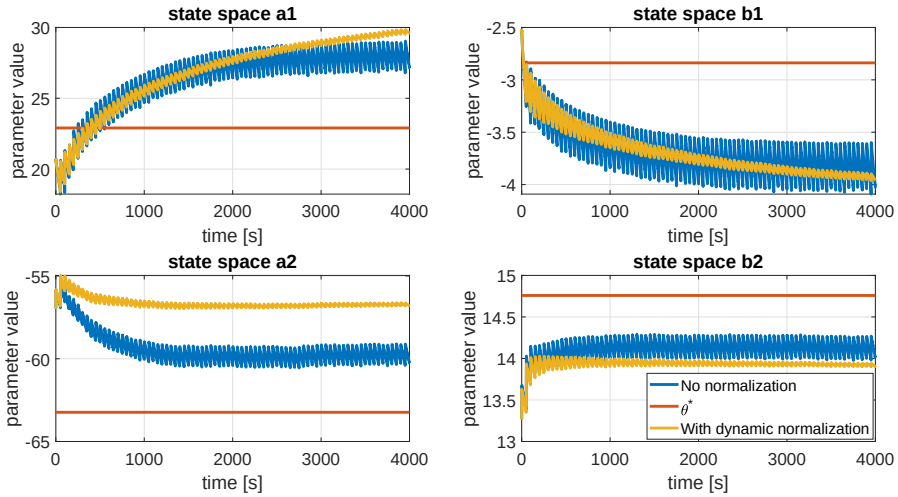


Figure 4-28: Influence on the parameter estimation when using dynamic normalization. Each 50 seconds a disturbance. GB

The results for the parameter estimation looks better when using dynamic normalization. Therefore the results are shown also for the gain matrix K (Figure 4-30) and the in- and outputs of the system (Figure 4-31 and Figure 4-32 respectively). Each five seconds there is a disturbance. For the in- and outputs, only the last 10 seconds of the simulations are showed.

As can be seen is the gain matrix less noisy when using dynamic normalization. Even though, there is almost no difference for the in- and outputs when using dynamic normalization.

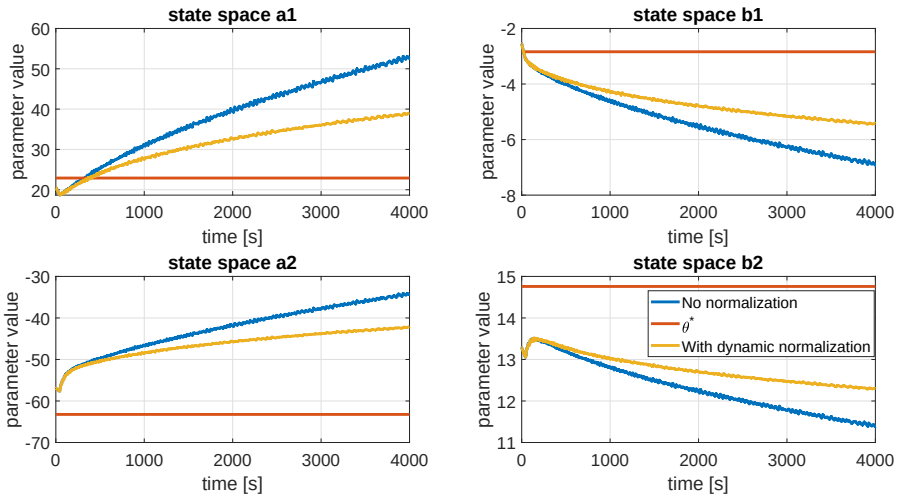


Figure 4-29: Influence on the parameter estimation when using dynamic normalization. No disturbance, only noise. GB

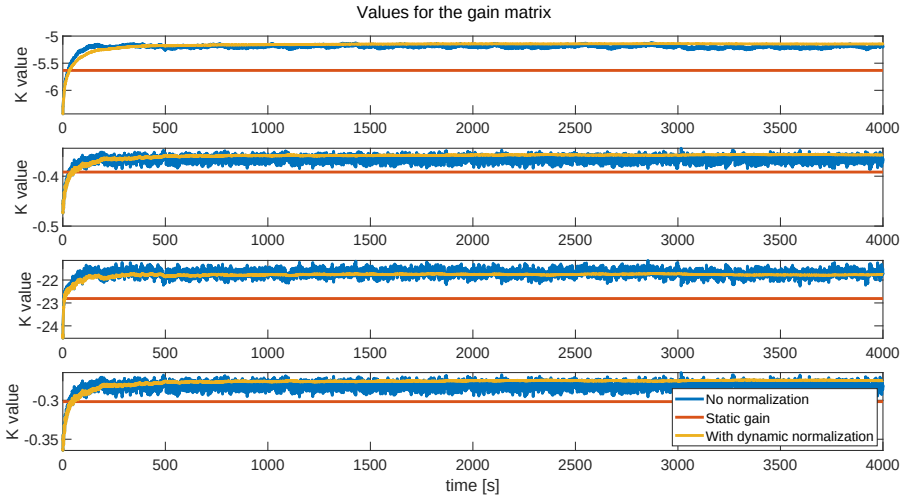


Figure 4-30: Influence on the gain matrix when using dynamic normalization. The non adaptive static gain is shown as reference. Each five seconds a disturbance. GB

Since the results for the GB method combined with dynamic normalization looks promising, the results are shown also for the RLS method in Appendix B.

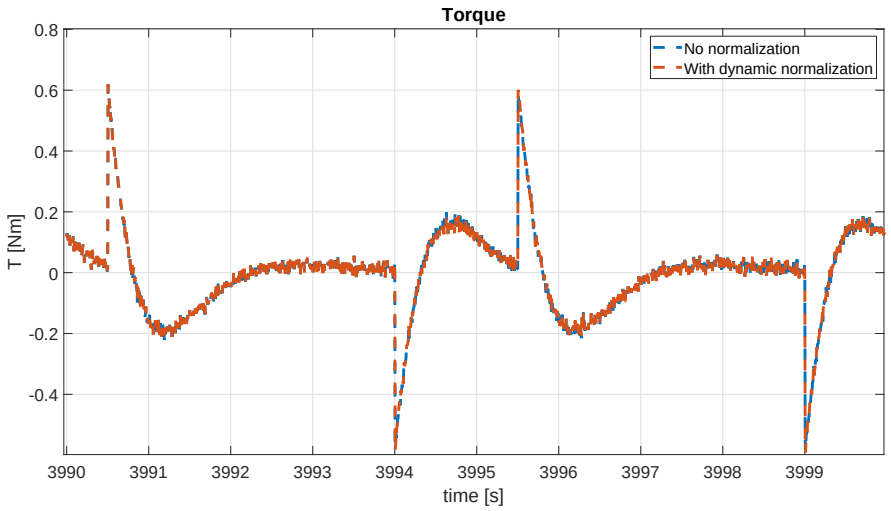


Figure 4-31: Influence on the torque when using dynamic normalization. Each five seconds a disturbance. GB

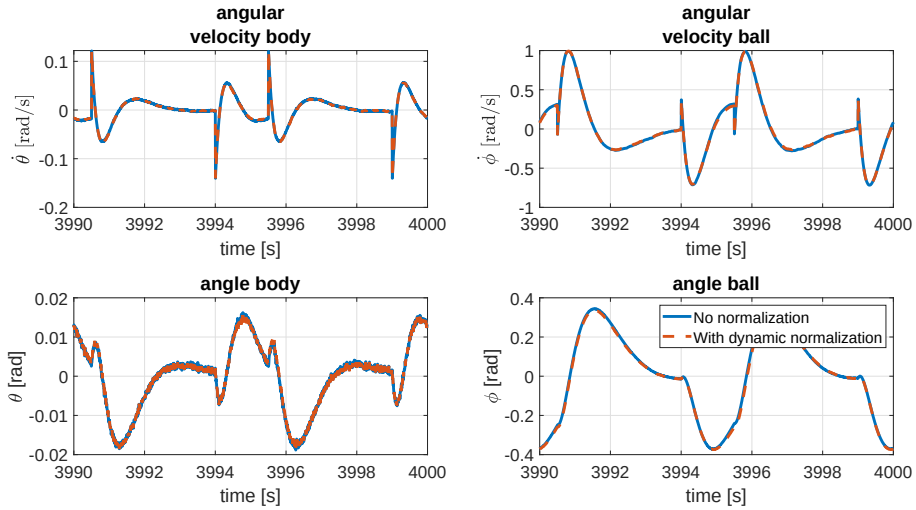


Figure 4-32: Influence on the states when using dynamic normalization. Each five seconds a disturbance. GB

4-4 Comparison

In Section 4-2 the different adaptive laws (GB method, RLS method) are tuned with the dynamic normalization as the only robust adaptive law which increased performance of the adaptive controller. With the knowledge gained from tuning, the controllers are compared with each other in this section.

In order to compare the different controllers simulations are performed. For these simulations a non-linear plant is used to simulate the responses to see how well the adaptive controllers can cope with the non-linear dynamics. Furthermore, the plant is changed at a certain point during the simulation by adding mass and changing the surface. For both situations the controllers are compared by how well they adapt to a changing environment.

4-4-1 Adding mass

First, five kg of mass is added to the top of the robot. This is influencing the total mass, the inertia and the centre of gravity (cog). This adjustment is seen back in the linearized state space system and therefore has an influence on the poles of the system.

The standard, not changing, pole placement is compared to when there is an ideal pole placement controller (which is changing with knowing that there is mass added). The mass is added half way the simulation at 500 seconds. The results are shown in Figure 4-33 and Figure 4-34.

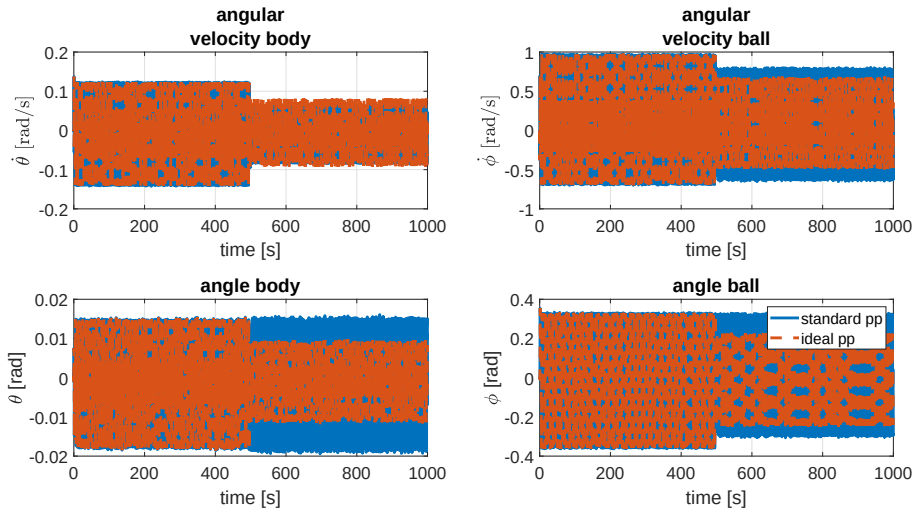


Figure 4-33: States of the system for when half way the simulation a 5 kg mass is added on top the robot. A standard, not changing pole placement controller is compared to an ideal pole placement controller which is changing with knowing there is mass added.

These results are used as a baseline. The ideal pole placement controller acts as the perfect adaptive controller. It's gain matrices are calculated for the two different systems, with and without added mass and the controller changes at the same time when the mass is added. In the real world it is unknown, if mass is added, when and how much mass is added. Therefore

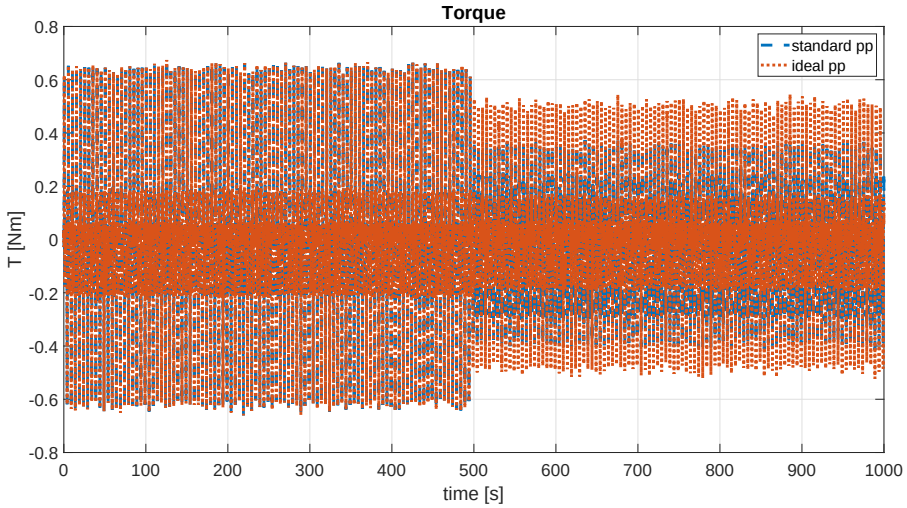


Figure 4-34: Input of the system for when half way the simulation a 5 kg mass is added on top the robot. A standard, not changing pole placement controller is compared to an ideal pole placement controller which is changing with knowing there is mass added.

the adaptive controllers are used. Four different adaptive controllers are compared to each other and the two baseline controllers:

- Standard pole placement
- Ideal pole placement
- GB method
- RLS
- GB method + dynamic normalization
- RLS + dynamic normalization

The results of the comparison is shown in the figures below. In Figure 4-35 the Root Mean Squares (RMS) values are plotted for the different controllers. The RMS values are calculated for sections of the simulations of 5 seconds. The RMS values for the total length of the simulation is shown in Table 4-3. This gives a better insight than plotting the values of the robot and ball angle and the torque itself. In Figure 4-36 the parameter estimation is shown for the different adaptive controllers.

As expected, the ideal pole placement controller gives the best results since it gives the lowest values for the RMS values. Both the GB method and GB method plus dynamic normalization perform really well, it converges faster than the RLS methods and has a RMS value close to the ideal pole placement controller. The RLS methods give worst results since the covariance matrix P is changing overtime and is only getting smaller. When the mass is added to the system, the covariance matrix P is so small that the parameters do not change as fast as the GB methods. This can be seen really clear in Figure 4-36.

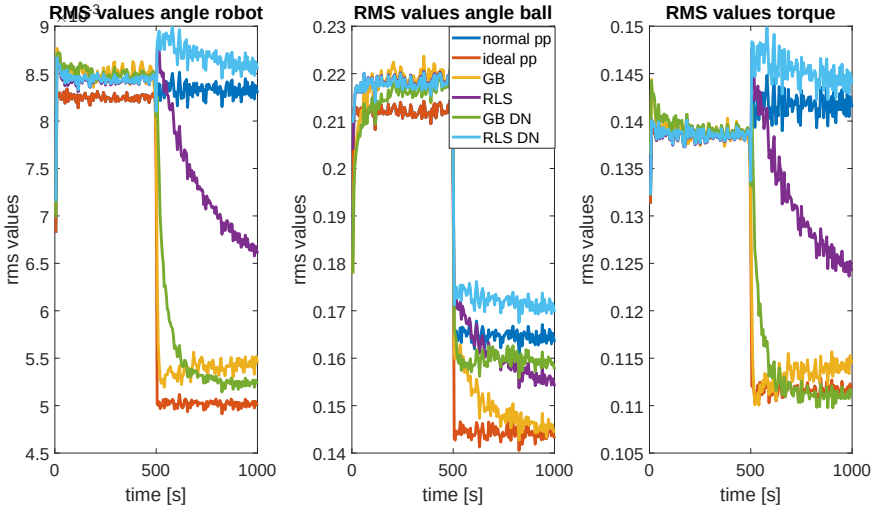


Figure 4-35: RMS values of the in- and outputs of the system when half way the simulation a 5 kg mass is added on top the robot. RMS values are calculated for sections of five seconds. Normal pole placement, ideal pole placement, GB method, RLS method, GB method + dynamic normalization and RLS method + dynamic normalization are compared.

Table 4-3: RMS values of the in- and outputs of the system when half way the simulation a 5 kg mass is added on top the robot. Normal pole placement, ideal pole placement, GB method, RLS method, GB method + dynamic normalization and RLS method + dynamic normalization are compared.

	angle robot	angle ball	torque
normal pole placement	0.0083	0.1897	0.1402
ideal pole placement	0.0068	0.1812	0.1258
GB	0.0071	0.1866	0.1267
RLS	0.0079	0.1914	0.1350
GB + dynamic normalization	0.0072	0.1890	0.1271
RLS + dynamic normalization	0.0086	0.1962	0.1420

Besides adding a mass of 5 kg, simulations were also done with adding 10 kg. The normal, not changing, pole placement controller became unstable while the adaptive controllers could cope with this added mass and change the controller such that the system stays stable.

In my opinion, the best adaptive controller is the GB method without adding dynamic normalization when adding mass. Both the GB methods with and without dynamic normalization give comparable results. Since the adaptive controller will be implemented on a real system, it would be favourable to have the least difficult controller without giving up on performance. Therefore the GB method is the best adaptive controller when adding mass.

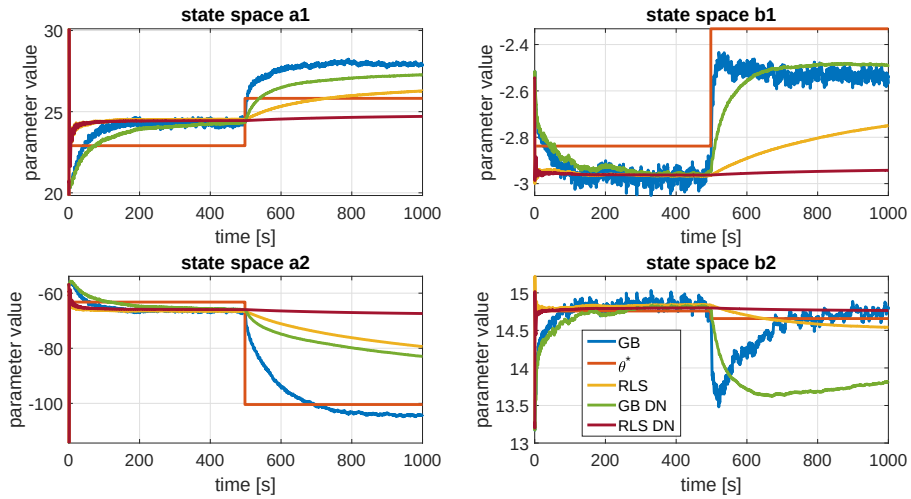


Figure 4-36: Parameter estimation when half way the simulation a 5 kg mass is added on top the robot. GB method, RLS method, GB method + dynamic normalization and RLS method + dynamic normalization are compared.

4-4-2 Changing the surface

Besides altering the mass of the robot, for Alten it is more interesting if the adaptive control can handle different surfaces where the BBR rides on. To do this the static friction (described by Equation (A-6)) between the floor and ball is changed after 500 seconds of simulating. The starting value is 0 Nm and it is changed to 0.3 Nm. Since the GB methods gave the best results for changing the mass, these methods are compared to the standard pole placement. The results are shown in Figure 4-37, where the RMS values are plotted. The RMS value is calculated for sections of five seconds. The RMS values for the total length of the simulation are displayed in Table 4-4. Figure 4-38 shows the parameter estimation.

Table 4-4: RMS values of the input and outputs of the system for when the static friction is changed. A standard, not changing pole placement controller is compared to the GB method and the GB method + dynamic normalization.

	angle robot	angle ball	torque
normal pole placement	0.0061	0.1913	0.1550
GB	0.0052	0.1532	0.1370
GB + dynamic normalization	0.0047	0.1336	0.1407

Even though changing the static friction is not resulting in a different state space model, the adaptive controller is changing the state space parameters, the parameters a_1 and a_2 are changing so much that they even change sign. Nevertheless the results for the RMS values are better (smaller) for the adaptive controller than the non adaptive controller and the new controller stabilizes the system, even with the big change in the parameter estimation.

From Figure 4-38 it can be seen that the dynamic normalization slows down the parameter estimation and therefore the adaptation goes slower. This also can be seen in Figure 4-27 till Figure 4-29. Since the adaptation goes slower for the GB method + dynamic normalization

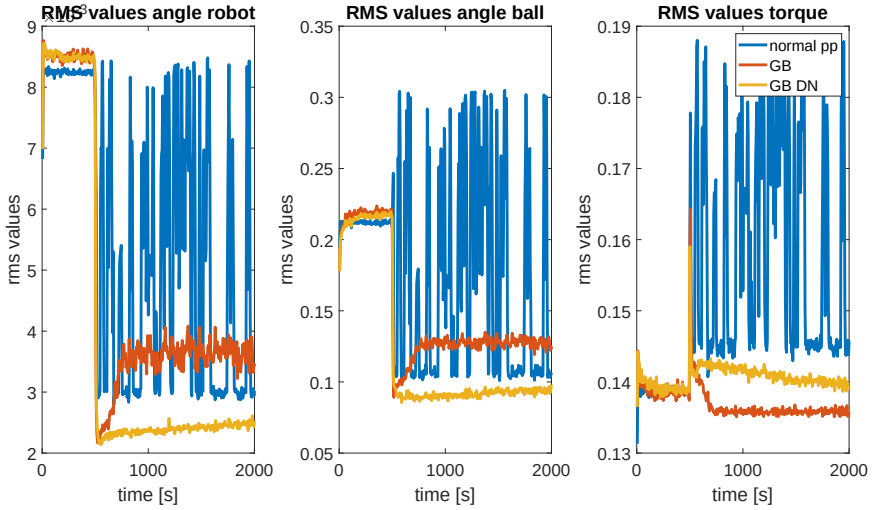


Figure 4-37: RMS values of the input and outputs of the system for when the static friction is changed. A standard, not changing pole placement controller is compared to the GB method and the GB method + dynamic normalization.

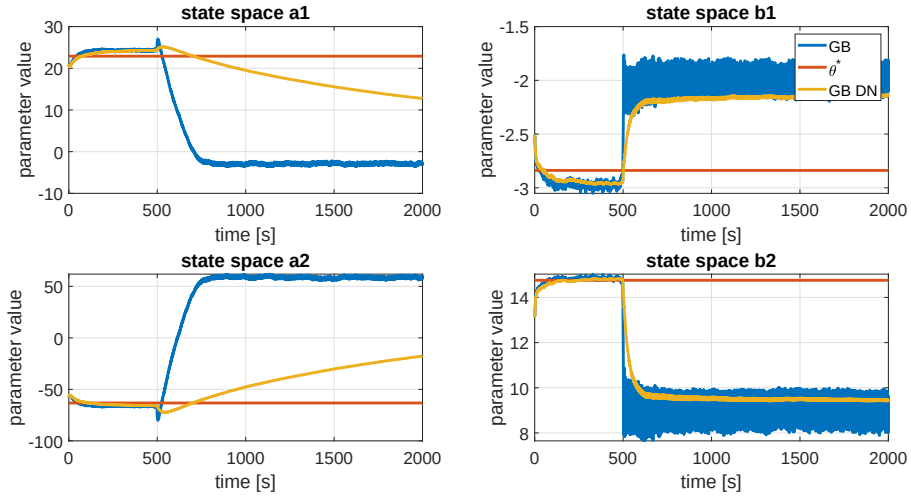


Figure 4-38: Parameter estimation for when the static friction is changed. A standard, not changing pole placement controller is compared to the GB method and the GB method + dynamic normalization.

and the controller is more complicated, the GB method without any added robust adaptive laws is the best adaptive controller to implement on the BBR when changing the surface.

4-5 Conclusion

In this Chapter simulations are used to tune and compare the adaptive controller and the adaptation laws. For the GB method it gave the following insights:

- The adaptation gain Γ indicates how fast the adaptation will go. When Γ is chosen too high, the adaptive controller becomes unstable.
- If the poles are changed, this has influence on the input and states of the system. The faster the poles, the bigger the input. It also influences the adaptation, the faster the poles the faster the adaptation. There is a certain optimum between the control effort, the performance of the controller and the speed of the adaptation.
- There was no region found for the starting values of the parameter estimation $\theta(0)$ which makes the adaptive controller unbounded or unstable.

For the RLS method it gave the following insights:

- The covariance matrix P indicates how fast the adaptation goes. Where Γ is a fixed value for the GB method, P is adapted as well, therefore it requires a start value $P(0)$. There is no limit found for $P(0)$ which makes the adaptive controller unstable. The bigger $P(0)$, the faster the adaptation will go. Overtime, P will decrease and therefore the adaptation is also slower.
- The same influence on the performance of the controller was noticed for either using the GB or the RLS method when changing the location of the poles or the the starting values of the parameter estimation $\theta(0)$.

When noise and unknown disturbances are added to the simulations, the parameter estimation is not bounded any more or the parameters converge to different values. Robust adaptive laws can bound the parameter estimation again but can not solve the problem that the parameters converge to different values. Dynamic normalization gave the best results compared with the other robust adaptive laws.

The GB method, RLS method and both methods extended with the dynamic normalization are compared with each other using simulations. During the simulation there is a mass added on top of the robot or the surface is changed where it rides on. The GB method and the GB method extended with dynamic normalization performed the best. These adaptation law adapted the controller fast when the plant was changed (add mass, change surface) while the control effort and the deviation of the angles were the lowest.

Chapter 5

Implementation

The controllers designed in Chapter 4 are implemented on the real Ball Balancing Robot (BBR) and compared in this Chapter. Equal to the simulations, a non-adaptive pole placement controller is implemented first. Afterwards the Adaptive Pole Placement Controller (APPC) is implemented. The adaptive controller is expanded with the robust adaptive laws, to study if these will improve the performance of the adaptive controller. For this Chapter, the model and controller for $y-z$ plane are used to show the results of the different controllers.

5-1 Adaptive Pole Placement

First a non-adaptive controller is implemented on the BBR. The motor controller settings are changed to torque command, since it is more convenient to use torque as input of your system instead of velocity (which is already an output of your system). Unfortunately, the torque commanded controller did not work good enough to make it adaptive. It could balance for a few seconds, but after a while the controller became unstable. The torque commanded controller has never worked before on the BBR, while the velocity commanded controller works good. An adaptive controller needs a stabilizable controller to start with, therefore it is chosen to restore the velocity commanded controller.

5-1-1 Simple pole placement

Figure 3-2 is used as structure for the pole placement controller. Since velocity commanded control is used, the plant parameters value are unknown, therefore the controller gain K is tuned by trial and error. The pole placement controller is extended with an integral action, as described by [16], to improve the performance of the controller. The integral is used for the angle of the robot θ_x. The values used for the controller are as follows:

$$K = [-300, -50, -14000, -250],\ I = 80000 \tag{5-1}$$

With this controller the following results are achieved: Figure 5-1 shows the output of the controller, Figure 5-2 shows the input of the system.

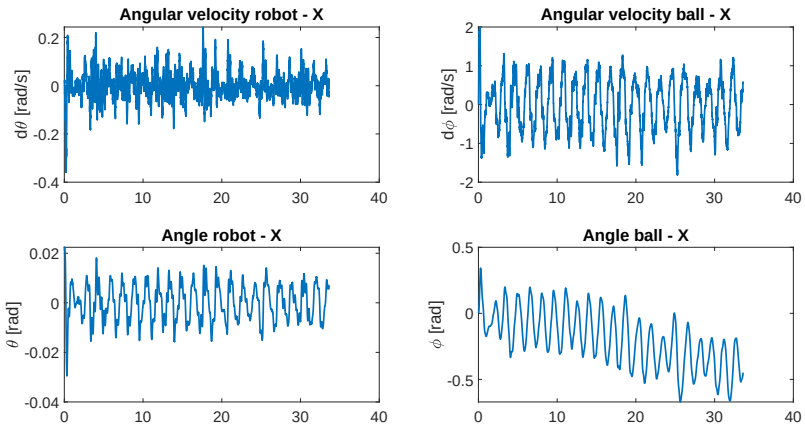


Figure 5-1: Output of the pole placement plus integral action control.

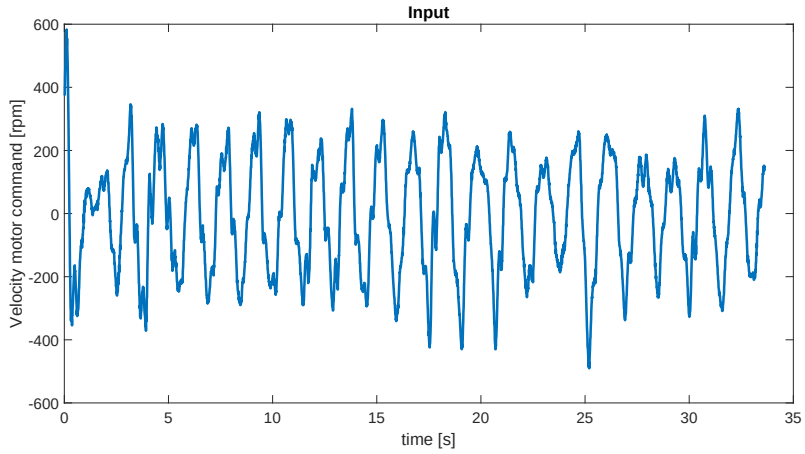


Figure 5-2: Input of the pole placement plus integral action control.

As can be seen, the BBR is balanced by the pole placement controller. One interesting thing that can be seen from the data of the angle of the ball (Figure 5-1) is that the ball drifts away. This is probably due to the calibration of the gyroscope and the accelerometer. Both sensors are calibrated before the controller starts, this is required, such that the controller knows when the robot is at zero degrees. The calibration is done by hand, by holding the BBR at zero degrees at best for five seconds. The mean value over this 5 seconds is used as 'zero' degrees. To compare the controllers and not how well the calibration is done, the calibration values are saved and used for all the different controllers. After a while the saved calibration values are not ideal any more, which can be detected by a drifting angle of the ball. The run shown above in Figure 5-1 had a slightly off calibrated value.

5-1-2 Tuning adaptive controller

From Section 4-4 it could be concluded that the most suitable APPC for the BBR is the Gradient Based (GB) method. The input output data from a real experiment (Figure 5-2 and Figure 5-1) is used in `MATLAB` to tune the adaptive controller. The following steps are used in an iterative way to tune the adaptive controller:

- Choose initial values of model parameters $\theta(0)$. θ_0 exist of $a_1(0)$, $a_2(0)$, $b_1(0)$, $b_2(0)$. Since the values of the model parameters are unknown (since using velocity commanded control), the values described in Equation (2-37) are used as an initial guess.
- Choose the starting controller gain matrix $K(0)$. With $K(0)$, $A(\theta(0))$ and $B(\theta(0))$ the closed loop poles are calculated ($\texttt{eig}(A - B \cdot K)$) which are used as poles for the adaptation law. The poles are checked if they are stable.
- Choose the adaptation gain matrix Γ.
- The integral gain I from the added integral action is not changed, neither by the adaptive controller. Since the integral gain does not have an influence on the location of the poles.

Without using the robust adaptive laws, the parameter estimation and the gain matrix calculations were not bounded. Therefore dead zone (Section 3-4-1) and dynamic normalization (Section 3-4-4) are implemented. A slightly different algorithm is used for dynamic normalization:

$$\epsilon(k) = \frac{\zeta(k) - \hat{\zeta}(k)}{m(k)^2} \tag{5-2}$$

$$m(k)^2 = 1 + \phi(k)^\top P \phi(k) + n_s(k)^2, \quad n_s(k)^2 = m_s(k) \tag{5-3}$$

$$m_s(k) = m_s(k-1) - (\delta_0 m_s(k-1) + \phi(k)^\top P \phi(k)) T_s, \quad m_s(0) = 0 \tag{5-4}$$

The difference is the extra weight matrix P such that weights can be added to the different entries of ϕ.

With these added robust adaptive laws the following parameters need to be tuned as well:

- Choose δ_0 and the weight matrix P for the dynamic normalization.
- Choose the upper bound on the estimation error g_0. When the estimation error is below this value, the adaptation is turned off. Since there are two parametric models (described by Equation (3-16) and Equation (3-17)), there are two upper bounds as well, $g_{1,0}$ and $g_{2,0}$.

The following values (Table 5-1) are used for the parameters described above for the APPC on the BBR.

Table 5-1: Parameters used for the APPC plus dead zone and dynamic normalization.

Parameter	Value
$a_1(0)$	1
$a_2(0)$	-0.003
$b_1(0)$	10
$b_2(0)$	0.001
$K(0)$	[-200, -50, -10000, -200]
Γ	$\begin{bmatrix} 10^8 & 0 \\ 0 & 10^4 \end{bmatrix}$
δ_0	0.001
P	$\begin{bmatrix} 10 & 0 \\ 0 & 10^6 \end{bmatrix}$
$g_{1,0}$	$2 \cdot 10^{-10}$
$g_{2,0}$	$3 \cdot 10^{-10}$

The states and input data showed in Figure 5-1 and Figure 5-2 respectively is used to show the results for the adaptation algorithm which calculates the parameter estimation θ (Figure 5-3) and the gain matrix K (Figure 5-4). The estimation errors e_1 and e_2 are shown in Figure 5-5.

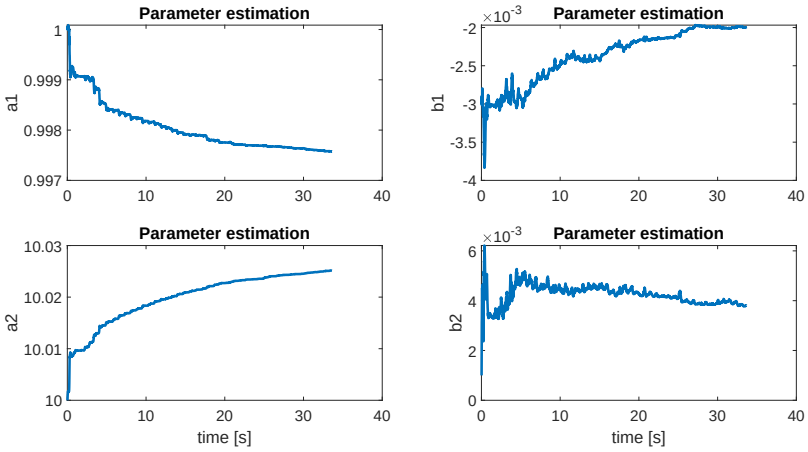


Figure 5-3: Parameter estimation using the GB method combined with dead zone and dynamic normalization. Input output data is used from a real run on the BBR to tune the adaptive controller.

When the parameters of the model are estimated, the plant should stay observable and controllable. A check is added which calculates each time step if the new estimated plant is observable and controllable. This is done by calculating the rank of both the controllability and observability matrix. The result is shown in Figure 5-6. As can be seen, the rank does not drop for both the controllability and observability matrix and therefore the plant stays controllable and observable.

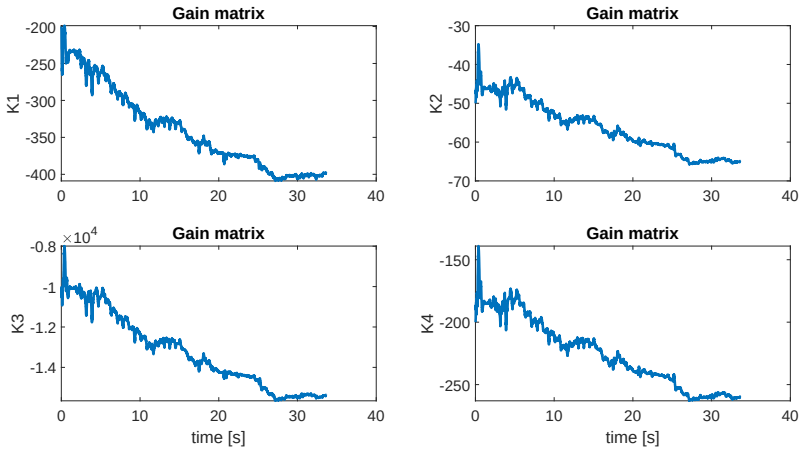


Figure 5-4: Controller gains using the GB method combined with dead zone and dynamic normalization. Input output data is used from a real run on the BBR to tune the adaptive controller.

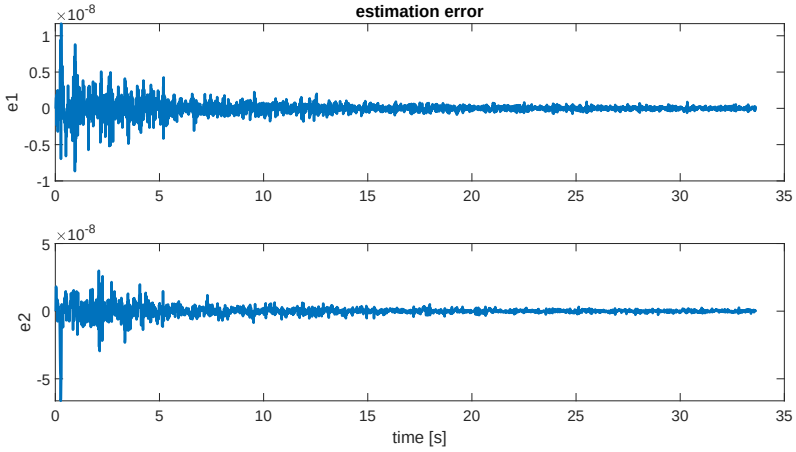


Figure 5-5: Estimation error e_1 and e_2 using the GB method combined with dead zone and dynamic normalization. Input output data is used from a real run on the BBR to tune the adaptive controller.

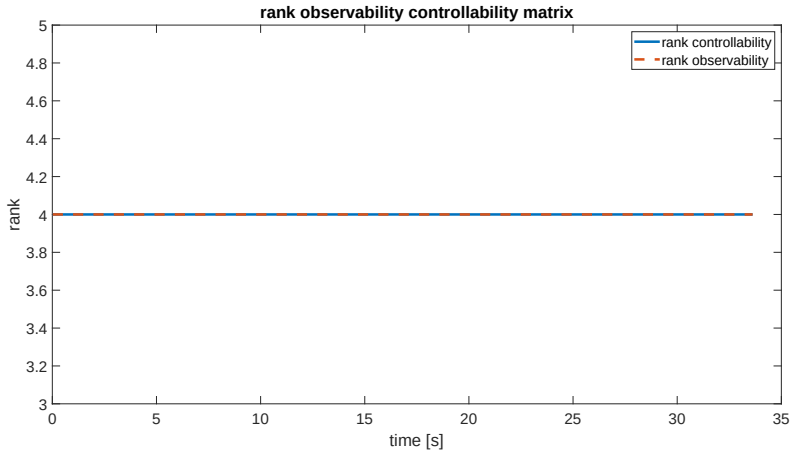


Figure 5-6: Rank of the controllability and observability matrix to check if the estimated model stays observable and controllable.

5-2 Comparison

The APPC described above is implemented on the real system and compared with the non-adaptive pole placement controller described in Section 5-1-1. The controllers are compared using two different cases. One where there is a mass added on top of the robot of 8 kg and one where two different surfaces are used where the BBR rides on.

The comparison is done by looking at the deviation of the states and the control effort. During the runs, the BBR is balancing and station keeping and the robot is not disturbed. So by looking at the deviation of the states it can be seen how well the controller performs at balancing and station keeping, and how much effort it costs.

5-2-1 Adding mass

Multiple runs are done with a non-adaptive and adaptive pole placement controller. After 30 seconds a mass of 8 kg is added on top of the robot. The results for one run of the APPC and the simple pole placement controller is shown in Figure 5-7 (states) and Figure 5-8 (input).

The results for the parameter estimation (Figure 5-9) and the gain matrix (Figure 5-10) are shown below for one of the runs of the APPC.

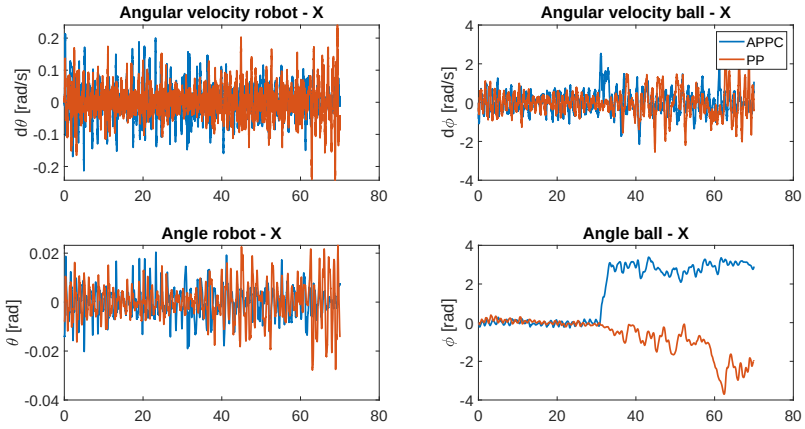


Figure 5-7: Comparison between the states of an adaptive and non-adaptive pole placement controller when 8 kg is added on top of the robot after 30 seconds.

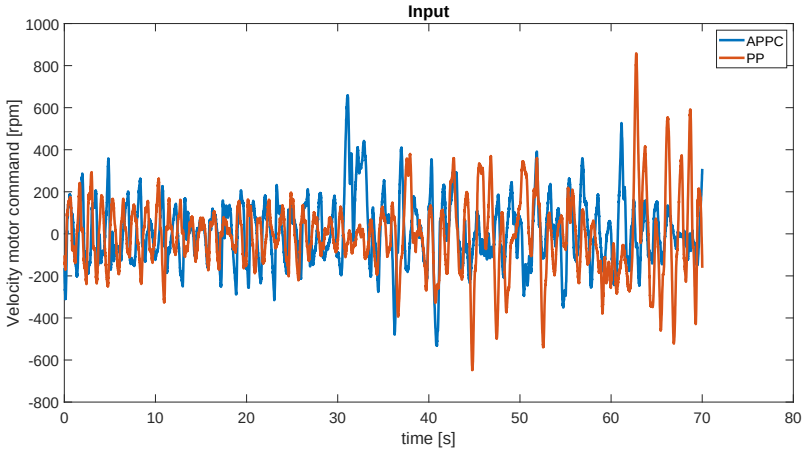


Figure 5-8: Comparison between the inputs of an adaptive and non-adaptive pole placement controller when 8 kg is added on top of the robot after 30 seconds.

As illustrated, the parameters and the gains do change when the mass is added. The results of the other runs look similar and are therefore not shown. Even though it can be seen that the adaptive controller is changing when the mass is added, it is hard to see from the above figures if the adaptive controllers performs better than the non-adaptive controller.

To compare both controllers the RMS values are calculated for the states and the input for segments of the run of 8 seconds. This is done for multiple runs for both controllers. The mean is taken of these RMS values for the different runs and is shown in Figure 5-11 for the states and Figure 5-12 for the input. The RMS value is also calculated for all the runs combined, these values for the states and the input are shown in Table 5-2.

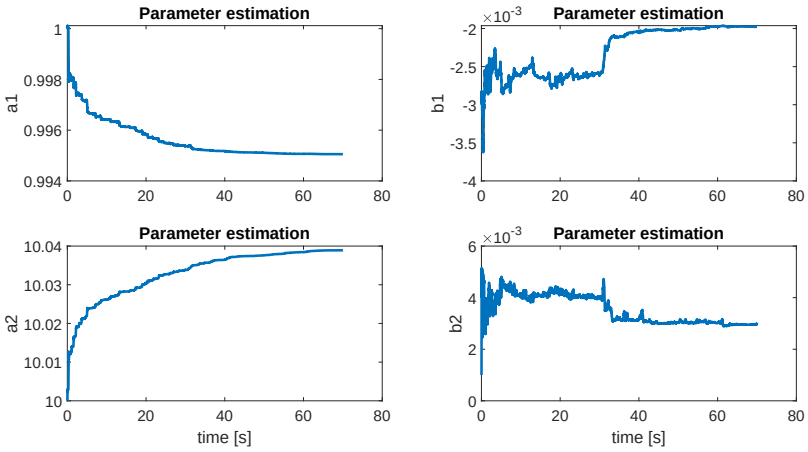


Figure 5-9: Parameter estimation when 8 kg is added on top of the robot after 30 seconds.

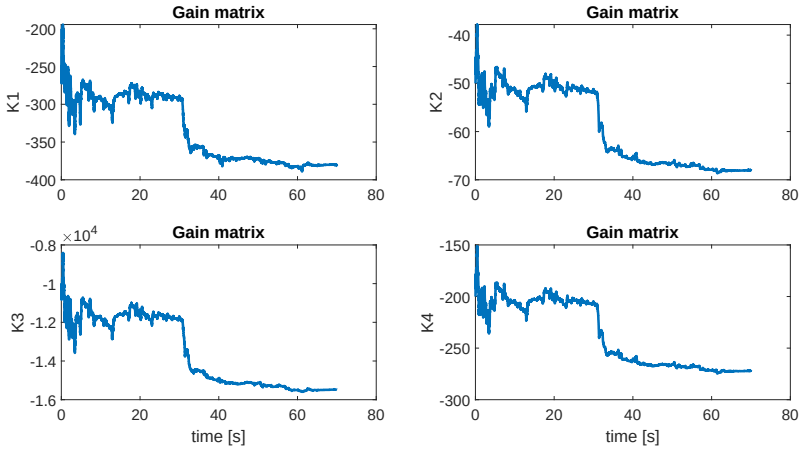


Figure 5-10: Controller gains when 8 kg is added on top of the robot after 30 seconds.

Table 5-2: RMS values of the states and the inputs for the total run, multiple runs combined.

	angular velocity robot	angular velocity ball	angle robot	angle ball	input
APPC	0.0299	0.3610	0.0036	1.3737	98.5682
pole placement	0.0426	0.5105	0.0057	0.4169	137.8231

As can be seen, the APPC performs better than the simple pole placement controller. The simple pole placement controller only performs better on the angle of the ball. This can have multiple reasons:

- The adaptive controller converge to a worse controller for controlling the position of the robot when mass is added.
- When the mass is added to the robot, it is hard to place the mass such that the centre of gravity (cog) does not change. If it is miss placed, the robot starts to drift away

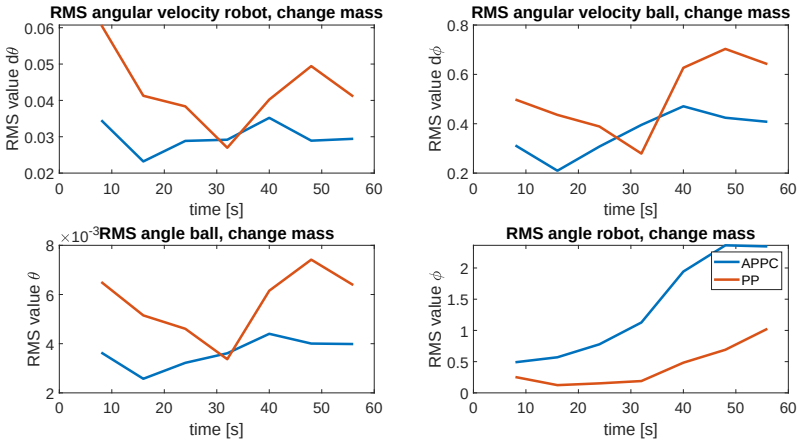


Figure 5-11: RMS values for the states when a mass of 8 kg is added on top of the robot for the APPC and pole placement controller.

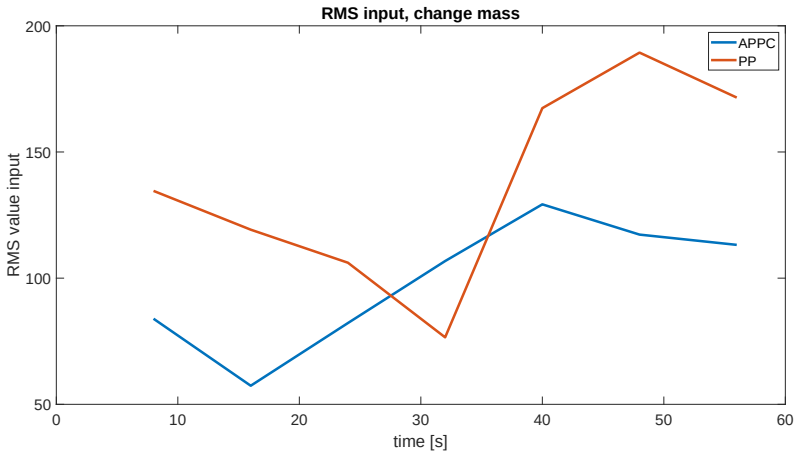


Figure 5-12: RMS values for the input when a mass of 8 kg is added on top of the robot for the APPC and pole placement controller.

and finds a new equilibrium, this phenomenon is clearly shown by Figure 5-7. It could be by chance that the mass was misplaced more for the APPC than the simple pole placement controller.

5-2-2 Changing the surface

In this section the adaptive and the non-adaptive controller are compared when the BBR is balancing on different surfaces. The controllers are tuned when the BBR is balancing on a linoleum floor. The controllers are then compared when the BBR is placed on a carpet floor. It is expected that there is not a big difference between the controllers when the BBR is placed on a linoleum floor. When they are used on a carpet floor it is expected that the adaptive controller adapts itself such that it performs better, which already could be seen in the results from the simulations (Section 4-4-2).

Figure 5-13 shows the adaptive controller gains when the BBR is placed either on linoleum or carpet. As can be seen is there not a difference between the adaptive controllers to which values they converge when the BBR is placed on either linoleum or carpet.

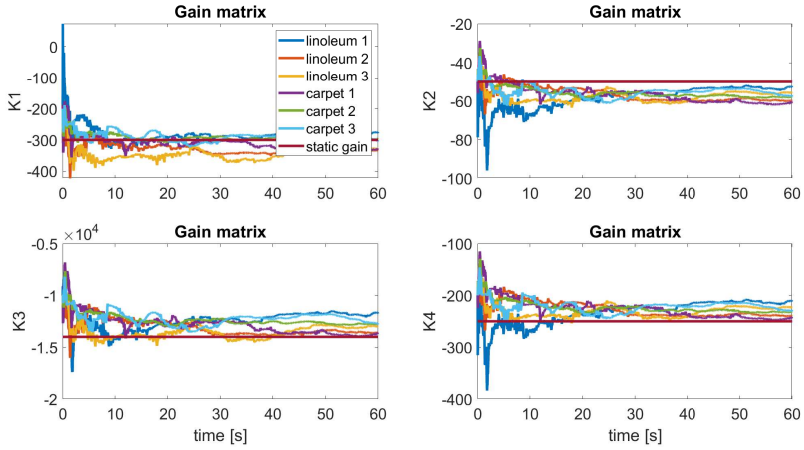


Figure 5-13: Controller gains when the BBR is either placed on linoleum or carpet. The non-adaptive static gain is shown for comparison.

Figure 5-14 and Figure 5-15 show the results of the RMS values of the states and input respectively for 3 different runs combined for the APPC and the non-adaptive pole placement controller for the linoleum floor. Figure 5-16 and Figure 5-17 show the results for the carpet floor. The RMS values shown, are calculated for segments of the run of 8 seconds. Table 5-3 shows the mean RMS values of the three runs for the total length of the run. Three different runs are done for each surface type.

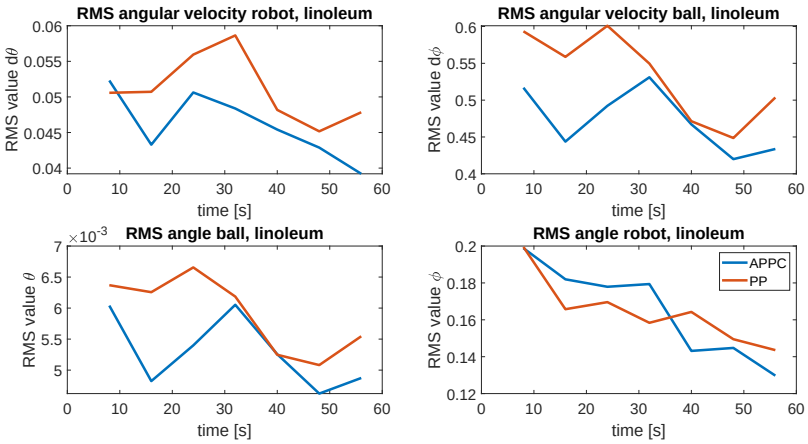


Figure 5-14: RMS values for the states when the robot rides on linoleum for the APPC and pole placement controller.

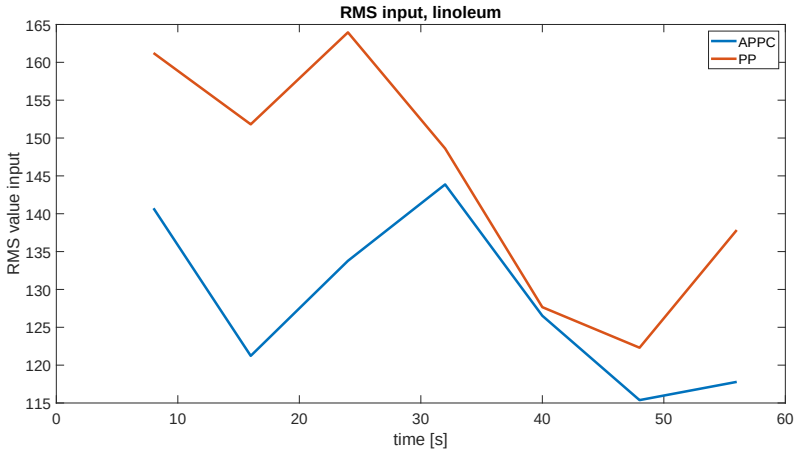


Figure 5-15: RMS values for the input when the robot rides on linoleum for the APPC and pole placement controller.

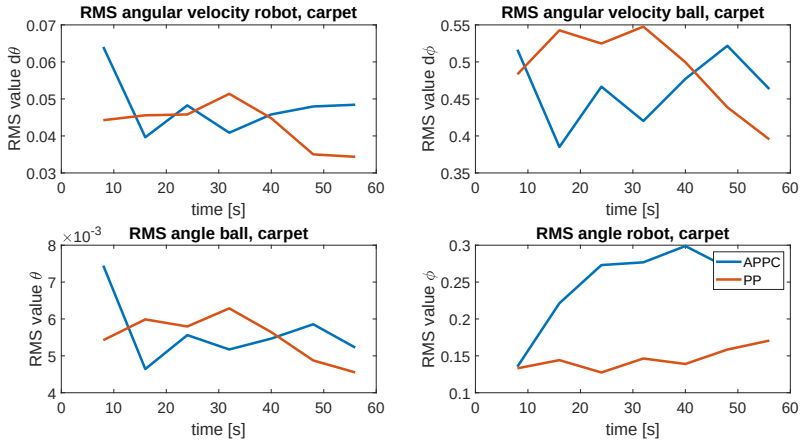


Figure 5-16: RMS values for the states when the robot rides on carpet for the APPC and pole placement controller.

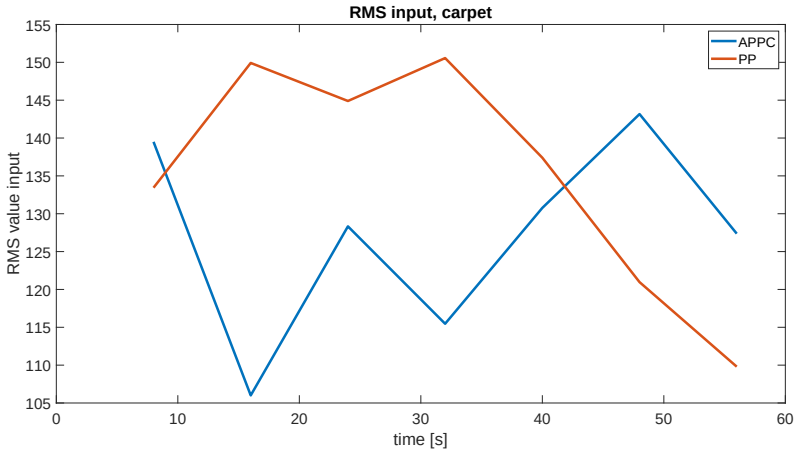


Figure 5-17: RMS values for the input when the robot rides on carpet for the APPC and pole placement controller.

Table 5-3: RMS values of the states and the inputs for the total run, multiple runs combined. When the BBR is either balancing on linoleum or carpet using a APPC or pole placement controller.

	APPC linoleum	PP linoleum	APPC carpet	PP carpet
angular velocity robot	0.0460	0.0510	0.0479	0.0430
angular velocity ball	0.4721	0.5323	0.4644	0.4903
angle robot	0.0053	0.0059	0.0056	0.0055
angle ball	0.1651	0.1644	0.2455	0.1456
input	128.47	144.77	127.23	135.28

The difference between the APPC and the simple pole placement controller is not big, which was expected, when balancing on a linoleum surface. The difference when balancing on a carpet floor is even smaller, which was not really expected. It was expected that the controller gains would converge to different values such that the adaptive controller would perform better than the non-adaptive controller. That this would not be the case, already could be deduced from Figure 5-14. There is no difference to which values the controller gains converge when either balancing on linoleum or carpet.

5-3 Conclusion

The adaptive controller described above can work on the BBR. It adapts its parameter estimation when a mass of 8 kg is added and thereby changing the controller gains K as well. The adaptive controller also performs better than the non-adaptive controller.

When the surface where the BBR rides on is changed, the adaptive controller can not identify this. Therefore the controller gains do not converge to different values and the difference between the performance of the adaptive and non-adaptive controller is nil.

Chapter 6

Conclusion and Recommendations

This chapter presents the conclusions which can be drawn from this thesis and gives the recommendations for further research and improvements for the Ball Balancing Robot (BBR). Furthermore, the answers on the research questions are discussed.

6-1 Conclusion

Earlier research showed that the simpler the controller, the bigger the chance that the controller can be implemented on a real BBR and works as expected. Therefore it is tried to keep the adaptive controller as simple as possible.

With simplicity in mind, three 2D models are derived (instead of one 3D model) to model the dynamics of the BBR. It was tried to identify the model parameters using closed-loop identification, but unfortunately the results were not good enough to create a useful model. Therefore, the model parameters are estimated and used to fill into the linearized model. The linearized model showed that the plant is unstable (positive poles) and non-minimum phase (positive zeros).

What is a suitable adaptive controller to implement on the BBR?

Literature showed that the Adaptive Pole Placement Controller (APPC) is a the most suitable adaptive controller to use for the BBR since it can cope with the non-minimum phase characteristic. The adaptive controller needs an adaptation law such that it adapts the online model and therewith the controller. The Gradient Based (GB) method and the Recursive Least Squares (RLS) method are used in this thesis as adaptation law. These can be extended with robust adaptations laws.

What is the influence of the different parts of the adaptive controller on the performance of the controller using simulations?

`MATLAB` and `SIMULINK` are used to tune and compare the adaptive controller and the adaptation laws. For the GB method it gave the following insights:

- The adaptation gain Γ indicates how fast the adaptation will go. When Γ is chosen too high, the adaptive controller becomes unstable.
- If the poles are changed, this has influence on the input and states of the system. The faster the poles, the bigger the input. It also influences the adaptation, the faster the poles the faster the adaptation. There is a certain optimum between the control effort, the performance of the controller and the speed of the adaptation.
- There was no region found for the starting values of the parameter estimation $\theta(0)$ which makes the adaptive controller unbounded or unstable.

For the RLS method it gave the following insights:

- The covariance matrix P indicates how fast the adaptation goes. Where Γ is a fixed value for the GB method, P is adapted as well, therefore it requires a start value $P(0)$. There is no limit found for $P(0)$ which makes the adaptive controller unstable. The bigger $P(0)$, the faster the adaptation will go. Overtime, P will decrease and therefore the adaptation is also slower.
- The same influence on the performance of the controller was noticed for either using the GB or the RLS method when changing the location of the poles or the the starting values of the parameter estimation $\theta(0)$.

When noise and unknown disturbances are added to the simulations, the parameter estimation is not bounded anymore or the parameters converge to different values. Robust adaptive laws can bound the parameter estimation again but can not solve the problem that the parameters converge to different values. Dynamic normalization gave the best results compared with the other robust adaptive laws.

The GB method, RLS method and both methods extended with the dynamic normalization are compared with each other using simulations. During the simulation there is a mass added on top of the robot or the surface is changed where it rides on. The GB method and the GB method extended with dynamic normalization performed the best. These adaptation laws adapted the controller fast when the plant was changed (add mass, change surface) while the control effort and the deviation of the angles were the lowest. For implementation, it is desired to have the simplest controller while performing good. Therefore it is chosen to implement the GB method on the BBR.

Which type of control is able to control the real BBR regarding both balancing and station keeping?

Before an adaptive controller can be implemented, a static non-adaptive controller needs to work on the BBR with the same structure as the adaptive controller. Since an APPC will be used for the BBR, a pole placement controller is designed. A pole placement controller was not sufficient enough to balance the BBR. The controller resulted in a shaky movement which resulted sometimes in a unstable controller. Therefore the pole placement controller was extended with an integral action.

Which controller shows the best performance on the set up of the BBR?

It was found when implementing the adaptive controller on the real BBR, that the controller was not bounded, therefore the GB method was extended with dynamic normalization and dead zone. The APPC is compared with a non-adaptive pole placement controller using two use cases: adding a mass of 8 kg on top of the robot after 30 seconds from the start of an experiment and changing the surface where the BBR rides on from linoleum to carpet. The APPC works better than the non-adaptive pole placement controller when a mass is added. The adaptive controller adapts its controller gains and therefore has a better performance. When changing the surface, there is no big difference between the APPC and the non-adaptive pole placement controller.

6-2 Recommendations

This sections describes the recommendations for further research and improvements for the BBR.

- There are no big differences when either using the APPC or non adaptive pole placement controller when changing the surface. The adaptive controller was tuned for when the mass would change of the robot. It would be interesting to study, if the adaptive controller performs better when the adaptive controller is tuned for when the surface is changed. Furthermore the controllers were compared when the BBR should balance and keep position. Therefore the deviations of the states and the input are small and therefore the adaptation law has less information to use for the adaptation. When a fixed impulse disturbance or a step reference signal is used, the deviations are bigger and therefore the adaptation law has more information. It can be studied if the adaptive controller performs even better when it has more information for the adaptation law and if a difference can be noticed between the controllers when placed on a carpet floor.
- Other use cases can be studied for the adaptive controller such as changing the dimensions of the ball or change the 3 kg medicine ball to a volleyball for instance.
- Three different pole placement controllers were designed for each plane of the robot. Therefore it misses the coupling terms between the planes. For the case where the BBR is balancing and station keeping this missing coupling term is not a big problem, but when the BBR should follow a certain trajectory it could benefit from these coupling terms. A 3D model can be derived for which a pole placement controller can be designed. It would be interesting to study if the 3D model (adaptive) pole placement controllers performs better than the 3 separate 2D model (adaptive) pole placement controllers.
- The closed-loop identification of the model parameters did not give results as expected, the fit was not good enough to work with while parameters like the inertia were negative. It would be interesting to use the indirect closed-loop identification method instead of the direct method to see if the results are better.
- At the beginning of each run, the gyroscope and the accelerometer needs to be calibrated. If this is not done properly, the BBR starts to drift away. An algorithm needs to be designed which notices this drift and automatically adjust the calibrated values such that the BBR does not drift away anymore.

Appendix A

Identification of the parameters

A-1 Estimation

The parameters in Table 2-1 are estimated in the following matter:

A-1-1 Masses

The mass of the ball, the robot and the wheels are measured.

A-1-2 Inertia

The inertia of the robot is calculated using the CAD model in SolidWorks. The inertia is calculated around the centre of gravity (cog) excluding the ball and the wheels.

The inertia of the omni wheel is estimated using the formula for a solid disk: $I = \frac{mr^2}{2}$ where m is the mass of the wheel m_w and r the radius of the wheel r_w. The inertia of the omni wheels around it's rotation axis is:

$$I_{ow,r} = \frac{m_w r_w^2}{2} \tag{A-1}$$

The inertia of the motor and the gears can be found in the their data sheets. Given the inertia of the omni wheel, the motor and the gears, the inertia of the virtual actuating wheel around the ith axis $I_{w,i}$ can be calculated. Since the inertia of the gears is low in comparison with the inertia of the wheel it is neglected. The rotational energy around each axis should be equal for the virtual actuating wheel and the real set up.

x-axis:

$$\begin{aligned}\frac{1}{2}I_{w,x}\dot{\psi}_x^2 =&\frac{1}{2}I_{ow}\big(\cos(\alpha)\dot{\psi}_x\big)^2 + \frac{1}{2}I_m\big(\cos(\alpha)k\dot{\psi}_x\big)^2 \\ &+ 2\Big(\frac{1}{2}I_{ow}\big(\cos(\alpha)\sin(1/6\pi)\dot{\psi}_x\big)^2 + \frac{1}{2}I_m\big(\cos(\alpha)\sin(1/6\pi)k\dot{\psi}_x\big)^2\Big) \\ =&\frac{1}{2}I_{ow}\cos(\alpha)^2\dot{\psi}_x^2 + \frac{1}{2}I_m\cos(\alpha)^2k^2\dot{\psi}_x^2 + \frac{1}{4}I_{ow}\cos(\alpha)^2\dot{\psi}_x^2 + \frac{1}{4}I_m\cos(\alpha)^2k^2\dot{\psi}_x^2\end{aligned}$$

Where k is the reduction ratio of the gears. From the above equation it follows that $I_{w,x}$ is equal to:

$$I_{w,x} = \frac{3}{2}\cos(\alpha)^2(I_{ow} + k^2I_m) \tag{A-2}$$

y-axis:

$$\begin{aligned}\frac{1}{2}I_{w,y}\dot{\psi}_y^2 =&2\Big(\frac{1}{2}I_{ow}\big(\cos(\alpha)\cos(1/6\pi)\dot{\psi}_y\big)^2 + \frac{1}{2}I_m\big(\cos(\alpha)\cos(1/6\pi)k\dot{\psi}_y\big)^2\Big) \\ =&I_{ow}\cos(\alpha)^2\frac{3}{4}\dot{\psi}_y^2 + I_m\cos(\alpha)^2\frac{3}{4}k^2\dot{\psi}_y^2\end{aligned}$$

From the above equation it follows that $I_{w,y}$ is equal to:

$$I_{w,y} = \frac{3}{2}\cos(\alpha)^2(I_{ow} + k^2I_m) \tag{A-3}$$

z-axis:

$$\begin{aligned}\frac{1}{2}I_{w,z}\dot{\psi}_z^2 =&3\Big(\frac{1}{2}I_{ow}\big(\sin(\alpha)\dot{\psi}_z\big)^2 + \frac{1}{2}I_m\big(\sin(\alpha)k\dot{\psi}_z\big)^2\Big) \\ =&\frac{3}{2}I_{ow}\sin(\alpha)^2\dot{\psi}_z^2 + \frac{3}{2}I_m\sin(\alpha)^2k^2\dot{\psi}_z^2\end{aligned}$$

From the above equation it follows that $I_{w,y}$ is equal to:

$$I_{w,z} = 3\sin(\alpha)^2(I_{ow} + k^2I_m) \tag{A-4}$$

The inertia for the ball is approximated by the inertia for a hollow sphere:

$$I_b = \frac{2m_br_b^2}{3} \tag{A-5}$$

A-1-3 Distances

Most of the distances are either known or measured such as the radius of the wheels and the ball. The distance between the cog of the ball and the robot is calculated using the SolidWorks model.

A-1-4 Friction parameters

The Coulomb friction for the ball is approximated by the rolling resistance times the radius of the ball plus a factor for the Coulomb friction for the motor. [20]

$$D_{c,b} = mgC_r r_b + D_{c,m} \tag{A-6}$$

Where m is total mass of the robot and ball, g the gravitational acceleration, C_r the rolling resistance coefficient and $D_{c,m}$ the Coulomb friction from the motor. C_r is approximated as ≈ 0.005. $D_{c,m}$ is approximated as ≈ 0.2. Such that the approximation of the total friction is $D_{c,b} \approx 0.3\,[\text{Nm}]$.

The viscous friction for the ball $D_{v,b}$ is approximated by the viscous friction of the motor $D_{v,m}$ since the rolling resistance of the ball is not depended on the velocity at all. It is hard to exactly calculate this value and therefore it is approximated as $D_{v,b} \approx 0.2\,[\text{Nms}\,\text{rad}^{-1}]$.

As a first approximation the Coulomb friction and the viscous friction of the robot are taken the same as the frictions for the motor so:

$$D_{c,r} = D_{c,m} \approx 0.2\,[\text{Nm}]$$
$$D_{v,c} = D_{v,m} = D_{v,b} \approx 0.2\,[\text{Nms}\,\text{rad}^{-1}]$$

A-2 Closed-loop identification

The direct closed-loop identification method is used to identify the parameters for the dynamic model. Multiple different identification algorithms are used to identify the model:

- The MATLAB output error function `oe` (estimates a transfer function of a specified size).
- The MATLAB state space estimation function `ssest` (estimates a state space function of specified size).
- The MATLAB optimization function `lsqnonlin` is used to identify a parametrized model. A parametrized state space model, a parametrized transfer function model and a parametrized non-linear model is used.

The different identified models are compared with each other using the MATLAB function `compare`. This function plots the output of the models and displays the Normalized Root Mean Square Error (NRMSE). It is calculated as follows:

$$\text{NRMSE} = 100\left(1 - \frac{||y - \hat{y}||}{||y - \text{mean}(y)||}\right) \tag{A-7}$$

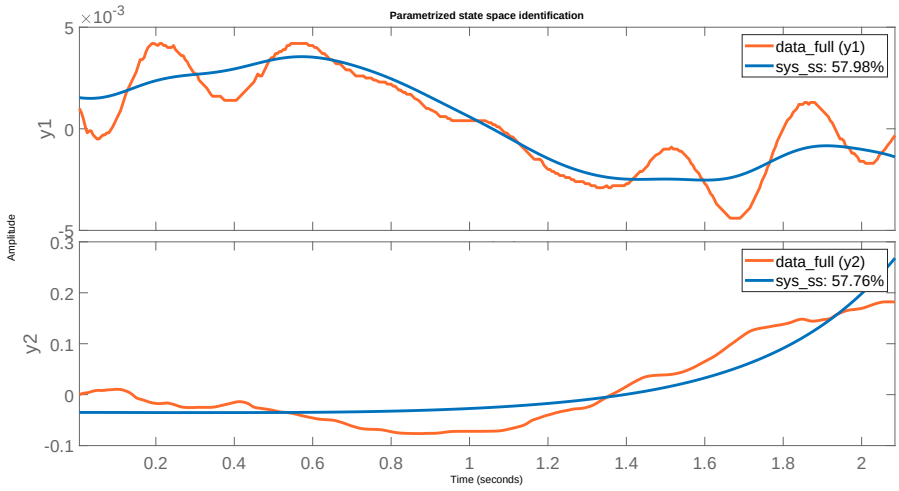


Figure A-1: Comparison of output data from an identified model (blue line) and measured data (orange line). y_1 is the angle of the robot and y_2 is the angle of the ball.

The best results (highest NRMSE) are achieved with the parametrized state space model, the results are shown in Figure A-1. The outputs of the model (y_1 and y_2) are compared with the measured angle of the robot and the ball.

The identified model is checked using a different data set than the one used for the identification. The result is shown in Figure A-2.

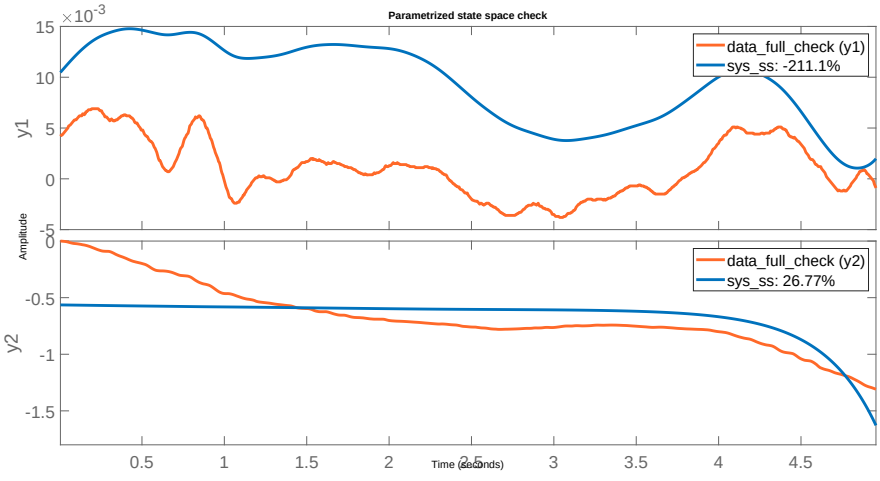


Figure A-2: Check for an identified model. Comparison of output data from an identified model (blue line) and measured data (orange line). y_1 is the angle of the robot and y_2 is the angle of the ball.

The results are not that good for the identification. The NRMSE is not that high, especially for the checked data and moreover are some of the identified parameters negative. A boundary condition is used for the parameters to avoid negative values. However, the results are less good, since the NRMSE is lower for the new bounded model.

Appendix B

Dynamic normalization results

This chapter shows the results when the Recursive Least Squares (RLS) method is extended with dynamic normalization.

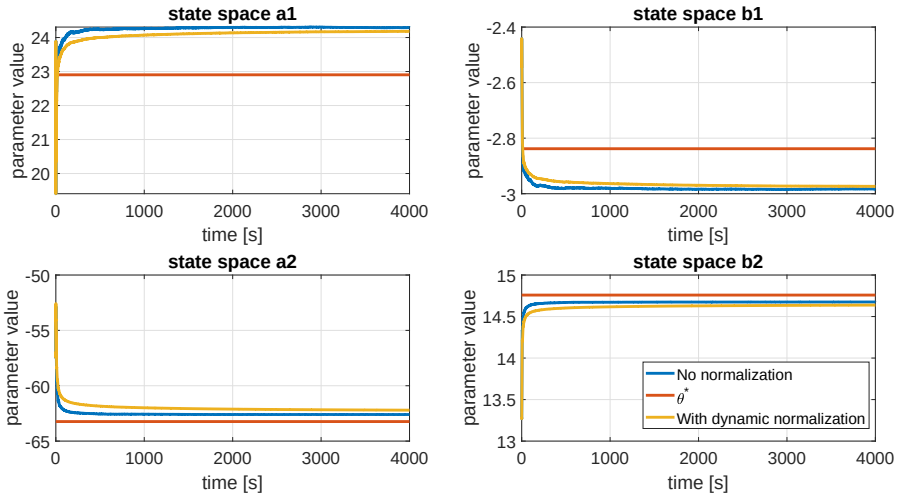


Figure B-1: Influence on the parameter estimation when using dynamic normalization. Each five seconds a disturbance. RLS

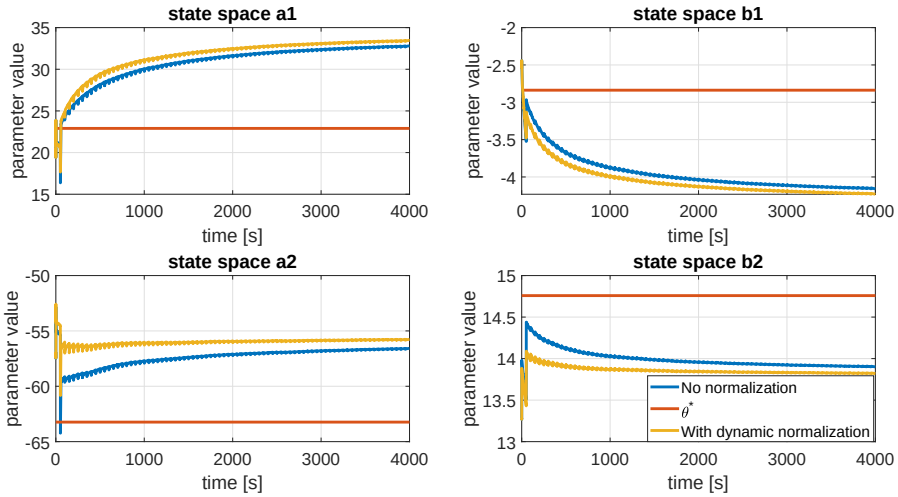


Figure B-2: Influence on the parameter estimation when using dynamic normalization. Each 50 seconds a disturbance. RLS

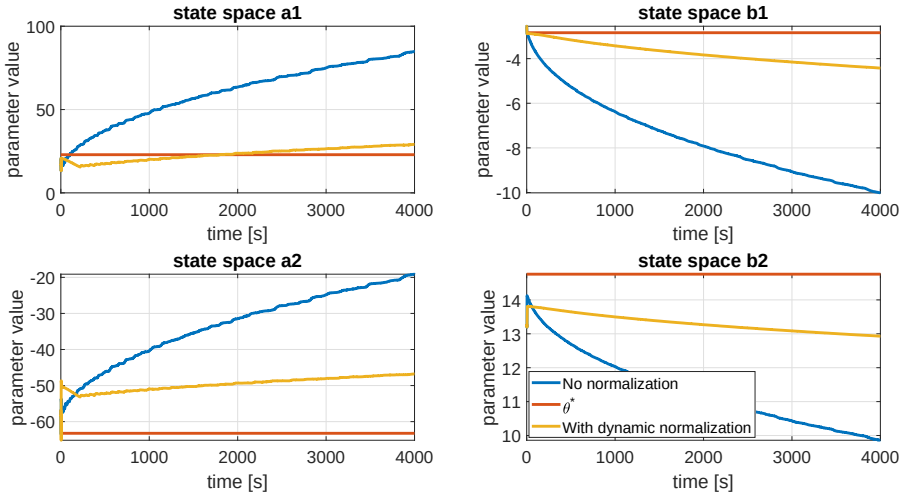


Figure B-3: Influence on the parameter estimation when using dynamic normalization. No disturbance, only noise. RLS

Bibliography

[1] P. Ioannou and J. Sun, *Robust Adaptive Control*. Courier Corporation, 2013.

[2] K. van der Blonk, "Modeling and control of a ball-balancing robot," Master's thesis, University of Twente, 2014.

[3] C. Verdier, "Geometric control of an underactuated balancing robot," Master's thesis, Technische Universiteit Delft, 2015.

[4] A. M. Omer Saleem Bhatti, Osama Bin Tariq and O. A. Khan, "Adaptive intelligent cascade control of a ball-riding robot for optimal balancing and station-keeping," *Advanced Robotics*, vol. 32.

[5] T. B. Lauwers, G. A. Kantor, and R. L. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, May 2006.

[6] C.-C. Tsai, "Self-balancing and position control using multiloop approach for ball robots," *International Conference on System Science and Engineering*.

[7] C.-W. Liao, C.-C. Tsai, Y. Y. Li, and C.-K. Chan, "Dynamic modeling and sliding-mode control of a ball robot with inverse mouse-ball drive," *SICE Annual Conference*, 2008.

[8] C.-C. Tsai, C.-K. Chan, and L.-C. Kuo, "Lqr motion control of a ball-riding robot," *The 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2012.

[9] P. Fankhauser and C. Gwerder, "Modeling and control of a ballbot," Master's thesis, Eidgenössische Technische Hochshule Zürich, 7 2010.

[10] M. Kumagai and T. Ochiai, "Development of a robot balancing on a ball," *International Conference on Control, Automation and Systems*, 2008.

[11] D. Jeltsema, *Modeling and Nonlinear Systems Theory, reader SC4092*. TU Delft, 2013.

[12] L. L. Urban Forssekk, "Closed-loop identification revisited - updated version," 1998.

[13] J. Anninga, "Literature study for implementation of an adaptive controller on a ball balancing robot," Master's thesis, Technische Universiteit Delft, 3 2019.

[14] K. J. Åstrom and B. Wittenmark, "On self tuning regulators," *Automatica*, vol. 9, pp. 185–199, 1973.

[15] S. Baldi, *Adaptive and Predictive Control (SC4060)*. 2016.

[16] K. J. Åstrom and B. Wittenmark, *Computer-Controlled Systems, Theory and Design*. Dover Publications, Inc., 2011.

[17] A. E.-N. Gene F. Franklin, J. David Powell, *Feedback Control of Dynamic Systems*. PEARSON, 2010.

[18] P. Ioannou and B. Fidan, *Adaptive Control Tutorial*. SIAM, 2006.

[19] B. Egardt, *Stability of Adaptive Controllers*. Lecture Notes in Control and Information Sciences, 1979.

[20] *Werktuigbouw.nl Formuleboekje*. Mikrocentrum.

Glossary

List of Acronyms

APPC	Adaptive Pole Placement Controller
BBR	Ball Balancing Robot
CMU	Carnegie Mellon University
cog	centre of gravity
ETH	Eidgenössische Technische Hochshule
GB	Gradient Based
MRAC	Model Reference Adaptive Control
NCHU	National Chung Hsing Universit
NRMSE	Normalized Root Mean Square Error
RLS	Recursive Least Squares
RMS	Root Mean Squares
SIMO	Single Input Multiple Output
TGU	Tohoku Gakuin University